2011 IEEE International Conference on High Performance Computing and Communications

# A Priority-type Resource Allocation Approach in Cluster Computing

Kaiqi Xiong
College of Computer and Information Science
Rochester Institute of Technology
Rochester, NY 14623 USA

Kyoung-Don Kang
Department of Computer Science
State University of New York at Binghamton
Binghamton, NY 13902 USA

Xiao Chen
Department of Computer Science
Texas State University
San Marcos, TX 78666 USA

## Abstract

*In cluster computing, a service provider must allocate necessary computing resources for large-scale scientific computations to process a customer's service request according to a service level agreement (SLA) that is a set of quality of services (QoS) and a fee agreed between a customer and a service provider. Thus, Resource allocation is a challenging but very necessary problem in cluster computing. In an effort to maximize a service provider's profit, it is commonplace and important to prioritize customer services in favor of those who are willing to pay higher fees. In this paper, we consider a set of computing resources owned by a service provider who serves differentiated customer services subject to an SLA for scientific applications that often require parallel computation. The QoS defined in the paper includes percentile response time and cluster utilization. We present an approach for optimal resource allocation in cluster computing systems in that we minimize the total cost of computing resources owned by a service provider while satisfying multiple priority customer service requirements. Our simulation experiments show that the proposed approach is applicable to the resource allocation in a cluster computing system with multiple customer services.*

**Key words:** *Cluster computing, priority-type customer service, scheduling theory, resource allocation, and percentile response time*

## 1 INTRODUCTION

Resource allocation plays a crucial role in computing systems including a cluster computing system that is a group of linked computers, working together closely thus in many respects forming a single computer. Cluster computing has become an ideal parallel computing paradigm used in solving large-scale scientific problems. In an effort to maximize a service provider's profit, it is commonplace and important to prioritize customer services in favor of those who are willing to pay higher fees, so that they can receive faster services, see McWherter et al. [10]. Thus, *priority-type* resource management plays a key role in such a computing system (see Menasce and Casalicchio [11]).

In the research, we consider a priority-type resource allocation problem in cluster computing where a service provider processes petascale service request jobs for business customers by using a set of computing resources called cluster nodes shown in Figure 1. In order to achieve best performance, we often need to process such service jobs in parallel. Here, a customer represents a business entity that generates a stream of service request jobs at a specified rate to be processed by a service provider's resources according to QoS requirements and a given fee defined in an SLA. A simple intuition here is that the higher fee a customer pays the faster it receives service. Thus, we classify those users who are willing to pay the same amount of fee for their services as one type of customer, called a customer of class $j$. Such users generate a stream of service requests issued at a specified rate. We assume that there are $R$ different fees paid by all the users in the cluster computing system. That is, there are $R$ classes of customers in the system. Thus, all the users that belong to customers $1, 2, \cdots, R$ generate *multiple* streams of service requests at *multiple* specified rates and these service streams are simply called priority-type customer (service) requests. Moreover, we consider a *preemptive resume* priority discipline in the paper. That is, the service of a class $r_2$ customer can be interrupted if a higher-priority customer of class $r_1$ ($r_2 > r_1$) arrives during its

978-0-7695-4538-7/11 $26.00 © 2011 IEEE
DOI 10.1109/HPCC.2011.43

service. The interrupted customer resumes its service from where it stopped after the higher-priority customer, and any other customers with priority higher than $r_2$ that may arrive during its service, complete their service. Also, any customers with the same priority complete their services in order of arrival, i.e., first-in-first-out (FIFO).

In a cluster computing system as shown in Figure 1, a service provider owns a group of linked cluster nodes to process parallel computation in support of multiple customer service requests (e.g., see Aron et al. [1], Heath et al. [6] as well as Xiao and Ni [18]). The priority-type resource allocation problem occurs in multiple customer service requests in the computing system. A service request job is transmitted to each cluster node sequentially. Upon the completion of a service request job at the service provider, the final result is sent back to the customer. Cluster nodes are the service provider's computing resources such as processors and cluster servers/machines as discussed in Shin et al. [16] as well as Xiao and Ni [18] and they have the ability to work together for processing customer service jobs. For presentation simplicity, we explicitly regard each node's computing resources as cluster servers. Meanwhile, "computing resource" and "server" are used alternatively in this paper.

The priority-type resource allocation problem is to minimize the overall cost of the service provider's computing resources of each node allocated to the multiple customers in terms of the number of servers at each cluster node while satisfying multiple customers' QoS requirements defined in an SLA and a fee that is negotiated and agreed between the customer and the application provider in the SLA. QoS requirements may include a variety of metrics in performance, availability, and security. In the research, we consider the QoS metrics below: (1) *Percentile response time*-$\gamma\%$ ($0 \leq \gamma \leq 100$) of the time the response time, i.e., the time to execute a multiple-type service request job, is less than a pre-defined value. Specifically, a statistical bound on its response time than an average response time is considered since an average response time is heavily influenced by "outliers"; (2) *Cluster utilization*-It is the percentage of the time that the cluster node is utilized. They are typical QoS metrics used in an SLA (e.g., see Martin and Nilsson [8] and INTERNAP [9]). As stated in Xiong and Suh [20], a customer is in general concerned about response time rather than throughput (e.g., in the case of online shopping, a consumer often concerns with the time to receive his/her order after an order is put). Thus, throughput is not included in the research. Figure 1 extends the model in Xiong and Suh [20] into a novel prioritized service model that addresses multiple customer services, showing the ability to address the complexity of a cluster computing system.

