Sustainable Computing: Informatics and Systems xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud

Altino M. Sampaio [a], Jorge G. Barbosa [b,*]

[a] CIICESI, Escola Superior de Tecnologia e Gestão, Instituto Politécnico do Porto, Felgueiras, Portugal
[b] LIACC, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal

ARTICLE INFO

Article history:
Received 15 May 2017
Received in revised form 5 October 2017
Accepted 28 November 2017
Available online xxx

Keywords:
Reactive and Proactive fault-tolerance
Failure prediction
VM migration
Deadline constrain scheduling
Energy minimization

ABSTRACT

As data centres continue to grow in size and complexity in order to respond to the increasing demand for computing resources, failures become the norm instead of an exception. To provide dependability at scale, traditional techniques to tolerate faults focus on reactive, redundant schemes. While the former relies on the checkpointing/restart of a job (which could incur significant overhead in a large-scale system), the latter replicates tasks, thus consuming extra resources to achieve higher reliability and availability of computing environments. Proactive fault-tolerance in large systems represents a new trend to avoid, cope with and recover from failures. However, different fault-tolerance schemes provide different levels of computing environment dependability at diverse costs to both providers and consumers.

In this paper, two state-of-the-art fault-tolerance techniques are compared in terms of availability of computing environments to cloud consumers and energy costs to cloud providers. The results show that proactive fault-tolerance techniques outperform traditional redundancies in terms of costs to cloud users while providing available computing environments and services to consumers. However, the computing environment dependability provided by proactive fault-tolerance highly depends on failure prediction accuracy.

1. Introduction

With trends in the service-oriented economy [1] being supported by distributed computing paradigms such as cloud computing, there is an increased concern about the quality and availability of offered services. However, providing quality of service (QoS) guarantees to users is a very difficult and complex task [2] due to the demands of the consumers' services vary significantly with time. Moreover, this problem is exacerbated when one considers computing node availability. By definition, a system is available if the fraction of its down-time is very small, either because failures are rare or because it can restart quickly after a failure. Therefore, availability is a function of reliability, which, according to the Institute of Electrical and Electronics Engineers (IEEE) [3], is the ability of a system or component to perform its required functions under stated conditions for a specified period of time. Reliability can be measured by the mean time between failures (MTBF), which in turn is estimated by the component manufacturer.

Despite the continuous improvements in reliability and overall quality of hardware components, an important remark is that the failure rate of a system grows in proportion to the number of processor chips in the system. Therefore, as networked computing systems continue to grow in scale and in the complexity of their components and interactions, component failures become the norm instead of an exception. According to Schroeder and Gibson [4], three of the most difficult and growing problems affecting large-scale computing infrastructures are avoiding, coping with and recovering from failures. Fu [5] reported that a node's MTBF is 1.25 h in a petaflop system. Considering that many scientific applications and experiments require a large number of nodes and weeks of computations to complete, failure and recovery are frequent events during an application's execution [6]. As a result, node failures make it difficult for applications to make forward progress, with consequential impacts on the QoS perceived by consumers. Furthermore, failures result in a loss of energy because the computation is lost and needs to be repeated after recovery.

It is evident that dealing with failure is essential in order to improve dependability at scale (a function of reliability and availability), which contributes to the success of ever-growing warehouse-sized data centres. A fault first causes an error in the service state of a system's component, which in turn may lead

* Corresponding author.
E-mail addresses: ams@estg.ipp.pt (A.M. Sampaio), jbarbosa@fe.up.pt (J.G. Barbosa).

https://doi.org/10.1016/j.suscom.2017.11.006
2210-5379/

to its subsequent service failure [7]. To this end, fault-tolerance mechanisms are aimed at failure avoidance, thus giving operational continuity to the system when one of the resources breaks down. Egwutuoha et al. [8] describes the major techniques to overcome failures and improve reliability and availability of networked systems. A full revision of fault-tolerant systems is beyond the scope of this paper. Further information about this subject can be found in [8,9]. Three of the most known and applied fault-tolerance mechanisms are checkpoint/restart, redundancy, and proactive fault-tolerance. Checkpoint/restart mechanisms continuously checkpoints a process (or a whole virtual machine (VM) in the case of cloud computing) and is known to incur significant overhead in large-scale systems. For example, Philp [10] estimated that the checkpoint overhead prolongs a 100-h job (in the absence of failure) by an additional 151 h in petaflop systems. This cost is expected to increase with exascale on the horizon, because an exascale machine is expected to experience a failure every few minutes [11,12] due to the extremely large number of components. Not even more recent improvements with respect to checkpoint/restart fault tolerance scheme (e.g., [13,14]) are able to limit the overhead to acceptable levels in the context of deadline-driven applications, which is the scenario considered in this study. For this reason, this paper does not consider the checkpoint/restart mechanism. In turn, redundancy is a traditional fault-tolerance technique which applies the replication of tasks. The third alternative, proactive fault-tolerance, represents a new trend on system dependability and leverages failure prediction techniques to anticipate failures by migrating a task from a health-deteriorating physical machine (PM) to a healthy PM.

The contribution of this paper consists of a realistic comparison of redundancy and proactive fault-tolerance techniques in terms of energy consumption and completion rate of jobs, where job characteristics follow the last version of Google cloud tracelogs [15]. For that purpose, a set of independent CPU-bound deadline-constrained jobs is submitted to a cloud data centre in which computer nodes are subject to failure. A failure prediction tool [5] is used to carry out experiments on proactive fault-tolerance. To provide a fair comparison, two similar Best-Fit-based scheduling algorithms are combined with failure tolerance mechanisms (namely, redundancy and proactive fault-tolerance) to implement highly available virtual computing environments. The objective is to understand to what extent redundancy can deliver reliable and available services to consumers when compared to proactive fault-tolerance and at what cost to cloud providers in terms of electricity bills. The analysis presented in this paper is an improved version of our previous work [16], in the sense that it additionally does the following: (1) applies the same scheduling algorithm with both fault-tolerance techniques to provide a fair comparison between the methods; (2) updates MTBF to 300 min, in accordance with previous studies (e.g., [17,14]); (3) submits a higher number of tasks, which characteristically follows the Google Cloud tracelogs [15]; (4) evaluates the performance of the schedules with a proactive fault-tolerance mechanism, with respect to the failure prediction accuracy; and (5) considers 4 types of VM.

The remainder of this paper is organized as follows. Section 2 discusses related works based on fault-tolerance techniques in the cloud. Section 3 presents the implemented scenario to conduct tests. Section 4 describes the metrics and discusses obtained results. Section 5 concludes this paper.

