Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 605439, 8 pages
http://dx.doi.org/10.1155/2015/605439

Research Article

Scheduling Method of Data-Intensive Applications in Cloud Computing Environments

Xiong Fu,[1] Yeliang Cang,[1] Xinxin Zhu,[1] and Song Deng[2]

[1]*School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[2]*Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*

Correspondence should be addressed to Xiong Fu; fux@njupt.edu.cn

Received 5 January 2015; Accepted 29 March 2015

Academic Editor: Emilio Insfran

The virtualization of cloud computing improves the utilization of resources and energy. And a cloud user can deploy his/her own applications and related data on a pay-as-you-go basis. The communications between an application and a data storage node, as well as within the application, have a great impact on the execution efficiency of the application. The locations of subtasks of an application and the data that transferred between the subtasks are the main reason why communication delay exists. The communication delay can affect the completion time of the application. In this paper, we take into account the data transmission time and communications between subtasks and propose a heuristic optimal virtual machine (VM) placement algorithm. Related simulations demonstrate that this algorithm can reduce the completion time of user tasks and ensure the feasibility and effectiveness of the overall network performance of applications when running in a cloud computing environment.

1. Introduction

Cloud computing has been presented as a brand new sharing network computation model of commercial resources in recent years. It has also been universally recognized as the third technology revolution, and it will continue to lead the business revolution in the coming two decades. According to the survey conducted by market research company Gartner in 2010, cloud computing has become one of the most crucial technologies for IT users. Meanwhile many IT giants have built large amount of data centers and provide cloud computing services for outside users. For example, Google has deployed 36 data centers and millions of computation nodes all over the world. The number of Microsoft's servers will be doubled in every 14 months, and there are hundreds of thousands of servers in its cloud computing data centers [1].

Currently, cloud computing centers based on the virtualization technology have become the most widely used hosting platforms for composite applications. As such, a large amount of the communication intensive applications and data-intensive applications has emerged. An application is usually divided into different subtasks. These subtasks are then allocated to virtual machines (VMs). The VMs are finally placed in specific computation nodes. Therefore, the communication traffic among subtasks and the data transmission rate between a computation node and a storage node have a great impact on the reaction and execution efficiency of the application. So guaranteeing high network performance has become the top issue in a cloud computing system and it has attracted a lot of attention [2, 3]. As the network topologies of cloud systems may be different and visualization technology itself affects the communication delay in a cloud, the execution of an application could largely depend on the network performance [4].

There are many types of applications in cloud computing systems, such as web applications (These applications are divided into different layers: web layer, application layer, and data layer, and there are communications between different layers.) and distributed applications (e.g., applications related to e-commerce or scientific computation, these applications are generally divided into multiple subtasks; there are computation behaviors within a subtask and data exchanges between one subtask and another). Communication capacity

affects the execution time and reaction time of subtasks or applications, and it is the bottleneck of multitask execution in a cloud system. As there are constraints of physical resources (mainly about CPU and memory resources), an application is usually divided into multiple subtasks. And the subtasks are distributed in computation nodes on a large scale. Therefore, the communication capacity among physical devices can greatly affect the completion time of an application.

In a current cloud computing environment, an application could be divided into multiple subtasks when deployed into a cloud computing system. Then these subtasks are assigned to VMs based on the subtasks' types; different subtasks may access different data in data centers, and there should be specific data exchanges between subtasks. As the types of applications vary, the communication behaviors among subtasks and storage nodes are different as well. For example, subtasks of a scientific computation application are running in a dependency relationship; namely, one certain subtask cannot be executed unless another specific subtask is finished. For some MPI applications, subtasks are executed with data being transmitted between them at the same time.

However, in a particular cloud system, VMs' configurations, such as the processor power, disk sizes, and memory capacities, are different. Moreover, there should be data exchanges between different VMs to finish an application, and each VM needs to access files in data centers. For example, in a web application, the data layer needs to access files in a database. The data transmission time can greatly affect the execution of an application. So it is pretty clear that how to determine the physical or logical locations of VMs plays an important role in a cloud computing system.