In this paper, we formulate the priority-type resource allocation problem as a constrained optimization one. Through modeling multiple customer services in each cluster node as a priority queue, we first develop a way to calculate the percentile response time of a priority-type service request job and cluster utilization. Then, we further present a QoS-guaranteed algorithm to solve the priority-type resource allocation problem in cluster computing by providing the solution of the constrained optimization problem in an expression of the number of servers required in each cluster node. To the best of our knowledge, our study is the first attempt to analytically solve the priority-type resource provisioning problem under the consideration of *percentile response time and cluster utilization* for priority-type cluster computing systems. Our method is to first model the multiple customer services in such computing systems as a queueing network, and then to characterize and quantify the above performance metrics so that the solution of the priority-type resource problem is achieved.

The rest of the paper is organized as follows. In Section 2 we formulate the priority-type resource allocation problem with the SLA performance metrics. In Section 3 we model the priority-type service request jobs processed in SLA-based cluster computing as a queueing network and give an approximation approach to computing the percentile response time of each priority-type customer service request job in the queueing network. We further propose a QoS-guaranteed algorithm for solving the priority-type resource allocation problem subject to an SLA. Related work is presented in Section 4. Numerical experiments are given in Section 5. In the end, we conclude our results in Section 6.

## 2 The Priority-type Resource Allocation Problem

The priority-type resource allocation problem is concerned with the customer service request jobs depicted in Figure 1 where a priority-type service request job is transmitted to $m$ cluster nodes within a service provider. For presentation simplicity, assume that there is only one type of cluster server associated with cost $c_j$ within each cluster node. If there are multiple types of servers, each cluster node should be decomposed into several individual sub-nodes so that there is only one type of servers with the same cost within each sub-node. Furthermore, denote by $N_j$ the number of servers at node $j$ ($j = 1, 2, \cdots, m$).

Then, the priority-type resource allocation problem is to minimize the overall cost of the computing resources required while satisfying multiple customer service requirements in a cluster computing system. More precisely, the multiple customer resource allocation problem is quantified by solving for $d_j$ in the following provisioning problem:

$$I = \min_{d_1,\cdots,d_m} \left[ \sum_{j=1}^{m} d_j \, c_j \right] \tag{1}$$

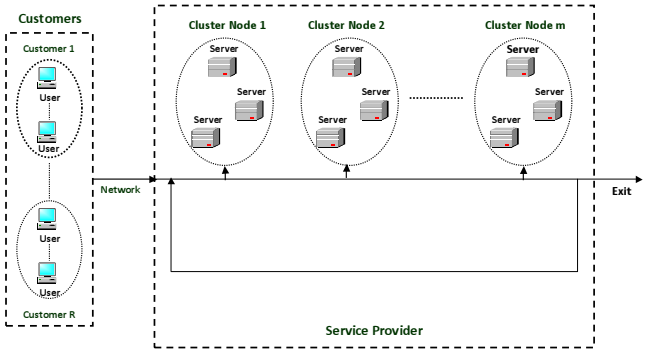


**Figure 1. Customer Service Request Jobs in SLA-based Cluster Computing**

subject to SLA constraints, where $d_j$ represents the number of servers required in cluster node $j$ and thus its value is 1, 2, $\cdots$ , or $N_j$, each server associated with cost $c_j$. For presentation simplicity, we only consider $R = 2$ in the paper. Performance and a service fee are the two most important components for a variety of SLAs in high performance computing such as cluster and grid computing to support parallel computing for multiple customer service applications. As discussed in Section 1, we only consider percentile response time and cluster utilization for the QoS performance requirements and a fee in this paper.

Denote by $f_{T^{(r)}}(t)$ and $F_{T^{(r)}}(t)$ the probability and cumulative distribution functions of response time $T^{(r)}$ with priority class $r$ (i.e., pdf and CDF), respectively. Denote by $T_D^{(r)}$ the desired target response time that a priority class $r$ customer agrees upon with its service provider based on a fee paid by the customer. Thus, the percentile response time is quantified in the following expression:

$$F_{T^{(r)}}(t) = \int_0^{T_D^{(r)}} f_{T^{(r)}}(t)\, dt \geq \gamma^{(r)}\% \qquad (2)$$

That is, $\gamma^{(r)}\%$ of the time a customer will receive its service in less than $T_D^{(r)}$.

As discussed in Section 1, we consider cluster utilization and the percentile of response time as the SLA performance metrics in the research. The cluster utilization is the percentage of the time that the cluster node is utilized. Its formulation will be given in Section 3.1. The priority-type resource allocation problem can be formulated as the following integer optimization problem.