2. Related work

Over recent years, several techniques have been developed to improve system availability. Redundancy and proactive fault-tolerance are two well-known techniques that aim to provide reliability and availability at scale. Regarding redundancy, Ferreira et al. [18] evaluated the viability of using state machine replication to address failures. Redundant copies of Message Passing Interface (MPI) processes, partially combined with checkpointing, provide failover capabilities. Results reveal that the state machine replication approach to exascale resilience outperforms traditional checkpoint/restart approaches. Casanova et al. [19] investigated two approaches for replication, including where entire application instances are replicated and where each process in a single application instance is replicated. Authors have concluded that replication is worthwhile when compared to the no-replication case (instead of using the pure checkpoint-recovery approach) in terms of expected application execution times. Mills et al. [20] proposes a shadow replication computing computational model in order to overcome failures while minimizing system power. A shadow process is an exact replica of the main process, scheduled to execute concurrently with the main process, but at a different computing node. To minimize energy, initial shadow processes execute at lower processor speeds and immediately take over the role of the main process at an increased speed if the main process fails. The idea is to maintain the application's QoS requirements in terms of application response time in the presence of system faults. However, authors do not compare their approach to proactive fault-tolerance schemes.

Regarding proactive fault-tolerance techniques, in [5] they investigated and proposed failure-aware compute node selection strategies for the construction and reconfiguration of virtual computing environments. The approach leverages proactive failure management techniques, based on virtual machines (VM) migration, and considers both the performance and reliability status of computing nodes. Later, Sampaio et al. [17] proposed a dynamic power- and failure-aware scheduling algorithm to build highly available virtual clusters to run cloud jobs for consumers. Authors proposed the Power- and Failure-Aware Relaxed Time Execution scheduling algorithm, which considers failure prediction on PMs, to construct highly available virtual clusters on top of failure-prone computing environments. A VM is migrated away from the PM predicted to fail to a healthy one. Moreover, Liu et al. [21] followed a proactive fault-tolerance approach to enhance cloud service reliability based on the CPU temperature model. Despite the interesting proactive fault-tolerance schemes proposed, authors do not compare their approach to traditional redundancy techniques in terms of resilience and energy consumption.

There are several studies in the literature that survey and compare different fault-tolerance mechanisms [22–24]. Unfortunately, these studies lack a quantitative comparison regarding energy costs to service providers and system availability and service performance to consumers. Moreover, despite redundancy possibly being significantly efficient, it may lead to waste due to replication of tasks (i.e., it requires additional resources to run the replications of the main task). In turn, proactive fault-tolerance highly depends on the ability to predict the failure. This study intends to compare these two fault-tolerant techniques in terms of costs to cloud providers and jobs executed to consumers.

3. Fault-tolerant cloud environment

This section provides a formal description of a fault-tolerant cloud environment.

3.1. System overview

A cloud computing environment consists of a cloud provider and multiple consumers. The computing infrastructure is composed of y PMs, where H is the vector representing the PMs such that $H = \{h_1, \ldots, h_y\}$. PMs are homogeneous in terms CPU capacity C, memory capacity M, network bandwidth N, and equal access to a shared

Please cite this article in press as: A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustain. Comput.: Inform. Syst. (2017), https://doi.org/10.1016/j.suscom.2017.11.006

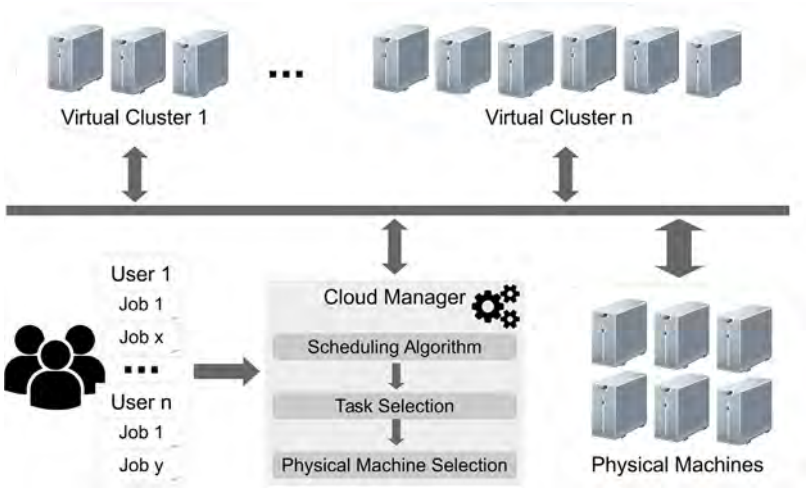


Fig. 1. Distributed system architecture.

storage space S for storing VM disk images and may have distinct predicted times for future failures F_i such that $h_i = \{C, M, N, S, F_i\}$. Fig. 1 depicts a typical usage scenario. Consumers submit jobs to the system, and the cloud manager module reserves the necessary resources from the cloud infrastructure to run their jobs. Each job J can be seen as a set of independent deadline-driven CPU-bound tasks $\tau_i, J = \{\tau_1, \ldots, \tau_n\}$. The cloud manager, having no knowledge of when jobs arrive, runs the resource allocation algorithm that creates task mappings to machines. Then, the cloud manager creates and manages VMs to execute the tasks. Each VM runs on top of a PM at a time. The set of VMs created to execute the tasks of a user form the user's virtual cluster execution environment. It is the responsibility of the cloud manager component to allocate, manage, and deallocate virtual clusters. A VM encapsulates the task execution environment and is the unit of migration in the system. A VM runs a single task. Multiple distinct VMs can be mapped to a single PM. Consumers specify a deadline for a job based on the job's longest task. The computing environment acts as best-effort, since resources may not be available at the submission time, delaying the start of jobs' executions. Job deadlines become activated right after submission.

Migration of VMs is required to implement a proactive fault-tolerance technique. The technique, which is used to transfer a VM's instance across PMs, has also served as a main approach to achieve better energy efficiency of virtualized data centres by means of server consolidation to allow more PMs to be turned off. The two types of available migration are 'stop and copy' and live migration [25]. Stop and copy migration involves halting the VM, copying its contents to the destination host and then resuming it. Live migration performs the same logical functionality but with almost no downtime for the transition. In live migration, the overhead introduced is a function of the frequency of the writing to memory pages, since dirty memory pages are continuously transferred until the set is fully transferred to the destination. In stop and copy migration, the migration time depends on the amount of memory allocated to the VM, which is completely transferred. Thus, the cost is mostly constrained by the network bandwidth. Several studies [5,26,17] on VM migration with respect to working set size have shown that live migration of VMs can almost double the migration time compared with the time associated with the migration of VMs using the stop and copy alternative. To clarify, the migration of VMs with the stop and copy method requires an average of 12 to 14 s, in contrast to the average of 14 to 25 s required for a live migration. Despite the shorter service downtime in the latter case, its overall migration time is longer compared to the former approach.

