RC25469 (WAT1405-043) May 21, 2014
Computer Science

IBM Research Report

Cloud Overbooking through Stochastic Admission Controller

Merve Unuvar, Yurdaer N. Doganata, Asser N. Tantawi, Malgorzata Steinder
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

Research Division
Almaden – Austin – Beijing – Cambridge – Dublin - Haifa – India – Melbourne - T.J. Watson – Tokyo - Zurich

Cloud overbooking through stochastic admission controller

Merve Unuvar, Yurdaer N. Doganata, Asser N. Tantawi and Malgorzata Steinder

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{munuvar,yurdaer,tantawi,steinder@us.ibm.com}

***Abstract*—Cloud providers apply overbooking to increase utilization of data center resources. Overbooking is associated with the risk of over-utilizing cloud resources. In this paper, we study an admission control technique that permits Cloud overbooking with bounded probability of resource over-utilization. The objective is to achieve a specified quality-of-service related to the probability of resource over-utilization in an uncertain loading condition, while maintaining high resource utilizations. Our method relies on an approximation of total resource demand generated by multiple tasks as a Beta Distribution. We perform a qualitative study to investigate the efficiency of using our method on Google Compute Cluster. We disclose empirical observations on how well the resource utilization can be estimated as Beta Distribution and we evaluate the effectiveness of the stochastic admission controller using the same data set.**

***Keywords*—admission control, cloud management, dynamic 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].

The most commonly used admission policy admits a request (such as a virtual machine or a job) into the Cloud if its full resource demand can be reserved on some physical resource in the Cloud. As resource requests often greatly exceed their actual time-varying demand, such policy leads to Cloud resources being under-utilized. Cloud providers want to take advantage of that unused capacity by resource overbooking. Overbooking allows the sum of admitted resource requests to exceed the capacity of physical resources in the Cloud. The choice of placing a request on a particular physical node is a decision of the *Placement Controller*, which is not the subject of this paper, but has been studied in many publications before. Here we are concerned with the decision to admit a request once its suggested physical host is selected. A request needs to be admitted to at least one physical node, in order to be admitted into the Cloud.

Overbooking carries with it the risk that the actual demand will exceed the usable capacity of physical resources leading to request interruption or poor performance. Admission control policies that permit overbooking must therefore be selected such that the risk of overload is minimized.

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 request based on its resource demand characteristics, such as maximum resource demand or average resource demand. Parameter-based schemas are easy to implement but their effectiveness is reduced by the inability to deal with workload variability; incorrect parameter settings can lead to under-utilization or over-utilization depending on actual workload resource demands.

A measurement-based admission control schema, on the other hand, takes advantage of measurements made in the real system to guide admission decisions in addition to request parameters. Such technique can better adapt to workload intensity and variability. In this paper, we introduce a measurement-based admission control policy for the Cloud. Our technique relies on easily available resource utilization measurements. Each resource in a physical machine 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. The admission control policy ensures a stochastic bound on the probability that a resource is over-utilized.

Each resource is described by the stochastic properties of its utilization. The probability density function (pdf) of the utilization of a resource is the convolution of all 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. In mathematical terms, resource k is stable if its utilization, U_k, satisfies the following constraint,

$$P(U_k > U_k^o) \leq \epsilon^o \tag{1}$$

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

In this study, we approximate the pdf of the aggregated resource utilization using the first and second moments as a Beta Distribution. 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 evaluate our technique using Google Compute Cluster usage data. The Google Cluster data discloses resource usage by millions of tasks running on a set of machines hosted in racks and connected by a high bandwidth network. Our aim is to verify the Beta Distribution assumption by aggregating usage of different number of tasks and check if we can fit a Beta Distribution to the actual resource utilization distribution. We also compare the over-utilization probability that we estimate using the Beta assumption with the actual resource over-utilization in Google Cluster data.

The paper is organized as follows. We introduce a mathematical representation for the arriving resource requests and the associated distribution for the resource demand in section II. The details of stochastic admission controller is presented in in section III. Our numerical results for Google Compute Cluster are presented in section IV. We review the related work in section V and summarize the conclusion and future work in section VI.

II. Formulation

A. Homogenous System

Consider p homogenous *PMs* subjected to a stream of homogenous requests with a Poisson arrival process with rate λ and a generally distributed lifetime with mean τ. A *PM* has K different resource types, each having capacity C_k, $k = 1, 2, \cdots, K$. 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, i.e. $Z_k^n = nD_k$. Thus, 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.

B. Heterogeneous System

Consider a system that is subjected to I request types. As before, we have p homogenous *PMs*. A type i request has a demand of D_{ik} for 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 $d_{ik} \in [D_{ik}^{min}, D_{ik}^{max}]$ and $i = 1, 2, \cdots, I$. We assume that $D_{ik}^{min} = 0$ and $D_{ik}^{max} > 0$. Let λ_i represent 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 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, $Z_k^{\mathbf{n}} = \sum_{i=1}^{I} n_i \; D_{ik}$. 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. Stochastic Admission Controller

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. If the admission criterion uses fixed 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 variance of resource utilization, we call it measurement-based.

First, for the sake of simplicity, we detail the description of stochastic admission controller for homogenous system only. Then, we show later how these can be extended to a heterogenous system. A request is admitted if the admission policy allows it given the current state of the system, otherwise it is rejected. The resulting request rejection probability δ, and the mean utilization of k^{th} resource $\overline{U_k}$ are related as

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

Admission based on a probabilistic bound over-utilization can be denoted by $\mathbb{P}(U_k^*, \epsilon, \mu_k, \sigma_k)$, where U_k^* is the utilization threshold, 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, and μ_k and σ_k are the mean and standard deviation of the utilization for resource k, respectively.

