RC25372 (WAT1304-059) April 23, 2013
Computer Science

IBM Research Report

Configuring Cloud Admission Policies under Dynamic Demand

Merve Unuvar, Yurdaer N. Doganata, Asser N. Tantawi
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 208
Yorktown Heights, NY 10598
USA

Configuring Cloud Admission Policies under Dynamic Demand

Merve Unuvar, Yurdaer N. Doganata, and Asser N. Tantawi
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{munuvar,yurdaer,tantawi@us.ibm.com}

***Abstract*—We consider the problem of admitting sets of, possibly heterogenous, virtual machines (*VMs*) with stochastic resource demands onto physical machines (*PMs*) in a Cloud environment. The objective is to achieve a specified quality-of-service related to the probability of resource over-utilization in an uncertain loading condition, while minimizing the rejection probability of *VM* requests. We introduce a method which relies on approximating the probability distribution of the total resource demand on *PMs* and estimating the probability of over-utilization. We compare our method to two simple admission policies: admission based on maximum demand and admission based on average demand. We investigate the efficiency of the results of using our method on a simulated Cloud environment where we analyze the effects of various parameters (commitment factor, coefficient of variation etc.) on the solution for highly variate demands.**

***Index Terms*—admission control, cloud management, dynamic resource demand, performance comparison, policies, virtual machines**

I. Introduction

In a Cloud system managing the utilization of physical resources with effective admission control policies is essential. Admission control policies ensure that sufficient resources are available in a cluster to provide fail-over protection and to ensure that virtual machine resource reservations are respected [1]. If the additional resources are not reserved, the power-on attempt fails and the fail-over protections cannot be realized. Hence, admission control policies reserve resources to ensure robustness for a potential fail-over and successful power-on procedures.

Each resource in a physical machine in the Cloud system has an over-utilization threshold. When this threshold is exceeded and the resources are over-utilized for longer periods of time, operations of physical machines may be interrupted or migration may become necessary. Over-utilization threshold is the maximum acceptable utilization percentage for a resource. As an example, if the over-utilization threshold is 90%, it is assumed that the operations will not be interrupted, as long as the resource utilization remains below 90%. In addition to over-utilization threshold, the likelihood of the resource being over-utilized is another parameter to be considered for the purpose of admission control. The second threshold is the percentage of time that the over-utilization can be tolerated. In other words, the likelihood of finding the resource over-utilized. Note that the over-utilization threshold can be exceeded for short periods of time. If the likelihood of fail-over or the virtual machine power-on is negligible, exceeding the over-utilization threshold may be tolerated. In this paper, we study the behavior of various admission control policies under dynamic resource demand and introduce a method for configuring admission control policies against over-utilization.

Admission control policies adopt admission criteria by which admission control schema accepts or rejects a request to be placed in the Cloud. In general, admission control schemas are either parameter-based or measurement-based. Parameter-based admission control schemas are based on $apriori$ knowledge of the input requests and provide for deterministic guarantees for uninterrupted Cloud operations. Examples of this type of admission control schemas include admitting a *VM* request based on its resource demand characteristics, such as maximum resource demand or average resource demand. Parameter-based schemas are easy to implement and guarantee Cloud operations under worst-case assumptions. A measurement-based admission control schema, on the other hand, utilizes the estimated resource utilization of physical machines in addition to input *VM* request parameters. In this case, the utilization of a resource is characterized by its stochastic properties and a probabilistic bound can be defined for the potential interruptions of Cloud operations. The probability density function (pdf) of the utilization of a resource is the convolution of all the resource demands of the accepted requests that utilize that resource. In such aggregation of independent resource demands, the probability that the aggregate utilization will reach the sum of the peak demand is infinitesimally small. Using the pdf of the aggregated resource utilization in admission criteria provides for probabilistic guarantees. That is, instead of providing deterministic bound for the worst case scenarios, measurement-based admission control policies guarantee a bound on the probability of over-utilization. In mathematical terms, resource k is stable if its utilization, U_k, satisfies the following constraint,

$$P(U_k > U_k^o) \leq \epsilon^o \qquad (1)$$

where U^o is the over-utilization threshold and ϵ^o is the probabilistic bound on over-utilization.

In this paper, we introduce a measurement-based admission control policy for the Cloud by approximating the pdf of the aggregated resource utilization using the first and second moments. Then, we employ (1) as the admission criterion to decide if the statistical properties of an arriving *VM* request will likely to drive the physical machine into over-utilization. Thus, we enforce an admission criterion that guarantees a bound on the probability of over-utilization. We compare the performance of two parameter-based admission control policies against a measurement-based control policy that we introduce. We also show how to configure the parameter-based admission control policies by using the probabilistic bound of the measurement-based policy. Note that parameter-based approaches provide a deterministic bound against worst case scenarios and their admission criteria do not change with the variations of resource demand. We also show how much parameter-based control policies are sensitive to variations in dynamic resource demand.

The paper is organized as follows. We describe the problem formulation in section II. Two parameter-based and one measurement-based admission control policies are introduced in section III. In section IV we describe how to configure the admission control criteria to reduce the likelihood of over-utilization. Our simulation results are presented in section V. We review related work in section VI and summarize the conclusion and future work in section VII.

II. Formulation

A. Homogenous System

Consider p homogenous *PMs* with K different resources with each having capacity C_k, subjected to a stream of homogenous requests with a Poisson arrival process with rate λ and a generally distributed lifetime with mean τ. A request has a demand D_k for resource k that is generally distributed with distribution function $F_{D_k}(d_k) = Pr[D_k \leq d_k]$, where $d_k \in [D_k^{min}, D_k^{max}]$. Without loss of generality we assume that $D_k^{min} = 0$ and $D_k^{max} > 0$. We denote the mean and standard deviation of the demand for resource k by μ_{D_k} and σ_{D_k}, respectively. Hence, the mean offered load for the k^{th} resource is given by

$$\rho_k = \lambda \; \tau \; \mu_{D_k}. \tag{2}$$

Let Z_k^n denote the sum of n independent k^{th} resource demands. Given that $Z_k^n = n \; D_k$, the mean of Z_k^n is $E[Z_k^n] = n \, \mu_{D_k}$, the variance is $V[Z_k^n] = n \, {\sigma_{D_k}}^2$, and the probability distribution, denoted by $F_{Z_k^n}(z_k)$, is the n-fold convolution of $F_{D_k}(d_k)$.

