Reliable and Energy Efficient Resource Provisioning and Allocation in Cloud Computing

Yogesh Sharma
Western Sydney University
NSW, Australia
y.sharma@westernsydney.edu.au

Bahman Javadi
Western Sydney University
NSW, Australia
b.javadi@westernsydney.edu.au

Weisheng Si
Western Sydney University
NSW, Australia
w.si@westernsydney.edu.au

Daniel Sun
DATA61-CSIRO
Australia
daniel.sun@data61.csiro.au

ABSTRACT

Reliability and Energy-efficiency is one of the biggest trade-off challenges confronting cloud service providers. This paper provides a mathematical model of both reliability and energy consumption in cloud computing systems and analyses their interplay. This paper also proposes a formal method to calculate the finishing time of tasks running in a failure prone cloud computing environment using checkpointing and without checkpointing. To achieve the objective of maximizing the reliability and minimizing the energy-consumption of cloud computing systems, three resource provisioning and virtual machine (VM) allocation policies using the aforementioned mathematical models are proposed. These three policies are named Reliability Aware Best Fit Decreasing (RABFD), Energy Aware Best Fit Decreasing (EABFD), Reliability-Energy Aware Best Fit Decreasing (REABFD). A simulation based evaluation of the proposed policies has been done by using real failure traces and workload models. The results of our experiments demonstrated that by considering both reliability and energy factors during resource provisioning and VM allocation, the reliability and energy consumption of the system can be improved by 23% and 61%, respectively.

KEYWORDS

Cloud Computing, Failures, Reliability, Energy Consumption, Virtual Machines, Resource Provisioning, Bag of Tasks, Checkpointing

1 INTRODUCTION

Cloud computing has revolutionized the Information Technology sector by giving computing a perspective of service to users. Though the acceptance of cloud computing technology is increasing every day, it is still facing numerous challenges because of its complex and large-scale architecture. Reliability and Energy-efficiency are two key challenges that need careful attention and investigation. In this study, reliability is considered as the probability with which an application/task will finish the execution before the occurrence of a failure. A failure in the services of a cloud costs significantly to both providers and customers. The report [12] from Ponemon institute in 2016 revealed that the average down-time cost of data centers due to outages is approximately $740,357 per year. According to the Information week, each year IT outages result in the revenue loss of more than $26.5 billion [19]. The energy requirement to operate the cloud infrastructure is also increasing in proportion to the operational costs. Approximately, 45% of the total operational expenses of IBM data centers goes in electricity bills. It has been estimated that the servers mounted in Microsoft's cloud based datacenters consumes around 2 terawatts-hours (TWh) of energy per year for which the company pays approximately $2.5 billion per year as electricity bills [1]. Apart from the operational costs, huge amount of energy consumption by cloud computing infrastructure causes huge amount of carbon and green house gases emission in the environment.

To maximize the reliability of the cloud computing services, all the cloud service vendors add back-up resources/hosts (hosts, nodes and resources are being used interchangeably in this work) and use replication as well as load balancing as fault tolerance mechanism. Adding extra resources increases energy consumption more steeply than the reliability. The key technique used to reduce the energy consumption is by running the resources on a low power scaling level or by turning off the under-utilized or idle resources such as back-up by migrating the running virtual machines (VM) from the under-utilized resources to other resources. Turning off the back-up resources will reduce the reliability of the system. For example, in the case of VM consolidation (key technique to reduce energy consumption in cloud computing systems), if a physical machine fails due to some hardware or software issues before the completion of tasks and there are no recovery/back-up resources, then all the VMs and their corresponding processes will have to start again. This will dramatically increase overheads such as energy consumption and resource utilization.

This creates a critical trade-off between the reliability and energy efficiency of the cloud computing systems. To make the cloud-computing systems reliable and energy-efficient, a mathematical framework is provided in this paper to show the interplay of these two factors. Following are the contributions of this work

UCC'17, 2017
© 2017 Association for Computing Machinery.
ACM ISBN ISBN 978-1-4503-5149-2/17/12…$15.00
https://doi.org/http://dx.doi.org/10.1145/

(1) A mathematical model to calculate the reliability and energy consumption while executing the tasks by the VMs on cloud computing resources in the presence of failures.
(2) A formal method to calculate the finishing time of the tasks in the presence of failures running on cloud resources with and without checkpointing.
(3) Resource provisioning and VM allocation policies using the proposed models to optimize the reliability and energy-efficiency of the cloud computing systems.

The remainder of the paper is organized as follows: Section 2 gives a brief survey about the existing work on reliability and energy-efficiency of cloud computing systems jointly. Section 3 explains the system architecture used in this work followed by the workload model and deadline model. Section 4 presents the reliability model and application/task finishing time with the given reliability using checkpointing and without checkpointing. Section 5 includes the formulation of energy consumption while executing the tasks in the presence of failures. In section 6, resource provisioning and VM allocation policies using the proposed models are presented. Section 7 consists of the details about the system set-up, workload model, performance evaluation metrics and reports the results in graphical form. Section 8 concludes this work.

2 RELATED WORK

Most of the work in the literature either explored reliability or energy consumption of cloud computing systems. Very limited work has been done by combining these two variables. A survey of the state of art in reliability and energy efficiency mechanisms has been provided in our previous work [23]. This section discusses the recent works covering both reliability and energy efficiency.

It has been claimed that as the operating frequency of a system increases wrt to supply voltage, reliability of the system increases but energy efficiency decreases [28]. This makes the task scheduling a challenge to achieve these two contradictive goals at the same time. In response to the challenge three different algorithms such as SHRHEFT, SHRCPOP and SHREERM have been proposed by Longxin Zhang et al. [27]. Dynamic voltage scaling and shared recovery technique have been used to regulate the energy consumption and to ensure the reliability, respectively. After performing the experiments, it has been concluded that SHREERM algorithm surpasses the rest. With the same objectives, a genetic algorithm (BOGA) for task-scheduling to optimize the reliability and energy-consumption of high performance computing systems has been proposed by the same authors [26]. The performance of the proposed algorithm has been compared with other two algorithms such as modified MODE and MOHEFT and the superiority of the new algorithm has been shown over the other algorithms. All the approaches proposed in [27] [26] are focused specifically on high performance computing systems. It has also been assumed that at most one failure will occur during the life time of a task. In our work, this assumption has been rejected with the injection of multiple failures.

