# Dynamic Resource Allocation of Shared Data Centers Supporting Multiclass Requests

Sai Rajesh Mahabhashyam and Natarajan Gautam
310 Leonhard Building,
The Pennsylvania State University,
University Park, PA 16802.
sairajesh@psu.edu, ngautam@psu.edu

## Abstract

*In this paper, we consider a shared data center (or web server) with requests from multiple companies: each company with two classes of requests: 1) streaming requests and 2) elastic requests. The processing capacity of the web server is shared by the companies' requests of the two classes. An analytical model is developed and matrix geometric method is used to derive the system performance measures. A cost model is developed to obtain a request admission control policy that utilizes the web server's processing capacity efficiently. The objective is to maximize the revenue of the data center, which is a function of the multiclass request characteristics and the quality of service measures. It is well known that the request characteristics like arrival rates vary dynamically over time. As a consequence, autonomic computing is the only practical way for determining on the fly how to partition the web server processing capacity over time among the companies.*

## 1. Introduction

With the increase in popularity, the internet has become an important medium for business, selling and buying goods and services. It is also an interface for applications like e-commerce, online trading etc., which require performance guarantees from the web server. As World Wide Web is becoming more prevalent, the internet data centers that host third party web applications and services, are receiving a lot of attention.

Internet data centers serve multiple service sites from a common hardware base. In the shared web server model, an application can share resources with other applications and so it becomes important to know how to partition resources among competing applications. The concept of data centers is driven by the potential for financial savings. But, providing guarantees is difficult under the unpredictable nature of the server load, and failure to meet the guarantees may result in financial loss and possible loss of customers. So, it is very important to utilize the resources optimally while providing Quality of Service (QoS) needs.

It is well known that the traffic in the internet vary dynamically over multiple time scales and estimation of such workloads itself is a topic of research (Shumway and Stoffer [13]). Also, the traffic may come from all over the world (for e.g. hosting olympic websites), which makes the traffic rates dynamically vary over time. As a result, static allocation of resources is either inefficient or risky: while over provisioning can lead to underutilization of resources, underprovisioning can lead to possible violation of guarantees, loss of customers, etc. A better approach is to dynamically vary the resources among the applications so that under and overprovisioning can be avoided as much as possible. So, the web servers need to be tuned appropriately from time to time to efficiently run the system and the only practical way of implementation is by using autonomic computing.

We consider multiple classes of requests with varying QoS needs. Specifically, there are two main classes, one is streaming class that has bandwidth requirements, and the other is elastic class that utilizes processor capacity unused by the real-time requests. We consider both the classes of requests in the web server scenario mentioned earlier. The crucial point is that, for optimization, both the blocking probability of streaming requests and the delay of elastic requests are minimized. The motivation for this comes from the fact that most web pages have both streaming and elastic requests that needs to be delivered to the users. Therefore, it is critical to give importance to both classes of requests. Optimizing the resources for the two classes under consideration would have a tremendous impact on the resource utilization. For the system to run efficiently, there is a need to control admission appropriately based on the variations in parameters at various times of day. In the next part of this section, we will discuss about some related research work in this area.

Chandra et al [3] developed an analytic model that considers dynamic resource allocation in servers that use generalized processor sharing. It considers web data traffic whose QoS requirement is mean response time. Some researchers (Abdelzaher et al [1]) focussed on designing adaptive systems for web servers based on control theoretic approach and some others (Lu et al [7]) designed adaptive systems that can react to workload changes in the context of storage systems. Another topic of interest for researchers (Chase et al [4]) is to obtain optimization techniques for reducing energy consumption in data centers. But none of the above articles have considered both the classes of traffic in the same model. The reason for this is that the two classes of traffic have different QoS needs and most techniques developed in the multimedia traffic are not applicable in the context of web server. In the networking community, although there are some articles (Ross [12], Roberts et al [11]) that deal with streaming traffic, but they ignore the needs of elastic traffic.

In this paper, we consider an internet data center with multiple companies, each supporting two classes of traffic, and with different arrival rates and holding times. The elastic traffic of all the companies share the remaining processing capacity left over by the streaming traffic. The objective is to minimize the cost obtained by combining the QoS of both the classes. The important contribution is how the admission of streaming requests of different companies is controlled at different times under dynamically varying arrival rates and holding times.

