Cloud Resource Provisioning and Contract Adjustment in the Backdrop of SLA Violation Risk Mitigation

The bulk of the work was done by a doctoral student.

Abstract: Service Level Agreements (SLA) in cloud computing, with their many constructs such as price, penalty, contract duration, and server uptime guarantee, represent notable trade-offs faced by the provider, specifically between provisioning cost control and SLA fulfillment. Here we address two components of the SLA-based contract design problem in cloud services: (1) the optimal service provisioning problem which aims to fulfill the uptime guarantee assured in the SLA and (2) the pricing problem in the context of SLA violation. An algorithm minimizing expected total cost by deriving the optimal number of backup virtual machines (VMs) for the purpose of mitigating SLA violation risk is presented. We also study specific contract adjustment measures that the provider can utilize to encourage the client to accept penalty payment deferrals in the event of SLA violation. Such measures allow the provider to possibly recover from server failures in a prior period by provisioning more resources in the next period. We also present some preliminary computational results which highlight the validity of some of our assumptions pertaining to the downtime distribution and demonstrate the robustness of our algorithm to specific parameter choices.

1. Introduction and Motivation

Cloud computing deemphasizes the need for companies to maintain significant in-house infrastructure. Server instances, also known as virtual servers/machines, with various sizes and different configurations of CPU, memory, and storage are now commonly offered by cloud service providers such as Amazon Web Services (AWS), Google and Microsoft. A client can

rent a set of virtual machines (VMs) to remotely access resources stored in these VM from any location via Internet without purchasing, storing and maintaining large scale and expensive physical servers locally. Therefore the cost involved in local resource management has been reduced dramatically, as well as eliminating the risk emanating from capacity under-utilization or demand non-fulfillment resulting from demand decreases or increases respectively by only paying what she actually used as a client of the cloud datacenter operator. These VMs are mapped onto physical servers within a cloud datacenter maintained by a provider. Such datacenters are known to susceptible to different types of failures ranging from frequent small scale failures (such as a few isolated individual server/switch failures) to less frequent but large-scale failures (such as a rack/pod or even an entire datacenter failure) due to hardware (including power, disk and network) failures, software bugs, human errors, hacks/sabotages, or natural disasters (see Dean 2009, Gill et al 2011). Failures at physical server level necessarily lead to the unavailability of running VMs due to loss of connectivity, etc.

A cloud service provider typically enters into a Service Level Agreement (SLA) with a client guaranteeing a pre-determined level of service, e.g., 99.9% uptime guarantee of the VMs provided for the contract duration. A penalty is incurred when the service provider violates this commitment, not to mention significant loss of goodwill and reputation. A cloud services SLA therefore typically comprises of contract constructs such as price, penalty, contract duration, uptime guarantee and the length of the service window over which the uptime guarantee is assured. These constructs all interact with each other and thus the overarching problem of determining an optimal SLA between a provider and a client becomes very complex. In this research, we address two related components of this larger contract determination problem: 1)

the optimal service provisioning component which aims to fulfill the uptime guarantee assured in the SLA and 2) the pricing problem as it relates to the likelihood of fulfilling this guarantee.

Significant progress is being made in the area of SLA design for cloud services, largely driven by the terrific growth in emerging cloud services, rapid increase in the number of service providers and more and more clients embracing the cloud to meet their needs. These works have focused on the establishment of SLA, the detection of SLA violations, and resource allocation to meet SLA requirement. Chhetri et al (2012) proposes a policy-based framework for the automated establishment of SLAs allowing both clients and providers the flexibility to choose amongst different pricing models, and Redl et al (2012) introduces a method to find semantically equal elements from different SLAs to enable automatic selection of optimal service offer for the client since there is no general standard for SLAs specification currently. Further, an architecture which is used to detect SLA violation by resource monitoring is presented in V.C. Emeakaroha et al (2012). This architecture allocates resources for a request from clients, monitors computing resources, and translates the low-level metrics into high-level SLAs using the specified mapping rules. Cost-minimization or profit-maximization through resource allocation in relation to SLA requirements is another major topic of interest. Goudarzi et al (2012) studies the SLA-based resource allocation problem to minimize the total energy cost of cloud computing by effective VM placement. Wu et al (2011) proposes algorithms to minimize VM cost and SLA violations in the context of SaaS (Software as a Service) by effective platform layer resource allocation strategies. Hu et al (2009) develops an algorithm to determine the allocation strategy that results in smallest number of servers required with two interactive job classes. However, to the best of our knowledge, the state-of-the-art in SLA research as represented by these papers do not consider: 1) optimal resource provisioning of backup VMs for the purpose of mitigating SLA

violation risk, 2) probabilistic modeling of SLA violation using downtime distributions of virtualized infrastructure, and 3) contract adjustment strategies in the event of SLA violation.

A common strategy followed by providers is to allocate a set of backup VMs to substitute the failed VMs as needed, thus reducing the probability of failing to meet the SLA-specified availability requirement level. For instance, if a client demands n VMs but is supplied k additional VMs, downtime would be incurred only when at least $k + 1$ VMs fail. Note that the client only pays for the n VMs he demanded; the k backups are used by the provider to mitigate the risk of SLA violation, thereby paying a penalty. Thus, deciding on the number of backup VMs is an operational decision which is typically not disclosed to the client. If the number of backup VMs needed is determined in an ad hoc manner, the provider runs the risk of either violating the SLA (through under-provisioning) or reducing his profit (by over-provisioning). The provider thus needs to determine the number of backup VMs that minimizes his expected total costs, which comprises of the operating cost and expected penalty in case of SLA violation.