In this probabilistic admission policy, the dynamic nature of resource demand is represented by its mean, μ_{D_k}, and standard deviation, σ_{D_k}. The utilization of resource k, U_k, is a random variable between [0,1]. It is characterized by its mean and standard deviation, μ_k and σ_k, respectively.

Admission criterion for, $\mathbb{P}(U_k^*, \epsilon_k, \mu_k, \sigma_k)$, utilizes U_k^*, ϵ_k, μ_k and σ_k to make an admission decision. Here μ_k and σ_k are the estimated mean and standard deviation of the measured utilization of resource k at the request arrival time. The admission criterion is given by

$$F_{Z_k^n}(U_k^*) \geq (1 - \epsilon_k), \tag{6}$$

otherwise the request is rejected.

The key to implementing the stochastic admission controller is the knowledge of $F_{Z_k^n}(U_k^*)$. We approximate the probability distribution function (pdf) of U_k as a Beta Distribution. The 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

characterize the utilization U_k with two parameters, α_k and β_k, associated with the first and second moments of U_k.

The values of α_k and β_k are computed from the estimated mean and variance values of the utilization of resource k in the *PM* as:

$$\alpha_k = \bar{R}_k \left(\frac{\bar{R}_k(1-\bar{R}_k)}{\bar{S}_k^2} - 1 \right), \tag{7}$$

$$\beta_k = (1-\bar{R}_k) \left(\frac{\bar{R}_k(1-\bar{R}_k)}{\bar{S}_k^2} - 1 \right), \tag{8}$$

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:

$$F_{Z_k^n}(U_k, \alpha_k, \beta_k) = B(U_k, \alpha_k, \beta_k)/B(\alpha_k, \beta_k), \tag{9}$$

where B is the *Beta* function. $\mathbb{P}$ does not need to be adjusted as the workload characteristics change, since it is dynamically adjusted with the measured statistical properties of resource utilization. The predefined parameters U_k^* and ϵ_k are set by the Cloud manager depending on the specifications of the physical machine.

In the heterogenous load case, admission controller still admits a request into a *PM* based on $\mathbf{n} = (n_1, n_2, \cdots, n_I)$, the number of requests of each type in the *PM* at the time of admission policy $\mathbb{P}(\phi)$ is applied. Hence, the convolution in Equation 6 is over the vector $\mathbf{n}$ rather than the scalar value n.

IV. Numerical Results

We use Google Compute Cluster data, which is available online[1], to support our assumption on representing resource utilization by Beta Distribution, and to estimate the mean and the standard deviation of real workloads to experiment the performance of stochastic admission controller. In this section, we first give explanation on Google Compute Cluster data, then detail our estimation method.

A. Google Compute Cluster Data

In this section, we describe the large scale Google production workload trace (version 2)[2] that is recently made publicly available by Google. Google Cluster is a set of machines hosted in racks and connected by a high bandwidth network. A cell is a group of machines, which share a single cluster management system, that places requests on to physical machines [2]. Google published 2 traces of their computing cells, which both contain usage trace, system resource and machine characteristics. The trace that we study in this section (version 2) represents 29 days worth of cell information on a single cluster collected in 500 time points.

Table I shows the content of available data that can be downloaded from Google's website. Some folders (job_events, task_constraints, task_events, task_usage) contain 500 csv files in time series form. Machine_attributes and machine_events folders contain single csv file to describe the properties of the machine attributes and events. The downloaded data also contains "schema.csv" file which describes the column names

[1]http://code.google.com/p/googleclusterdata/

[2]https://code.google.com/p/googleclusterdata/wiki/TraceVersion2

TABLE I: Contents of Google Cluster Trace

Folder name	Format
job_events	time series
machine_attributes	properties- single file
machine_events	properties- single file
task_constraints	time series
task_events	time series
task_usage	time series

in each file for all sub-folders. Before we start describing how we benefit from the Google Cluster data for our work, let us first highlight the statistical properties of the data, how it is stored and what is publically made available by Google.

Workloads (*requests*) arrive to a cell in the form of jobs where job consists of one or more tasks. Each task has resource requirements that are used by the Google scheduler to place the tasks onto physical machines. Every job and every machine in the dataset are represented by a unique 64-bit identifier (job ID and machine ID) whereas tasks are indexed as 0-based indices for the job they are associated with [3]. There are total of 2,012,242 observations recorded in the trace where there are 672,074 unique Job IDs [3]. In job_events dataset, there are 933 unique users and 183,955 unique job names and logical job names [3].

Each machine in the machine_events dataset has a unique machine ID. There are total 12,583 unique machine IDs [3]. Machine_events stores two capacity values; CPU and memory which are normalized physical capacity of each machine in the cell (between [0,1]). There are three different level of CPU core capacities which are [0.25, 0.50, 1] and there are ten different memory capacities which are [0.0308, 0.0616, 0.1241, 0.2493, 0.2498, 0.4995, 0.5000, 0.7490, 0.9678, 1.0000].

The task_usage dataset consists of 500 csv files. Each record in the tables has a timestamps attribute, which is in microseconds since 600 seconds before the beginning of the trace period, and is recorded as a 64-bit integer [2]. Task_usage table records the mean and the maximum resource usage for CPU, memory, and disk (labels are CPU usage, memory usage, disk I/O time) for the period of time that the monitoring is done. Each measured records is extracted from typically 300 seconds time interval. Most of the resource usage measurements and requests have been scaled by Google by an unknown linear transformation function due to confidentiality reasons. CPU is measured in CPU core- seconds/second, memory in bytes, and disk time fraction in I/O seconds/second.

