Optimal management of virtual infrastructures under flexible cloud service agreements

Zhiling GUO
Singapore Management University, ZHILINGGUO@smu.edu.sg

Jin LI
Xidian University

Ram RAMESH
State University of New York at Buffalo

Optimal Management of Virtual Infrastructures under Flexible Cloud Service Agreements

Zhiling Guo[a], Jin Li[b,*], Ram Ramesh[c]

[a] School of Information Systems, Singapore Management University, Singapore 178902
[b] School of Economics and Management, Xidian University, Xi'an, China 710071
[c] Department of Management Science and Systems, State University of New York, Buffalo, New York 14260
[*] Corresponding author

Contact: zhilingguo@smu.edu.sg, http://orcid.org/0000-0002-9058-4710 (ZG); jinli@xidian.edu.cn, http://orcid.org/0000-0003-3340-1516 (JL); rramesh@buffalo.edu, http://orcid.org/0000-0002-6232-1841 (RR)

Abstract

A cloud service agreement entails the provisioning of a required set of virtual infrastructure resources at a specified level of availability to a client. The agreement also lays out the price charged to the client and a penalty to the provider when the assured availability is not met. The availability assurance involves backup resource provisioning and the provider needs to allocate backups cost-effectively by balancing the resource provisioning costs with the potential penalty costs. We develop stochastic dynamic optimization models of the backup resource provisioning problem, leading to cost-effective resource management policies in different practical settings. We present two sets of dynamic provisioning strategies: *periodic* policies where resources are adjusted at regular intervals, and *aperiodic* policies that allow flexible timing of such interventions. A *closed-loop (CL)* optimization model under conservative resource control and a *certainty-equivalent (CE)* optimization model under aggressive resource control are developed for periodic resource management. Similarly, aperiodic resource management is modeled using two different strategies: *single intervention with single look-ahead (SISL)* and *multiple interventions with single look-ahead (MISL). Online optimization* algorithms for both the periodic and aperiodic models are developed. The worst-case behavior of the algorithms is studied using *competitive ratio analysis*, and the expected behavior using computational investigations. Using these studies, managerial guidelines for choosing the best resource management strategy under different client-specific, service-specific and system-specific resource optimization conditions are presented. We validate our models based on use cases constructed from Amazon EC2 with their actual pricing and service credit data. The practical guidelines from this study will aid contract administrators in cloud data centers to both efficiently formulate service level agreements and cost-effectively manage the virtual infrastructure resources committed in such agreements.

Keywords: cloud computing, service level agreement (SLA), dynamic programming (DP), online algorithm, virtual machines (VMs), cloud resource management

1. Introduction

The broad concept of cloud computing entails three fundamental models of service hosting and delivery: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). IaaS functions at the lowest level by providing virtualized computing, storage and network services that would support PaaS and SaaS (Colman 2013). A cloud service level agreement (SLA) for IaaS typically includes dynamic metrics such as infrastructure service availability, performance latency and response delay for emergencies, and a host of medium to long term metrics such as data security, privacy and integrity.[1] Availability is a central commitment in all SLA abstractions—compute, network and storage (Cloud Standards Customer Council 2017), and a failure to meet the committed level of availability by a service provider results in either penalties or service credits given by the provider to the client. For example, Dimension Data proposes a system of service credits and other vendors such as Amazon Web Services (AWS) also provide similar terms to compensate clients for system downtime that exceed the promised level of availability in an SLA. These contractual arrangements lend valuable flexibility to a service provider in fulfilling the availability commitment in an SLA, especially when faced with system failures and other service disruptions that may be hard to predict and control.

Central to IaaS is the concept of *virtualization*. Virtualization is essentially software that encapsulates multiple operating systems individually and enables them to co-exist and run independently on the same physical server platform. Each such operating system is a *virtual machine* (VM). Multiple virtual machines that are co-hosted on a physical server would execute concurrently and independently. Accordingly, a *virtual infrastructure* typically comprises of a set of virtual machines that are hosted on a set of physical servers; the virtual machines provide the basic functionalities of computing, storage and networking in the distributed cloud architecture. This IaaS model is widely adopted by several cloud service providers such as AWS, Google, Microsoft, Rackspace, SalesForce and many others.[2] Under the notion of virtual infrastructures, when a client requests cloud services, the client requirements translate to an SLA for IaaS that broadly specify the number of VMs needed, their critical dynamic service-level metrics, the service window over which the metrics are evaluated, and the service pricing and service credit schemes (Cloud Standards Customer Council 2017).

[1] Dimension Data Cloud Terms of Services and Related Service Levels Descriptions: https://www.dimensiondata.com/en-US/Solutions/Cloud/Pages/Service-level-agreement-of-Public-IaaS.aspx

[2] The Essential Guide−Infrastructure as a Service: http://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS

Denoting the VMs required in an SLA as *primary VMs*, the service provider usually builds redundancy in the virtual infrastructure in the form of *backup VMs* to ensure the availability of the required number of VMs at the level promised in the SLA. This redundancy enables fault-tolerance by checkpointing and rollback recovery, and guidelines for VM checkpointing using backups are given in the Microsoft communication.[3] Various models of backup checkpointing exist in both literature and practice and the widely adopted models are the *powered-on* models under conventional VM failures and especially when critical applications are supported (Du et al. 2015). In the above context of an SLA for a virtual infrastructure under powered-on checkpointing, Yuan et al. (2018) address the trifecta of backups, VM pricing and penalty for shortfall in the assured availability in an SLA. As we increase the number of backup VMs, the likelihood of SLA violation decreases and hence the expected penalty cost, but the VM provisioning cost increases. For a given price-penalty combination, Yuan et al. (2018) derive the optimal level of backups to minimize the expected total cost over a contract period by balancing the trade-off between the VM provisioning cost and expected penalty cost.

A limitation of the study of Yuan et al. (2018) is the single and static determination of the level of backups needed for a service agreement at the beginning of the contracted period, and its enforcement throughout the period. In the presence of practical random events such as system failures and recovery, such static determination could be sub-optimal in real-world data center operations, especially when system downtimes are common and significant. Consequently, cloud resource provisioning should be a dynamic rather than a static decision. As the cumulative system downtime randomly grows with the progression of service in a contracted period, a cloud data center needs to take advantage of the observed downtime information, reassess existing resource commitments, and make dynamic decisions on backup deployment. While such dynamic management would lead to efficient resource allocation, the challenge is how to time such backup adjustments and determine the optimal adjustment levels, especially under the transient random failure and recovery processes involved.

Motivated by the above, we develop dynamic optimization models to guide the service provider's resource provisioning strategies. By modeling the number of primary and backup VMs deployed as resources, and denoting the times of resource adjustment as *interventions*, we develop both *periodic* and *aperiodic* policies for dynamic resource management. In periodic resource optimization, the service provider would intervene regularly at fixed time intervals and may adjust the backup deployment at each

[3] Virtual Machine Checkpoints: https://technet.microsoft.com/en-us/library/bb740891.aspx

intervention. This policy leads to two implementation strategies: *conservative* versus *aggressive* backup VM provisioning. Under conservative provisioning, the service provider initially provides a small number of backup VMs when no downtime is incurred yet. As the cumulative downtime increases over time, the service provider becomes more concerned about potential contract violation, and consequently could fine-tune or even increase the number of backups if needed. Under aggressive provisioning, the service provider initially provides a large number of backup VMs to ensure sufficient redundant capacity to cope with future downtime uncertainty. As time goes by, when the service provider becomes less uncertain about future downtime, the level of backup deployment can be fine-tuned or even reduced if warranted. We develop a *closed-loop* (CL) stochastic dynamic optimization model of the conservative strategy and a *certainty-equivalent* (CE) dynamic optimization model of the aggressive strategy.[4] The CL model always produces optimal solutions under all conditions. However, the CE model requires significantly less computational effort than the CL model. Furthermore, the CE model produces near-optimal solutions when (1) the ratio of penalty to backup provisioning cost is relatively low, (2) the failure and recovery processes are less predictable [5], and (3) the availability level required in the SLA is high. Hence, the CE model is recommended due to its significant computational effectiveness when these conditions are prevalent.

While periodic resource management focuses on how much adjustment to the backup VM deployment should be made when the number and timing of such adjustments are fixed, aperiodic resource management focuses on the quanta of VM adjustments, number of such adjustments and their timing all at once. Flexible timing of adjustments would bring a number of benefits, such as cost savings from less frequent assessments and adaptations than fixed interventions. However, the underlying optimization may be computationally overwhelming. Accordingly, we develop a tractable approach to aperiodic optimization by categorizing such policies along two dimensions: interventions and look-aheads. A *look-ahead* is the determination of the expected performance over the remaining part of the contract period after an intervention. This optimization occurs recursively over the continuum of time, rolling over from intervention to intervention. Therefore, such policies could lead to multiple interventions that are unevenly spaced in the contract period. In this research, we develop two aperiodic

[4] The notion of conservatism used here refers to backup VM management; hence, conservative backup VM management is equivalent to aggressive penalty management, and aggressive backup VM management corresponds to conservative penalty management.

[5] In general, the failure and recovery processes of fault-tolerant systems are more predictable than the fault-prone system.

policies denoted as *Single Intervention with Single Look-ahead* (SISL) and *Multiple Interventions with Single Look-ahead* (MISL). Intuitively, as the number of interventions increases, the total costs of aperiodic policies would decrease up to a certain optimal number of interventions, which asymptotically converge to the costs of frequently periodic policies. However, under conditions of (1) greater predictability of the failure and recovery processes, (2) not too stringent availability requirements in an SLA, and (3) small-sized service contracts, aperiodic policies outperform periodic policies. Our extensive computational studies confirm this.

Finally, we develop *online* as well as *offline* implementation algorithms for both periodic (CL and CE) and aperiodic (SISL and MISL) management policies. The offline algorithm of a model solves the problem at the start of the planning horizon and produces solution as a pre-determined control policy regardless of the information revealed during the course of the contract duration. The online algorithm implements the first-step solution of an offline algorithm, resolves the model every time new pieces of information arrive dynamically and yields new control decisions using the new information. We implemented the online algorithms based on use cases constructed from Amazon Elastic Compute Cloud (EC2) with their actual pricing and service credit data. We derived practical guidelines for contract administrators in cloud data centers to cost-effectively manage their virtual infrastructure resources.

The rest of the paper is organized as follows. Section 2 focuses on the research context and summarizes our key contributions. Section 3 reviews the relevant literature. Section 4 develops the periodic policies under conservative resource management and Section 5 under aggressive management. Section 6 develops aperiodic intervention policies under the SISL and MISL resource management. Section 7 presents the computational results and Section 8 further validates our models based on use cases constructed from Amazon EC2 service structures. Section 9 discusses the managerial implications of the proposed resource management policies. Section 10 concludes with directions for future research. The Online Supplement provides more details of our computational study and the proofs to all our theoretical results.

2. Research Context and Contributions

Cloud service providers typically offer a wide range of flexible contracts and pricing mechanisms. Based on a survey of 19 leading cloud service vendors across 27 types of service offerings, Kauffman et al. (2014) classify the state-of-the-art cloud pricing and service provision into two major categories: reservation-based pricing for *reserved services* delivery and usage-based pricing for *on-demand services* delivery. This observation is also supported by large-scale cloud platforms such as Amazon EC2.[6] For example, Amazon offers four classes of EC2 instances: on-demand instances, spot instances, reserved instances and dedicated hosts (which provide EC2 instance capacity on dedicated physical servers). Under on-demand pricing, customers pay for computing capacity by hour without long-term commitment; under spot pricing, customers request spare Amazon EC2 computing capacity for up to 90% off the on-demand price. These two pricing models are suitable for applications with short-term, unpredictable workloads. In contrast, reserved instances and dedicated hosts can be purchased with the commitment of 1 or 3 years contract term. These models are more suitable for applications with relatively stable workloads for dedicated usage. Our proposed virtual infrastructure resource management framework is in the context of IaaS providers' service contracts of reserved instances or dedicated hosts, where pre-committed VM resources are provided to clients in a service contract over a fixed contract period.

We develop a multi-disciplinary approach to study the IaaS virtual infrastructure resource management problem in the cloud. The proposed approach draws upon research from computing resource optimization, algorithm design and statistical estimation of downtime distributions, together leading to the development of effective IT policies for cloud data centers. We model the service downtime allowed in an SLA as perishable commodity in the underlying service period. This leads to inventory-like approaches to dynamic VM management in the cloud. Drawing from the literature on dynamic systems control and online algorithm design, we develop effective VM resource management strategies that are both relevant and easily implementable in cloud data centers. Thus, our modeling approach integrates ideas from Management Science and Computer Science in addressing resource management problems in the cloud IT domain.