Du et al (2012) provides a set of algorithms to derive the downtime distribution of a given server configuration based on the limiting behavior of the birth-death process of VM failures and repairs, using a sample path analysis. We build on that research by designing an algorithm to efficiently determine the optimal number of backup VMs to meet an SLA by utilizing certain properties of the downtime distribution. Note that downtime distribution need not follow a well-behaved probability distribution (e.g., exponential). In fact, preliminary testing using real world server log data from a large high performance computing center reveals this difficulty of finding a good fit with most well-known distributions. We have not provided the details of this distribution fitting exercise due to space limitations. Note that even if we were to find a reasonably good fit with a particular distribution, there is no assurance that data collected

from a different data center would also fit that distribution. We therefore develop a computationally efficient algorithm which first derives a piece-wise linear approximation of the downtime distribution using a method developed by Wang and Chaovalitwongse (2013) and then determines the optimal number of backup VMs to satisfy the uptime guarantee in a given SLA.

Next, we study the issue of SLA violation, i.e., non-fulfillment of the uptime guarantee, particularly how price may then be used as a tool to defer the penalty payments to a client. In each service window, the client in our strategy has the option of either redeeming any penalty accrued or defer it to the next period and thereby allowing the provider to possibly make up for the service shortfall in the next period. We derive the maximum price that the provider can charge under such conditions of non-fulfillment and still motivate the client to defer the penalty payments. Extant research does not examine contract adjustment strategies in the event of SLA violation, which is a necessary study given the high rate of failures in cloud datacenters.

The rest of the paper is organized as follows. Section 2 provides the algorithm to derive the optimal number of backup VMs to service a particular client SLA. Section 3 studies contract adjustment strategies (particularly using pricing) in the event of SLA violation. Section 4 provides some preliminary computational analysis providing key managerial insights and Section 5 concludes with a discussion of our ongoing research in this area.

2. Optimal Backup Provisioning Model

Since uptime guarantees are provided on a client by client basis, we study a contract between a provider and a client. We make some simplifying assumptions for analytical tractability. We assume that the provider faces a single type of failure and a 1:1 mapping of physical to virtual servers in the datacenters. This is reasonable since datacenter providers may

choose to spread out the VMs for a given client across multiple racks to manage failure risks and avoid potential SLA violation. The client requires n servers. The service provider provides $n + k$ servers in order to provide some degree of failure resilience. When any of the n VMs fails, one of the k backup VMs is used to replace the failed VM. Du et al (2012) derives downtime distributions under two scenarios: one where this replacement is instantaneous and another where some non-negligible time is expended in the process. Our models are applicable to either of these scenarios. Table 1 lists additional notations used in this paper along with brief definitions.

α:	Uptime guarantee specified in the SLA, $0 \leq \alpha \leq 1$
t:	Service window over which the uptime guarantee has to be fulfilled (E.g, if SLA specifies 99% availability in each week, then service window is one week.)
T:	Contract duration ($T \geq t$)
p:	Price for each of the *n* VMs demanded per unit of time
π:	Penalty per unit of time as a result of SLA violation
τ:	Accumulated downtime within one service window *t*
$v(\tau)$:	Probability distribution function of the total downtime within a service window *t* for an (*n, k*) VM configuration, derived by the algorithm specified in Du et al (2012)
h:	Operating cost per VM per unit of time

Table 1: Notations

The provider can reduce the likelihood of SLA violation by providing more backup VMs, however this entails a higher operating cost for provisioning these backups while the client pays only for the *n* VMs demanded. The operating cost is therefore traded-off against the expected penalty faced by the provider. The provider therefore needs to determine the optimal number of backup VMs such that his total expected costs are minimized. The provider's operating cost is modeled as hkt, while the expected penalty is $\pi \int_{(1-\alpha)t}^{t} v(\tau)(\tau - (1-\alpha)t)d\tau$, as the provider does not incur a penalty if total downtime does not exceed $(1-\alpha)t$. Further, since τ corresponds to the total downtime incurred within a service window, we have to subtract the

allowable downtime in the calculation of the expected penalty. We therefore have the following expected total cost function for a specific *k*:

$$Q_k = hkt + \pi \int_{(1-\alpha)t}^{t} v(\tau)(\tau - (1-\alpha)t)d\tau \quad (1)$$

We assume that $\int_{(1-\alpha)t}^{t} v(\tau, k)d\tau$ is a decreasing and convex function on k. This is a valid assumption as the probability of SLA violation is likely to decline as the provider increases the number of backup VMs, ceteris paribus. In Section 4, we validate this assumption. Next, we define *expected penalizable downtime* as the amount of downtime accumulated during the service window in excess of the downtime allowable under the SLA specified uptime guarantee, α, which is denoted by $\int_{(1-\alpha)t}^{t} v(\tau)(\tau - (1-\alpha)t)d\tau$. Penalties are incurred only for the expected penalizable downtime. We can then derive the following two lemmas.

Lemma 1: The expected penalizable downtime is decreasing (1a) and convex (1b) in k.

Proof of Lemma 1(a): Let $\tau_0 = (1-\alpha)\mathrm{t},\ \mathrm{k}_1 < \mathrm{k}_2$. We get $\int_{\tau_0}^{\mathrm{t}} \mathrm{v}(\tau,\ \mathrm{k}_1)\mathrm{d}\tau > \int_{\tau_0}^{\mathrm{t}} \mathrm{v}(\tau,\ \mathrm{k}_2)\mathrm{d}\tau$ from our assumption. Since $(\tau - \tau_0) > 0$ when $\tau \in (\tau_0, \mathrm{t})$, we get the following inequality:

$\int_{\tau_0}^{\mathrm{t}} \mathrm{v}(\tau, \mathrm{k}_1)(\tau - \tau_0)\mathrm{d}\tau > \int_{\tau_0}^{\mathrm{t}} \mathrm{v}(\tau, \mathrm{k}_2)(\tau - \tau_0)\mathrm{d}\tau$.

