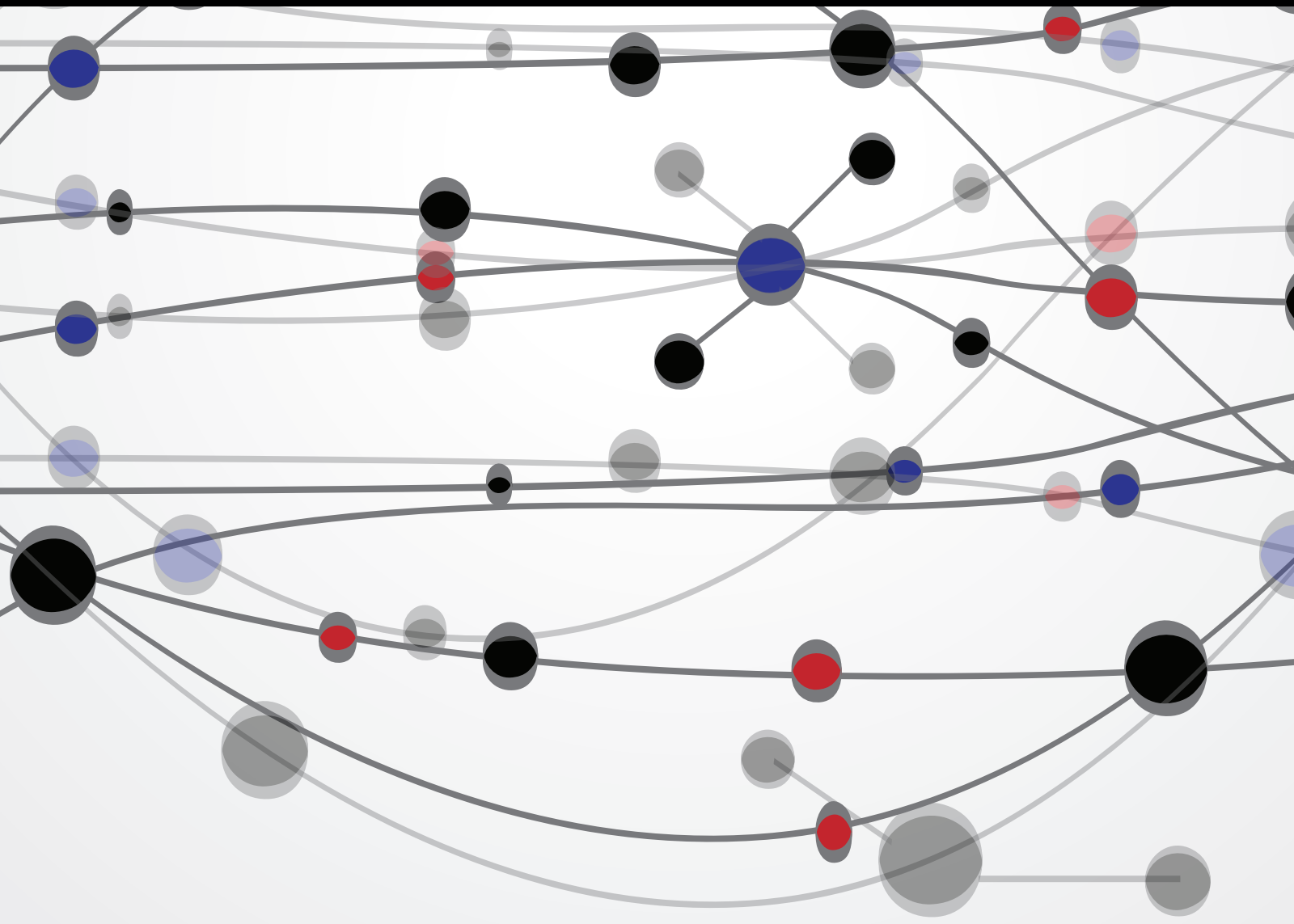
The Scientific World Journal

Recent Advancements in Computer & Software Technology

Guest Editors: K. K. Mishra, A. K. Misra, Peter Mueller, Gregorio Martinez Perez, Sanjiv K. Bhatia, and Yong Wang



Recent Advancements in Computer & Software Technology

The Scientific World Journal

Recent Advancements in Computer & Software Technology

Guest Editors: K. K. Mishra, A. K. Misra, Peter Mueller, Gregorio Martinez Perez, Sanjiv K. Bhatia, and Yong Wang

Contents

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 609512, 1 page
http://dx.doi.org/10.1155/2014/609512

Editorial
Recent Advancements in Computer & Software Technology

K. K. Mishra,[1] A. K. Misra,[1] Peter Mueller,[2] Gregorio Martinez Perez,[3] Sanjiv K. Bhatia,[4] and Yong Wang[5]

[1] *Department of Computer Science, MNNIT, Allahabad, India*
[2] *IBM Zurich Research Laboratory, Saeumerstraße 4, 8803 Rueschlikon, Switzerland*
[3] *University of Murcia, Murcia, Spain*
[4] *University of Missouri-St. Louis, MO, USA*
[5] *School of Information Science and Engineering, Central South University, Changsha, China*

Correspondence should be addressed to K. K. Mishra; mishrakrishn@gmail.com

Received 18 May 2014; Accepted 18 May 2014; Published 5 June 2014

Over the last few decades, advancements in computer and software technologies have reached an impressive level. These technologies improve not only very common areas of our daily life, but also areas of education, health, production industries, and so on. Thus, recent advancements in computer and software technologies are the base for the society of tomorrow.

The feasibility of future developments strongly relies on the existence of key technologies and their deployments. Analyzing global trends in cloud computing including all its services reveals cornerstone fields, such as distributed parallel processing, advanced software engineering, image processing, and security solutions. These fields require different sets of resources like computing hardware, Internet, software and hardware tools, mobility technologies, storage, system management, and security technology.

The objective of this special issue is to present a collection of articles that cover recent research results and comprehensive reviews on relevant computer and software technologies. In particular, it aims to present highly technical papers describing the areas mentioned above: distributed parallel processing, advanced software engineering, image processing, and security solutions.

This special issue received an overwhelming response from the community. Due to the limited space, only 10 papers from the initial 109 manuscripts have been selected. These papers represent the most up-to-date research work covering topics such as software process improvement and software testing, vulnerability and video quality assessment, artificial immune systems, improvements in file systems, and component models for distributed simulation, amongst others. Due to the space limitation we will not reiterate the contents here. We hope the reader will find this special issue informative and stimulating.

We would like to thank all the contributors who have submitted their high quality papers.

K. K. Mishra
A. K. Misra
Peter Mueller
Gregorio Martinez Perez
Sanjiv K. Bhatia
Yong Wang

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 478013, 18 pages
http://dx.doi.org/10.1155/2014/478013

Research Article

Using Heuristic Value Prediction and Dynamic Task Granularity Resizing to Improve Software Speculation

Fan Xu, Li Shen, Zhiying Wang, Bo Su, Hui Guo, and Wei Chen

National University of Defense Technology, Changsha, Hunan 410073, China

Correspondence should be addressed to Fan Xu; xfdadada@gmail.com

Received 24 December 2013; Accepted 2 March 2014; Published 20 May 2014

Academic Editors: K. K. Mishra and A. K. Misra

Exploiting potential thread-level parallelism (TLP) is becoming the key factor to improving performance of programs on multicore or many-core systems. Among various kinds of parallel execution models, the software-based speculative parallel model has become a research focus due to its low cost, high efficiency, flexibility, and scalability. The performance of the guest program under the software-based speculative parallel execution model is closely related to the speculation accuracy, the control overhead, and the rollback overhead of the model. In this paper, we first analyzed the conventional speculative parallel model and presented an analytic model of its expectation of the overall overhead, then optimized the conventional model based on the analytic model, and finally proposed a novel speculative parallel model named HEUSPEC. The HEUSPEC model includes three key techniques, namely, the heuristic value prediction, the value based correctness checking, and the dynamic task granularity resizing. We have implemented the runtime system of the model in ANSI C language. The experiment results show that when the speedup of the HEUSPEC model can reach 2.20 on the average (15% higher than conventional model) when depth is equal to 3 and 4.51 on the average (12% higher than conventional model) when speculative depth is equal to 7. Besides, it shows good scalability and lower memory cost.

1. Introduction

Exploiting potential thread-level parallelism (TLP) is becoming the key factor to improving performance of programs on multicore systems [1]. A series of productions provide effective solutions to parallel programming, such as OpenMP [2], MPI [3], and TBB [4]. However, as the processor cores increase and software application becomes more and more diverse, the traditional parallel programming frameworks are facing new challenges. First, the complexity of dependencies makes the program code hard to be parallelized effectively by traditional parallel programming tools. Programs with lots of conflict variables (CVARs, the variables involved in cross-iteration dependencies) usually cannot be parallelized smoothly. To solve this problem, some parallel programming tools offer explicit synchronization and communication interfaces for programmers, but this will increase the difficulty of parallel programming. Second, traditional parallel programming tools cannot support multiple parallelism modes. For example, OpenMP can support DOALL mode well but lacks support for DOACROSS or PIPELINE mode [5]. Third, as the processor core number increases, the scalability of traditional parallel programming methods faces additional challenge, too.

Speculative parallel execution model offers a solution to the problems above. It offers underlying hardware or software for correctness checking so that the programming interface is simpler. Programmers using Transactional Memory (TM) [6–9] or Thread Level Speculation (TLS) [10–14] models do not have to know the details about the dependencies between threads. They can neglect the dependencies while they are parallelizing the program and focus on the algorithm optimization or task partition. The underlying hardware or runtime system will help them to insure the program against errors. Speculative parallel model can drastically exploit parallelism in the program and reach a high performance, without increasing burden of programmers. The Stanford Hydra [15] with its TLS mechanism and various kinds of transactional memories are typical works of speculative parallel execution models.

Although the speculative parallel model is of high efficiency and practicability, there are defects of its mechanism.

For conventional hardware supported models, the changes in microarchitecture are costly and less scalable. To avoid these problems, many researches on the speculative parallel models are based only on software in recent years. A series of works such as BOP [16, 17], CorD [18, 19], and SpiceC [5] are proposed and the evaluation results of them are quite good. However, for the software-only speculative parallel models, there are still two kinds of obstacles. First, the missing of hardware support usually leads to both higher control overhead and rollback overhead. Second, the static task partitioning leads to the imbalance of the loads of each speculative thread. To overcome these obstacles, special strategies are needed to reduce the overall overhead and balance the load.

Aiming at the defects in the software-only speculative parallel models, in this paper, we try to use a novel value prediction scheme and a dynamic task partitioning scheme to improve the conventional models. The paper proposes our new software-based speculative parallel model, called HEUSPEC. Two main contributions are included.

(i) The model uses heuristic value prediction (HVP) mechanism to reduce the high misspeculative rate in the conventional speculative parallel models. The mechanism can generate predicted values of CVARs via multiple approaches, including history value prediction scheme. It uses a scorekeeper to evaluate and select the prediction results. The mechanism can improve the accuracy of speculative read in the model and reduce the rollback overhead.

(ii) The model involves dynamic task granularity resizing (DTGR) mechanism. The mechanism can optimize the overhead of the model at runtime by resizing the granularity of each parallel task. It can augment the task size when the misspeculation rate is at low level and deflate the task size when the rate is high. Thus it can reduce the overall time cost remarkably.

The rest of the paper is organized as follows. In Section 2, the overview of the HEUSPEC model is introduced. In Section 3, the key techniques are proposed in detail. The implementation of the model is proposed in Section 4. The evaluation and experiment results are given and analyzed in Section 5. In Section 6 we introduce some related works. Finally in Section 7, the conclusions are given.

2. Overview of the HEUSPEC Model

The HEUSPEC parallel model is a coarse-grained parallel programming framework. It consists of two parts: the runtime library and the source-to-source compiler. HEUSPEC uses two stage compiling methods. Figure 1 shows the hierarchy structure of HEUSPEC. The programmers can parallelize the sequential program easily with HEUSPEC. First, the original source code of the sequential program is labeled by the programmer; second, programmer uses the HEUSPEC source-to-source compiler to transform the labeled code into parallel code, with multiple parallel functions implemented in the HEUSPEC runtime library. Finally, the parallel code is compiled by a normal compiler and transformed to the parallel binary code.

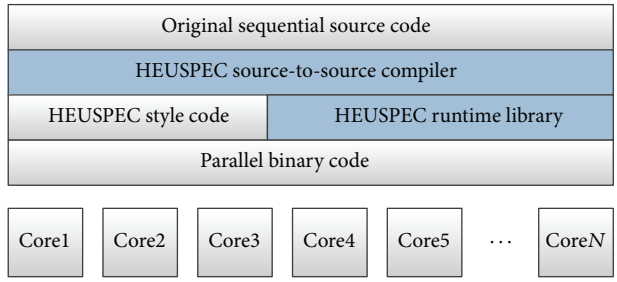


Figure 1: The hierarchy structure of HEUSPEC model (the shadowed parts are the HEUSPEC source-to-source compiler and the runtime library).

The HEUSPEC abstract code structure is shown in Figure 2. The HEUSPEC model has one main thread and multiple speculative threads. The main thread executes the HEUSPEC_MAIN_BODY, which includes several control modules handling management work, such as speculative threads creating, CVARs management, and correctness checking. All the speculative threads are created by the main thread, which run the HEUSPEC_THREAD_FUNC code. There is no communication between the speculative threads. However, a speculative thread can communicate with the main thread via HEUSPEC messages during the speculative reading and correctness checking. As the main thread occupies a processor core during the execution, the upper bound of the speedup of HEUSPEC on an N-core platform is $N - 1$.

For the CVARs involved in dependencies between iterations, HEUSPEC uses the software state isolation mechanism. This mechanism is applied by the CorD [19, 20] parallel model proposed by Tian et al, which is proposed by Tian et al. in 2008, Riverside. Under this mechanism, each CVAR has a committed version and multiple speculative versions. The committed version is stored in the committed memory space, which can only be accessed by the main thread. The speculative versions are generated by the speculative threads and stored in their own private space when they start. Meanwhile, for each CVAR, the mapping relations between the committed version and speculative versions are also created by the speculative thread and recorded in the mapping table, which can be searched by main thread during the correctness checking. The speculative threads can access the speculative versions of CVARs directly in their own private space. The access trace of each speculative thread is recorded in the Read Mapping Table or Write Mapping Table. The initial value of a speculative version of a CVAR is generated by the speculative read mechanism in HEUSPEC (see Section 3.1).

Figure 3 shows the state isolation mechanism in HEUSPEC. We assume that two CVARs in the code section, a and b, are copied to the private space when the speculative threads start. During the parallel execution, the speculative threads can read or modify the speculative version of a and b stored in their own private space, while the committed

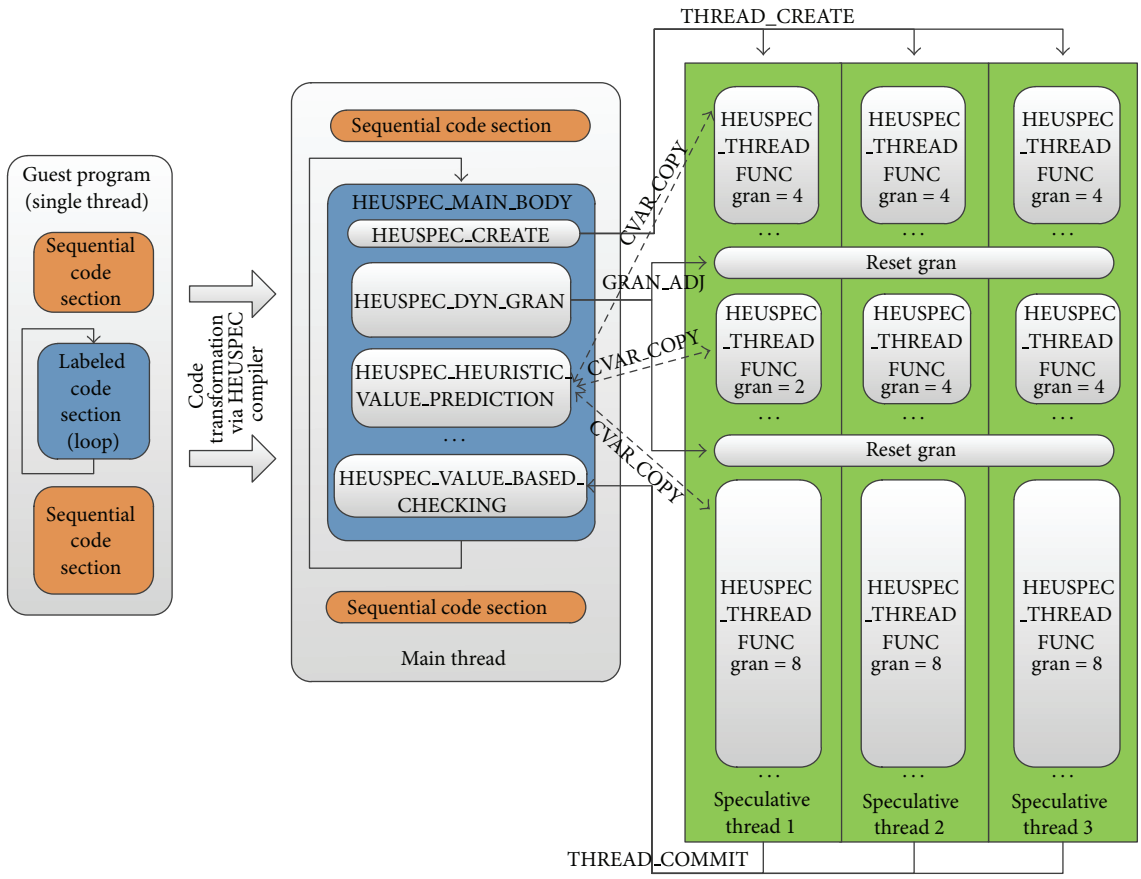


Figure 2: The abstract code structure of HEUSPEC model (between the main thread and speculative thread, there are several kinds of message shown in the figure, such as CVAR_COPY for copying and CVARs and GRAN_ADJ for granularity resizing. The details of the HEUSPEC message are given in Section 4).

version is protected in the shared space. When the computing is finished, the speculative threads send messages asking the main thread for correctness checking. The main thread checks the correctness of each speculative read operation by searching in Read Mapping Table and Address Mapping Table. If there is no misspeculation, the main thread copies each modified speculative version of CVAR in the private space back to the shared space and overwrites its committed version. Or else, the speculative thread rerolls and the speculative versions of CVARs are invalidated.

HEUSPEC adopts dynamic task assignment. For example, if the labeled code section is a loop with N iterations, the main thread at runtime packs several successive iterations into a task and assigns the task to an idle speculative thread. When all the tasks are finished, the main thread confirms that the speculative parallel section is finished and terminates all the speculative threads. Though dynamic task assignment introduces some additional control overheads, it enables the main thread to adjust the granularity of the task at runtime and eliminates unnecessary interim thread creating and killings processes. Therefore, it definitely benefits the overall performance. We proposed dynamic task granularity resizing (DTGR) mechanism based on the dynamic task assignment (see Section 3.3).

To insure the correctness of speculative parallel execution, the speculative parallel model must include a commit mechanism (or conflict detection mechanism), so that the correct task can be committed, and the failed task can be rerolled. Most of the conventional speculative parallel models apply version based correctness checking mechanism. Under this mechanism, each copy of CVAR has its own version number: speculative version numbers for the speculative versions and committed version numbers for the committed versions. The speculative version number can be modified during a speculative writing to the CVAR. While the committed version number can only be modified while a task commits successfully. During the correctness checking, for each CVAR, the speculative version number recorded in the RAT will be compared to the current committed version number, so as to determine the correctness of the task.

In HEUSPEC, to support HVP, we must change the conventional version based correctness checking mechanism to the value based correctness checking. This mechanism was applied in the BOP [16, 17] proposed by Ding et al. in 2007 to reduce some avoidable rollback caused by written but not changed CVARs. The key idea of the value based correctness checking is that during the correctness checking, for each CVAR, the values instead of the version numbers

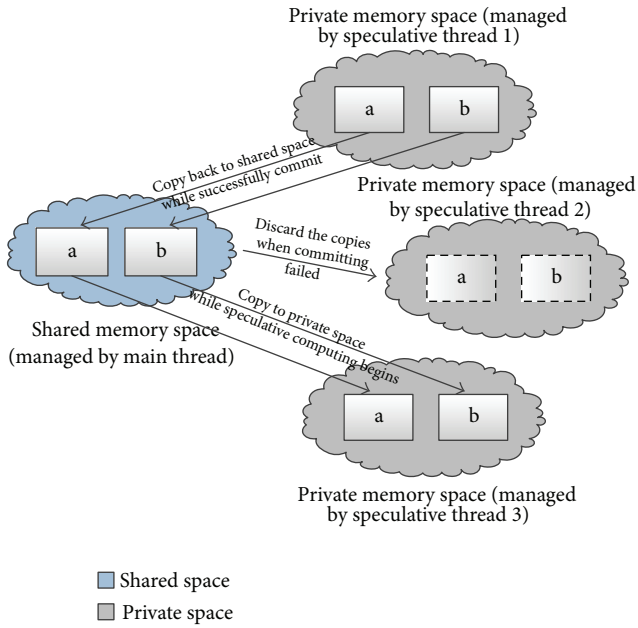


Figure 3: The state isolation mechanism (in the figure, we assume that the speculative thread 1 has successfully committed, the speculative thread 2 has failed during committing, and speculative thread 3 has just initialized. Therefore, the figure shows the functions of state isolation mechanism in 3 different cases, namely, copying back CVARs to shared space, discarding the CVAR copies, and copying CVAR to private space).

of the speculative versions are compared to the value of the committed version to determine the correctness of a speculative task during the correctness checking. To support this, some changes in the structures of global tables (Address Mapping Table, Read Mapping Table, and Write Mapping Table) must be applied. For example, the version number fields in the tables are replaced by the size of CVAR fields. And additional memory space is required to store the speculative versions generated by HVP. For the details about the global tables, see Section 4.1.

3. Significant Techniques in HEUSPEC

The conventional software-based speculative parallel models are diversified in the implementations of conflict detecting and conflict solving mechanisms. However, from most models, the 3 factors that affect the performance can be abstracted, namely, the misspeculation rate, the average rollback overhead, and the average controlling overhead. The relationship between the global overhead and the 3 factors is as follows:

$$O_{\text{global}} = N_{\text{task}} \times \left(O_{\text{Control In Average}} + R_{\text{miss}} \times O_{\text{Rollback In Average}}\right). \quad (1)$$

The variables used in the equation are shown as follows:

(i) O_{global}: global overhead of the model,

(ii) N_{task}: total number of tasks,

(iii) $O_{\text{Control In Average}}$: average control overhead of each task,

(iv) $O_{\text{Rollback In Average}}$: average reroll overhead of each task,

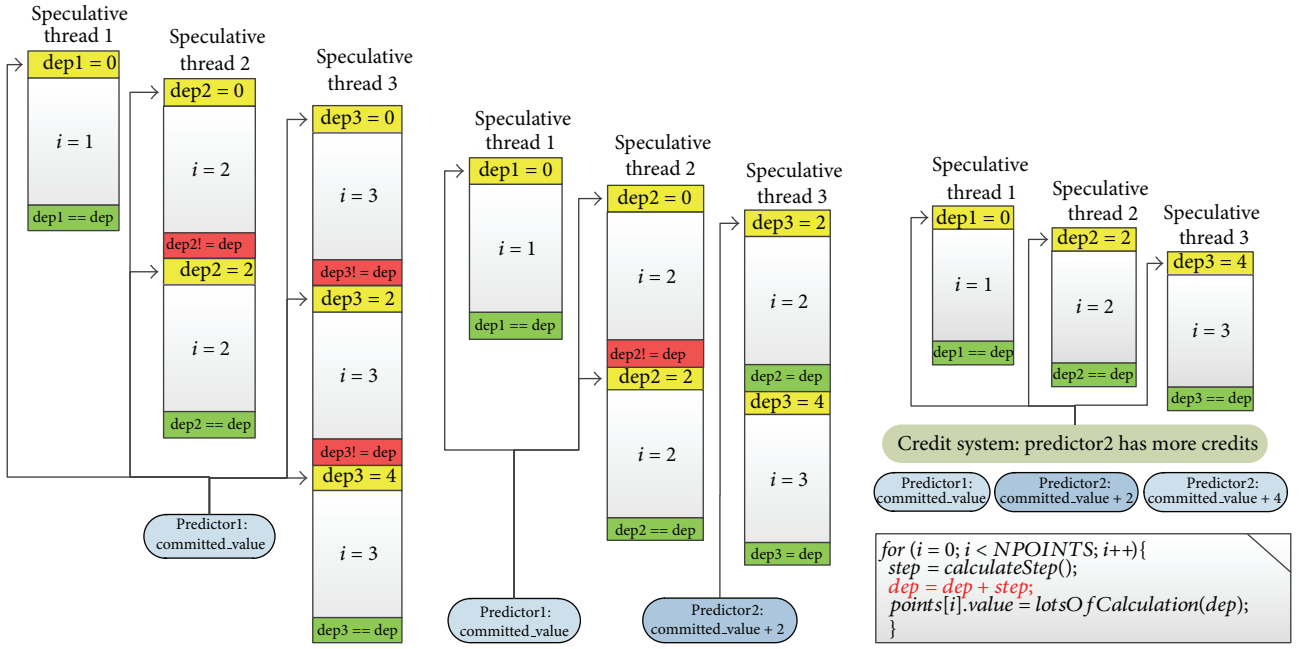
(v) R_{miss}: misspeculation rate.

From (1), we can conclude that there are 3 ways to reduce O_{global}, namely, to reduce $O_{\text{Control In Average}}$, $O_{\text{Rollback In Average}}$, or R_{miss}. In fact, there is tradeoff between the 3 factors; take CorD as an example; it uses precomputing to reduce R_{miss} and checkpoint mechanism to reduce $O_{\text{Rollback In Average}}$; however, both of them increase the $O_{\text{Control In Average}}$ remarkably. In the HEUSPEC model, we proposed 2 key technologies to overcome the defects in the conventional models. The heuristic value prediction (HVP) mechanism can reduce the high R_{miss} in the guest program with low cost, and the dynamic task granularity resizing mechanism manages to balance the $O_{\text{Control In Average}}$ and the $O_{\text{Rollback In Average}}$ in order to optimize the O_{global}.

3.1. Heuristic Value Prediction. The misspeculation rate (R_{miss}) is tightly correlated with the global overhead. However, conventional speculative parallel models without value prediction have high R_{miss} while executing a loop with dependencies. Take the code section in Figure 4(c) as an example; the loop in the figure has lots of potential parallelism; however, a CVAR *dep* is within the loop. Using conventional model, we assume that *dep_privateN* is the speculative version of CVAR *dep* in the speculative thread *N*. If there is no explicit synchronization, the conventional model always copies the value of committed *dep* in the shared space when generating *dep_privateN*. This will cause many conflicts, make the task reroll frequently, and impact the performance of paralellized code section seriously.

To solve the problem, some previous works adopted value prediction schemes [18, 20]. However, most of them use random algorithm correlated with multiple execution (more than one processor to execute the same loop iteration) scheme in the value prediction, which is processor-consumptive and lowers down the upper limit of the overall speedup. In this paper, we try to find an effective and less processor-consumptive way to lower the R_{miss}, hoping that the speculative read mechanism can be more "rational," that is, to predict the value validly with some information such as loop index or history values rather than predict blindly. Therefore we proposed heuristic value prediction (HVP). We add a group of value predictors in the conventional model. Just as Figure 4(c) shows, for a single CVAR, each predictor predicts its value by a specific rule. A credit system is created to evaluate the "validity" of all the predictors. The speculative thread always adopts the value from the predictor with more credits.

The effectiveness of the HVP depends on two aspects. The first aspect is the predictability of the CVARs. If the value changing trace of a CVAR follows a specific rule potentially during the sequential execution, the variable is considered to be predictable. The second aspect is that whether the rule matches a specific predictor. If they are matched, the predictor



(a) Conventional model without prediction (equivalent to predicting by committed value)

(b) Model with multiple prediction and execution

(c) Heuristic value prediction in HEUSPEC model

FIGURE 4: Prediction scheme in 3 different kind of models. (The code section is shown in (c). The bold line is the sentence that causes the dependency.)

will probably pass the correctness checking. Therefore, for those CVARs whose values change randomly, the HVP hardly improves the R_{miss}. However, for those predictable CVARs, the mechanism can reduce the R_{miss} remarkably. For example, assume that the function *calculateStep()* in the code section in Figure 4(c) always returns 2; the value of the CVAR *dep* presents in a linear form. Therefore, a simple linear predictor can match it, and the pitfall of rollback introduced by *dep* will be reduced considerably, just as Figure 4(c) shows.

To apply the HVP, two hypotheses should be proved. First, there are quite a number of "predictable" CVARs in the practical applications. Second, the values of these CVARs can be predicted by some simple methods with low overheads, so that the prediction will not increase the overhead of the model too much. To prove them, we carried through an investigation to a series of applications. We found the CVARs and categorized them by their value changing rules.

Figure 5 shows typical examples of 6 categories. The example variables are selected from a loop in the benchmark *256.bzip*. The loop has 97 iterations. Each subgraph shows the value changing trace of a single CVAR during the loop execution. Generally speaking, variables with random value changing traces are hard to predict, and the variables in the CONSTANT, BOOLEAN, LADDER, and LINEAR category are easier. In the selected benchmarks in our investigation, the CVARs of the last two categories (RANDOM and RESTRICTED RANDOM) account for about 19%; the rest are of the former four categories (CONSTANT, BOOLEAN, LINEAR, and LADDER). Obviously, for the CONSTANT and the LADDER category, the conventional mechanism which uses the committed value of the CVAR is the best. The BOOLEANs can be predicted with the mechanism similar to the Branch Predicting Buffer in the microprocessors. For the LINEARs, if the trace of their value can be learned, they can be predicted precisely by linear extrapolation.

Through this investigation, we can get the basic ideas of the HVP. First, among all the CVARs, there are several "predictable" CVARs, whose values are changing regularly in the loop. Second, a predictable CVAR's value can be predict through a low cost way with their history value, such as linear prediction or bool prediction. Third, the changing rule of the value of a predictable CVAR is probably steady in a period. Based on these three ideas, we developed the HVP. First, we build a group of predictor, in each of which implemented a low cost prediction way. Once a speculation read (a speculative thread reading a CVAR's value, may cause a misprediction) happens, each predictor generates a speculative value of the CVAR. One of these values will be selected as the result of the speculative read. As the changing rule of the value is probably steady, sometimes the value of a CVAR may be "catched" by a certain predictor; therefore, we can use a mechanism like scoreboard to evaluate which predictor is most probably matching the CVAR. This mechanism is called the "credit system" which is described in Section 3.2.

3.1.1. HEUSPEC Predictors. Based on the analysis above, we designed the HVP predictors. Figure 6 shows the 4-field structure of a HVP predictor. During the prediction,

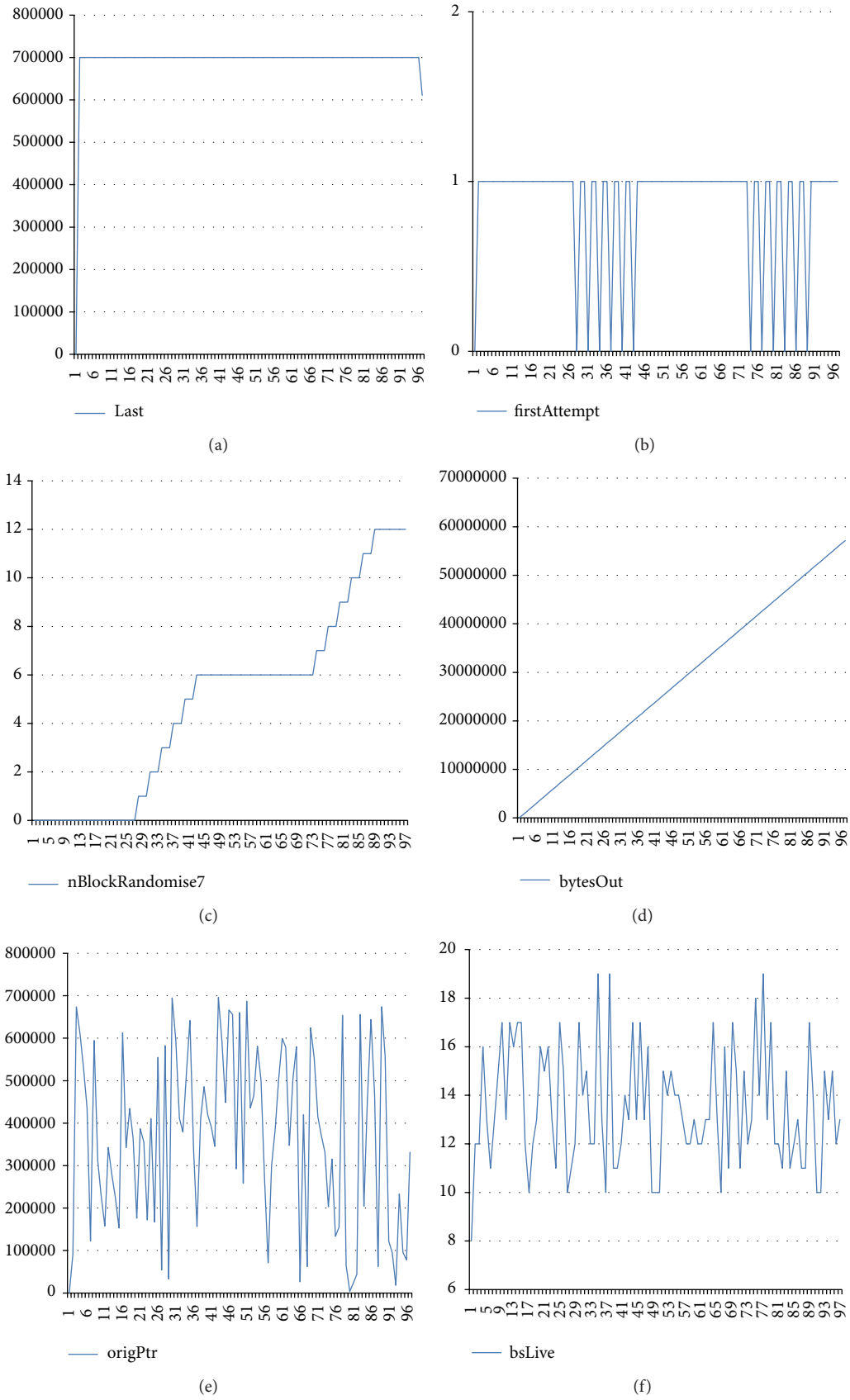


Figure 5: 6 different patterns of CVAR's value changing.

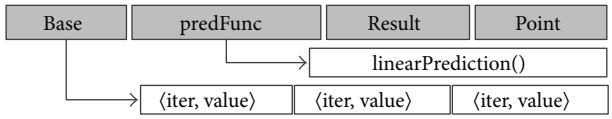


Figure 6: The structure of a HEUSPEC predictor (a single predictor includes four fields, namely, the *Base* field which points to an array storing the information used in prediction, the *predFunction* field which is the entry of the prediction function, the *Result* field which stores the prediction result, and the *Points* field which records the times of correct speculation the predictor has made. The elements in the *Base* array and the *Result* field are ⟨iter, value⟩ pairs. The figure shows a linear predictor. The size of *Base* array is 3, and the *predFunc* points to the function *linearPrediction()*).

the prediction function pointed by *predFunction* pointer generates the *Result.value* based on the information in the *Base* array and *Result.iter*. Most of the time, not all elements in the *Base* array are used. For example, for the conventional predictor and the reversal predictor, only the last committed ⟨iter, value⟩ pairs are used. To simplify the HEUSPEC prediction mechanism, the max element number of the *Base* array is 3.

Table 1 shows all the predictors implemented in the HEUSPEC model. Among these 5 predictors, the conventional predictor inherits the speculation mechanism in the conventional models, which always uses the committed value in the shared space. The reverse predictor is for the Bool type CVARs. In the scheme of the reverse predictor, we assume that the value of the CVAR always reverses between two adjacent iterations. Therefore, the predictor can calculate the speculative value in the current iteration. For example, if the *Base[0]* is $\langle 1, 1\rangle$ which means that the number of the last committed iteration is 1 and the predicted value is 1 and the current iteration number is 2, the *Result* should be $\langle 2, 0\rangle$. The restricted random, linear, and guadratic predictors are for integers. The linear and quadratic predictors take the elements in the *Base* array as a series of points in 2-D space and use them to generate the *Result* via extrapolation method. The restricted random predictor uses the *Base* array to record the upper and lower limit.

3.1.2. Credit System. The commonness of the 5 predictors is that they use, more or less, the history values of the CVARs to guide predictions. However, a single predictor has little probability to make a correct prediction. To augment the probability, the credit system is applied. For a speculative read, each predictor produces a candidate value. Figure 6 shows the structure of the predictors. The *Points* field in each predictor records the correct speculation it has made. Through this, the credit system can quantify the "rational" level of the predictors, and select an appropriate speculative value among them.

Figure 7 shows the workflow of the credit system. During the speculative read process, each predictor generates a speculative value candidate (*depN* in the figure) for *dep*. Only the value generated by the predictor with the highest point is selected via the select function HEUSPEC_selectPredictor(). If multiple predictors have the same highest point, the select function selects one from them randomly. Once the speculative value is selected, it is returned to the speculative read function and used in the calculation of the speculative thread. Finally During the commit process, the commit function *HEUSPEC_commit()* compares all the generated speculative value, no matter used or not, with the committed value in the shared space. If a predictor did a correct prediction, it gains an additional 1 point. Therefore, the HVP can generate speculative values of CVARs more rationally, making use of their history values.

Although the HVP cannot insure that the predictions are always correct, it can improve the speculation accuracy by a certain extent. Especially for those CVARs whose values changing pattern matches a given predictor, most of the rollback can be eliminated. The evaluation and result of HVP are discussed in Section 5.1.

3.2. Dynamic Task Granularity Resizing. The Dynamic task granularity resizing (DTGR) is intended to reduce the global overhead of the HEUSPEC model and balance the load. Models with static task granularity always suffer from much additional overhead and imbalanced load. As Figure 8 shows, two different code sections (DEPENDENCY = 1 or not) are executed under a conventional model with static task granularity. Obviously, for a high misspeculation rate (R_{miss}), granularity should be lowered down to avoid additional rollback overhead. However, for a low R_{miss}, finer tasks will break the continually of the calculation and cause a lot of control overhead, so in this case the bigger granularity is better. In fact, the appropriate granularity task is related not only to the rollback rate, but also to the computation in the parallel section and control overhead caused by task creating and committing. Therefore, to find the appropriate granularity, we use the dynamic optimization technique.

We believe that the global speculation overhead can be optimized to adapt the runtime behavior of a program via dynamic optimization. In Section 3, we have created the analytical model of global overhead in the HEUSPEC, which is described by (1). We have analyzed that the global overhead depends on the 3 factors, namely, the average control overhead ($O_{\text{Control In Average}}$), the misspeculation rate (R_{miss}), and the average rollback overhead ($O_{\text{Rollback In Average}}$). In this section we give a further discussion about the O_{global}. The variables used are listed:

(i) N_{task}: total task number,

(ii) N_{iter}: total iteration number of the loop,

(iii) R_{miss}: miss rate,

(iv) N_{miss}: total miss time of the loop,

(v) N_{thread}: total number of speculative threads,

(vi) gran: task granularity,

(vii) $O_{\text{Control In Average}}$: average control overhead of each task,

(viii) $O_{\text{Rollback In Average}}$: average rollback overhead of each task,

(ix) $O_{\text{Rollback Per Iter}}$: rollback overhead of a single iteration.

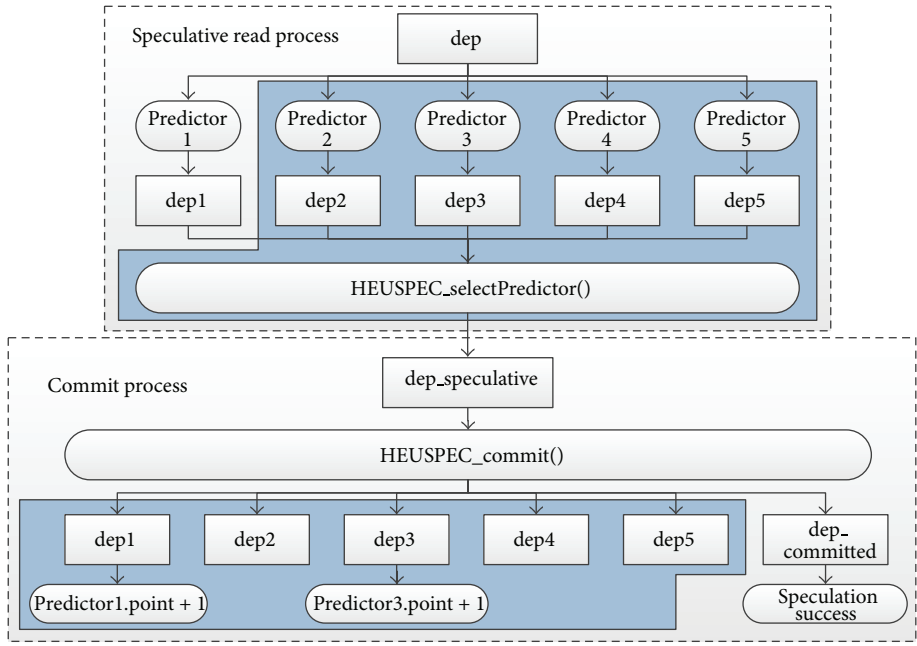


Figure 7: The speculative read and commit processes of heuristic value prediction. (We use the code section in Figure 4. The shadowed part of the figure shows the new workflow introduced by the heuristic value prediction. The unshadowed part is the original speculation-commit workflow of the conventional speculation model.)

Table 1: The HEUSPEC predictors.

Name	Base elements used	Especially applicable category	Speculative value depends on	Example
Reverse	1	BOOLEAN	The reverse of the value in the last iteration	Base[0] = ⟨1, 1⟩ **Result** = ⟨**2**, **Reverse**(**1**)⟩
Conventional	1	CONSTANT, LADDER	The committed value in the shared space	Base[0] = ⟨1, 135⟩ **Result** = ⟨**2**, **135**⟩
Restricted random	2	RESTRICTED RANDOM	A random number in a restricted value space	Base[0] = ⟨1, 20⟩ Base[1] = ⟨2, 5⟩ **Result** = ⟨**3**, **Rand**(**5**, **12**)⟩
Linear	2	LINEAR	The linear extrapolation of last 2 committed values	Base[0] = ⟨1, 20⟩ Base[1] = ⟨2, 5⟩ **Result** = ⟨**3**, **Linear Extra**(**1**, **20**, **2**, **5**)⟩
Quadratic	3	LINEAR, CONSTANT	The quadratic extrapolation of last 3 committed values	Base[0] = ⟨1, 20⟩ Base[1] = ⟨2, 5⟩ Base[2] = ⟨3, 8⟩ **Result** = ⟨**4**, **Quad Extra**(**1**, **20**, **2**, **5**, **3**, **8**)⟩

In a period, if task granularity is constant, we have $N_{\text{task}} = N_{\text{iter}}/\text{gran}$, $R_{\text{miss}} = N_{\text{miss}}/N_{\text{task}}$, and $O_{\text{Rollback In Average}} = \text{gran} \times O_{\text{Rollback Per Iter}}$; therefore, we can transform (1) as follows:

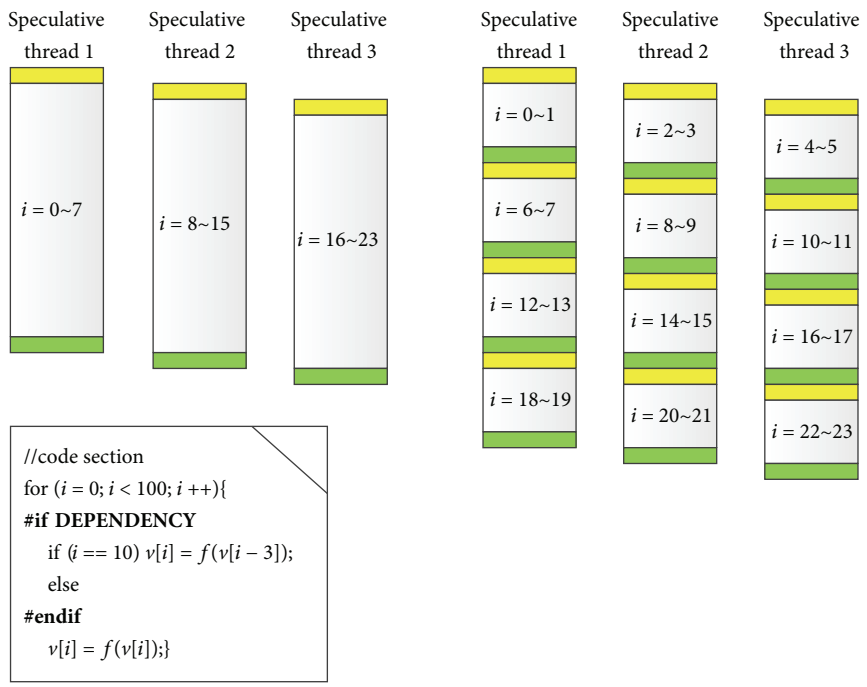
$$O_{\text{global}} = \left(\frac{N_{\text{iter}}}{\text{gran}}\right) \times \left(O_{\text{Control In Average}} + \left(\frac{N_{\text{miss}}}{(N_{\text{iter}}/\text{gran})}\right) \times \text{gran} \times O_{\text{Rollback Per Iter}}\right). \quad (2)$$

Equation (2) can be simplified as follows:

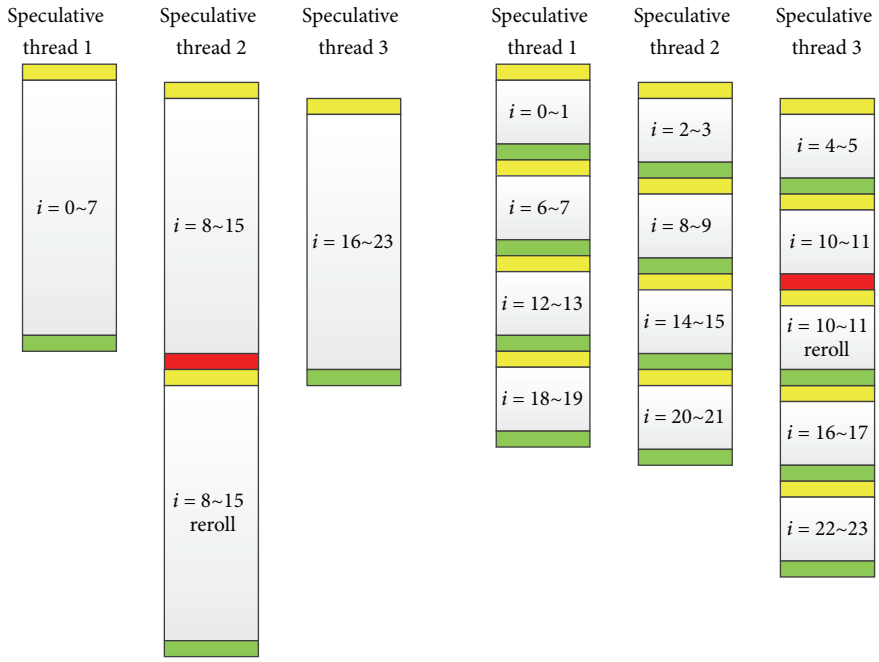
$$O_{\text{global}} = \left(\frac{N_{\text{iter}}}{\text{gran} \times O_{\text{Control In Average}}}\right) + \left(N_{\text{miss}} \times \text{gran} \times \text{O}_{\text{Rollback Per Iter}}\right). \quad (3)$$

Since gran ≥ 1, to minimize the GO, we derive both sides of (3) by gran. Therefore we have

$$O'_{\text{global}}(\text{gran}) = \left(-N_{\text{iter}} \times O_{\text{Control In Average}} \times \frac{1}{\text{gran}^2}\right) + \left(N_{\text{miss}} \times O_{\text{Rollback Per Iter}}\right). \quad (4)$$



(a) Execution with DEPENDENCY = 0 (bigger task granularity wins)



(b) Execution with DEPENDENCY = 1 (smaller task granularity wins)

Figure 8: The execution flow of a code section under the conventional speculative model with static task granularity (the two parts of the figure share the same code section, in which the "#ifdef" part in the code section brings a dependency. The shadowed part in the task shows the control overhead brought by task creating and committing. The (a) part of the figure shows the execution flow with DEPENDENCY = 0, while the (b) part shows the execution flow with DEPENDENCY = 1. The bigger task granularity is 8, while the smaller task granularity is 2).

Let $O'_{\text{global}}(\text{gran}) = 0$; if $N_{\text{miss}} \neq 0$, we have

$$\text{gran} = \sqrt{\frac{\left(N_{\text{iter}} \times O_{\text{Control In Average}}\right)}{\left(N_{\text{miss}} \times O_{\text{Rollback Per Iter}}\right)}} \quad (N_{\text{miss}} \neq 0), \tag{5}$$

From above analysis, we can make conclusion. If we take a certain number of iterations as an adjusting period (AP), we optimize gran according to (5). During an adjusting period, the N_{iter} is a constant, while the $O_{\text{Control In Average}}$, the $O_{\text{Rollback Per Iter}}$, and the N_{miss} can be calculated by a group of

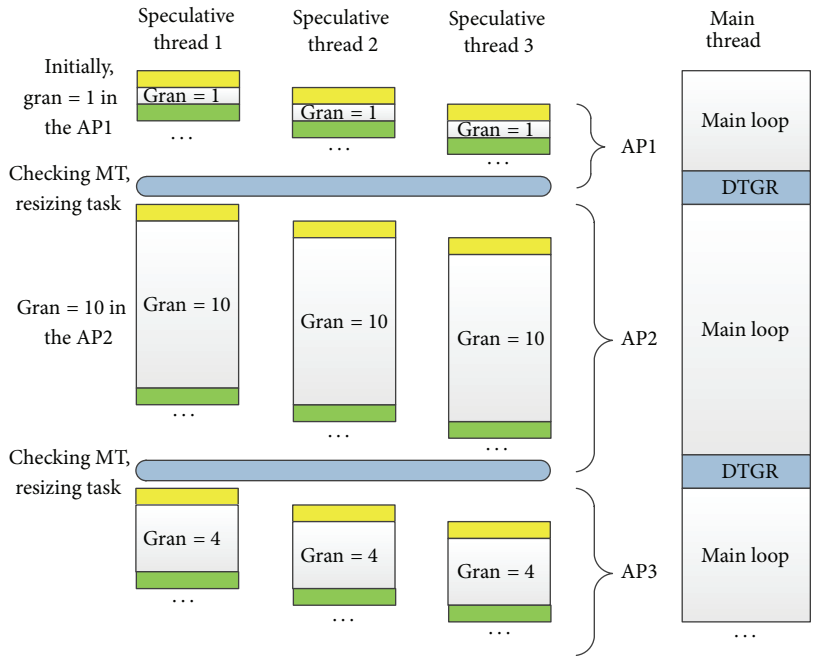


Figure 9: Dynamic task granularity resizing (in the speculative thread columns, the shadowed parts represent the control overheads).

profiling counters at runtime. Therefore, we can calculate the optimized gran dynamically. However, since the N_{miss} can be zero and the calculation may fail, in that case, we prescribe that when $N_{\text{miss}} = 0$, we let gran $= N_{\text{iter}}/N_{\text{thread}}$. N_{thread} represents the number of speculative thread.

In the HEUSPEC, the DTGR is implemented as a module embedded in the HEUSPEC_MAIN_BODY (shown in Figure 2), which is executed by the main thread. Initially, we set gran to a certain value (in the experiment in Section 5, it is 1) then recalculate and update the gran at the beginning of every adjusting period. After that, the succeeded tasks are assigned with the granularity equal to the newly updated gran. The DTGR assures that tasks are assigned betimes to idle speculative threads. Figure 9 shows the details of DTGR. We let a constant number of iterations be an adjusting period (AP). The main thread executes task granularity resizing function to adjust the task granularity with dynamically calculated N_{iter}, $O_{\text{Control In Average}}$, N_{miss}, and $O_{\text{Rollback Per Iter}}$. After resizing, the main thread assigns task with new granularity.

Through the DTGR, programmers are able to optimize the global overhead and balance the load without considering the task partition scheme. According to our experiment, however, some profiling information such as $O_{\text{Control In Average}}$ and $O_{\text{Rollback Per Iter}}$ can just reflect the average behavior of the loop iterations in a constant time period. Therefore, this method can assuredly help the programs without much computation difference across iterations (such as the loops in *MatMul* or *LU*). For those loops with large across-iteration computation difference, we cannot ensure that the granularity will have converge to the best value. But this method includes the dynamic assignment, with which the performance loss can be made up, and we still can get an acceptable speedup. The details of the experiment results and analysis are given in Section 5.

3.3. Other Optimizations. Besides the two key techniques, we have adopted some other optimizations. The out-of-order confirming mode is used for those benchmarks without dependencies. Because there is no dependency between the adjacent speculative threads, thus the implicit synchronization in the in-order confirming mode can be eliminated. The on-the-fly copying is adopted for those read-only CVARs. If a CVAR is read-only in the parallel section, it is not necessary to generate its speculative version. Instead, we can use the committed version directly. The out-of-order confirming mode and on-the-fly copying are adopted in the HEUSPEC model, in order to further reduce its time and memory cost.

4. Implementation

As Figure 1 shows, the HEUSPEC includes a source-to-source compiler and a runtime library. We implemented the compiler via LLVM framework and the runtime library via POSIX thread library in ANSI C. The HEUSPEC model is implemented based on the traditional software-based speculative model. The two key techniques proposed in Section 3 are implemented in the runtime library. In this section we show details of the global tables and the implementation of the HEUSPEC.

4.1. Global Tables. We have introduced in Section 2 that the state isolation mechanism is the basic mechanism of

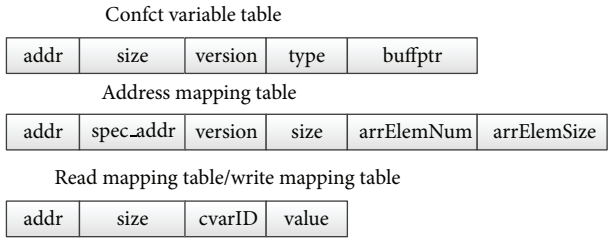


Figure 10: The structures of the global tables (there are only one Conflict Variable Table. The number of Address Mapping Tables and Read Mapping Tables/Write Mapping Tables is equal to the N_{thread}).

the HEUSPEC model. To manage the different versions of CVARs, it is necessary to create the mapping relations between the speculative versions and committed version for each CVAR. Meanwhile, the access information should be recorded, too. Therefore, the sufficiency of the information for the correctness checking and task committing can be insured. In the HEUSPEC model, the information related to speculative execution is stored in several global tables. Figure 10 shows the structures of the global tables.

The *Conflict Variable Table*, which can only be accessed by the main thread, is used to store the basic information of the CVARs, such as address (for the committed version), size, or type. For each CVAR, there is a record in the CVAR Table.

The *Address Mapping Table* is used to maintain the mapping relation between the addresses of committed version (*addr*) and the speculative version (*spec_addr*). Each speculative thread has its own Address Mapping Table.

The *Read Mapping Table* and *Write Mapping Table* are used to keep the record of accesses to the CVARs for each speculative thread. The *value* field in the two tables is used to store the speculative value of the CVARs while being read or written. This value will be used in the committing process.

4.2. HEUSPEC Style Code. The HEUSPEC source-to-source compiler can translate the labeled C code into HEUSPEC style C code. During this process, the original C code is transformed, mixed with HEUSPEC runtime library function, and finally transformed to the code which can be parallel executed. In Figure 2 we have shown the abstract code structure of the HEUSPEC code. Algorithm 1 shows an example of the code transformation of a real benchmark (*Pi* in the OmpSrc 2.0).

5. Experiments and Evaluation

To test the performance of the HEUSPEC model and prove its advantage comparing with the conventional model, we designed and carried on a series of experiments. We choose the hardware platform with two Xeon5450 processors, which have 4 processor cores. The capacity of the memory is 24 GB. The software environment includes a Linux OS (kernel version 2.6.32) and a C compiler. We chose the benchmarks of different CVAR numbers. Table 2 lists the benchmarks we used in the experiments.

Table 2: The benchmarks used in experiment.

Name	Package	CVAR number
backprop	Rodinia 2.1	5
heartwall	Rodinia 2.1	2
kmeans	Rodinia 2.1	8
hotspot	Rodinia 2.1	10
leucocyte	Rodinia 2.1	16
srad_v1	Rodinia 2.1	13
lavaMD	Rodinia 2.1	5
adpcm	mibench	2
susan-s	mibench	8
183.equake	Spec2K INT	5
179.art	Spec2K INT	13
LU	OmpSrc 2.0	4
mandelbrot	OmpSrc 2.0	3
MD	OmpSrc 2.0	7
Pi	OmpSrc 2.0	2
blackscholes	Parsec 2.1	0
badloop	self-coded	1
MatMul	self-coded	0

We have designed four experiments to test the performance of the HEUSPEC. First, to show how speculation accuracy improved by HVP; we did the experiment and gathered the miss rate with two typical benchmark on multiple levels of task granularity. Second, to reflect the performance gain by the HEUSPEC, we have contrasted the performance speedup of a program executed under the HEUSPEC against that under the conventional model. Third, to reflect the scalability of the HEUSPEC, we have tested the speedup of each benchmark under the HEUSPEC as the speculative depth (the number of concurrent speculative threads) increases. Forth, we have tested the control overhead introduced by the HEUSPEC.

5.1. Speculation Accuracy Improvement. We choose the benchmark *badloop* and *fluidanimate* and run them under the HEUSPEC. To show the miss rate improvement with different task granularity levels, we shut down the DTGR. For *badloop*, we choose 6 levels of task granularity, and for *fluidanimate*, we choose 3 levels of task granularity.

Figure 11 shows the experiment result: the miss rate of badloop reduced by 12.9% on the average and the miss rate of fluidanimate reduced by 25.6%. The experiment shows that the HVP actually reduced the miss rate of speculation.

5.2. Speedup. Figure 12 shows the speedup of each benchmark under the HEUSPEC against that under the conventional model. In this experiment, we run the benchmarks under the conventional model (without HVP and DTGR) and the HEUSPEC, respectively. We used out of order confirm mode for *MatMul*, *lavaMD*, *adpcm*, and *blackscholes*,

(a) original labeled C code

```
int main(int argc, char * argv[]) {
  ...
#pragma heuspec parallel for (w, pi)
  for (i = 0; i < N; i++) {
      local = (i + 0.5) * w;
      pi += 4.0/(1.0 + local * local);
  }
  ...
}
```

(b) HEUSPEC style C code

```
int main(int argc, char * argv[]) {
  ...
  #ifdef ENABLE_HEUSPEC
  HEUSPEC_register_cvar_in_version_table(&w, sizeof(double), 1, NORMAL);
  HEUSPEC_register_cvar_in_version_table(&pi, sizeof(double), 2, RD_PLUS);
  HEUSPEC_main_body(&threadfunc, N),
  #else
  for (i = 0; i < N; i++) {
      local = (i + 0.5) * w;
      pi += 4.0/(1.0 + local * local);}
  #endif
...}
```

(c) HEUSPEC_main_body

```
int HEUSPEC_main_body(void * (*threadfunc)(long *)) {
  HEUSPEC_initiallization();
  for (i = 0; i < NUM_THREAD; i++) {
    HEUSPEC_create_thread(I, threadfunc);
```

Algorithm 1: Continued.

```
    HEUSPEC_assign_task(i); }
  for
(i = 0; i < NUM_THREAD&&exceed_flag[i]! = 1; i = (i + 1)%NUM_THREAD) {
    while (1) {
      j = HEUSPEC_catch_message(msg_buffer, i);
      if (msg_buffer[j].type == FINISH && i == j) //in-order commit
            HEUSPEC_commit()
      if (msg_buffer[j].type == EXCEPTION) //do HVP
            HEUSPEC_HeuristicValuePrediction(); }
    if(taskno ≥ AP) //an AP passed, do DTGR
      gran = HEUSPEC_DynamicTaskGranResizing(OC, OR, MT, AP, gran));
    If(taskno > quan || HEUSPEC_break()) {
      set_exceeded_flag(i);
      if(HEUSPEC_check_exceed_flag() == 1) break;
      else continue; }
    HEUSPEC_assign_new_task(); }
  HEUSPEC_main_body_end(); }
```

(d) HEUSPEC_threadFuncion

```
void * HEUSPEC_threadfunc(unsigned long * child_args) {
  double pi, w;
  while (1) {//main loop on threadfunc
    TLS_wait_start_msg(_i);
    mt[_i][0].PAddr = &w;
    mt[_i][1].PAddr = &pi;
    HEUSPEC_getHeadAndTail(&head, &tail, _i);
    pi = 0.0; //it is a RD_PLUS cvar
    TLS_spec_read(&w, 0);
    TLS_spec_read(&pi, 1);
```

Algorithm 1: Continued.

```
for (i = head; i < tail; i++) {
        local = (i * 0.5) * w;
        pi += 4.0/(1.0 + local * local); }
TLS_spec_write(&(pi), 1);
TLS_send_finish_msg(_i);
TLS_wait_confirm_msg(_i);
TLS_task_terminated(taskno);
TLS_send_ready_msg(_i); }
return; }
```

Algorithm 1: A code example optimized by HEUSPEC model. (The benchmark is Pi in the OmpSrc package. (a) the labeled sequential code. The loop is bolded. (b) the HEUSPEC style code after transformation. (c) the HEUSPEC_main_body() function. The HVP and the DTGR are bolded. (d) the HEUSPEC_threadFunc().)

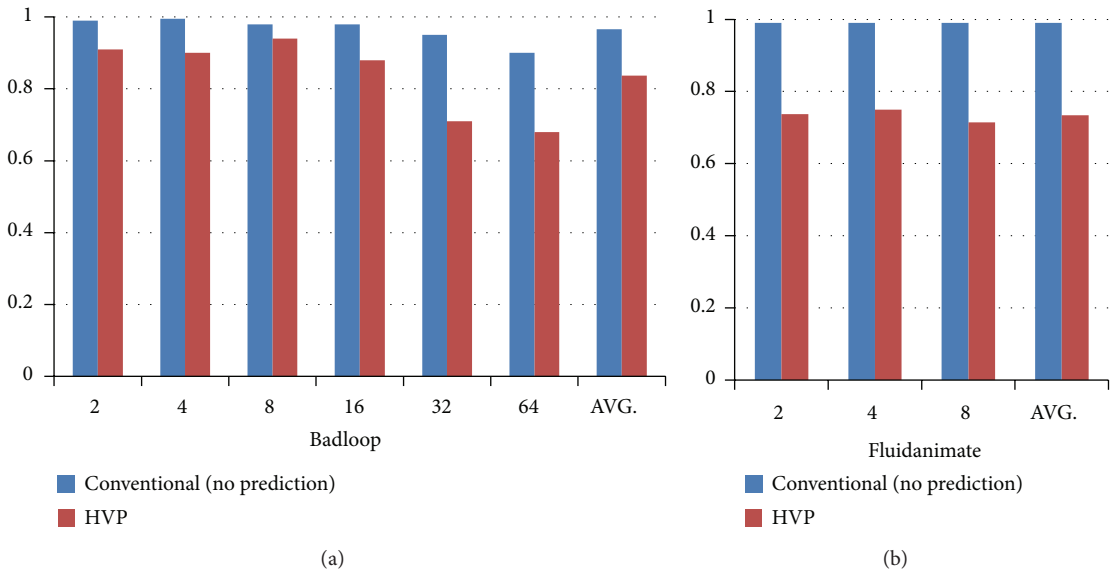


Figure 11: The miss rate under HEUSPEC with HVP.

thus greatly improving the performance of those benchmarks without dependencies. According to our experiment, when the speculative depth is 3, the average speedup of the HEUSPEC is 2.20, about 15% higher than that of the conventional model(1.91). The speedup of the HEUSPEC can reach a high level when the speculative depth is 7, about 4.51 on the average, about 12% higher than that of the conventional model(4.02).