The paper is organized as follows. The problem description with notation is given in section 2. In section 3, an analytical model is developed to obtain the performance measures. The pricing scheme and the optimization model will be introduced in section 4. The analysis and optimization is illustrated using examples in section 5. Finally, section 6 concludes the paper.

## 2. Problem definition

### 2.1. Notation

In the problem under consideration, there are two types of requests, streaming requests like video on demand and internet telephony, and elastic requests of files, where bandwidth reservation is not as important as it is in streaming requests. In this paper, we will call the former as class 1 requests and the latter as class 2 requests.

Consider a shared data center with multiple companies. The data center serves third party web applications through the respective companies. Let the processing speed or capacity of the webserver be $S$. We consider a static admission control policy. For each company $k$, the maximum number of class 1 requests allowed simultaneously is $n_k$ i.e., class 1 requests are rejcted if there are $n_k$ ongoing requests. Each class 1 request from company $k$ is allocated a fixed bandwidth $b_k$, which is reserved when admitted. However, all class 2 are admitted and share the remaining processor capacity (not used by all class 1 requests of all companies) equally. This is accomplished by choosing $c > 0$ such that $S = n_1 b_1 + n_2 b_2 + .... + n_s b_s + c$, where $s$ is the total number of companies. Our aim is to determine $n_k$ optimally. At any given time, requests of class 1 arrive at rate $\lambda_1(k)$ per unit time. Each class 1 lasts for an average of $1/\mu_1(k)$ time (holding time). Class 2 requests arrive at the rate of $\lambda_2(k)$ per unit time and the average class 2 file size is $1/\mu_2(k)$.

At any given time $t$, let $X_k(t)$ and $Y(t)$ be the number of ongoing class 1 requests for company $k$ and class 2 requests in the channel respectively. Class 2 requests share the processor capacity not used by class 1. Therefore, since the unused capacity is $S_{rem} = S - b_1 X_1(t) - .... - b_s X_s(t) = [n_1 - X_1(t)]b_1 + [n_2 - X_2(t)]b_2 + ... + [n_s - X_s(t)]b_s + c$, each class 2 request is allocated a processing speed of $S_{rem}/Y(t)$, whenever $Y(t) > 0$.

The companies pay $\$t_1$ and $\$t_2$ for each admitted class 1 request and each class 2 request respectively. The webserver returns $\$r$ and $\$h$ for each rejection of class 1 request and each unit delay of class 2 request respectively. This can be considered as a penalty that the web server is paying the companies for rejection of class 1 and delay for class 2. The objective is to maximize the total revenue obtained by the webserver from all the companies put together. When the request characteristics (like arrival rate) vary over time, the system will not run efficiently with a static admission control policy. So, $n_k$ can be determined dynamically using autonomic computing. The notation used above and throughout the paper (in section 4) is summarized in Table 1.

| Var | Explanation |
|---|---|
| $\lambda_1(k)$ | Arrival rate of class 1 traffic of company $k$ |
| $\lambda_2(k)$ | Arrival rate of class 2 traffic of company $k$ |
| $\frac{1}{\mu_1(k)}$ | Average holding time of class 1 request of company $k$ |
| $\frac{1}{\mu_2}$ | Average class 2 file size for all companies |
| $s$ | Total number of companies |
| $n_k$ | Maximum no. of class 1 requests of company $k$ allowed simultaneously |
| $b_k$ | Bandwidth allocated to each class 1 request when admitted |
| $c$ | Minimum bandwidth for class 2 requests |
| $S$ | Total bandwidth or capacity of the channel |
| $X_k(t)$ | No. of class 1 requests of company $k$ ongoing at time t |
| $Y(t)$ | No. of class 2 requests ongoing at time t |
| $\theta_{i_1..i_s}$ | Rate of completion of class 2 requests when there are $i_k$ requests $k = 1..s$ |
| $p_{n_k}$ | Steady state probability that there are $n_k$ class 1 ongoing requests |
| $e$ | Column vector of ones |
| $W$ | Average delay of class 2 requests |
| $S^{(i)}$ | State space with $i$ class 2 requests |
| $\pi^{(i)}$ | Steady state probability vector for state space $S^{(i)}$ |
| $R$ | Auxilliary matrix in matrix geometric method |
| $t_1$ | Revenue per unit class 1 request accepted |
| $t_2$ | Revenue per unit class 2 request |
| $r$ | Cost of rejection of a class 1 request |
| $h$ | Holding cost / unit delay of class 2 request |

**Table 1. Notation**

## 2.2. Performance measures

1. *Loss Probability of class 1 requests*: A request is rejected or blocked if and only if there are $n_k$ ongoing requests at the time of arrival of that new request. The probability of a request getting rejected is called *loss probability* or *blocking probability.*

2. *Average delay of class 2 requests*: There is no loss or blocking for class 2 as all the requests are accepted but the time to transfer the file increases when new requests arrive. Hence, the measure chosen to represent the performance of class 2 is *average latency* or *average delay.* The service rate changes dynamically and is dependent on the number of class 1 users at that instant of time. So it is a non-trivial task to calculate average delay of class 2 user experiences.

