2-2006

Allocating Non-Real-Time and Soft Real-Time Jobs in Multiclusters

Ligang He
University of Warwick, liganghe@dcs.warwick.ac.uk

Stephen A. Jarvis
University of Warwick, saj@dcs.warwick.ac.uk

Daniel P. Spooner
University of Warwick, dps@dcs.warwick.ac.uk

Hong Jiang
University of Nebraska-Lincoln, jiang@cse.unl.edu

Donna N. Dillenberger
IBM T.J. Watson Research Center, engd@us.ibm.com

See next page for additional authors

Digital Object Identifier: 10.1109/TPDS.2006.18

Allocating Non-Real-Time and Soft Real-Time Jobs in Multiclusters

Ligang He, *Student Member*, *IEEE*, Stephen A. Jarvis, *Member*, *IEEE*, Daniel P. Spooner, Hong Jiang, *Member*, *IEEE Computer Society*, Donna N. Dillenberger, *Member*, *IEEE*, and Graham R. Nudd

Abstract—This paper addresses workload allocation techniques for two types of sequential jobs that might be found in multicluster systems, namely, non-real-time jobs and soft real-time jobs. Two workload allocation strategies, the Optimized mean Response Time (ORT) and the Optimized mean Miss Rate (OMR), are developed by establishing and numerically solving two optimization equation sets. The ORT strategy achieves an optimized mean response time for non-real-time jobs, while the OMR strategy obtains an optimized mean miss rate for soft real-time jobs over multiple clusters. Both strategies take into account average system behaviors (such as the mean arrival rate of jobs) in calculating the workload proportions for individual clusters and the workload allocation is updated dynamically when the change in the mean arrival rate reaches a certain threshold. The effectiveness of both strategies is demonstrated through theoretical analysis. These strategies are also evaluated through extensive experimental studies and the results show that when compared with traditional strategies, the proposed workload allocation schemes significantly improve the performance of job scheduling in multiclusters, both in terms of the mean response time (for non-real-time jobs) and the mean miss rate (for soft real-time jobs).

Index Terms—Scheduling, parallel systems, distributed systems, real-time systems, numerical algorithms.

1 Introduction

Clusters are now recognized as popular high-performance computing platforms for both scientific and commercial applications. Separate clusters are also being interconnected to create multicluster computing architectures [5]. The reason for this is two-fold: First, applications are increasingly exhibiting intensive computing requirements that have been shown to exceed the processing capability of any existing single cluster. A viable solution, therefore, is to integrate multiple clusters that can then be effectively viewed as a single system, although these constituent clusters may have different performance and supporting architectures, and may be located within a single organization or indeed across different geographical sites. A number of commercial products supporting multicluster environments are available, including the Platform LSF Multicluster [26], for example. Second, in some scenarios, a cluster needs to be partitioned into several subparts in order to support resource maintenance and data integrity [5], [14]. An approach to supporting the latter is to partition a cluster into multiple subclusters and employ a quorum mechanism to determine the subcluster whose results should be trusted [14].

- *L. He, S.A. Jarvis, D.P. Spooner, and G.R. Nudd are with the Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK. E-mail: {liganghe, saj, dps, grn}@dcs.warwick.ac.uk.*
- *H. Jiang is with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588-0115. E-mail: jiang@cse.unl.edu.*
- *D.N. Dillenberger is with the IBM T.J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: engd@us.ibm.com.*

Manuscript received 6 Jan. 2005; revised 25 May 2005; accepted 14 July 2005; published online 28 Dec. 2005.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-0006-0105.

Job scheduling in a distributed system can be categorized into two classes of activity based on the type of information on which the scheduling decisions are made [21], [25]. This information may be either

1. based on *averages*, including metrics such as the mean job arrival rate and the average processing capabilities of the constituent processing nodes, or
2. based on *instantaneous* measures, such as the execution time of the current job requesting execution or the current residual load on each computational resource.

Instantaneous scheduling schemes usually perform better when compared with their average-based counterparts [25], [12]. However, obtaining instantaneous system information for the scheduling of every job imposes a high overhead. This is especially true when the computational resources are geographically distributed, where the delay of retrieving system information may ultimately be intolerable [12]. Moreover, in some distributed systems consisting of autonomous servers, the system information recorded at each individual server may not always be available [12]. Hence, it remains necessary to develop average-based scheduling schemes to gain desirable performance improvements at a lower cost.

Average-based scheduling for sequential jobs in distributed systems usually consists of two fundamental components, workload allocation and job dispatching [24], [25]. The workload allocation scheme determines the proportion of workload directed to each resource, while the job dispatching strategy distributes the incoming jobs to each resource as the jobs arrive and, in so doing, satisfies the proportion of workload specified by the workload allocation scheme.

The workload allocation technique is a key factor for achieving desirable performance in average-based job scheduling. The metric for evaluating performance varies according to the job type. When jobs have soft real-time requirements (i.e., a fraction of jobs are permitted to miss their real-time requirements), performance metrics are usually chosen to measure the extent of requirements compliance. Slack [28] and deadline [16] are two commonly used real-time metrics, where slack is defined as the maximum wait time that a job can tolerate before execution. Slack is used in this paper as the soft real-time metric. The real-time requirements of a job are satisfied if the job's waiting time (in the system) is less than its slack. The performance metric *miss rate* is used to measure the proportion of jobs whose real-time requirements have been missed. If jobs have no real-time requirements, then a common performance goal is to reduce the *mean response time* [25].

In this paper, optimization techniques are investigated for scheduling both non-real-time and soft real-time sequential jobs in multicluster architectures, which consist of multiple homogeneous clusters with different service rates. Two workload allocation strategies, *Optimized mean Response Time* (ORT) and *Optimized mean Miss Rate* (OMR), are developed. ORT aims to achieve the optimized mean response time for incoming non-real-time job streams and OMR aims to achieve the optimized mean miss rate for soft real-time job streams.

The workload allocation strategies presented in this paper can be categorized as average-based scheduling as the workload parameter that both ORT and OMR take into account is the mean arrival rate. In addition, the OMR scheme also considers the probability distribution of job slack. The system parameters considered by ORT and OMR include the number of nodes in each cluster and the mean job service rate of the nodes. These workload allocation strategies do not require the size of every job and the workload status in each cluster.

Workload allocation in multiclusters is mathematically modeled using optimization equation sets. Numerical solutions with low time complexities are developed to solve the workload allocation for each cluster. The objective functions constructed for both ORT and OMR demonstrate similar properties and can therefore be solved using similar numerical solutions. These numerical solutions take as input the jobs' mean arrival rate λ. When the change in λ exceeds a predefined threshold, the numerical solutions are invoked on-the-fly so as to recalculate the proportion of workload for each cluster. This dynamic readjustment mechanism is feasible because of the low time complexities of the proposed numerical solutions. With this approach, the frequency in which the workload allocation algorithms are invoked can be reduced, since the current workload proportions are maintained until λ reaches the predefined threshold. This mechanism is also examined in the experimental studies presented in this paper.

Weighted Random (Rand) and Weighted Round-Robin (RR) are two commonly used job-dispatching strategies applied in real heterogeneous systems [25]. It has been shown that a Round-Robin policy typically attains higher scheduling performance than a Random policy [28]. In this paper, the proposed ORT and OMR strategies are combined with these two job dispatching strategies to generate four new job scheduling algorithms: ORT-RR, ORT-Rand, OMR-RR, and OMR-Rand. These scheduling algorithms treat incoming jobs equally and process them on a First-Come-First-Served basis (FCFS) [10].

This work is motivated by the need to provide effective e-commerce services in geographically distributed Web-server environments, supported by underlying multicluster architectures. In typical e-commerce environments, the requests sent by users for services may fall into different service classes (according to predefined Service-Level-Agreements) [23]. Each service class is associated with specific Quality-of-Service (QoS) requirements, which are often represented by the delay that the requests in this class can tolerate [23], [28] (corresponding to the slack in the soft real-time jobs defined above). The requests without any QoS requirements are referred to as the best-effort requests [23] (corresponding to non-real-time jobs). In this scenario, it is crucial to design judicial strategies to route these requests to individual cluster-based servers so as to optimize the desired performance, such as profit or mean response time of requests, across all the distributed servers. The profit under the Service-Level-Agreements constraints is maximized if the QoS requirements of requests are satisfied. Profit, therefore, depends on the miss rate of the requests' QoS requirements. Although this work is motivated by the application to e-commerce environments, the methodology employed can also be applied to broader workload management problems for general sequential jobs, where instantaneous workload and system information is difficult to obtain.

The remainder of this paper is organized as follows: Section 2 presents related work and the system and workload model assumed in this paper is introduced in Section 3. Two optimized workload allocation strategies are presented in Section 4 and the performance of these strategies is evaluated in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

There is now considerable literature to support research into multicluster systems [2], [4], [5], [13], [15], [17]. A multicluster model is presented in [5], for example, that integrates different workstation clusters into a virtual parallel machine. A supporting multiprotocol communication library is presented in [2] which is highly appropriate for multicluster systems. However, this research does not discuss suitable job scheduling schemes for multicluster systems.

