Journal of Circuits, Systems, and Computers
Vol. 31, No. 1 (2022) 2250012 (15 pages)

DOI: 10.1142/S0218126622500128

World Scientific
www.worldscientific.com

Mitigating Lifetime-Energy-Makespan Issues in Reliability-Aware Workflow Scheduling for Big Data*

Yu-Jie Xiong[†,‡,¶], Song-Yang Cheng[†] and Bin Chen[§]

[†]*School of Electronic and Electrical Engineering,*
Shanghai University of Engineering Science,
Shanghai 201620, P. R. China

[‡]*Shanghai Key Laboratory of*
Multidimensional Information Processing,
East China Normal University,
Shanghai 200241, P. R. China

[§]*School of Mechanical and Automotive Engineering,*
Shanghai University of Engineering Science,
Shanghai 201620, P. R. China
[¶]*xiong@sues.edu.cn*

Received 12 December 2020
Accepted 10 June 2021
Published 26 July 2021

The emergence of cloud computing in big data era has exerted a substantial impact on our daily lives. The conventional reliability-aware workflow scheduling (RWS) is capable of improving or maintaining system reliability by fault tolerance techniques such as replication and checkpointing based recovery. However, the fault tolerant techniques used in RWS would inevitably result in higher system energy consumption, longer execution time, and worse thermal profiles that would in turn lead to a decreased hardware lifespan. To mitigate the lifetime-energy-makespan issues of RWS in cloud computing systems for big data, we propose a novel methodology that decomposes the complicated studied problem. In this methodology, we provide three procedures to solve the energy consumption, execution makespan, and hardware lifespan issues in cloud systems executing real-time workflow applications. We implement numerous simulation experiments to validate the proposed methodology for RWS. Simulation results clearly show that the proposed RWS strategies outperform comparative approaches in reducing energy consumption, shortening execution makespan, and prolonging system lifespan while maintaining high reliability. The improvements on energy saving, reduction on makespan, and increase in lifespan can be up to 23.8%, 18.6%, and 69.2%, respectively. Results also show the potentiality of the proposed method to develop a distributed analysis system for big data that serves satellite signal processing, earthquake early warning, and so on.

Keywords: Cloud computing; reliability-aware workflow scheduling; energy; makespan; lifespan; replication; Big Data.

*This paper was recommended by Regional Editor Tongquan Wei.
[¶]Corresponding author.

1. Introduction

Cloud computing is a successful commercial computing paradigm in big data era that delivers users service over communication networks and virtualizes infrastructure resources of computing, storage, and communication into ordinary commodities utilized in a pay-as-you-go manner.[1] Figure 1 shows a three-tier cloud business market structure that contains three entities: the cloud infrastructure provider, the cloud service provider, and cloud customers. To achieve increasing profit in the cloud computing market, ensuring high-quality cloud services (i.e., completing service requests within the deadline, maintaining high reliability of service request processing, and increasing hardware lifespan) and reducing servers' energy consumption are becoming major concerns of cloud service providers. Among all the concerns, system reliability is of the highest priority since high reliability is a necessity to guarantee the successful operation of service requests.

Workflow is widely used in describing service requests in cloud computing for big data. A workflow is composed of two parts: a set of tasks and the data or control dependencies among these tasks. Due to this structure, directed acyclic graphs (DAGs) can be utilized to represent workflow applications. In a DAG, nodes are used to denote tasks and edges are used to denote the data or control dependencies. Reliability-aware workflow scheduling (RWS) is an effective approach to ensure the high system reliability and therefore has attracted lots of attention in cloud computing community. RWS is able to maintain or increase cloud system's reliability by using techniques such as replication,[2] frequency speedup,[3] rollback recovery,[4] etc. However, these techniques would inevitably bring in many problems such as high

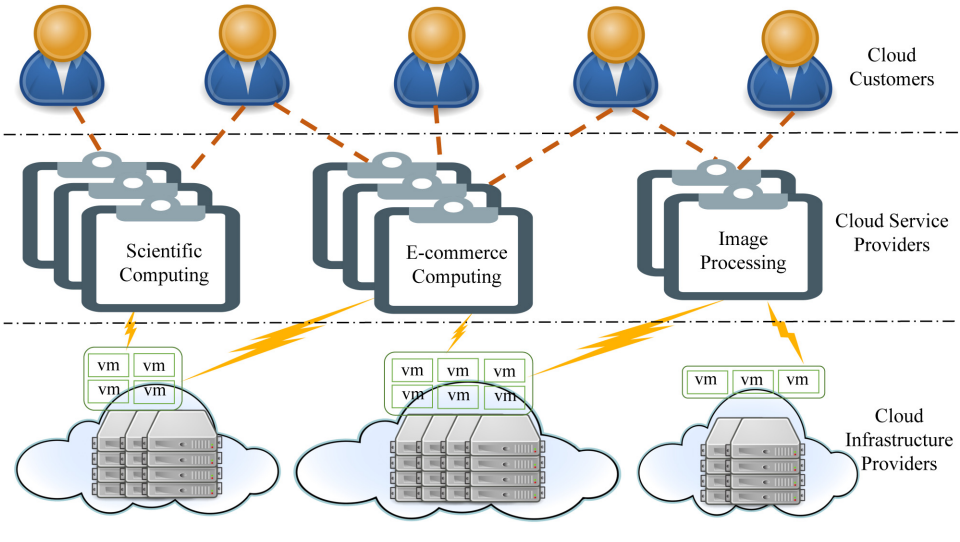


Fig. 1. The three-tier cloud business market structure.