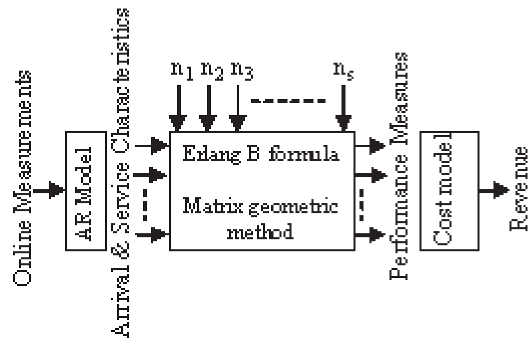


**Figure 1. System representation**

## 2.3. Solution methodology

In the problem description, we introduced the notation and the parameters in the problem like arrival rate, processing times etc. The parameters are estimated using autoregressive techniques and time series analysis. There is lot of research done in that area and we will not focus on that aspect. Given the input parameters, we develop an analytical model to calculate the performance measures described in section 2.2. The blocking probability of class 1 users can be obtained using standard queueing system analysis but calculating the average delay of class 2 is a non-trivial task. In section 3.2, we use the matrix geometric method to calculate the average delay in class 2. Once we have both the performance measures and pricing scheme, we develop a cost function. We come up with an optimization model whose objective is to minimize the penalty cost incurred by the webserver for unit rejection and unit delay, which we will explain in detail in section 4. Thus, the web server resources are allocated among the companies with multiclass requests in an optimal manner. The process is described schematically in figure 1.

When the request characteristics vary over time, it would be inefficient to use the same admission control policy for class 1 over the entire period of analysis. The entire period of analysis can be split into several intervals of time and the arrival and service rates can be obtained by online measurements using a time series (Shumway and Stoffer [13]) model. The solution methodology can be repeated at the beginning of each interval to obtain optimal $n_k$. Thus, the system would run efficiently if the $n_k$ values are changed over time using autonomic computing.

# 3. Analytical Model

In this section, we will develop an analytical model to calculate the performance measures that we defined in section 2.2. Let the arrivals in both class 1 and class 2 be according to Poisson processess with parameters $\lambda_1(k)$ (for company $k$) and $\lambda_2(k)$ respectively. Also, let the holding time of class 1 requests be exponentially distributed with mean $1/\mu_1(k)$ and let the file size of class 2 be exponentially distributed with mean size $1/\mu_2$. The exponential distribution is assumed so as to keep the analysis tractable following Chandra et al [3].

## 3.1. Blocking probability of class 1

To obtain the blocking probability of class 1 requests, we only need to consider transitions of class 1 as they are not affected by class 2 requests. Note that $X_k(t)$ is the number of ongoing class 1 requests at time t. The process $\{X_k(t), t \geq 0\}$ is modeled as a continuous time Markov chain with transitions out of state $X_k(t) = i$ $(0 < i < n)$ as $i \rightarrow i+1:\ \lambda$, $i \rightarrow i-1:\ i\mu_1$.

This is a birth and death process with birth rate $\lambda_1(k)$ and death rate $\mu_1(k)$. From a queuing stand point, this is a standard $M/M/n_k/n_k$ system and the loss probability is given by

$$p_{n_k} = \frac{1/n_k!(\lambda_1(k)/\mu_1(k))^{n_k}}{\sum_{p=0}^{n} 1/p!(\lambda_1(k)/\mu_1(k))^p}$$

where $p_{n_k}$ denotes the steady state probability that there are $n_k$ ongoing requests of class 1 for company $k$. This is well known Erlang B formula (Kulkarni [5]).