On one hand, the HVP aims at the cross-iteration dependencies of the loop. Therefore, it is more efficient on those benchmarks which have more predictable CVARs. On the other hand, a loop with bigger iteration number and intensive computation can introduce a larger space of optimization for DTGR. Therefore, the DTGR prefers the benchmarks with this feature. From Figure 12, we can see that some benchmarks (*badloop*, *LU*, *Molecular Dynamic*, *Mandelbrot*, and *kmeans*) show remarkable improvement compared with the conventional model (especially with 7 speculative threads). That is mainly because they fit the two conditions we mentioned above. Some benchmarks (*heartwall*, *blackscholes*, *hotspot*, *leucocyte*, *MatMul*, *lavaMD*, and *adpcm*) show a high speedup compared with the serial execution, but little improvement compared with the conventional model. That means that HVP and DTGR are less efficient in these benchmarks, because they have little predictable CVAR, and less optimized space for DTGR. Some other benchmarks (backprop, srad_v1,

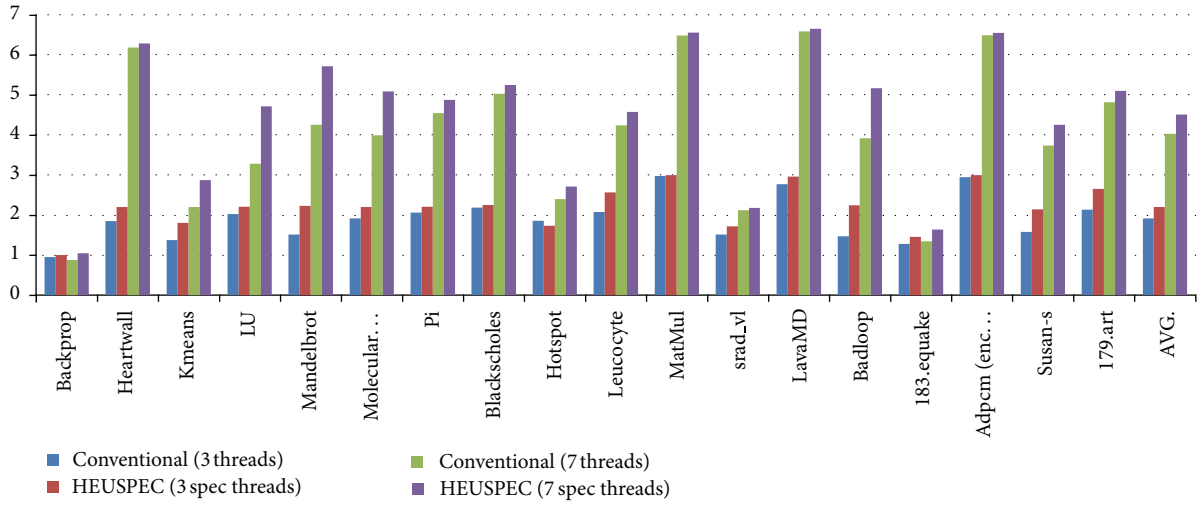


Figure 12: The speedup of HEUSPEC against conventional model. (We show the speedup under the conventional model with 3 speculative threads and the speedup under the HEUSPEC with 3 and 7 speculative threads, resp.)

and 183.equake) have low speedup compared with other benchmarks. That is mainly because the computation in the parallelized loop in these benchmarks is not enough, and the global overhead of HEUSPEC is too much for them.

Due to the unavoidable rollback overhead or the high control overhead, several benchmarks show low speedups under the speculative parallel model, such as *backprop*, *183.equake*, and *srad_v1*. However, most of the benchmarks show remarkable performance gain on this experiment.

5.3. Scalability. Figure 13 shows that the speedup improved along with the speculative depth increases under the HEUSPEC, which can reflect the scalability of the HEUSPEC model to a certain extent. In our experiment, the performances of all the benchmarks improve as the speculation depth increases. Among them, *adpcm* and *MatMul* show better scalability, while some other benchmarks show worse, such as *backprop* or *183.equake*.

The speedup of a benchmark depends on two factors. On the one hand, the rate of the parallel section is relative to the speedup of the benchmark. For example, the parallel section of the *MatMul* benchmark accounts for more than 97% code, while the parallel section of the *backprop* benchmark accounts for less than 30%. Thus the former shows better speedup and scalability than others while the latter performances are worse. On the other hand, the code structure of the parallel section can also influence the speedup. For example, the parallel section of the benchmark *183.equake* is in the function *smvp_opt()*; this function is repeatedly called in another loop in the *main()*. This makes the program call the *HEUSPEC_main_body()* repeatedly, bringing much control overhead. Therefore, it shows a bad speedup and scalability.

5.4. Time and Space Overhead. Figure 14 shows the control overhead introduced by the HEUSPEC. The control overhead includes the time cost on the CVAR copy, task creating and eliminating, correctness checking, and communication between speculative threads and main thread. The experiment is carried with the speculation depth equaling to 7. We compared the result with that of the conventional model. The overall control overhead is 6% on the average, about 7% lower than that of the conventional model. Except several benchmarks such as LU, kmeans, and 181.equake with higher control overhead, for most benchmarks, the control overhead is lower than 4%.

Figure 15 shows the additional space overhead introduced by HEUSPEC. We carried on this experiment with 7 speculative threads and used on-the-fly copying to reduce the memory cost further. According to our experiment results, the average memory cost increased by 21% on the average under the HEUSPEC. Compared to other software-based speculative models, the additional space overhead of the HEUSPEC is much lower.

6. Related Works

The hardware based speculation model has not been widely used due to its limited availability. The researchers concentrated on software speculation mechanism in recent years. To reduce the overhead and to improve the accuracy are the key problems in the software speculation model research in recent years.

Ding et al. have proposed behavior oriented parallelism (BOP) mechanism [16, 17]. In the BOP, the UNIX process is used to encapsulate the speculative thread information. The shared variables are copied to the private space of each speculative thread when the UNIX process is forked. For each CVAR, the BOP allocates a single page to store it. Compared with traditional conflict detecting techniques, the BOP uses value based correctness checking, rather than version based checking, which can avoid some unnecessary rerolls of speculative threads and improve the overall performance. BOP supports DOACROSS parallel model through the dynamic dependence hints.

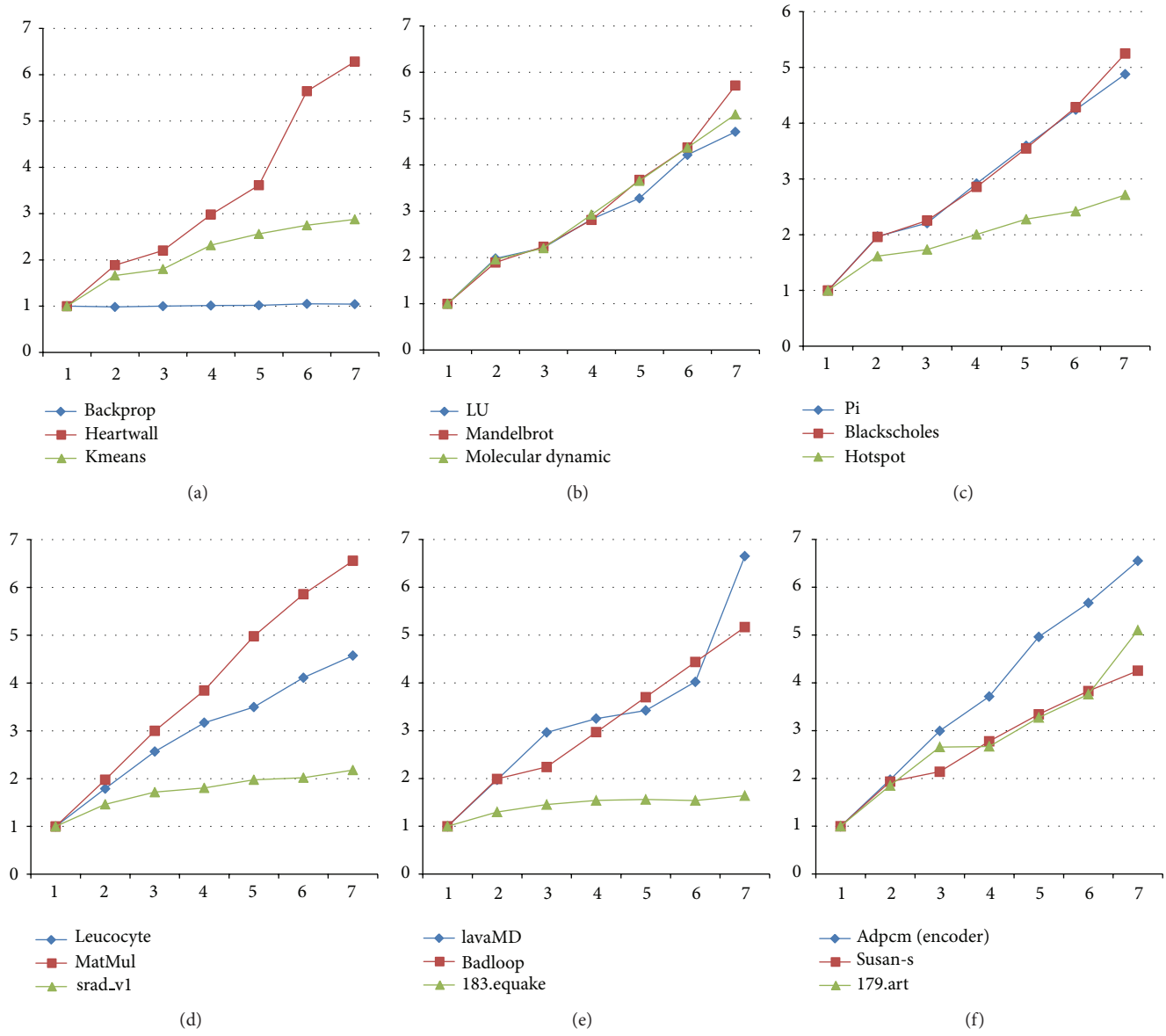


Figure 13: The speedup of the benchmarks under HEUSPEC with the speculation depth changes from 3 to 7.

The copy or discard (CorD) [18, 19] execution model implemented by Tian et al. is another software speculation mechanism. In the CorD, the variables may have dependencies identified by compiler and "copied-in" to the private memory space (P space) of the speculation threads. Conflicts are detected and handled by main threads, which is a manager thread without doing any calculation. Ideally, the overall speedup using CorD can approach to $p - 1$ in a p-core platform. To reduce the misspeculation rate, CorD has brought in the "multiple random value prediction" mechanism, which uses 3 or more predictors to generate the speculative values of the CVARs. Under this mechanism, several processor cores are used to execute same iterations to increase the speculation accuracy. The "pre-computing" technique is also used to improve the accuracy of the speculation.

Liu et al., in University of California Irvine, have performed speculative execution with multiple value prediction on GPUs [20]. Similar to the CorD, for each CVAR, it uses multiple random value prediction mechanism. A single loop iteration may have several copies executed in different threads with different sets of predicted CVAR values. For each loop iteration and its copies, the earliest finished one which passed the correctness checking can submit, while others are discarded. This mechanism can improve the speculation accuracy remarkably with large hardware thread consumption (a task with n CVARs and m possible values for each CVAR may have m^n copies and need the same number of hardware threads to execute them in parallel). With the help of GPU architecture, the number of the predictors is very large. The CVAR values generated by different predictors are

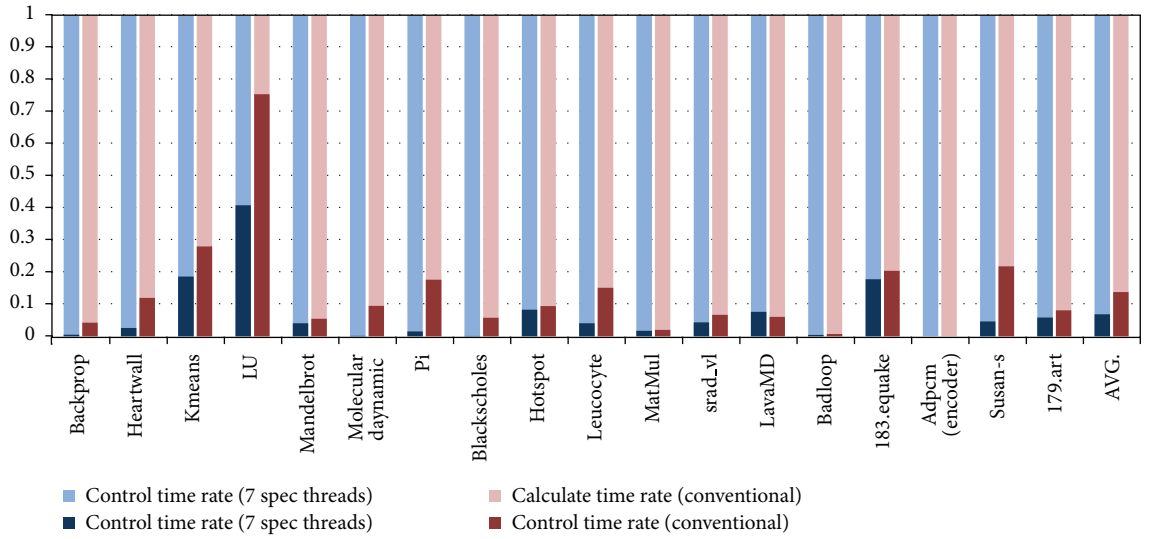


Figure 14: The control overhead introduced by the HEUSPEC model. (The benchmarks with higher control overhead is mainly due to the repeated calling of HEUSPEC_main_body(), such as LU, kmeans and 183.equake.)

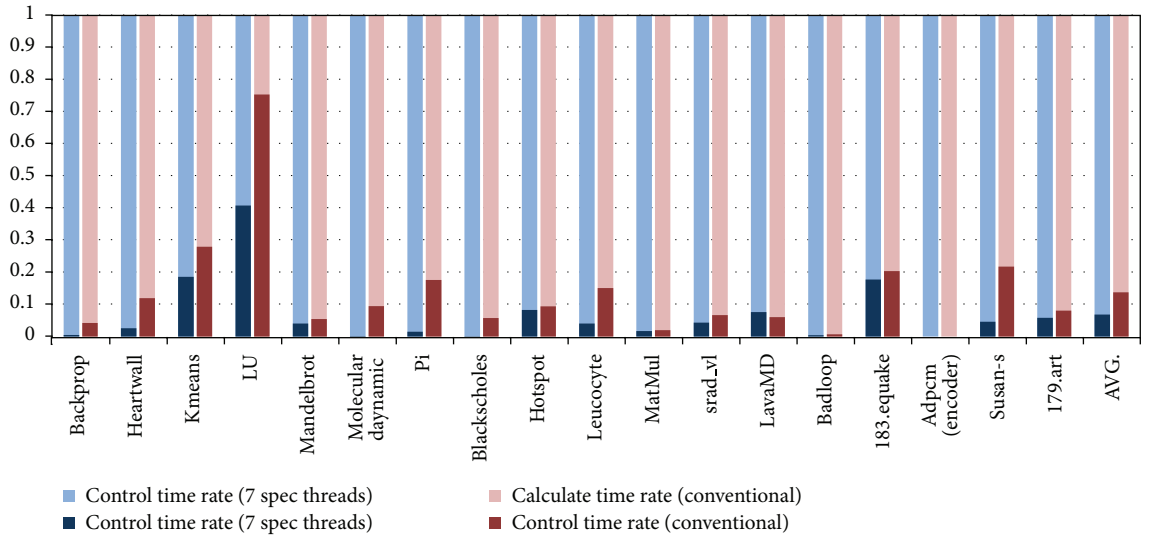


Figure 15: The additional memory cost introduced by HEUSPEC. (The experiment is done with 7 speculative threads. The increasing rate of the memory cost is about 21% on the average.)

mapped to different speculative threads which run in parallel, thus providing a high speculation accuracy and reducing the rollback overhead.

7. Conclusion

We presented a novel speculation parallel execution model: the HEUSPEC. Based on the conventional software speculation parallel execution model, the HEUSPEC adopts 2 key techniques, heuristic value prediction (HVP) and dynamic task granularity resizing (DTGR). The HVP is adopted to reduce the misspeculation rate. The DTGR is implemented to reduce the global overhead and balance the load of the speculative threads. With 18 different benchmarks and 7 speculative threads, our experiments show that the HEUSPEC achieves a speedup of 4.51 on the average (12% higher than conventional model), and 6.56 of the highest on a 8-core platform. The model also shows good scalability and low time and space overheads.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by China National 863 Program (no. 2012AA010905), the National Natural Science Foundation of China (no. 61070037, 61272143, 61103016, and 61202121), the NUDT Innovation Foundation For Excellent Postgraduate (no. B120604), and the Hunan Provincial Innovation Foundation For Postgraduate (no. CX2012B209).

References

[1] J. G. Steffan and T. C. Mowry, "Potential for using thread-level data speculation to facilitate automatic parallelization," in *Proceedings of the 4th International Symposium on High-Performance Computer Architecture (HPCA '98)*, pp. 2–13, February 1998.

[2] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Computational Science & Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[3] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message Passing Interface*, The MIT Press, Cambridge, Mass, USA, 1999.

[4] G. Contreras and M. Martonosi, "Characterizing and improving the performance of Intel Threading Building Blocks," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC '08)*, pp. 57–66, Seattle, Wash, USA, September 2008.

[5] M. Feng, R. Gupta, and Y. Hu, "SpiceC: scalable parallelism via implicit copying and explicit Commit," in *Proceedings of the 16th ACM Symposium on Principles and Practice of Parallel Programming (PPoPP '11)*, pp. 69–79, February 2011.

[6] K. E. Moore, J. Bobba, M. J. Moravan, M. D. Hill, and D. A. Wood, "LogTM: log-based transactional memory," in *Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA '06)*, pp. 254–265, February 2006.

[7] B. Saha, A.-R. Adl-Tabatabai, and Q. Jacobson, "Architectural support for software transactional memory," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '06)*, pp. 185–196, Orlando, Fla, USA, December 2006.

[8] N. Shavit, "Software transactional memory: where do we come from? What are we? Where are we going?" in *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing (IPDPS '09)*, p. 1, Rome, Italy, May 2009.

[9] L. Hammond, V. Wong, M. Chen et al., "Transactional memory coherence and consistency," in *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA '04)*, pp. 102–113, June 2004.

[10] J. G. Steffan, C. B. Colohan, A. Zhai, and T. C. Mowry, "A scalable approach to thread-level speculation," in *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA '00)*, pp. 1–12, June 2000.

[11] M. K. Prabhu and K. Olukotun, "Using thread-level speculation to simplify manual parallelization," in *Proceedings of the 9th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '03)*, pp. 1–12, June 2003.

[12] S. Wang, X. Dai, K. S. Yellajyosula, A. Zhai, and P.-C. Yew, "Loop selection for thread-level speculation," in *Proceedings of the 18th International Workshop on Languages and Compilers for Parallel Computing (LCPC '05)*, pp. 289–303, 2005.

[13] J. T. Oplinger, D. L. Heine, and M. S. Lam, "In search of speculative thread-level parallelism," in *Porceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT '99)*, pp. 303–313, October 1999.

[14] N. Ioannou and M. Cintra, "Complementing user-level coarse-grain parallelism with implicit speculative parallelism," in *Proceedings of the 44th Annual IEEE/ACM Symposium on Microarchitecture (MICRO '11)*, pp. 284–295, December 2011.

[15] L. Hammond, B. A. Hubbert, M. Siu, M. K. Prabhu, M. Chen, and K. Olukotun, "The Stanford Hydra CMP," *IEEE Micro*, vol. 20, no. 2, pp. 71–84, 2000.

[16] C. Ding, X. Shen, K. Kelsey, C. Tice, R. Huang, and C. Zhang, "Software behavior oriented parallelization," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '07)*, pp. 223–234, June 2007.

[17] C. Ke, L. Liu, C. Zhang, T. Bai, B. Jacobs, and C. Ding, "Safe parallel programming using dynamic dependence hints," in *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA '11)*, pp. 243–258, October 2011.

[18] C. Tian, M. Feng, V. Nagarajan, and R. Gupta, "Copy or discard execution model for speculative parallelization on multicores," in *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '08)*, pp. 330–341, Lake Como, Italy, November 2008.

[19] C. Tian, C. Lin, M. Feng, and R. Gupta, "Enhanced speculative parallelization via incremental recovery," in *Proceedings of the 16th ACM Symposium on Principles and Practice of Parallel Programming (PPoPP '11)*, pp. 189–199, February 2011.

[20] S. Liu, C. Eisenbeis, and J.-L. Gaudiot, "Speculative execution on GPU: an exploratory study," in *Proceedings of the 39th International Conference on Parallel Processing (ICPP '10)*, pp. 453–461, September 2010.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 743604, 18 pages
http://dx.doi.org/10.1155/2014/743604

Research Article

On the Performance of Video Quality Assessment Metrics under Different Compression and Packet Loss Scenarios

Miguel O. Martínez-Rach,[1] Pablo Piñol,[1] Otoniel M. López,[1] Manuel Perez Malumbres,[1] José Oliver,[2] and Carlos Tavares Calafate[2]

[1] *Department of Physics and Computer Engineering, the Miguel Hernández University, Avenida de Universidad s/n, Elche, 03202 Alicante, Spain*
[2] *Department of Computer Engineering, Polytechnic University of Valencia, Camino de Vera s/n, Building G1, 46022 Valencia, Spain*

Correspondence should be addressed to Miguel O. Martínez-Rach; mmrach@umh.es

Received 7 January 2014; Accepted 14 April 2014; Published 20 May 2014

Academic Editors: H. Su and W. Su

When comparing the performance of video coding approaches, evaluating different commercial video encoders, or measuring the perceived video quality in a wireless environment, Rate/distortion analysis is commonly used, where distortion is usually measured in terms of PSNR values. However, PSNR does not always capture the distortion perceived by a human being. As a consequence, significant efforts have focused on defining an objective video quality metric that is able to assess quality in the same way as a human does. We perform a study of some available objective quality assessment metrics in order to evaluate their behavior in two different scenarios. First, we deal with video sequences compressed by different encoders at different bitrates in order to properly measure the video quality degradation associated with the encoding system. In addition, we evaluate the behavior of the quality metrics when measuring video distortions produced by packet losses in mobile ad hoc network scenarios with variable degrees of network congestion and node mobility. Our purpose is to determine if the analyzed metrics can replace the PSNR while comparing, designing, and evaluating video codec proposals, and, in particular, under video delivery scenarios characterized by bursty and frequent packet losses, such as wireless multihop environments.

1. Introduction

In the past years, the development of novel video coding technologies has spurred the interest in developing digital video communications, where evaluation mechanisms to assess the video quality play a major role in the overall design of video communication systems.

The most reliable way of assessing zthe quality of a video is subjective evaluation, because human beings are the ultimate receivers in most applications. The mean opinion score (MOS), which is a subjective quality metric obtained from a number of human observers, has been regarded for many years as the most reliable form of quality measurement. However, the MOS method is too cumbersome, slow, and expensive for most applications. Objective quality assessment metrics (QAM) are valuable because they provide video designers and standard organizations with means for making meaningful quality evaluations without convening viewer panels.

Recently, new objective image and video quality metrics have been proposed. They emulate human perception of video quality since they produce results which are very similar to those obtained from subjective methods. Most of these proposals were tested and compared in the different phases carried out by the video quality experts group (VQEG), which was formed to develop, validate, and standardize new objective measurement and comparison methods for video quality. The models that the VQEG forum validates result in International Telecommunication Union (ITU) recommendations and standards for objective quality measurement for both television and multimedia applications [1]. Some of the QAM proposals are designed to be as generalist as possible, that is, to be able to assess quality for a wide set of different

distortion types, while other QAM focus their design on the detection of one, two, or a reduced set of specific distortions.

It would be desirable to find a QAM for image and video that exhibits a good behavior for any set of video and/or image distortions, that is, that detects accurately (as close as possible to the human perceived quality) any distortion regardless of its type and degree. Also, it would be desirable that the time required to obtain a quality measurement is short enough in order to have a practical use or even to be able to use it in real time.

But quality is by definition a highly subjective feature that is influenced not only by the intrinsic characteristics of the signal but also by psychological and environmental factors. Therefore, the task of choosing "the best QAM" is influenced by too many factors and sources of inaccuracy. These sources of inaccuracy are, for example, the reliability of unbiased subjective reference data, the selection of video or image contents, the degree of the impairments and where they appear (in space and time), the procedure used to map between subjective and objective quality values, and even the use and interpretation of the correlation indicators. These factors must be taken into account when making comparisons between metrics [2].

The selection of a QAM may also depend on the target application where it will be used. Examples of applications are a real-time monitor that adaptively adjusts the image quality in a video acquisition or transmission system, a benchmarking image processing system, or even algorithms and encoder proposals that are embedded into image processing systems to decide about the preprocessing and postprocessing stages.

We work with a set of the most relevant quality assessment metrics whose source code or test software has been made available by their authors. So, we can use them in our own evaluation tests.

As mentioned before, we will analyze the behavior of the candidate metrics in two test environments. The first one, is the compression environment, where the quality of compressed sequences at different bitrates with different encoders is compared by means of QAM. The most common way of doing the comparisons between existing image/video coding approaches, improvements over these approaches, or completely new codec designs is by performing a rate/distortion (R/D) analysis. When using R/D, the distortion is usually measured in terms of PSNR (peak signal-to-noise ratio) values, where rates are often measured in terms of bpp (bits per pixel) for images or bps (bits per second) for video. So, in this test environment, we work with the selected QAM as candidates to replace the PSNR as the distortion metric in the R/D comparisons. We will also consider the QAM complexity in order to determine their applicability. The second one is the packet loss environment, where we will analyze the behavior of the candidate metrics in the presence of packet losses under different mobile ad hoc networks (MANET) scenarios. In particular, we are going to compare the behavior of QAM when measuring the quality degradation of an H.264/AVC video delivery in a MANET. We use a hidden Markov model (HMM) to accurately reproduce the packet loss patterns typical of these networks, including variable network congestion levels and different degrees of node mobility. For each particular network scenario, we perform a bitstream erasure process based on the loss patterns suggested by the HMM model. The resulting bitstream is delivered to the H.264/AVC decoder in order to get the resulting HRC that will be used to calculate the QAM value.

The organization of the paper is as follows. In the next section, we will describe the main frameworks addressing objective QAM. In Section 3, we will expose some key aspects of how to compare heterogeneous metrics and the method used to compare the metrics under evaluation. In Section 4, we show the behavior of several available quality metrics in the compression environment. In Section 5, the models and the methods used for the packet loss environment are explained and a behavioral analysis of the metrics is made for different network scenarios. Finally, in Section 6, we present the main conclusions of this work.

2. Objective Quality Assessment Metrics

In the past years, a big effort has been done in the field of QAM. A large number or objective metrics can be found in the literature. Some of them have been designed for a specific kind of distortions, while others are more generalist and try to assess quality regardless of the distortion type. Besides, each metric design is different. Objective evaluation of picture quality in line with human perception is still difficult [3–9] due to the complex, multidisciplinary nature of the problem, including aspects related to physiology, psychology, vision research, and computer science. Nevertheless, with proper modeling of major underlying physiological and psychological phenomena and by obtaining results from psychophysical tests and experiments, it is possible to develop better visual quality metrics to replace nonperceptual criteria as PSNR or MSE being still widely used nowadays.

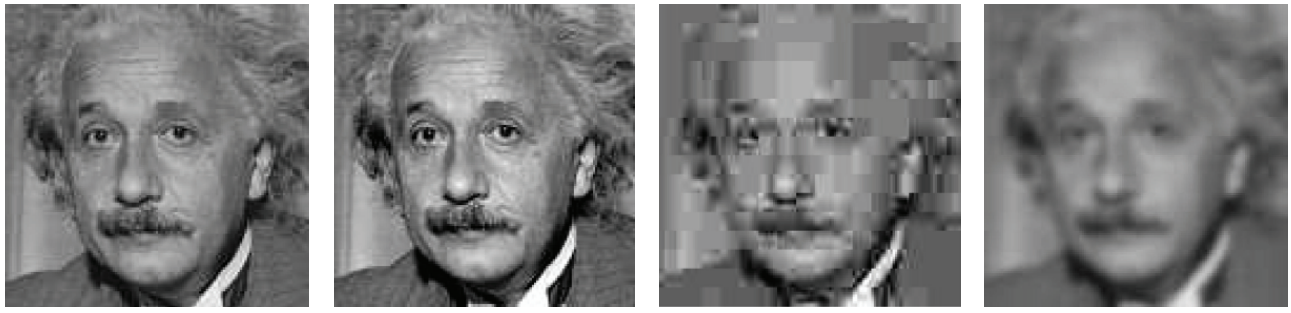
In the literature, we can find different classifications and frameworks that group several QAM depending on the way they are designed. In this section, we will briefly describe the main ideas behind the different frameworks, along with their main QAM.

There is a consensus in a primer classification of objective quality metrics [10, 11] attending to the availability of original nondistorted info (video reference) to measure the quality degradation of available distorted versions.

(i) Full reference (FR) metrics perform the distortion measure with full access to the original image/video version, which is taken as a perfect reference.

(ii) No reference (NR) metrics have no access to the reference image/video. So, they have to perform the distortion estimation based on the distorted version only. In general they have lower complexity but are less accurate than FR metrics and are designed for a limited set of distortions and video formats.

(iii) Reduced reference (RR) metrics have access to partial information about the original video. A RR metric defines what information has to be extracted from original video, so it can be compared with the the same one extracted from the distorted version.



(a) (b) (c) (d)

Figure 1: Example of three figures with different impairments and the same PSNR values: (a) original, (b) contrast stretched 26.55 dB, (c) JPEG compressed 26.60 dB, and (d) blurred 26.55 dB.

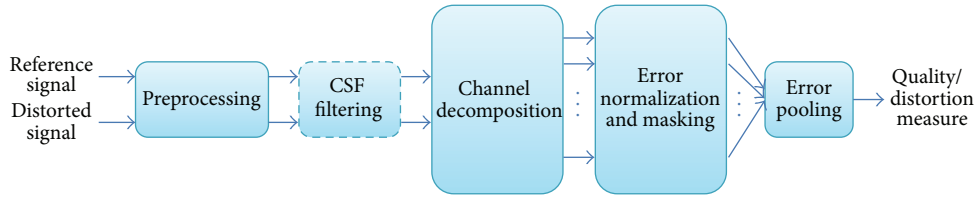


Figure 2: Common block diagram of the error sensitivity framework.

The most widely used FR objective video quality metrics are the mean square error (MSE) and the peak signal-to-noise ratio (PSNR). They are simple and quick to calculate, providing a good way to evaluate the video quality [12]. However, it is well known that these metrics do not always capture the distortion perceived by the human visual system (HVS). In Figure 1, an original image has been distorted in different ways. The PSNR metric gives almost the same value for each distortion, indicating that the quality of the distorted images is the same, but as it can be seen, the perceived quality is different for each image. Moreover, it is not unusual that the perceived quality of image in Figure 1(b) is higher than the one given to the original one, Figure 1(a). That is, a distorted image has better perceptual quality than the original one. If PSNR is used for measuring the quality of the resulting images/videos produced by the different coding proposals, how can we certify that one coding proposal has a better perceptual quality than another?

In this section, we will briefly describe also the main ideas behind the different frameworks and the most relevant and cited QAM of each one. QAM can be classified by many factors such as the metric architecture (number and type of blocks and stages or algorithms used in the metric design), the primary domain (space or frequency) where they work, and the inclusion or not of HVS characteristics or HVS models in their design.

2.1. HVS Model Based Framework. A basic idea of any metric based on a HVS model is that subjective differences between two images cannot be extracted directly from the given images (original and distorted one) but from their perceived versions, that is, from the version that our brain perceives. As it is known, the HVS produces several visual scene information reductions, carried out in different steps. The way in which this information reduction process is modeled is the key to obtain a good subjective fidelity metric.

This framework includes the metrics that are clearly based on a HVS model, that is, their design follow the stages of any of the available HVS models. We include here metrics from the error sensitivity framework (ESF) [7] and also some other RR and NR metrics that are based on HVS models.

This framework mainly include FR metrics based on HVS models that measure errors between the reference and the distorted content using a HVS model.

In general, the emulation of HVS is a bottom-up approach that follows the first retina processing stages to continue with different models of the visual cortex behavior. Also, some metrics deal with cognitive issues about the human visual processing modeling that are included as additional stages.

The main difference between the FR metrics of this framework is related to the way they perform the subband decomposition inspired by the complex HVS models [13–15], low cost decompositions in DCT [16, 17] or wavelet [18] domains, and with other HVS related issues like in [19] where foveal vision is also taken into account and in [20] where focus of attention is also considered. It is worth noting that most of proposed FR quality assessment models share the error sensitivity based philosophy which is motivated from psychophysical vision science research [11].

Figure 2 shows a block diagram with the typical processing stages of a FR metric. In the preprocessing stage, different operations are done in order to adequate some characteristic of the reference and the distorted input versions. These operations commonly include pixel alignment, image cropping, color space transformations, device calibrations, PSF filtering, light adaptation, and other operations. Not all the metrics perform all these operations; each metric processes both signals in a different way. After the preprocessing stage,

usually HVS models first decompose the input signal into spatiotemporal subbands at both the reference and distorted signals.

The contrast sensitivity function (CSF) can be implemented in the channel decomposition step by the use of linear filters that approximate the frequency responses to the CSF like in [21]. But most of the metrics choose to implement the CSF as weighting factors that are applied to the channels after the channel decomposition, providing for each channel a different perceptual sensitivity.

As mentioned before, frequency decomposition is one of the biggest differences between models and hence between metrics. Complex HVS frequency channel decomposition models are used in QAM designs, but some of these models are simplified attending to computational constraints. In this sense, other QAM use the DCT [16] or wavelet [18] transforms showing good MOS correlation results. Depending also on the metric type and the distortions it handles, metrics use different channel decomposition models.

Cortical receptive fields are represented by 2D Gabor functions, but the Gabor decomposition is hard to compute and is not suitable for some operations as invertibility, reconstruction by addition, and so forth. In [22], Watson modeled a frequency and orientation decomposition with profiles similar to the 2D Gabor functions but computationally more efficient. Other authors like Lubin [23], Daly [24], Teo and Heeger [13], and Simoncelli et al. [25] provided different models trying to approximate as close as possible the HVS channel decomposition.

There are also some models that use temporal frequency decomposition in order to account for the characteristics of the temporal mechanisms in the HVS [21, 26]. The design of temporal filter banks is typically implemented using infinite impulse response filters (IIR) with a delay of only a few frames; other authors use finite response filters that despite their higher delay are simpler to implement.

The next step is error normalization and masking. Masking occurs when a stimulus that is visible by itself cannot be detected due to the presence of another stimulus. In contrast, facilitation occurs when a nonvisible stimulus becomes visible due to the presence of another stimulus. Most of the HVS models implement error normalization and masking as a gain-control mechanism, using the contrast visibility thresholds to weight the error signal at each channel. Some metrics [14], due to complexity and performance issues, use only intrachannel masking, while others [13] include interchannel masking, as there are evidences that channels are not totally independent in the HVS. Other authors [27] include also in this stage the luminance masking, also called light adaptation. In [28, 29], some comparisons of different masking models and some considerations about how to include them into an image encoder are made. In [30], authors propose a contrast gain-control model of the HVS that incorporates also a contrast sensitivity function for multiple oriented bandpass channels.

The last processing step (Figure 2) is the error pooling, which is in charge of combining the error signals in different channels into a single distortion/quality interpretation, providing different importance to errors depending on the channels where they appear. For most QAM, a Lp norm or Minkowski norm is used to produce an image spatial error maps. From the spatial error map, a frame-level distortion score is computed. In video quality assessment, we obtain the corresponding sequence-level distortion score by averaging frame scores. For the time domain, some metrics use temporal HVS models or information to accurately reproduce human scores, while others simply do not assess time domain. Other QAM that may be included in the model based framework may be found in [13, 15–21, 26, 27, 31–36].

2.2. HVS Properties Framework. In this framework we consider the metrics that, although are not based on a specific HVS model, are still inspired in features of the HVS. We also include those metrics that are designed to detect specific impairments produced by any of the processing stages of image and video coding, like quantization, transmission errors, and so forth.

The Institute for Telecommunication Sciences (ITS) presented in [37] an objective video quality assessment system that was based on human perception. They extract several features from the original and degraded video sequences that were statistically analyzed in comparison with the corresponding human rating extracted form subjective tests. This analysis provide parameters to adjust objective measures for these features and, after being combined in a simple linear model, they provide the final predicted scores. Some of the extracted features require the presence of the original sequence, while others are extracted in a no-reference mode. The proposed metric exploits spatial and temporal information. The processing include Sobel filtering, Laplace filtering, fast Fourier transforms, first-order differencing, color distortion measures, and moment calculation.

In [38], authors proposed a RR metric for in-service quality monitoring system. Their metric is built on a set of spatiotemporal distortion metrics that can be used for monitoring in-service of any digital video system. Authors expose that a digital video quality metric, in order to be widely applicable, must accurately emulate subjective responses, must work over the full range of quality (from very low bit rate to very high), must be computationally efficient, and should work for end-to-end in-service quality monitoring. The metrics are based on extracted features from the video sequence as in [37] and in order to satisfy the last condition (to be able to work in in-service monitoring systems), these features, extracted from spatiotemporal regions are sent, compressed following the ITU-R Recommendation BT.601, through an ancillary data channel so that it can be continuously transmitted. In the paper, the authors describe these spatiotemporal distortion metrics in detail, so that they can be implemented by researchers.

Later, through The National Telecommunications and Information Administration (NTIA), the same authors, proposed the general model of the video quality measurements techniques (known as VQM metric [39, 40]) for estimating video quality and its associated calibration techniques. This metric was submitted to be independently evaluated on MPEG-2 and H.263 video systems by the video quality experts group (VQEG) in their phase II full reference

television (FR-TV) test. In [41], authors reduce the requirements of some of the features extracted in the NTIA general model in order to achieve a monitoring system that uses less than 10 kbits/s of reference information.

We also can find metrics based on watermarking techniques that analyze the quality degradation of the embedded image [42]. There are metrics that are designed for measurement-specific distortions types and those produced by specific encoders [43, 44]. Another representative metrics in this framework are the ones proposed in [43–49].

2.3. Statistics of Natural Images Framework. Some drawbacks of the model based HVS framework are reviewed in [7, 50]. Some of these drawbacks are, for example, that the HVS models work appropriately for simple spatial patterns, like pure sine waves; however, when working with natural images, where several patterns coincide in the same image area, their performance degrades significantly. Another drawback is related to the Minkowski error pooling, as it is not a good choice for image quality measurement. As authors show, different error patterns can lead to the same final Minkowski error.

Therefore, several authors argue that the approach to the problem of perceptual quality measurement must be a top-down approach, analyzing the HVS to emulate it at a higher abstraction level. The authors supporting this approach propose using the statistics of the natural images. Some of them propose the use of image statistics to define the structural information of an image. When this structural information is degraded, then the perceptual quality is also degraded. In that sense, a measurement of the structural distortion should be a good approximation to the perceived image distortion. These metrics are able to distinguish between distortions that change the image structure and distortions that do not change it, like changes in luminance and contrast.

In [7, 51], authors define a Universal Quality Index that is able to determine the structural information of the scene. This index models any distortion as a combination of three different factors: (a) the loss of correlation between the original signal and the distorted one, (b) the mean distortion that measures how close the mean of the original and distorted version are, and (c) the variance distortion that measures how similar the variances of the signals are. The dynamic range of the Quality Index is $[-1, 1]$. A value of 1 indicates that both signals are identical. They apply this index in a 8×8 window for an image, obtaining a quality map of the image. The overall index is the average of the quality map.

Authors in [50] further improve their previous quality index and in [52] propose a generalization of their work where any distortion may be decomposed into a linear combination of different distortion components. In [53], the model is extended to the complex wavelet domain in order to design a robust metric to scaling, rotation, and translation effects.

Authors in [54] proposed a video quality metric following a frame by frame basis. It takes quality measures for different blocks of each frame taking into account their spatial variability, the movement, and other effects (like blocking) by means of a specifically adapted NR metric [45].

Other authors use also statistics of the natural scene in a different way. They state that the statistical patterns of natural scenes have modulated the biological system, adapting the different HVS processing layers to these statistics. First a general model of the natural images statistics is proposed. The modeled statistics are those captured with high quality devices working in the visual spectrum (natural scenes). So, text images, computer generated graphics, animations, draws, random noise or image, and videos captured with nonvisual stimuli devices like radar, sonar, X-ray, and so forth are out of the scope of this approach. Then, for a specific image, the perceptual quality is measured taking into account how far its own statistics are from the modeled ones. In [55], a statistical model of a wavelet coefficient decomposition is proposed, and in [56] the authors propose an NR metric derived from previous work.

Some metrics defined under this approach take the objective quality assessment as an information loss problem, using techniques related to information theory [57, 58].

2.4. Metrics under Study. Now, we introduce the metrics we will use in our study. The criteria to choose these metrics, and no other ones, was the availability of their code (source or executable) to reproduce their behavior as follows.

(i) The DMOSp-PSNR metric: we translate the traditional PSNR to the DMOS space applying a scale-conversion process. We call the resulting metric DMOSp-PSNR.

(ii) The Mean Structural SIMilarity index [50] (MSSIM) from the structural distortion/similarity framework: in the reference paper, this FR metric was tested against JPEG and JPEG2000 distortion types. We test its performance with the new distortion types available in the second release of Live Database, "Live2 Database" since it is considered a generalist metric.

(iii) The visual information fidelity (VIF) metric [59] from the Statistics of Natural Images Framework. A FR metric that quantifies the information available in the reference image and determine how much of this reference information can be extracted from the distorted image.

(iv) The no-reference JPEG2000 quality assessment (NRJPEG2000) [54] from the Statistics of Natural Images Framework. A NR metric that uses natural scene statistical models in the wavelet domain and uses the Kullback-Leibler distance between the marginal probability distributions of wavelet coefficients of the reference and distorted images as a measure of image distortion.

(v) Reduced-reference image quality assessment (RRIQA) [57] from the Statistics of Natural Images Framework. The only RR metric under study. It is based on a natural image statistical model in the wavelet transform domain.

(vi) The no-reference JPEG quality score (NRJPEGQS) [43] from the HVS properties framework. A NR

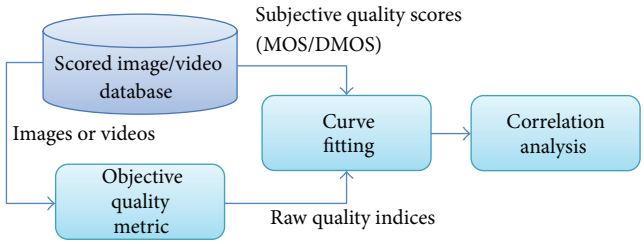


Figure 3: Block diagram of the QAM evaluation process.

metric designed specifically for JPEG compressed images.

(vii) The video quality metric [40] (VQM general model) from the HVS properties framework. The VQM uses RR parameters sent through an ancillary channel that requires at least 14% of the uncompressed sequence bandwidth. Although being conceptually an RR metric, it was submitted to the VQEG FR-TV test because the ancillary channel can be used to receive more detailed and complete references from the original frames, even the original frames themselves.

3. Comparing Heterogeneous Metrics

As previously mentioned, each QAM gets the quality of the image/video using its own and specific scale that depends on its design. Therefore, these raw quality scores cannot be compared directly, even though the range of the values (scale) is the same. In order to compare fairly the behavior of various metrics for a set of images or sequences, the objective quality index obtained from each metric has to be converted into a common scale.

When reviewing the performance comparisons that authors made in their new QAM proposals, few details are provided about the comparison procedure itself. So, it is difficult to replicate these results. Authors in [2] reviewed the sources of inaccuracy of each step of the QAM comparing process, shown at Figure 3. The sources of inaccuracy may be related to many factors as the reliability of the subjective reference data, the types and grade of the distortions in the images or videos, the selection of the content that made up the training and testing sets, and even the use and interpretation of the correlation indicators. These sources of inaccuracy can lead to quantitative differences when the same QAM is tested by different authors, even when the tests are correctly done. Although different tests can provide slightly varying results for a set of metrics, their results should be in line as explained in [2].

These issues encouraged and guided us to perform our own comparison test with the selected QAM in order to adapt the test to the target applications we are interested in. The results of our test, as expected, were slightly different from other comparison tests but remain in line with their results [2].

We use the method and mapping function proposed by the VQEG [6, 60] with some refinements proposed in other relevant comparison tests [61]. The chosen target scale is the DMOS scale (differences mean opinion score) which is the one used by the VQEG and other authors [61] when comparing metric proposals.

In order to compare several QAM, first a subjective test must be done, for example, a Double Stimulus Continuous Quality Scale (DSCQS) method as suggested and explained in [6], in order to get the subjective quality assessment of a set of images or sequences. The scale used by the viewers goes from 0 to 100. Raw scores obtained in subjective tests are converted into difference scores and processed further [58] to get a linear scale in the 0–100 range. The mean opinion score (MOS) can be calculated for the source and distorted versions of each image or sequence in this set. The DMOS is therefore the difference between the MOS value obtained for the original image/sequence and the MOS value obtained for the distorted one. So, for a particular image or sequence, its DMOS value gives the mean subjective value of the difference between the original and the distorted versions. A value of 0 means no subjective difference found between the images by all the viewers. Due to the nature of the subjective test this value is very unlikely.

In this work, we have not done such a subjective test. Instead of this, we have used directly the DMOS values published in the Live Database Release 2 [62] and in the VQEG Phase I Database [63].

Basically, the raw score of each metric must be converted into a value in this predicted DMOS (DMOSp) scale. This is done in the curve fitting step, shown in Figure 3. The final result of this scale-conversion process allows the quality score given by a metric for a specific image/sequence to be directly comparable with the one given by the other metrics for the same image/sequence.

We use the nonlinear mapping function between the objective and the subjective scores, as suggested in the VQEG Phase I and Phase II testing and validation tests [6, 60] as well as in other extensive metrics comparison tests [61]. This function is shown in (1). It is a parametric function which is able to translate a QAM raw score to the DMOSp space. As suggested in [2, 64], the performance evaluation of the metrics (correlation analysis step in Figure 3) is computed after a nonlinear curve fitting process.

A linear mapping function cannot be used because quality scores are rarely scaled uniformly in the DMOS scale, because different subjects may interpret vocabulary and intervals of the rating scale differently, depending on the language, viewing instructions, and individual psychological characteristics. Therefore, a linear mapping function would give too pessimistic view of the metric performance. Several mapping functions could be selected for this purpose, such as cubic, logistic, exponential, and power functions, with monotonicity being the main property that the function must comply with, at least in the relevant range of values.

Consider

$$\text{Quality}(x) = \beta_1 \text{logistic}(\beta_2, (x - \beta_3)) + \beta_4 x + \beta_5, \quad (1)$$

$$\text{logistic}(\tau, x) = \frac{1}{2} - \frac{1}{1 + \exp(\tau x)}. \quad (2)$$

Table 1: Equation parameters of metrics under study.

	β_1	β_2	β_3	β_4	β_5
MSSIM	−39.5158	14.9435	0.8684	−10.8913	46.4555
VIF	−3607.3040	−0.5197	−1.6034	−476.0144	−693.3585
NRJPEGQS	37.6531	−0.9171	6.6930	−0.2354	40.7253
NRJPEG2000	37.3923	0.8190	0.6011	−0.8882	74.5031
RRIQA	−18.9995	1.5041	3.0368	6.4301	5.0446
PSNR-DMOSp	23.2897	−0.4282	28.7096	−0.6657	61.5160
VQM-GM	−163.6308	6.3746	−7.6192	114.4685	76.6525

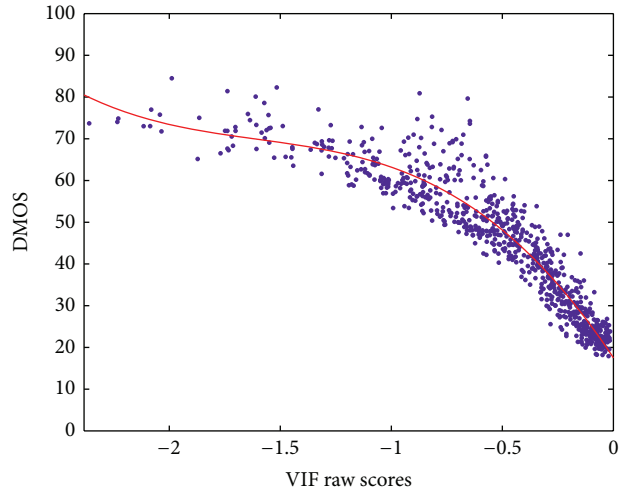


Figure 4: Dispersion plot used for the VIF metric including the curve fit for (1).

Equation (1) has five parameters, from β_1 to β_5, that are fixed by the curve fitting process that achieves the best correlation between the QA metric values and the subjective DMOS values. We have not found in the literature any mapping function with its parameters for any image/video database. So, we have calculated these parameters based on sets of images and sequences that conform with our "training sets".

As an example, Figure 4 shows the dispersion plot used in the fitting process for one of the metrics, in this case the VIF metric. Each point of the scatter-plot corresponds to an image of the training set used, Live2 Database [62]. For each image in the training set, we get the average DMOS value obtained in the subjective test and we run each metric in order to get its raw quality scores. Each metric gives its score in its own scale.

The x-axis of Figure 4 corresponds to the raw values given by the VIF implementation used, where 0 corresponds to the highest quality reported by the metric and decreasing values report lower quality. In the y-axis, we have the corresponding DMOS values. The curve fitting process gives us the parameters for (1), which is represented by the solid curve in Figure 4.

The quality of the images in the subjective test is variable, covering a large range of distortion types and intensities for each distortion. Image distortions go from very highly distorted to practically undistorted ones. The viewers gave their scores for each image in the set, obtaining the average DMOS value. As shown in Figure 4, the dynamic range of the average DMOS values does not reach the limits of the DMOS scale (0 and 100) for any distortion type; therefore, the fitted curve predicts DMOSp values inside the same dynamic range. This is the reason why for a raw score of 0 (the best possible quality for the metric in this case), the predicted DMOSp value is not 0; that is, there was no image scored with an average DMOS value of 0, instead of that, the best DMOSp value obtained is around the value of 20. So, in the case of the VIF metric its dynamic DMOSp range varies from 20 to 80.

Table 2: Goodness of DMOSp-DMOS fitting.

	PCC	RMSE	SROCC
MSSIM	0.8625	7.9682	0.851
VIF	0.9529	0.0516	0.9528
NRJPEGQS	0.936	3.0837	0.902
NRJPEG2000	0.9099	7.056	0.9021
RRIQA	0.9175	4.4986	0.9194
PSNR-DMOSp	0.85257	9.0969	0.8197
VQM-GM	0.8957	7.6746	0.9021

Having fixed the beta parameters for each metric (see Table 1), (1) can be used to estimate or predict the DMOSp value for any objective metric score.

In Table 2, the performance of our fittings is shown. These performance parameters show the degree of correlation between the DMOSp values and the subjective DMOS values provided by the viewers. Performance validation parameters are the Pearson correlation coefficient (PCC), the root mean squared error (RMSE), and the Spearman rank order correlation coefficient (SROCC).

Another key point to consider while comparing QAM [2] is the selection of the image or video sequence set used as "training set." The "training set" is used to perform the curve fitting process. This set should be chosen with special care and must be excluded from validation tests. So for each metric, the fitting process must be done using images or sequences with impairments that the metric is designed to handle. See [2] for details of how an incorrect selection of the image "training set" can influence the final interpretation of the statistics used in the correlation analysis.

Once the metric has been evaluated in the correlation analysis step, it will work with another set of images or sequences that we call the "*testing set.*" For the "testing set," the DMOS values are unknown; therefore, we obtain them via (1).

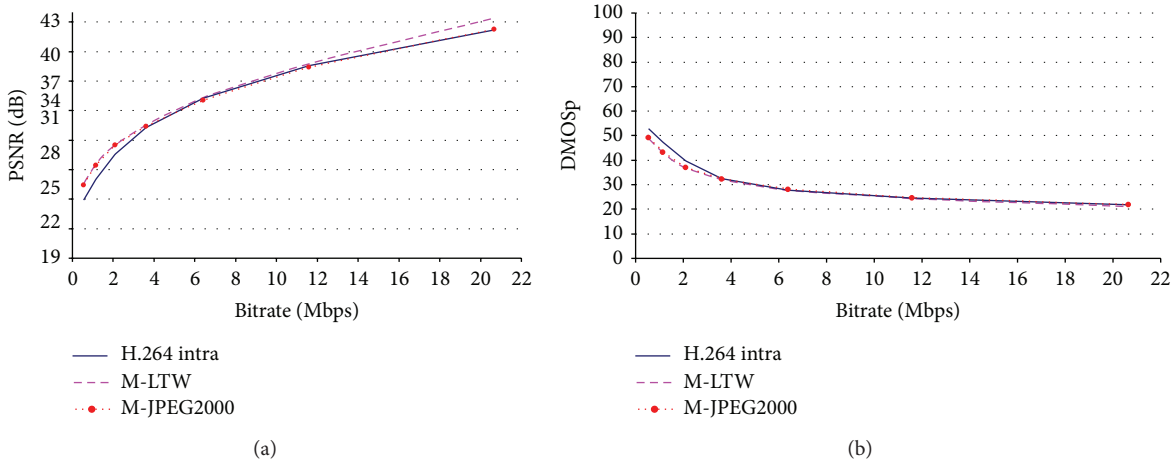


Figure 5: PSNR versus DMOSp-PSNR for the evaluated codecs (mobile sequence).

In our study, all the metrics have been "trained" only with the luminance information. The MSSIM, VIF, RRIQA, and DMOSp-PSNR metrics were "trained" with the whole Live2 Database because they are intended to be generalist metrics.

The NRJPEGQS was "trained" only with the JPEG distorted images of Live2 database as this metric is designed only to handle this type of distortions. And for the same reason the NRJPEG2000 was "trained" only with the JP2 K distorted images of Live2 Database and the VQM-GM was "trained" with a subset of 8 video sequences and its 9 corresponding HRCs of the VQEG Phase I Database in a bitrate range of 1 to 4 Mb/s.

It is important to mention that each of these "training sets" has different dynamic ranges in the DMOS scale depending on the degree of distortions applied to the images in each set.

We define as "homogeneous metrics" those which were trained with the same sets, and therefore, we use the term "heterogeneous metrics" to refer to metrics that were trained with different sets.

Our "testing set" comprises different standard video sequences that are commonly used in video coding evaluation research, as shown in Table 3. For FR-metrics, both reference and distorted images/sequences are used as input. For NR-metrics only the distorted image/sequence is available. For RR-metrics, the reference image/sequence is the input of the features extraction step, and both the extracted features and the distorted image/sequence are the input for the final metric evaluation step. Image metrics were applied to each frame of the sequences and the mean raw value for all the frames was translated to the DMOSp scale. Hence, we finally obtain comparable DMOSp values for all images/sequences.

Table 3: Sequences included in the "test set".

Sequence	Frame	F. number	F. rate
Foreman	QCIF: 176 × 144	300	30 fps.
Container			
Foreman	CIF: 352 × 288		
Container			
Mobile	640 × 512	40	

4. Analyzing Metrics Behavior in a Compression Environment

In this section, we will study the behavior of the QAM under evaluation when assessing the quality of compressed images and sequences with different encoders. As exposed before, in the development of a new encoder or when performing modifications to existing ones, the performance of the proposals must be evaluated in terms of perceived quality by means of the R/D behavior of each encoder. The distortion metric commonly used in the R/D comparisons is PSNR.

So, in this test environment, we will work with the selected metrics as candidates to replace the PSNR as the quality metric in a R/D comparison of different video codecs. In this case, we will use a set of video encoders and video sequences in order to create distorted sequences hypothetical reference circuit (HRC) at different bitrates and analyze the results of the different QAM under study. Also, we will consider the metric complexity in order to determine their scope of application. For the tests, we have used an Intel Pentium 4 CPU Dual Core 3.00 GHz with 1 Gbyte RAM. The programming environment used is Matlab 6.5 Rel.13. The codecs under test are H.264/AVC [65], Motion-JPEG2000 [66], and Motion-LTW [67]. The fitting between objective metric values and subjective DMOS scores was done using the Matlab curve fitting toolbox looking for the best fit in each case.

A R/D plot of the different video codecs under test, using the traditional PSNR as a distortion measure, is shown in Figure 5(a). It is usual to evaluate performance of video codecs in a PSNR range varying from 25–27 dB to 38–40 dB, because it is difficult to determine which one is better with PSNR values above 40 dB. This saturation effect, at high qualities, is not captured by the traditional PSNR that increases steadily as the bitrate rises, as shown in Figure 5(a).

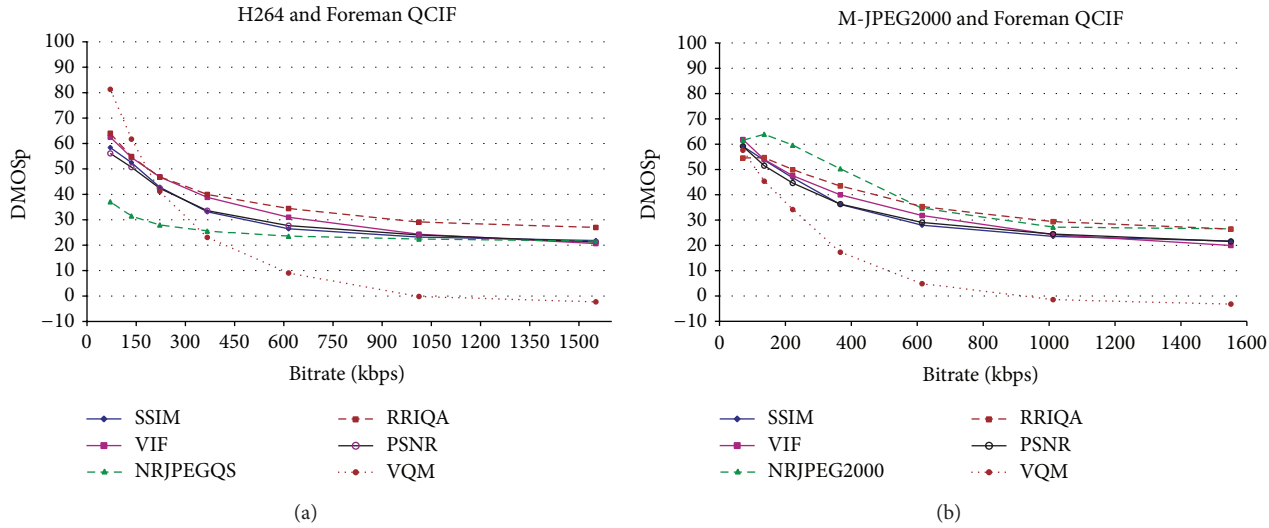


Figure 6: QAM comparison using the same sequences with different codecs.

We convert the traditional PSNR to a metric that we call DMOSp-PSNR by applying the scale-conversion process explained in Section 3. We can consider the DMOSp-PSNR metric to be the "*subjective*" counterpart of the traditional PSNR. It is the same metric, though expressed in a different scale. The DMOSp scale denotes distortion, thereby quality increases as DMOSp value decreases. The main difference between PSNR and its counterpart DMOSp-PSNR is that the saturation effect is fixed, as we can see in Figure 5(b). As it can be seen, subjective saturation effect is noticeable above a specific quality value. At bitrates above 11.5 Mbps, the DMOSp values practically do not change. This behavior is the same for all the evaluated codecs and video formats, confirming that there is no noticeable subjective difference when watching the sequences at the two highest evaluated bitrates (11.7 and 20.7 Mbps).

But as mentioned before the only modification that has been done to the PSNR metric was the mapping process with the DMOS data; that is, the raw values of the PSNR have not changed; therefore, DMOSp-PSNR metric does not fix the known drawbacks shown in Figure 1. For bitrates values below the saturation point (11.5 Mbps in the case of Figure 5(b)), the behavior of the two R/D curves is the same. In fact, the DMOSp-PSNR metric below the saturation point arranges the codecs by quality in the same order as the PSNR does, agreeing also with the results of subjective tests. This behavior is the same for all evaluated sequences and bitrates.

Since PSNR, and therefore DMOSp-PSNR, are known to be inaccurate perceptual metrics for image or video quality assessment, we now analyze the remaining metrics under study for all codecs and bitrates. These metrics have a better perceptual behavior and they offer different scores for the images in Figure 1.

The expected behavior of a QAM scoring an image or sequence at different bitrates is as follows.

(i) It should give a decreasing quality value as the bitrate decreases when bitrate values are below saturation threshold.

(ii) The quality value should be almost the same when bitrate values are above saturation threshold.

So, we run all the metrics for each HRC and analyzed the resulting data between consecutive bitrates, obtaining the quality scores in the DMOSp space. A simple subjective DSCQS test was performed with 23 viewers in order to detect if there was really perceived differences above threshold in these sequences at high bitrates (above saturation 11.5 Mbps). In the tests, the three HRCs (for each sequence and encoder) with higher bitrates were presented to the viewers: the first HRC (the first located below saturation point, 6.4 Mbps) and the last two HRCs (two rightmost points from curves in Figure 5, 11.58 and 20.65 Mbps) that are locate in the saturation region. The test concluded that no perceptual differences were detected above saturation threshold, whereas all the viewers detected some perceptual differences bellow threshold. The predicted DMOSp differences for these HRCs above threshold vary from 0.82 to 4.91 DMOSp points, so we can initially conclude that above saturation these small differences in DMOSp values are perceptually indistinguishable.

In Figure 6 we can see examples of the R/D plots used for comparing the metrics where all the evaluated QAM were applied to the same sequence. In Figure 6(a), the HRCs were encoded with the H.264/AVC codec. The NRJPEG2000 metric is omitted because it is not designed to handle DCT transform distortions. In the same way, in Figure 6(b), where HRCs were encoded with M-JPEG2000, the NRJPEGQS metric is omitted because it is not designed to handle the distortions related to the wavelet transform. We can see that the perceptual saturation is captured by all the QAM at high bitrates (high quality) regardless of the encoder. The same holds for all the sequences and encoders.

As mentioned in Section 3, monotonicity is expected in the mapping function. So, the expected behavior of the metrics should also be monotonic; that is, metrics should indicate lower quality values as the bitrates decrease. However, if we look at Figure 6(b) and focusing on the two lowest bitrates, the quality score given by both the RRIQA and

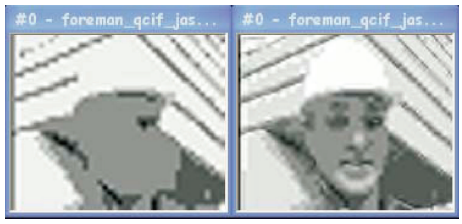


Figure 7: First frame of Foreman QCIF encoded at 70 Kbps (left) and 135 Kbps (right).

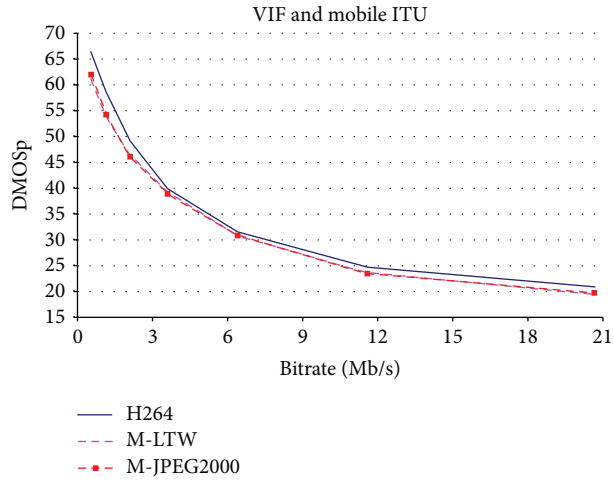


Figure 8: R/D performance evaluation of the three video codecs using mobile ITU video sequence by means of VIF metric.

NRJPEG2000 metrics increases as the bitrate value decreases. This is contrary to the expected behavior of a QAM. Figure 7 shows the first frame of the Foreman QCIF frame size sequence at these bitrates. Clearly, the right image (135 Kbps) receives a better subjective score than the left one (70 Kbps), though the mentioned metrics state just the opposite in this particular case. Our results for the compression environment show that NRJPEG2000 offers wrong quality scores between the two highest compression ratios with the M-JPEG2000 codec, for all the sequences and frame sizes tested. RRIQA also failed with this codec at high compression ratios, but only for small video formats. All the other metrics exhibit a monotonic behavior for all bitrates regardless of the encoder and sequence being tested.

Figure 6 will also help us to explain what it was exposed in Section 3; heterogeneous metrics should not be compared directly because the dynamic range of the subjective quality scores in each training set is different. Looking at Figure 6(a) and focusing on the lowest bitrate, the DMOSp rating differences between metrics arrive surprisingly up to 44.21 DMOSp units.

In fact, there are three different behaviors corresponding to the use of three different training sets: VQM-GM was trained with VQEG sequences, NRJPEGQS was trained only with the JPEG distorted images, and the rest of the metrics trained with the whole set of distorted images in the Live2 Database. This is the main reason of these anomalous behaviors in Figure 6.

So, when including in the same R/D plot curves from different metrics it should be checked that the metrics are homogeneous in order to avoid misleading conclusions.

Determining how good a metric works depends on how good the metric predicts the subjective scores given by human viewers. This goodness of fit is measured in parameters like those of Table 2. Our performance validation data tells that the VIF metric is the one which best fits the subjective DMOS values among the metrics in the same "training set."

Figure 8 represents the common R/D plots used when comparing the performance of the encoders being tested. In this case the plot shows how the VIF metric evaluates the performance of the encoders. If the mapping function of the metrics was obtained with the same "training set," then the ranking order of the encoders should agree with the subjective ranking order for each bitrate being evaluated.

We performed a simple subjective test with 23 viewers in order to evaluate if we can trust the codec ranking; that is, for a specific bitrate, the metric should arrange the encoders by quality, in the same order that a human observer does. For each rate and sequence, the reconstructed sequence of each encoder was presented simultaneously to the subjects. The ordering of the three sequences varies for each HRC, so that the subjects had no knowledge about the encoder order. The subjects ranked the sequences by perceptual quality if no differences were detected between pairs of sequences; they also annotated this fact. After analyzing the users scores and removing outliers, the test confirms that the ranking order of the metrics was the same as the subjective ranking.

In the cases where viewers scored no perceptual difference between sequences, the metrics gave always values lower than 2.9 DMOSp units of difference between encoders. In this test, for slightly higher differences, for example, 3.11 DMOSp units at 2.1 Mb/s between H264/AVC and M-JPEG2000 in Figure 8, most of the viewers could see some perceptual differences between the sequences, since they ranked H264/AVC to have better perceptual quality than M-JPEG2000 and M-LTW.

