Fault-Tolerant Resource Provisioning with Deadline-Driven Optimization in Hybrid Clouds

Emmanuel Ahene
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan 611731, P. R. China

Kingsley Nketia Acheampong
School of Software Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan 611731, P. R. China

Heyang Xu
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan 611731, P. R. China

***Abstract*—Resource provisioning remains as one of the challenging research problems in cloud computing, more importantly when considered together with service reliability. Fault-tolerance techniques such as fault-recovery is one of the techniques that can be employed to improve on service reliability. Technically, fault-recovery has obvious impact on service performance. Such impact requires detailed studies. Only few works on hybrid cloud resource provisioning address fault recovery and its impact. In this paper, we investigate the problem of resource provisioning in hybrid Clouds, considering the probability of hybrid cloud resource failure during job execution. We formulate this problem as an optimization model with operational cost as an objective function subject to the deadline constraint of jobs. Based on our proposed optimization model, we design a heuristic-based algorithm called dynamic resource provisioning algorithm (DRPA). We then perform extensive experiments to evaluate performance of the proposed algorithm based on a real world set of data. The results confirm the obvious impact of fault recovery on the performance metrics (operational cost and deadline violation rate) and also confirms that DRPA can be useful in minimizing operational cost.**

***Keywords*—Deadline; fault recovery; hybrid Clouds; resource provisioning; software-as-a-service**

I. Introduction

Cloud computing is a promising computing paradigm which has attracted more research attention in both the academia and the industry [1]. Among its benefit, the Cloud enables its users to have access to a pool of configurable computing resources across the internet independently without reference to its underlying hosting infrastructure.

The resource pool in the Cloud are often deployed in any of the four well known models, namely: public, private, community and hybrid deployment models. In the hybrid Cloud model, the Cloud infrastructure is made up of a combination of the private and public Cloud infrastructure. Provisioning of configurable resources with a private Cloud is known for its main advantage of being more secured than its alternatives. This is because, data is controlled on servers that no other company has access to except its users. However, since the resources in private Clouds are limited, organizations (such as SaaS providers) are faced with the challenge of capacity limitation when there is a high demand for resource by users. Compared to purchasing additional physical servers or building a new datacenter, many of such organizations prefer to rather scale up their service capacity to the public Cloud so as to meet their users' need. The process where an organization would leverage both its private Cloud resource and the resource in the public Cloud to process its user's workload is called the hybrid Cloud resource provisioning. Currently, Open Text, a leading software provider in enterprise information management, employs the hybrid Cloud model to demonstrate their enterprise content management software. Moreover, one of the world's leading game companies, SEGA, has adopted hybrid Cloud to improve its development process [2].

Its worth noting that, Cloud applications tend to be inevitably useful in major business operations and so most users are always bent on having assurance from providers with respect to service delivery. These assurances are realized through Service Level Agreements (SLAs) established between the providers and users. In this way, providers express their commitment to deliver services to satisfy the user's requirement. To users, it is a warranty. However, to SaaS providers, it is a challenge to ensure efficient resource provisioning policies. Moreover, how to optimize the operational cost involved in the efficient provisioning of resources remains an important and challenging problem to SaaS providers especially, when there is a high workload of job requests with QoS constraints.

A. Our Contribution

Over the years, many researchers have done relevant studies on the problem of resource provisioning in both hybrid and public Cloud. However, one relevant facet of the problem that is rarely addressed is on the probability of resource failures. The tendency of resource failures during the execution of jobs cannot be overlooked because it has obvious impact on service performance. Such impact requires detailed studies. Our previous work [3] in addition to other works such as [4] [5] [6] happens to be part of the few works that address this issue. Fault-tolerance techniques such as fault recovery is one of the techniques that can be employed to improve reliability. It employs checkpoint and rollback/roll-forward scheme, that enables a resource to recover from an error and resume execution [14]. In this paper, we investigate the problem of resource provisioning in hybrid Clouds considering the probability of resource failure during job execution. We employ fault-recovery to address the problem of resource failure. Our contributions are as follows:

We present a hybrid Cloud model that enables the scalability of the resource-base of a software-as-a-service SaaS provider who intends to leverage resources in both private and public Cloud for job execution. We propose DRPA to address the problem of operational cost minimization associated with the leveraging of hybrid Cloud resources, considering the probability that resources (i.e.Virtual machines (VM) and communication links) may fail and recover. In addition, we take into account some practical issues such as communication cost, cost incurred at local (private cloud) and jobs that are dynamic in nature. Finally, we perform extensive experiments to evaluate the performance of the proposed algorithm in terms of operational cost and deadline violation rate.

The rest of the paper is organized into six sections. Section 2 presents related works whiles section 3 presents the system model and problem formulation. Section 4, presents the proposed DRPA, and its provisioning policies. In section 5, we discuss the experiments and their results against existing works. The paper is finally concluded in section 6.

II. Related Works

The optimal provisioning of the Cloud's configurable compute resources to meet certain predefined performance criteria is a complex problem. It has attracted much research attention [7], [8]. This problem comprise the steps involved in allocating suitable compute resources to tasks with the aim of optimizing certain objectives. Some widely known objective functions are; minimizing cost, minimizing the time of task completion and maximizing the utilization of resources.

To address the problem of operational cost minimization in Cloud resource provisioning context, authors in [9], proposed a new MapReduce Cloud service model called Cura, which automatically creates the best configuration of clusters for tasks so as to approach a global resource optimization. Specifically, the system employs a deadline-awareness method, which defers the execution of certain tasks, and necessitates global optimization with reduced cost. Wu et al. [10] proposed an SLA based resource allocation method, which is compatible to the heterogeneity of infrastructure and adaptable to the dynamic change of customer jobs. Their approach seeks to maximize the profit of the SaaS provider by minimizing the number of SLA violations and the cost by reusing VMs. Other related works [9], [11], [12], [13] and [14] also focused on cost-optimization strategies for resource provisioning. This paper is also focused on cost optimization and also consider workloads that are dynamic in nature. However, it is considered in the hybrid Clouds.

Unlike hybrid Clouds, extensive research work on resource provisioning has been conducted in public Clouds. Nevertheless, many industries have started using Hybrid Cloud for Cloud businesses [2]. In 2011, Tak et al. [15] did investigate the economic issues of the application deployment choice in the hybrid Cloud. The output of their research indicated the need to research on the solutions for various optimization problems associated with the hybrid SaaS concept. Subsequently, related works such as [16] and [17] focused on the design of cost efficient VM migration algorithms which enables the scale up of local data center resources to the public Cloud in the context of Cloud bursting. However, their works considers workload characteristics as static. In [18], a model that extends the physical site cluster with Cloud resources elastically was proposed to adapt to the dynamic demands of applications. The central component of this model is an elastic site manager that handles resource provisioning. In [19] Li et al. implements Lyapunov optimization techniques to develop an online dynamic provisioning algorithm in the hybrid Cloud setting. Their algorithm seeks to minimize the operational cost of a hybrid SaaS provider with a delay-aware optimization. In their work, they focus on how the SaaS provider can leverage three types of VMs from the public in addition to the VMs at local for job execution. However, their work does not consider the probability of VM and communication link failure.

