Hindawi
Scientific Programming
Volume 2017, Article ID 2621089, 12 pages
https://doi.org/10.1155/2017/2621089

Research Article

A Comparative Study of Reliability-Ignorant and Reliability-Aware Energy Management Schemes Using UPPAAL-SMC

Dai Shengxin, Hong Mei, and Guo Bing

College of Computer Science, Sichuan University, Chengdu 610065, China

Correspondence should be addressed to Hong Mei; hongmei@scu.edu.cn and Guo Bing; guobing@scu.edu.cn

Received 30 September 2016; Revised 19 March 2017; Accepted 24 April 2017; Published 23 August 2017

Academic Editor: Frank Hannig

Energy saving and high reliability are two key concerns in the design of real time systems. However, high reliability and low energy consumption are conflicting objects, and they are generally contrasted with temporal correctness. In this paper, we propose comparing reliability-ignorant and reliability-aware power management schemes with statistical model checking approach using UPPAAL-SMC. The power management schemes and the relevant components are modeled in the form of stochastic timed automata. And the analysis objectives are expressed as verification queries. With the model and queries as inputs, UPPAAL-SMC returns the probability of system failure and the expected value of energy consumption. In this analysis, we have considered three reliability-ignorant power management schemes and two reliability-aware power management schemes. Based on the comparative study, we provided guidelines for choosing the suitable scheme for a given system. One thing that should be emphasized is that our methodology is not limited to the schemes involved in this paper. The modeling and evaluating procedure can be applied to analyse other energy management schemes in practical application.

1. Introduction

The performance of modern computing systems has increased rapidly on account of the advancement in microprocessor technologies. The performance improvement comes at the expense of dramatically increased power consumption. Dynamic Voltage Scaling (DVS) [1] is an efficient energy management technique, which scales down the processor supply voltage and operating frequency simultaneously to save energy. Since changing the operating frequency may violate the timing constraints, Real Time DVS (RT-DVS) [2] has been proposed to achieve energy saving while preserving temporal correctness. In the recent past, a number of RT-DVS algorithms have been proposed for energy saving, which exploit the available static and/or dynamic slacks in the system. They adjust the supply voltage dynamically and collaborate with the Earliest Deadline First (EDF) or Rate Monotonic (RM) scheduling policy.

Another important concern in real time systems is reliability [3–5]. Reliability of a real time job is defined as the probability of the job being correctly executed before its deadline. For safety-critical real time systems, whenever a fault occurs, it must be detected and appropriately fixed. Otherwise, the consequence of fault can be catastrophic. Faults can be classified as transient faults and permanent faults, and the transient faults occur much more frequently than permanent faults [4]. With the continued scaling of semiconductors and advanced technologies of VLSI circuit design, practically all the digital devices will be more vulnerable to transient faults. Thus, we concentrate on transient faults in this work.

The backward error recovery is a cost effective technique, which exploits the system slack for rollback execution in order to tolerate transient faults. Obviously, since both DVS and backward recovery technique rely on the utilization of system slack, thus, there is an interesting tradeoff between system reliability and energy consumption. Moreover, it has been shown that voltage scaling has a direct and negative effect on system reliability. Scaling down voltage and frequency for energy saving could lead to dramatically increased

transient fault rates. For instance, though 63% active energy is saved by scaling down the supply voltage, it leads to approximately 200 times higher probability of failure [5]. Recently, both system reliability and energy consumption are considered in real time system design. Zhu et al. [3–5] have studied the effects of energy management schemes on system reliability and proposed two fault rate models related to frequency and voltage scaling. Based on the proposed models, several reliability-aware power management schemes have been proposed, which achieve energy saving without degrading system reliability.

The aim of this paper is comparing the reliability-ignorant and reliability-aware power management schemes, so that the impact of DVS methods on system reliability and energy consumption of the hard real time systems can be observed. In this study, the investigated schemes are static DVS scheme [2], cycle conserving DVS scheme [2], look-ahead DVS scheme [2], and reliability-aware power management (RA-PM) scheme [4]. The comparison is based on statistical model checking approach, in which the system is modeled as a network of stochastic timed automata and the analysis objectives are expressed as verification queries. Subsequently, the probability of system failure and the expected value of energy consumption are obtained by verifying the system model with respect to the corresponding queries using UPPAAL-SMC. According to the results of comparative study, we propose several guidelines for designers, so that they can decide which scheme is more suitable for a given application.

There are several advantages of our methodology. It assists in the analysis of reliability-aware power management schemes with the recent branch of UPPAAL tool suite, UPPAAL-SMC, which enables the analysis with statistical model checking. Traditionally, reliability-aware power management schemes are analysed through extensive simulations, which are time-consuming and do not guarantee the accuracy of the results. Our methodology enables analysis of power management schemes under different configurations, without prototyping and measurement for each alternative. It can be applied at the design stage, since the source codes of systems and simulator platforms are not required. In addition, it should be noted that our methodology is not limited to the schemes involved in this paper. The modeling and evaluating procedure can be applied to analyse other energy management schemes in practical application. What is more, we illustrate the usefulness of statistical model checking as a backend for analysing the reliability-aware power management schemes.

The rest of the paper is structured as follows. Section 2 introduces the analytical models for reliability-aware power management schemes. Section 3 gives a short introduction of statistical model checking and the model checker UPPAAL-SMC. Section 4 illustrates our methodology. Section 5 presents the modeling procedure in detail. Section 6 shows the comparison and analysis procedure with experiments. Section 7 reviews the related work. Finally, Section 8 concludes this paper and outlines the future work.

2. Analytical Models

2.1. Application and Platform Model. In this paper, we consider a real time application which consists of a set of independent periodic real time tasks $\Gamma = \{\tau_1, \tau_1, \ldots, \tau_n\}$. Each task is characterized by a pair (P, C), where P represents its period (which is also the relative deadline) and C denotes its worst case execution time (WCET). Task τ_i generates infinite jobs, with the first job arriving at time zero and the jth job arriving at time $(j-1)\cdot P_i$ and has a deadline of $j \cdot P_i$. The task set utilization (which is also the system utilization) is defined as $U = \sum_{i=1}^{n} u_i$, where $u_i = C_i/P_i$ is the utilization of task τ_i.

We consider the tasks running on a DVS-enabled processor, the supply voltage may vary between $V_{\min}$ and $V_{\max}$ continuously, and the operating frequency can be scaled continuously between $f_{\min}$ and $f_{\max}$ accordingly. This assumption is realistic; even if current DVS-enabled processors only have a few discrete frequencies, each desired frequency can be emulated by two adjacent frequencies or the next higher discrete frequency can be used instead. It is shown in [4] that the difference on the energy consumption for discrete frequencies and continuous frequency is within 2%. In DVS setting, the WCET C_i of task τ_i is given under the maximum operating frequency $f_{\max}$. For simplicity, we assume that the actual execution time of a task is linearly dependent on the operating frequency. That is, at the scaled frequency f_s, the actual execution time of task τ_i is assumed to be $C_i \cdot f_{\max}/f_s$. The tasks are executed on the processor according to the preemptive EDF policy, and on the basis of the feasibility condition for EDF, we assume that $U \leq 1$; otherwise, the system will be not schedulable.