The unique contributions of our research are threefold. First, we develop both periodic and aperiodic dynamic decision models for cloud VM provisioning under an SLA. Under the periodic focus, we model both conservative and aggressive approaches to address the question of how many VMs to allocate in a periodic manner over the contract window; and under the aperiodic focus, we develop dynamic approaches to concurrently address the questions of how many VMs to allocate and when to allocate over the contract period of an SLA. In particular, the closed-loop conservative optimization solution can be easily translated into a reference chart to guide backup resource provisioning in real

[6] Amazon EC2 service offerings and pricing: https://aws.amazon.com/ec2/pricing/

time. Second, we design tractable online and offline algorithms to solve for the optimal solutions with reasonable computational effort, which is critical to provide practical decision support for cloud service providers' cost-effective resource allocation decisions. Third, using extensive computational studies we evaluate how the client-level parameters (i.e., service availability and contract duration), service-level parameters (i.e., VM requirements and VM provisioning and penalty costs) and system-level parameters (i.e., mean time between failures and mean time to repair of the VMs) would affect the service provider's optimal resource management. These studies, together with Amazon use cases evaluation, lead to practical insights into the service provider's flexible resource provisioning to minimize its total operational cost and a relevant policy framework to guide SLA contract administrators in their VM resources management under different conditions.

3. Related Literature

We review the literature on cloud pricing and service delivery mechanisms using the classification scheme of Kauffman et al. (2014). Since SLA and resource management differ across the service provisioning models (i.e., reserved or on-demand) and the types of service (i.e., IaaS, PaaS, and SaaS), we present a taxonomy of service models and pricing structures in the online supplement. Based on this taxonomy, we summarize the relevant literature as follows.

Reserved Services Provisioning and Service Contracts. Recent trends in IaaS cloud focus on virtual infrastructure resources including computing, storage and networking that are deployed over a number of VMs in a distributed manner (Chowdhury and Boutaba 2010). Due to virtualization, the cloud infrastructure is more prone to a wide range of hardware and software failures (Gill et al. 2011). Since robust and failure-resilient infrastructure is critical to cloud consumers, infrastructure availability is the most important Quality of Service (QoS) metric in the IaaS cloud SLA for reserved services. A common practice to ensure infrastructure availability is to deploy redundant VMs to increase fault tolerance in the cloud provider's service provision (Qiu et al. 2014). Since full redundancy is costly, a more effective approach is to allocate a set of backup VMs to replace the failed primary VMs as needed (Lu et al. 2012, Xu et al. 2012). Technically, this can be achieved by checkpointing (Goiri et al. 2010). The checkpointing mechanism periodically saves the execution state of a running task (e.g., a VM image file), and enables the task to be resumed from the latest saved state after failure occurs (Du et al. 2015). Zhou et al. (2017) propose a network topology-aware backup VM placement approach to minimizing the consumption of network communication resources when the primary VM failures need to be recovered by backup VMs under the k-fault-tolerance reliability constraints.

In the context of PaaS and SaaS, a broader range of SLAs have been studied for different applications hosted on the cloud infrastructure. Zhao et al. (2015) propose a consumer-centric SLA management framework for cloud-hosted databases. Wang et al. (2008) develop a resource management framework to support multi-tier web applications in shared data centers, which helps to meet different service quality targets at minimum operational cost. Wu et al. (2011) propose resource allocation algorithms for SaaS providers to schedule enterprise applications on VMs that minimize infrastructure cost and SLA violation. Nevertheless, most of these works consider lower-level cloud architectural design to support cloud-based applications, as well as QoS metrics like latency and execution time that affect application delivery (Goudarzi et al. 2012). In contrast, our research focuses on availability-aware infrastructure resource provisioning, which is the most important consideration in IaaS cloud SLAs.

On-Demand Services Provisioning and Service Contracts. Bruneo (2014) studies QoS in IaaS clouds under the on-demand service model. Liu et al. (2015) define quick response time as the key QoS metric in the service contracts. They propose an aggressive VM provisioning strategy to minimize the adaptation time and maintain a high level of QoS of hosted services. Singh et al. (2017) present an SLA-aware autonomic management framework aiming to reduce SLA violation rates for on-demand cloud service delivery.

Since workload uncertainty is a key feature of the on-demand services, dynamic IaaS resource provisioning is an emerging research topic in cloud data center research (Bilal et al. 2014). Substantial amount of research in this area has studied server consolidation, optimal VM placement and sizing to reduce operating cost in the IaaS cloud (Bobroff et al. 2007, Ahmad et al. 2015). Silva et al. (2018) provide a recent survey for the approaches to optimizing VM placement and migration in the cloud environment. Various algorithms and frameworks have been proposed in the literature for this purpose. For example, Laalaoui and Al-Omari (2018) propose an iterative direct move heuristic approach for reassigning VMs into clusters in the IaaS cloud platform. In the context of PaaS/SaaS, Shabeera et al. (2017) develop a metaheuristic algorithm based on ant colony optimization to simultaneously optimize VM placement and location of data for hosting data-intensive applications in the cloud.

IaaS Dynamic Resource Provisioning under Reserved/On-Demand Services. Since most cloud providers offer both reservation and on-demand services, Chase and Niyato (2017) consider that resources can be utilized under the reservation plan, or provisioned under the on-demand plan at a higher rate. They propose joint optimization of VMs and bandwidth allocation to account for the VM over- and under-provisioning risks. Similarly, Ran et al. (2017) present a dynamic instance provisioning strategy by controlling the predicted overload probability of a service below a threshold level to ensure QoS, as well as a reserved instance provisioning strategy for further reducing the total cost. From the IaaS provider's point of view, Mistry et al. (2018) propose a dynamic metaheuristic optimization model to compose long-term service requests under reserved instances, subject to resource and QoS constraints. Different from their approaches that focus on stochastic arrival of requests, we consider the uncertainty involved in the infrastructure availability. Although failure-aware resource management has been studied in the literature (Fu 2010), dynamic backup resource provisioning to fulfil service contracts in the IaaS cloud is an underexplored area of research. We propose an availability-aware cloud resource management framework to fill this research gap.

Other Cloud-Based Business Models. In the PaaS and SaaS environment, a few studies have focused on resource optimization by taking into consideration the cloud consumers' requirements. For example, Liu et al. (2010) study the resource allocation policies for personalization services on content delivery sites. The website trades off the benefit from providing optimal personalized content with long delay and suboptimal content with less waiting time. Johar et al. (2014) propose an optimal control model to determine the size and composition of the firm's offer set to engage a customer. Our work is different from this line of research since we focus on cloud providers' infrastructure resource management rather than cloud consumers' personalized content offerings. We propose a forward-looking, dynamic optimization model for resource provisioning, and incorporate various factors including client-specific, system-specific, and service-specific parameters into the cloud service provider's decision making. We aim to provide higher-level policy recommendations that help guide data center managers to choose the appropriate resource management strategy in the course of their availability-aware SLA execution.

In addition to the operational cost models of resources, pricing models of services have been widely studied in the cloud IT domain. Cheng et al. (2016) show that price heterogeneity exists among different cloud computing providers because of the network latency differentials. Sen et al. (2009) propose a dynamic priority-based price-penalty mechanism for fulfilling the SLA, considering the users' preference variance and their demand fluctuations. In addition to pricing, service contract design has been widely studied in various other contexts such as software outsourcing (Dey et al. 2010), online storage services (Das et al. 2011), and supply chain coordination (Sieke et al. 2012). In the cloud infrastructure, Yuan et al. (2018) study the SLA contract pricing problem based on the tradeoff between resource provisioning cost to ensure availability and penalty cost for potential service downtime. We complement their work by focusing on efficient resource allocation under an SLA to optimize the service provider's resource provisioning decisions in the dynamic cloud environment.

Integrating methods from Operations Research and Computer Science has demonstrated great promise in business problem solving. Technically, our solution approach follows the multi-stage dynamic programming optimization models in Operations Research and online algorithms in Computer Science. To deal with the "curse of dimensionality" (Bellman 1957), characterized as the exponentially increased computational requirements to solve the dynamic programming models as the problem's size increases, we employ techniques such as certainty equivalent models and limited steps of look-ahead models to handle the computational challenges (Bertsekas 1995). In addition, online optimization is crucial to provide practical decision support (Jaillet and Wagner 2012). The assessment of the solution of an online algorithm when compared with the solution of a corresponding ideal offline algorithm which knows the entire input sequence in advance, is termed as competitive analysis (Sleator and Tarjan 1985). We show that the online algorithms developed in this research are competitive and the resource management strategies we propose can be used to support IaaS cloud SLA management in practice.

4. Periodic Resource Optimization: Conservative Strategy

The SLA considered in this research is as follows. Let the client require n VMs over contract duration T with a required level of service availability $\alpha \in (0,1)$. Whenever the number of VMs available is less than n, the overall system of VMs in the contract is said to be *down*, resulting in SLA violation. Hence, the allowed total downtime without any penalty to the service provider by contract is $B = (1 - \alpha)T$. Any downtime incurred in excess of B is denoted as penalizable downtime (Yuan et al. 2018), and the service provider compensates the client at the unit penalty rate π for the penalizable downtime. Linear penalty functions for SLA violations in IaaS have been commonly used in the literature (Mistry et al. 2018). In addition, several variances of such functions (e.g. step penalty function) have been used in practice (e.g., Amazon EC2, Alibaba, Microsoft Azure, Google). Please see the online supplement for details. We further assume the cost of provisioning one VM per unit of time is h.

Without loss of generality, we assume one-to-one mapping of virtual to physical servers for a given client in the data center. While the same physical server may host multiple VMs, the VMs for a given client may be spread across different physical server racks in order to mitigate the SLA violation risk from single point of failure. Using the powered-on checkpointing model, we assume k backup VMs are provided. The total expected cost consists of the provisioning cost of $(n + k)$ VMs over the contract period and the expected penalty of not meeting the availability guarantee. Assuming independent server failures, an SLA violation would occur when more than k servers are concurrently down at a given time. Denoting the duration of SLA violation in a contract period T as *System Downtime,* Du et al. (2015) model the states of the system of $(n + k)$ VMs in terms of the number of VMs that have failed at any time and obtain a birth-death recurrent Markov process over the system states. Using a sample path randomization approach, they derive the transient probability distribution of system downtime. Since system steady-states may not be guaranteed in many practical data center operations, the derivation of the transient distribution is needed. In this research, we adapt the approach by Du et al. (2015) to obtain the transient downtime distribution as input to the proposed optimization models. A summary of the methodology is presented in the accompanying online supplement.

Periodic resource optimization involves decisions on VM deployment at a fixed set of stages in the contract duration T. We denote the decision stages as $\delta = 1,2,\ldots S$, where S is the total number of stages. Although the stages need not be equally spaced in T, for the sake of simplicity in presentation and without loss of generality, we assume that the stages are equally spaced. A system state in stage δ is characterized as a pair (x_δ, k_δ), where x_δ is the total incurred downtime and k_δ is the number of backup VMs at the beginning of stage δ. Furthermore, x_δ ranges in the interval $[0, \frac{(\delta-1)T}{S}]$ and $k_\delta = 0,1,\ldots,K$, where K is the maximum number of backups available for the SLA. Since there is no downtime to begin with stage 1, we have $x_1 = 0$. The state transition equation is $k_{\delta+1} = k_\delta + u_\delta$, where u_δ is the integer decision variable at the beginning of stage δ, and $-k_\delta \le u_\delta \le K - k_\delta$. Therefore, $u_\delta(x_\delta, k_\delta) = \{-k_\delta, (-k_\delta + 1), \ldots, 0,1,2,\ldots, K - k_\delta\}$. These decisions lead to the state transitions from any stage δ to a state in the subsequent stage as illustrated in the left panel of Figure 1.

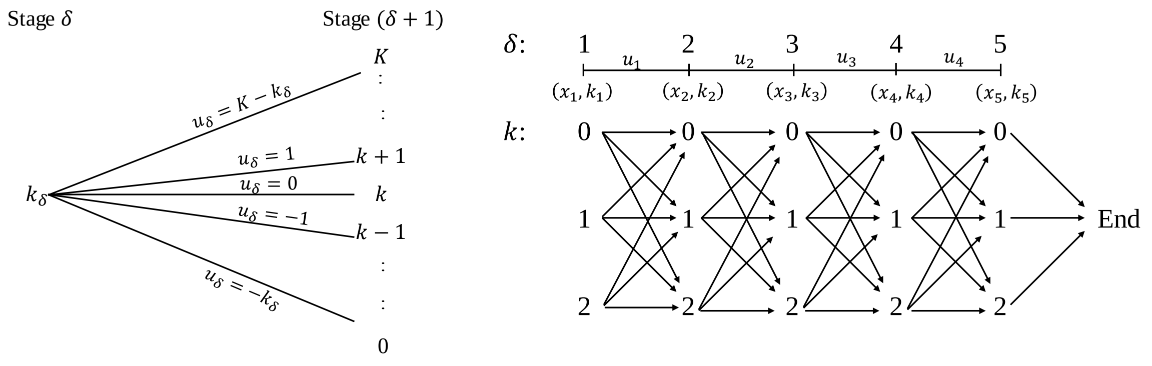


Figure 1. Staged VM Deployment and a 4-Stage Transition Paths Example

To focus on the backup adjustment decision and to simplify our notation, we assume the number of primary backup VMs is fixed at n in our theoretical model development. We denote $D(k)$ as the random downtime incurred in any given stage with k backup VMs, and $d(k)$ the expected downtime in the given stage. Therefore, we have $x_{\delta+1} = x_\delta + D(k_\delta + u_\delta)$. The right panel of Figure 1 illustrates a 4-stage problem with $K = 2$. The top of this panel illustrates the initial states and controls in each stage, where stage 5 is the ending stage at which no further cost is incurred and hence, no control is necessary. The bottom of this panel shows the full set of transition paths.