In contrast, other works such as [5] [6] [3] and [4] do not overlook the probability of VM or communication link failure. In [4], Javadi et al. studied on a problem that is comparatively similar to ours, however, they consider the probability of failure in the private Cloud only. In addition, they do not consider failure in communication links as well. Although we address a similar problem in this paper, we focus on some practical issues such as communication cost, the cost incurred at local (private Cloud), jobs that are dynamic in nature, and the probability that a VM (in both private and public) and a communication link between the public and private Cloud may fail. Specifically, this work considers how the SaaS provider can leverage VM instances at local and the On-demand VM instances in the public Cloud to process the job request of its users. Unlike [4] and [19], this paper proposes a resource provisioning algorithm which relies on the job runtime estimate, the prices of VM instances, and the availability-state of rented/local resource to decide what are the best types of VM instances to run each job and when jobs should run in the hybrid Cloud setting.

III. System Model and Problem Formulation

A. System Model

In this paper, we assume that there is a SaaS provider who owns a private data center which comprise a finite number of local servers that implements virtualization concepts to run user jobs. The SaaS provider is assumed to have the ability to seamlessly scale up the capacity of its services by renting for On-demand VM instances from the public Cloud when there is a spike in resource demand. The provider can as well scale down the rented VM instances once they are not needed. Since the rendering of service with VMs at local (private cloud) and the renting of the public Cloud VMs is associated with much monetary cost, it is imperative for the SaaS provider to reduce such cost. The SaaS provider is therefore faced with the challenge of minimizing total operational cost without violating the deadline for job completion considering the probability of VM and communication link failure.

To address the challenge, the SaaS provider needs to decide on the number of VMs to rent and when to rent them. Emphatically, a job is first directed to the private Cloud but once the private VMs are busy, they are directed to a waiting queue. The SaaS provider decides on the number of VMs to rent when the waiting time of the job has exceeded a threshold delay value W_k. The decision of the SaaS provider is based on the runtime estimate of the job, the availability-state of

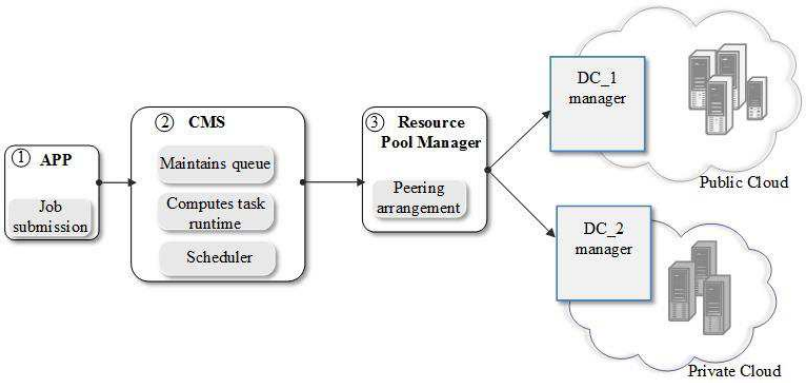


Fig. 1. Hybrid Cloud Architecture

the VMs and their associated prices. Technically, there is the probability of VM or communication link failure, however, since we adopt the fault recovery mechanism, it is assumed that all failures are recoverable. Given that all failures are recoverable, it is obvious that after some time (recovery time) VMs and communication links resumes execution [3]. The occurrence of failures, failure times, and recovery times are assumed to be mutually s-independent [5]. Table I shows a list of notations and their descriptions as used in this paper.

VM instances are assumed to be equivalent to the instance types available at Amazon EC2. Since there are diverse VM types in the public Clouds, we assume that the VM instances on public Cloud have similar capacity in terms of memory size and EC2 compute units (ECUs) as in the private Cloud [20]. However, for each type of similar instance, there exist an infinite number of them on the public Cloud whiles that of the private is limited. One ECU provides a processing capacity which is equivalent to a CPU capacity of 1.0 to 1.2 GHz 2007 AMD Opteron or 2007 Intel Xeon processor. In this paper, we consider VM instances with possible ECU values of $\{1, 8, 13, 16,$ and $26\}$ [21]. The EC2 types which are of exact ECU characteristics are m1.small (1 ECU), c4.large (8 ECUs), m4.large (13 ECUs), c4.xlarge (16 ECUs), and m4.2xlarge (26 ECUs). Let R_{priv} represent the pool of VM instances in private Cloud and R_{pub} the pool of public Cloud VM instances. The total pool of VM instances $R_p = R_{priv} \cup R_{pub}$. Thus the union($\cup$) of the set of private and public VMs. Emphatically, the number of VMs in R_{pub} is assumed to be infinite whiles that of R_{priv} is finite. For the purposes of clarity in representation, we represent the j^{th} VM in either the public or private Cloud by R_j respectively, where $1 \leq j \leq n$. Each R_j has the following parameters, number of ECUs U_j, availability a_j, bandwidth bw_j, memory Me_j, processor speed ps_j and price P_j. R_j=$\{U_j, a_j, bw_j, Me_j, ps_j, P_j\}$. The availability a_j of an instance refers to the state where that VM instance is not in use by other jobs. It is considered as an indicator value $\{0, 1\}$, which determines whether or not a VM can be provisioned. If a VM is in use regardless of the number of free compute units, the value of $a_j = 0$. However, as soon as a VM has finished processing the current job, $a_j = 1$ which indicates that VM is now idle for provisioning.

User jobs are assumed to contain parallel tasks with average parallelism (A_k) and a coefficient of the variance of parallelism as σ_k [22]. Each job can run on several virtual processors(ECU) but restricted to one VM instance. There are l dynamic jobs that can be submitted to the SaaS provider for provisioning. Let the k^{th} job in l number of jobs be represented as j_k. Each job J_k, where $1 \leq k \leq l$ has t_i number of tasks, number of required processors rc_k, job length l_k and deadline for job completion D_k.

TABLE I. NOTATIONS

Symbols	Description
t_i	The i^{th} Task which is a subset of a job
S	Scheduler
μ_r	Service rate of task
T_k^W	The waiting time of job
N	The maximum number of jobs allowed to be in queue
D_k	The deadline for the completion of job
T	The overall execution time with fault recovery
α_{kj}	the runtime estimate of the k^{th} on the j^{th} VM
W_k	Threshold waiting time of a job
b_w	Bandwidth
a_j	The availability-state of the j^{th} VM instance
R_p	The total pool of VM instances (public and private)
R_{priv}	The set of VM instances in the private Cloud
R_{pub}	The set of VM instances in the public Cloud
R_j	The j^{th} VM instance (public or private)
ps_j	Processing speed of a VM instance j
$_j$	Failure rate
μ_j	Recovery rate
β	The fraction of the maximum power in idle state
ω_{max}	Maximum power consumed in active state
G	The communication cost
C	Overall cost considering fault recovery
ϵ	Constant price for power consumption in private Cloud
A_k	average parallelism of a job
σ_k	coefficient of the variance of parallelism
S_{kj}	Speedup of the K^{th} job on the j^{th} VM