To handle this problem, some of the current VM placements focus on the consumption of energy; the placement of assigning VMs to the same physical node, for instance, can limit the number of running physical machines and cut down the energy consumption [5]. Other placements either focus on the CPU dimension of physical machines only [6] or concern the problem from the user's perspective like the VM placement based on SLA [7]. There are also some placements in which only physical resources are taken into account [8, 9]. However, few of them are about the communication capacity. The works in [2, 3] only focus on the communication capacity and design an overall network topological structure to improve the network performance of a cloud system, but no concrete VM placements are found in these two papers. The effect of data transmission rate between a computation node and a storage node is discussed in [10], but the communications between VMs are not. Then communication delays among VMs are also not discussed in [11] which only focuses on the data transmission between VMs and storage nodes. All of the placement algorithms that we discussed above are not taking into account the communication traffic between VMs, and therefore there is much higher completion time of applications.

In this paper, a heuristic VM placement algorithm is proposed. Not only the communications between subtasks but also the data transmissions between computation nodes and storage nodes are concerned in this algorithm, so it can effectively shorten the overall completion time of an application.

The remainder of this paper is organized as follows. We cover related work in Section 2. Section 3 presents the cloud system model. Then the heuristic VM placement algorithm is presented in Section 4. Next, we show some simulation results and analysis in four different algorithms in Section 5. Finally, Section 6 presents future work.

2. Related Work

In a cloud system, the subtasks of an application are allocated to VMs. And the key problem of dynamic resource management in a cloud computing system is how to place and manage VMs effectively. In the infrastructure layer of cloud computing, the problem of how to place VMs can be adapted into the classic problem—Bin-Packing problem. And it satisfies the condition that the number of the running physical nodes is at its minimum level and resources required by VMs in a host should not exceed the host's capacity at the same time.

The placement problem of VMs is an NP-hard variant of the N-dimensional Bin-Packing problem which has no polynomial algorithms for optimal solutions [12]. Many heuristic or greedy algorithms are introduced to get close to the global optimal solution of this problem. A lot of additional simple rules are also presented on the basis of these heuristic algorithms such as suboptimal fit, first fit, and optimal fit. In addition, many heuristic algorithms use the conventional method which features single-point search. This method tends to fall into the local area and thus cannot get the global optimal but partial solution. In some cases, this method cannot get a solution at all. Another way to get the optimal solution is using a Constraint Programming (CP) engine [8, 9, 13, 14]. Employing CP to optimize VM placement is a convenient technique of elegance and flexibility which simplifies the way of getting solutions of the problem. But the quality of constraint conditions directly affects the quality of the final solutions obtained by CP.

Current solutions to VM placement in a cloud computing environment can be divided into two categories: one focuses on the optimization of single objective while another focuses on that of multiple objectives. The single objective includes the minimum number of host nodes [5], ensuring high efficiency of service level [7], reducing VM migration times [15], cutting down the energy consumption of data centers [16], promising high availability for users [7], and decreasing the use of network I/O in cloud computing systems [17]. But defects of the single objective are also evident and some problems that the single objective deals with above are in conflict with themselves. For example, we can allocate more VMs to fewer physical hosts and shut down the idle hosts to reduce the energy consumption and management expenses. But this will lead to more VM migration times. On the contrary, to achieve the minimum migration times, there must be more physical hosts. So the strategy of multiobjective optimization has arisen [18, 19]. It is an optimization process that concerns all of these optimal conditions and makes a trade-off among them. Many of the existing VM placements

of multiobjective optimization phase to solve the VM placement problem. They only consider one objective at a time. Few of them can take multiple objectives into account at the same time. Therefore, they cannot get the globally but locally optimal solution most of the time. The work in [18] divides the VM placement problem into combinational optimization problems. It adopts genetic algorithm in dealing with the multiobjective optimization problem of VM placement and then optimizes multiple objectives which include minimizing the overall resource waste, energy consumption, and heat dissipation. But it does not take into account the overhead of VM migrations.

The performance of the solutions to the multiobjective optimization presented above, whatever Bin-Packing solutions or multistage solutions, is not good as we may wish in the aspect of time complexity. All things considered, we can make an acceptable balance between the time complexity of the algorithm and the precision of the result depending on reasonable regulations of a heuristic algorithm.

