Design and Analysis of a Load Balancing Strategy in Data Grids

Xiao Qin

Department of Computer Science
New Mexico Institute of Mining and Technology
Socorro, NM 87801, Email: xqin@cs.nmt.edu

Abstract

Developing Data Grids has increasingly become a major concern to make Grids attractive for a wide range of data-intensive applications. Storage subsystems are most likely to be a performance bottleneck in Data Grids and, therefore, the focus of this paper is to design and evaluate a data-aware load-balancing strategy to improve the global usage of storage resources in Data Grids. We build a model to estimate the response time of job running at a local site or remote site. In light of this model, we can calculate slowdowns imposed on jobs in a Data Grid environment. Next, we propose a load-balancing strategy that aims to balance load of a Data Grid in such a judicious way that computation and storage resources in each site are simultaneously well utilized. We conduct experiments using a simulated Data Grid to analyze the performance of the proposed strategy. Experimental results confirm that our load-balancing strategy can achieve high performance for data-intensive jobs in Data Grid environments.

Key words: load balancing, data-intensive, datagrids, performance evaluation
PACS:

1 Introduction

In the last decade, Data Grids have increasingly become popular for a wide range of scientific and commercial applications [1]. A Data Grid can be envisioned as a collection of geographically dispersed computing and storage resources that provide a diversity of services [2][3] to fit needs of data-intensive

⋆ Article published in Future Generation Computer Systems: The Int'l Journal of Grid Computing, vol. 23, no. 1, pp. 132-137, Jan. 2007.

applications like simulation analysis tools [4] and high-energy physics applications [5][6]. The Data Grid integrates data archives stored in an array of distributed sites into a single virtual data management system [7]. Due to a widening performance gap between CPU and storage devices [8], the impact of storage resources in a Data Grid on data-intensive applications becomes highly significant. As such, it is extremely important to develop load-balancing schemes providing effective usage of global storage resources as well as that of CPU and memory.

In an effort to improve performance of storage resources in Data Grids, we propose in this paper a Data-aware Load-Balancing strategy (or DLB for short) for Data Grids. Our scheme aims at balancing load of a Data Grid in a judicious way that computation and storage resources at each site are simultaneously well utilized for a variety of data-intensive applications. After building a model to estimate the response time of job running at a local site or remote site, we conduct experiments using a simulated Data Grid to confirm that our approach can achieve high performance for data-intensive jobs running on a resource-sharing Data Grid.

The rest of the paper is organized as follows. In the section that follows, related work in the literature is briefly reviewed. Section 3 describes the system model of a Data Grid. In Section 4, we propose the data-aware load-balancing strategy. Section 5 evaluates the performance of the new strategy by comparing it with existing approaches. The paper concludes with Section 6.

2 Related Work

A handful of previous studies are geared towards leveraging data replication techniques to achieve high performance in Data Grids [3][9][10]. Existing data replication approaches in Data Grids can be divided into four camps, namely, (1) Architecture and management for data replication, (2) Data replication placement, (3) Data replication selection, and (4) Data consistency [7].

Load balancing and scheduling play a critical role in achieving high utilization of resources in Data Grids [3][11]. A large body of work has been done in the area of load balancing [12]. For example, many load balancing polices were develop to achieve high performance by improving utilization of CPU [13][14][15][16] or memory resources [17][18]. Most existing load-balancing strategies are inadequate for Data Grids supporting data-intensive applications, since the existing schemes ignore the issue of load-balancing among storage resources.

Several research efforts have been focused on design of cache and buffer tech-

niques to optimize performance of storage systems [19][20][21]. Existing cache and buffer techniques are orthogonal to our load-balancing strategies, meaning that the implementation of our strategies can achieve extra performance improvements when applied in concert with the cache and buffer schemes.

Much attention has been paid to the development of load-balancing policies for disk systems [22][23]. Although previous disk techniques can improve storage performance by fully utilizing hard drives. However, these approaches are unable to effectively handle complex workload conditions where data-intensive applications are sharing resources with other types of applications. Unlike the existing disk load-balancing schemes, our proposed strategy takes into account both computation and storage resources.