In order to determine how much difference expressed in the DMOSp scale is perceptually detectable, deeper studies and subjective tests must be done. From our studies, we detect that the perceptual meaning of the difference depends on the point in the DMOSp scale where we are working. For example, for high quality (as stated before in previous tests), DMOSp value differences up to 4.91 DMOSp points were imperceptible; however, at lower quality levels, smaller differences (3.11) can be perceived.

Finally, Table 4 shows, for different frame sizes, the mean frame evaluation time and the evaluation time for the whole sequence needed by each metric to assess its raw quality value. Times for the two steps of RRIQA, features extraction (f.e.), and quality evaluation (eval.) have been separately measured. For a CIF sequence (calibration and colour conversion time is not included) the VQM-GM is faster than the other metrics, except NRJPEGQS and DMOSp-PSNR. DMOSp-PSNR is by far the less computationally expensive metric at all frame sizes. On the other hand, RRIQA and VIF are the slowest

Table 4: QAM average scoring times (seconds) at frame and sequence level.

	QCIF		CIF		640 × 512	
	Frame	Seq.	Frame	Seq.	Frame	Seq.
MSSIM	0.028	8.4	0.147	44.1	0.764	30.5
VIF	0.347	104.1	1.522	456.5	6.198	247.9
NRJPEGQS	0.01	3	0.049	14.6	0.201	8.1
NRJPEG2000	0.163	48.9	0.486	145.9	1.595	63.8
RRIQA (f.e.)	4.779	1433.7	6.95	2084.9	10.111	404.5
RRIQA (eval.)	0.201	60.2	0.635	190.6	2.535	101.4
DMOSp-PSNR	0.001	0.3	0.006	1.7	0.02	0.8

metrics (they run a linear multiscale, multiorientation image decomposition), although in our tests the VIF is the most accurate metric among the general purpose metrics.

5. Analyzing Metrics Behaviour in a Packet Loss Environment

Our objective in this section is to analyze the behavior of the candidate metrics in the presence of packet losses under different MANET scenarios. In order to model the packet losses in these error prone scenarios, we use a three-state hidden Markov model (HMM) and the methodology presented in [68]. HMMs are well known for their effectiveness in modeling bursty behavior, relatively easy configuration, quick execution times, and general applicability. So, we consider that they fit our purpose of accelerating the evaluation process of QAM for video delivery applications on MANET scenarios, while offering similar results to the ones obtained by means of simulation or real-life testbeds. Basically, by the use of the HMM, we define a packet loss model for MANET that accurately reproduces the packet losses occurring during a video delivery session.

The modeled MANET scenario is composed of 50 nodes moving in an 870 × 870 square meters area. Node mobility is based on the random way-point model, and speed is fixed at a constant value between 1 and 4 m/s. The routing protocol used is DSR. Every node is equipped with an IEEE 802.11g/e enabled interface, transmitting at the maximum rate of 54 Mbit/s up to a range of 250 meters. Notice that a QoS differentiated service is provided by IEEE 802.11e [69]. Concerning traffic, we have six sources of background traffic transmitting FTP/TCP traffic in the best effort MAC access category. The foreground traffic is composed by real traces of an H.264 video encoded (using the Foreman CIF video test sequence) at a target rate of 1 Mbit/s. The video source is mapped to the video MAC acess category.

We apply the HMM described above to extract packet arrival/loss patterns for the simulation traces and later replicate these patterns for testing. We describe two environments: (a) congestion related environment and (b) mobility related environment.

The congestion environment is composed of 6 scenarios with increasing level of congestion, from 1 to 6 video sources. The mobility environment is composed of 3 scenarios with only one video source, but with increasing degrees of node mobility (from 1 to 4 m/s).

For each of these scenarios, we get different packet loss patterns provided by the HMM that represents each scenario.

After an analysis of the packet losses, different patterns are defined as follows.

(i) Isolated small bursts represent less than 7 consecutive lost packets. As each frame is split in 7 packets at source, isolated bursts will affect 1 or 2 frames, but none of them will be completely lost. This error pattern is mainly due to network congestion scenarios, where some packets are discarded due to transitory high occupancy in the wireless channel or buffers at relaying nodes.

(ii) Large packet loss bursts. Large Bursts cause the loss of one or more consecutive frames. Large packet error bursts are typically a consequence of high mobility scenarios, where the route to the destination node is lost and a new route discovery process should be started. This will keep the network link in down state during several seconds, losing a large number of consecutive packets.

We have used the H.264/AVC codec adjusting the error resilience parameters to the values proposed in [70], so that the decoder is able to reconstruct sequences even when large packet loss bursts occur. H.264/AVC is configured to produce one I frame every 29 P frames, with no B frames and to split each frame in 7 slices, so we put each slice into a separate packet and encapsulate its output in RTP packets. As suggested in [70], we also force 1/3 of the macroblocks of each frame to be randomly encoded in intramode.

We have used the Foreman CIF seq. (300 frames at 30 fps) to build an extended video sequence by repeating the original one up to the desired video length. After running the encoder for each extended video sequence, we get RTP packet streams. Then, we delete from the RTP packet stream, those packets that have been marked as lost packets by the HMM model. This process simulates packet losses in the MANET scenarios, so a distorted bitstream will be delivered to the decoder. The decoder behavior depends on the packet loss burst type as follows.

(i) When an isolated small bursts appear, the decoder is able to apply error concealment mechanisms to repair

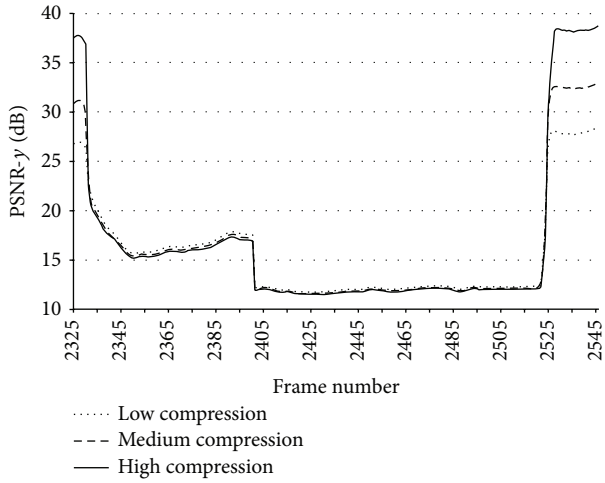


Figure 9: PSNR frame values during a long packet loss burst (from frame 2327 to 2525) at different bitrates.

the affected frames. The video quality decreases, and just after the burst, the reconstructed video quality recovers the quality by means of the random intra-coded macroblock updating. When the next I frame arrives, it completely stops error propagation.

(ii) When the decoder faces large bursts, it stops decoding and waits until new packets arrive. This produces a sequence in the decoder that is shorter than the original one. Therefore, both sequences are not directly comparable with the QAM and so we freeze the last completely decoded frame until the burst ends.

Once we have comparable video sequences (original and decoded video sequences with the same length), we are able to run the QAM. Each metric produces an objective quality value for each frame in its own scale. Then, we perform the scale-conversion to the DMOSp scale (see Section 3).

Figure 9 shows the objective quality value in the traditional PSNR scale at three different compression levels (low compression, medium compression, and high compression) during a large packet loss burst. We observe the evolution of quality during the burst period. What the observer sees during this large burst is a frozen frame, with more or less quality depending on the compression level. The PSNR metric reports that quality drops drastically with the first frame affected by the burst and decrease even more as the difference between the frozen frame and the current frame increases. Nearly at the middle of the burst, an additional drop of quality can be observed. It corresponds to a scene change (with the beginning of a new cycle of the foreman video sequence). At this point, the drastic scene change makes the differences between sequences even higher, and the PSNR metric scores with even worse values, reaching values as low as 10–12 dBs.

On the other hand, the perceived quality which changes at these levels is quite difficult to evaluate. So, a better perceptually designed QAM should not score such a quality drop in this situation because quality saturates. When the burst ends, quality rapidly increases because of the arrival of packets belonging to the same frame number than the current one in the original sequence (frame 2525 in Figure 9).

If during such a burst a QAM takes into account only the quality of the frozen frame, disregarding the differences with the original one (which changes over time), the effect of the burst would remain unnoticed for that metric, that is, quality remain constant.

Figure 10 shows the evolution of the candidate QAM during a large burst (similar to Figure 9 but in this case in the DMOSp space). There is a panel for each compression level: Figure 10(a) corresponds to high compression, Figure 10(b) to middle compression, and Figure 10(c) to low compression. We observe some interesting behaviors that we proceed to analyze.

From a perceptual point of view, quality must drop to a minimum when one or more frames are lost completely and should remain that way until the data flow is recovered. It should not matter if a scene change takes place inside the large burst. VIF and MSSIM behaves this way. At the point of the burst, where the scene change takes place, both the VIF and MSSIM metrics have almost reached their "bad quality" threshold regardless of the compression level and therefore there is no substantial change in the reported quality. The drop of quality to the minimum at the beginning of the burst evidence the lost of whole frames.

NR metrics do not detect the presence of a frozen frame (by dropping the quality score) as expected because the quality given by these metrics remain at the level scored for the frozen frame during the burst duration. So, NR metrics could not detect the beginning of a large burst, since lost frames will be replaced with the last correctly decoded frame (frozen frame) and the reference frames are not available for comparison. However, NR metrics detect the end of such bursts. Figure 11 will help us to explain this behavior, showing how reconstruction is done after a large burst. This figure shows the impairments produced when the large burst ends. Figure 11(a) is the current frame, the one being transmitted. Figure 11(b) is the frozen frame that was repeated during the burst duration. When the burst ends, the decoder progressively reconstruct the sequence using the intramacroblocks from the incoming video packets. So the decoder partially updates the frozen frame with the incoming intramacroblocks. This is shown in Figures 11(c) and 11(d) where the face of the foreman appears gradually.

The gradual reconstruction of the frame with the incoming macroblocks is interpreted in a different way by NR metrics and FR metrics. When the macroblocks begin to arrive, what happens at frame 2522 (see Figure 12), the NR metrics react scoring down quality, while the FR metrics begin to increase their quality score, just the opposite behavior. For a NR metric, without a reference frame, Figure 11(c) has clearly worse quality than Figure 11(b). But for a FR metric the corresponding macroblocks between Figures 11(c) and 11(a) help to increase the scored quality.

So, NR metrics react only when the burst of lost packets affects frames partially, that is, isolated bursts and at the end of a large burst. The NRJPEGQS metric reacts harder (i.e., it shows higher quality differences) than the NRJPEG2000

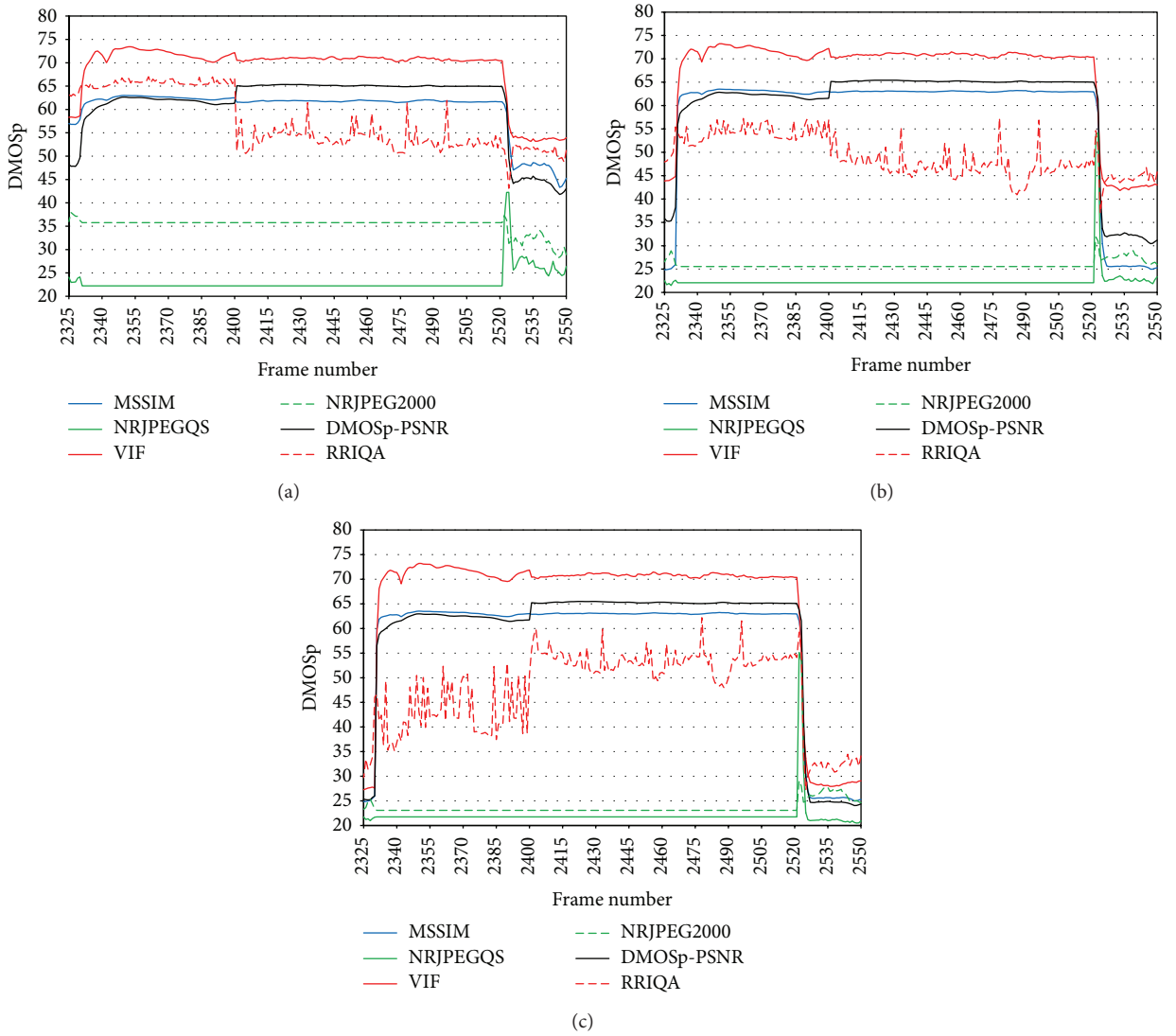


Figure 10: Metric comparison in the DMOSp space during a very large burst.

Figure 11: Frame reconstruction after a large burst: (a) original frame, (b) last frozen frame, and (c) (d) first and second reconstructed frames after the burst.

because it was designed to detect the blockiness introduced by the discrete cosine transform. When the frame is fully reconstructed then the score obtained with NR and FR metrics approaches again the values achieved before the burst, which depends on the compression rate.

The RRIQA metric shows high variability in its scores between consecutive frames inside bursts. These variations become more evident as the degree of compression decreases. The nature of the data sent through the ancillary channel, 18 scalar parameters obtained form the histogram of the wavelet

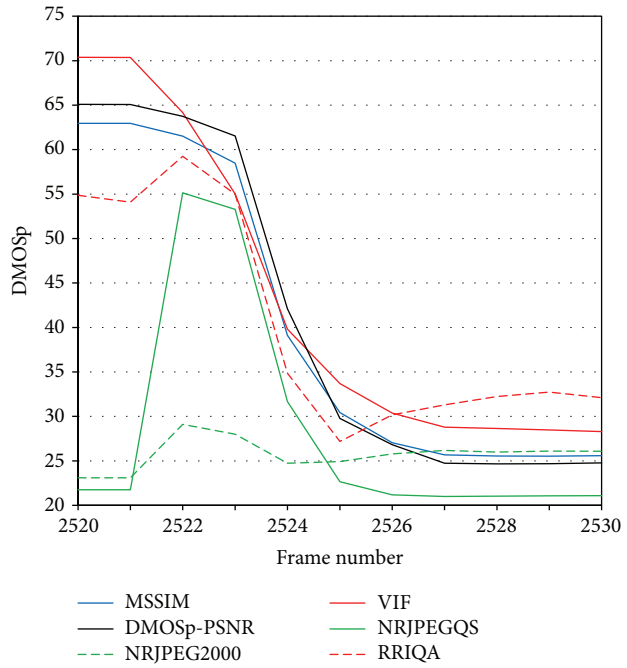


Figure 12: End of the large burst for the low compression panel. FR and NR metrics show the opposite behavior.

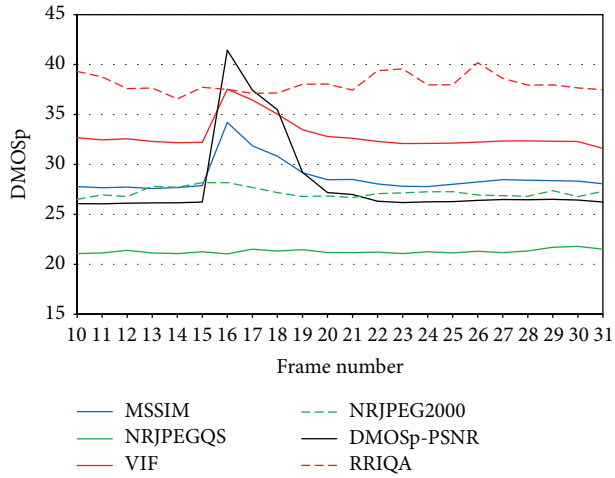


Figure 13: Metric comparison for an isolated burst.

subbands of the reference image, is very sensitive to loss of synchronism between the reference frame and the frozen one. On the decoder, the same extracted parameters are statistically compared with the received through the ancillary channel. When this comparison is performed with two sets of parameters obtained from different frames, unexpected results appear.

Concerning the FR metrics, MSSIM, VIF, and PSNR-DMOSp show a similar behavior or trend. MSSIM and PSNR-DMOSp show closer quality scores between them than the ones obtained with the VIF metric, which gives lower quality values than the other two metrics. This behavior is the same regardless of the compression level inside the large burst. Leaving aside the PSNR-DMOSp, which is not really a QAM, the other two FR metrics (VIF and MSSIM) have the same behavior when facing large bursts.

Figure 13 shows an isolated burst. In this case, blur and edge shifting impairments are introduced altering only one frame. This fact is perceived only by the FR metrics and the NRJPEG2000, which is designed to detect this type of impairments. The error concealment mechanism of H.264/AVC needs up to 6 frames to achieve the same quality scores obtained before the burst. Figure 14 shows the original frame (a) and three subsequent frames (b, c, d), where the effect of the lost packets is concealed by the H.264/AVC decoder.

As defined previously, an isolated burst can affect one or two consecutive frames. In the last case, the behavior of the QAM when facing the isolated burst resembles the behavior of the metrics with a large burst. The difference is that the concealment mechanisms and the correct reception of part of the frames avoid the largest drop in the quality.

Figure 15 shows multiple consecutive bursts (large and isolated) that behave as exposed previously. From left to right, we see a large burst followed by an isolated one. This pattern repeats again one more time, and at the right most part of the figure, between frames 352 and 372, two large bursts occur consecutively, having a gap between them where new incoming packets arrive for a short period of time (frames 361 and 362).

In Figure 16, we zoom into this area (frames 352 to 372) to analyze why the behavior of the DMOSp-PSNR metric differs from the other FR metrics during the gap between bursts. In the gap, the encoder is not able to reconstruct a whole frame because the gap is too small, that is, between the two large bursts only a small amount of packets arrive, and this is not enough to reconstruct a whole frame. So the involved frames (361 and 362) are partially reconstructed (Figures 17(b) and 17(c)). Both frames exhibit perfect correspondence in the lower half with the original one (Figure 17(a)). Therefore, the scored quality must increase, at least to some extent, compared to the quality of the previous frozen frame, as occurs at the end of a large burst. This fact is only reflected by the VIF and MSSIM metrics. The PSNR-DMOSp metric is not able to detect this because it is computed using information from the whole frame. For the VIF and the MSSIM, which are perceptually driven, the lower half of the frame increases their raw scores, in the same way as the human scores do. After frame 362, quality decreases again since the following frame is frozen too. So, VIF and MSSIM detect two consecutive loss burst, while PSNR-DMOSp and the other metrics consider only a single larger one.

6. Conclusions

The main goal of this work was focused on looking for a quality assessment metric that could be used instead of the PSNR when evaluating compressed video sequences with different encoder proposals at different bitrates and to analyze the behavior of such metrics when compressed video is transmitted over error prone networks such as MANETs.

(a) (b) (c) (d)

Figure 14: Packet loss affecting only one frame. (a) Original frame and (b, c, d) next three decoded frames.

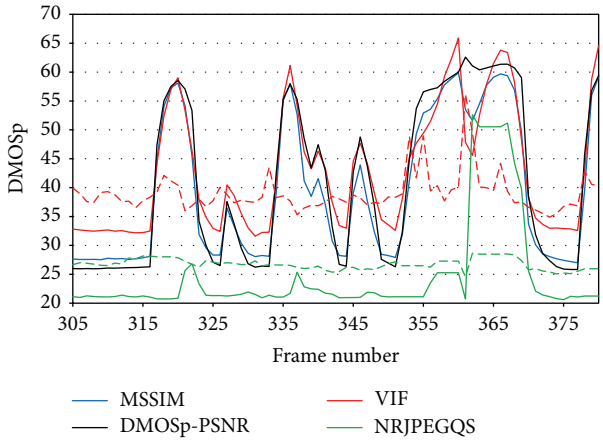


Figure 15: Frame interval where different type of bursts occurs consecutively.

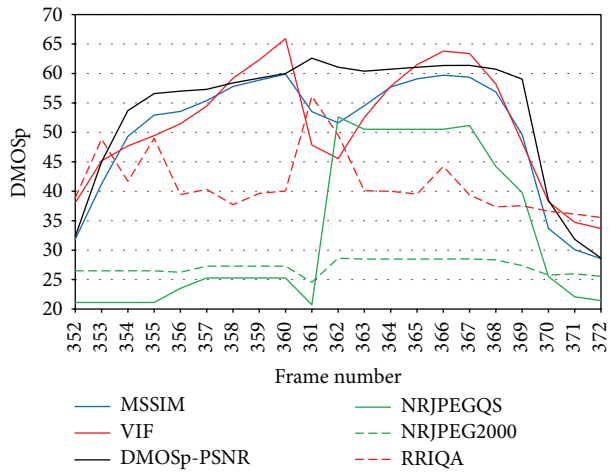


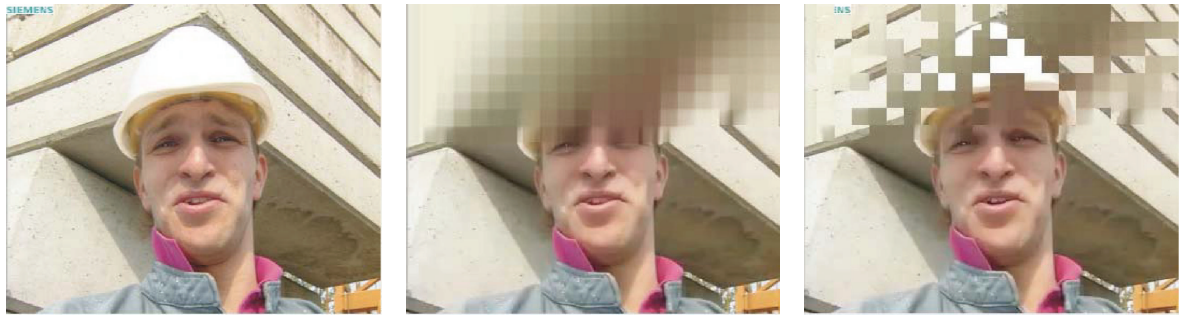
Figure 16: Detail from two consecutive long burst with incoming packets between them.

We explained the procedures that we followed to compare QAM metrics and alerted about some issues that arise when a comparison between heterogeneous metrics is made. The metrics must be compared using a common scale, since the raw scores of the metrics are not directly comparable. The scale-conversion process involves subjective tests and the use of mapping functions between the subjective MOS values and the metrics raw values. The parameters for the mapping function we used are provided in the paper. The metrics were first trained with a set of images from two open source image and video databases with known MOS values. The metrics were tested with another set of images and videos also taken from available databases. In order to perform a fair comparison, the training and testing sets used with each metric must use only impairments which the metric was designed to handle. We defined as heterogeneous metrics those that were trained with different sets of images or sequences. The R/D comparisons of heterogeneous metrics must be done carefully, focusing not only on the absolute quality scores, but also on the relative scoring between consecutive bitrates as the differences between DMOSp values are perceptually detected (or not) depending on the quality range. When metrics are trained with the same training set, differences in DMOSp values have the same perceptual meaning for all the metrics, but this may not be true between heterogeneous metrics. Normalizing the DMOSp scale when comparing heterogeneous metrics helps to detect these differences.

We performed the comparison between metrics in two environments: a compression environment and a packet loss environment. We performed several subjective tests in order to confirm that the analysis and the behavior of the metrics were consistent with human perception. Our tests included the comparisons of three encoders by replacing the PSNR as distortion metric in their R/D curves with each of the candidate metrics.

From our results of the compression environment, we conclude that we can trust the quality provided by the VIF metric, which is the one that obtains a better fit in terms of DMOS during the calibration process and on how it ranks the performance of the tested encoders for the bitrate range under consideration. The NRJPEG2000 and the RRIQA metrics break monotonicity for very high compression levels when M-JPEG2000 is the evaluated encoder. For the rest of the bitrates, all the other metrics show a monotonic behavior for all the bitrate range and for all encoders.

The choice of a QAM to replace the traditional PSNR, when working in a compression framework with no packet losses, depends on the availability of the reference sequence. In applications where the reference sequence is not available, RRIQA is our choice because its behavior is similar to FR metrics. If the reference sequence is available, the choice depends on the weight given to the tradeoff between computational cost and accuracy. If time is the most important parameter, we



(a) (b) (c)

Figure 17: Decoded frames between two consecutive bursts: (a) original frame; reconstructed frames (b) 361 and (c) 362.

will choose DMOSp-PSNR followed by VQM and MSSIM. If accuracy is more important, then the choice will be VIF and MSSIM metrics.

In the loss-prone environment, we analyzed the metrics behavior when measuring reconstructed video sequences encoded and delivered through error prone wireless networks, like MANETs. In order to obtain an accurate representation of delivery errors in MANETs, we adopted an HMM model able to represent different MANET scenarios.

The results of our analysis are as follows. (a) NR metrics are not able to properly detect and measure the sharp quality drop due to the loss of several consecutive frames. (b) The RR metric has a nondeterministic behavior in the presence of packet losses, having difficulties in identifying and measuring this effect when the video is encoded with moderate to high compression rates. (c) Concerning the other metrics, MSSIM, DMOSp-PSNR, and VIF show a similar behavior in all cases. In summary, we consider that although they exhibit slight differences in the packet loss framework, we propose the use of the MSSIM metric as a tradeoff between a high quality measurement process (resembling human visual perception) and computational cost.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was supported by the Spanish Ministry of Education and Science under Grant no. TIN2011-27543-C03-03.S.

References

[1] K. Brunnstrom, D. Hands, F. Speranza, and A. Webster, "VQEG validation and ITU standardization of objective perceptual video quality metrics," *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 96–101, 2009.

[2] J. Korhonen, N. Burini, J. You, and E. Nadernejad, "How to evaluate objective video quality metrics reliably," in *Proceedings of the 2012 4th International Workshop on Quality of Multimedia Experience (QoMEX '12)*, pp. 57–62, July 2012.

[3] M. P. Eckert and A. P. Bradley, "Perceptual quality metrics applied to still image compression," *Signal Processing*, vol. 70, no. 3, pp. 177–200, 1998.

[4] T. N. Pappas and R. J. Safranek, "Perceptual criteria for image quality evaluation," in *Handbook of Image and Video Processing*, pp. 669–684, Academic Press, 2000.

[5] VQEG, "Final report from the video quality experts group on the validation of objective models of video quality assessment," phase I, 2000.

[6] VQEG, "Final report from the video quality experts group on the validation of objective models of video quality assessment," phase II, 2003.

[7] Z. Wang, A. C. Bovik, and L. Lu, "Why is image quality assessment so difficult?" in *Proceedings of the 2002 IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP '02)*, vol. 4, pp. 3313–3316, May 2002.

[8] S. Winkler and P. Mohandas, "The evolution of video quality measurement: from PSNR to hybrid metrics," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 660–668, 2008.

[9] F. Porikli, A. Bovik, C. Plack et al., "Multimedia quality assessment [DSP Forum]," *IEEE Signal Processing Magazine*, vol. 28, no. 6, pp. 164–177, 2011.

[10] S. Winkler, "Issues in vision modeling for perceptual video quality assessment," *Signal Processing*, vol. 78, no. 2, pp. 231–252, 1999.

[11] Z. Wang, H. R. Sheikh, and A. C. Bovik, "Objective video quality assessment," in *The Handbook of Video Databases: Design and Applications*, chapter 41, pp. 1041–1078, CRC Press, 2003.

[12] B. Girod, "What's wrong with mean-squared error," in *Digital Images and Human Vision*, pp. 207–220, 1993.

[13] P. C. Teo and D. J. Heeger, "Perceptual image distortion," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '94)*, vol. 2, pp. 982–986, 1994.

[14] C. J. van den Branden Lambrecht and O. Verscheure, "Perceptual quality measure using a spatiotemporal model of the human visual system," *Storage and Retrieval for Image and Video Databases*, vol. 2668, pp. 450–461, 1996.

[15] A. B. Watson, J. Hu, and J. F. McGowan III, "Digital video quality metric based on human vision," *Journal of Electronic Imaging*, vol. 10, no. 1, pp. 20–29, 2001.

[16] J. Malo, A. M. Pons, and J. M. Artigas, "Subjective image fidelity metric based on bit allocation of the human visual system in

the DCT domain," *Image and Vision Computing*, vol. 15, no. 7, pp. 535–548, 1997.

[17] A. B. Watson, "Toward a perceptual video-quality metric," in *Human Vision and Electronic Imaging III*, Proceedings of SPIE, July 1998.

[18] M. Masry and Y. Sermadevi, "A scalable wavelet-based video distortion metric and applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 260–272, 2006.

[19] Z. Wang, A. C. Bovik, L. Lu, and J. Kouloheris, "Foveated wavelet image quality index," in *Applications for Digital Image Processing XXIV*, Proceedings SPIE, pp. 42–52, August 2001.

[20] A. Cavallaro and S. Winkler, "Segmentation-driven perceptual quality metrics," in *Proceedings of the 2004 International Conference on Image Processing (ICIP '04)*, vol. 5, pp. 3543–3546, October 2004.

[21] C. J. van den Branden Lambrecht, "A working spatio-temporal model of the human visual system for image restoration and quality assessment applications," in *Proceedings of the 996 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP' 96)*, vol. 4, pp. 2291–2294, May 1996.

[22] A. B. Watson, "The cortex transform: rapid computation of simulated neural images," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 311–327, 1987.

[23] J. Lubin, "The use of psychophysical data and models in the analysis of display system performance," in *Digital Images and Human Vision*, pp. 163–178, MIT Press, Cambridge, Mass, USA, 1993.

[24] S. Daly, "The visible differences predictor: an algorithm for the assessment of image fidelity," in *Digital Images and Human Vision*, pp. 179–206, MIT Press, Cambridge, Mass, USA, 1993.

[25] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multiscale transforms," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 587–607, 1992.

[26] S. Winkler, "Perceptual distortion metric for digital color video," in *Proceedings of the 1999 Human Vision and Electronic Imaging IV*, pp. 175–184, January 1999.

[27] A. B. Watson, "Dct quantization matrices visually optimized for individual images," 1993.

[28] M. Nadenau, *Integration of human color vision models into high quality image compression [Ph.D. thesis]*, STI, Lausanne, Switzerland, 2000.

[29] M. J. Nadenau, J. Reichel, and M. Kunt, "Performance comparison of masking models based on a new psychovisual test method with natural scenery stimuli," *Signal Processing: Image Communication*, vol. 17, no. 10, pp. 807–823, 2002.

[30] A. B. Watson and J. A. Solomon, "Model of visual contrast gain control and pattern masking," *Journal of the Optical Society of America A*, vol. 14, no. 9, pp. 2379–2391, 1997.

[31] C. Lambrecht and O. Verscheure, "Perceptual quality measure using a spatio-temporal model of the human visual system," vol. 2668 of *Proceedings of the SPIE*, pp. 450–461, San Jose, Calif, USA, January-February 1996.

[32] A. B. Watson, "Perceptual optimization of dct color quantization matrices," in *Proceedings of the 1994 IEEE International Conference on Image Processing (ICIP '94)*, vol. 1, pp. 100–104, 1994.

[33] S. Winkler, "Quality metric design: a closer look," in *Human Vision and Electronic Imaging*, vol. 3959 of *Proceedings of SPIE*, pp. 37–44, January 2000.

[34] A. B. Watson, G. Y. Yang, J. A. Solomon, and J. Villasenor, "Visibility of wavelet quantization noise," *IEEE Transactions on Image Processing*, vol. 6, no. 8, pp. 1164–1175, 1997.

[35] Z. Yu, H. R. Wu, S. Winkler, and T. Chen, "Vision-model-based impairment metric to evaluate blocking artifacts in digital video," *Proceedings of the IEEE*, vol. 90, no. 1, pp. 154–169, 2002.

[36] Y. Sermadevi and S. S. Hemami, "Linear programming optimization for video coding under multiple constraints," in *Proceedings of the Data Compression Conference (DCC '03)*, pp. 53–62, March 2003.

[37] A. A. Webster, C. T. Jones, M. H. Pinson, S. D. Voran, and S. Wolf, "An objective video quality assessment system based on human perception," in *Human Vision, Visual Processing, and Digital Display IV*, Proceedings of SPIE, pp. 15–26, September 1993.

[38] S. Wolf and M. H. Pinson, "Spatial-temporal distortion metrics for in-service quality monitoring of any digital video system," in *Proceedings of the 1999 Multimedia Systems and Applications II*, pp. 266–277, September 1999.

[39] S. Wolf and M. Pinson, "Video quality measurement techniques," NTIA Technical Report TR-02-392, 2002.

[40] M. H. Pinson and S. Wolf, "A new standardized method for objectively measuring video quality," *IEEE Transactions on Broadcasting*, vol. 50, no. 3, pp. 312–322, 2004.

[41] S. Wolf and M. H. Pinson, "Low bandwidth reduced reference video quality moni- toring system," in *Proceedings of the 1st International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, January 2005.

[42] S. Winkler, E. D. Gelasca, and T. Ebrahimi, "Perceptual quality assessment for video watermarking," in *Proceedings of the International Conference on Information Technology: Coding and Computing*, pp. 90–94, April 2002.

[43] Z. Wang, H. R. Sheikh, and A. C. Bovik, "No reference perceptual quality assessment of JPEG compressed images," in *Proceedings of the International Conference on Image Processing (ICIP '02)*, pp. 477–480, September 2002.

[44] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi, "Perceptual blur and ringing metrics: application to JPEG2000," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 163–172, 2004.

[45] Z. Wang, A. C. Bovik, and B. L. Evans, "Blind measurement of blocking artifacts in images," in *International Conference on Image Processing (ICIP 2000)*, vol. 3, pp. 981–984, Vancouver, Canada, September 2000.

[46] A. C. Bovik and S. Liu, "DCT-domain blind measurement of blocking artifacts in DCT-coded images," in *Proceedings of the 2001 IEEE Interntional Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 3, pp. 1725–1728, May 2001.

[47] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi, "A No-reference perceptual blur metric," in *Proceedings of the International Conference on Image Processing (ICIP'02)*, vol. 3, pp. 57–60, September 2002.

[48] T. M. Kusuma and H. J. Zepernick, "A reduced-reference perceptual quality metric for in-service image quality assessment," in *Proceedings of the Joint First Workshop on Mobile Future and Symposium on Trends in Communications (SympoTIC '03)*, pp. 71–74, 2003.

[49] P. Gastaldo, R. Zunino, I. Heynderickx, and E. Vicario, "Objective quality assessment of displayed images by using neural networks," *Signal Processing: Image Communication*, vol. 20, no. 7, pp. 643–661, 2005.

[50] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[51] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, 2002.

[52] Z. Wang and E. P. Simoncelli, "An adaptive linear system framework for image distortion analysis," in *Proceedings of the IEEE International Conference on Image Processing 2005 (ICIP '05)*, vol. 3, pp. 1160–1163, September 2005.

[53] Z. Wang and E. P. Simoncelli, "Translation insensitive image similarity in complex wavelet domain," in *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, vol. 2, pp. 573–576, March 2005.

[54] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment using structural distortion measurement," in *Proceedings of the International Conference on Image Processing (ICIP'02)*, vol. 3, pp. 65–68, September 2002.

[55] E. P. Simoncelli, "Modeling the joint statistics of images in the wavelet domain," in *44th Annual Meeting*, vol. 3813 of *Proceedings of SPIE*, pp. 188–195, July 1999.

[56] H. R. Sheikh, A. C. Bovik, and L. Cormack, "No-reference quality assessment using natural scene statistics: JPEG2000," *IEEE Transactions on Image Processing*, vol. 14, no. 11, pp. 1918–1927, 2005.

[57] Z. Wang and E. P. Simoncelli, "Reduced-reference image quality assessment using a wavelet-domain natural image statistic model," in *Human Vision and Electronic Imaging X*, vol. 5666 of *Proceedings of SPIE*, pp. 149–159, January 2005.

[58] H. R. Sheikh, A. C. Bovik, and G. de Veciana, "An information fidelity criterion for image quality assessment using natural scene statistics," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2117–2128, 2005.

[59] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, 2006.

[60] VQEG, "Final report from the video quality experts group on the validation of objective models of video quality assessment," phase I, 2000.

[61] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3440–3451, 2006.

[62] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik, "Live image quality assessment database release 2," http://live.ece.utexas.edu/research/quality/.

[63] Video Quality Experts Group (VQEG), "Vqeg fr-tv phase i database," http://www.its.bldrdoc.gov/vqeg/downloads.aspx.

[64] A. M. Rohaly, P. J. Corriveau, J. M. Libert et al., "Video quality experts group: current results and future directions," in *Visual Communications and Image Processing*, K. N. Ngan, T. Sikora, and M. T. Sun, Eds., vol. 4067 of *Proceedings SPIE*, pp. 742–753, May 2000.

[65] "Coding of audiovisual objects part 10: advanced videocoding," ISO/IEC 14496-10:2003, ITUT Recommendation H264 Advanced video codingfor generic audiovisual services, 2003.

[66] "JPEG 2000 image coding system, part 1: core coding system," ISO/IEC 15444-1, 2000.

[67] J. Oliver and M. P. Malumbres, "Low-complexity multiresolution image compression using wavelet lower trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 11, pp. 1437–1444, 2006.

[68] C. T. Calafate, P. Manzoni, and M. P. Malumbres, "Speeding up the evaluation of multimedia streaming applications in MANETs using HMMs," in *Proceedings of the 7th ACM Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (ACM MSWiM '04)*, pp. 315–322, October 2004.

[69] "Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements," IEEE 802.11 WG. 802.11e, 2005.

[70] C. T. Calafate, M. P. Malumbres, and P. Manzoni, "Performance of H.264 compressed video streams over 802.11b based MANETs," in *Proceedingsof the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW '04)*, vol. 7, pp. 776–781, March 2004.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 813521, 8 pages
http://dx.doi.org/10.1155/2014/813521

Research Article

Log-Less Metadata Management on Metadata Server for Parallel File Systems

Jianwei Liao,[1] Guoqiang Xiao,[1] and Xiaoning Peng[2]

[1] *College of Computer and Information Science, Southwest University of China, Beibei, Chongqing 400715, China*
[2] *HuaiHua College of Computer Science and Technology, Huaihua, Hunan 418008, China*

Correspondence should be addressed to Jianwei Liao; liaotoad@gmail.com

Received 27 December 2013; Accepted 30 March 2014; Published 27 April 2014

Academic Editors: Y. Blanco Fernandez, Y.-L. Chen, and P. Muller

This paper presents a novel metadata management mechanism on the metadata server (MDS) for parallel and distributed file systems. In this technique, the client file system backs up the sent metadata requests, which have been handled by the metadata server, so that the MDS does not need to log metadata changes to nonvolatile storage for achieving highly available metadata service, as well as better performance improvement in metadata processing. As the client file system backs up certain sent metadata requests in its memory, the overhead for handling these backup requests is much smaller than that brought by the metadata server, while it adopts logging or journaling to yield highly available metadata service. The experimental results show that this newly proposed mechanism can significantly improve the speed of metadata processing and render a better I/O data throughput, in contrast to conventional metadata management schemes, that is, logging or journaling on MDS. Besides, a complete metadata recovery can be achieved by replaying the backup logs cached by all involved clients, when the metadata server has crashed or gone into nonoperational state exceptionally.

1. Introduction

Distributed and parallel file systems employ multiple parallel I/O devices by striping file data across the I/O nodes and then through using high aggregate bandwidth to meet the growing I/O requirements of parallel scientific applications [1, 2]. In addition, decoupling file's metadata from read and write operations has been proven to be an effective strategy to improve the concurrency in the parallel file systems, since the operations on metadata and real file data could be processed in parallel [3]. Generally speaking, in a parallel file system, a client file system (client) communicates with the active metadata server (MDS), which manages all properties of the whole file system, to get the permission to operate on the file and file's layout information that indicates the locations of storage servers (OSTs), on which the stripes belonging to the target file are stored. Then the client accesses the corresponding OSTs, which handles management of actual file data on the storage devices, to perform the real file I/O operations after parsing the file's layout information.

The metadata server (MDS) plays an intermediary role in a parallel file system, and the metadata is essential to the whole file system. Both interruption of the metadata service and inconsistent metadata may lead the entire file system to become unavailable [4]. Most of the parallel file systems, such as Gfarm [5], Ceph [6], and GFS [7], employ logging or journaling metadata updates on the MDS; therefore a complete and up-to-date metadata snapshot can be yielded by resorting to the logs or journals, which have been committed to the nonvolatile storage devices, when the former active MDS has crashed. Without doubts, however, logging all metadata updates slows down the speed of metadata processing on MDS, which is because the metadata response cannot be sent until the corresponding metadata update has been committed to the nonvolatile storage devices.

In order to eliminate the negative effect of logging metadata changes conducted by the MDS, based on our previous work [8, 9] this paper proposes a log-less metadata management mechanism on the metadata server. To put it from another angle, the metadata server does not create any logs to record metadata changes. In the newly proposed

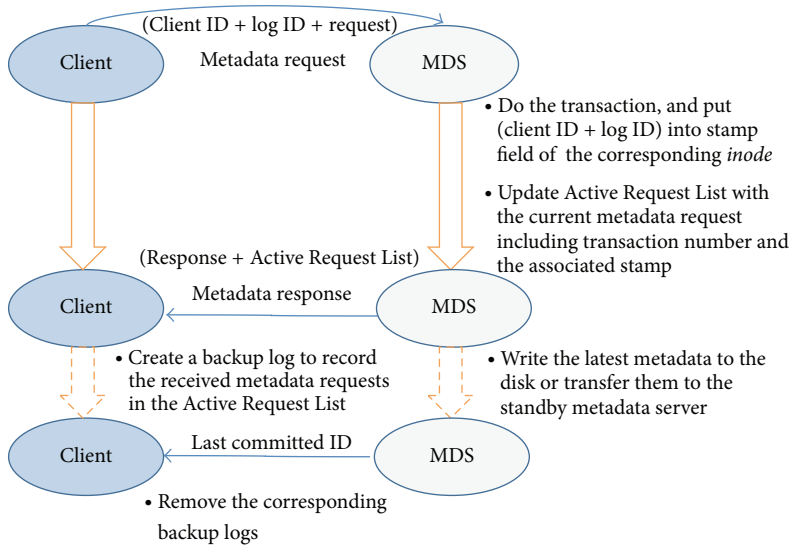


Figure 1: The interaction between client and MDS.

mechanism, all clients back up metadata requests that incorporated with the required metadata response sent by the MDS till the associated metadata changes have been committed to the nonvolatile storage. Once the former MDS has crashed, all involved clients are supposed to resend the backup metadata requests, so that the current MDS can restore all lost metadata changes by replaying these requests. Since every metadata request is backed up by several clients in this newly proposed mechanism, the complete and up-to-date metadata recovery is still achievable even though one or more clients failed to resend the backup requests. As a result, the MDS is freed from logging metadata updates to nonvolatile storage; thus, the speed of metadata processing can be improved to a great extent, which is the main goal of this newly proposed metadata management mechanism.

This paper is organized as follows: we first present the design and implementation of the log-less metadata management mechanism in Section 2; next, the evaluation experiments and results are demonstrated in Section 3; then, Section 4 describes the related work about metadata management for ensuring metadata consistency; finally, we make concluding remarks.

2. Design and Implementation

2.1. The Interactivity between Client and MDS. Figure 1 illustrates the normal case of interactivity between the client and the MDS in the log-less metadata management mechanism. The client sends the request with a log ID and client ID; after the transaction involving the metadata request, the MDS reads the stamp field of the associated *inode* (a data structure in the file systems that stores all the information about the file system objects, such as file and directory, but without data and name) and sends it along with the metadata response to the client; finally, it puts the client ID and log ID into the stamp field of the corresponding *inode* to indicate the last metadata request related to the *inode* and updates Active Request List with the current metadata request including the sequential number generated by the MDS (i.e., the corresponding transaction number) and the relevant stamp. The client, after receiving the response replied by MDS, creates a backup log in the memory to record all metadata requests (the requests in the Active Request List) sent by the MDS with the response.

The backup logs can be removed from memory after the associated metadata changes have been written to the disk. Once the active MDS has crashed, the rebooted MDS or the standby MDS can restore the lost metadata by reexecuting the backup requests stored on the client side.

2.2. Client Caching Metadata Requests. As we mentioned in Section 2.1, the client file system backs up certain metadata requests. In fact, responding to every metadata request sent by the client, the MDS sends the desired metadata response with the metadata requests in the Active Request List, which holds certain most recent metadata requests. In fact, every metadata request in the Active Request List includes the original request sent by the client, unique sequential number generated by the MDS, and the relevant stamp. On the other hand, after receiving the reply from the MDS, the client creates an in-memory log to cache the metadata requests sent by the MDS for the possible metadata recovery when the active MDS has crashed in the future. Besides, for the purpose of improving the reliability, each metadata request can be cached by more than one client; thus, the backup metadata request is still available though one of the host clients has failed to resend the backup requests.

We assume that there are 3 clients and they send metadata requests sequentially; besides, the size of Active Request List is configured as 3, which means 3 metadata requests handled by the MDS most recently will be cached on the MDS side. To put it from another angle, there should be 3 metadata requests in each backup log on the client side and each metadata request will be backed up by 3 clients. Without doubts, when the MDS has totally received less than 3 metadata requests, the number of metadata requests in

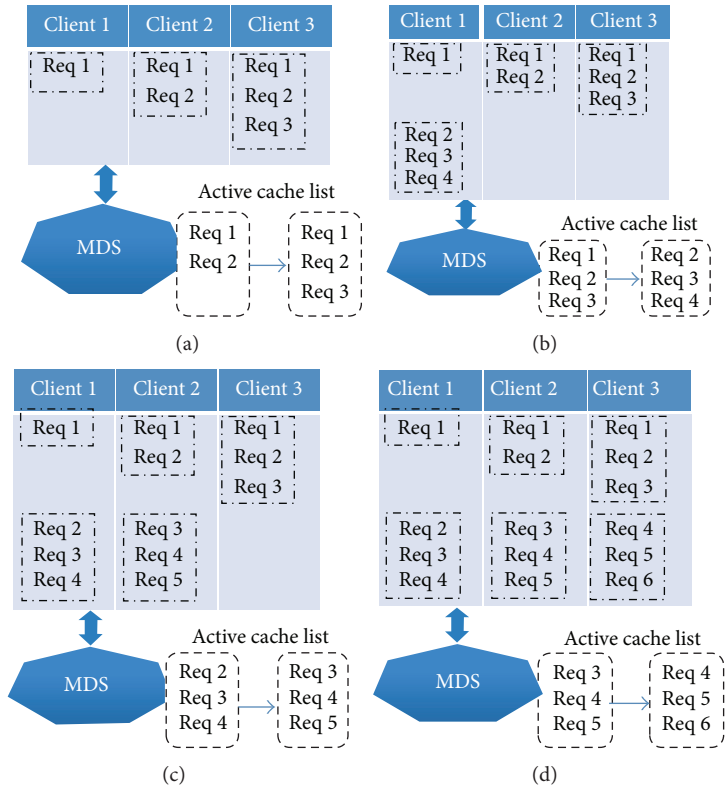


Figure 2: Backing up multiple requests on client side.

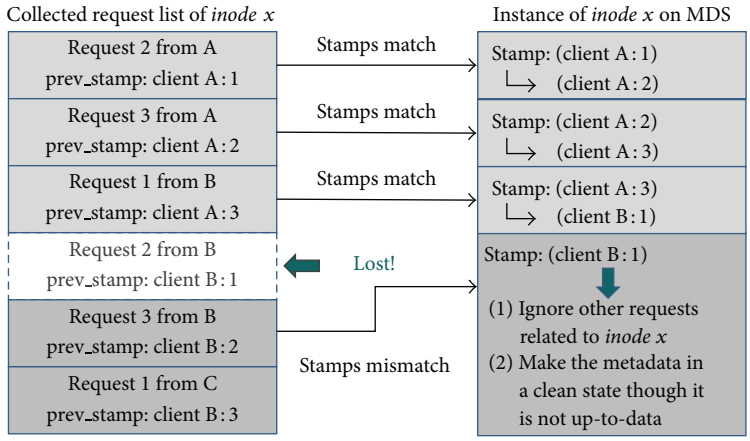


Figure 3: Stamp-based metadata recovery.

the backup log should be less than 3, as well. Namely, while the number of dead client is less than the size of Active Request List, the MDS can still collect a complete metadata request list resent by other active clients. Figure 2 shows the instance of caching multiple metadata requests on the client side in detail (the stamp information is ignored in this subsection). In Figure 2(a), all 3 clients have their own backup logs after sending their metadata requests; after *client 1* sends a new metadata request, the MDS first updates the Active Request List and then responds to *client 1* with the corresponding metadata response and the metadata requests in the Active Request List; next *client 1* creates a backup log to record the received metadata requests, which is illustrated in Figure 2(b); finally, *client 1* works as normal, such as parsing the metadata response and communicating with associated storage servers. Figures 2(c) and 2(d), respectively, demonstrate the situations when *client 2* and *client 3* have sent their metadata requests sequentially.

2.3. Stamp-Based Metadata Recovery. A stamp-based metadata recovery is employed by the log-less metadata management mechanism; Figure 3 describes the main idea of stamp-based recovery. In fact, each entry in the collected request list has a metadata request and the previous stamp of the

associated *inode*. On the other hand, the MDS first checks whether the stamp of *inode* matches the previous stamp of the collected request or not. If stamps match, the MDS plays the backup request and then updates the stamp of *inode* with the client ID (e.g., *client A*) and log ID (e.g., *request 3*) of the last reexecuted metadata request; in case the stamps mismatch, all collected metadata requests associated with the same *inode* will be thrown away from the collected request list for keeping metadata in a clean state though it might not be up-to-date.

2.4. Implementation. We have implemented a prototype parallel file system from scratch in C and run it in the Linux environment. The implementation has three modules running at the user level:

(i) the module of active metadata server, which works for providing metadata service;

(ii) the module of storage server which is responsible for the management of real file data;

(iii) the module of client file system which has been designed and implemented based on FUSE [10].

Since our implementation is a prototype system used for illustrating whether the ideas presented in this paper are feasible or not, for fairness in the comparison experiments, we have also implemented other parallel file systems with different properties (such as the parallel file system that employs log-based metadata management mechanism) based on the source code we have developed as our comparison counterparts.

3. Experiments and Evaluation

This section describes the experimental methodology for evaluating our implemented file system and reports the experimental results. First, we introduce the experimental platform for conducting all experiments. Then we show the experimental results related to the overhead associated with backing up metadata requests on the client side. Next, the benefit of log-less metadata management mechanism to metadata processing will be demonstrated and highlighted. Finally, the I/O throughput will be measured and presented.

3.1. Experimental Platform and Benchmark. We employed two cluster, which labeled as cluster 1 and cluster 2, to conduct our evaluation experiments. Consequently, one active MDS, 4 storage servers are deployed on the 5 nodes of cluster 1; the client file systems are installed on the 32 nodes of cluster 2; these two clusters are connected by 10 GbE Ethernet. Tables 1 and 2 show the specifications of the nodes on the two clusters. Moreover, the following benchmarks are used in the evaluation experiments.

(i) mdtest HPC Benchmark 1.8.3 is an MPI-coordinated metadata benchmark. It is the most frequently used benchmark to test the performance of the metadata server under intensive create/stat/remove operations on empty files and directories in parallel file systems [11].

Table 1: Specification of nodes on cluster 1.

CPU	2 × Intel Xeon E5502 1.86 GHz
Memory	6 × 4 GB 1066 MHz/DDR3 Memory
Disk storage for MDS	3 ∗ 160 GB 7200 rpm SATA HDD
Disk storage for OST	5 ∗ 160 GB 7200 rpm SATA HDD
Network	Intel 82598EB, 10 GbE Ethernet
Operating system	Debian GNU/Linux 5 (Kernel 2.6.27)

Table 2: Specification of nodes on cluster 2.

CPU	AMD Quad-Core Opteron 8356 2.3 GHz
Memory	32 GB 1066 MHz/DDR3 Memory
Local disk storage	250 GB 7200 rpm SATA HDD
Network	IP over Myrinet
Operating system	Centos 5.1 (Kernel 2.6.18)

(ii) NAS-BTIO 3.3 is an extension of NAS BT benchmark. It is derived from computational fluid dynamics (CFD) applications and widely used to test the I/O data output capabilities of parallel systems [12]. NAS-BTIO is designed to solve 3D compressible Navier-Stokes equations. Since the access pattern in NAS-BTIO is noncontiguous in memory and in the file, the MPI I/O library is used for its on-disk file access [13].

(iii) MADbench2 is an I/O benchmark derived from a real world application analyzing massive cosmic microwave background radiation in the sky from noisy pixelated datasets from satellites [14]. An extremely large amount of data is written to a disk and then read back from the disk as the calculation progresses. Since MADbench2 performs large, contiguous mixed read and write patterns as matrix operations with a variety of parameters (SHARED or UNIQUE files, POSIX versus MPI I/O, etc.), it has become a popular and often used benchmark in the parallel I/O community [15].

3.2. Overhead of Backing up Requests on Client. For the purpose of metadata recovery, the log-less metadata management mechanism adopts backing up sent metadata requests on the client side; once the former active MDS has crashed, the clients will then resend the logged, uncommitted metadata requests to the rebooted MDS or the standby MDS for restoring the lost metadata. In order to investigate the overhead due to backing up requests on the client side in our mechanism (such as making backup log records), we chose a benchmark, which simply copies an empty file 10000 times per minute. Each copy operation contains several metadata requests. Equation (1) shows the components of the metadata operations in detail:

$$1\ \text{copy} = 1\ \text{getattr} + 3\ \text{lookups} + 1\ \text{mkmod} + 2\ \text{opens} + 1\ \text{read}. \quad (1)$$

For each metadata operation, the client should make a corresponding backup metadata request. Thus, the client

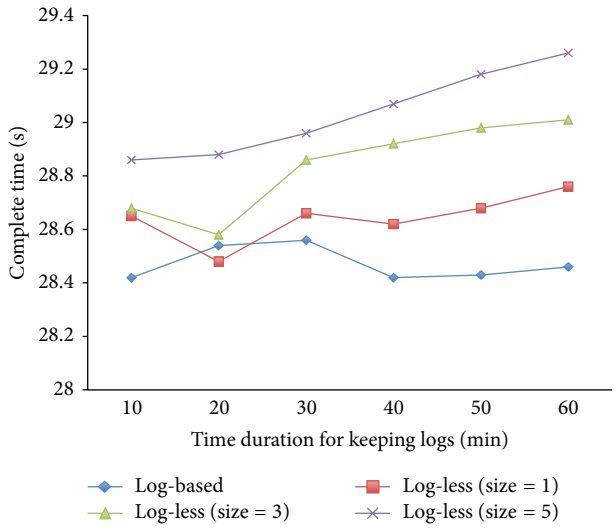


Figure 4: The overhead of backing up requests on client side.

should keep a lot of backup metadata requests after the copy operations. We should mention that there is no *write* metadata operation in (1), because the source file is an empty file. In the case of reading 0 bytes from the source file, our implemented system does not write any data to the newly created file.

Figure 4 shows the average execution times on both the log-based metadata management mechanism and log-less metadata management mechanisms (with different sizes of Active Request List, which means each log created by the client should include different number of backup metadata requests) for copying 10000 file per minute, which has been repeated 60 times in sequence (i.e., 60 minutes running time). The x-axis indicates the duration of time for which metadata requests are kept on the client side before committing the metadata changes to the disk on the MDS side. After that, the corresponding logs kept by the clients can be released. It is obvious that keeping logs on the client side brings about no more than 2.9% overhead other than for certain memory space needed for storing the backed up logs temporarily.

Moreover, it is clear that more time and more memory space are consumed while the size of Active Request List is becoming larger, that is because the more requests should be included in each client log. On the other hand, the larger size of Active Request List means much more reliability; it can tolerate more crashes of clients or loss of the backup logs. Since the size of Active Request List is configurable, it is not difficult to balance the reliability and performance overhead with the agreeable size of Active Request List.

3.3. Metadata Processing. To improve the metadata processing throughput, all metadata is kept in the memory of metadata server in the implemented file system. As a matter of fact, metadata performance is critical to the whole file system; this information has been used to find bottlenecks in the important area of metadata processing as a growing file system performance issue. The mdtest benchmark [11] was used to test the metadata performance of our implemented prototype file system, and we measured metadata performance with various clients from 1 to 16. In the experiments, we configured one task per client node, and every task executed the following command: *mdtest -u -d/mnt/pfs/newfs/temp/-b 3 -z 5 -I 100 -i 3*.

Figures 5(a) and 5(b) show the results when clients created and deleted objects (i.e., files), respectively. In these figures, x-axis represents the number of clients involved in the tests, and y-axis denotes the number of completed I/O operations per second (called IOPS, higher is better). From the results reported in Figures 5(a) and 5(b), it is safe to conclude that log-less mechanisms with different sizes of Active Request List outperform the log-based mechanism; namely, in contrast to the conventional metadata management mechanism, the newly proposed log-less mechanism can improve the speed of metadata processing, which is critical to metadata-intensive applications. In addition, the log-less mechanism with larger size of Active Request List performs a little worse than the mechanism with smaller size of Active Request List since more metadata requests should be handled.

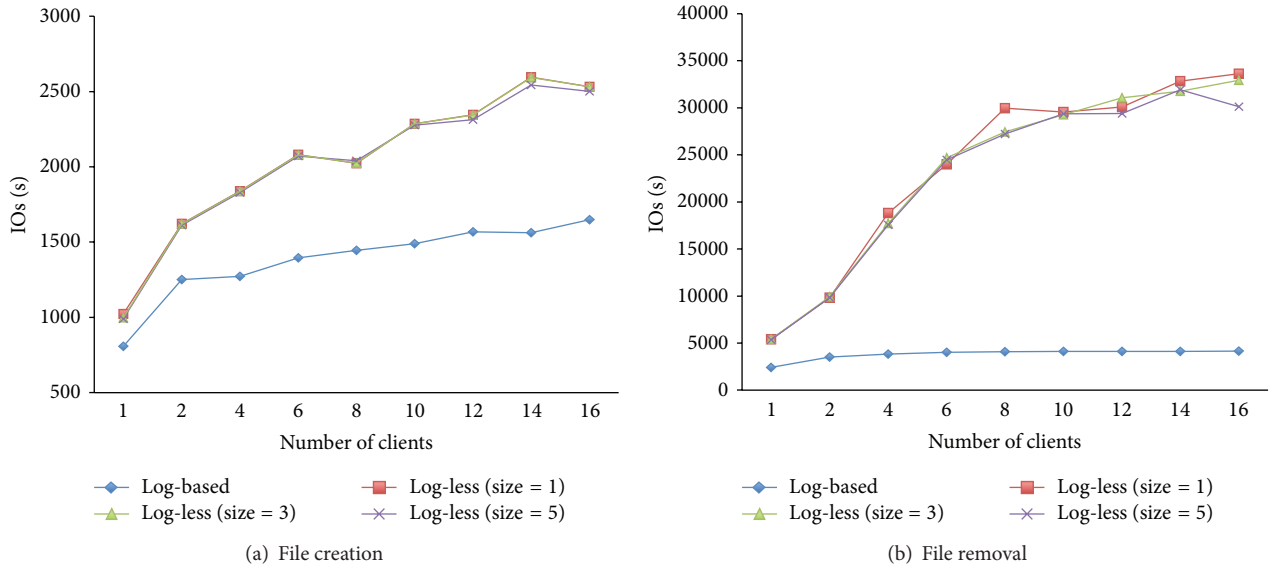
3.4. I/O Throughput. We also selected BTIO benchmark and madbench2 benchmark to measure the I/O data rate of the file systems with different properties. Figures 6(a) and 6(b) show the results of BTIO benchmark while the subtype is FULL and SIMPLE, respectively. It is clear that the log-less mechanism can obtain more I/O data rate than the log-based mechanism, for example, more than 15% improvement, while the filetype is SIMPLE and the subclass is D, which is shown in Figure 6(b). The reason for the less data throughput when adopting Log-based mechanism is due to making logs to nonvolatile storage on the MDS must cause negative effect on I/O data throughput. In addition, the log-less mechanism with smaller size of Active Request List outperforms the log-less mechanism with larger size of Active Request List because both clients and the MDS just need to process less backup metadata requests.

Figures 7(a) and 7(b) demonstrate the experimental results of MADbench2 with unique and shared filetype, respectively. As a matter of fact, the results have similar trend to that of BTIO. From the experimental results presented in the section, we can safely make a brief summary that log-less metadata management mechanism on the MDS can improve not only the speed of metadata processing, but also the I/O data throughput.

4. Related Work

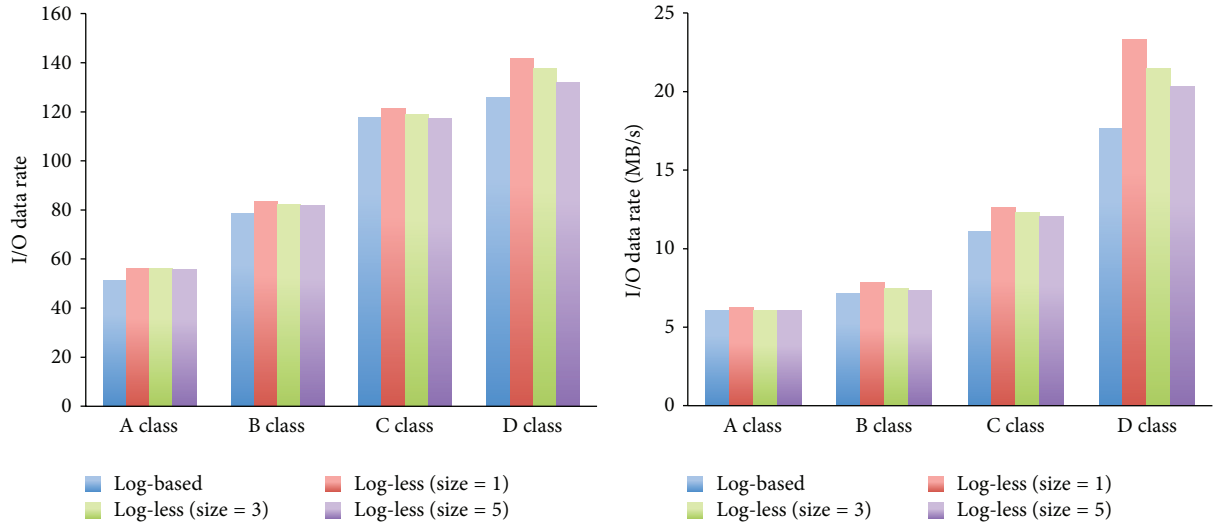
In this section, we will outline several metadata management mechanisms for restoring lost metadata updates in the conventional parallel file systems.

(i) *Logging Metadata Updates.* The traditional logging metadata updates mean every log should be flushed to nonvolatile storage before responding to the client requests. This mechanism is quite straightforward and can ensure metadata consistency correctly, but it



(a) File creation (b) File removal

Figure 5: Metadata performance: creation operation.



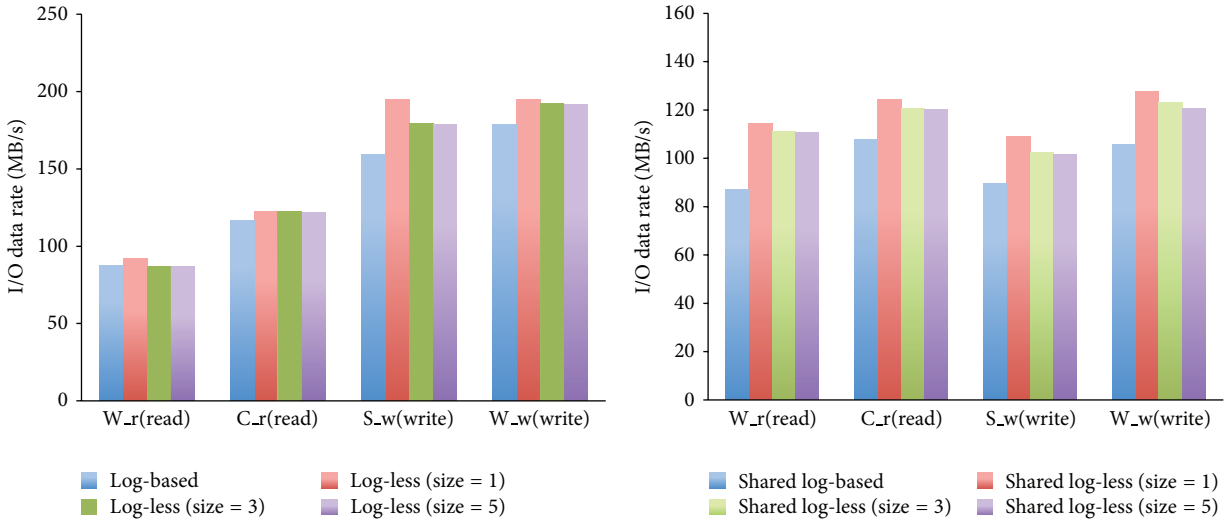
(a) Subtype: FULL (b) Subtype: SIMPLE

Figure 6: The results of BTIO benchmark.

may result in quite large overhead for I/O operations, and then impose negative effect on normal metadata processing. As a matter of fact, a major part of conventional distributed and parallel file systems, such as the Gfarm file system [5] and the Google file system [7], employs this kind of mechanism to ensure the metadata consistency even though the former MDS has crashed unexpectedly.