Due to imperfect failure prediction, the implemented proactive fault-tolerance system applies stop and copy migration since the overall migration time is shorter, which is of paramount importance in the face of failure. For simplicity, tasks with deadlines extending past the failure time of their node migrate $\xi = 3$ min before the node's predicted failure time. A previous study [17] has shown that migrating a VM instance 3 min before a failure was predicted to occur on a PM resulted in a good commitment between migration time and the time to the next forecasted failure by the failure predictor. In the considered scenario of the computing infrastructure, the network bandwidth was set to 1 Gbit/s.

3.2. Nodes power consumption

Since the power consumed by the active PMs is mainly dictated by the CPU resource [27], only the CPU power consumption in our energy model is considered. The power consumption of a PM i (i.e., P_i) can be estimated based on the linear power model presented in (1).

$$P_i = p1 + p2 \times CPU\%, \quad (1)$$

where $CPU\%$ is the percentage of CPU utilization for a given time interval, measured for a PM i at runtime. The $p1$ and $p2$ factors are the power consumption when the PM is in idle mode and the additional power consumption from CPU utilization, respectively. The factor $p2$ is typically proportional to the overall system load. In this study, $p1$ and $p2$ were set to 112 watts and 48 watts, respectively. These values follow that an idle server represents 60–70% of the power consumed when it is fully utilized [28].

3.3. Workloads and failures description

A set of synthetic jobs was created to conduct simulations. The set consisted of 30 different users submitting a total of 760 synthetic jobs which totaled 1400 tasks, with characteristics that followed an extensive study about the last version of Google Cloud tracelogs [15]. According to this study, the jobs inter-arrival time, tasks length, CPU task requirements, and the number of tasks per job can be modelled using a Lognormal distribution. The authors of the study also argue that approximately 75% of jobs only run one task and most of the jobs have less than 28 tasks that determine the overall system throughput. The average length of a job is 3 min and the majority of jobs complete in less than 15 min, although there are a small number of jobs that run longer than 300 min. The CPU usage varies from almost 0% to approximately 25%. Job inter-arrival

Please cite this article in press as: A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustain. Comput.: Inform. Syst. (2017), https://doi.org/10.1016/j.suscom.2017.11.006

mean time was 4 s. Regarding RAM usage, most of the tasks use less than 2.5% of the node's RAM. Each task deadline was rounded up to 10% more than its minimum necessary execution time. The job deadline equals the deadline of its longest task.

Regarding the reliability of nodes, Schroeder and Gibson [4] stated (using a large-scale study of failures in high-performance computing systems) that the MTBF of physical nodes and the entire system is well described by a Weibull distribution with a shape parameter of 0.8. The same authors stated that the mean time to repair (MTTR) should have a Lognormal distribution applied. This paper leverages such distributions to simulate failures on PMs. The MTBF was programmed according to [17] with an average value of 300 min. Regarding MTTR, failed nodes stay unavailable during a period with a mean time set to 15 min and varied up to 60 min. Proactive fault-tolerance was implemented following the description of the failure predictor tool presented in [5,29]. The authors of the tool argued that the predicted tool is able to accurately predict failure occurrences with an average accuracy of 76.5% and with a prediction error due to false negatives equal to 3.7%. According to the same authors, the prediction error in forecasting the time at which the next failure is going to occur in a certain PM is defined as err = (ActualTime − PredictedTime)/ActualTime * 100%. The sensitivity of scheduling algorithms to the accuracy of failure prediction was well studied and properly documented in [5,17]. In this study, a failure event is defined as the occurrence of any anomaly caused by hardware or software faults and unstable behaviour that precludes computing infrastructure components from working.

3.4. Instance specification

A VM instance encapsulates a set of resources and represents the execution environment of a task. In this regard, the resource allocator, incorporated in the cloud manager module, has to select from a set of types of VM the most appropriated VM instance to run a task. The existing types of VM deployed by the cloud provider are described in Table 1. Each type offers a specific CPU performance capability, expressed in GFlops/s, and consumers are charged $ every 5 min. As an example, consider a task τ that performs a matrix multiplication that requires $2n^3$ flops to compute matrices of size (n, n). If the task τ only requires n amount of resources to execute at the maximum speed [30], the task will need $2n^3/n$ s to complete its execution. Please note that the amount of required resources n can eventually be less than the maximum processing capacity deployed by a VM instance. This study assumes that n is defined by the user. The VM migration overhead with a stop and copy migration is presented in the "Migration Time(s)" column, as measured in [5]. In the case of the proactive fault-tolerance technique, first a scheduling algorithm decides to which healthy nodes the VMs are going to be migrated away from the suspicious node. Then, VMs are placed in queue and await migration. In this sense, migration of VMs occurs sequentially, meaning that VMs migrate one at a time from/to a PM. The scheduling algorithm has access to the queue size and necessary migration time for each VM instance in the queue in order to make resource allocation decisions. An in-queue VM instance continues its execution while migration does not start.

Table 1
Characterization of the VM instances on the cloud infrastructure (granularity of billing cycles is 5 min).

VM type	GFlops/s	Cost ($)	Migration time (s)	RAM (MB)
A	0.323	0.0258	9	256
B	0.537	0.0508	10	512
C	0.685	0.0725	12	1024
D	0.753	0.0833	16	2046

3.5. Scheduling algorithms

To construct highly available fault-tolerant virtual clusters of VMs on top of PMs to which user tasks are scheduled and executed, this study leverages a common scheduling algorithm similar to the well-known Best-Fit Decreasing heuristic [31]. The resource allocator in the cloud manager module utilizes power, cost, and failure models to estimate the future system state and select the best placement based on the two step optimization criteria: (1) selection of the less costly type of VM that provides required resources to complete the task by its deadline; (2) placement of the VM in the fullest PM that still provides required resources. In the first step, defined by (2), the scheduler algorithm selects from the set of available types of VM, $\{v_A, v_B, v_C, v_D\}$ (please, refer to Table 1) the cheapest VM q (i.e., v_q) that provides resources $r(v_q)$ so that the task τ is able to run with enough resources (i.e., $r(\tau)$) to conclude by its deadline d_τ.

