Resource Provisioning in SLA-Based Cluster Computing

Kaiqi Xiong and Sang Suh

Department of Computer Science, Texas A&M University,
Commerce, TX 75429, USA
kaiqi_xiong@tamu-commerce.edu

Abstract. Cluster computing is excellent for parallel computation. It has become increasingly popular. In cluster computing, a service level agreement (SLA) is a set of quality of services (QoS) and a fee agreed between a customer and an application service provider. It plays an important role in an e-business application. An application service provider uses a set of cluster computing resources to support e-business applications subject to an SLA. In this paper, the QoS includes percentile response time and cluster utilization. We present an approach for resource provisioning in such an environment that minimizes the total cost of cluster computing resources used by an application service provider for an e-business application that often requires parallel computation for high service performance, availability, and reliability while satisfying a QoS and a fee negotiated between a customer and the application service provider. Simulation experiments demonstrate the applicability of the approach.

Keywords: Cluster computing, scheduling theory, resource provisioning, service level agreement, and percentile response time.

1 Introduction

In computer science, scheduling theory is concerned with the optimal allocation of scarce resources such as servers, processors and network links to computer service activities over time, with the objective of optimizing one or several computer performance measures (e.g., see Levner [13]). Cluster computing is excellent for parallel computation. It has become increasingly popular. The management of computing service resources is fundamental to cluster computing. The increasing pervasiveness of network connectivity and the proliferation of on demand e-business applications and services in public domains, corporate networks, as well as home environments give rise to the need for the design of appropriate service management solutions in cluster computing. Accurately predicting e-business application and scientific computation performance based on service statistics and a customer's perceived quality allows an application service provider (simply called a service provider) not only to assure quality of services but also to avoid over provisioning to meet a service level agreement (SLA).

E. Frachtenberg and U. Schwiegelshohn (Eds.): JSSPP 2010, LNCS 6253, pp. 1–15, 2010.
© Springer-Verlag Berlin Heidelberg 2010

Job scheduling has been a fruitful area of research for many decades. It involves answers to the following questions:

1. How are jobs assigned to computing resources, such as processors and machines?
2. What orders should we use to process jobs in a single computing resource?
3. How to allocate sufficient computing resources to match the requirements of submitted jobs in terms of ensuring QoS guarantees?

The above three job scheduling questions are usually called the parallel job scheduling problem, the job sequencing problem, and the resource provisioning problem (also called the resource matching problem), respectively. While the job sequencing problem is relatively simple, the parallel job scheduling problem and the resource allocation problem are difficult to solve. Generally speaking, both are NP-hard (see Du and Leung [7]). In this paper, we focus on the resource provisioning problem that has been extensively researched over the years (see Feitelson et al. [8] and Yom-Tov and Aridor [19]). In particular, we consider the problem for avoiding the over-provisioning of computing resources. With over-provisioning, computing resources are allocated more than service request jobs actually need due to the over-determined requirements of service request jobs, which should not occur as desired by a service provider for high profits.

Yom-Tov and Aridor [19] gave an example of two machines to explain how badly over-provisioning affects machine utilization. However, if allocated computing resources fall below a certain level or are insufficient, service request jobs cannot complete to meet customer service requirements. Hence, the resource provisioning problem plays a key role in job scheduling. It is an extremely important but very challenging problem as shown in Liu et al. [12], Naik et al [16], and Yom-Tov and Aridor [19].

In this paper, we consider a resource provisioning problem in SLA-based cluster computing where a service provider processes e-business application request jobs for business customers subject to an SLA. Such request jobs often require parallel computation for high service performance, availability, and reliability. As shown in Figure 1, a customer represents a business 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 for a given fee. A service request job is transmitted to a service provider in a cluster computing system consisting of a group of cluster nodes that are linked together to support parallel computation (e.g., see Aron et al. [1], Heath et al. [10] as well as Xiao and Ni [35]). Upon the completion of a service request job at the service provider, the final result is sent back to the customer. The service provider's cluster nodes have or are a set of computing resources so that they are capable to collaborate each other in parallel for processing the service request job. Such computing resources in each cluster node may include processors and cluster servers/machines as discussed in Shin et al. [17] as well as Xiao and Ni [35]. For presentation purpose, we explicitly think the computing resource of each node as cluster servers. (Note that in this paper "computing resource" or "server" are used alternatively.)

