# Optimal Server Resource Allocation Using an Open Queueing Network Model of Response Time

Alex Zhang, Pano Santos, Dirk Beyer, Hsiu-Khuern Tang
Intelligent Enterprise Technology Laboratory
HP Laboratories Palo Alto
HPL-2002-301
October 21st , 2002*

service level agreement, web performance tuning, software performance modeling, response time, queueing, optimization

We present a nonlinear integer optimization model for determining the number of machines at each tier in a multi-tier server network. We utilize a result from an open queueing network model on the average response time; the optimization model is to minimize a weighted sum of total machines while satisfying the average response time constraint (or a service level expression that is convertible to the average response time). We then discuss the solution to this optimization model and present an algorithm that utilizes a bounding procedure.

* Internal Accession Date Only
Approved for External Publication

# 1 Introduction

This report examines the issue of dynamically provisioning computing resources, in the form of server machines, to meet a prescribed service level on average response time. This issue frequently arises in ASPs (Application Service Providers) and in web-service operations (e-commerce sites, for example).

The web-service system that we consider has a tiered structure. Typically three tiers are configured: web servers, application servers, and database servers. Within each tier, multiple machines can be provisioned to share the incoming workload which consists of a series of different types of web requests. The response time is defined in this report to be the time taken for a web request to go through the three-tiered system (i.e. "server-side response time").

Our question is to allocate an adequate number of machines to each tier in order to meet certain service level requirement. Applications that allow different number of machines at each tier are called horizontally scalable. We deal with such a horizontally scalable system where service level requirement is expressed in average response time of the system. Since the system response time is the sum of response times at each of the three tiers, the service level requirement can be met by different configurations in the number of machines at each tier. For example, the configuration of 3 web servers, 2 application servers, and 1 database server may achieve the same average response time as the configuration of 1 web server, 3 application servers, and 1 database server. The question, then, is to find a minimum number of machines in total to meet a prescribed service level requirement. Realizing that machines at different tiers might be different and incur different costs, the more general optimization problem is to minimize the total weighted sum of the machines, with the weights reflecting the "costs" of some kind (such as operating costs) of machines at different tiers.

This problem is a sub-problem in a more general problem of web transaction analysis and optimization (TAO) discussed in HPL Technical Report HPL-2002-45 (Garg et al 2002). We refer to this report for a more detailed discussion of the system and modeling efforts. Appleby et al (2001) describe various aspects of IBM's Oceano project which is centered on service level management and which includes many issues that the TAO project also sets out to solve. Menasce and Almeida (1998) present general issues in capacity planning issues for web performance.

In another HPL technical report HPL-2002-64, Santos, Zhu and Crowder (2002) address the issue of allocating resources (machines) in a tree-like topology of a data center,

considering performance constraints such as link bandwidth and switch capacity while minimizing communication delay between the assigned servers. They propose a mathematical optimization model with binary variables for optimally configuring the topology. This report differs from Santos et al (2002) in that we consider only the average response time performance measure. Our topology is a tiered structure. The use of the queueing model in expressing the average response time leads to a general non-linear integer problem, but with only a few general integer variables. These simplifications allow us to characterize the solution of the problem and devise an algorithm to find the solution efficiently.

We point out that the solution of the problem is relevant to the dynamic resource allocation at a Utility Data Center (UDC). The dynamic resource allocation at a UDC requires the integration of demand and capacity planning. This concept is known as Capacity on Demand. Capacity on Demand consists of optimizing the assignment of buffers (or shared) resources to satisfy the requirements of multiple applications. A UDC enables multiple applications to be hosted on a collection of shared resources, where the resources assigned to the applications may be increased when workload increases and reduced when workload reduces. This dynamic resource allocation allows flexible service level agreements (SLAs) in an environment where peak workload is much greater than the normal steady state. A key problem with Capacity on Demand is that the user needs are translated into a logical configuration, and physical resources are then assigned to the logical configuration to satisfy the resource requirements of the application. The problem addressed in this paper is precisely to determine the optimal resource requirements of an application under a workload in such a way that application SLAs are satisfied.

This report is organized as follows. We will next describe the queueing model which gives the expected (average) response time as a function of demand and the amount of resources allocated. In Section 3, we will then formulate the resource allocation problem as a mathematical program consisting of a linear objective function and a nonlinear constraint, with general integer decision variables. Sections 4 and 5 examine the two-tier problem and the general $n$-stage problem, and give the main mathematical results on feasibility conditions, bounds, and relaxed problem solutions. Section 6 presents our main algorithmic results on solving the practical integer optimization problems. In Section 7, we show that our optimization model is applicable to any service level metric that can be translated into the average response time.

