Hybrid Cloud Resource Provisioning Policy in the Presence of Resource Failures

Bahman Javadi
School of Computing, Eng. and Math.
University of Western Sydney, Australia
Email: b.javadi@uws.edu.au

Jemal Abawajy
School of Information Technology
Deakin University, Australia
Email: jemal@deakin.edu.au

Richard O. Sinnott
Dept. of Computing and Information Sys.
The University of Melbourne, Australia
Email: rsinnott@unimelb.edu.au

***Abstract*—Resource provisioning is an important and challenging problem in the large-scale distributed systems such as Cloud computing environments. Resource management issues such as Quality of Service (QoS) further exacerbate the resource provisioning problem. Furthermore, with the increasing functionality and complexity of Cloud computing, resource failures are inevitable. Therefore, the question we address in this paper is how to provision resources to applications in the presence of resource failures in a hybrid Cloud computing environment. To this end, we propose three Cloud resource provisioning policies where we utilize workflow applications to drive the system workload. The proposed strategies take into account the workload model and the failure correlations to redirect requests to appropriate Cloud providers. Using real failure traces and workload models, we evaluated the performance and monetary cost of the proposed policies. The results of our experiments show that we can decrease the deadline violation rate of users' requests to as low as 20% with a limited cost on Amazon public Cloud.**

***Keywords*-Resource provisioning; Hybrid Cloud Computing; Failure-prone systems; Deadline;**

I. Introduction

Cloud computing platforms provide easy access to a wide range of IT resources such as raw computing and storage in the form of Virtual Machines (VMs). Cloud computing platforms provide massive scalability, high reliability, and high performance to end users while at the same time hiding the complexity of managing an IT infrastructure from end users. One particular type of Cloud computing, which is known as Infrastructure-as-a-Service (IaaS), allows users to customize and configure the Cloud resources based on their application demands.

Utilizing public Cloud services along with local resources (e.g., private Cloud) to support *Hybrid Clouds* [34], is one of the most widely used Cloud computing models. Hybrid Cloud platforms help scientists and businesses leverage the scalability and cost effectiveness of the public Cloud by paying only for IT resources consumed (server, connectivity, storage) while delivering the levels of performance and control available in private Cloud environments without changing their underlying IT setup. However, efficient policies to integrate public and private Cloud to assure QoS target of the users remain a challenge. These policies are further complicated when facing resource *failures* in the system.

In this paper, we propose three brokering strategies that take into account a workload model and resource failure correlations when selecting specific Cloud providers. Several papers have presented the adoption of public Cloud platforms and services by the scientific community. Most of these work demonstrate the performance and monetary cost-benefits for scientific applications [5], [6], [21], [30]. Recently, Assunção *et al.* [5] have investigated how an organization using a local cluster can utilize Cloud resources to improve the performance of its users' requests. The existing strategies take into account neither the workload type nor the resource failure to make decision about redirection of requests.

In this paper, we consider resource failure that refers to any anomaly caused by hardware or software faults that lead to unavailability in service. Also, we assume that the policies take advantage of a *knowledge-free approach*, i.e., they do not need any information about the failure model. Our policies are proposed in the context of the Australian Urban Research Infrastructure Network (AURIN)[1] project, which is an initiative aiming to develop an e-Infrastructure supporting research in the urban and built environment domain [33]. It will deliver a *lab in a browser* infrastructure providing federated access to heterogeneous data sources and facilitate data analysis and visualization in a collaborative environment to support multiple urban research activities. We consider the workflow applications in the AURIN project as a workload with the parallel jobs requesting for resources in the form of VMs. Moreover, we assume that end-user QoS requirements are defined as the *deadline*.

The rest of the paper is organized as follows. In Section II, we present the system architecture of interest's and its implementation. Then, the provisioning policies which include brokering strategies as well as scheduling algorithms are proposed in Section III. The performance evaluation of the proposed policies is presented in Section IV. Related work is described in Section V. Conclusions and future directions are presented in Section VI.

II. System Overview

In this section, we briefly overview the architecture and implementation of the AURIN project as well as the hybrid Cloud system that is used in this paper.

[1] http://aurin.org.au/

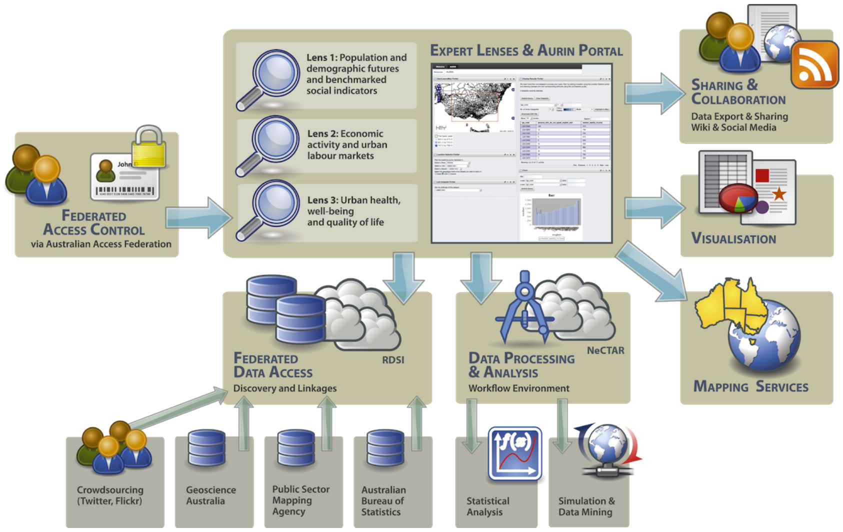


Fig. 1. The AURIN architecture.