Xiwei Qiu et al. [21] provides a theoretical correlation model for the fine grained measurement of reliability, power consumption and performance of cloud computing systems. A frequency scaling based power model while considering maximum CPU utilization

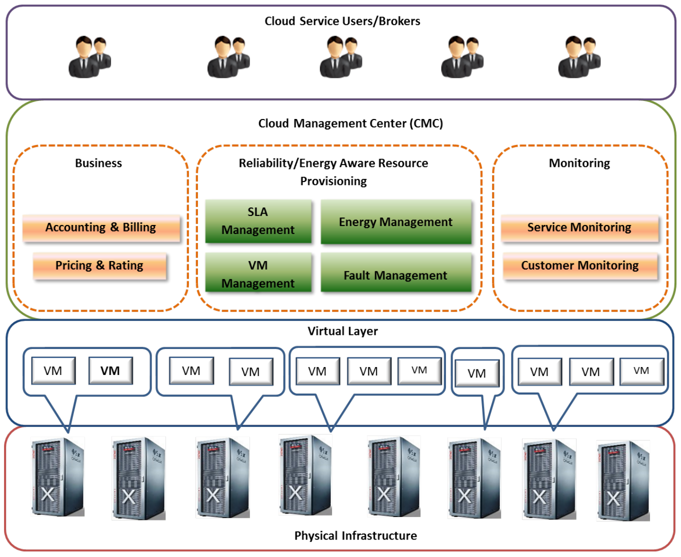

Figure 1: System Architecture

has been used where as we have formulated the power consumption based on variable utilization while operating CPU at maximum frequency. The proposed work has been analysed numerically except the reliability model which has been simulated. However, our evaluation of the proposed formulation has been done both analytically and by using simulation though only simulation based results have been reported.

Amir Varasteh et al. [24] have studied the interplay of energy consumption and reliability while performing the VM consolidation in cloud computing systems. A fine grained mathematical model has been provided to minimize the total data center cost while regulating the energy consumption and reliability. The proposed model has been analysed using matlab based simulation where as we have shown the interplay of both metrics using simulation based results using real life data. Authors have calculated the reliability of homogeneous systems as the function of system activity (on-off cycles) where as we have calculated the reliability under the occurrence of failures in a heterogeneous environment. Authors have used the random VM allocation to the physical machines to test their models. However, we have proposed three heuristics to allocate the VMs besides the random allocation.

3 SYSTEM ARCHITECTURE

The cloud computing environment considered in this work consists of a pool P of failure prone heterogeneous resources/nodes. From the resource pool, resources gets provisioned to run the heterogeneous VMs executing the tasks arriving at a specific rate. In Figure 1, the model has been divided into four different layers. The bottom layer consisting of the resources such as servers on top of which the VMs are mounted. The Virtual layer is responsible for the allocation of VMs according to the decisions made at the Cloud Management Center (CMC). The CMC is the heart of the architecture where all the reliability and energy aware resource provisioning and allocation policies are existing. The role of CMC is to gather the parameters from the energy managment and fault management modules and takes the decision about the VM allocation so that the reliability of the system will be maximized and energy consumption will be minimized. Users/Brokers are submitting their tasks to the CMC seeking execution before the deadline.

On the arrival of new tasks, the current status of the resources (gathered at monitoring module in CMC) and resource requirements of new tasks get evaluated at CMC. On the basis of the evaluation, optimized decisions about the resource provisioning and VM allocation takes place according to the proposed algorithms to regulate the reliability and energy-efficiency of the system. The number of VMs running on a provisioned node will not exceed the number of cores available on the node. One core will be allocated to each VM and VMs are not allowed to share the cores with each other (such configuration can be obtained in Xen [2] hypervisor). Memory of the host is being shared by the running VMs and to avoid the interference during run time, each VM has an exclusive share of the host memory.

3.1 Application Model

Bag-of-Task(BoT) applications, composed of bags or groups of independent and sequential compute intensive tasks arriving at a specific arrival rate (Table 2) have been used in this work. In BoT applications which can also be seen referred as Parameter Sweep Applications [8], there is no dependency and communication between the tasks. The most common examples of BoT applications are image manipulation applications (astronomy, image rendering, video survelliance), data mining applications, Monte Carlo simulations and intensive search applications. Each BoT consists of set of independent tasks, $T = \{t_i \mid 1 \leq i \leq n\}$. Every task t_i has a corresponding length, l_i. In this work, the length of a task has been refered to the number of instructions. For each task t_i a VM $vm_i \in VM$, where $VM = \{vm_i \mid 1 \leq i \leq n\}$ has been instantiated. To launch the desired number of VMs, a number of physical machines or nodes, $N = \left\{n_j \mid 1 \leq i \leq m\right\}$ gets provisioned according to the available CPU cores. Every task has a deadline d_i associated to it which has been calculated according to the model proposed in the following section.

3.2 Deadline Model

The tasks are non-preemtable and the execution of the application will be finished when the last task will complete the execution. Every task t_i has a corresponding deadline d_i, which has been calculated as

$$d_i = \begin{cases} s_i + (f.l_i), & \text{if } [s_i + (f.l_i)] < c_i \\ c_i, & \text{otherwise} \end{cases} \tag{1}$$

s_i, l_i and c_i are the starting time, task length and finishing time of a task t_i, respectively [15]. f is the stringency factor that defines the deadline strictness i.e. higher the value of f is, higher deadline relaxation the task has. All the proposed resource provisioning and allocation policies have been evaluated by using $f = 1.2$ i.e. normal deadline.

Rather than rejecting a task for a deadline miss (hard deadline), the soft deadline concept has been adopted which means it reduces

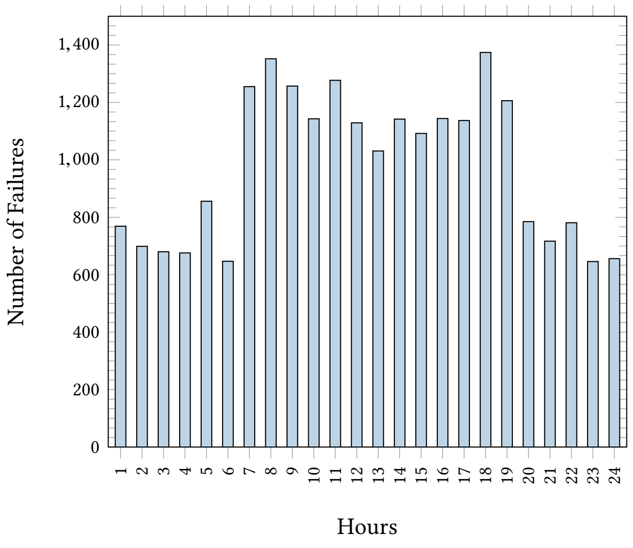

Figure 2: Failure count vs. daily hours

the value of the computation for the users [6]. More the execution of a task will be delayed, more the value will be reduced.