## 2 A Queueing Model of Response Time

In HPL-2002-45, we have modeled the 3-tiered web system as an open queueing network, with a single type of critical resource (such as CPU). Related work addressing systems performance predictions on queueing time is the layered model by Rolia and Sevcik (1995), and Shiekh et al (1997). Here we present a simple (non-iterative) model for queueing time. Its simplicity is a computational advantage when used as input in an optimization routine which needs to determine the optimal resource allocation frequently and quickly. In our simple model, each machine is modeled as a processor sharing queue. Assuming that all machines at the same tier are identical, and that the workload balancing and scheduling is such that the workload is shared equally among the machines at the same tier, this queueing model gives the average response time of the 3-tier system as follows:

$$E[R] = \frac{E[S_1]}{1 - \lambda_1 E[S_1]/N_1} + \frac{E[S_2]}{1 - \lambda_2 E[S_2]/N_2} + \frac{E[S_3]}{1 - \lambda_3 E[S_3]/N_3} \quad (1)$$

where $E[S_1]$, $E[S_2]$, and $E[S_3]$ are the expected processing times (or service demands) of a request on the critical resource (such as CPU) at web, application and database tiers, respectively; $\lambda_1$, $\lambda_2$, and $\lambda_3$ are the arrival rates to the three tiers (e.g. $\lambda_1$ is the *total* arrival rate to all the web servers in the web tier); and $N_1$, $N_2$ and $N_3$ are the number of servers in each of the three tiers, respectively.

The only inputs to the average response time model are the arrival rates (or throughput) ($\lambda_1$, $\lambda_2$, $\lambda_3$), the number of servers at each tier ($N_1$, $N_2$, $N_3$), and the average service times ($E[S_1]$, $E[S_2]$, $E[S_3]$). The average service times (service demands) can be computed from the measured utilization rates of the critical resource (CPU), through the following relationship:

$$E[S_1] = U_1/(\lambda_1/N_1),$$

where $U_1 < 1$ is the measured per-machine average utilization rate; $(\lambda_1/N_1)$ is the per-machine average throughput rate.

We have tested the accuracy of the average response time formula (1) using measured response time data recorded from a test bed (the "PetStore" e-tailing web site, see Garg et al 2002) consisting of 1 web server, 2 application servers, and 1 database server. The data log also includes CPU utilization percentages for all machines in 5-minute intervals. Under various workload scenarios (different levels of request intensity and different mix percentages of request types), the queueing formula (1) was found to be from 4% to 27% of the actual average response time.

# 3 Resource Allocation with the Average Response Time Measure

Our model of resource optimization would then be to minimize a total weighted "cost" (represented by machine replacement cost or operating cost) of the system, subject to the constraint that the average system response time to be less than or equal to a certain level ($T$, such as 1 second). That is:

$$\min_{N_1,N_2,N_3} \quad h_1N_1 + h_2N_2 + h_3N_3 \tag{2}$$

$$\text{s.t.} \quad E[R] = \frac{E[S_1]}{1-\lambda_1 E[S_1]/N_1} + \frac{E[S_2]}{1-\lambda_2 E[S_2]/N_2} + \frac{E[S_3]}{1-\lambda_3 E[S_3]/N_3} \leq T; \tag{3}$$

$$N_1, N_2, N_3 \text{ positive integers,}$$

where weights $h_1$, $h_2$, and $h_3$ (all assumed to be strictly positive) are to reflect the difference in the costs of the different servers; for example, if a database server machine is twice as expensive as a web server machine, then $h_3 = 2 \cdot h_1$. As we mentioned earlier, if costs are difficult to ascertain, a convenient simplification would be to minimize the total number of machines ($N_1 + N_2 + N_3$); i.e. to set $h_1 = h_2 = h_3 = 1$.

When $N_1$ increases, the values of $\lambda_1$ and $E[S_1]$ remain the same, as $\lambda_1$ is the aggregate arrival rate to the web tier. Hence, parameters $\lambda_1, \lambda_2, \lambda_3$ and $E[S_1], E[S_2], E[S_3]$ are true input parameters in the optimization model.

Finally, although we have given the explicit formulation for the 3-tiered system, we could easily generalize the formulation to any $n$-stage system (each tier is a stage). However, we will begin our discussion about the solution of the optimization model with the two-stage model.

# 4 The Two-Stage System

If the number of servers at a particular tier is fixed (say there can be only one database server), the optimization problem is simplified. The 2-stage problem provides a case for mathematical examination.

Consider the 2-stage version of this problem:

$$\min_{N_1,N_2} \quad h_1N_1 + h_2N_2 \tag{4}$$

$$\text{s.t.} \quad \frac{s_1}{1-u_1/N_1} + \frac{s_2}{1-u_2/N_2} \leq T; \tag{5}$$

$$N_1, N_2 \text{ positive integers.}$$

where $s_i \equiv E[S_i]$ and $u_i \equiv \lambda_i E[S_i]$ are input parameters. Here $u_i$ can be interpreted as an expression of workload; for example, $u_1 = 1.5$ signifies that the aggregate workload on web server tier is equivalent to 1.5 web server machines working full time (at 100% utilization). Clearly, we can also replace the non-negativity constraints $N_1, N_2$ positive integers by a tighter $N_1 \geq \max(u_1, 1)$ and $N_2 \geq \max(u_2, 1)$.