B. Statistics on Resource Usage Data

We want evaluate if a real-life workload characteristics can be correctly represented as a Beta Distribution. For this purpose, we analyze a representative sample of jobs/tasks from Google data set. Google Cluster contains multiple job types [4], which differ in resource consumption and duration. Our objective is not characterizing these types - we aim for a representative mix, and therefore our samples are randomly selected. Here we show how such samples are extracted from the Google data set.

Figure 1 plots the aggregated CPU usage in core-hours for average of 17 jobs that involve 18 tasks at a given time over 500 snapshots in 29 days among total of 1,771 unique

jobs consisting of 3,061 tasks that are running on a particular machine with an ID "351618647". Here, the available CPU usage data is converted into to core-hours metric from core-seconds per second. The core-hours is the product of average CPU usage and its duration (duration is 300 seconds = 5 minutes) and divided by 12 since records are collected for 5 minute intervals. The GB-hours for memory is calculated in a similar way. As it is shown in Figure 1, the variation of CPU usage is high compare to memory usage over the same period as shown in Figure 2. The coefficient of variation for CPU usage is found to be 15%. Investigating the reason for the fluctuation (steadyness) in the CPU (memory) usage is not the scope of this paper. Our main focus is to investigate how effectively we can approximate the distribution of resource utilization as a Beta Distribution by using Google usage data. Then, using this approximation to show that we can reduce the probability of over-utilization by using the stochastic admission controller developed by Unuvar et al. [5].

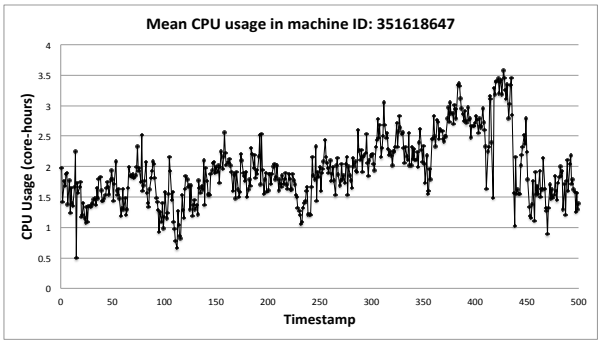


Fig. 1: CPU usage for average of 18 tasks, mean = 1.95, variance= 0.3

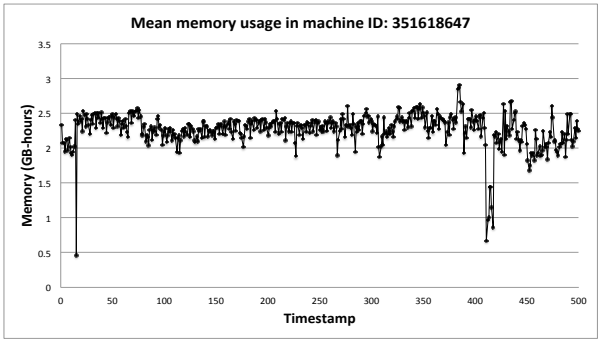


Fig. 2: Memory usage for average 18 tasks, mean = 2.27, variance = 0.05

As shown by Figure 1 and 2, the variation of CPU usage is more compared to that of memory usage. In the rest of the paper, we study the CPU usage utilization since; 1-Due to its dynamic nature, CPU usage will need a better stochastic admission controller, 2-CPU is the bottleneck resource rather than memory most of the time.

C. Fitting Sample Usage Data to Beta Distribution

In this section, we explain the method of calculating the CPU utilization from the sample usage data and using this utilization data to fit a Beta Distribution. In order to calculate the utilization of a resource, the machine resource capacities need to be known exactly. Since Google did not publish the exact machine capacities but published the normalized values, we cannot use that information directly to calculate the resource utilization. Rather, we take the maximum resource usage throughout the 29 days of time span and set that level of utilization to be 90% for the CPU utilization. Google reports that not all of the resource capacity is available to the tasks therefore 10% of the machine is assumed to be reserved for the cluster scheduler and operating system [2]. For example, for the same sample (average of 17 jobs involving 18 tasks running on machine ID= "351618647"), the maximum CPU usage over 29 days is 3.97 core-hours. We set the usage of 3.97 core-hours to be 90% utilization and normalize the rest of the usage data by dividing to $3.97/0.9$. Since actual resource consumption is done by the tasks, we will be using only the number of tasks rather than number of jobs through the rest of the paper. Figure 3 plots the distribution of the CPU utilization for 29 days when calculated by using the 18 tasks on average. Table II shows the general statistics for the usage of 18 tasks, transformed to utilization data, in each row. There are total of 500 observations since we calculate the resource utilization at every data measurement point.

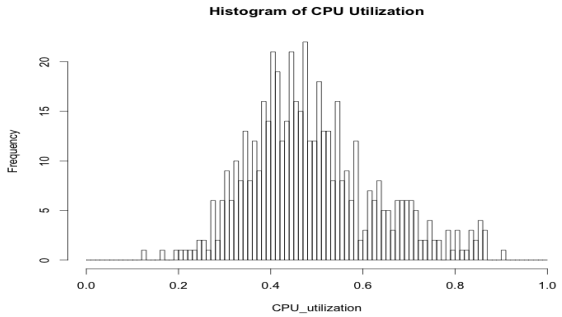


Fig. 3: CPU utilization for 29 days by 18 tasks on average, mean = 0.49, variance = 0.02

TABLE II: Summary statistics for CPU utilization for average of 18 tasks

Variable	Observations	Min	Max	Mean	Std. dev.
CPU utilization	500	0.124	0.901	0.49	0.14