It has been shown that it is nontrivial to optimize workload allocation in heterogeneous systems [3], [22], [25]. For example, it is shown in [22] that allocating workload proportional to computing capability does not achieve the best performance unless the system workload is very high. However, the study in [22] does not quantitatively develop a scheme to optimize performance. A similar problem is addressed in [3], where an optimization function is established. However, the solution to the objective function is not given and the optimization function is limited to

multicomputer systems as opposed to multicluster systems, where the former is an operational equivalent of a single heterogeneous cluster while the latter includes a heterogeneous cluster of clusters. A static workload allocation technique is addressed in [25] which aims to optimize mean response times in a heterogeneous cluster. Both an optimization function and its solution are given. Their solution is in fact a special case of this work, where each cluster in the multicluster architecture assumed here has only one computing node.

Workload allocation strategies have also been analyzed in terms of their ability to satisfy performance requirements in heterogeneous multiple processor systems [11]. The performance metrics used in [11] include mean response time, throughput, and system time. The methodology employed is instantaneous, that is, the proposed strategies make scheduling decisions for each job by taking as input the current number of jobs in each processor. In addition, the strategies in [11] are once again limited to a single-cluster model and do not consider soft real-time jobs.

Performance-based load balancing schemes are presented in [7] for enterprise application environments. Their work takes into account both system and application-oriented statistics, and dynamically calculates a workload-allocation weight for each server. The methodology applied is essentially instantaneous, as the system and application information at each server are needed for the load balancer to calculate appropriate weights. An interesting aspect of their work is that application-oriented statistics are also utilized, including the transactional success rates of applications and the topology of the different service classes.

Using non-real-time clusters to process soft real-time jobs is gaining in popularity [1], [10], [18], [19], [28]. The work presented in [20] documents the possibility of using two identical non-real-time servers to provide a soft real-time service, and the work in [28] extends this by investigating the feasibility of using a homogeneous cluster for soft real-time service. The performance of soft real-time job scheduling in terms of miss rate is also evaluated in [28]. However, their work is confined to a single cluster and does not consider the optimization of miss rate through judicial workload allocation.

3 System and Workload Model

The multicluster architecture assumed in this paper consists of n different clusters, where each cluster comprises a set of homogeneous computing nodes. Cluster $i(1 \leq i \leq n)$ is modeled using an $\mathrm{M}/\mathrm{M}/m_i$ queueing model [21], where m_i is the number of computing nodes in cluster i. The mean service rate of a node in cluster i is u_i. The multicluster architecture has two levels of scheduler, a global scheduler and multiple local schedulers, as shown in Fig. 1. The global scheduler is able to estimate the mean arrival rate of incoming jobs (there are existing techniques to do this, see [12]). As the jobs arrive, the global scheduler dispatches them immediately to the individual clusters using a combination of a First-Come-First-Served (FCFS) and a predefined workload allocation policy. Each local scheduler uses a single waiting queue to accommodate the jobs received from the global scheduler and sends these jobs on a FCFS basis to free processing nodes for execution. The execution is nonpreemptive.

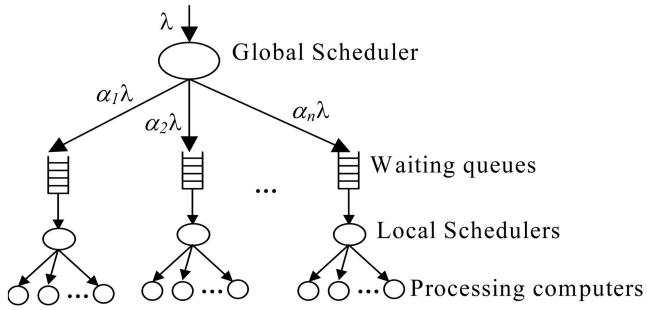


Fig. 1. The multicluster architecture.

There are multiple discrete service classes in a typical e-commerce environment. The requests belonging to the same service class have the same QoS requirements (slack). Hence, the slack of all requests usually follows a q-spike distribution [20], where q is the number of service classes offered by the service provider. Its probability density function $S(x)$ is formulated in (1). A 2-spike distribution is illustrated in Fig. 2a.

$$S(x) = \begin{cases} \beta_1 & x = s_1 \\ \beta_2 & x = s_2 \\ \vdots & \\ 1 - \sum_{i=1}^{q-1} \beta_i & x = s_q. \end{cases} \quad (1)$$

It is assumed in [20], [28] that the slack of a general soft real-time job follows a uniform distribution, as illustrated in Fig. 2b. Its probability density function $S(x)$ is given in (2),

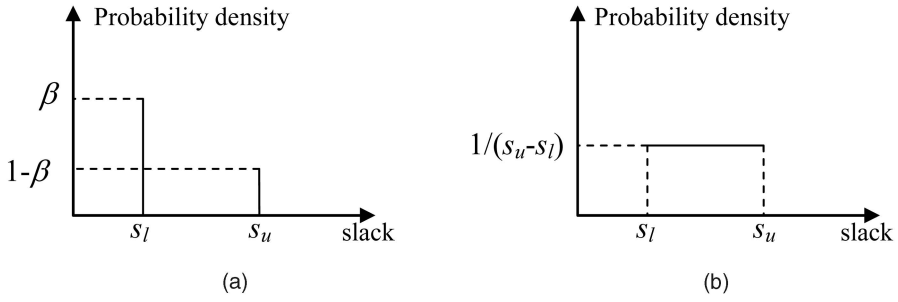


Fig. 2. Slack distribution. (a) Two-spike distribution. (b) Uniform distribution.

where s_u and s_l are the upper and lower limits of the slack, respectively.

$$S(x) = \frac{1}{s_u - s_l}. \quad (2)$$

A uniform distribution is used in this paper under which the proposed optimization techniques are derived. We also demonstrate that a q-spike distribution can be accommodated by this framework.

4 Workload Allocation

When a job stream with an average arrival rate λ is presented to the global scheduler (as shown in Fig. 1), it is decomposed by applying a job scheduling scheme and, as a result, a fraction α_i of all jobs are allocated to cluster i. The aim of the workload allocation scheme is to determine $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, a process that can be invoked dynamically taking λ as input.

4.1 Optimized Mean Response Time Strategy

For a non-real-time job stream, the workload allocation strategy aims to optimize the mean response time of the job stream in the multicluster system. The response time of a job is defined as the time from when the job arrives at the system until it is completed.

Intuitively, this workload allocation strategy might take into account the heterogeneity of the clusters performance, so that the workload fraction α_i allocated to cluster $i (1 \le i \le n)$ is proportional to its processing capability, $m_i u_i$. Hence, α_i is computed as

$$\alpha_i = \frac{m_i u_i}{\sum_{i=1}^{n} m_i u_i}. \quad (3)$$

This strategy is called *weighted workload allocation*. In the rest of this section, a detailed analysis is provided of the development of a workload allocation scheme for optimizing the mean response time.

The response time of a job is its waiting time in the queue plus its execution time. Hence, the average response time of the jobs in cluster i, denoted as R_i, can be computed by (4), where W_i is the mean waiting time of the jobs in cluster i.

$$R_i = W_i + \frac{1}{u_i}. \quad (4)$$

According to [21], the mean waiting time of jobs, W_i, can be computed as shown in (5), where ρ_i is the utilization of cluster i and W_{0i} is the mean remaining execution time of the job in service when a new job arrives.

$$W_i = \frac{W_{0i}}{1 - \rho_i}. \quad (5)$$

The formula for W_{0i} is given in (6), where P_{mi} is the probability that the system has no less than m_i jobs [8].

$$W_{0i} = \frac{P_{mi}}{m_i u_i}. \quad (6)$$

Suppose that the fraction of workload allocated to cluster i is α_i, then,

$$\rho_i = \frac{\alpha_i \lambda}{m_i u_i}. \quad (7)$$

The variable P_{mi} in (6) can be calculated by (8) [8], [21]

$$P_{mi} = \frac{(m_i \rho_i)^{m_i}}{(1 - \rho_i) m_i! \left[\sum_{k=0}^{m_i - 1} \frac{(m_i \rho_i)^k}{k!} + \frac{(m_i \rho_i)^{m_i}}{(1 - \rho_i) m_i!} \right]}. \quad (8)$$

Combining (4) to (8), we derive the formula for R_i in terms of α_i, as shown in (9):

$$R_i = \frac{m_i u_i \left(\frac{\alpha_i \lambda}{u_i}\right)^{m_i}}{\left[m_i! \sum_{k=0}^{m_i - 1} \frac{\left(\frac{\alpha_i \lambda}{u_i}\right)^k}{k!} + \frac{\left(\frac{\alpha_i \lambda}{u_i}\right)^{m_i}}{\left(1 - \frac{\alpha_i \lambda}{m_i u_i}\right)} \right] (m_i u_i - \alpha_i \lambda)^2} + \frac{1}{u_i}. \quad (9)$$