B. Overall Execution Time

For the processing of each job, a real life situation is considered, where the VM running tasks has the probability of failing and recovering. Jobs are made up of parallel tasks and each k^{th} job has i number of tasks where $1 \leq i \geq k$. The time for the execution of a job on VM j without considering the possibility of VM failure is denoted by $T_{kj}^{(e)}$. Mathematically,

$T_{kj}^{(e)}$ can be computed as shown in (1) ;

$$\tau_{ij} = \frac{length(t_i)}{ps_j}$$
$$T_{kj}^{(e)} = max\{\tau_{ij}\} \quad (1)$$

Where ps_j is the processing speed of VM *j*, $length(t_i)$ is the length of task i and max $\{\tau_{ij}\}$ is the maximum finish time of the i^{th} task on *j* (i.e. the time the last task is completed on *j*). Assuming all failures are recoverable, then after some time (recovery time), the VM resumes execution. The failure rate on *j* follows a Poisson process with rate λ_j. Hence, the total number of failures $N_j(t)$ during a time interval of (0, t] can be computed as shown in (2) [6];

$$Pr\{N_j(t) = b\} = \frac{(\lambda_j t)^b}{b!} e^{-\lambda_j t}, b = 0, 1, ... \quad (2)$$

Given that *b* is the instance for an occurrence of failure. Moreover, let $RT_j^{(b)}$ represent the b^{th} recovery time on *j*, where $b = 0, 1, ...$ Emphatically, all $RT_j^{(b)}$ are exponential random variables with parameter μ_j, given that μ_j is the rate of recovery on j. Let the total recovery time during a time interval of (0, t] on *j* be denoted as $RT_j(t)$. $RT_j(t)$, can be computed as in (3) [3].

$$RT_j(t) = \sum_{b=1}^{N_j(t)} RT_j^b \quad (3)$$

it is obvious that $RT_j(t)$ is a compound Poisson process, whose mean value is;

$$E[RT_j(t)] = \frac{\lambda_j(t)}{\mu_j} \quad (4)$$

Hence, the actual execution time of a job denoted by $AT_{kj}^{(e)}$, with fault recovery, is the sum of $T_{kj}^{(e)}$ and the recovery time of job *k* on *j*.

$$AT_{kj}^{(e)} = T_{kj}^{(e)} + RT_{kj} \quad (5)$$

With an expectation of

$$E[AT_{kj}^{(e)}] = E[T_{kj}^{(e)} + RT_{kj}] \quad (6)$$

The value RT_{kj} is the recovery time of job *k* on *j*. Moreover, based on our scenario jobs can be transmitted from the private cloud to the public cloud. Hence, there exist a time for communication. Let $\Theta(j_k)$ be the set of links that can be used to transfer jobs. Let the communication time without considering the possibility of a link failure be denoted by T_c. T_c, can be computed as shown in (7). Mathematically, T_c is dependent on the amount of data transferred through link *c* to its destination and the bandwidth b_w such that c $\in \Theta(j_k)$ [5].

$$T_c = \frac{data_c(j_k)}{b_w} \quad (7)$$

Assuming all failures are recoverable, then after some time (recovery time), the communication link gets connected and allows for data transfer. Let γ_c be the failure rate on link *c* [3]. The failure rate follows a Poisson process with rate γ_c. Hence, the total number of failures $X_c(t)$ during a time interval of (0,t] can be computed as;

$$Pr\{X_c(t) = b\} = \frac{(\gamma_c t)^b}{b!} e^{-\gamma_c t}, b = 0, 1, ... \quad (8)$$

Here, *b* is the instance for an occurrence of failure. Moreover, let the b^{th} recovery time on link *c*, where *b=0,1,...* be denoted as $RT_c^{(b)}$. All $RT_c^{(b)}$ are exponential random variables with parameter (δ_c). Given that δ_c is the rate of recovery on link c. Let the total recovery time on link *c* be denoted as $RT_c(t)$, the total recovery time during a time interval of (0,t] can be computed as [5];

$$RT_c(t) = \sum_{b=1}^{X_c(t)} RT_c^b \quad (9)$$

It can be seen that $RT_c(t)$, is a compound Poisson process, whose mean value is

$$E[RT_c(t)] = \frac{\gamma_c(t)}{\delta_c} \quad (10)$$

If RT_c is the recovery time for communication failure, then the communication time considering fault and recovery denoted as T^{ce} can be computed as;

$$T^{ce} = T_c + RT_c \quad (11)$$

with an expectation of

$$E[T^{ce}] = E[T_c + RT_c] \quad (12)$$

Finally, there is the probability that an incoming job may find other jobs in the system on its arrival. At the arrival of a job, if the queuing system currently has sufficient vacancies, then the job will enter the queue, else it will be blocked and the job request fails. Details for computing the blocking probability can be found in [5]. All jobs that enter the queue successfully will possibly reach the scheduler *S* (*S* refers to homogeneous schedulers, $S > 1$) after some waiting time. For *k* jobs, the expected waiting time of jobs of size *l* in a queue for $l \leq S$ is given by:

$$E[T_k^W] = \begin{cases} 0 & 0 \leq k \leq S - l \\ \frac{k-S+l}{\mu_r S} & S - l \leq k \leq N - l \end{cases} \quad (13)$$

Moreover, the expected waiting time of a job of size *l* in a queue given that $l > S$ is given by:

$$E[T_k^W] = \frac{k - S + l}{\mu_r S}, 0 \leq k \leq N - l \quad (14)$$

where *N*, μ_r is the maximum number of jobs allowed to be in a queue and the service rate respectively. Details of this model have been well studied in [5] [23].

Let the overall execution time for each job *k* on *j* be denoted by T_{kj}. T_{kj} is therefore the sum of the actual execution time AT_{kj}, the actual communication time $T^{(ce)}$ and the waiting time T_k^W. Moreover, let T denote the overall execution time

for processing all user jobs considering fault recovery. T and T_{kj} can be computed as follows:

$$T_{kj} = AT_{kj}^{(e)} + T^{(ce)} + T_k^W$$
$$T = \sum_{k=1}^{l}\sum_{j=1}^{n} T_{kj} \tag{15}$$

with an expectation of

$$E[T] = E[\sum_{k=1}^{l}\sum_{j=1}^{n} T_{kj}] \tag{16}$$

C. Total Operational Cost

1) Cost for Private VMs: The total operational cost refers to the overall expense the SaaS provider realizes for provisioning resources to execute client request. It is worth noting that, some related works such as [4] and [20], which focused on a similar problem to that of this paper, do not factor into account the cost incurred at private Cloud. Similar to [19], we argue that there is a cost for running local or private datacenters and such cost has an effect on the total expenditure of the SaaS provider. The cost incurred at local is therefore modeled as an energy cost. In the computation of the energy cost, private servers are assumed to be in two states, namely; idle and active state. A CPU of a server in its idle state is assumed to consume averagely 70% of the power consumed by a fully active CPU [24]. The energy cost is therefore modeled as follows: the power consumption rate P_u of servers is given by;

$$P_u(U) = \beta \cdot \omega_{max} + (1-\beta) \cdot \omega_{max} \cdot U) \tag{17}$$