2.2. Power Model. For embedded systems, the power consumption consists of three components: static power, frequency-independent active power, and frequency-dependent active power. The static power includes the power to maintain the system circuits and keep the clock running. This part of power can be removed only by turning the system off. The frequency-independent active power includes any power in the system that is independent of the operating frequency. It can be removed by putting the system into sleep states, which is achieved by dynamic power management (DPM) schemes. The frequency-dependent active power consists of any power that depends on the operating frequency. We consider that the operating frequency is approximately linearly related to the supply voltage. In this paper, we adopt the system level power model originally proposed in [3] and subsequently used in several recent papers [6, 7], where the power consumption $P(f)$ of an embedded system at frequency f is given by

$$\begin{aligned} P(f) &= P_s + \eta \cdot (P_{\text{ind}} + P_d) \\ &= P_s + \eta \cdot (P_{\text{ind}} + C_{\text{ef}} \cdot f^m), \end{aligned} \tag{1}$$

where P_s is the static power, P_{ind} is the frequency-independent active power, and P_d stands for the frequency-dependent active power. If the system is in progress, $\eta = 1$. Otherwise, when the system is turned off, $\eta = 0$. The frequency-dependent active power P_d equals $C_{\text{ef}} \cdot f^m$, where C_{ef} is the effective switching capacitance, m is the dynamic power exponent (both are system dependent constants; in general $2 \leq m \leq 3$), and f is the operating frequency.

Since energy is the integral of power over time, the energy consumed to execute a given job can be determined by

$$E = P_s \cdot D + \left(P_{\text{ind}} + C_{\text{ef}} \cdot f^m\right) \cdot \frac{c}{f}, \tag{2}$$

where D is the operation interval and c is the WCET of the task, so that c/f is the execution time of the job at frequency f. It can be seen that P_s is always consumed during the operation interval, and the energy saving gained from DVS is independent of it. Thus, for simplicity, we assume $P_s = 0$ as done in [4].

2.3. Fault Model. During the operating time, a fault may be caused by various reasons, such as hardware failures, software errors, or cosmic ray radiations. In this paper, we focus on transient faults, which are caused by cosmic ray radiations or high energy particles. And backward recovery techniques are used to tolerate them.

We adopt the fault model proposed in [3–5], in which it is assumed that the occurrence of transient faults follows a Poisson distribution and the average transient fault rate for system operating at frequency f can be expressed as

$$\lambda(f) = \lambda_0 \cdot g(f), \tag{3}$$

where λ_0 is the average fault rate corresponding to frequency $f_{\max}$ and $g(f_{\max}) = 1$ ($g(f) > 1$ for $f < f_{\max}$). Specifically, we adopt the exponential fault rate model proposed in [4, 5]:

$$\lambda(f) = \lambda_0 \cdot g(f) = \lambda_0 \cdot 10^{d(1-f)/(1-f_{\min})}, \tag{4}$$

where d is a constant indicating the sensitivity of fault rates to DVS, f is the normalized frequency, and $f_{\min}$ is the lowest frequency. The maximum fault rate is assumed to be $\lambda_{\max} = \lambda_0 \cdot 10^d$, which corresponds to the lowest frequency. Therefore, it can be observed that reducing the supply voltage leads to increased failure rate and increased execution time; both will result in higher possibly of failure.

2.4. Reliability-Ignorant and Reliability-Aware Power Management. According to (2), lower frequency leads to less frequency-dependent active energy consumption. However, low frequency scales the application execution time, which results in more static and frequency-independent active energy. Therefore, there exists an energy-efficient frequency f_{ee}; based on (1) and (2), it is easy to find out that [4]

$$f_{\text{ee}} = \sqrt[m]{\frac{P_{\text{ind}}}{C_{\text{ef}} \cdot (m-1)}}. \tag{5}$$

Consequently, for energy efficiency, in this paper, the processing frequency is limited to the range $[f_{\text{ee}}, f_{\max}]$.

Reliability-Ignorant Power Management Schemes. These schemes propose to utilize DVS for energy saving as much as possible, ignoring the concern of reliability, which may dramatically increase the probability of failure. There are three kinds of these schemes, namely, static DVS, cycle conserving DVS, and look-ahead DVS. In this paper, we use static EDF [2], ccEDF [2], and laEDF [2] on behalf of each kind of scheme.

Reliability-Aware Power Management Schemes. These schemes were originally proposed in [4, 5], which achieve energy saving while preserving system reliability. Zhu and Aydin [4] proposed reliability-aware task-level and job-level power management schemes, which exploit the available slack to schedule a recovery job and utilize the remaining slack for energy saving. It is shown in [4] that there exists an optimal value $X_{\text{opt}} = \text{sc} \cdot ((P_{\text{ind}} + C_{\text{ef}})/(m \cdot C_{\text{ef}}))^{1/(m-1)}$ that minimizes the total fault free energy consumption. RA-PM schemes select the tasks among the task set, where the summation of the selected tasks should be closed to X_{opt}. And the selected tasks are scaled down appropriately to minimize energy consumption.

3. Statistical Model Checking and UPPAAL-SMC

Statistical model checking [8] is one of the quantitative verification approaches, which efficiently addresses verification problem by combining the Monte Carlo method with temporal logic model checking [9]. It automatically samples the traces (or simulations) of the system model, checks whether the samples satisfy or violate the property, and finally applies a statistical estimation technique to compute an approximate value for the probability that the property is satisfied [9]. The estimation is correct up to some confidence level that can be set by user.

UPPAAL-SMC [10] is a stochastic and statistical model checking extension of UPPAAL toolbox for verification of stochastic hybrid systems. The modeling formalism of UPPAAL is an extension of classical timed automata with additional features such as integer variables, data structures, user-defined functions, and synchronization. And the modeling formalism of UPPAAL-SMC is based on a stochastic interpretation and extension of the timed automata used in UPPAAL; it has been extended to handle stochastic and nonlinear dynamic behaviors, discrete probabilities, a stochastic interpretation for timed delays, and even dynamic process creation [10]. In UPPAAL-SMC, a system is represented as a network of interacting stochastic timed automata (STA), which communicate via broadcast channels and shared variables [10]. Three types of queries are supported by UPPAAL-SMC, that is, probability estimation, hypothesis testing, and probability comparison. The probability confidence interval can be estimated by the query Pr[bound](φ), the hypothesis testing is achieved by the query Pr[bound](φ) >= p (where p is a specified threshold), and probability comparison is achieved by the query Pr[bound$_1$](φ_1) >= Pr[bound$_2$](φ_2), where *bound* defines how to bound the runs. There are three ways to bound the runs including by time (specifying <= N, where N is positive integer), by cost (specifying x <= N, where x is a specific clock), and by number of steps (specifying # <= N) [10].