4 RELIABILITY MODEL

The proposed reliability model is based on the utilization of the system. Figure 2 shows the failure count per hour in LANL failure traces taken from Failure Trace Archieve [17], an online repository of failure traces gathered from various sites. It has been observed that during the working hours from 7 am to 7 pm, the number of failures are higher than the non-working hours. It can be interperated as a correlation between the occurrence of failures and system activity/utilization. During the working hours the utilization/activity of the systems is higher than the non-working hours which increases the occurrence of failures. The same conclusion has been drawn in [14] [22]. On the basis of the given conclusion, a linear failure rate (failure rate and hazard rate are being used interchangeably in this work) model directly proportional to the utilization has been proposed as follows

$$\lambda_{ij} = \lambda_{max_j} u_i^{\beta} \tag{2}$$

where, λ_{ij} is the failure rate of a VM vm_i running on a node n_j with utilization u_i and λ_{max_j} is the failure rate of node n_j at maximum utilization. In this work, it has been assumed that all the running VMs shares the maximum hazard rate similar to the physical node n_j which they are running on. β is a sensitivity factor which express the sensitivity of failure rate towards the utilizaiton. In this study, a strict linear relationship between the hazard rate and utilization of a system has been considered by taking β equal to 1. As failures in cloud computing systems are inevitable, every node n_j in the resource pool has a Mean Time between Failures ($MTBF_{max_j}$) at maximum utilization that has been calculated by the cloud coordinator at CMC empirically from the node history.

Hazard rate or Failure rate (λ_{max_j}) for a node n_j at maximum utilization will be calculated as

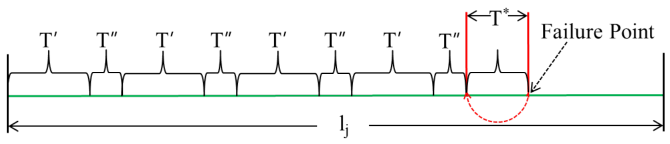

Figure 3: Recovery from Failure with Checkpointing

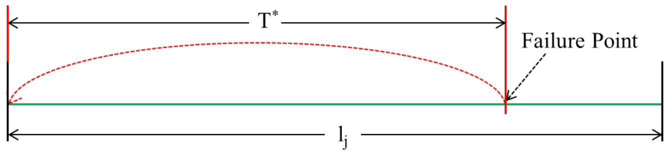

Figure 4: Recovery from Failure without Checkpointing

$$\lambda_{max_j} = \frac{1}{MTBF_{max_j}} \tag{3}$$

The failure rate has been assumed to be following poisson distribution [28][27] and remains constant for each utilization level. So the probability with which vm_i running on a node n_j with utilization u_i with hazard rate λ_{ij} will finish the execution of a task t_i of length l_i without the occurrence of a failure will be given as

$$R_{vm_{ij}} = \exp^{-\lambda_{ij} \times l_i} \tag{4}$$

To get the utilization of each VM, the length of corresponding tasks gets normalized wrt to the task of maximum length, l_{max}. In this work, VMs fail only when the node which they are running on fails. So the node and VMs have serial relationship such that when the node fails, all the associated VMs fail. The probability with which a node n_j will be able to finish the execution of all the m tasks being executed by m VMs before the occurrence of a failure has been calculated as

$$R_j = \prod_{i=1}^{m} (R_{vm_{ij}}) \tag{5}$$

However provisioned nodes fail independently [21]. Between the provisioned nodes, neither the serial relationship nor parallel relationship exists. So the reliability of the system has been calculated as the average of the reliabilitiy values possessed by all the provisioned nodes at a particular instance.

4.1 Fault Tolerance and Task Execution Model

In this work, two methods have been used to recover from a failure i.e. Checkpointing and Restart from the beginning. In the literature, checkpointing mechanism has been used intensively to provide fault-tolerance in cloud computing systems. Though the solutions provided by using checkpointing method are elegant, they are very costly in terms of overheads which make it some-times impractical. If a failure will not occur, checkpointing adds the overhead of 151 hours for a job of length 100 hours in a petaflop system [20]. However, if a failure occurs, re-execution of the failed task from the last checkpoint (Fig. 3) saves a good amount of re-execution period.

4.1.1 Finishing Time with Checkpointing. In the Figure 3, T' represents the checkpoint interval and T'' represents the checkpoint overhead such that time taken by the system to save the checkpoint of the running task on some stable storage. In this study, the value of T'' is equal to 20 seconds, which has been considered as the optimized minimum value in the previous studies [9]. We have assumed that the storage where all the checkpoints are getting stored is failure free. T^* represents the time required to re-execute the part of the task that was lost because of the occurrence of a failure. To further reduce the checkpointing overhead, risk based checkpointing mechanism [18] has been used in this work. In risk based checkpointing, if the expected amount of lost work before the checkpoint is smaller than the cost of checkpoint T'' then skip the checkpoint.

To calculate the length of the lost part of the failed task t_i of length l_i that need to be re-executed on a node n_j, first it is required to calculate the interval of checkpointing T'_j for that node, which has been calculated as follows by using Young's formula [25]

$$T'_j = \sqrt{2 \times T'' \times MTBF_j} \tag{6}$$

$T^{\#}$ is the part of a task that had been executed before the occurrence of a failure. The number of checkpoint intervals that took place before the occurrence of a failure on a node n_j while executing the task t_i will be calculated as

$$N'_{ij} = \left\lfloor \frac{T^{\#}_{ij}}{T'_j} \right\rfloor \tag{7}$$

The length of the lost part of failed task t_i that need to be re-executed will be calculated as

$$T^*_{ij} = \left(\frac{T^{\#}_{ij}}{T'_i} - N'_{ij} \right) \times T'_i \tag{8}$$