β, refers to a fraction of the maximum power used in an idle state while ω_{max} refers to the maximum power consumed by server in an active state and U is the utilization factor of the servers. If U=0 then it means the server is in an ideal state. Hence, the fraction of the maximum power will be the value of the energy consumed. The utilization of the servers varies with respect to time due to the workload variability. Accordingly, the utilization is a function of time and it is represented as U(t). The total energy consumption on a compute node is given by [24];

$$P_U^T = \int_0^1 P_u U(t) \tag{18}$$

Given that the price for the power consumption of a compute node is a constant ε, as extensively studied in [70]. The price P_j^{priv} of a private VM is simply considered as the product of the constant ε and the number of compute units of that VM instance. That is $P_j^{priv} = \varepsilon \cdot U_j$ where U_j is the number of ECU in that VM instance. However, if a compute node is in idle state it is reasonable to consider the price of its VM as 70% of the active VMs price. That is although it is not in use but it consumes 70% of the power consumed by a fully active compute node as studied in [24]. Since in this paper all idle VMs are released immediately for allocation, the cost of running jobs on private VMs is considered only for active state VMs. Hence, the cost C_{kj}^{priv} for running the k^{th} job on private VM j where $1 \leq k \leq l$ and $1 \leq j \leq n$ can be computed as:

$$C_{kj}^{priv} = P_j^{priv} \cdot T_{kj} \tag{19}$$

Here P_j^{priv} is the price of the j^{th} VM instance in the private Cloud, and T_{kj} is the overall time for the completing job k on that j^{th} VM instance.

2) Cost for Public VMs: The cost of using public VMs is modeled to follow the Amazon EC2 billing model. Each instance runs for a minimum of one hour, hence the cost is computed per the execution time for job completion. Given that the price of a VM instance is represented as P_j^{pub}, let the cost for renting a VM at public be denoted as C_{kj}^{pub}. The cost for renting a public VM can be computed as:

$$C_{kj}^{pub} = P_j^{pub} \cdot T_{kj} \tag{20}$$

where $1 \leq k \leq l$ and $1 \leq j \leq n$. Therefore the cost for running VMs at local and renting On-demand VMs can be computed as shown in (21), where G is the communication cost between the VMs across the public and private Cloud. G is considered to be a constant.

$$Cost_{kj} = C_{kj}^{pub} + C_{kj}^{priv} + G \tag{21}$$

Furthermore, the total operational cost for running all jobs on available VM instances can be computed as:

$$C = \sum_{k}^{l}\sum_{j}^{n} Cost_{kj} \tag{22}$$

Here $1 \leq k \leq l$ and $1 \leq j \leq n$. With an expectation of;

$$E[C] = E[\sum_{k}^{l}\sum_{j}^{n} Cost_{kj}] \tag{23}$$

D. Problem Formulation

First, it is expected that a VM or a communication link may fail during the execution of jobs. The SaaS provider is challenged to make decisions on the number of VM instances to rent or instantiate at local such that total operational cost is reduced. Technically, clients have an estimated time (deadline) by which they expect their jobs to be completed. That means although there is the need to reduce total operational cost, the execution time of a job must not be greater than the deadline for the job. In this section, we present the optimization model for this problem. The problem is to minimize the total operational cost while ensuring that jobs are completed within its deadline. Formally:

$$min \quad E[C] \tag{24}$$

s.t.

$$\forall k \in \{1...l\}, \forall j \in \{1...n\} \tag{25}$$

$$E[T_{kj}] \leq D_k \tag{26}$$

(24): This is the optimization objective. It refers to the decision to minimize the total or overall operational cost. For instance, assuming there are five instance types of ECU values $\{1, 8, 13, 16,$ and $26\}$, and the runtime in seconds for running **job 1** on a standard VM instance of 1 ECU is 2, then the runtime in seconds for running **job 1** on all the other VM

instances are $\{16, 26, 32, \text{and } 52\}$ respectively. To achieve (24), the choice of VM instance chosen to run **job 1** should satisfy the constraint (25) and (26).

(25): This constraint indicates that a job is restricted to one VM during execution.

(26): An SLA contract is signed between the SaaS provider and users specifying the deadline for each job. The provider must therefore ensure that the overall execution time for each job does not exceed the clients' estimated deadline.

IV. Dynamic Resource Provisioning Algorithm

The VMs and communication links across the hybrid Cloud is considered to be failure prone. Therefore, the problem of minimizing operational cost without deadline violation is non-trivial. Based on our optimization model we are able to address this problem with two provisioning policies, namely: the private and public provisioning policies. These policies are implemented by an efficient algorithm (DRPA). Fig.1 gives an overview of the framework of DRPA. First, jobs arrive at the Cloud management system (CMS). The CMS maintains queue and computes the runtime estimate of jobs based on the information provided by the resource pool manager (RPM). The RPM handles the peering arrangement between Datacenter 1 (DC_1) manager and Datacenter 2 (DC_2) manager. It verifies whether or not a job must be transferred to the public Cloud for execution based on the runtime estimate of the incoming jobs, the availability state of the VMs and their associated prices. Once verified, the scheduler then schedules the job on the VM of choice. The VM of choice refers to the VM which satisfies the objective in (24) with respect to its constraint in (25) and (26).

A. Runtime Estimation

Studies have shown that Cloud provisioning strategies based on user supplied runtime estimate of job sometimes leads to overestimation or overprovisioning of resources [25], [26]. Several works have proposed approaches to predict job runtime [25], [27], [28], where the system computes the estimated runtime of a job and uses it rather than the users' only. Moreover, as indicated by [26], no single method of runtime estimation has proven to work well in all scenarios. The runtime estimation technique is popularly used in backfilling service techniques. However, we employ it to aid the decision making involved in efficiently leveraging VM instances across the hybrid Cloud. Without loss of generality, four approaches for computing runtime of jobs are considered in this paper. The four approaches are generated by adjusting the *user supplied runtime estimate*. The models are namely: *user supplied, fraction of user supplied, user runtime with error* and *recent average runtime estimation models*. Based on each VM instance comprising a number of compute units, we compute the runtime estimate of each job. We employ the speedup model of Downey [29] which was also used by [27] and [30]. Downey's model [29] depends on two important factors, namely; the degree of parallelism of a job (A_k) and the coefficient of the variance of parallelism σ_k. The values of A_k and σ_k are modeled based on the job characteristics provided by users using the model of Cirne & Berman [31]. This model [31] was also adopted in [27].Given the speedup of k^{th} job on the j^{th} VM as S_{kj}, the runtime estimate α_{kj} of a job *k* on VM *j* is computed as shown in (27). U_j refers to the number of ECUs in the *jth* VM.

$$\alpha_{kj} = \frac{U_j}{S_{kj}} \tag{27}$$

In the computation of the *user supplied runtime*, the job length used is assumed to be the job length given by the user at job submission. Since studies have shown that the user-supplied values leads to overestimation, an alternate approach have been proposed. Thus the *fraction of user supplied runtime* estimation. This approach uses a value equal to 1/3 of original value of the job length as used in the user supplied approach. In computing the *user runtime with error*, the estimate is obtained by using a slightly modified value of the job length as used in the user supplied runtime estimate. The modification is done by adding up a uniformly distributed random percentage between 0 and ten percent (10%) to the job length. Finally, in computing the *recent average runtime*, the average runtime of at least two completed jobs are used to generate the runtime of the next incoming job. At the extent of decision-making, if there exists less than two completed jobs, then the estimated value is assumed to be given by the user supplied approach [27], [32].