B. Heterogeneous System

The same notations can be extended to a system that is subjected to heterogenous requests. A request type is characterized by the amount of demand for resources. A request is classified to type $i = 1, 2, \cdots, I$, where I is the number of types. Let the system consist of, as before, p homogenous *PMs* with each having C_k capacity for the resource type k. A type i request has a demand of D_{ik} for the resource k with mean and standard deviation of $\mu_{D_{ik}}$ and $\sigma_{D_{ik}}$, respectively, and a distribution function of $F_{D_{ik}}(d_{ik}) = Pr[D_{ik} \leq d_{ik}]$, where $d_{ik} \in [D_{ik}^{min}, D_{ik}^{max}]$. Similar to the homogenous system, we assume that $D_{ik}^{min} = 0$ and $D_{ik}^{max} > 0$. Let λ_i represents the mean arrival rate for Poisson arrivals and τ_i represent the generally distributed mean lifetime of the i^{th} request type. Hence, the mean offered load for a given type i for the k^{th} resource is given by

$$\rho_{ik} = \lambda_i \; \tau_i \; \mu_{D_{ik}}. \tag{3}$$

The total mean offered load for resource k is:

$$\rho_k = \sum_{i=1}^{I} \rho_{ik} = \sum_{i=1}^{I} \lambda_i \; \tau_i \; \mu_{D_{ik}}. \tag{4}$$

Let $\mathbf{n} = (n_1, n_2, \cdots, n_I)$ denote the number of requests of each of the I types in the system. Given $\mathbf{n}$, the sum of independent k^{th} resource demands in the heterogenous system, $Z_k^{\mathbf{n}}$ is given by, $Z_k^{\mathbf{n}} = \sum_{i=1}^{I} n_i \; D_{ik}$ where n_i represents the number of type i requests. The mean of $Z_k^{\mathbf{n}}$ is $E[Z_k^{\mathbf{n}}] = \sum_{i=1}^{I} n_i \; \mu_{D_{ik}}$, the variance is $V[Z_k^{\mathbf{n}}] = \sum_{i=1}^{I} n_i \; {\sigma_{D_{ik}}}^2$, and the probability distribution, denoted by $F_{Z_k^{\mathbf{n}}}(z_k)$, is the convolution of $F_{D_{ik}}(d_{ik})$ over i.

III. Cloud Admission Policies

An admission controller admits a request into the Cloud based on some policy $\mathbb{P}(\overline{\phi})$, with a set of parameters $\overline{\phi}$ used in admission criteria. The parameter set $\overline{\phi}$ includes elements that characterize the requests, such as the maximum demand, average demand and elements that characterize the resources in the cloud such as resource capacity. As discussed in the Introduction section, if the admission criterion uses fix parameter values based on the characterization of the input request and the Cloud resource, we call the admission policy parameter based. On the other hand, if the admission criterion uses measurements to capture the stochastic nature of the current state, such as the mean and the variance of resource utilization, we call it measurement based.

In this section, for the sake of simplicity, we detail the description of policies for homogenous system only. In section IV, we show how these policies can be extended to a heterogenous system without loss of generality. In parameter based admission control policies, the maximum number of requests that can be accommodated by the Cloud for each resource k is denoted by N_k^{max}. Let n be the number of requests in the *PM* at the time admission policy is applied. Thus, a request is admitted to a *PM* if $n < N_k^{max}$ and is rejected for that particular *PM* otherwise. If a request cannot be placed to any of the *PMs*, it is rejected from the Cloud. By using the resulting request rejection probability δ, the mean utilization of k^{th} resource $\overline{U_k}$ is calculated as

$$\overline{U_k} = \frac{(1 - \delta) \; \rho_k}{C_k \; p}. \tag{5}$$

We consider three policies from the class of admission policies described above. The first two admission control policies are parameter based and the third one is measurement based.

1) Admission based on the maximum value of demand with a commitment factor: $\mathbb{P}_1(\kappa, D_k^{max})$, where D_k^{max} is the maximum demand for resource k, κ is the commitment factor for the resources on a *PM* and $\kappa > 1$.
2) Admission based on the average value of the demand with a commitment factor: $\mathbb{P}_2(\theta, \mu_{D_k})$, where μ_{D_k} is

the average demand of a request for resource k and θ is the commitment factor for the resources on a *PM* and $0 < \theta < 1$.

3) Admission based on a probabilistic bound over-utilization: $\mathbb{P}_3(U_k^*, \epsilon, \mu_k, \sigma_k)$, where U_k^* is the utilization threshold and ϵ_k is the probabilistic bound on the over-utilization probability for resource k such that the probability of over-utilization being above U_k^* is limited to ϵ_k. μ_k and σ_k are the mean and the variance of the utilization for resource k.

Details of these three different admission policies are explained below.

A. Policy 1: Admission based on the maximum value of demand with a commitment factor:

In this admission control policy, the maximum demand values for the resources are taken into account for admission decision. Let us denote this policy with $\mathbb{P}_1(\kappa, D_k^{max})$, where κ is the commitment factor for all types of resources and D_k^{max} is the maximum value of the demand for resource type k. The respective maximum allowed concurrent requests $N_{max}(\mathbb{P}_1)$ in a *PM* for this admission policy is

$$N_{max}(\mathbb{P}_1) = \min_{k \in K} \left\{ \lfloor \frac{\kappa \, C_k}{D_k^{max}} \rfloor \right\}. \tag{6}$$

Here K is the number of resources a request is demanding in a physical machine, *PM*. This policy accepts requests to a *PM* as long as the total number of requests in the *PM* is less than or equal to $N_{max}(\mathbb{P}_1)$ at the time of admission. $N_{max}(\mathbb{P}_1)$ is the maximum number of requests that can be accommodated without over-committing any of the resources beyond κ.

In this policy, the commitment factor, κ is used to prevent the under-utilization of a resource on a *PM*. Since the accepted requests are not always demanding their maximum value, selecting $\kappa > 1$ reduces under-utilization. The performance of the Cloud depends on selecting the κ value properly. As long as the requests are demanding less than their maximum value, particular *PM* is guaranteed to function properly with $\kappa = 1$. This selection, however, will cause under-utilization of the resources. On the other hand, if the *VMs* that are accepted to that *PM* demand their maximum values and when $\kappa > 1$, then resources on *PM* face an over-utilization since total demand exceeds the resource capacity. This problem causes "crashing" on the physical machine if particular resource is a memory. Higher the κ values, the more likely for $\mathbb{P}_1$ to accept *VMs* thus more likely to over-utilize a resource on a *PM*.

Note that selecting a value for κ provides a deterministic bound on the number of requests to be accepted. In order to utilize the resources effectively, κ value needs to be adjusted against the stochastic variations of resource utilization. In practice, it is not common to change the κ values frequently.