Besides the checkpointing overheads and re-execution part of the failed task, time to return (TTR) from a failed state to running state also adds to the finishing time of a task. So, the finishing time of a task t_i of length l_i executing on node n_j after the occurrence of n failures and m checkpoints has been calculated as follows

$$T^{\$}_{ij} = \begin{cases} l_i + \sum\limits_{k=0}^{n} T^*_{(ij)_k} + \left(T'' \times \sum\limits_{q=0}^{m} N'_{(ij)_q} \right) + \sum\limits_{k=0}^{n} TTR_{(ij)_k}, & \\ & \text{if } k, q > 0 \\ l_i & \text{otherwise} \end{cases} \tag{9}$$

4.1.2 Finishing Time without Checkpointing. Due to the expensive implementation of checkpointing mechanism, restarting of the execution of a failed task or job from the beginning (Fig. 4) is more preferable in practice because of the less overheads. We are claiming the adoption of task restart mechanism on the basis of the discussions and surveys done at the Fujitsu Primergy high-performance, distributed-memory cluster named Raijin located at National Computing Infrastructure (NCI) facility in Australia[2]. As there are no checkpoints in this case, the lost part of the task that

[2]https://nci.org.au/systems-services/national-facility/peak-system/raijin/

need to be re-executed T^* is equal to the part of the task length that has been executed before the occurrence of a failure $T^{\$}$. So the finishing time of a task t_i of length l_i executing on node n_j after the occurrence of n failures has been calculated as follows

$$T_{ij}^{\$} = \begin{cases} l_i + \sum_{k=0}^{n} T^*_{(ij)_k} + \sum_{k=0}^{n} TTR_{(ij)_k}, \text{if } k > 0 \\ l_i \quad \text{otherwise} \end{cases} \tag{10}$$

For both of the above mentioned task recovery mechanisms, the total finishing time of all the t tasks running on a provisioned node n_j will be calculated as

$$T_j^{\$} = \sum_{i=1}^{t} T_{ij}^{\$} \tag{11}$$

Total finishing time of all the tasks running on n provisioned nodes in the presence of failures will be calculated as

$$T_{total}^{\$} = \sum_{j=1}^{n} T_j^{\$} \tag{12}$$

The change in the execution length of a BoT application consisting of t tasks because of the occurrence of failures will be given as the difference between the total finishing time and actual length of the application such that the total finishing time without failures

$$\Delta T^{\$} = T_{total}^{\$} - \sum_{i=1}^{t} l_i \tag{13}$$

5 ENERGY MODEL

Many devices such as CPU, storage, memory, network interfaces and other PCI devices contribute to the power consumption of the system. But in the literature, it has been argued that CPU is the biggest power consumer despite of the advancement in the hardware and software technology [4] [16]. Based on the literature, this work has been focused on the energy minimization by regulating the utilization of CPU while operating at maximum frequency. As assumed, nodes in the resource pool has different minimum (P_{min}) and maximum power (P_{max}) cosumptions. The power consumption by a VM vm_i with utilization u_i running on a node n_j will be calculated as

$$P_j(u_i) = (frac_j \times P_{max_j}) + ((1 - frac_j) \times P_{max_j} \times u_i) \tag{14}$$

$frac_j$ is the fraction of minimum, min_j and maximum, max_j power consumption for a node n_j [3]. The energy consumption is the amount of power consumed per unit time. In the presence of failures, the energy consumption is the sum of energy consumed while executing the task length and energy wastage because of the failure overheads. So the energy consumption by a VM, vm_i running on node n_j while executing a task of length l_i in the presence of failures will be calculated as

$$E_{vm_{ij}} = \left(P_j(u_i) \times l_i\right) + E_{waste_{ij}} \tag{15}$$

As shown in equations 9-10, the finishing time of a task changes because of the occurrence of failures. Along with the re-execution time, there are other factors that adds to the execution time of a task such as down-time (TTR) i.e. time that system takes to restart the execution and other overheads. Only re-execution time of the task, checkpoint overheads and system down-time have been considered in this study to calculate the energy wastage. During the down-time, system is in non-working condition so it does not contribute to the energy wastage.

5.1 Energy wastage with checkpointing

For the checkpointing, the energy wastage will further split into the energy consumption while saving the checkpoints and energy consumption while re-executing the lost part of a task.

$$E_{waste_{ij}} = E_{checkpoint_{ij}} + E_{re-execute_{ij}} \tag{16}$$

The power consumption while saving the checkpoints on a disk consumes less power than power consumption during the execution of a task. In this study, $1.15 \times P_{min}$ has been taken as the power consumption while saving the checkpoints [10]. So the energy wastage while using the checkpointing has been calculated as

$$E_{checkpoint_{ij}} = \begin{cases} 1.15 \times P_{min_j} \times \left(T'' \times \sum_{q=0}^{m} N'_{(ij)_q}\right), \text{if } q > 0 \\ 0 \quad \text{otherwise} \end{cases} \tag{17}$$

The energy consumed while re-executing the lost part of the task t_i because of the occurrence of n failures has been calculated as

$$E_{re-execute_{ij}} = \begin{cases} P_j(u_i) \times \sum_{k=0}^{n} T^*_{(ij)_k}, \text{if } k > 0 \\ 0 \quad \text{otherwise} \end{cases} \tag{18}$$

5.2 Energy wastage without checkpointing

In the absence of checkpointing, the only energy that will be wasted is the energy consumption while performing the re-execution of the lost part of a task because of the occurrence of n failures

$$E_{waste_{ij}} = \begin{cases} P_j(u_i) \times \sum_{k=0}^{n} T^*_{(ij)_k}, \text{if } k > 0 \\ 0 \quad \text{otherwise} \end{cases} \tag{19}$$

So, the total energy consumption by n provisioned nodes allocated to m VMs while finishing all the tasks of BoT application in the presence of failures using checkpointing and without checkpointing will be calculated as

$$E_{total} = \sum_{i=1}^{n} \sum_{j=1}^{m} E_{vm_{ij}} \tag{20}$$

6 RESOURCE PROVISIONING AND VM ALLOCATION POLICIES

With the given BoT application consisting of a set of BoTs with n independent tasks in each and a pool of failure prone cloud resources, the challenge is how to provision the resources and allocate the VMs executing the tasks in order to maximize the reliability and minimize the energy consumption while keeping the turnaround time of every task less than corresponding deadline.