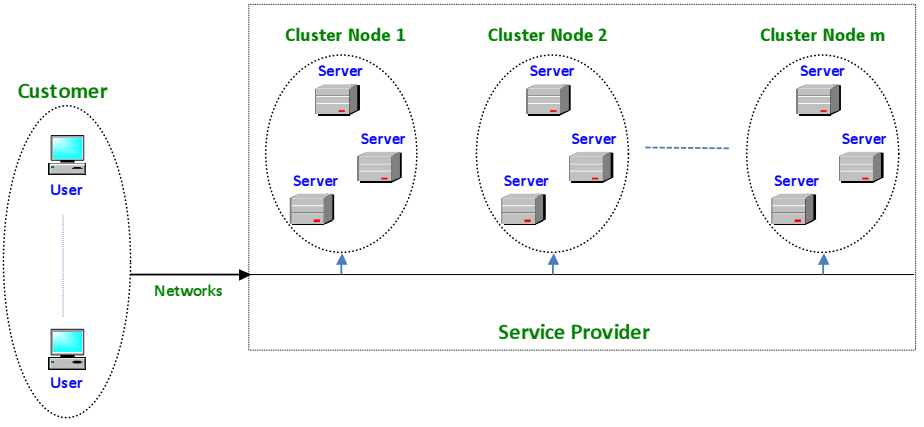

Fig. 1. Customer Service Request Jobs in SLA-based Cluster Computing

The resource provisioning problem is to minimize the overall cost of the service provider's computing resources of each node allocated to the business customer in terms of the number of servers at each cluster node while satisfying an SLA agreement. The SLA is a contract negotiated and agreed between the customer and the application provider. It defines the quality of service (QoS) and a fee. In this paper, the QoS metrics include:

1. *Percentile response time*-γ% ($0 \leq \gamma \leq 100$) of the time the response time, i.e., the time to execute a service request job, is less than a pre-defined value;
2. *Cluster utilization*-It is the percentage of the time that the cluster node is utilized.

Both of them are often called SLA performance metrics in the literature. These QoS requirements are typical metrics included in an SLA (e.g., see INTERNAP [25] as well as Martin and Nilsson [28]). As an end user of e-business applications, a customer is in general concerned about response time rather than throughput (for example, in an online business, an buyer often concerns about how soon his/her order will be processed and completed). Hence, we do not include throughput as a metric in this study. Security, reliability and survivability may be included in an SLA as well as described in Jacob [26]. We will discuss them in another paper.

In this paper, we present the resource provisioning problem by minimizing the total cost of each cluster node's computer resources required to ensure a given percentile of the response time and cluster utilization. We formulate the provisioning problem as a constrained optimization problem. By modeling a typical customer service scenario as a queueing network, we first propose an approach to computing the percentile response time of a service request job. We note that the proposed approach can be also applied to a queueing network whose cluster nodes are arbitrarily linked as long as the link can be quantified. Then,

we present an approach to solving the constrained optimization problem by calculating the computing resource of each cluster node required in each service provider's node. To the best of our knowledge, our study is the first attempt to analytically solve the resource provisioning problem under the consideration of *percentile response time and cluster utilization* for cluster computing by using a queueing network method.

The rest of the paper is organized as follows. Related work is presented in Section 2. In Section 3 we formulate the resource provisioning problem with the SLA performance metrics. In Section 4 we model the 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 a customer service request job in the queueing network. We further propose an approach for solving the provisioning problem. Numerical experiments are given in Section 5. We conclude our results in Section 6.

2 Related Work

The job scheduling questions presented in Section 1 have been extensively studied. They play 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 (e.g., see Cook [6], Garey and Johnson [9], and Papadimitriou [15]).

Resource management including resource monitoring as well as resource matching and/or resource provisioning has been researched over many years. Feitelson et al. [8] and Yom-Tov and Aridor [19] have studied resource provisioning for job scheduling in heterogeneous server clusters. Ngubiri and Vlient [14] discussed a processor provisioning problem in multi-cluster systems. Bucur [3], Bucer and Epema [4], and Jones [11] have considered the problem of resource provisioning for Distributed ASCI Supercomputer (DAS). Bucur and Epema [4] proposed and analyzed resource provisioning approaches in different scenarios. Jones [11] focused on scheduling techniques and how they are affected by network characteristics like latencies.

In the paper, we consider a job as a stream of customer service requests in cluster computing. The problem of multiple heterogeneous resources allocated to a single job has been discussed in Liu et al. [12]. It is an one-to-many matching problem under the constraints of application specific global aggregations, for example, total memory sizes and processor capacities.