A. The AURIN Architecture

The AURIN project is tasked with developing an e-Infrastructure through which a wide range of urban and built environment research activities will be supported [33]. The AURIN technical architecture approach is based on the concept of a single sign-on point of entry portal[2] (Figure 1). The sign-on capability is implemented through the integration of the Australian Access Federation (AAF)[3], which provides the backbone for the Internet2 Shibboleth-enabled[4] decentralized identity provision (authentication) across the Australian university sector. The portal facilitates access to a diverse set of data interaction capabilities implemented as JSR-286 compliant portlets. The portlets represent the user interface component of the capabilities integrated within a loosely coupled service-oriented architecture, exposing data search and discovery, filtering and analytical capabilities, coupled with a mapping service, and various visualization capabilities.

A rich library of local (e.g., Java) and federated (REST or SOAP services) analytical tools are exposed through the workflow environment based on the Object Modeling System (OMS) framework [19]. These analytical processes allow for advanced statistical analysis of spatial and aspatial data, and also expose complex modeling environments to urban researchers. The users in AURIN are able to compose different types of workflows that might themselves be compute/data intensive. Using the OMS framework as the basis for the AURIN workflow environment allows to enact user workflows on both Cluster and Cloud computing environments [19].

B. The Hybrid Cloud System

Hybrid Cloud [34] enables organizations that wants to supply their users' with computing capacity from a public Cloud provider when the local resources (i.e., private Cloud) fail short to meet the needs of the users. Figure 2 shows the system architecture used in this paper. In the architecture, we used the InterGrid components designed and implemented in the Cloudbus[5] research group [7] for its realization. The core component for this architecture is the *broker* that selects suitable resource providers offering computing services for incoming requests. This broker is realized as an InterGrid Gateway (IGG), which allows various types of resource manager to interconnect. Currently, it is possible to connect to the OpenNebula [11] or Eucalyptus [27] to manage the local resources. In addition, two interfaces to connect to a grid middleware (i.e., Grid'5000) and IaaS providers (i.e., Amazon's EC2 [1] and the NeCTAR Research Cloud [10]) have been developed.

C. Workload Model

In AURIN different users run a range of simulations and access and use data in a variety of scenarios that can be arbitrarily complex [19], [33]. Generally, it is the case that different jobs will require potentially large number of resources over a short period of time. As such, we model workflows as a job with several tasks that are sensitive to communication networks and resource failures. The AURIN workflows defined thus far are *tightly coupled*, i.e., where the simulations run on a single Cloud environment (and not concurrently across Cloud providers). As such, we assume that jobs are allocated to virtualized resources from a single provider.

A user enacts a workflow by submitting a request to the broker in the form of a request for VMs (see Figure 2). Each user's request has the following characteristics: type of virtual machine; number of virtual machines; estimated duration of the request; deadline for the request. When such a request arrives at the broker, it determines which resource provider to use. In the following section, we propose different provisioning policies to be utilized by the broker to handle such user's requests.

[2] http://portal.aurin.org.au
[3] http://www.aaf.edu.au/
[4] http://shibboleth.internet2.edu/
[5] http://www.cloudbus.org

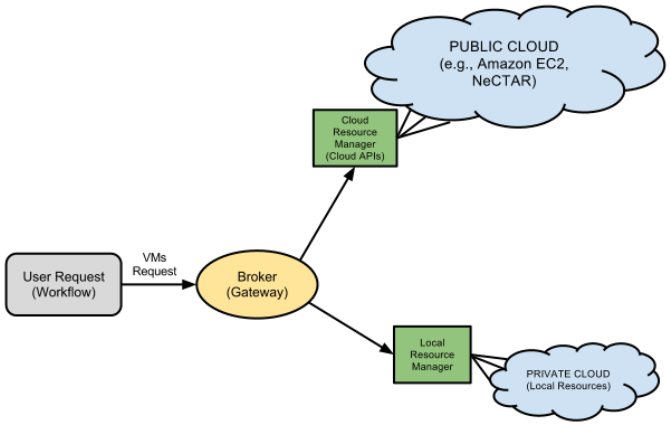


Fig. 2. The hybrid Cloud architecture

III. The Proposed Resource Provisioning Policies

In this section, we propose the resource provisioning policies that include the scheduling algorithms of the local resources (private Cloud) and the public Cloud as well as brokering strategies to share the incoming load with the public Cloud providers. The public Cloud providers adopt carefully engineered modules that include redundant components to cope with resource failures [15]. We assume that this design style is too expensive to consider for private Clouds which make them less reliable as compared to public Clouds. Thus, we concentrate on resource failures of private Clouds.