## 3.2. Average delay of class 2: Matrix geometric methodology

For class 2 class analysis, it is important to consider class 1 as well due to its inherent dependence. The values of $X_k(t)$ vary from 0 to $n_k$ respectively where as $Y(t)$ can be anywhere between 0 to $\infty$. For $0 < i_k < n_k$ and $j > 0$, the transition states are $j \rightarrow j+1:\ \lambda_2$ and $j \rightarrow j-1:\ \mu_2[\sum_{k=1}^{s}(n_k - i_k)b_k + c]$. Let us call the latter $\theta_{i_1..i_s}$. The available bandwidth for each of $j$ files in class 2 is $\theta_{i_1..i_s}/j$. So the average rate of transfer of a single class 2 file is $\mu_2\theta_{i_1..i_s}/j$. So the transition rate from $j$ to $j-1$ is $\theta_{i_1..i_s}$.

There does not exist any closed form solution for the steady state joint probability. Therefore, the average delay for class 2 requests cannot be obtained in closed form. The stochastic process being a Quasi-birth-death (QBD) process which has a special structure, can be solved using a technique called matrix geometric method (MGM) (Latouche et al [6], Ramaswami [9], Thorne [14]). The infinitesimal generator matrix of QBD is of the form

$$Q_{QBD} = \begin{bmatrix} B_1 & A_0 & 0 & 0 & 0 & \dots \\ A_2 & A_1 & A_0 & 0 & 0 & \dots \\ 0 & A_2 & A_1 & A_0 & 0 & \dots \\ 0 & 0 & A_2 & A_1 & A_0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \text{where}$$

$A_0$, $A_1$, $A_2$and $B_1$are matrices.

One main advantage of that is MGM can overcome the rapid growth of the state space while constructing the infinitesimal matrix of the birth-death process. In matix geometric method, an auxillary matrix called $R$ is used in the calculation of stationary probabilities and other measures of interest like waiting time and mean queue length.

### 3.2.1 Computation of matrix R

The matrix $R$ has a following quadratic relation $A_0 + RA_1 + R^2A_2 = 0$. For simplicity, we will show the waiting time equations for the case with one company. It can be extended to the case with multiple companies.

Neuts [6] defines infinite-state Markov chains with a repetitive structure with state space partitioned into boundary states $S^{(0)} = \{s_0^0, ..., s_0^n\}$ and a set of states $S^{(i)} = \{s_i^0, ..., s_i^n\}$, that correspond to the repetitive portion of the chain. Let $\pi^{(i)}$be the steady state probability vector of states $S^{(i)}$. Then $\pi^{(i)} = \pi^{(1)}R^{i-1}\forall i \geq 1$.

Solving $\pi Q_{QBD} = 0$ will give both $\pi^{(0)}$and $\pi^{(1)}$.The following set of equations are obtained

(Riska and Smirni [10]).

$$\begin{aligned}\pi^{(0)}B_0+\pi^{(1)}A_2 &= 0\\ \pi^{(0)}A_0+\pi^{(1)}(A_1+RA_2) &= 0\\ \pi^{(0)}e+\pi^{(1)}(I-R)^{-1}e &= 1.\end{aligned}$$

Once $\pi^{(0)}$and $\pi^{(1)}$are obtained, the expected holding time of a job in the system can be calculated as follows:

$$W=\lambda_2(\pi^{(1)}(I-R)^{-1}e+\pi^{(1)}R(I-R)^{-2}e).$$

For the problem with one company, $A_0$, $A_1$, $A_2$ and $B_1$are as follows.

$$A_0=\begin{bmatrix}\lambda_2 & 0 & 0 & \dots & 0\\ 0 & \lambda_2 & 0 & \cdots & 0\\ 0 & 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & \lambda_2\end{bmatrix}$$

$$A_1=\begin{bmatrix}-\lambda_1-\theta_0 & \lambda_1 & 0 & \dots & 0\\ \mu_1 & s(1) & \lambda_1 & \cdots & 0\\ 0 & 2\mu_1 & s(2) & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & n\mu_1 & s(n)\end{bmatrix}$$ where

$s(i)=-\lambda_1-\theta_i-i\mu_1$

$$A_0=\begin{bmatrix}\theta_2 & 0 & 0 & \dots & 0\\ 0 & \theta_2 & 0 & \cdots & 0\\ 0 & 0 & \theta_2 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & \theta_2\end{bmatrix}$$

$$B_1=\begin{bmatrix}-\lambda_1 & \lambda_1 & 0 & \dots & 0\\ \mu_1 & u(1) & \lambda_1 & \cdots & 0\\ 0 & 2\mu_1 & u(2) & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & n\mu_1 & u(n)\end{bmatrix}$$ where

$u(i)=-\lambda_1-i\mu_1$. Substituting the above matrices in the set of equations, we get $\pi^{(0)}$ and $\pi^{(1)}$. Thus we obtain the average delay for class 2.