3 System Model

In this study we model a data grid as a collection of n sites, each of which is comprised of computational facilities and a storage subsystem. Let $G = S_1, S_2, ..., S_n$ denote a set of sites, where each site S_i is modeled as a vector with five parameters $S_i = (u_i, b_i, m_i, C_i, D_i)$, where u_i is the number of computational nodes, b_i is the bandwidth of the storage subsystem, m_i is the memory capacity, C_i is a set of jobs allocated to S_i, and D_i is an array of datasets accessed and generated by data-intensive applications.

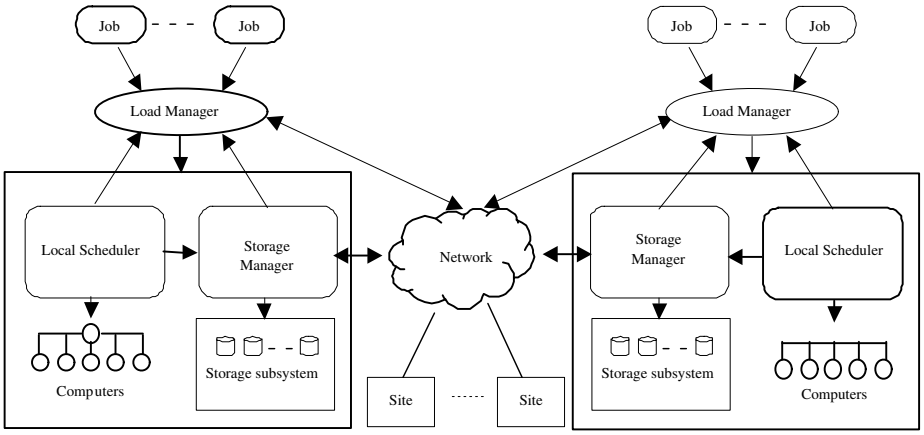

Fig. 1. Block diagram of the system model of a Data Grid.

Fig. 1 shows the block diagram of the system model of a Data Grid. Each site in the Data Grid has a load manager that accommodates submitted jobs. The load manager aims at tracking global load information by periodically exchanging load status with other load managers in the system. Depending on load balancing policies employed in the Data Grid, the load manager may

execute locally submitted jobs or transfer them to other sites. If the local site decides to have a job executed by a remote site, the job will be dispatched to the remote site through a network. If the local site is not overloaded, the job will be accommodated and handled by a local scheduler. The functionalities of storage managers include monitoring local datasets and handling dataset requests issued by local schedulers [24]. A data-intensive job can be characterized by a set of parameters, i.e., $J_j = (q_j, d_j, A_j, tc_j, td_j, y_j)$, where q_j is the number of requested nodes, l_j is the size of required datasets, and A_j represents a set of sites storing the datasets. tc_j is the required computational time, td_j is the required time spent in data accesses, y_j is the memory requirement of J_j. td_i is expressed by $\frac{d_i}{b_k}$, where d_i the amount of accessed data and b_k is the bandwidth of the kth dedicated storage subsystem.

The load manager makes use of the following two equations to quantify computational and data access load of the local site. Note that Eqs. 1 and 3 take into account heterogeneities in computation and data accesses.

$$L_i^C = \frac{max_{k=1}^{n}\left(p_k\right)}{p_i} \cdot \sum_{J_j \in C_i} \left(q_j \cdot tc_j\right), \tag{1}$$

$$L_i^M = \frac{max_{k=1}^{n}\left(m_k\right)}{m_i} \cdot \sum_{J_j \in C_i} u_j, \tag{2}$$

$$L_i^D = \frac{max_{k=1}^{n}\left(b_k\right)}{b_i} \cdot \sum_{J_j \in C_i} d_j. \tag{3}$$

An important metric used to evaluate performance of load-balancing strategies is slowdown, which is measured by the ratio between the time to execute a job in a resource-shared setting and the time to execute the same job in a dedicated environment. The slowdown of job J_j executing on site S_i can be expressed by Eq. 4,

$$s_{i,j} = \frac{w_{i,j} + tc_i + td_j \cdot \sum_{k=1}^{p_i}(pd_{i,k} \cdot k) + tp_{i,j} + tn_j \cdot \sum_{k=1}^{p_i}(pn_{i,k} \cdot k)}{tc_j + td_j + tp_j}, \tag{4}$$

where
- $w_{i,j}$ = the waiting time of J_j at S_i,
- p_i = the number of additional jobs running at S_i,
- $pd_{i,k}$ = the probability that k jobs will access data at S_i at the same time,
- $pn_{i,k}$ = the probability that k jobs at S_i will communicate at the same time,
- tn_j = the data communication time,
- $tp_{i,j}$ = the paging time of J_j at S_i, and
- tp_j = the paging time of J_j in a dedicated environment.