(ii) *Soft Updates.* The soft updates mechanism tracks dependencies among changes to cached (i.e., in-memory) copies of metadata and enforces these dependencies, via update sequencing, as the dirty metadata blocks are written back to nonvolatile storage [16]. Compared with the mechanism of logging metadata updates, the mechanism of soft update can yield performance improvements for metadata-update-intensive applications; however, since certain changes are cached in the memory, metadata consistency cannot be ensured affirmatively once the MDS goes into nonoperational state.

(iii) *Synchronous Metadata Replication.* With the mechanism of synchronous metadata replication, all metadata changes are replicated in the standby MDS before the active MDS responds to the clients. Wang et al. [17] have designed and implemented a hot standby



(a) Filetype: unique

(b) Filetype: shared

Figure 7: MADbench2 results: IOMETHOD = MPI, SYNC, 18KPIX, 16BIN.

replication mechanism in the Hadoop file system [18] to provide a highly available metadata service. This replication mechanism incurs around three times the delay in metadata responses. In case the active MDS crashes, the hot standby MDS replaces the failed one and continues to provide metadata service for the outside clients based on its current state, without any lost metadata changes. However, hot replication model results in the latency in the responses to the clients, because every metadata response cannot be delivered until the corresponding metadata change has been replicated to the standby one. Consequently, it incurs performance degradation of I/O data throughput definitely.

5. Concluding Remarks

This paper has proposed a novel metadata management scheme on the metadata server for distributed and parallel file systems. In this newly proposed log-less metadata management mechanism, all client file systems back up the sent metadata requests till the associated metadata changes have been committed to the nonvolatile storage by the active MDS, and the evaluation experiments show that backing up metadata requests on the involved clients only results in no more than 2.9% overhead. On the other hand, the log-less mechanism makes the MDS freed from logging metadata changes to nonvolatile storage systems; thus, compared with conventional log-based metadata management mechanisms, the speed of metadata processing and I/O data throughput can be improved significantly. As a matter of fact, the log-less metadata mechanism presented in this paper can be also applied to other conventional parallel file systems such as the PVFS file system [19] and the Hadoop file system, or their extensions, as well.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported partially by National Natural Science Foundation of China (no. 61303038) and Natural Science Foundation Project of CQ CSTC (no. CSTC2013JCYJA40050). The authors would like to thank anonymous reviewers for their thorough reviews and highly appreciate their comments and suggestions to revise this paper.

References

[1] J. M. Kunkel and T. Ludwig, "Performance evaluation of the PVFS2 architecture," in *Proceedings of the 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP '07)*, pp. 509–516, February 2007.

[2] Y. Li, D. Feng, and Z. Shi, "An effective cache algorithm for heterogeneous storage systems," *The Scientific World Journal*, vol. 2013, Article ID 693845, 9 pages, 2013.

[3] B. Welch, M. Unangst, Z. Abbasi et al., "Scalable performance of the Panasas parallel file system," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST '08)*, pp. 17–33, USENIX Association, 2008.

[4] Z. Chen, J. Xiong, and D. Meng, "Replication-based highly available metadata management for cluster file systems," in *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER '10)*, pp. 292–301, IEEE Computer Society, 2010.

[5] The Gfarm File System, http://datafarm.apgrid.org/.

[6] A. Weil, A. Brandt, L. Miller et al., "Ceph: a scalable, high-performance distributed file system," in *Proceedings of the 7th*

Symposium on Operating Systems Design and Implementation (OSDI '06), pp. 307–320, USENIX Association, 2006.

[7] S. Ghemawat, H. Gobioff, and S. Leung, "The google file system," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (OSDI '03)*, pp. 29–43, October 2003.

[8] J. Liao and Y. Ishikawa, "Partial replication of metadata to achieve high metadata availability in parallel file systems," in *Proceedings of the 41st International Conference on Parallel Processing (ICPP '12)*, pp. 168–177, IEEE Computer Society, Los Alamitos, Calif, USA, 2012.

[9] J. Liao, L. Li, H. Chen, and X. Liu, "Adaptive replica synchronization for distributed file systems," *IEEE Systems Journal*, 2014.

[10] Filesystem in Userspace, http://fuse.sourceforge.net/.

[11] mdtest HPC Benchmark, http://sourceforge.net/projects/mdtest/.

[12] NAS BTIO Benchmark, http://www.nas.nasa.gov/publications/npb.html.

[13] X. Zhang, K. Davis, and S. Jiang, "QoS support for end users of I/O-intensive applications using shared storage systems," in *Proceedings of the ACM/IEEE Conference on Supercomputing (SC '11)*, ACM, Seattle, Wash, USA, November 2011.

[14] MADbench2, http://crd.lbl.gov/groups-depts/computational-cosmology-center/c3-research/madbench2/.

[15] J. Borrill, L. Oliker, J. Shalf, and H. Shan, "Investigation of leading HPC I/Operformance using a scientific-application derived benchmark," in *Proceedings of the ACM/IEEE Conference on Supercomputing (SC '07)*, pp. 10:1–10:12, ACM, Reno, Nev, USA, November 2007.

[16] G. R. Ganger, M. K. McKusick, C. A. N. Soules, and Y. N. Patt, "Soft updates: a solution to the metadata update problem in file systems," *ACM Transactions on Computer Systems*, vol. 18, no. 2, pp. 127–153, 2000.

[17] F. Wang, J. Qiu, J. Yang, B. Dong, X. Li, and Y. Li, "Hadoop high availability through metadata replication," in *Proceedings of the 1st International Workshop on Cloud Data Management (CloudDB '09)*, pp. 37–44, ACM, Hong Kong, November 2009.

[18] Hadoop Distributed File System (HDFS), http://hadoop.apache.org/.

[19] Parallel Virtual File System (PVFS), http://www.pvfs.org/.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 316014, 9 pages
http://dx.doi.org/10.1155/2014/316014

Research Article

Using Fuzzy Logic in Test Case Prioritization for Regression Testing Programs with Assertions

Ali M. Alakeel

Faculty of Computing and Information Technology, University of Tabuk, P.O. Box 1458, Tabuk 71431, Saudi Arabia

Correspondence should be addressed to Ali M. Alakeel; alakeel@ut.edu.sa

Received 25 October 2013; Accepted 2 December 2013; Published 27 April 2014

Academic Editors: S. K. Bhatia and A. K. Misra

Program assertions have been recognized as a supporting tool during software development, testing, and maintenance. Therefore, software developers place assertions within their code in positions that are considered to be error prone or that have the potential to lead to a software crash or failure. Similar to any other software, programs with assertions must be maintained. Depending on the type of modification applied to the modified program, assertions also might have to undergo some modifications. New assertions may also be introduced in the new version of the program, while some assertions can be kept the same. This paper presents a novel approach for test case prioritization during regression testing of programs that have assertions using fuzzy logic. The main objective of this approach is to prioritize the test cases according to their estimated potential in violating a given program assertion. To develop the proposed approach, we utilize fuzzy logic techniques to estimate the effectiveness of a given test case in violating an assertion based on the history of the test cases in previous testing operations. We have conducted a case study in which the proposed approach is applied to various programs, and the results are promising compared to untreated and randomly ordered test cases.

1. Introduction

Program assertions have been recognized as a supporting tool during software development, testing, and maintenance [1–5]. Therefore, software developers place assertions within their code in positions that are considered to be error prone or that have the potential to lead to a software crash or failure [4]. An assertion specifies a constraint that applies to a state of computation. When an assertion is evaluated to be *false* during program execution (this scenario is called an assertion violation), there is an incorrect state in the program. Many programming languages support assertions by default, for example, Java and Perl. For languages that do not have built-in support, assertions can be added in the form of annotated statements. For example, Korel and Al-Yami [2] present assertions as commented statements that are preprocessed and converted into Pascal code before compilation. Many types of assertions can be easily generated automatically, such as boundary checks, division by zero, null pointers, and variable overflow/underflow. For this reason and to enhance their confidence in their software, programmers can be encouraged to write more assertions into their programs.

Recognizing the importance of program assertions, some recent research efforts have been devoted to the development of algorithms and methods that are specifically designed for programs that have assertions. For example, Korel et al. reported in [6] an algorithm for assertion revalidation during software maintenance. In [3], an algorithm is presented for efficient processing and analysis in which a large number of assertions are present in the program. Additionally, a regression testing method for programs with assertions was proposed in [7], and an assertion placements scheme for string-matching algorithms is presented [8].

Similar to other types of software, programs with assertions must be maintained. Software maintenance usually involves activities during which the software is modified for different reasons. Some of the reasons for which the software could be modified are fixing faults, introducing new functionality, and improving the performance of some parts of the software through the introduction of new

algorithms. A study in [9] shows that there is a probability of 50–80% of introducing faults to the modified software during software maintenance. Therefore, regression testing is performed during software maintenance for the purpose of testing the modified software to ensure its correctness after maintenance operations. There are many regression testing methods, which could be classified as specification-based or code-based. Specification-based regression testing strategies, for example, [10–12], generate test cases based on the specification of the software, while code-base regression testing, for example, [7, 13–16] strategies depend on the software structural elements to generate the test cases.

Regression testing is very labor intensive and could be responsible for approximately 50% of software maintenance costs [17]. In a systematic software development environment, all of the types of regression testing methods usually involve the usage of an original test suite, which is used for the purpose of testing the original program before it has been modified. Therefore, many regression testing methods usually utilize an existing previous test suite in some form or another during regression testing. For example, a simple regression testing strategy would rerun an existing testing suite, on an as-is basis, on the modified program, while introducing new test cases to test new features. Although this method is simple, it is not practical for commercial software because an existing test suite is usually very large and could take weeks to rerun on the new modified software. Therefore, regression test selection techniques, for example, [18], test suite minimization techniques, for example, [19], and test case prioritization techniques, for example, [20–27], are proposed in the literature to mitigate the cost that is associated with running the entire suite of previous, existing tests.

The main objective of regression test selection techniques and test suite minimization techniques is to select a representative subset of the original test suite by using information about the original program, its modified version and the original test suite. It should be noted that both the regression test selection and test suite minimization techniques eliminate some elements of the original test suite, which could undermine the performance of these techniques. Test case prioritization techniques, however, order elements of the original test suite based on a given criterion. Furthermore, test case prioritization techniques do not involve the selection of a subset of the original test suite. In this presentation, we will concentrate on test case prioritization techniques; therefore, regression test selection and test suite minimization will not be discussed any further.

With regard to programs with assertions, assertions could also undergo some modifications during maintenance. Some assertions could be modified, while new assertions could also be introduced into the new version of the program. Additionally, some assertions could be kept the same as in the original program.

This paper presents a novel approach for test case prioritization during regression testing of programs with assertions using fuzzy logic. The main objective of this approach is to prioritize test cases according to their estimated potential to violate a given program assertion. Note that it has been shown in [2] that violating an assertion implies revealing a programming fault. To develop the proposed test case prioritization approach, we utilize fuzzy logic techniques [28, 29] to measure the effectiveness of a given test case in violating an assertion based on the history of the test cases in previous testing operations. The proposed method builds on previous research in the fields of assertions-based software testing and assertions revalidation, as reported in [6, 7].

The remainder of this paper is organized as follows. Related work and background is discussed in Section 2. We present our proposed fuzzy test case prioritization model in Section 3. To evaluate our proposed approach, a case study is presented in Section 4, and conclusions and future work are discussed in Section 5.

2. Related Work

Previous research on using fuzzy logic for the purpose of test case prioritization is scant. In [30], a fuzzy expert system is reported in which the system is used during regression testing of a telecommunication application. To build the required knowledge base for the expert system reported in this research, the researchers had to acquire knowledge from different sources, such as customer profiles, past test results, system failure rates, and the history of system architecture changes. Although this expert system has shown promising results with respect to the specific application that it was designed for, it is necessary to acquire a new knowledge base for new applications. The proposed test case selection model in [30] treats the software under test as a black box; therefore, it cannot be used for the purpose of regression testing programs with assertions.

Recently, a test case prioritization concept that is based on software agents and fuzzy logic was reported in [26]. In that research, software agents are used to gather information from different sources related to the environment surrounding the software. These sources include an architectural model, test management tool, fault management tool, and change management tool. After analyzing data that is gathered from various sources, this approach assigns each software module a test importance (TI) value in the range of 1 to 10. A high TI value indicates that this module should be tested more than another module with a lower TI value. Additionally, this approach assigns each test case a local priority (LP) value based on its ability to cover a certain software module. In the end, test cases are ordered based on global priority (GP) values, which are estimated by combining the values of the module TI values and test case LP values. The concept presented in [26] is interesting; however, the amount of data that must be gathered and analyzed by software agents could be very large and costly for large industrial software. Additionally, there has not been any information provided about what type of fuzzy logic technique has been used to estimate the TI, LP, and GP values. Furthermore, the prioritization approach reported in [5] prioritizes test cases based on their coverage ability on the module level and not on the statement level. This arrangement is a drawback because program faults are usually caused by errors at the statement level and not at the module level, which makes this approach

difficult to adapt and compare with most of the existing test case prioritization methods.

2.1. Test Case Prioritization. The main goal of the prioritization techniques is to increase the probability of detecting faults at an earlier stage of testing [20–27]. Additionally, the test case prioritization technique objective is the utilization of previous test cases for the purpose of future testing. As stated in [21], there could exist several goals of test case prioritization, such as (1) to increase the test suite fault detection rate; (2) to minimize the time required to satisfy a testing coverage criterion; (3) to enhance a tester's confidence in the reliability of the software in a shorter time period; (4) to be able to detect risky faults as early as possible; and (5) to increase the chances of detecting faults that are related to software modification during regression testing.

In [21], an extensive study of nine different test case prioritization techniques was presented and compared according to their ability to perform fault detection during regression testing. During that study, a detection rate function is used to reorder test cases according to their ability to reveal program faults during regression testing. In [23], the Extended Finite State Machine (EFSM) system model is proposed to be used instead of real programs in order to apply the same technique presented in [24] and to reduce the cost of running test cases with real programs. Bryce et al. presented in [22] a test prioritization model for Event-Driven software. This model concentrates on testing those parts that are related to the interface in GUI applications. Several experimental results have been reported in [20], which study the cost-benefits of applying test case prioritization techniques. Recently, a method for test case prioritization using genetic algorithms was presented in [27]. In that research, a genetic algorithm is proposed to order the test cases according to their historical data with regard to their abilities to perform fault detection. A survey study of different test case prioritization techniques and mythologies has been reported in [25].

2.2. Regression Testing for Programs with Assertions. This section briefly introduces the concept of regression testing for programs that have assertions. For more detail, the reader is referred to [7]. Given an original program P_o and a modified version of this program P_m, let $A_o = \{a_{o1}, a_{o2}, a_{o3}, \ldots, a_{on}\}$ be a set of assertions found in P_o and $A_m = \{a_{m1}, a_{m2}, a_{m3}, \ldots, a_{mz}\}$ be a set of assertions found in P_m. Let $V \subseteq A_m$ be a set of assertions that are nominated for revalidation [6], using previous test suites, during the process of regression testing the modified version P_m. Depending on the type of modification that is applied to the modified version P_m, some assertions might have been kept the same; some assertions might have been modified, and new assertions might have been introduced. The main objective of regression testing for programs that have assertions, as reported in [7], is to reduce the cost of regression testing of programs that have assertions through the utilization of previous test suites that are used during the initial development process. Furthermore, this method concentrates on assertions that are kept the same and those that are modified; new assertions are not covered because new test cases must be generated to explore these assertions. This method is presented in more detail in the next paragraph.

Let $a_{mi} \in A_m$ be an assertion that is found in P_m. Assume that a_{mi} was not changed from its original form in P_o nor was it affected by the modifications introduced to produce P_m. Therefore, a_{mi} will be nominated by the proposed approach, to belong to the set V; that is, $a_{mi} \in V$. Suppose that assertions-oriented testing, as reported in [2], has been performed on the original version P_o, and a set of test cases were generated during this process and were kept for later usage during regression testing. In particular, let $a_{ok} \in A_o$ be an assertion that was found in P_o, and let $T(a_{ok}) = \{t_{k1}, t_{k2}, t_{k3}, \ldots, t_{kr}\}$ be the set of test cases that were generated to explore this assertion during the application of assertion-oriented testing [2] on the original program P_o. To ensure that faults are not introduced during the production of the modified version P_m, regression testing must be performed on P_m, which has a set of assertions A_m. Given $a_{ok} \in A_o$, $T(a_{ok}) = \{t_{k1}, t_{k2}, t_{k3}, \ldots, t_{kr}\}$ and $a_{mi} \in V$, it has been shown in [7] that the old test suit, $T(a_{ok})$, could be used to revalidate assertion a_{mi} during regression testing of the modified version P_m. Furthermore, it has been shown that using previous test suites to revalidate assertions could uncover faults in the modified version if these revalidated assertions were violated. More specifically, faults for which the assertions were originally designed to guard against in the original version of the program could have been reintroduced in the modified version P_m [7].

Although the regression testing method for programs with assertions, as presented in [7], could succeed in utilizing previous test suites and therefore reduce testing time, this method still considers using *all* test cases found in the previous test suite. Therefore, the method presented in [7] might not perform well in the presence of a large previous test suite with thousands of test cases. In this paper, we propose a test case prioritizing method that uses fuzzy logic concepts to select only a *subset* of the previous test cases. The proposed method is described in Section 3.

2.3. Assertions Revalidation. To address assertions in modified programs during regression testing, an assertions revalidation model was proposed in [6]. That approach is based on data dependency analysis and program slicing. In particular, that approach is based on the computation of a static slice [31, 32] for each assertion found in both the original and the modified program. These program slices are then compared to decide which assertions are to be revalidated. Although this method is very useful in identifying assertions that must be revalidated, new test cases to revalidate the assertions are generated from scratch for each assertion. For industrial size programs with a possibly large number of assertions, this approach could be very expensive.

2.4. Fuzzy Logic Background. In our daily life, we use words and terms that are vague or fuzzy, such as:

"The server is *slow*,"

"The weather is *hot*," or

"John is *tall*."

Fuzzy Logic concepts, for example, [28, 29], give us the ability to quantify and reason with words that have ambiguous meanings, such as the words (*slow*, *hot*, and *tall*) mentioned above. In fuzzy sets [28], an object can *partially* belong to a set, as opposed to classical or "crisp" sets, in which an object can belong to a set or not. For example, in a universe of heights (in feet) for adult people defined as $\mu = \{5, 5.5, 6, 6.5, 7, 7.5, 8\}$, a fuzzy subset TALL can be defined as follows:

$$\text{TALL} = [0/5, 0.125/5.5, 0.6/6, 0.875/6.5, 1/7, 1/7.5, 1/8]. \quad (1)$$

In this example, the degree of membership for the members of the universe, μ, with respect to the set TALL can be interpreted as the value "6" belongs to the set TALL 60% percent of the time, while the value 8 belongs to the set TALL all of the time.

3. A Fuzzy Test Case Prioritization Technique

The main objective of the proposed approach in this paper is to prioritize test cases according to their effectiveness when violating an assertion. More specifically, given a set of test cases, our objective is to reorder these test cases according to their estimated potential to violate a given program assertion. Note that it has been shown in [2] that violating an assertion can strongly imply uncovering program faults.

More formally, the problem investigated in this research can be stated as follows. Given an original program P_o and a modified version of this program P_m, let $A_o = \{a_{o1}, a_{o2}, a_{o3}, \ldots, a_{on}\}$ be a set of assertions found in P_o, and let $A_m = \{a_{m1}, a_{m2}, a_{m3}, \ldots, a_{mz}\}$ be a set of assertions found in P_m. Suppose that we are performing regression testing for the modified version P_m, while using some regression testing method, for example, [7]. Let $T_o = \{t_1, t_2, t_3, \ldots, t_q\}$ be a previous test suite that was used during the process of assertion-oriented test data generation [2] of the original version P_o. Given an assertion $a_{ok} \in A_o$ and a test suite $T(a_{ok}) = \{t_{k1}, t_{k2}, t_{k3}, \ldots, t_{kr}\}$, which was generated to explore assertion a_{ok} during the application of assertion-oriented testing [2] on the original program P_o. Our goal is to reorder the set $T(a_{ok})$ according to the effectiveness of a given test case $t_{kj} \in T(a_{ok})$ to violate a given program assertion $a_{mr} \in A_m$ during the regression testing process of the modified version P_m. We call the following effectiveness: Assertion Violating Potential (AVP) of a test case t_{kj}, which is represented as AVP(t_{kj}). To estimate AVP(t_{kj}), we analyze the performance of each t_{kj} in previous tests of the original program P_o together with the revalidations [6] history of assertions found in the modified version P_m.

We propose using the model that is shown in Figure 1, which can be described as follows. First, we analyze both P_o and P_m to classify the assertions, A_m, that were found in P_m with respect to how much the modifications placed in P_m had affected those assertions. To perform this analysis, we use an assertions revalidations model [6] to classify the set of assertions, A_m, which are found in P_m into three different sets: "Affected," "Partially Affected," and "Not Affected." This

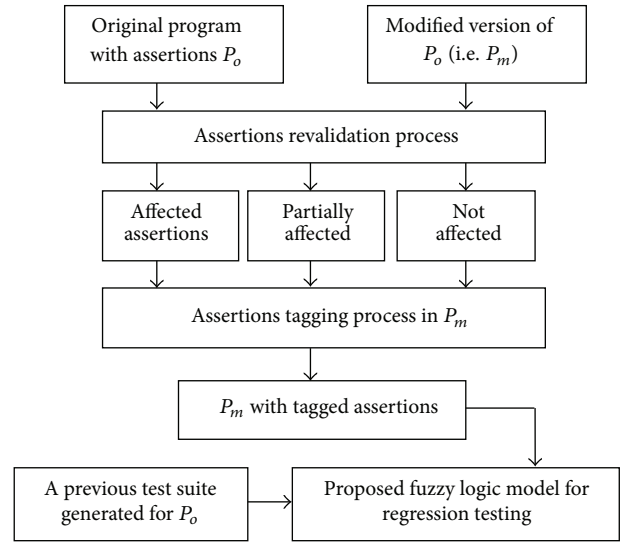


Figure 1: Fuzzy Regression Testing Model for programs with assertions.

categorization is based on how much each assertion has been affected by changes that were made in the modified program version P_m. Because it is very difficult to express this categorization with normal sets that dictate drawing crisp lines between each category, we create a fuzzy set [25] called AFFECTED in which each assertion will only belong to the set by a membership value in the range $[0, 1]$. The assignment of membership values (grades) is based on the *S*-function [33], which is shown in (4) and will be described shortly. Note that other fuzzy clustering techniques other than the *S*-function can be used for the purpose of building up fuzzy sets and the assignment of membership functions. In this research, our estimation of the modifications incurred on each assertion, A, is based on the number of variables modified in this assertion. More formally, let N_A be the number of variables that constitute an assertion A. Based on our empirical experiments, *S*-function parameters (α, β, and γ) are expressed as follows:

$$\begin{aligned} \alpha &= 0, \\ \beta &= 0.4 * N_A, \\ \gamma &= 0.8 * N_A. \end{aligned} \quad (2)$$

For example, if we have assertion A with five variables, that is, $N_A = 5$, then we will have the following *S*-function: $S_A(x; 0, 2, 4)$. Based on the number of variables, x, modified in assertion A, we substitute this number in $S_A(x; 0, 2, 4)$ to obtain the membership value for assertion A, m_A(AFFECTED), in the fuzzy set AFFECTED. Suppose that in this example only one variable was modified in assertion A; the membership of A will be computed as m_A(AFFECTED) = 0.125. On the other hand, if three variables were modified, then the membership of A will be m_A(AFFECTED) = 0.875, and so on.

Based on the results of assertions categorization performed in the first step, the next step is to categorize test cases

according to their expected effectiveness during regression testing of the modified version of the program, that is, P_m. Because the "effectiveness" of a test case is a fuzzy term that is very hard to measure with a crisp value, we propose using fuzzy logic techniques to address measuring the effectiveness of a given test case. For this purpose, we create a fuzzy set called EFFECTIVENESS. Test cases will belong to the fuzzy set EFFECTIVENESS with a membership value or grade that corresponds to their Assertion Violating Potential (AVP) values. More specifically, let $t_{kj} \in T(a_{ok})$ be a test case that was used to explore assertion $a_{ok} \in A_o$ during the initial testing of a program P_o. To measure the effectiveness of t_{kj} (with AVP(t_{kj})) in violating the corresponding assertion $a_{mr} \in A_m$ in the modified version P_m during the process of regression testing the program P_m, we use the following formula:

$$\mathrm{AVP}\left(t_{kj}\right) = 1 - \mathrm{ma}_{mr}\left(\mathrm{AFFECTED}\right), \tag{3}$$

where ma_{mr}(AFFECTED) is the membership values of assertion a_{mr} in the fuzzy set AFFECTED.

Therefore, test cases related to any assertion $a_{ok} \in A_o$, where a_{ok} belongs to the fuzzy set AFFECTED with a high membership value, will have *low* effectiveness in exploring the corresponding assertion in the modified version of the program. Similarly, test cases that are related to any assertion $a_{ok} \in A_o$, where a_{ok} belongs to the fuzzy set AFFECTED with moderate grade values, will have *moderate* effectiveness in exploring the corresponding assertion in the modified version of the program. By the same token, test cases related to any assertion $a_{ok} \in A_o$, where a_{ok} belongs to the fuzzy set AFFECTED with low membership values, will have *high* effectiveness in exploring the corresponding assertion in the modified version of the program.

3.1. The S-Function. S-functions can be described as follows [33]:

(i) A mathematical function that is used in fuzzy sets as a membership function.

(ii) A simple but valuable tool in defining fuzzy functions, such as the word "*tall*".

(iii) The objects x are elements of some universe X. In this research, x represents the set of test cases that we are addressing during our prioritization mechanism, where these test cases are elements of the universe of possible program input data.

(iv) α, β, and γ are parameters that can be adjusted to fit the desired membership data. The parameter α represents the minimum boundary, and γ represents the maximum boundary. The parameter β is the middle point between α and γ and is computed as $(\alpha + \gamma)/2$.

The S-function

$$S\left(x;\alpha,\beta,\gamma\right) = \begin{cases} 0 & \text{for } x \le \alpha \\ 2\left(\dfrac{x-\alpha}{\gamma-\alpha}\right)^2 & \text{for } \alpha \le x \le \beta \\ 1 - 2\left(\dfrac{x-\alpha}{\gamma-\alpha}\right)^2 & \text{for } \beta \le x \le \gamma \\ 1 & \text{for } x \ge \gamma. \end{cases} \tag{4}$$

Depending on the application, a membership function can be controlled from different sources [28]. For example, in an expert system, the membership function will be constructed based on the experts' opinion modeled by the system. In this research, values of the parameters α and γ are determined after experimentation with the proposed approach.

For illustration, consider the program shown in Program 1 to be the original version P_o, and its modified version, P_m, is the program represented in Program 2. The function of P_o is to compute the minimum and maximum of an array of integers. Suppose that P_o is modified to introduce a new functionality, which is to compute the sum of the array elements. This modification is shown in Program 2. Furthermore, suppose that during this modification, a fault is introduced in which statement number 12 of the modified version is "incorrectly" misplaced in an incorrect position. This seeded fault will cause the program of (4) to compute the maximum element incorrectly for certain combinations of the array's elements. Note that the seeded fault could be uncovered through the violation of assertion #2, which is shown in statement number 13 of Program 2.

Using our notation above, let the identifiers "a_{o2}" and "a_{m2}" be used to represent assertion number 2 of Program 1 and assertion number 2 of Program 2, respectively. Note that the text of these assertions is identical in both versions of the program. Suppose that during the original application of assertion-oriented test data generation [2] on the original version of Program 1, a test suite, $A(a_{o2})$, is produced during the exploration of assertion a_{o2} of Program 1. Suppose that $A(a_{o2})$ is composed of five test cases, as follows:

$$\begin{aligned} t_{21} &= (10, [17, 645, -900, 3, 88, 24, 190, -10, 1003, 115]), \\ t_{22} &= (10, [600, 200, 10000, 7, 99, 88, 42, -2000, -100, 28]), \\ t_{23} &= (10, [101, 5202, 700, 1, 32, 11, 270, -10, -575, 9]), \\ t_{24} &= (10, [-765, 33, 2009, -16, -20, 113, 800, 19, -1, -99]), \\ t_{25} &= (10, [-301, 2045, 760, 10, 609, 24, 21, -6, -14, 912]). \end{aligned} \tag{5}$$

Note that assertions-oriented testing [5] is originally proposed to be used *after* other forms of traditional software testing, such as black box (e.g., boundary value analysis) and white box (e.g., branch coverage), to increase the confidence in the software under consideration. Therefore, the test cases used in this example are only for the purpose of assertion-oriented testing [5]; hence, invalid test cases (e.g., boundary value analysis) are not included in the test suite presented above in this example.

```
(1) public int computeMinMax_O(int elements, int Data[]){
          int inData[] = new int[50]; int i, j, min = 0, max = 0;
          // Assuming a PreCondition of: "0 < elemetns <= 10"
(2), (3) for (j = 0; j < Data.length; j++) inData[j] = Data[j];
(4)       if (elements > 0 && elements <= 10){
(5)            min = inData[0];
(6)            max = inData[0];
(7)            i = 1;
(8)            while (i < Data.length){
(9)    assert (i >= 0 && i < Data.length)    // assertion #1
(10), (11)       if (min > inData[i]) min = inData[i];
(12)   assert (i >= 0 && i < Data.length)    // assertion #2
(13), (14)       if (max < inData[i]) max = inData[i];
(15)               i++;
                  }
(16)    System.err.println("\nMin is:" + min + " Max is:" + max);
(17)    System.err.println
(18)    return 1;}
(19)   else{
(20)   if (elements == 0)
(21)      System.err.println("Empty array provided!");
(22)      else System.err.println("Violation of precondition... Out of
range    array!!!    Elements:" + elements);
(23)     return −1;}
} // ComuteMinMax_O
```

PROGRAM 1: A sample Java program with assertions.

Note that assertions-oriented testing [5] is originally proposed to be used *after* other forms of traditional software testing, such as black box (e.g., boundary value analysis) and white box (e.g., branch coverage), to increase the confidence in the software under consideration. Therefore, test cases used in this example are only for the purpose of assertion-oriented testing [5]; hence, invalid test cases (e.g., boundary value analysis) are not included in the test suite presented above in this example.

Because assertion a_{m2} in the program of Program 2 is identical to assertion a_{o2} of the original version of Program 1, and because a_{m2} is not affected by the modifications [6] introduced to P_m in Program 2, the test suite generated to explore assertion a_{o2}, that is, $A(a_{o2})$, could be used to explore assertion a_{m2} during the regression testing of P_m. Note that, in this example, only two test cases, t_{23} and t_{25}, in $A(a_{o2})$ have the potential of violating assertion number 2 of Program 2, which results in uncovering the fault in P_m. It should be noted that assertion a_{m2} can only be violated by test cases that place the maximum element in the *second* position of the input array. As a result of these test cases, the program in Program 2 will compute the maximum element of the array incorrectly.

Applying our proposed test case prioritization approach to this example, assertion a_{m2} will be added to the set "Not Affected," and the two test cases (t_{23} and t_{25}) will be added to the fuzzy set "*High Effectiveness*." Therefore, the proposed approach will reorder the five test cases in the test suite $A(a_{o2})$, as follows: $(t_{23}, t_{25}, t_{22}, t_{21}, t_{24})$. This rearrangement means that t_{23} and t_{25} will be considered first, and then, the remaining test cases are considered in random order.

4. Case Study

To evaluate the effectiveness of our proposed approach for test case prioritization during regression testing of programs with assertions, we have conducted an experiment in which a set of nine programs with assertions were used. These sets of programs are borrowed and are considered to be our original versions. As reported in [2], prior to assertion-oriented testing, these programs have been thoroughly tested using traditional black box testing (e.g., boundary value analysis) and white box testing (e.g., branch coverage). Additionally, from these original programs, a total of 40 new versions are created, as will be described later. Detailed information on programs used in this case study is reported in Table 1, as follows. The first and second columns show the name of the original program and the number of lines of code of this version, respectively. The number of assertions in the original program is shown in the third column, and the fourth column shows the number of modified versions created from the original program. The fifth column represents the total number of assertions in all of the versions of the same program. In the first phase of this case study, for each program from this set, we have designated a single version that we considered to be the original. For each original version, we conduct assertion-oriented test data generation as described in [2], up to a designated search time threshold of 2 minutes. During this process, we build test suites for assertions in each original program as follows.

For each assertion, we save each test case that succeeds in reaching this assertion [2, 3]. For the purpose of this

```
(1) public int computeMinMax_M(int elements, int Data[]){
                int inData[] = new int[50]; int i, j, min = 0, max = 0; int sum;
                // Assuming a PreCondition of: "0 < elements <= 10"
(2), (3)    for (j = 0; j < Data.length;j++) inData[j] = Data[j];
(4)       if (elements > 0 && elements <= 10){
(5)             min = inData[0];
(6)             max = inData[0];
(7)             i = 1;
(8)             while (i < Data.length){
(9)   assert (i >= 0 && i < Data.length)    // assertion #1
(10), (11)      if (min > inData[i]) min = inData[i];
(12)            i++;    // this fault will cause the program to produce
                            an incorrect maximum number
(13)  assert (i >= 0 && i < Data.length)    // assertion #2
(14), (15)      if (max < inData[i]) max = inData[i];
                }
(16)            System.err.println(" \ nMin is:" + min + " Max is:" + max);
(17)            System.err.println(" \ nInput data:");
                // report the sum of the input data
(18)            sum = 0;
(19)            for (i = 0; i < Data.length; i++){
(20)  assert (i <= 0 && i < Data.length)    // assertion #3
(21)              sum + = inData[i];  }
(22)          System.err.println(" \ nSum is:" + sum);
(23)          return 1;}
(24)          else{
(25)           if (elements == 0) System.err.println("Empty array
              provided!");
(26)           else System.err.println("Violation of precondition... Out of range
(27)              array!!! Elements:" + elements);
(28)          return −1;
              }
} // ComuteMinMax_M
```

PROGRAM 2: Modified version of the program in Program 1.

TABLE 1: Programs used in the case study.

Program name	Lines of code	Number of assertions	Number of versions	Total number of assertions
Concatenation	19	4	4	20
Average	54	7	3	21
RestrictedAverage	50	6	4	30
FunnyAverage	30	5	3	15
RealNumberFormat	39	5	5	30
Bank	336	15	9	140
MinMax	23	3	6	18
Total	52	8	3	27
RestrictedSubstitute	29	3	3	10

experiment, the assertion's exploration process does not stop by violating an assertion—as in the original assertion-oriented testing [2]. Instead, the process of test data generation continues to produce more test cases up to a given number of violations, for example, two violations in this experiment or the exhaustion of a designated search time. The outcome of this process is a test suite for each assertion in each original program. At the end of this phase, we investigate the cause of each assertion's violation and correct faults in the original program to the best of our knowledge. In the second phase of this case study, from each original program, we create a set of modified versions. Modified versions are created by introducing three types of changes: keeping assertions the same while modifying the functionality of

the new version; modifying some assertions while keeping the same functionality of the original program; and modifying some assertions and modifying the functionality of the new version. Note that new assertions could be introduced at any point in these modifications.

Additionally, and most importantly, in each modified version, we have seeded a fault that should be uncovered by an assertion's violation. In the third phase of this case study, we have identified assertions for revalidation using data dependency techniques reported in [3, 6]. In the fourth and last phase of this case study, we have conducted regression testing on all of the modified versions of each original program. During the fourth phase, previous test suites, generated during the first phase of this experiment, are used to revalidate assertions in the modified version that were identified for revalidation [6] during the third phase, and the results of this step are recorded. If a given previous test suite succeeds in reaching any assertions in the modified version, it is considered to be a success of this test suite. By reaching an assertion, we mean directing the program control flow to execute the assertion [2, 3]. If the assertion is violated, then the test suite has succeeded in uncovering the fault that was seeded in the modified version.

In this experiment, we compare the performance of the proposed "Fuzzy" test case prioritization approach to that of "Untreated" and "Random" prioritization techniques. The "Untreated" is not genuinely a technique; instead, it is used as a control. For the purpose of the untreated approach, we use the original test suites that were used for assertion-based testing of the nine original programs [7]. For all of the 40 versions of modified programs, we applied the random and fuzzy techniques to each of the 150 test suites that we created for this experiment. For the untreated, we kept the original 150 test suites because there is not any prioritization.

For each test with a prioritized test suite, using our proposed fuzzy model, we estimated its effectiveness in violating assertions found in modified versions. More specifically, for each test suite, we measured the weighted average of the percentage of Violated assertions (WAPVA) relative to the set of assertions provided with those modified programs during the execution of a given test suite. The result of this case study is depicted in Figure 2. As shown in the box plot, a noticeable improvement in the rate of assertion violations is achieved by the proposed fuzzy prioritization approach compared to the random and untreated techniques. The random approach also produced some improvements compared to the untreated approach. Note that the maximum average percentage rate of assertion violations is approximately 25%. This finding is expected because assertion-based testing is applied only *after* traditional white-box and black-box testing techniques have been applied to the programs [2]. Therefore, the number of assertions that are expected to be violated would not be large because most program faults should have been detected by applying white-box and black-box testing techniques.

Although the result of this evaluation study is encouraging, it cannot be generalized for all types of programs with assertions. Good performance could be biased by the type of faults seeded in the programs and by the relationships among assertions found in these programs. This result, however, indicates that the proposed approach for test case prioritization enhances the effectiveness of previous test suites in violating assertions during regression testing of modified programs.

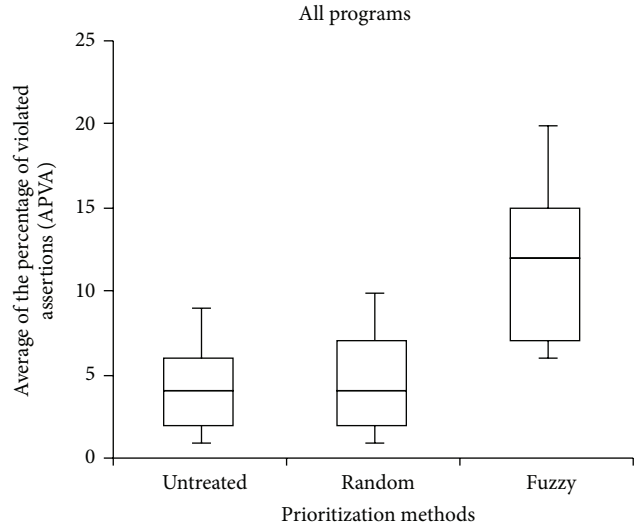


Figure 2: Average percentage of assertion violations achieved by each prioritization method.

5. Conclusions and Future Work

In this paper, we presented a test case prioritization approach for programs that have assertions. The proposed technique employs fuzzy logic concepts to measure the effectiveness of a given test case in violating program assertions during the regression testing of modified programs. Our proposed method builds upon the concepts of previous research in the fields of assertions-based software testing and assertions revalidation. We have conducted an experimental study to evaluate the proposed approach, and the results are encouraging. Nevertheless, further investigation is still required to evaluate this approach for commercial-size software.

Conflict of Interests

As the author of the paper, the author hereby declares that the support of this research by the University of Tabuk absolutely has no effect on the scientific process or the validity of this research and also has no conflict of interests with any other party.

Acknowledgments

The author would like to acknowledge financial support for this work form the Deanship of Scientific Research (DSR), University of Tabuk, Saudi Arabia, under Grant no. S-1434-0002. An earlier version of this paper was presented at the Sixth Int. Conference on Advanced Engineering Computing and Applications in Sciences, Barcelona, Spain, September 2012.

References

[1] D. S. Rosenblum, "Towards a method of programming with assertions," in *Proceedings of the International Conference on Software Engineering*, pp. 92–104, May 1992.

[2] B. Korel and A. M. Al-Yami, "Assertion-oriented automated test data generation," in *Proceedings of the 18th International Conference on Software Engineering*, pp. 71–80, Berlin, Germany, March 1996.

[3] A. M. Alakeel, "An algorithm for efficient assertions-based test data generation," *Journal of Software*, vol. 5, no. 6, pp. 644–653, 2010.

[4] A. M. Alakeel and M. Mahashi, "Using assertion-based testing in string search algorithms," in *Proceedings of the 3rd International Conference on Advances in System Testing and Validation Lifecycle*, pp. 1–5, Barcelona, Spain, 2011.

[5] A. M. Alakeel, "A framework for concurrent assertion-based automated test data generation," *European Journal of Scientific Research*, vol. 46, no. 3, pp. 352–362, 2010.

[6] B. Korel, Q. Zhang, and L. Tao, "Assertion-based validation of modified programs," in *Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation (ICST '09)*, pp. 426–435, Denver, Colo, USA, April 2009.

[7] A. M. Alakeel, "Regression testing method for programs with assertions," *American Journal of Scientific Research*, no. 11, pp. 111–122, 2010.

[8] A. M. Alakeel, "Intelligent assertions placement scheme for string search algorithms," in *Proceedings of the 2nd International Conference on Intelligent Systems and Applications*, pp. 122–128, Venice, Italy, April 2013.

[9] W. Hetzel and B. Hetzel, *The Complete Guide to Software Testing*, John Wiley & Sons, New York, NY, USA, 1991.

[10] S. Beydeda and V. Gruhn, "An integrated testing technique for component-based software," in *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, pp. 328–334, 2001.

[11] W.-T. Tsai, X. Bai, R. Paul, and L. Yu, "Scenario-based functional regression testing," in *Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC '01)*, pp. 496–501, October 2001.

[12] B. Korel, L. H. Tahat, and B. Vaysburg, "Model based regression test reduction using dependence analysis," in *Proceedings of the IEEE International Conference on Software Maintenance*, pp. 214–223, October 2002.

[13] Y.-F. Chen, D. S. Rosenblum, and K.-P. Vo, "Test tube: a system for selective regression testing," in *Proceedings of the 16th International Conference on Software Engineering*, pp. 211–220, May 1994.

[14] R. Gupta, M. Harrold, and M. Soffa, "An approach to regression testing using slices," in *Proceedings of the IEEE International Conference on Software Maintenance*, pp. 299–308, 1992.

[15] B. Korel and A. Al-Yami, "Automated regression test generation," in *Proceedings of the ACM International Symposium on Software Testing and Analysis*, pp. 143–152, 1998.

[16] G. Rothermel and M. J. Harrold, "A safe, efficient regression test selection technique," *ACM Transactions on Software Engineering and Methodology*, vol. 6, no. 2, pp. 173–210, 1997.

[17] B. Beizer, *Software System Testing and Quality Assurance*, Thomson Computer Press, 1996.

[18] G. Rothermel and M. J. Harrold, "Safe, efficient algorithm for regression test selection," in *Proceedings of the IEEE International Conference on Software Maintenance*, pp. 358–367, 1994.

[19] W. Masri, A. Podgurski, and D. Leon, "An empirical study of test case filtering techniques based on exercising information flows," *IEEE Transactions on Software Engineering*, vol. 33, no. 7, pp. 454–477, 2007.

[20] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, "The effects of time constraints on test case prioritization: a series of controlled experiments," *IEEE Transactions on Software Engineering*, vol. 36, pp. 593–617, 2010.

[21] G. Rothermel, R. H. Untcn, C. Chu, and M. J. Harrold, "Prioritizing test cases for regression testing," *IEEE Transactions on Software Engineering*, vol. 27, no. 10, pp. 929–948, 2001.

[22] R. C. Bryce, S. Sampath, and A. M. Memon, "Developing a single model and test prioritization strategies for event-driven software," *IEEE Transactions on Software Engineering*, vol. 37, no. 1, pp. 48–64, 2011.

[23] B. Korel, G. Koutsogiannakis, and L. H. Tahat, "Application of system models in regression test suite prioritization," in *Proceedings of the 24th IEEE International Conference on Software Maintenance (ICSM '08)*, pp. 247–256, October 2008.

[24] B. Korel, L. H. Tahat, and M. Harman, "Test prioritization using system models," in *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM '05)*, pp. 559–568, September 2005.

[25] C. Cagatay and D. Mishra, "Test case prioritization: a systematic mapping study," *Software Quality Journal*, vol. 21, no. 3, pp. 445–478, 2013.

[26] C. Malz, N. Jazdi, and P. Gohner, "Prioritization of test cases using software agents and fuzzy logic," in *Proceedings of the 5th IEEE International Conference on Software Testing, Verification and Validation (ICST '12)*, pp. 483–486, April 2012.

[27] Y.-C. Huang, K.-L. Peng, and C.-Y. Huang, "A history-based cost-cognizant test case prioritization technique in regression testing," *Journal of Systems and Software*, vol. 85, no. 3, pp. 626–637, 2012.

[28] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[29] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1992.

[30] Z. Xu, K. Gao, and T. M. Khoshgoftaar, "Application of fuzzy expert system in test case selection for system regression test," in *Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI '05)*, pp. 120–125, August 2005.

[31] S. Horwitz, T. Reps, and D. Binkley, "Interprocedural slicing using dependence graphs," *ACM Transactions on Programming Languages and Systems*, vol. 12, no. 1, pp. 26–60, 1990.

[32] M. Weiser, "Program slicing," *IEEE Transactions on Software Engineering*, vol. 10, no. 4, pp. 352–357, 1984.

[33] J. Giarratano and G. Riely, *Expert Systems: Principles and Programming*, PWS-KENT, Boston, Mass, USA, 1989.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 273807, 12 pages
http://dx.doi.org/10.1155/2014/273807

Research Article

AP-IO: Asynchronous Pipeline I/O for Hiding Periodic Output Cost in CFD Simulation

Ren Xiaoguang and Xu Xinhai

State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha, Hunan 410073, China

Correspondence should be addressed to Ren Xiaoguang; hbszrxg@gmail.com

Received 26 December 2013; Accepted 2 March 2014; Published 3 April 2014

Academic Editors: S. K. Bhatia and P. Muller

Computational fluid dynamics (CFD) simulation often needs to periodically output intermediate results to files in the form of snapshots for visualization or restart, which seriously impacts the performance. In this paper, we present asynchronous pipeline I/O (*AP-IO*) optimization scheme for the periodically snapshot output on the basis of asynchronous I/O and CFD application characteristics. In *AP-IO*, dedicated background I/O processes or threads are in charge of handling the file write in pipeline mode, therefore the write overhead can be hidden with more calculation than classic asynchronous I/O. We design the framework of *AP-IO* and implement it in OpenFOAM, providing CFD users with a user-friendly interface. Experimental results on the *Tianhe-2* supercomputer demonstrate that *AP-IO* can achieve a good optimization effect for the periodical snapshot output in CFD application, and the effect is especially better for massively parallel CFD simulations, which can reduce the total execution time up to about 40%.

1. Introduction

Currently, computational fluid dynamics (CFD) is increasingly applied in science and engineering with the continuous development of computer technology, especially the ever-growing computing power of supercomputers [1–3]. In order to meet the needs of visualization or restart, CFD simulation has to periodically output intermediate results to files in the form of *snapshot*.

Unfortunately, this kind of periodic snapshot output has gradually become a performance bottleneck in CFD simulations on massively parallel system. In addition, according to some experimental data [4–6], the periodic large volume snapshot file output can take more than 80% time of the whole CFD simulation process in extreme case. On the one hand, the growth rate of I/O speed is far behind the growth rate of computing capacity [7, 8], which makes I/O become a bottleneck; on the other hand, the requirement of all-refined numerical simulation promotes the mesh scale of many CFD applications to million scale, resulting in the snapshot file tremendous, that is, Gigabytes or even Terabytes [9, 10]. Therefore, it is urgent to optimize the periodic large volume snapshot file output in massively parallel CFD simulations.

Asynchronous I/O is a widely used I/O optimization technique, in which dedicated I/O processes/threads are used to handle the file write operations, and the I/O overhead is hidden by overlapping I/O operations with calculation operations [11, 12]. The key of the asynchronous I/O optimization is to get enough calculation operations to hide the I/O overhead on the premise of data consistency; that is, for any set of data, its asynchronous write operation should be finished before the next update operation.

A CFD simulation, as one kind of typical compute-intensive application, has an immense potential to use asynchronous I/O technology to optimize large-volume file output. However, the snapshot output in traditional CFD simulation always happens at the end of a timestep, which is close to the update operation of the same data at the beginning of next timestep. Thus, there is no enough time for asynchronous I/O to hide the file write overhead.

By analyzing the simulation procedure of CFD applications, which is based on the time step iterations, we find

that each timestep's snapshot generally consists of multiple physical data arrays called *field*, and these fields are updated at different time. This potentially allows to output these fields asynchronously in pipeline mode. In this paper, we propose *AP-IO*, a new asynchronous pipeline I/O optimization method for periodic large-volume snapshot output in massively parallel CFD applications. Instead of outputting snapshot's multiple fields together at the end of the time step, AP-IO asynchronously outputs these different fields in a pipeline mode.

The main contributions of this paper are summarized below.

(i) We introduce *AP-IO*, an asynchronous pipeline I/O method, to solve the I/O performance bottleneck problem of periodic large volume snapshot output in CFD applications.

(ii) We propose an application level calculation segments scheduling method, which schedules the order of the calculation segments of different fields, to obtain more hidden time for *AP-IO*.

(iii) We implement the user-level *AP-IO* framework in the open source CFD software OpenFOAM with the Pthread and compiler-directed instruction techniques.

(iv) We show by experiments on *Tianhe-2* supercomputer with three typical CFD application cases that *AP-IO* optimization method can significantly enhance the performance of periodic large-volume snapshot output in CFD applications. Specifically, the file write overhead can be reduced 60% on average, and the total execution time can be reduced 18% on average.

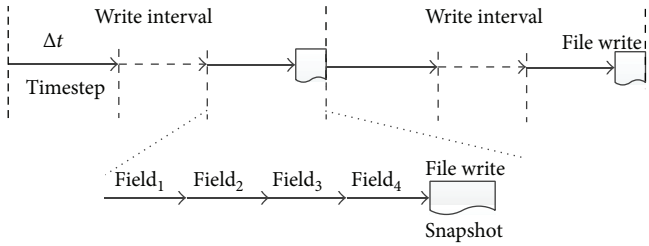
The rest of the paper is organized as follows. Section 2 introduces the motivation of our work. Section 3 introduces the basic idea of *AP-IO*. Section 4 describes the framework of *AP-IO*, including the programming model, the compiler-directed foundation framework, and the calculation segments scheduling optimization. Section 5 introduces the implementation of *AP-IO* in OpenFOAM software. Section 6 describes our evaluation methodology and demonstrates the superiority of *AP-IO* by 3 CFD applications. Finally, Section 7 concludes the paper.

2. Motivation

2.1. Snapshot Output in CFD. In a typical CFD application, on the one hand, the simulation procedure is discretized into several timesteps, and each timestep is denoted with Δt. It is notable that different timesteps are simulated in the same computing mode, but with different initial condition value. On the other hand, the simulation region is discretized into a mesh system, and the physical property in the simulation region is represented by a *filed*, that is, a data array with the same size of the mesh. As shown in Figure 1(a), each timestep is usually simulated by computing multiple fields one by one. The set of these fields, namely, *snapshot*, is the intermediate simulation result of the timestep.

```
While(timestep++)
{
   update(field1);
   update(field2);
   update(field3);
   if(output)
      write(field1, field2, field3)
}
```

(a) Example code of Snapshot output



(b) The snapshot output mode

Figure 1: Example code and snapshot output mode of a typical CFD application.

For the purpose of visualization or restart, a CFD simulation procedure needs to periodically output the snapshot to files on the disk. As shown in Figure 1(b), the snapshot is outputted at the end of *write interval*, which is consisted of fixed number of timesteps.

Nowadays, the mesh size of massively parallel CFD applications reaches million level, and the volume of a snapshot often scales to Gigabyte level. Unfortunately, outputting such large snapshot on HPC system seriously affects the performance of the whole CFD simulation. The reasons are as follows: (a) multiple users or multiple processes competing for shared I/O channel; (b) with the scale of HPC increasing, the performance of computing capability improves more quickly than that of I/O; (c) the redundant data produced by the parallel task partitioning makes the volume of snapshot rise. All these influences will be discussed in detail in Section 6.

According to our experiments as well as in related documents, the file write overhead consumes 20% of the whole CFD simulation time on average, which can reach even more than 80% in extreme circumstances [4–6]. Thus, it is critically important to optimize the periodical large volume snapshot output in massively parallel CFD applications.

2.2. Traditional Optimization for File Write. I/O performance optimizations mainly performed in two ways: (a) reducing I/O overhead and (b) hiding I/O overhead.

Buffer mechanism [8], which reduces I/O overhead through buffer caching, is the most universal technology used for reducing I/O overhead in the modern computer system. However, the buffer mechanism is restrained by buffer capacity and difficult to be applied in large-volume file output. In most modern computer systems, the buffer size

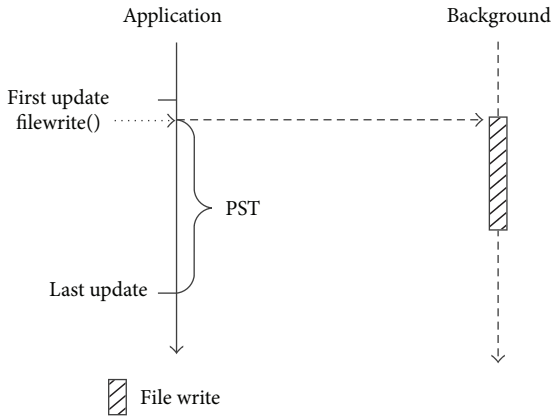


Figure 2: Asynchronous I/O for large volume file.

is less than 1 Megabyte, then the buffer mechanism cannot optimize the CFD snapshot output, which is usually more than hundreds of Megabytes.

Asynchronous I/O is another optimization technology used for hiding I/O overhead. The core idea of asynchronous I/O is to use dedicated routines (processes/threads) in background to perform I/O operation, which will hide the file write overhead with calculation operations [13]. As shown in Figure 2, the asynchronous write operations in the background overlap with calculation operations and the file write overhead is hidden. To guarantee the data consistency, for any set of data, the background asynchronous write operation must be finished before the next update operation of the same data.

So, for asynchronous I/O, it is critical that there are enough calculation operations to hide the file write overhead. As a matter of convenience, we define *potential shield time (PST)* as the calculation time used to hide the write overhead. To a data set, if the output operation is closed to the next update operation of the same data, the *PST* will not be sufficient to hide the file write overhead.

3. Basic Idea

3.1. The Basic Idea of AP-IO. In the parallel CFD simulation, there are multiple simulation processes which we call as *compute processes*. As shown in Figure 3(a), it is an example of the asynchronous I/O method applied for snapshot output in the CFD simulation. For each compute process, there is a corresponding dedicated write *service routine* (process/thread) to deal with the file write requests. When a compute process needs to do a write operation, it sends a relevant write request to its corresponding write service routine, which handles the request in background.

A prominent problem of the above traditional asynchronous I/O for the snapshot output in the CFD simulation is the shortage of *PST*. The outputs of all the fields of a snapshot are concentrated at the end of the timestep, which is close to their update operations in the next timestep. In asynchronous I/O, to guarantee the data consistency, these fields' update operations in the next timestep cannot be performed until the asynchronous write operations in the current timestep are completed. This situation makes it clear that there is no *PST* for the asynchronous snapshot output.

In order to solve this problem, we proposed the *AP-IO* method. *AP-IO* method mainly originates from the updating characteristic of the fields of the snapshot. In general, as shown in Figure 4(a), a field may be updated for several times in a timestep, We call an updating procedure as *calculation segment*, and the calculation segments of a field are stagger with other fields' calculation segments. For a field in a single timestep, we define the *First Update (FU)* time and the *Last Update (LU)* time. The *FU* time of a field is the starting point of the field's first calculation segment in a timestep, and the *LU* time is the end point of the field's last calculation segment in a timestep.

In *AP-IO*, as shown in Figure 3(b), the fields of the snapshot are not outputted together at the end of the time step. On the contrary, for each field, its output follows its *LU* time in the current timestep. To guarantee the data consistency, the field's output operation should be finished before its *FU* time in the next time step. So, the *PST* of a field is the time between the *LU* time in the current timestep and the *FU* time in the next timestep. Compared with the centralized output at the end of timestep, *AP-IO* gets more *PST*, and the field's output operation will be overlap with other fields' calculation operations.

3.2. Analysis of PST. Providing more *PST* is the key advantage of *AP-IO*, and we will analyze the characteristics of PST in CFD simulation in this section.

The distribution of a field's calculation segments in a single timestep has a direct influence on the length of its *PST* in the *AP-IO* mode. Considering the CFD simulation process shown in Figure 4(a), the $field_0$'s calculation operations go across the entire timestep. $field_0$'s *LU* time in the current timestep is overlapped with its *FU* time in the next timestep, resulting in the fact that there is no *PST* for $field_0$. To obtain more *PST* for $field_0$, as shown in Figure 4(b), we exchange $field_1$'s first calculation segment with $field_0$'s first calculation segment. Now, $field_0$ can get a *PST* PST_0 as shown in Figure 4(b).

In order to increase a fields' *PST*, we need to improve the distribution of fields' calculation segments in the snapshot. A calculation segments scheduling optimization method, aiming at improving the distribution of fields, is proposed in this paper, which will be described in Section 4.3.

4. Framework

We design the user-level framework of *AP-IO* by combing the compiler-directed technology with the *AP-IO* library, which is convenient for CFD users to carry out the *AP-IO* optimization in their CFD applications.

4.1. Programming Model. We define two compiler directives for the *AP-IO* optimization, and their semantics are listed as follows.

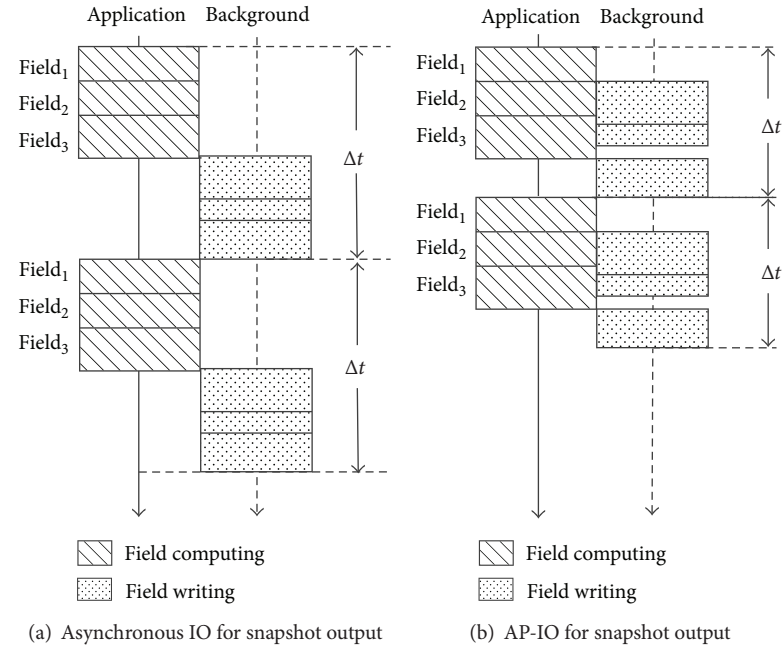


Figure 3: Asynchronous IO versus AP-IO for CFD snapshot output.

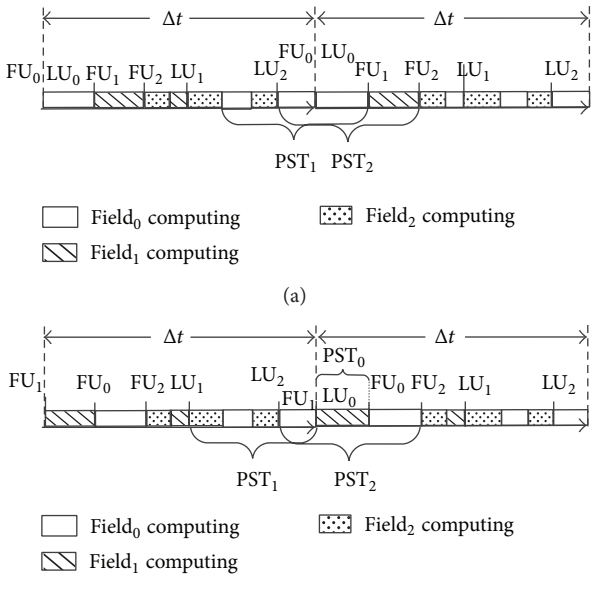


Figure 4: Example of *PST*.

(i) "#FU $field_1$": to explicitly identify $field_1$'s *FU* time, which means when the execution goes across this point, the asynchronous write of $field_1$ is no longer safe, and it is required to take a blocking wait to guarantee the finish of $field_1$'s asynchronous write.

(ii) "#LU $field_1$": to explicitly identify $field_1$'s *LU* time, which means that from this point on, the request of $field_1$'s write can be sent to the background write service routine to perform asynchronous write.

We will illustrate the *AP-IO* programming model with an example. The left part of Figure 5 shows a typical CFD application code based on timestep iteration. The update procedure of $field_1$ in the code has three calculation segments: $update1(), update2(), update3()$. If we use *AP-IO* to optimize the periodic snapshot output, we should add the compile directives in the corresponding position of the code for each field. Take $field_1$ as an example, we need to add compile directives "#FU $field_1$" in front of $update1()$ and "#LU $field_1$" after $update3()$, respectively. These two compile directives explicitly define the *PST* of $field_1$.

This compiler-directive based programming model makes minimum modification of the original CFD application code and enables the CFD application developers to optimize with the *AP-IO* method at minimum cost. It is notable that the compiler directives can be automatically inserted through compile techniques. However, it is not the focus of this paper.

4.2. Framework of AP-IO. Figure 5 presents our *AP-IO* framework, the core components which include both the *AP-IO* compiler and *AP-IO* library. The *AP-IO* compiler is a compile directives based source-to-source compiler. The CFD application code in the left part of Figure 5 is compiled into the code including *AP-IO* library calls which is shown in the right part of Figure 5.

The *AP-IO* library provides semantic interfaces support for the application code generated by the *AP-IO* compiler. As shown in Figure 5, after preprocessing by the *AP-IO* compiler, the compiler directives, "#LU $field_1$" and "#FU $field_1$", are, respectively, replaced with *AP-IO* library calls: $Iwrite(field_1)$ and $Iwait(field_1)$, whose semantic functions are, respectively, listed as follows.

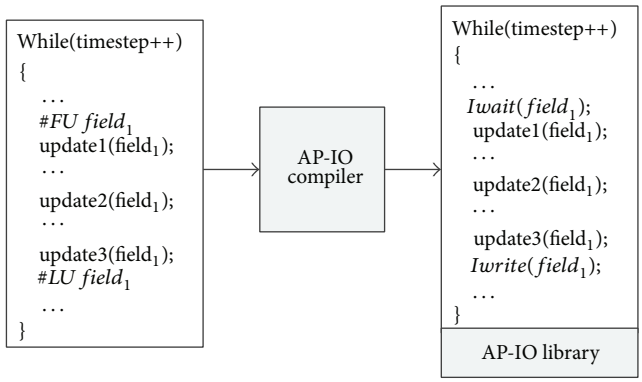


Figure 5: Framework of *AP-IO*.

(i) *Iwrite(field$_1$)*: responsible for sending the *field$_1$*'s write request to the background service routine. The function will return as soon as the request is successfully sent. *Iwrite()* semantics encapsulate three work: the send of asynchronous write request, FIFO-mode queuing of write requests in the write service routine, background write, and other operations.

(ii) *Iwait(field$_1$)*: responsible for checking whether the corresponding field's asynchronous write operation has been finished. If it has been already finished, a success code will be returned. Otherwise, a blocking wait must be hold until the finish of the asynchronous write operation, to avoid updating the field whose write operation is underway.

4.3. Calculation Segments Scheduling Optimization. In order to gain the benefit of *AP-IO*, our AP-IO compiler can also help the developer to optimize the distribution of the calculation segments of each field in a single timestep. Achieving the most optimal scheduling solution is a NP-hard problem. Therefore, we design a heuristic algorithm for the scheduling of field's calculation segments in a single timestep, which is shown in Algorithm 1.

The core idea of our heuristic algorithm is to improve the distribution of each field's calculation segments, that is, reduce the intersection between different fields' calculation segments. First, the scheduling algorithm numbers the fields in a single timestep according to the order of each field's *FU* time. During the scheduling, the algorithm moves the odd number field's segments as close as possible to its first calculation segment, and the even number field's segments as close as possible to its last calculation segment. "As possible" means maintaining the dependencies between the calculation segments, and moving the calculation segments as much as possible to the corresponding direction.