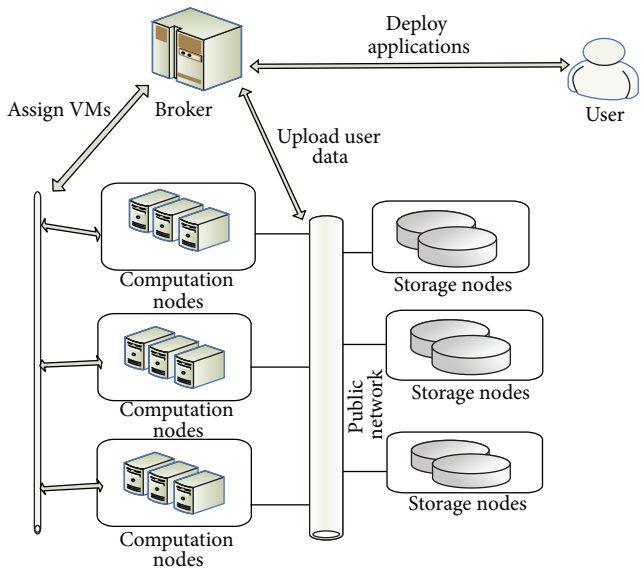

Figure 1: The structure of self-adapting resource monitoring model.

3. Problem Statement

3.1. Scenario. In a classic cloud computing system, there are lots of flexible data storage nodes, computation nodes, and brokers, and all of them can communicate with each other based on the network topology structure. Cloud users can interact with each other through the broker. They can transfer data and deploy their own applications. In this case, cloud users only need to care about the working state of applications and they do not need to care about the allocated locations of their own tasks. The scenario is shown in Figure 1.

In this scenario, cloud users request the cloud broker to deploy applications and upload data that they will use; the cloud broker allocates the related subtasks to the corresponding VMs in the computation nodes and uploads the data to the storage nodes.

3.2. Cloud Model. Before deploying applications, we assume that the related data has been uploaded to the storage nodes in advance. And the bandwidths between storage nodes, computation nodes, and the broker are already known.

As is shown in Section 3.1, there are many computation nodes and data nodes in a cloud computing system. We define S as a set that includes all of the storage nodes. Let S_i denote a certain storage node ($S_i \in S$, $1 \le i \le M$). M is the number of storage nodes. Let H denote the set of computation nodes. And H_j is a physical host that belongs to H ($H_j \in H$, $1 \le j \le N$). N refers to the number of computation nodes. H^i_{cpu} refers to the available CPUs of host H_j and H^j_{ram} is the remaining memory resource in host H_j. The sign H^j_{disk} denotes the disk size of host H_j.

The data transmission rates or bandwidths among all the physical devices in a cloud can be calculated using the function Rate $(\text{ps}, \Delta t)$. ps refers to the size of the package and Δt is the package transfer time slot.

The matrix CSH represents data transmission rates between storage nodes and computation nodes. Each element $\text{CSH}_{i,j}$ in CSH denotes the data transmission rate between computation node H_j and storage node S_i. Similarly, CHH represents data transmission rates among computation nodes, and $\text{CHH}_{a,b}$ is the data transmission rate between computation nodes a and b where $1 \le a \le M$ and $1 \le b \le M$. $\text{CHH}_{a,b} = \infty$ when $a = b$. In other words, we disregard the data transmission time among VMs (or subtasks) in the same hosts. And the equation $\text{CHH}_{a,b} = \text{CHH}_{b,a}$ means that the data transmission rate from host a to host b is the same as that from host b to host a.

3.3. Task Model. The present broker will divide an application into multiple subtasks when it receives the requests of the application. Let A denote the subtask set of the present applications, and each subtask is allocated to the corresponding VM whose resources of CPU, memory, and disk are already known. The VM is then placed on a specific computation node. What we should do is to determine the final computation node that a subtask is allocated to. $v^A_l \in A$ denotes the subtask that belongs to the application A where $1 \le l \le L$ and L denotes the number of subtasks. And the signs of R^l_{cpu}, R^l_{ram}, and R^l_{disk} denote the processor capacity, memory capacity, and disk size of the subtask v^A_l, respectively. And F is the set of files that the application A will request. f_r is the element of F ($1 \le r \le R$). R represents the number of files in F. We define T^l_{exec} as the completion time of subtask v^A_l.

Let $p^A = \{p_1, p_2, p_3, \ldots, p_l, \ldots, p_L\}$ denote the distribution path of the subtasks of A where p_l represents the allocated host of subtask v^A_l ($p_l \in H$).

We can define the file storage matrix D as

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & d_{1,3} & \cdots & d_{1,M} \\ d_{2,1} & d_{2,2} & d_{2,3} & \cdots & d_{2,M} \\ d_{3,1} & d_{3,2} & d_{3,3} & \cdots & d_{3,M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{R,1} & d_{R,2} & d_{R,3} & \cdots & d_{R,M} \end{bmatrix}. \quad (1)$$

The element $d_{r,m}$ in D represents the size of the data block of file f_r in storage node s_m ($s_m \in S, 1 \le r \le R, 1 \le m \le M$).