Therefore, $\int_{\tau_0}^{\mathrm{t}} \mathrm{v}(\tau, \mathrm{k}_2)(\tau - \tau_0)\mathrm{d}\tau - \int_{\tau_0}^{\mathrm{t}} \mathrm{v}(\tau, \mathrm{k}_1)(\tau - \tau_0)\mathrm{d}\tau < 0$, i.e. $\int_{\tau_0}^{\mathrm{t}} \mathrm{v}(\tau, \mathrm{k})(\tau - \tau_0)\mathrm{d}\tau$ is a decreasing function in k.

Proof of Lemma 1(b): For any $\mathrm{k}_1, \mathrm{k}_2 \in \mathrm{k}$, according to our assumption, we have

$\int_{\tau_0}^{t} v(\tau, \frac{(k_1+k_2)}{2})d\tau < \frac{\int_{\tau_0}^{t} v(\tau,k_1)d\tau+\int_{\tau_0}^{t} v(\tau,k_2)d\tau}{2}$. Since $(\tau-\tau_0) > 0$ when $\tau \in (\tau_0, t)$, we get the following inequality: $\int_{\tau_0}^{t} v\left(\tau, \frac{(k_1+k_2)}{2}\right)(\tau-\tau_0)d\tau < \frac{\int_{\tau_0}^{t} v(\tau,k_1)(\tau-\tau_0)d\tau+\int_{\tau_0}^{t} v(\tau,k_2)(\tau-\tau_0)d\tau}{2}$.

Therefore, $\int_{\tau_0}^{t} v(\tau)(\tau-\tau_0)d\tau$ is a convex function in *k*. **QED.**

Lemma 2: The expected total cost is a convex function in k.

Proof: From Equation (1), for any $k_1, k_2 \in k, k_1 \neq k_2$,

$$Q_{\frac{k_1+k_2}{2}} - \frac{Q_{k_1}+Q_{k_2}}{2} = h\frac{(k_1+k_2)}{2}t + \pi\int_{\tau_0}^{t} v\left(\tau, \frac{(k_1+k_2)}{2}\right)(\tau-\tau_0)d\tau$$

$$- \frac{hk_1t+\pi\int_{\tau_0}^{t} v(\tau,k_1)(\tau-\tau_0)d\tau+hk_2t+\pi\int_{\tau_0}^{t} v(\tau,k_2)(\tau-\tau_0)d\tau}{2}$$

$$= \pi\int_{\tau_0}^{t} v\left(\tau, \frac{(k_1+k_2)}{2}\right)(\tau-\tau_0)d\tau - \frac{\pi\int_{\tau_0}^{t} v(\tau,k_1)(\tau-\tau_0)d\tau + \pi\int_{\tau_0}^{t} v(\tau,k_2)(\tau-\tau_0)d\tau}{2}$$

$$= \pi\left(\int_{\tau_0}^{t} v\left(\tau, \frac{(k_1+k_2)}{2}\right)(\tau-\tau_0)d\tau - \frac{\int_{\tau_0}^{t} v(\tau,k_1)(\tau-\tau_0)d\tau+\int_{\tau_0}^{t} v(\tau,k_2)(\tau-\tau_0)d\tau}{2}\right).$$

From Lemma 1, we get,

$$\int_{\tau_0}^{t} v\left(\tau, \frac{(k_1+k_2)}{2}\right)(\tau-\tau_0)d\tau < \frac{\int_{\tau_0}^{t} v(\tau,k_1)(\tau-\tau_0)d\tau+\int_{\tau_0}^{t} v(\tau,k_2)(\tau-\tau_0)d\tau}{2}.$$

Therefore, $\pi\left(\int_{\tau_0}^{t} v\left(\tau, \frac{(k_1+k_2)}{2}\right)\tau d\tau - \frac{\int_{\tau_0}^{t} v(\tau,k_1)\tau d\tau+\int_{\tau_0}^{t} v(\tau,k_2)\tau d\tau}{2}\right) < 0$. That is,

$Q_{\frac{k_1+k_2}{2}} - \frac{Q_{k_1}+Q_{k_2}}{2} < 0$, or $Q_{\frac{k_1+k_2}{2}} < \frac{Q_{k_1}+Q_{k_2}}{2}$. Therefore, the expected total cost is a convex function in *k*. **QED.**

From Equation (1), in order to find the closed form solution to the optimal number of backup VMs, k^*, we need Lemma 2 and a differentiable functional form of the downtime distribution. We ran extensive distribution fitting exercises on downtime distributions, specifically testing against exponential, gamma, Weibull, log normal and log logistic distributions, using the chi square goodness of fit test. None of these fit the distributions satisfactorily. We therefore use a method prescribed by Wang and Chaovalitwongse (2013) to derive a piecewise linear approximation of $v(\tau)$. Since most piecewise linear approximation algorithms rely extensively on some data-dependent threshold-based strategies, there exists a tradeoff between approximation accuracy and compression rate, which the user typically resolves via a trial and error process. Instead, we employ a two-stage top-down segmentation approach following Wang and Chaovalitwongse (2013) that first decomposes data recursively into non-overlapping intervals until all partitioned segments satisfy a data-independent threshold which is used to control approximation accuracy, and then fine-tune the linear approximation by performing least-squares linear regression. We outline this process in Figure 1.

Inputs: $v(\tau)$, approximation accuracy r
Step 1: Calculate r_m for each partitioned segment (the initial input $v(\tau)$ is considered as a single segment at the very first step)
Step 2: If $minimum\ r_m < r$, partition the segment with the minimum r_m into two parts, then go to Step 1.
Step 3: Perform linear regression on each segment and calculate the intersection points of the regression lines.
Return Piecewise linear approximation of $v(\tau)$ by connecting the intersection points obtained in step 3.