As we know, in the above literature the authors only considered the average metric value of a job stream as a performance metric. 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. Thus, we use the percentile response time as our performance metric in the paper.

Resource provisioning with the constraints of a variety of QoS metrics such as response time, cluster utilization, or packet loss rate for other computing infrastructures such as a network system have been extensively studied in the literature as well. Bouillet, et al. [20] considered a routing and resource management problem subject to the requirements of aggregate bandwidth from ingress to egress nodes. In [21], Chassot et al. dealt with a communication architecture with guaranteed end-to-end QoS in an IPv6 environment. The end-to-end QoS includes an end-to-end delay (i.e., response time). Chassot et al. only discussed and measured the maximal, minimal and average values of response time. Cao and Zegura [22] considered the bandwidth allocation scheme for an available bit rate service. In Liao and Campbell [27], a mechanism was developed with the capability of delivering capacity provisioning in an efficient manner providing quantitative delay-bounds with differentiated loss across per-aggregate service classes.

3 The Resource Provisioning Problem

In this section, we study the customer service request jobs depicted in Figure 1 where a service request job is transmitted to m cluster nodes within a service provider. For presentation purposes, we assume that each cluster node has only one type of cluster server associated with cost c_j. If they have multiple types of servers, we can decompose each cluster node into several individual sub-nodes so that each one only contains one type of servers with the same cost.

Let N_j be the number of servers at node j ($j = 1, 2, \cdots, m$). Thus, the resource provisioning problem is to minimize the overall cost of the computing resources required while satisfying SLA requirements in cluster computing. That is, the resource provisioning problem is quantified by solving for d_j in the following provisioning problem:

$$I = \min_{d_1,\cdots,d_m} (d_1\, c_1 + \cdots + d_m\, c_m) \tag{1}$$

subject to SLA constraints, where d_j represents the number of servers required in cluster node j and hence its value is 1, 2, $\cdots$, or N_j, each server associated with cost c_j. 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 business applications. In this paper, the SLA constraints include the aforementioned percentile response time and cluster utilization as well as a service fee.

As discussed in Section 1, we consider cluster utilization and the percentile of response time as the SLA performance metrics. The cluster utilization is the percentage of the time that the cluster node is utilized. It will be discussed in detail in Section 4.1. The cluster utilization within a service provider is not

observed by a customer (see Martin and Nilsson [28]). Instead, response time can be directly measured by a customer. It directly reflects service performance as stated in Martin and Nilsson [28], Paxson [30] and Padhye et al. [31].

As described earlier, in the literature, typically the average response time (or an average execution time) is used (e.g., see Martin and Nilsson [28] as well as Menasce and E. Casalicchio [29]). The average response time is heavily influenced by "outliers," which occur in almost all measurements. Therefore, although the average response time is relatively easy to calculate, it may not address the concerns of a customer. Typically, a customer is more inclined to request a statistical bound on its response time than an average response time. For instance, a customer can request that 95% of the time its response time should be less than a desired value. Hence, in this paper we are concerned with the statistical bound on the response time.

The response time is the time it takes for a service request job to be executed on the service provider's cluster nodes and then sent its completed job back to the customer. Let T be a random variable representing the response time, and let $f_T(t)$ and $F_T(t)$ be its probability and cumulative distributions pdf and CDF, respectively. Also, let T^D be the desired target response time that a customer requests and agrees with its service provider based on a fee paid by the customer. The statistical bound on the response time can be expressed by

$$F_T(t)|_{t=T^D} = \int_0^{T^D} f_T(t)\,dt \geq \gamma\% \quad (0 \leq \gamma \leq 100) \tag{2}$$

which is called *percentile response time*. This means that $\gamma\%$ of the time a service request job will be executed in less than T^D.

As an example let us consider an $M/M/1$ queue with arrival rate λ and service rate μ. The service discipline is FIFO. The steady-state probability of the system is $p_0 = 1 - \rho$, and $p_k = (1-\rho)\rho^k$, $k > 0$, where $\rho = \frac{\lambda}{\mu}$ (see Perros [32]). The response time T is exponentially distributed with the parameter $\mu(1-\rho)$, i.e., its probability distribution is given by

$$f_T(t) = \mu(1-\rho)e^{-\mu(1-\rho)t}$$

Using the definition given in (2), we have that

$$F_T(t)|_{t=T^D} = 1 - e^{-\mu(1-\rho)T^D} \geq \gamma\% \tag{3}$$

For example, to ensure that in a 95% (=$\gamma\%$) of time, customer service request jobs can be executed in T^D. It follows from (3) that

$$e^{-\mu(1-\rho)T^D} \leq 5\%$$

which is equivalent to

$$\mu \geq \lambda + \frac{\ln 20}{T^D}$$

Furthermore, the resource provisioning problem can be formulated as the following integer optimization problem.