Let D^l denote the set of files that subtask v_l^A will request, and d_s^l is the element in D^l ($D^l \subseteq F, 1 \le s \le R$). The size of the file d_s^l can be calculated by the function Size (d_s^l); namely,

$$\text{Size}\left(d_s^l\right) = \sum_{m=1}^{M} d_{d_s^l,m}. \quad (2)$$

Let matrix CS denote the communication data sizes among all the subtasks that belong to A; namely,

$$\text{CS} = \begin{bmatrix} 0 & \text{cs}_{1,2} & \text{cs}_{1,3} & \cdots & \text{cs}_{1,L} \\ \text{cs}_{2,1} & 0 & \text{cs}_{2,3} & \cdots & \text{cs}_{2,L} \\ \text{cs}_{3,1} & \text{cs}_{3,2} & 0 & \cdots & \text{cs}_{3,L} \\ \vdots & \vdots & \vdots & \text{cs}_{x,y} & \vdots \\ \text{cs}_{L,1} & \text{cs}_{L,2} & \text{cs}_{L,3} & \cdots & 0 \end{bmatrix}. \quad (3)$$

Element $\text{cs}_{x,y}$ represents the size of data that needs to be transferred between subtask x and subtask y, and $\text{cs}_{x,y} = 0$ when $x = y$, $1 \le x, y \le L$. The equation $\text{cs}_{x,y} = \text{cs}_{y,x}$ denotes that the size of data that needs to be transferred mutually between the two subtasks is the same.

Then we can define the total data transmission time between the subtask v_l^A and all the related data files as

$$\begin{aligned} T_{\text{file}} &= \sum_{d_s^l \in D_s^l} \sum_{m=1}^{M} \frac{d_{d_s^l,m}}{\text{CSH}_{m,p_l}}, \\ \text{s.t.} \quad & 1 \le j \le N, \quad R_{\text{cpu}}^l \le H_{\text{cpu}}^j, \quad R_{\text{ram}}^l \le H_{\text{ram}}^j, \\ & R_{\text{disk}}^l \le H_{\text{disk}}^j. \end{aligned} \quad (4)$$

And the total communication time between subtask v_l^A and other subtasks that belong to A can be denoted as

$$\begin{aligned} T_{\text{task}} &= \sum_{y=1}^{L} \frac{\text{cs}_{l,y}}{\text{CHH}_{p_l,p_y}}, \\ \text{s.t.} \quad & p_y, p_l \in P^A, \quad R_{\text{cpu}}^l \le H_{\text{cpu}}^j, \quad R_{\text{ram}}^l \le H_{\text{ram}}^j, \\ & R_{\text{disk}}^l \le H_{\text{disk}}^j. \end{aligned} \quad (5)$$

3.4. Problem Definition. The problem of an application deployment can be adapted to the problem of reducing completion time of all the subtasks as much as possible. T^A denotes the overall completion time of application A. We can shorten the overall completion time based on the definitions of subtasks and the cloud model that we discussed above and finally get the allocation path P^A. The subtasks are then allocated to computation nodes based on P^A.

We can get the completion time of all subtasks that belong to A using formulas (4) and (5); namely,

$$T^A = \sum_{l=1}^{L} \left(\sum_{d_s^l \in D_s^l} \sum_{m=1}^{M} \frac{d_{d_s^l,m}}{\text{CSH}_{m,p_l}} + \sum_{y=1}^{L} \frac{\text{cs}_{l,y}}{\text{CHH}_{p_l,p_y}} + T_{\text{exec}}^l \right) \quad (6)$$

s.t.

$$1 \le i \le M, \quad (7)$$

$$1 \le l \le L, \quad (8)$$

$$1 \le y \le L, \quad (9)$$

$$1 \le m \le M, \quad (10)$$

$$\sum_{p_l=h_i} R_{\text{cpu}}^l \le H_{\text{cpu}}^{h_i}, \quad \forall i, l, \quad (11)$$

$$\sum_{p_l=h_i} R_{\text{ram}}^l \le H_{\text{ram}}^{h_i}, \quad \forall i, l, \quad (12)$$

$$\sum_{p_l=h_i} R_{\text{disk}}^l \le H_{\text{disk}}^{h_i}, \quad \forall i, l, \quad (13)$$

$$l \le y, \quad (14)$$

$$p_l \in H. \quad (15)$$

Condition (11), (12), and (13) mean that the sum of the physical resources requested by all subtasks in computation node h_i does not exceed the remaining resource capacity, such as processor capacity, memory capacity, and disk size. And condition (14) means that the size of data transferred mutually between the two subtasks is the same, so we only need to calculate a single transmission time. The problem that we ought to solve is how to get the allocation path P^A so that we can minimize the overall completion time T^A.