Denote $J_\delta(x_\delta, k_\delta)$ as the *cost-to-go function*, which is the minimum expected total cost from stage δ to the end of stage S, which is the end of the horizon. The terminal cost incurred at the end of the horizon is $J_{S+1}(x_{S+1}, k_{S+1}) = 0$. The dynamic stochastic optimization problem at any stage δ, where $\delta = 1, \ldots, S$ and u_δ the control decision is as follows:

Problem [CL]

$$J_\delta(x_\delta, k_\delta) = \min\left[h(k_\delta + u_\delta)\frac{T}{S} + E[\pi \max\{0, x_\delta + D(k_\delta + u_\delta) - \max(B, x_\delta)\}] + J_{\delta+1}(x_{\delta+1}, k_{\delta+1})\right]$$

$$s.t.\ x_{\delta+1} = x_\delta + D(k_\delta + u_\delta),\ \ k_{\delta+1} = k_\delta + u_\delta,\ \ -k_\delta \le u_\delta \le K - k_\delta, \text{ and } u_\delta \text{ is integer}$$

The cost-to-go function $J_\delta(x_\delta, k_\delta)$ minimizes the expected total cost in the current stage δ plus the minimum expected total cost from stage $\delta + 1$ to the end of the horizon $J_{\delta+1}(x_{\delta+1}, k_{\delta+1})$. The first term in the minimization function is the backup provisioning cost in stage δ. The second term is the expected penalty cost in stage δ. Note that if $x_\delta > B$, then the penalizable downtime incurred in stage δ is $D(k_\delta + u_\delta)$. If $x_\delta \le B$, then the penalizable downtime in stage δ is $\max\{0, x_\delta + D(k_\delta + u_\delta) - B\}$.

Closed-loop (CL) optimization derives a control policy that depends on the current state information on the system. Accordingly, the closed-loop solution to Problem [CL] yields rules for choosing u_δ for each stage δ with knowledge of the current level of downtime incurred x_δ and the current level of backup provision k_δ. Since there is no closed-form solution to characterize the closed-loop strategy, we develop an efficient algorithm to solve for the optimal state-contingent strategies. Building upon the approach proposed by Du et al. (2015) to estimate transient system downtime, Section 1 of the online supplement details our strategy to obtain the empirical estimation of system downtime in our current context. For any given (n, k) configuration, define $f_k(\tau)$ and $F_k(\tau)$ as the density function and cumulative distribution function of incurring τ downtime over the contract period T, and η_k as the mean percentage of downtime incurred over the contract period. We next show that the downtime distribution satisfies the following properties:

Lemma 1. *If the number of backup VMs $k_1 < k_2$, then the downtime distribution has the following properties: (i) $F_{k_1}(\tau) \le F_{k_2}(\tau)$; that is, $F_{k_1}(\tau)$ first-order stochastically dominates $F_{k_2}(\tau)$; (ii) $\int_0^\tau [F_{k_2}(t) - F_{k_1}(t)]dt \ge 0$; that* *is,* $F_{k_1}(\tau)$ *second-order stochastically dominates* $F_{k_2}(\tau)$*; (iii) The mean percentage of downtime* $\eta_{k_1} > \eta_{k_2}$.

These distributional properties are important to guide our numerical optimization in Section 6. In particular, Lemma 1(iii) implies that all else being equal, the mean percentage of downtime (or equivalently, uptime) over a contract period decreases (increases) as the service provider increases the number of backup resources.

4.1 Closed-Loop Optimization

Figure 2 presents a backward-recursive algorithm to solve the optimization problem [CL] and derive the closed-loop solution. Using the characterization of decision stages and the states within each stage as described above, the algorithm finds the stage-wise, state-dependent cost function $J_\delta^*(x_\delta, k_\delta)$ and the optimal decision $u_\delta^*(x_\delta, k_\delta)$. These functions are computed recursively backward in time, starting from stage S and ending at stage 1. The optimal expected cost is given by the last step of the algorithm.

```
Input:     SLA contract parameters: contract duration $T$, guaranteed service level $\alpha$, unit provisioning cost $h$,
           unit penalty cost $\pi$;
           System parameters: Number of primary VMs $n$, server mean time between failures $MTBF$, server
           mean time to repair $MTTR$;
           User choices: maximum number of stages $S$, maximum number of backup VMs $K$, discretization
           interval $\Delta t$
Output:    Minimum expected cost-to-go $J_\delta^*(x_\delta, k_\delta)$ and optimal policy $u_\delta^*(x_\delta, k_\delta)$ for $\delta = 1, \ldots, S$
Let $W = \frac{T}{S}$;
Call EmpDistribution($n,k,W$) for $k = 0,1, \ldots, K$; /* see online supplement for the EmpDistribution procedure */
    Get downtime probability distribution $p_w(i, n, k)$, for $i = 1, \ldots, \left\lceil \frac{W}{\Delta t} \right\rceil$;
Let $z_\delta^{max} = (\delta - 1)\left\lceil \frac{W}{\Delta t} \right\rceil$ for $\delta = 1, \ldots, S$;
Let $k_\delta = 0, \ldots, K$ for $\delta = 1, \ldots, S$;
Begin
           $\delta = S$;
           $J_{S+1}^*(x_{S+1}, k_{S+1}) = 0$;
           Repeat
                Let $z_\delta = \{0,1, \ldots, z_\delta^{max}\}$ and denote $x_\delta = z_\delta \Delta t$ as the cumulated downtime at the beginning
                    of stage $\delta$
                For each $(x_\delta, k_\delta)$, solve Problem [CL] for $u_\delta = -k_\delta, \ldots, K - k_\delta$;
                Let $u_\delta^*(x_\delta, k_\delta)$ denote the optimal solution to [CL]
                Let $J_\delta^*(x_\delta, k_\delta)$ denote the minimum expected cost
                $\delta = \delta - 1$;
           Until $\delta = 0$
End
```

Figure 2. Closed-Loop Optimization Algorithm

4.2 Online Solution Implementation

In many multi-stage decision problems, decisions at any given stage may have to be made either with incomplete knowledge of the future or under not very reliable distributional assumptions on the future (Bertsekas 1995). In such cases, online optimization should be used (Jaillet and Wagner 2012). An online algorithm resolves the decision model based on sequentially arriving new information on the system behavior as it evolves over time. The closed-loop resource provisioning strategy presented above is in essence an online algorithm implementation, where the realized downtime serves as input in a piece-by-piece serial fashion as the SLA contract evolves over time. Based on the observed downtime information, different state-contingent control decisions will be made. Therefore, by design, the proposed CL optimization model provides the online solutions. The following example shows the online implementation of the closed-loop solution.

Assume the number of primary VMs $n = 100$, and the number of backup VMs can vary from $k = 0,1, \ldots 3$. Let the service window $T = 120$. Let the mean time between failures and mean time to repair of a VM be as follows: $MTBF = 2{,}400$, and $MTTR = 20$. Using these parameters, we generated 5,000 samples with a discrete grid $\Delta t = 0.1$ to derive the empirical downtime distribution. Further assume the cost of provisioning 1 VM per unit of time is $h = 1$, the penalty per unit of time is $\pi = 100$, and the SLA service availability requirement $\alpha = 90\%$. Now, consider a 4-stage decision problem with this data. Figure 3 shows the optimal transition paths based on the closed-loop solution to this problem. The nodes denote decisions on the number of backup VMs at each stage, and the associated conditions on x_δ, $\delta = 1,2,3,4$, are labeled on the edges.

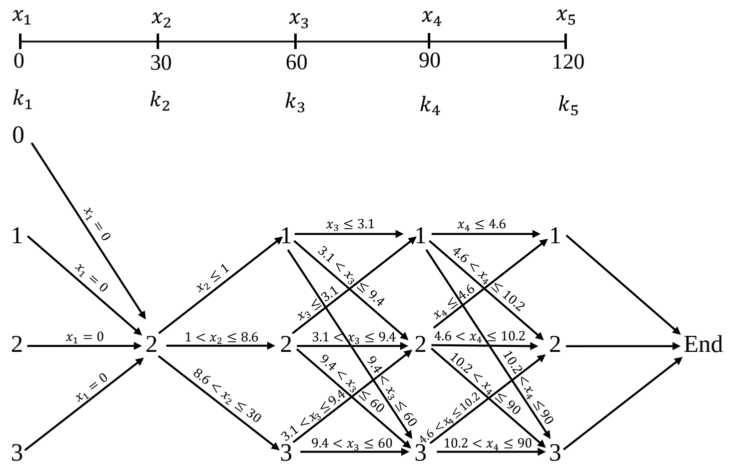


Figure 3. Optimal Transition Paths of Dynamic Resource Provisioning

Note that the length of each stage is 30, since $T = 120$, and $S = 4$. Since there is no downtime to begin with, we have $x_1 = 0$. The optimal closed-loop decision is to provide 2 backup VMs in the first stage. Given an initial allocation k_1, the service provider will then adjust the provision accordingly. For example, if the initial allocation is 2 backups, then the service provider will not make any adjustment. If the initial allocation is 3 backups, then the service provider will remove 1 backup VM.

The adjustment decision in the second stage will depend on the realized downtime in the first stage. As per the closed-loop solution, if the realized downtime is low $(x_2 \le 1)$, then the optimal decision is to remove 1 backup VM in the second stage. If the realized downtime is medium ($1 < x_2 \leq 8.6$), then the optimal decision is to keep 2 backup VMs in the second stage. If the realized downtime in the first stage is high ($8.6 < x_2 \leq 30$), then the optimal decision is to increase to 3 backup VMs. Decisions for other stages can be interpreted in the same way. Figure 4 translates the optimal decision rules into a resource provisioning reference chart based on the observed state conditions. The horizontal axis shows the possible range of values for $x_2 \in [0,30]$, $x_3 \in [0,60]$, and $x_4 \in [0,90]$. The vertical axis shows the optimal number of backup VMs.

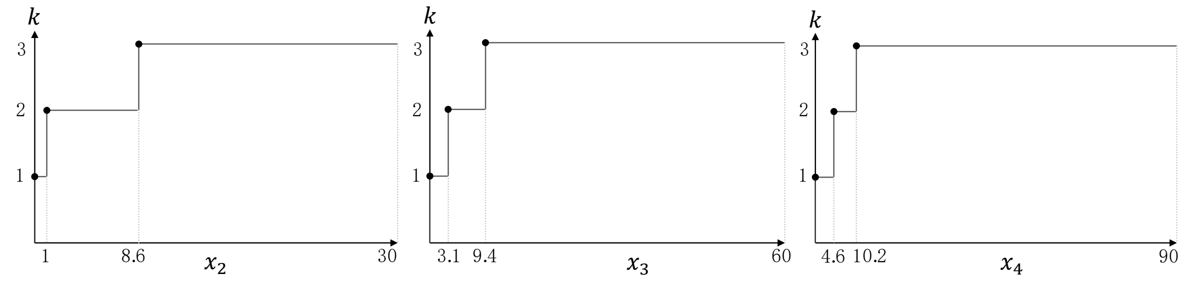


Figure 4. Dynamic Resource Provisioning Reference Chart

Clearly, the closed-loop solution yields an easy reference chart to the service provider for making decisions on VM allocation at each stage after observing the actual accrued downtime. For example, at $t = 90$, the service provider needs to make the fourth stage decision. Using the third panel of Figure 4, the following decisions can easily be identified for this stage: If $x_4 \leq 4.6$, then 1 backup VM is needed; if $4.6 < x_4 \leq 10.2$, then 2 backups are needed; and if $x_4 \geq 10.2$, then 3 backups are necessary.

Clearly, our closed-loop dynamic optimization model finds the state-dependent optimal resource provisioning policies, based on which practical reference charts as in Figure 4 can be created. As time goes by, the service provider just needs to monitor the incurred downtime in real-time and make optimal resource adjustment decisions using the reference charts.

Because the service provider does not have full information about the future, the online algorithm may not perform as well as an offline algorithm which knows the entire sequence of information in advance and responds optimally. The *competitive ratio* is defined as the maximum of the ratio between the cost incurred by the online algorithm and that of an ideal offline algorithm over all possible input sequences (Sleator and Tarjan 1985). It is also the worst-case performance ratio between the online algorithm and an optimal offline algorithm. We establish the following competitive ratio bound based on worst-case scenario analysis. The bound shows that the closed-loop optimization is competitive.

Proposition 1. *The competitive ratio for the closed-loop optimization algorithm r_{CL} is bounded by $r_{CL} <$*

$1+\frac{K}{n}<2$.