B. Provisioning Policies

1) Private Provisioning Policy: In our scenario, the SaaS provider intends to first run the incoming jobs on the private VMs. However, the provider can scale up for more VMs from the public Cloud when the need arises, i.e. public Cloud VMs are rented only when the waiting time of a job has exceeded its' threshold delay value W_k such that no other VM instance at local is available (i.e. a_j=0). The threshold delay value W_k of a job *k* is the maximum estimated time the job can wait for a VM to be provisioned such that its due deadline is still met. Incoming jobs may eventually be redirected to the public Cloud once its waiting time for the VM of Choice exceeds the threshold delay value and all other VMs at local are busy (i.e. a_j=0). W_k is computed as shown in (28), where D_k is the deadline of the waiting job, T_k^{ar} is the arrival time of the job, α_{kj} is the runtime estimate of the job on its VM of Choice and ϱ is the modifying factor. That means, $D_k - T_k^{ar}$ refers to the time until the deadline of the job.

$$W_k = max(0, \quad D_k - T_k^{ar} - \alpha_{kj} \cdot \varrho) \tag{28}$$

The modifying factor introduced in the computation of W_k supports the decision making for renting VM resources from the public Cloud. Algorithm 1 becomes more conventional when the value of the modifying factor ϱ is greater. Specifically, W_k approaches 0 with higher values of ϱ. A value of W_k=0 indicates that a resource must be provisioned immediately from the public Cloud to complete the job within the given deadline. On the other hand, lower values of ϱ indicates that the value of W_k would be greater than 0, leading to postponement of provisioning actions and a high probability of running jobs on private VMs which are comparatively cheaper than VMs in the public Cloud but associated with deadline violations. The provisioning of VM for incoming job *k* at local must satisfy (29), such that (25) and (26) holds. Thus the minimum resulting value that is yielded when the price of

each VM in the private Cloud is multiplied by the runtime estimate α_k of the k^{th} job.

$$\mathbf{P1} = min(P_j^{priv} \cdot \alpha_{kj}) \quad (29)$$

Let the private provisioning policy problem be called **P1**

2) Public Provisioning Policy: Public Cloud VMs are rented only when the waiting time of a job has exceeded the threshold delay value W_k. A job may wait to be assigned to a VM of choice if the VM is expected to be free soon. However, if the waiting time of the job exceeds its threshold delay value W_k, then the SaaS provider must decide on the VM to rent from the public Cloud. The VM to rent from the public Cloud can be determined by solving (30). The VM of choice, with respect to the public provisioning policy, refers to the VM which satisfies the objective in (30)) whiles constraint in (25) and (26) holds.

$$\mathbf{P1} = min(P_j^{pub} \cdot \alpha_{kj}) \quad (30)$$

P_j^{pub}, is the price of the j^{th} VM instance in the public Cloud and α_{kj} is the runtime estimate of the job *k* on that instance. Let the public Cloud provisioning be called **P2**.

To achieve the optimization objective, we designed a heuristic-based dynamic resource provisioning algorithm DRPA as shown in Algorithm 1 to solve **P1** and **P2**. Each job is considered to have a number of parallel tasks. Jobs are either transferred to the public Cloud or run on the private Cloud based on its runtime estimate, deadline, waiting time as against the threshold value, the availability state and the prices of the VMs. Initially, no VM is rented from the public Cloud. For each job *k* that arrives at the CMS, its runtime estimate α_{kj} is computed. Given that the pool of VM instance is R_p, the job is first directed to the private Cloud on the condition that the available VMs in the private Cloud are free and satisfies **P1** (Step 1-8). The determination of the VM of choice in the private cloud is initiated by solving **P1**. The result of **P1** is simply the minimum resulting value that is yielded when the price of each VM in the private Cloud is multiplied by the runtime estimate α_{kj} of the k^{th} job. The VM of choice is then scheduled to run the k^{th} job. (step 9). Next, the availability-state of the VMs in the private cloud are updated (step 10). If the VM of choice is not available (i.e a_j=0) then the job j_k must wait provided that its' waiting time does not exceed the delay threshold factor W_k. If the job can wait, then it is added to the waiting list **W** and scheduled (added to the waiting list **W**) to run on the VM expected to be free or idle soon (Steps 12-14). As indicated in (steps 25-30), for each job found in **W**, the algorithm will first search for other VMs in the private Cloud which may be free and satisfies *P1* to complete the job. Otherwise, the VM instance in the public Cloud which satisfies **P2** will be rented On-demand to run the job (steps 16-18). The solution of **P2** is is simply the minimum resulting value that is yielded when the price of each VM rented from the public Cloud is multiplied by the runtime estimate α_k of the k^{th} job. After the job is completed, the VM is shut down to avoid extra billing (step 19). The queue is updated and the next job follows (Step 34). The application of methods such as job runtime estimation, job postponing, and idle VM instance termination assures that the provisioning policy keeps an all encompassing view of the incoming jobs which are dynamic in nature.

Algorithm 1 Dynamic Resource Provisioning

Input: set of jobs, VM prices, VM instances
Output: Dynamic provisioning of VM resources with minimum cost

1: **for** each job j_k **do**
2: $\alpha_{kj} \leftarrow$ compute runtime estimate on all instances;
3: $R_p \leftarrow$ resource pool ($R_{priv} \cup R_{pub}$);
4: $R_{pub} \leftarrow \emptyset$;
5: decision $\leftarrow select R_{priv}$;
6: compute W_k;
7: **for** each VM resource R_j in private Cloud **do**
8: **if** ($a_j = 1$) **then**
9: solve **P1**;
10: Update VMs a_j state;
11: **else**
12: **if** ($W_k > 0$) **then**
13: Delay job until W_k;
14: Add job to waiting list **W**;
15: **else**
16: decision $\leftarrow$ select R_{pub};
17: **for** each VM resource R_j in public Cloud **do**
18: Solve **P2**;
19: VMidle.Shutdown;
20: **end for**
21: **end if**
22: **end if**
23: **end for**
24: **end for**
25: **for** each job $J_k \in W$ **do**
26: decision $\leftarrow$ select R_{priv};
27: **for** each VM resource R_j in private Cloud **do**
28: **if** (a_j=1) **then**
29: Solve **P1**;
30: Remove job J_k from **W**;
31: **end if**
32: **end for**
33: **end for**
34: Update queue;

V. Performance Evaluation

In this section, we study the performance of the proposed DRPA. We run a set of experiments using a real world workload log obtained from the Distributed ASCI Supercomputer 2 (DAS2 fs4) available at the Parallel Workloads Archive [33]. The original workload is composed of a total of 33,795 with parallel tasks submitted over a period of 11 months by a total of 40 users. This workload is suitable for the simulation in this work because of its property of parallelism and varied job lengths. It contains an information about the user supplied runtime estimate. However, as stated earlier, we shall conduct a further evaluation of the DRPA using other runtime estimation methods namely: *user runtime estimate with error, fraction of user supplied runtime estimate* and the *recent average runtime estimate*. The objective of our experimental study is to evaluate the performance of DRPA based on two metrics; cost and deadline violation rate in the presence of VM and communication link failure. In order to capture the performance of the proposed algorithm, our results are averaged on 50 simulation runs with varying number of jobs between 3,000 and 33,000.