4. Placement Algorithm of Subtasks

The key to solve the optimization problem presented in formula (6) is to settle the placement problem of every subtask so that we can get the final path P^A of the application A. We can use a heuristic placement algorithm of subtasks to get the overall optimal placement of the application and this algorithm is called HRVP for short.

Step 1. Initialize the computation node set H, storage node set S, and data transmission rate matrices CHH and CSH based on the characteristic of the cloud system.

Step 2. Initialize the subtask set A, including computation capacity R_{cpu}^l, memory capacity R_{ram}^l, and disk size R_{disk}^l of

each subtask. Finally initialize the file set D^l and communication data matrix CS.

Step 3. We can initialize the file storage matrix D by traversing all of the storage nodes in S and the file information of each storage node.

Step 4. Let the cyclic variable Count = 0 denote the number of subtasks that have been allocated. And one more variable l is defined.

Step 5. Initialize the variable l = Count + 1 and traverse the set H to find a host $H_i \in H$ satisfying the formula

$$\text{Min}\left(\sum_{d_s^l \in D_s^j}\sum_{m=1}^{M}\frac{d_{d_s^l,m}}{\text{CSH}_{m,h_i}} + \sum_{y=1}^{\text{Count}}\frac{\text{cs}_{l,y}}{\text{CHH}_{h_i,p_y}} + T_{\text{exec}}^l\right)\&\&$$
$$R_{\text{cpu}}^l \le H_{\text{cpu}}^i \&\& R_{\text{ram}}^l \le H_{\text{ram}}^i \&\& R_{\text{disk}}^l \le H_{\text{disk}}^i. \tag{16}$$

Step 6. Calculate formula (16), get H_i, and save its value; namely, $p_{\text{Count}} = H_l$. Update the remaining physical resources of related computation nodes using the following formulas:

$$H_{\text{cpu}}^i = H_{\text{cpu}}^i - R_{\text{cpu}}^l, \qquad H_{\text{ram}}^i = H_{\text{ram}}^i - R_{\text{ram}}^l,$$
$$H_{\text{disk}}^i = H_{\text{disk}}^i - R_{\text{disk}}^l. \tag{17}$$

Step 7. Update the variable Count = Count + 1. If Count = L, go to the next step or else go back to Step 5.

Step 8. Get the final allocation path P^A.

Complexity Analysis. The time complexity of initializing the matrices D and CS is $O(M \cdot R)$ and $O(L^2)$, respectively. So the time complexity of the heuristic algorithm in Step 3 is not more than $\max\{O(M \cdot R), O(L^2)\}$.

5. Performance Evaluation

The model we presented is based on the IaaS cloud computing system which supplies cloud users with visually endless resources. The placement of the VM allocated to a subtask plays an important role in the system. It is a great challenge to implement and experiment an algorithm repeatedly in a large-scale computing infrastructure. We chose the simulation toolkit CloudSim [20, 21] as our experimental platform. This toolkit can simulate not only a variety of cloud physical resources and user tasks but also network topology of the whole cloud, and it plays an important role in our experiment.

To show the superiority of the proposed algorithm, we compare it with another two already known placement algorithms of VMs. One of the algorithms is energy efficient placement of VMs [5] whose objective is to minimize the number of running physical hosts and increase resource utilization, namely, allocating all of the VMs to the running and qualified hosts and switching off the idle hosts as many as possible. We call this algorithm MRP for short. Another VM placement policy of CloudSim, known as VMSimpleAllocationPolicy or simply as SAP, allocates the VM to the least utilized host. And all of the results obtained by using the three algorithms above are compared with the optimal solution obtained by using integer linear programming approach which is called ILP for short.

Physical Resources. There are 10 data storage nodes. The capacity of each node is 1 T and the average data transmission rate between storage nodes and computation nodes is 100 Mb/s. There are 15 computation nodes, and the configuration of each computation node is 4 CPUs, 4 GB memory, 100 GB disk size. Each computation node uses Xen as the visualization platform [22]. The average data transmission rate between computation nodes is 100 Mb/s.