Thus, the mean response time of the incoming job stream over these n clusters, denoted by R, can be computed as in (10)

$$R = \sum\nolimits_{i=1}^{n} \alpha_i R_i. \quad (10)$$

Hence, in order to achieve the optimal mean response time of the job stream in the multicluster, the aim is to find a workload allocation $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ that minimizes (10) subject to $\sum_{i=1}^{n} \alpha_i = 1$ and $0 \le \alpha_i \le \frac{m_i u_i}{\lambda}$ (the constraint $\alpha_i \le \frac{m_i u_i}{\lambda}$ is used to ensure that cluster i does not become saturated). This is a constrained-minimum problem and, according to the Lagrange multiplier theorem [6], solving this problem is equivalent to solving (11)

$$\begin{cases} \sum_{i=1}^{n} \alpha_i = 1, \quad 0 \le \alpha_i \le \frac{m_i u_i}{\lambda} & (11a) \\ \frac{\partial}{\partial \alpha_k} \left(\sum_{i=1}^{n} \alpha_i R_i \right) \\ -v \frac{\partial}{\partial \alpha_k} \left(\sum_{i=1}^{n} \alpha_i - 1 \right) = 0 \quad 1 \le k \le n. & (11b) \end{cases}$$

Since α_i is the only unknown variable in the expression of R_i, (11) can be reduced to (12) by solving the partial differential equations in (11b).

$$\begin{cases} \sum_{i=1}^{n} \alpha_i = 1, \quad 0 \le \alpha_i \le \frac{m_i u_i}{\lambda} & (12a) \\ \frac{\partial}{\partial \alpha_k} (\alpha_k R_k) = v \quad 1 \le k \le n. & (12b) \end{cases}$$

It is impossible to find the general symbolic solution $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ from (12) due to the complexity of R_i. However, a property of (12b) is revealed (below) that enables us to develop a numerical solution for (12). This property is summarized in Theorem 1.

Theorem 1. *$\frac{\partial}{\partial \alpha_k} (\alpha_k R_k)$ is a monotonically increasing function of α_k.*

Proof. $\frac{\partial}{\partial \alpha_k} (\alpha_k R_k)$ can be transformed into (13)

$$\frac{\partial}{\partial \alpha_k} (\alpha_k R_k) = \alpha_k \frac{\partial R_k}{\partial \alpha_k} + R_k. \quad (13)$$

As in queueing theory [21], the mean response time of jobs (R_k) is a monotonically increasing function of the average job arrival rate α_k. Furthermore, the slope of the function (i.e., $\frac{\partial R_k}{\partial \alpha_k}$) also monotonically increases with the increase in α_k. Using (13), $\frac{\partial}{\partial \alpha_k} (\alpha_k R_k)$ is therefore identified as a monotonically increasing function of α_k.□

Based on Theorem 1, we develop a numerical solution to (12) and derive the optimized workload allocation $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. The numerical solution is shown in Algorithm 1.

Algorithm 1. Computation of workload allocation for optimized mean response time

Input: the lower and upper limit of v in (12), denoted as v_lower and v_upper

1: **while** $(v\_lower \leq v\_upper)$
2: $v\_mid = (v\_lower + v\_upper)/2$;
3: **for** each cluster $i(1 \leq i \leq n)$ **do**
4: **if** $(v\_mid < \frac{\partial}{\partial \alpha_i}(\alpha_i R_i)|_{\alpha_i=0})$ **then**
5: $\alpha_i = 0$;
6: **else if** $(v\_mid > \frac{m_i u_i}{\lambda}$) **then**
7: $v\_upper = v\_mid$;
8: **go to** Step 2;
9: **endif**
10: **else**
11: $\alpha\_lower = 0$; $\alpha\_upper = 1$;
12: **while** $(\alpha\_lower \leq \alpha\_upper)$
13: $\alpha\_mid = (\alpha\_lower + \alpha\_upper)/2$;
14: $v\_cur = \frac{\partial}{\partial \alpha_i}(\alpha_i R_i)|_{\alpha_i=\alpha\_mid}$;
15: **if** $(|v\_cur - v\_mid| \leq v\_valve)$ **then**
16: $\alpha_i = \alpha\_mid$;
17: $i = i + 1$;
18: **go to** Step 4;
19: **endif**
20: **if** $(v\_cur < v\_mid)$ **then**
21: $\alpha\_lower = \alpha\_mid$;
22: **else**
23: $\alpha\_upper = a\_mid$;
24: **endif**
25: **endwhile**
26: **endif**
27: **end for**
28: $\alpha\_sum = \sum_{i=1}^{n} \alpha_i$;
29: **if** $(|\alpha\_sum - 1| \leq \alpha\_valve)$ **then**
30: $\alpha_i(1 \leq i \leq n)$ is the correct workload allocation; exit with success;
31: **else if** $(\alpha\_sum < 1)$ **then**
32: $v\_lower = v\_mid$;
33: **endif**
34: **else**
35: $v\_upper = v\_mid$;
36: **endif**
37: **end while**

Algorithm 1 is explained as follows: According to (12), $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)$ $(1 \leq i \leq n)$ have to equal a common value v. Since $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)$ $(1 \leq i \leq n)$ is the monotonically increasing function of α_i, in Algorithm 1, the lower limit of v, v_lower, can be set as the minimum of $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)$ $(1 \leq i \leq n)$ evaluated at $\alpha_i = 0$; while the upper limit of v, v_upper, can be set as the maximum of $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)$ $(1 \leq i \leq n)$ evaluated at $\alpha_i = 1 - \xi$, where ξ is an infinitesimal quantity (e.g., 0.001). Since the differential of $\alpha_i R_i$ at $\alpha_i = 0$ decreases as $m_i u_i$ increases, v_lower is effectively $\frac{\partial}{\partial \alpha_k}(\alpha_k R_k)|_{\alpha_k=0}$, where k satisfies the condition that the value of $m_k u_k$ is the greatest of all the clusters (which means that cluster k has the greatest computing capabilities). For an arbitrary v_mid between v_lower and v_upper, the algorithm searches for a suitable α_mid (the search space of α_i is [0, 1]) so that the difference between v_mid and v_cur, computed by $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)|_{\alpha_i=\alpha\_mid}$, is less than a predefined valve v_valve. In this way, a set of specific values of $\alpha_i(1 \leq i \leq n)$ can be obtained. Then, the algorithm adds $\alpha_i(1 \leq i \leq n)$ to obtain α_sum. If the sum is greater than 1, by a predefined valve α_valve, it means that the current v_mid is too high and a lower value should therefore be used for computing a new set of α_i. If, on the other hand, the sum is less than 1, by α_valve, a higher value of v should be used for the next iteration of the computation of α_i.

Since this binary search technique is used to search for v and α_i in their respective search spaces $[v\_lower, v\_upper]$ and $[\alpha\_lower, \alpha\_upper]$ (e.g., [0, 1]), the time complexity of Algorithm 1 is $O(n \log k_v \log k_\alpha)$, where n is the number of clusters in the multicluster system, and k_v and k_α are the number of elements in the search spaces of v and α_i, which can be calculated using

$$k_v = \frac{v\_upper - v\_lower}{\varphi}$$
$$k_\alpha = \frac{\alpha\_upper - \alpha\_lower}{\gamma},$$

where φ and γ represent the precision in the calculation. Since φ and γ are predefined constants (in the experiments in this paper, φ and γ are set to 10^{-3} and 10^{-4}, respectively), the time complexity is linear with the number of clusters, n.

The feasibility and effectiveness of Algorithm 1 are proven in Theorem 2.

Theorem 2. *The workload allocation $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ computed by Algorithm 1 minimizes the average response time of the incoming job stream in a multicluster system of n clusters.*

Proof. We need to prove two cases in order to prove the theorem: 1) Algorithm 1 can generate a set $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ that satisfies (12), and 2) the generated workload allocation strategy can lead to the minimal mean response time of the job stream over these n clusters.

$\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)$ $(1 \leq i \leq n)$ is a monotonically increasing function of α_i. Hence, for any v in $[v\_lower, v\_upper]$, if v is in the value field of $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)$, which is $[\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)|_{\alpha_i=0}$, $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)|_{\alpha_i=1-\xi}]$ $(1 \leq i \leq n)$, using the binary search, we can find such a value of α_i in its value field [0, 1] that satisfies $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i) = v$. If, for some cluster $k(1 \leq k \leq n)$, v is not in the value field of $\frac{\partial}{\partial \alpha_k}(\alpha_k R_k)$ (i.e., $v < \frac{\partial}{\partial \alpha_k}(\alpha_k R_k)|_{\alpha_k=0}$), α_k is set to 0. Suppose we obtain a workload allocation $\{\alpha'_1, \alpha'_2, \ldots, \alpha'_n\}$ from $v = v'$ and obtain another $\{\alpha''_1, \alpha''_2, \ldots, \alpha''_n\}$ from $v = v''$. Since $\frac{\partial}{\partial \alpha_i}(\alpha_i R_i)$ monotonically increases over α_i, if $v'' > v'$, there must be some $\alpha''_i > \alpha'_i$ $(1 \leq i \leq n)$ and, if $v'' < v'$, then $\alpha''_i < \alpha'_i$ $(1 \leq i \leq n)$. Therefore, using the binary search we can find some value of v so as to satisfy $\sum_{i=1}^{n} \alpha_i = 1$. The first part of the proof therefore holds. Equation (12) is the equivalent expression of the Lagrange multiplier theorem. Thus, the solution to (12), $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, minimizes the mean response time of the job stream. The theorem as a whole, therefore, holds.□

4.2 Optimized Mean Miss Rate Strategy

In this section, a workload allocation strategy, called OMR, is developed to optimize the mean miss rate of the

incoming soft real-time job stream in a multicluster. Every soft real-time job has some slack, following a uniform distribution, as defined in (2).