5. Periodic Resource Optimization: Aggressive Strategy

The above closed-loop optimization follows a conservative approach to VM provisioning; it begins with low level of backup VM provisioning and subsequently increases or adjusts the allocation as actual downtimes are successively realized over the stages. On the other hand, an aggressive approach would involve a sufficient backup provisioning to begin with, and subsequently decrease or adjust the level of backup VMs as actual downtimes are realized over time. Since the chances of contract violation under the aggressive strategy are significantly less than the conservative strategy, an expected value analysis of the stochastic downtime may be sufficient for the underlying optimization problem. We thus develop an online Certainty-Equivalent (CE) optimization model using the mean value of downtime. The CE model serves as both an approximation to the computationally extensive full stochastic CL model and an implementation of the aggressive strategy. While closed-loop optimization is purely an online algorithm, certainty-equivalent optimization has both online and offline counterparts. In the following discussion, we first present the offline model, and subsequently develop the online approach.

5.1. Offline Certainty-Equivalent Optimization

Certainty equivalent control applies at each stage the control that would be optimal if the uncertain downtime were fixed at some "typical" values, such as the mean values. It solves a deterministic optimal control problem at each stage. The advantage of the CE model is that the potentially expensive determination of the expected cost is replaced by the calculation of single stage-control trajectory. To develop this model, we define the following variables:

$G_\delta(k_\delta) =$ The minimum expected total cost from state k_δ of stage δ till the end

$\Omega_\delta(k_\delta) =$ Total expected downtime from state k_δ of stage δ till the end along the path corresponding to $G_\delta(k_\delta)$

We present the optimization model underlying the certainty-equivalent approach as follows.

Problem [CE]:

$$G_\delta(k_\delta) = \min\Big[h\frac{T}{S}(k_\delta+u_\delta) + \pi\max\{0, d(k_\delta+u_\delta) + \Omega_{\delta+1}(k_\delta+u_\delta) - \max\{B, \Omega_{\delta+1}(k_\delta+u_\delta)\}\} + G_{\delta+1}(k_\delta+u_\delta)\Big]$$

s.t. $k_{\delta+1} = k_\delta + u_\delta$, $-k_\delta \le u_\delta \le K - k_\delta$, and u_δ is integer.

The cost-to-go function in Problem [CE] is similar to that of Problem [CL]. However, there are several differences. First, the state variable in [CE] is (k_δ), rather than (x_δ, k_δ). Second, the cost computation is simplified. The expectation outside of the max function in [CL] is replaced by $d(k_\delta + u_\delta)$ and $\Omega_{\delta+1}(k_\delta + u_\delta)$, which are mean expected downtimes. These simplifications significantly reduce the computation time of the CE model. Compared with the CL model, the CE model could be suboptimal, but is computationally more tractable because of the dramatically reduced state-space of the dynamic programming model and hence, the computational effort. Furthermore, because CE model uses the mean value of the random downtime, it is a deterministic optimization. We identify a unique path that fully characterizes the resource provisioning and adjustment strategy. This is the offline solution. Proposition 2 shows that the offline CE solution is not unique, and the provision of backup VMs is not stage-dependent in an offline implementation.

Proposition 2. *If the optimal offline CE solution yields the backup provision sequence* $(k_1, k_2, \dots, k_S)$, *then any permutation of the sequence also provides an optimal offline backup provision.*

The intuition of Proposition 2 is as follows. Note that the deterministic downtime in stage δ is uniquely determined by the number of backup VMs in that stage. In the offline implementation, since the service provider only implements a fixed sequence of controls, the order of the sequence does not matter in terms of total cost minimization. Therefore, any permutation of the sequence yields the same provisioning cost and penalty cost in the deterministic environment. To support an aggressive resource provisioning strategy, the CE solution would pick a decreasing backup provisioning sequence. This high initial number of backup VMs will minimize the occurrence of downtime at beginning stages of the contract period. The service provider will then reduce the number of backup VMs in later stages of the contract period if the actual incurred downtime is low.

The offline CE solution is an *open-loop* solution without knowledge of the future states, thus it may not be optimal. However in practice, the service provider could simply choose to implement only the control for the first stage at the beginning of the contract period. As stages arrive over time, the available non-penalizable downtime B can be updated with the observed actual downtime and the CE model can be re-solved with this updated information at every stage. This leads to online CE optimization, and is presented below.

5.2. Online Certainty-Equivalent Optimization

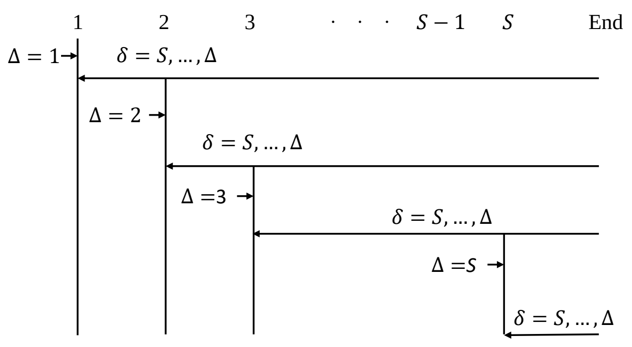
The online CE algorithm is presented in Figure 5. This algorithm consists of two passes: a forward pass and a backward pass. The forward pass constitutes the outer loop and the backward pass constitutes the inner loop of the algorithm. The evaluation strategy is as follows. Consider any time in the contract period where stages $1, \dots, \Delta - 1$ have been realized, and we are at stage Δ to decide upon the allocation. Let C_Δ denote the accrued total realized downtime from stages 1 to stage Δ. The forward-passing outer loop starts with $\Delta = 1$. At each realization of stages, Δ advances to $\Delta + 1$ until $\Delta = S + 1$ where it ends. At each advance of the outer loop, a backward-passing inner loop is performed. This inner loop starts from $\delta = S$, and ends at $\delta = \Delta$. The inner loop is embedded within the outer loop, as illustrated in Figure 6. The solution from the inner loop is a control trajectory $(u_\Delta^*, \dots, u_S^*)$. But only u_Δ^* is adopted in the decision making at stage Δ. In stage $\Delta + 1$, the actual downtime till then is observed, the allowable downtime $\hat{B}$ is updated, and another CE optimization problem is solved using the inner loop.

```
Outer Loop
Begin
    Set $\Delta \leftarrow 1$; $C_\Delta \leftarrow 0$; $k_\Delta \leftarrow k^*$;
    Repeat
    Run the Inner Loop;
    Allocate $(k_\Delta + u_\Delta^*)$ backup servers for stage $\Delta$;
    Observe actual downtime in stage $\Delta$ with this decision and denote it as $\hat{X}_\Delta(k_\Delta + u_\Delta^*)$;
    $C_{\Delta+1} \leftarrow C_\Delta + \hat{X}(k_\Delta + u_\Delta^*)$;
    $k_\Delta \leftarrow k_\Delta + u_\Delta^*$;
    $\Delta \leftarrow \Delta + 1$;
    Until $\Delta = S + 1$;
End

Inner Loop
Begin
    $\hat{B} = Max\{0, B - C_\Delta\}$ Allowable total downtime at the beginning of stage $\Delta$;
    $\Omega_S(k), k = 0, \dots, K$;
    $G_S(k) = h\frac{T}{S}k + \pi Max\{0, \Omega_S(k) - \hat{B}\}, k = 0, \dots, K$;
    $\delta = S - 1$;
    Repeat
        $B = \hat{B}$;
        Solve for Problem [CE];
        Let $u_\delta^*$ denote the optimal solution to [CE];
        $\Omega_\delta(k) = d(k + u_\delta^*) + \Omega_{\delta+1}(k + u_\delta^*), k = 0, \dots, K$;
        $\delta = \delta - 1$;
    Until $\delta = \Delta - 1$.
End
```

Figure 5. Online Certainty-Equivalent Optimization Algorithm

Two versions of CE optimization are possible. The first version is a simple *offline* CE solution obtained from a single complete backward pass from stage S to stage 1 of the inner loop in Figure 5. The second version is the online solution presented in Figure 6.



1
2
3
· · ·
$S-1$
S
End
$\Delta = 1$
$\delta = S, \ldots, \Delta$
$\Delta = 2$
$\delta = S, \ldots, \Delta$
$\Delta = 3$
$\delta = S, \ldots, \Delta$
$\Delta = S$
$\delta = S, \ldots, \Delta$

Figure 6. Online Certainty-Equivalent Optimization Strategy

Since the CE model does not assume any uncertainty, it identifies a fixed control trajectory. The full set of staged decisions in the offline model is implemented right at the beginning of the contract period. In contrast, the online CE solution could achieve better performance because the decisions are made in a stage-wise manner over time, after observing the actual downtime at each stage and updating the non-penalizable downtime accordingly. Similar to the CL optimization model, we derive the competitive ratio for the online CE algorithm as shown in Proposition 3.

Proposition 3. *The competitive ratio for the online CE algorithm is no lower than that under the CL optimization algorithm and it is bounded by* $r_{CL} \leq r_{CE} < 1 + \frac{K}{n} < 2$.

Although the competitive ratio under the CE algorithm is greater than or equal to that under the CL optimization, both algorithms are competitive since their competitive ratios are bounded by 2.

6. Aperiodic Resource Optimization: Flexible Interventions

An *intervention* pertains to the action of adjusting the backup allocation at any time in the contract period. In periodic optimization, an intervention corresponds to a decision stage whose timing is pre-fixed; contrarily, in aperiodic resource optimization, we allow the time of intervention to be flexible and determined through cost optimization rather than following a set schedule. Accordingly, the decision problem involves two concurrent decisions: *When* to intervene next and *how much* backup to allocate at each intervention. This involves a triad of decisions at an intervention: the backup allocation from the current time till the next intervention, the time of next intervention, and the new backup allocation after the intervention time. The underlying optimization requires an estimation of the expected total cost from the current intervention time till the end of the horizon. We term the estimation of the cost from the next intervention time till the end as the *look-ahead*. The service provider may perform single or multiple interventions in the contract period. If the choice is to perform just a single intervention, then there will be just one look-ahead period. This is termed as the Single Intervention with Single Look-ahead (SISL) strategy. If the choice is to perform multiple interventions, then we could either employ just a single look-ahead period or recursively use multiple look-ahead in estimating the post-intervention costs at each intervention. We term these as Multiple Interventions with Single Look-ahead (MISL) and Multiple Interventions with Multiple Look-aheads (MIML) strategies, respectively. MIML is inherently more complex because a flexible intervention time has to be determined for each look-ahead period. Determining the intervention time for each future period needs the knowledge of the intervention time in the current period, which is unknown at the time of decision making. So we only focus on SISL and MISL. Our objective is to solve an aperiodic optimization problem that is simpler and computationally tractable.

6.1. Single-Intervention with Single Look-Ahead (SISL) Strategy

Assume the service provider only intervenes once over the entire contract period. Denote the intervention time $t \in [0,T]$, and the number of backups before and after the intervention time is k_1 and k_2, respectively. Denote $D_t(k_1)$ and $D_{T-t}(k_2)$ as the random downtime occurred before and after the intervention. The SISL decision problem is to determine an optimal intervention time t^* and the optimal resource provisioning (k_1^*, k_2^*) to minimize the expected cost function $C(k_1, k_2, t)$:

Problem [SISL]

$$\min_{k_1,k_2,t} C(k_1,k_2,t) = hk_1 t + hk_2(W-t) + E[\pi \max\{0, D_t(k_1) + D_{T-t}(k_2) - B\}]$$

The objective function minimizes the expected total cost over the two stages: the first two terms are the backup provisioning cost before and after the intervention, and the third term is the expected penalty cost over the two stages. Next we examine some properties of the expected cost function.

Lemma 2. *The expected cost function of SISL is symmetric: $C(k_1,k_2,t) = C(k_2,k_1,T-t)$.*

Lemma 2 is intuitive. Because the downtime of the two stages is independent, we can simply switch the two stages and the symmetric property holds. Also, the random downtime incurred over the period T is separable. Define $D_{t,T-t}(k_1,k_2) = D_t(k_1) + D_{T-t}(k_2)$ and denote the expected downtime as $d_{t,T-t}(k_1,k_2)$. We have the following property.

Lemma 3. *For given (k_1,k_2), the mean downtime $d_{t,T-t}(k_1,k_2) = T\eta_{k_2} + (\eta_{k_1} - \eta_{k_2})t$.*

Immediately from Lemma 3, we know that the expected downtime with one intervention is a linear function of t. By Lemma 1, $\eta_{k_1} > \eta_{k_2}$ if $k_1 < k_2$, so the mean downtime linearly increases in t. Similarly, the mean downtime linearly decreases in t if $k_1 > k_2$, and is constant over t if $k_1 = k_2$. Without loss of generality, we next focus on the case $k_1 < k_2$ by assuming the service provider adopt a conservative strategy. The case $k_1 > k_2$ is the mirror case of $k_1 < k_2$.