4. Illustration of Our Methodology

The block diagram of our methodology is shown in Figure 1, in which dash lines represent manual steps, while solid lines

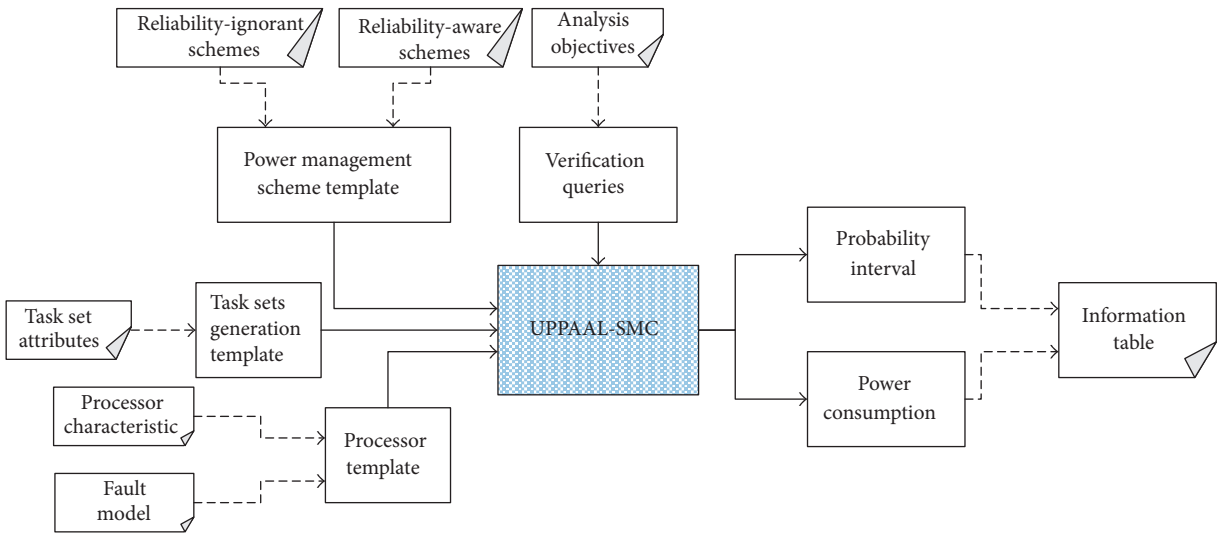


Figure 1: Block diagram of our approach.

represent automated steps. That is, the modeling procedure of all the templates and the expressing of verification queries are done manually.

Four inputs are required for analysing power management schemes. Each power management scheme is modeled as an individual template (the model is called template in UPPAAL-SMC) in form of stochastic timed automata. The task sets generation template generates the task sets to which the schemes can be applied. It consists of a task attributes generator template that produces timing attributes of each task, and the task template captures the behavior of real time tasks. The processor template captures the behavior of the processor and records the variation of energy consumption. What is more, the analytical fault model is also implanted in the processor template, which captures the occurrence of transient faults. The analysis objectives are expressed as verification queries using temporal logic. With the templates and queries as inputs, UPPAAL-SMC returns the probability of system failure and the expected value of energy consumption. For each power management scheme, we alter the parameters in task set attributes and invoke the verification process repetitively. It should be noted that the task attributes generator template is involved in every verification process. Thus, all the schemes are applied to task sets with the same attributes instead of prior generated task sets. This is necessary to obtain unbiased results, since prior generated task sets may be biased. We record the results provided by UPPAAL-SMC in information tables, so that further analysis can be accomplished. Finally, we visualize the information tables in form of line charts, so that the difference between reliability-ignorant and reliability-aware power management schemes can be observed. In addition, the impact of task set attributes upon different power management schemes can also be observed.

5. Modeling in UPPAAL-SMC

In this section, we introduce the formal framework for analysing reliability-aware power management schemes. Each template is described in detail, with the channels and shared variables; UPPAAL-SMC will compose all the templates so as to obtain the entire model for analysis.

The templates of reliability-ignorant power management schemes are not depicted in this paper. This is because we had depicted these templates in the previous paper [11], so we only depict the template of reliability-aware power management in this section.

5.1. Task Attributes Generator. The generator template generates the timing attributes of each task. Two key parameters for task attributes generation are the task set utilization Uti and cardinality number N. The generator sets the utilization of each task; meanwhile, it guarantees the task set utilization $\sum_{i=1}^{n} u_i$ = Uti, where u_i is the utilization of τ_i. Once a task's period has been chosen, its WCET and WCEC (worst case execution cycles, that is, the value obtained by multiplying WCET by the maximum frequency) can be obtained subsequently.

As shown in Figure 2, UUnifast [12] algorithm is implemented in form of UPPAAL template, which takes two parameters, task set utilization Uti and cardinality number N, respectively. In the generator, tasks' periods are randomly selected in the range of [5 ∗ PERIOD, 10 ∗ PERIOD], where PERIOD is the task cardinality period. This model iteratively generates the attributes of each task with respect to the task set parameters.

5.2. Real Time Task. The task template is depicted in Figure 3, which takes a single parameter, namely, the task identifier tid that is used to index the task attributes arrays as well. Each task is represented by an instance of the task template

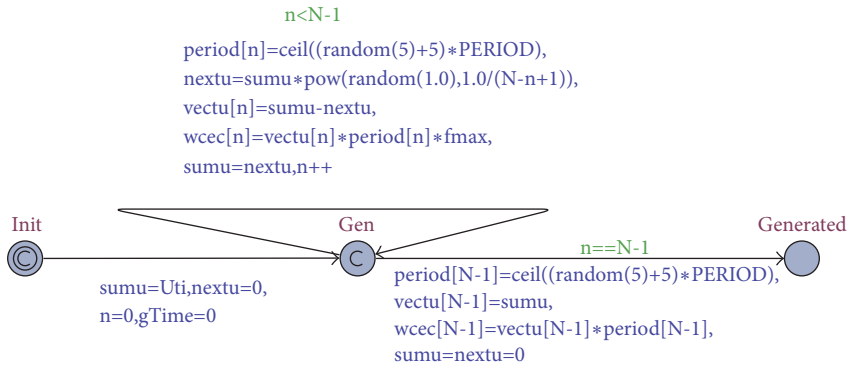


Figure 2: Task attributes generator template.

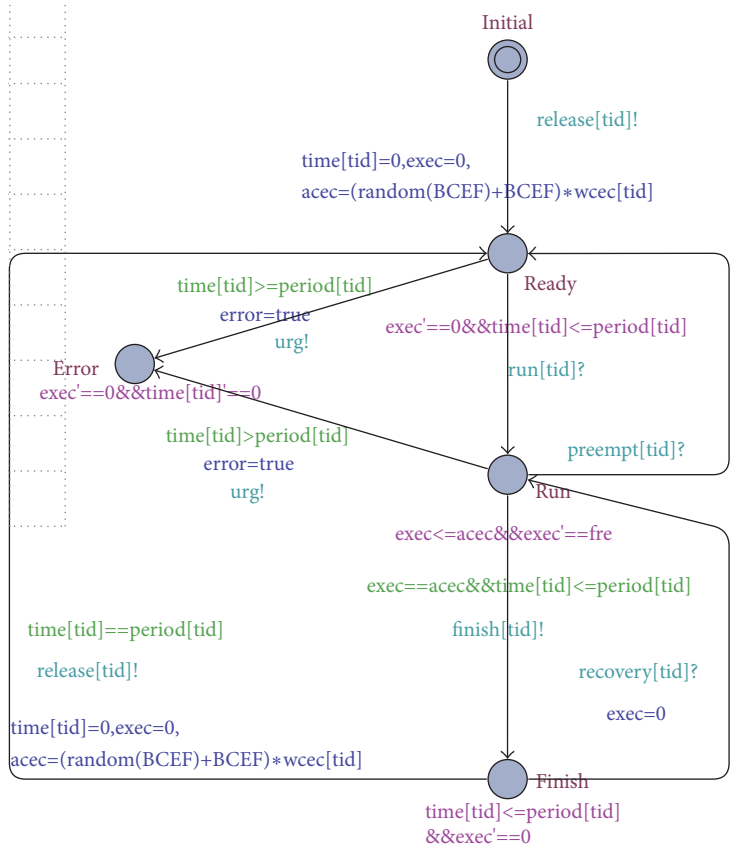


Figure 3: Task template.