The basic tradeoff expressed through the above optimization problem is that of balancing fewer expensive machines with a resulting longer response time against more numerous cheaper machines with a resulting shorter response time.

We first look at the feasibility of this simple optimization problem.

**Proposition 1**. The 2-stage problem is feasible if and only if $s_1 + s_2 < T$.

**Proof**. Say $s_1 + s_2 < T$. Then let $N_1 \to \infty$ and $N_2 \to \infty$, we have

$$\frac{s_1}{1 - u_1/N_1} + \frac{s_2}{1 - u_2/N_2} \to s_1 + s_2 < T.$$

Hence the constraint can be satisfied by large $N_1$ and $N_2$; the problem is feasible.

Now assume that the problem is feasible. Then there exist positive integers $N_1^0$ and $N_2^0$ such that

$$\frac{s_1}{1 - u_1/N_1^0} + \frac{s_2}{1 - u_2/N_2^0} \leq T.$$

Hence,

$$s_1 + s_2 < \frac{s_1}{1 - u_1/N_1^0} + \frac{s_2}{1 - u_2/N_2^0} \leq T.$$

||

We also find lower bounds on the feasible $N_1$ and $N_2$:

**Proposition 2**. The feasible $N_1$ and $N_2$ satisfy

$$N_1 > \frac{u_1(T - s_2)}{T - s_1 - s_2}; \quad N_2 > \frac{u_2(T - s_1)}{T - s_1 - s_2}. \tag{6}$$

**Proof**. From the feasibility condition,

$$T \geq \frac{s_1}{1 - u_1/N_1} + \frac{s_2}{1 - u_2/N_2} > \frac{s_1}{1 - u_1/N_1} + s_2;$$
$$\frac{s_1}{1 - u_1/N_1} < T - s_2;$$
$$N_1 > \frac{u_1(T - s_2)}{T - s_1 - s_2}.$$

Similar derivation for the inequality for $N_2$. ||

**Remark**. Since $N_1$ and $N_2$ are required to be integers, we can tighten the inequality to $N_1 \geq \lceil \frac{u_1(T-s_2)}{T-s_1-s_2} \rceil$ where $\lceil x \rceil$ stands for the smallest integer greater than or equal to $x$.

**Example**. Let $s_1 = 0.3$, $s_2 = 0.5$, $u_1 = 0.3$, $u_2 = 0.4$, $T = 1$. Notice that $s_1 + s_2 = 0.8 < T = 1$, hence the necessary and sufficient condition for feasibility of the optimization problem is satisfied. Then

$$\begin{aligned} N_1 &> \frac{0.3(1-0.5)}{1-0.3-0.5} = 0.75; \quad N_1 \geq 1; \\ N_2 &> \frac{0.4(1-0.3)}{1-0.3-0.5} = 1.4; \quad N_2 \geq 2. \end{aligned}$$

||

The bounds on $N_1$ and $N_2$ will facilitate the numerical search for the optimal values. In general, the numerical search will have to enumerate all integer pairs $(N_1, N_2)$ satisfying Proposition 2. In section 6, we will give an iterative algorithm for tightening these bounds, which we will utilize in an algorithm for obtaining the integer optimal solution. The continuous-variable version of the optimization problem, however, has a closed form solution.

**Proposition 3**. The continuous-variable version of the 2-stage problem has the following optimal solution:

$$\begin{aligned} N_1^c &= u_1 + \sqrt{\gamma s_1 u_1 / h_1}; \\ N_2^c &= u_2 + \sqrt{\gamma s_2 u_2 / h_2}; \end{aligned}$$

where $\gamma > 0$ is the shadow price of the required average response time and is given by

$$\sqrt{\gamma} = \frac{\sqrt{h_1 s_1 u_1} + \sqrt{h_2 s_2 u_2}}{T - s_1 - s_2}.$$

The objective function value is given by

$$Z^c = h_1 u_1 + h_2 u_2 + \sqrt{\gamma} \left( \sqrt{s_1 u_1 h_1} + \sqrt{s_2 u_2 h_2} \right).$$

**Proof**. Consider the Lagrangian:

$$L(N_1, N_2, \gamma) = h_1 N_1 + h_2 N_2 + \gamma \left( \frac{s_1}{1 - u_1/N_1} + \frac{s_2}{1 - u_2/N_2} \right).$$

Take partial derivative w.r.t. $N_1$ and set to zero we obtain:

$$\frac{\partial L}{\partial N_1} = h_1 - \gamma \cdot \frac{s_1 u_1}{(N_1 - u_1)^2} = 0,$$

hence

$$N_1^c = u_1 + \sqrt{\gamma s_1 u_1 / h_1}.$$

Similarly we obtain $N_2^c$. Since $N_1^c$ and $N_2^c$ are continuous, the constraint

$$\frac{s_1}{1 - u_1/N_1} + \frac{s_2}{1 - u_2/N_2} \leq T$$

must be binding ($= T$); substituting $N_1^c$ and $N_2^c$ into the equality and simplify algebraically we can solve for $\gamma$. ||

**Remark**. Proposition 1 guarantees $T - s_1 - s_2 > 0$ if the problem is feasible. Also, $N_1^c > u_1$ and $N_2^c > u_2$.