We tested two applications A and B in our experiment. Application A is the Workflow App provided by the CloudSim framework and it includes three subtasks. Two of them send data package to the other subtask and each subtask has its own computation phase. There are two data files and the sizes are 500 MB, respectively. The values of the three subtasks $[R_{\text{cpu}}^l, R_{\text{ram}}^l, R_{\text{disk}}^l]$ of application A are $[1, 1024, 5120]$, $[1, 2048, 2048]$, and $[2, 1024, 5120]$, respectively. The corresponding units of the three parameters are the number of CPUs, MB, and MB.

To prove the performance and efficiency of the proposed algorithm in various environments, we assess the performance of each application from two aspects: change the communication traffic between subtasks and files gradually under the condition that the communication traffic between subtasks keeps stable; change the communication traffic between subtasks gradually under the condition that the communication traffic between subtasks and files keeps stable.

In Figure 2, we assume that the overall communication traffic between the three subtasks of application A and files is fixed to 2 GB. And when we change the overall communication traffic between the three subtasks gradually, we can see the final complete time of the application after conducting the algorithms we presented above.

The optimal solution is obtained by using integer linear programming approach which is called ILP for short. From Figure 2, we can see that HRVP has the best performance compared with MRP and SAP.

In Figure 3, the overall communication traffic between the three subtasks of application A is 2 GB. The communication traffic between the three subtasks and files is changing gradually.

From Figures 2 and 3, the HRVP has the performance most close to that of ILP compared with other two algorithms. In Figure 2, HRVP has an overwhelming advantage over MRP and SAP. MRP can improve physical resources utilization, but it has longer completion time compared with SAP and HRVP. On the contrary, the application completion time of SAP is close to HRVP in Figure 3. But SAP allocates VMs to the least utilized hosts, and therefore it will cause low utilization of physical resources.

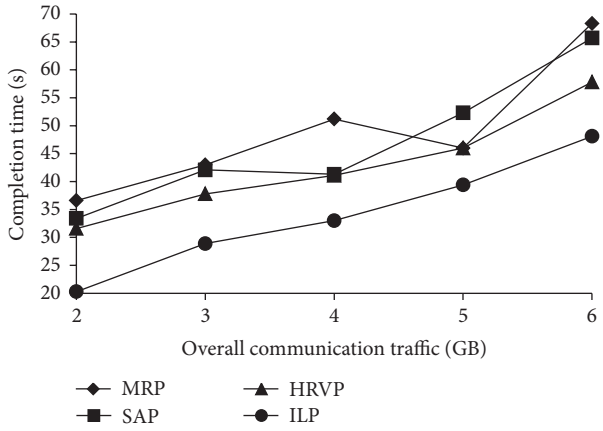

Figure 2: The completion time of application A when the communication traffic between subtasks and files keeps stable.

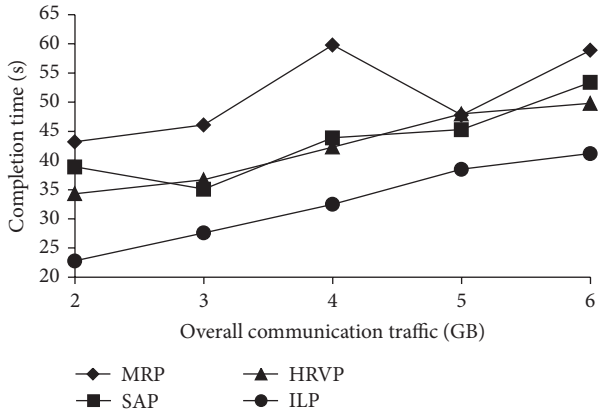

Figure 3: The completion time of application A when the communication traffic between subtasks keeps stable.

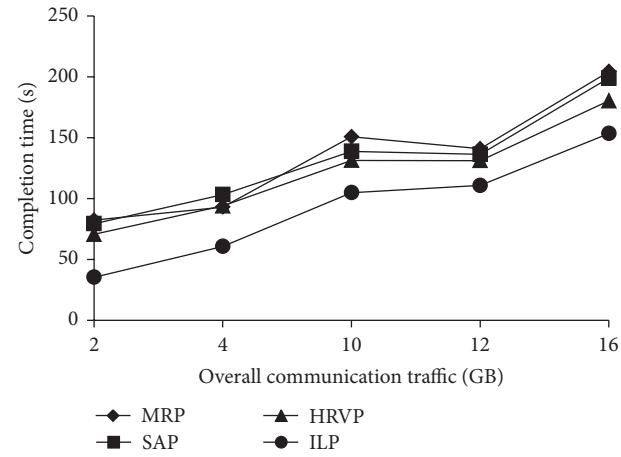

Figure 4: The completion time of application B when the communication traffic between subtasks and files keeps stable.