Next, we calculate the shape parameters for Beta Distribution from the mean and the standard deviation of the sample as described in section III. Table II shows the mean CPU utilization and the standard deviation for the 500 sample as 0.49 and 0.14 respectively. By substituting the calculated mean and the standard deviation to Equation 7 and 8, we estimate Beta Distribution shape parameters as follows: $\alpha = 5.76, \beta = 5.93$. Figure 4 depicts the Q-Q plot showing how well this estimation is, based on our sample. Q-Q plot is a probability graph that graphically compares two probability distributions by plotting their quantiles against each other, [6]. If the two distributions being compared are similar, the plotted points in the Q-Q plot will approximately lie on $y = x$ line. As Figure 4 shows, the

estimation of the parameters are fitting to $y = x$ line indicating that the estimated shape parameters are well chosen.

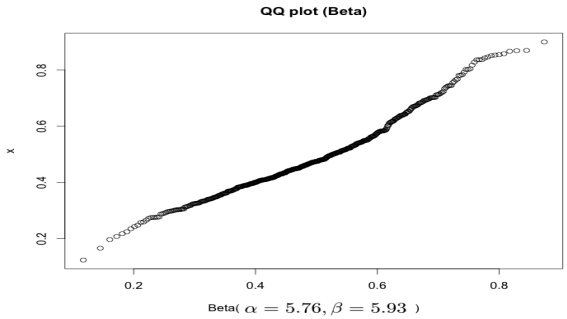


Fig. 4: CPU Utilization for average of 18 tasks fit to Beta Distribution parameters

Further, Table III shows additional parameters of the input data and the estimated Beta Distribution function related to; the first (mean), second (standard deviation), third (skewness) and fourth (kurtosis) moments. Skewness is a measure of symmetry, or more precisely measure of the lack of symmetry where Kurtosis is a measure of whether the data is peaked or flat relative to a Normal Distribution, [7]. Kurtosis coefficient measures heavy tails, sharp peaks or flat tops. As it is supported by Q-Q plot, the mean and the standard deviation are very well estimated however the skewness and the kurtosis coefficients are not well estimated for this sample data. The result of not being able to fit skewness and kurtosis coefficients well is seen in Figure 5 where we plot the actual sample distribution and the estimated Beta Distribution in the form of histogram. As Figure 5 supports, the mean and the variance of Beta Distribution and the actual sample are quite similar however, the skewness and kurtosis which are extensions of third and fourth moments, are not fitted perfectly to the CPU utilization sample.

TABLE III: Statistics estimated on the input data and computed using the estimated parameters of the Beta Distribution

Statistics	Data	Parameters
Mean	0.490	0.493
Variance	0.019	0.020
Skewness (Pearson)	0.601	0.015
Kurtosis (Pearson)	0.160	-0.408

Figure 6 is another way to visualize the distribution of actual CPU utilization values and the estimated Beta Distribution function. Even though the Beta shape parameters fit perfectly, the skewness and the peak of actual data and the estimated Beta Distribution do not quite match. We further perform Kolmogrov-Smirnov (K-S) test to understand if this graphical conclusion is valid at 0.05 significance level. We use Kolmogrov-Smirnov test to compare the fit of actual CPU utilization sample to estimated Beta Distribution sample. The K-S test is one of the most useful and general nonparametric methods for comparing the empirical distribution functions, [8]. K-S test measures a distance between these empirical distributions by drawing samples from the same null distribution of the desired statistics, defined in null hypothesis. We state

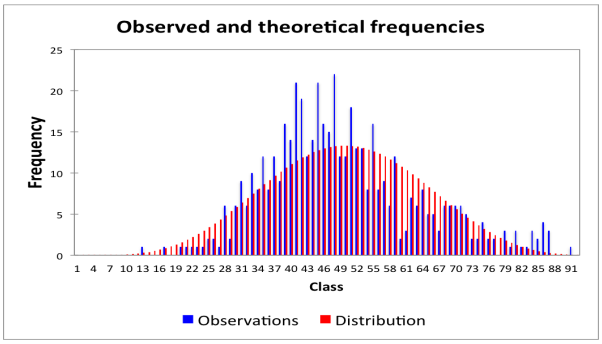


Fig. 5: Histogram of actual CPU Utilization for average of 18 tasks and fitted Beta Distribution

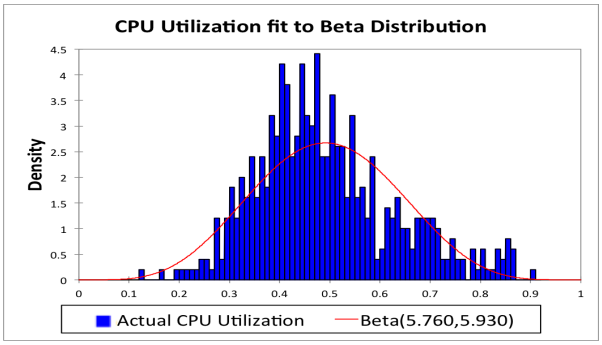


Fig. 6: CPU Utilization fit to Beta Distribution for average of 18 tasks

our null and alternative hypothesis as:

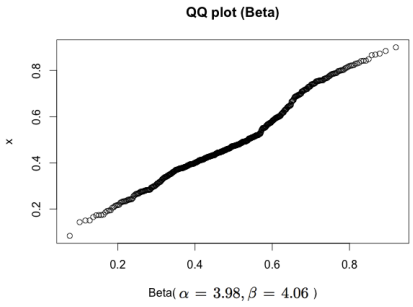
H_0 : The sample follows a Beta Distribution

H_a : The sample does not follow a Beta Distribution.