The proposed policies are part of the broker (i.e., IGG) as described in Section II-B. The proposed strategies are based on the workload model as well as the failure correlation and take advantage of knowledge-free approach, so they do not need any information about the failure model. As a result, the implementation of these policies in the AURIN environment is simplified.

A. User Request

In the system under study, each request can be thought of as a rectangle whose length is the user requested duration (T) and the width is the number of required VMs (S). Since the resources in the private Cloud are failure-prone, we would have some failure events (E_i) in the nodes while a request is getting serviced. In addition, it has been shown that there are *spatial* and *temporal* correlations in the failure events as well as dependency of workload type and intensity on the failure rate [12], [13], [36]. Spatial correlation means multiple failures occur on different nodes within a short time interval, while temporal correlation in failures refers to the skewness of the failure distribution over time. Therefore, we might have several *overlapped* failure events in the system.

Let $T_s(.)$ and $T_e(.)$ be the function that return the start and end time of an failure event. Let H be the sequence of overlapped failure events ordered according to increasing start time that is,

$$H = \{F_i \mid F_i = (E_1, ..., E_n), T_s(E_{i+1}) \leq T_e(E_i)\} \qquad (1)$$

where $1 \leq i \leq n-1$. Since we are dealing with the tightly coupled workflows, all VMs must be available for the whole requested duration, so any failure event in any VM would stop the execution of the whole request. In this case, the downtime of the service would be as follows:

$$D = \sum_{\forall F_i \in H} (max\{T_e(F_i)\} - min\{T_s(F_i)\}) \qquad (2)$$

The above analyses reveal that even in the presence of an *optimal* fault-tolerant mechanism (e.g., perfect checkpointing) in the private Cloud, a given request is faced with D time units delay which may consequently breach the request's deadline. In other words, if the request has been stalled for the duration of overlapping failures, a long delay may arise and cause service unavailability. This is the justification of utilizing highly reliable services from a public IaaS Cloud provider. To deal width these failure properties, we proposed three different strategies which are based on the workload model for the general failure events.

B. Size-based Strategy

Several studies have explored the spatial correlation in failure events in distributed systems [12], [13], i.e., where multiple failures occur on different nodes within a short time interval. This property can be very detrimental where each request needs all VMs to be available for the whole required duration. Moreover, as mentioned in Equation (2), the downtime is strongly dependent on the number of requested VMs. Therefore, the more VMs requested, the more likely they jobs will be affected by simultaneously failures.

To cope with this situation, we proposed a redirecting strategy that sends wider requests with larger S to more reliable public Cloud systems, while minimizing requests sent to potentially more failure-prone local resources. A key element of this strategy is the mean number of VMs required per request.

To determine the mean number of VMs per request, we need the probability of different numbers of VMs in incoming requests. Assume that P_1 and P_2 are probabilities of request with one VM and power of two VMs in the workload, respectively. So, the mean number of virtual machines required by requests is given as follows:

$$\overline{S} = P_1 + 2^{\lceil k \rceil}(P_2) + 2^k\left(1 - (P_1 + P_2)\right) \qquad (3)$$

Based on the parallel workload models, the size of each request follows a two-stage uniform distribution with parameters (l, m, h, q) [22], [23]. This distribution consists of two uniform distributions where the first distribution would be in the interval of $[l, m]$ with probability of q and the second one with the probability of $1-q$ would be in the interval of $[m, h]$. So, m is the middle point of possible values between l and h. Hence, k can be found as the mean value of this distribution as follows:

$$k = \frac{ql + m + (1-q)h}{2} \qquad (4)$$

The redirection strategy submits requests to the public Cloud provider if the number of requested VMs is greater than $\overline{S}$, otherwise the request will be serviced using resources in the private Cloud.

C. Time-based Strategy

In addition to spatial correlation, failure events are often correlated in the time domain which can result in a skewing of the failure distribution over time [12]. As a result, the failure rate is time-dependent and some periodic failure patterns can be observed in different time-scales [36]. The longer requests mainly affected by this temporal correlation since these requests stay longer in the system and are likely to have more failures. So, again there is a strong relation between down time and the request duration. On the other hand, in the real distributed systems, request duration (job runtime) are *long tailed* [8], [28]. This means that a very small fraction of all requests are responsible for the main part of the load.

The time-based strategy offers an efficient technique to solve both above mentioned properties. We use the mean request duration as the decision point for the gateway to redirect the incoming requests to the Cloud providers. In other words, if the request duration is less than or equal to the mean request duration, the request will be run using local resources. By this technique, the majority of short requests could meet the workload deadline as they are less likely to encounter failures. Moreover, longer requests will be served by more reliable public Cloud provider resources where they can meet the deadline under enhanced resource availability.

The mean request duration can be obtained from the fitted distribution of the workload model. For instance, the request duration of the parallel workloads in DAS-2 multi-cluster system is the Lognormal distribution with parameters μ, σ, so the mean value is given as follows [22]:

$$\overline{T} = e^{\mu + \frac{\sigma^2}{2}} \tag{5}$$

The redirection strategy submits requests to the public Cloud provider if the duration of request is greater than $\overline{T}$, otherwise will be serviced using local resources. Another advantage of this strategy is better utilization of the Cloud resources. For example with Amazon's EC2, if a request uses a VM for less than one hour, the cost of one hour must be paid. So, when we redirect long requests to the public Cloud provider, this waste of money will be minimized.