We continue to model cluster i (of m_i nodes) as an $M/M/m_i$ queue $(1 \le i \le n)$. As shown in [21], in an $M/M/m_i$ queue, the probability distribution function of the job waiting time, $P_w(x)$ (i.e., the probability that the job waiting time is less than x is $P_w(x)$), is calculated using (14) [21], where ρ_i and P_{mi} are the same variables as those found in (5) and (6).

$$P_w(x) = 1 - P_{mi} e^{-m_i u_i (1-\rho_i)x}. \tag{14}$$

Using the probability density function of the slack, the miss rate of the soft real-time jobs allocated to cluster i, denoted by MR_i, can be computed using (15)

$$MR_i = \int_{s_l}^{s_u} S(x)(1 - P_w(x))dx. \tag{15}$$

Applying (2) and (14) and solving the integral, (15) becomes (16), where the workload fraction α_i for cluster i is the only unknown variable.

$$MR_i = \frac{m_i u_i \left(\frac{\alpha_i \lambda}{u_i}\right)^{m_i} \left[e^{-(m_i u_i - \alpha_i \lambda)s_l} - e^{-(m_i u_i - \alpha_i \lambda)s_u}\right]}{\left[m_i! \sum_{k=0}^{m_i-1} \frac{\left(\frac{\alpha_i \lambda}{u_i}\right)^k}{k!} + \frac{\left(\frac{\alpha_i \lambda}{u_i}\right)^k}{\left(1 - \frac{\alpha_i \lambda}{m_i u_i}\right)}\right] (m_i u_i - \alpha_i \lambda)^2 (s_u - s_l)}. \tag{16}$$

The mean miss rate of the incoming soft real-time job stream over these n clusters, denoted by MR, can be computed using (17)

$$MR = \sum_{i=1}^{n} \alpha_i \times MR_i. \tag{17}$$

Similar to the case of minimizing the mean response time, this is a constrained-minimum problem. We need to find a workload allocation that minimizes MR in (17) subject to $\sum_{i=1}^{n} \alpha_i = 1$ and $0 \le \alpha_i \le \frac{m_i u_i}{\lambda}$, which is equivalent to solving the following equation set:

$$\begin{cases} \sum_{i=1}^{n} \alpha_i = 1, \quad 0 \le \alpha_i \le \frac{m_i u_i}{\lambda} & (18a) \\ \frac{\partial}{\partial \alpha_k}(\alpha_k \times MR_k) = v \quad 1 \le k \le n. & (18b) \end{cases}$$

In the previous section, we stated that the numerical solution to (12) is based on the property that $\frac{\partial}{\partial \alpha_k}(\alpha_k R_k)$ is a monotonically increasing function of α_k. Theorem 3 is introduced to demonstrate that $\frac{\partial}{\partial \alpha_k}(\alpha_k \times MR_k)$ in (18) also monotonically increases over α_k. Theorem 3 can be proven directly from Theorem 1 and the detailed proof is therefore omitted. With this property, a numerical solution is also developed to solve (18). The solving algorithm is similar to Algorithm 1 and the proof of the algorithm's effectiveness is similar to Theorem 2. Hence, they are also omitted in this paper.

Theorem 3. *$\frac{\partial}{\partial \alpha_k}(\alpha_k \times MR_k)$ is a monotonically increasing function of α_k.*

The above derivation assumes that the slack follows a uniform distribution. When the slack follows a q-spike distribution as defined in (1), the miss rate of the soft real-time jobs in cluster i, MR_i, can be computed using (19) (corresponding to (15) for a uniform distribution)

$$MR_i = \sum_{i=1}^{q} \beta_i (1 - P_w(s_i)). \tag{19}$$

It can be shown that, with (19), the property described in Theorem 3 is preserved. Hence, the numerical solution remains effective.

5 Experimental Evaluation

In this section, the workload allocation strategies presented in this paper are evaluated through extensive simulations.

5.1 Experimental Setting

An experimental simulator has been developed to evaluate the performance of the proposed workload allocation techniques under a wide range of system configurations and workload levels. The experimental parameters are chosen either based on those used in the literature [9], [16], [25] or so that they represent a realistic workload. The simulator consists of a collection of clusters. In every cluster, a central node, or head node, acts as the local scheduler and the local schedulers in all clusters are connected to a global central node in the multicluster system. The global central node acts as the global scheduler, which receives all incoming jobs and schedules them onto the constituent clusters.

Two types of job streams (non and soft real-time) are generated using the same parameters except that every soft real-time job has one additional metric, the *slack*, which follows a uniform distribution. Experimental evaluations have also been conducted using the q-spike distributions. As the results exhibit a similar behavior, they are therefore omitted. Each job stream includes 500,000 independent jobs. The first 100,000 jobs are considered to be the initiation period, allowing the system to achieve a steady state, and the last 100,000 jobs are considered to be the ending period. Statistical data are therefore collected from the middle 300,000 jobs. The mean size of the incoming jobs, $\overline{e}$, in terms of the processing time on a node with speed 1.0, is set to be 84.0 seconds.

Based on the mean job size, the job arrival rate λ_s, at which the system is saturated, can be computed as follows: The workload levels in these experiments are measured by the percentage of the saturated arrival rate.

$$\lambda_s = \sum_{i=1}^{n} m_i u_i \text{ no./sec.}$$

The burstiness of the job arrivals may cause performance deterioration [25], [27]. The burstiness can be measured by the Coefficient of Variation (CV) of the job interarrival times (the CV of a random variable is calculated as its standard deviation divided by its mean) [25], [27]. For a Poisson arrival, the CV of the interarrival times is 1. However, the job arrivals in real environments tend to be burstier than this. Job traces in a real computing system are analyzed in [27], which shows that the CV of the interarrival times is 2.6. The work presented in [25] shows that the job arrivals can be modeled using a hyperexponential distribution.

TABLE 1
Combinations of the Workload Allocation and the Job Dispatching Strategies

		Workload allocation schemes		
		ORT	OMR	Weighted
Job dispatching strategies	Random	ORT-Rand	OMR-Rand	W-Rand
	Round-Robin	ORT-RR	OMR-RR	W-RR

In this work, experiments have been conducted in which job arrivals follow a Poisson process and the job sizes follow an exponential distribution, as assumed in the theoretical derivation. We have also conducted experiments in Section 5.6 to investigate the impact of the CV of the interarrival times on the performance of the ORT and OMR strategies, in which a two-stage hyperexponential distribution is used to model the job arrival process.

Various real-world studies have shown that job sizes often exhibit a heavy-tailed distribution. The experiments in this paper also utilize job sizes that are modeled by a Bounded Pareto distribution [25] (see Section 5.6), which preserves this heavy-tailed property.

The global scheduler takes λ as input and invokes a workload allocation algorithm (ORT or OMR) to calculate the workload proportion for each cluster. When the global scheduler detects a change in λ, it maintains the current workload allocation until the change reaches a certain threshold. Experiments have been conducted (and the results are presented in Section 5.7) in order to gain an insight into determining this threshold.

When a given level of incoming workload is presented to the multicluster, the global scheduler calculates the workload proportion for each cluster. Then, the level of the incoming workload is gradually changed. The resulting workload allocation is performed in the following ways: 1) approximate allocation—the global scheduler maintains the initial workload proportions and 2) accurate allocation—the global scheduler takes as input the current mean arrival rate and generates the new workload proportions. The performance difference between these two methods can be viewed as the effect of maintaining the workload allocation as the workload level changes.

Three workload allocation strategies (ORT, OMR, and weighted workload allocation) and two job dispatching strategies (weighted Random and weighted Round-Robin) are tested in the experiments. In weighed Random dispatching, a cluster is randomly selected while, in weighted Round-Robin dispatching, a cluster is selected in a round-robin fashion by the global scheduler. Both strategies satisfy the condition that the probability that a new job is sent to cluster i is α_i. Six scheduling algorithms are evaluated, each of which is the combination of a workload allocation scheme and a job dispatching strategy. These six algorithms (ORT-Rand, ORT-RR, OMR-Rand, OMR-RR, W-Rand, and W-RR) are listed in Table 1.

The performance metrics used in the experiments include the *mean response time* and the *mean miss rate*. Each point in the performance curves is plotted as the average result of five independent runs of the job streams with different initialization random numbers. In order to gain insight into the difference in the allocation behaviors between the OMR and the ORT strategies, ORT is also used to allocate the soft real-time job stream.