$$\exists v_q \in \{v_A, v_B, v_C, v_D\} : \quad r(\tau) \leq r(v_q) \wedge v_q = \underset{k \in \{A,B,C,D\}}{arg\,min}\, Cost(v_k). \tag{2}$$

The second step aims at minimizing energy consumption by consolidating multiple VMs in a PM. However, this optimization problem is constrained by the amount of resources on the PM h_i, as is indicated by (3), where $r(v_k)$ is the resource required by VM k (in GFlops/s), $\{v_1, v_2, \ldots, v_n\}$ is the set of VMs running on the PM i (i.e., h_i), and $r(h_i)$ is the resource that the PM can supply.

$$r(v_q) + \sum_{k=1}^{n} r(v_k) \leq r(h_i). \tag{3}$$

To compare the two state-of-the-art fault-tolerant techniques, two related Best-Fit Decreasing-based scheduling algorithms were created: (1) Common Best-Fit for Replication (CBFIT_R); and (2) Common Best-Fit for Proactive fault-tolerance (CBFIT_PFT). The CBFIT_R scheduling algorithm does not consider the reliability of physical nodes when scheduling tasks, and consequently, it does not perform migrations to avoid failure. As such, when a physical node fails, all the tasks running in it have to re-initiate from scratch. CBFIT_R applies the replication of tasks to implement fault-tolerance by means of redundancy. This study simulates redundancy based on 0, 1, and 2 task replicas. For simplicity, for the remainder of this paper, these redundancy schemes will be denoted as CBFIT_R0, CBFIT_R1, and CBFIT_R2, respectively. In this sense, CBFIT_R0 means that no fault-tolerance technique is applied. For each main task, CBFIT_R also schedules the pre-defined number of exact replicas to execute concurrently but on different computing nodes. To optimize energy and resource consumption, the replications of a main task are terminated as soon as one of them successfully completes. This study does not consider an additional number of replications due to the amount of submitted jobs and the limited number of resources (100 physical nodes). In this regard, running an additional number of replications could ultimately lead to the opposite effect in the completion rate of users' jobs, due to resource contention and the incoming workloads that must be finished by their deadlines. Moreover, because an additional number of resources would be used to run replications, the costs due to electric bills would increase significantly. Nevertheless, CBFIT_R0, CBFIT_R1, and CBFIT_R2 represent a good approach to understanding how energy consumption and fault-tolerance evolve with a diverse number of replications.

Regarding the CBFIT_PFT scheduling algorithm, it considers the predicted failure occurrences on PMs to proactively move VMs away from suspicious nodes. The algorithm starts by creating a list of sorted tasks in ascending order according to the difference between relative task deadlines and their minimum execution

Please cite this article in press as: A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustain. Comput.: Inform. Syst. (2017), https://doi.org/10.1016/j.suscom.2017.11.006

time. Then, CBFIT_PFT iteratively picks up the first task from the list, chooses the less costly VM providing required resources to complete the task by its deadline (please, refer to (2)), and places the VM in the PM provided the minimum required CPU capacity. In the case of several PMs providing the same minimum required processing capacity, the algorithm chooses the PM with higher reliability. Two situations may occur regarding the selection of a PM to place the VM which will execute the task: (1) the deadline d_τ of the task τ is shorter than the predicted instant of failure δ_i on the PM i; (2) the deadline is equal to or longer than the predicted instant of failure on the PM. The first case is expressed by (4), where $W(\tau, \mu)$ (in Gflops) is the remaining workload for task τ, at the current instant μ.

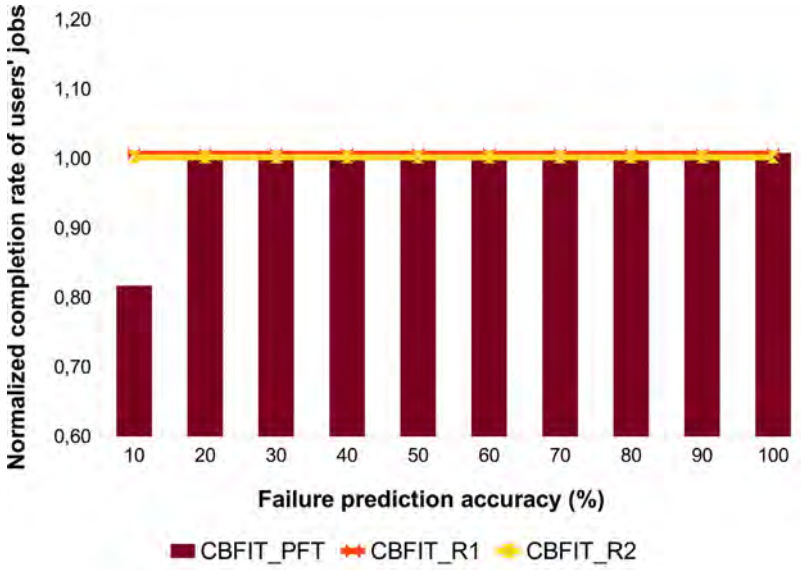
$$r(\tau) \times (d_\tau - \mu) \geq W(\tau, \mu) \quad \text{if} \quad \left((\delta_i - \mu) > \frac{W(\tau, \mu)}{r(\tau)}\right). \tag{4}$$

In the second case, CBFIT_PFT verifies that the PM i can execute part of the task before a failure in i occurs, as evaluated by (5).

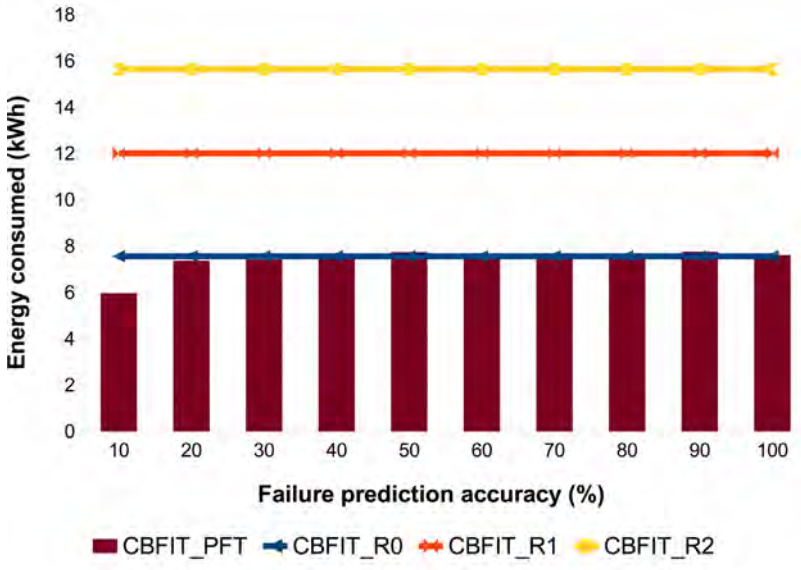
$$r(\tau) \times (\delta_i - \mu - m_{ei}) + r(\tau) \times (d_\tau - \delta_i - m_{ei}) \geq W(\tau, \mu), \quad \text{if} \quad \left((\delta_i - \mu - m_{ei}) \leq \frac{W(\tau, \mu)}{r(\tau)}\right), \tag{5}$$