equipped with the timing attributes generated by the generator. The task identifier is used as the index to associate the template and the attributes array.

Whenever a job (an instance of a task, and all the jobs of one task share the same task identifier) is released, it signals the scheduler that it is ready for execution using release[tid]. Meanwhile, two clock variables called time[tid] and exec are initialized, where time[tid] represents the time since the beginning of the current period and exec is a stopwatch used for counting the execution cycles in current period. Moreover, the actual execution cycles, acec, of this job are calculated with formula acec = (random(BCEF) + BCEF) ∗ wcec[tid]; that is, acec is a value in the range of [BCEF ∗ WCEC, WCEC], where BCEF (in general, $0 \leq \text{BCEF} \leq 1$) stands for the best case fraction of worst case execution cycles.

The invariant on location *Ready* indicates that exec does not progress (exec == 0); that is, the job is not running, and it cannot stay in this location longer than the task's period. The location *Run* corresponds to busy executing, in which the growth rate of exec is fre, that is, the current normalized frequency. When exec is equal to acec, the template moves to *Finish* location and stays in this location until the next

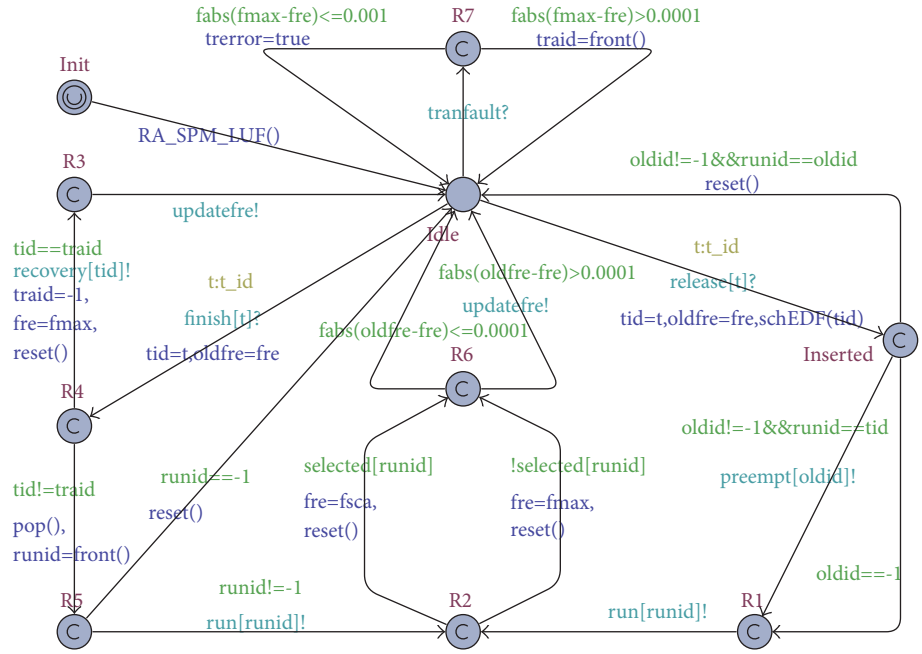


Figure 4: RA-SPM-LUF template.

period starts. When the job is running, it may be preempted by a higher priority job through preempt[tid]. Whenever a job misses its deadline, that is, time[tid] > period[tid], the template moves to *Error* location and sets the global variable *error* to TRUE.

If the task was selected by RA-DPM to scale down the frequency and a transient fault was detected at the end of the scaled job's execution, a recovery job is activated immediately. This scenario is modeled as a transition from *Finish* location to *Run* location, which is triggered by recovery[id] signal.

5.3. RA-PM Scheme. The RA-PM scheme template manages an array that keeps the released tasks' identifier. This template dispatches tasks to run on the processor by means of managing the order of elements in the array.

The RA-SPM-LUF() [4] scheme template is depicted in Figure 4. Initially, the template invokes RA-SPM-LUF() function to initialize all the elements in the array to −1, subsequently computing the optimal utilization (X_{opt}) and scaled frequency f_{sca}; finally the function selects tasks to meet X_{opt} based on largest utilization task first policy. If task τ_i is selected then set selected[i] = true. The template of RA-SPM-SUF [4] scheme is the same as this template, except that, in the initialization function, the tasks are selected based on smallest utilization task first policy. After initialization, the template stays in *Idle* location. Upon receiving a *release* signal, the template will record the current running job's identifier and operating frequency in *oldid* and *oldfre*, respectively. Then, the template puts the incoming job's identifier into the array based on EDF policy; this is achieved by schEDF() function. Afterwards, the template alters the state of the relevant jobs: if the incoming job has higher priority than the running job, the scheduler preempts the running job and dispatches the incoming job to execute; however if the array is empty, the incoming job is dispatched to execute; otherwise, no operation is performed.

If the running task was selected by RA-SPM-LUF scheme, the processor will operate at f_{sca}. Whenever the operating frequency is changed, an *updatefre* signal is released in order to inform the processor template.

When the processor is operating, a transient fault may occur. If the processor is operating at its maximum frequency (no matter whether the running job is a normal job or a recovery job), a system failure is reported. Otherwise, the template records the current running job's identifier in *traid*.

Upon receiving a *finish* signal, if the task is a preselected task and a transient fault had occurred, a recovery job will be activated. Otherwise, the template pops the job out of the array and selects the next highest priority job (if any) to execute.

5.4. Processor. As shown in Figure 5, the processor template alternates between *Idle* and *Run* locations which indicate the processor being free and busy running, respectively. One variable named *Energy* is defined to capture energy consumption. The invariants on both locations describe the variation of the energy consumption based on formula (1).

Since the occurrence of transient faults follows a Poisson distribution, the interval time between two faults will follow an exponential distribution. We define a variable t to represent the time since the last occurrence of a fault and a function $F\exp(\text{fre})$ that computes the actual delay at frequency fre. The function computes delay with formula $-\ln(\text{random}(1.0))/\lambda$, where random(1.0) returns a number in $[0, 1]$ with the uniform distribution and λ is the average fault rate corresponding to frequency fre.