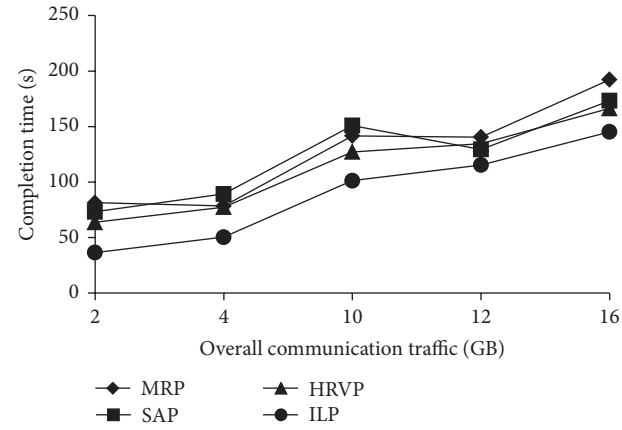

Figure 5: The completion time of application B when communication traffic between subtasks keeps stable.

Application B is an extensional application. The description is as follows: the application in this experiment includes 6 subtasks, and the physical resource value of each subtask is $[1, 2048, 2048]$, $[2, 2048, 4096]$, $[1, 1024, 5120]$, $[1, 2048, 2048]$, $[2, 1024, 5120]$, and $[2, 2048, 4096]$ where the corresponding units of the three parameters are the number of CPUs, MB, and MB.

The experiment is conducted in five groups ($abcde$). The communication traffic between all the subtasks and files in each group is unchangeable while the data communication traffic between VMs is not. And the relationship of the communication traffic between VMs in the five groups is $a < b < c < d < e$. The related five data files are distributed on the ten storage nodes in advance. And the sizes of the five data files are 657 MB, 350 MB, 500 MB, 400 MB, and 750 MB, respectively. For each subtask, its computation phase is included.

Assuming that the overall communication traffic between subtasks of B and files is fixed to 4 GB, the application completion time of B is shown in Figure 4. When the overall communication traffic between the 6 subtasks is changing gradually, we can see the final complete time of the application after conducting the algorithms we presented above.

Figure 4 shows a trend that the more the overall communication traffic increases, the closer the execution result of HRVP gets to that of the optimal solution (ILP). But the performance of MRP and SAP is getting worse. That is because MRP and SAP are not bandwidth related, and they cannot adapt to the bandwidth change.

In Figure 5, the overall communication traffic between subtasks of application B is fixed to 4 GB. Change the communication traffic between the subtasks and files gradually, and the result is shown as in Figure 5.

From the descriptions of the figures we learned that the HRVP algorithm can ensure the ideal complete time of an application in each case. The complete time is close to the optimal level. In Figures 2, 3, 4, and 5, because the bandwidths between computation nodes and storage nodes are different, the performances of MRP and SAP are unstable. But HRVP keeps a better property over the previous two algorithms. That is for the reason that HRVP is bandwidth efficient

and allocates VMs to computation nodes in consideration of the bandwidths between physical equipments like hosts or storage nodes.

6. Algorithm Analysis

In the beginning of the proposed algorithm, matrixes like CS, D, CSH, and CHH should be initialized. The time complexity of initializing these matrixes is $O(L^2)$, $O(M \cdot R)$, $O(M \cdot N)$, and $O(N \cdot N)$, respectively. Among these matrixes, CS and D will change according to a specific application. Because different applications will have different communication data sizes between their subtasks these applications may also access different files. So the main initialization part of this algorithm is about initializing matrices CS and D. When the number of subtasks of an application increases, the communication matrix will become more complex and more files will be accessed. Therefore, the complexity of the initialization time and execution time will increase.

To solve this problem, we can divide files into bigger parts, and the number of the subfiles will decrease. So the expression $O(M \cdot R)$ will get a smaller value. However, there is not a better way to simplify the process of initializing matrix CS because of the increasing number of subtasks. To compensate this weakness, the elements in CS can be sorted so that they can be accessed more quickly when used in Step 5 of the proposed algorithm.

7. Conclusions

A heuristic algorithm that targets the task placement problem of data-intensive applications in a cloud computing system is proposed in this paper. Not only the data transmission time between subtasks and storage nodes but also the communication traffic between the subtasks is taken into account. It can obtain shorter completion time of applications compared with other several algorithms. And the application completion time of the proposed algorithm is pretty close to that of the optimal solution obtained by using linear programming approach.