5. Implementation of *AP-IO*

We implement the *AP-IO* method in the open source CFD software OpenFOAM with the source-to-source compiler technology and the Pthread technology, which mainly includes two parts: the compiler support layer and the *AP-IO* library.

5.1. Compiler Support Layer. With the source-to-source compile technology, the compiler support layer mainly includes two parts: transforming the compiler directives to *AP-IO* library function calls and scheduling the fields' calculation segments in the origin OpenFOAM application code with the heuristic scheduling algorithm in Algorithm 1.

5.2. AP-IO Library. The *AP-IO* library mainly involves three parts: compute processes, write service threads, and write task queue, as shown in Figure 6.

5.2.1. Iwrite. First, a Pthread thread is created for each parallel simulation process (compute process) as the write service thread. A write task queue is shared by each compute process and its write service thread. When the compute process trigger the function call *Iwrite()*, a write request will be added to the write task queue. The write service thread in the background constantly scans the write task queue, and all the tasks in the queue are processed in FIFO mode.

A status variable *write_state* is set for each field, and it has three states: *WAIT_WRITE*, *WRITING*, and *WRITE_FINISHED*, the meanings of them are listed as follows.

(i) *WAIT_WRITE* means that this field's write request in the write task queue is waiting to be handled by the write service thread.

(ii) *WRITING* means that this field's write request is being handled by the write service thread and this field is writing in the background now.

(iii) *WRITE_FINISHED* means that this field's write request had been finished, or there is no write request for this field.

For a field, the default status of *write_state* is *WRITE_FINISHED*. When the function *Iwrite(field$_1$)* is called in compute process, the write request of *field$_1$* is added into the write task queue, and *write_state* of *field$_1$* is set as *WAIT_WRITE*. When the write request begins to be handled by the write service thread, *write_state* of *field$_1$* will be set as *WRITING*. Finally, the status will be set to *WRITE_FINISHED* when the write request is finished.

5.2.2. Iwait. To guarantee the data consistency, the compile-directive "#FU *field$_1$*" explicitly specifies the opportunity of checking whether *field$_1$*'s write has been finished. The function call *Iwait()* in the compute process is in charge of the checking work. The checking work of *Iwait()* is performed by reading the status of *write_state*. If the status of *write_state* is *WRITE_FINISHED*, *Iwait()* returns with a success code. Otherwise, a blocking wait is launched and the waiting procedure will not finish until the status of *write_state* is updated to *WRITE_FINISHED*.

INPUT: the origin order set of calculation segments in a time step F, which includes n calculation segments; the matrix of dependence relationship between calculation segments D;
OUTPUT: the new order set of calculation segments in a time step F_{new};
(1) **procedure** SCHEDULING(F, D) // calculation segments scheduling procedure
(2) $F_{new} = F$
(3) **while** $F_{new} \neq F_{old}$ **do**
(4) $F_{old} = F_{new}$;
(5) **for** each $i \in [1, n]$ **do**
(6) **if** $(K(S_i)\%2 == 0 \&\& D(i, i+1) == 0 \&\& S_i \neq S^{first}_{k(S_i)})$ **then** // $k(S_i)$ indicates the field number which S_i belongs to
(7) S_i exchange with S_{i+1};
(8) **end if**
(9) **if** $(K(S_i)\%2 == 1 \&\& D(i-1, i) == 0 \&\& S_i \neq S^{last}_{k(S_i)})$ **then**
(10) S_{i-1} exchange with S_i;
(11) **end if**
(12) **end for**
(13) **if** $(F_{new} == F_{old})$ **then**
(14) break;
(15) **end if**
(16) **end while**
(17) **end procedure**

Algorithm 1: A heuristic algorithm for the scheduling of fields' calculation segments in a single timestep.

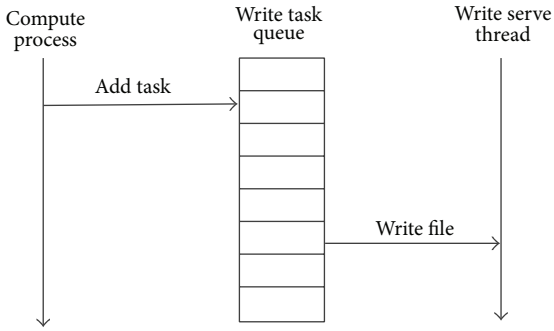


Figure 6: Implement of *AP-IO*.

5.2.3. Threading Optimizations. The above mechanism can basically perform the *AP-IO* optimization in the OpenFOAM platform, but there are two situations that should be taken into account for the CPU consumption, which influence the performance optimization.

The first situation is shown in Figure 7(a). In the *Iwait()* function call, if the status of *write_state* is *WRITING* or *WAIT_WRITE*, a blocking wait is needed. This blocking wait has two implementation ways. One is to constantly check the status of *write_state* with *while* iteration, which will seriously consume CPU resources, resulting in low efficiency. So, we take another more efficient way. In the procedure of the blocking wait, the compute process is suspended and sleeping to release CPU resources. Once the write service thread finishes the current write request, it awakens the sleeping compute process by sending a signal to it. Therefore, the compute process can greatly reduce the CPU resources consumption during the sleep time.

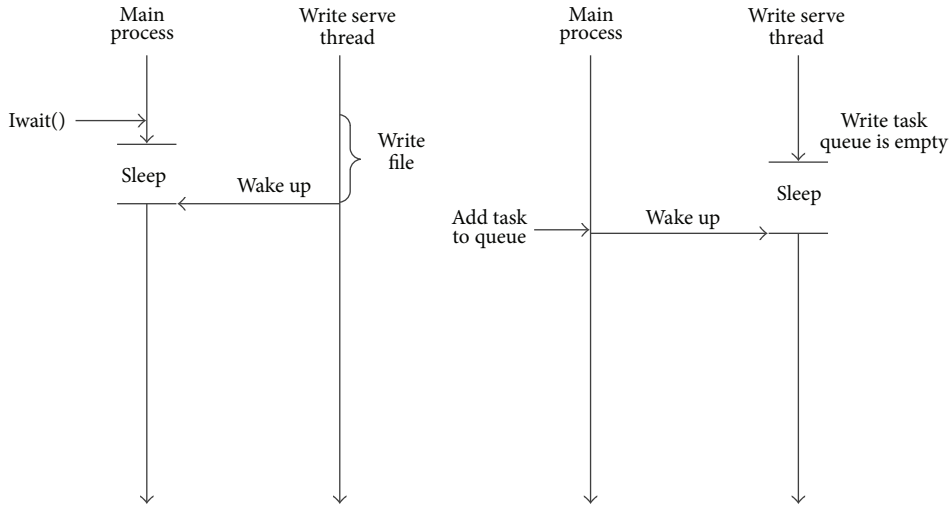
The second situation is shown in Figure 7(b), it is related to the write service thread's scan of the write task queue. When the queue is empty and no write request needs to be handled, to timely responds to the new write request, the write service thread has to pay close attention to the change of the write task queue. Similar to the first situation, once the write service thread finds that the write task queue is empty, it suspends itself to sleep and releases CPU resources. The compute process always tries to send an awake signal to the write service thread when a new write request arrives. If the write service thread is sleeping, it will be awakened at the time of new request's arrival.

The two suspend/awake mechanism can ensure that the compute process and the write service thread will not consume too many CPU resources, which guarantees the efficient execution of *AP-IO* in CFD simulation.

6. Experiment

We demonstrate the superiority of our *AP-IO* method on the Tianhe-2 supercomputer with three typical CFD application cases. Section 6.1 introduces the computer platform Tianhe-2 used in the experiments. Section 6.2 discusses the benchmarks selected. Section 6.3 describes the experiment environment and the evaluation methodology. Section 6.4 presents and analyzes our results.

6.1. Platform. The *Tianhe-2* supercomputer was built by China's National University of Defense Technology (NUDT), which is the world's fastest supercomputer with a 33.86 petaflop peak performance according to the TOP500 list in June 2013 [14]. Our experiments use a subsystem of *Tianhe-2* with 128 compute nodes as the test platform, and



(a) Thread optimizations for *Iwait()* (b) Thread optimizations for write task queue

Figure 7: Threading optimizations for *AP-IO*.

each computing node comprises two Intel Ivy Bridge Xeon processors, three Xeon Phi chips (accelerator card), and 64 Gigabyte DRAM [15]. It should be pointed out that our experiments only use the processors, and do not use the accelerator card.

The whole I/O subsystem of *Tianhe-2* has a disk array with a 12.4 Petabyte capacity, and all the nodes in the system share the disk array through a global interconnecting network. The interconnecting network uses the optoelectronics hybrid transport technology, connecting all the compute nodes through 576 connection ports with 13 large routers, and the transmission rate reaches 6.36 GB/s.

It is noticeable that although the subsystem we use is exclusive for us, the I/O system is globally shared with other users, which can lead to competition with other users and impact the precision of the experimental results. A detailed analysis of this problem will be described in Section 6.3.1.

6.2. Benchmark. To verify the superiority of *AP-IO*, we select three typical large-scale CFD applications as benchmarks. They are *dambreak*, *cavity,* and *pitzDaily*. The three benchmarks are briefly introduced as follows.

(i) *Dambreak*: *dambreak* is a simulation of dambreak based on the volume of fluid (VOF) method. The two-phase flow solver *interFoam* in OpenFOAM is used.

(ii) *Cavity*: *cavity* is the lid-driven cavity flow simulation, using the incompressible flow solver *icoFoam* in OpenFOAM.

(iii) *PitzDaily*: *pitzDaily* is a large eddy simulation (LES) case. The solver used is *pisoFoam* together with the *oneEqEddy* eddy viscosity model.

Some important parameters of the three benchmarks are shown in Table 1. All the benchmarks are in 3D with the grid scale between 5 M and 17 M. To verify the effect of the write interval to the performance of snapshot output, we tested five write interval conditions, including the none condition with no snapshot output. All the three benchmarks take *Scotch* as the decompose method for parallel grid partition.

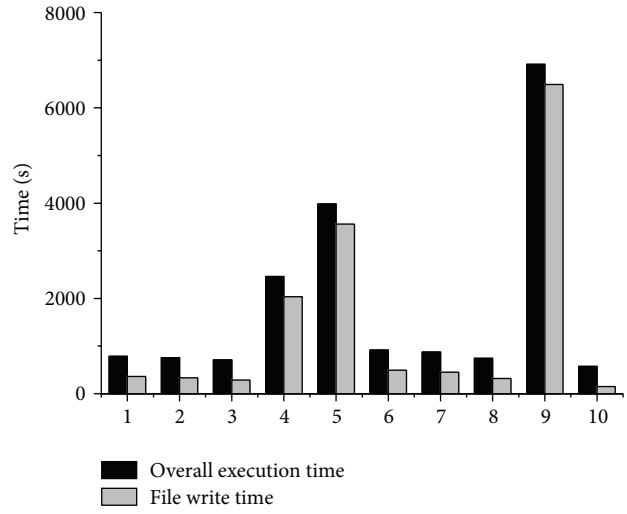


Figure 8: The execution time of *cavity* runs on *Tianhe-2* for 10 times.

6.3. Experiment Design

6.3.1. Experiment Environment Analysis. As mentioned in Section 6.1 that we use the I/O subsystem shared with other users, our experimental results are more or less random. To measure this randomness, we test the *cavity* benchmark for 10 times with 256 processors. The test is performed in a continuous time, during which the *Tianhe-2* system is running hundreds of other users' tasks at the same time, and these tasks may also perform file write operations to the disk array. The execution times are shown in Figure 8.

Table 1: The parameters of the three benchmark.

	Dambreak	Cavity	PitzDaily
Solver	interFoam	icoFoam	pisoFoam
Dimensionality	3D	3D	3D
Number of cells	6229504	15625000	16131636
Δt	$1.00E-03$	$1.00E-04$	$2.00E-04$
Number of time steps	100	100	100
Write interval (time Steps)	1/2/4/10/none	1/2/4/10/none	1/2/4/10/none
Number of fields	5	8	8
Decomposition method	Scotch	Scotch	Scotch

According to Figure 8, we can find that the execution times of the 4th, 5th, and 9th test are much higher than those of others. The gap reaches 10 times, which indicates severe abnormalities. Therefore, to eliminate the influence on I/O performance from the multiuser and multitask environment, we perform a test for multiple times and choose the average of the execution time as experiment result. Generally speaking, all the following experiments results are the average value of multiple tests.

6.3.2. Evaluation Methodology. The purpose of our experiments is to obtain the performance optimization effect of *AP-IO* in OpenFOAM by comparison of results before and after the optimization. The tests are performed in two dimensions: multiple numbers of processors and multiple write intervals. Our tests start from 4 processors and scale to 1024 processors with a scale coefficient of 2. Meanwhile, to verify the write interval's effect on I/O performance, we have to get all the benchmarks tested, respectively, with $1\Delta t$, $2\Delta t$, $4\Delta t$, $10\Delta t$, and none (no snapshot output) write interval conditions.

Our experiments mainly tested three indicators of the performance: the growth of the output file's volume that scales with the number of processors, the original file write overhead without *AP-IO* optimization, and the optimization effect with *AP-IO* method. The first two aspects mainly analyze the I/O characteristics of the tree benchmarks, which will be taken as the baseline. The third aspect is to measure the effect of *AP-IO* optimization directly.

First, to measure the growth of the output volume with the influence of the number of processors, we get the three benchmarks tested under the number of processors ranging from 2 to 1024 and four write interval conditions ($1\Delta t$, $2\Delta t$, $4\Delta t$, and $10\Delta t$). Recording the volume of the output file, we will get the characteristics of file volume growth due to the parallel grid partition.

Second, we have to measure the overhead of the periodic snapshot output without *AP-IO* optimization. Suppose that the execution time under the none write interval condition is $T_{0\Delta t}$, and the execution time under $n\Delta t$ write interval condition without optimization is $T^{\text{prim}}_{n\Delta t}$, then the file write overhead $T^{\text{prim}}_{\text{file}}$ can be approximated by the following methods as

$$T^{\text{prim}}_{\text{file}} = T^{\text{prim}}_{n\Delta t} - T_{0\Delta t}. \quad (1)$$

Finally, the optimization effect of *AP-IO* is measured with two indicators: one is the optimization ratio to the snapshot output overhead, and the other is the optimization ratio to the total simulation time. We define the total execution time of the simulation with *AP-IO* under $n\Delta t$ write interval condition as $T^{\text{opt}}_{n\Delta t}$, and then the file write overhead with *AP-IO* optimization $T^{\text{opt}}_{\text{file}}$ can be approximated as s

$$T^{\text{opt}}_{\text{file}} = T^{\text{opt}}_{n\Delta t} - T_{0\Delta t}. \quad (2)$$

The optimization ratio of *AP-IO* to the file write overhead α is

$$\alpha = \frac{T^{\text{prim}}_{\text{file}} - T^{\text{opt}}_{\text{file}}}{T^{\text{prim}}_{\text{file}}}. \quad (3)$$

The optimization ratio of *AP-IO* to the total simulation time β is

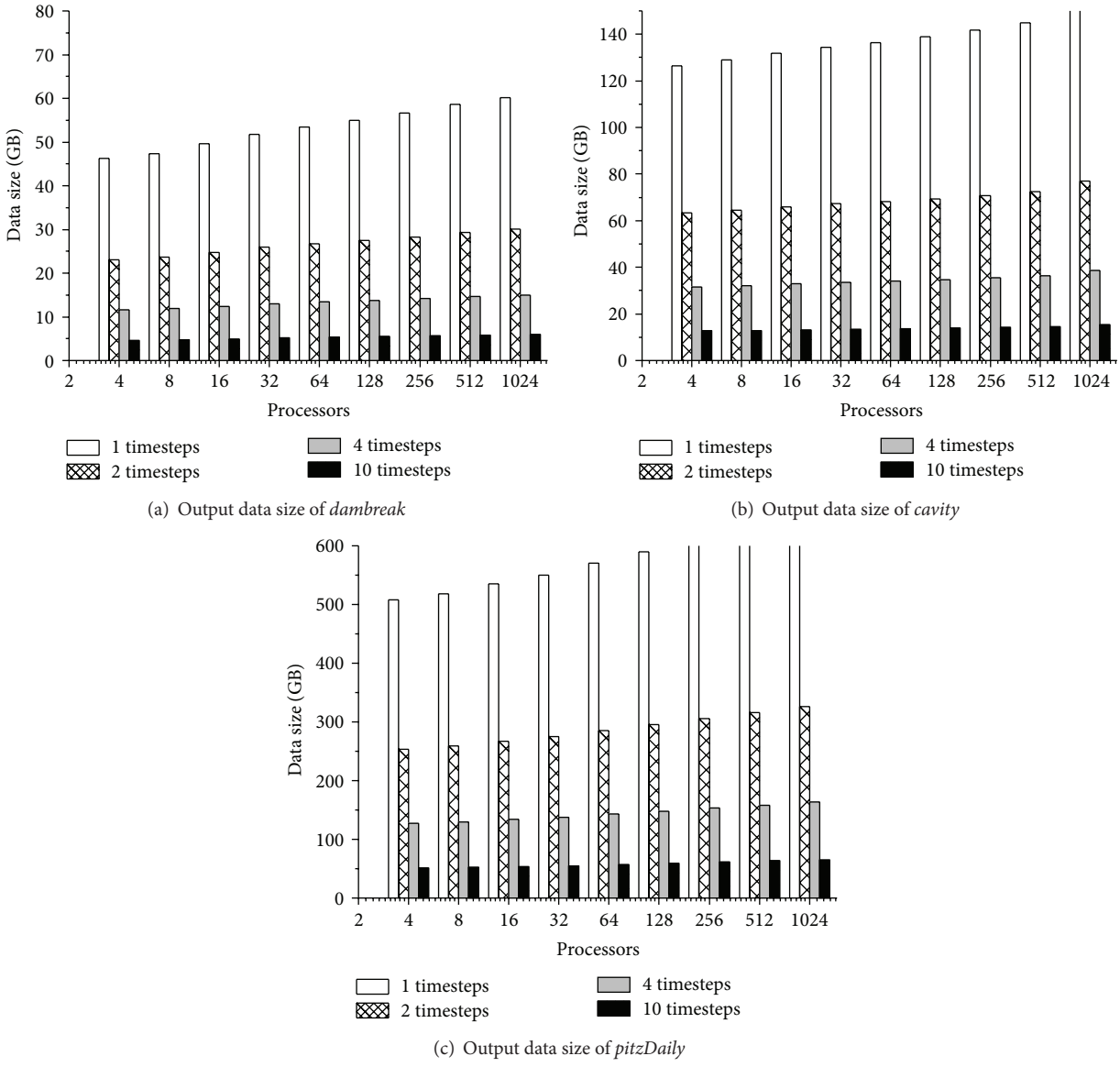
$$\beta = \frac{T^{\text{prim}}_{n\Delta t} - T^{\text{opt}}_{n\Delta t}}{T^{\text{prim}}_{n\Delta t}}. \quad (4)$$

6.4. Results and Analysis

6.4.1. Volume of the Snapshot Output. As shown in Figure 9, they are the total volumes of the three benchmarks' (*dambreak, cavity, and pitzDaily*) snapshot output tested with the number of processors from 2 to 1024 under different write interval conditions. We can find that the total volume scales with the same rate as the number of processors grows; the rate is about 2% for *dambreak*, about 6% for *cavity*, and 3% for *pitzDaily*. The growth of the volumes is mainly due to the data redundancy in the grid and fields produced by parallel grid domain decompose. Although the growth rate is less than 10%, it still has severe influence on I/O performance, which will be verified and analyzed in Section 6.4.2.

6.4.2. File Write Overhead. To get the baseline of the *AP-IO* optimization, we first have to measure the periodical snapshot output overhead for the entire simulation. Figure 10 shows the three benchmarks' file write overhead.

As mentioned before, we measure the file write overhead indirectly with the difference between the execution time under certain write interval conditions and the none write interval condition. Figures 11(a), 11(d), and 11(g) are the



(a) Output data size of *dambreak*

(b) Output data size of *cavity*

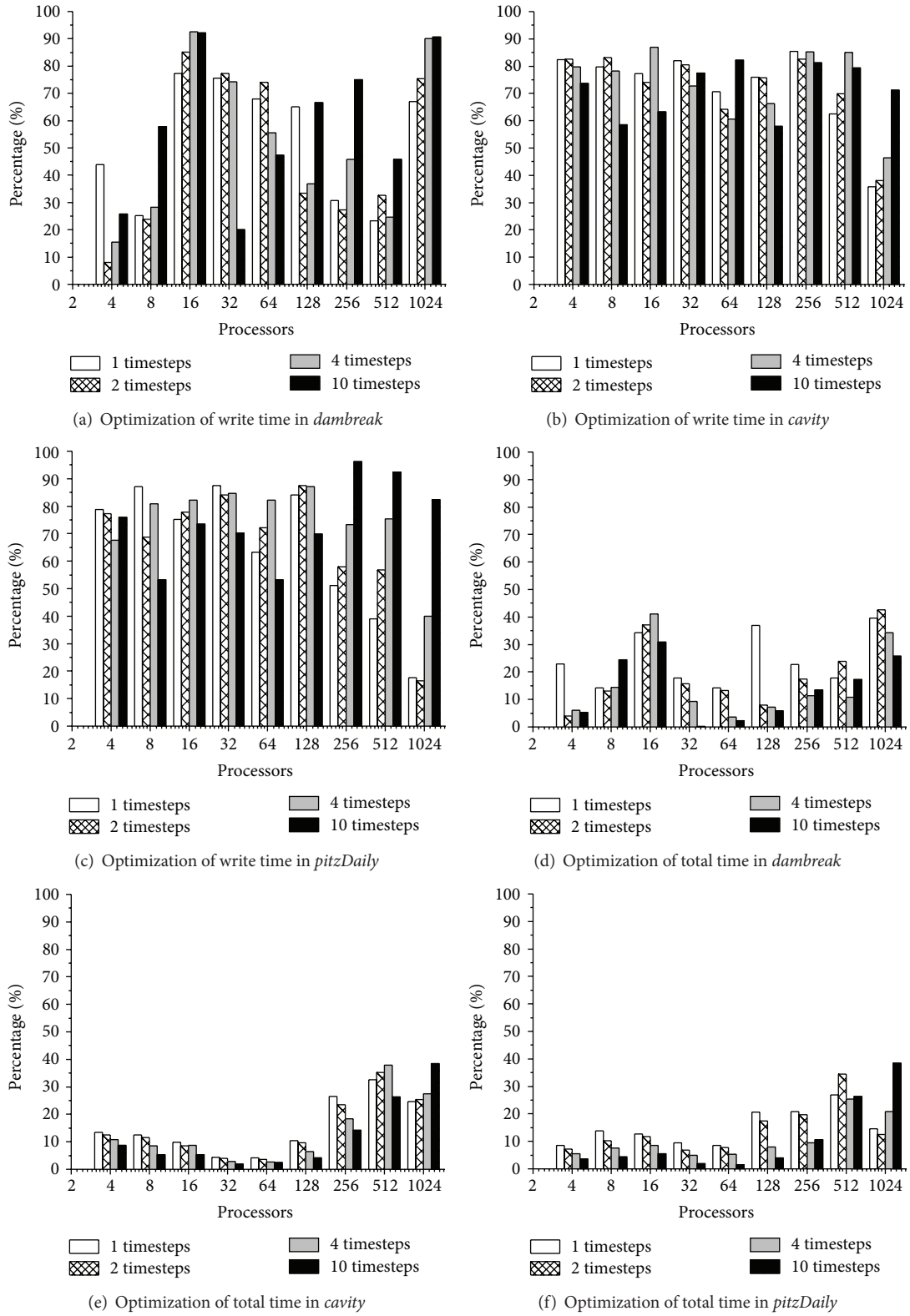
(c) Output data size of *pitzDaily*

Figure 9: CFD application file output volume increases with scale of the number of processors.

execution time of the three benchmarks under the none write interval condition, respectively. It can be found that the performance improves as the number of processors increases, but the scalability is limited at 512 processors.

Figures 11(b), 11(e), and 11(h) show the three benchmarks' file write overhead under different write interval conditions and the number of processors. It can be found that the file write overhead declines with the scale of the number of processors firstly. This phenomenon is mainly due to the fact that the write operations are also performed in parallel with the parallel of processors. On the premise that the I/O channel bandwidth is sufficient, the snapshot output speed scales with the number of processors. But when the number of processors scales to a certain value (64 in our results), the I/O channel bandwidth is not sufficient, and the competition caused by the concurrent write operations makes the write speed decline.

Figures 11(c), 11(f), and 11(i) show how the three benchmarks' proportion of file write overhead in the total simulation time scales with the number of processors. It can be found that although the write file overhead declines with the scale of the number of processors, the proportion of the overhead in the total simulation time ascends. This is due to the fact that the growth of the number of processors makes the file write overhead descend, but its descending speed is far less than that of the calculation time, which makes the proportion of the snapshot output overhead in total simulation time show an ascending tendency. It is notable that the snapshot output overhead portion of *dambreak* at 1024 processors is lower than that at 512 processors in Figure 10(c), which is due to the rise of calculation time caused by this benchmark's negative speedup at 1024 processors.



(a) Optimization of write time in *dambreak*

(b) Optimization of write time in *cavity*

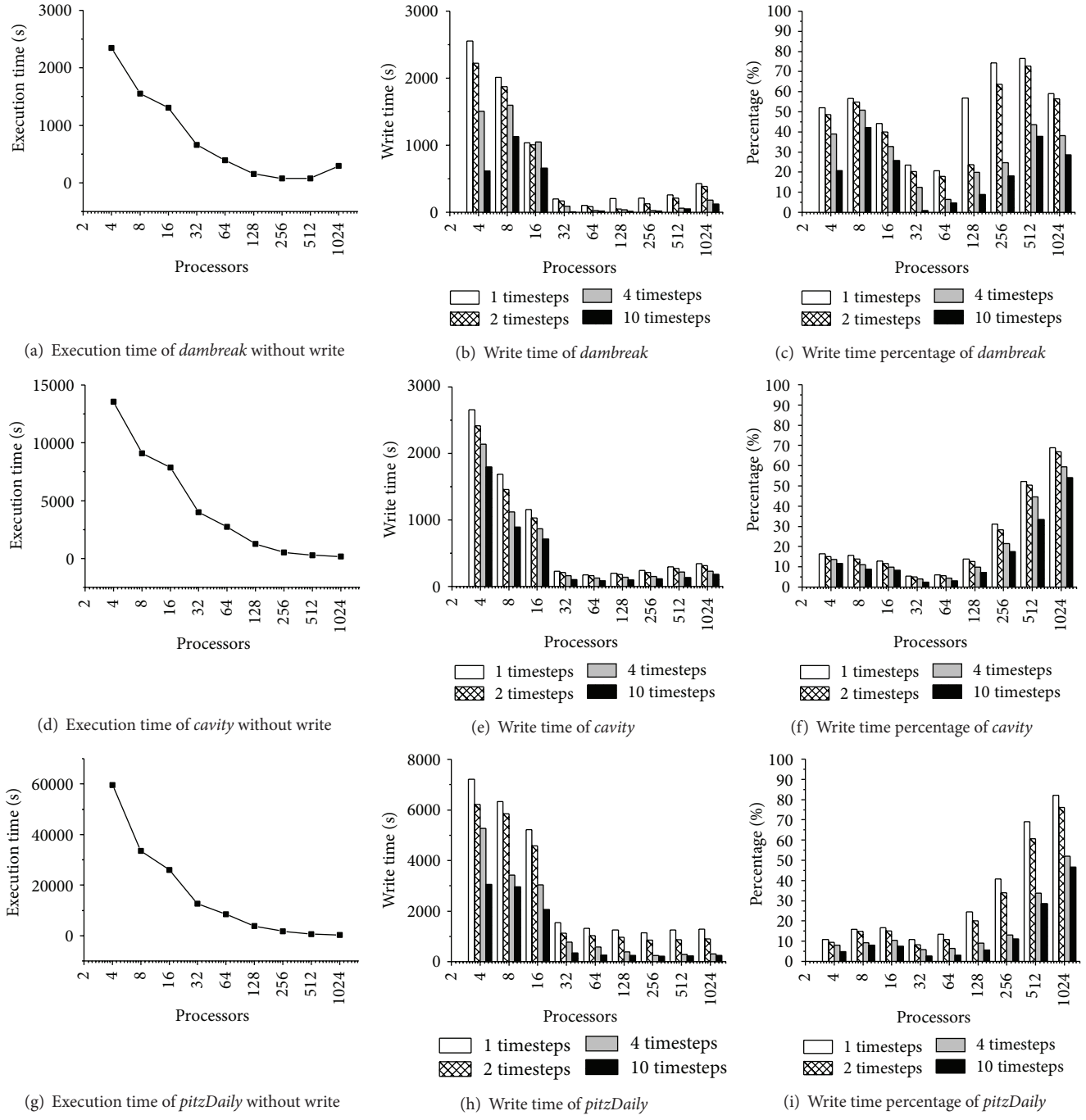
(c) Optimization of write time in *pitzDaily*

(d) Optimization of total time in *dambreak*

(e) Optimization of total time in *cavity*

(f) Optimization of total time in *pitzDaily*

Figure 10: Overhead and proportion of periodical file write.



(a) Execution time of *dambreak* without write
(b) Write time of *dambreak*
(c) Write time percentage of *dambreak*
(d) Execution time of *cavity* without write
(e) Write time of *cavity*
(f) Write time percentage of *cavity*
(g) Execution time of *pitzDaily* without write
(h) Write time of *pitzDaily*
(i) Write time percentage of *pitzDaily*

Figure 11: Optimization effect of AP-IO.

6.4.3. Optimization Effect of AP-IO. Figure 11 shows the optimization effect of *AP-IO*, which is measured with the optimizing effect on the snapshot output overhead and the total simulation time, respectively. As shown in Figures 10(a)–10(c), the optimization ratio of snapshot output overhead can reach more than 50% in average and 90% in the best case.

As shown in Figures 11(d)–11(f), the optimization ratio of the total simulation time can reach to 10% with *AP-IO* on average. However, as the snapshot output overhead is hidden by the calculation operation, this optimization ratio of the total simulation time cannot exceed 50%. The trends in the figure show that *AP-IO* is more effective for the simulations under the intensive write interval condition. The optimization effect of the total simulation time tends to decrease when the number of processors is smaller than 64. However, when the number of processors is more than 64, the optimization effect of the total simulation time tends to increase. This phenomenon agrees with the trends of the file write overhead ratio in the total simulation time shown in Figure 10.

The experimental results show that *AP-IO* can well optimize the periodic snapshot output in CFD simulation and gains better optimization effect in massively parallel simulations.

7. Conclusions

We have proposed an *AP-IO* optimization method based on the features analysis of CFD simulation for the problem of periodic large volume snapshot output in CFD simulation. We design the *AP-IO* framework for CFD application by combining the compiler-directed technology and the application library. We implement *AP-IO* with multithreading technology in the open source CFD software OpenFOAM. Experimental results demonstrate that our *AP-IO* optimization technique can achieve a good optimization effect for the periodical snapshot output in CFD simulation, which can reduce 10% of the total execution time on average.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

We are pleased to acknowledge the National Natural Science Foundation of China (Grant nos. 61221491, 61303071, and 61120106005), Fund (no. 134200026) from Guangzhou Science and Information Technology Bureau, and Fund of State Key Laboratory of High Performance Computing (no. 201303-01).

References

[1] D. Roose and R. Vandriessche, Eds., *Parallel Computers and Parallel Algorithms for CFD: An Introduction*, 1995.

[2] T. Norton and D.-W. Sun, "Computational fluid dynamics (CFD)—an effective and efficient design and analysis tool for the food industry: a review," *Trends in Food Science and Technology*, vol. 17, no. 11, pp. 600–620, 2006.

[3] A. Iserles, "Parallel computational fluid dynamics: implementation and results," *Journal of Fluid Mechanics*, vol. 254, pp. 722–723, 1993.

[4] J. No, R. Thakur, and A. Choudhary, "High-performance scientific data management system," *Journal of Parallel and Distributed Computing*, vol. 63, no. 4, pp. 434–447, 2003.

[5] H. Mickler, A. Knüpfer, M. Kluge, M. S. Müller, and W. E. Nagel, "Trace-based analysis and optimization for the semtex CFD application—hidden remote memory accesses and I/O performance," in *Euro-Par 2008 Workshops—Parallel Processing*, pp. 295–304, Springer, Berlin, Germany, 2009.

[6] G. P. Lim, F. Yang, T. Chu, P. Chen et al., "Ibuprofen suppresses plaque pathology and inflammation in a mouse model for Alzheimer's disease," *The Journal of Neuroscience*, vol. 20, no. 15, pp. 5709–5714, 2000.

[7] X. Ma, M. Winslett, J. Lee, and S. Yu, "Improving MPI-IO output performance with active buffering plus threads," in *Proceedings of the 17th International Symposium on Parallel and Distributed Processing (IPDPS '03)*, pp. 68.2–68.5, IEEE Computer Society, Washington, DC, USA, 2003.

[8] X. Ma, J. Lee, and M. Winslett, "High-level buffering for hiding periodic output cost in scientific simulations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 3, pp. 193–204, 2006.

[9] D. J. Mavriplis and S. Pirzadeh, "Large-scale parallel unstructured mesh computations for 3D high-lift analysis," *Journal of Aircraft*, vol. 36, no. 6, pp. 987–998, 1999.

[10] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, "High-performance parallel implicit CFD," *Parallel Computing*, vol. 27, no. 4, pp. 337–362, 2001.

[11] S. Iyer and P. Druschel, "Anticipatory scheduling: a disk scheduling framework to overcome deceptive idleness in synchronous I/O," *SIGOPS: Operating Systems Review*, vol. 35, no. 5, pp. 117–130, 2001.

[12] A. Acharya, M. Uysal, R. Bennett et al., "Tuning the performance of I/O-intensive parallel applications," in *Proceedings of the 4th Annual Workshop on I/O in Parallel and Distributed Systems (IOPADS '96)*, pp. 15–27, ACM, May 1996.

[13] N. Ali, P. Carns, K. Iskra et al., "Scalable I/O forwarding framework for high-performance computing systems," in *Proceedings of the IEEE International Conference on Cluster Computing and Workshops (CLUSTER '09)*, New Orleans, La, USA, September 2009.

[14] http://www.top500.org/.

[15] J. Dongarra, *Visit to the National University for Defense Technology*, 2013.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 104072, 10 pages
http://dx.doi.org/10.1155/2014/104072

Research Article

Process Correlation Analysis Model for Process Improvement Identification

Su-jin Choi,[1] Dae-Kyoo Kim,[2] and Sooyong Park[1]

[1] *Sogang University, Seoul, Republic of Korea*
[2] *Oakland University, Rochester' MI 48309, USA*

Correspondence should be addressed to Dae-Kyoo Kim; kim2@oakland.edu

Received 28 December 2013; Accepted 25 February 2014; Published 24 March 2014

Academic Editors: S. K. Bhatia and P. Muller

Software process improvement aims at improving the development process of software systems. It is initiated by process assessment identifying strengths and weaknesses and based on the findings, improvement plans are developed. In general, a process reference model (e.g., CMMI) is used throughout the process of software process improvement as the base. CMMI defines a set of process areas involved in software development and what to be carried out in process areas in terms of goals and practices. Process areas and their elements (goals and practices) are often correlated due to the iterative nature of software development process. However, in the current practice, correlations of process elements are often overlooked in the development of an improvement plan, which diminishes the efficiency of the plan. This is mainly attributed to significant efforts and the lack of required expertise. In this paper, we present a process correlation analysis model that helps identify correlations of process elements from the results of process assessment. This model is defined based on CMMI and empirical data of improvement practices. We evaluate the model using industrial data.

1. Introduction

Software process improvement (SPI) is carried out to improve the efficiency of software development process by analyzing involved practices and their relations in consideration of available resources in an organization. SPI is initiated by assessing the current process to identify strengths and weaknesses. Based on findings, improvement plans are developed to reinforce strengths and improve weaknesses.

In general, a reference model is used throughout SPI as the base. A widely used model is Capability Maturity Model Integration (CMMI) [1] which has shown its impact on product quality, development cost, and time-to-market across the industry [2–4]. CMMI provides a structured process assessment framework defined upon a set of process areas (PAs). 22 PAs (e.g., project planning and requirement definition) are defined each describing specific goals to be achieved and specific practices to be carried out. PAs and their components (goals and practices) are highly correlated, which reflects the iterative nature of software development process [5]. These correlations capture dependencies among PAs and components which should be considered in improvement planning [6, 7].

However, the current practice focuses only on individual PAs and often overlooks correlations of PAs, which diminishes the efficiency of improvement plans. This is mainly attributed to significant efforts and the lack of required expertise. There exist a few studies on identifying correlations of improvement items (e.g., CMMI practices and improvement agendas) and improvement planning [7–10]. The existing work, however, largely relies on analyst's expertise and manual work which make it difficult for the less experienced to practice. Some works (e.g., [7, 8, 10]) propose manual approaches for reviewing and relating process items and some others (e.g., [9]) present an expert-friendly template for describing improvement agendas.

In this paper, we present a process correlation analysis model for identifying correlations of PAs, goals, and practices based on CMMI and empirical data collected from SPI projects where the analysis of findings' correlations have been done. CMMI is used to infer correlations of PAs and their components and the inferred correlations are affirmed

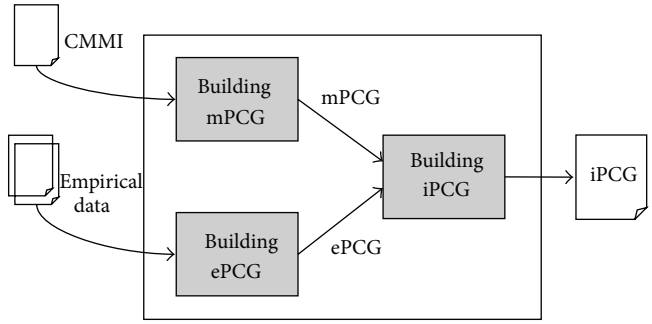


Figure 1: Process correlation model.

and complemented using empirical data. The model takes as input the findings of process assessment and identifies their correlations by analyzing the findings to corresponding practices correlated in CMMI and empirical data. The model then produces a graph representing correlations of findings. We evaluate the model in terms of precision, recall, F-measure, and accuracy using five different industrial SPI projects. We also demonstrate tool support for the presented model. In our previous work, we presented ReMo, a model for developing improvement recommendations which use manual correlation analysis. This work complements the previous work by providing a systematic way of identifying process correlations.

The paper is structured as follows. Section 2 gives an overview of related work, Section 3 details the proposed process correlation analysis model and its supporting tool, Section 4 presents the evaluation results using industrial data, and Section 5 concludes the paper with future work.

2. Related Work

There is some work on identifying improvements using a certain type of relationships. Gorscheck and Wohlin [8] propose *DAIIPS*, a method for packaging improvement issues by prioritizing them based on analysis of their dependencies. This method is designed for small organizations where resources for addressing improvement issues are limited. The degree of dependency between improvement issues is decided by vote in a workshop. That is, the decision on dependency is greatly influenced by the level of participants' expertise. Moreover, they do not provide criteria for the qualification of workshop attendees and guidelines for making dependency decisions. Such a manual process is time consuming and requires significant efforts.

Calvo-Manzano Villalón et al. [9] present an improvement specification template called *Action Package* for describing organizational, technical, and management aspects of process improvements. The template consists of twelve items including policy, training, and metrics. However, the template is designed for experts, providing little details as to how the items should be filled out. This makes it difficult for the less experienced to use the template.

Sun and Liu [10] present a CMMI-based method for relating improvement issues to CMMI practices using the Quality Function Deployment technique [11]. Similar to Gorscheck and Wolin's work, they prioritize improvement issues by the number of relations to practices. The higher the number of relations, the higher the priority. However, the method does not describe how identified relations may be used.

Chen et al. [7] present a practice dependency model for CMMI using six process areas at level 2. Dependencies between practices are identified via the flow of work products between practices based on a textual analysis of the CMMI specification. Their work is similar to our work in that our work also uses CMMI. However, the specification of CMMI involves many ambiguities, which limits the extent to which CMMI alone may provide information about correlations. To address this, we make use of empirical field data from industry in addition to CMMI.

3. Correlation Analysis Model

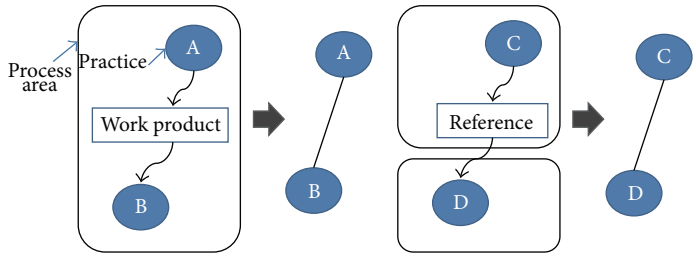
In this section, we describe a process correlation analysis model that identifies correlations from a given assessment finding set. The model is built upon CMMI and empirical field data. CMMI is used as a base for identifying an initial set of common correlations of practices from its descriptions on PAs and their components (e.g., goals and practices). Identified correlations are represented as a graph. The graph is referred to as *mPCG* which denotes a process correlation graph (PCG) for the considered model (CMMI). However, the description in CMMI involves ambiguities and inconsistencies due to its general nature. To address this, we use empirical data collected from various SPI projects in addition to CMMI. Correlations are identified from empirical data and represented as a graph referred to as *ePCG*. *mPCG* and *ePCG* are then combined to produce an integrated graph *iPCG*. Figure 1 illustrates the process of the model. Henceforth, we use terms correlation and relation interchangeably.

3.1. Building mPCG. We use CMMI as the base for identifying practice correlations. CMMI describes a PA in terms of *Goals*, *Practices*, and *Subpractices*. Goals may be generic or specific. A specific goal is achieved by a set of specific practices producing certain work products. A specific practice may consist of a set of subpractices. The specific goals, practices, and subpractices of a PA may reference the components of

Table 1: Process area examples.

Process area	Goal	Practice	Description	Reference information
Requirement management (REQM)	[a]SG 1		Manage requirements	**Refer to** the *Project Monitoring and Control* process area for more information about managing corrective action to closure.
		[b]SP 1.2	Obtain commitment to requirements	**Refer to** the *Project Monitoring and Control* process area for more information about monitoring commitments.
		SP 1.4	Maintain bidirectional traceability of requirements	*Not defined*
Project planning (PP)	SG 1		Establish estimates	*Not defined*
		SP 1.1	Estimate the scope of the project	**Refer to** the *Supplier Agreement Management* process area. . .
	SG 2		Develop a project plan	*Not defined*
		SP 2.2	Identify project risks.	**Refer to** the *Monitor Project Risks* specific practice in the *Project Planning* process area for more information about risk monitoring activities.
Project monitoring and control (PMC)	SG 1		Monitor the project against the plan	*Not defined*
		SP 1.2	Monitor commitments	*Not defined*
		SP 1.3	Monitor project risks	**Refer to** the *Project Planning* process area for more information about identifying project risks.
	SG 2		Manage corrective action to closure	*Not defined*
		SP 2.1	Analyze issues	**Refer to** the *Establish the Budget and Schedule* specific practice in the *Project Planning* process area for more information about corrective action criteria.

[a]Specific goal, [b]specific practice.



(a) IPC (b) EPC

Figure 2: Identifying practice correlations in CMMI.

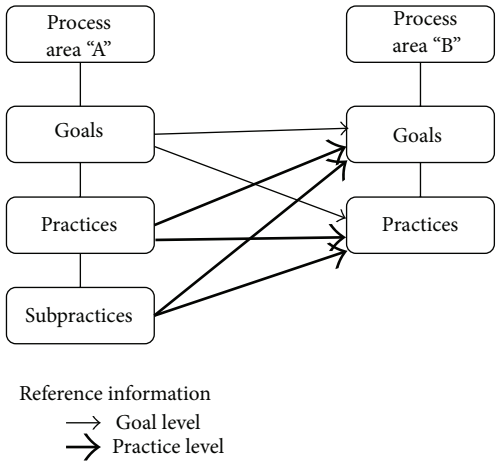


Figure 3: Structure of process areas in CMMI.

another PA. Generic goals are shared by PAs and, thus, there is no reference for generic goals.

Given the PA descriptions in CMMI, we look for information about internal practice correlations (IPCs) and external practice correlations (EPCs). IPCs exist within the same process area while EPCs cut across process areas. To identify IPCs, we make use of the description of practices and subpractices and their related work products. Specifically, we focus on relationships of input and output work products among practices. Two internal practices are considered as internally correlated if one practice has output work products that are used as input work products of the other. Figure 2(a) shows an IPC. EPCs are identified based on the description of related process areas and external references. Two external practices are considered as externally correlated if one practice refers to the other, which is shown in Figure 2(b).

Goals, practices, and subpractices are defined at different levels as shown in Figure 3. The goal level is the highest followed by the practice level and then the subpractice level. We consider two components (e.g., goals) as correlated if they reference each other. A component referencing another component at a lower level is considered as related to that specific component only. On the other hand, a component referencing another component at a higher level is considered as related to all the subcomponents of the component. Two PAs are considered as correlated if a component of one PA refers to a component of the other or one PA refers to the other PA in its description.

Table 1 shows examples of PAs and a subset of their components defined in CMMI. The table describes three PAs—*Requirement Management (REQM)*, *Project Monitoring and Control (PMC)*, and *Project Planning (PP)*. *REQM* has one specific goal (*SG 1*) with two specific practices (*SP 1.1* and *SP 1.2*). Other PAs can be explained similarly. A component may be accompanied with reference information. For instance, *SG 1* in *REQM* has the following reference description: "*Refer to the* Project Monitoring and Control *process area for more information about* managing corrective action to closure." From this description, one can obviously infer a relation of *REQM SG 1* to the *SG 2* goal of *PMC*, which is an example of a goal-level reference. Similarly, from the reference description of *REQM SP 1.2*, it can be easily inferred that *REQM SP 1.2* is related to the *SP 1.2* practice in *PMC*.

Based on elicited correlations of PAs and their components, a mPCG is built. A mPCG is a nondirected graph capturing correlations of practices where a node represents a practice and an edge represents a correlation. An edge between nodes p_m and p_n has a weight denoted by $W_{m(p_m,p_n)}$. All edges in a mPCG have their weight 1 denoting the existence of a correlation. Correlations may be found within the same PA or across PAs. Figure 4 shows an example of a mPCG.

Note that CMMI descriptions are not always explicit. For instance, subpractices are described mostly about output work products while having little description on input products. Goal-level references across PAs are described abstractly providing little information on how they influence related practices. CMMI descriptions on process areas also involve ambiguities.

3.2. Building ePCG. To address the above, we make use of empirical data collected from a set of industrial SPI projects in addition to CMMI. The data is postproject data including findings and their correlations are already analyzed and used in completed projects. Postproject findings are often tagged with relevant CMMI practices. Accordingly, we assume that they are all tagged in this work.

Each project is analyzed to identify correlated practices which are captured in a PCG. PCGs of all the considered projects are combined to produce an ePCG. The resulting ePCG is then merged with the mPCG of CMMI. However,

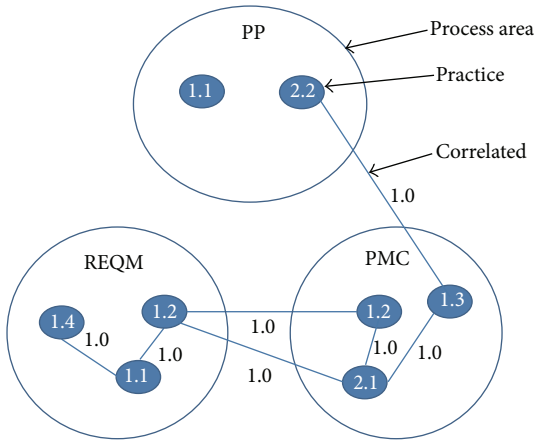


Figure 4: mPCG example.

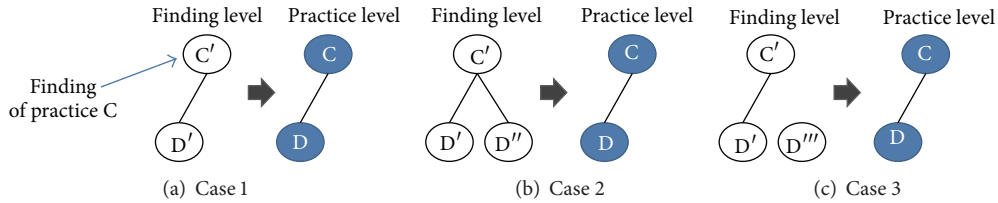


Figure 5: Correspondence between findings and practices.

unlike CMMI where correlations are described for practices, empirical data are described for findings which are implementations of practices. This is because an SPI project is an implementation of CMMI. Given that, findings in empirical data should be abstracted to practices to align the level of abstraction of ePCGs with the mPCG of CMMI. Abstraction is carried out as follows.

(R1) For each finding, identify the corresponding practice in CMMI.

(R2) Two practices are considered as correlated if a finding of the practice is related to a finding of the other practice. This is illustrated in Figure 5(a). The same holds regardless the number of instances (findings) of a practice. Figures 5(b) and 5(c) show the cases where a practice has multiple instances with different relations. Both cases result in the same abstraction as Figure 5(a) per R3.

We demonstrate abstraction using the findings in Table 2. The table shows four findings *REQM-W-01*, *PP-W-01*, *PP-W-02*, and *PP-S-01* where *REQM-W-01* is an instance of the practice *REQM SP 1.4* in Table 1 and *PP-W-01*, *PP-W-02*, and *PP-S-01* are instances of *PP SP 1.1*. From the description of the findings, *REQM-W-01* and *PP-W-01* are found related since requirement traceability needs to be maintained for development tasks and work products defined in work breakdown structure. Similarly, *REQM-W-01* and *PP-S-01* are found related since *PP-S-01* is a strength practice of defining development tasks and work products. Given this relation, the corresponding practices *REQM SP 1.4* and *PP SP 1.1* to these findings are considered also as related. On the other hand, *REQM-W-01* and *PP-W-02* are found not related because COTS products are already made. Thus, requirements traceability to development tasks and work products is not necessary. However, their corresponding practices are considered as related as they have already been so for other pairs.

PCGs of SPI projects are merged by combining nodes and edges to create an ePCG for the considered project set. Each edge is weighted by the number of projects having the edge identified in their PCG. Let $|P_{p1,p2}|$ be the number of the projects whose PCG has practices $p1$ and $p2$ correlated and $|P_{\widehat{p1,p2}}|$ the number of the projects whose PCG has $p1$ and $p2$ identified as correlated. Then, the weight $W_{e(p_m,p_n)}$ of the edge between p_m and p_n is defined as follows:

$$W_{e(p_m,p_n)} = \frac{\left|P_{\widehat{(p_m,p_n)}}\right|}{\left|P_{(p_m,p_n)}\right|}, \quad \left(0 \le W_{e(p_m,p_n)} \le 1\right). \tag{1}$$

As an example, consider Figure 6. In the figure, there are three projects *ProjetX*, *ProjectY*, and *ProjectZ* where the PCG of *ProjetX* has one related practice pair ((*PP 1.1, REQM 1.4*)), the PCG of *ProjectY* has two related pairs ((*PP 1.1, REQM 1.4*), (*PP 1.1, REQM 1.1*)), and the PCG of *ProjectZ* has one related pair ((*REQM 1.4, REQM 1.1*)). The practices are the same as those in Figure 4. PCGs of the projects are merged into an ePCG by adding all the nodes (*PP 1.1*, *REQM 1.4*, *REQM 1.1*) and their relations involved in the PCGs. The weight of the edge between *PP 1.1* and *REQM 1.1* is measured at 0.5 (1/2) as the number of the projects that involve *PP 1.1* and *REQM*

Table 2: Assessment findings and corresponding practices.

Finding ID	Finding description	Practice ID
REQM-W-01	It is difficult to trace defined requirements to development tasks and work products.	REQM SP 1.4
PP-W-01	Work break down structure are developed in high level (e.g., milestone) without detailed tasks and work product.	PP SP 1.1
PP-W-02	Product components to be purchased as COTS (commercial-off-the-shelf) are not documented.	PP SP 1.1
PP-S-01	In some projects, WBS with detail description on tasks and work products is developed with the support of Quality Assurance team.	PP SP 1.1

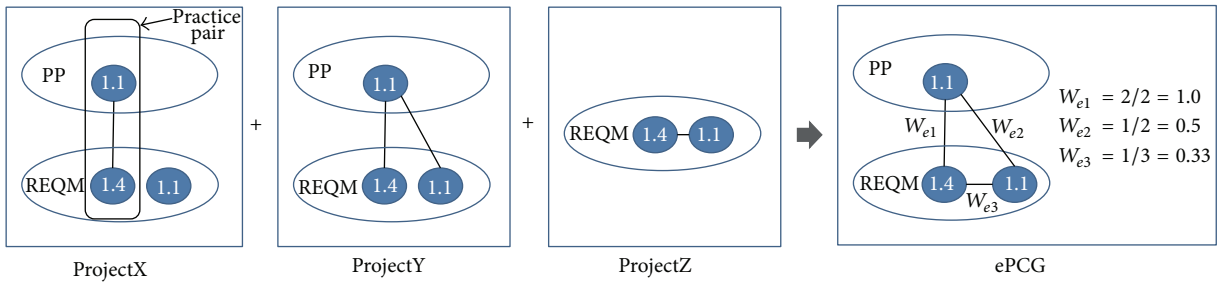


Figure 6: Building an ePCG.

1.4 is two (ProjectX, ProjectY) and the number of the projects that have *PP 1.1* and *REQM 1.1* identified as correlated is one (ProjectY). Weight for other edges can be measured similarly.

3.3. Building iPCG by Combining mPCG and ePCG. The mPCG resulting from Section 3.1 and the ePCG from Section 3.1 are merged to produce an integrated graph iPCG. Figure 7 shows an example which combines the mPCG in Figure 4 and the ePCG in Figure 6.

The weight $W_{i(p_m,p_n)}$ of an edge between p_m and p_n in the iPCG is computed as follows:

$$W_{i(p_m,p_n)} = \frac{W_{m(p_m,p_n)} + W_{e(p_m,p_n)}}{2}, \quad \left(0 \le W_{i(p_m,p_n)} \le 1\right). \tag{2}$$

A threshold is set for weight and any edge having its weight lower than the threshold is considered as not related. We use 0.25 for the threshold in this work. This means that a practice pair in an iPCG is considered as correlated if it is identified in CMMI ($W_m = 1$) or in the half or more of the empirical projects ($W_e \ge 0.5$).

Figure 8 shows the application process of the presented model. The findings identified in the process assessment of an SPI project are abstracted to practices using the abstraction rules in Section 3.2. The resulting practices are then input to the iPCG to produce a PCG of the project described in terms of practices. The practices in the PCG are concretized back to findings using the same mapping used as in abstracting the findings to practices at the beginning.

Figure 9 shows an example of a resulting correlation matrix. The matrix is symmetric having the same set of findings in column and row and the values represent weights. In the figure, the box in bold line shows that the weight of the (*REQM-W-03, CM-W-01*) pair is measured as 0.50 which indicates that the practices in the pair are correlated in either CMMI ($W_m = 1$) or the empirical data ($W_e = 1$) used in the model. Those pairs that have zero weight are not related. The resulting matrix is suggestive. That is, the process analyst may modify the matrix at his discretion. The matrix can be also used for identifying more significant findings by prioritizing them by the number of correlations or accumulated weights (i.e., adding all the weights by column).

3.4. Supporting Tool. We have developed a prototype tool supporting the presented model. Figure 10 shows the architecture of the tool. It consists of two components—a correlation analysis engine and a PCG viewer. The correlation analysis engine is Excel-based consisting of (1) a knowledge base containing CMMI-based practice correlations and project-based practice correlations, (2) a PCG generator responsible for building PCGs, and (3) a PCG executor applying an iPCG and generating the output correlations in an Excel file. The PCG viewer displays the resulting PCG using the *yED Graph Viewer*, an open source application for visualizing object connections [12]. The viewer takes an input file in various formats including the Excel files generated by the PCG executor. Figure 11 shows a screenshot of the PCG viewer. Nodes are grouped to denote different PAs of relevance and correlation weights are displayed with mouse-over on edges.

4. Evaluation

We evaluate the presented model in terms of recall, precision, F-measure, and accuracy [13] by comparing the resulting correlations to manually identified correlations by experts. Recall is measured by the number of correlations produced by the

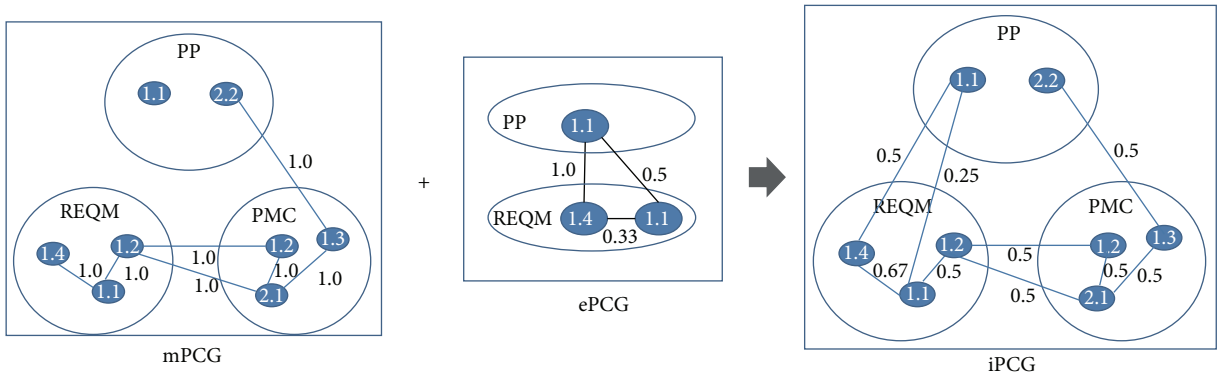


Figure 7: Building an iPCG.

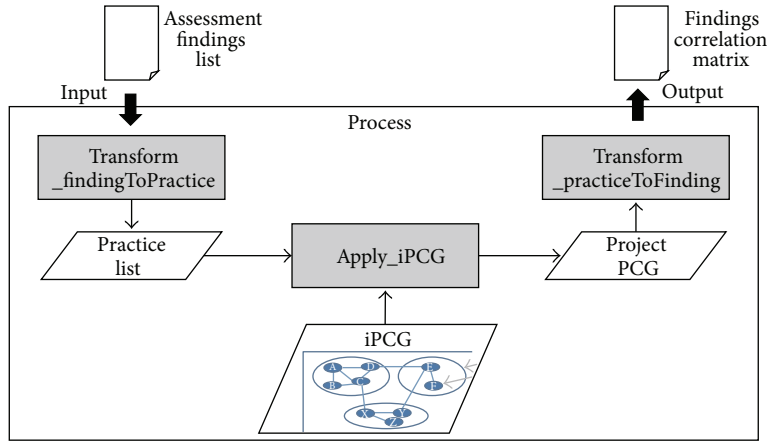


Figure 8: Application process of the correlation model.

A set of correlated findings for a given finding "REQM-W-03"

Finding		CM-W-01	MA-W-01	REQM-W-02	REQM-W-03	VER-W-02
	Practice	CM_SP1.3	MA_SP1.1	REQM_SP1.3	REQM_SP1.4	VER_SP1.1
CM-W-01	CM_SP1.3	0.00	0.50	0.50	0.50	0.25
MA-W-01	MA_SP1.1	0.50	0.00	0.33	0.75	0.00
REQM-W-02	REQM_SP1.3	0.50	0.33	0.00	0.88	0.00
REQM-W-03	REQM_SP1.4	0.50	0.75	0.88	0.00	0.50
VER-W-02	VER_SP1.1	0.25	0.00	0.00	0.50	0.00

Figure 9: Example of resulting finding correlation.

model over the number of manually identified correlations. Similarly, precision is measured by the number of manually produced correlations over the number of correlations produced by the model. The accuracy is measured by the number of correctly identified correlations to the total number of practice pairs.

We use five industry SPI projects to evaluate the model each from a different company. Four projects are used for building the iPCG in the model and the remaining project is used as the target project to which the model is applied. The target project is rotated in the five projects so that the model can be applied to all the five projects. As the target project is rotated, the project used as the target project is excluded from the project set used to build the iPCG in the model. In this way, the iPCG is not biased to the target project. The five projects are all CMMI-based targeting the maturity level from 2 to 4.

Table 3 shows an overview of the five projects used in the evaluation. In the table, *CompA* and *CompB* aim at level 3, *CompC* and *CompD* aim at level 2, and *CompE* aims at level 4. Given that, it can be observed in the table that the four PAs (*requirement management*, *project planning*, *measurement and analysis*, *and configuration management process*) in *CompC* and *CompD* are from level 2 and the two additional PAs (*technical solution* and *verification process*) in *CompA*, *CompB*, and *CompE* are from level 3. The table shows that for *CompA*, 20 findings are found in six PAs and they are all related to 18 practices. 49 practice correlations are found out of total 153 practice pairs in the six PAs. Other projects can be explained similarly. The six PAs involve total 50 practices in CMMI of which 40 practices (accounting for 80%) are addressed in the five projects.

Table 4 shows measured data from the evaluation. *True Positive* denotes the number of correlations that are identified by the presented model as related and also evaluated as

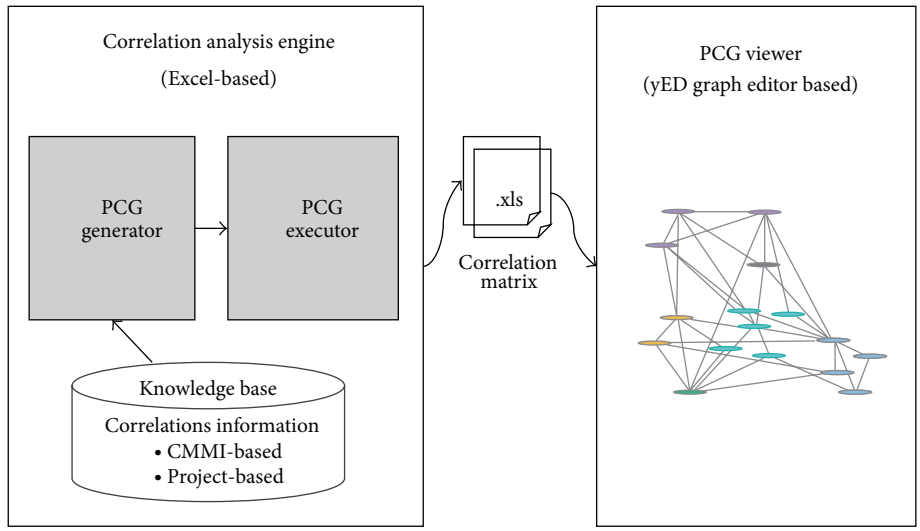


Figure 10: Prototype tool components.

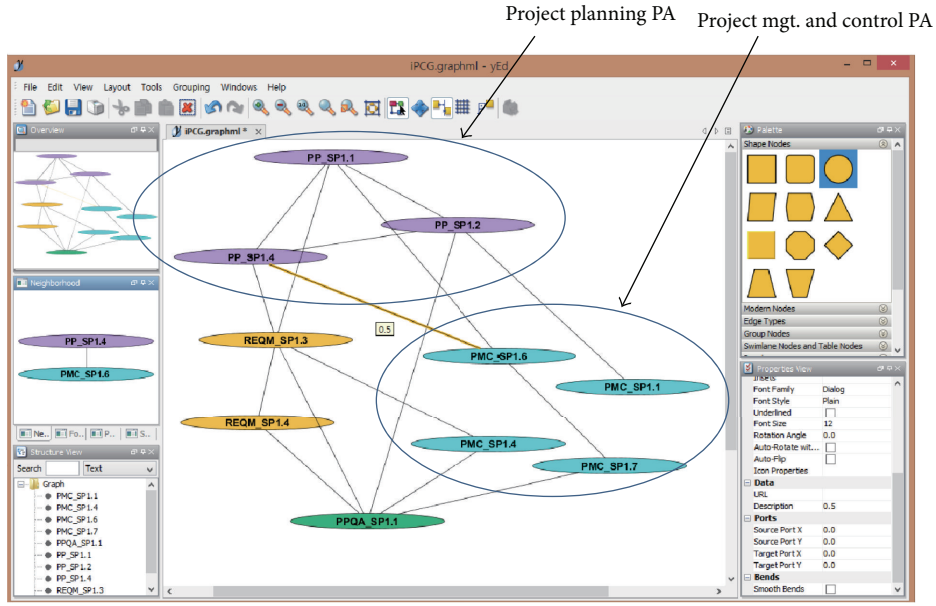


Figure 11: PCG viewer.

Table 3: Summary on collected data.

Company	Number of process areas[1]	Number of Findings	Number of practices[1]	Number of practice pairs	
				Total	Correlated
CompA	6	20	18	153	49
CompB	6	21	19	171	32
CompC	4	12	15	105	52
CompD	4	11	10	45	33
CompE	6	22	20	190	49
Total	6	86	40	664	215

[1]Total means number of distinct process areas or practices.

Table 4: Measured data.

Company	True positive	True negative	False positive	False negative
CompA	25	76	28	24
CompB	11	105	34	21
CompC	20	44	9	32
CompD	23	8	4	10
CompE	30	121	22	17
Sum	109	354	97	104

Table 5: Evaluation results.