B. Policy 2: Admission based on the average value of the demand with a commitment factor

In this admission policy, the average demand of a *VM* request is taken into account for admission decision. Let us denote this policy with $\mathbb{P}_2(\theta, \mu_{D_k})$, where θ is the commitment factor for all resources and μ_{D_k} is the mean value of the demand for the resource type k. The respective maximum allowed concurrent requests $N_{max}(\mathbb{P}_2)$ in a *PM* for this admission policy is

$$N_{max}(\mathbb{P}_2) = \min_{k \in K} \left\{ \lfloor \frac{\theta \, C_k}{\mu_{D_k}} \rfloor \right\}. \tag{7}$$

In this admission control schema, requests are accepted to a *PM* as long as the total number of requests in the *PM* is less than or equal to $N_{max}(\mathbb{P}_2)$ at the time of admission.

The commitment factor, θ is used to prevent the over-utilization of resources on a *PM* since the accepted requests do not always demand their average value. This factor helps to maintain the utilization of resources on a *PM* to be under an upper limit for the times when the admitted requests demand higher than their average value. Usually, θ is selected as: $0 < \theta < 1$. Smaller the θ value, it is more likely to reject the requests thus less likely to over-utilize resources on a *PM*.

Note that $N_{max}(\mathbb{P}_2)$ is a deterministic bound. As in the case of $\mathbb{P}_1$, the parameter of $\mathbb{P}_2$, θ, needs to be adjusted when the demand at a particular instance is different from the average demand, μ_{D_k} in order to maintain the utilization under a certain threshold. Frequent adjustments to θ, however, is not practical as in case of Policy 1.

C. Policy 3: Admission based on a probabilistic bound over utilization

In this admission policy, dynamic nature of a demand for a resource is represented with its mean, μ_{D_k} and standard deviation, σ_{D_k}. Let each *PM* consists of K resources, the utilization of each resource, U_k, is a random variable between [0,1] and characterized by its first and second moments:

U_k: k^{th} resource utilization of PM where $k \leq K$
μ_k: Mean of U_k
σ_k: Standard deviation of U_k

We approximate the probability distribution function (pdf) of U_k as a Beta distribution since Beta distribution is a good approximation for the maximum entropy probability distribution for all classes of distributions with the same first and second moments (see Appendix A for the reasoning of this assumption). Beta distribution is a family of continuous probability distributions defined on the interval [0,1] by two positive shape parameters, denoted by α and β. Hence, we also characterize the utilization U_k with two parameters, α and β, associated with the first and second moments of U_k. For more information on Beta distribution, see Appendix B.

Admission criterion for Policy 3, $\mathbb{P}_3(U_k^*, \epsilon_k, \mu_k, \sigma_k)$, utilizes U_k^*, ϵ_k, μ_k and σ_k to make an admission decision. Here U_k^* is the over-utilization threshold, ϵ_k is the probabilistic bound on over-utilization, μ_k and σ_k are the estimated mean and the standard deviation of the measured utilization of resource k after the request arrival. The admission criterion for Policy 3 is given by

$$F_{Z_k^n}(U_k^*) \geq (1 - \epsilon_k) \tag{8}$$

If equation (8) is not satisfied with the new request arrival, the request is rejected. The respective maximum allowed concurrent requests $N_{max}(\mathbb{P}_3)$ in a *PM* for Policy 3 is expressed as:

$$N_{max}(\mathbb{P}_3) = \min_{k \in K} \left\{ \sup \left\{ n \mid F_{Z_k^n}(U_k^*) \geq (1-\epsilon_k) \right\} \right\}. \quad (9)$$

Here $U_k \in [0,1]$ is the utilization of resource k and $U_k \sim Beta(\alpha_k, \beta_k)$. As described in Appendix B, the corresponding α_k and β_k values are found from the estimated mean and variance values of the utilization of resource k in the *PM* as:

$$\alpha_k = \bar{R}_k \left(\frac{\bar{R}_k(1-\bar{R}_k)}{\bar{S}_k^2} - 1 \right) \quad (10)$$

$$\beta_k = (1-\bar{R}_k) \left(\frac{\bar{R}_k(1-\bar{R}_k)}{\bar{S}_k^2} - 1 \right) \quad (11)$$

where $\bar{R}$ and $\bar{S}$ are estimations for μ_k and σ_k respectively. Hence the cumulative distribution function $F_{Z_k^n}(U_k)$ for the utilization of resource k is expressed as:

$$F_{Z_k^n}(U_k, \alpha_k, \beta_k) = B(U_k, \alpha_k, \beta_k)/B(\alpha_k, \beta_k) \quad (12)$$

where B is the *Beta* function. Note that $N_{max}(\mathbb{P}_3)$ is not a deterministic bound, but it changes as the mean and the variance of the utilization change dynamically. Unlike $\mathbb{P}_1$ and $\mathbb{P}_2$, $\mathbb{P}_3$ does not need to be adjusted, since $\mathbb{P}_3$ is dynamically adjusted with the measured statistical properties of resource utilization. The predefined thresholds for U_k^* and ϵ_k are set by the Cloud manager depending on the specifications of the physical machine.

IV. Configuration of Admission Policies

A. Homogenous System

So far, we have obtained maximum allowed concurrent requests $N_{max}(\mathbb{P}_k)$ in a physical machine for each policy as:

1) $N_{max}(\mathbb{P}_1) = \min_K \left\{ \lfloor \frac{\kappa\, C_k}{D_k^{max}} \rfloor \right\}$
2) $N_{max}(\mathbb{P}_2) = \min_K \left\{ \lfloor \frac{\theta\, C_k}{\mu_{D_k}} \rfloor \right\}$
3) $N_{max}(\mathbb{P}_3) = \min_K \left\{ \sup \left\{ n \mid F_{Z_k^n}(U_k^*) \geq (1-\epsilon_k) \right\} \right\}$.

The first two are deterministic bounds for the number of concurrent requests that can be accommodated in the Cloud and the third one is a probabilistic bound. The first bound guarantees accommodation in the Cloud for requests with maximum demand D_k^{max} on resource k with a commitment factor κ. Similarly, the second bound guarantees accommodation in the Cloud for requests with the average demand μ_{D_k} on resource k with a commitment factor θ. Note that these deterministic bounds cannot guarantee that resources will not be over-utilized. Also, there is no guarantee that the requests will be not be rejected as long as resources are available at the time arrival. This is merely because the first and the second admission control policies do not take into account the dynamic nature of the resource utilization of existing virtual machines in the Cloud. The third policy, however, uses the measured utilization statistics in the admission criterion, thus $N_{max}(\mathbb{P}_3)$ is continuously adjusted.

For the same request arrival process, all three policies perform the same when $N_{max}(\mathbb{P}_1) = N_{max}(\mathbb{P}_2) = N_{max}(\mathbb{P}_3)$. In this case, the rejection rate, as well as the overload factor, will be the same for all policies. As the statistical characteristics of PM resource utilization change, the over-utilization probability in Policy 3 remains below ϵ_k. This is not the case for Policy 1 and 2. Regardless, over-utilization probabilities for the first two policies can be controlled by configuring κ and θ values using the measured resource utilization statistics and $N_{max}(\mathbb{P}_3)$.