Although the heuristic algorithm we proposed can reduce the completion time of an application, there remains a problem that needs to be solved, namely, the increasing complexity of the initialization time and execution time because of the excessive number of subtasks. What we will do next is to simplify the initialization process and cut down the dimension of subtasks. Therefore the execution efficiency and accuracy of the algorithm can be improved.

Conflict of Interests

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

Acknowledgments

This work is sponsored by the National Science Foundation of China (nos. 61202354 and 61272422) and technological innovation fund for technology-based enterprises of Jiangsu Province in Jiangsu (BC2014195).

References

[1] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.

[2] A. Greenberg, J. Hamilton, and D. Maltz, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2009.

[3] S. Ijaz, E. U. Munir, W. Anwar, and W. Nasir, "Efficient scheduling strategy for task graphs in heterogeneous computing environment," *The International Arab Journal of Information Technology*, vol. 10, no. 5, pp. 75–86, 2013.

[4] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of Amazon EC2 Data Center," in *Proceedings of the IEEE INFOCOM*, pp. 1–9, IEEE, San Diego, Calif, USA, March 2010.

[5] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '10)*, pp. 577–578, May 2010.

[6] B. Urgaonkar, A. L. Rosenberg, and P. Shenoy, "Application placement on a cluster of servers," *International Journal of Foundations of Computer Science*, vol. 18, no. 5, pp. 1023–1041, 2007.

[7] D. Breitgand and A. Epstein, "SLA-aware placement of multi-virtual machine elastic services in compute clouds," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM '11)*, pp. 161–168, May 2011.

[8] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, pp. 331–340, May 2007.

[9] H. N. Van, F. D. Tran, and J.-M. Menaud, "Autonomic virtual resource management for service hosting platforms," in *Proceedings of the ICSE Workshowp on Software Engineering Challenges of Cloud Computing (CLOUD '09)*, pp. 1–8, Vancouver, Canada, May 2009.

[10] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Proceedings of the 9th International Conference on Grid and Cloud Computing (GCC '10)*, pp. 87–92, Nanjing, China, November 2010.

[11] K. Zamanifar, N. Nasri, and M.-H. Nadimi-Shahraki, "Data-aware virtual machine placement and rate allocation in cloud environment," in *Proceedings of the 2nd International Conference on Advanced Computing and Communication Technologies (ACCT '12)*, pp. 357–360, January 2012.

[12] X. Zhu, D. Young, B. J. Watson et al., "1000 islands: an integrated approach to resource management for virtualized data centers," *Cluster Computing*, vol. 12, no. 1, pp. 45–57, 2009.

[13] R. Dechter, *Constraint Processing*, Morgan Kaufmann, 2003.

[14] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: algorithms and applications," *European Journal of Operational Research*, vol. 119, no. 3, pp. 557–581, 1999.

[15] T. S. Kang, M. Tsugawa, J. Fortes, and T. Hirofuchi, "Reducing the migration times of multiple VMs on WANs using a feedback

controller," in *Proceedings of the IEEE 27th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW '13)*, pp. 1480–1489, Cambridge, Mass, USA, May 2013.

[16] J. Yuan, X. Jiang, L. Zhong, and H. Yu, "Energy aware resource scheduling algorithm for data center using reinforcement learning," in *Proceedings of the 5th International Conference on Intelligent Computation Technology and Automation (ICICTA '12)*, pp. 435–438, January 2012.

[17] K. Sato, H. Sato, and S. Matsuoka, "A model-based algorithm for optimizing I/O intensive applications in clouds using vm-based migration," in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*, pp. 466–471, May 2009.

[18] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proceedings of the IEEE/ACM International Conference on Green Computing and Communications*, pp. 179–188, December 2010.

[19] S. Wang, H. Gu, and G. Wu, "A new approach to multi-objective virtual machine placement in virtualized data center," in *Proceedings of the IEEE 8th International Conference on Networking, Architecture and Storage (NAS '13)*, pp. 331–335, Xi'an, China, July 2013.

[20] S. K. Garg and R. Buyya, "NetworkCloudSim: modelling parallel applications in cloud simulations," in *Proceedings of the 4th IEEE/ACM International Conference on Utility and Cloud Computing (UCC '11)*, pp. 105–113, December 2011.

[21] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. de Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[22] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 164–177, October 2003.