Company	Precision	Recall	*F*-measure	Accuracy
CompA	0.47	0.51	0.49	0.66
CompB	0.24	0.34	0.28	0.68
CompC	0.69	0.38	0.49	0.61
CompD	0.85	0.70	0.77	0.69
CompE	0.58	0.64	0.61	0.79
Average	**0.57**	**0.51**	**0.54**	**0.69**
Standard Deviation	0.23	0.16	0.18	0.07

related by experts. *True Negative* represents the number of correlations that are identified by the model as not related and also evaluated by experts as not related. *False Positive* shows the number of the correlations that are identified by the model, but are evaluated as irrelevant by experts. *False Negative* is the number of the correlations that are evaluated as related by experts, but not identified by the presented model. The table shows *CompB* having a low true positive. This is due to the lack of thoroughness in the project which was conducted hastily with nonexperts involved. *CompD* shows a relatively higher true positive and a low false positive. This is because there are only 10 practices involved in the project. Thus, it was easier for experts to identify correlations.

Based on Table 4, precision, recall, f-measure, and accuracy of the model are measured for the five projects. Precision is measured by *True Positive*/(*True Positive* + *False Positive*) while recall is measured by *True Positive*/(*True Positive* + *False Negative*); Table 5 shows the results. The table shows that the average precision is 0.57 which implies that about 60% of the correlations identified by the presented model are also identified as correlated in the five projects. The average of recall is measured as 0.51 which means that about the half of the manually identified correlations are also identified by the model. *F*-measure, which is the harmonic mean of precision and recall, is measured at 0.54.

A higher precision implies that experts have more correlations identified manually that match with the correlations identified by the model. On the other hand, a higher recall implies that the model identified more correlations matching with manually identified correlations. *CompB* shows the lowest precision and recall. This is because there is only a small number of manually identified correlations to the total number of practice pairs (which is only 19% while the average is 46%). On the other hand, *CompD* has 73% of the total number of practice pairs identified as correlated, which contributes to its highest precision and recall. This observation was expected as precision and recall depend on the number of manually identified correlations.

Accuracy is measured by (*True Positive* + *True Negative*)/(*True Positive* + *True Negative* + *False Positive* + *False Negative*). Accuracy is measured at 0.69 in average, which means that about 70% of the correlations identified by the presented model are considered as correct in the five projects. The standard deviation of accuracy is measured at 0.07 which is relatively low compared to precision and recall. This means that the presented model is consistent to the considered projects.

5. Conclusion

We have presented a process correlation analysis model for identifying correlations of findings. A major advantage of the model is the use of empirical data which complements the CMMI specification being ambiguous and abstract. The evaluation of the model shows 0.51 for recall and 0.69 for accuracy which demonstrates the potential of the model. It is noteworthy to mention that the model is developed in response to the need of techniques for identifying finding correlations from the field. We shall continue to improve and evaluate the model by applying it to more case studies. As more case studies are conducted, we shall extend the evaluation to look into the efficiency aspect of the model over manual analysis.

We also plan to investigate an effective way of packaging findings based on identified correlations to build improvement plans. Using correlation information, a finding that has more correlations can be found and as such a finding can lend itself as a seed for forming a package. The package may include findings that are directly correlated by the seed. The resulting package then serves as an early version of an improvement plan and may evolve throughout a series of refinement activities.

The model can be also used in the case where empirical data is not available. In such a case, one may start analyzing correlations using only mPCG and then use the empirical data from the current project as it is completed. The data then can be used as input to build an ePCG for the next project. We expect that the model is to be more accurate as more empirical data is used.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the Korean Institute of Energy Technology Evaluation and Planning (KETEP) under the international collaborative R&D program (20118530020020) and Next-Generation Information Computing Development

Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (2012M3C4A7033348).

References

[1] CMMI Product Team, "CMMI for Development, Version 1.3," Tech. Rep. CMU/SEI-2010-TR-033, Software Engineering Institute, Carnegie Mellon University, 2010.

[2] M. Staples and M. Niazi, "Systematic review of organizational motivations for adopting CMM-based SPI," *Information and Software Technology*, vol. 50, no. 7-8, pp. 605–620, 2008.

[3] M. Swinarski, D. H. Parente, and R. Kishore, "Do small IT firms benefit from higher process capability?" *Communications of the ACM*, vol. 55, no. 7, pp. 129–134, 2012.

[4] M. Unterkalmsteiner, T. Gorschek, A. K. M. M. Islam, C. K. Cheng, R. B. Permadi, and R. Feldt, "Evaluation and measurement of software process improvement—a systematic literature review," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 398–424, 2012.

[5] D. Damian and J. Chisan, "An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management," *IEEE Transactions on Software Engineering*, vol. 32, no. 7, pp. 433–453, 2006.

[6] S. Choi, D. K. Kim, and S. Park, "ReMo: a recommendation model for software process improvement," in *Proceedings of the International Conference on Software and Systems Process (ICSSP '12)*, pp. 135–139, 2012.

[7] X. Chen, M. Staples, and P. Bannerman, "Analysis of dependencies between specific practices in CMMI maturity level 2," in *Software Process Improvement*, vol. 16 of *Communications in Computer and Information Science*, pp. 94–105, 2008.

[8] T. Gorschek and C. Wohlin, "Packaging software process improvement issues: a method and a case study," *Software: Practice and Experience*, vol. 34, no. 14, pp. 1311–1344, 2004.

[9] J. A. Calvo-Manzano Villalón, G. C. Agustín, T. S. F. Gilabert, A. de Amescua Seco, and L. G. Sánchez, "Experiences in the application of software process improvement in SMES," *Software Quality Control*, vol. 10, no. 3, pp. 261–273, 2002.

[10] Y. Sun and X. Liu, "Business-oriented software process improvement based on CMMI using QFD," *Information and Software Technology*, vol. 52, no. 1, pp. 79–91, 2010.

[11] Y. Akao, Ed., *Quality Function Deployment: Integrating Customer Requirements Into Product Design*, Productivity Press, 1990.

[12] yED Graph Editor Homepage, http://www.yworks.com/en/index.html.

[13] M. Junker, A. Dengel, and R. Hoch, "On the evaluation of document analysis components by recall, precision, and accuracy," in *Proceedings of the 5th International Conference on Document Analysis and Recognition (ICDAR '99)*, pp. 713–716, 1999.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 156790, 11 pages
http://dx.doi.org/10.1155/2014/156790

Review Article

A Survey of Artificial Immune System Based Intrusion Detection

Hua Yang,[1,2] Tao Li,[1] Xinlei Hu,[1] Feng Wang,[1] and Yang Zou[1]

[1] *College of Computer Science, Sichuan University, Chengdu 610064, China*
[2] *Computer School, China West Normal University, Nanchong 637002, China*

Correspondence should be addressed to Hua Yang; hyang.yh@gmail.com and Tao Li; litao@scu.edu.cn

Received 28 November 2013; Accepted 30 December 2013; Published 23 March 2014

Academic Editors: K. K. Mishra and A. K. Misra

In the area of computer security, Intrusion Detection (ID) is a mechanism that attempts to discover abnormal access to computers by analyzing various interactions. There is a lot of literature about ID, but this study only surveys the approaches based on Artificial Immune System (AIS). The use of AIS in ID is an appealing concept in current techniques. This paper summarizes AIS based ID methods from a new view point; moreover, a framework is proposed for the design of AIS based ID Systems (IDSs). This framework is analyzed and discussed based on three core aspects: antibody/antigen encoding, generation algorithm, and evolution mode. Then we collate the commonly used algorithms, their implementation characteristics, and the development of IDSs into this framework. Finally, some of the future challenges in this area are also highlighted.

1. Introduction

Computer security refers to information security as applied to computers and networks, which is an important problem in the world today. This field covers all the processes and mechanisms by which computer based equipment, information and services are protected from unintended or unauthorized access, change, or destruction. With the development of the networks, computer security is facing enormous challenges. To solve this problem, Intrusion Detection Systems (IDSs) have become an indispensable component for detecting abnormal behaviors before they cause widespread damage.

How can we effectively detect all the unauthorized use, misuse, and abuse of computer system? Many researchers have made efforts. Anderson [1] first pointed out the computer Intrusion Detection (ID) problem in 1972. Then he proposed the concept of IDS in 1980 [2] which was one of the earliest works on ID. Between 1984 and 1987, Denning first proposed an IDS model [3]. This prototype was named as the Intrusion Detection Expert System (IDES). 1990 is a watershed in IDS development history. This year, Heberlein developed the Network Security Monitor (NSM) [4]. Then IDS was officially formed as two camps: network based IDS (NIDS) and host based IDS (HIDS). Now, ID is a hot topic in the area of computer security and many prototypes have been developed using different approaches. This paper will discuss various ID methods using Artificial Immune System (AIS).

Computer science has a great tradition of stealing nature's good ideas. The brain has inspired the neural network model which is the basis of many attempts to develop artificial intelligence. The HIS (Human Immune System) is made up of interdependent cell types which protect the body from various harmful pathogenic infections, such as bacteria, viruses, and parasites. It does this largely without prior knowledge of the structure of these pathogens (a more detailed introduction of the HIS can be found in [5, 6]). The goal of HIS is typically referred to as the differentiation of self (molecules and cells that belong to the host organisms) from potentially harmful nonself (molecules and cells that are recognized as foreign molecules). This property has in recent years made it the focus of computer science and ID communities. Hence, applying theoretical immunology and observed immune functions to IDS has gradually developed into a research field called AIS [7]. These years, researchers have made considerable contributions to the development of AIS. A large number of AISs have been built for a wide range of applications including fraud detection [8], optimization [9], machine learning [10], robotics [11], and computer security [12]. Most reviews about AIS based IDS are summarized from the view point of used algorithms or system development.

There are so many methods of AIS, which one on earth should we use? Is there any law to follow? This paper will provide a general framework to the area of AIS based IDS and discussion from three aspects: antibody/antigen encoding, generation algorithm, and evolution mode.

In the following sections, we briefly introduce the areas of IDS and AIS. Section 2 mainly gives the framework for the design of AIS based IDS and introduces the background of AIS. From Section 3 to Section 5, we provide a detailed discussion about the conjunction of IDS and AIS in view of our framework, respectively, antibody/antigen encoding, generation algorithm, and evolution mode. Finally, we present our conclusion and discuss future work of investigation.

2. The Framework for the Design of AIS Based IDS

The purpose of the IDS is not only preventing the attack to be happened but also reporting all the abnormal behaviors of the system. In order to design a successful AIS based IDS, the first thing that should be considered is the problem presentation of the system in ID domain and then the combination of AIS methods to IDS. Here, we first introduce AIS briefly. Then, we present the framework design of AIS based IDS.

2.1. Background of Artificial Immune System. AIS research began in the mid-1980s with Farmer, Packard, and Perelson's study [13]. Their study suggested that computer science might borrow from the immune system. The great formative AIS researches for computer security were those that proposed the immune system as an analogy for IDSs. One of the classical theories is Negative Selection (NS) [14] which is abstract model of biological NS. In this theory, the detector model generated in censoring phase is intended to monitor the self-state and detect whether or not self has been changed. Then they estimated the method feasibility as a change-detection method on the problem of computer virus detection. Based on the above analysis, Kephart successfully applied immune mechanisms to antivirus problems [15]. With the development of HIS principle, Negative Selection Algorithm (NSA) [14], Clonal Selection Algorithm (CSA) [16], Immune Network Algorithm (INA) [12], and Danger Theory Algorithm (DTA) [17] become the most representative algorithms in the AIS theory. Aickelin et al. [18] provided a detailed overview of immune system approaches to ID. He gave a review of methodologies, algorithms, and research groups in the application of AISs to ID. Kim et al. summarized six immune features that are desirable in an effective IDS [19]. They provided an overview in the view of the research development history.

2.2. The Framework for the Design of AIS Based IDS. Although there are many papers that have summarized the works for this topic, these reviews just divided the current methods into different groups and cannot provide enough guidance information for the design of the AIS based ID methods. In this review, we will introduce these methods from basic elements that a framework for AIS based IDS requires, which are shown in Figure 1.

In order to apply AIS to IDS, three steps are followed in this framework. The first step (the left gray box in Figure 1) is to represent the elements of the system and interaction of individuals in an immune-like form. The goal of this step is to represent the ID elements in an immunology way (e.g., creating abstract models of immune cells, molecules, etc.) and quantify the interaction of these elements by affinity measures. For example the abnormal behavior in IDS is presented as the antigen (nonself) in AIS. In ID domain, affinity means the similarity between detectors and data. Different representations can adopt different affinity measures. The second step is to generate the initial repertoires (generation algorithm), and the third step is to optimize the algorithm (evolution mode). More immune algorithms can be selected for these two steps. This framework can be thought of as a design procedure for engineer AIS inspired IDS. On this foundation three issues will be discussed in the next sections: antibody/antigen encoding, generation algorithm, and evolution mode.

3. Antibody/Antigen Encoding

The core of HIS is self and nonself discrimination performed by lymphocytes, which is similar to the IDS that distinguishes normal and abnormal behavior. The key of modeling of this mechanism in AIS based IDS is how to represent the elements in problem domain and decide the matching rules. Antibodies are generated by random combinations of a set of gene segments. Therefore, representation of detectors is to encode them as gene sequences. In AIS based IDS, we follow [12] in assuming the general case that each antibody Ab is a detector represented by an L-dimensional vector $\mathrm{Ab} = \langle \mathrm{Ab}_1, \mathrm{Ab}_2, \ldots, \mathrm{Ab}_L \rangle$ and each antigen Ag is a data to be classified which is represented by an L-dimensional vector $\mathrm{Ag} = \langle \mathrm{Ag}_1, \mathrm{Ag}_2, \ldots, \mathrm{Ag}_L \rangle$, where L is the length of the vector. Each antibody is then matched against each of the antigens and recognized them. The affinity, when mapped into the ID domain, means the similarity between Ag and Ab.

Because any data are eventually implemented as binary bits in computers, researches focused on binary representation as mainstream. That is why binary string is the most commonly adopted coding scheme in AIS. The first AIS model adopted binary encoding, which is suggested by Forrest et al., simulated the self-nonself discrimination principle of the HIS [14]. NSA is the core of this model, by which invalid detectors are eliminated when they matched self data. The NSA adopts binary encoding to simulate antibody/antigen. It breaks 32-bit string into eight substrings as antigen and antibody. Although not many immune features were employed, it shows the feasibility of this algorithm. LISYS (Lightweight Immune SYStem) is a relatively early model system used to protect the LAN from network based attacks [20]. In this system, each detector is a 49-bit binary string, mainly for TCP SYN packet; see Figure 2.

Later, virus-oriented CDIS [23] extended LYSIS further and used 320-bit binary string for each antibody signature,

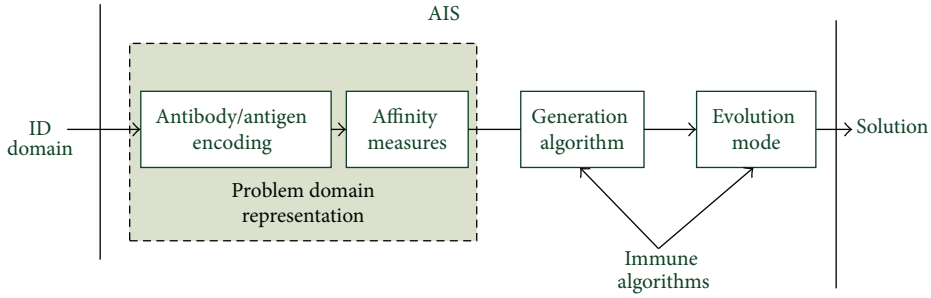


Figure 1: The framework for AIS based IDS design.

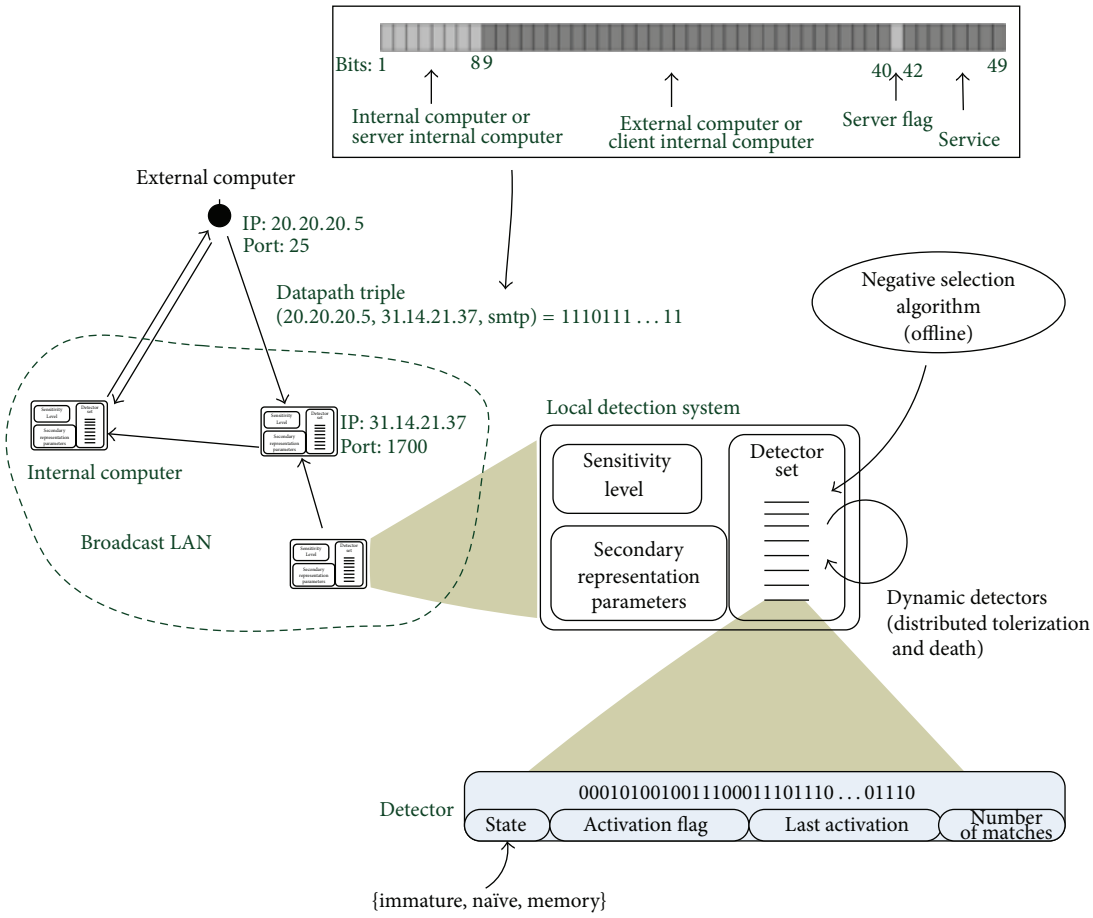


Figure 2: LISYS encoding of a TCP SYN packet [20].

comprising 29 of the possible data fields in a network protocol packet, to detect TCP, UDP, and ICMP. Kim and Bentley used a static CSA with NS operator as one component of the AIS for Network ID (NID). The component was especially developed for the purpose of building a misuse detector in a more efficient way [21]. They use binary genotypes to encode the conjunctive rule detectors, as shown in Figure 3. Then they investigated the dynamic clonal selection, and they found that it can adapt to novel data in NID [24]. A cooperative immunological approach for detecting network anomaly presented set of self as a binary vector for the communication triple (source, destination IP and Port, and protocol) [25].

By changing the encoding from binary to Gray code, the performance can be improved [26]. The reason is that codifications of two consecutive numbers have small Hamming distance. And this method still belongs to the binary encoding.

Most works have been restricted to binary representation of given data and detectors, but they use different

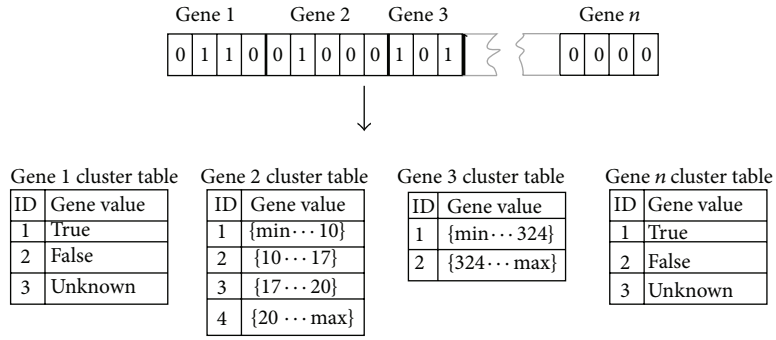


Figure 3: The DynamiCS gene representation [21].

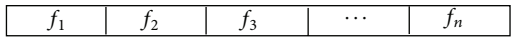


Figure 4: Real-value representation.

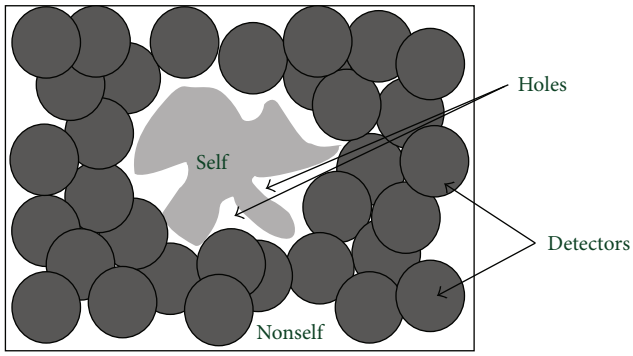


Figure 5: The NSA. Randomly generate candidate detectors (represented by dark circle); if they match any self (i.e., if any of the points covered by the detector are in the self-set), they are eliminated and regenerated until getting enough valid detectors [20].

affinity measures, for example, r-contiguous bits matching [14], r-chunks matching [27], landscape-affinity matching [23], Hamming distance [28], and Rogers and Tanimoto (R&T) matching [29], and so forth. However, this antibody/antigen encoding shows several drawbacks. The most significant problem is that the affinity relation between two binary strings represented by the matching rules results in a poor coverage of the problem space [30]. Moreover, the exponential growth of computational time caused by the number of generated detectors is large enough. In order to solve these problems, another different NSA was proposed by Gonzalez et al. [31]. In their method, antibodies were not represented as bit-strings; instead they were represented as hyperspheres. Gonzalez et al. called this approach, real-valued NS; each feature belongs to the range $[0.0, 1.0]$ as shown in Figure 4. They focused on real-valued anomaly detection problems rather than ID problems. This algorithm generates hyperspheres with equal radius lengths. Kim used NSA to build an anomaly detector for NID [32]. In the encoding of detectors, each gene of a detector uses decimal notation. The self profile has 33 different fields and this number determines the total number of corresponding genes in the detectors.

In real-valued NS algorithms, a large number of constant-sized detectors are needed to cover large area of nonself space, while no detectors may fit in the small area of nonself space, especially near the boundary between self and nonself [33, 34]. Hence a variable radius was suggested in the variable-sized detectors (termed V-detector) algorithm [35]. V-detector algorithm generates candidate detectors randomly, in which the radius of a detector is dynamically resized until the boundary of the region comes in contact with the nearest hypersphere of a self element. The algorithm terminates if a predefined number of detectors are generated or a predetermined proportion of nonself space is covered. The flexibility provided by the variable radius is easy to realize. Ostaszewski also calculated variable parameters of detectors to cover nonself space [36]. Besides that, a feedback NSA was proposed to solve the anomaly detection, which adjusts adaptively the self and detection radius and the number of detectors according to the detection result [37].

The issue of holes (the nonself region that cannot be covered by any valid detectors, see Figure 5) induced the geometrical detectors which means that not only the detector radius but also the shape of detector can be changed. Zhou Ji mentioned that detector variability can also be achieved by detector shapes or matching rules and so forth. NS with Detector Rules (NSDR) uses a genetic algorithm to evolve detectors with a hyperrectangle shape that can cover the nonself space. They used a sequential niching technique to evolve multiple detectors in the initial version [38] and then used deterministic crowding as the niching technique in the improved version [39]. In addition, Shapiro et al. used hyperellipsoids instead of hyperspheres to express detectors [40]. Hyperellipsoid is a special hypersphere; it can be stretched and reoriented to fit the boundary of self and nonself. Balachandran et al. incorporated these multiple hypershape detectors together to cover nonself area [41]. Their experimental results demonstrate that multishaped detectors provide better coverage of nonself space than other approaches using only a single type of detectors and less time.

When dealing with real-valued data, the majority of AIS researches use the Euclidean and Manhattan distances on the shape space [42]. Moreover, the difference between Euclidean and Manhattan distances has been discussed by Freitas and

Timmis [43]. More information about the other matching rules can be found in [42].

Finally, hybrid representations are possible and intuitively desirable when coping with data sets having attributes of different data types [44]. Numeric attributes are encoded in real-valued format, and category attributes are encoded in strings. In [45], authors chose parameters vector to represent the network pattern, including number of bytes and flag values. Nonetheless, some algorithms cannot handle that data. For instance, [26] apply NSA to a multidimensional personnel data containing both categorical and numeric data. However, instead of using a hybrid categorical/numeric representation and taking all the attributes into account, they simply ignore categorical attributes and work only with numeric attributes.

4. Generation Algorithm

Generating accurate and efficient detectors is important when AIS is applied to a detection problem. A good detector must not cover self space and should have minimum overlap with the rest of the detectors. Most NSA based methods randomly generate detectors as described in Forrest's original NSA. Random generation is uniformly distributed among nonself space and resolves problem of unknown nonself space. In training phase, the algorithm randomly generates a set of detectors; each fails to match any element in self. Then in test phase, these detectors are applied to classify new data as self or nonself, like Figure 5.

Although this method is frequently adopted in other research works, as pointed out by Stibor et al. [46], it increases the possibilities of generating invalid detectors. With the increase of self set size, the runtime complexity of detector generation has an exponential growth.

D'haeseleer et al. introduced two detector generating algorithms: linear time detector generating algorithm and greedy detector generating algorithm [47]. They were compared with the Forrest method which is called "exhaustive detector generating algorithm." The linear algorithm solves a counting recurrence for the number of strings unmatched by strings in candidate detectors and then uses the enumeration imposed by the counting recurrence to pick detectors randomly from this set of candidate detectors. Compared to the exhaustive algorithm, the advantage of linear algorithm is obvious, because it removes the pattern strings which will not become valid detector strings. The greedy algorithm improves upon the linear algorithm through the elimination of redundant detectors. It spreads the detectors apart and provides the maximum coverage for a given number of detectors. Nevertheless it sacrifices the speed of detector generation; the time will increase linearly with the size of self set. Castro and Timmis proposed the NS with mutation algorithm (NSMutation) which has better performance in terms of time complexity. NSMutation has a slight modification of the exhaustive stage of the NS by introducing somatic hypermutation [12]. The goal of NSMutation algorithm is to guide the candidate detector away from self set during the process of mutating a candidate detector. In [48], the authors drew a conclusion that NSMutation is similar to the exhaustive algorithm with the difference of eliminating redundancy and possessing parameters that can be optimized for better performance. All these detector generating algorithms time and space complexities are shown in Table 1, where m is the alphabet cardinality, l is the string length, r is matching threshold, N_S is the number of self, and N_R is the number of detectors.

In HIS, clonal selection is used to proliferate and differentiate the stimulation of cells with antigens. Burne proposed in 1959 [49] that we can improve the random detector generation by clonal selection principle. The artificial form of clonal selection was popularized by de Castro and Von Zuben. They gave an algorithm called CSA [50], which was then modified and renamed as CLONALG [9]. Garrett introduced an adaptive CSA as a modification of CLONALG [51]. CSA has always been used as strategy towards optimization and pattern recognition [52]. It is a colony search mechanism in nature, which enables detectors to clone their parents according to a mutation mechanism with high rates. This strategy evolves the immune systems so that they can deal with antigens that it has encountered in the past. From this, researchers combine clonal selection with other methods to solve ID problems. Kim and Bentley adopted the clonal selection as one component of the AIS for NID [25, 26, 52]. Liu et al. applied the CSA to the process of modeling normal behavior in ID, and experimental results showed that the algorithm has higher detection rate (DR) and lower false alarm rate (FA) [53], compared with the algorithms which apply the genetic algorithm to ID or apply the NSA of the AIS to ID. Tang et al. presented an avidity model based CSA for NID, which also has higher DR and lower FA compared with other approaches [54]. Besides that, many other approaches were mentioned in [55]. Additionally, the famous immune network model aiNet [56] also uses CLONALG with added network interactions. The mechanism used by the aiNet model is based on the ideas of clonal selection, and it mainly combines with the immune network theory. A network of stimulatory and suppressive interactions exists between antibodies that affects the concentrations of each type of antibody and then reaches a state of equilibrium. For more information, please refer to [57].

According to the features of AIS, many methods and techniques have been combined with AIS to better detect the abnormal behavior, like artificial neural networks, fuzzy systems, and genetic algorithms. For instance, [31] combined NSA and a conventional classification algorithm to perform anomaly detection; [58] presents an immunofuzzy approach to anomaly detection, because fuzzy logic can provide a better definition of the boundary between normal and abnormal behavior; Dasgupta et al. proposed a Multilevel Immune Learning Algorithm (MILA) to detect intrusions and issue alarms [59]. MILA detection used multiple strategies to generate detectors, where T detectors performed a low-level continuous bitwise match, while the B detectors performed a high-level match at noncontiguous positions of strings. Activated T detectors will further provide a signal to help activate B detectors. This model further simulated NSA, CSA, and somatic hypermutation of mature T cells and B cells. A

Table 1: Time and space complexities of all detector generating algorithms [48].

Algorithm	Time	Space
Exhaustive	$O(m^l \cdot N_S)$	$O(l \cdot N_S)$
Linear	$O((l-r+1) \cdot N_S \cdot m^r) + O((l-r+1) \cdot m^r) + O(l \cdot N_R)$	$O((l-r+1)^2 \cdot m^r)$
Greedy	$O((l-r+1) \cdot N_S \cdot m^r) + O((l-r+1) \cdot m^r \cdot N_R)$	$O((l-r+1)^2 \cdot m^r)$
NSMutation	$O(m^l \cdot N_S) + O(N_R \cdot m^r) + O(N_R)$	$O(l \cdot (N_S + N_R))$

hybrid system composed of AIS and self organising map is presented in [60]. Their experimental results showed higher detection and classification rate for Denial-of-Service and User-to-Root attacks.

Self and nonself discrimination is the fundamental principle which guides the AIS development. Therefore, NS acts as an important role in AIS. However, Matzinger proposed the Danger Theory (DT) and claimed that immune responses are triggered by the danger signals that are sent out when cells die an unnatural death, not by nonself antigens [61, 62]. It provides a fresh idea for AIS. Based on this idea, Aickelin and his research group applied DT to IDSs [17, 63]. In their research, danger signals are represented as numbers. Then, Twycross and Aickelin presented a libtissue framework incorporating ideas from innate immunity into AISs. The libtissue has a client/server architecture. Clients in libtissue collect antigen and external signals and transmit them to the libtissue server. The servers implement the AIS algorithm. They used libtissue for dynamic anomaly detection. From the dendritic cells and their interaction with T cells of the DT, the Dendritic Cell Algorithm (DCA) and Toll-Like Receptor Algorithm (TLRA) were proposed by Greensmith and Aickelin, Twycross and Aickelin, respectively. The DCA plus libtissue framework can scan port [64, 65]. The TLRA was deployed in the libtissue framework to detect process anomaly [66, 67]. Nonetheless, the DCA relies on the signal processing aspect by using multiple input and output signals, while the TLRA only uses danger signals. But the DTA is still controversial among immunologists about how to clearly define the danger signals.

5. Evolution Mode

With the development of the system, the detectors will increase. However, the system is finite, like the body; we cannot generate detectors infinitely. The old and invalid detectors must be eliminated. Whilst the intrusion behaviors appear every day, the new detectors must generate and evolve to detect them. Instead of inefficiently throwing away detectors that match self samples, Hofmeyr suggested changing the detectors over time, that is, to make them dynamic [20]. He gave each detector a finite lifetime; at the end of lifetime, the detector will be eliminated and replaced by a new randomly generated detector. He gave a figure of the lifecycle for a detector as shown in Figure 6.

Ayara et al. [48] and González and Dasgupta [68] tried to give detectors a period of time before eliminating them. Kim and Bentley investigated a further extension of Dynamic CS [69]; when memory detectors show a poor degree of self-tolerance to new antigens, they will be eliminated. Li proposed a receptor editing inspired real NSA [70]. For the detector that matches self, algorithm uses directional receptor editing to make a new life, and, for the detector that does not match self, algorithm uses direction receptor editing for identifying identical nearest self to expand coverage of nonself space.

If new detectors are generated by taking some feedback from previous detectors instead of random, then the new detector can be better suited for the nonself antigens. Hightower et al. [71], Perelson et al. [72], and Oprea and Forrest [73] employed a Genetic Algorithm (GA) to study the effects of evolution in the genetic encoding of the antibody molecules, which can be seemed as a feedback strategy. Moreover, in [74] Kim and Bentley embedded gene library evolutionary stage in their artificial immune model for NID. The gene library is a dynamic evolutionary library which stores the potential genes of detectors and diverse genetic mechanisms generate new detectors. The potential genes are the selected fields of profiles to describe anomalous network traffic patterns. After that, they use deleted memory detectors as the virtual gene library [75]. In fact, their method is consistent with the HIS theory, because deleted detectors also come from gene libraries. Zeng also uses gene library to generate the new detectors in initial IDS [76]. Thus, gene libraries provide a way of remembering past encounters so that antibody creation is more likely to match novel clusters which are nevertheless similar to those seen some time ago. More information about evaluation of the gene libraries in the AIS can be found in [77].

Gene library is an approach which guides the generation process to create antibodies with a good probability of success. However, gene library approaches are relatively complex. In addition to changing the radius and shape of the detector, another approach to improve the effectiveness is just moving the position of the detector. González and Dasgupta calculated the k-nearest neighbors of detector in the self set, and then the median distance of these k-neighbors is computed. If this median distance is less than a threshold, the detector is considered to match self and moves away to the opposite direction. This strategy is good to be robust to noise and outliers [68]. Laurentys et al. allocated the detectors in nonself space mixed moving detectors and generating detectors with constant radius and V-detector together [78].

An IDS evaluates a suspected intrusion once it has taken place and signals an alarm. In fact, most current ID methods

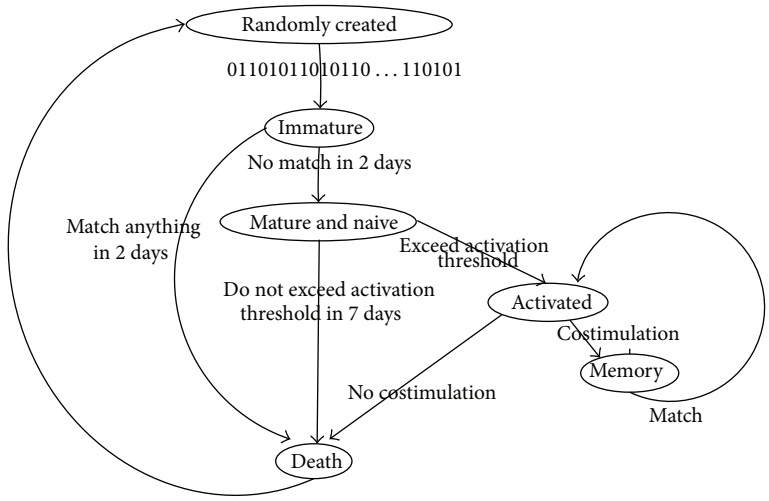


Figure 6: The lifecycle of a detector [20].

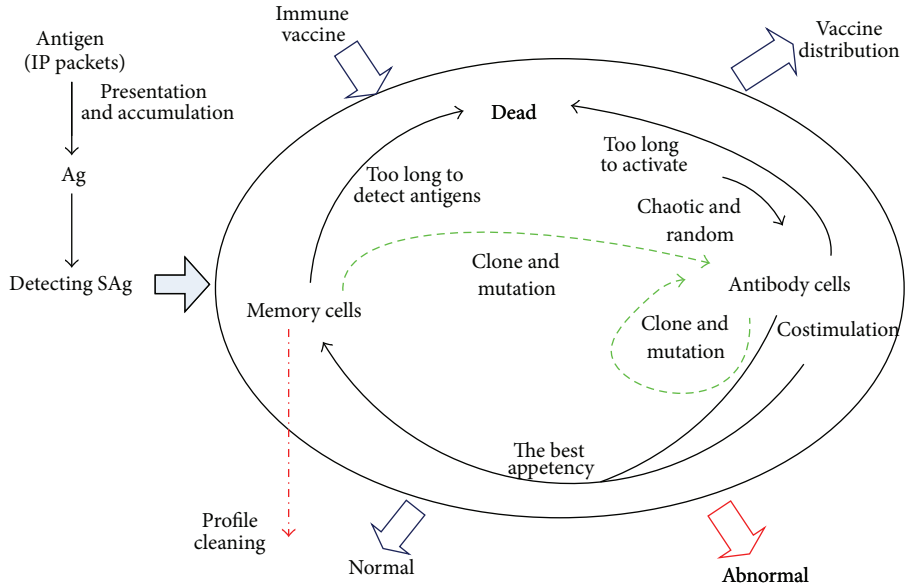


Figure 7: Dynamical real-time anomaly detection with immune NS [22].

cannot process large amounts of audit data for real-time operations. The roles of self and nonself may dynamically exchange; that is, the legal behaviors this time may be dangerous the next time, and vice versa. In the past few years, computer scientists have designed immune inspired algorithms that could detect the abnormal behavior effectively. DynamiCS has done a trial on this situation [24]. It can be able to deal with a real environment where self behaviors change after a certain period. DynamiCS introduced three important parameters: tolerization period of an immature detector, activation threshold of a mature detector, and the life span of a mature detector, but only one detection period for the self updating; it is too short to collect enough self elements. Li proposed a new immune based dynamic ID (Idid) model [79]. In Idid, the dynamic models and the corresponding recursive equations of the lifecycle of mature lymphocytes and the immune memory are built; the self and nonself dynamic description is solved. Yang et al. presented a model of network security based on AIS which utilized distributed agents to capture the network traffic in real time [80]. The model depicted the dynamic evolutions of self, antigens, immune-tolerance, lifecycle of mature agent, and immune memory. Their experimental results show that it has the features of real-time processing and self-adapting. Peng et al. proposed a Dynamic Anomaly Detection Algorithm with Immune NS (DADAI) [22], combining the antibody's clone theory and vaccination. It established dynamic evolvement formulations of detection profiles which can dynamically synchronize detection profiles with the real network environment. The algorithm is contained in Figure 7. Theoretical

analysis and experimental results showed that DADAI can be effectively deployed on the real-time NID under high-speed network environment.

6. The Future of Intrusion Detection

This review concentrated on the AIS based IDS. It first presented a brief introduction to the AIS in order to provide the readers with the background to understand. The main contribution of this paper is the framework for the design of AIS based IDS. Based on the framework, three aspects were described, followed by explorations of the literatures about IDSs. These theories and approaches based on AIS are able to combine to serve as a base for effective ID through our analysis. From the analysis of our framework, we find that system with real-valued representation is better suited for IDS, in which detectors effectively generate and dynamically evolve.

In the more recent years, AIS research has drifted away from more biologically appealing models to biological details, such as DCA, which is inspired by the role of dendritic cells (a specialized antigen presenting cells that provide a vital link between the innate and adaptive immune system) [81]. It is more useful in computer security, as not all abnormal events represent attacks [65, 82]. Grossman's Tunable Activation Threshold (TAT) hypothesis [83] is another perspective. TAT posits that each individual immune cell has its own tunable activation threshold whose value reflects the recent history of interactions with the surrounding environment. Antunes and Correia [84] described the deployed TAT based AIS for NID; [85] gives the analysis of TAT model. There are many useful and powerful algorithms that have already arisen and can arise when more than two of the different approaches are hybridized or new HIS theory is proposed.

Like [85] and [82–84], many summaries of the research in AIS were reported. HIS embodies the features of robustness, distribution, lightweight, self-organizing, and self-adapting. AISs are highly abstract models of their biological counterparts applied to solve problems in different areas. The analogy between the HIS and IDS naturally attracts computer scientists to make research on immune system approaches to ID. AISs have also been used in conjunction with other approaches in order to create more powerful models and improve individual performances.

Despite the existing advantages of AIS, now IDSs still have many problems, for example, lack of support of IPv6 addressing scheme, high levels of false positive and false negative alarm rates, lack of quick response for the unknown attacks. And AIS is a relatively young field; AIS based IDS faces many difficulties: real-world environments are much more complicated, self set constantly changes, and detection is in real time. In order to resolve all these issues and make progress for this research, our future IDSs should focus on the questions of quick response and less false alarm and false negative. In the future, depending on the biological immune mechanism, it will be able to propose effective ID models and algorithms, although there will be a difficult and winding road.

Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (no. 61173159).

References

[1] J. P. Anderson, *Computer Security Technology Planning Study*, vol. 2, James P. Anderson Company, Fort Washington, Pa, USA, 1972.

[2] J. P. Anderson, "Computer security threat monitoring and surveillance," Tech. Rep., James P. Anderson Company, Fort Washington, Pa, USA, 1980.

[3] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, pp. 222–232, 1987.

[4] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. D. Wolber, "A network security monitor," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 296–304, Oakland, Calif, USA, May 1990.

[5] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer Immunology," *Communications of the ACM*, vol. 40, no. 10, pp. 88–96, 1997.

[6] C. A. Janeway, P. Travers, M. Walport, and M. Shlomchik, *Immunobiology: The Immune System in Health and Disease*, Garland Science, New York, NY, USA, 2005.

[7] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: a review," *Applied Soft Computing Journal*, vol. 10, no. 1, pp. 1–35, 2010.

[8] M. F. A. Gadi, X. Wang, and A. P. do Lago, "Credit card fraud detection with artificial immune system," in *Artificial Immune Systems*, vol. 5132 of *Lecture Notes in Computer Science*, pp. 119–131, Springer, Berlin, Germany, 2008.

[9] L. N. de Castro and F. J. von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.

[10] A. Watkins, J. Timmis, and L. Boggess, "Artificial immune recognition system (AIRS): an immune-inspired supervised learning algorithm," *Genetic Programming and Evolvable Machines*, vol. 5, no. 3, pp. 291–317, 2004.

[11] J. Timmis, A. Tyrrell, M. Mokhtar, A. Ismail, N. Owens, and R. Bi, "An artificial immune system for robot organisms," in *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, pp. 268–288, Springer, Berlin, Germany, 2010.

[12] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, Berlin, Germany, 2002.

[13] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1–3, pp. 187–204, 1986.

[14] S. Forrest, L. Allen, A. S. Perelson, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 202–212, Oakland, Calif, USA, May 1994.

[15] J. O. Kephart, "A biologically inspired immune system for computers," in *Artificial Life IV: Proceedings of the Fourth*

International Workshop on the Synthesis and Simulation of Living Systems, pp. 130–139, MIT Press, Cambridge, Mass, USA, 1994.

[16] L. N. de Castro and F. J. von Zuben, "The clonal selection algorithm with engineering applications," in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO '00)*, pp. 36–39, Las Vegas, Nev, USA, July 2000.

[17] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod, "Danger theory: the link between AIS and IDS?" in *Artificial Immune Systems*, vol. 2787 of *Lecture Notes in Computer Science*, pp. 147–155, Springer, Berlin, Germany, 2003.

[18] U. Aickelin, J. Greensmith, and J. Twycross, "Immune system approaches to intrusion detection-a review," in *Artificial Immune Systems*, vol. 3239 of *Lecture Notes in Computer Science*, pp. 316–329, Springer, Berlin, Germany, 2004.

[19] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross, "Immune system approaches to intrusion detection—a review," *Natural Computing*, vol. 6, no. 4, pp. 413–466, 2007.

[20] S. A. Hofmeyr and S. Forrest, *An Immunological Model of Distributed Detection and Its Application to Computer Security*, The University of New Mexico, Albuquerque, NM, USA, 1999.

[21] J. Kim and P. J. Bentley, "Towards an artificial immune system for network intrusion detection: an investigation of clonal selection with a negative selection operator," in *Proceedings of the Congress on Evolutionary Computation (CEC '01)*, pp. 1244–1252, Seoul, Korea, May 2001.

[22] L. Peng, W. Chen, D. Xie, Y. Gao, and C. Liang, "Dynamically real-time anomaly detection algorithm with immune negative selection," *Applied Mathematics & Information Sciences*, vol. 7, no. 3, pp. 1157–1163, 2013.

[23] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont, "An artificial immune system architecture for computer security applications," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 252–280, 2002.

[24] J. Kim and P. J. Bentley, "Towards an artificial immune system for network intrusion detection: an investigation of clonal selection," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1015–1020, Honolulu, Hawaii, USA, May 2002.

[25] T. S. Sobh and W. M. Mostafa, "A cooperative immunological approach for detecting network anomaly," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 1275–1283, 2011.

[26] D. Dasgupta and N. S. Majumdar, "Anomaly detection in multidimensional data using negative selection algorithm," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1039–1044, Honolulu, Hawaii, USA, May 2002.

[27] J. Balthrop, F. Esponda, S. Forrest, and M. Glickman, "Coverage and generalization in an artificial immune system," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pp. 3–10, July 2002.

[28] S. Forrest and S. Hofmeyr, "Immunity by design: an artificial immune system," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, pp. 1289–1296, Morgan-Kaufmann, San Francisco, Calif, USA, 1999.

[29] P. K. Harmer, "A distributed agent architecture for a computer virus immune system," DTIC Document, 2000.

[30] F. González, D. Dasgupta, and J. Gómez, "The effect of binary matching rules in negative selection," in *Genetic and Evolutionary Computation-GECCO 2003*, vol. 2723 of *Lecture Notes in Computer Science*, pp. 195–206, Springer, Berlin, Germany, 2003.

[31] F. Gonzalez, D. Dasgupta, and R. Kozma, "Combining negative selection and classification techniques for anomaly detection," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, vol. 1, pp. 705–710, Honolulu, Hawaii, USA, May 2002.

[32] J. Kim and P. J. Bentley, "An evaluation of negative selection in an artificial immune system for network intrusion detection," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '01)*, pp. 1330–1337, 2001.

[33] Z. Ji, "A boundary-aware negative selection algorithm," in *Proceedings of the 9th IASTED International Conference on Artificial Intelligence and Soft Computing (ASC '05)*, Acta Press, Benidorm, Spain, 2005.

[34] D. Wang, F. Zhang, and L. Xi, "Evolving boundary detector for anomaly detection," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2412–2420, 2011.

[35] Z. Ji and D. Dasgupta, "Real-valued negative selection algorithm with variable-sized detectors," in *Genetic and Evolutionary Computation-GECCO 2004*, vol. 3102 of *Lecture Notes in Computer Science*, pp. 287–298, Springer, Berlin, Germany, 2004.

[36] M. Ostaszewski, F. Seredynski, and P. Bouvry, "Coevolutionary-based mechanisms for network anomaly detection," *Journal of Mathematical Modelling and Algorithms*, vol. 6, no. 3, pp. 411–431, 2007.

[37] J. Zeng, T. Li, X. Liu, C. Liu, L. Peng, and F. Sun, "A feedback negative selection algorithm to anomaly detection," in *Proceedings of the 3rd International Conference on Natural Computation (ICNC '07)*, pp. 604–608, Haikou, China, August 2007.

[38] D. Dasgupta and F. González, "An immunity-based technique to characterize intrusions in computer networks," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 281–291, 2002.

[39] F. A. Gonzalez and D. Dasgupta, "An imunogenetic technique to detect anomalies in network traffic," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pp. 1081–1088, Morgan Kaufmann, 2002.

[40] J. M. Shapiro, G. B. Lament, and G. L. Peterson, "An evolutionary algorithm to generate hyper-ellipsoid detectors for negative selection," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 337–344, Atlanta, Ga, USA, June 2005.

[41] S. Balachandran, D. Dasgupta, F. Nino, and D. Garrett, "A framework for evolving multi-shaped detectors in negative selection," in *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence (FOCI '07)*, pp. 401–408, Honolulu, Hawaii, USA, April 2007.

[42] Z. Ji and D. Dasgupta, "Revisiting negative selection algorithms," *Evolutionary Computation*, vol. 15, no. 2, pp. 223–251, 2007.

[43] A. A. Freitas and J. Timmis, "Revisiting the foundations of artificial immune systems: a problem-oriented perspective," in *Artificial Immune Systems*, vol. 2787 of *Lecture Notes in Computer Science*, pp. 229–241, Springer, Berlin, Germany, 2003.

[44] X. Hang and H. Dai, "An extended negative selection algorithm for anomaly detection," in *Advances in Knowledge Discovery and Data Mining*, vol. 3056 of *Lecture Notes in Computer Science*, pp. 245–254, Springer, Berlin, Germany, 2004.

[45] V. D. Kotov and V. I. Vasilyev, "Immune model based approach for network intrusion detection," in *Proceedings of the 3rd International Conference on Security of Information and Networks (SIN '10)*, pp. 233–237, Taganrog, Russia, September 2010.

[46] T. Stibor, P. Mohr, and J. Timmis, "Is negative selection appropriate for anomaly detection?" in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 321–328, Washington, DC, USA, June 2005.

[47] P. D'haeseleer, S. Forrest, and P. Helman, "Immunological approach to change detection: algorithms, analysis and implications," in *Proceedings of the 17th IEEE Symposium on Security and Privacy*, pp. 110–119, May 1996.

[48] M. Ayara, J. Timmis, R. de Lemos, L. N. de Castro, and R. Duncan, "Negative selection: how to generate detectors," in *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS '02)*, pp. 89–98, 2002.

[49] S. F. M. Burnet, *The Clonal Selection Theory of Acquired Immunity*, vol. 3, Vanderbilt University Press, Nashville, Tenn, USA, 1959.

[50] L. N. de Castro and F. J. Von Zuben, "Artificial immune systems: part I-basic theory and applications," Tech. Rep., Universidade Estadual de Campinas, Campinas, Brazil, 1999.

[51] S. M. Garrett, "Parameter-free, adaptive clonal selection," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, pp. 1052–1058, June 2004.

[52] S. M. Garrett, "How do we evaluate artificial immune systems?" *Evolutionary Computation*, vol. 13, no. 2, pp. 145–177, 2005.

[53] F. Liu, B. Qu, and R. Chen, "Intrusion detection based on immune clonal selection algorithms," in *AI 2004: Advances in Artificial Intelligence*, vol. 3339 of *Lecture Notes in Computer Science*, pp. 1226–1232, Springer, Berlin, Germany, 2004.

[54] W. Tang, X.-M. Yang, X. Xie, L.-M. Peng, C.-H. Youn, and Y. Cao, "Avidity-model based clonal selection algorithm for network intrusion detection," in *Proceedings of the IEEE 18th International Workshop on Quality of Service (IWQoS '10)*, pp. 1–5, Beijing, China, June 2010.

[55] D. Dasgupta, S. Yu, and F. Nino, "Recent advances in artificial immune systems: models and applications," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1574–1587, 2011.

[56] L. Nunes de Casto and F. J. Von Zuben, "An evolutionary immune network for data clustering," in *Proceedings of the 6th Brazilian Symposium on Neural Networks*, pp. 84–89, Rio de Janeiro, Barzil, 2000.

[57] J. C. Galeano, A. Veloza-Suan, and F. A. González, "A comparative analysis of artificial immune network models," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 361–368, Washington, DC, USA, June 2005.

[58] J. Gómez, F. González, and D. Dasgupta, "An immuno-fuzzy approach to anomaly detection," in *Proceedings of the 12th IEEE International Conference on Fuzzy Systems (FUZZ '03)*, pp. 1219–1224, Baton Rouge, La, USA, May 2003.

[59] D. Dasgupta, S. Yu, and N. S. Majumdar, "MILA-multilevel immune learning algorithm and its application to anomaly detection," *Soft Computing*, vol. 9, no. 3, pp. 172–184, 2005.

[60] S. T. Powers and J. He, "A hybrid artificial immune system and Self Organising Map for network intrusion detection," *Information Sciences*, vol. 178, no. 15, pp. 3024–3042, 2008.

[61] P. Matzinger, "Tolerance, danger, and the extended family," *Annual Review of Immunology*, vol. 12, pp. 991–1045, 1994.

[62] P. Matzinger, "Essay 1: the danger model in its historical context," *Scandinavian Journal of Immunology*, vol. 54, no. 1-2, pp. 4–9, 2001.

[63] U. Aickelin and S. Cayzer, "The danger theory and its application to artificial immune systems," in *Proceedings of the 1st Internat Conference on ARtif icial Immune Systems (ICARIS '02)*, pp. 141–148, Canterbury, UK, 2002.

[64] J. Greensmith and U. Aickelin, "Dendritic cells for real-time anomaly detection," in *Proceedings of the Workshop on Artificial Immune Systems and Immune System Modelling (AISB '06)*, pp. 7–8, Bristol, UK, April 2006.

[65] J. Greensmith and U. Aickelin, "Dendritic cells for SYN scan detection," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 49–56, London, UK, July 2007.

[66] J. Twycross and U. Aickelin, "An immune inspired approach to anomaly detection," in *Handbook of Research on Information Assurance and Security*, charpter 10, pp. 109–121, Information Science Reference, New York, NY, USA, 2007.

[67] J. P. Twycross and U. Aickelin, *Integrated innate and adaptive artificial immune systems applied to process anomaly detection [Ph.D. thesis]*, University of Nottingham, Nottingham, UK, 2007.

[68] F. A. González and D. Dasgupta, "Anomaly detection using real-valued negative selection," *Genetic Programming and Evolvable Machines*, vol. 4, no. 4, pp. 383–403, 2003.

[69] J. Kim and P. J. Bentley, "Immune memory in the dynamic clonal selection algorithm," in *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS '02)*, pp. 59–67, 2002.

[70] G. Y. Li and T. Guo, "Receptor editing-inspired real negative selection algorithm," *Computer Science*, vol. 39, pp. 246–251, 2012.

[71] R. Hightower, S. Forrest, and A. S. Perelson, "The evolution of secondary organization in immune system gene libraries," in *Proceedings of the 2nd European Conference on Artificial Life*, pp. 458–470, Brussels, Belgium, 1994.

[72] A. S. Perelson, R. Hightower, and S. Forrest, "Evolution and somatic learning in V-region genes," *Research in Immunology*, vol. 147, no. 4, pp. 202–208, 1996.

[73] M. Oprea and S. Forrest, "How the immune system generates diversity: Pathogen space coverage with random and evolved antibody libraries," Tech. Rep. 99-02-014, 1999.

[74] J. Kim and P. Bentley, "The artificial immune model for network intrusion detection," in *Proceedings of the 7th European Conference on Intelligent Techniques and Soft Computing (EUFIT '99)*, Aachen, Germany, 1999.

[75] J. Kim and P. J. Bentley, "A model of gene library evolution in the dynamic clonal selection algorithm," in *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS '02)*, Canterbury, UK, 2002.

[76] J. Zeng, X. Liu, T. Li, G. Li, H. Li, and J. Zeng, "A novel intrusion detection approach learned from the change of antibody concentration in biological immune response," *Applied Intelligence*, vol. 35, no. 1, pp. 41–62, 2011.

[77] S. Cayzer, J. Smith, J. A. R. Marshall, and T. Kovacs, "What have gene libraries done for AIS?" in *Artificial Immune Systems*, vol. 3627 of *Lecture Notes in Computer Science*, pp. 86–99, Springer, Berlin, Germany, 2005.

[78] C. A. Laurentys, G. Ronacher, R. M. Palhares, and W. M. Caminhas, "Design of an artificial immune system for fault detection: a negative selection approach," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5507–5513, 2010.

[79] T. Li, "An immune based dynamic intrusion detection model," *Chinese Science Bulletin*, vol. 50, no. 22, pp. 2650–2657, 2005.

[80] J. Yang, X. Liu, T. Li, G. Liang, and S. Liu, "Distributed agents model for intrusion detection based on AIS," *Knowledge-Based Systems*, vol. 22, no. 2, pp. 115–119, 2009.

[81] J. Greensmith, U. Aickelin, and S. Cayzer, "Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection," in *Artificial Immune Systems*, vol. 3627 of *Lecture Notes in Computer Science*, pp. 153–167, Springer, Berlin, Germany, 2005.

[82] J. Kim, P. Bentley, C. Wallenta, M. Ahmed, and S. Hailes, "Danger is ubiquitous: detecting malicious activities in sensor networks using the dendritic cell algorithm," in *Artificial Immune Systems*, vol. 3627 of *Lecture Notes in Computer Science*, pp. 153–167, Springer, Berlin, Germany, 2005.

[83] Z. Grossman and A. Singer, "Tuning of activation thresholds explains flexibility in the selection and development of T cells in the thymus," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 25, pp. 14747–14752, 1996.

[84] M. Antunes and M. Correia, "TAT-NIDS: an immune-based anomaly detection architecture for network intrusion detection," in *Proceedings of the 2nd International Workshop on Practical Applications of Computational Biology and Bioinformatics (IWPACBB '08)*, pp. 60–67, Salamanca, Spain, 2009.

[85] P. S. Andrews and J. Timmis, "Tunable detectors for artificial immune systems: from model to algorithm," in *Bioinformatics for Immunomics*, vol. 3, pp. 103–127, Springer, New York, NY, USA, 2010.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 696904, 12 pages
http://dx.doi.org/10.1155/2014/696904

Research Article

Reusable Component Model Development Approach for Parallel and Distributed Simulation

Feng Zhu,[1,2] Yiping Yao,[1,2] Huilong Chen,[1] and Feng Yao[1]

[1] *State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China*
[2] *College of Information System and Management, National University of Defense Technology, Changsha 410073, China*

Correspondence should be addressed to Feng Zhu; zhufeng@nudt.edu.cn

Received 21 September 2013; Accepted 24 December 2013; Published 3 March 2014

Academic Editors: S. K. Bhatia, K. K. Mishra, A. K. Misra, and P. Muller

Model reuse is a key issue to be resolved in parallel and distributed simulation at present. However, component models built by different domain experts usually have diversiform interfaces, couple tightly, and bind with simulation platforms closely. As a result, they are difficult to be reused across different simulation platforms and applications. To address the problem, this paper first proposed a reusable component model framework. Based on this framework, then our reusable model development approach is elaborated, which contains two phases: (1) domain experts create simulation computational modules observing three principles to achieve their independence; (2) model developer encapsulates these simulation computational modules with six standard service interfaces to improve their reusability. The case study of a radar model indicates that the model developed using our approach has good reusability and it is easy to be used in different simulation platforms and applications.

1. Introduction

With the rapid development of simulation platforms and simulation applications during the last decades, in particular for parallel and distributed simulation, one of the most important challenges is how to respond quickly to new application requirements while reducing the development costs [1–3]. Building application from existing simulation models rather than from scratch is considered as a promising approach to improve the development efficiency, as well as to minimize engineering efforts and resource costs [4–6]. Reuse-oriented models are developed to be reused across simulation platforms with little or even no modification. At the same time, the reuse of component models together with visual programming technology makes it possible to drag and drop existing component models to assemble the simulation application, thus significantly reducing development time [7–9]. In addition, model reuse not only improves productivity but also has a positive impact on the quality of software products because of the obvious fact that a simulation application will work properly if it has already worked before [10, 11].

Motivated by the advantages of model reuse, various reusable model development approaches have been developed. (1) Based on a specific modeling language in which people designs special simulation function module as primitive and control module as simulator, different models created with the same modeling language can be reused for the corresponding simulator. However, most of simulation modeling languages are usually related to domain knowledge closely and each research field involves some specific platforms, so that there is congenitally deficiency for them to create models that can be reused across multidomain platforms, such as continue system simulation language *ACSL* [12], discrete event system simulation language *GPSS* [13], and multifield physical simulation language *Modelica* [14]. (2) Using a modeling specification which defines uniform internal structure, behavior constraint, and external interfaces for models, models will have good reusability if they do not bind with any platforms. However unfortunately, existing modeling specifications do not emphasize that model development should be independent with other simulation platforms. For example, *ESA* proposed a reusable simulation model description specification *SMP* (Simulation Model Portability Standards) [15, 16]. It is not supported well to reuse the models on other runtime platforms, because the execution of a *SMP* model depends on services provided

by the SMP simulation platform. *The University of Arizona* gives a thorough discussion of parallel discrete event system specification *DEVS* [17], which mostly focuses on the hierarchical structure of components. (3) Some models are based on simulation environment which provides runtime supporting platform for model running. The reusability of them is also limited. Taking *HLA* [18] for instance, each federate provides some interfaces which compiles with the *HLA* interface specification, and federates developed by different developers can communicate with each other via the runtime infrastructure using these interfaces. But these federates are hard to use in other simulation platforms, such as *SUPE* [19], *POSE* [20], and *Charm++* [21], because these platforms cannot "identify" any HLA service interfaces. In summary, as far as the authors know, there is not an adapted approach to guide users to develop models to achieve model reuse across platforms currently. Therefore, we have to identify a reliable way to develop what we mean by a "reusable" component model. Such an approach may help us not only to learn how to build reusable components but also to recognize whether a component model can be reused in a new simulation application.

In this paper, we first discussed a reusable component model framework composed of a simulation computational module and six well-defined standard interfaces. Next we proposed a reusable component model development approach, which contains two phases. In the first phase, the independent simulation computational modules are implemented under three principles. In the next phase, model developers encapsulated these simulation modules with the six standard interfaces to improve their reusability. Our final goal is to achieve flexible cross-platform reuse and fast composition of component model in parallel and distributed simulation.

Section 2 discusses the process of a reusable component model-based simulation application development. Section 3 introduces a reusable component model framework and discusses the execution flow of six service interfaces. Section 4 presents our reusable model development approach which elaborates the three principles that domain experts must observe and the encapsulation process. Section 5 describes a case study in the field of military simulation, to create a reusable radar model which can be reused in air-defense and antimissile of naval vessel system. Finally, our conclusion will be made with an indication of the future work.

2. Reusable Component Model-Based Simulation Application

The term *reusable component model* refers to an independent replaceable part of a simulation application that can be independently developed and delivered as a unit and reused in different platforms [22]. Such component models can be selected from a model resource library, thus reducing both model developing time and costs when compared with a new development [23]. Meanwhile, reuse increases the reliability of the components, furnishing added "testing," with the consequence that library component models are more reliable and less prone to faulty behavior [24].

Component-based software development is associated with a shift from object-oriented coding to system building by plugging together components [25]. Figure 1 shows the simulation application development process based on reusable component models in parallel and distributed simulation. Component models are selected from a model resource library to construct simulation executable units, namely, simulation entities, which are used to assemble a simulation application running on the corresponding simulation platform. Unidirectional dotted lines between component models represent that simulation entity dispatches component models to compute according to the input and output relationship, while bidirectional active lines between simulation entities represent that simulation entities communicate with each other via external services provided by the corresponding simulation platform. Because simulation entities are different between various simulation platforms, if there are no constrains for model development, these models will be hard to reuse across different simulation platforms. As discussed earlier, models which call *HLA* interfaces are hard to be used in any other discrete event simulation platforms. Thus, to achieve model reuse not only in *HLA*, but also in many other simulation environments, we have to define a component model framework as a standard reusable model specification first.

3. Reusable Component Model Framework

Our reusable component model framework mainly contains a simulation computational module and six standard service interfaces. The simulation computational module is the internal implementation which contains a set of mathematical functions to model a process or function in real world. Six service interfaces, including state restoring interface, input interface, dynamic data driven interface, business process interface, state getting interface, and output interface, are used to encapsulate the internal mathematical functions provided by the simulation computational model. Thanks to the six standard interfaces, the reusable component model is very easy to be assembled or replaced rapidly, which will decrease software development time.

Figure 2 shows the reusable component model framework. As the figure shows, there is a configuration file parsing interface outside the simulation computational module. That is because most of component models have several attributes which need to be configured to satisfy the different demands for different simulation applications. Taking a radar model for example, we can assign different values for radar cross-section, probability of false alarm, or length of antennas to implement suitable radars for some specific applications. Moreover, our component model framework provides a state rolling back strategy, which makes the component model used in the simulation application with optimism mechanism. Different from many other methods storing the state of a model by itself, our strategy uses the state getting interface to get the model state data by the outside simulation entity. Then the simulation entity will define a roll back variable to save the state data, so no models need to do the "saving" thing. When rolling back happened, the simulation entity gets the value of

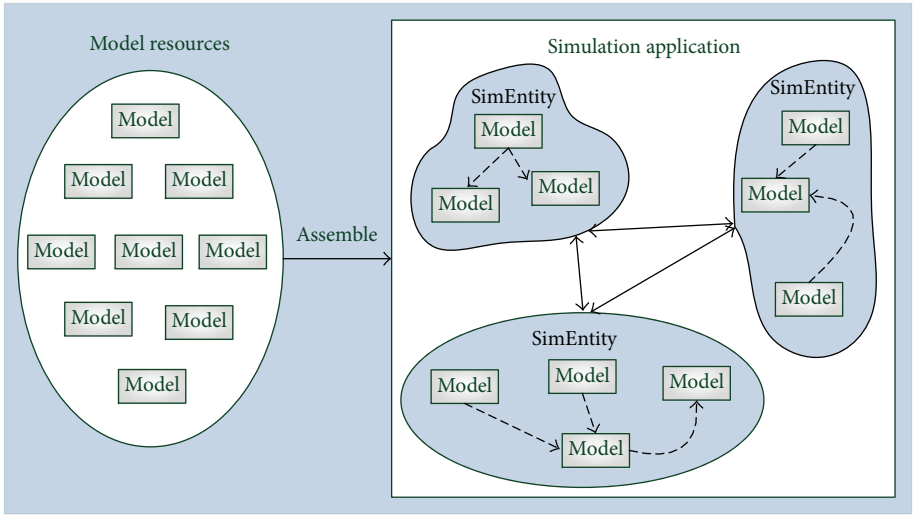


Figure 1: Component model-based simulation application development process in parallel and distributed simulation.