where m_{ei} is the migration cost of a VM/task, from node e to node i and which value is reported in Table 1. This equation indicates that task τ can be (re)scheduled or migrated from PM e to a non-reliable PM i at time $\mu + m_{ei}$ which is predicted to fail at instant δ_i, (1) it provides the resources $r_i(\tau)$ required by task τ during the $\delta_i - \mu$ time interval, and (2) task τ will have to migrate to another currently unknown PM where it cannot require more resources than those needed to execute at maximum speed $r(\tau)$.

Since both scheduling algorithms are best-fit heuristics in terms of CPU capacity (i.e., it selects the PM that has the minimum required CPU capacity that is required to complete a task within



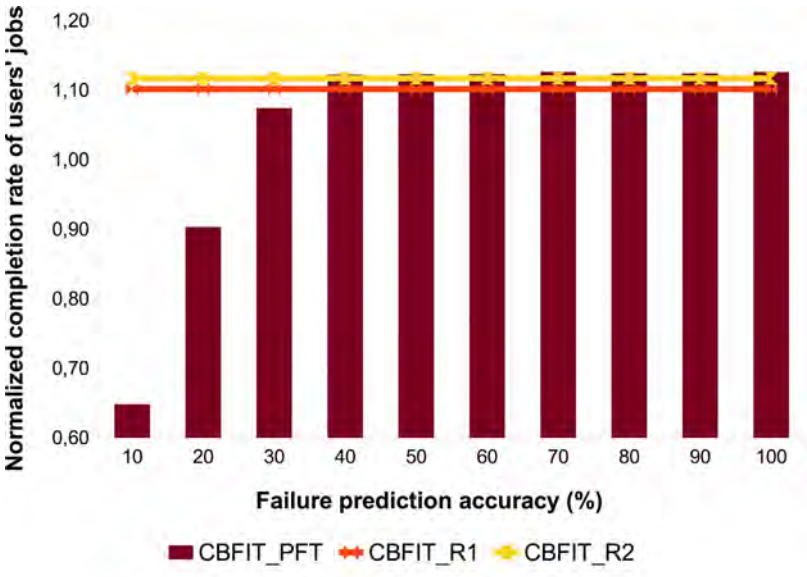
(a) Normalized completion rate of users' jobs



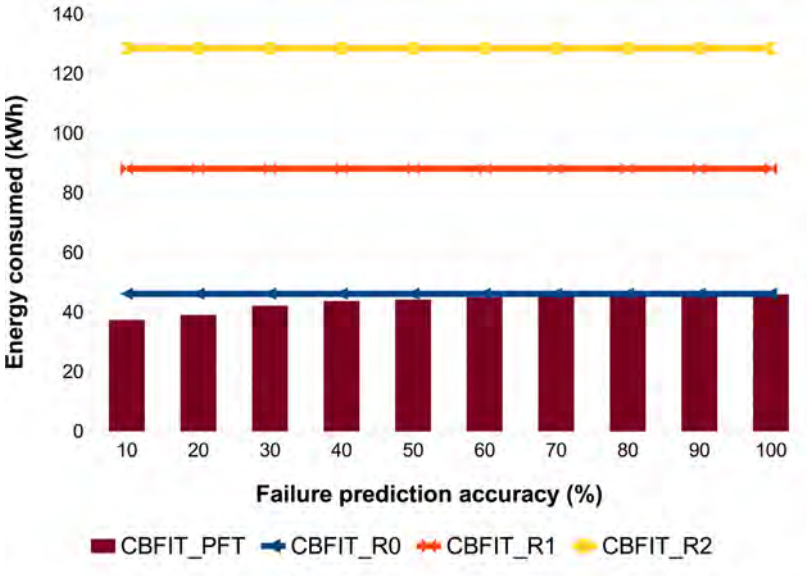
(b) Energy consumption

Fig. 2. Performance of fault-tolerance mechanisms, for an average task length to MTBF ratio of 0.01 (MTBF = 300 min). The completion rate of users' jobs is normalized with reference to CBFIT_R0.

Please cite this article in press as: A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustain. Comput.: Inform. Syst. (2017), https://doi.org/10.1016/j.suscom.2017.11.006



(a) Normalized completion rate of users' jobs



(b) Energy consumption

Fig. 3. Performance of fault-tolerance mechanisms, for an average task length to MTBF ratio of 0.10 (MTBF = 300 min). The completion rate of users' jobs is normalized with reference to CBFIT_R0.

its deadline), and considering that compute nodes are homogeneous, the scheduling will result in consolidating the running tasks (i.e., VMs) in fewer resources so that energy consumption may be minimized. As such, the aim is to build energy-efficient and fault-tolerant virtual cluster execution environments to execute user's jobs, based on two distinct state-of-the-art fault-tolerance mechanisms.

4. Evaluation and results

This section describes the metrics used to evaluate the two fault-tolerance mechanisms and analyses the results obtained through simulation.

4.1. Evaluation metrics

The two metrics defined to assess the performance of the two fault-tolerance mechanisms are the completion rate of users' jobs (E_J) and energy consumption (E_E). The completion rate of users' jobs, E_J, expressed as (6), is the ratio of completed jobs J_C (i.e., the jobs to which all the tasks have been executed within their deadlines) to the number of submitted jobs J_S. The value of the ratio falls in the interval [0, 1]. The availability of the computing environment and services provided increases as E_J approaches 1.

$$E_J = \frac{J_C}{J_S}. \tag{6}$$

The energy consumption E_E, expressed in kWh and presented by (7), is determined by dividing the average power consumption (in watts) of the computing infrastructure (i.e., for all active PMs u

Please cite this article in press as: A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustain. Comput.: Inform. Syst. (2017), https://doi.org/10.1016/j.suscom.2017.11.006