temperature (and thus shortened lifetime),[5] large energy consumption,[6] and long makespan,[7] which are often ignored in conventional RWS. To overcome these challenges, we aim to handle the lifetime, energy, makespan problems and propose an effective framework for RWS in cloud systems. The framework contains three parts that are used to increase system lifetime, reduce system energy consumption, and decrease makespan under the requirements of reliability and deadline. Specifically, in this paper, we make the following major contributions:

- The problem of optimizing lifetime, energy, and makespan at the same time under the reliability and deadline constraints is decomposed into three sub-problems. The three sub-problems are very relevant since the concerns (i.e., lifetime, energy, and makespan) for RWS in the three sub-problems are same. In addition, these concerns influence each other and are determined by the same group of operations.
- Three RWS mechanisms are designed to solve the decomposed sub-problems for the reliability-aware cloud computing systems.
- A series of simulation experiments are carried out to validate the proposed scheme. Simulation results reveal that the proposed scheme outperforms the existing peer approaches in many aspects.

The remainder of the paper is organized as follows. Section 2 reviews the related work on RWS in cloud computing. Section 3 presents the preliminaries, including the bag-of-task model, reliability model, energy model, temperature model, and lifetime model. The details of the proposed methodology for RWS are given in Sec. 5. Section 6 validates the proposed methodology using simulation experiments. Section 7 concludes the paper and discusses future work.

2. Related Work

Considerable research efforts have been devoted to improving reliability against soft errors. For example, a resource management method is proposed for jointly optimizing system soft-error reliability (SER) and lifetime reliability (LTR).[8] To solve the resource management problem, the authors develop a novel static evolutionary algorithm that maximizes SER and LTR for real-time homogeneous MPSoC systems under the constraints of task deadline, energy budget, and task precedence. Similarly, a new time-constrained reliability-aware HEFT algorithm is presented in Ref. 9, which uses the list scheduling algorithm to derive the plan solution, and finds its solution based on the concept of fuzzy dominance, and finally finds the best SER-LTR trade-off solutions. However, all the above works are designed for embedded systems, rather than cloud systems.

There exist a large amount of research works on RWS in cloud systems. Zhou *et al.*[10] propose a dependable algorithm for scheduling workflow applications on cloud-assisted cyber-physical systems. It uses slack to recover failed tasks, and

realizes static scheduling of tasks and dynamic allocation of recovery by first determining the priority of the task, and then assigning the maximum frequency to each task. Wu *et al.*[11] design a soft-error aware energy-saving task scheduling method for the workflow of a DVFS-enabled cloud center. The proposed method generates energy-saving task plans for the workflow by assigning tasks to appropriate virtual machines with specific operating frequencies, and meet the reliability and completion time constraints required by tenants. In Ref. 12, the authors present an incentive public auditing scheme for non-manager groups in clouds. Based on the blockchain technology, they use a threshold signature technology to ensure data integrity, and blind signature technology to achieve high reliability, security and privacy protection in the public auditing scheme. Although all these methods are effective in increasing system reliability, they do not consider energy, makespan, lifetime issues in RWS.

Recently, researchers have investigated energy, makespan, and lifetime issues in RWS separately. Zhang *et al.*[13] employ a novel and effective evolutionary algorithm to maximize the energy efficiency and the resource utilization of the reserved cloud data centers by exploring energy efficient VM allocation solutions. Moreover, to accelerate the exploration of VM allocation solutions, they design an efficient simulation engine for cloud simulator CloudSim. The workflow scheduling problem with resolving the task execution order, task-to-VM allocation, and VM type assignment is solved in Ref. 14. The authors provide two genetic algorithm (GA) based approaches to tackle a single-objective optimization problem that minimizes execution cost under deadline constraint and a multi-objective optimization problem that attempts to minimize execution cost and makespan simultaneously. To solve the multi-objective workflow scheduling problem, a new list scheduling algorithm called FDHEFT is presented in Ref. 15, which combines fuzzy dominance ranking with HEFT and uses fuzzy dominance to measure the relative suitability of solutions in a multi-objective domain (i.e., makespan and cost).

3. Preliminary

This section introduces the preliminaries used in the paper, including the bag-of-task model, reliability model, energy model, temperature model, and lifetime model.