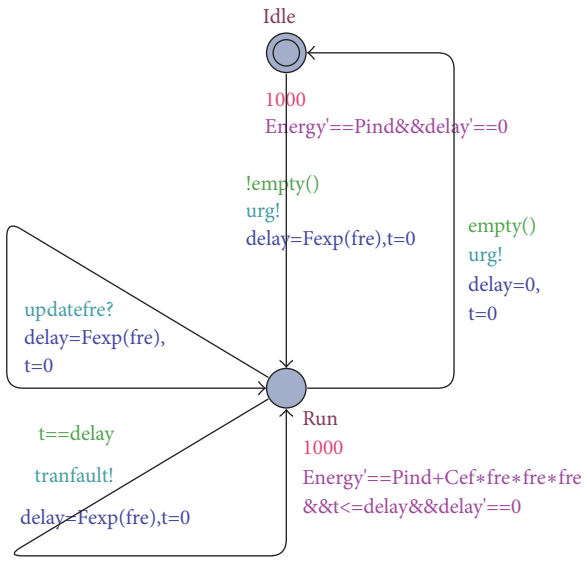


Figure 5: Processor template.

Whenever the processor frequency is changed or a transient fault has occurred, *delay* is recalculated and t is reset to 0. When a transient fault occurs, the template releases a *tranfault* signal. After receiving a *tranfault* signal, RA-DPM template decides to report a system failure or activate a recovery job.

6. Comparison of the Power Management Schemes

In this section, we compare the power management schemes under different task set configurations. In addition, the impact of task set attributes upon these schemes is also investigated.

For each power management scheme, we alter the parameters in task set attributes and invoke the verification process repetitively. In each experiment, we alter one parameter and keep the remaining parameters constant, so that the impact of that parameter upon different schemes can be observed. We compare the system reliability and energy consumption under each power management scheme, and basic EDF policy is used as a baseline.

UPPAAL-SMC is utilized to verify the model with respect to specified queries, so that the probability of failure (RA-DPM template reports a transient fault) and power consumption can be obtained. Besides RA-DPM-LUF and RA-DPM-SUF templates, we have implemented the template of EDF, staEDF, ccEDF, and laEDF for comparison.

The two analytical objectives of our approach are the probability of failure and the energy consumption. These two objectives are expressed in form of UPPAAL-SMC query language as follows.

The probability interval of having a system failure can be obtained by the following query:

$$\Pr\left[<= T\right](<> \text{trerror}), \tag{6}$$

where trerror is a bool variable that changes to true when a system failure occurrs. This query computes the probability interval of the occurrence of a system failure up to T time units.

What is more, the energy consumption can be obtained with the following query:

$$\Pr\left[\text{Energy} <= \text{MAX}\right](<> g\text{Time} >= T), \tag{7}$$

where MAX is a constant larger than the reachable range of the energy consumption of all runs. This query computes the probability interval of reaching time beyond T within a large energy bound, which is trivially true. But in addition, the tool provides the mean value of *Energy* variable, which is a key metric in comparison.

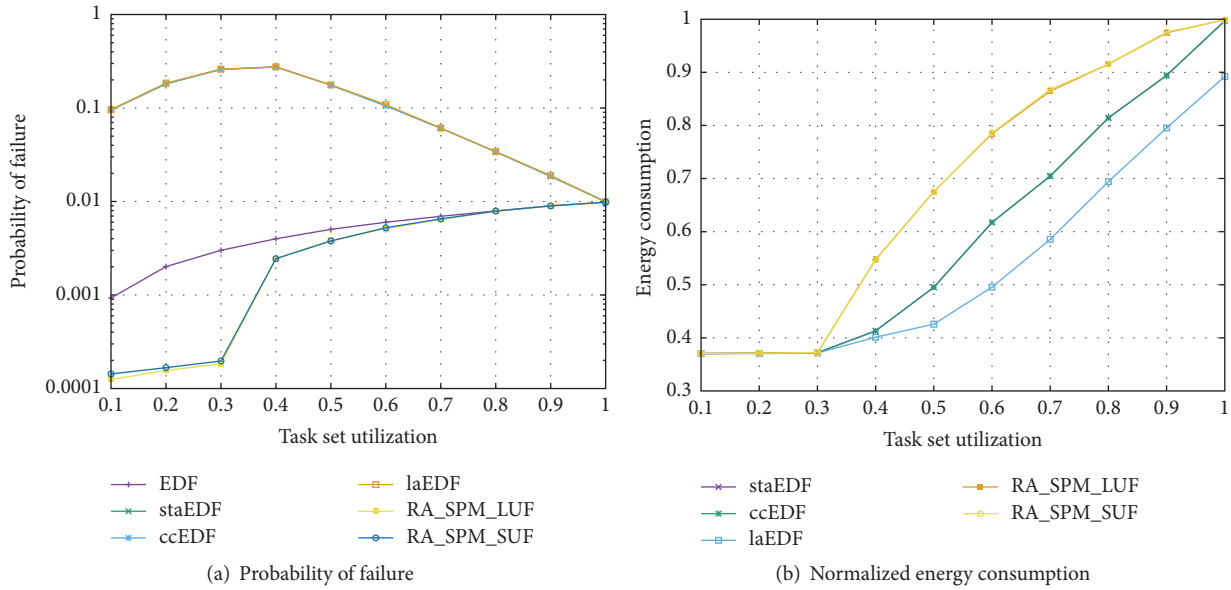
In our analysis, task sets are automatically generated with the generator template. The task set utilization is varied with a step size of 0.1 within the range of $[0.1, 1.0]$. The task set cardinality number is 10 and the task period cardinality is 100. The same task set attributes have also been used in [4]. For processor parameters, we use normalized frequencies with $f_{\max} = 1$ and assume the frequency can be varied continuously. Moreover, we assume $P_{\text{ind}} = 0.1$, $C_{\text{ef}} = 1$, and $m = 3$ in formula (1) and the average fault rate as $\lambda_0 = 10^{-4}$ at $f_{\max}$ based on the data provided in [4, 5]. With these processor characteristics, the minimum energy-efficient frequency $f_{\text{ee}} = 0.37$.

The parameters for statistical model checking are set as follows: the probability of false negatives $\aleph = 0.001$ and the probability uncertainty $\varepsilon = 10^{-5}$, so that the result is correct up to the confidence level $(1 - \aleph)$ and the confidence interval width is less than 2ε. And we set the time bound $T = 100000$ as done in [4].

All the experiments are performed by a machine learning enhanced version of UPPAAL-SMC; the average runs required to get the expected accuracy are about two hundred thousand. And the experimental platform is as follows: CPU: Intel E5-2600v4; memory: DDR4 128 G; OS: Red Hat Linux 6.5. The uppaal code is available in https://github.com/Chelseeeea/model-checking.

6.1. Experiment Results and Analysis. In the first experiment, we compare the schemes under different task set utilizations. The task set utilization is varied, and the remaining parameters are kept constant. We assume that all jobs consume their WCEC; that is, acec = wcec in the task template. Therefore, under this configuration, only the static system slack is available; thus ccEDF and staEDF have the same behavior. This assumption is made in order to eliminate the potential impact of actual execution cycles on the actual task set utilization.