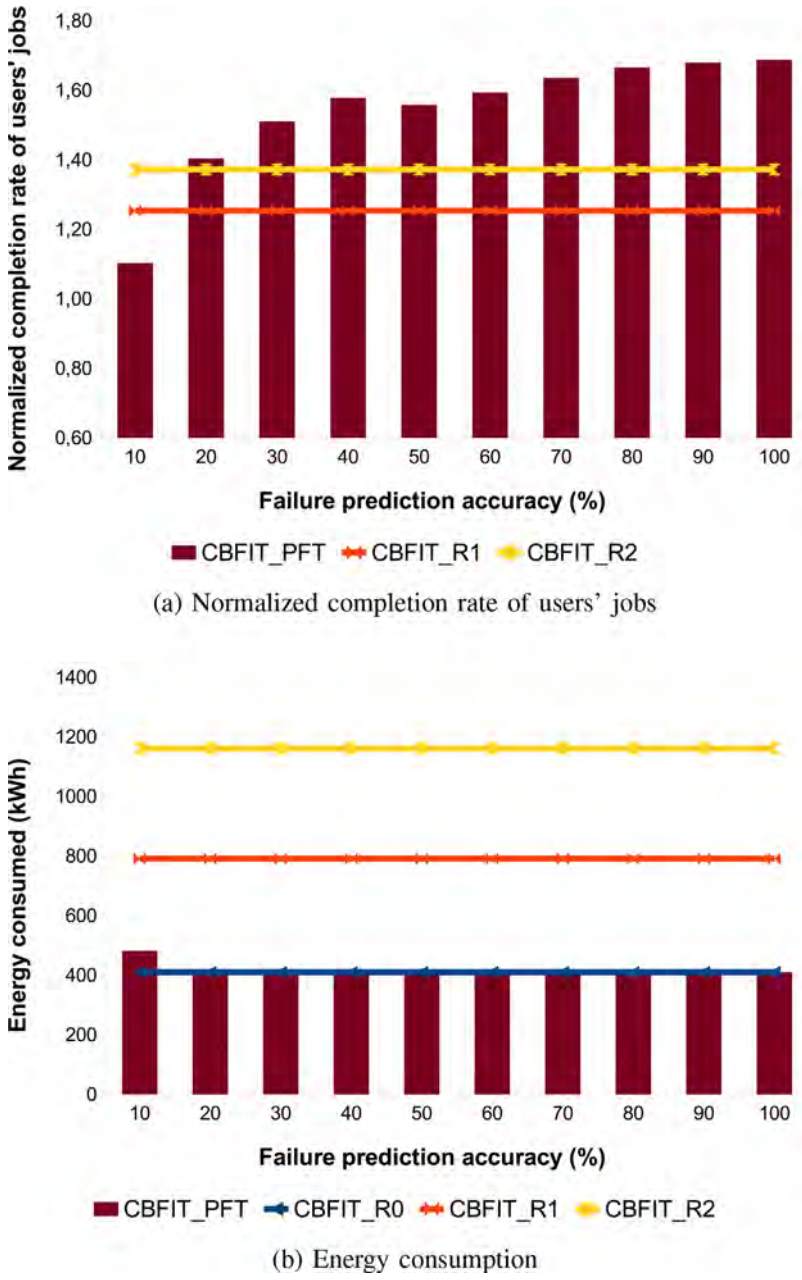


Fig. 4. Performance of fault-tolerance mechanisms, for an average task length to MTBF ratio of 1.00 (MTBF = 300 min). The completion rate of users' jobs is normalized with reference to CBFIT_R0.

at all sample times f) by 60 (since power consumption samples are obtained each minute).

$$E_E = \frac{\sum_{s=1}^{f}\sum_{i=1}^{u}\theta_i \times \frac{P_i}{1000}}{60}, \qquad \theta_i = \begin{cases} 1, & \text{if PM } i \text{ is active at instant } s \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Considering that energy costs (per kWh) is c, the electricity bill B_E may be calculated based on energy consumption as $B_E = E_E \times c$. From the business point of view, the cloud provider is interested in reducing energy costs that augment operational costs in order to maximize revenues. Thus, availability must be protected in a cost-effective manner and E_J and E_E may be used to select the appropriate fault-tolerance technique. In this regard, the optimization objective is the maximization of the completion rate of users' jobs, E_J, while energy consumption E_E is minimized.

4.2. Results analysis

Three sets of simulations were conducted and comprised proactive fault-tolerance (i.e., CBFIT_PFT) and redundancy based on 0, 1, and 2 redundant copies of tasks (i.e., CBFIT_R0, CBFIT_R1, and CBFIT_R2, respectively). Each set considered an average task length to MTBF ratio of 0.01, 0.10, and 1.00, respectively. The rationale is to measure the impact of the average task length to MTBF ratio in the performance of the fault-tolerance techniques. Additionally, the failure prediction accuracy was varied within the interval [10 . . . 100] for the CBFIT_PFT scheduling algorithm in order to evaluate the sensitivity of the fault-tolerance mechanism due to changing accuracy of the failure prediction tool. These simulations

Please cite this article in press as: A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustain. Comput.: Inform. Syst. (2017), https://doi.org/10.1016/j.suscom.2017.11.006

were conducted in a computing infrastructure comprised of 100 homogeneous nodes with MTBF fixed at 300 min.

The results for an average task length to MTBF ratio of 0.01 are shown in Fig. 2. In Fig. 2a, the results for the completion rate of users' jobs are normalized with respect to CBFIT_R0. One can observe that for a low average task length to MTBF ratio, proactive fault-tolerance and redundancy achieve similar results. The CBFIT_R0 algorithm (i.e., the no fault-tolerance technique is applied) is able to complete approximately 99.2% of jobs and is outperformed by others in approximately 0.8% only. However, the CBFIT_PFT algorithm is the only algorithm that reaches 100% job completion. However, the performance of CBFIT_PFT degrades for low prediction accuracy values, performing worse than any CBFIT_R for prediction accuracies less than 20%. To understand these results, please note that failure prediction accuracy is defined as the difference between the actual and predicted failure time expressed as a percentage. For example, if a physical node is supposed to fail at minute 100, the failure predictor tool will predict the failure at minute 10 if the failure prediction accuracy is 10%. This means that failure prediction inaccuracies have a direct impact on the MTBF perceived by the CBFIT_PFT algorithm that considers node reliability. Unlike CBFIT_PFT, the CBFIT_R scheduling algorithm does not consider the reliability of physical nodes, and consequently, it reaches a similar completion rate of users' jobs irrespective of δ_i. These results are in line with those registered by two previous related studies [5,17].