Figure 1: Algorithm to Derive Piecewise Linear Approximation for $v(\tau)$

Suppose the downtime distribution is partitioned into m non-overlapping segments denoted by $(s_1, s_2, \dots s_m)$ according to the algorithm in Figure 1. Let γ_i be the downtime (τ) corresponding to the intersection of segments s_i and s_{i+1} $(i = 1,2, \dots m-1)$. Then,

γ_0 and γ_m correspond to the smallest and the largest accumulated downtime values respectively. Therefore, for given vector (n, k, t), we get the approximated linear segments $\hat{v}(\tau)$ as shown in Table 2. Varying the values of (n, k, t), we can get a four-dimensional matrix depicting the corresponding piecewise linear approximation functions of $v(\tau)$.

Segment	s_1 $[\gamma_0, \gamma_1]$	s_2 $[\gamma_1, \gamma_2]$	…	s_m $[\gamma_{m-1}, \gamma_m]$
Piecewise linear approximation for given (n, k, t)	$\hat{v}(\tau)$ for the vector (s_1, n, k, t)	$\hat{v}(\tau)$ for the vector (s_2, n, k, t)	…	$\hat{v}(\tau)$ for the vector (s_3, n, k, t)

Table 2: Linear Segmentation of Downtime Distribution

Once the piecewise linear functional forms for $v(\tau)$ are derived, the expected penalizable downtime can also be readily obtained. For instance, suppose an SLA specifies $t = 90$ days and $= 90\%$. Then $(1 - \alpha)t = 9$ days , is the maximum allowable downtime over the 90 day period for which no penalties are levied. Suppose the downtime of 9 days belongs to the segment *i* where s_i is its upper bound. We can then compute the expected penalizable downtime for a given k as follows: $\int_{(1-\alpha)t}^{t} \hat{v}(\tau)(\tau - (1-\alpha)t)d\tau = \int_{9}^{90} \hat{v}(\tau)(\tau - 9)d\tau$

$$= \int_{9}^{s_i} \hat{v}(\tau, s_i)(\tau - 9)d\tau + \int_{s_i}^{s_{i+1}} \hat{v}(\tau, s_{i+1})(\tau - 9)d\tau + \cdots + \int_{s_{m-1}}^{90} \hat{v}(\tau, 90)(\tau - 9)d\tau \ .$$

We can subsequently compute the expected total cost using equation (1). Similarly, varying the value of k, we can get $Q_1, Q_2, \ldots Q_K$. Therefore, optimal k^* can be obtained by inspection from the series, $(1, Q_1), \ldots (K, Q_K)$. The largest value of *k* inspected can be arbitrarily set to *n.* Since the expected total cost function is convex, one may also start with *k = 1* and progressively increase *k* as the expected total cost decreases and then subsequently starts to increase again. The inspection can be stopped shortly after the uptick in expected total cost is observed. In our

experiments in Section 4, we confirm the convexity of the expected total cost function in *k*. Note that when k is sufficiently large, $\int_0^{\tau_0} \hat{v}(\tau)d\tau = 1$, i.e., no expected penalizable downtime is incurred. Using *k* values larger than this would entail over-provisioning for which there are no additional returns to the provider even though it costs him more. So inspection may also be stopped as soon as a *k* is reached where $\int_0^{\tau_0} \hat{v}(\tau)d\tau = 1$. Figure 2 provides a brief outline of our algorithm to derive the optimal number of backup VMs.

Inputs: n, t, α, $v(\tau)$, approximation accuracy
Step 1: Derive the piecewise linear approximation functions for $v(\tau)$
Step 2: Using (1) compute Q_k, for k = 1,…K
Step 3: $k^* = \arg\min_k(Q_k)$
Return k^*

Figure 2: Algorithm to Derive Optimal Number of Backup VMs

3. Pricing Problem in the Context of SLA Violation

If the α uptime guarantee in the SLA is not met, the client is eligible to a pre-determined penalty. The provider may consider deferring the penalty payout to the end of the next service window, in the hopes of eventually fulfilling the availability guarantee. We now derive the highest price the provider can charge such that the client is sufficiently incentivized to defer the penalty, in the event of SLA violation. Let α_i^{obj} be the target level of service set by the provider in a particular service window i, α_i be the actual service level provided by the end of the service window i, and p_i be the unit price in i. The provider and client may negotiate and partition the entire contract duration into smaller service windows at point of contract formation. At the beginning of each service window i, the provider sets an "objective" service level α_i^{obj}, assuming that this service window is the last chance for her to satisfy the promised service level α from service windows 1 to i. Therefore, α_i^{obj} becomes a function of α and accumulated actual

service level α_{i-1}. Note that at the end of each service window i, the actual service level provided will be either $\alpha_i \geq \alpha_i^{obj}$ (surplus) or $\alpha_i < \alpha_i^{obj}$(deficit).

If $\alpha_i \geq \alpha_i^{obj}$, the availability surplus may be applied towards the next service window $i+1$, which would affect the α_{i+1}^{obj} set by the provider. That is, the provider may set $\alpha_{i+1}^{obj} < \alpha$ in service window $i+1$ and still attain α from i to $i+1$. In case of a deficit ($\alpha_i < \alpha_i^{obj}$), the client has two choices. In the first, the client may accept the penalty payment from the provider at the end of service window *i* and the provider starts the next window with α_{i+1}^{obj} = α, regardless of the actual service level α_i provided in the last window. Alternatively, if the provider sufficiently incentivizes the client with a lower price, she may defer the penalty payment till the next service window $i+1$. In this case, the provider takes the deficit from i into account when calculating α_{i+1}^{obj}. In this case, instead of paying a penalty at the end of service window i, the provider needs to set $\alpha_{i+1}^{obj} > \alpha$ in the service window $i+1$ to attain accumulated α across both service windows. Furthermore, p_{i+1} should be less than p_i in order to appropriately incentivize the client. Otherwise, the client would always prefer immediate penalty payments.