Let the estimated mean and standard deviation of utilization be $\bar{\mu_k}$ and $\bar{\sigma_k}$ for resource k, respectively . The next request with demand statistics μ and σ will increase the mean utilization to $\bar{\mu_k}^* = \bar{\mu_k} + \mu$ and the variance to $\bar{\sigma_k^2}^* = \bar{\sigma_k^2} + \sigma^2$. As a result, over-utilization probability after the new arrival is found as in (12),

$$F_{Z_k^n}(U_k^*, \alpha_k^*, \beta_k^*) = Pr(U_k > U_k^*). \quad (13)$$

Here α_k^* and β_k^* are found from the estimated mean and variance, $\bar{\mu_k}^*$ and $\bar{\sigma_k^2}^*$, respectively, as explained in equations (10) and (11). If equation (13) is greater than the probabilistic bound ϵ_k, then the request is rejected.

Assume that the resource utilization demand for all independent arrival requests have the same mean and variance, μ and σ, the probability of over-utilization for N concurrent requests is given by

$$F_{Z_k^n}(U_k^*, \alpha_k(N), \beta_k(N)) = Pr(U_k(N) > U_k^*). \quad (14)$$

Here $U_k(N)$ is the resource utilization of N concurrent requests for the resource k. The parameters of the beta function $\alpha_k(N)$ and $\beta_k(N)$ are obtained from (10) and (11) by substituting $\bar{R} = N\mu_k$ and $\bar{S}^2 = N\sigma_k^2$. This yields the probabilistic bound $N_{max}(\mathbb{P}_3)$ as defined above. $N_{max}(\mathbb{P}_3)$ is the smallest number of concurrent requests that causes over-utilization for any resource k. Once $N_{max}(\mathbb{P}_3)$ is computed with estimated values of μ_k and and σ_k, then the commitment factors for Policy 1 and 2 are found as follows:

$$\kappa = \frac{N_{max}(\mathbb{P}_3)\; D_k^{max}}{C_k}$$
$$\theta = \frac{N_{max}(\mathbb{P}_3)\; \mu_{D_k}}{C_k}. \quad (15)$$

Since $\mathbb{P}_3$ considers the dynamic demand, it can be used to tune the parameters of the other policies. This forces the system to run with same rejection rate, thus with the same overload factor for all three policies. As the probability distribution function of requests, $F_{Z_k^n}$, changes, new $N_{max}(\mathbb{P}_3)$ values are calculated, thus new parameters for $\mathbb{P}_1$ and $\mathbb{P}_2$ are generated.

We have also shown how to compute the distribution of the number of requested arrivals in the Cloud by using independent arrival assumptions in Appendix D. This approximation may be useful to estimate the under-utilization probabilities which we left out of the scope of this paper.

B. Heterogenous System

For the heterogenous case, admission controller still admits a request into a *PM* based on n, the number of requests in the *PM* at the time of admission policy $\mathbb{P}(\phi)$ is applied. Denote the resulting request rejection probability by δ_i, and the mean utilization of k^{th} resource by $\overline{U_k}$. It is given by

$$\overline{U_k} = \frac{\sum_{i=1}^{I}(1-\delta_i)\; \rho_{ik}}{C_k\;\; |PM|}. \quad (16)$$

The three policies that are described before are still applicable in the heterogenous system with some modifications. The N_{max} value that is calculated by policies now have a list of arrays where each array contains feasible combination of acceptable numbers of each type of *VMs*. Let N^j_{max} represent one of the possible combination of feasible maximum vector for different types of *VMs* such that $N^j_{max} = \{n^1, \ldots, n^I\}$, where n^i represents the number of allowed type i requests in the *PM* and there is a total of I types of requests. The $\overline{N_{max}} = \{N^1_{max}, \ldots, N^J_{max}\}$ represents all possible combinations of feasible admissions that will reveal the same maximum value and there is a total of J number of lists that satisfy the total maximum number of requests including all types.

Let the policies be now adjusted per request type i such that; $\mathbb{P}_1(\kappa, D^{max}_{ik})$, $\mathbb{P}_2(\theta, \mu_{D_{ik}})$, and $\mathbb{P}_3(U^*_k, \epsilon, \mu_{D_{ik}}, \sigma_{D_{ik}})$. Each of the above policies per type determines its respective maximum value on the allowed concurrent request type i by assuming that the number of other types of requests placed in the same physical machine is zero. In particular, we have an upper bound on n^i values such that:

1) $n^i(\mathbb{P}_1) \leq \min_K \left\{ \lfloor \frac{\kappa\, C_k}{D^{max}_{ik}} \rfloor \right\}$
2) $n^i(\mathbb{P}_2) \leq \min_K \left\{ \lfloor \frac{\theta\, C_k}{\mu_{D_{ik}}} \rfloor \right\}$
3) $n^i(\mathbb{P}_3) \leq \min_K \left\{ \sup \left\{ n \mid F_{Z^n_k}(U^*_k) \geq (1 - \epsilon_k) \right\} \right\}$

By using the relationships between upper limits and D_{ik}s, we can write the list of $\overline{N_{max}} = \{N^1_{max}, \ldots, N^J_{max}\}$ and construct admission decision based on these vectors.

V. Numerical Results

A. Description of setup

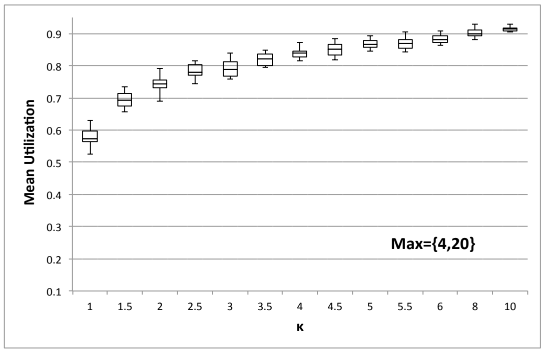
We consider a Cloud with 15 *PMs*, each with a CPU capacity of 80 cores. There are two types of *VM* requests, small and large depending on their size hence we are working with a heterogenous system. A small *VM* requires 2 CPU cores and a large *VM* requires 10 CPU cores on average. The variation in demand is characterized by the coefficient of variation with respect to demand and is in the range of 0.5 to 5 with 0.5 increments. The over-utilization threshold for CPU is set to 95% and it is not allowed for the utilization to violate this threshold more than 1% of time.

We simulate the above system, starting from an empty system, leading up to an offered loading of 95% average *PM* CPU utilization. *VM* requests arrive as a Poisson process and the lifetime of a *VM* is exponentially distributed in such a way to maintain the 95% average load. The mix of small and large *VMs* is governed by a Bernoulli process with probabilities 0.3 and 0.7, respectively.