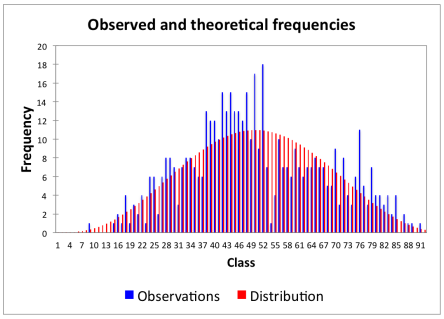
K-S test result is as follows: D value is 0.084, $p - value$ is 0.002 and the significance level, α is 0.05. D value is the maximum difference in cumulative fraction thus small D value is expected while comparing similar distributions. The result of K-S test is usually interpreted by the $p - value$ and the significance level, α. As the computed $p - value$ is lower than the significance level $\alpha = 0.05$, one should reject the null hypothesis H_0, and accept the alternative hypothesis H_a for this particular test. The risk to reject the null hypothesis H_0 while it is true is lower than 0.05%. This result indicates that Beta Distribution is not a good fit for the average of 18 tasks worth of usage data. This may be caused by the lack of representative usage sample. In other words, the convolution of the usage distribution of 18 tasks did not fit to a Beta Distribution. In order to minimize the uncertainty caused by the smaller subsets, we increase our task sample size to 49, 88 and 154. We show that as we sample the usage data from more tasks uniformly, the resource utilization fits to Beta Distribution better. The results of larger samples are described in the next section.

D. Effect of Sample Size on Beta Fitness Test

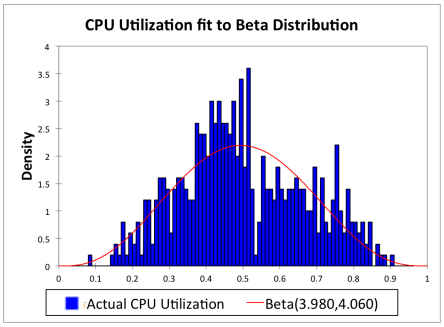
Even though the utilization distribution for 18 task usage on average graphically looked like an "almost" good fit to Beta



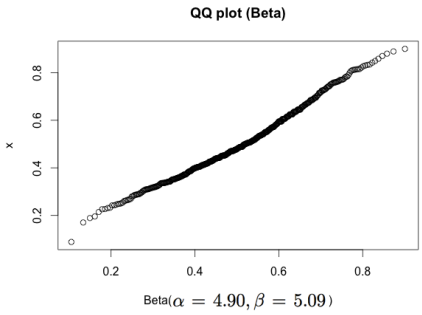
(a) CPU Utilization fit to Beta Distribution parameters for average of 49 tasks



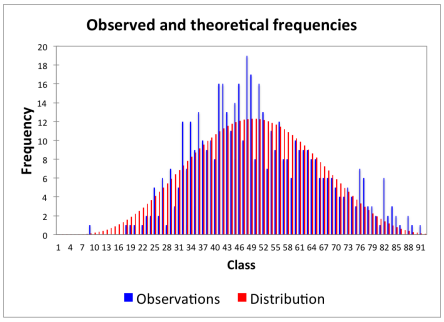
(b) Histogram of actual CPU Utilization and fitted Beta Distribution for average of 49 tasks



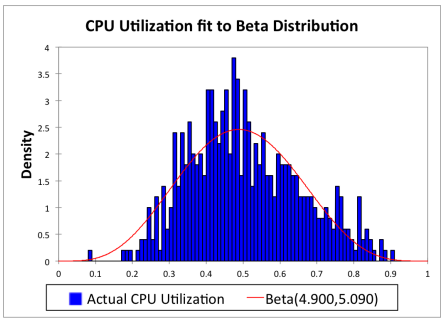
(c) CPU Utilization fit to Beta Distribution for average of 49 tasks



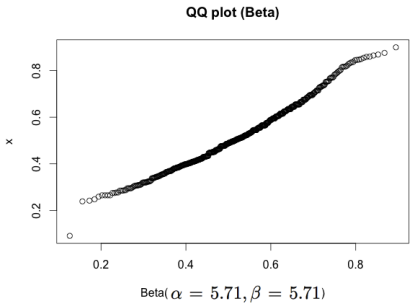
(d) CPU Utilization fit to Beta Distribution parameters for average of 88 tasks



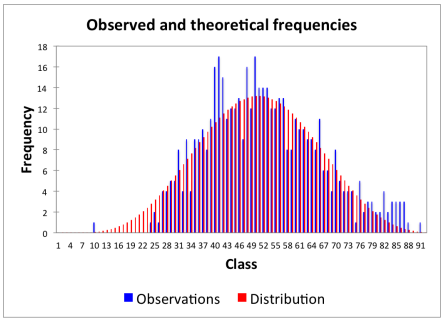
(e) Histogram of actual CPU Utilization and fitted Beta Distribution for average of 88 tasks



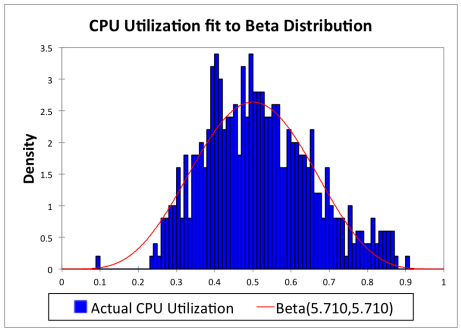
(f) CPU Utilization fit to Beta Distribution for average of 88 tasks