## 4. Optimization

In section 3, analysis for obtaining the performance measures, viz., blocking probability of class 1 and average delay of class 2 have been discussed. In this section, we discuss the pricing scheme and a model to maximize the revenue and allocate resources in an optimal manner. We will discuss the cost function and a search algorithm to obtain values of $n_k$ for each company $k$.

### 4.1. Cost function: Choosing optimal $n_k$

We first analyze how the rejection probability and average delay of class 2 vary versus resource allocation. For clarity and to avoid messy notation, we consider a web server with only one company. It is reasonable to consider $S$ and $b_1$ to be fixed by infrastructure and applications respectively. However, $n_1$ could be chosen such that the channel is utilized optimally. If $n_1$ is increased, there is a decrease in $c=S-n_1b_1$ and thus the probability that class 1 user is rejected decreases while average class 2 file transfer time increases.

With increase in $n_1$, the loss probability of class 1 requests reduces but at the same time the average delay of class 2 requests increases. So, there is always a trade off between the conflicting objectives. Consider the pricing scheme in section 2.1 for a data center with $s$ companies. The data center advertises the following parameters. A fixed cost charged per unit bandwidth for each accepted class 1 customer (say \$$t_1$). Another fixed cost charged per class 2 customer (say \$$t_2$). Let \$$r$ be the cost of rejection of a class 1 request and let \$$h$ be the holding cost per unit time of delay of class 2 request. That is if a class 1 request is rejected, the data center gives the company back \$$r$. Similarly, the data center gives \$$h$ back per unit delay of class 2 customer. Note that $W$ is the average delay for class 2 of all companies together. Then the total revenue for the data center is

$$\sum_k t_1\lambda_1(k)(1-p_{n_k})b_k+\sum_k t_2\lambda_2+\sum_k K_k-\sum_k r\lambda_1(k)p_{n_k}-h\lambda_2 W.$$

The first term is a revenue for admitting requests of class 1. The second term is the revenue for requests in class 2. The constant $K_k$ (third term) would comprise of other revenues that are functions of file size of class 2 for company $k$, holding time of class 1 etc. The webserver gives back the amount in fourth term for blocking class 1 requests. The fifth term is the amount that the webserver gives back for the delay of class 2 request. We do not have to consider terms that would not be affected by the allocation of resources and thus play no role in the optimization.We can safely ignore the unaffected revenues (terms 1 (part), 2 and 3) and minimize the cost (terms 1(part), 4 and 5) that is dependent on resource allocation. The total cost incurred per unit time is given by $C=\sum_k\lambda_1(k)p_{n_k}(r+t_1b_k)+\lambda_2 Wh$.

At some $n=n^*=(n_1^*,..,n_s^*)$, $C$ is minimized and the revenue is maximized since the other terms are not affected by changing $n$. In this way, we can choose optimal $n$ value. Note that even with any other pricing scheme, the same procedure can be followed.

### 4.2. Search Algorithm

The cost function is convex and does not have derivative information. Since it is defined only at integral points of $n_k$, we modify the dichotomous search algorithm, which falls under the category of sequential line search problems where the values of function at the previous iterations are used to determine the suceeding points (Bazaara et al [2]).

1. Let the allowable final length of uncertainity, $l > 0$. We can take $l$=1. Let $[a_1, b_1]$ be the initial interval of uncertainity, let $k = 1$, and go to step 2. Then, $a_1 = 1$, $b_1 = \lceil R/b \rceil - 1$.
2. If $b_k - a_k < l$, stop; the minimum point lies in the interval $[a_k, b_k]$. Otherwise, $\theta_k$ and $\mu_k$ are given by $\theta_k = \lfloor (a_k + b_k)/2 \rfloor$ , $\mu_k = \theta_k + 1$.
3. If $f(\theta_k) < f(\mu_k)$, let $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$. Otherwise, let $a_{k+1} = \theta_k$ and $b_{k+1} = b_k$. Replace $k$ by $k + 1$, and go to step 2.

Example:

Parameters:$\lambda_1 = 2$, $\lambda_2 = 10$, $\mu_1 = 1$, $\mu_2 = 1$, $R = 17$, $b = 2$.