3.1. *Bag-of-task model*

Bag-of-tasks (BoTs) are a type of applications that consist of numerous independent tasks which could be processed in parallel without synchronization.[16] The independence means that there is no data or execution precedence between any two tasks in the application. BoTs have been widely adopted in many domains like image processing and big data processing.[16] Due to its high computing capability and flexible pricing strategy, cloud computing is a natural solution to execute the BoT applications. Consider a BoT application Γ that is composed of N independent real-time

tasks $\{\tau_1, \tau_2, \ldots, \tau_N\}$. The application is assumed to be executed on a cloud hardware platform that provides M cores $\{C_1, C_2, \ldots, C_M\}$. Each core C_m $(1 \leq m \leq M)$ is dynamic voltage scaling (DVS)-enabled and supports a group of frequencies. A task in the BoT is represented as a triplet $\tau_i : \{\mu_i, c_i, D\}$ $(1 \leq i \leq N)$, where μ_i, c_i, and D are the task's active factor, worst case execution cycles, and deadline, respectively. Assume $t_{\text{start}}(\tau_i)$, $t_{\text{finish}}(\tau_i)$, and $t_{\text{exe}}(\tau_i)$ are task τ_i's start time, finish time, and execution time. We can easily derive that $t_{\text{finish}}(\tau_i) = t_{\text{start}}(\tau_i) + t_{\text{exe}}(\tau_i)$. In clouds, if faults happened, the task needs to be recovered and the corresponding time is the worst case execution time. Otherwise, the execution time is the best case execution time. The makespan is actually the latest completion time of all tasks in the BoT application. It can be expressed as

$$t_{\text{makespan}} = \max_{i=1}^{N} t_{\text{finish}}(\tau_i)\,. \tag{1}$$

3.2. *Reliability model*

Soft errors are generally modeled by the exponential distribution that is characterized by an average arrival rate θ. The error rate θ is the expected number of failures occurring per second and a core's error rate highly relies on core's frequency f.[6] The error rate at frequency f is expressed as

$$\theta(f) = \theta_{f_{\max}} \cdot 10^{\frac{d(1-f)}{1-f_{\min}}}, \tag{2}$$

where $\theta_{f_{\max}}$ and $f_{\min}$ are the average error rates at the maximal frequency $f_{\max}$ and the minimal frequency $f_{\min}$. d is a hardware related parameter constant representing the sensitivity of error rates to frequency scaling.

A task's reliability is calculated as the probability that the task has been successfully executed without suffering soft errors. Concretely, for a task τ_i with the error rate $\theta(f_i)$, its reliability[17] executing at the frequency f_i is

$$R_i = e^{-\theta(f_i)\frac{c_i}{f_i}}\,. \tag{3}$$

The correct operation of a BoT application running on a cloud system depends on the successful execution of all tasks in the BoT application. In this sense, the system reliability, represented by R_{sys}, is modeled as the product of reliabilities of N BoT tasks. Thus, we can derive that

$$R_{\text{sys}} = \prod_{i=1}^{N} R_i\,. \tag{4}$$

Replication, which explores time and space redundancy to increase the probability of successful task execution, has been widely used in increasing reliability. Details on the replication technique are suggested to refer Ref. 2.

3.3. *Energy model*

Let E_{sys} represent the energy consumption consumed by all the tasks in the BoT application Γ. It is formulated as

$$E_{\text{sys}} = \sum_{i=1}^{N} E_i \,, \tag{5}$$

where E_i is the energy consumed by task τ_i and is computed as the product of power consumption and execution time of task τ_i. The core's power consumption highly depends on the core's frequency and the task execution time is decided by the task's operating frequency. Specifically, a core's power consumption is computed as the sum of static power and dynamic power. Using the CMOS model, the power consumption ϕ can be derived as

$$\phi = \mu(\lambda f^3 + \xi f + \kappa f T) \,, \tag{6}$$

where μ is an active factor manifesting the heterogeneous nature of tasks, f is the frequency, λ is the effective switching capacitance, and T is chip temperature. ξ and κ are two curve fitting constants depending on the core.

3.4. *Temperature and lifetime model*

Let $T_{\text{peak}}(\tau_i)$ be the peak temperature of task τ_i, which is expressed as $T_{\text{peak}}(\tau_i) = \max : \{T(t) | \forall\, t \in [t_{\text{start}}(\tau_i), t_{\text{finish}}(\tau_i)]\}$. $T(t)$ is the instantaneous temperature and can be obtained by HotSpot.[18] Let T_{peak} be the peak temperature of tasks on processors, then it is calculated as

$$T_{\text{peak}} = \max_{i=1}^{N} T_{\text{peak}}(\tau_i) \,. \tag{7}$$

The instantaneous temperature at time t is obtained by

$$T(t) = T_{\text{sst}} - (T_{\text{sst}} - T_{\text{int}}) \times \exp\{-t/(\alpha \times \beta)\} \,, \tag{8}$$

where T_{sst} is the steady state temperature and T_{int} is the initial temperature. α and β is the thermal resistance and capacitance, respectively. Both α and β are two core architecture-dependent constants. T_{sst} is calculated by

$$T_{\text{sst}} = T_{\text{amb}} + \phi \times \alpha \,, \tag{9}$$

where T_{amb} is the chip die's ambient temperature and α is the thermal resistance. ϕ is the power consumption that can be derived by Eq. (6).