Table 1: Nomenclature used in algorithms and functions

Notation	Explanation
B	Set of Bag of Tasks
T	Set of Tasks in a Bag
t_i	i_{th} task
l_i	Length of i_{th} task
l_{max}	Length of a longest task in T
R	List of Resources/Nodes
R_{sorted}	Sorted list of Resources
V	Set of Virtual Machines (VMs)
V_{sorted}	Sorted list of VMs
r_j	j_{th} node
λ_j	Current Hazard Rate of j_{th} node
P_j	Current Power consumption of j_{th} node
$MTBF_j$	Current Mean Time Between Failure of j_{th} node
Ψ_j	Ratio of $MTBF_j$ and P_j
vm_i	i_{th} virtual machine
u_i	Utilization corresponding to i_{th} VM
RC_j	Remaining cores of j_{th} node
S_j	State of j_{th} node $i.e.$ failed or active
rel_{ij}	Reliability of i_{th} virtual machine on j_{th} node
pow_{ij}	Power consumption of i_{th} virtual machine on j_{th} node

The cloud co-ordinator periodically receives the BoTs at a fixed interval rate. As assumed, for each task in a BoT, one VM will be launched. So the number of VMs will be equal to the number of tasks that need to be executed. All the nodes at physical infrastructure layer (Fig. 1) have the hazard rate and power consumption at maximum utilization that has been recorded at CMC during previous executions. For simplicity, we have assumed that resource requirements of all the VMs will be fulfilled by the resources present in the cloud based data center. Depending on the objectives of the cloud provider three list based greedy heuristics such as Reliability Aware Best Fit Decreasing (RABFD), Energy Aware Best Fit Decreasing (EABFD) and Reliability-Energy Aware Best Fit Decreasing (REABFD) have been proposed to provision the resources and to allocate the VMs. As the base line policy, Opportunistic Load Balancing (OLB)[5] has been used.

Function 1 Reliability Aware Best Fit Decreasing (RABFD)

```
   function ReliabilityAware($R$)
   // Calculate the current Hazard Rate of a resource by using equation 2
1: for all $j \in R$ do
2:     $\lambda_j \leftarrow r_j$.calculateCurrentHazardRate()
3: end for
4: for all $j \in R$ do
5:     $R_{sorted} \leftarrow \lambda_j$.sortHazard-rateIncreasing()
6: end for
7: return $R_{sorted}$
   end function
```

Algorithm 1 Resource Provisioning and VM Allocation

```
1:  Input: Set of Bag of Tasks, B; List of Resources, R and Policy
2:  Output: Set of Provisioned Resources and Allocated VMs
3:  if ($Policy == RABFD$) then
4:      $R_{sorted} \leftarrow$ReliabilityAware($R$)
5:  else if ($Policy == EABFD$) then
6:      $R_{sorted} \leftarrow$EnergyAware($R$)
7:  else if ($Policy == REABFD$) then
8:      $R_{sorted} \leftarrow$ReliabilityandEnergyAware($R$)
9:  else
    //Default case is for OLB policy
10: end if
11: for all $b \in B$ do
12:     for all $i \in T$ do
13:         $vm_i = t_i$.taskAssignment()
14:         $V \leftarrow vm_i$
    // Calculating utilization of each VM
15:         $u_i = l_i/l_{max}$
16:         $U \leftarrow u_i$
17:     end for
    // Sorting VMs in decreasing order according to their utilization
18:     for all $i \in V$ do
19:         $V_{sorted} \leftarrow vm_i$.sortUtilizationDecreasing()
20:     end for
21:     for all $i \in V_{sorted}$ do
22:         $VM_{cores_i} \leftarrow vm_i.coresRequired()$
23:         for all $j \in R_{sorted}$ do
24:             if (($RC_j \geq VM_{cores_i}$) && ($S_j \neq failed$)) then
25:                 $r_j \leftarrow vm_i$.allocateHost()
26:                 $RC_j = RC_j - VM_{cores_i}$
    // Calculate VM reliability by using equation 4
27:                 $rel_{ij} \leftarrow vm_i$.calculateReliability()
    // Estimate VM power consumption by using equa- equation 14
28:                 $pow_{ij} \leftarrow vm_i$.estimatePower()
29:                 if ($RC_j == 0$) then
30:                     $R_{sorted} = R_{sorted} - R_{sorted_j}$
31:                 end if
32:                 break
33:             end if
34:         end for
35:     end for
36: end for
```

6.1 Reliability Aware Best Fit Decreasing (RABFD)

In this policy (Function 1), all the VMs executing tasks of each incoming BoT will be sorted in decreasing order according to their utilization and all the resources will be sorted in increasing order according to their hazard-rate corresponding to the current utilization (equation 2). After sorting, the resource provisioning will be done and VM corresponding to each task will be instantiated (Algorithm 1). After the instantiation, the allocation of VMs will be done in such a way that the VM with maximum utilization will get allocated to a resource with minimum hazard-rate.

6.2 Energy Aware Best Fit Decreasing (EABFD)

This policy has been proposed in Function 2 to optimize the energy consumption by the VMs in the presence of failures. In this policy all the resources will be sorted in the increasing order according to their power consumption corresponding to the current utilization (equation 14), so that the VM with maximum utilization will be allocated to a node with minimum power consumption.

Function 2 Energy Aware Best Fit Decreasing (EABFD)

```
function EnergyAware($R$)
// Calculate the current power consumption of a resource by using equation 14
1: for all $j \in R$ do
2:     $P_j \leftarrow r_j$.calculateCurrentPowerConsumption()
3: end for
4: for all $j \in R$ do
5:     $R_{sorted} \leftarrow P_j$.sortPowerIncreasing()
6: end for
7: return $R_{sorted}$
end function
```

6.3 Reliability-Energy Aware Best Fit Decreasing (REABFD)

The objective of this policy is to optimize the reliability and energy consumption both at the same time. In the given policy in Function 3, the ratio of $MTBF_j$ and power consumption, P_j corresponding to the current utilization for each node has been used to rank the resources. All the resources will be sorted in decreasing order according to the calculated ratio. A VM with maximum utilization gets allocated to a node with highest ratio value (Algorithm 1).