In this paper, we develop the foundations of this model by considering a tractable two-period problem. The client's decision tree can then be depicted as shown in Figure 3. In this problem, the client makes her decision on whether to receive the penalty immediately or to defer it when the actual service level is less than the promised level ($\alpha_1 < \alpha_1^{obj}$) at the end of the first service window. In order to design the incentives appropriately, the provider needs to model the client's expected costs under each choice. The client's expected cost comprises of the payment for *n* servers less expected penalty. Let EC_A be the expected cost in period 2 if the client chooses to accept the penalty in period 1, which is modeled as follows.

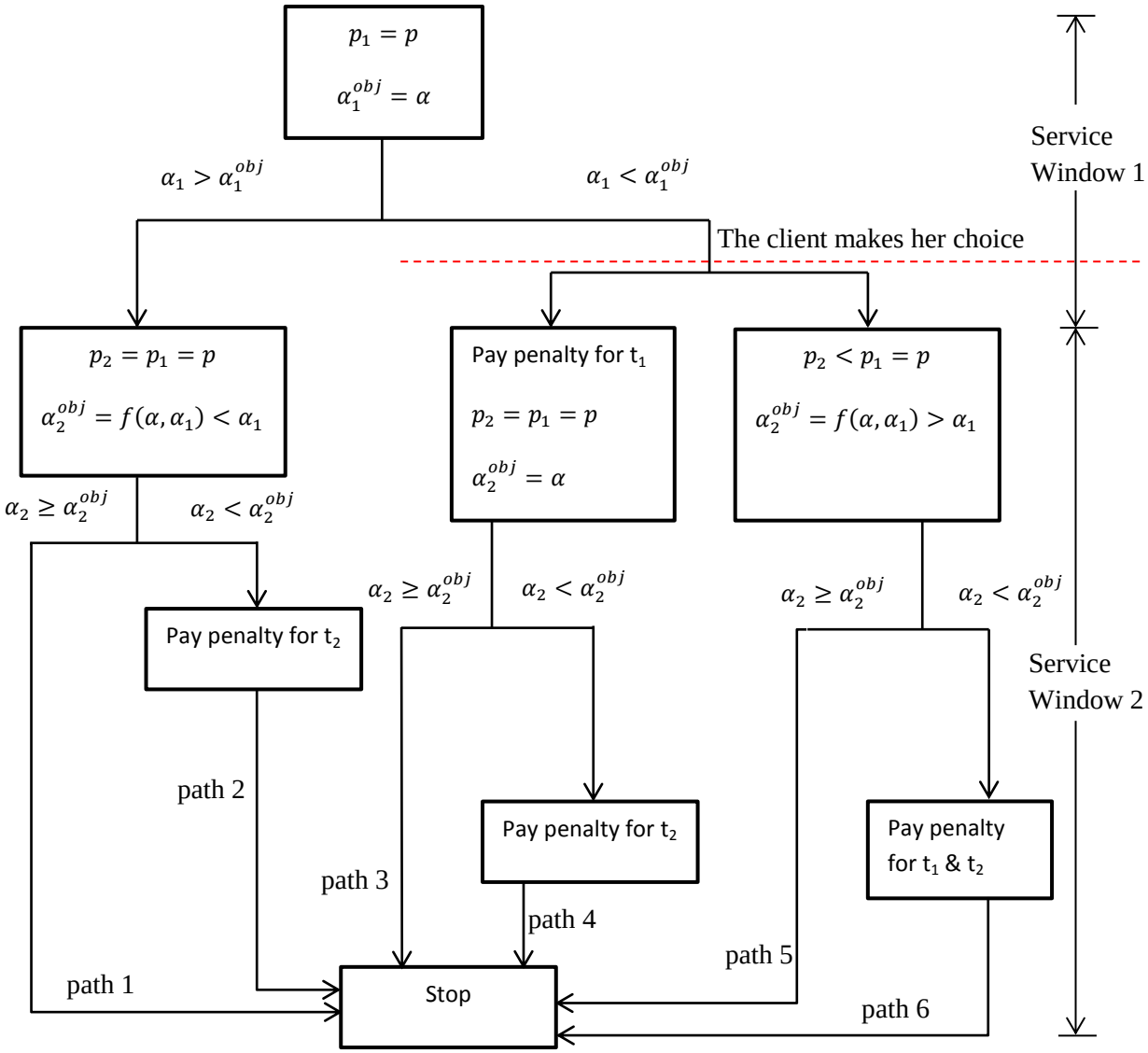


Figure 3:Client's Decision Tree for the Two-Period Problem

$EC_A =$

$prob\left(\alpha_2 \geq \alpha_2^{obj}\right) * expected\ cost(path\ 3) + prob\left(\alpha_2 < \alpha_2^{obj}\right) * expected\ cost(path\ 4)$

$= \int_0^{\left(1-\alpha_2^{obj}\right)*t_2} v(\tau)d\tau * \left(p * n * t_1 - \pi * \int_{(1-\alpha_1^{obj})t_1}^{t_1} v(\tau)(\tau - \left(1-\alpha_1^{obj}\right)t_1)d\tau + p * n * t_2\right) +$

$\int_{\left(1-\alpha_2^{obj}\right)*t_2}^{t_2} v(\tau)d\tau * (p * n * t_1 - \pi * \int_{(1-\alpha_1^{obj})t_1}^{t_1} v(\tau)(\tau - \left(1-\alpha_1^{obj}\right)t_1)d\tau + p * n * t_2$

$- \pi * \int_{(1-\alpha_2^{obj})t_2}^{t_2} v(\tau)(\tau - \left(1-\alpha_2^{obj}\right)t_2)d\tau)$, where $\alpha_1^{obj} = \alpha\ , \alpha_2^{obj} = \alpha.$