D. Area-based Strategy

The two aforementioned strategies are based on the only one aspect of the request: number of VMs (S) or duration (T). A third strategy aimed at making a compromise between the size-based and time-based strategy. Here we utilize the area of a request as the decision point for the gateway - this is given as the rectangle with length T and width S. Using this model, we are able to calculate the mean request area by multiplying the mean number of VMs by the mean request duration, as follows:

$$\overline{A} = \overline{T} \cdot \overline{S} \tag{6}$$

The redirection strategy submits requests to the public Cloud provider if the area of the request is greater than $\overline{A}$, otherwise these requests will be serviced using local resources. This strategy sends long *and* wide requests to the public Cloud, so it would be more conservative than a size-based strategy and less conservative than a time-based strategy.

E. Scheduling Algorithms

As described before, a resource provisioning policy consists of a brokering strategy as well as an algorithm for scheduling the requests across private and public Cloud resources. Based on this, we utilize two well-know algorithms: *conservative* [26] and *selective* backfilling [35]. With conservative backfilling, each request is scheduled when it is submitted to the system, and requests are allowed to leap forward in the queue if they do not delay other queued requests. Selective backfilling grants reservations to a request when its expected slowdown exceeds a threshold, i.e., where the request has waited long enough in the queue. The expected slowdown of a given request is also called *eXpansion Factor* and can be given as $XFactor = \frac{W_i + T_i}{T_i}$, where W_i and T_i are the waiting time and the run time of request i, respectively. We use the Selective-Differential-Adaptive scheme proposed in [35], which lets the XFactor threshold be the average slowdown of previously completed requests.

After submitting requests to the scheduler, each VM runs on one available node. In the case of resource failure during the execution, we assume checkpointing so that the application is started from where it left off when the node becomes available again. The checkpointing issues are not in the scope of this research and interested readers should refer to [3] to see how checkpoint overheads and periods can be computed base on the associated failure model.

IV. Performance Evaluation

In order to evaluate the performance of proposed policies, we implemented a discrete event simulator using CloudSim [4]. We used simulation as experiments are reproducible and the cost of conducting experiments on real public Cloud resources would be prohibitively expensive.

The performance metrics which are considered in all simulation scenarios are the violation rate and the bounded slowdown [9]. The violation rate is the fraction of requests that do not meet a given deadline. The bounded slowdown for the M requests is defined as:

$$Slowdown = \frac{1}{M} \sum_{i=1}^{M} \frac{W_i + max(T_i, bound)}{max(T_i, bound)} \tag{7}$$

where $bound$ is set to 10 seconds to eliminate the effect of very short requests [9].

To compute the cost of using resources from a public Cloud, we use the amounts charged by Amazon to run basic virtual machines and network usage at EC2. The cost of using EC2 for policy pl can be calculated as follows:

$$Cost_{pl} = (H_{pl} + M_{pl} \cdot H_u)\, C_n + (M_{pl} \cdot B_{in})\, C_x \tag{8}$$

TABLE I
INPUT PARAMETERS FOR THE WORKLOAD MODEL

Input Parameters	Distribution/Value
Inter-arrival time	Weibull ($\alpha = 23.375, 0.2 \leq \beta \leq 0.3$)
No. of VMs	Loguniform ($l = 0.8, m, h = log_2 N_s, q = 0.9$)
Request duration	Lognormal ($2.5 \leq \mu \leq 3.5, \sigma = 1.7$)
P_1	0.02
P_2	0.78

where H_{pl} is the Cloud usage per hour for the policy pl. That means, if a request uses a VM for 45 minutes for example, the cost of one hour is considered. M_{pl} is the fraction of requests which are redirected to the public Cloud. H_u is the startup time for initialization of operating systems on a VM which is set to 80 seconds [29]. We take into account this value as Amazon commences charging users when the VM process starts. B_{in} is the amount of data which needs to be transferred to the Amazon EC2 for each request. This is set to 100 MB per request. The cost of one specific instance on the EC2 is determined as C_n and considered as 0.085 USD per VM per hour for a small instance. The cost of data transfer to the Amazon EC2 is also considered as C_x which is 0.1 USD per GB. It should be noted that we consider a case where requests' output are very small and can be transferred to the local resources for free [1].

A. Experimental Setup

Considering workflow applications in the AURIN project as the parallel applications, we used the parallel job model of the DAS-2 system which is a multi-cluster Grid [22] as the workload model for evaluation scenarios. Based on the workload characterization, the inter-arrival time (based on Weibull distribution), the request size (based on Loguniform distribution), and the request duration (based on Lognormal distribution). These distributions with their parameters are listed in Table I.