**The Priority-type Resource Provisioning Problem in Cluster Computing:**

Find integers $d_j$ ($0 \leq d_j \leq N_j$; $j = 1, 2, \cdots, m$) in the $m$-dimensional provisioning problem (1) under the constraints of $I \leq C^D$, the percentile response time as expressed by $F_{T^{(r)}}(T_D^{(r)}) \geq \gamma^{(r)}\%$, and the cluster utilization satisfying $\rho_j \leq \zeta_j\%$ and $\rho^{overall} \leq \zeta\%$, respectively, where $C^D$ is a fee negotiated and agreed upon between a customer and the service provider, $\rho_j$ is the average cluster utilization of node $j$, and $\rho^{overall}$ is the average cluster utilization of all the cluster nodes within the service provider. Parameters $\zeta_j$ and $\zeta$ are pre-defined values in the SLA ($j = 1, 2, \cdots, m$).

## 3 The QoS-guaranteed Approach for Solving The Priority-type Resource Allocation Problem

We model the priority-type service model depicted in Figure 1 as a queueing network presenting the path that service request jobs have to follow through the cluster nodes' resources owned by the service provider. The queueing network is depicted in Figure 2. We refer to the queueing network model as a priority-type service request job model since it describes the computing resources used to provide computing services to respond a priority-type customer's service jobs.

As shown in Figure 1, the priority-type service request job model consists of a single infinite server, and $m$ service provider's stations or simply called nodes. (Without any confusion station and node are alternatively used in the paper.) These nodes are numbered sequentially from 1 to $m$. After a customer service job exits from the single infinite server, it will continue to be served at all $m$ cluster nodes. Upon completion of its service at the $m$-th node, a customer service job may exit the queueing network with probability $\alpha$, or may return to the beginning the queueing network with probability $1 - \alpha$, which characterizes the retransmission of a service request job within the service provider depicted in Figure 2.

We model each cluster node $j$ as a single multiple-class priority queue, each providing a service at the rate $\psi_j^{(r)}(d_j)\mu_j^{(r)}$ ($r = 1, 2, \cdots, R$), where $\psi_j^{(r)}(d_j)$ is a known function of $d_j$ and depends on the configuration of servers

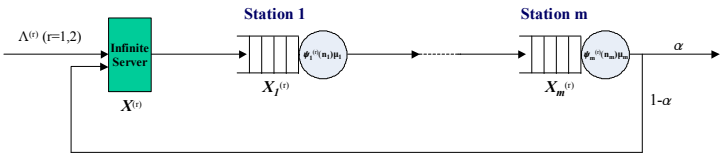


**Figure 2. A Service Request Job Model**

at each cluster node under study with $\psi_j^{(r)}(1) = 1$, and $R$ is the number of priority classes. $\psi_j^{(r)}(d_j)$ is non-decreasing and can be inverted, i.e., $(\psi_j^{(r)})^{-1}$ exists. For instance, suppose that a cluster represents a group of CPUs for the service model as discussed in [16]. Then, $\psi_j^{(r)}(d_j)$ can be seen as a CPU scaling factor for the number of CPUs from 1 to $d_j$. According to [2], $\psi_j^{(r)}(d_j) = (\varepsilon_j^{(r)})^{\log_2 d_j}$, where $\varepsilon_j^{(r)}$ is a basic scaling factor from 1 CPU to 2. So, $(\psi_j^{(r)})^{-1}(d_j) = (\varepsilon_j^{(r)})^{-\log_2 d_j}$. Furthermore, let $\Lambda^{(r)}$ be the external arrival rate to the infinite server due to customer service jobs, and let $\lambda^{(r)}$ and $\lambda_j^{(r)}$ be the effective arrival rates to the infinite server and station $j$, $j = 1, 2, \cdots, m$. We assume that all service times are exponentially distributed and the external arrival to the infinite server occurs in a Poisson fashion.

The infinite server represents the total propagation delay from the first cluster node through the $m$-th cluster node. The first station in Figure 2 models the architecture and elements (i.e., servers) of the first cluster node in Figure 1. The $j$-th station in Figure 2) ($j = 2, 3, \cdots, m$) models the architecture and elements of the $j$-th cluster node in Figure 1.

We have the traffic equations: $\lambda^{(r)} = \Lambda^{(r)} + (1-\alpha)\lambda_m^{(r)}$ and $\lambda_j^{(r)} = \lambda^{(r)}$ that implies $\lambda_j^{(r)} = \lambda^{(r)} = \frac{\Lambda^{(r)}}{\alpha}$, and the utilization of each station is $\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(d_j)\mu_j^{(r)}} = \frac{\Lambda^{(r)}}{\alpha\mu_j^{(r)}\psi^{(r)}(d_j)}$ ($j = 1, 2, \cdots, m$). Note that the infinity server has the same effective arrival rate as node $j$. Denote by $p(t)$ and $p_j(t, \psi^{(r)}(d_j)\mu_j^{(r)})$ the pdfs of response time at the infinity server and node $j$, and $L_{X^{(r)}}(s)$ and $L_{X_j^{(r)}}(s, \psi^{(r)}(d_j)\mu_j^{(r)})$ its corresponding Laplace transform at the infinite server and node $j$ respectively, where $X^{(r)}$ is the service time at the infinite server, and $X_j^{(r)}$ is the time elapsed from the moment a service request job arriving at node $j$ to the moment it departs from the node.