Likewise, let EC_B be the expected cost in period 2 if the client chooses to defer the collection of penalties in exchange for a lower price in period 2. This cost is modeled as follows:

$$EC_B =$$

$$prob\left(\alpha_2 \geq \alpha_2^{obj}\right) * expected\ cost(path\ 5) + prob\left(\alpha_2 < \alpha_2^{obj}\right) * expected\ cost(path\ 6)$$

$$= \int_0^{\left(1-\alpha_2^{obj}\right)*t_2} v(\tau)d\tau * (p * n * t_1 + p_2 * n * t_2) + \int_{\left(1-\alpha_2^{obj}\right)*t_2}^{t_2} v(\tau)d\tau * (p * n * t_1 + p_2 * n *$$

$$t_2 - \pi * \int_{(1-\alpha_1^{obj})t_1}^{t_1} v(\tau)(\tau - \left(1-\alpha_1^{obj}\right)t_1)d\tau - \pi * \int_{(1-\alpha_2^{obj})t_2}^{t_2} v(\tau)(\tau - \left(1-\alpha_2^{obj}\right)t_2)d\tau) \quad ,$$

where $\alpha_1^{obj} = \alpha$,

$$\alpha_2^{obj} = \frac{\alpha*(t_1+t_2)-E[\alpha_1]*t_1}{t_2} = \frac{\alpha*(t_1+t_2)-\frac{t_1-E[\tau_1]}{t_1}*t_1}{t_2} \ , E[\tau_1] = \frac{\int_{\left(1-\alpha_1^{obj}\right)*t_1}^{t_1} v(\tau)\tau d\tau}{Prob\left(\alpha_1<\alpha_1^{obj}\right)} = \frac{\int_{\left(1-\alpha_1^{obj}\right)*t_1}^{t_1} v(\tau)\tau d\tau}{\int_{\left(1-\alpha_1^{obj}\right)*t_1}^{t_1} v(\tau)d\tau} \ .$$

The client is indifferent between the two choices when $EC_A = EC_B$. Hence, we get

$$p_2^* = p * n * t_2 + \pi * \int_{(1-\alpha)t_1}^{t_1} v(\tau)(\tau - (1-\alpha)t_1)d\tau * \left(\int_{\left(1-\alpha_2^{obj}\right)*t_2}^{t_2} v(\tau)d\tau - 1\right) - \pi *$$

$$\int_{(1-\alpha)*t_2}^{t_2} v(\tau)d\tau * \int_{(1-\alpha)t_2}^{t_2} v(\tau)(\tau - (1-\alpha)t_2)d\tau + \pi * \int_{\left(1-\alpha_2^{obj}\right)*t_2}^{t_2} v(\tau)d\tau *$$

$$\int_{(1-\alpha_2^{obj})t_2}^{t_2} v(\tau)(\tau - \left(1-\alpha_2^{obj}\right)t_2)d\tau /(n * t_2)$$

where

$$\alpha_2^{obj} = \frac{\alpha*(t_1+t_2)-E[\alpha_1]*t_1}{t_2} = \frac{\alpha*(t_1+t_2)-\frac{t_1-E[\tau_1]}{t_1}*t_1}{t_2} \ , E[\tau_1] = \frac{\int_{\left(1-\alpha_1^{obj}\right)*t_1}^{t_1} v(\tau)\tau d\tau}{Prob\left(\alpha_1<\alpha_1^{obj}\right)} = \frac{\int_{(1-\alpha)*t_1}^{t_1} v(\tau)\tau d\tau}{\int_{(1-\alpha)*t_1}^{t_1} v(\tau)d\tau}$$

Therefore, the client would choose to defer the penalty so long as $p_2 \leq p_2^*$.

4. Experimental Study

Here we present some preliminary computational results highlighting the validity of certain assumptions in our model, emphasizing some of our analytical results and adding insights pertaining to contract design and service provisioning.

4.1 Relationship Between Probability of SLA Violation and the Number of Backup VMs

We derive downtime distributions for $n = 20$ using the sample path algorithm from Du et al (2012) which divides the service window into $t/\Delta t$ consecutive intervals of length Δt of vanishing size. We ran two sets of experiments using t = 10 days and t = 20 days, with Δt = 5 minutes. Each set specified k at 1, 2, 3 and 4. The SLA violation probability, $(\int_{(1-\alpha)t}^{t} v(\tau, k)d\tau)$, drops to zero at k = 4 and beyond. At each level of k, the allowable downtime, $(1-\alpha)t$, is set at four different levels – 5, 10, 15 and 20 mins. Figure 4 shows that the SLA violation probability decreases as the number of backup VMs increases in each set of experiments, confirming our assumption in Section 2.