High-temperature and frequent-temperature variations would accelerate chip wear-out due to electro-migration (EM), time-dependent dielectric breakdown (TDDB), stress migration (SM), and thermal cycling (TC).[19] The accelerated wear-outs eventually would lead to permanent faults occurring earlier and shorten hardware lifespan. Hardware lifespan is typically quantified by the mean time to failure

(MTTF), which can be derived by a well established tool.[19] The MTTFs due to the above-mentioned four failure mechanisms are all highly dependent on temperature, can be increased by reducing core's temperature.

4. Methodology Overview

A versatile RWS methodology should not only have a high reliability but also be able to solve the makespan, energy, lifespan issues and satisfy deadline requirements. However, this cannot be true since that (i) minimizing energy consumption and execution makespan of workflow applications are antagonistic and therefore cannot be optimized simultaneously. Concretely, if workflow applications' operating frequencies are reduced for decreasing system energy consumption, makespan is inevitably increased because of the delayed task execution, and vice versa. On the other hand, if the tasks are executed at high frequencies for decreasing makespan, the increase of system energy consumption would be also inevitable. (ii) If a system's peak temperature exceeds a safe threshold, the system will possibly fall into the predicament of hardware failures. To avoid this, a system designer should focus on prolonging system lifetime by controlling the system's peak temperature. Based on the two observations, we cannot optimize energy and makespan simultaneously and we prolong system lifetime if the temperature constraint is not considered. Therefore, a new approach is proposed in this paper that divides the co-optimization problem of makespan, energy, and lifespan with high reliability for RWS into three concerns. For each concern, we provide an RWS mechanism in this paper.

Figure 2 shows the overview of the proposed methodology. It uses an energy-efficient RWS (`EERWS`) approach to minimize system energy consumption under the reliability, deadline and temperature constraints, a makespan-aware RWS (`MARWS`) to minimize makespan under the reliability, deadline and temperature constraints, and a lifetime-aware RWS (`LARWS`) to improve reliability and lifespan under the deadline constraint. The proposed three RWS approaches are very useful to battery-

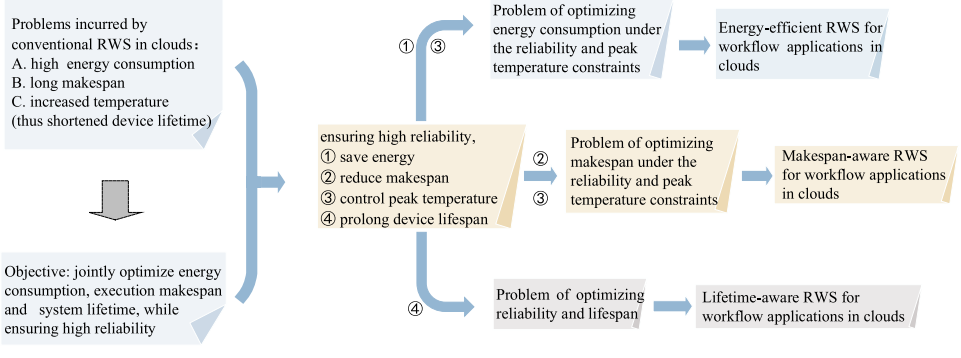


Fig. 2. The high-level overview of the proposed methodology.

powered server systems and service-oriented edge devices. The details of the proposed methodology are presented in the following section.

5. Details on the Proposed RWS Methodology

As discussed above, we propose a new and effective methodology to mitigate lifetime-energy-makespan issues in RWS for cloud systems. The details on the proposed RWS methodology are described in Algorithms 1. It first divides the studied big problem into three subproblems and then calls the procedures `EERWS`, `MARWS`, and `LARWS` to solve the three decomposed problems (lines 1–2).

The goal of our proposed procedure `EERWS` is to generate an energy optimum schedule of workflow applications meeting reliability and temperature constraints. The procedure takes as input workflow application Γ, cloud hardware system C, the system reliability goal R_{goal}, the temperature constraint $T_{\max}$, and an arbitrarily small positive number ϵ (line 3). Unlike the traditional RWS methods for corner cases, the uncertainty in soft error occurrences is considered in our algorithm, which uses an error adaptation variable ω to model the uncertainty (line 4). ω adapts its value from 0 to 1, indicating the error-free case to the error occurrence case. In the error occurrence case, the task execution time should contain the error recovery time. The procedure then reduces system energy consumption by deriving an energy-efficient workflow schedule for tasks in application Γ, which selects the core to execute tasks and determines the operating frequency. After deciding the allocation and frequency of every task, the procedure then uses the thermal-aware task sequencing technique to decide the execution order of tasks on every core for lowering the peak temperature T_{peak} (line 8). The task sequencing technique fully exploits tasks' thermal characteristics to lower the peak temperature by iteratively alternating task execution.