### 3.1 The QoS-guaranteed Algorithm for The Priority-type Resource Allocation Problem

Our approach for solving the priority-type resource allocation problem is to first compute performance metrics: percentile response time and cluster utilization, and then develop an QoS-guaranteed algorithm to solve a constrained optimization that is formulated from the resource allocation problem. In order to calculate the percentile response time, we need to derive the Laplace-Stieltjes transforms (LST) of the probability distribution of the response time.

Let $T^{(r)}(k)$ be the response time of $k$-th visit at the infinite server, the first node, the second node, ..., and $m$-th node for class $r$ customer jobs. Then, $T^{(r)}(k)$ is considered as the sum of the response time of the k-th pass at the infinite server plus the response time of the k-th pass at all the $m$ stations for class $r$ customer jobs:

$$T^{(r)}(k) = X^{(r)} + X_1^{(r)} + X_2^{(r)} + \cdots + X_m^{(r)}$$

where each cluster node is assumed to be independent of each other for class $r$ customer jobs. That is, the waiting time of a service request job at a station or a node is independent of its waiting times at other stations or nodes. Thus, the total response time of a service request for class $r$ customer jobs is

$$T^{(r)} = \sum_{k=1}^{\infty} p(k) T^{(r)}(k)$$

where $p(k)$ is the steady state probability that a request will circulate $k$ times at the infinite server and the $j$-th station through the computing system. $p(k)$ is determined by

$$p(k) = \alpha(1-\alpha)^{k-1}$$

Therefore, the LST of the response time $T^{(r)}$ for class $r$ customer service jobs is:

$$L_{T^{(r)}}(s) = \frac{\alpha L_{X^{(r)}}(s) \Pi_{j=1}^{m} L_{X_j^{(r)}}(s, \psi^{(r)}(d_j)\mu_j^{(r)})}{1 - (1-\alpha) L_{X^{(r)}}(s) \Pi_{j=1}^{m} L_{X_j^{(r)}}(s, \psi^{(r)}(d_j)\mu_j^{(r)})} \quad (3)$$

where $L_{X^{(r)}}(s)$ and $L_{X_j^{(r)}}(s, \psi^{(r)}(d_j)\mu_j^{(r)})$ ($j = 1, 2, \cdots, m$) are the LST of the response time $X^{(r)}$ and the response time $X_j^{(r)}$.

We then compute the probability distribution $f_{T^{(r)}}(t)$ and the cumulative distribution $F_{T^{(r)}}(t)$ of the response time $T^{(r)}$ by inverting $L_{T^{(r)}}(s)$ and $L_{T^{(r)}}(s)/s$ respectively, that is,

$$f_{T^{(r)}}(t) = L^{-1}\left(L_{T^{(r)}}(s)\right) \text{ and } F_{T^{(r)}}(t) = L^{-1}\left(\frac{L_{T^{(r)}}(s)}{s}\right) \quad (4)$$

Notice that $f_{T^{(r)}}(t)$ and $F_{T^{(r)}}(t)$ are usually nonlinear functions of $t$ and $d_j$. Thus, the priority-type resource allocation problem is an $m+R$-dimensional linear optimization problem subject to nonlinear constraints. It is much more difficult to solve this problem compared to the one for single-class service jobs in Xiong and Suh [20]. However, the complexity of the problem can be significantly reduced by postulating that the utilization of each node in Figure 2 should be the same for all nodes. That is, we find the optimum value of $d_1, \cdots, d_m$ such that

$$\rho_1^{(r)} = \cdots = \rho_m^{(r)} \stackrel{def}{=} \hat{a}^{(r)}$$

where $\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(d_j)\mu_j^{(r)}}$ is the average cluster utilization of the $j$-th node ($j = 1, 2, \cdots, m$). This is called *the balanced condition*. (We note that in production lines, it is commonly assumed that the service stations are balanced whose further justification can be found in Xiong [19]). Under this condition, it follows from (3) that

$$L_{T^{(r)}}(s) = \frac{\alpha L_{X^{(r)}}(s)\left[L_{X_j^{(r)}}(s, \psi^{(r)}(d_j)\mu_j^{(r)})\right]^m}{1-(1-\alpha)\left[L_{X_j^{(r)}}(s, \psi^{(r)}(d_j)\mu_j^{(r)})\right]^m} \quad (5)$$