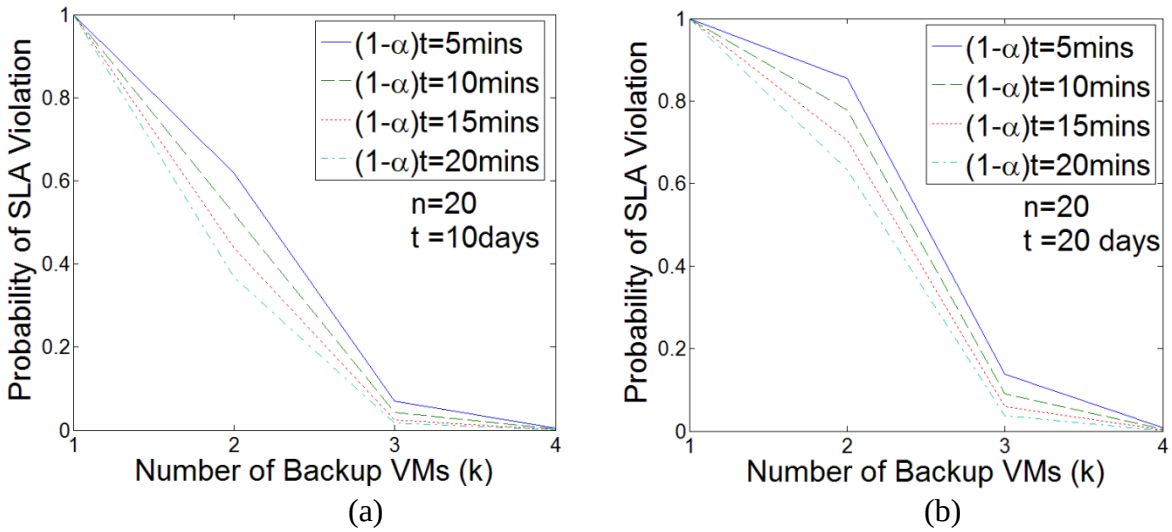


Figure 4: SLA Violation Probability Decreases with Increasing Backup VMs

4.2 Impact of Approximation Accuracy

Accuracy is a specified parameter in the piecewise linear approximation algorithm which affects the number of segments into which $v(\tau)$ is partitioned. As expected, in Figure 5, we can see that in general the number of segments increases along with an increase in the accuracy level. This increase in the number of segments is especially dramatic beyond the accuracy level of 0.9 and particularly so for $k = 1$, regardless of the length of the service window. As k increases, more segments may still be formed at higher accuracy levels but the rate of this increase steadily slows down. This makes sense as the downtime distribution is likely to have higher variance when fewer backup VMs are allocated. While the number of segments seems to be sensitive to approximation accuracy, especially at smaller k, it should be noted that the computation time is negligible with the longest being 0.065 seconds.

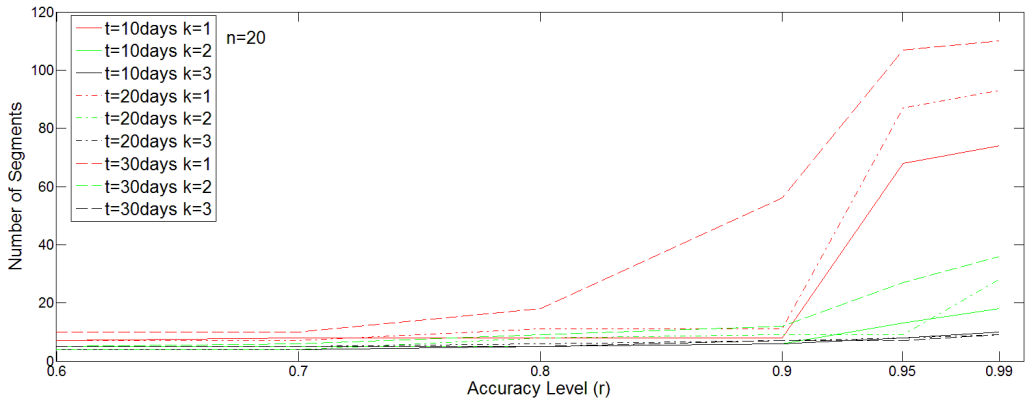


Figure 5: Increase in Number of Segments with Approximation Accuracy

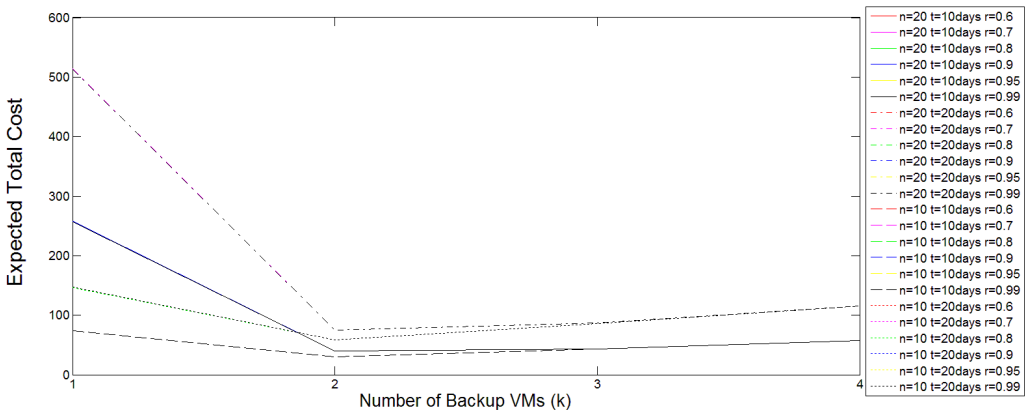
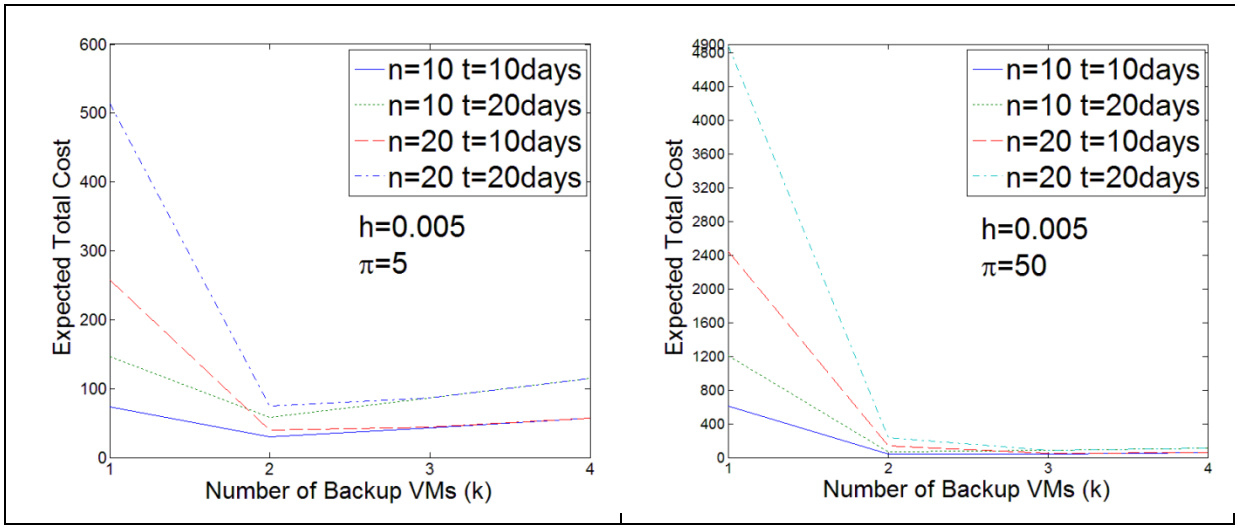