Lemma 4. *If $k_1 < k_2$, then the expected penalizable downtime $E\left[max[0, D_{t,T-t}(k_1,k_2) - B]\right]$ is either linearly increasing in t or convex in t.*

Define $\Phi_k(\tau)$ as the normalized cumulative downtime distribution function when k backups are provided, where τ is interpreted as percentage of downtime incurred over the contract period rather than the real downtime. Based on Lemma 4, the following Proposition shows that the cost function is well-behaved in t and the unique optimal solution exists.

Proposition 4. *If $k_1 < k_2$, then the expected cost function $C(k_1, k_2, t)$ has the following properties:*

(i) *If $\Phi_{k_1}(1-\alpha) = 1$, then $C(k_1,k_2,t)$ is a linear and strictly decreasing function of t;*

(ii) *If $\Phi_{k_2}(1-\alpha) < 1$, then $C(k_1,k_2,t)$ is a linear and strictly increasing function of t If $\frac{\pi}{h} > \frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$; otherwise, $C(k_1,k_2,t)$ is a linear and strictly decreasing function of t if $\frac{\pi}{h} < \frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$.*

(iii) *If $\Phi_{k_2}(1-\alpha) = 1$ and $\Phi_{k_1}(1-\alpha) < 1$, then $C(k_1,k_2,t)$ is convex in t.*

Note that when $t = 0$, the total number of backups is k_2, and when $t = T$, the total number of backups is k_1. There are three cases. Proposition 4(i) shows that, if no penalty would be incurred when k_1 backups are provided over the entire time period T, or equivalently, k_1 is so high such that all occurrences of its normalized percentage of downtime are below the critical threshold $1-\alpha$ (the same holds for k_2), then the expected penalizable downtime is always 0 as t increases. The total expected cost only consists of the resource provisioning cost, which linearly increases in k. So as t increases, the total provisioning cost decreases.

Proposition 4(ii) shows that, if k_2 is not large enough such that there is positive probability that the normalized percentage of downtime in some cases would exceed the critical value $1-\alpha$ (the same holds for k_1), then the expected penalizable downtime is linear in t and the expected total cost is also linear in t in this case. If the per unit penalty cost is relatively more expensive than the per unit provisioning cost such that the condition $\frac{\pi}{h} > \frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$ is satisfied, then the expected penalty cost dominates the provisioning cost and thus the total cost increases in t. Otherwise, if the provisioning cost is relatively higher than the penalty cost, the total cost decreases in t.

Proposition 4(iii) shows the scenario where k_2 is large enough so that no penalty would be incurred if k_2 backups are provided over the entire time period T, but k_1 is not large enough so that some penalty would be incurred if k_1 backups are provided over the entire time period T. Since the provisioning cost decreases in t whereas the expected penalty is convex and increases in t under this case, the expected total cost is convex in t. There exists an optimal intervention time to trade off the total provisioning cost against the expected penalty cost. In sum, the optimal intervention time can be identified as follows.

Proposition 5. *Assume $k_1 < k_2$. The optimal intervention time occurs at:*

(i) $t^* = 0$ *if* $\Phi_{k_2}(1-\alpha) < 1$ *and* $\frac{\pi}{h} > \frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$;

(ii) $t^* = \tilde{t}$ *if* $\Phi_{k_2}(1-\alpha) = 1$, $\Phi_{k_1}(1-\alpha) < 1$, *and* $\frac{\pi}{h} > \frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$;

(iii) $t^* = T$ *if any of the following cases holds: a)* $\Phi_{k_1}(1-\alpha) = 1$; *b)* $\Phi_{k_2}(1-\alpha) < 1$ *and* $\frac{\pi}{h} < \frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$ *; or c)* $\Phi_{k_2}(1-\alpha) = 1$, $\Phi_{k_1}(1-\alpha) < 1$, *and* $\frac{\pi}{h} < \frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$.

Note that the threshold value $\frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$ is determined by the backup provisioning choices, their corresponding downtime distributions, as well as the availability requirement, but is independent of the penalty to provisioning cost ratio $\frac{\pi}{h}$. All else being equal, if per unit penalty is likely more expensive than the per unit of resource provision, then the service provider tends to intervene early to provide higher number of backups and avoid possible penalty cost.

When providing k_1 backups over the entire contract period T would incur penalty, but providing k_2 backups would not, then the optimal intervention time $\tilde{t}$ is the time when the combined downtime variable $D_{t,T-t}(k_1,k_2) = D_t(k_1) + D_{T-t}(k_2)$ just starts to have non-zero probability of incurring penalizable downtime. The condition $\frac{\pi}{h} > \frac{k_2-k_1}{\eta_{k_1}-\eta_{k_2}+\int_0^{1-\alpha}[\Phi_{k_1}(\tau)-\Phi_{k_2}(\tau)]d\tau}$ suggests that, to warrant the intervention, the savings from the reduced downtime as well as the penalty cost should be larger than the additional provisioning cost.

Under the conditions specified in Proposition 5, $t^* = 0$ and $t^* = T$ can be easily identified. Given (k_1, k_2), it might be optimal to keep the number of backup VMs constant over the entire period. When the cost function $C(k_1, k_2, t)$ is convex in t, the optimal intervention time occurs at $\tilde{t}$, where $\tilde{t}$ is defined in the online supplement. Although we can theoretically characterize this time, we still need to design an efficient algorithm to empirically search for it because the intervention time is affected by the empirical downtime distribution, which is a function of the number of backup VMs provided.

6.2. The SISL Algorithm

When the empirical downtime distribution satisfies conditions specified in Proposition 5(ii), we need to search for the optimal intervention time. This could involve significant computational effort as the intervention time is a continuous decision variable. We propose an efficient, offline, two-step approach to solve this problem as shown in Figure 7. The inner loop employs a search algorithm to find the optimal solution $C(k_1,k_2,t^*)$ for any given (k_1,k_2) combinations. The outer loop enumerates the limited number of (k_1,k_2) combinations to find the global optimal $C(k_1^*,k_2^*,t^*)$.

```
Input: maximum number of backups $K$ and contract duration $T$;
Output: $I(k_1^*,k_2^*,t^*)$ and corresponding total cost $TC$;
Main: SISL($T$)
Begin
    $I = (0,0,0)$;
    $TC = G$, where $G$ is a very large number;
    For $k = 0,\ldots,K$, do
        Compute $C(k,k,T) = hkT + E[\pi \max\{0, D_T(k) - B\}]$;
        If $C(k,k,T) < TC$ then
            $TC = C(k,k,T)$;
            $I = (k,k,T)$;
        End If
    End
    For $k_1 = 0,\ldots,K-1$, do
        For $k_2 = k_1+1,\ldots,K$, do
            If conditions in Proposition 5($i$) are satisfied
                $t^* = 0$;
                $C(k_1,k_2,t^*)$;
            Elseif conditions in Proposition 5($iii$) are satisfied
                $t^* = T$;
                $C(k_1,k_2,t^*)$;
            Else
                Call Search ($k_1$,$k_2$,$T$); /*the search algorithm is executed under conditions in Proposition 5($ii$)*/
            End
            If $C(k_1,k_2,t^*) < TC$ then
                $TC = C(k_1,k_2,t^*)$;
                $I = (k_1,k_2,t^*)$;
            End If
        End
    End
End
```

Figure 7. Offline SISL Algorithm

Note that there are $(K+1)^2$ possible combinations of the (k_1,k_2) pairs. However, the $(K+1)$ combinations where $k_1 = k_2$ can be directly solved without any intervention. This is the first part of the algorithm in Figure 7. Due to the symmetric property demonstrated in Lemma 2, we can further reduce the search by half its size. Thus, the total number of combinations in search is at most $\frac{(K+1)^2-(K+1)}{2} = \frac{K(K+1)}{2}$. This is the second half of the SISL algorithm in Figure 7.

The offline SISL algorithm employs a module called Search (k_1,k_2,T), where we propose a *ternary partition search algorithm* to numerically search for the optimal intervention time when conditions in Proposition 5(ii) are satisfied. Please refer to the online supplement for details. The online SISL algorithm would outperform the offline SISL algorithm due to the value of information. It follows the offline algorithm to implement k_1 backups in stage 1 until the intervention time t_1. At time t_1, the total realized downtime $\tau = C_1$ is observed. The online algorithm then adjusts the allowable downtime $\hat{B} = \max[0, (1-\alpha)T - C_1]$, based on which k_2 is re-optimized assuming there is no more adjustment in the remaining $(T - t)$ period. For every realized downtime C_1, we obtain the minimum cost $C(k_2|k_1, C_1)$. The expected cost of the online SISL algorithm is computed as $\int_0^{t_1} C(k_2|k_1, C_1) f_{k_1}(\tau)\, d\tau$.

6.3. Multiple Interventions with Single Look-Ahead (MISL) Strategy

Under MISL, the optimal cost-to-go is approximated by assuming there is only one intervention opportunity in the remaining time period. Starting from $\delta = 1$, the MISL strategy repeats the following steps: (1) At the beginning of stage δ (the last intervention time $t_{\delta-1}$), the service provider determines $(k_\delta, t_\delta, k_{\delta+1})$ to minimize the expected cost over the contract horizon $T - t_{\delta-1}$, assuming there is one intervention in the remaining time period, based on which the service provider implements k_δ till the next intervention time t_δ. (2) If $t_\delta = T$, following the decision k_δ till the end of contract period; otherwise, following the decision k_δ till t_δ, $\delta = \delta + 1$, and repeat (1). Note that the original decision $k_{\delta+1}$ from the previous stage may not be followed as new decisions about backup resource provision are made at times t_δ. Figure 8 graphically illustrates the decision sequence.

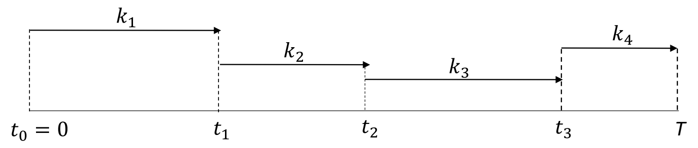


Figure 8. Diagram of MISL Strategy

Problem [MISL]

$$H_\delta(x_\delta, k_\delta) = \min[hk_\delta(T - t_{\delta-1}) + hu_\delta(T - t_{\delta-1} - t)$$
$$+ E[\pi \max\{0, x_\delta + D(k_\delta + u_\delta) - \max(B, x_\delta)\} + H_{\delta-1}(x_{\delta-1}, k_{\delta-1})]]$$

$s.t.\ x_{\delta+1} = x_\delta + d(k_\delta + u_\delta)$, $k_{\delta+1} = k_\delta + u_\delta$; $-k_\delta \le u_\delta \le K - k_\delta$, and u_δ is integer

Here $H_\delta(x_\delta, k_\delta)$ is the cost-to-go function and $T - t_{\delta-1}$ is the remaining contract window at the beginning of stage δ. $H_{\delta-1}(x_{\delta-1}, k_{\delta-1})$ is the cost incurred up to stage $\delta - 1$ and the initial cost $H_0(x_0, k_0) = 0$. This optimization problem follows the same interpretation of the [CL] problem. The difference is that now the time of intervention is also a decision variable and we use a forward strategy to solve the problem. The full MISL model implementation is thus more challenging than CL model because it not only depends on the observed downtime information, but needs to search for the optimal intervention times.

Figure 9 shows the offline MISL algorithm, which calls for a series of SISL optimizations until no more intervention can be found. The online MISL algorithm involves an update of B that is based on the realized downtime observed at the intervention time, rather than the expected downtime as in Figure 9. Consider the example in Figure 8. The online algorithm follows the offline MISL solution (k_1, t_1) to implement k_1 backups and to intervene at t_1. At t_1, following the downtime distribution $F_{k_1}(\tau)$, the total realized downtime $\tau = C_1$ is observed. The online algorithm then adjusts the allowable downtime $\hat{B} = \max[0, (1-\alpha)T - C_1]$, and resolves the offline MISL algorithm in the remaining period $(T - t_1)$ to yield an optimal first-stage solution (k_2, t_2, k_3) and minimum expected cost $C(k_2, t_2, k_3 | k_1, C_1)$, where only the first part of the solution (k_2, t_2) will be implemented, and so on.

```
**Input:** maximum number of backups $K$ and contract duration $T$;
**Output:** $I_W = (k_1^*, k_2^*, t^*)$ and corresponding $C(k_1^*, k_2^*, t^*)$;
**Begin**
    $W = T$;
    **While** $W > 0$
        **Call** SISL ($W$);
        $W = W - t^*$;
        $B = \max[0, B - t^* \eta_{k_1^*}]$;
    **End While**
**End**
```

Figure 9. Offline MISL Algorithm

The expected cost of the online MISL algorithm with two interventions is computed as $\int_0^{t_1} C(k_2, t_2, k_3 | k_1, C_1) f_{k_1}(\tau)\, d\tau$. Clearly, the expected cost of the online MISL algorithm depends on the realized downtime in each step of the algorithm implementation (C_1, C_2, etc.). The state space grows exponentially as the number of interventions increases. However, as shown in Section 5.4 of the online supplement, the algorithm runs in polynomial time and thus is considered as "fast."