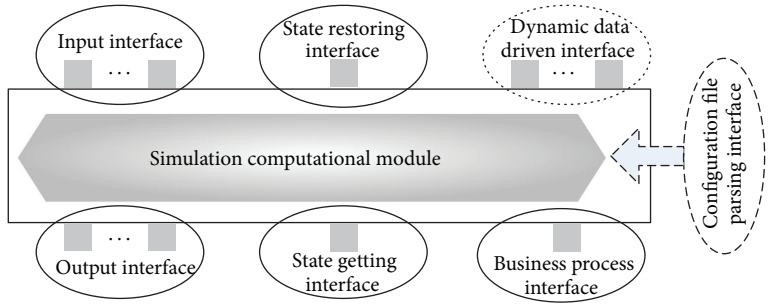


Figure 2: Reusable component model framework.

state variable referent to the current simulation time, and then it calls the state restoring interface using the state variable as parameter to roll back the state of the reusable component model.

Figure 3 shows the execution flow for a reusable component model, the details of which are described in the following.

(1) Call state restoring interface *setstate* (S) to restore the state of the component model to S, which is a variable saving the state data of the simulation model. If this interface is called in the first time, S is on behalf of initial state data; otherwise, S is the state data stored by external simulation entity after once computation.

(2) Call dynamic data driven interface *driven* (t, x) to provide dynamic data for the component model during its execution. t is the current time and x represents dynamic external input parameter set which depends on external simulation entity, not always needed for model computation each time.

(3) Call input interface *input* (t, x) to provide input data for the component model during its execution. t is the current simulation time and x represents input parameter set which is always needed for model computation each time.

(4) Call business process interface *process* (t') to implement some important behaviors in a model of interest though internal state transition functions. After its execution, the local simulation time of the component model advances to t'.

(5) Call output interface *output* (t', y) to output the result after model processing. t' is the current local simulation time and x represents an output parameter.

(6) Call state getting interface *getstate* (S') to get the state data of the component model after model processing and S' represents the state data of the component model to be saved.

4. Two-Phase Development Approach

As discussed above, a component model can be considered as an implementation of a system, process, or function. It usually has a close relationship with multidisciplinary domain knowledge such as hydromechanics and aerodynamics, so that a software developer without domain expert's help cannot afford to create a model with high quality. On the other hand, if we know there are some component models that we can use again, probably with some slight

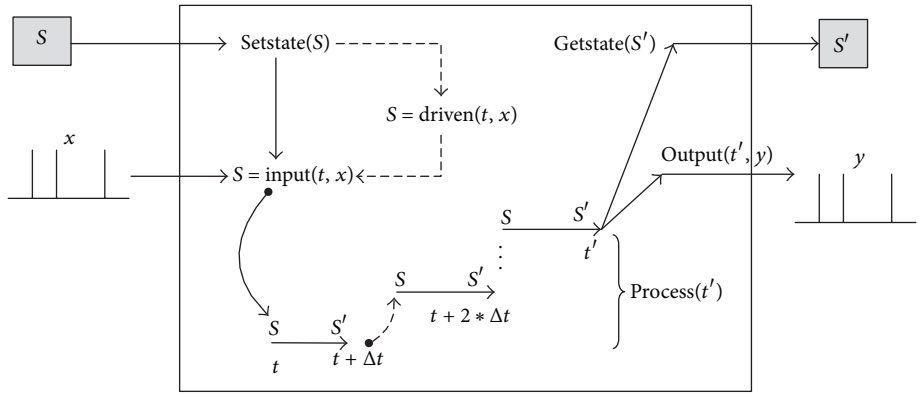


Figure 3: Execution flow of reusable component model.

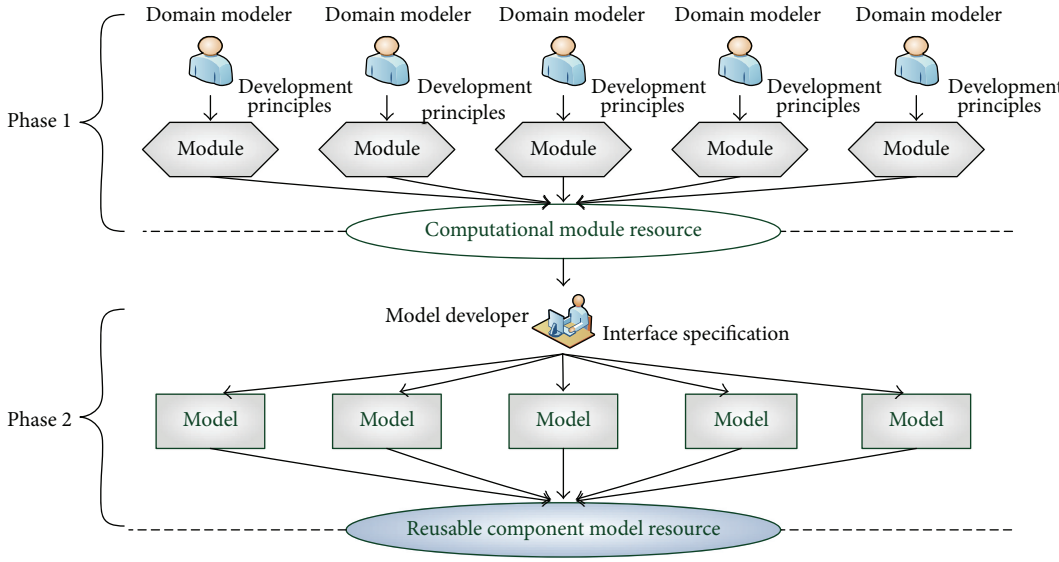


Figure 4: Two-phase reusable component model development process.

modification, then we will do so as long as we trust the person who creates the model [26].

As lack of guideline for component model encapsulation, the models are usually characteristic of diversiform interfaces, coupled tightly, and bound together with simulation platforms. This implies that the integration of these models to a simulation application is a great challenge for developers, not to mention to reuse them.

In view of the above reasons, our reusable model development approach contains two phases shown in Figure 4. In the first phase, domain modelers create a simulation computational module distributed observing three principles in the following, not only to reduce the overlap time of development, but also to improve its credibility. In the next phase, this simulation computational module is encapsulated to a reusable component model by a model developer according to the above six service interfaces to improve its reusability.

Phase 1: Domain Modeler Implements Internal Simulation Computational Module. According to component-based simulation application development paradigm, component models are to be reused across various products and product families. Since the internal state transmission functions are provided by the simulation computational module, to achieve model reuse across different platforms, the simulation computational module must be characterized and constrained properly. There are three principles that domain modeler must observe while creating simulation computational modules.

(1) Simulation Computational Module Should Be Independent with Other Modules. To achieve loose coupling and modularity, the execution of each module cannot depend on any other modules. This means that each module cannot call functions or access shared data in other modules. Modules which are developed by different domain modelers only communicate with each other via the external simulation entity. Taking Figure 5 for example, there are two situations according to whether two communicated models are in the same simulation entity or not. If the two models are in the

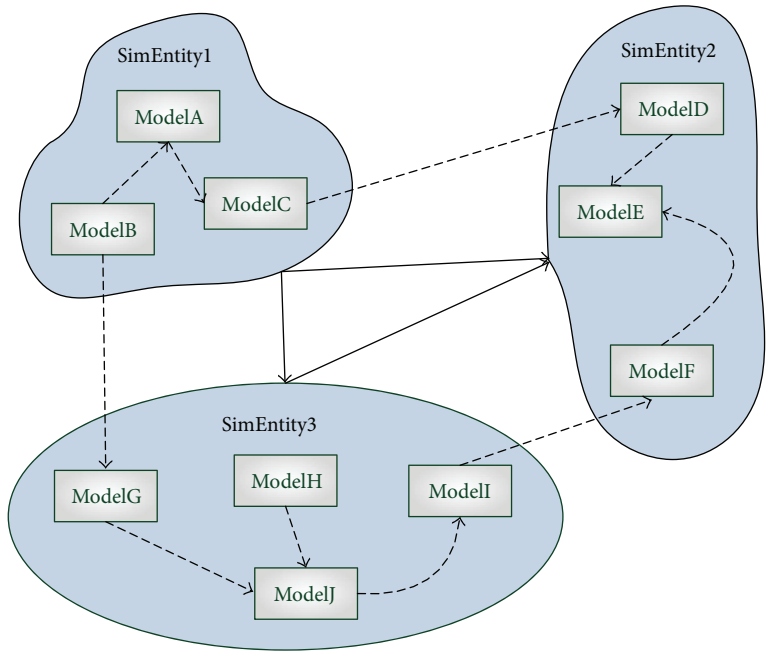


Figure 5: Communication between component models.

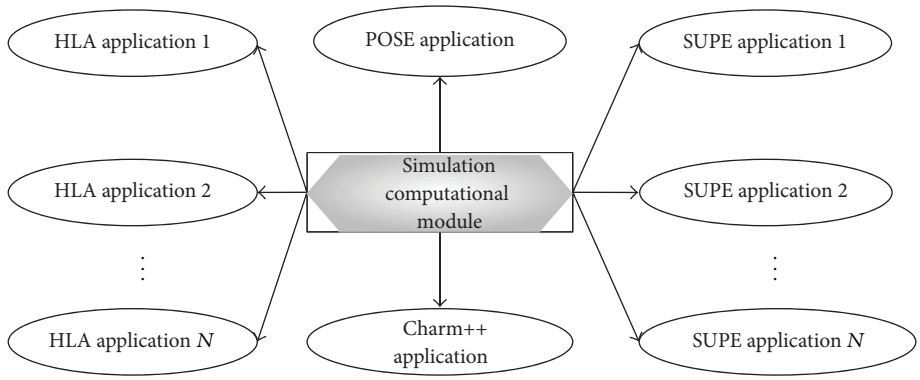


Figure 6: Model reuse across different platforms.

same simulation entity, like *ModelA* and *ModelB*, *SimEntity1* will dispatch *ModelB* to compute first and then takes the output data of *ModelB* as the input data of *ModelA*. If the two models are in different simulation entity, like *ModelB* and *ModelG*, *SimEntity1* will also dispatch *ModelB* to compute first and then transfers the output data of *ModelB* to *SimEntity3*, which will receive the data and paper it as the input data of *ModelG*.

(2) Simulation Computational Modules Should Be Independent with Simulation Platforms. Simulation execution can be taken as several simulation computational modules running on a simulation platform. The simulation platform provides runtime environment for these computational modules. However, there are great differences between the mechanisms of different simulation platforms. For example, in Figure 6, in *HLA*, component models are used to construct federates which communicate with each other via *HLA* service interfaces provided by the runtime infrastructure, while, in discrete event simulation, component models are used to construct simulation objects which communicate with each other by calling event-scheduling interface provided by corresponding simulation engine. As a consequence, to achieve model reuse across different simulation platforms, the implementation of computational modules should be independent with simulation platforms; in other words it cannot call any service provided by any simulation platforms. If a component model calls *HLA* service interfaces directly, it is hard to be reused in other simulation platforms.

(3) Simulation Computational Modules Should Provide Important Description Information for Users. With the purpose of protecting intellectual property rights, modules are always encapsulated as a "black box" to model developers. The implication of this is that, unless a computational module is quite simple, a model developer will have to spend a great

deal of time understanding how the computational module works. Thus, to facilitate model developers to encapsulate computational module, some important informal descriptions should be provided, which encompasses the following characteristics.

Phase 2: Model Developers Encapsulate the Simulation Computational Module with Six Standard Service Interfaces. The simulation computational module developed by a domain modeler has high reliability but pool usability. How to transform it to a component model with good reusability is our focus. Therefore on the precondition of no changes that happed for the business logic of the simulation computational module, model developers need to encapsulate the simulation computational module with six standard service interfaces in the following.

(1) State Restoring Interface (int setstate(string & SimuState)). This interface aims to restore the state of models to a storage state, where *SimuState* is a reference to a string parameter which contains the state data of the model related to the current simulation time. If the interface is called first, it means that *SimuState* represents the initial state data.

(2) Input Interface (int input(InputDataTypeX inputData)). This interface aims to provide input data for this model, where *InputDataTypeX(X=0, 1, 2 . . .)* is a user defined data type and *inputData* contains data of all the input variables needed to be assigned. Before calling this interface, the simulation entity applies enough memory for *inputData* to save data derived from another model output. The simulation entity usually needs to call this interface more than once for different input variables assignment because of various *InputDataTypeX*.

(3) Dynamic Data Driven Interface (int driven(Dynamic DataTypeX dynamicData)). This interface aims to provide dynamic data for this model, where *DynamicDataTypeX (X=0, 1, 2, . . .)* is a user defined data type and *dynamicData* contains data of all the dynamic input variables needed to be assigned. We design this interface to support that a model usually needs to receive some unexpected data to amend its behavior under execution, such as adjusting radar cross-section or length of antennas for a radar model. Before calling this interface, the simulation entity applies enough memory for *dynamicData* to store data derived from outside environment. The simulation entity usually needs to call this interface more than once for different dynamic input variables assignment.

(4) Business Process Interface (int process(double dSimuTime)). This aims to call internal transmission functions provided by the simulation computational module to implement the business logic, where *dSimuTime* refers to the current simulation logical time.

(5) Output Interface (int output(OutputDataType & ouputData)). This interface aims to get output data from this model, where *outputDataTypeX(X=0, 1, 2,. . .)* is a user defined structure type and *ouputData* contains all of the output variables. Before calling this interface, the simulation entity needs to define a variable of *OutputDatatTypeX*. The simulation entity usually needs to call this interface more than once to get different output variables because of various *outputDataTypeX*.

(6) State Getting Interface (int getstate(string & simuState)). This interface aims to get the current state of simulation model, which will be stored in a string parameter, where *simuState* is a reference to a string parameter defined in the external simulation entity. After the component model execution finished, the simulation entity will use *simuState* as a parameter to call state restoring interface, where the state data of this model will be stored into parameter *simuState*.

After encapsulation finished, component models can be tested and executed in other projects. In order to facilitate component models reused across various products and product families, interface parameters of models will be added into model's informal descriptions, which with "domain," "purpose," and "usage time" together will be written into a model description file using XML technology to help users understanding how the component model works; see Table 1.

Table 1: Informal description of a computational module.

Domain	Application domains should be considered when characterizing the context for a component model, for example, hydromechanics, aerodynamics, mechanical control, and so on.
Purpose	Component models can be considered as replaceable building blocks of application. This includes the problem that the model solves.
Usage time	Usage time is a characteristic of a component that reflects its stability and maturity. When a component is first introduced, there is a high risk associated with its usage.
Parameter	This records the parameter description of the internal state transmission function, including name, type, unit, identification, and function, especially for user defined parameter types.
Sequence	The logical behavior of this simulation computational module is implemented though scheduling internal state transmission function according to correct sequence.

5. Case Study: A Reusable Radar Model

Model reusability appears to be a topic of notable interest in the simulation research community and in selected application areas [27], especially in military simulation. Radar was originally developed to satisfy the needs of the military for surveillance and weapon control [28]. In this section, we take the radar model under a complex electromagnetic environment, for example, to test and verify our reusable model development approach.

5.1. Create Simulation Computational Module. The basic parts of a radar model in complex system simulation

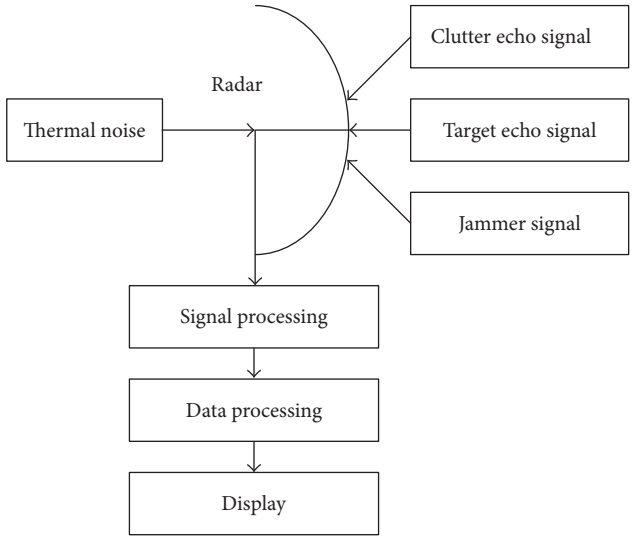


Figure 7: A radar model detection process.

application are illustrated in the simple block diagram of Figure 7. The radar signal is generated by the transmitter and radiated into space by the antenna. Reflecting objects (targets) intercept and reradiate a portion of the radar signal, a small amount of which is returned in the direction of the radar. The returned echo signals which contain target echo signal, clutter echo signal, and interfering signal under complex electromagnetic environment are collected by the radar antenna. The thermal noise of radar itself will also be considered. According to the radar equation, after signal processing and data processing, we compute statistical probability of radar to detect a target. If the output of the radar receiver is sufficiently larger than a random variable, detection of a target is said to occur [29].

(1) The radar equation gives the range of a radar in terms of the radar characteristics, one form of which gives the received signal power as [29]

$$P_{rs} = \frac{P_t G_t}{4\pi R^2} \times \frac{\sigma}{4\pi R^2} \times \frac{G_r \lambda^2}{4\pi L}. \quad (1)$$

The received signal powers are influenced by three factors to represent the physical process. The first factor is the power density at distance R meters from radar that radiates a peak power of P_t watts from an antenna of transmission gain G_t. The value of the second factor is the target cross-section σ in square meters. The third parts contain antenna reception gain G_r, radar wavelength λ, and radar comprehensive loss L.

(2) The receiver noise power generated by radar is described as [29]

$$P_n = kTB_r F_n. \quad (2)$$

The thermal noise is equal to kTB_r, where k is *Boltzmann's* constant, $T = 290$ K is the temperature (approximately room temperature), and B_r is receiver bandwidth. The receiver noise is the thermal noise multiplied by the factor F_n.

(3) When the detection of the radar signal is influenced by an external noise source, such as a deliberate noise jammer, the receiver noise power is now that determined by the jammer (radar active interfering) rather than the receiver noise [30]

$$P_{rj} = \frac{P_t G_t}{4\pi R_j^2} \times \frac{B_r \lambda^2}{4\pi R_j^2} \times \frac{1}{B_j L_j}, \quad (3)$$

where R_j = jammer range from radar, B_j = jammer bandwidth, P_j = jammer power, G_j = jammer antenna gain, L_j = jammer comprehensive loss.

(4) The interfering unwanted clutter echoes can severely limit the detect ability of the target, such as chaff interfering signal in military simulation. So the receiver passive interfering signal power is described as [31]

$$P_{rc} = \frac{P_t G_t G_r \lambda^2 \sigma_c}{(4\pi)^3 R_c^4 L}, \quad (4)$$

where G_r is chaff antenna gain, σ_c is chaff cross-section, and R_c is chaff range from radar.

(5) The minimum detectable signal can be expressed as the signal-to-noise ratio required for reliable detection times the receiver noise [32]

$$\text{SNR} = \frac{P_{rs} D_j}{P_{rj} + P_{rc} + P_n}. \quad (5)$$

So in complex electromagnetic environment, the probability of radar to detect target can be written as

$$P_d = \left(\frac{\eta \text{SNR} + 1}{\eta \text{SNR}} \right)^{\eta - 1} P_f^{1/(1+\text{SNR})}, \quad (6)$$

where D_j is radar anti-interference factor, η is number of pulses, and P_f is the probability of a false alarm.

(6) Because radar equation generates statistical probability of detection, it needs a random variable to make decision. Supposing μ between $[0, 1]$ is a random variable generated through Monte Carlo, if $\mu \leq P_d$, we said that the target has been found.

We analyze the above mathematical formulas and transform them to the state transmission functions of the radar module in Table 2. According to parameter sources, we divided them into two categories. One is initialization parameter which is derived from the radar configure file, while the other is input parameter, which is derived from the output of other models.

5.2. Encapsulate with Service Interfaces. According to the state transmission functions of radar module in Table 2, *ParseConfigure* provides access to radar configured file for external simulation entity. The dynamic data driven interfaces *driven (JammerStruct jammer)* and *driven (ChaffStruct chaff)* provide dynamic data for functions *ComputeJammerPower* and *ComputeChaffPower*. The input interface *input(TargetStruct target)* provides target data for function ComputeEchoPower. The business process interface will call *ComputeEchoPower*, *ComputeJammerPower*, *ComputeChaffPowe*, *ComputeProbability,* and *ComputeDetected* to implement behavior logic of the radar model according to the

Table 2: Functions of radar computational module.

Function	Parameter	User defined type	Value
ParseConfigure	$P_t, G_t, \lambda, B_r, F_n, D_j, \eta, P_f$	*RadarStruct*	Radar configure file
ComputeEchoPower	σ, R	*TargetStruct*	Target output
ComputeJammerPower	R_j, B_j, P_j, G_j, L_j	*JammerStruct*	External dynamic data
ComputeChaffPower	B_r, σ_c, R_c	*ChaffStruct*	External dynamic data
ComputeProbability	D_j, η, P_f	—	Radar configure file
ComputeDetected	*Detected*	—	Radar output

sequence recoding in the module informal descriptions. State restoring interface and state getting interface are utilized to implement state saving and restoring that make the component model used in simulation applications with optimism mechanism. Use state getting interface to get value of parameter *detected* that represents whether the target has been found or not. The execution flow of the radar reusable component model is described in the following; see Figure 8.

To illustrate the code structure of our reusable component model framework, we use C++ programming language for example to implement the radar component model shown in Algorithm 1.

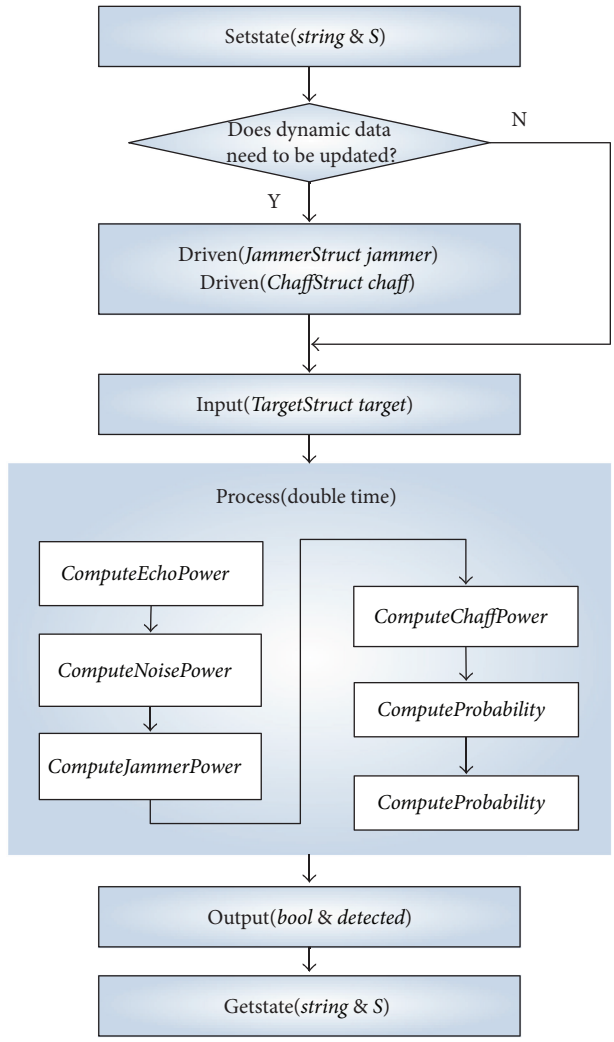


Figure 8: Reusable radar model execution flow.

6. Results and Discussion

Air-defense and antimissile of naval vessel is a typical scene in military simulation application [33, 34]. It usually contains three kinds of simulation entities: naval vessel, battle plane, and guided missile. The procedure of air-defense and antimissile of naval vessel is that, firstly, the battle plane opens its radar to search targets (naval vessel). Once it finds an enemy naval vessel, the battle plane will launch a guided missile to attack it; secondly, the guided missile flies to the enemy naval vessel under the control, and in the end of flight, it will opens its radar to locate the target by itself; finally, the naval vessel detects the guided missile using its radar and then uses jammer and interceptor missile for antimissile and air-defense.

Figure 9 shows the discrete event simulation application of air-defense and antimissile of naval vessel built with our visual modeling tool. A rectangular block represents a simulation entity and a small picture within the rectangular block represents a model in the right editing area. The component models, such as radar model, jammer model, and interceptor missile model, can be dragged and dropped from model resource panel on the left to construct simulation entities. They are connected with bidirectional active lines to build the air-defense and antimissile of naval vessel simulation application. Comparing with more than one hour cost through writing code manually, we only take less than 10 minutes to finish it using the visual modeling tool shown in Figure 10. It is only needed to move the mouse and the application code will be generated automatic in the background, thus significantly reducing development time.

We can see that each simulation entity, the naval vessel, the battle plane, or the guided missile, contains a radar component model. Although there are discrepancies between the radar component models in different simulation entities, maybe different wavelength, different transmission gain, or different interface parameters, it only needs to modify the configured file or adjust the external parameters to satisfy different demands rather than developing different models. For instance, while comparing the radar model in the guided missile within the naval vessel, the former has jammer and chaff data as parameters in its execution, but the latter has not. In this situation, the radar model in naval vessel will not

```
Class CRadarModel
{
    public:  //External six standard service interfaces
             int setstate(string& simuState);
             int input(TargetStruct& target);
             int driven(JammerStruct& jammer);
             int driven(ChaffStruct& chaff);
             int proces(double dSimuTime);
             int output(bool& detected);
             int getstate(string& simuState);

    private: //Internal transmit functions
             void ParseConfigure();
             void ComputeEchoPower(TargetStruct& target);
             void ComputeJammerPower(JammerStruct& jammer);
             void ComputeChaffPower(chaffStruct& chaf);
             void ComputeProbability();
             void ComputeDetected();

             //Internal status variables
             RadarStruct radar;
             ...
};
```

Algorithm 1: Reusable radar model code structure.

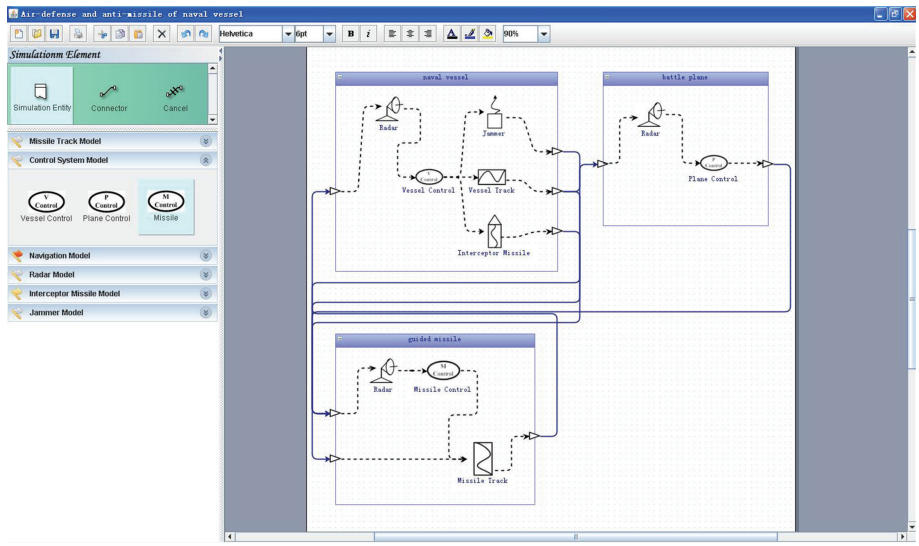


Figure 9: Snapshot of air-defense and antimissile of naval vessel simulation application.

call driven interface, and that is why we call it dynamic data driven interface. We can also assign different values for radar transmission gain or radar wavelength in the configure file to achieve suitable radars for the three kinds of simulation entities. Thus the same model can be used in different simulation entities that illustrates that the models developed using our approach have good reusability.

Moreover, the radar model has been used in a training simulation system based on HLA [35, 36]. Figure 11 shows the architecture of the training simulation system. The airborne radar federate and the ground radar are the detection radars which are used to detect targets in two orientations. The radar countermeasure federate searches and parses the detection radar signal under complex electromagnetic environment. Based on the detection radar signal information, the radar jammer federate produces the external noise to interfere the two detection radar federates. The display federate, the data process federate, the effect evaluation federate, and the scene simulation federate are used to process data, display detecting result, and interfering effect.

In the radar countermeasure training simulation system, the airborne radar federate and the ground radar federate use

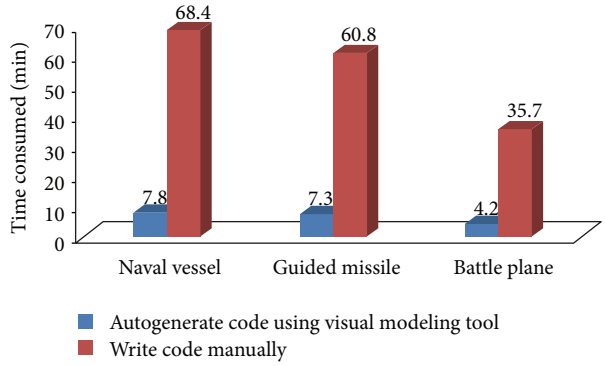


Figure 10: Time consumed of two approaches for developing three simulation entities.

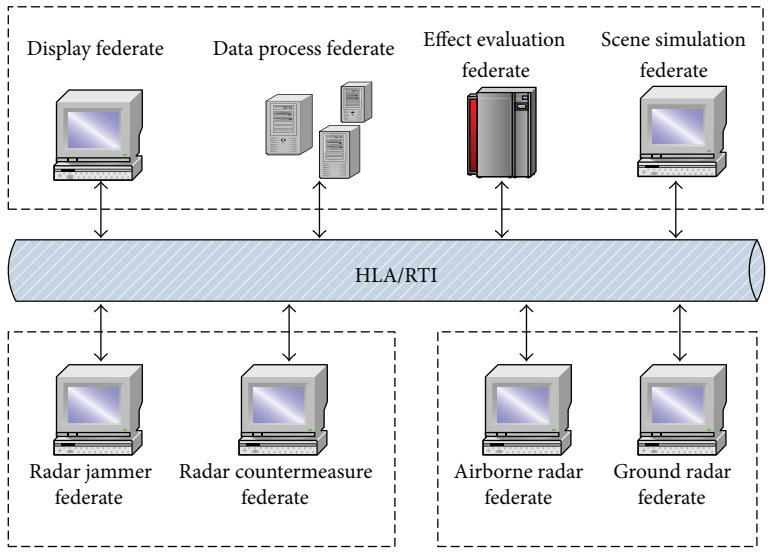


Figure 11: The architecture of a radar training simulation system.

the same radar model to implement their functionality. The practical project illustrates that the radar model is also very easy to be assembled in *HLA*-based distributed simulation system.

7. Conclusions and Future Work

Model reusability is very important for parallel and distributed simulation application development, not only for minimizing engineering efforts and resource costs, but also for improving the reliability. Current model development methods face several obstacles hindering model reuse across simulation platforms and simulation applications, such as diversiform interfaces, coupled tightly, and bound together with simulation platforms. To solve these problems, in this paper, we first presented a framework to define what we mean by "reusable" component model and then proposed a two-phase model development approach. In the first phase, the domain modeler builds a computational module that is independent with other models and simulation platforms. In the second phase, the model developer encapsulates the module with the framework as a model specification. At last, we tested our reusable component model development approach using a radar model and its application of air-defense and antimissile of naval vessel. Our method has been successfully applied in several projects. The case study and these practical applications illustrate that the simulation models created with our method have good reusability and facilitation to be assembled. They can also be used in different simulation platforms and applications.

As for our future work, we plan to study the algorithm for component models selection to suggest how to select component model to fulfill the requirements of different clients.

Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors appreciate the support from National Natural Science Foundation of China (no. 61170048) and Research

Project of State Key Laboratory of High Performance Computing of National University of Defense Technology (no. 201303-05).

References

[1] R. M. Fujimoto, "Parallel and distributed simulation," in *Proceedings of the Winter Simulation Conference Proceedings (WSC '99)*, pp. 122–131, December 1999.

[2] M. A. Hofmann, "Challenges of model interoperation in military simulations," *Simulation*, vol. 80, no. 12, pp. 659–667, 2004.

[3] R. Sandor and N. Fodor, "Simulation of soil temperature dynamics with models using different concepts," *The Scientific World Journal*, vol. 2012, Article ID 590287, 8 pages, 2012.

[4] A. Pos, P. Borst, J. Top, and H. Akkermans, "Reusability of simulation models," *Knowledge-Based Systems*, vol. 9, no. 2, pp. 119–125, 1996.

[5] N. B. Gill, "Importance of software component characterization for better software reusability," *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 1, pp. 1–3, 2006.

[6] D. Chen, S. J. Turner, W. T. Cai, and M. Z. Xiong, "A decoupled federate architecture for high level architecture-based distributed simulation," *Journal of Parallel and Distributed Computing*, vol. 68, no. 11, pp. 1487–1503, 2008.

[7] O. Balci, A. I. Bertelrud, C. M. Esterbrook, and R. E. Nance, "Visual simulation environment," in *Proceedings of the Winter Simulation Conference (WSC '98)*, pp. 279–287, December 1998.

[8] R. Muetzelfeldt and J. Massheder, "The Simile visual modelling environment," *European Journal of Agronomy*, vol. 18, no. 3-4, pp. 345–358, 2003.

[9] X. Hu, B. P. Zeigler, and S. Mittal, "Variable structure in DEVS component-based modeling and simulation," *Simulation*, vol. 81, no. 2, pp. 91–102, 2005.

[10] S. Robinson, R. E. Nance, R. J. Paul, M. Pidd, and S. J. E. Taylor, "Simulation model reuse: definitions, benefits and obstacles," *Simulation Modelling Practice and Theory*, vol. 12, no. 7-8, pp. 479–494, 2004.

[11] B. Allan, R. Armstrong, F. Bertrand et al., "A component architecture for high-performance scientific computing," *International Journal of High Performance Computing Applications*, vol. 20, no. 2, pp. 163–202, 2006.

[12] E. E. L. Mitchell and J. S. Gauthier, "Advanced continuous simulation language (ACSL)," *Simulation*, vol. 26, no. 3, pp. 72–78, 1976.

[13] S. W. Cox, "GPSS World: a brief preview," in *Proceedings of the Winter Simulation Conference Proceedings*, pp. 59–61, December 1991.

[14] S. E. Mattsson, H. Elmqvist, and M. Otter, "Physical system modeling with Modelica," *Control Engineering Practice*, vol. 6, no. 4, pp. 501–510, 1998.

[15] Y. L. Lei, W. P. Wang, Q. Li, and Y. F. Zhu, "A transformation model from DEVS to smp2 based on MDA," *Simulation Modelling Practice and Theory*, vol. 17, no. 10, pp. 1690–1170, 2009.

[16] C. Koo, H. Lee, and Y. Cheon, "SMI compatible simulation scheduler design for reuse of model complying with SMP standard," *Journal of Astronomy and Space Sciences*, vol. 27, no. 4, pp. 407–412, 2010.

[17] B. P. Zeigler, "DEVS today: recent advances in discrete event-based information technology," in *Processdings of the 11TH IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS '03)*, pp. 148–161, 2003.

[18] SISC, *IEEE Standard for Modeling and Simulation High Level Architecture(HLA)-Framework and Rules*, 2000.

[19] Y.-P. Yao and Y.-X. Zhang, "Solution for analytic simulation based on parallel processing," *Journal of System Simulation*, vol. 20, no. 24, pp. 6617–6621, 2008.

[20] T. L. Wilmarth and L. V. Kalé, "POSE: getting over grainsize in parallel discrete event simulation," in *Proceedings of the International Conference on Parallel Processing (ICPP '04)*, pp. 12–19, August 2004.

[21] K. R. Bisset, A. M. Aji, E. Bohm et al., "Simulating the Spread of Infectious Disease over Large Realistic Social Networks Using Charm++," in *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium WorkShop*, pp. 507–518, 2012.

[22] O. M. Ulgen, T. Thomasma, and N. Otto, "Reusable models: making your models more user-friendly," in *Proceedings of the Winter Simulation Conference Proceedings*, pp. 148–151, December 1991.

[23] G. Gossler and J. Sifakis, "Composition for component-based modeling," *Science of Computer Programming*, vol. 55, no. 1–3, pp. 161–183, 2005.

[24] G. T. Mackulak, F. P. Lawrence, and T. Colvin, "Effective simulation model reuse: a case study for AMHS modeling," in *Proceedings of the Winter Simulation Conference (WSC '98)*, pp. 979–984, December 1998.

[25] W. Pree, "Component-based software development-a new paradigm in software engineering," in *Proceedings of the Software Engineering Conference , Asia Pacific and International Computer Science Conference (APSEC '97) and (ICSC '97)*, pp. 523–524, 1997.

[26] E. Page and J. Opper, "Observations on the complexity of composable simulation," in *Proceedings of the Winter Si-mulation Conference*, pp. 553–560, 1999.

[27] M. Pieraccini, "Monitoring of civil infrstructures by interferometric radar: a review," *The Scientific World Journal*, vol. 2013, Article ID 786961, 8 pages, 2013.

[28] S. Robinson, "Modes of simulation practice: approaches to business and military simulation," *Simulation Modelling Practice and Theory*, vol. 10, no. 8, pp. 513–523, 2002.

[29] M. I. Skolnik, *Radar Handbook*, McGraw-Hill, New York, NY, USA, 1970.

[30] C. Gong and D. Wei, "Radar detection probability model under complex electromanetic environment," *Automatic Measurement and Control*, vol. 26, no. 3, pp. 78–81, 2007.

[31] Z. He, F. Zhao, B. Zhao, Z. Liu, and X. Wang, "Research on computation model of radar target detection probability with noise interference," *Modern Defense Technology*, vol. 40, no. 1, pp. 119–124, 2012.

[32] J. Hao and B. Gan, "An air defense radar detection model in complex electromagnetic environment," *Computer Simulation*, vol. 26, no. 6, pp. 33–37, 2009.

[33] Z. X. Xiong, Y. K. Sun, C. S. Jiang, and D. S. Liu, "Implement on simulation of air-defense and anti-missile on naval vessel based on STAGE," *Journal of System Simulation*, vol. 16, no. 6, pp. 1358–1360, 2004.

[34] C. P. Pan, W. J. Gu, J. Cheng, and H. C. Zhao, "Design and accomplishment of diturbedf anti-warship missile attack-defence combat simulation system," *Journal of Projectiles, Rockets, Missiles and Guidance*, vol. 25, no. 3, pp. 277–279, 2005.

[35] W. G. Wang, Y. P. Xu, X. Chen, Q. Li, and W. P. Wang, "High level architecture evolved modular federation object model," *Journal of Systems Engineering and Electronics*, vol. 20, no. 3, pp. 625–635, 2009.

[36] J. J. Shi and B. Lin, "Design on radar countermeasure training simulation system based on HLA architecture," *Aerospace Electronic Warfare*, vol. 29, no. 2, pp. 61–65, 2013.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 491549, 11 pages
http://dx.doi.org/10.1155/2014/491549

Research Article

A Master-Slave Surveillance System to Acquire Panoramic and Multiscale Videos

Yu Liu, Shiming Lai, Chenglin Zuo, Hao Shi, and Maojun Zhang

Department of System Engineering, College of Information System and Management, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Yu Liu; jasonyuliu@hotmail.com

Received 9 November 2013; Accepted 24 December 2013; Published 3 March 2014

Academic Editors: S. K. Bhatia and K. K. Mishra

This paper describes a master-slave visual surveillance system that uses stationary-dynamic camera assemblies to achieve wide field of view and selective focus of interest. In this system, the fish-eye panoramic camera is capable of monitoring a large area, and the PTZ dome camera has high mobility and zoom ability. In order to achieve the precise interaction, preprocessing spatial calibration between these two cameras is required. This paper introduces a novel calibration approach to automatically calculate a transformation matrix model between two coordinate systems by matching feature points. In addition, a distortion correction method based on Midpoint Circle Algorithm is proposed to handle obvious horizontal distortion in the captured panoramic image. Experimental results using realistic scenes have demonstrated the efficiency and applicability of the system with real-time surveillance.

1. Introduction

Digital video surveillance has become commonly used in public and private places such as government buildings, military bases, car parks, and banks, and so forth. Traditional monitoring cameras can only cover a limited area, leading to "blind spots." Developments in panoramic imaging technology offer significant advantages over traditional surveillance systems. They can monitor an area that covers 180° or 360° and so replace several traditional cameras. Nevertheless, images captured by panoramic cameras have limited range of scale due to the relatively reduced resolution. Comparatively PTZ dome cameras can focus on areas of interest rapidly by decreasing or increasing focal length. The master-slave camera composed of a fish-eye panoramic camera and PTZ dome camera combines the advantages of both (Figure 1). The fish-eye panoramic camera is responsible for acquiring global and wide images in the large surveillance area, and the PTZ dome camera is used to acquire multiscale videos for more detailed information.

Figure 2(b) demonstrates the internal structure of the proposed system, where a panoramic camera acts as master camera and is mounted next above the traditional PTZ dome camera (Figure 2(a)). Upgrading from existing surveillance systems to a master-slave system is simple (shown in Figure 2(c)). This system can automatically direct a slave PTZ dome camera(s) to zoom into target areas of interest, in which details of object appearance are available at a higher resolution.

Different master-slave camera systems use various compositions of PTZ dome cameras and mixtures of other types of camera. Regardless of how the system is composed of camera types, the critical technique is to develop a suitable calibration algorithm for accurate interaction between the cameras. Through calibration, we hope that the target appointed by master camera can steer the slave camera to focus on the same position at the pixel level. The simplest and most direct way to precisely calibrate two cameras is to manually find every pixel in an image captured by one camera and correspond these with pixels in an image captured using another camera. Dense mapping such as this is impractical and it seriously limits the applicability. In practice, fewer points are required, which can be interpolated within some degree of accuracy. However, often the level of accuracy is not unacceptable. It has been shown experimentally that calibration based on 200 sample points may take several

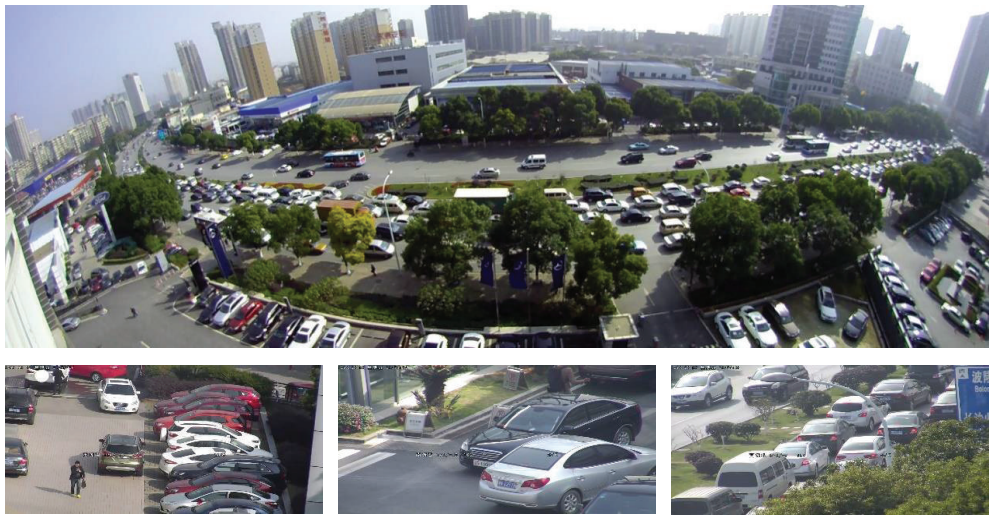


Figure 1: The fish-eye panoramic image captured by the master camera and zoom-in image captured by slave camera with distant details.

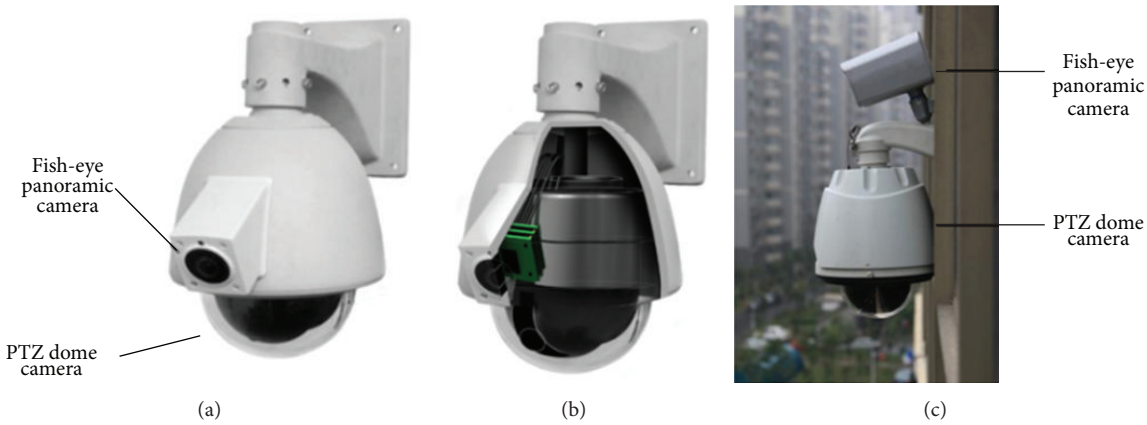


Figure 2: (a) The master-slave visual surveillance system. (b) The internal structure of the proposed system. (c) Most existing surveillance systems can be upgraded to the master-slave system by adding a fish-eye camera above the traditional PTZ camera, or adding a PTZ dome camera underneath the panoramic camera.

hours. Hence, a practical calibration method is required for master-slave camera systems.

To ensure the panoramic camera captures the same scene content that the PTZ camera covers, the fish-eye camera is fixed inclining towards the gravity direction (Figures 3(a)–3(c)). However, this design brings serious image distortion in the horizontal direction (Figure 3(d)). So, an additional challenge for the proposed system is to find a suitable image correction to handle these distortions.

Our contributions over existing competing systems are twofold: (1) in terms of camera calibration, an efficient and accurate calibration method is proposed to accomplish the calibration between stationary and dynamic cameras. This method does not require specific camera setup or a particular grid pattern; (2) in terms of fish-eye distortion correction, our technique correctly handles the particular type of distortion introduced in fish-eye panoramic images. By adjusting the values of interrelated parameters, the extent of the distortion can be controlled. Moreover, the proposed algorithm can be applied to an embedded camera platform without any extra hardware resources due to its low computational cost.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces the calibration method between panoramic and PTZ dome cameras. Section 4 describes the proposed distortion correction algorithm. In Section 5, experiments are implemented and the experimental results are shown. Finally, Section 6 concludes this paper.

2. Related Work

As mentioned, the main challenge in the application of the proposed system is to actively control a PTZ dome camera to correctly focus on the same target in the panoramic scene. The precision of this interaction largely depends on the accuracy of the spatial calibration, which can be considered as the mapping between each of pixels in fish-eye panoramic image and the pan-tilt angles of PTZ dome camera.

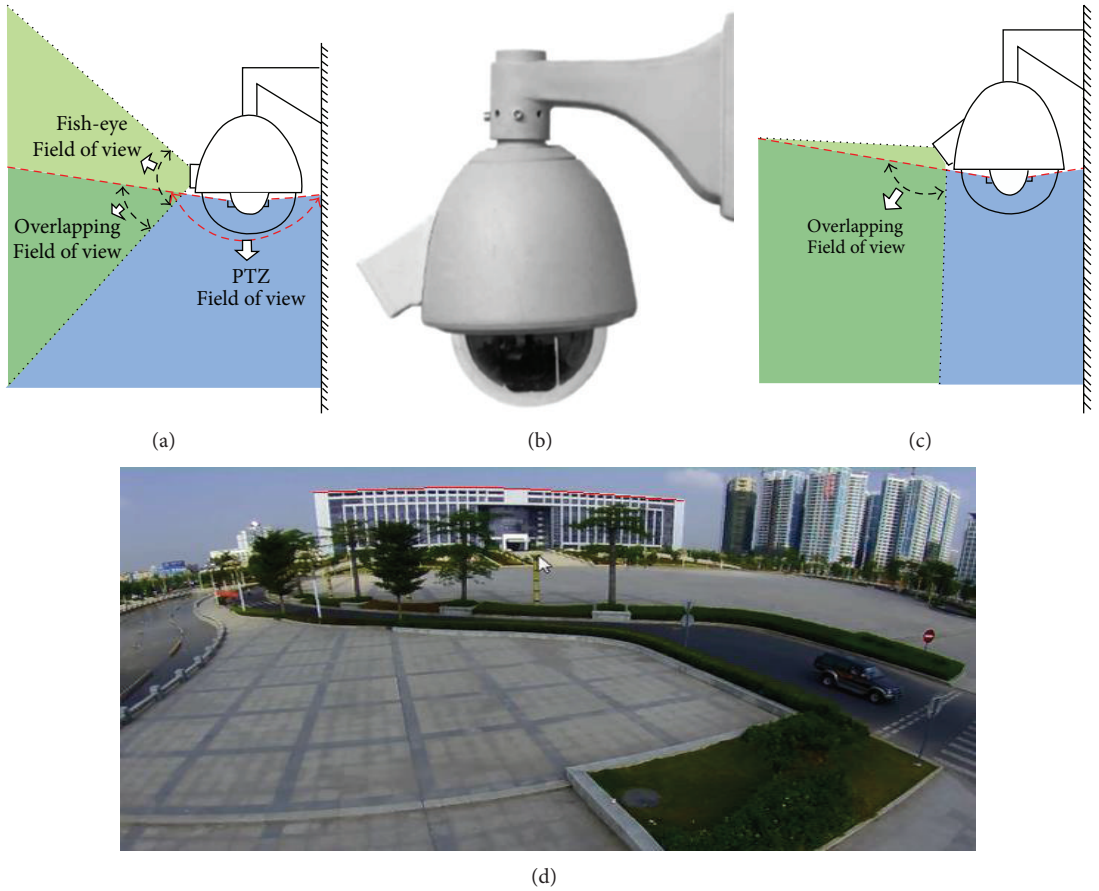


Figure 3: (a) If the fish-eye camera is mounted parallel to front, two cameras only have small overlapping FOV (b) The side view of master-slave visual surveillance system (c) The inclining angle ensures that the PTZ dome camera covers the FOV of fish-eye camera. (d) Image distortion appeared on the panoramic camera marked by the red curve line.

A practical calibration method should not only be efficient and effective but also need no particular system setup or human intervention. For the proposed system, it also has a problem of horizontal distortion when the master camera is mounted with an angle towards the gravity direction. Here we start with reviewing the calibration methods and approaches used in maser-slave camera system.

Current calibration methods can be divided into two main categories: geometry calibration and data fitting calibration. Geometry is a calibration method through single camera calibration and dual camera joint calibration to obtain the mapping relationship. However, it requires a priori knowledge about camera imaging model and geometric environment. Sato et al. [1] present an indoor monitoring system with multiple camera units, which includes panoramic camera and PTZ camera. They calculated the PTZ camera rotation angle corresponding to the point $P(P_x, P_y)$ on the panoramic image using the following equation:

$$\theta_P = 360 \times \frac{P_x}{W + E_x},$$
$$\theta_T = \left(|OmniTop| + |OmniBottom|\right) \times \frac{P_y}{H + E_y}, \quad (1)$$

where E_x and E_y are the distances between panoramic camera and PTZ camera in the horizontal and vertical direction respectively. *OmniTop* is the height from PTZ camera to the ground, and *OmniBottom* is the height from PTZ camera's bottom to the ground; H and W are the height and width of panoramic image. This simple method has large error because PTZ rotation angle is calculated through the position of point P in the panoramic image and relative shift of PTZ rotation angle. Scotti et al. [2] employed the master-slave camera in which the optical axis of catadioptric panoramic camera coincides with the horizontal axis of PTZ camera. They approximately assumed that the optical centers of two cameras overlap with each other due to the close installation. As the polar angle of the pixel in the panoramic image equals the corresponding horizontal rotation angle of PTZ camera, it simplifies the calculation greatly. It also reduces the error of the horizontal rotation angle which is brought by the change of the optical center with PTZ's movement. However, the method assumes that objects in the scene should be located on the same space ground. This hypothesis is invalid for most practical circumstance. The distance measurement between PTZ camera and the ground is required to be manually set, which also limits the installation procedure.

Geometry calibration generally needs to know two priories and satisfy a hypothesis, so the mapping relationship largely depends on both the accuracy of the priori and the validity of the hypothesis. While, the data fitting models the relationship between the panoramic coordinates and PTZ rotation angle by fitting the sample points. Both the camera imaging model and relative position of two cameras can be ignored, so this type of methods is more flexible. Hampapur et al. [3] triangulated a position by two or more calibrated cameras and determine the steering parameters for a third PTZ camera which is also calibrated. Chen et al. [4] proposed a versatile method for a variety of cameras. However, their method is at the cost of reducing the accuracy. They sampled the pixel coordinates (x, y) in the panoramic image and the correspondent PTZ rotation angle and hope to find the best fitting polynomial to describe their relationship. Tan [5] proposed a mapping method based on image piecewise fitting to obtain an improvement. Different polynomials were used according to the distortion degree in different areas. Nevertheless, the accuracy does not meet requirements. Senior et al. [6] used a master-slave camera that is composed of a fixed super wide-angled camera and a PTZ camera. They selected several sample points in the FOV of super wide-angled camera and determined nonsample point mapping relationship by a linear interpolation. Zhou et al. [7] selected a number of pixel locations in a static camera. For each pixel, manually move the slave camera to center the slave image and record the corresponding slave pan-tilt angles to obtain a lookup table. It links the static camera coordinates with the pan and tilt angles. Although their method is accurate enough to initialize the track of dynamic camera, it is time consuming and inconvenient. You et al. [8] employed a mosaic image created by snapshots of slave camera to estimate the relationship between static master camera plane and pan-tilt controls of slave camera. Compared with other approaches, this solution provides an efficient and automatic way to calibration of a master-slave system. Nevertheless, the mapping determined by a liner interpolation is inaccurate.

In terms of fish-eye panoramic image distortion correction, Devernay and Faugeras [9] assumed the presence of straight lines in the scene. Distortion parameters are sought which lead to lines being imaged as straight in the corrected image. Kannala and Brandt [10] proposed a novel calibration method for fish-eye lens cameras that was based on viewing a planar calibration pattern. This method was proven suitable for different kinds of omnidirectional cameras as well as for conventional cameras. Wang et al. [11] presented a new model of camera lens distortion that utilized two angular parameters and two linear parameters. These parameters were used to determine the transform from an ideal plane to real sensor array plane, which governs the lens distortion. Yu [12] proposed a lens geometric and photometric distortion correction method to obtain a high quality image. By using a simplified camera calibration technique, lens geometric coefficient can be estimated. Photometric distortion was corrected using a nonlinear model fitting of a proposed photometric distortion model function. Ying et al. [13] used spherical perspective projection model to calibrate the fish-eye lenses. Based on straight line spherical perspective projection constraint, the mapping between a fish-eye image and its corresponding spherical perspective image was determined. Once the mapping is obtained, the fish-eye lenses can be calibrated. Since orthographic spherical perspective projection was employed, these algorithms can only be applied to orthographic fish-eye cameras but not for equidistant fish-eye cameras [14, 15]. Li et al. [16] presented an embedded real-time fish-eye image distortion correction algorithm, which can be applied in an IP network camera. However, this algorithm only aimed to correct the distortion in the vertical direction. Moreover, methods that adapted the projection to content in the scene were also presented [17–19]. However, these methods require human intervention, and the corrected image has to be cropped.

Most previous distortion correction research focuses on constructing and calculating the internal reference model, which can express the mapping between the three-dimensional world and the two-dimensional image. Based on the internal reference model, the distorted image is mapped onto a three-dimensional spherical surface or parabolic surface. By using perspective projection, the distortion can be corrected. However, these methods aim to correct the distortion of conventional fish-eye panoramic images, and few methods have been proposed to correct the particular type of distortion in the fish-eye panoramic image captured by a master-slave camera. The proposed system draws inspiration from Midpoint Circle Algorithm (MCA) [16] and applies this algorithm to correct these distortions.

3. Calibration

Through a specific calibration, the target appointed by master camera can steer the slave camera to focus on the same position. To achieve this goal, the core technique is to determine the geometric relationship between the master camera image pixel coordinates and the pan-tilt angles of the slave camera.

3.1. Analysis of Coordinate Systems. The basis of calibration is to establish three coordinate systems: a panoramic coordinate system, a PTZ coordinate system, and a spherical coordinate system. As a master-slave camera is composed of a fish-eye panoramic camera and a PTZ dome camera, the panoramic coordinate system is a fish-eye coordinate system based on the fish-eye panoramic images, the PTZ coordinate system is a coordinate system based on PTZ camera taking pan angle and tilt angle as parameters, and the spherical coordinate system is an auxiliary coordinate system transforming from the panoramic coordinate system to the PTZ coordinate system.

Figure 4(a) illustrates the panoramic coordinate system. The x-axis and the y-axis represent the horizontal and vertical directions, respectively. The optical center is O, and OO' is the optical axis. W and H are the width and height of the panoramic image respectively. $AngleW$ and $AngleH$ are, the horizontal and vertical angle of view of the panoramic image, respectively. The focal length of fish-eye panoramic camera C_f can be calculated by $W/AngleW$.

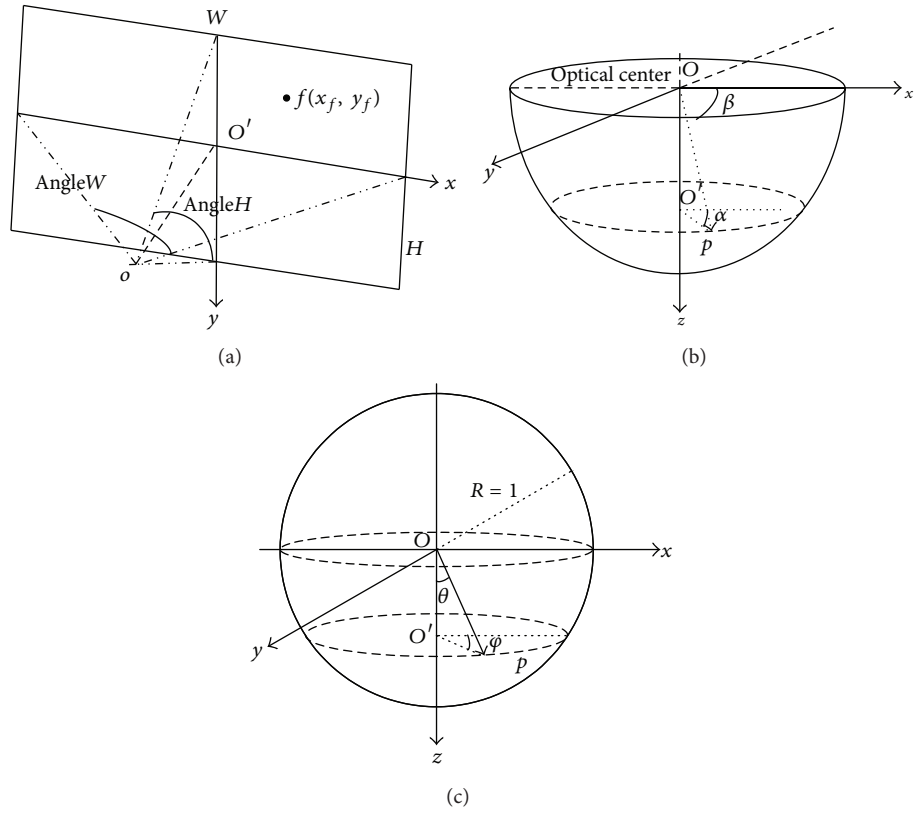


Figure 4: (a) Panoramic coordinate system, (b) PTZ camera coordinate system, and (c) PTZ image coordinate system.

Here we set (x_f, y_f) as the pixel positions in the panoramic image. As shown in Figure 4(b), the PTZ coordinate system contains two parameters. p is a point on the surface of sphere. α represents the pan angle between O'_p and positive x-axis. It increases in the anticlockwise direction viewing from positive z-axis, which ranges from 0° to 359°. β is the tilt angle that ranges from 0° to 89°. It is the angle between O_p and XOY plane which increases in the clockwise direction viewing from positive x-axis. The PTZ coordinate system is defined within a hemisphere on the XOY plane with the z-axis pointing downwards.

Figure 4(c) illustrates the unit spherical coordinate system. Starting point O and every axis correspond to the PTZ dome coordinate system. The coordinate of point p is denoted by (x_s, y_s, z_s). φ is the angle between O'_p and positive x-axis which increases in the anticlockwise direction viewing from positive z-axis and ranges from 0° to 359°. θ is the angle between O_p and positive z-axis which increases in the anticlockwise direction viewing from positive x-axis and ranges from 0° to 89°.

3.2. Transformation between Coordinate Systems. When the transformation from the panoramic coordinate system to the PTZ coordinate system is obtained, we can determine the mapping relationship between each of pixels in the image captured by the fish-eye camera and image captured by the PTZ camera. However, the transformation cannot be calculated in a single step directly. We need to use spherical coordinate system to link with these two coordinate systems. The process is completed in three steps. Firstly, assume $(x_f, y_f, 1)$ to be any pixel with homogeneous coordinates in the panoramic coordinate system with a corresponding spherical homogeneous coordinate $(x_{Ps}, y_{Ps}, z_{Ps}, 1)$. Secondly, set the mapped homogeneous coordinate to be $(x_P, y_P, z_P, 1)$. In the PTZ coordinate system, its corresponding spherical homogeneous coordinate is $(x_{Hs}, y_{Hs}, z_{Hs}, 1)$. Last, establishing a mapping relationship between $(x_{Ps}, y_{Ps}, z_{Ps}, 1)$ and $(x_{Hs}, y_{Hs}, z_{Hs}, 1)$ as follows:

$$(x_{Hs}, y_{Hs}, z_{Hs}, 1) = (x_{Ps}, y_{Ps}, z_{Ps}, 1) \times T_{PH}. \quad (2)$$

T_{PH} is a 4 × 4 matrix which represents the transformation from a panoramic spherical coordinate to PTZ dome spherical coordinate. Before the transformation, there are two constraints for the master-slave camera system. Firstly, the camera's optical axis is perpendicular to the image plane and point of intersection is at the center of the image plane. Secondly, the fish-eye panoramic camera's optical center and the PTZ dome camera's optical center are in the same vertical plane. If above two constraints are not satisfied, great errors may result. The mapping T is obtained through a process of transformation among coordinates:

(1) Transformation from panoramic coordinate system to spherical coordinate system.

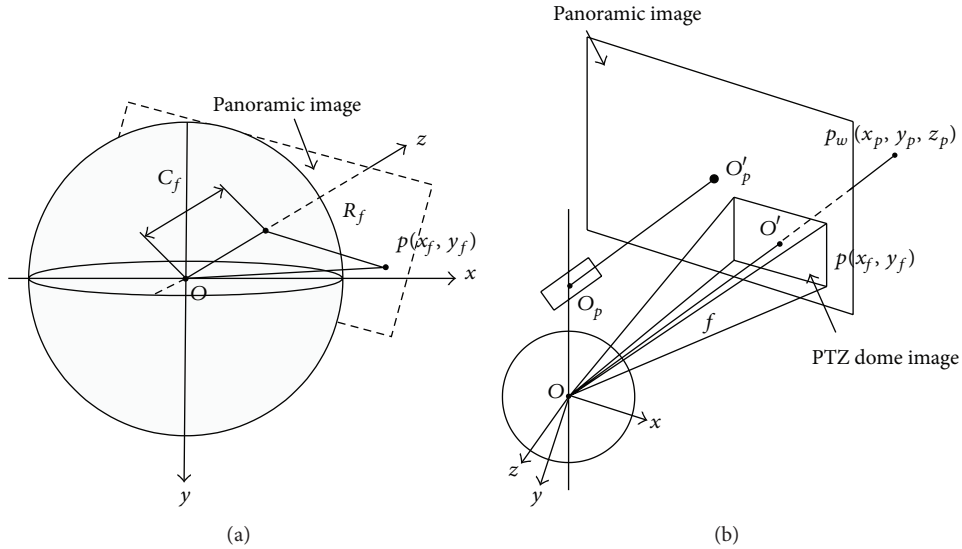


Figure 5: (a) PTZ image coordinate system imaging and (b) PTZ camera's pinhole.

As shown in Figure 3(a), we can obtain the distance R_f between point p and the center. Based on the fish-eye imaging model, the radical angle of point p off-center is

$$\theta = \frac{R_f}{C_f}. \tag{3}$$

According to R_f and θ, the z-axis of the point p can be calculated. So the transformation is

$$\begin{aligned} x_{Ps} &= Sx_f, \\ y_{Ps} &= Sy_f, \\ z_{Ps} &= S\frac{R_f}{\tan\left(R_f/c_f\right)}. \end{aligned} \tag{4}$$

S is the normalization constant to ensure $\sqrt{x_{Ps}^2 + y_{Ps}^2 + z_{Ps}^2} = 1$.

(2) Transformation from PTZ coordinate system to spherical coordinate system.

As PTZ dome camera model is a hemisphere model, the PTZ coordinate system is identical with the spherical coordinate system. The transformation can be written as

$$\begin{aligned} x_{Hs} &= \cos\varphi \sin\theta, \\ y_{Hs} &= \sin\varphi \sin\theta, \\ z_{Hs} &= \cos\theta. \end{aligned} \tag{5}$$

According to the transformation using the two steps described above, the final transformation can be obtained.