The goal of our proposed procedure `MARWS` is to generate a makespan optimized workflow schedule meeting temperature and reliability constraints. The procedure takes as input workflow application Γ, cloud hardware system C, and the temperature constraint $T_{\max}$. It is easy to find that the execution makespan of workflow application Γ is optimal if all cores' schedule lengths are equal. Motivated by this, the procedure derives the optimal workload of cores which leads to equal or nearly equal workflow schedule length (line 12). It is realized by dividing the N tasks into M groups and assigning the M groups to M cores, respectively (line 13). The procedure first sets the operating frequency of each task in workflow application to the maximal frequency of its assigned core (line 14). To ensure the system reliability, task replication is also adopted in this method (line 15). Under this setup, the procedure will exit if the timing constraint violates (lines 16–18). The procedure also checks the temperature constraint by comparing T_{peak} with the limit $T_{\max}$ (line 19). If T_{peak} is higher than $T_{\max}$, the procedure uses the task sequencing technique to reduce tasks' temperatures (line 20). But due to the high frequencies, T_{peak} may be still higher than

Algorithm 1. The proposed RWS methodology

Input: $\Gamma \&\& C \&\& R_{\text{goal}} \&\& T_{\max} \&\& \epsilon$;

divide the studied problem into three subproblems;
call the procedures `EERWS`, `MARWS`, and `LARWS` to solve the decomposed problems;
Procedure `EERWS`$(\Gamma, C, R_{\text{goal}}, T_{\max}, \epsilon)$;
randomly set a value in the rang of $[0, 1]$ to the error adaptation variable ω;
repeat
    update the execution time of application Γ based on the selected ω and then update the value of ω;
    obtain an energy-efficient workflow schedule for tasks in application Γ to reduce system total energy consumption E_{tot} under the deadline constraint D;
    decide the execution sequence of tasks on cores to lower T_{peak} under the temperature constraint $T_{\max}$;
    calculate the system reliability R_{sys} of executing the workflow application Γ;
until $(R_{\text{sys}} - R_{\text{goal}}) \geq \epsilon > 0$;
Procedure `MARWS`$(\Gamma, C, T_{\max})$;
derive the optimum workload for all cores;
classify the N tasks in workflow application Γ into M sets according to the optimum workloads in an FF manner and allocate the M sets to all cores;
adopt the maximum frequency of the assigned cores as the working frequency of tasks;
determine the number of replication of tasks for satisfying system reliability constraint;
if $t_{\text{makespan}} > D$ **then**
    exit;
end
if $T_{\text{peak}} > T_{\max}$ **then**
    reduce T_{peak} by the temperature-aware sequencing;
    if $T_{\text{peak}} > T_{\max}$ **then**
        lower T_{peak} under the constraints of deadline and reliability using dynamic frequency scaling;
    end
end
Procedure `LARWS`(Γ, C);
produce an initial task allocation and frequency assignment strategy, represented by $(\mathcal{A}, \mathcal{F})$;
derive the lifespan of individual cores and the maximal/minimal lifespan $L_{\max}/L_{\min}$;
Pre $= 0$, Cur $= L_{\min}$;
while $L_{\min} < L_{\max}$ and Pre $\neq$ Cur **do**
    Pre $= L_{\min}$;
    if $L_{\min} = \text{MTTF}_P(C_{\min})$ **then**
        LMF_Reallocation$(C_{\max}, C_{\min})$;
    end
    if $L_{\min} \neq \text{MTTF}_P(C_{\min})$ **then**
        LMF_Replication$(C_{\max}, C_{\min})$;
    end
    update $\mathcal{A}$, $\mathcal{F}$, $L_{\max}$, $L_{\min}$, $C_{\max}$, $C_{\min}$;
    Cur $= L_{\min}$;
end

$T_{\max}$ (line 21). For this case, dynamic frequency scaling is utilized to decrease T_{peak} (line 22).