The Resource Provisioning Problem in SLA-based 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 (2), 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, ρ_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 ζ_j and ζ are pre-defined values in the SLA $(j = 1, 2, \cdots, m)$.

4 The Solution of the Resource Provisioning Problem

In this section, we study a queueing network model that depicts the path that service request jobs have to follow through the cluster nodes' resources owned by the service provider described in Figure 1. The queueing model is shown in Figure 2. We refer to the queueing model as a service request job model since it depicts the computing resources used to provide computing services to respond a customer's service job requests.

The service request job model consists of a single infinite server, and m service provider's stations (or simply called nodes. In the rest of this paper, without any confusion station and node are alternatively used) numbered sequentially from 1 to m as shown in Figure 2. After a customer exits from the single infinite server, it will continue to be served at all m nodes. Upon completion of its service at the m-th node, a customer may exit the queueing network with probability α, 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, shown in Figure 2.

As seen in Figure 1, each cluster node consists of multiple servers that are linked together to support for parallel computations. The servers of each cluster node are commonly, but not always, connected to each other through fast local area networks. Cluster nodes are usually deployed to improve performance and/or availability over that of a single computer, while typically being much

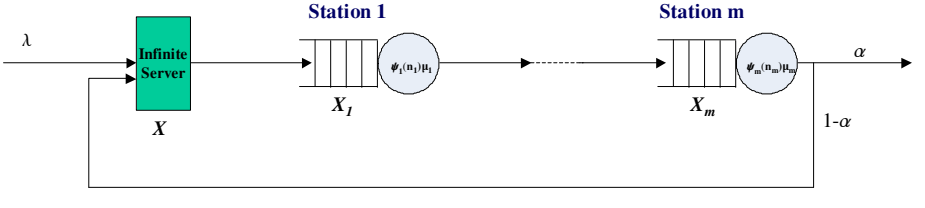

Fig. 2. A Service Request Job Model

more cost-effective than single computers of comparable speed or availability (see Luke [2]). Each cluster node has a group of linked servers to work together closely so that it is treated as a single computer in many respects. Thus, in the following discussion each service provider's cluster node is modeled as a single $G/G/1$ queue with arrival rate λ_j and service rate $\psi(d_j)\mu_j$, where $\psi(d_j)$ is a known function of d_j and depends on the configuration of servers at each node or station. It is non-decreasing and can be inverted, i.e., ψ^{-1} exists. For instance, suppose that a station represents a group of CPUs. Then, $\psi(n)$ can be seen as a CPU scaling factor for the number of CPUs from 1 to n. According to Chang [5], $\psi(n) = \xi^{\log_2 n}$, where ξ is a basic scaling factor from 1 CPU to 2. So, $\psi^{-1}(n) = \xi^{-\log_2 n}$.

Let Λ be the arrival rate generated by a customer as well as λ and λ_j be the effective arrival rates to the infinite server, respectively. 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: $\lambda = \Lambda + (1 - \alpha)\lambda_m$ and $\lambda_j = \lambda$ that implies $\lambda_j = \lambda = \frac{\Lambda}{\alpha}$, and the utilization of each station is $\rho_j = \frac{\lambda_j}{\psi(d_j)\mu_j} = \frac{\Lambda}{\alpha\mu_j\psi(d_j)}$ ($j = 1, 2, \cdots, m$). Note that the infinity server has the same effective arrival rate as node j. Thus, let $p(t)$ and $p_j(t, \psi(d_j)\mu_j)$ be the pdfs of response time at the infinity server and node j (these pdfs can be at least determined by a curve fitting of measurement data as discussed in Zandt [36]), and $L_X(s)$ and $L_{X_j}(s, \psi(d_j)\mu_j)$ its corresponding Laplace transform at the infinite server and node j respectively, where X is the service time at the infinite server, and X_j is the time elapsed from the moment a service request job arriving at node j to the moment it departs from the node.

4.1 An Algorithm for the Resource Provisioning Problem

In order to present our approach for solving the resource provisioning problem, we need to derive the Laplace-Stieltjes transforms (LST) of the probability distribution of the response time.