**Example** (continued). Let $s_1 = 0.3$, $s_2 = 0.5$, $u_1 = 0.3$, $u_2 = 0.4$, $T = 1$. Introduce $h_1 = 1$, $h_2 = 2$. We solve the continuous-variable version of the 2-stage problem as follows.

$$\begin{aligned}
\sqrt{\gamma} &= \frac{\sqrt{h_1 s_1 u_1} + \sqrt{h_2 s_2 u_2}}{T - s_1 - s_2} = \frac{\sqrt{1 \cdot 0.3 \cdot 0.3} + \sqrt{2 \cdot 0.5 \cdot 0.4}}{1 - 0.3 - 0.5} = 4.6623; \\
N_1^c &= u_1 + \sqrt{\gamma s_1 u_1 / h_1} = 0.3 + 4.6623\sqrt{0.3 \cdot 0.3/1} = 1.6987; \\
N_2^c &= u_2 + \sqrt{\gamma s_2 u_2 / h_2} = 0.4 + 4.6623\sqrt{0.5 \cdot 0.4/2} = 1.8743; \\
Z^c &= h_1 N_1^c + h_2 N_2^c = 5.4473.
\end{aligned}$$

The integer-optimal solution (found through enumeration) is $N_1^* = N_2^* = 2$, with achieved average response time of 0.978 seconds (better than the required $T = 1$ second), and objective function value of 6 (worse than 5.4473 of the continuous solution).

||

From Proposition 3, the continuous-variable solution of the problem has some interesting properties. The shadow price ($\gamma$) of the required average response time is the increase in the objective function value (cost) per unit of decrease on the response time; for example, "$\gamma$ = \$20/second" denotes that each second in decreased response time will cause an approximately \$20 increase in the total objective function value (cost). The shadow price as an output of the mathematical optimization model may have pricing

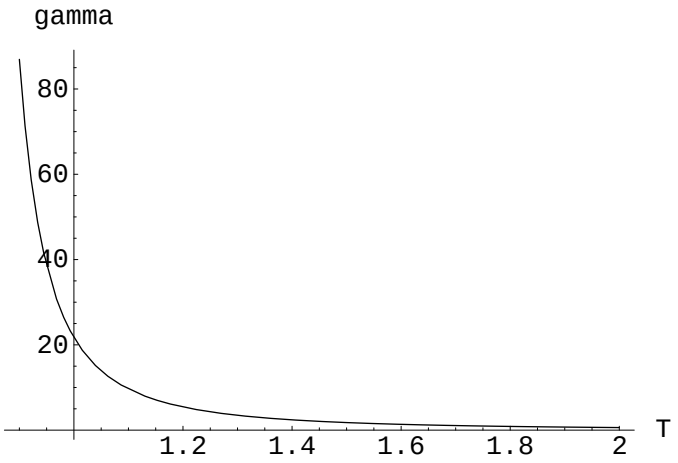


Figure 1: Shadow price $\gamma$ for the required average response time $T$. Parameters: $s_1 = 0.3$, $s_2 = 0.5$, $u_1 = 0.3$, $u_2 = 0.4$, $h_1 = 1$, $h_2 = 2$.

implications on SLAs. We notice that the shadow price is related to $T$ by the following relationship: $\gamma \propto 1/(T-c)^2$ (where $c \equiv s_1 + s_2$). The shadow price decreases rapidly as the allowable response time increases. On the other hand, the number of machines at each tier, as well as the objective function value (cost), is approximately inversely proportional to the required response time ($N_i^c \propto 1/(T-c)$; $Z^c \propto 1/(T-c)$). Figure 1 and Figure 2 show the relationship of $\gamma$, $N_1^c$ and $N_2^c$ as a function of the required response time $T$.

# 5 The General Case: An $n$-Stage System

All three propositions in the 2-stage system case can be generalized to $n$-stage systems. Here are the restatements of the old results.