Function 3 Reliability and Energy Aware Best Fit Decreasing (REABFD)

```
function ReliabilityAndEnergyAware($R$)
1: for all $j \in R$ do
2:     $MTBF_j \leftarrow r_j$.calculateCurrentMTBF()
3:     $P_j \leftarrow r_i$.calculateCurrentPowerConsumption()
4:     $\Psi_j \leftarrow (MTBF_j) / (P_j)$
5: end for
6: for all $j \in R$ do
7:     $R_{sorted} \leftarrow \Psi_j$.sortMTBFPowerRatioIncreasing()
8: end for
9: return $R_{sorted}$
end function
```

6.4 Opportunistic Load Balancing(OLB)

This policy has been used as a baseline policy. In OLB, no criteria has been used to rank the resources and no preprocessing of VMs has been done based on their utilization as done in the previous policies. All the VMs executing tasks associated to incoming BoTs gets allocated in random order as they are arriving to the next available node (Algorithm 1).

7 PERFORMANCE EVALUATION

In order to evaluate the architecture proposed in Figure 1 and to evaluate the proposed resource provisioning and VM allocation policies, we have extended a popular cloud computing simulator i.e. CloudSim [7] by adding failure injectors and fault tolerance mechanisms.

7.1 Datacenter Configuration

The hardware configuration of more than 2000 hosts of the datacenter has been taken from Los Alamos National Laboratory (LANL) data set of Failure Trace Archive (FTA)[17]. FTA is an online public repository providing failure traces gathered from 26 different computing sites. This work has used LANL traces specifically because of the precise details provided regarding the failure start time and end time, causes of failures and node configuration. Traces from LANL systems were collected between year June 1996 to November 2005 and covers data from 23 high performance computing systems consisting of 4750 nodes in total. The mean time between failures (MTBF) and mean time to return (MTTR) for each node at maximum utilization have been calculated by using the failure information provided in the traces. To calculate an accordant value of MTBF and MTTR, only the nodes with event count more than 3 in the traces have been considered in this work.

To calculate the power consumption, the values of minimum and maximum power consumptions corresponding to a node are taken from spec2008 benckmark[3]. To select the realistic datacenter nodes, we have matched the core count and memory capacity of the nodes with the values provided in the traces. This approach has been used by Peter Garraghan et al. [11] by using Google trace logs. On the basis of the match, we have selected Intel Platform SE7520AF2 Server Board, HP ProLiant DL380 G5, HP ProLiant DL758 G5, HP ProLiant DL560 Gen9 and Dell PowerEdge R830 as 2, 4, 32, 128 and 256 core nodes with 4GB, 16GB, 32GB, 64GB and 256GB memory, respectively.

7.2 Workload Model

To generate the BoTs workload, model proposed by Iosup et al. [13] has been used with parameters given in table 2. In the analysis it has been established that the arrival of jobs behaves differently in peak and off-peak times. To provision the enough number of nodes for the fair evaluation of proposed policies, the inter-arrival time has been modeled using peak time workload following Weibull distribution with scale and shape parameters equal to 4.25 and 7.86, respectively. Every incoming BoT consists of 2^x tasks where x follows Weibull distrubution with scale and shape parameters given in Table 2. The length or execution time of each task in a BoT has been modeled as normal distribution with mean and standard deviation (SD) values equal to 2.73 and 6.1, respectively. Every task has a corresponding deadline that has been calculated using equation 1 with stringency factor f = 1.2 i.e. normal deadline.

[3]https://www.spec.org/power_ssj2008/results/

Table 2: Workload Generation Parameters

Input Parameters	Distribution	Values
Inter-arrival time (BoT)	Weibull	Scale = 4.25, Shape = 7.86
Number of Tasks per BoT	Weibull	Scale = 1.76, Shape = 2.11
Average runtime per task	Normal	Mean = 2.73, SD = 6.1

Table 3: Simulation Configuration Parameters

Input Parameters	Values
Stringency Factor (f)	1.2
Sensitivity Factor (β)	1
Checkpoint Overhead (T_s)	20 secs

7.3 Performance Evaluation Metrics

To evaluate the performance of the proposed resource provisioning and VM allocation policies, the results of following metrics have been reported

(1) **Reliability**: The reliability with which the application has been executed on the provisioned resources.
(2) **Energy Consumption**: Energy consumption incurred by the provisioned resources while executing the application.
(3) **Energy Wastage**: The amount of energy wasted while re-executing the lost part of the task because of failures and related overheads.
(4) **Turnaround Time**: It is the time taken by each task of BoT application to finish.
(5) **Deadline-Turnaround Time Fraction**: It is the margin by which the turnaround time has been exceeded from the deadline.
(6) **Benefit Function**: It is the ratio of first two parameters, reliability and energy consumption.

All the reported results are the average of 50 simulations with Confidence Interval (CI) of 95%. All the simulations have been performed by using 1000 BoTs with total number of tasks ranges between 100000 to 120000.

7.4 Results and Discussions

Figure 5 presents the average reliability for each policy using checkpointing and without checkpointing. It can be seen that REABFD gives better reliability by approximately 5% from RABFD, 16% from OLB and 17% from EABFD with checkpointing. Also in the scenario of without checkpointing, REABFD gives better reliability by 6% from RABFD, 15% from OLB and 23% from EABFD. Figure also shows that policies with checkpointing gives better reliability by 5% to 9% than without checkpointing. This is due to the fact that after an event of a failure, the task length generally gets reduced in process of recovery from the last checkpoint (if any). For shorter task length, the system possess high reliability to execute the task without or before the occurrence of a failure. However, in the scenario without checkpointing, the task restarts from the beginning after an event of a failure. As the size of a resubmitted task remains same, reliability of a system also remains unchanged and relatively higher than checkpointing scenario. Moreover, in terms of reliability, REABFD without checkpointing has achieved almost similar reliability achieved by RABFD with checkpointing.