Let $T(k)$ be the response time of k-th visit at the infinite server, the first node, the second node, ..., and m-th node. Then, $T(k)$ 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:

$$T(k) = X + X_1 + X_2 + \cdots + X_m$$

where we assume that each router is independent of each other. That is, we assume that 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. Then, the total response time of a service request is

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

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

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

Thus, the LST of the response time T is

$$L_T(s) = \sum_{k=1}^{\infty} p(k) L_X^k(s) L_{X_1}^k(s, \psi(d_1)\mu_1) \cdots L_{X_m}^k(s, \psi(d_m)\mu_m)$$

which can be re-written as follows:

$$L_T(s) = \frac{\alpha L_X(s)\, \Pi_{j=1} L_{X_j}(s, \psi(d_j)\mu_j)}{1 - (1-\alpha)\, L_X(s) \Pi_{j=1}^{m} L_{X_j}(s, \psi(d_j)\mu_j)} \tag{4}$$

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

The probability distribution $f_T(t)$ and the cumulative distribution $F_T(t)$ of the response time T can be calculated by inverting $L_T(s)$ and $L_T(s)/s$ respectively, that is,

$$f_T(t) = L^{-1}\left(L_T(s)\right) \quad \text{and} \quad F_T(t) = L^{-1}\left(\frac{L_T(s)}{s}\right) \tag{5}$$

We observe that $f_T(t)$ and $F_T(t)$ are usually nonlinear functions of t and d_j. Hence, the resource provisioning problem is an m-dimensional linear provisioning problem subject to nonlinear constraints. In general, it is not easy to solve this problem. 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

$$\rho_1 = \cdots = \rho_m \stackrel{def}{=} \hat{a}$$

where $\rho_j = \frac{\lambda_j}{\psi(d_j)\mu_j}$ 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 [18]).

We further consider the cluster utilization of the service model within the service provider's node, and derive the following result.

Proposition: The average cluster utilization of all the cluster nodes within the service provider is

$$\rho^{overall}(\hat{a}) = \frac{\hat{a}^m}{1 - (1-\alpha)\hat{a}^m} \tag{6}$$

Proof. From the structure of the queueing network, the average cluster utilization of this SLA-based cluster model within the service provider can be computed by

$$\rho^{overall}(\hat{a}) = \sum_{k=1}^{\infty} p(k) \rho_1(\hat{a}) \cdots \rho_j(\hat{a})$$

where $p(k) = \alpha(1-\alpha)^{k-1}$ and $\rho_j(\hat{a}) = \rho_j$. Due to the balanced condition, we have $\rho_j(\hat{a}) = \hat{a}$, and then easily get (6). The proof is complete.

As presented in the resource provisioning problem, the constraint of cluster utilization at each node: $\rho_j(\hat{a}_j) \leq \zeta_j\%$, and the constraint of the average cluster utilization of all the cluster nodes within the service provider: $\rho^{overall}(\hat{a}^u) \leq \zeta\%$. To ensure the cluster utilization guarantees, we require that $\hat{a}_j = \hat{a} \leq \zeta_j\%$ and $\frac{\hat{a}^m}{1-(1-\alpha)\hat{a}^m} \leq \zeta\%$. This implies that

$$\hat{a} \leq \min\left\{\zeta_1\%, \cdots, \zeta_m\%, \sqrt[m]{\frac{\zeta\%}{1+(1-\alpha)\zeta\%}}\right\} \tag{7}$$

In addition, note that $\frac{\lambda_j}{\psi(d_j)\mu_j} = \hat{a}$. Hence, $\psi(d_j) = \frac{\lambda}{\hat{a}\,\mu_j}$, i.e., $d_j = \psi^{-1}\left(\frac{\lambda}{\hat{a}\,\mu_j}\right)$ for $j = 1, 2, \cdots, m$. This implies $\sum_{j=1}^{m} c_j d_j$ reduces to a function of variable $\hat{a}$. Thus, we have the following algorithm for solving the resource provisioning algorithm.