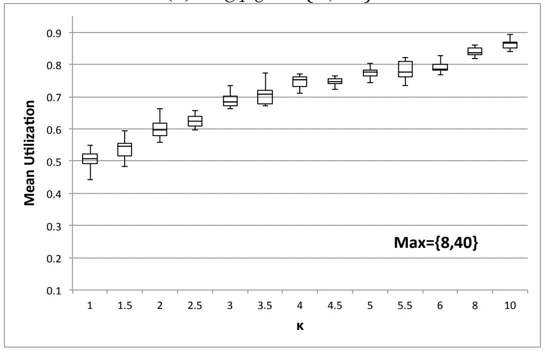
We implemented the three policies under study. First, we investigated the impact of the parameters of the policy on its performance. Then, we compared the behavior of the policies with respect to demand variation. Later, we showed how to configure parameter based policies by using Policy 3.

B. Effect of parameters on mean utilization

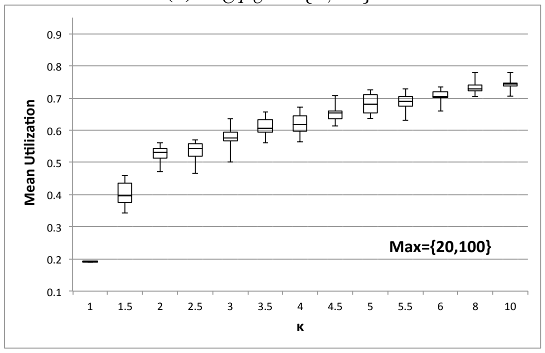
In order to compare the performances of parameter-based policies $\mathbb{P}_1$ and $\mathbb{P}_2$ in terms of utilization we vary tunable parameters and observe the behavior of mean utilization. For $\mathbb{P}_1(\kappa, D^{max}_k)$, we varied κ from 1 to 10 with increments of 0.5 for three different D_{max} values: $\{4, 20\}, \{8, 40\}, \{20, 100\}$, where each tuple represents the value of D_{max} for small and large *VMs*, respectively. We obtained the rejection rates from the simulator and calculated the mean utilization using (16). As depicted in Fig. 1(a), (b), (c) we observe an increase in mean utilization with the increase in the commitment factor, for all maximum demand values. Further, we observe that for the same commitment factor κ, as the D^{max}_{CPU} decreases, utilization increases since smaller D^{max}_{CPU} values admit more thus the policy rejects less number of requests.



(a) $D^{max}_{CPU} = \{4, 20\}$.



(b) $D^{max}_{CPU} = \{8, 40\}$.



(c) $D^{max}_{CPU} = \{20, 100\}$.

Fig. 1. Utilization in policy ($\mathbb{P}_1$) as a function of commitment factor κ

For $\mathbb{P}_2(\theta, \mu_{D_k})$, we varied θ in the range of $[0.5, 1]$ with increments of 0.05 for the mean demand of $\{2, 10\}$. The mean utilization as a function of θ is depicted in Fig. 2. We observe that an increment in θ value allows the admission controller to accept more requests, thus increasing the mean utilization.

Fig. 1 and Fig. 2 show the corresponding resource utilization for selected policy parameters and provide a benchmark for Cloud administrators to select parameter values for the desired

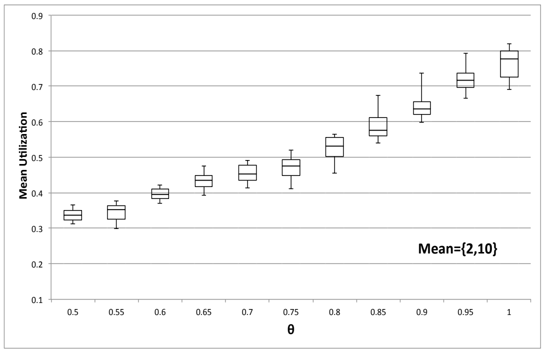


Fig. 2. Utilization in policy $(\mathbb{P}_2)$ as a function of θ

utilization values. As an example, when D_{CPU}^{max} is $\{4,20\}$ for different *VM* types, the average utilization is limited to 0.90 if $\kappa = 8$ is chosen as seen in Fig. 1 for the 95% average load in Policy 1. Similarly, the corresponding θ value for the desired average utilization for Policy 2 is found from Fig. 2.

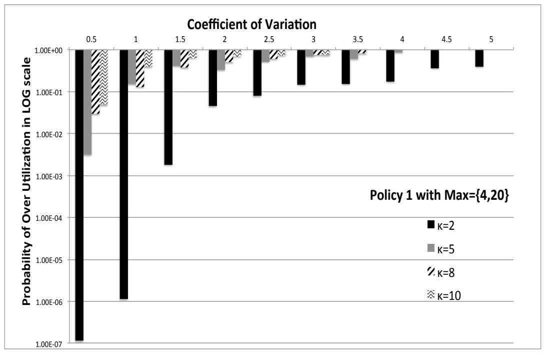
$\mathbb{P}_3$, on the other hand, is not parameter-based and it ensures the given quality-of-service and keeps the utilization below the desired level based on the measured resource utilization statistics.

C. Effect of demand variation on probability of over-utilization

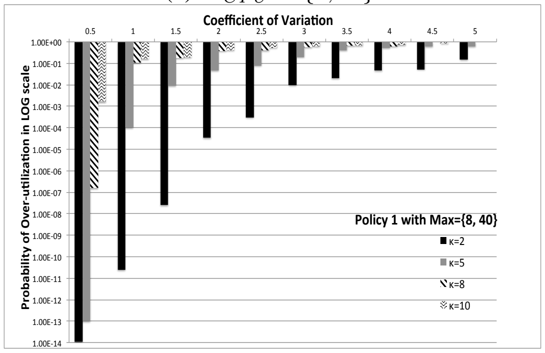
In $\mathbb{P}_1$ and $\mathbb{P}_2$, resource demand variations are not taken into account in selecting the policy parameters. In this section, we investigate the impact of variation in CPU demand on probability of over-utilization with respect to fixed parameters of $\mathbb{P}_1$ and $\mathbb{P}_2$.