Steps of the algorithm:

Step 1: The initial interval is

$a_1 = 1, f(1) = 8.4476$, $b_1 = \lceil 17/2 \rceil - 1 = 8, f(8) = 4.6127$.

Step 2: $k = 1$, if $b_k - a_k = 1$, stop. $b_1 - a_1 = 7$. The minimum point lies between 1 and 8.

$$\theta_1 = \lfloor (a_1 + b_1)/2 \rfloor = \lfloor (1 + 8)/2 \rfloor = 4, \mu_1 = \theta_1 + 1 = 5.$$

Step3: $f(\theta_1 = 4) = 4.4261$,$f(\mu_1 = 5) = 4.3869$.

Since $f(\theta_1) > f(\mu_1)$, the minimum lies between $a_2 = \theta_1 = 4$ and $b_2 = b_1 = 8$.

Step2: $k = 2$, if $b_2 - a_2 < 3$, stop. $b_2 - a_2 = 4$. The minimum point lies between 4 and 8.

$$\theta_2 = \lfloor (a_2 + b_2)/2 \rfloor = \lfloor (4 + 8)/2 \rfloor = 6, \mu_2 = \theta_2 + 1 = 7.$$

Step 3: $f(\theta_2 = 6) = 4.4912$, $f(\mu_2 = 7) = 4.5741$.

Since $f(\theta_2) < f(\mu_2)$, the minimum lies between $a_3 = a_2 = 4$ and $b_3 = \mu_2 = 7$.

Step 2: $k = 3$, if $b_3 - a_3 < 3$, stop. $b_3 - a_3 = 3$. The minimum point lies between 4 and 7.

$$\theta_3 = \lfloor (a_3 + b_3)/2 \rfloor = \lfloor (4 + 7)/2 \rfloor = 5, \mu_3 = \theta_3 + 1 = 6.$$

Step 3: $f(\theta_3 = 5) = 4.3865$, $f(\mu_3 = 6) = 4.4912$.

Since $f(\theta_3) < f(\mu_3)$, the minimum lies between $a_4 = a_3 = 4$ and $b_4 = \mu_3 = 6$.

Step 2: $k = 4$, if $b_4 - a_4 < 3$, stop. The minimum occurs at 5 and the minimum cost is 4.3865.

Thus we show how the search algorithm can be used to find the minimum. With this, all the steps in the solution methodology are complete. In section 5, we will illustrate with an example how the methodology is used.

## 5. Illustration

In this section, we start with a scenario where the input parameters remain unchanged through out the period of analysis, which we call 'single period analysis'. Section 5.2 has an example where the parameters vary over time, which we call 'multi period analysis'. The purpose of section 5.1 is to illustrate the methodology for fixed parameter scenario, which would help us understand the example with parameter variations with time better.

### 5.1. Single Period analysis

Let us consider a webserver where both the classes of requests access it. By single period, we mean that all the parameters of the problem remain the same through out the analysis.

#### 5.1.1 Example

In the example, let us consider 5 companies with the following parameters that the shared server is serving. The parameters are $S = 12$, $\lambda_2 = 5$, $\mu_2 = 1$ and other parameters are given in table 2. Consider the scenario when 5 companies approach the shared server with the average parameters like arrival rate, holding time of class 1 requests and an overall class 2 requests. The problem is how to allocate resources to these 5 companies so that the revenue is maximized. In other words, the problem is to find the values of $n_1$, $n_2$, $n_3$, $n_4$ and $n_5$. Note that the bandwidth required or allocated for each class 1 application are different for each company. Based on the analysis done using matrix geometric method, cost function and the search algorithm, we can find the values of $n_1$, $n_2$, $n_3$, $n_4$ and $n_5$.The values of $r$ and $h$ taken are respectively 5 and 1. For the example problem, the solution obtained is $n_1 = 1$,$n_2 = 1$,$n_3 = 3$,$n_4 = 1$ and $n_5 = 4$. Thus, optimal $n$ values can be set to maximize the revenue or minimize costs.