Now we turn to competitive ratio analysis. The following Proposition establishes the competitive ratios of both SISL and MISL online algorithms.

Proposition 6. *(i) The competitive ratio of the SISL and MISL online algorithm is the same, which is bounded by 2 (i.e., $r_{SISL} = r_{MISL} < 1 + \frac{K}{n} < 2$); (ii) The competitive ratio of SISL and MISL online algorithm is lower than that of the CL algorithm if the number of decision stages in CL is large enough (i.e., $r_{SISL} = r_{MISL} < r_{CL}$ if $t_1^* > \frac{T}{S}$).*

The competitive ratio analysis of all our online algorithms (Propositions 1,3, and 6) show that our proposed solutions are competitive and perform well in the worst case scenarios. In particular, when the number of decision stages in the periodic intervention policy is relatively large, the aperiodic intervention algorithms tend to yield better online performance than the periodic online algorithms.

7. Computational Analyses

In this section, we comprehensively evaluate the solution quality and computational performance of both the periodic and aperiodic resource optimization models under different system configurations.

We classify the parameters of the cloud resource management problem into four general categories. We choose two values for each parameter in our main experiments. The first category includes two *client-specific* parameters: required service availability α (0.9 for low availability and 0.99 for high availability) and contract duration T (30 days for short term and 180 days for long term). The second category involves two *service-specific* parameters: the number of primary VMs n (100 for small size service contract and 1000 for large size service contract), and the penalty/provision cost ratio per VM (100 for low penalty and 1000 for high penalty). The third category consists of two *system-specific* operational level parameters: the mean time between failures (MTBF) of a physical server (60 days for failure-prone systems and 300 days for fault-tolerant systems), and the mean time to repair (half day). The fourth category pertains to *resource optimization* parameters. It involves the service provider's decisions regarding the maximum number of backup VMs K to provide, which we set as 10% of the number of primary backup VMs required in SLA ($10\%n$). The service provider can choose from 4 online algorithm choices corresponding to periodic intervention (CL and online CE) or aperiodic intervention (online SISL and online MISL) policies. The rationale of the experiment design and detailed solutions of all experiments are provided in the online supplement. We implemented our algorithms in R 3.4 (64-bit edition). An Intel(R) Core(TM) i7-7500U 2.90 GHz processor equipped with 8GB of RAM was used for all experiments.

7.1. Effect of Intervention

Using the static, no intervention (denoted as NI) optimal solution of Yuan et al. (2018) as a benchmark, we first demonstrate the performance improvement under the periodic policies (the CL model and the online CE model denoted as CE^{on}). Denote the expected costs under NI, CL, and CE^{on} models as $E[NI]$, $E[CL]$, and $E[CE^{on}]$. We define the *expected cost performance ratio* as $R[CL] = \frac{E[NI]-E[CL]}{E[NI]}$ and $R[CE^{on}] = \frac{E[NI]-E[CE^{on}]}{E[NI]}$, respectively, to measure the relative cost reduction of the CL and CE^{on} model from the no-intervention benchmark. The larger the ratio, the higher the expected cost reduction.

SLA Requirements			**MTBF Low (Fault-Prone)**				**MTBF High (Fault-Tolerant)**			
			π/h Low		π/h High		π/h Low		π/h High	
n	T	α	$R[CL]$	$R[CE^{on}]$	$R[CL]$	$R[CE^{on}]$	$R[CL]$	$R[CE^{on}]$	$R[CL]$	$R[CE^{on}]$
Small	Short Term	Low	16.5%	11.5%	38.4%	16.7%	48.1%	44.6%	43.4%	-3.6%
		High	3.4%	3.0%	15.5%	9.8%	15.3%	13.4%	20.8%	8.4%
	Long Term	Low	17.2%	12.1%	39.0%	19.9%	49.8%	45.7%	44.8%	3.0%
		High	7.5%	7.4%	18.9%	12.6%	20.5%	17.0%	24.7%	17.2%
Large	Short Term	Low	6.6%	5.5%	11.9%	2.6%	23.6%	19.0%	23.1%	-1.6%
		High	0%	0%	6.6%	4.5%	3.0%	2.7%	14.3%	5.8%
	Long Term	Low	7.0%	5.7%	12.2%	3.5%	24.2%	19.5%	23.7%	1.8%
		High	1.4%	1.3%	8.7%	6.6%	5.3%	5.1%	17.1%	11.1%

Table 1. Expected Cost Performance Ratios of CL and Online CE Models

Table 1 shows the expected cost performance ratios under different configurations when there are 8 decision stages to represent relatively frequent periodic interventions. Tables A4 and A5 in the online supplement provides the complete model solutions and the expected costs of the NI, CL and CE^{on} models. Several interesting observations can be made from Table 1. First, both the online CE model and the CL model have the potential to significantly reduce the expected cost due to the ability to adjust the backup provision dynamically over the contract duration. The highest cost reduction achieved is around 44-46% for the online CE model and 48-50% for the CL model when the service contract size is small, the availability is low, the penalty cost is low compared to provisioning cost, and in the fault-tolerant system.

Second, we observe two scenarios where the cost performance of the online CE model is comparable to that of the no-intervention benchmark (-3.6% and -1.6% in Table 1). This is because the CE model uses the deterministic mean downtime approximation rather than the stochastic downtime distribution in making the dynamic adjustment decisions. Both cases occur for short-term contracts when the required availability is low, provision/penalty cost ratio is high, and is managed by fault-tolerant servers. Fortunately, in such cases the CL model can significantly reduce the costs. Thus, we caution the contract managers of such cases and recommend CL model in these instances.

Third, comparing the best-performing CL model with the no-intervention benchmark, we identify scenarios where there is little need for intervention (within 2% expected cost reduction). These scenarios occur when the contract size is large, the required availability is high, the penalty cost is low compared to provisioning cost, and in the fault-prone system. Since the expected cost reduction is limited, the service provider may find the computationally expensive CL model not be economically justified. In such cases, if the service provider still prefers to perform dynamic interventions, we recommend the online CE model due to the closeness of its solution to the optimum and its computational efficiency.

On average, across all configurations the online CE model and the CL model achieved cost reduction of 10.4% and 19.4%, respectively. Overall, the benefits of dynamic optimization are more significant for small contracts with low availability in fault-tolerant systems.

7.2. Effect of Flexible Timing

In the following discussion, we compare periodic and aperiodic intervention policies to examine under what conditions it is beneficial to perform flexible interventions. We choose the best performing CL model under the periodic policy as the benchmark, and compare it with online SISL and online MISL solutions. Obviously, the number of decision stages affects the CL solution. Since there is one intervention in the online SISL model by design, we choose the two-stage CL model (CL2) for its comparison. Note that the MISL model with two interventions establishes a performance lower bound of MISL policy. We use it as a conservative benchmark to compare the performance. Denote the corresponding three-stage CL model as CL3. Define the expected cost performance ratio of the online SISL and MISL models as $R[SISL^{on}] = \frac{E[CL2]-E[SISL^{on}]}{E[CL2]}$ and $R[MISL^{on}] = \frac{E[CL3]-E[MISL^{on}]}{E[CL3]}$, respectively. So a positive ratio indicates expected cost reduction of the aperiodic intervention over the periodic intervention policies. Since the contract length does not seem to significantly affect the solution, we focus on short-term contracts for illustration purposes. Table 2 presents expected cost performance ratios of the online SISL and online MISL policies under different system configurations.

SLA Requirements		**MTBF Low (Fault-Prone)**				**MTBF High (Fault-Tolerant)**			
		π/h Low		π/h High		π/h Low		π/h High	
n	α	$R[SISL^{on}]$	$R[MISL^{on}]$	$R[SISL^{on}]$	$R[MISL^{on}]$	$R[SISL^{on}]$	$R[MISL^{on}]$	$R[SISL^{on}]$	$R[MISL^{on}]$
Small	Low	-6.0%	-7.5%	0.9%	1.1%	23.3%	10.7%	14.3%	20.3%
	High	-3.1%	-4.2%	-1.7%	-0.6%	-2.7%	-6.8%	-0.3%	2.4%
Large	Low	0.7%	0.0%	-1.6%	-1.6%	0.8%	-0.8%	-1.4%	-2.3%
	High	0.0%	0.0%	-0.1%	-0.5%	-0.3%	-0.9%	0.7%	-1.2%

Table 2. Expected Cost Performance Ratios of Online SISL and Online MISL Models

We note that the aperiodic models may not necessarily outperform the periodic models. The main reason is the use of single look-ahead till the end of the horizon at each intervention stage of the MISL strategy. As Table 2 shows, there is a clear tradeoff between the potential gain brought by flexible timing and the potential loss due to the myopic single look-ahead. We find that the performance differences in small-sized contracts are more than those in large-sized contracts. When the contract size is small, it is better to use the CL model when the penalty/provisioning cost ratio is low in fault-prone systems, and when both the penalty/provisioning cost ratio is low and the availability is high in fault-tolerant systems.

On average the highest cost reduction of the online SISL algorithm over CL2 model is 23.3% and of the online MISL algorithm over CL3 model is 20.3%, respectively. Please see Table A7 of the online supplement for expected costs under different models. The aperiodic models outperform the periodic models significantly when the contract size is small, availability is low, and in fault-tolerant systems.

Intuitively, this is because under these scenarios, the total required number of backups is relatively small, thus the marginal effect of adding or removing one backup is significant. Flexible timing allows for fine-tuning the marginal benefit-cost trade-off and thus turns out to be beneficial.

8. Model Validation with Amazon EC2 Service Structure

In this section, we validate our models based on actual pricing and service credit data on dedicated hosts obtained from the Amazon EC2 website. Consider the case of a client requesting to contract with Amazon for n instances (VMs). A dedicated host is configured to support one VM at a time. For simplicity, we consider both small and large contract sizes, denoted by $n = \{100,1000\}$ primary VMs, respectively.

The contract can have different configurations based on Amazon instance types and its pricing/penalty structures[7]. For illustration purposes, we choose the 1-year contract for the cheapest instance type *a1* and a similar 1-year contract for the most expensive instance type *p3*. These instance type designations are from the Amazon EC2 website. The monthly price p for one *a1* VM is \$206.59 and for one *p3* VM is \$13,415.94. Since service credits for violations of uptime guarantees are offered as fractions of the prices charged, it is realistic to consider the low-cost *a1* hosts to be less fault-tolerant (or equivalently, more fault-prone) than the high-cost *p3* hosts. Accordingly, we term the two instance types *a1* and *p3* considered in this study as fault-prone and fault-tolerant instances, respectively. As we do not have access to the MTBF and MTTR data from Amazon, we obtained these parameters from the server logs provided by the Center for Computational Research (CCR) at the University at Buffalo, which is a high-performance computing node. Using these parameters as surrogates for the Amazon data center operations, we conducted a detailed computational study of the proposed algorithms using their price and penalty structures for the *a1* and *p3* instance types. These results can be easily replicated if the server logs data from Amazon are available.

Our experiment consisted of four independent factors: contract size {small, large}, level of fault-tolerance of the instances {fault-prone, fault-tolerant}, provisioning cost {low, medium, high} and penalty cost {low, medium, high}. Accordingly, a fully-crossed experimental design consisting of 36 unique treatments has been carried out. In all these treatments, we consider a one month contract window. We set $\psi = 12{,}000$ as the number of discrete time intervals in the one month evaluation period, which is equivalent to about 4 minutes per interval. This is consistent with what is done in

[7] Amazon dedicated hosts pricing: https://aws.amazon.com/ec2/dedicated-hosts/pricing/

practice to measure downtime. For example, Amazon S3 tracks and calculates the error rate (i.e., downtime) for each Amazon S3 service account in every 5-minute interval in the monthly billing cycle.

Define the selling price per VM per unit time interval as $\bar{p} = p/\psi$. First, we assume the resource provisioning cost per VM per unit time is a percentage of the selling price per VM per unit time: $h = \{10\%, 30\%, 50\%\} * \bar{p}$. These define three levels of the provisioning cost. Second, we estimate the penalty cost per unit time π based on Amazon penalty-price structures. There are three cases: (1) Amazon pays no penalty if the monthly uptime percentage is greater than or equal to 99.95%; (2) it provides 10% of the monthly fee as service credit if the monthly uptime percentage is between 99% and 99.95%, and (3) it pays 30% of the monthly fee as service credit if the monthly uptime percentage is below 99%. Clearly Amazon EC2 SLA belongs to the high availability (high α) scenario. For a given n, recall that the density and cumulative distribution functions of the random downtime for k backup servers are $f_k(\tau)$ and $F_k(\tau)$, respectively. We use the baseline distribution $F_0(\tau)$ (i.e., when no backup is provided) as the basis for calculating the expected unit penalty cost as follows:

$$\hat{\pi} = \frac{10\%\times n\times p\times[F_0(0.01)-F_0(0.005)]+30\%\times n\times p\times[1-F_0(0.01)]}{\mu(\tau-0.01|\tau\geq 0.01)+j\times\sigma(\tau-0.01|\tau\geq 0.01)}.$$

Here, the numerator is the expected total service credit amount paid for n VMs, and the denominator is the (expected penalizable downtime + j * standard deviation of penalizable downtime), where we set $j = \{-1,0,1\}$ to derive three (i.e., high, medium and low) estimates of π. The expected penalizable downtime is computed as $\mu(\tau - 0.01|\tau \geq 0.01) = \int_{0.01}^{1}(\tau - 0.01)f_0(\tau)d\tau$, and the variance of penalizable downtime is calculated as $\sigma^2(\tau - 0.01|\tau \geq 0.01) = \int_{0.01}^{1}(\tau - 0.01)^2 f_0(\tau)d\tau - [\int_{0.01}^{1}(\tau - 0.01)f_0(\tau)d\tau]^2$. Based on Amazon data, Table 3 shows the values of the π/h ratio used in the 36 treatments of the experiment. Together, the 36 treatments cover a wide variety of real world use cases. This validation can be repeated for any other system parameters in a cloud data center.