**Proposition 1**. The $n$-stage problem is feasible if and only if $\sum_{i=1}^{n} s_i < T$.

**Proposition 2**. The feasible $N_i$, $i = 1, 2, \ldots, n$, satisfy

$$N_i > \frac{u_i(s_i + T - \sum_{j=1}^{n} s_j)}{T - \sum_{j=1}^{n} s_j}. \tag{7}$$

**Proposition 3**. The $n$-stage problem has the following continuous-variable solution:

$$N_i^c = u_i + \sqrt{\gamma s_i u_i / h_i}; \tag{8}$$

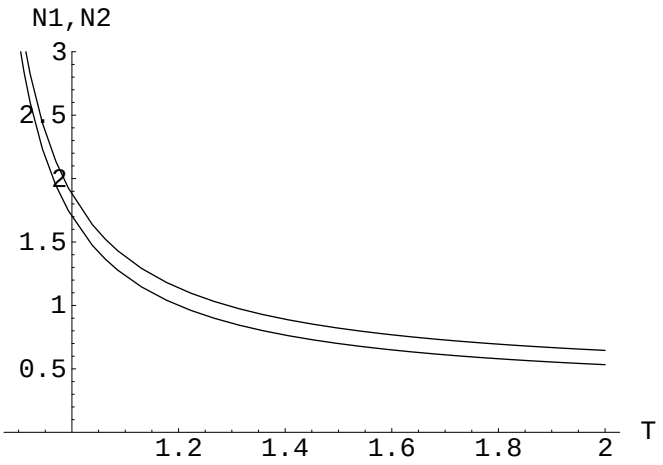


Figure 2: $N_1^c$ (lower curve) and $N_2^c$ (top curve) as a function of the required average response time $T$. Parameters: $s_1 = 0.3$, $s_2 = 0.5$, $u_1 = 0.3$, $u_2 = 0.4$, $h_1 = 1$, $h_2 = 2$.

where $\gamma > 0$ is the shadow price of the expected response time and is given by

$$\sqrt{\gamma} = \frac{\sum_{i=1}^{n} \sqrt{h_i s_i u_i}}{T - \sum_{i=1}^{n} s_i}. \tag{9}$$

The objective function value is

$$Z^c = \sum_{i=1}^{n} h_i u_i + \sqrt{\gamma} \sum_{i=1}^{n} \sqrt{s_i u_i h_i}. \tag{10}$$

# 6 The Integer Program: A Solution Algorithm

We point out that the continuous-variable version of the problem is only a relaxation of the original integer problem. We emphasize that the integer optimal solutions might be far away from the continuous solutions as given by equation (8); that is, if the continuous solution is $(N_1, N_2) = (1.44, 1.79)$, the integer optimal solution might be $(1, 4)$. Rounding up each of the continuous $N_i$ will clearly lead to a feasible, but not necessarily optimal, solution.

Proposition 2 gives a lower bound on each $N_i$, and can be utilized as the starting values in a numerical search algorithm. The following proposition further gives a (dynamic) upper bound on the search.

**Proposition 4**. If $(N_1^0, \ldots, N_n^0)$ are integers and satisfy the expected average response time constraint, then any other integer solutions $(N_1, \ldots, N_n)$, where $N_i \geq N_i^0$ for all $i$

with strict inequality for at least one $i$, cannot be optimal.

**Proof**. Suppose integers $N_i \geq N_i^0$ for $i = 2, \ldots, n$, with $N_1 > N_1^0$. Then $(N_1, N_2, \ldots, N_n)$ is a feasible integer solution, since the average response time $E[R]$ is strictly decreasing in each $N_i$. However, $\sum_{i=1}^n h_i N_i > \sum_{i=1}^n h_i N_i^0$; hence $(N_1, N_2, \ldots, N_n)$ cannot be optimal. ||

We further obtain bounds on the integer-optimal objective function values (costs).

**Proposition 5**. Let $Z^*$ be the integer-optimal objective function value, and $N_i^c$ be the continuous-variable solution as given by (8) with objective function value $Z^c$ as given by (10). Then

$$\sum_{i=1}^n h_i N_i^c \equiv Z^c \leq Z^* \leq Z^{UB} \equiv \sum_{i=1}^n h_i \lceil N_i^c \rceil. \tag{11}$$

**Proof**. Since the continuous-variable model is a relaxation of the integer model, we have $Z^c \leq Z^*$. Since each $\lceil N_i^c \rceil \geq N_i^c$, the integer solution $(\lceil N_1^c \rceil, \lceil N_2^c \rceil, \ldots, \lceil N_n^c \rceil)$ is a feasible solution since the response time function (the left-hand-side of the constraint) is decreasing in each $N_i$. Hence the second inequality holds. ||

For the previous numerical example introduced in previous sections, $N_1^c = 1.6987$, $N_2^c = 1.8743$, and $Z^c = 5.4473$, hence Proposition 5 yields the bounds $5.4473 \leq Z^* \leq 6$ (the true $Z^* = 6$).

Figure 3 illustrates the use of Proposition 5 in bounding the integer optimal solution.