Note that either a high-priority or a low priority customer is only served at time. Thus, $\psi^{(1)}(d_j)\mu_j^{(1)}$ should be considered as the same as $\psi^{(2)}(d_j)\mu_j^{(2)}$, simply referred as to $\psi(d_j)\mu_j$ where no priority class is explicitly stated. Due to the preemptive-resume priority, the waiting time for the high-priority jobs (class 1) is the same as in the single class FIFO $M/M/1$. Thus, $L_{X^{(r)}}(s)$ is the LST of the service time $D^{(r)}$ given by

$$L_{X^{(r)}}(s) = \frac{\lambda^{(r)}}{s+\lambda^{(r)}} \quad (6)$$

and

$$L_{X_j^{(1)}}(s) = \frac{\psi^{(1)}(d_j)\mu_j(1-\rho_j^{(1)})}{s+\psi^{(1)}(d_j)\mu_j(1-\rho_j^{(1)})} \quad (7)$$

and $L_{X_j^{(2)}}(s)$ is the LST of the low-priority response time given by

$$L_{X_j^{(2)}}(s) = \frac{(1-\rho_j)\delta_j^{(1)}}{1-\rho_j\,\delta_j^{(1)}} \quad (8)$$

due to (25) in Xiong and Perros [19], where $\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(d_j)\mu_j}$, $\rho_j = \frac{\lambda_j^{(1)}+\lambda_j^{(2)}}{\psi^{(r)}(d_j)\mu_j}$, $\delta_j^{(1)} = \frac{\eta_j - \sqrt{\eta_j^2 - 4\,\psi^{(1)}(d_j)\lambda_j^{(1)}\,\mu_j^{(1)}}}{2\,\psi^{(1)}(d_j)\lambda_j^{(1)}}$, and $\eta_j = s+\lambda_j^{(1)}+\psi^{(1)}(d_j)\mu_j^{(1)}$ for $j = 1, 2, \cdots, m$. Simliar to Xiong [19], we can derive the percentile response time for priority $r$ customer services with $r > 2$.

Furthermore, the cluster utilization at cluster node $r$ is given by

$$\begin{aligned}\rho_j &= \sum_{r=1}^{R} \frac{\lambda_j^{(r)}}{\psi^{(r)}(d_j)\mu_j^{(r)}} = \frac{\sum_{r=1}^{R}\lambda_j^{(r)}}{\psi(d_j)\mu_j} = \frac{\sum_{r=1}^{R}\Lambda^{(r)}}{\alpha\psi(d_j)\mu_j} \\ &= \frac{\Lambda}{\alpha\psi(d_j)\mu_j} \stackrel{def}{=} \hat{a}\end{aligned}$$

where $\Lambda = \sum_{r=1}^{R}\Lambda^{(r)}$.

Furthermore, the priority-type resource allocation problem is equivalent to the following $1+R$ dimensional mixed optimization one:

$$\min_{\hat{a},\, e^{(r)};\, r=1,\cdots,R} \mathcal{I}(\hat{a}, e^{(1)}, \cdots, e^{(R)}) \quad (9)$$

subject to the requirements of the percentile response time and the cluster utilization, where $\mathcal{I}(\hat{a}, e^{(1)}, \cdots, e^{(R)})$ is equal to

$$\sum_{j=1}^{m} c_j \psi^{-1}\left(\frac{\Lambda}{\alpha\hat{a}\,\mu_j}\right) + \sum_{r=1}^{R} e^{(r)} c_e^{(r)}$$

That is, we reduce the dimensionality of the priority-type resource allocation problem from $m+R$ to $1+R$.

Moreover, the overall cluster utilization owned by the service provider is

$$\rho^{overall}(\hat{a}) = \frac{\hat{a}^m}{1-(1-\alpha)\hat{a}^m} \quad (10)$$

As presented in the priority-type resource allocation problem, the constraint of cluster utilization at each node is expressed as $\rho_j(\hat{a}_j) = \sum_{r=1}^{R}\rho^{(r)}{}_j(\hat{a}_j) \le \zeta_j\%$, and the constraint of the average cluster utilization of all the cluster nodes within the service provider is given by $\rho^{overall}(\hat{a}^u) = \sum_{r=1}^{R}\rho^{(r)}{}_{overall}(\hat{a}) \le \zeta\%$. To ensure the cluster utilization guarantees, we require that $\hat{a}_j = \hat{a} \le \zeta_j\%$ and $\frac{\hat{a}^m}{1-(1-\alpha)\hat{a}^m} \le \zeta\%$. This implies that

$$\hat{a} \le \min\left\{\zeta_1\%, \cdots, \zeta_m\%, \sqrt[m]{\frac{\zeta\%}{1+(1-\alpha)\zeta\%}}\right\} \quad (11)$$

**The Algorithm for Ensuring QoS-guaranteed Services in Priority-type Cluster Computing:**

a. Find $\hat{a}^{(r)}$ in the following minimization problem of a percentile response time and its corresponding optimum values of $d_j^{(r)}$ where $d_j^{(r)}$ are integers:

$$\hat{a}_{min}^{(r)} \leftarrow \arg\min_{\hat{a}^{(r)}} F_{T^{(r)}}(t)|_{t=T^D}$$

subject to the constraint: $F_{T^{(r)}}(t)|_{t=T^D} \ge \gamma^{(r)}\%$ at $\hat{a}^{(r)} = \hat{a}_{min}^{(r)}$, where $F_{T^{(r)}}(t)$ is given by (4). Then,

the optimum values of $d_j^{(r)}$ for the percentile response time guarantee are given by

$$d_j^{(r)} = (\psi^{(r)})^{-1}\left(\frac{\lambda_j^{(r)}}{\hat{a}_{min}^{(r)}\,\mu_j^{(r)}}\right)$$

for $j = 1, 2, \cdots, m$ and $r = 1, 2, \cdots, R$.