As shown in Figure 6(a) (note the log scale of y-axis), compared to reliability-ignorant schemes, reliability-aware schemes reduce the probability of failure distinctly. As the task set utilization increases, the probability of failure under classic EDF increases linearly. The main reason is that, with the increased task set utilization, the executing time of tasks increases, which increases the probability of occurring transient fault. Compared to classic EDF, the probability of failure



(a) Probability of failure

(b) Normalized energy consumption

Figure 6: Probability of failure and normalized energy consumption under different task set utilization.

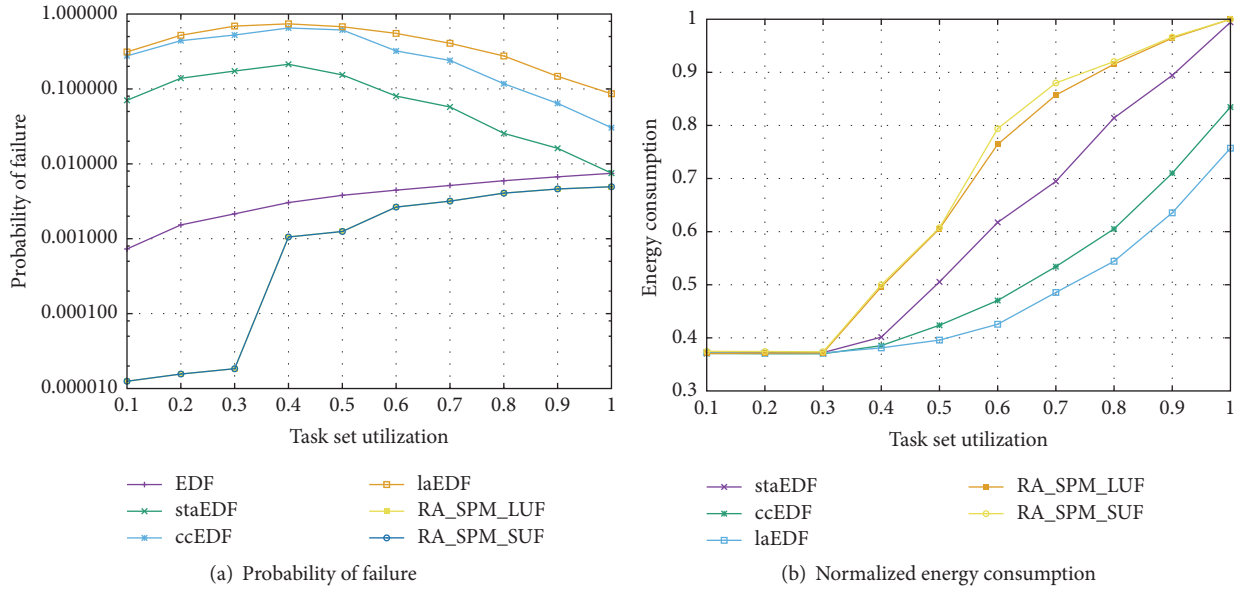
under reliability-ignorant power management schemes is relatively larger; this is because these schemes scale down the operating frequency that increases the fault rate and extends jobs' execution time; both will increase the probability of failure. Comparatively, reliability-aware schemes preserve the system reliability, meanwhile achieving reasonable energy saving. It should be noted that when the task set utilization is low (e.g., $U < 0.37$), tasks always run at f_{ee} and the probability of failure increases slightly with increased utilization. As the task set utilization continues to increase, this results in operating frequency increasing, which has lower fault rates and thus leads to a lower probability of failure. Besides, the probability of failure under laEDF is negligibly larger than that under ccEDF. However, it is indiscernible in the figure.

Figure 6(b) illustrates the normalized energy consumption under different schemes with classic EDF as baseline. It can be seen that when the task set utilization is low, all the schemes have the similar normalized energy consumption. When the task set utilization is higher than 0.37, reliability-aware schemes always consume more energy than staEDF and laEDF. This is because the reliability-aware schemes reserve part of system slack for potential backward recovery; thus there is less spare capacity available for energy management.

In the second experiment, we compare the schemes when dynamic system slack is available. Thus, we relieve the limitation on the actual execution cycles of jobs. We assume that acec of jobs are in the range of [0.5 ∗ WCEC, WCEC]; that is, BCEF = 0.5 and acec = (random(0.5) + 0.5) ∗ *wcec* in the task template. Comparing Figure 6(a) to Figure 7(a), it can be observed that when the dynamic system slack is available, for ccEDF and laEDF, the probability of failure increases dramatically. This is because ccEDF and laEDF utilize the dynamic system slack to further scale down the operating frequency that leads to a higher probability of failure. And it can be seen that the aggressive algorithm laEDF has obviously larger probability of failure than staEDF. This is because laEDF pushes the jobs as much as possible and executes the jobs with the lowest frequency that ensures the deadline, which increases the probability of failure. Moreover, as shown in Figure 7(b), the normalized energy consumption decreases under ccEDF and laEDF and that under the remaining schemes is not highly effected; this is because the former two schemes utilize the dynamic system slack generated by early termination to save energy.

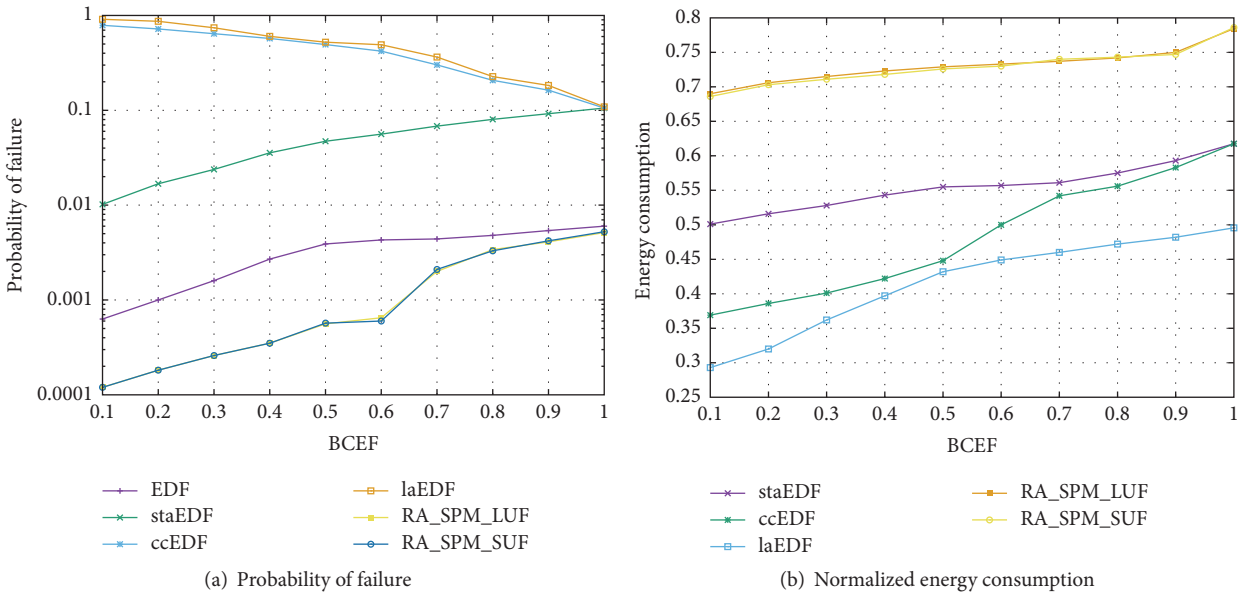
In the third experiment, we further compare the schemes when different amount of dynamic systems slack is available. We vary the best case fraction (BCEF) of worst case execution cycles, with step size of 0.1 in the range of [0.1, 1.0], and keep the task utilization equal to 0.6. It is shown in Figure 8(a) that as the BCEF increases, the probability of failure under ccEDF and laEDF dramatically decreases, and that under staEDF, classic EDF, and reliability-aware schemes linearly increases. Moreover, as shown in Figure 8(b), the normalized energy consumption slightly increases. This is because, for ccEDF and laEDF, when BCEF is low, more dynamic slack can be utilized to scale down the operating frequency that increases the probability of failure and decreases the energy consumption. The remaining schemes do not utilize the dynamic slack, so that as BCEF increases, the jobs' actual execution time increases accordingly which leads to larger probability of failure.