For $\mathbb{P}_1$, we fixed the commitment factor, κ, to $\{2, 5, 8, 10\}$ for various levels of fixed D_{CPU}^{max}: $\{4, 20\}, \{8, 40\}, \{20, 100\}$ values. We obtained the average number of accepted requests from the simulator and calculated the convoluted probability distribution function as described in section II. Fig. 3(a), (b), (c) depicts that for all κ and D_{CPU}^{max} values, as the coefficient of variation increases, the probability of over-utilization increases due to increase in the uncertainty. (Fig. 3 shows the probabilities in logarithmic scale.) For the same κ values, higher D_{CPU}^{max} values result in rejecting more requests thus yielding less over-utilization probability. For instance, the probability of over-utilization for the same coefficient of variation is less in Fig. 3(c) than Fig. 3(a). Moreover, when the coefficient of variation increases for fixed D_{CPU}^{max} values (any D_{CPU}^{max}), κ values need to be decreased to reduce the chance of over-utilization. This experiment clearly indicates that the impact of demand variations can be fenced by selecting appropriate κ values in $\mathbb{P}_1$ and likelihood of higher-utilization is reduced.

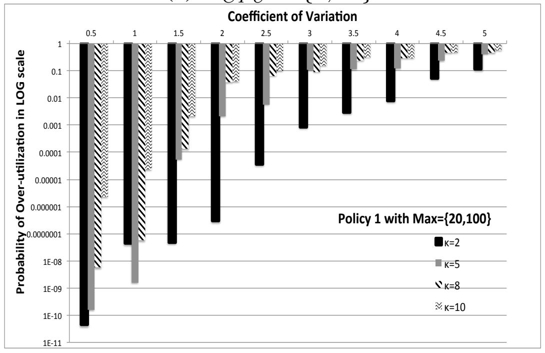
Similarly, in $\mathbb{P}_2$, as the coefficient of demand variation increases, the probability of over-utilization increases regardless of the commitment factor, θ. Fig. 4 also shows that for higher θ values, the probability of over-utilization is higher. (Fig. 4 is in logarithmic scale as well). When the coefficient of variation increases, θ values need to be decreased in order to reduce the probability of over-utilization. Thus, in order to limit the probability of over-utilization below a threshold, ϵ for $\mathbb{P}_2$, the commitment factor, θ, needs to be tuned accordingly.



(a) $D_{CPU}^{max} = \{4, 20\}$.



(b) $D_{CPU}^{max} = \{8, 40\}$.



(c) $D_{CPU}^{max} = \{20, 100\}$.

Fig. 3. Probability of over-utilization in policy $(\mathbb{P}_1)$, $P(U_{CPU} \geq 0.95)$, as a function of the coefficient of variation

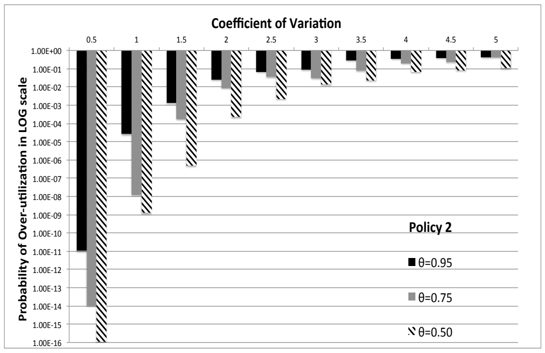


Fig. 4. Probability of over-utilization in policy $(\mathbb{P}_2)$ as coefficient of variation increases

Since $\mathbb{P}_3$ is a measurement-based policy, it always ensures that the probability of over-utilization is under ϵ, which is 0.01 for this specific example, regardless of changes in coefficient of variation. Fig. 5 illustrates that over-utilization probability in $\mathbb{P}_3$ does not depend on demand variations.

Unlike $\mathbb{P}_1$ and $\mathbb{P}_2$, $\mathbb{P}_3$ monitors the system and makes the acceptance decision based on the state of the Cloud in terms of over-utilization probability. Not only it ensures the stability of the Cloud but also it can be used to tune the parameters of $\mathbb{P}_1$ and $\mathbb{P}_2$ to keep the over-utilization probability under ϵ threshold. The relationship between $\mathbb{P}_3$ and other policies is described in the next section.

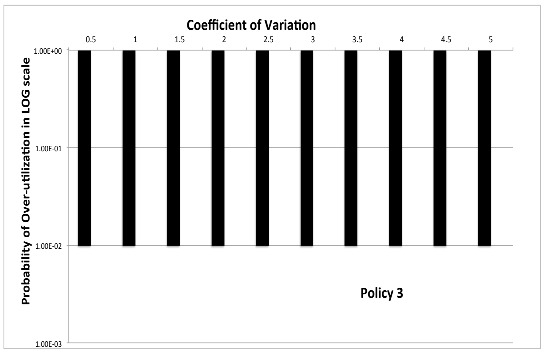


Fig. 5. Probability of over-utilization in policy $(\mathbb{P}_3)$ as coefficient of variation increases

D. Relationship between policies

As previous experiments indicate, admissions with $\mathbb{P}_1$ and $\mathbb{P}_2$ with fixed parameters will under-perform as the coefficient of variation of demand changes, whereas $\mathbb{P}_3$ always ensures the required quality-of-service for the Cloud. The commitment factor, κ, for $\mathbb{P}_1$ and θ for $\mathbb{P}_2$ need to be adjusted as the variation in the CPU demand changes in order to maintain the quality-of-service on over-utilization. One can use the probability of over-utilization function for $\mathbb{P}_3$ to adjust the parameters of the other two policies. We obtained the average number of accepted requests by $\mathbb{P}_3$ from the simulator and used (15) to obtain various κ values for different levels of variations. Fig. 6 shows the relationship between the commitment factor, κ, and the coefficient of variation for different D_{CPU}^{max} values for $\mathbb{P}_1$. We note that the probability of over-utilization is kept under ϵ in $\mathbb{P}_1$ by employing the same number of concurrent requests allowed $N_{max}(\mathbb{P}_3)$ that we found for $\mathbb{P}_3$ in $\mathbb{P}_1$. For instance, by employing this approach, as the coefficient of variation increases from 0.5 to 2.5 for $D_{CPU}^{max} = \{4, 20\}$, κ value needs to be dropped from 6 to 2 in order to keep the probability of over-utilization under 0.01. The relationship between κ and the coefficient of variation follows an exponential behavior, and as D_{CPU}^{max} values increases, the degree of the exponent decreases.

Similarly, admissions under $\mathbb{P}_2$ is not responsive to changes in coefficient of variation unless the commitment factor, θ, is adjusted. Fig. 7 shows the relationship between θ and the coefficient of variation for $\mathbb{P}_2$. We note that $\mathbb{P}_2$ maintains the probability of over-utilization under ϵ by again using the probability of over-utilization function of $\mathbb{P}_3$. There is

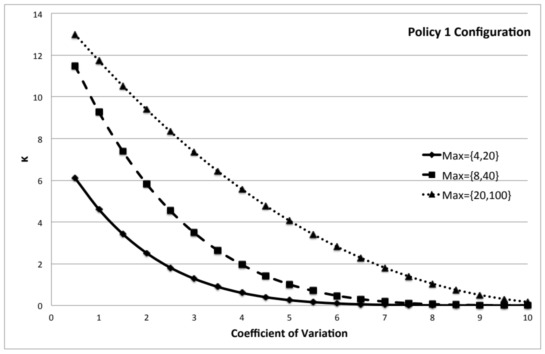


Fig. 6. Maintaining $(\mathbb{P}_3)$ utilization in $(\mathbb{P}_1)$ by tuning commitment factor

almost a linear relationship between these two attributes for $\mu = \{2, 10\}$. For instance, when the coefficient of variation increases from 2 to 4, θ value needs to be dropped from 0.6 to 0.25.