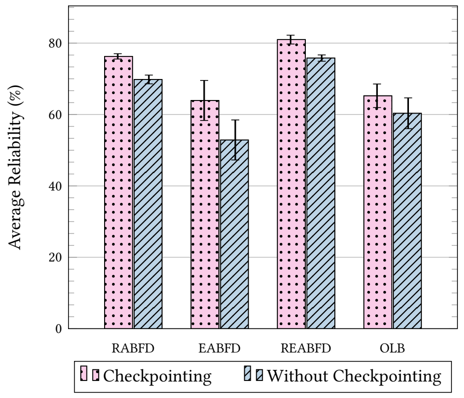

Figure 5: Average Reliability

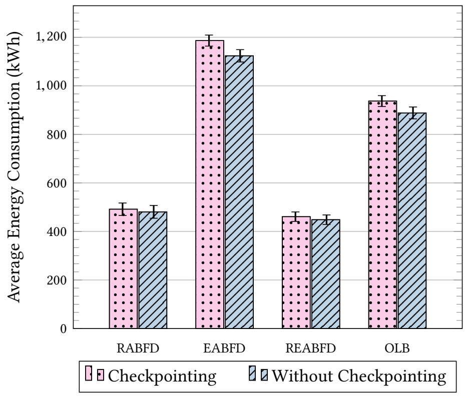

Figure 6: Average Energy Consumption

Figure 6 shows the average energy consumption incurred by all the policies with and without checkpointing. The energy consumption by REABFD is less in comparison to other policies with minimum difference of 7% from RABFD with and without checkpointing and maximum difference of 61% from EABFD with and without checkpointing. An interesting behaviour has been seen for EABFD policy as it consumes the maximum energy despite of the fact that it focuses on the provisioning of most energy efficient resources. From the given behaviour, it can be argued that the results are adverse. Rather than reducing the energy consumption, we ends up consuming more energy due to the energy losses incurred because of failure overheads (Fig. 7). In fact, it is better to use the random resource provisioning (OLB policy) than the energy aware resource provisioning in the presence of failures. In terms of energy wastage (Fig. 7), again REABFD outperforms all the policies with minimum improvement of approximately 8% and 11% over RABFD policy with and without checkpointing, respectively and maximum improvement of 67% and 70% over EABFD policy with and without

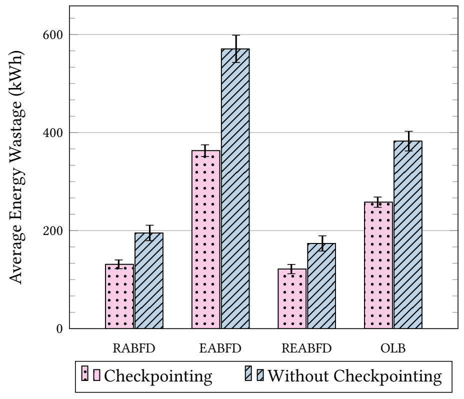

Figure 7: Average Energy Wastage

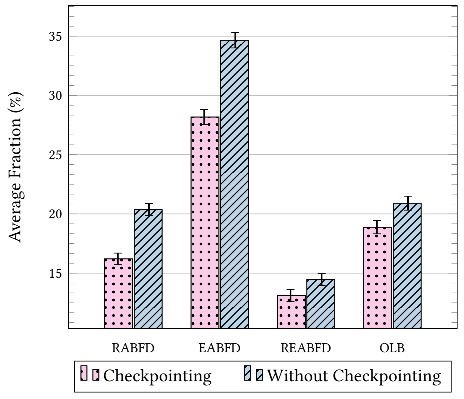

Figure 9: Average Deadline-Turnaround Time Fraction

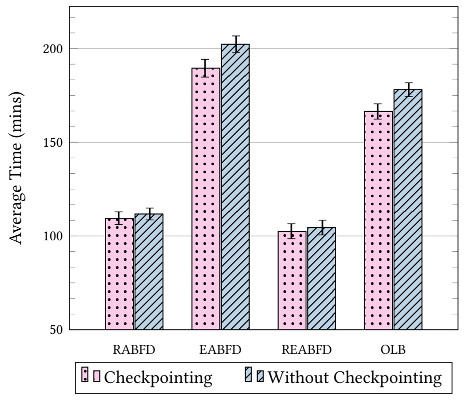

Figure 8: Average Turnaround Time

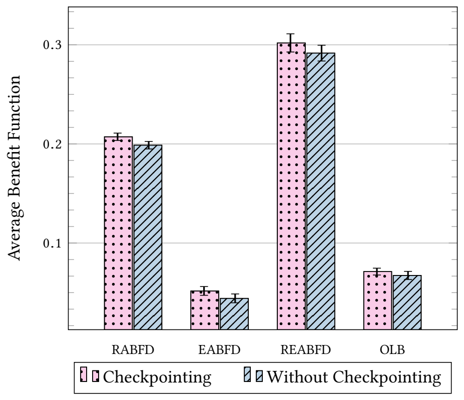

Figure 10: Average Benefit Function

checkpointing, respectively. Absence of any fault tolerance mechanism such as checkpointing further adds to these energy losses upto 36% because of the large re-execution overheads.

Figure 8 shows the average turnaround time, which is the time taken by each task of BoT application to finish. It can be seen that REABFD policy has the minimum turnaround time such that it took less time than what other policies took to finish the same application because of the less overheads incurred. REABFD policy has achieved better turnaround time by 7% from RABFD, 39% from OLB and 46% from EABFD for both checkpointing and without checkpointing scenarios. However, all the proposed policies have achieved better turnaround time while using checkpointing by 7% than without checkpointing because of less re-execution overheads occurred during the event of a failure. The achieved improvement in the turnaround time further justifies the improvements achieved in reliability and energy consumption for the REABFD policy.

Figure 9 shows, while executing the application by what percentage the makespan has been exceeded from the predefined deadline calculated by using equation 1. It can clearly be seen that for the secnarios without checkpointing, the makespan has been exceeded more upto 7% in comparison to the scenarios with checkpointing. This is because if a failure hits a scenario without checkpointing, then the re-execution overhead is huge which is found to be approximately 36% higher in comparison to checkpointing, which makes the deadlines violated with greater margin than the checkpointing scenarios. Among all the proposed policies, REABFD again outperforms the other proposed policies by exceeding less by 3%, 6% and 15% with checkpointing and 6%, 7% and 20% without checkpointing in comparison to RABFD, OLB and EABFD, respectively.

To measure the effectiveness of all the proposed policies in terms of reliability and energy consumption at the same time, we have used the benefit function (Fig. 10), which is the ratio of reliability and energy consumption. It is infered that the policies considering reliability factor (RABFD and REABFD) while provisioning the resources has the better benefit function than the policy considering only energy-efficiency (EABFD) and policy considering neither reliability nor energy-efficiency (OLB). Among all the policies, REABFD performs better by improving the benefit function by 29% over RABFD, 82% over EABFD and 76% over OLB with checkpointing. However, in the secnario without checkpointing REABFD policy gives better value of benefit function by 34% over RABFD, 85% over EABFD and 78% over OLB.