In the first part of our experiments, we study the effects of the four runtime estimation approaches on the DRPA in two cases: first, considering fault and recovery; second, without considering fault and recovery. The objective is to determine the kind of runtime approach which works better as against our performance metrics. In the second part of our experiment, we conduct a comparative study with two benchmark approaches namely: the best and worse case approaches [19] [27] and also with a recent algorithm called Size-based Selective-backfilling hybrid Cloud provisioning policy [4]. The objective of such comparison is to demonstrate the efficiency of the proposed algorithm.

A. Experimental Setup

We performed all our experiments on a Pentium(R) Dual-Core processor with a processor speed of 2.8GHz and a memory of 4GB. The CloudSim toolkit [34] was used to simulate a Cloud system consisting of two datacenters, namely $datacenter1$ and $datacenter2$. $datacenter1$ is the datacenter for the public Cloud and $datacenter2$ is the datacenter for the private Cloud. The NetworkTopology component of the CloudSim was modified to represent the communication between the Datacenters. Five instance types were simulated in the experiments. They were modeled after the characteristics of Amazon EC2 types [21]. The set of possible values of ECU is U_j: 1, 8, 13, 16, and 26. The EC2 types which are of the exact ECU characteristics are m1.small (1 ECU), c4.large (8 ECUs), m4.large (13 ECUs), c4.xlarge (16 ECUs), and m4.2xlarge (26 ECUs). The five VM instance types are hosted on each datacenter with distinguishing properties with respect to their ECUs. Since the public Cloud is perceived to have unlimited resources, five times of the VM instance type in the $datacenter1$ are hosted on $datacenter2$. However, this number increases as may be required for renting. Moreover, to obtain the values of the processor speeds and prices in the public Cloud, we adopt a similar method as applied in [35], because their approach was modeled after Amazon EC2 services. The processor speed of each host is uniformly distributed within the range [100, 1000] with the average speed of 550 MIPS. Without loss of generality, the price of a VM have roughly linear relationship with its processor speed, so as to generally ensure that a faster host yields more execution cost than a slower host in executing the same job. The method to derive the processor speed in the private Cloud is generated in the same way as in the public. However, its prices are generated according the energy cost model. The value of the constant ε as used in the experiment is 0.04 (ε is the constant price assumed for energy consumption). The failure rates of the VMs and communication links are randomly generated from the interval of [0.01, 0.1] [5] [6]. All recovery rates are randomly generated from the interval of [0.05, 0.15]. The average bandwidth between the compute nodes is set to 10 Mbps as used in [4], and finally, the modifying factor ϱ is a uniformly distributed random values between 0 and 2.

B. Effects of the Four Runtime Estimation Approaches

First, we evaluated DRPA using each of the runtime estimation approach against the performance metrics considering fault and recovery. The results in Fig. 2a and Fig.2b, shows the behavior of DRPA after job execution. It can be seen that, the Total operational cost increases slowly when the workload is relatively small. This is attributed to the fact that at such point most of the jobs are run on the private Cloud. However, when the workload shoots up there is a relative increase in total operational cost due to the cost for renting resources from the public Cloud. Moreover, the peak cost for each of the approach varies. The fraction of user supplied runtime approach yields the cheapest cost with comparatively, the worst deadline violation as can be shown in Fig.2b. This is due to the underestimation of the user job length. This underestimation resulted in the acquisition of relatively minimum runtime estimate which led to the deployment of VMs with cheap cost, hence causing its total operational cost to be the cheapest. However, this caused much deadline violation, hence causing its deadline violation rate to be 25.46%. According to Fig.2a and Fig.2b, it is obvious that the user runtime with error approach achieved the best results with respect to the two performance metrics; Total operational cost and deadline violation rate. It comparatively achieved a total operational cost of $3,749.00 which is about 14.4% better than the total operational cost of the user supplied runtime approach. Moreover, it yielded a deadline violation rate of 5.34% which is approximately 13% greater than the deadline violation rate of the user supplied runtime approach. The user runtime with error approach of the proposed ODPA performed consistently better.

Secondly, we evaluate DRPA using the four runtime approaches as against the performance metrics without considering fault and recovery. Fig.3a and Fig. 3b, shows the comparative results of the four approaches as against Total cost and deadline violation with and without fault recovery respectively. First, it is obvious to point out that without considering fault and recovery the various runtime approaches performs relatively better against cost and deadline violation than the case where fault and recovery is considered. In essence, this demonstrates the impact of fault recovery on service performance. It can be deduced that, it is relevant to develop a Cloud system that factors into account fault recovery, since it has relative impact on the cost and the execution time for job completion.

In summary, it can be observed that, for a given workload, having a perfect runtime estimate does not actually translate into a more efficient provisioning, especially in terms of cost. This is due to the fact that all of the approaches had a relatively low or high deadline violation. This can be attributed to moderate over or under estimations, which caused cheaper or relatively expensive instances to be requested. Analytically, the user runtime with error approach in this work appears to be comparatively better than the rest of the approaches against the two performance metrics (cost and deadline violation).

C. Comparative Analysis

In this experiment, we simulated three other algorithms namely: Size-based selective-backfilling algorithm [4], Best case algorithm [19], and worst case [27] algorithm. In the worst case, it is assumed that all jobs are processed with VMs rented form the public Cloud in an On-demand fashion. However, the information for each job is known. Jobs are run on the most efficient VMs. Similar to [19], in the best case scenario, it is assumed that there is an oracle that knows all the future

TABLE II. COMPARATIVE ANALYSIS OF DRPA WITH OTHER ALGORITHMS

Algorithms	Number of Jobs	Cost for Renting($)	Private Cost($)	Total Operational Cost($)	Deadline Violation Rate (%)
Best Case [19]	33,000	0	2,937.00	2,937.00	0
Worst Case [27]	33,000	7,764.00	0	7,764.00	0
SBSB [4]	33,000	3892.00	NC	3892.00	9.938
DRPA(USRWE)	33,000	3,531.00	218.00	3,749.00	5.34

DRPA(USRWE)- Dynamic resource provisioning algorithm with User runtime with error approach, NC- not considered

information and can use the most efficient VM instance for each job to achieve an ideal solution. It is worth noting that, the best case scenario is not real. However, it is hypothetically considered as a lower bound implemented to evaluate the cost-effectiveness of the proposed policy. The Size-based selective-backfilling algorithm (SBSB) [4] was proposed by Javadi et al. They considered a similar hybrid Cloud resource provisioning problem to ours. However, they do not consider the cost incurred at the private Cloud as well as the probability of failure in both the private and public Cloud. They assumed that only the VMs in the private Cloud are failure prone.