Contract Size	$\boldsymbol{h/\bar{p}}$	**Fault-Prone VMs**			**Fault-Tolerant VMs**		
		$\hat{\pi}(j=1)$	$\hat{\pi}(j=0)$	$\hat{\pi}(j=-1)$	$\hat{\pi}(j=1)$	$\hat{\pi}(j=0)$	$\hat{\pi}(j=-1)$
Small	**0.1**	477.53	546.40	638.48	1,454.36	2,108.53	3,832.27
	0.3	159.18	182.13	212.83	484.79	702.84	1,277.42
	0.5	95.51	109.28	127.70	290.87	421.71	766.45
Large	**0.1**	3,030.06	3,037.95	3,045.87	3,532.41	3,785.64	4,078.00
	0.3	1,010.02	1,012.65	1,015.29	1,177.47	1,261.88	1,359.33
	0.5	606.01	607.59	609.17	706.48	757.13	815.60

Table 3. The π/h Ratio under Different Treatments

In practice, service providers such as Amazon typically configure the resource provisioning at the time the service contract is offered. They adopt a static resource provisioning strategy as required by the service contract and may or may not deploy backups. We assume Amazon optimally chooses the number of backups to deploy and the deployment does not change over the one-month period. We implemented our periodic intervention and aperiodic intervention strategies in each of the 36 treatments using their respective parametric settings. We demonstrate the potential cost savings that can be generated using our dynamic resource provisioning framework in the following discussion.

8.1. Periodic Model Implementation

We compare our periodic dynamic resource provisioning model with the static model commonly used in practice. Table 4 presents the percentage of cost savings under different treatment conditions.

Contract Size	$h/\bar{p}$	**Fault-Prone VMs**			**Fault-Tolerant VMs**		
		$\hat{\pi}(j=1)$	$\hat{\pi}(j=0)$	$\hat{\pi}(j=-1)$	$\hat{\pi}(j=1)$	$\hat{\pi}(j=0)$	$\hat{\pi}(j=-1)$
Small	**0.1**	13.79%	13.94%	14.14%	21.80%	23.86%	30.03%
	0.3	9.10%	11.30%	14.1%	21.79%	20.94%	21.36%
	0.5	2.93%	4.27%	6.07%	23.69%	22.22%	20.84%
Large	**0.1**	9.64%	9.65%	9.67%	20.88%	21.53%	22.28%
	0.3	6.60%	6.60%	6.60%	14.66%	14.86%	15.09%
	0.5	4.80%	4.81%	4.82%	13.78%	13.85%	13.95%

Table 4. Expected Cost Savings of Periodic Intervention over Static Backup Deployment

Other things being equal, as the unit provisioning cost decreases and the unit penalty cost increases, the percentage of cost savings increases. This is not surprising because lower provisioning cost directly leads to operational cost savings. The high unit penalty cost justifies the benefit of using a dynamic adjustment strategy rather than a static provisioning strategy. Overall, the savings over the static model is up to 30%.

Similar to the insights gained from the main experiments, we find that managing small contracts yields higher cost savings than large contracts, and fault-tolerant systems achieve higher cost savings than fault-prone systems. Intuitively, this is because the mean and variance of the downtime distribution are small in both cases, which require small number of backups. The larger marginal benefit of adjusting VM provisioning leads to higher cost savings.

8.2. Aperiodic Model Implementation

In the following, we demonstrate the benefit of implementing the aperiodic intervention strategy over the periodic intervention. To have a fair comparison, we should allow the MISL and CL models to have the same number of stages (S). Since a MISL model has at least 3 stages, for illustration purposes, we choose $S = 3$ and denote the 3-stage CL model as CL3. As the number of stages increases, the advantage of aperiodic intervention over periodic intervention would diminish, and this is intuitive. Table 5 presents the percentage of cost savings of online MISL model over CL3 model under different treatment conditions, given by $\frac{E[CL3]-E[MISL^{on}]}{E[CL3]}$. Note that a positive value indicates that the MISL model yielded lower expected cost than the CL3 model.

Contract Size	$h/\overline{p}$	Fault-Prone VMs			Fault-Tolerant VMs		
		$\hat{\pi}(j=1)$	$\hat{\pi}(j=0)$	$\hat{\pi}(j=-1)$	$\hat{\pi}(j=1)$	$\hat{\pi}(j=0)$	$\hat{\pi}(j=-1)$
Small	**0.1**	0.49%	0.64%	0.59%	-2.3%	-7.9%	-5.21%
	0.3	-2.29%	-2.16%	-1.96%	8.08%	5.7%	-0.53%
	0.5	-4.03%	-4.68%	-6.12%	3.76%	7.5%	4.91%
Large	**0.1**	-0.67%	-0.67%	-0.67%	0.5%	0.49%	0.31%
	0.3	-0.47%	-0.47%	-0.47%	-0.9%	-0.77%	-0.63%
	0.5	-0.97%	-0.98%	-0.98%	-1.71%	-1.65%	-1.58%

Table 5. Expected Cost Savings of Aperiodic Intervention over Periodic Intervention

We see that the highest percentage of cost savings of the aperiodic intervention over periodic intervention is 8.08%. This occurs for small-sized contracts with fault-tolerant VMs when the penalty/provisioning cost ratio is moderately high (i.e., 484.79). Based on these experimental assessments, a decision tree for the choice of intervention strategy by Amazon EC2 can be formulated as in Figure 10.

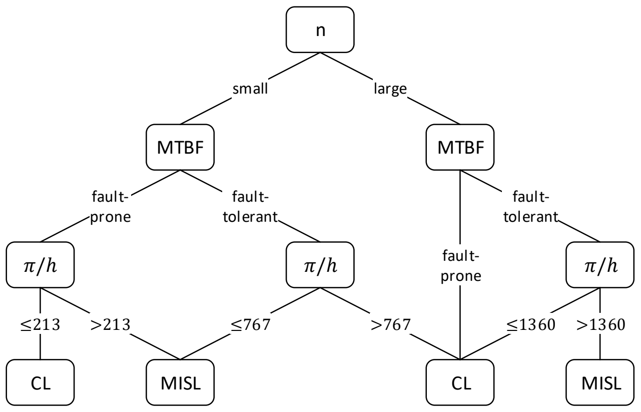


Figure 10. Decision Tree Calibration of CL vs. MISL Intervention Policies

The above analysis yields a proof-of-concept of the proposed intervention framework and demonstrates the efficiencies of the proposed algorithms using Amazon EC2 price/penalty data as a testbed of use cases, in conjunction with the system-specific parameters drawn from the CCR data. This study also yields a decision tree based strategy for any contract administrator to follow in the choice of intervention policies by appropriately calibrating them according to the prevailing contract-specific, system-specific and service-specific parameters.

9. Discussion of Managerial Implications

In this research, we develop different resource management strategies to support the cloud contract administrators' needs to conservatively or aggressively manage virtual infrastructure resources, and preferences to periodically or aperiodically adjust the backup VMs.

Our CL model supports conservative resource provisioning under the periodic intervention policy. The CL algorithm yields a reference chart of optimal allocations for a given service agreement and the service provider would dynamically allocate the backup VMs according to the realized conditions during the course of the contract. It offers valuable guidance for cloud service providers to structure the VM resources in the data center based on client service agreements in a dynamic and uncertain operating environment. The dynamic optimization significantly outperforms static optimization (no intervention), especially for small contracts that require low availability in fault-tolerant systems.

In the event where the cloud service providers prefer to use an aggressive resource management strategy, we recommend the use of the computationally cost-effective online CE model. Although the CL model would yield optimal allocations, the online CE model is quite cost-effective and yields near-optimal allocations in most cases, and especially for large contracts when penalty/provisioning cost ratio is low and the required availability is high in fault-prone systems.

In general, solving the aperiodic optimization problem is more computationally demanding than the periodic optimization. The trade-off between the periodic and aperiodic optimization is the additional cost reduction brought by flexible interventions and the increased time complexity to solve the aperiodic models. Based on the evaluation of use cases constructed from Amazon EC2 price and service credit structure, we find that it is more beneficial to employ aperiodic intervention than periodic intervention when managing small-sized service contracts with fault-tolerant VMs under high availability requirements.

To guide contract administrators to choose between the best-performing periodic (CL) and aperiodic (online MISL) intervention policies, we compare their expected costs based on which we can construct a decision tree recommendation. This analysis serves as a proof-of-concept for any cloud data center to adopt our analytical strategy in SLA resource management. Through comprehensive testing of our model using both synthetic data and Amazon use cases, we provide collaborative evidences that our proposed dynamic backup resource provisioning strategy can generate significant cost savings over the static backup provisioning approach.

Overall, our model is most suitable to IaaS provider' virtual infrastructure resource management based on their pre-committed resource requirements in the service contracts. Our model can be applied to both stateless and stateful VMs as long as appropriate checkpointing strategy and backup VMs are used to guarantee service continuity in the case of primary VM failure. Thus, our model can be extended and implemented in a PaaS environment where the PaaS service providers handle both infrastructure and services. However, we note that our current model is not directly applicable to big data management on the cloud. For example, MapReduce proposed by Google and its open source implementation Hadoop are the most popular big data management frameworks today. By moving big data and its processing to cloud, data is stored in storage clouds and computation is done with compute clouds. The Hadoop distributed file system (HDFS) handles large data sets on commodity servers. Data needs to be copied from the location where it is stored to the instance on which the computation will occur. This will incur data transmission costs as well as time delay according to the size of the data transmitted. Network structure and data transfer latency may be some important factors to consider in such applications. In addition, when using a backup, there is cost involved in the frequency and volume of updates, which can be an important issue in service contracts design, negotiation with the clients and solution implementation. Since PaaS involves client-level applications, it is different from the primary goal of this research as we do not model the operational details such as the data replication structures and how the VMs are used from the client's perspective. Extending our framework along the above discussed dimensions are interesting and important future research directions.

10. Conclusion

In this research, we develop cost-effective solutions to the optimal management of virtual infrastructure resources in cloud service agreements by an integration of ideas from multiple disciplines. More specifically, this integration involves the following concepts: (i) dynamic resource optimization from Operations Management, (ii) availability modelling based on sample path randomization techniques derived from the disciplines of Statistics and Machine Learning, and (iii) design of online algorithms drawn from Computer Science. We develop stochastic optimization models to support both periodic and aperiodic cloud infrastructure resource management strategies in a dynamic environment. We propose computationally efficient online algorithms that utilize their corresponding offline solutions to achieve both high computational performance and solution quality. We perform both competitive ratio analysis and expected value analysis to investigate the worst-case and average model performance, respectively. We further conduct comprehensive computational experiments and validate our model performance based on use cases constructed from Amazon EC2 price and service credit data. Findings from this research provide practical decision support for cloud data-center's virtual resource management under flexible service agreements.

Under the periodic intervention policy, we develop dynamic optimization models to adjust the provision of backup VMs based on the observed system downtime at regular time intervals. We examine both a conservative strategy (the CL model) that is computationally expensive but minimizes the expected total operational cost, and an aggressive strategy (the CE model) that is computationally less expensive but would only yield near-optimal solutions. We find that the dynamic periodic intervention significantly outperforms its static no-intervention counterpart when service contracts are small-sized, require low availability and demand low contract penalty in fault-tolerant systems. In addition, we recommend the use of the aggressive strategy for managing large service contracts with high availability in fault-prone systems with low penalty/provisioning cost ratios, as it produces near optimal solution to the conservative strategy but is more computationally effective.

Under the aperiodic intervention strategies, we allow flexible timing of intervention and examine both single intervention and multiple intervention policies. We find that the flexible intervention time is more beneficial than the fixed time of intervention when the size of service contract is small and availability is low in fault-tolerant systems. We also find that it might be sufficient to use single intervention rather than multiple interventions for small-sized service contracts when the penalty/provisioning cost ratio is low and the availability is high.