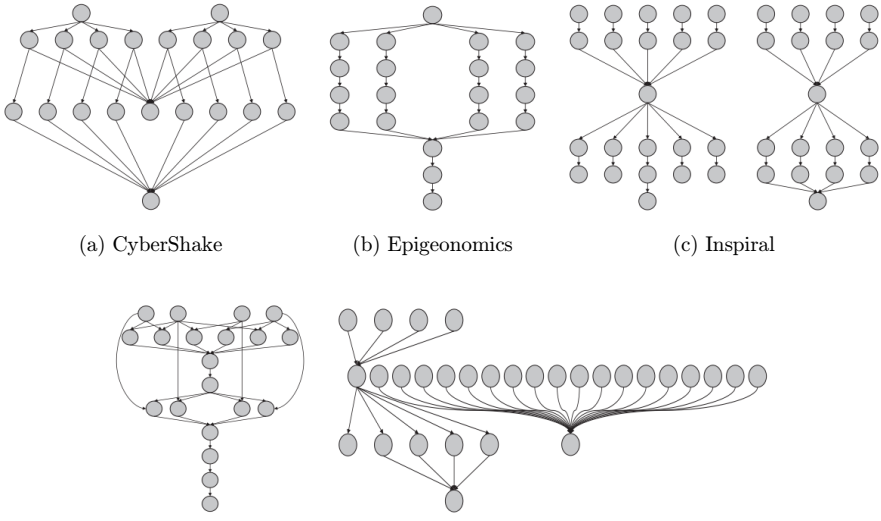
The goal of our proposed procedure `LARWS` is to maximize system lifespan and reliability of cloud systems. In this procedure, we use MTTF as the metric for quantifying both system lifespan and soft error reliability. With the common metric, it is easy to guide the tradeoff between lifespan and soft error reliability. In this procedure, we use $\text{MTTF}_P(C_m)$ to denote the MTTF of core C_m related to permanent faults and $\text{MTTF}_S(C_m)$ be the MTTF of core C_m related to soft errors. The two MTTFs can be derived by the existing tools and models. Thus we omit the details here. After obtaining the two MTTFs, the lifespan of core C_m denoted by $\text{MTTF}(C_m)$ is formulated as $\min\{\text{MTTF}_P(C_m), \chi\text{MTTF}_T(C_m)\}$, where χ is ratio representing the dominance of two MTTFs in lifespan. Using this, all cores' lifespan can be derived and the maximal/minimal lifespan are $L_{\max}/L_{\min}$. We use $C_{\max}$ and $C_{\min}$ to represent the core with the maximal and minimal lifespan, respectively. Clearly, the whole system lifespan is decided by the $L_{\min}$ of core $C_{\min}$. The procedure `LARWS` takes as input workflow application Γ and cloud multicore system C (line 25). It first produces an initial task allocation and frequency selection strategy $(\mathcal{A}, \mathcal{F})$ (line 26). Then, the lifespan of all cores can be calculated and hence the maximal/minimal lifespan $L_{\max}/L_{\min}$ are derived (line 27). As discussed above, $C_{\max}$ is the most suitable to be utilized for improving system lifespan. Therefore, the procedure iteratively reallocates task to cores or adjust the replication of tasks on $C_{\min}$ and $C_{\max}$ (lines 29–39). In each iteration, the procedure needs to find which MTTF dominates the lifespan of $C_{\min}$. If it is dominated by $\text{MTTF}_P(C_{\min})$, which means MTTF_P of the core needs to be increased, the task allocation of $C_{\min}$ and $C_{\max}$ is then adjusted by LMF_Reallocation (line 32). The reallocation operation chooses a suitable task and moves this task from $C_{\min}$ to $C_{\max}$. The migration of this task should lead to the largest ΔMTTF_P. If the lifespan is dominated by $\Delta\text{MTTF}_S(C_{\min})$, which means that MTTF_S of the core needs to be increased, the task allocation of $C_{\min}$ and $C_{\max}$ is then adjusted by LMF_Replication (line 35). The replication of the selected task should lead to the largest ΔMTTF_S. After the reallocation and replication, $\mathcal{A}$, $\mathcal{F}$, $L_{\max}$, $L_{\min}$, $C_{\max}$, $C_{\min}$ are all updated (lines 37 and 38). The iteration will repeat until the MTTF of $C_{\min}$ cannot be improved.

6. Evaluation

The objective of this work is to solve the lifetime-energy-makespan issues in RWS for cloud systems. Since the proposed methodology is composed of three procedures `EERWS`, `MARWS`, and `LARWS`, we validate the proposed methodology by verifying the effectiveness of the three procedures separately. Specifically, we carry out three simulation experiments to verify `EERWS`, `MARWS`, and `LARWS`, respectively. In the first simulation, the energy consumption of five benchmark applications using `EERWS` is compared with that of the state-of-art approaches `NOEM` and `HYEM`. `NOEM` is a baseline

that does not use any energy management techniques while `HYEM` is a hybrid method that uses RMFF for allocating tasks to cores and uses task sequencing and frequency scaling for determining execution order and operating frequency of tasks in the workflow application. In the second simulation, we compare the makespan of five benchmark applications using `MARWS` with that of the state-of-art approaches `NOMR` and `LSMR`. `NOMR` is a baseline that does not use any makespan reduction techniques while `LSMR` is a list scheduling-based makespan reduction approach that also considers system reliability. In the third simulation, the system lifespan of five benchmark applications using `LARWS` is compared with that of the state-of-the-art approaches `LALEF` and `LALUF`. The two approaches perform the reallocation and replication operation in a lowest-energy-first and largest-utilization-first manner, respectively.

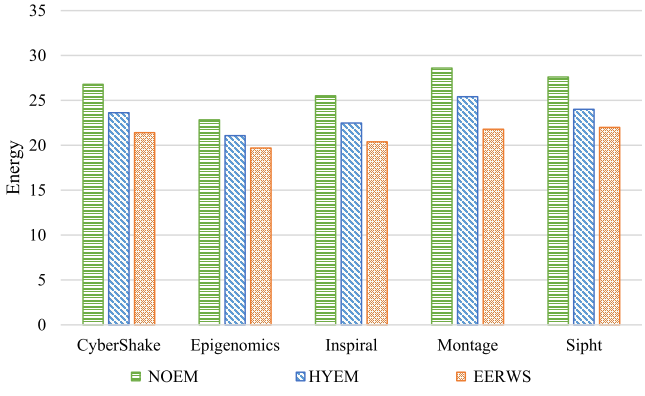
The five workflow benchmarks used in the simulation are CyberShake, Epigenomics, Inspiral, Montage, and Sipht.[14] These benchmarks have been widely adopted in testing the performance of workflow scheduling algorithms in clouds. Their data and structure are quite different. The structures of these five benchmarks are given in Fig. 3. The number of nodes and edges as well as data size can be found in Ref. 15. In the simulation, we derive the temperature profiles using a tool HotSpot[18] and the hardware lifetime is obtained by the tool.[19] Replication is used to increase system soft error reliability. The setups of five workflow benchmarks are different and they would be adjusted depending on the structure of these workflows. The experiments are implemented in Java and run on a cloud hardware platform that is composed of four homogeneous cores of 1.80 GHz.