b. Calculate $\hat{a}$ given in (11) to ensure the guarantees of cluster node utilization. Their maximal values $a_j^{(u)}$ for stations 1, 2, and 3 are computed by

$$a_j^{(u)} = \frac{\sum_{r=1}^{R}\lambda_j^{(r)}}{\mu_j}\max\left\{(\zeta_j\%)^{-1},\ \sqrt[m]{\frac{1+(1-\alpha)\zeta\%}{\zeta\%}}\right\}$$

Thus, its corresponding optimum values of $d_j^{(u)}$ are equal to $d_j^{(u)} = \psi^{-1}\left(a_j^{(u)}\right)$ for $j = 1, 2, \cdots, m$.

c. Calculate the maximum values $d_j^M$ such that

$$d_j^M = \max\{d_j^{(1)}, \cdots, d_j^{(R)}, d_j^{(u)}\}$$

and then we choose the optimum values of $d_j$ as $d_j^M$, i.e., $d_j = d_j^M$ where $j = 1, 2, \cdots, m$.

d. Let $e^{(r)} = e_{min}^{(r)}$. Check if $0 \le d_j \le N_j$, $0 \le e^{(r)} < 1$, and $I \le C^D$ are satisfied, where $j = 1, 2, \cdots, m$ and $r = 1, 2$. If yes, the obtained $d_j$ is the optimum number of servers required at each cluster node. That is, the service provider should allocate at least $d_j$ servers at each cluster node to ensure the SLA guarantee for multiple customer service jobs. Otherwise, the priority-type resource allocation problem subject to the SLA cannot be solved. In this case, the service provider will inform the customer "We need to re-negotiate the SLA."

Denote by $T_1$ the run-time for the inversion of the LST and $T_2$ the time to find $\hat{a}^{(1)}$ except the time to invert the LST of the response time, respectively. As seen before, the second one is an one-dimensional minimization problem. Thus, $T_2$ is relatively smaller than $T_1$. Furthermore, the total run-time for the QoS-guaranteed algorithm is $O(T_1 + T_2 + m)$.

As discussed before, the total run-time for the QoS-guaranteed algorithm is mainly determined by $O(T_1)$, which depends on the number of function evaluations required for each value of $t$ that is varied in each numerical approximation method for the inversion of a Laplace transform. In our numerical experiments in Section 5, it took a couple of minutes to complete the evaluation.

Moreover, we must point it out that if we require that each node has the same pre-defined $\zeta_j$ in the QoS-guaranteed algorithm, then the constraints of $\rho_j(\hat{a}_j) \le \zeta_j\%$ ($j = 1, 2, \cdots, m$) reduce to the only one constraint: $\rho_1(\hat{a}_1) \le \zeta_1\%$, due to the above proposition.

## 4 Related Work

Job scheduling has a long history. It has been extensively studied. In particular, almost all desktop and laptop computers sold today use dual-core and quad-core chips. Servers and special purpose processors have even much more cores. The largest shift toward parallel computing has now occurred. Thus, the resource allocation questions in the paper plays an important role in not only parallel computation but also other areas. Many real-world problems can be modeled as scheduling problems. For example, the relationship between jobs and computing resources is similar to the one between the following pairs: students and teachers, patients and doctors, as well as ships and docks. Only a few scheduling problems have been shown to be tractable, that is, they are solvable in polynomial time. For the remaining ones, the only way to secure optimal solutions is usually by enumerative methods, requiring exponential time. The scheduling problem becomes even more difficult when a priority-type discipline is imported due to the consideration of multiple customers.

In the research, we study a job as a stream of multiple customer service requests in priority-type cluster computing systems (see McWherter et al. [10]). Liu et al. [7] has considered the problem of multiple heterogeneous resource allocation for a single job. It is an one-to-many matching problem under the constraints of application specific global aggregations, for example, total memory sizes and processor capacities.

In the literature, one usually considers the *average or mean* response time of a job stream in a queue with priorities as a performance metric (see Martin and Nilsson [8]). For example, Miller [12], Mitrani and King [13], and Ngo and Lee [14]. Miller [12], and Ngo and Lee [14] obtained the *mean* response time for each priority class by partitioning into blocks and "super-blocks," based on the number of customers in a queue. This is because an average metric value is relatively easy to calculate. However, customer is more inclined to request a statistical bound on its response time than an average response time (see Martin and Nilsson [8]). A statistical bound on its response time is a percentile response time. Hence, we use it as a performance metric in the research. As shown in the paper, we show that the calculation of the percentile response time is equivalent to finding *the distribution of response time for multiple-priority customer services*.