(g) CPU Utilization fit to Beta Distribution parameters for average of 154 tasks



(h) Histogram of actual CPU Utilization and fitted Beta Distribution for average of 154 tasks



(i) CPU Utilization fit to Beta Distribution for average of 154 tasks

Fig. 7: Q-Q plot, CPU utilization & Beta Distribution fit and observed & theoretical utilization for 29 days for usage of 49, 88 and 154 tasks

Distribution, the fit, however, did not pass the K-S test at 0.05 significance level. We believe this is caused by the random selection of 18 tasks, which was not representative enough for the general workload to convolute as Beta Distribution. In this section, we perform the Beta fit method described above on 3 different samples separately. We randomly, on average, first select 49 then 88 and finally 154 tasks over 29 days to see if higher sample sizes would fit utilization to Beta Distribution better. We aggregate the usage of the tasks at each day over 29 days for every sample. We again set the maximum usage level over 29 days to be 90% resource utilization to leave 10% of the capacity for cluster management & operating systems.

Table IV shows the general statistics for the usage of, 49, 88 and 154 tasks, transformed to utilization data, in each row. As we sample usage from more tasks, the mean CPU utilization approaches to 50% and the standard deviation remains around 0.15. By using the calculated mean and standard

TABLE IV: Summary statistics for CPU utilization levels for usage of 49, 88 and 154 tasks

Sample	Variable	Obs.	Min	Max	Mean	Std. dev.
49 tasks	CPU Utilization	500	0.085	0.900	0.500	0.169
88 tasks	CPU Utilization	500	0.088	0.901	0.504	0.151
154 tasks	CPU Utilization	500	0.091	0.900	0.516	0.142

deviation values in Table IV, we estimate the shape parameters for three usage samples. The estimation of shape parameters for samples of 49, 88 and 154 tasks is more accurate than 18 task sample. Table V lists the estimated and actual values for mean, variance, skewness and kurtosis for 3 samples.

TABLE V: Statistics estimated on the input data and computed using the estimated parameters of the Beta Distribution

Experiment	Statistics	Data	Parameters
49 tasks	Mean	0.500	0.495
	Variance	0.029	0.028
	Skewness (Pearson)	0.202	0.012
	Kurtosis (Pearson)	-0.674	-0.543
88 tasks	Mean	0.504	0.490
	Variance	0.023	0.023
	Skewness (Pearson)	0.340	0.021
	Kurtosis (Pearson)	-0.415	-0.461
154 tasks	Mean	0.516	0.500
	Variance	0.020	0.020
	Skewness (Pearson)	0.374	0.000
	Kurtosis (Pearson)	-0.274	-0.416

Figure 7 shows the Q-Q plot, distribution of the actual CPU utilization and the estimated Beta Distribution for all samples. As Figure 7a, 7d and 7g show, the Q-Q plot is almost on $y = x$ line, indicating that the shape parameter estimations are very accurate for all 3 samples. Further, Figure 7b, 7e and 7h depicts the actual data distribution and the Beta Distribution in the form of histogram. In all 3 samples, estimated Beta Distribution, in graphical comparison, seem to be a good fit. In Figure 7c, 7f and 7i, we see the Beta function over the actual data distribution which visually support Beta Distribution fit argument again.

Even though the Figure 7 supports the Beta Distribution fit graphically, we still perform the K-S test to see how well the Beta Distribution fit to our samples. Table VI shows that with sample of 49 tasks, the $p - value$ is 0.098 meaning that as the computed $p - value$ is greater than the significance level $\alpha = 0.05$, one cannot reject the null hypothesis H_0. The risk to reject the null hypothesis H_0 while it is true is 9.76%. For the sample of 88 tasks, the computed $p - value$ is still higher than the significance level therefore one cannot reject the null hypothesis, H_0 again. The risk to reject the null hypothesis H_0 while it is true is 19.61%. For the sample of 154 tasks, the value of $p - value$ is even higher indicating that the better uniform sampling represent the workload characteristics better thus the Beta Distribution fit is more accurate. In the 154 tasks case, as the computed $p - value$ is greater than the significance level $\alpha = 0.05$, one cannot reject the null hypothesis H_0. The risk to reject the null hypothesis H_0 while it is true is 20.93%.

TABLE VI: Kolmogorov- Smirnov Test

Test Result	49 tasks	88 tasks	154 tasks
D	0.055	0.051	0.048
$p - value$	0.098	0.150	0.196
α	0.05	0.05	0.05

The above analysis showed that the quality with which the Beta Distribution fits the distribution of U_k depends on the number of request random variables included in the convolution. In practical terms, this means that in a system where a physical compute node can host a large number of tasks simultaneously, the Beta Distribution is a good representation of the utilization generated by the convolution of these tasks.

E. Estimation Error for Stochastic Admission Controller

Given the Beta Distribution of resource usage, it is useful to consider the potential benefit a stochastic admission controller could bring when applied to a particular system and how the inaccuracy resulting from using Beta as an approximation can affect this result. To this end, we compare the actual over-utilization observed in Google data set with the estimation thereof computed by the stochastic admission controller under Beta assumption.

The stochastic admission controller has two parameters: the over-utilization threshold and the probabilistic bound on the over-utilization as defined in section III. Figure 8 shows the Beta Distribution that fitted over the actual usage for 154 tasks data. For example, if the over-utilization threshold is set to 0.8 by the user, the probability of over-utilization -regardless of the admission controller- is estimated as $P(U_{CPU} \geq 0.8) = 0.024$ -the green portion of the Beta curve in Figure 8. The stochastic admission controller bounds the over-utilization probability to a user selected parameter, the probabilistic bound, by rejecting the arrivals that would potentially increase the likelihood of over-utilization beyond the selected parameter. Thus, the better the Beta Distribution fit, the closer the approximation of over-utilization upper bound is performed by the stochastic admission controller.