It has been shown that the dynamic least load algorithm can be used as the lower bound of the mean response time obtained by the average-based algorithms in a single cluster [25]. In the experiments presented in this paper, the performance of the dynamic least load algorithm (DLL) for the multicluster architecture is also used as the ideal bound. In the DLL algorithm, when a job arrives at the global scheduler, it schedules the job to the cluster with the least workload. Similarly, a dynamic least miss-rate (DLM) algorithm is used as the lower bound of the mean miss rate of the soft real-time job stream in the multicluster system. The DLM algorithm schedules the newly arriving soft real-time jobs to the cluster offering the least waiting time. These dynamic algorithms incur significant overheads as the global scheduler has to gather information on the current workload or waiting time from all constituent clusters when scheduling every incoming job. This makes the global scheduler a likely source of performance bottlenecks. Furthermore, when the clusters are geographically distributed, the delay brought about in retrieving system information may be intolerable and the collected information may be so out of date that the scheduler is unable to make accurate scheduling decisions.

5.2 Effect of Workload

Fig. 3a and Fig. 3b show the impact of the workload level on the mean response time and the mean miss rate of the jobs (for increased clarity the results for W-Rand and OMR-Rand are omitted in Fig. 3a and the results for W-Rand and ORT-Rand are omitted in Fig. 3b). The multicluster in this experiment consists of four clusters whose configurations are listed in Table 2. For the soft real-time jobs, the slack follows a uniform distribution in the range [0, 30].

It can be seen in Fig. 3a that the ORT-RR algorithm performs significantly better than W-RR and OMR-RR, while ORT-Rand outperforms W-Rand and OMR-Rand. Moreover, ORT-Rand also performs better than W-RR. This suggests that the ORT strategy performs much better than other strategies in terms of the mean response time. Furthermore, the performance difference increases as the workload decreases. For example, ORT-RR outperforms W-RR by 47.4 percent when the workload is 0.1, while the difference is 13.3 percent when the workload is 0.9. This trend can be explained as follows: The weighted workload allocation strategy allocates the same fraction of workload to a cluster even if the workload varies. However, the waiting time accounts for a lower proportion of the response time as the workload decreases. Hence, in order to reduce the response time, a higher proportion of the incoming workload should be allocated to the cluster with the greater u_i. The ORT strategy is able to satisfy this allocation requirement. For example, when the workload is 0.1, the proportion of the workload for clusters 1 and 2 is 0.752 and 0.248, while the proportion for clusters 3 and 4 is 0, as shown in Table 3.

Another observation from Fig. 3a is that under OMR, the jobs' mean response time first decreases and then increases as the workload increases. This is a result of the allocation characteristic of OMR. The strategy pursues the optimized

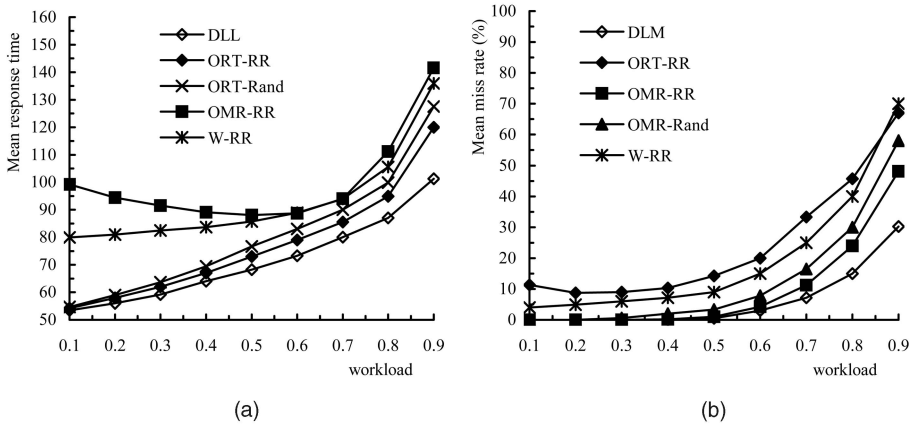


Fig. 3. Impact of workload on (a) mean response time and (b) mean miss rate.

mean miss rate and takes the probability distribution of the waiting time into account. This may lead to a different workload allocation from that governed by ORT. An example is shown in Table 3. It suggests that it is necessary to utilize different workload allocation strategies to optimize the respective performance.

Fig. 3b shows the impact of the incoming workload on the mean miss rate. It can be observed that the OMR-RR and the OMR-Rand scheduling outperform the other scheduling algorithms at all workload levels. This suggests that the OMR allocation strategy performs better than the weighted allocation strategy in terms of the mean miss rate.

It can be observed from both figures that, under the same workload allocation strategies, the algorithms employing weighted Round-Robin dispatching outperform those using the weighted Random dispatching. This is because Round-Robin dispatching can reduce the burstiness of job arrivals in a cluster compared with random dispatching. Hence, the results for ORT-Rand, OMR-Rand, and WW-Rand are omitted in the following sections.

In both figures, although DLL outperforms ORT-RR and ORT-Rand, the performance difference is small, especially when the workload is low. A similar pattern can be observed between the DLM and OMR-RR algorithms. This suggests that it is beneficial to apply low-cost optimized workload allocation, especially when the workload is low.

5.3 Effect of Node Speed

Since the nodes in the clusters are homogeneous, the heterogeneity of the multicluster is measured by the difference between the cluster size and node speed in each cluster. Fig. 4 demonstrates the impact of the difference of node speed. Here, the multicluster consists of four clusters and the number of nodes in each cluster is set to four. The speed of the nodes in cluster 1 varies from 21 to six with decrements of three, while the speed of all nodes in the three remaining clusters increases from one to six with increments of one. Thus, the multicluster ranges from a highly heterogeneous system to a homogeneous system, while the average speed of all nodes remains constant (i.e., six). The slack follows a uniform distribution in [0, 10].

Fig. 4a shows the impact of the difference of the node speeds on the mean response time. It can be observed in Fig. 4a that, as the speed difference increases, the mean response time decreases significantly under the ORT allocation strategy (the maximum decrease is 68 percent), while it remains approximately the same under the weighted allocation strategy. This is because, as the speed difference increases, despite the average node speed remaining constant, a higher proportion of the workload is sent to cluster 1 under the ORT strategy (higher than $m_1u_1/\sum_{i=1}^{n} m_iu_i$), while the weighted allocation strategy does not make full use of the computing power of cluster 1. This suggests that, under the ORT strategy, the speed difference among the clusters is a critical factor governing the mean response time.

An initial observation from Fig. 4b is that the OMR strategy performs better than the other strategies in all speed combinations. Moreover, under OMR, the mean miss rate remains approximately the same as the speed difference varies. The experimental results for the other levels of workload also show similar patterns. This suggests that, under OMR, the speed difference among clusters is not an important parameter for the mean miss rate. This differs from the characteristics of ORT for mean response time. This divergence may originate from the difference between the expressions of the response time and the miss rate (see (12) and (16)): There is an additional value $1/u_i$ in (12) while the value is absent in (16).

Table 4 shows the corresponding workload proportion in the clusters under these three workload allocation strategies when the speed difference decreases from [21, 1] to [15, 3]. The data in this table can be used to gain an insight into the

TABLE 2
System Setting in Fig. 3

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
The number of nodes	3	5	7	9
Relative speed of every node	20	16	12	8

TABLE 3
Fraction of the Jobs Allocated to Each Cluster (Workload = 0.1)

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
ORT	0.752	0.248	0	0
W	0.203	0.270	0.284	0.243
OMR	0.048	0.202	0.351	0.399

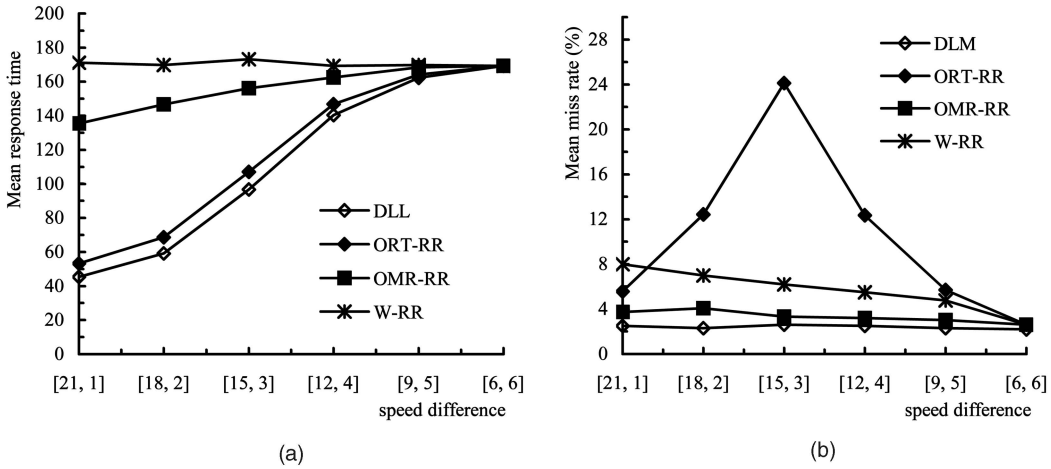


Fig. 4. The impact of speed difference on (a) the mean response time and (b) the mean miss rate; the arrival rate is 50 percent of the saturated arrival rate.

allocation behaviors of these strategies. It can be seen in Table 4 that, in all cases, the proportion of workload in cluster 1 is highest under ORT, lowest under the weighted allocation strategy, and OMR sits between these. The workload sent to cluster 1 under the weighted allocation scheme does not make full use of the computing power of these fast nodes. The mean miss rate is therefore improved when a higher proportion of workload is sent to cluster 1 under OMR. However, if a higher proportion than that under OMR is sent to cluster 1, the mean miss rate will be compromised despite the fact that the mean response time keeps improving.