In terms of computational performance, all our online algorithms have competitive ratios bounded by a factor less than 2. It shows that the algorithms we developed are economical and can achieve good performance guarantee at the worst-case scenarios. In terms of solution quality, the potential advantage of flexible timing would diminish when the number of stages in periodic intervention increases. Our extensive computational experiments and Amazon use cases evaluation offer us empirical evidences of the effectiveness and robustness of our model performance under various system-level, service-level, and user-level parametric conditions. We also construct a decision tree to provide intervention policy recommendations and managerial guidelines with insights that facilitate cloud contract administrators to execute their service contracts cost-effectively. This study leads to several important, viable and practical directions for future research. We outline some of these avenues in the following discussion.

First, in this study, we do not explicitly model the cost of intervention and cost of server repair. Presumably, the server repair cost can be factored into the server provisioning cost. In this case, both the number of backup VMs and the intervention frequency would be reduced compared to our base model. Although the main insights obtained from this study would still be valid, specifically considering the costs of intervention and repair could affect the cost and benefit trade-offs in the service provisioning. It would be an interesting new dimension of investigation for future.

Second, we assume the simpler architecture of powered-on without delay in the backup provision, where a set of backup VMs regularly capture the system states from the primaries and ensure continuity of service when primary VMs fail. An alternative consideration is the powered-off model where a single large backup that periodically captures the primary VM states and is used as recovery and rollback mechanism in the case of VM failures. While the powered-on without delay architecture is common in normal data center operations, future research may consider the powered-off model, which is common in highly fault-resilient and high-performance computing platforms.

Third, we focus on the Exponential failures in deriving the transient downtime distribution, which represents constant random failure rates in the cloud infrastructure. While exponential failures are common in practical data center operations, servers in certain data centers could have changing failure rates in their lifecycles. The case of decreasing failure rates over time can be modelled using the Weibull distribution which represents the *infant mortality* of servers. In this case, the servers more frequently fail during the early stages of their lifecycles and attain stability over time. Similarly, the case of increasing failure rates over time can be modelled using the Erlang distribution which captures the *aging* condition of servers. In this case, servers tend to fail more often as they age. Different failure rate distributions will affect the server downtime empirical distribution, which in turn would affect the cloud service provider's backup VM provisioning decisions. Future research on Weibull and Erlang failures would generate their appropriate virtual resource management strategies with richer practical insights.

In conclusion, IaaS cloud resource management is a complex issue. Performance metrics such as delay, bandwidth overhead, computation overhead, reliability and security have to be taken into consideration in the design of resource management schemes. In addition, modeling the backup replication structure and user requirements at PaaS level is an interesting future research direction.

Appendix: Notation Table

This notation table includes all variables defined and used in the main paper and the online supplement.

$\alpha \in (0,1)$	Required availability level (the uptime guarantee) specified in the SLA
T	Contract period over which the uptime guarantee should be fulfilled
$B = (1-\alpha)T$	Total allowable downtime by SLA contract
h	Provisioning cost per VM per unit of time
π	SLA violation penalty per unit of time
n	Number of primary VMs needed in the SLA contract
$k = \{1,2,\dots K\}$	Number of backup VMs, where K is the maximum number of backups
$f_k(\tau), F_k(\tau)$	The downtime density function and cumulative distribution function
$\phi_k(\tau), \Phi_k(\tau)$	The normalized percentage of downtime density function and cumulative distribution function
η_k	Mean percentage of downtime when the number of backup VMs is k
MTBF	Mean time between failures of primary and backup servers
MTTR	Mean time to repair of primary and backup servers
$\delta = \{1,2,\dots S\}$	Index of stage, where there are total of S decision stages

x_δ	Total incurred downtime at the beginning of stage δ
k_δ	Number of backup VMs at the beginning of stage δ
$u_\delta \in [-k_\delta, K - k_\delta]$	Backup VM adjustment decision in stage δ
$\hat{v}(\tau, n, k)$	Estimated probability distribution function of τ downtime on $[0, T]$ for an (n, k) VM configuration
Δt	Discretized time intervals to track downtime over $[0, T]$
$\psi = \left\lceil \frac{T}{\Delta t} \right\rceil$	The total number of discretization intervals over the contract period T
$D_W(k)$	Random downtime in a stage with length W when the number of backup VMs is k
$d_W(k)$	Expected downtime in a stage with length W when the number of backup VMs is k
$D_w(i, n, k)$	Discretized downtime in a stage with length W
$p_w(i, n, k)$	Discretized probability corresponding to i number of downtime intervals in a stage with length W
$C(k_1, k_2, t)$	Cost function of SISL strategy with the number of backup VM provisions before and after intervention as (k_1, k_2) and the intervention time t
$J_\delta(x_\delta, k_\delta)$	Cost-to-go function in stage δ of Problem [CL], which is the minimum expected total cost from stage δ to the end of stage S
$G_\delta(k_\delta)$	Cost-to-go function in stage δ of Problem [CE], which is the minimum expected total cost from stage δ to the end of stage S
$H_\delta(x_\delta, k_\delta)$	Cost-to-go function in stage δ of Problem [MISL], which is the minimum expected total cost from stage δ to the end of stage S
$\Omega_\delta(k_\delta)$	Total expected downtime from state k_δ of stage δ till the end of stage S along the path corresponding to $G_\delta(k_\delta)$
C_Δ	Accrued total realized downtime from time 0 to stage Δ
$\hat{X}_\Delta$	Observed actual downtime in stage Δ
$\hat{B}$	Allowable total downtime at the beginning of stage Δ
p	Monthly price of an instance (VM)
$\bar{p} = p/\rho_{max}$	Selling price per VM per unit time interval

References

Ahmad, R.W, A. Gani, S.H. Hamid, M. Shiraz, A. Yousafzai and F. Xia. 2015. A Survey on Virtual Machine Migration and Server Consolidation Frameworks for Cloud Data Centers. *Journal of Network and Computer Applications*, 52:11-25.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press, Princeton, N.J.

Bertsekas, D.P. 1995. *Dynamic Programming and Optimal Control* (Vol. 1, 3rd edition). Belmont, MA: Athena scientific.

Bilal, K., S.U.R. Malik, S.U. Khan and A.Y. Zomaya. 2014. Trends and Challenges in Cloud Datacenters. *IEEE Cloud Computing*, 1(1): 10-20.

Bobroff, N., A. Kochut and K. Beaty. 2007. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *IEEE International Symposium on Integrated Network Management*, 119-128.

Bruneo, D. 2014. A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems. *IEEE Transactions on Parallel and Distributed Systems*, 25(3): 560-569.

Chase, J. and D. Niyato. 2017. Joint Optimization of Resource Provisioning in Cloud Computing. *IEEE Transactions on Services Computing*, 10(3): 396-409.

Cheng, H.K., Z. Li and A. Naranjo. 2016. Research Note - Cloud Computing Spot Pricing Dynamics: Latency and Limits to Arbitrage. *Information Systems Research*, 27(1): 145-165.

Chowdhury, N.M.M.K. and R. Boutaba. 2010. A Survey of Network Virtualization. *Computer Networks*, 54(5): 862-876.

Cloud Standards Customer Council. 2017. *Practical Guide to Cloud Computing (version 3.0)*, http://www.cloud-council.org/deliverables/CSCC-Practical-Guide-to-Cloud-Computing.pdf

Colman, E. 2013. When to Use SaaS, PaaS, and IaaS? *Computenext.com* (August 27), https://www.computenext.com/blog/when-to-use-saas-paas-and-iaas/

Das, S., A.Y. Du, R. Gopal and R. Ramesh. 2011. Risk Management and Optimal Pricing in Online Storage Grids. *Information Systems Research*, 22(4): 756-773.

Dey, D., M. Fan and C. Zhang. 2010. Design and Analysis of Contracts for Software Outsourcing. *Information Systems Research*, 21(1): 93-114.

Du, A.Y., S. Das, Z. Yang, C. Qiao and R. Ramesh. 2015. Predicting Transient Downtime in Virtual Server Systems: An Efficient Sample Path Randomization Approach. *IEEE Transactions on Computers*, 64(12): 3541-3554.

Fu, S. 2010. Failure-Aware Resource Management for High-Availability Computing Clusters with Distributed Virtual Machines. *Journal of Parallel and Distributed Computing*, 70(4): 384-393.

Gill, P., N. Jain and N. Nagappan. 2011. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In *Proceedings of the ACM SIGCOMM Conference*, 350-361. Toronto, Canada.

Goiri, I., F. Julia, J. Guitart, and J. Torres. 2010. Checkpoint-based Fault Tolerant Infrastructure for Virtualized Service Providers. In *IEEE Network Operations and Management Symposium (NOMS)*, 455-462.

Goudarzi, H., M. Ghasemazar and M. Pedram. 2012. SLA-Based Optimization of Power and Migration Cost in Cloud Computing. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*.

Jaillet, P. and M.R. Wagner. 2012. *Online Optimization*. Springer.

Johar, M., V. Mookerjee and S. Sarkar. 2014. Selling vs Profiling: Optimizing the Offer Set in Web-based Personalization. *Information Systems Research*, 25(2): 285-306.

Kauffman, R.J., D. Ma, R. Shang, J. Huang and Y. Yang. 2014. On the financification of cloud computing: An agenda for pricing and service delivery mechanism design research. *International Journal of Cloud Computing*, 2(1): 1-14.

Laalaoui, Y., and J. Al-Omari. 2018. A Planning Approach for Reassigning Virtual Machines in IaaS Clouds. *IEEE Transactions on Cloud Computing*, forthcoming.

Liu, D., S. Sarkar and C. Sriskandarajah. 2010. Resource Allocation Policies for Personalization in Content Delivery Sites. *Information Systems Research*, 21(2): 227-248.

Liu, J., Y. Zhang, Y. Zhou and D. Zhang. 2015. Aggressive Resource Provisioning for Ensuring QoS in Virtualized Environments. *IEEE Transactions on Cloud Computing*, 3(2): 119-131.

Lu, P., B. Ravindran and C. Kim. 2012. VPC: Scalable, Low Downtime Checkpointing for Virtual Clusters. In *Proceedings of the IEEE 24th International Symposium on Computer Architecture and High Performance Computing* (SBAC-PAD), 203-210.

Mistry, S., A. Bouguettaya, H. Dong and A.K. Qin. 2018. Metaheuristic Optimization for Long-term IaaS Service Composition. *IEEE Transactions on Services Computing*, 11(1): 131-143.

Qiu, W., Z. Zheng, X. Wang, X. Yang and M.R. Lyu. 2014. Reliability-Based Design Optimization for Cloud Migration. *IEEE Transactions on Services Computing*, 7(2): 223-236.

Ran, Y., J. Yang, S. Zhang and H. Xi. 2017. Dynamic IaaS Computing Resource Provisioning Strategy with QoS Constraint. *IEEE Transactions on Services Computing*, 10(2): 190-202.

Shabeera, T.P., S.M. Kumar, S.M. Salam and K.M. Krishnan. 2017. Optimizing VM Allocation and Data Placement for Data-Intensive Applications in Cloud using ACO Metaheuristic Algorithm. *Engineering Science and Technology, an International Journal*, 20(2): 616-628.

Sen, S., T.S. Raghu and A. Vinze. 2009. Demand Heterogeneity in IT Infrastructure Services: Modeling and Evaluation of a Dynamic Approach to Defining Service Levels. *Information Systems Research*, 20(2): 258-276.

Sieke, M. A., R.W. Seifert and U.W. Thonemann. 2012. Designing Service Level Contracts for Supply Chain Coordination. *Production & Operations Management*, 21(4): 698-714.

Silva Filho, M.C., C.C. Monteiro, P.R. Inácio and M.M. Freire. 2018. Approaches for Optimizing Virtual Machine Placement and Migration in Cloud Environments: A Survey. *Journal of Parallel and Distributed Computing*, 111: 222-250.

Singh, S., I. Chana and R. Buyya. 2017. STAR: SLA-Aware Autonomic Management of Cloud Resources. *IEEE Transactions on Cloud Computing*, forthcoming.

Sleator, D.D. and R.E. Tarjan. 1985. Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28(2): 202-208.

Wang, X., Z. Du, Y. Chen and S. Li. 2008. Virtualization-Based Autonomic Resource Management for Multi-Tier Web Applications in Shared Data Center. *Journal of Systems and Software*, 81(9): 1591-1608.

Wu, L., S.K. Garg and R. Buyya. 2011. SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments. In *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*.

Xu, J., J. Tang, K. Kwiat, W. Zhang and G. Xue. 2012. Survivable Virtual Infrastructure Mapping in Virtualized Data Centers. In *Proceedings of the IEEE 5th International Conference on Cloud Computing*, 196-203.

Yuan, S., S. Das, R. Ramesh and C. Qiao. 2018. Service Agreement Trifecta: Backup Resources, Price and Penalty in the Availability-Aware Cloud. *Information Systems Research*, 29(4): 947-964.

Zhao, L., S. Sakr and A. Liu. 2015. A Framework for Consumer-Centric SLA Management of Cloud-Hosted Databases. *IEEE Transactions on Services Computing*, 8(4): 534-549.

Zhou A., S. Wang, B. Cheng, Z. Zheng, F. Yang, R.N. Chang, M.R. Lyu and R. Buyya. 2017. Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization. *IEEE Transactions on Services* Computing, 10(6):902-13.