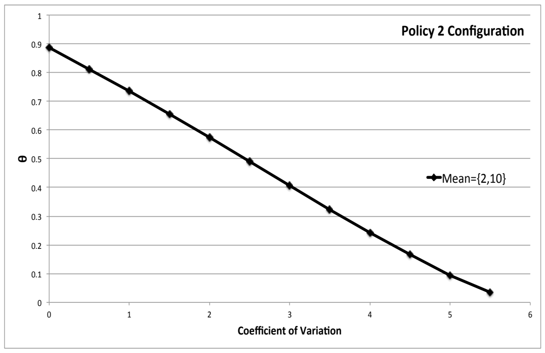


Fig. 7. Maintaining $(\mathbb{P}_3)$ utilization in $(\mathbb{P}_2)$ by tuning commitment factor

VI. Related Work

The problem of admission control in data centers, and in the Cloud in general, has been addressed from different angles. In [2], a data center that is subjected to a stream of *VM* requests of different types is considered. Due to the large size of the problem, an approximate dynamic programming technique is proposed. The allocation of *VMs* on *PMs* is performed assuming a fixed demand of resources. But, in practice, the demand varies over time, suggesting the inclusion of the variability in demand when admitting a *VM*. In [3], this variability in CPU usage enabled the overbooking of resources.

In the area of task allocation in distributed systems, the variability in resource demand has also been addressed. In [4], a genetic algorithm is employed to dynamically schedule heterogeneous tasks on heterogeneous processors. A different approach is proposed in [5] that is based on annealing models and simulated annealing.

Fundamentally, admission control is similar in many ways to loss systems. In such systems, a collection of resources with some capacities are provided to a stream of requests, where each request specifies its resource demand. Typically, the resource demand is fixed. Even in this case, and given Poisson arrivals of requests and generally distributed request residence times, the probabilistic analysis, say to evaluate the loss probability, which corresponds to the rejection rate in admission control, is challenging due to the need to compute a normalizing constant [6]. To overcome this challenge, asymptotic approximations have been obtained for load factors that correspond to light, critical, and overload conditions [7].

As for the case of stochastic demand, there has been extensive research in the area of stochastic bin packing. For an overview see [8]. An example of packing items with random sizes is given in [9]. An application in stochastic load balancing is presented in [10].

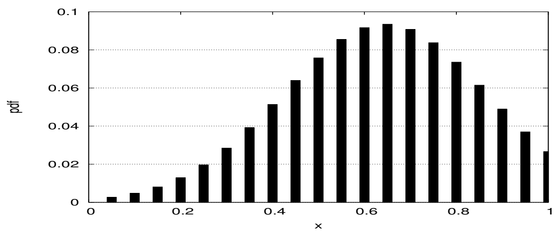
Back to the Cloud computing environment, there the variability in demand depends on the nature of the resource. For example, for the CPU resource on a *PM*, the total CPU demand of all *VMs* hosted on that CPU may very well exceed 100%. This will simply result in congestion and degraded performance. However, for the memory resource, such an overload is unacceptable since it may lead to crashing. Hence, a method for estimating the probability of overload is crucial in deciding on admitting a *VM* into the system. And, that is exactly the goal of this paper.

VII. Conclusion and future work

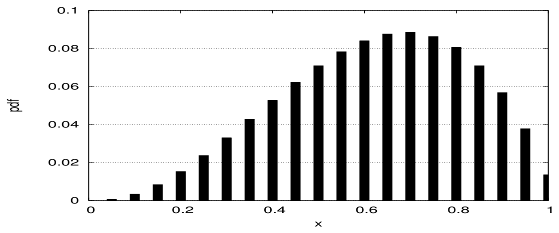
We address the problem of making admission decision for Cloud on sets of, possibly heterogenous, *VMs* with dynamic resource demands. The admission problem is trivial if the demand is assumed to be fixed during the admission process hence quality-of-service on over-utilization is satisfied as long as there is enough capacity on the *PM*. For *VMs* with dynamic demand, it is not as simple to consider only the realization of the demand during the admission time however it is necessary to take into account the demand distribution during its lifetime to be able to meet the predefined quality-of-service throughout its lifetime on a *PM*.

We have introduced a method for admitting sets of *VMs* with dynamic (stochastic) resource demands in Cloud while maximizing the utilization and achieving a specified quality-of-service to avoid over-utilization of resources in the system. Our solution method approximates the probability distribution of total resource demand on *PMs* as beta distribution and admits the *VMs* into the system by estimating the probability of over-utilization. We compare our admission policy with two parameter based policies where each considers the resource demand as fixed. The experiments indicate that probability of over-utilization increases significantly with the increase in demand variation for both of these policies when the parameters are fixed to a predetermined value. Our policy, $\mathbb{P}_3$, maintains robust performance with highly variate demands thus always ensures the satisfaction of quality-of-service for different distributions of demand. Also, importantly, our method can be used to adjust the parameters of $\mathbb{P}_1$ and $\mathbb{P}_2$ to maintain the desired quality-of-service level.

As a future work, we intend to evaluate the system on a larger scale. Another extension of this work that we wish to pursue is automating the adjustment of parameters for a given parameter based policy by using our method. Prioritizing admissions by assigning different quality-of-service levels to requests and optimizing such system is another open research study that we would like to investigate in the future.



(a) Maximum entropy distribution.



(b) Beta distribution.

Fig. 8. Fitting maximum entropy and Beta distributions.

Appendix A
Choice of Beta Distribution

We select a probability distribution for the utilization of resource k, U_k, from the first and second moments of the distribution. In general, when given the first few moments of a probability distribution, the most likely distribution function is the one that maximizes entropy [11]. As an example, given a mean of 0.6 and a standard deviation of 0.2 for a distribution in [0,1], we plot the distribution which maximizes entropy in Fig. 8(a). Matching the first and second moments, we plot the corresponding Beta distribution in Fig. 8(b). As noted, the Beta distribution approximates well the maximum entropy distribution.