In the last experiment, we compare the schemes when more tasks are in the task sets. We change the task set cardinality number to 20 and keep BCEF equal to 1, meanwhile, varying the task set utilization. Comparing Figure 6 to Figure 9, it can be observed that, with increased task set cardinality number, regardless of the schemes, the probability of failure is dramatically decreased and the normalized energy consumption is not highly affected. And the shape of



(a) Probability of failure (b) Normalized energy consumption

FIGURE 7: Probability of failure and normalized energy consumption under different task set utilization with random actual execution cycles.



(a) Probability of failure (b) Normalized energy consumption

FIGURE 8: Probability of failure and normalized energy consumption under different BCEF.

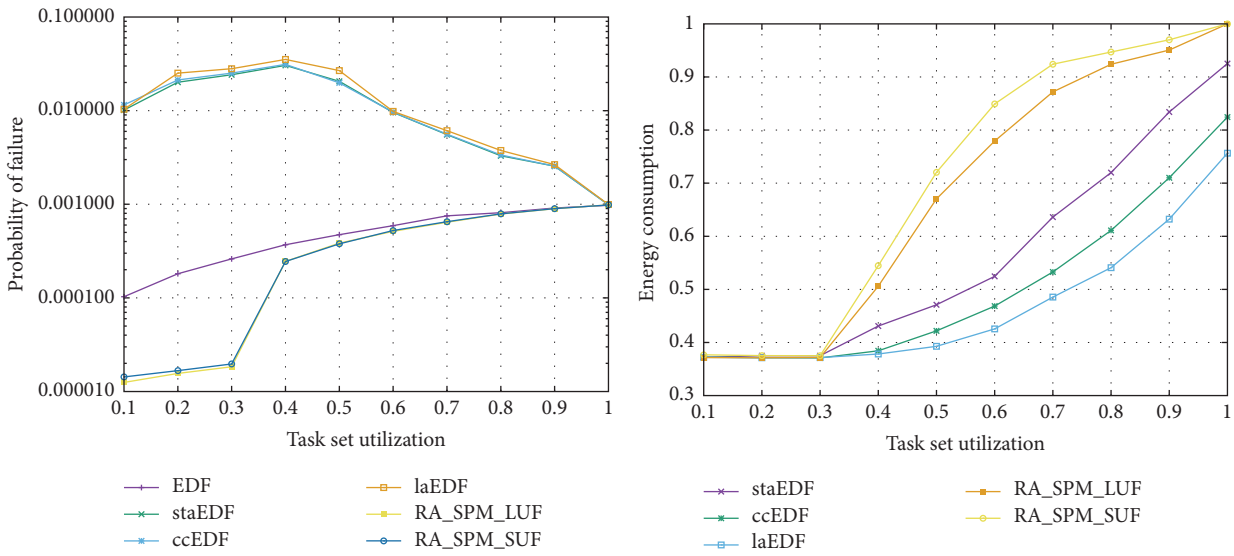
the curves is preserved. This is because, with the increased task set cardinality number, each task has smaller utilization that leads to shorter execution time and thus decreases the probability of failure. However, the normalized energy consumption mainly depends on the task set utilization; it will not be highly affected by task set cardinality number.

6.2. Summary and Guidelines. The results obtained in the experiments can be confirmed by the results proposed in [4, 5], which demonstrate the correctness and usability of our methodology. According to the obtained results, we have the following summaries:

(i) The most impacting factors of system reliability are the task set utilization and cardinality number.

(ii) The best case fraction (BCEF) of worst case execution cycles will highly affect the system reliability under dynamic power management schemes (ccEDF and laEDF), but it linearly affects the static power management schemes.

(iii) The normalized energy consumption is mainly influenced by task set utilization, and BCEF will highly affect that under dynamic power management schemes.



(a) Probability of failure

(b) Normalized energy consumption

Figure 9: Probability of failure and normalized energy consumption with increased task set cardinality number.

In consequence, we would like to give the following guidelines:

(i) When system reliability is the major concern and the system workload is low, reliability-aware schemes are the proper methods.

(ii) Static power management schemes are more stable in the context of reliability, since they are not highly affected by the variation of BCEF.

(iii) The most convenient way to improve system reliability is decomposing the big tasks into small tasks, since increasing the task set cardinality number dramatically decreases the probability of failure.

(iv) When energy consumption is the major concern, the dynamic power management schemes are the proper methods.

7. Related Work

In [13], Aminzadeh and Ejlali used analytical models to explore and compare four different energy management methods. They proposed to use Markov models to obtain analytical models for reliability and energy consumption. Hardware replication is used to achieve fault tolerance under their assumption. In their paper, they considered a system with two identical processors. One processor is used as a primary unit and the other one as a backup. Based on the consideration above, they proposed to use Markov models to obtain analytical models for reliability and energy consumption. They considered four existing methods: classic dynamic power management (DPM), classic DVS, postponement method, and hybrid method. They have provided reasonable results with the analytical models. By contrast, we consider the system with one processor and use backward recovery to tolerate the fault. Their approach is based on analytical models and only considers a single period of job execution, which limits its applicability. In general, their work is a theoretical study under idealized consideration, and it is based on mathematical formulas and calculation. Comparatively, the objective of our approach is to provide a practical methodology for analysis of the reliability-aware power management schemes.

In [3], Zhu et al. explored the effects of energy management on system reliability for real time embedded systems. They considered two fault rate models, the linearly decreasing fault rate model under frequency scaling and the exponentially increasing fault rate model under voltage scaling. After that, in [4], they proposed a RA-PM framework for periodic real time tasks and studied task-level utilization based static RA-PM schemes and a job-level dynamic RA-PM scheme. Checkpointing techniques are further explored to more efficiently use the dynamic slack in [5]. In [14], Mittal and Vetter presented a survey of techniques for improving the reliability of computing systems. They underscored the crucial importance of reliability in future systems and identified research directions that merit further exploration. These papers presented the basic theories of reliability-aware power management. All the analytical models provided in these papers are the foundation of our approach, and their experimental results are used as the criterion to judge the correctness of our approach.