5.4 Effect of Cluster Size

Fig. 5a and Fig. 5b show the impact of the difference of cluster size on the mean response time and mean miss rate. The number of nodes in cluster 1 decreases from 21 to six with decrements of three, while the number of nodes in clusters 2-4 increases from one to six with increments of one. The total number of nodes in the multicluster remains constant, i.e., 24. The speed of all nodes is set to be 10 and the incoming workload is 50 percent of the saturated workload.

It can be observed from Fig. 5a that the ORT strategy performs better than other workload allocation strategies in terms of the mean response time. A further observation is that the mean response time decreases as the difference in the number of nodes increases. This suggests once again that the ORT strategy is able to make full use of the clusters with more computing power as the heterogeneity of the multicluster system increases. When the system heterogeneity changes from [6, 6] (homogeneous system) to [21, 1], the mean response time decreases by 24 percent. According to the experimental results from Fig. 4a, the decline in the mean response time is as much as 68 percent when the system varies from the speed difference of [21, 1] to a homogeneous system. It suggests that the difference in the node speed has a more significant impact on the mean response time than the difference in cluster size.

As can be seen in Fig. 5b, OMR outperforms the other strategies in terms of mean miss rate. Furthermore, the mean miss rate is improved as the difference in the number of nodes decreases. This result differs from those in Fig. 4b, where the speed difference has no significant impact. This is because the miss rate depends on the probability distribution of the waiting time of jobs, on which the number of nodes has greater impact than node speed.

TABLE 4
Fraction of Jobs Allocated to Each Cluster as the Speed Difference Varies

Speed difference	OMR (cluster1, cluster 2-4)	ORT (cluster1, cluster 2-4)	Weighted (cluster1, cluster 2-4)
[21, 1]	(0.912, 0.029)	(1, 0)	(0.875, 0.042)
[18, 2]	(0.802, 0.066)	(1, 0)	(0.75, 0.083)
[15, 3]	(0.676, 0.108)	(0.962, 0.013)	(0.625, 0.125)

5.5 Effect of Slack

For the soft real-time job stream, job slack is an important parameter influencing the miss rate. Fig. 6a shows the impact of slack on the mean response time under the OMR strategy (the slack has no impact on the mean response time for the other strategies); Fig. 6b demonstrates the impact of job slack on the OMR and weighted workload strategies.

In Fig. 6a, the mean response time is approximately the same as the slack varies. This suggests that the fraction of workload in each cluster remains approximately the same. It can be observed from Fig. 6b, that under both OMR and weighted allocation, the mean miss rate improves as the slack increases. Further, the mean miss rate demonstrates a linear decrease under the weighted allocation strategy, while its improvement diminishes under the OMR strategy as the slack increases. This is because the OMR strategy takes into account the distribution of job waiting times while the weighted allocation strategy has no such capability.

5.6 Effect of the Coefficient of Variation

In the experiments presented in this section, job arrivals are modeled by a two-stage hyperexponential distribution, in which the CV can be adjusted by changing the distribution parameters. Job sizes are modeled by a Bounded Pareto distribution [25] (exhibiting a heavy-tailed property) as follows, where k and q are the lower and upper limit of job size, and α is a parameter that reflects the variability of job size. In the experiments, these parameters are set to:

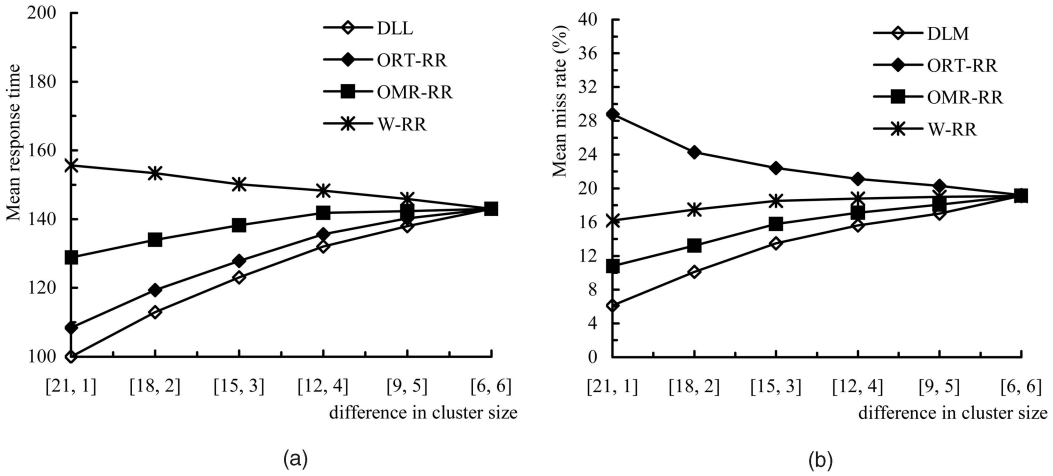


Fig. 5. The impact of the difference in the number of nodes in each cluster on (a) the mean response time and (b) the mean miss rate; the incoming workload is 0.5.

$k = 15.0$, $q = 4241.0$, $\alpha = 1$. The mean job size is the same as that in Fig. 2a:

$$f(x) = \frac{\alpha k^{\alpha}}{1 - (k/q)^{\alpha}} x^{-\alpha-1} \ (k \le x \le q).$$

Fig. 7a shows the effect of the CV of the interarrival times on the mean response time, where the CV increases from 1.0 to 3.0 with increments of 0.5 while the average arrival rate remains unchanged. We only show the results for the case where the workload level is 0.3 as the results for other workload levels demonstrate similar patterns.

It can be seen in Fig. 7a that, under these three strategies, the mean response time increases as the CV increases, as is to be expected. When the CV is 3.0, ORT-RR still outperforms W-RR by 23.1 percent, while the performance difference between ORT-RR and DLL is only 6.6 percent. These results suggest that the burstiness of job arrivals does not notably impair the advantages of the ORT workload allocation strategy.

Fig. 7b shows the mean response time as a function of the workload level when the CV is set to 3.0. As can be observed from Fig. 7b, the performance curves of ORT-RR, DLL, and W-RR demonstrate similar patterns to those previously seen in Fig. 3a. When the workload is 0.1, ORT-RR outperforms W-RR by 41.9 percent while the advantage is 10.3 percent when the workload is 0.9. These results imply once again that ORT-RR consistently performs better than W-RR even if there exists higher burstiness in job arrivals.

The experimental results for OMR-RR with gradually increasing CV lead to similar conclusions, i.e., the burstiness in job arrivals does not significantly weaken the advantage of OMR-RR over W-RR in terms of the mean miss rate, and the performance difference between OMR-RR and DLM remains small. These experimental results are not shown for brevity.

5.7 Effect of the Approximate Allocation

When the global scheduler detects a change in λ, it uses an approximate allocation, that is, it maintains the current workload allocation until the change in λ reaches a certain threshold, rather than immediately recalculating the workload proportions as in accurate allocation.

Fig. 8a demonstrates the performance comparison (in terms of mean response time) between the approximate and accurate allocation, when the initial workload level is

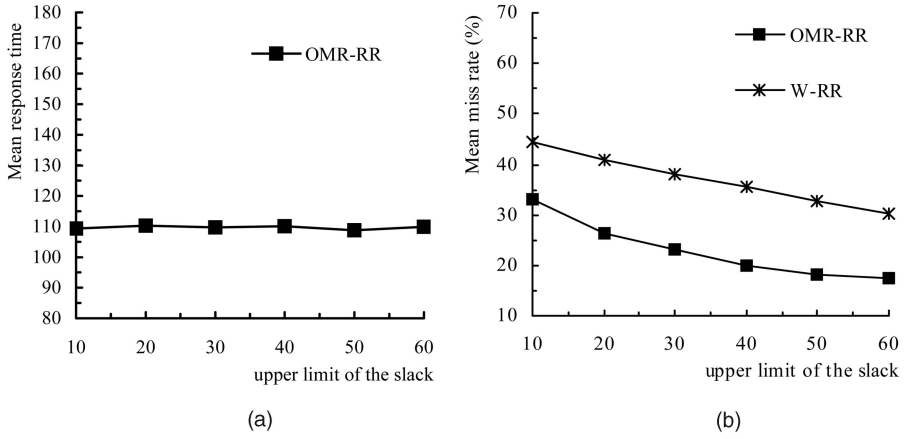


Fig. 6. The impact of job slack on the mean response time and the miss rate; (a) the mean response time; (b) the miss rate; the incoming workload is 0.8; the slack follows a uniform distribution between 0 and its corresponding upper limit.