Algorithm

a. Find $\hat{a}$ in the following minimization problem of a percentile response time and its corresponding optimum values of $d_j^{(1)}$:

$$\hat{a}^{(1)} \leftarrow \arg\min_{\hat{a}} F_T(t)|_{t=T^D}$$

subject to the constraint: $F_T(t)|_{t=T^D} \geq \gamma\%$ at $\hat{a} = \hat{a}^{(1)}$, where $F_T(t)$ is given by (5). Then, the optimum values of $d_j^{(1)}$ for the percentile response time guarantee are given by $d_j^{(1)} = \psi^{-1}\left(\frac{\lambda_j}{\hat{a}^{(1)}\,\mu_j}\right)$ for $j = 1, 2, \cdots, m$.

b. Calculate $\hat{a}$ given in (7) to ensure the guarantees of cluster node utilization. Their maximal values $a_j^{(2)}$ for stations 1, 2, and 3 are computed by

$$a_j^{(2)} = \frac{\lambda_j}{\mu_j}\max\left\{(\zeta_j\%)^{-1}, \sqrt[m]{\frac{1+(1-\alpha)\zeta\%}{\zeta\%}}\right\}$$

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

c. Calculate the maximum values d_j^M such that $d_j^M = \max\{d_j^{(1)}, d_j^{(2)}\}$, and then choose the optimum values of d_j are equal to d_j^M $(j = 1, 2, \cdots, m)$.

d. Check if $0 \leq d_j \leq N_j$ $(j = 1, 2, \cdots, m)$ and $I \leq C^D$ are satisfied. 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. Otherwise, the resource provisioning 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," or both.

Note that if we cannot get a solution for the resource provisioning problem using the above algorithm, then the service provider cannot execute service request jobs in the SLA-based cluster computing due to at least one of the following reasons: (i) the service provider has insufficient computing resources (i.e., μ_j, N_j, or both are too small), (ii) a pre-specific fee is too low (i.e., $I > C^D$), or (iii) at least one cluster node is over-utilized. Using these information, we may detect and debug a service provider's capacity problem, that is, the SLA needs to be re-negotiated.

In this algorithm, the run-time for Steps b, c and d have the same run-time $O(m)$. The efficiency of this algorithm is determined by the run-time for inverting the LST of the response time in Step a, which can be efficiently done as well (see Graf [24]). Let T_1 be the run-time for the inversion of the LST and T_2 be the time to find $\hat{a}^{(1)}$ except the time to invert the LST of the response time. (This is an one-dimensional minimization problem. So, generally speaking, T_2 is relatively smaller than T_1.) Thus, the total run-time for the Algorithm is $O(T_1 + T_2 + m)$.

As we see, the total run-time for the 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, it usually took a couple of minutes to complete the evaluation.

Remarks: In the above algorithm, if we require that each node has the same pre-defined ζ_j, then the constraints of $\rho_j(\hat{a}_j) \leq \zeta_j\%$ $(j = 1, 2, \cdots, m)$ reduce to the only one constraint: $\rho_1(\hat{a}_1) \leq \zeta_1\%$, due to the above proposition.

5 Numerical Experiments

In this section we demonstrate how to apply our algorithm to solve the resource provisioning problem subject to an SLA.

Clearly, our proposed method heavily depends on the computation of the inverse Laplace transform of $L_T(s)$. Many studies have been done in the past a few decades as described in Graf [24]. Since the numerical computation of an inverse Laplace transform is an ill-posed problem, no single method works for any inverse Laplace transform problem (see Graf [24]). This is because in this case there is a singular point that significantly affects the numerical computation of an inverse Laplace transform. Thus, we employed several different numerical methods for inverting a given $L_T(s)$. If two or more methods can reach about the same results, then we are confident that the derived numerical inverse Laplace transform is correct. These numerical methods include the inversion methods using Laguerre functions and Fourier functions in Graf [24], Gaussian quadrature formulas in Piessens [33], and the method by Gaver [23] and Stehfest [34]. The Laguerre method in Graf [24] and the Gaver-Stehfest method in Gaver [23] and Stehfest [34] compute more rapidly but are slightly less accurate compared to the Gaussian quadrature formulas in Piessens [33].

We consider the service request job model shown in Figure 2. For presentation purpose, we only consider a three-station model, i.e., $m = 3$. The values of

Table 1. The Values of c_1, c_2, c_3, C^D, N_1, N_2, N_3, T^D, γ, α, ζ_1, ζ_2, ζ_3, and ζ

c_1	c_2	c_3	c^D	N_1	N_2	N_3	T^D	γ	α	ζ_1	ζ_2	ζ_3	ζ
8	8	3	800	50	80	100	0.08	98	0.8	0.78	0.9	0.92	0.58

parameters c_j, C^D, N_j, $T^D = 0.08$, γ, α, ζ_1, ζ_2, ζ_3, and ζ are given in Table 1 for j=1, 2, 3.