To evaluate our proposed algorithm, we adopt the user runtime with error approach (USRWE) as the base policy of DRPA and compare its result to the simulation results of the Best Case, Worst Case and the SBSB algorithms. The result in Fig. 4a, Fig. 4b and in Table II shows the behavior of the four algorithms after running 33,000 jobs. Comparing the results obtained in Table II for the cost of renting VMs in the case of each algorithm, the proposed DRPA(USRWE) outperforms the size-based selective backfilling (SBSB) and the Worse Case algorithms by a percentage of approximately 9.3% and 54.52% respectively. However, the Best Case algorithm yields a result that outperforms all the other three algorithms including our algorithm. It yielded no cost all ($0.00), because in the Best Case scenario all jobs are run on the private Cloud.

On the other hand, our proposed DRPA(USRWE) outperforms the Best Case algorithm in terms of the operational cost at the private Cloud. The Worst Case yielded no cost at all ($0.00) since in this scenario all jobs are run with VMs in the public Cloud. Hence, it outperformed both our DRPA(USRWE) which yielded a cost of $218.00 and the Best Case which yielded a cost of $2,937.00. In contrast, the SBSB algorithm does not factor into account the operational cost at the private Cloud. It must be noted that the cost of the Best case is high because in this scenario all the 33,000 jobs were run on the private Cloud. Moreover, considering the results of our proposed DRPA(USRWE) in terms of the cost for renting and the cost at the private Cloud, it can be observed that the cost for renting is fairly larger than the cost at private. This can be attributed to two factors: first, the assigned price for a VM at the private Cloud was fairly lower than that of the cost in the public Cloud. Second, about 55% of the jobs were run with VMs on the public Cloud due to the need to satisfy the deadline constraint of jobs.

Considering the total operational cost of all the algorithms, the proposed DRPA(USRWE) yielded $3,749.00 which is approximately 51.71% better than the Worse Case. However, when compared with the Best Case, it is approximately 28% worse. Also it is 3.71% better than the SBSB algorithm. There was no deadline violation in the Best and Worst Case comparatively. Unlike Best and the Worst Case, the deadline violation rate of the proposed DRPA(USRWE) is 5.34% which is 46.27% better than the SBSB algorithm.

In summary, the obtained results show that, by the adoption of the proposed DRPA(USRWE), SaaS providers can save 51.71% of total cost compared to the Worst Case scenario, even in the presence of VM and communication link failures, with a relatively low deadline violation rate of 5.34%. Moreover, the proposed DRPA(USRWE) performs better than SBSB by a percentage of 9.3% with respect to the cost for renting VM instances.

VI. CONCLUSION

In the hybrid Cloud paradigm, SaaS providers are faced with the challenge of minimizing cost without sacrificing resource provisioning service performance. The probability of VM and communication link failure is a real life factor which has an effect on service performance. This paper focuses on the problem of minimizing total operational cost involved in hybrid Cloud resource provisioning considering fault recovery. Specifically, we propose a heuristic-based algorithm which applies methods such as job runtime estimation, job postponing, and idle VM instance termination to assure that the provisioning policy keeps an all encompassing view of the incoming jobs which are dynamic in nature. The results obtained from our experimental study show that, by the adoption of the proposed DRPA(USRWE), SaaS providers can save 51.71% of total cost compared to the Worst Case scenario, even in the presence of VM and communication link failures, with a relatively low deadline violation rate of 5.34%. Moreover, the proposed DRPA(USRWE) outperforms the recently proposed SBSB by a percentage of 9.3% with respect to the cost for renting VM instances.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging {IT} platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, 2009.

[2] D. Davis, *Companies Enjoying Big Success with Hybrid Cloud*. Skytap blog, 2013. [Online]. Available: https://www.skytap.com/blog/companies-enjoying-big-success-with-hybrid-cloud/

[3] W. Q. Heyang Xu, Bo Yang and E. Ahene, "A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery," *KSII Transactions on Internet and Information Systems*, vol. 10, no. 3, pp. 976–995, March 2016.

[4] B. Javadi, J. Abawajy, and R. Buyya, "Failure-aware resource provisioning for hybrid cloud infrastructure," *Journal of Parallel and Distributed Computing*, vol. 72, no. 10, pp. 1318 – 1331, 2012.

[5] B. Yang, F. Tan, and Y.-S. Dai, "Performance evaluation of cloud service considering fault recovery," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 426–444, 2013.

[6] B. Yang, H. Hu, and S. Guo, "Cost-oriented task allocation and hardware redundancy policies in heterogeneous distributed computing systems considering software reliability," *Comput. Ind. Eng.*, vol. 56, no. 4, pp. 1687–1696, May 2009.

[7] R. Raman, M. Livny, and M. Solomon, "Matchmaking: An extensible framework for distributed resource management," *Cluster Computing*, vol. 2, no. 2, pp. 129–138, 1999.

[8] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," 2001.

[9] B. Palanisamy, A. Singh, L. Liu, and B. Langston, "Cura: A cost-optimized model for mapreduce in a cloud," in *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, May 2013, pp. 1275–1286.

[10] L. Wu, S. K. Garg, and R. Buyya, "Sla-based resource allocation for software as a service provider (saas) in cloud computing environments," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, May 2011, pp. 195–204.

[11] S. Chaisiri, R. Kaewpuang, B. S. Lee, and D. Niyato, "Cost minimization for provisioning virtual servers in amazon elastic compute cloud," in *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, July 2011, pp. 85–95.

[12] Q. Li, Q. Hao, L. Xiao, and Z. Li, "Adaptive management of virtualized resources in cloud computing using feedback control," in *2009 First International Conference on Information Science and Engineering*, Dec 2009, pp. 99–102.

[13] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34–40, September 2012.

[14] T. R. G. Nair and M. Vaidehi, "Efficient resource arbitration and allocation strategies in cloud computing through virtualization," in *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, Sept 2011, pp. 397–401.

[15] B. C. Tak, B. Urgaonkar, and A. Sivasubramaniam, "To move or not to move: The economics of cloud computing," in *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 5–5.

[16] M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 243–254, Aug. 2010.

[17] T. Guo, U. Sharma, T. Wood, S. Sahu, and P. Shenoy, "Seagull: Intelligent cloud bursting for enterprise applications," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 33–33.

[18] P. Marshall, K. Keahey, and T. Freeman, "Elastic site: Using clouds to elastically extend site resources," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, May 2010, pp. 43–52.

[19] S. Li, Y. Zhou, L. Jiao, X. Yan, X. Wang, and M. R. Lyu, "Delay-aware cost optimization for dynamic resource provisioning in hybrid clouds," in *Web Services (ICWS), 2014 IEEE International Conference on*, June 2014, pp. 169–176.

[20] B. Javadi, J. Abawajy, and R. O. Sinnott, "Hybrid cloud resource provisioning policy in the presence of resource failures," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, Dec 2012, pp. 10–17.

[21] A. Inc, "Ec2 instances," pp. 1136 – 1142, 2007. [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[22] A. B. Downey, "A model for speedup of parallel programs," Berkeley, CA, USA, Tech. Rep., 1997.

[23] P. Vijaya Laxmi and U. Gupta, "Analysis of finite-buffer multi-server queues with group arrivals: Gix/m/c/n," *Queueing Systems*, vol. 36, no. 1, pp. 125–140, 2000.

[24] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012, special Section: Energy efficiency in large-scale distributed systems.