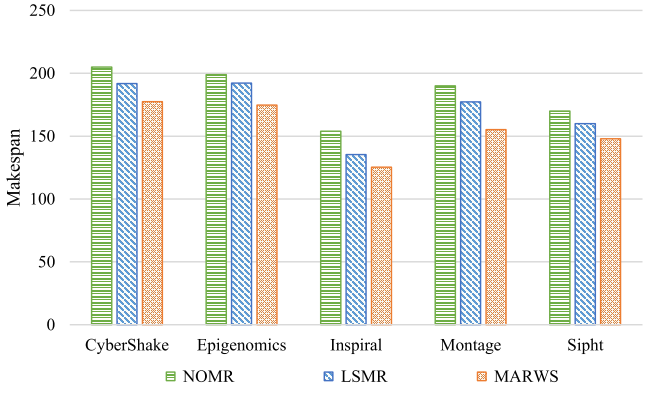
(a) CyberShake (b) Epigeonomics (c) Inspiral

(d) Montage (e) Sipht

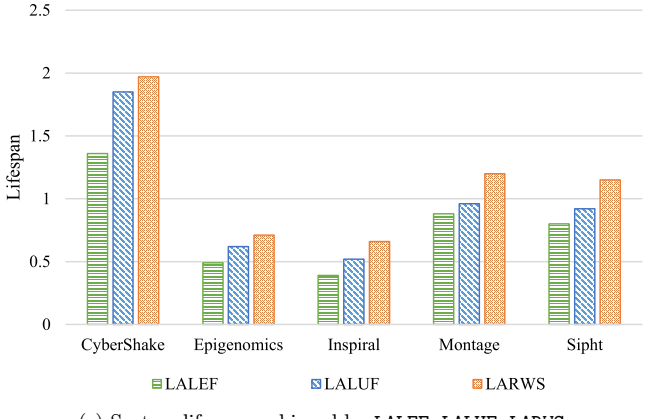
Fig. 3. The five workflow benchmarks used in the simulation experiments.



(a) System energy consumption of `NOEM`, `HYEM`, `EERWS`.



(b) Execution makespan realized by `NOMR`, `LSMR`, `MARWS`.



(c) System lifespan achieved by `LALEF`, `LALUF`, `LARWS`.

Fig. 4. Energy consumption, execution makespan, and lifespan of five benchmarks using different methods.

Figure 4 shows the energy consumption, makespan, and lifespan of benchmarks CyberShake, Epigenomics, Inspiral, Montage, and Sipht using the proposed strategies and comparative algorithms. Specifically, Fig. 4(a) presents the energy consumption of the five workflow benchmarks using `NOEM`, `HYEM`, and `EERWS`. Compared to the other two approaches, our proposed `EERWS` has the lowest energy consumption and it can reduce energy consumption by up to 23.8%. The execution makespan of the five workflow benchmarks using `NOMR`, `LSMR` and `MARWS` are presented in Fig. 4(b). As can be seen from the figure, our proposed `MARWS` has the shortest makespan when compared to the other two approaches. The reduction in execution makespan achieved by `MARWS` can be up to 18.6%. Figure 4(c) plots the system lifespan of five benchmarks executing on the cloud system using `LALEF`, `LALUF`, and `LARWS`. As can be observed from the figure, our proposed `LARWS` has the maximal system lifespan when compared with `LALEF` and `LALUF`. The enhancement of system lifespan realized by our `LARWS` can be up to 69.2%. We also test the CPU runtime overhead of the proposed scheme and the state-of-the-arts HYEM, LSMR, LALEF, LALUF. They are taken either less than one second or a few seconds, which are acceptable.

7. Summary

In this paper, we aim to mitigate lifetime-energy-makespan issues in reliability-aware workflow scheduling. To this end, we propose a new methodology for reliability-constrained workflow applications running on multicore based cloud systems. The proposed methodology is composed of three RWS strategies, `EERWS`, `MARWS`, and `LARWS`. The three RWS strategies can be used to solve the energy, execution makespan, and lifespan issues in reliability-aware workflow scheduling. We carry out extensive simulation experiments to validate the proposed three RWS strategies by comparing them with other state-of-art approaches with respect to energy, makespan, and lifespan. Simulation results reveal that, our proposed methodology is effective in reducing energy consumption, shortening execution makespan, and increasing system lifespan while ensuring high system reliability. In this work, we consider the homogeneous multicore system as the hardware platform in cloud and assume the cloud system is a public cloud. It shows the potentiality of the proposed method to develop a distributed analysis system for big data that serves satellite signal processing, earthquake early warning, and so on. However, this in fact is not always true in many real-world scenarios since heterogeneous multicore systems have become the mainstream and private cloud also becomes common. Therefore, in the future, we will consider heterogeneous multicore systems for hybrid clouds and extend our proposed methodology to solve the RWS problems in hybrid clouds.