Many existing studies have been done for resource al-

location in a variety of computing systems subject to the constraints of a variety of QoS metrics such as response time, cluster utilization, or packet loss rate (e.g., see He et al. [5] and Menasce and Casalicchio [11]). He et al. [5] has studied optimizing static workload management for multi-cluster computing systems based on average response time and mean miss rate .

## 5 Experimental Examples

We will use the above QoS-guaranteed algorithm to solve the priority-type resource allocation problem subject to an SLA where QoS metrics include priority-type percentile response time and cluster utilization. As seen above, the accuracy of the QoS-guaranteed algorithm relies on the computation of the inverse Laplace transform of $L_{T^{(r)}}(s)$. As is well-known, the numerical computation of an inverse Laplace transform is an ill-posed problem. Thus, no single method works for any inverse Laplace transform problem (see Graf [4]). This is because in this case there is a singular point that significantly affects the numerical computation of an inverse Laplace transform. Many researchers have developed numerical solutions of inverting a Laplace transform for the past a few decades as described in Graf [4]. In the paper, we employed several different numerical methods for inverting a given $L_T(s)$. If two or more methods can reach consistent solutions, then we are confident that the obtained numerical solutions for the inverse Laplace transform are correct (see Gaver [3], Graf [4], Piessens [15], and Stehfest [17]).

We implement the proposed algorithm. That is, we study the priority-type service request job model shown in Figure 2. Let us still consider a four-station model. The values of parameters $c_j$, $N_j$, $\gamma^{(r)}$, $\zeta_j$, and $\zeta$ are given in Table 1 for $j$=1, 2, 3, and 4; $r = 1$ and 2.

Let us also select $\alpha = 0.67$, $C^D = 600$, $T_D^{(1)} = 0.078$, $T_D^{(2)} = 0.08$, $\Lambda^{(1)}$ =75, $\Lambda^{(2)}$ =25, $\mu_1^{(1)}$ = 58, $\mu_2^{(1)}$ = 28, $\mu_3^{(1)}$ = 35, $\mu_4^{(1)}$ = 16, $\mu_1^{(2)}$ = 9, $\mu_2^{(2)}$ = 19, $\mu_3^{(2)}$ = 11, and $\mu_4^{(2)}$ = 5. Also, let $\psi^{(1)}(d_j) = 1.45^{\log_2 d_j}$ and $\psi^{(2)}(d_j) = 1.56^{\log_2 d_j}$ for $j = 1, 2, 3, 4$. Then, $\lambda^{(1)} = \Lambda^{(1)}/\alpha = 111.94$ and $\lambda^{(2)} = \Lambda^{(2)}/\alpha = 37.31$.

According to our algorithm in Section 3, we calculate the optimum numbers of $d_1$, $d_2$, $d_3$, and $d_4$ by using the following steps.

We first start by solving for $\hat{a}^{(r)}$ in the Step a of the proposed QoS guaranteed algorithm. That is, we need to find the values of $\hat{a}^{(r)}$ to minimize $F(t)|_{t=T_D^{(r)}}$ such that $F(t)|_{t=T_D^{(r)}} = F(T_D^{(r)}) \geq 0.958$, where $F(T_D^{(r)})$ is computed by (5) and $L(f_{X_j^{(r)}}(t))$ is the LST of $f_{X_j^{(r)}}(t)$ for $j = 1, 2, 3, 4$ given in (7) and (8). Thus, we get $\hat{a}^{(1)} = 0.855$, $\hat{a}^{(2)} = 0.91$, Therefore, $d_1^{(1)} = 5$, $d_2^{(1)} = 18$, $d_3^{(1)} = 12$, $d_4^{(1)} = 51$, $d_1^{(2)} = 11$, $d_2^{(2)} = 4$, $d_3^{(2)} = 8$, and $d_4^{(2)} = 27$.

Then, we use Step b of the above algorithm to compute:

$$a_1^{(3)} = \max\{2.2706, 2.3387\} = 2.3387,$$
$$a_2^{(3)} = \max\{4.3933, 4.8444\} = 4.8444,$$
$$a_3^{(3)} = \max\{3.3666, 3.8755\} = 3.8755,$$
$$a_4^{(3)} = \max\{7.3645, 8.4778\} = 8.4778.$$

Thus, $d_1^{(3)} = 5$, $d_2^{(3)} = 19$, $d_3^{(3)} = 13$, and $d_4^{(3)} = 54$.

By using Step c,we get $d_1^M = 11$, $d_2^M = 19$, $d_3^M = 13$, and $d_4^M = 54$. We further choose $d_j = d_j^M$, and verify that $I = \sum_{j=1}^{3} c_j d_j = 573.1405 < C^D$. This means that the optimum values are $d_1 = 11$, $d_2 = 19$, $d_3 = 13$, and $d_4 = 54$, which ensures QoS-guaranteed services for multiple customers.

Extensive numerical results point to the fact that the proposed method provides an efficient way to calculate computing resources required for the SLA assurance.