### 5.2. Multi-period analysis

It is quite common that the arrival rates and holding times vary over time according the time of

| Company | $b$ | $\lambda_1$ | $\mu_1$ |
|---|---|---|---|
| 1 | 3 | 4 | 2 |
| 2 | 2 | 4 | 1 |
| 3 | 1 | 4 | 1 |
| 4 | 2 | 8 | 1 |
| 5 | 1 | 6 | 1 |

**Table 2.** Example for Single period analysis

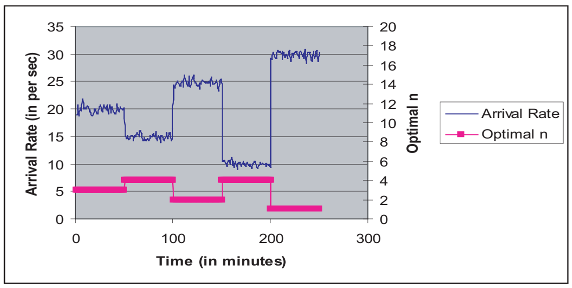


**Figure 2.** Multiperiod analysis with varying $\lambda_2$

the day and day of the week. There are some peak periods and non-peak periods when the arrival rates change over time. The processing capacity remains constant over the entire period of analysis. The value of $n$ can be optimized after every time interval. The data of the arrival times for a small window of time can be taken and updated arrival rates can be calculated and can be taken as the new input for the optimization. We will clearly see how autonomic control can be effectively used for optimization in webserver. The time interval should not be too big so that we have stale information. At the same time, we assume that the time period is long enough that steady state is reached.

#### 5.2.1 Example 1

We will illustrate how $n_k$ can be changed over time under time varying request characteristics. For that purpose, we consider a system with only one company where $\lambda_2$ varies with time. The request characteristics are: $\lambda_1 = 2/\text{s}$, $\mu_1 = 1/\text{s}$, $\mu_2 = 2/kb$ remain unchanged. The other parameters are $R = 24$ and $b_1 = 5$. The figure 2 shows how the parameter $\lambda_2$ changes with time and how $n_1$ is changed accordingly.

| Period | I | II | III | IV | |
|---|---|---|---|---|---|
| Comp. | $\lambda_1, \mu_1$ | $\lambda_1, \mu_1$ | $\lambda_1, \mu_1$ | $\lambda_1, \mu_1$ | $b$ |
| 1 | 4,1 | 4,2 | 4,1 | 6,0.8 | 4 |
| 2 | 3,1 | 3,1 | 3,1.5 | 3,1.5 | 2 |
| 3 | 2,1 | 2,1 | 2,1.5 | 1,1 | 2 |
| 4 | 5,1 | 5,1 | 5,1.5 | 5,2.5 | 3 |
| 5 | 4,1 | 4,1 | 4,2 | 2,3 | 1 |
| $\lambda_2, \mu_2$ | 120,20 | 120,20 | 120,20 | 100,20 | |

**Table 3. Multiperiod Analysis:Example 2**

#### 5.2.2 Example 2

Let us take an example problem with 5 companies over four time periods. The arrival rates of the class 1 and class 2, holding times of class 1 and average file size of class 2 remain the same throughout the small time window (1 hour). The parameters mentioned above, however, change from one time window to another. The parameters $b_k$ remain constant throughout the period of analysis. The parameters are $S = 18$, $t_1 = 15$, $r = 5$ and $h = 1$. The other parameters are given in table 3.

Suppose the webserver did not have sophistication to autonomically adjust the resource allocation. For time period I, $n =$ (1,1,1,1,4) is obtained by the single period analysis. A server without autonomic adjustment would have the $n$ value fixed over all the four periods. In time period II, there is a change in $\lambda_1(1)$, everything else unchanged. The optimal value of $n$ is (2,1,1,1,2). If the resources are allocated accordingly, there will be 3% savings. Similarly, in time period III, there are some small changes in some other parameters. The savings, if $n$ is optimally chosen, is 5%. If there is variation in all the parameters, the savings could be quite large as illustrated by time period IV. The saving in period IV is as large as 12%. The primary purpose of this section is for illustration of the fact that there can be saving if the system is run optimally. When the magnitude of the arrival prrocessing rates are large, the savings could be a lot more than what we showed. Thus, the example illustrates how the total costs incurred can be minimized or revenue is maximized if the resources are tweaked optimally when the parameters change over time.