In [11], Dai et al. illustrated the applicability of statistical model checking to the area of RT-DVS algorithms evaluation. The relevant components involved in RT-DVS algorithms evaluation are modeled as stochastic timed automata and implemented in statistical model checker UPPAAL-SMC. There are four evaluation metrics in their approach, namely, the utilization bound, energy efficiency, battery awareness, and temperature awareness. This is the prepositive study of our approach, which evaluates the power management schemes without considering system reliability.

Statistical model checking is powerful approach that has been widely used in verification and analysis of various systems. In [15], Zhou et al. proposed to verify the underlying consistency with the specification for service oriented systems using UPPAAL-SMC. In [16], Boudjadar et al. proposed to utilize UPPAAL-SMC for schedulability analysis of hierarchical scheduling systems. In [17], Nouri et al. presented SBIP, an extension of BIP for stochastic systems, and proposed a statistical model checking approach to verify the systems described by SBIP. In [18], Zuliani reviewed recent work on the use of statistical model checking techniques for biological applications. In [19], Dal Corso et al. proposed to utilize UPPAAL-SMC to compare two ad hoc routing protocols under the possibly lossy communication scenario. In [20], Grosu et al. explored the usability of statistical model checking for measuring quantitative properties of systems. In [21], Legay and Viswanathan reviewed the existing results on statistical model checking and discussed its research direction.

8. Conclusion and Future Work

Energy saving and high reliability are two major concerns in hard real time systems. Many RT-DVS algorithms have been proposed to achieve energy saving while guaranteeing the time constraint. However, these reliability-ignorant power management schemes will increase the probability of failure. Reliability-aware power management schemes are proposed to achieve energy saving and preserve reliability at the same time.

In this paper, we proposed to compare the reliability-ignorant and reliability-aware power management for hard real time systems. We illustrated the applicability of statistical model checking for analysing reliability-aware power management schemes. We have compared the system reliability and normalized energy consumption under different power management schemes. Our methodology enables analysis of power management schemes under different configurations, without prototyping and measurement for each alternative. And the modeling and evaluating procedure can be applied to analyse other energy management schemes in practical application.

Our results show that the most impacting factors of system reliability are system workload and task number, and system workload is the key factor of energy consumption. We have proposed some guidelines for the designers, so that they can decide which scheme is more suitable for a given application.

A suggestion for continuing this research area is to consider the job-level reliability-aware schemes. Another further work is to consider the task sets with precedence constraints.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the State Key Program of National Natural Science Foundation of China under Grant no. 61332001, the National Natural Science Foundation of China under Grant nos. 61772352 and 61472050, and the Science and Technology Planning Project of Sichuan Province under Grant nos. 2014JY0257, 2015GZ0103, and 2014-HM01-00326-SF.

References

[1] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of energy-cognizant scheduling techniques," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1447–1464, 2013.

[2] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *ACM SIGOPS Operating System Review*, vol. 35, no. 5, pp. 89–102, 2001.

[3] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," in *Proceedings of the 2004 IEEE/ACM International Conference on Computer-aided Design, ICCAD' 04*, pp. 35–40, IEEE Computer Society, Washington, DC, USA, 2004.

[4] D. Zhu and H. Aydin, "Reliability-aware energy management for periodic real-time tasks," in *Proceedings of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS'07*, pp. 225–235, April 2007.

[5] D. Zhu, "Reliability-aware dynamic energy management in dependable embedded real-time systems," *ACM Transactions on Embedded Computing Systems*, vol. 10, no. 2, pp. 1–27, 2011.

[6] Z. Li, S. Ren, and G. Quan, "Energy minimization for reliability-guaranteed real-time applications using DVFS and checkpointing techniques," *Journal of Systems Architecture*, vol. 61, no. 2, pp. 71–81, 2015.

[7] M. Lin, Y. Pan, L. T. Yang, M. Guo, and N. Zheng, "Scheduling co-design for reliability and energy in cyber-physical systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 2, pp. 353–365, 2013.

[8] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: an overview," in *Runtime Verification: First International Conference, RV 2010, St. Julians, Malta, November 1–4, 2010. Proceedings*, vol. 6418, pp. 122–135, Springer, Berlin, Germany, 2010.

[9] E. Clarke and P. Zuliani, "Statistical model checking for cyber-physical systems," in *Automated Technology for Verification and Analysis: 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11–14, 2011. Proceedings*, T. Bultan and P.-A. Hsiung, Eds., vol. 6996, pp. 1–12, Springer, Berlin, Germany, 2011.

[10] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal SMC tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015.

[11] S. Dai, M. Hong, B. Guo et al., "A formal approach for RT-DVS algorithms evaluation based on statistical model checking," *Mathematical Problems in Engineering*, vol. 2015, Article ID 815230, 12 pages, 2015.

[12] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.

[13] S. Aminzadeh and A. Ejlali, "A comparative study of system-level energy management methods for fault-tolerant hard real-time systems," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 60, no. 9, pp. 1288–1299, 2011.

[14] S. Mittal and J. S. Vetter, "A survey of techniques for modeling and improving reliability of computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1226–1238, 2016.

[15] Y. Zhou, J. Ge, P. Zhang, and W. Wu, "Model based verification of dynamically evolvable service oriented systems," *Science China Information Sciences*, vol. 59, no. 3, 2016.

[16] A. Boudjadar, A. David, J. H. Kim et al., "Statistical and exact schedulability analysis of hierarchical scheduling systems, Science of Computer Programming," in *Proceedings of the 1th International Symposium on Formal Aspects of Component Software (FACS) Colocated with the 11th International Conference on Integrated Formal Methods (IFM)*, vol. 127, pp. 103–130, Bertinoro, Italy, September, 2014.

[17] A. Nouri, S. Bensalem, M. Bozga, B. Delahaye, C. Jegourel, and A. Legay, "Statistical model checking QoS properties of systems with SBIP," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 2, pp. 171–185, 2015.

[18] P. Zuliani, "Statistical model checking for biological applications," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 527–536, 2014.

[19] A. Dal Corso, D. Macedonio, and M. Merro, "Statistical model checking of ad hoc routing protocols in lossy grid networks," in *NASA Formal Methods*, K. Havelund, G. Holzmann, and R. Joshi, Eds., vol. 9058 of *Lecture Notes in Computer Science, 7th NASA Formal Methods Symposium, Pasadena, CA, USA*, pp. 112–126, NASA, 2015.

[20] R. Grosu, D. Peled, C. R. Ramakrishnan, S. Smolka, S. Stoller, and J. Yang, "Using statistical model checking for measuring systems," in *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, T. Margaria and B. Steffen, Eds., vol. 8803 of *Lecture Notes in Computer Science, 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA), Imperial, Greece, October 2014*, pp. 223–238, 2014.

[21] A. Legay and M. Viswanathan, "Statistical model checking: challenges and perspectives," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 369–376, 2015.