We further choose Λ =200, $\mu_1 = 48$, $\mu_2 = 38$, and $\mu_3 = 25$. Also, let $f_{X_1}(t)$ and $f_{X_3}(t)$ be Erlang-2 distributions with $\nu_1 = \psi(d_1)\mu_1$ and $\nu_3 = \psi(d_3)\mu_3$ for cluster nodes 1 and 3 respectively, $f_{X_2}(t)$ is an Erlang-1 distribution with $\nu_2 = \psi(d_2)\mu_2$, where $\psi(d_j) = 1.5^{\log_2 d_j}$ for $j = 1, 2, 3$. Then, $\lambda = \Lambda/\alpha = 250$.

According to our algorithm in Section 4, we calculate the optimum numbers of d_1, d_2 and d_3 using the following steps.

We first solve for $\hat{a}^{(1)}$ in the Step a of Algorithm. That is, let us find the minimum value of $\hat{a}$ such that $F(t)|_{t=T^D} = F(T^D) \geq 0.98$, where $F(T^D)$ is computed by

$$F(t) = L^{-1}\Big\{\frac{200}{s(s+250)}\frac{\Pi_j^3 L(f_{X_j}(s))}{1-0.2\Pi_j^3 L(f_{X_j}(s))}\Big\}$$

and $L(f_{X_j}(t))$ is the LST of $f_{X_j}(t)$ for $j = 1, 2, 3$. Thus, we get $\hat{a} = 0.85$. Consequently, $d_1^{(1)} = 23$, $d_2^{(1)} = 34$, and $d_3^{(1)} = 68$.

Then, we use Step b of the Algorithm to compute $a_1^{(2)} = \max\{6.6774, 6.4780\}$ $= 6.6774$, $a_2^{(2)} = \max\{7.3099, 8.1828\} = 8.1828$ and $a_3^{(2)} = \max\{10.8696, 12.4379\}$ $= 12.4379$. Thus, $d_1^{(2)} = 26$, $d_2^{(2)} = 37$, and $d_3^{(2)} = 75$.

By using Step c,we get $d_1^M = 26$, $d_2^M = 37$, and $d_3^M = 75$. We further choose $d_j = d_j^M$, and verify that $I = \sum_{j=1}^3 c_j d_j = 729 < C^D$. This means that the optimum values are $d_1 = 26$, $d_2 = 37$ and $d_3 = 75$.

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

6 Conclusions

Cluster computing is excellent for parallel computation. It has become increasingly popular. We have proposed an approach for resource provisioning in a typical SLA-based cluster computing environment, whereby we minimize the total cost of computing resources allocated to a customer so that a given set of SLAs including percentile of the response time and cluster utilization is satisfied.

We have further formulated the resource provisioning problem as an optimization problem subject to SLA constraints for a typical SLA-based cluster computing system, and developed an efficient approach to solving the problem. Finally, we have demonstrated how to use our proposed approach to finding the

minimum values of computing resources required for the customer SLA guarantee by conducting numerical experiments.

Most importantly, we should point out that the proposed approach of this paper provides a framework for addressing and solving this type of resource provisioning problems subject to a given set of SLAs for high-performance computing systems including cluster and grid computing systems. Moreover, this approach can be extended to study a service request job model whose cluster nodes are arbitrarily linked as long as the defined link can be quantified. In this paper, 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 discussed in another paper.

References

1. Aron, M., Sanders, D., Druschel, P., Zwaenepoel, W.: Scalable content-aware request distribution in cluster-based network servers. In: Proceedings of USENIX 2000 Technical Conference (June 2000)
2. Lucke, R.: Buidling Clustered Liux Systems. Prentice-Hall, Englewood Cliffs (2005)
3. Bucur, A.: Performance analysis of processor co-allocation in multicluster systems, PhD Thesis, Delft University of Technology, Delft, The Netherlands (2004)
4. Bucur, A., Epema, D.: Local versus global schedulers with processor co-allocation in multicluster systems. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2002. LNCS, vol. 2537, pp. 184–204. Springer, Heidelberg (2002)
5. Chang, J.: Processor performance: Update 1, `http://SQL-Server-Performance.com`
6. Cook, S.: The complexity of theorem proving procedures. In: Proceedings of the Third Annual ACM Symposium on the Theory of Computing, pp. 151–158. ACM, New York (1971)
7. Du, J., Leung, T.: Complexity of scheduling parallel task systems. SIAM Journal on Discrete Mathematics 2, 473–487 (1989)
8. Feitelson, D., Rudolph, L., Shwiegelshohn, U.: Parallel job scheduling: a status report. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 1–16. Springer, Heidelberg (2005)
9. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
10. Heath, T., Diniz, B., Carrera, E.V., Meira Jr., W., Bianchini, R.: Self-configuring heterogeneous server clusters. In: Proceedings of the Workshop on Compilers and Operating Systems for Low Power (September 2003)
11. Jones, W.: Improving parallel job scheduling performance in multi-clusters through selective job co-allocation. PhD dissertation, Clemson University, Clemson, South Carolina, USA (2005)
12. Liu, C., Yang, L., Foster, I., Angulo, D.: 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 (HPDC 2002), Washington, DC, USA, p. 63. IEEE Computer Society, Los Alamitos (2002)