Regarding the energy consumption, Fig. 2b shows that CBFIT_PFT and CBFIT_R0 are the most energy-efficient fault-tolerance alternatives, since both consume about the same energy, and the two alternatives outperform redundant techniques for multiple task replications. Despite CBFIT_R1 and CBFIT_R2 using an additional number of tasks, the energy consumption is not linear to the number of replications because the replications of a main task are terminated as soon as one of them successfully completes. In this sense, CBFIT_PFT and CBFIT_R0 represent the best alternatives for cloud providers dealing with a computing infrastructure with an MTBF much higher than the average length of users' tasks. They consume less energy (thus less associated electricity bills) and complete a higher rate of jobs.

Fig. 3 shows the results for an average task length to MTBF ratio of 0.10. As presented by Fig. 3a, in which the results for completion rate of users' jobs are normalized with respect to CBFIT_R0, the performance of the algorithms degrades relatively to the average task length to MTBF ratio of 0.01. In the case of CBFIT_R0 (i.e., no fault-tolerance technique is applied), the completion rate of users' jobs decreases from nearly 99.2% to approximately 88.7% as the average task length to MTBF ratio increases from 0.01 to 0.10. In turn, CBFIT_R1 and CBFIT_R2 achieve rates of 97.7% and 99.1%, respectively, while CBFIT_PFT reaches 99.8% an average failure prediction accuracy of 75%. Compared to an average task length to MTBF ratio of 0.01, the CBFIT_PFT performance now decreases faster for low failure prediction accuracies, being outperformed by CBFIT_R1 and CBFIT_R2 alternatives for failure prediction accuracies equal to or lower than 30%. Regarding energy consumption, Fig. 3b shows that the CBFIT_PFT is the most energy-efficient fault-tolerance technique. Moreover, energy consumption significantly increases when in the case of an average task length to MTBF ratio of 0.01, since tasks now run for a longer time. A final remark is that the gap in energy consumption among fault-tolerance alternatives is larger with the average task length to MTBF ratio.

Relative to the last set of experiments, Fig. 4 shows the results for an average task length to MTBF ratio of 1.00. Fig. 4a indicates an evident decline in the completion rate of users' jobs for all the scheduling algorithms when compared to the previous results for average task length to MTBF ratios of 0.01 and 0.10. In particular, CBFIT_R0, which represents the scheduling algorithm that does not apply any fault-tolerance measures, achieves the worst results,

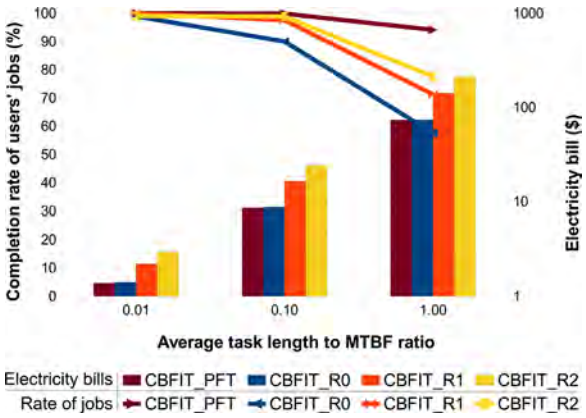


Fig. 5. Performance of fault-tolerance mechanisms, for an average failure prediction accuracy of 75% (MTBF = 300 min; energy cost per kWh is $0.189; rectangles represent the electricity bill while lines indicate completion rate of jobs).

completing almost 40% less jobs than CBFIT_PFT with a failure prediction accuracy of 100%. Furthermore, if one considers that the author of the failure prediction tool used in this study claims it is able to predict failure occurrences with an average accuracy equal to 75%, it becomes clear that CBFIT_PFT outperforms that redundancy-based fault-tolerance alternative, completing almost 94% of submitted jobs, and outperforms the second better alternative, CBFIT_R2, in almost 16%. Similar to the results obtained in the other two sets of experiments, redundancy can be interesting when compared to proactive fault-tolerance to achieve higher completion rates of users' jobs in cases where the failure prediction accuracy drops below 20%. Concerning energy consumption, Fig. 4b shows CBFIT_PFT as the fault-tolerance alternative that consumes less energy, thus representing the most attractive technique to cloud providers in terms of infrastructure costs. Moreover, we observe a tendency to increase the gap in the energy consumption among fault-tolerance alternatives as the average task length to MTBF ratio increases.

Taking into consideration that the author of the failure prediction tool claims that it is able to predict failures with an average accuracy near to 75%, the next set of experiments fixes the failure prediction accuracy in this value for the CBFIT_PFT scheduling algorithm and varies the average task length to MTBF ratio within {0.01, 0.10, 1.00}. Furthermore, one defines the energy cost per kWh as $0.189 in order to evaluate the electricity bill based on the consumed energy necessary to execute submitted jobs. The performance of CBFIT_PFT is compared to CBFIT_R0, CBFIT_R1, and CBFIT_R2 under enunciated conditions. Fig. 5 shows the completion rate of users' jobs and the electricity bill expressed in dollars due to the consumed energy, here represented by the lines and rectangles, respectively. As before, the tendency is that the completion rate of users' jobs decreases as the average task length to MTBF ratio increases from 0.01 to 1.00. The impact on the performance of the fault-tolerance mechanisms is more evident in the case of an average task length to MTBF ratio equal to 1.00. CBFIT_PFT outperforms the redundancy-based fault-tolerance mechanisms CBFIT_R0, CBFIT_R1, and CBFIT_R2 in terms of the completion rate of users' jobs for all average task length to MTBF ratios. Moreover, CBFIT_PFT consumes less energy than CBFIT_R1 and CBFIT_R2 and about the same as CBFIT_R0. In this regard, an important result is that CBFIT_PFT is able to complete more jobs while consuming less energy, consequently resulting in lower electricity bills. If one considers that energy consumption heavily contributes to increasing provider operational costs and carbon footprints to the environment, one can conclude that CBFIT_PFT benefits both consumers

Please cite this article in press as: A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustain. Comput.: Inform. Syst. (2017), https://doi.org/10.1016/j.suscom.2017.11.006

and providers, since it is able to achieve higher rates of completed users' jobs at lower expenses for cloud providers.

5. Conclusion

This paper has evaluated the effectiveness of two state-of-the-art fault-tolerance mechanisms in providing dependable cloud services to consumers. Failures represent a threat to the availability and dependability of a system and services deployed. Unfortunately, with an ever-growing number of warehouse-sized data centres built to address the increasing demand for computing resources and services, component failures become the norm instead of an exception. As such, the success of petascale/exascale computing will depend on the ability to provide reliability and availability at scale. To address this problem, different fault-tolerance mechanisms exist to provide dependable computing environments. However, dependability of services at all costs is not a solution, as it may increase energy consumption and operational costs for providers.