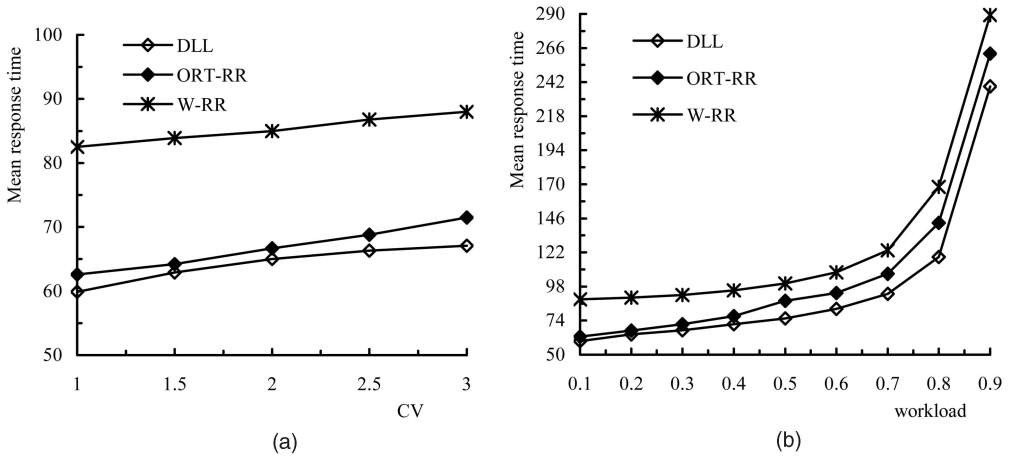


Fig. 7. Effect of the Coefficient of Variation (CV). (a) The effect of the CV on the mean response time; the workload level is 0.3; job arrivals follow a two-stage hyperexponential distribution. (b) Performance comparison under different workload levels; the CV is 3.0; job arrivals follow a two-stage hyperexponential distribution.

0.3 and the workload level changes from $(0.3 - 0.3 \times 30\%)$ to $(0.3 + 0.3 \times 30\%)$.

As can be observed from Fig. 8a, the performance achieved by the approximate allocation is worse than that by the accurate allocation when the workload level deviates from its initial value. Moreover, the performance deterioration increases as the deviation of the workload level increases, as is to be expected. The global scheduler invokes the workload allocation algorithm to recalculate the workload proportions only when the change in workload level reaches such a percentage that the performance deteriorates by a predefined level (e.g., 3 percent). This mechanism is feasible since the performance deterioration is very moderate. For example, the performance deteriorates by 3 percent when the workload level changes by 12 percent (or -11 percent). When the workload level is 0.3, the ORT strategy outperforms the weighted allocation by 33 percent.

Fig. 8b shows the incremental percentage of the initial workload level when the performance (mean response time) obtained by the approximate allocation deteriorates by 3 percent, as compared with the performance demonstrated by the accurate allocation. Under the initial workload level of 0.5, for example, performance deteriorates by 3 percent when the workload level increases by 9 percent (up to 0.545). The experiments have been conducted under various initial levels of workload increasing from 0.1 to 0.9 with increments of 0.1. The results for other levels of performance deterioration (e.g., 5 percent) are omitted since they show similar patterns.

It can be observed from Fig. 8b that the incremental percentage of workload first decreases and then increases as the workload level increases. This can be explained as follows: When the initial workload level is very low (e.g., 0.1), the incremental percentage has to be relatively high to make a large difference in the absolute increase. For example, when the initial workload level is 0.1, the incremental percentage is 33 percent and the absolute increase is only 0.033. When the initial workload level is very high (e.g., 0.9), the reason why the incremental percentage is high is different from the case where the initial workload level is very low. As demonstrated in Fig. 3a (showing the effect of the workload level on mean response time), the performance advantage of the ORT strategy over the weighted allocation strategy diminishes when the workload level becomes very high. This result

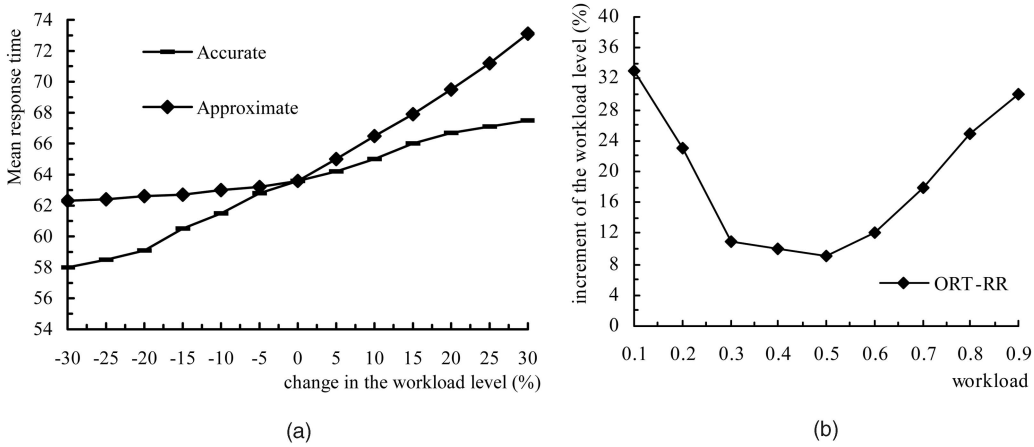


Fig. 8. Effect of the approximate workload allocation. (a) Effect of the change in the workload level on mean response time; the initial workload level is 0.3. (b) Incremental percentage of the workload level when the performance deterioration in terms of the mean response time reaches 3 percent; system and workload parameters are the same as those in Fig. 3a.

suggests that, when the workload level is high, the performance is less sensitive to the change of workload proportions among clusters, which will also explain why the incremental percentage has to be high to make the performance deteriorate by a given degree (which is 3 percent in Fig. 8b).

In Fig. 8b, the minimal incremental percentage is 9 percent. In addition, the experimental results show that the decreasing percentage is approximately symmetric along the x-axis (the figure is omitted in this paper). Hence, a conservative way to guarantee less than 3 percent deterioration in terms of mean response time is to ensure that the global scheduler employes ORT to recalculate the workload proportions when it detects a change in the workload of 9 percent.

The experimental results for OMR show similar patterns to that of ORT. For example, when the system and workload parameters are the same as those in Fig. 3b, the experimental results demonstrate that, in order to guarantee less than 1 percent performance deterioration in terms of mean miss rate, the global scheduler can recalculate the workload allocation when it detects a 6 percent change in the workload level. These results are omitted in this paper for brevity.

The experiments presented in this section are conducted in order to investigate the effect of the approximate allocation on performance when the workload level changes. However, the results also indicate that the workload allocation strategies presented in this paper can tolerate certain levels of inaccuracies in the estimation of the mean arrival rate. When the estimation of the mean arrival rate is inaccurate, the performance deterioration is moderate.

6 Conclusions

In this research, workload allocation strategies are developed for multicluster architectures (a heterogeneous cluster of clusters). Two average-based workload allocation strategies (ORT and OMR) are proposed that deal with different types of job. The ORT strategy can optimize the mean response time of non-real-time jobs, while the OMR strategy can optimize the mean miss rate of a soft real-time job stream. These proposed workload allocation strategies are combined with job dispatching strategies based on Weighted Random and Weighted Round-Robin policies. Both ORT and OMR strategies take the mean arrival rate as input and calculate workload proportions for each cluster. When a change in the mean arrival rate reaches a certain threshold, the proportions of workload are updated dynamically.

The effectiveness of these two strategies is proved through theoretical analysis. The proposed scheduling algorithms are also evaluated through extensive experimentation. The results show that:

1. the scheduling algorithms that utilize ORT and OMR perform significantly better than the algorithms that do not employ these optimization techniques;
2. the performance achieved by ORT and OMR approaches that obtained by the schemes based on instantaneous measures, particularly when the workload level is low;
3. bursty arrivals of incoming jobs do not notably impair the performance advantages of the ORT and OMR strategies; and
4. the performance deterioration of these schemes is moderate when the mean arrival rate of the jobs change, which implies that it is feasible that the workload proportions are recalculated only when the change in the mean arrival rate reaches a certain threshold.

The current system and workload model assumed in this paper imply that all the submitted requests can be executed successfully. However, this may not always be the case in some e-commerce environments. Future work includes incorporating into the current model possible failures in request executions, and developing new workload allocation schemes that not only preserve the advantages of the current strategies but also integrate additional fault-tolerant functionality.

References

[1] B. Adelberg, H. Garcia-Molina, and B. Kao, "Emulating Soft Real-time Scheduling Using Traditional Operating System Schedulers," *Proc. 1994 IEEE Real-Time Systems Symp.,* pp. 292-298, 1994.

[2] O. Aumage, "Heterogeneous Multi-Cluster Networking with the Madeleine III Communication Library," *Proc. 16th Int'l Parallel and Distributed Processing Symp. (IPDPS 2002),* pp. 172-183, 2002.

[3] S.A. Banawan and N.M. Zeidat, "A Comparative Study of Load Sharing in Heterogeneous Multicomputer Systems," *Proc. 25th Ann. Simulation Symp.,* pp. 22-31, 1992.