3.3. Calibration Theory between Master and Slave Camera. (x_f, y_f) are the coordinates of pixel p in the panoramic coordinate system. (x_p, y_p, z_p) are the coordinates of the corresponding point in the PTZ coordinate system. The mapping T can be determined by finding a matrix that makes the optical centers of the two cameras coincide with each other. Namely, a panoramic image can be considered as a large PTZ dome image. A PTZ dome camera satisfies the theory of pinhole imaging. According to Tsai [20], the theory of pinhole camera imaging is illustrated in Figure 5(b), in which $s\widetilde{m} = PX_c$.

In the proposed system, a fish-eye panoramic camera is mounted above a PTZ dome camera and their optical centers are on the same vertical plane. Here O_p is the optical center of panoramic camera and O'_p is its principal point. O_p is the optical center of PTZ dome camera and O' is its principle point. p is a point in the panoramic coordinate system and p_w is the corresponding object point in the world coordinate system. We project the PTZ dome image on the panoramic image. According to pinhole imaging theory, if the PTZ dome camera shoots at p_w, O' will coincide with p, which is the result we hope to obtain. In this work, the mapping T is calculated through a process of selecting sampling points. According to the algorithm presented by Zhang [21], it has to select at least 3 PTZ dome image to solve the transformation matrix.

3.4. Feature Points Matching. This step requires a method to detect and match visual feature that is robust to scale, rotation, viewpoint, and lightning. The Scale Invariant Feature Transform (SIFT) [22] exhibits great performance under these requirements. In this work, we employ SIFT to detect feature points in both the panoramic image and the PTZ dome image. Consider that $\{P_{Hi}^j\} = \{(X_{Hi}^j, Y_{Hi}^j, Z_{Hi}^j)\}^T$ $(i = 1, \ldots 6;\ j = 1, \ldots n,\ n \geq 6)$ is the j_{th} feature point in the spherical coordinate of the i_{th} PTZ dome image. Consider that $\{(P_{Pi}^j)\} = \{(X_{Pi}^j, Y_{Pi}^j, Z_{Pi}^j)\}^T$ $(i = 1, \ldots 6;\ j = 1, \ldots n,\ n \geq 6)$ is the panoramic spherical coordinate that corresponds

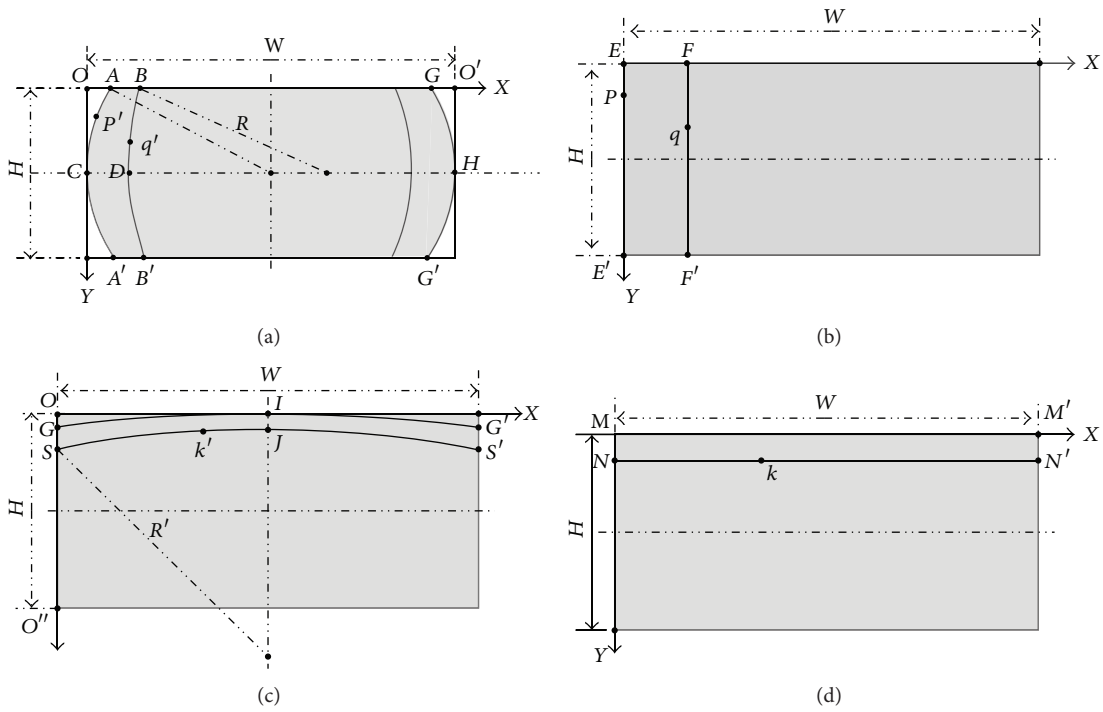


Figure 6: (a) The structure of captured fish-eye panoramic image in vertical direction. (b) VCI in vertical direction. (c) VCI in horizontal direction. (d) HVCI in horizontal direction.

to $\{P_{Hi}^{j}\}$. A_{Hi} is the feature point matrix in the ith PTZ dome image. A_{Pi} is the matching feature point matrix in the panoramic image:

$$T_i = A_{Pi} \times \left(A_{Hi}^T A_{Hi}\right)^{-1} A_{Hi}^T, \tag{6}$$

where $A_{Hi} = [P_{Hi}^1, P_{Hi}^2, \ldots, P_{Hi}^m]$ and $A_{Pi} = [P_{Pi}^1, P_{Pi}^2, \ldots, P_{Pi}^m]$. P_{Hi}^0 is the center point spherical coordinate of the ith PTZ dome image. P_{Pi}^0 is the panoramic spherical coordinate that corresponds to P_{Pi}^0:

$$P_{Pi}^0 = T_i \times P_{Hi}^0 \quad (i = 1, \ldots n). \tag{7}$$

Here A_H is the center point spherical coordinate matrix in the PTZ dome image. A_P is the matching point matrix in the panoramic image:

$$T_{PH} = A_P \times \left(A_H^T A_H\right)^{-1} A_H, \tag{8}$$

where $A_H = [P_{H1}^0, P_{H2}^0, \ldots, P_{Hn}^0]$ and $A_P = [P_{P1}^0, P_{P2}^0, \ldots, P_{Pn}^0]$. The calibration can be considered as calculating the transformation matrix. As a result, for given any pixel, we can calculate the corresponding rotation angle. The feature points matching step is employed to solve the unknown in the transformation matrix T_{PH}.

4. Distortion Correction

As mentioned by Strand and Hayman [23], a straight line in world coordinates can be projected to a corresponding circle on the fish-eye image plane, which means that the mapping process can be calculated directly using MCA [16]. The proposed distortion correction method can be divided into two parts. Firstly, the coordinate mapping is calculated based on MCA between a column of the first corrected image and the arc line of the original fish-eye panoramic image in the vertical direction, which is named Vertical Correction Image (VCI) in this work. Secondly, following the same principle, the coordinate mapping between a row of the second corrected image and the arc line image of VCI in horizontal direction is calculated, which is named Horizontal Vertical Correction Image (HVCI) here.

Figures 6(a) and 6(b) demonstrate the structure of captured fish-eye panoramic image and VCI in vertical direction, whose width and height both are W and H. Figures 6(c) and 6(d) show the VCI and HVCI in the horizontal direction.

For vertical distortion correction, the first column EE' in VCI corresponds to the first arc line ACA' in the fish-eye panoramic image. For an arbitrary point P on EE', there is a corresponding point P' on I. Since the distortion is corrected only in the vertical direction, ordinate value of point P is the same with its corresponding point P'. In fish-eye panoramic image, OA does not project to VCI, so its length L is called Vertical Distortion Redundancy Length (VDRL).

For an arbitrary column FF' in VCI, assume that its corresponding arc line image in the fish-eye panoramic image is BDB'. The point $F(X_f, Y_f)$ on FF' has its corresponding point $B(X_b, Y_b)$ on BDB'. Based on MCA, for an arbitrary point $q(X_q, Y_q)$ on FF', assume that its corresponding point

on BDB' is $q'(X_0, Y_0)$. Thus the corresponding point position can be calculated as follows:

$$X_0 = \frac{\lambda^2 + \delta^2 - X_f^2}{2(\delta - X_f)} - X_f + X_q - \sqrt{\frac{\lambda^2 + \delta^2 - X_f^2}{2(\delta - X_f)} - X_f^2 - (\lambda - Y_0)^2} \quad \left(X_q \le \frac{W}{2}\right),$$

$$X_0 = \frac{\lambda^2 + (W - \delta)^2 - (W - X_f)^2}{2(X_f - \delta)} - \omega + X_q + \sqrt{\frac{\lambda^2 + (W - \delta)^2 - (W - X_f)^2}{2(X_f - \delta)} - \omega^2 - (\lambda - Y_0)^2} \quad \left(\frac{W}{2} < X_q \le W\right),$$

$$Y_0 = Y_K, \tag{9}$$

where $\lambda = H/2$, $\delta = L + X_f((W - 2L)/W)$, and $\omega = H - Y_k$. The extent of the distortion correction can be controlled by the value of VDRL.

The distortion correction in the horizontal direction follows the same logic. In HVCI, an arbitrary row has its corresponding arc line in the VCI, such as the first row line image MM' and its corresponding first arc line GIG'. With VDRL in fish-eye panoramic image, there also is Horizontal Distortion Redundancy Length (HDRL) L' in the VCI, which is the length of $|OG|$. Because the distortion is corrected only in horizontal direction, the abscissa value of a point in HVCI is the same as its corresponding point in VCI. For an arbitrary row line NN' in HVCI, assume that its corresponding arc line in VCI is SJS'. A point $k(X_k, Y_k)$ on NN' has its corresponding point $k'(X_1, Y_1)$ on SJS'. The coordinate values of point k' can be derived as follows:

$$Y_1 = \frac{\lambda^2 + \delta^2 - Y_k^2}{2(\delta - Y_k)} - \sqrt{\left(\frac{\lambda^2 + \delta^2 - Y_k^2}{2(\delta - Y_k)} - Y_k\right)^2 - (\lambda - X_1)^2} \quad \left(Y_k \le \frac{H}{2}\right),$$

$$Y_1 = \frac{\lambda^2 + (H - \delta)^2 - (H - Y_k)^2}{2(Y - \delta)} - H + 2Y_k + \sqrt{\left(\frac{\lambda^2 + (H - \delta)^2 - (H - Y_k)^2}{2(Y - \delta)} - (H - Y_k)\right)^2 - (\lambda - X_1)^2} \quad \left(\frac{H}{2} < Y_k \le H\right),$$

$$X_1 = X_k, \tag{10}$$

where $\lambda = W/2$ and $\delta = L' + Y_k((H - 2L')/H)$. By traversing the HVCI, all positions can be mapped to corresponding points in VCI. Thus, the distortion in the horizontal direction can be corrected, and the extent of distortion correction can also be controlled by adjusting the value of HDRL.

After correcting the distortion in the vertical and horizontal directions, the distortion in the fish-eye panoramic image captured by master-slave camera can be corrected effectively and efficiently.

5. Experiments and Results

With regard to the time consuming of the proposed calibration method. The process can be considered as two steps: image subtracting and image matching. Capturing one image takes 2 seconds roughly. For one pair of images, image matching procedure takes about 5 seconds. In this circumstance, the entire process only takes 50 seconds. Compared with the traditional manual calibration which normally consumes several hours, the proposed automatic calibration algorithm increases the efficiency dramatically.

For the accuracy evaluation, Figure 7 demonstrates the images captured by the proposed master-slave camera. Here we focus on four areas of the panoramic image which includes distinguished features.

Figure 8 shows the mean error between requested pixel (p_x, p_y) and the ground truth pixel (p'_x, p'_y) to evaluate the distribution of error.

In Figure 8, the dark blue area represents 0~35 pixels, the green area represents 35~70 pixels, and the red area represents 70~105 pixels. In this work, the size of the PTZ dome image is 1920 ∗ 1080 pixels. It shows that error can be under-controlled within 1° over 95% of the scene. However, three main factors could influence the accuracy during the calibration procedure. Firstly, the PTZ dome camera only has accuracy of 1° other than 0.1°. Under this condition, 3.5 pixels is equivalent to 1° which bring the result that the centering error greatly reduces. Secondly, the selection of the PTZ dome images' position. Once the chosen 6 PTZ dome images are in uniform distribution in the panoramic image, the red area would not appear in Figure 8. Last but not least, it is the feature point matching process and the numbers of match points directly influence the result of calibration.

Figure 9 shows the images after distortion correction with different HDRL values, and the correction results are marked by red lines to offer a more direct expression. The distortion in Figure 9(a) is corrected in a small scale without any obvious noticeability. Although the distortion correction result in Figure 9(b) is close to the realistic scene, it is still away from being ideal, while in Figure 9(c), the HDRL value is 120 in which the distortion in horizontal direction obtains a desirable correction effect. When the HDRL value is 140 (Figure 9(d)), the setting is obviously overdone and causes overdistortion. It is worth mentioning that the HDRL needs to be tested to obtain a relatively ideal value since it differs in different panoramic images.

Table 1 shows the comparison of time consumption on dealing with different resolution images with the algorithm

Table 1: Comparison of time and resource consumption.

Image resolution	Ying's algorithm [13]			Our algorithm		
	Time (ms)	CPU (%)	Memory (Kb)	Time (ms)	CPU (%)	Memory (Kb)
640 × 480	416	15	3284	30	1	1516
1280 × 720	1371	20	9524	212	4	1992
1920 × 1080	3125	20	19604	274	6	2336
2560 × 896	3605	20	25560	419	14	2692

Figure 7: Experimental result after the calibration. Several areas are selected in the panoramic image as focus of interest, and corresponding multiscaled image is captured by PTZ dome camera.

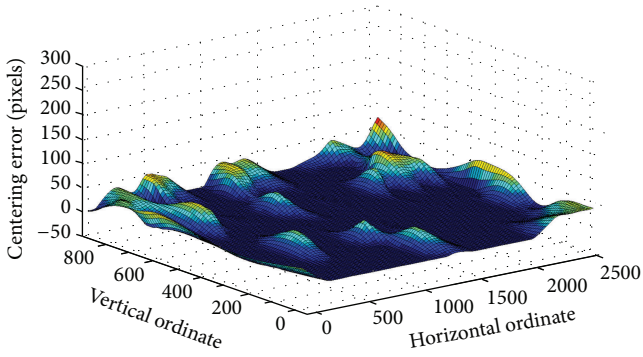
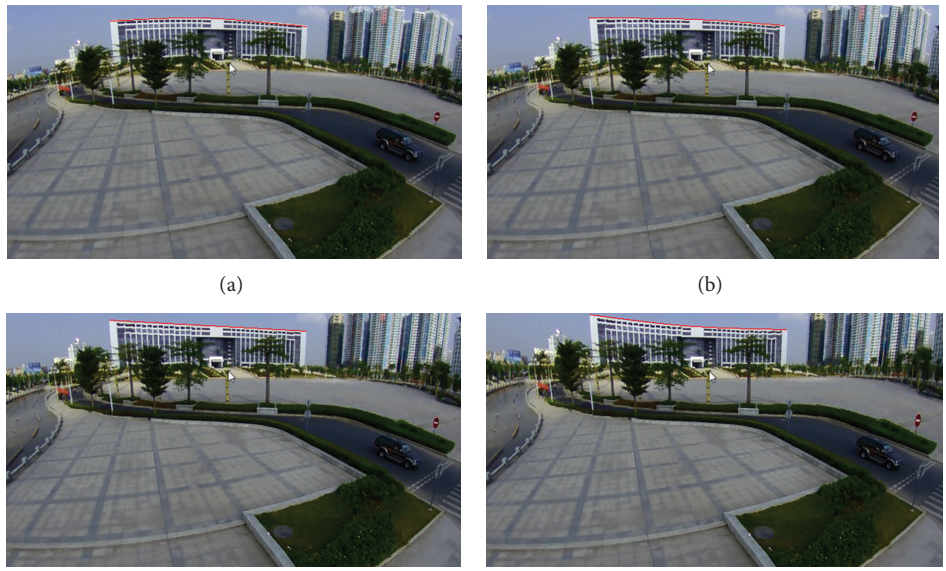


Figure 8: Mean error between requested and the ground truth pixel.

proposed by Ying et al. [13]. The comparison was conducted with single-threaded implementation on 3.20 GHz Intel Core i5-3470 CPU and 4.00 GB RAM computer by using Microsoft Visual Studio 2008 software. The result shows that the proposed algorithm has much less time consumption than Ying's algorithm under different resolutions. Moreover, the calculation resource consumption between Ying's algorithm and the proposed algorithm is also listed, which includes the utilization of CPU and memory. As represented, less resource is required by the proposed correction method which enables the proposed surveillance system to achieve real-time performance.

6. Conclusion and Discussion

A master-slave camera system that is composed of panoramic and PTZ dome cameras is proposed for stationary and dynamic visual surveillance. A panoramic camera observes a scene with a large field of view, and PTZ dome cameras simultaneously capture high-resolution images with multi-scale information. It can roughly cover 2-square-kilometer area with one camera, especially suitable for the large area surveillance such as squares and stadiums. More specifically, we present a calibration method for obtaining the mapping relations between master camera and slave camera. The availability and accuracy of the method are validated by the experiments shown in this paper. Additionally, we propose a correction approach to correct the particular type of distortion in fish-eye panoramic image captured by this



(a) (b) (c) (d)

Figure 9: Different images after distortion correction with different Horizontal Distortion Redundancy Length (HDRL) values. (a)~(d) show the corrected images when HDRL values are 60, 90, 120, and 140, respectively.

camera system. It has been applied on embedded camera platform without any extra hardware resources due to its low computational cost. In order to achieve the more precise interaction, future work would consider a calibration method based on panoramic image mosaic to obtain the pixel level mapping relation between the fish-eye image and the PTZ camera's motion parameters.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by Scientific Research Foundation for Returned Overseas Chinese Scholars, Doctoral Fund of Ministry of Education of China under Project (20134307120040), and National University of Defense Technology (JC13-05-02).

References

[1] Y. Sato, K. Hashimoto, and Y. Shibata, "A new video surveillance video tracking system based on omni-directional and network controlled cameras," in *Proceedings of the International Conference on Advanced Information Networking and Applications (AINA '09)*, pp. 602–607, May 2009.

[2] G. Scotti, L. Marcenaro, C. Coelho, F. Selvaggi, and C. Regazzoni, "Dual camera intelligent sensor for high definition 360 degrees surveillance," *IEEE Proceedings on Vision, Image and Signal Processing*, vol. 152, pp. 250–257, 2005.

[3] A. Hampapur, S. Pankanti, A. W. Senior, Y. Tian, L. Brown, and R. Bolle, "Face cataloger: multi-scale imaging for relating identity to location," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 13–20, Miami, Fla, USA, July 2003.

[4] C.-H. Chen, Y. Yao, D. Page, B. Abidi, A. Koschan, and M. Abidi, "Heterogeneous fusion of omnidirectional and PTZ cameras for multiple object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1052–1063, 2008.

[5] S. Tan, *Research on cooperative tracking between omnidirectional camera and active camera [Ph.D. thesis]*, College of Information System and Management, National University of Defense Technology University, Hunan, China, 2011.

[6] A. W. Senior, A. Hampapur, and M. Lu, "Acquiring multi-scale images by pan-tilt-zoom control and automatic multi-camera calibration," in *Proceedings of the 7th IEEE Workshop on Applications of Computer Vision (WACV '05)*, vol. 1, pp. 433–438, January 2005.

[7] X. Zhou, R. Collins, T. Kanade, and P. Metes, "A master-slave system to acquire biometric imagery of humans at distance," in *Proceedings of the 1st ACM International Workshop on Video Surveillance*, pp. 113–120, November 2003.

[8] L. You, S. Li, and W. Jia, "Automatic weak calibration of master-slave surveillance system based on mosaic image," in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR '10)*, pp. 1824–1827, August 2010.

[9] F. Devernay and O. Faugeras, "Straight lines have to be straight," *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, 2001.

[10] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006.

[11] J. Wang, F. Shi, J. Zhang, and Y. Liu, "A new calibration model of camera lens distortion," *Pattern Recognition*, vol. 41, no. 2, pp. 607–615, 2008.

[12] W. Yu, "An embedded camera lens distortion correction method for mobile computing applications," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 894–901, 2003.

[13] X. Ying, Z. Hu, and H. Zha, "Fisheye lenses calibration using straight-line spherical perspective projection constraint," in *Proceedings of the 7th Asian Conference on Computer Vision (ACCV '06)*, pp. 61–70, 2006.

[14] C. Hughes, R. McFeely, P. Denny, M. Glavin, and E. Jones, "Equidistant (f θ) fish-eye perspective with application in distortion centre estimation," *Image and Vision Computing*, vol. 28, no. 3, pp. 538–551, 2010.

[15] C. Hughes, P. Denny, M. Glavin, and E. Jones, "Equidistant fish-eye calibration and rectification by vanishing point extraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2289–2296, 2010.

[16] Y. Li, M. Zhang, Y. Liu, and Z. Xiong, "Fish-eye distortion correction based on midpoint circle algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2224–2228, October 2012.

[17] R. Carroll, M. Agrawal, and A. Agarwala, "Optimizing content-preserving projections for wide-angle images," *ACM Transactions on Graphics*, vol. 28, no. 43, pp. 1–9, 2009.

[18] J. Kopf, D. Lischinski, O. Deussen, D. Cohen-Or, and M. Cohen, "Locally adapted projections to reduce panorama distortions," *Computer Graphics Forum*, vol. 28, no. 4, pp. 1083–1089, 2009.

[19] J. Wei, C. Li, S. Hu, R. Martin, and C. Tai, "Fisheye video correction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 10, pp. 1771–1783, 2012.

[20] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[21] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] R. Strand and E. Hayman, "Correcting radial distortion by circle fitting," in *Proceedings of the British Machine Vision Conference*, 2005.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2013, Article ID 946768, 10 pages
http://dx.doi.org/10.1155/2013/946768

Research Article

Vulnerability Assessment of IPv6 Websites to SQL Injection and Other Application Level Attacks

Ying-Chiang Cho and Jen-Yi Pan

Department of Electrical Engineering, National Chung Cheng University, Chia-Yi 62102, Taiwan

Correspondence should be addressed to Ying-Chiang Cho; silvergun@mail2000.com.tw

Received 14 October 2013; Accepted 2 December 2013

Academic Editors: S. K. Bhatia and A. K. Misra

Given the proliferation of internet connected devices, IPv6 has been proposed to replace IPv4. Aside from providing a larger address space which can be assigned to internet enabled devices, it has been suggested that the IPv6 protocol offers increased security due to the fact that with the large number of addresses available, standard IP scanning attacks will no longer become feasible. However, given the interest in attacking organizations rather than individual devices, most initial points of entry onto an organization's network and their attendant devices are visible and reachable through web crawling techniques, and, therefore, attacks on the visible application layer may offer ways to compromise the overall network. In this evaluation, we provide a straightforward implementation of a web crawler in conjunction with a benign black box penetration testing system and analyze the ease at which SQL injection attacks can be carried out.

1. Introduction

IPv6 was developed by Engineering Task Force [1] to solve the issue of insufficient number of addresses provided by the IPv4 protocol [2]. With the proliferation of internet enabled devise, the limits of IPv4 have been reached. IPv6 is composed of 128 bits, generating a total of 3.4×1038 addresses, which is 7.9×1028 times the address space of IPv4. Because of this greatly increased address space, most normal "war-dialing" type attacks [3, 4] are not feasible, and thus IPv6 offers an increased level of security versus IPv4. Therefore, most information security research with regard to IPv6 focuses mainly on the discussion of IP layer [5, 6] seeking to show that the underlying protocol is resistant to attack. However, this research ignores the overall nature of the Internet; that is, devices are inherently interconnected, and that once a vulnerable device can be identified linking information exists that allows one to identify other devices on the network. Therefore, it is trivial to traverse the links to attack the device of interest.

More specifically, most attacks are against organization rather than individual devices. Most of these organizations are connected to the internet in such a way where they can be searched via publically available search engines or a hyperlink structure can be traversed to reach them. After the initial web server or database has been identified and breached, other devices belonging to the organization can then be identified and compromised in turn, with the increased address space of IPv6 not making a difference. Thus, the increased address space provided by IPv6 does not offer any practical barriers to finding targets to attack.

To demonstrate this vulnerability, we will utilize the keyword search of publically available search engines such as Google, Bing, and Yahoo in conjunction with a web crawler with a black box penetration testing kit [7–9] to show how this can be done in principle.

2. Algorithm Principle

Briefly, the overall component of this system is a web crawler [10, 11] that takes an initial website, traverses the links on the front page, and tries to identify vulnerable links that can be exploited. This system also has a secondary component which utilizes search engines such as Google, Bing, and Yahoo to search for specific URLs that might be vulnerable to injection.

2.1. Web Crawling Algorithm. An effective web crawler needs to implement four key elements, a selection policy, a revisitation policy, a politeness policy, and parallelization scheme [12–15]. Briefly, the selection policy defines which sites that one will visit and includes aspects such as whether a link has been previously visited. A revisitation policy defines how often a link should be refreshed in order to detect changes. Politeness reflects the scheme by which a server is not overloaded with requests, and finally the parallelization scheme defines how the process can be parallelized for efficient searching.

The typical web crawler works via breadth first search [16], in which a frontier of unvisited links is first presented. These nodes are traversed, and a new frontier of unvisited sites is found after which the process repeats. One complication, however, is that to mark a site as visited, we normally rely upon a hashing protocol that functions on a canonical web address rather than just storing the address in its entirety. The overall process is given in Figure 1. However, one complication that needs to be addressed is that malicious websites or "spider traps" can be crafted so that web crawlers are trapped in an infinite loop [17]. Therefore, the hashing strategy must also take into account some of the content associated with the page.

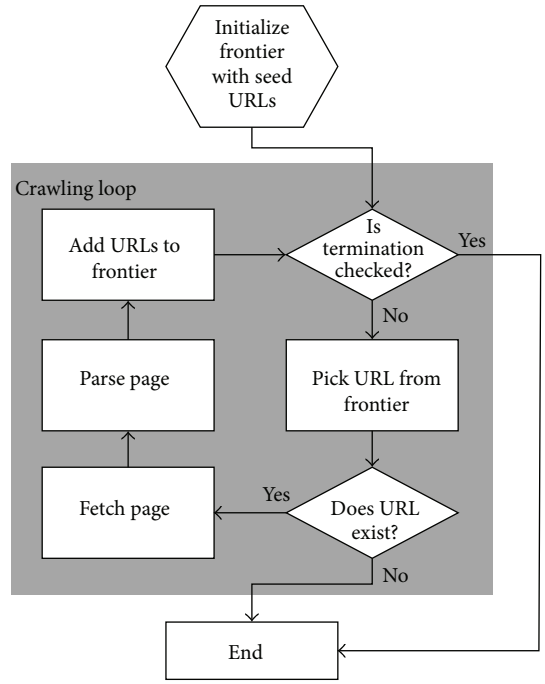


Figure 1: Web crawling flow chart.

2.2. Dynamic Analysis. Dynamic analysis identifies security problems by directly interacting with a functioning website. In other words, dynamic analysis relies on simulating user interactions with web pages, including interactions designed with potentially malicious intent. Because dynamic analysis uses a real website to find vulnerabilities in real time, found vulnerabilities are much more likely to be real than with static analysis, which has problems with detecting false positives [18, 19]. Black box testing determines whether a web application has a vulnerability by inputting testing data to the application and analyzing its response [9], as opposed to white box testing which focuses on source code parsing and analysis. White box testing tends to have lower efficiency because it does not factor in the dynamic interplay between the web server, application server, and database server [20]. Therefore, it is more common to use black box testing to more holistically analyze web application's vulnerability [21]. Penetration testing is a method to estimate the security of computer system or internet security by actively simulating attacks [22, 23]. This method analyzes all possible security weakness in the system, so the testing result is very valuable and convincing. The end product is not simply potential vulnerabilities but verified vulnerabilities and exploits. Honest testing result can form a bridge between developer and information security communities [24, 25].

2.3. Testing for SQL Injection. SQL injection [26–28] takes advantage of the process of web applications accessing databases with queries based on improperly validated user input. The website security mining system finds SQL injection attacks which can bypass firewall and identity authorization to control the database [29]. SQL injection can penetrate any type of database that relies on SQL, regardless of whether the underlying web application is written in ASP, PHP, or JSP as long as the program has a severe yet common logic error. Although there are well-known techniques to combat SQL injection attacks [30, 31], they are still quite common, and, therefore, there has been much interest in developing methods to inspect web applications and detect these vulnerabilities [29, 32, 33].

2.4. Testing for Brute Force. A complete dictionary file is important to our research. The web crawler we designed will detect the pages with weak password [34], and hence the detection ability [35, 36] will be decided on quality of the dictionary file. In particular in case of crawling database form, ineffective dictionary file causes slow crawling while insufficient dictionary file fails to determine whether the webpage's password detected reaches the information security standards [37, 38]. With thousands of real experiments, our system refers to many related literatures [9, 39–43] and web vulnerability scanners practically applied. We provide function of adjustable parameters to handle different environment through flexible adjustment (for instance, Http versus FTP, whether complying with protocol of robots.txt, priority of attacks facing numerous database, the depth of crawling a webpage and whether detecting broken links, etc.). However, we do not focus on these issues here.

2.5. The Security Issues from IPv4 to IPv6. There are three techniques to transform IPv4 to IPv6 addresses which are dual stack, tunneling, and translation [44]. Most of the security research with respect to IPv6 has focused upon these translation layers as well as in the authentication and encryption of the individual data packets [45–48]. The primary

```
<?xml version="1.0" encoding="utf-8"?>
<config>
<name>Name of Vulnerability</name>
<date>Releasing Date of Vulnerability</date>
<author>Author</author>
<version>Version Number</version>
<type>Type of Vulnerability</type>
<description>Description of Vulnerability</description>
<file>File of Causing Vulnerability </file>
<bugfile>The URL that used for Vulnerability Testing</bugfile>
<successkeyword>Successful keyword shown on the page after
error appears</successkeyword>
</config>
```

Algorithm 1: XML format.

concern that security researchers have tried to address is the problem of incorrect redirection and spoofing. However, it should be noted that the majority of attacks against the current IPv4 infrastructure do not occur at the transport layer but rather at the application layer [49] and that these attacks still apply to IPv6. For instance, attacks such as sniffing, application layer attacks [50], rogue devices, man-in-the-middle attacks, and flooding are still applicable. Both IPv4 and IPv6 are vulnerable when facing application layer attacks [51], as shown in Figure 2 [52].

Among many application level attacks methods, SQL injection is the most well known. Furthermore, because it attacks databases which may store information related to accounts and authentication, they are an attractive target to hit. In our evaluation we combine the discovery module which utilizes web crawling with black box penetration methods [53] to implement a system which is called website security mining system. It has two modules and six functions. The modules are the dynamic scanning module and static mining module. The functions are the syntax analysis function, search engine assistant function, hidden page searching function, vulnerability updating function, specific file searching function, and multidomain lookup with single IP function. The experimental targets are each country's IPv6 official website. We use the system to crawl each website 24 hours and gather statistics to each site's found e-mails and injectable URLs to compare the security protection done in each country's IPv4 website.

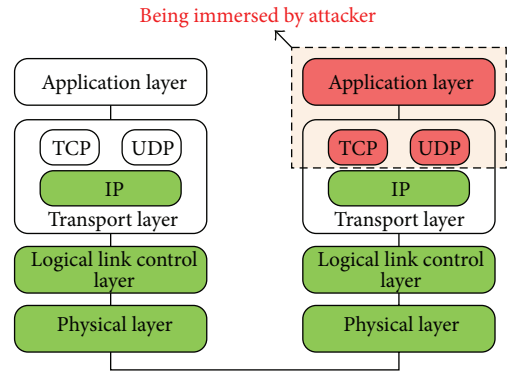


Figure 2: Application attack.

3. System Implementation

In order to inspect if information stored on the web presents a security risk, this research combines a web crawler, like those used in search engines, with the concept of application vulnerability inspection, specifically black box and penetration testing. The end product is the website security mining system, a tool to evaluate a website's security. This system can be separated into two main modules which are the Static Mining module and the Dynamic Scanning module. The Static Mining module inspects a specific website's robots.txt, E-mails, potential SQL injection URLs, files, and broken links. The Dynamic Scanning module uses the system's vulnerability-inspecting function by typing keywords into a search engine's query box to inspect many websites.

Both the Static Mining and Dynamic Scanning modules leverage the system's vulnerability inspecting function, which has two parts: known website vulnerability inspection and SQL injection inspection. The former compiles a database of open source website vulnerabilities into an XML file which is used to inspect the website to see whether it has the same vulnerability. Algorithm 1 is the format of an XML file. The bug file parameter is a base64 hash and other parameters are converted from the open source website vulnerabilities database. Our system updates its vulnerability database by adopting new vulnerabilities that have been announced on the Exploit Database regularly [54]. By updating the vulnerability database, we can ensure that the vulnerability samples are always updated, similar to how antivirus software regularly updates its virus database.

The website security mining system can find vulnerabilities in a variety of database engines, specifically MS-SQL, MySQL, Access, Oracle, Postgresql, and Sqlite. The steps to identify SQL injection vulnerabilities are as follows. First, an injectable point must be identified by inspecting the website for places where user input may be used in SQL queries. If such an injectable point is found, then further tests are

conducted to identify the specific type of database engine. To do this, we take advantage of how different databases use different function names for certain tasks. For example, MS-SQL and MySQL use len( ) to calculate length; however, Oracle uses length( ) to do it. In other words, when you use len("s") = 1 to test and receive a normal response, that means the target website uses MS-SQL or MySQL. On the other hand, if this does not work, then the database might be Oracle or other types of database. There are several other functions that can help us determine what the database is. After getting the database's type, we find table and column names and finally get the database's data.

The website security mining system can run on any operating systems that are supported by Java. We describe the two basic modules in more detail below.

3.1. Static Mining Module. The Static Mining module runs depth mining on a specific website. There is an option to determine whether you want to follow the website's robot.txt rules. Robot.txt [55] is an ASCII-encoded file stored in the website's root directory that lists files that can and cannot be accessed by search engine crawlers. There is no official standards body or RFC for the robots.txt format. It was created by consensus and ultimately serves only as a recommendation for crawlers, so it cannot protect the website's private data completely [56]. Other functions of the Static Mining module are identifying e-mail information, potentially injectable URLs, downloadable files, and broken links, which may contain private information.

The Static Mining module starts with a specific web site and then collects all the related pages from it using a breath-first search algorithm. The system assumes that web pages have close relations to the original web page if the link distance is within a certain range [57–59], so it will fetch all links inside the original page then iterate through all of those URLs to fetch all links within them. This type of method can be easily parallelized to improve fetching speed. After files are downloaded by the web crawler, an HTML parser process extracts pages' URLs and then adds it into the URL queue. Also, the system will call vulnerability inspecting process to inspect URLs, checking whether it has potential vulnerabilities or not.

Figure 3 shows the process of mining a college's website [60] by our system. Several injectable URLs were found and by exploiting these vulnerabilities we were able to retrieve the database information shown in Figure 4.

Additionally, we determined that the operating system (OS) of the host was "Microsoft Windows XP Professional," as shown in Figure 5, which could open up the possibility for further OS-based exploits.

3.2. Dynamic Scanning Module. The most popular search engines today are Google, Yahoo, Baidu, Microsoft Bing, NHN, eBay, Ask.com, Yandex, and Alibaba. With the help of search engines, we can find billions of web pages and their URLs. Our system inspects these websites to determine whether they have vulnerabilities by analyzing the results retrieved from search engines. Our system supports the kinds

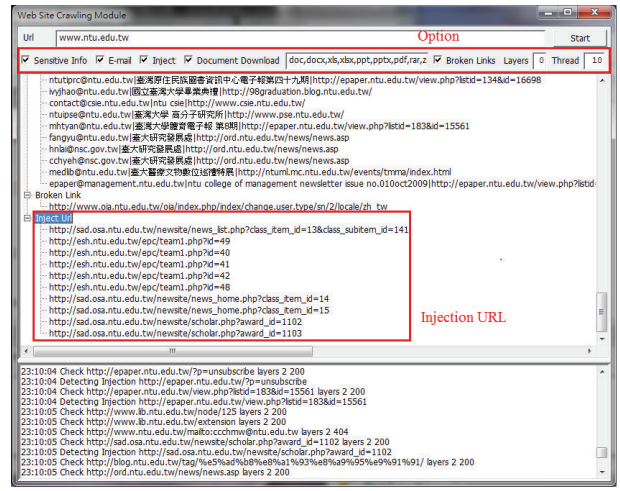


Figure 3: Static Mining module.

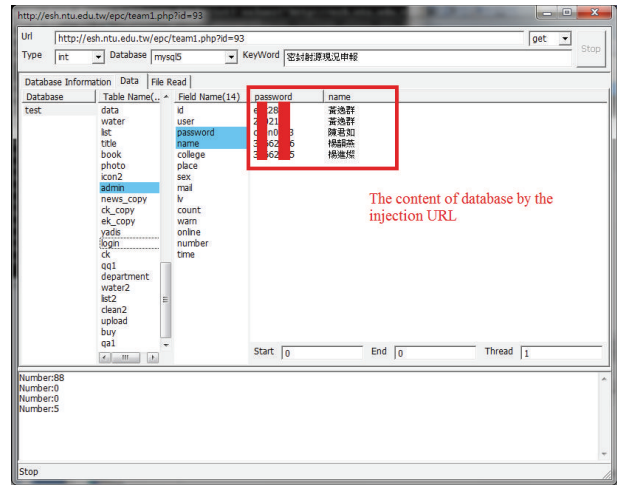


Figure 4: Database content found by injectable URLs.

of query syntaxes used in modern search engines. After you input the keywords, the system can find all related web pages and inspect whether they are at risk for vulnerabilities or not.

Figures 6, 7, and 8 show the different query syntaxes used in different search engines. For Google, it is "inurl:.asp? site:edu nobel prize"; Yahoo is "inurl:.php? site:edu education"; Bing is "inurl:.aspx? site:edu academic".

This research used the same command, inurl:.asp? |.jsp?|.php?|.aspx? site:com new, to search the ten most popular search engines. 800 web pages were retrieved from each search engine. We found 550 SQL injectable URLs and 21 known website vulnerabilities out of this total of 8000 web pages, which are shown in Figure 9 below. This highlights the fact that SQL injection problems are still very severe on the internet.

Despite SQL queries injection, this system provides functions of backend systems detection [61], session hijacking [62], Cookie poisoning [63], form manipulation [64], URL parameter tampering [65], HTTP header modification [66], bypassing intermediate forms in a multiple-form set [61],

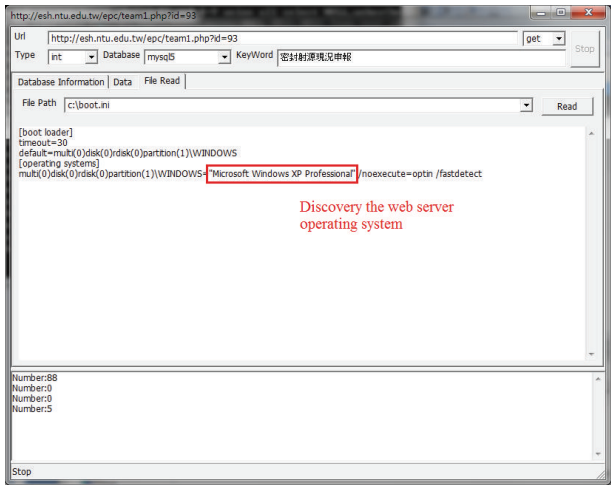


Figure 5: Host operating system.

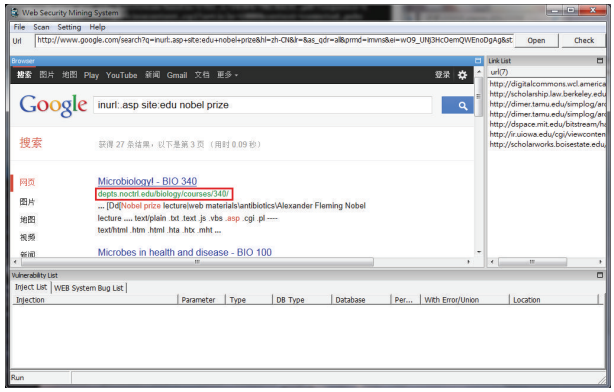


Figure 6: Work on Google search engine.

cross-site scripting [67], third party software misconfiguration [61], forceful browsing [43], and several security tests related to application layer. However, we do not focus on these issues here.

4. Experiments

We designed two experiments to test the security situation of IPv6's website on the internet. Experiment 1 uses WSMS's Dynamic Scanning module to compare the numbers of injectable URL in each IPv4 and IPv6 website. Experiment 2 uses WSMS's Static Mining module to crawl the websites that have been authorized by IPv6 forum [68], which can help us realize the situation of e-mail leakage and database leakage in IPv6's websites.

4.1. Experiment 1. We constructed WSMS in the pure IPv4 environment; it will show "getaddr info failed URL Error" message and stop if it crawled IPv6's website. In the case where we wish to explore IPv6 addresses, the converse will be true; that is, getaddr info URL failed will be returned for IPv4 address. We uses Dynamic Scanning module to search these four type web pages (asp/aspx/php/jsp) in three

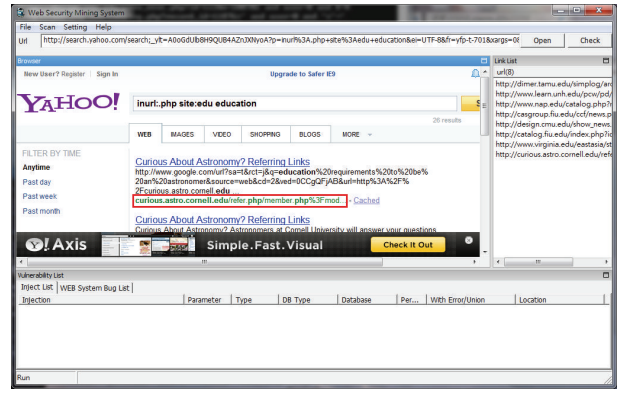


Figure 7: Work on Yahoo search engine.

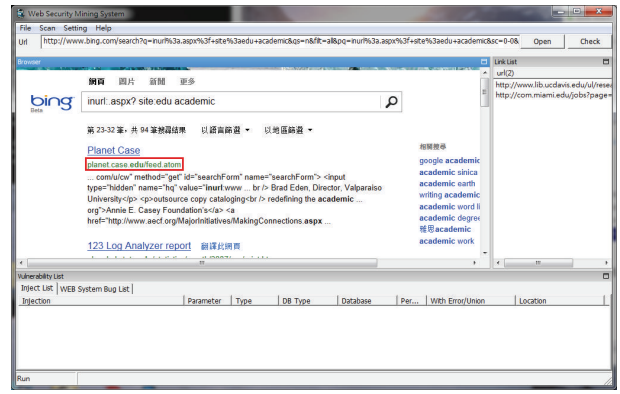


Figure 8: Work on Bing search engine.

Table 1: Pure IPv4 website detection statistics.

(X, Y, Z)	Google	Yahoo!	Bing
ASP	(831, 292, 22)	(558, 123, 7)	(575, 157, 15)
ASPX	(875, 286, 8)	(559, 68, 5)	(623, 153, 1)
PHP	(917, 311, 15)	(741, 177, 7)	(655, 220, 10)
JSP	(866, 290, 12)	(501, 171, 13)	(571, 152, 12)

different search engines (Google/Yahoo!/Bing) with "world peace" as the keyword. The statements which we type in Google, Yahoo!, and Bing are shown as

Google => inurl:asp? world peace

Yahoo! => world peace asp?

Bing => world peace asp?

We assumed that X represents the pure IPv4 web pages that contain world peace, Y represents the number of URLs that have been inspected by WSMS, and Z represents the number of URLs that are injectable. The operating process and data results are shown by Tables 1, 2, and 3 (also see Figure 10).

As shown in Table 4 analyzed through type of website, results of Analysis of variance (ANOVA) [69] (P value = 0.873 > 0.05) showed that the type of website has no significant contribution to the rate of vulnerability in either IPv4 or IPv6.

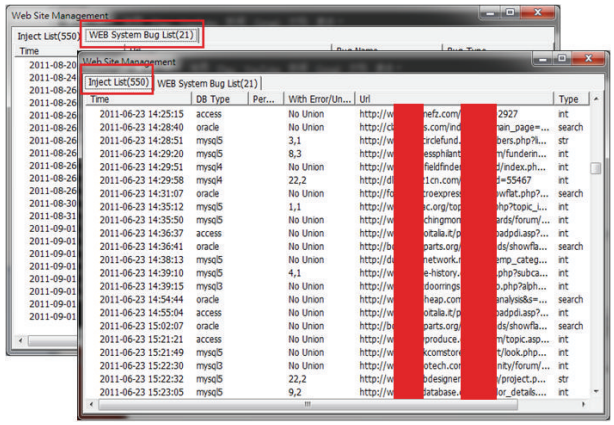


Figure 9: Dynamic scanning results.

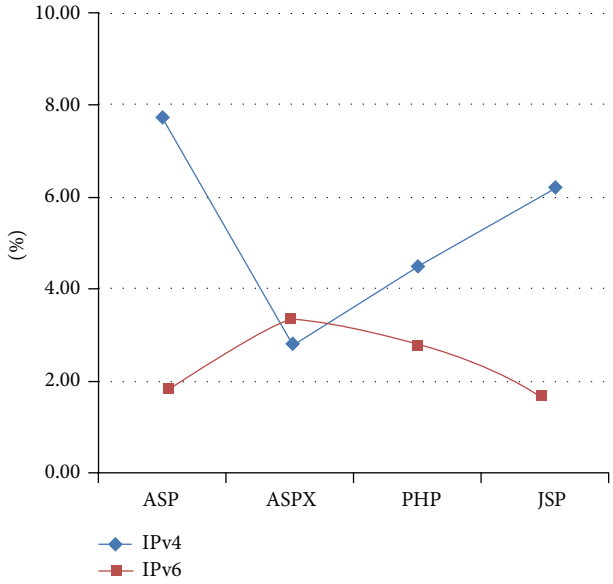


Figure 10: Ratio of injectable URL.

Table 2: Pure IPv6 website detection statistics.

(S, T, U)	Google	Yahoo!	Bing
ASP	(482, 222, 4)	(504, 25, 1)	(517, 22, 0)
ASPX	(368, 97, 2)	(502, 31, 1)	(506, 18, 2)
PHP	(520, 175, 3)	(607, 45, 2)	(639, 29, 2)
JSP	(128, 36, 0)	(516, 16, 0)	(546, 6, 1)

Table 3: Ratio of injectable URL.

	ASP	ASPX	PHP	JSP
IPv4	7.69%	2.76%	4.52%	6.04%
IPv6	1.86%	3.42%	2.81%	1.72%
Average	4.78%	3.09%	3.67%	3.88%

In Tables 5 and 6 analyzed through type of network environment, we knew that IPv4 and IPv6 have great difference in the prevalence of URL injection from results of independent

Table 4: ANOVA.

	SS	DF	MS	F	P
Ratio of injectable URL					
Between	0.001	3	0.000	0.232	0.873
Inner	0.023	20	0.001		
Sum	0.024	23			

Table 5: Group statistics.

	Web page	Number	Mean
Ratio of injectable URL	IPv4	12	0.05545133
	IPv6	12	0.02937988

Table 6: Independent samples test.

	Levene's test for equality of variances		t-test for equality of means		
	F test	P	t	DF	P (2-tailed)
The ratio of injectable URL	0.368	0.550	2.147	22	0.043

Table 7: Page ∗ type cross-tabulation.

	Type				Sum
	ASP	ASPX	PHP	JSP	
IPv4 page					
Amount	44	14	32	37	127
Ratio	34.6%	11.0%	25.2%	29.1%	100.0%
IPv6 page					
Amount	5	5	7	1	18
Ratio	27.8%	27.8%	38.9%	5.6%	100.0%
Sum					
Amount	49	19	39	38	145
Ratio	33.8%	13.1%	26.9%	26.2%	100.0%

Table 8: Chi-square test.

	Value	DF	Asymp. Sig. (2-tailed)
Pearson Chi-square	10.329[a]	3	0.016

a: 3 cells (37.5%) have expected count less than 5. The minimum expected count is 2.24.

samples t-test. In other words, upon the confident level of 95%, the average prevalence rate of injectable URL of IPv4 (5.55%) is larger than IPv6 (2.94%). However, the reasons for above statement remain unclear. We make hypothesis that the main reason that IPv6 sites have better security than IPv4 sites is because the IPv6 sites are newer and the programmers of these sites are more cognizant of vulnerabilities such as SQL injection and have already taken steps to mitigate these issues.

The experiment result shows that the number of inspectable URLs in Google is the highest above all because it supports the function of parameter "inurl." In IPv4's situation, ASPX has the least injection problem while ASP gets the worst outcome. In IPv6's situation, JSP has the least injection

Table 9: Experiment 2 result output.

Region/country	Tags	URL	MAIL disclosure amount	Inject URL amount
Mexico	Enterprise site	http://arteria.com.mx	91	0
Brazil	IT site	http://bgp.net.br	169	0
Denmark	Other	http://mirrors.dotsrc.org	13	0
Russia	Other	http://rusnavi.org	46	0
Belgium	Government site	http://www.awt.be	565	3
Argentina	Education site	http://ipv6solutions.com.ar	0	0
Spain	Education site	http://www.cba.upc.edu	557	0
Britain	Education site	http://www.ecs.soton.ac.uk	1697	0
Canada	Personal site	http://www.ampedcanada.com	0	0
America	Not-for-profit cooperative site	http://www.fairfaxcirclechurch.org	12	0
Germany	Other	http://www.das-labor.org	71	0
New Zealand	Other	http://www.geekzone.co.nz	97	0
Italy	Government site	http://www.imaa.cnr.it	33	0
India	Government site	http://www.nixi.in	77	0
Japan	Personal site	http://www.robata.org	82	0
Taiwan	Education site	http://www.yfp.ks.edu.tw	3	0
Thailand	Education site	http://ns6.cpe.rmutt.ac.th	0	0
China	Education site	http://www.zzrvtc.edu.cn	430	59
Switzerland	Other	http://www6.itu.int	1691	13
Poland	Education site	http://zsp6siedlce.pl	3	0

problem while ASPX performs poorly. In general, ASPX has much fewer problems and ASP has most problems. From Tables 7 and 8, IPv4 and IPv6 have great difference in the virus number detected in view of types of website, while from the result of Chi-square test, we discovered that ASP website accounts for 34.6% of virus detected in IPv4, followed by JSP with 29.1%; these two kinds of website perform better in IPv6 environment. In IPv6, PHP accounts for 38.9% of virus detected, followed by ASPX with 27.8%, and these two kinds of website perform better in IPv4.

4.2. Experiment 2. We constructed WSMS on the pure IPv6 environment, using Static Mining module to crawl twenty websites which were randomly selected from twenty regions from the IPv6 forum, and then we gathered the amount of E-mail address and injectable URL again (shown in Table 9); we see a significant number of websites in both IPv6 and IPv4 that are susceptible to attack, with IPv6 showing a lower level of vulnerability.

From the above two experiments, we can see that the migration to IPv6 still leaves a great deal of vulnerabilities present in the application level infrastructure with a great deal of vulnerabilities still existing. These vulnerabilities while known can represent the initial springboard for more targeted attacks.

5. Conclusion

One of the messages from this evaluation is that with respect to the majority of the attacks that are commonly used, IPv6 does not offer any increased level of security versus IPv4. This is not surprising given the fact that the application layer attacks bypass the majority of the security infrastructure built into IPv6. Therefore, the results of this evaluation are hardly surprising. However, one interesting consequence of IPv6 is that given the large address space, it becomes more difficult to identify where malicious attacks are coming from due to the fact that an attacker no longer has to be tied to a small number of IP addresses but instead has a much larger pool with which to hide. Without the need to be readily discoverable by the general public, this level of anonymity becomes a much stronger weapon for the attacker than it is for the defender. That being said, with a better understanding vulnerability, we see that newer systems are much more robust than legacy systems. This perhaps is the most important result of this paper.

The website figures sampled from the experiment can prove that, even though the injection problem of IPv6 website is less than the IPv4 one, IPv6's security protection on the transport layer does nothing to mitigate shortcomings on the application layer. Therefore, the programming habit [26, 70, 71] of the programmer is still critical. We all know that the information stored online is not one-hundred-percent safe, but one of the measures that an end user can take is to increase the complexity of its password setting [72, 73]. As for the server database side, the plain text password should be encrypted [74] before it is stored in the database, so that the hacker will not obtain easily authentication tokens when he breaks in the website and obtains the content of the database. The empirical measures show that aside from the website

programming logic and database security management, the encrypted storage of the data is also important.

Acknowledgments

The authors thank the National Science Council, Taiwan, for partially supporting this research under contract no. NSC 102-2221-E-194-036. The authors also feel grateful to anonymous reviewers for their very helpful and constructive criticism. Their suggestions have enabled the authors to improve the quality of this work.

References

[1] S. Deering and R. Hinden, IETF RFC2460, Internet Protocol, Version 6, 1998, http://www.ietf.org/rfc/rfc2460.txt.

[2] M. Boucadair, J. Grimault, P. Lévis, A. Villefranque, and P. Morand, "Anticipate IPv4 address exhaustion: a critical challenge for internet survival," in *Proceedings of the 1st International Conference on Evolving Internet (INTERNET '09)*, pp. 27–32, Cannes La Bocca, France, August 2009.

[3] M. Gunn, "War dialing," 2002.

[4] Wikipedia, "War dialing," 2013, http://en.wikipedia.org/wiki/War_dialing.

[5] R. Oppliger, "Security at the internet layer," *Computer*, vol. 31, no. 9, pp. 43–47, 1998.

[6] S. Weber and L. Cheng, "A survey of anycast in IPv6 networks," *IEEE Communications Magazine*, vol. 42, no. 1, pp. 127–132, 2004.

[7] E. Fong and V. Okun, "Web application scanners: definitions and functions," in *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS '07)*, Waikoloa, Hawaii, USA, January 2007.

[8] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao, "A static analysis framework for detecting SQL injection vulnerabilities," in *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC '07)*, pp. 87–96, Beijing, China, July 2007.

[9] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the art: automated black-box web application vulnerability testing," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '10)*, pp. 332–345, Oakland, Calif, USA, May 2010.

[10] G. Pant, P. Srinivasan, and F. Menczer, *Crawling the Web*, 2004.

[11] A. Heydon and M. Najork, "Mercator: a scalable, extensible web crawler," *World Wide Web*, vol. 2, no. 4, pp. 219–229, 1999.

[12] H. Y. Kao, S. H. Lin, J. M. Ho, and M. S. Chen, "Mining web informative structures and contents based on entropy analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 41–55, 2004.

[13] I. S. Altingövde and O. Ulusoy, "Exploiting interclass rules for focused crawling," *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 66–73, 2004.

[14] V. Shkapenyuk and T. Suel, "Design and implementation of a high-performance distributed web crawler," in *Proceedings of the 18th International Conference on Data Engineering*, pp. 357–368, San Jose, Calif, USA, March 2002.

[15] C. Castillo, "Effective web crawling," Computer Science, The University of Chile in fulfillment: ACM SIGIR Forum, 2004.

[16] S. Even, *Graph Algorithms*, Cambridge University Press, New York, NY, USA, 2011.

[17] A. Paraskevas, I. Katsogridakis, R. Law, and D. Buhalis, "Search engine marketing: transforming search engines into hotel distribution channels," *Cornell Hospitality Quarterly*, vol. 52, no. 2, pp. 200–208, 2011.

[18] M. Weiser, "Program slicing," *IEEE Transactions on Software Engineering*, vol. 10, no. 4, pp. 352–357, 1984.

[19] A. Phalgune, "Testing and debugging web applications: an end-user perspective," in *Proceedings of the IEEE Symposium on Visual Languages and Human Centric Computing*, pp. 289–290, Rome, Italy, September 2004.

[20] N. El Ioini and A. Sillitti, "Open web services testing," in *Proceedings of the IEEE World Congress on Services (SERVICES '11)*, pp. 130–136, Washington, DC, USA, July 2011.

[21] N. Khoury, P. Zavarsky, D. Lindskog, and R. Ruhl, "An analysis of black-box web application security scanners against stored SQL injection," in *Proceedings of the IEEE 3rd International Conference on Privacy, Security, Risk and Trust (passat) and IEEE 3rd International Conference on Social Computing (SocialCom '11)*, pp. 1095–1101, Boston, Mass, USA, October 2011.

[22] M. Bishop, "About Penetration Testing," *IEEE Security & Privacy*, vol. 5, no. 6, pp. 84–87, 2007.

[23] N. Antunes and M. Vieira, "Enhancing penetration testing with attack signatures and interface monitoring for the detection of injection vulnerabilities in web services," in *Proceedings of the IEEE International Conference on Services Computing (SCC '11)*, pp. 104–111, Washington, DC, USA, July 2011.

[24] H. J. Kam and J. J. Pauli, "Work in progress—web penetration testing: effectiveness of student learning in Web application security," in *Proceedings of the Frontiers in Education Conference (FIE '11)*, pp. F3G-1–F3G-3, Rapid City, SD, USA, November 2011.

[25] C. Mainka, J. Somorovsky, and J. Schwenk, "Penetration testing tool for web services security," in *Proceedings of the IEEE 8th World Congress on Services (SERVICES '12)*, pp. 163–170, Honolulu, Hawaii, USA, June 2012.

[26] D. A. Kindy and A. K. Pathan, "A survey on SQL injection: vulnerabilities, attacks, and prevention techniques," in *Proceedings of the 15th IEEE International Symposium on Consumer Electronics (ISCE '11)*, pp. 468–471, Singapore, June 2011.

[27] R. Johari and P. Sharma, "A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection," in *Proceedings of the International Conference on Communication Systems and Network Technologies (CSNT '12)*, pp. 453–458, Rajkot, India, May 2012.

[28] M. Junjin, "An approach for SQL injection vulnerability detection," in *Proceedings of the 6th International Conference on Information Technology: New Generations (ITNG '09)*, pp. 1411–1414, Las Vegas, Nev, USA, April 2009.

[29] V. Chapela, *Advanced SQL Injection*, 2005.

[30] R. Overstreet, *Protecting Yourself from SQL Injection Attacks*, 2006.

[31] S. W. Boyd and A. D. Keromytis, "SQLrand: preventing SQL injection attacks," pp. 292–302.

[32] C. Anley, *More Advanced SQL Injection*, An NGSSoftware Insight Security Research (NISR) Publication, 2002.

[33] C. Anley, *Advanced SQL Injection in SQL Server Application*, An NGSSoftware Insight Security Research (NISR) Publication, 2002.

[34] E. H. Spafford, "OPUS: preventing weak password choices," *Computers and Security*, vol. 11, no. 3, pp. 273–278, 1992.

[35] D. P. Jablon, "Extended password key exchange protocols immune to dictionary attack," in *Proceedings of the 6th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 248–255, Cambridge, Mass, USA, June 1997.

[36] S. M. Bellovin and M. Merritt, "Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 244–250, ACM, November 1993.

[37] B. Schneier, "Attack trees," *Dr. Dobb's Journal*, vol. 24, no. 12, pp. 21–29, 1999.

[38] L. R. Knudsen and M. J. B. Robshaw, "Brute force attacks," in *The Block Cipher Companion*, Information Security and Cryptography, pp. 95–108, Springer, Berlin, Germany, 2011.

[39] M. Vieira, N. Antunes, and H. Madeira, "Using web security scanners to detect vulnerabilities in web services," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '09)*, pp. 566–571, Lisbon, Portugal, July 2009.

[40] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks," in *Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing (PRDC '07)*, pp. 365–372, Melbourne, Australia, December 2007.

[41] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic, "SecuBat: a web vulnerability scanner," in *Proceedings of the 15th International Conference on World Wide Web*, pp. 247–256, ACM, May 2006.

[42] N. Antunes and M. Vieira, "Detecting SQL injection vulnerabilities in web services," in *Proceedings of the 4th Latin-American Symposium on Dependable Computing (LADC '09)*, pp. 17–24, Joao Pessoa, Brazil, September 2009.

[43] L. Auronen, "Tool-based approach to assessing Web application security," in *Seminar on Network Security*, vol. 11, pp. 12–13, Helsinki University of Technology, 2002.

[44] E. Nordmark and R. Gilligan, IETF RFC4213, Basic Transition Mechanisms for IPv6 Hosts and Routers, 2005, http://www.ietf.org/rfc/rfc4213.txt.

[45] A. R. Choudhary, "In-depth analysis of IPv6 security posture," in *Proceedings of the 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom '09)*, November 2009.

[46] S. Szigeti and P. Risztics, "Will IPv6 bring better security?" in *Proceedings of the 30th EUROMICRO Conference*, pp. 532–537, September 2004.

[47] E. Davies, S. Krishnan, and P. Savola, IETF RFC4942, IPv6 Transition/Coexistence Security Considerations, 2007, http://www.ietf.org/rfc/rfc4942.txt.

[48] R. Priyadarshini, S. Aishwarya, and A. A. Ahmed, "Search engine vulnerabilities and threats—a survey and proposed solution for a secured censored search platform," in *Proceedings of the International Conference on Communication and Computational Intelligence (INCOCCI '10)*, pp. 535–539, Erode, India, December 2010.

[49] Wikipedia, "Application security," 2012, http://en.wikipedia.org/wiki/Application_security.

[50] D. Watson, "Web application attacks," *Network Security*, vol. 2007, no. 10, pp. 10–14, 2007.

[51] R. Radhakrishnan, M. Jamil, S. Mehfuz, and M. Moinuddin, "Security issues in IPv6," in *Proceedings of the 3rd International Conference on Networking and Services (ICNS '07)*, p. 110, Athens, Greece, June 2007.

[52] D. Yang, X. Song, and Q. Guo, "Security on IPv6," in *Proceedings of the 2nd IEEE International Conference on Advanced Computer Control (ICACC '10)*, pp. 323–326, March 2010.

[53] Y. W. Huang, C. Tsai, T. Lin, S. Huang, D. T. Lee, and S. Kuo, "A testing framework for web application security assessment," *Computer Networks*, vol. 48, no. 5, pp. 739–761, 2005.

[54] O. Security, "The exploit database," 2012, http://www.exploit-db.com/.

[55] Y. Sun, I. G. Councill, and C. L. Giles, "BotSeer: an automated information system for analyzing Web robots," in *Proceedings of the 8th International Conference on Web Engineering (ICWE '08)*, pp. 108–114, Yorktown Heights, NJ, USA, July 2008.

[56] Y. Sun, I. G. Councill, and C. L. Giles, "The ethicality of web crawlers," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, WI 2010*, pp. 668–675, Toronto, Canada, September 2010.

[57] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering," in *Proceedings of the 7th International Conference on World Wide (WWW '98)*, pp. 161–172, 1998.

[58] V. Shkapenyuk and T. Suel, "Design and implementation of a high-performance distributed web crawler," in *Proceedings of the 18th International Conference on Data Engineering*, pp. 357–368, March 2002.

[59] M. Najork, "Breadth-first search crawling yields high-quality pages," in *Proceedings of the 10th International Conference on World Wide (WWW '01)*, pp. 114–118, 2001.

[60] National Taiwan University, 2012, http://www.ntu.edu.tw/english/.

[61] N. Gaur, "Assessing the security of your web applications," *Linux Journal*, vol. 2000, no. 72, article 3, 2000.

[62] P. Noiumkar and T. Chomsiri, "Top 10 free web-mail security test using session Hijacking," in *Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology (ICCIT '08)*, vol. 2, pp. 486–490, Busan, Republic of Korea, November 2008.

[63] D. Gollmann, "Securing Web applications," *Information Security Technical Report*, vol. 13, no. 1, pp. 1–9, 2008.

[64] D. Scott and R. Sharp, "Abstracting application-level web security," in *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*, pp. 396–407, ACM Press, May 2002.

[65] D. Scott and R. Sharp, "Abstracting application-level web security," in *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*, pp. 396–407, ACM Press, May 2002.

[66] D. J. Bryce and T. C. Williams, "HTTP header intermediary for enabling session-based dynamic site searches," U.S. Patent Application 11/299, 525.

[67] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna, "Cross site scripting prevention with dynamic data tainting and static analysis," in *Proceeding of the Network and Distributed System Security Symposium (NDSS '07)*, 2007.

[68] Forum, I. IPv6 Enabled WWW Web Sites List, 2012, http://www.ipv6forum.com/ipv6_enabled/approval_list.php.

[69] C. M. Judd, G. H. McClelland, and C. S. Ryan, *Data Analysis: A Model Comparison Approach*, Routledge/Taylor & Francis Group, 2009.

[70] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: a static analysis tool for detecting web application vulnerabilities," in *Proceedings of the IEEE Symposium on Security and Privacy (S and P '06)*, pp. 258–263, Berkeley/Oakland, Calif, USA, May 2006.

[71] M. A. Howard, "A process for performing security code reviews," *IEEE Security and Privacy*, vol. 4, no. 4, pp. 74–79, 2006.

[72] P. Cisar and S. M. Cisar, "Password—a form of authentication," in *Proceedings of the 5th International Symposium on Intelligent Systems and Informatics (SISY '07)*, pp. 29–32, Subotica, Serbia, August 2007.

[73] S. Riley, "Password security: what users know and what they actually do," Usability News 8.1, 2006.

[74] X. Zheng and J. Jidong, "Research for the application and safety of MD5 algorithm in password authentication," in *Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '12)*, pp. 2216–2219, Sichuan, China, 2012.