8 CONCLUSION

In this paper, the problem of reliability and energy aware resource provisioning in cloud computing systems has been addressed. In solving this problem, a scalable and elastic cloud computing architecture has been proposed with three list based greedy heuristic algorithms such as Reliability Aware Best Fit Decreasing (RABFD), Energy Aware Best Fit Decreasing (EABFD) and Reliability-Energy Aware Best Fit Decreasing (REABFD). For fault tolerance, both re-execution from the beginning and checkpointing mechanism for task recovery have been considered. Our extensive experiments revealed that if the emphasis is given only to the energy optimization without consideration of reliability, then the results will be contrary to what we expect. Rather than reducing the energy consumption, we ends up consuming more energy due to the energy losses incurred because of failure overheads. Among the proposed policies, Reliability-Energy Aware Best Fit Decreasing (REABFD) policy outperforms all the other policies and reveals that if both reliability and energy efficiency factors of resources are considered at the same time then both factors can be improved to a larger extent than being regulated individually.

In future work, we will explore the use of machine learning methods to predict the occurrence of correlated failures. With the prediction results, VM consolidation mechanism will be adopted to further optimize the fault-tolerance and energy consumption of the cloud computing systems.

REFERENCES

[1] Sebastian Anthony. 2013. Microsoft now has one million servers–less than Google, but more than Amazon, says Ballmer. *ExtremeTech. ExtremeTech* 19 (2013).

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. In *ACM SIGOPS operating systems review*, Vol. 37. ACM, 164–177.

[3] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems* 28, 5 (2012), 755–768.

[4] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, Albert Zomaya, et al. 2011. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in computers* 82, 2 (2011), 47–111.

[5] Tracy D Braun, Howard Jay Siegel, Noah Beck, Ladislau L Bölöni, Muthucumaru Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen, et al. 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed computing* 61, 6 (2001), 810–837.

[6] Rodrigo N Calheiros and Rajkumar Buyya. 2014. Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through DVFS. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on.* IEEE, 342–349.

[7] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* 41, 1 (2011), 23–50.

[8] Fabricio AB da Silva and Hermes Senger. 2011. Scalability limits of Bag-of-Tasks applications running on hierarchical platforms. *J. Parallel and Distrib. Comput.* 71, 6 (2011), 788–801.

[9] Nosayba El-Sayed and Bianca Schroeder. 2014. Checkpoint/restart in practice: When âĂŸsimple is betterâĂŹ. In *2014 IEEE International Conference on Cluster Computing (CLUSTER).* IEEE, 84–92.

[10] Nosayba El-Sayed and Bianca Schroeder. 2014. To checkpoint or not to checkpoint: Understanding energy-performance-i/o tradeoffs in hpc checkpointing. In *2014 IEEE International Conference on Cluster Computing (CLUSTER).* IEEE, 93–102.

[11] Peter Garraghan, Ismael Solis Moreno, Paul Townend, and Jie Xu. 2014. An Analysis of Failure-Related Energy Waste in a Large-Scale Cloud Environment. *IEEE Transactions on Emerging Topics in Computing* 2, 2 (2014), 166–180.

[12] Ponemon Institute. 2016. Cost of Data Center Outages. (2016), 1–21.

[13] Alexandru Iosup, Ozan Sonmez, Shanny Anoep, and Dick Epema. 2008. The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th international symposium on High performance distributed computing.* ACM, 97–108.

[14] Ravishankar K Iyer and David J Rossetti. 1986. A measurement-based model for workload dependence of CPU errors. *IEEE Trans. Comput.* 100, 6 (1986), 511–519.

[15] Bahman Javadi, Jemal Abawajy, and Rajkumar Buyya. 2012. Failure-aware resource provisioning for hybrid Cloud infrastructure. *Journal of parallel and distributed computing* 72, 10 (2012), 1318–1331.

[16] Tarandeep Kaur and Inderveer Chana. 2015. Energy efficiency techniques in cloud computing: A survey and taxonomy. *ACM Computing Surveys (CSUR)* 48, 2 (2015), 22.

[17] Derrick Kondo, Bahman Javadi, Alexandru Iosup, and Dick Epema. 2010. The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems. In *10th International Conference on Cluster, Cloud and Grid Computing (CCGrid).* IEEE/ACM, Melbourne, Victoria, Australia, 398–407.

[18] Adam J Oliner, Larry Rudolph, Ramendra K Sahoo, José E Moreira, and Manish Gupta. 2005. Probabilistic qos guarantees for supercomputing systems. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on.* IEEE, 634–643.

[19] Martin Perlin. 2012. Downtime, outages and failures-understanding their true costs. *Retrieved November* 25 (2012), 2012.

[20] Ian Philp. 2005. Software failures and the road to a petaflop machine. In *HPCRI: 1st Workshop on High Performance Computing Reliability Issues, in Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA-11).* San Francisco, California, USA, 125–128.

[21] Xiwei Qiu, Yuanshun Dai, Yanping Xiang, and Liudong Xing. 2016. A hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46, 3 (2016), 401–412.

[22] Bianca Schroeder, Garth Gibson, et al. 2010. A large-scale study of failures in high-performance computing systems. *IEEE Transactions on Dependable and Secure Computing* 7, 4 (2010), 337–350.

[23] Yogesh Sharma, Bahman Javadi, Weisheng Si, and Daniel Sun. 2016. Reliability and energy efficiency in cloud computing systems: Survey and taxonomy. *Journal of Network and Computer Applications* 74 (2016), 66–85.

[24] Amir Varasteh, Farzad Tashtarian, and Maziar Goudarzi. 2017. On Reliability-Aware Server Consolidation in Cloud Datacenters. *arXiv preprint arXiv:1709.00411* (2017).

[25] John W Young. 1974. A first order approximation to the optimum checkpoint interval. *Commun. ACM* 17, 9 (1974), 530–531.

[26] Longxin Zhang, Kenli Li, Changyun Li, and Keqin Li. 2016. Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems. *Information Sciences* 379 (2016), 241–256.

[27] Longxin Zhang, Kenli Li, Keqin Li, and Yuming Xu. 2016. Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems. *International Journal of Electrical Power & Energy Systems* 78 (2016), 499–512.

[28] Dakai Zhu, Rami Melhem, and Daniel Mossé. 2004. The effects of energy management on reliability in real-time embedded systems. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD-2004).* IEEE, 35–40.