This study assessed two state-of-the-art fault-tolerance mechanisms, namely, redundancy and proactive fault-tolerance. The obtained results based on simulations show that proactive fault-tolerance outperforms redundancy up to 2 replications (i.e., three copies of the same task running concurrently) in terms of jobs successfully executed and less energy consumption. However, the performance of the proactive fault-tolerance alternative is highly dependent on failure prediction accuracy. In fact, the fault-tolerance alternative based on failure prediction mechanisms is outperformed by redundancy when the prediction accuracy drops below 20%.

References

[1] D. Armstrong, K. Djemame, Towards quality of service in the cloud, Proc. of the 25th UK Performance Engineering Workshop (2009).
[2] V. Stantchev, C. Schröpfer, Negotiating and enforcing qos and slas in grid and cloud computing, in: International Conference on Grid and Pervasive Computing, Springer, 2009, pp. 25–35.
[3] T. Critchley, High Availability IT Services, CRC Press, 2014.
[4] B. Schroeder, G. Gibson, A large-scale study of failures in high-performance computing systems, IEEE Trans. Dependable Secure Comput. 7 (4) (2010) 337–350.
[5] S. Fu, Failure-aware resource management for high-availability computing clusters with distributed virtual machines, J. Parallel Distrib. Comput. 70 (4) (2010) 384–393.
[6] P. Garraghan, I.S. Moreno, P. Townend, J. Xu, An analysis of failure-related energy waste in a large-scale cloud environment, IEEE Trans. Emerg. Top. Comput. 2 (2) (2014) 166–180.
[7] A.M.S. Sampaio, Energy-Efficient and SLA-Based Management of IAAS Cloud Data Centers, 2015.
[8] I.P. Egwutuoha, D. Levy, B. Selic, S. Chen, A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems, J. Supercomput. 65 (3) (2013) 1302–1326.
[9] I. Koren, C.M. Krishna, Fault-Tolerant Systems, Morgan Kaufmann, 2010.
[10] I. Philp, Software failures and the road to a petaflop machine, HPCRI: 1st Workshop on High Performance Computing Reliability Issues, in Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA-11) (2005).
[11] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig, et al., The international exascale software project roadmap, Int. J. High Perform. Comput. Appl. 25 (1) (2011) 3–60.
[12] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, M. Snir, Toward exascale resilience, Int. J. High Perform. Comput. Appl. 23 (4) (2009) 374–388.
[13] E. Meneses, O. Sarood, L.V. Kalé, Assessing energy efficiency of fault tolerance protocols for HPC systems, in: 2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), IEEE, 2012, pp. 35–42.
[14] G. Aupy, A. Benoit, M.E.M. Diouri, O. Glück, L. Lefèvre, Energy-aware checkpointing strategies, in: Fault-Tolerance Techniques for High-Performance Computing, Springer, 2015, pp. 279–317.
[15] I.S. Moreno, P. Garraghan, P. Townend, J. Xu, Analysis, modeling and simulation of workload patterns in a large-scale utility cloud, IEEE Trans. Cloud Comput. 2 (2) (2014) 208–221.
[16] A.M. Sampaio, J.G. Barbosa, A comparative cost study of fault-tolerant techniques for availability on the cloud, in: International Symposium on Ambient Intelligence, Springer, 2017, pp. 263–268.
[17] A.M. Sampaio, J.G. Barbosa, Towards high-available and energy-efficient virtual computing environments in the cloud, Future Gener. Comput. Syst. 40 (2014) 30–43.
[18] K. Ferreira, J. Stearley, J.H. Laros, R. Oldfield, K. Pedretti, R. Brightwell, R. Riesen, P.G. Bridges, D. Arnold, Evaluating the viability of process replication reliability for exascale systems, in: 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), IEEE, 2011, pp. 1–12.
[19] H. Casanova, F. Vivien, D. Zaidouni, Using replication for resilience on exascale systems, in: Fault-Tolerance Techniques for High-Performance Computing, Springer, 2015, pp. 229–278.
[20] B. Mills, T. Znati, R. Melhem, Shadow computing: an energy-aware fault tolerant computing model, in: 2014 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2014, pp. 73–77.
[21] J. Liu, S. Wang, A. Zhou, S. Kumar, F. Yang, R. Buyya, Using proactive fault-tolerance approach to enhance cloud service reliability, IEEE Trans. Cloud Comput. (2016).
[22] A. Ganesh, M. Sandhya, S. Shankar, A study on fault tolerance methods in cloud computing, in: Advance Computing Conference (IACC), 2014 IEEE International, IEEE, 2014, pp. 844–849.
[23] S.S. Sathya, K.S. Babu, Survey of fault tolerant techniques for grid, Comput. Sci. Rev. 4 (2) (2010) 101–120.
[24] M.N. Cheraghlou, A. Khadem-Zadeh, M. Haghparast, A survey of fault tolerance architecture in cloud computing, J. Netw. Comput. Appl. 61 (2016) 81–92.
[25] S.-Y. Jing, S. Ali, K. She, Y. Zhong, State-of-the-art research study for green cloud computing, J. Supercomput. (2013) 1–24.
[26] A.B. Nagarajan, F. Mueller, C. Engelmann, S.L. Scott, Proactive fault tolerance for HPC with XEN virtualization, in: Proceedings of the 21st Annual International Conference on Supercomputing, ACM, 2007, pp. 23–32.
[27] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768.
[28] X. Zhang, J.-J. Lu, X. Qin, X.-N. Zhao, A high-level energy consumption model for heterogeneous data centers, Simul. Model. Pract. Theory 39 (2013) 41–55.
[29] S. Fu, C.-Z. Xu, Exploring event correlation for failure prediction in coalitions of clusters, in: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, ACM, 2007, p. 41.
[30] M. Stillwell, F. Vivien, H. Casanova, Dynamic fractional resource scheduling versus batch scheduling, IEEE Trans. Parallel Distrib. Syst. 23 (3) (2012) 521–529.
[31] E. Feller, L. Rilling, C. Morin, R. Lottiaux, D. Leprince, Snooze: a scalable, fault-tolerant and distributed consolidation manager for large-scale clusters, in: Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, IEEE Computer Society, 2010, pp. 125–132.

Please cite this article in press as: A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustain. Comput.: Inform. Syst. (2017), https://doi.org/10.1016/j.suscom.2017.11.006