[4] S. Banen, A.I.D. Bucur, and D.H.J. Epema, "A Measurement-Based Simulation Study of Processor Co-Allocation in Multicluster Systems," *Proc. Ninth Workshop Job Scheduling Strategies for Parallel Processing,* pp. 105-128, 2003.

[5] M. Barreto, R. Avila, and P. Navaux, "The MultiCluster Model to the Integrated Use of Multiple Workstation Clusters," *Proc. Third Workshop Personal Computer-Based Networks of Workstations,* pp. 71-80, 2000.

[6] D.P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods,* ISBN: 1-886529-04-3, 1996.

[7] A. Bivens, D. Dillenberger, C. Chhuor, G. Ferris, W. Chou, and J. Fenton, "CISCO Load Balancing with SASP and Enterprise Workload Management," Oct. 2004.

[8] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi, *Queueing Networks and Markov Chains, Modeling and Performance Evaluation with Computer Science Applications.* John Wiley & Sons, 1998.

[9] A.I.D. Bucur and D.H.J. Epema, "The Maximal Utilization of Processor Co-Allocation in Multicluster Systems," *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS 2003),* pp. 60-69, 2003.

[10] K. Chen and L. Decreusefond, "Just How Bad is the FIFO Discipline for Handling Randomly Arriving Time-Critical Messages," *Proc. 1995 IEEE Int'l Workshop Factory Comm. Systems,* pp. 183-190, 1995.

[11] Y.C. Chow and W.H. Kohler, "Models for Dynamic Load Balancing in Heterogeneous Multiple Processor System," *IEEE Trans. Computers,* vol. 28, no. 5, pp. 354-361, 1979.

[12] M. Colajanni and P.S. Yu, "Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers," *Proc. 18th Int'l Conf. Distributed Computing Systems,* pp. 295-302, 1998.

[13] M. Chu, K. Fan, and S. Mahlke, "Region-Based Hierarchical Operation Partitioning for Multicluster Processors," *Proc. ACM SIGPLAN 2003 Conf. Programming Language Design and Implementation,* pp. 300-311, 2003.

[14] M.M. Deris, J.H. Abawajy, and H.M. Suzuri, "An Efficient Replicated Data Access Approach for Large-Scale Distributed Systems," *Proc. Fourth IEEE Int'l Symp. Cluster Computing and the Grid,* pp. 588-594, 2004.

[15] L. He, S.A. Jarvis, D.P. Spooner, X. Chen, and G.R. Nudd, "Hybrid Performance-Based Workload Management for Multiclusters and Grids," *IEE Proc. Software,* special issue on performance eng., vol. 151, no. 5, pp. 224-231, Oct. 2004.

[16] L. He, S.A. Jarvis, D.P. Spooner, X. Chen, and G.R. Nudd, "Dynamic Scheduling of Parallel Jobs with QoS Demands in Multiclusters and Grids," *Proc. Fifth IEEE/ACM Int'l Workshop Grid Computing (Grid 2004),* pp. 402-409, 2004.
[17] L. He, S.A. Jarvis, D.P. Spooner, and G.R. Nudd, "Optimising Static Workload Allocation in Multiclusters," *Proc. 18th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS '04),* 2004.
[18] L. He, S.A. Jarvis, D.P. Spooner, and G.R. Nudd, "Dynamic Scheduling of Parallel Real-Time Jobs by Modelling Spare Capabilities in Heterogeneous Clusters," *Proc. Fifth IEEE Int'l Conf. Cluster Computing (Cluster '03),* pp. 2-10, 2003.
[19] L. He, Z. Han, H. Jin, and L. Pang, "DAG-Based Parallel Real Time Task Scheduling Algorithm on a Cluster," *Proc. Seventh Int'l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA 2000),* pp. 437-443, 2000.
[20] B. Kao and H. Garcia-Molina, "Scheduling Soft Real-Time Jobs over Dual Non-Real-Time Servers," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 1, pp. 56-68, Jan. 1996.
[21] L. Kleinrock, *Queueing System.* John Wiley & Sons, 1975.
[22] R. Leslie and S. McKenzie, "Evaluation of Load Sharing Algorithms for Heterogeneous Distributed Systems," *Computer Comm.,* vol. 22, no. 4, pp. 376-389, 1999.
[23] Z. Liu, M.S. Squillante, and J.L. Wolf, "On Maximizing Service-Level-Agreement Profits," *Proc. Third ACM Conf. Electronic Commerce,* pp. 213-223, 2001.
[24] N.G. Shivaratri, P. Krueger, and M. Singhal, "Load Distribution for Locally Distributed Systems," *Computer,* vol. 8, no. 12, pp. 33-44, Dec. 1992.
[25] X.Y. Tang and S.T. Chanson, "Optimizing Static Job Scheduling in a Network of Heterogeneous Computers," *Proc. 29th Int'l Conf. Parallel Processing,* pp. 373-382, 2000.
[26] M.Q. Xu, "Effective Metacomputing Using LSF Multicluster," *Proc. First Int'l Symp. Cluster Computing and the Grid (CCGrid '01),* pp. 100-105, 2001.
[27] S. Zhou, "A Trace-Driven Simulation Study of Dynamic Load Balancing," *IEEE Trans. Software Eng.,* vol. 14, no. 9, pp. 1327-1341, 1988.
[28] W. Zhu and B. Fleisch, "Performance Evaluation of Soft Real-Time Scheduling on a Multicomputer Cluster," *Proc. 20th Int'l Conf. Distributed Computing Systems (ICDCS 2000),* pp. 610-617, 2000.

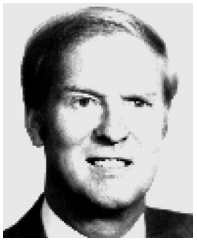
Ligang He received the Bachelor's and Master's degrees from the Huazhong University of Science and Technology, Wuhan, China. He is currently a PhD student and research associate in the High Performance Systems Group in the Department of Computer Science at the University of Warwick. His areas of interest are grid and cluster computing, parallel and distributed processing, high performance computing, real-time processing, and performance modelling and evaluation. He is the primary author on more than 20 high-quality conference and journal papers. He is also a student member of the IEEE, the IEEE Computer Society, and the ACM.

Stephen A. Jarvis is a senior lecturer in the High Performance Systems Group at the University of Warwick. He has authored more than 90 refereed publications (including three books) in the area of software and performance evaluation. While previously at the Oxford University Computing Laboratory, he worked on the development of performance tools with Oxford Parallel, Sychron Ltd., and Microsoft Research in Cambridge. He has close research ties with IBM, including current projects with the IBM T.J. Watson Research Center in New York and with IBM Hursley Park in the UK. He is also principle investigator on the recently awarded EPSRC Fundamentals of Computer Science for e-Science project (Dynamic Operating Policies for Commercial Hosting Environments), which draws on research collaboration with IBM, BT at Martlesham Heath, Hewlett Packard Research Laboratories in Bristol and Palo Alto, and the National Business to Business Centre and the University of Newcastle. Dr. Jarvis sits on a number of international program committees for high-performance computing. He is an external advisor for the Netherlands Organization for Scientific Research, coorganizer for one of the UK's high end scientific computing training centers, and manager of the Midlands e-Science Technical Forum on Grid Technologies. He is a member of the IEEE.

Daniel P. Spooner is a lecturer in the Department of Computer Science at the University of Warwick, Coventry, UK, and a member of the High Performance Systems Group. His primary research interest is the application of performance prediction techniques to improve resource scheduling in grid environments. Other interests include distributed network management architectures and high performance Web services.

Hong Jiang received the BSc degree in computer engineering in 1982 from Huazhong University of Science and Technology, Wuhan, China, the MASc degree in computer engineering in 1987 from the University of Toronto, and the PhD degree in computer science in 1991 from the Texas A&M University. Since August 1991, he has been at the University of Nebraska-Lincoln, where he is a professor and vice chair in the Department of Computer Science and Engineering. His present research interests are computer architectures, parallel/distributed computing, cluster and grid computing, computer storage systems and parallel I/O, performance evaluation, real-time systems, middleware, and distributed systems for distance education. He has more than 100 publications in major journals and international conferences in these areas and his research has been supported by the US National Science Foundation (NSF), DOD, and the State of Nebraska. Dr. Jiang is a member of the ACM, the IEEE Computer Society, and ACM SIGARCH.

Donna N. Dillenberger is a senior technical staff member, IT Optimization Architect for IBM's Systems Group, and manager of zSeries Research and IT Optimization at IBM's Watson Research Center. She joined IBM in 1988, and has worked on future hardware, mainframe operating systems, workload management, scalable java virtual machines, Web servers, stream servers, and J2EE object containers. All of her work has shipped as IBM products and impacted external standard groups: IETF, OpenGroup, and Java. She is currently working on enterprise workload management and virtualization technologies. She is a member of the IEEE.



Graham R. Nudd is chair of the Computer Science Department and head of the High Performance Systems Group at the University of Warwick. Prior to joining Warwick, Professor Nudd was head of Information Systems and Architectures at the Hughes Research Laboratories in Malibu, California. He has authored or coauthored several hundred publications and is a fellow of the Royal Academy of Engineering.