For each simulation experiment, statistics were gathered for a two-month long period of DAS-2 workloads. The first week of workloads during the *warm-up* phase were ignored to avoid bias before the system reached a steady-state. Each data point represents the average of 30 simulation rounds. The number of resources in the private and the public Cloud were set equally to $N_s = N_c = 64$ with a homogeneous computing speed of 1000 MIPS[6]. The time to transfer the application (e.g., configuration file or input file(s)) for the private Cloud is negligible as the local resources are interconnected by a high-speed network, so $L_s = 0$. However, to execute the application on the public Cloud we must send the configuration file as well as input file(s). Given this, we consider a network transfer time of $L_c = 80$ sec., which is the time to transfer of 100 MB data on a 10 Mbps network connection.

The failure trace for the experiments is obtained from the Failure Trace Archive [20]. We used the failure trace of a cluster in the Grid'5000 with 64 nodes for a duration of 18 months, which includes on average 800 events per node. The average availability and unavailability time in this trace archive is 22.26 hours and 10.22 hours respectively.

In order to generate different workloads, we systematically modified three parameters of the workload model. To change the inter-arrival time, we modified the second parameter of the Weibull distribution (the *shape* parameter β) as shown in Table I. To have requests with different durations, we changed the first parameter of the Lognormal distribution between 2.5 and 3.5 as described in Table I. Moreover, we also varied the middle point of the Loguniform distribution (i.e., m) to generate workloads with different number of VMs per request where $m = h - \omega$ and ω is between 1.5 to 3.3. Thus the larger value of ω, the fewer VMs required to service requests. The same scheduling algorithms are used for the private and public Cloud providers in all scenarios.

To generate the request deadlines we utilize the same techniques given in [18], which provide a feasible schedule for jobs. To obtain deadlines, we conducted experiments based on scheduling requests on local resources without failure events using aggressive backfilling. We used the following equations to calculate the deadline for each request i:

$$d_i = \begin{cases} st_i + (f \cdot\ ta_i), & \text{if } [st_i + (f \cdot\ ta_i)] < ct_i \\ ct_i, & \text{otherwise} \end{cases} \tag{9}$$

where st_i is the request's submission time, ct_i is its completion time, ta_i is the request's turn around time (i.e., $ct_i - st_i$). We define f as a stringency factor that indicates how urgent deadlines are. If $f = 1$, then the request's deadline is uses an aggressive backfilling scenario to ensure completion. We evaluate strategies with different stringency factors, however only report results where $f = 1.3$ (i.e., a normal deadline).

B. Results and discussions

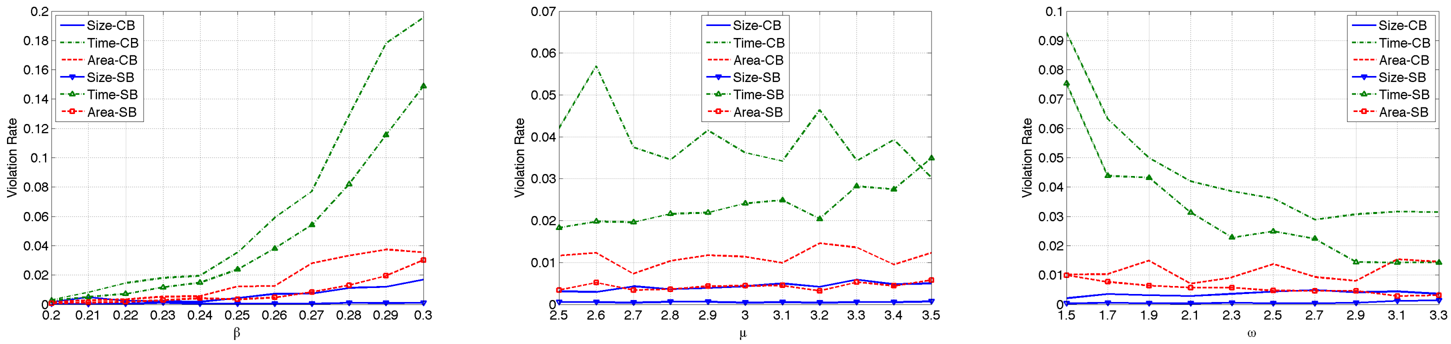
The results of simulating violation rates versus different workloads are depicted in Figure 3 for different provisioning policies. In each figure, three brokering strategies are plotted for a scheduling algorithm. In all the figures, Size, Time, and Area refer to size-based, time-based and area-based brokering strategies, respectively. Moreover, CB and SB stand for Conservative and Selective Backfilling, respectively.

Based on Figure 3, by increasing the workload intensity (i.e., arrival rate, duration or size[7] of requests), we observe an increase in the violation rate for all provisioning policies. As illustrated in this figure, the size-based brokering strategy yields a very low violation rate where the area-based strategy also shows a comparable performance. The time-based strategy has the worst performance in terms of violation rate especially when the workload intensity increases.

It is worth noting that the violation rate of size-based brokering strategy, in contrast to others, has an inverse relation with the request size, i.e., we observe an increase in the number of fulfilled deadlines by reducing the size of requests. This behavior is due to increasing the number of redirected requests

[6]This assumption is made just to focus on performance degradation due to failure.

[7]The larger value of ω, the fewer VMs required to service requests.