To provide a heuristic solution to the integer model, we first solve the continuous-variable problem (in closed-form), and simply round *up* each $N_i^c$ if $N_i^c$ is not an integer (i.e. to the next higher integer), we will obtain an integer feasible solution, with a guaranteed performance within $Z^{UB} - Z^c = \sum_{i=1}^n h_i(\lceil N_i^c \rceil - N_i^c)$ from the true optimum. When $N_i^c$, $i = 1, 2, \ldots, n$, are relatively large (say greater than 3), the relative gap between the heuristic solution and the true integer optimal solution is quite small.

## 6.1 Improved Bounds with an Iterative Procedure

Proposition 5 is useful in the search for integer optimal solution in that we know any integer optimal solution must satisfy $h_1 N_1 + h_2 N_2 + \cdots + h_n N_n \geq Z^c$; hence if $N_1, \cdots, N_{n-1}$ have been determined, we must have $N_n \geq (Z^c - h_1 N_1 - \cdots - h_{n-1} N_{n-1})/h_n$. This narrows the search range in the $n$-th dimension $(N_n)$.

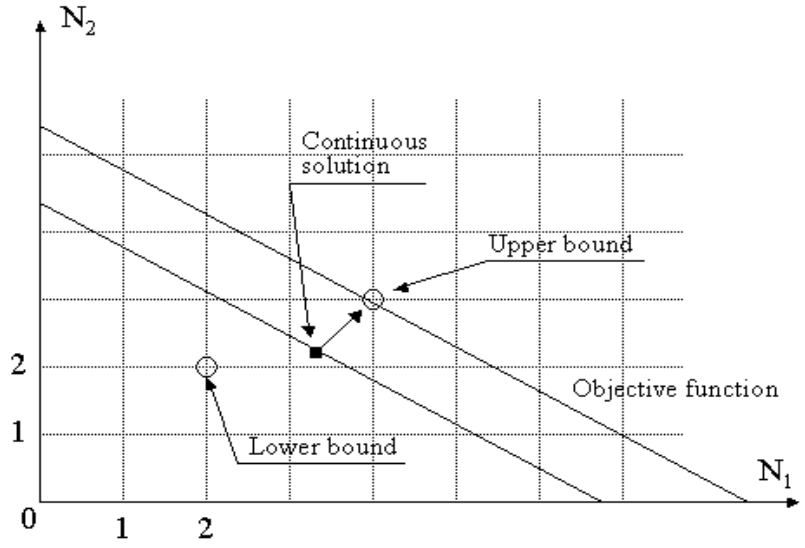


Figure 3: An illustration of the lower and upper bounds

Proposition 5 also enables us to improve the bounds as given by Proposition 2. The procedure is as follows.

Step 0. (Initial lower bounds) Following Proposition 2, set $N_i^{LB} = \lceil \frac{u_i(s_i + T - \sum_{j=1}^n s_j)}{T - \sum_{j=1}^n s_j} \rceil$.

Step 1. (Upper bounds) For $i = 1, 2, \ldots, n$, obtain an upper bound on $N_i$ by using the inequality $N_i \leq (Z - \sum_{j \neq i} h_j N_j)/h_i \leq (Z^{UB} - \sum_{j \neq i} h_j N_j^{LB})/h_i$. Set $N_i^{UB} = \lfloor (Z^{UB} - \sum_{j \neq i} h_j N_j^{LB})/h_i \rfloor$.

Step 2. (Improved lower bounds) For $i = 1, 2, \ldots, n$, obtain an improved lower bound on $N_i$ by using the inequality $\frac{s_i}{1 - u_i/N_i} + \sum_{j \neq i} \frac{s_j}{1 - u_j/N_j} \leq T$ where for $j \neq i$, $N_j$ is set to its upper bound $N_j^{UB}$. Set new $N_i^{LB} = \lceil \frac{u_i}{1 - (s_i/t_i)} \rceil$ where $t_i \equiv T - \sum_{j \neq i} \frac{s_j}{1 - (u_j/N_j^{UB})}$.

Step 3. (Improved bounds) If for some $i$, the new $N_i^{LB}$ is better (greater) than the old $N_i^{LB}$, then replace all old bounds by the new bounds and go back to Step 1; Otherwise, stop and output the current $N_i^{LB}$ and $N_i^{UB}$.

The above algorithm uses the service level constraint (on average response time) to find the lower bounds, and then in turn uses the optimality condition $\sum_{i=1}^n h_i N_i \leq Z^{UB}$ to

find the upper bounds. The upper bounds may lead to improved lower bounds from the response time constraint; the iteration terminates when no further improvements can be made.

We illustrate the above bounding algorithm using the previous numerical example (where $n = 2$). Recall $s_1 = 0.3$, $s_2 = 0.5$, $u_1 = 0.3$, $u_2 = 0.4$, $h_1 = 1$, $h_2 = 2$.