## 6 Conclusions and Future Work

Cluster computing has been an ideal parallel computing paradigm used in solving large-scale scientific problems. In an effort to maximize a service provider's profit, it is commonplace and important to prioritize customer services in favor of those who are willing to pay higher fees. We have proposed an approach for priority-type resource allocation in such a cluster computing environment, whereby we minimize the total cost of computing resources allocated to multiple customers so that a given set of QoS requirements including percentile of the response time and cluster utilization is satisfied. We have further formulated the priority-type resource allocation problem as an optimization problem subject to QoS constraints and develop the algorithm to solve it, which is used to find the minimum values of computing resources for the multiple customer service guarantee through an illustrated example.

Moreover, our proposed QoS-guaranteed algorithm can be extended to solve resource allocation for a multiple customer service model whose cluster nodes are arbitrarily linked as long as the defined link can be quantified. In addition, we only considered a percentile of response time and cluster utilization in the SLA. Other metrics such as security, availability, vulnerability, and reliability will be investigated in our future study.

## Acknowledgement

Xiong's work was supported in part by the U.S. National Science Foundation (NSF) under the grant number: 1065665. The contents of this paper do no necessarily reflect the position or the policies of the U.S. Government.

**Table 1. The Values of Parameters $c_j$, $N_j$, $\gamma^{(r)}$, $\zeta_j$, and $\zeta$**

| $c_1$ | $c_2$ | $c_3$ | $c_4$ | $N_1$ | $N_2$ | $N_3$ | $N_4$ | $\gamma^{(1)}$ | $\gamma^{(2)}$ | $\zeta_1$ | $\zeta_2$ | $\zeta_3$ | $\zeta_4$ | $\zeta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 9 | 3 | 5 | 15 | 50 | 50 | 100 | 95.8 | 95.8 | 0.85 | 0.91 | 0.95 | 0.95 | 0.69 |

## References


[1] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel. Scalable content-aware request distribution in cluster-based network servers. In *Proceedings of USENIX00 Technical Conference*, 2000.

[2] J. Chang. Processor performance: Update 1. In http://SQL-Server-Performance.com.

[3] D. Gaver, M. Handley, J. Padhye, and J. Widmer. Observing stochastic processes, and approximate transform inversion. *Operations Research*, 14(3), 1966.

[4] U. Graf. *Applied Laplace Transforms and z-Transforms for Scientists and Engineers*. Birkhauser Verlag, 2004.

[5] L. He, S. Jarvis, D. Spooner, and G. Nudd. Optimising static workload allocation in multiclusters. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[6] T. Heath, B. Diniz, E. Carrera, W. Meira, and R. Bianchini. Self-configuring heterogeneous server clusters. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power*, 2003.

[7] C. Liu, L. Yang, I. Foster, and D. Angulo. Design and evaluation of a resource selection framework for grid applications. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing HPDC-11 2002 (HPDC02)*, 2002.

[8] J. Martin and A. Nilsson. On service level agreements for IP networks. In *Proceedings of the IEEE INFOCOM*, June 2002.

[9] INTERNAP. The INTERNAP route optimization solution: executive summary. In `http://www.internap.com/learning/whitepapers`.

[10] D. McWherter, B. Schroeder, N. Ailamaki, and M. Harchol-Balter. Priority mechanisms for OLTP and transactional web applications. In *Proceedings of the 20th International Conference on Data Engineering (ICDE)*, April 2004.

[11] D. Menasce and E. Casalicchio. A framework for resource allocation in grid computing. In *Proceedings of the 12th IEEE MASCOTS*, pages 259–267. IEEE, October 2004.

[12] D. Miller. *Steady-state algorithm analysis of $M/M/c$ two priority queues with heterogeneous servers. In Applied probability - Computer Science, The Interface, volume II, R. L. Disney and T. J. Ott, editors*. Birkhauser, 1992.

[13] J. Mitrani and P. King. Multiprocessor systems with preemptive priorities. *Performance Evaluation*, 1(2):118–125, 1980.

[14] B. Ngo and H. Lee. Analysis of a pre-emptive priority $m/m/c$ model with two types of customers and restriction. *Electronics Letters*, 26:1190–1192, 1990.

[15] R. Piessens. Gaussian quadrature formulas for the numerical integration of Bromwich's integral and the inversion of the Laplace transform. *Journal of Engineering Mathematics*, 5(1), 1971.

[16] M. Shin, S. Chong, and I. Rhee. Dual-resource TCP/AQM for processing-constrained networks. In *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM)*, April 2006.

[17] H. Stehfest. Algorithm 386, numerical inversion of Laplace transforms. *Communcations of the ACM*, 13(1), January 1970.

[18] X. Xiao and L. Ni. Internet QoS: a big picture. *IEEE Network*, 1999.

[19] K. Xiong. Multiple priority customer service guarantees in cluster computing. In *Proceedings of the 23rd IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2009.

[20] K. Xiong and S. Suh. *Resource Provsioning in SLA-based Cluster Computing, The Job Scheduling Strategies for Parallel Processing, Lecture Notes, LNCS6253, editored by E. Frachtenberg and U. Schwiegelshohn*. Springer, 2010.