As an example, with estimated Beta Distribution for the usage data, stochastic admission controller ensures that the probability of over-utilization (over-utilization being 0.8) as 2.4%. This can be verified by equation (6). The actual distribution of CPU, however, is higher than the over-utilization threshold 4.4% of the time. Hence, the stochastic admission controller under-estimates the over-utilization by 2.0% for the 154 task sample. Table VII shows that as the beta fit gets better (higher $p - value$ in K-S test), the error in estimating over-utilization probability decreases.

TABLE VII: Error in estimating $P(U_{CPU} \geq 0.8)$

Sample size	$p - value$	Actual prob.	Estimated prob.	Error
18 tasks	0.002	0.066	0.011	0.055
49 tasks	0.096	0.064	0.031	0.033
88 tasks	0.15	0.042	0.016	0.028
154 tasks	0.196	0.044	0.024	0.020

Since Beta Distribution underestimates the over-utilization probability, the stochastic controller will deliver higher ratio of over-utilization samples than its theoretical goal. At the same time, the controller is unlikely to reject requests unnecessarily. The knowledge of the direction in which the controller is likely to err, should be helpful in adjusting its parameters to obtain the desired system behavior.

F. Implementing Stochastic Admission Controller on Google Compute Cluster

In this section, we apply the stochastic admission controller to Google data set and measure its effectiveness.

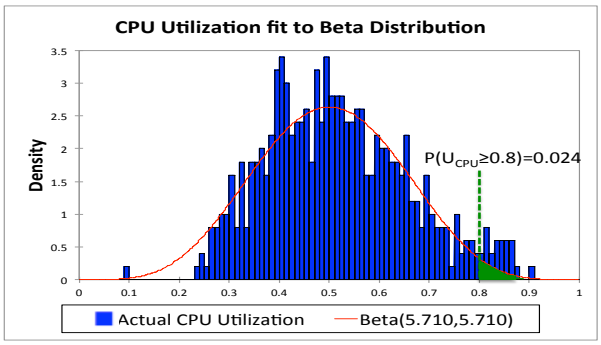


Fig. 8: Estimated and actual probability of over-utilization

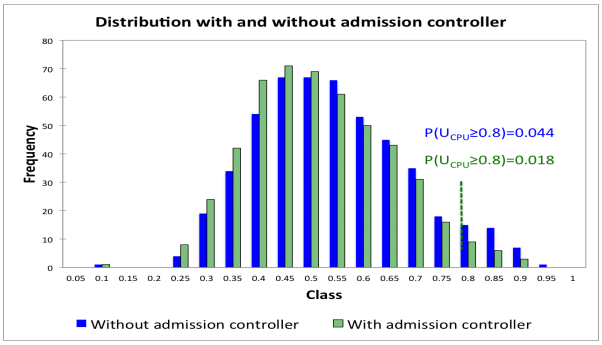


Fig. 9: Distribution of CPU utilization with and without the stochastic admission controller

We apply the controller to 30,000 task arrivals randomly selected from Google data set over 29 day time span. There are 154 tasks on average running in the system (same distribution of tasks in Figure 8). As the first step of the analysis we calculate the mean and the standard deviation of resource demands of each task. Then, given the parameters of individual task distributions, we apply the controller criterion to decide the admission or the rejection of each task. Rejected tasks are removed from the distribution and do not contribute to the total resource usage in the resultant distribution. This way we end up with two data sets: the original data set of 30,000 tasks from Google data and the resultant distribution with a subset of these tasks accepted by our controller.

Stochastic admission controller, changes the shape of the utilization distribution. For example, in Figure 8 if an admission controller policy on CPU, $\mathbb{P}(0.8, 0.01, 0.516, 0.142)$ was applied, then almost half of the actual data points beyond the green dashed line would not have been accepted into the system. This would change the shape of the actual distribution yielding a change in the estimated Beta Distribution. The new curve would have a longer tail then the current one which would decrease the likelihood of over-utilization of the resources significantly.

Figure 9 shows the distribution of actual tasks' contribution to utilization without the admission controller and with the admission controller. After applying the stochastic admission controller, $\mathbb{P}(0.8, 0.01, \mu, \sigma)$, where μ and σ are calculated at every new arrival, we see that for the green bars, the probability of over-utilizing the CPU is 0.018. For the blue distribution, which is the utilization level without the admission controller, the probability of over-utilizing the CPU is 0.044. With the admission controller, 1.8% of time, this particular sample of tasks may over-utilize the machine. Even though the ϵ for the upper bound on probability of over-utilization is defined in the policy as 0.01, we were able to bound the probability of over-utilization by 0.018 due to estimation error shown in Table VII.

Table VIII shows the effect of the stochastic controller for various values of over-utilization probability bounds, ϵ. It demonstrates that increasing ϵ leads to lower request rejection rate and higher over-utilization probability. It also shows that our technique adjusts to different settings of ϵ although it does not perfectly achieve ϵ, and that it consistently delivers improvement in over-utilization probability relative to the system with no admission controller.