Acknowledgments

This work is jointly sponsored by the National Key Research and Development Program of China (Grant No. 2019YFC1509202), National Natural Science

Foundation of China (Grant Nos. 62006150 and 61802251), Shanghai Young Science and Technology Talents Sailing Program (Grant No. 19YF1418400), Shanghai Key Laboratory of Multidimensional Information Processing (Grant No. 2020MIP001), and Fundamental Research Funds for the Central Universities.

References

1. T. Wang, J. Zhou, G. Zhang, T. Wei and S. Hu, Customer perceived value- and risk-aware multiserver configuration for profit maximization, *IEEE Trans. Parallel Distrib. Syst.* **31** (2020) 1074–1088.
2. M. A. Haque, H. Aydin and D. Zhu, On reliability management of energy-aware real-time systems through task replication, *IEEE Trans. Parallel Distrib. Syst.* **28** (2017) 813–825.
3. J. Zhou, X. S. Hu, Y. Ma and T. Wei, Balancing lifetime and soft-error reliability to improve system availability, *Proc. Int. Conf. Asia and South Pacific Design Automation*, 25–28 January 2016, Macao, China, pp. 685–690.
4. Y. Zhang and K. Chakrabarty, A unified approach for fault tolerance and dynamic power management in fixed-priority real-time embedded systems, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **25** (2006) 111–125.
5. Y. Ma, T. Chantem, R. P. Dick and X. S. Hu, Improving system-level lifetime reliability of multicore soft real-time systems, *IEEE Trans. Very Large Scale Integr. Syst.* **25** (2017) 1895–1905.
6. D. Zhu, R. Melhem and D. Mosse, The effects of energy management on reliability in real-time embedded systems, *Proc. Int. Conf. Computer-Aided Design*, 7–11 November 2004, San Jose, CA, United States, pp. 35–40.
7. I. Assayad, A. Girault and H. Kalla, A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints, *Proc. Int. Conf. Dependable Systems and Networks*, 28 June-1 July 2004, Florence, Italy, pp. 347–356.
8. J. Zhou, J. Sun, X. Zhou, T. Wei, M. Chen, S. Hu and X. S. Hu, Resource management for improving soft-error and lifetime reliability of real-time MPSoCs, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **38**(12) (2019) 2215–2228.
9. J. Zhou, M. Zhang, J. Sun, T. Wang, X. Zhou and S. Hu, DRHEFT: Deadline-constrained reliability-aware HEFT algorithm for real-time heterogeneous MPSoC systems, *IEEE Trans. Reliab.*, in press, (2020), doi: 10.1109/TR.2020.2981419.
10. J. Zhou, J. Sun, M. Zhang and Y. Ma, Dependable scheduling for real-time workflows on cyber-physical cloud systems, *IEEE Trans. Ind. Inform.*, in press, (2020), doi: 10.1109/TII.2020.3011506.
11. T. Wu, H. Gu, J. Zhou, T. Wei, X. Liu and M. Chen, Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud, *J. Syst. Architect.* **84** (2018) 12–27.
12. L. Huang, J. Zhou, G. Zhang, J. Sun, T. Wei, S. Yu and S. Hu, IPANM: incentive public auditing scheme for non-manager groups in clouds, *IEEE Trans. Dependable Secure Comput.*, in press, (2020), doi: 10.1109/TDSC.2020.3004827.
13. X. Zhang, T. Wu, M. Chen, T. Wei, J. Zhou, S. Hu and R. Buyya, Energy-aware virtual machine allocation for cloud with resource reservation, *J. Syst. Softw.* **147** (2019) 147–161.
14. J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei and M. Chen, Cost and makespan-aware workflow scheduling in hybrid clouds, *J. Syst. Architect.* **100** (2019) 101631.

15. X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei and S. Hu, 'Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT, *Future Gener. Comput. Syst.* **93** (2019) 278–289.
16. Y. Zhang, J. Zhou and J. Sun, Scheduling bag-of-tasks applications on hybrid clouds under due date constraints, *J. Syst. Architect.* **101** (2019) 101654.
17. S. Aminzadeh and A. Ejlali, A comparative study of system-level energy management methods for fault-tolerant hard real-time systems, *IEEE Trans. Comput.* **60** (2011) 1288–1299.
18. K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy and D. Tarjan, Temperature-aware microarchitecture: Modeling and implementation, *ACM Trans. Architect. Code Optim.* **1** (2004) 94–125.
19. Y. Xiang, T. Chantem, R. P. Dick, X. S. Hu and S. Li, System-level reliability modeling for MPSoCs, *Proc. Int. Conf. Hardware/Software Codesign and System Synthesis*, 24–29 October 2010, Scottsdale, AZ, United States, pp. 297–306.