(a) Request arrival rate

(b) Requset duration

(c) Request size

Fig. 3. Violation rate for all provisioning policies versus different workloads.

to the failure-prone private Cloud in the size-based brokering strategy. Additionally, in all experiments, the policies using a selective backfilling scheduler have a better performance than the correspondent experiments using conservative backfilling.

Figure 4 expresses the slowdown of requests for all provisioning policies versus different workloads with the same configuration as per the previous experiments. As illustrated in Figure 4(a), increasing the request arrival rate results in a slowdown with size-based and area-based strategies, both of which have a more gradual slope than the time-based strategy. Moreover, the slowdown versus request duration has been plotted in Figure 4(b) reveals that this slowdown is not very sensitive to the request duration given the almost non-variant lines. Based on the results in Figure 4(c), slowdown decreases by decreasing the request size (number of VMs per request) for the time-based and the area-based strategies. However, the size-based strategy has gradually increased slowdown due to increasing the number of redirected requests to the more failure-prone private Cloud. Nevertheless, the size-based strategy has the better slowdown in the most cases with respect to other strategies for different workload types.

Figure 5 shows the amount of money spent on the EC2 per month to respond to the incoming requests with different workload types. It is worth noting that Cloud usage is independent of the scheduling algorithm as depicted in these figures. As observed in all workload types, the size-based strategy utilizes more Cloud resources than other strategies, and this is the reason for lower violation rates and slowdown as described previously. Moreover, the time-based strategy has the lowest Cloud cost on the EC2 while the area-based strategy always incurs the cost between the size-based and time-based strategies.

Since the proposed strategies have different cost and performance, selecting a suitable policy for a real case is strongly dependent on many issues such as desired level of QoS as well as budget constraints. For instance, by using the time-based brokering policy (CB or SB), we can decrease the deadline violation rate of users' requests to as low as 20% with the less than 1200 USD on Amazon public Cloud (see Figures 3(a) and 5(a)).

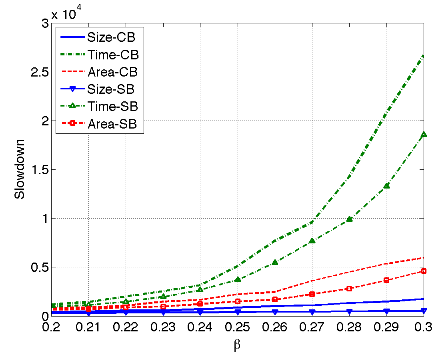
V. Related Work

Related work can be classified into two groups: load sharing in the distributed systems and solutions utilizing Cloud computing resources to extend the capacity of existing infrastructures. Several load sharing mechanisms have been proposed for different types of distributed systems. Iosup *et al.* [17] proposed a matchmaking mechanism for enabling resource sharing across computational Grids. Balazinska *et al.* [2] investigated a mechanism for migrating stream processing operators in a federated distributed system. We address resource provisioning using leased resources from a Cloud provider to improve the users' and organisations' QoS in the presence of failure.

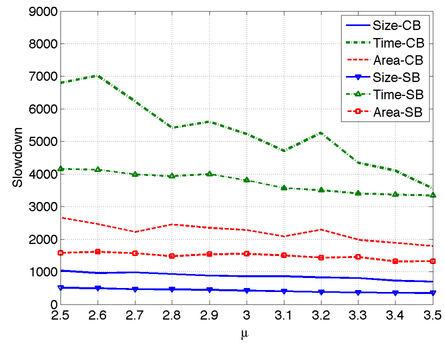
VioCluster [32] is a system in which a broker is responsible for dynamically managing a virtual domain by borrowing and lending machines between clusters. Montero *et al.* [31] also used GridWay to deploy virtual machines on a Globus Grid; They also proposed GridGateWay [16] as a means to support interoperability of Globus Grids. In [34], the authors developed virtual infrastructure management through two open source projects: OpenNebula and Haizea[8]. In contrast to this, we adopted the InterGrid environment that is based on the virtual machine technology and can be connected to any distributed systems through a Virtual Machine Manager (VMM) [7]. Moreover, we consider a new type of platform which is commonly called Hybrid Cloud and propose three brokering strategies which are part of InterGrid Gateway (IGG) and allow to utilize public Cloud resources.

The applicability of public Cloud services for Grid computing has been demonstrated in existing work. In [30], the authors consider the Amazon data storage service S3 for scientific data-intensive applications. They conclude that monetary costs are as high as the collective costs for storage