Step 0. From Proposition 2, $N_1^{LB} = 1$; $N_2^{LB} = 2$.

Step 1. Upper bounds. $N_1 \leq 6 - 2N_2^{LB} = 6 - 2 \cdot 2 = 2$; hence $N_1^{UB} = 2$.
$N_2 \leq (6 - N_1^{LB})/2 = (6 - 2)/2 = 2$; hence $N_2^{UB} = 2$. (Since $N_2^{LB} = N_2^{UB} = 2$, we have $N_2 = 2$.)

Step 2. Improved lower bounds. $t_1 = T - \frac{s_2}{1-(u_2/N_2^{UB})} = 1 - \frac{0.5}{1-(0.4/2)} = 0.375$;
$N_1^{LB} = \lceil \frac{u_1}{1-(s_1/t_1)} \rceil = \lceil \frac{0.3}{1-(0.3/0.375)} \rceil = \lceil 1.5 \rceil = 2$ (an improvement).

Further iterations do not improve the current bounds, hence the procedure terminates with $N_1^{LB} = N_1^{UB} = N_2^{LB} = N_2^{UB} = 2$. In this numerical example, the bounding procedure happens (by chance) to lead to the unique optimal integer solution (without applying any optimization procedure).

## 6.2 A Numerical Search Algorithm

We first give a simple algorithm for solving the integer optimization problem with $n = 2$ (two tiers). We then give a recursive algorithm for solving a general $n$-stage problem.

Algorithm **TwoStage($T$)**: Given a required average response time $T$, find the optimal $N_1$ and $N_2$ to minimize $h_1N_1 + h_2N_2$.

Step 0. Feasibility test: If $s_1 + s_2 \geq T$ then the problem TwoStage($T$) is infeasible. Terminate and output "infeasible".

Step 1. Bounds: Compute the continuous-variable solution from equations (8), (9) and (10). Compute the lower and upper bounds $N_i^{LB}$ and $N_i^{UB}$ for $i = 1, 2, \ldots, n$ following the iterative bounding procedure outlined in the previous section. Set the optimal objective function value $Z$ to infinity.

Step 2. Set $N_1$ to its lower bound $N_1^{LB}$.

Step 3. Set $t_2 = (T - \frac{s_1}{1-u_1/N_1})$ and $N_2 = \lceil \frac{u_2}{1-(s_2/t_2)} \rceil$.

Step 4. Compute objective function value $h_1N_1 + h_2N_2$. If this is lower than $Z$, then set $Z = h_1N_1 + h_2N_2$ and remember $(N_1, N_2)$.

Step 5. If $N_1 < N_1^{UB}$ then increment $N_1$ by one and return to Step 3; otherwise terminate and output $(N_1, N_2)$.

To explain the above **TwoStage($T$)** algorithm, we note that we loop over $N_1$ from its lower bound to its upper bound. For any fixed $N_1$, we can calculate the smallest $N_2$ that still makes the required response time $T$ feasible; that is, $\frac{s_1}{1-u_1/N_1} + \frac{s_2}{1-u_2/N_2} \leq T$ (Step 3). We do not need to consider any other $N_2$, as any $N_2$ less than that given in Step 3 will be infeasible, and any $N_2$ greater than that given in Step 3 will result in a higher objective function value (by Proposition 4). The $(N_1, N_2)$ from Step 3 thus form the integer boundary of the feasible region.

It is possible to terminate **TwoStage($T$)** early if $N_2$ from Step 3 turns out to be equal to $N_2^{LB}$. Any higher $N_1$ (and same or higher $N_2$) will only lead to higher objective function value (Proposition 4).

The following is an algorithm that recursively solves a higher-dimension problem.

Algorithm $n$-**Stage($T$)**:

Step 0. Feasibility test: If $\sum_{i=1}^{n} s_i \geq T$ then the problem $n$-Stage($T$) is infeasible. Terminate and output "infeasible".

Step 1. Bounds: Compute the continuous-variable solution from equations (8), (9) and (10). Compute the lower and upper bounds $N_i^{LB}$ and $N_i^{UB}$ for $i = 1, 2, \ldots, n$ following the iterative bounding procedure. Set the optimal objective function value $Z$ to infinity.

Step 2. Set $N_n$ to its lower bound $N_n^{LB}$.

Step 3. Solve the problem $(n-1)$-**Stage($t_n$)** where $t_n = T - \frac{s_n}{1-u_n/N_n}$. If $(n-1)$-**Stage($t_n$)** returns "infeasible" then go to Step 4; Otherwise obtain (the returned) $(N_1, \ldots, N_{n-1})$. If $Z > \sum_{i=1}^{n} h_i N_i$, then set $Z = \sum_{i=1}^{n} h_i N_i$ and remember $(N_1, \ldots, N_{n-1}, N_n)$.