Figure 6: Sensitivity of Optimal k^* to Varying Approximation Accuracy

Next, we explore whether a highly accurate fitting is even necessary for the piece-wise linear approximation of the downtime distribution. If k^* does not change with a change in the accuracy level, then we may conclude that lower accuracy levels (with fewer segments) will suffice for our problem. This is especially true in light of the large number of segments when $r \geq 0.9$. We generated the expected total cost curves for 24 different cases, using $\alpha = 0.999$, n = 10 and 20; t = 10 and 20 days and r set at six levels from 0.6 to 0.99. The expected total costs across the different accuracy levels for a given (n, t, k) combination all lie within a fairly close range, which is the reason for the observing only four curves in Figure 6 in spite of actually plotting all 24 curves. The optimal k corresponds to the minimum expected total cost. As can be seen in Figure 6, regardless of the accuracy level, in each of the four sets of experiments of different (n, t) combinations, k^* is the same. Expected penalty in the expected total cost function is calculated on the basis of the expected penalizable downtime. There is greater volatility in the range pertaining to the allowable downtime i.e., 0 to $(1-\alpha)t$ range than in the range pertaining to the penalizable downtime, i.e., $(1-\alpha)t$ to t. As a result, when greater accuracy is demanded, more segments are formed within the allowable downtime range while the segments in the penalizable downtime range remains largely unaffected. Thus, it appears that the determination of k^* is quite

robust as far as approximation accuracy is concerned. Finally, we also note that the curves in Figure 6 validate the convexity result for the expected total cost function shown in Lemma 2.



(a) (b)

Figure 7: Sensitivity of Optimal k^* to Varying Penalty and Demand

4.3 Impact of Penalty on Backup VM Provisioning

Here we vary penalty to observe its impact on the optimal number of backups VMs provisioned. As Figure 7(a) illustrates, at the lower penalty level ($5 for every 5 minutes of downtime), k^* remains unchanged at two regardless of the number of VMs demanded and length of service window. However, when penalty is increased to $50 for every 5 minutes of downtime, k^* increases to three when n goes from 10 to 20. At that point, the penalty is large enough where it is worthwhile for the provider to allocate one additional VM for the higher demand case. This experiment highlights the importance of setting the penalty appropriately during the negotiation process as it is shown to affect backup VM provisioning, especially at higher demand levels.

5. Conclusion and Future Research

Cloud server failures are quite common, as evidenced by some of the recent well-publicized examples involving AWS affecting the services of Netflix, Instagram, etc. Other large

scale failures are rarely publicized, albeit they are commonplace. In light of this, from the perspective of an Infrastructure as a Service (IaaS) provider, we propose an algorithm to determine the optimal number of backup VMs in order to minimize the expected total cost in the face of SLA violation risks. In the event of an SLA violation, we also study contract adjustment strategies using pricing as a mechanism to entice the client into deferring penalty payments. We are presently extending our experimental study in this context to explore the relationships between the second period price under SLA violation and other contract parameters such as penalty and length of service window. We are also extending our model to more comprehensively capture the multi-parameter contract determination problem as an iterative negotiation process that evolves between the provider and the client.

References

Dean, J. 2009. "Designs, Lessons and Advice from Building Large Distributed Systems". *3rd ACMSIGOPS Intl. Workshop on Large Scale Distributed Systems and Middleware (LADIS)*.

P. Gill, N. Jain, and N. Nagappan. 2011. "Understanding network failures in data centers: measurement, analysis, and implications". *SIGCOMM*. 350–361.

Du, A. Y., Das, S., Qiao, C., Ramesh, R. and Yang, Z. 2012. "Downtime Predictions for Virtual Servers: A Study under Two Check-pointing Scenarios". *Conference on Information Systems and Technology (CIST 2012).*

Shouyi Wang, Wanpracha Art Chaovalitwongse. 2013. "An Efficient and Robust Approach for Automated Online Time Series Segmentation". *Working paper*.

C. Redl, I. Breskovic, I. Brandic, and S. Dustdar. 2012. "Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets." *ACM/IEEE 13th International Conference on Grid Computing*.

M. B. Chhetri, Q. B. Vo, and R. Kowalczyk. 2012. "Policy-Based Automation of SLA Establishment for Cloud Computing Services." *The 12th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing (CCGrid 2012).*

V.C. Emeakaroha et al. 2012. "Towards autonomic detection of SLA violations in Cloud infrastructures." *Future Generation Computer Systems.* 28. 1017–1029.

H. Goudarzi, M. Ghasemazar, and M. Pedram. 2012. "SLA-based Optimization of Power and Migration Cost in Cloud Computing." *The 12th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing (CCGrid 2012).*

L. Wu, S. K. Garg, and R. Buyya. 2011. "SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments." *The 11th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing (CCGrid 2011).*

Ye H., Wong J., Iszlai G., Litoiu M. 2009. "Resource Provisioning for Cloud Computing." *Proceedings of CASCON 2009.*