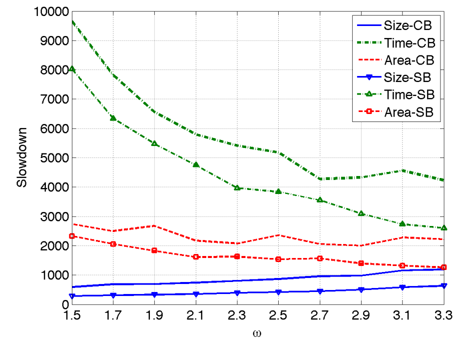
[8]http://haizea.cs.uchicago.edu



(a) Request arrival rate

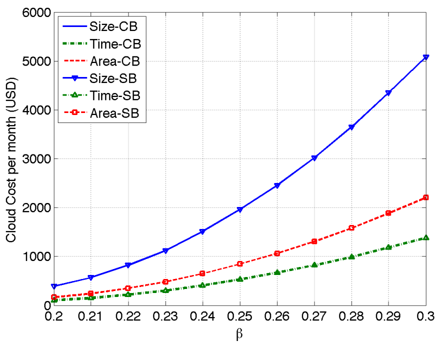


(b) Request duration

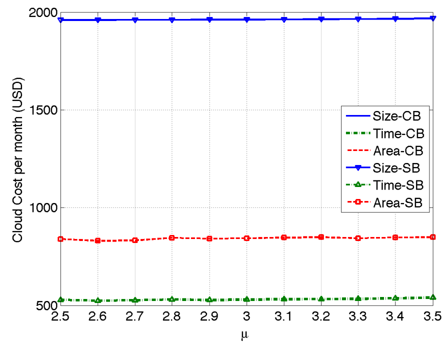


(c) Request size

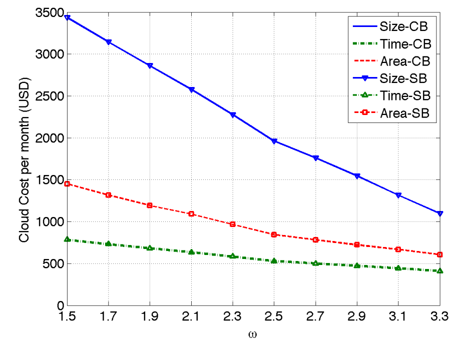
Fig. 4. Slowdown for all provisioning policies versus different workloads.



(a) Request arrival rate



(b) Requset duration



(c) Request size

Fig. 5. Cloud cost on EC2 per month for all provisioning policies versus different workloads.

service groups such as availability, durability, and access performance. In contrast, data-intensive applications often do not need all of these features. In [14], the author conducts a general cost-benefit analysis of Clouds. However, no specific type of scientific application is considered. In [6], the authors determine the cost of running a scientific workflow over a Cloud. They found that the computational costs outweighed storage costs for their Montage application. Kondo *et al.* [21] also provided a cost-benefit analysis of Cloud computing versus desktop grids for compute-intensive tasks. In our work, in contrast to others, we consider workload based on user' requests which can be data or compute intensive workflows.

In [24], the authors developed a model of an *Elastic Site* that utilized services provided by a site, to take advantage of elastically provisioned resources in a public Cloud. The authors in [5] investigated whether an organization using a local cluster could benefit from using Cloud providers to improve the performance of their user' requests. The authors in [25] utilized gang scheduling to dispatch parallel jobs to a cluster of VMs hosted on Amazon EC2. In this paper, we take into account the workload model and failure correlation to borrow the public Cloud resources. Moreover, we evaluate the performance of the system under realistic workloads and failure traces.

VI. Conclusions

In this paper, we considered the problem of QoS-based resource provisioning in a hybrid Cloud computing system where private Clouds can potentially be failure-prone. We proposed a variety of brokering strategies based upon a hybrid Cloud model where an organization that operates its own private Cloud aims to improve the QoS for its user' requests by utilizing public Cloud resources. Various brokering strategies which adopt the workload model and take into account the failure correlations were described and presented.

The proposed policies take advantage of a context/knowledge-free approach in that they do not need any statistical information about the failure model of the local resources in the private Cloud. We evaluated the proposed policies and considered their impact on different performance metrics such as deadline violation rate and slowdown. Experimental results under realistic workload and failure events, reveal that we are able to adopt user the workload model to provide flexibility in the choice of strategy based on the desired level of QoS, the needed performance, and the available budget.

In future work, we intend to use a set of real workflow applications from the AURIN project and run real experiments. For this purpose, we shall investigate different checkpointing mechanisms augmenting our analysis and implementation.

References

[1] Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2.

[2] M. Balazinska, H. Balakrishnan, and M. Stonebraker. Contract-based load management in federated distributed systems. In *1st Symposium on Networked Systems Design and Implementation (NSDI)*, pages 197–210, San Francisco, USA, March 2004.

[3] M. Bouguerra, T. Gautier, D. Trystram, and J.-M. Vincent. A flexible checkpoint/restart model in distributed systems. In *9th International Conference on Parallel Processing and Applied Mathematics*, volume 6067 of *LNCS*, pages 206–215. Springer, 2010.

[4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[5] M. D. de Assunção, A. di Costanzo, and R. Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *18th International Symposium on High Performance Parallel and Distributed Computing (HPDC 2009)*, pages 141–150, Garching, Germany, June 2009.

[6] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the Cloud: The montage example. In *19th ACM/IEEE InternationalConference on Supercomputing (SC 2008)*, pages 1–12, Piscataway, NJ, USA, 2008.

[7] A. di Costanzo, M. D. de Assunção, and R. Buyya. Harnessing cloud technologies for a virtualized distributed computing infrastructure. *IEEE Internet Computing*, 13(5):24–33, 2009.

[8] D. G. Feitelson. *Workload Modeling for Computer Systems Performance Evaluation*. 2009.