Step 4. If $N_n <$ its upper bound $N_n^{UB}$ then increment $N_1$ by one and return to Step 3; otherwise terminate and output $(N_1, N_2, \ldots, N_n)$.

We note that the computational complexity of the above algorithm (number of enumerations) is of the order of $\prod_{i=1}^{n}(N_i^{UB} - N_i^{LB})$ where $n$ is typically a small number (2 or 3).

# 7 Optimization Model Using the Critical Quantile Response Time

It is possible to specify the service level requirement in a critical quantile of the response time, such as "95% of the response times to be less than 5 seconds." In mathematical language, $\Pr\{R \leq Q\} \geq 0.95$, where $R$ is the response time, $Q$ is the critical quantile ($Q = 5$ seconds here).

If response time $R$ is exponentially distributed, then the requirement $\Pr\{R \leq Q\} \geq 0.95$ is equivalent to $E[R] \leq \alpha \cdot Q$ where $\alpha = 1/[-\ln(1 - 0.95)] \approx 0.33381$. Hence in case of exponentially distributed response time, the response time quantile requirement can be exactly translated into a requirement on the average response time.

In cases when the response time is not exponentially distributed, we can use Markov's inequality

$$\Pr\{R \geq Q\} \leq (E[R]/Q) \quad \text{or} \quad \Pr\{R \leq Q\} \geq 1 - (E[R]/Q). \tag{12}$$

to produce a surrogate (approximate) inequality $1 - (E[R]/Q) \geq 0.95$, or equivalently $E[R] \leq 0.05Q$. Hence, by letting $T \equiv 0.05Q$, the resulting optimization model would be identical to the one we have discussed.

The use of inequality $1 - (E[R]/Q) \geq 0.95$ guarantees the satisfaction of the requirement $\Pr\{R \leq Q\} \geq 0.95$. We want to point out that we found, through actual test data, that Markov's inequality is a loose inequality in the sense that the actual achieved response time performance when using the surrogate constraint $1 - (E[R]/Q) \geq \alpha$ is much better than predicted; i.e. the actual probability $\Pr\{R \leq Q\}$ is much greater than $\alpha$.

# 8 Concluding Remarks

In this document, we have looked at an optimization model using our computationally simple queueing results on the average response time. The optimization model determines the number of servers at each of the $n$ tiers, reflecting the cost differentials among the tiers. So the model will likely designate fewer expensive machines together with more numerous cheap machines so as to achieve the overall average response time.

The continuous-variable version of the optimization model leads to the closed form solution as given by (8), (9) and (10), which gives the "ideal" (in the sense if fractional machines are permissible) machine configurations. Figures 1 and 2 show that the ideal

number of machines is inversely proportional to the required average response time $T$. It is possible to consider the sensitivity of the ideal $N_i^c$ to other problem parameters ($h_i$, $s_i$, and $u_i$), by taking partial derivative of $N_i$ w.r.t. these parameters in equation (8) and noting that $\gamma$ is also a function of $h_i$, $s_i$, and $u_i$.

Finally, the optimization model is applicable (can be extended) to any service level expression that can be translated into the average response time.

# 9 Acknowledgement

We would like to thank Xiaoyun Zhu for her tremendously helpful comments which have clarified numerous aspects in the description of the problem context. We also like to thank Opher Baron for his comments on an earlier version of this report.

# 10 References

Appleby, K., S. Fakhouri, L. Fong, G. Goldszmidt, and M. Kalantar. 2001. Oceano — SLA based management of a computing utility, in IFIP/IEEE Intl. Symp. on Integrated Network Management, May 2001.

Garg, Pankaj K., Ming Hao, Cipriano Santos, Hsiu-Khuern Tang, Alex Zhang. 2002. Web Transaction Analysis and Optimization. HP Laboratories Technical Report, HPL-2002-45, March 4, 2002. Palo Alto, CA 94304.

Menasce, Daniel A. and Virgilio A.F. Almeida, 1998. *Capacity Planning for Web Performance*, Prentice Hall PTR, Upper Saddle River, NJ 07548.

Rolia, Jerry A. and K.C. Sevcik. 1995. The Method of Layers. *IEEE Transactions on Software Engineering*, Vol. 21, No. 8, August 1995.

Santos, Cipriano, Xiaoyun Zhu, Harlan Crowder. 2002. A Mathematical Optimization Approach for Resource Allocation in Large Scale Data Centers. HP Laboratories Technical Report, HPL-2002-64. Palo Alto, CA 94304.

Sheikh, Fahim, Jerry Rolia, Pankaj Garg, Svend Frolund, and Allan Shepherd. 1997. Layered Performance Modeling of a CORBA-based Distributed Application Design. *Proc. 4th International Conference on Analytical and Numerical Modeling Techniques with application to QUALITY OF SERVICE (QoS) MODELLING*, Singapore, September 1-4,

1997.