13. Levner, E.: Multiprocessor Scheduling: Theory and Applications. I-Tech Education and Publishing, Vienna (December 2007)
14. Ngubiri, J., Vliet, M.: Group-wise performance evaluation of processor co-allocation in multi-cluster systems. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2007. LNCS, vol. 4942, pp. 24–36. Springer, Heidelberg (2008)
15. Papadimitriou, C.: Computational Complexity, 1st edn. Addison-Wesley, Reading (1994)
16. Naik, V., Liu, C., Yang, L., Wagner, J.: On-line resource matching in a heterogeneous grid environment. In: Proceedings of the International Symposium on Cluster Computing and the Grid (CCGrid 2005). IEEE Computer Society, Los Alamitos (2005)
17. Shin, M., Chong, S., Rhee, I.: Dual-resource TCP/AQM for processing-constrained networks. In: Proceedings of the IEEE INFOCOM (April 2006)
18. Xiong, K.: Resource Optimization and Security in Distributed Computing. Ph.D. Dissertation, North Carolina State University, USA (December 2007)
19. Yom-Tov, E., Aridor, Y.: A self-optiimized job scheduler for heterogeneous server clusters. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2007. LNCS, vol. 4942, pp. 169–187. Springer, Heidelberg (2008)
20. Bouillet, E., Mitra, D., Ramakrishnan, K.: The structure and management of service level agreements in networks. IEEE Journal on Selected Areas in Communications 20(4), 691–699 (2002)
21. Chassot, C., Garcia, F., Auriol, G., Lozes, A., Lochin, E., Anelli, P.: Performance Analysis for an IP Differentiated Services Network. In: Proceedings of IEEE Internnational Conference on Communication (ICC 2002), pp. 976–980 (2002)
22. Cao, Z., Zegura, E.: Utility max-min: An application-oriented bandwidth allocaton scheme. In: Proceedings of the IEEE INFOCOM (March 1999)
23. Gaver, D., Handley, M., Padhye, J., Widmer, J.: Observing stochastic processes, and approximate transform inversion. Operation Research 14(3) (1966)
24. Graf, U.: Applied Laplace Transforms and z-Transforms for Scientists and Engineers. Birkhauser Verlag, Basel (2004)
25. INTERNAP, The INTERNAP route optimization solution: executive summary, `http://www.internap.com/learning/whitepapers`
26. Jacob, B., et al.: On Demand Operating Environment: Managing the Infrastructure, IBM Redbooks (June 2005)
27. Liao, R., Campbell, A.: Dynamic core provisioning for quantitative differentiated services. IEEE/ACM Transactions on Networking 12(3), 429–442 (2005)
28. Martin, J., Nilsson, A.: On service level agreements for IP networks. In: Proceedings of the IEEE INFOCOM (June 2002)
29. Menasce, D., Casalicchio, E.: A framework for resource allocation in grid computing. In: Proceedings of the MASCOTS (October 2004)
30. Paxson, V.: End-to-end Internet packet dynamics. In: Proceedings of the ACM SIGCOMM (1997)
31. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP throughput: a simple model and its empirical validation. In: Proceedings of the ACM SIGCOMM (2004)
32. Perros, H.: Queueing Network with Blocking, Exact and Approximate Solutions. Oxford University Press, Oxford (1994)

33. Piessens, R.: 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)
34. Stehfest, H.: Algorithm 386, numerical inversion of Laplace transforms. Communcations of the ACM 13(1) (January 1970)
35. Xiao, X., Ni, L.M.: Internet QoS: a big picture. IEEE Network (March/April 1999)
36. Zandt, T.: How to fit a response time distribution, `http://citeseer.ist.psu.edu/552295.html`