TABLE VIII: Stochastic Admission Controller with different ϵ bounds one probability of over-utilization

ϵ	$P(U_{CPU} > 0.8)$ with admission controller	$P(U_{CPU} > 0.8)$ without admission controller	Rejection %
0.005	0.012	0.044	10.83%
0.01	0.018	0.044	8.27%
0.02	0.038	0.044	4.12%

V. Related Work

The problem of admission control in data centers, and in the Cloud in general, has been addressed from different angles. In [9], a data center that is subjected to a stream of *VM* requests of different types is considered. In practice, the demand varies over time, suggesting the inclusion of the variability in demand when admitting a *VM*. In [10], this variability in CPU usage enabled the overbooking of resources. [11] allowed overbooking to improve the Cloud utilization with an admission controller that monitors the infrastructure capacities.

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 [12]. To overcome this challenge, asymptotic approximations have been obtained for load factors that correspond to light, critical, and overload conditions [13].

Beta Distributions are a type of probability distribution that is commonly used to describe uncertainty about a fraction or a prevalence. In general, the data that comes from surveys, experiments, or computer executions helps us to describe the uncertainty. Quality Control [14], epidemiology, disease control [15], economics [16], risk analysis [17] are some of the areas where Beta Distribution is often used to measure uncertainty. In our paper, we use the Beta Distribution to measure the uncertainty regarding Cloud resource over-utilization.

Google Compute Cluster data properties, how to access and process the data, are documented in [2]. Statistical analysis, such as workload characteristics for the same dataset is studied

by [3] and for a similar Google production data by [4]. In [4], authors used k-means clustering to group the similar workloads together thus help schedular to understand the behavior of the workloads better. We benefit from both [3] and [4] while deriving simple statistics and conclusions about the characteristics of the data.

Back to the Cloud computing environment, 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.

VI. Conclusion and future work

There are certain conveniences in assuming that the aggregated resource usage distribution in a Cloud machine follows a Beta Distribution. Beta is a family of distributions with two parameters where the parameters are associated with the first and the second moments of the resource usage. Thus, it covers a wide range of usage distribution possibilities. In addition, an expression for over-utilization probability can also be expressed conveniently when the resource usage distribution is assumed to be Beta. The stochastic admission control policy that we introduced in section III bounds the specified over-utilization probability to a user selected level. In this study, we evaluate the stochastic admission policy against the real usage data produced in a Google Compute Cluster. The resource usage data that is obtained from Google Cluster suggests that a Beta Distribution is a good fit for the resource utilization distribution, provided that enough number of tasks are aggregated. Thus, our stochastic admission policy can be effectively applied against the usage data that we extracted from Google Cluster.

We observe that the resource demand distribution of small number of tasks may not always satisfy the K-S criterion for Beta Distribution fit. This corresponds to the cases where machines with smaller capacities are used to run the tasks or resource demands are big or not granular enough to fit large number of tasks in one machine. Thus, we conclude that when the number of tasks running on a single machine is large enough, then the resource utilization distribution fits Beta Distribution better. As a result, the stochastic admission controller gives more reliable over-utilization bounds. When the number of tasks is not large enough, on the other hand, the stochastic admission controller under estimates the over-utilization probability.

Our experiment with Google Cluster data shows that the resource distribution consistently fits to Beta Distribution when the number of tasks running on a single machine is above 50 and the stochastic admission controller bounds the over-utilization probability better. Our future work will focus on statistical analysis of the Google Cluster data to determine the conditions around the number of tasks running in a single machine to justify the Beta assumption.

The stochastic admission controller avoids over-utilization with certain confidence. When the stochastic admission controller is implemented based on the Beta assumption, the admission controller is found to be over-confident against over-utilization compared to the actual usage distribution when Beta assumption fails. This can be avoided if we can fit the tails of the distribution better. We are working on modifying our approach to find a Beta Distribution that fits better at the upper tail. This will help improving the confidence for avoiding over-utilization.

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] Google, "https://docs.google.com/file/d/0b5g07t_grdg9njztzzrfbmm."

[3] M. Rasheduzzaman, I. M. Amirul, I. Tasvirul, H. Thamid, and M. R. Rashedur, "Task shape classification and workload characterization of google cluster trace," *IEEE International Advance Computing Conference*, 2014.

[4] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: Insights from google compute clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, March 2010.

[5] M. Unuvar, N. Y. Doganata, and A. N. Tantawi, "Configuring cloud admission policies under dynamic demand," Modeling, Analysis, Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on. IEEE, 2013.

[6] G. Blom, *Statistical estimates and transformed beta variables,*. John Wiley and Sons, 1958.

[7] H. Pham, *Handbook of Engineering Statistics*. Springer, 2006.

[8] N. Smirnov, "Table for estimating the goodness of fit of empirical distributions," *Annals of Mathematical Statistics*, vol. 19, pp. 279–281, 1948.

[9] 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.

[10] 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.

[11] L. Tomas and J. Tordsson, "Improving cloud infrastructure utilization through overbooking," *CAC '13 Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, vol. 5, 2013.

[12] 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.

[13] 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.

[14] C. Cogun and B. Kilinc, "An attempt to model distributions of machined component dimensions in production," *KSME International Journal*, vol. 16, no. 1, pp. 60–74, January 2002.

[15] I. F. Lewis and P. R. Torgerson, "A tutorial in estimating the prevalence of disease in humans and animals in the absence of a gold standard diagnostic," *Emerging Themes in Epidemiology*, vol. 9, no. 9, 2012.

[16] P. R. D. S. Duangkamon, C. and and T. K. K., "Estimating income inequality in china using grouped data and the generalized beta dsitribution," *Review of Income and Wealth Review of Income and Wealth*, vol. 53, no. 1, pp. 127–147, March 2007.

[17] N. Y. Doganata and F. Curbera, "Effect of using automated auditing tools on detecting compliance failures in unmanaged processes," *BPM '09: Proceedings of the 7th International Conference on Business Process Management*, pp. 310–326, 2009.