### 5.3. Putting into practice

The procedure to be followed to put everything into practice is as follows. This would also knit all the sections of the paper together. The first step

| Period | $n^*$ | mincost | const $n$ | % diff |
|---|---|---|---|---|
| I | 1,1,1,1,4 | 477 | 477 | - |
| II | 2,1,1,1,2 | 378 | 388 | 3 |
| III | 1,1,1,2,3 | 502 | 526 | 5 |
| IV | 1,2,1,2,1 | 531 | 600 | 12 |

**Table 4. Solution to example 2**

is to identify the time period interval so that it is short enough for the arrivals to be assumed independent and identically distributed. The next step is to estimate the arrival rates and processing times over the interval based on time series model i.e., the previous interval requests and the general pattern based on time of the day and day of the week. Due to lack of space, we have not discussed ways to estimate arrival rates, which can be found in several articles. Once we have inputs ready, the performance measures, have to be calculated using the analysis done in section 3. The resource allocation is done by appropriately chosing $n_k$ as done in section 4, where the cost function is calculated (section 4.1) and optimal allocation is done using search algorithm (section 4.2) as illustrated in section 5.2. This would be practically impossible to implement if it is done manually. The solution to the problem obviously is autonomic computing, which would enable the system to run efficiently over time.

## 6. Concluding remarks

We consider a shared data center with multiclass requests for various companies. We develop an analytical model to obtain the system performance measures. Using the performance measures of the two classes and a pricing scheme, a cost model was developed. The objective was to control the maximum number of class 1 requests to be allowed inside or control the admission of class 1 requests appropriately to maximize the revenue of the shared data center. Under dynamically varying web loads, static admission control would lead to either inefficient utilization or loss of revenue and customers. The only way to efficiently run the system is to autonomically control the admission of the requests of the streaming traffic.

**Acknowledgements**

The research is partially supported by NSF Grants ANI-0219747 and ACI-0325056. The authors thank Prof. George Kesidis, Prof. Raj Acharya, Prof. Anand Sivasubramaniam, Dr. Donna Ghosh, Prof. Venkatesh Sarangan for initial discussions that led to the formulation of this research.

## References

[1] T. Abdelzaher, K.G. Shin, and N. Bhatti. Performance Guarantees for Webserver End-Systems: A Control-Theoretical Approach. *IEEE Transactions on Parallel and Distributed Systems*, 13(1), January, 2002.

[2] M. S. Bazaraa, H. D. Sherali, C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. 2nd Edition, John Wiley and Sons, Inc, 1993.

[3] A. Chandra, W. Gong, P. Shenoy. Dynamic Resource Allocation for Shared Data Centers using Online Measurements, in *Proceedings of the Eleventh IEEE/ACM International Workshop on Quality of Service, Monterey, CA*, June 2003.

[4] J. Chase, D. Anderson, P. Thakkar, A. Vahdat, and R, Doyle. Managing Energy and Server Resources in hosting Centers. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, pp. 103-116, October 2001.

[5] V. G. Kulkarni. *Modeling and Analysis of Stochastic Systems.* Chapman and Hall texts in Statistical series, CRC press, 1995.

[6] G. Latouche, C. Pierce, and P.Taylor. Invariant measures for quasi-birth and death processes, *Stochastic Models*, 1997.

[7] C. Lu, G. Alvarez, and J.Wilkes. Aqueduct: Online Data Migration with Performance Guarantees. In *Proceedings of the Conference on File and Storage Technologies*, January 2002.

[8] M.F. Neuts. *Matrix Geometric Solutions in Stochastic Models: An algorithmic Approach.* John Hopkins University, University Press, 1981.

[9] V. Ramaswami. Algorithmic analysis of stochastic models: The changing face of mathematics, *Ramanujam Endowment Lecture at Anna University*, Chennai, India, December, 2000.

[10] A. Riska and E. Smirni. Mamsolver: A matix analytical methos tool, in the *Proceedings of the 12-th International Conference on Modeling Techniques and Tools for Computer and Communication Systems Performance Evaluation*, vol. LNCS 2324, pp.205-211, April 2002.

[11] J. W. Roberts. *A Service System with Heterogenous User Requirements*, Performance of Data Communication Systems and their Applications, pp. 423-431, 1981.

[12] K. W. Ross. *Multiservice Loss Models for Broadband Telecommunication Networks.* New York: Springer-Verlag, 1995.

[13] R. Shumway and S. Stoffer. *Time Series Analysis and its Applications.* Springer, New York, NY, 2000.

[14] J. R. Thorne. Modeling Complex Queuing Situations with Markov Processes, *URL: charybdis.mit.csu.edu.au/mantolov/CD/ICITA2002/papers/077-1.pdf*.