[25] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Backfilling using system-generated predictions rather than user runtime estimates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 789–803, June 2007.

[26] D. Tsafrir, *Using Inaccurate Estimates Accurately*, E. Frachtenberg and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[27] W. Voorsluys, S. K. Garg, and R. Buyya, "Provisioning spot market cloud resources to create cost-effective virtual clusters," in *Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ser. ICA3PP'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 395–408. [Online]. Available: http://dl.acm.org/citation.cfm?id=2075416.2075453

[28] W. Tang, N. Desai, D. Buettner, and Z. Lan, "Analyzing and adjusting user runtime estimates to improve job scheduling on the blue gene/p," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, April 2010, pp. 1–11.

[29] A. B. Downey, "A model for speedup of parallel programs," Berkeley, CA, USA, Tech. Rep., 1997.

[30] 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*, vol. 12, no. 6, pp. 529–543, Jun 2001.

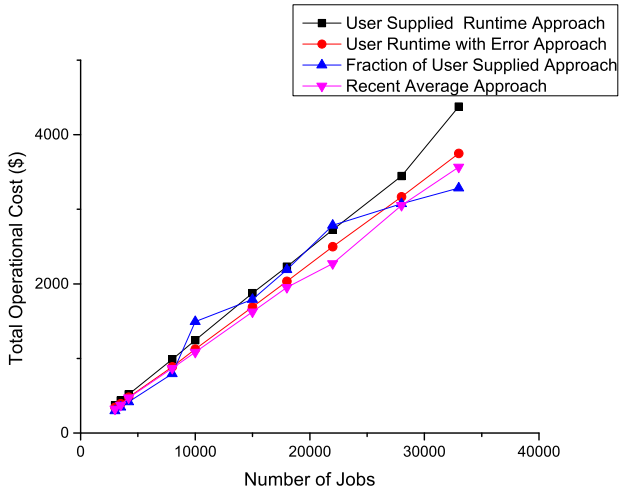
[31] W. Cirne and F. Berman, "Using moldability to improve the performance of supercomputer jobs," *Journal of Parallel and Distributed Computing*, vol. 62, no. 10, pp. 1571 – 1601, 2002.

[32] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Modeling user runtime estimates," in *In 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2005*. Springer-Verlag, 2005, pp. 1–35.

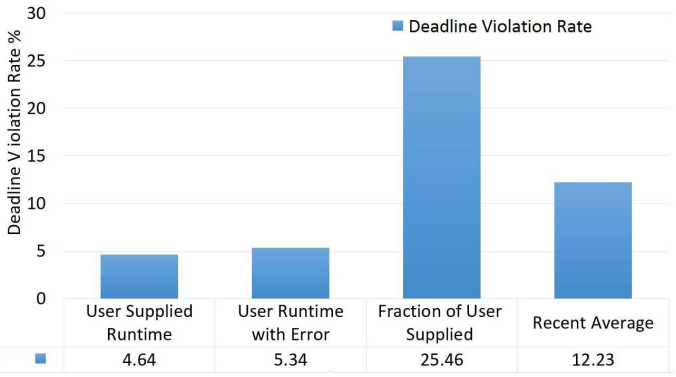
[33] W. Archive, "Logs of real parallel workloads from production systems," 2003. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/

[34] 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," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[35] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158 – 169, 2013, including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

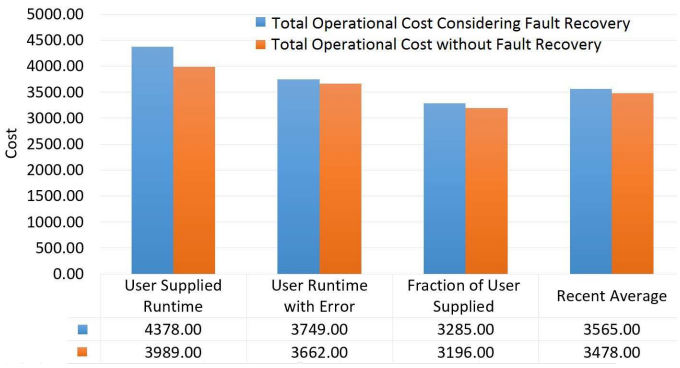


(a) Total Operational Cost of the four Runtime Approaches

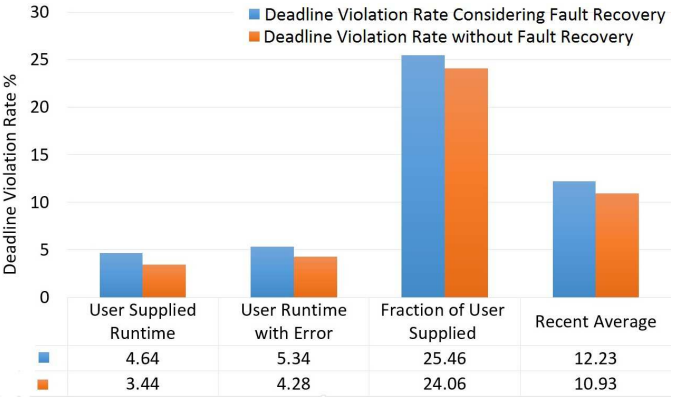


(b) Deadline Violation rate of the four Runtime Approaches

Fig. 2. Simulation results using the four approaches

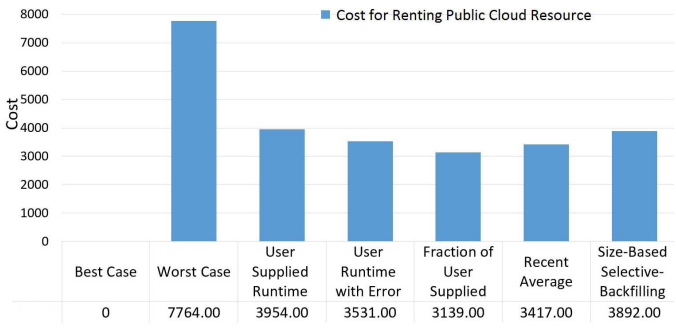


(a) Impact of Fault Recovery on Total Operational Cost

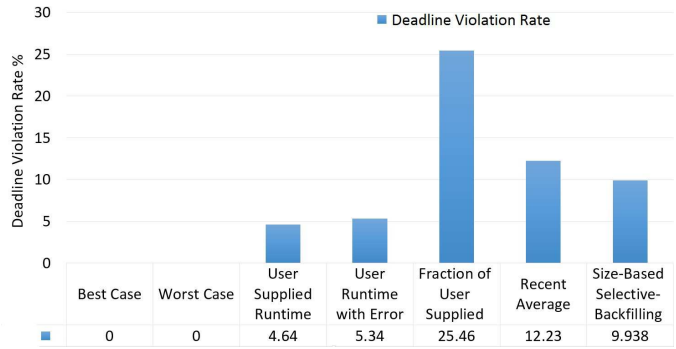


(b) Impact of Fault Recovery on Deadline Violation Rate

Fig. 3. Simulation results depicting the impact of Fault Recovery on DRPA



(a) Comparative Cost for Renting Public Cloud Resource



(b) Comparative Analysis of Deadline Violation Rate

Fig. 4. Simulation results depicting the performance of DRPA against other Algorithms