[9] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong. Theory and practice in parallel job scheduling. In *3rd Job Scheduling Strategies for Parallel Processing (IPPS'97)*, pages 1–34, London, UK, 1997.

[10] T. Fifield. NeCTAR research Cloud node implementation plan. Research Report Draft-2.5, Melbourne eResearch Group, The University of Melbourne, October 2011.

[11] J. Fontán, T. Vázquez, L. Gonzalez, R. S. Montero, and I. M. Llorente. OpenNEbula: The open source virtual machine manager for cluster computing. In *Open Source Grid and Cluster Software Conference – Book of Abstracts*, San Francisco, USA, May 2008.

[12] S. Fu and C.-Z. Xu. Quantifying event correlations for proactive failure management in networked computing systems. *Journal of Parallel and Distributed Computing*, 70:1100–1109, November 2010.

[13] M. Gallet, N. Yigitbasi, B. Javadi, D. Kondo, A. Iosup, and D. Epema. A model for space-correlated failures in large-scale distributed systems. In *Euro-Par 2010*, volume 6271 of *Lecture Notes in Computer Science*, pages 88–100. 2010.

[14] S. Garfinkel. Commodity grid computing with Amazons S3 and EC2. In *USENIX LOGIN*, volume 32, pages 7–13, 2007.

[15] U. Hoelzle and L. A. Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, San Rafael, CA, 2009.

[16] E. Huedo, R. S. Montero, and I. M. Llorente. Grid architecture from a metascheduling perspective. *IEEE Computer*, 43(7):51 –56, Jul. 2010.

[17] A. Iosup, D. H. J. Epema, T. Tannenbaum, M. Farrellee, and M. Livny. Inter-operating Grids through delegated matchmaking. In *18th ACM/IEEE Conference on Supercomputing (SC 2007)*, pages 1–12, New York, USA, November 2007.

[18] M. Islam, P. Balaji, P. Sadayappan, and D. K. Panda. QoPS: A QoS based scheme for parallel job scheduling. In *9th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '03)*, volume 2862 of *LNCS*, pages 252–268, Seattle, WA, USA, 2003.

[19] B. Javadi, M. Tomko, and R. O. Sinnott. Decentralized orchestration of data-centric workflows using the object modeling system. In *CCGRID*, pages 73–80, 2012.

[20] D. Kondo, B. Javadi, A. Iosup, and D. H. J. Epema. The Failure Trace Archive: Enabling comparative analysis of failures in diverse distributed systems. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID)*, pages 398–407, 2010.

[21] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson. Cost-benefit analysis of Cloud computing versus desktop grids. In *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2009)*, pages 1–12, Rome, Italy, 25-28 May 2009. IEEE Computer Society, Washington, DC.

[22] H. Li, D. Groep, and L. Wolters. Workload characteristics of a multi-cluster supercomputer. In *10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, pages 176–193, New York, USA, 2004.

[23] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, 2003.

[24] P. Marshall, K. Keahey, and T. Freeman. Elastic site: Using clouds to elastically extend site resources. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID)*, pages 43–52, 2010.

[25] I. Moschakis and H. Karatza. Evaluation of gang scheduling performance and cost in a cloud computing system. *The Journal of Supercomputing*, pages 1–18, 2010.

[26] A. W. Mu'alem and D. G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, 2001.

[27] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus open-source cloud-computing system. In *1st Workshop of Cloud Computing and Its Applications*, October 2008.

[28] L. F. Orleans and P. Furtado. Fair load-balancing on parallel systems for QoS. In *36th International Conference on Parallel Processing (ICPP 2007)*, pages 22 –22, 2007.

[29] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. A performance analysis of EC2 cloud computing services for scientific computing. In *1st International Conference on Cloud Computing (CloudComp 2009)*, pages 115–131, Munich, Germany, 2009.

[30] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel. Amazon S3 for science Grids: a viable solution? In *1st International Workshop on Data-aware Distributed Computing (DADC'08) in conjunction with HPDC 2008*, pages 55–64, New York, NY, USA, 2008.

[31] A. J. Rubio-Montero, E. Huedo, R. S. Montero, and I. M. Llorente. Management of virtual machines on Globus Grids using GridWay. In *21st IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007)*, pages 1–7, Long Beach, USA, March 2007.

[32] P. Ruth, P. McGachey, and D. Xu. VioCluster: Virtualization for dynamic computational domain. In *7th IEEE International Conference on Cluster Computing (Cluster 2005)*, pages 1–10, Burlington, USA, September 2005.

[33] R. O. Sinnott, G. Galang, M. Tomko, and R. Stimson. Towards an e-infrastructure for urban research across australia. In *eScience*, pages 295–302, 2011.

[34] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13(5):14 –22, Sep. 2009.

[35] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Selective reservation strategies for backfill job scheduling. In *8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '02)*, volume 2537 of *LNCS*, pages 55–71, London, UK, 2002.

[36] N. Yigitbasi, M. Gallet, D. Kondo, A. Iosup, and D. Epema. Analysis and modeling of time-correlated failures in large-scale distributed systems. In *8th IEEE/ACM International Conference on Grid Computing*, October 2010.