Appendix B
Parameters of the Beta Distribution

Beta distribution is a family of continuous probability distributions defined on the interval [0,1] by two positive parameters, denoted by α and β with following probability density function:

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \tag{17}$$

where B is the beta function. Hence, we also characterize the utilization x with two parameters, α and β, associated with the first and second moments of x. First moment of the beta distribution: Using the method of moments estimator, the sample mean $\bar{R}$ and the variance $\bar{S}^2$ of the observed utilization are set to the population mean and the variance and expressed in terms of the mean and the standard deviation of the associated beta distribution as:

$$\bar{R} = \frac{\alpha}{\alpha+\beta} \tag{18}$$

$$\bar{S}^2 = \frac{\alpha\beta}{(\alpha+\beta+1)(\alpha+\beta)^2} \tag{19}$$

From (18) and (19), α and β are solved in terms of population mean and variance as:

$$\alpha = \bar{R}\left(\frac{\bar{R}(1-\bar{R})}{\bar{S}^2} - 1\right) \tag{20}$$

$$\beta = (1-\bar{R})\left(\frac{\bar{R}(1-\bar{R})}{\bar{S}^2} - 1\right) \tag{21}$$

For the estimated α and β values, the cumulative distribution function can be expressed in terms of incomplete beta function $F(x;\alpha,\beta)$ as:

$$F(x;\alpha,\beta) = \frac{B(x;\alpha,\beta)}{B(\alpha,\beta)} = I_x(\alpha,\beta) \tag{22}$$

Here, (22) gives the x^{th} percentile of the beta distribution. As an example, the 90^{th} percentile of the resource utilization is expressed as $F(0.9,\alpha,\beta) = I_{0.9}(\alpha,\beta)$ which is the probability that the utilization is below 90%.

Appendix C
Adding a VM

The impact of a virtual machine to a physical machine can be measured by how much the likelihood of exceeding the over-utilization threshold is increased. If μ and σ are the first and second moments of the utilization demand, respectively, on resource R_k by the new arrival, then the new mean and variance value for resource utilization after the arrival is found as:

$$\bar{R}' = \bar{R} + \mu \tag{23}$$

$$\bar{S^2}' = \bar{S}^2 + \sigma^2. \tag{24}$$

Here, we assume that the utilization demand of the newly arriving *VM* is independent of the utilization of the physical machine. The probability density function for the new utilization is characterized by α' and β' values associated with the new mean and variance values. If the contribution of a virtual machine to the first and second moments of a resource changes the α and β values to α' and β', then the increase in the 90^{th} percentile due to this particular virtual machine is found as: $I_{0.9}(\alpha,\beta) - I_{0.9}(\alpha',\beta')$. As an example if the newly presented virtual machine reduces the 90% percentile of the utilization from $I_{0.9}(\alpha,\beta) = 0.25$ to $I_{0.9}(\alpha',\beta;) = 0.20$, then it is 5% more likely that the resource utilization will go over 90% utilization with the new arrival. This is the impact of a virtual machine on a physical machine.

Appendix D
Distribution of the number of requests accommodated in the Cloud

Given an N_{max} we can evaluate the stationary distribution of the number of requests in the system by noting that the equivalent model is that of an $M/G/m/m$ loss queueing system [12], where $m = |PM| \times N_{max}$. In particular, in the case of exponentially distributed lifetimes, we have an $M/M/m/m$ loss queue whose stationary distribution is given by [13]

$$\pi_n = \pi_0 \, \frac{(\lambda\,\tau)^n}{n!}, \quad n = 1, 2, \cdots, m, \tag{25}$$

and π_0 is given by the normalizing constant

$$\pi_0 = \left[\sum_{n=0}^{m} \frac{(\lambda\,\tau)^n}{n!}\right]^{-1}. \tag{26}$$

The average occupancy is given by

$$\overline{N} = \sum_{n=0}^{m} n\;\pi_n. \tag{27}$$

Thus, given N_{max}, we have the average utilization of resource k as:

$$\overline{U_k} = \frac{\overline{N}\;\mu_{D_k}}{C_k\;|PM|} \tag{28}$$

and the rejection probability

$$\delta = \pi_{N_{max}}. \tag{29}$$

Define an overload factor $\eta(U_k^*) = Pr[U_k > U^*]$. Since $U_k = N\,D_k$, then by conditioning on N we get

$$\begin{aligned}\eta_k(U_k^*) &= \sum_{n=1}^{N_{max}} \pi_n \; Pr[U_k > U_k^* | n] \\ &= \sum_{n=1}^{N_{max}} \pi_n \; \left(1 - F_{Z_k^n}(U_k^*|n)\right). \end{aligned} \tag{30}$$

Acknowledgment

We would like to thank Malgorzata Steinder for valuable discussions.

References

[1] Vmware vsphere ® high availability 5.0 deployment best practices. [Online]. Available: www.vmware.com/files/pdf/techpaper/vmw-vsphere-high-availability.pdf

[2] Z. Feldman, M. Masin, A. N. Tantawi, D. Arroyo, and M. Steinder, "Using approximate dynamic programming to optimize admission control in cloud computing environment," in *Proceedings of the Winter Simulation Conference*, ser. WSC '11. Winter Simulation Conference, 2011, pp. 3158–3169.

[3] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 239–254, Dec. 2002.

[4] A. J. Page and T. J. Naughton, "Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005, pp. 189a–189a.

[5] H. W. D. Chang and W. J. B. Oldham, "Dynamic task allocation models for large distributed computing systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 6, no. 12, pp. 1301–1315, 1995.

[6] G. L. Choudhury, K. K. Leung, and W. Whitt, "An algorithm to compute blocking probabilities in multi-rate multi-class multi-resource loss models," *Advances in Applied Probability*, pp. 1104–1143, 1995.

[7] A. Simonian, J. Roberts, F. Theberge, and R. Mazumdar, "Asymptotic estimates for blocking probabilities in a large multi-rate loss network," *Advances in Applied Probability*, pp. 806–829, 1997.

[8] E. Coffman, K. So, M. Hofri, and A. Yao, "A stochastic model of bin-packing," *Information and Control*, vol. 44, no. 2, pp. 105–115, 1980.

[9] W. T. Rhee, "Optimal bin packing with items of random sizes," *Mathematics of Operations Research*, vol. 13, no. 1, pp. 140–151, 1988.

[10] A. Goel and P. Indyk, "Stochastic load balancing and related problems," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE, 1999, pp. 579–586.

[11] D. Dowson and A. Wragg, "Maximum-entropy distributions having prescribed first and second moments (corresp.)," *Information Theory, IEEE Transactions on*, vol. 19, no. 5, pp. 689–693, 1973.

[12] L. Kleinrock, *Queueing Systems*. New York: John Wiley & Sons, 1975, vol. 1: Theory.

[13] E. Gelenbe and G. Pujolle, *Introduction to Queueing Networks*. John Wiley & Sons, 1987.