$tp_{i,j}$ can be expressed as $\rho_{i,j} \cdot tc_j \cdot t_p$, where $\rho_{i,j}$ is the page fault rate of J_j at S_i and t_p is the paging overhead. Similarly, tp_j is given by $\rho_j \cdot tc_j \cdot t_p$, where ρ_j

is the page fault rate in the dedicated environment. Given m jobs submitted to a Data Grid for execution, we design a load-balancing strategy to minimize the mean slowdown, which is given by $\frac{\sum_{i=1}^{m} s_i}{m}$.

4 Load Balancing in Data Grids

In this section we present a Data-aware Load-Balancing (DLB) strategy for Data Grids. In addition to allocating data-intensive jobs in a way to balance computing resources, the proposed strategy strives to balance data access load in a Data Grid. Our load-balancing strategy is outlined in Figure 2 below.

The DLB load-balancing strategy:
Input: A job J_j submitted to a local site S_i;
begin
  if $L_i^D = max_{k=1}^{n}\{L_k^D\}$ **then**
    chose a site $S_g \in A_j$ where $L_g^D = min_{S_k \in A_j}\{L_k^D\}$;
  else $g = i$;
  if $\sum_{J_k \in C_i} y_k > m_i$ **then**
    choose a site S_h, where $L_h^M = min_{k=1}^{n}\{L_k^M\}$;
  else choose a site S_h, where $L_h^C = min_{k=1}^{n}\{L_k^C\}$;
  estimate the response time $R(J_j, S_i)$ of J_j on S_i;
  estimate the response time $R(J_j, S_h, S_g)$ of J_j on S_h with data access on S_g;
  if $R(J_j, S_i) \leq R(J_j, S_h, S_g)$ **then**
    $g = i$;
    $h = i$;
  endif
  notify S_g to handle data accesses of J_j;
  allocate J_j to S_h;
  for $S_k \in A_j, S_k \neq S_g$ **do**
    notify the storage manager of S_k to maintain data consistency;
  endfor
end;

Fig. 2. The data-aware load balancing strategy (DLB)

Given a job J_j submitted to the local site S_i, the DLB strategy gives the highest priority to storage resources. This is because jobs running under data-intensive workload conditions are likely to experience waiting time on data accesses. As such, in case that data access load in the system is unbalanced, DLB manages to notify the best candidate site from list A_j to handle the data accesses of J_j. The candidate site must have the lightest load in terms of data access. After choosing the best candidate site to deal with data access, DLB balances global memory usage if the memory load of S_i exceeds its available memory amount. Thus, the memory resources is balanced by allocating J_j to a site with the lightest memory load. When data access and memory loads are

well balanced, DLB strives to improve the usage of global computing resources by evenly distributing computational load. In other words, in this case DLB migrates S_j to a remote site with the lightest computational load across the entire Data Grid. Next, the DLB strategy checks whether the response time of J_j on the remote site is less than that of J_j on the local site. In doing so, DLB guarantees that each remote execution can improve performance of the Data Grid. After J_j is allocated to the target site, the corresponding storage managers are required to maintain data consistency for J_j during the course of its execution.

Let $R(J_j, S_i)$ denote the response time of job J_j on site S_i. Let $R(J_j, S_h, S_g)$ be the response time of J_j on site S_h with data access on site S_g. The DLB strategy leverages the values of $R(J_j, S_i)$ and $R(J_j, S_h, S_g)$ to determine whether having J_j remotely executed can improve performance of the Data Grid. $R(J_j, S_i)$ can be obtained as:

$$R(J_j, S_i) = w_{i,j} + tc_i + td_j \cdot \sum_{k=1}^{p_i} (pd_{i,k} \cdot k) + tp_{i,j}. \tag{5}$$

The four terms on the right-hand side of Eq. 5 represent the waiting time, computational time, data access time, and paging time, respectively. The following variables are necessary to facilitate the derivation of $R(J_j, S_h, S_g)$, which can be written as Eq. 6.

- θ = the fixed cost of remote execution,
- b_{hg} = the bandwidth of the network link between S_h and S_g,
- p_{hg} = the number of jobs sharing the link between S_h and S_g, and
- $pn_{hg,k}$ = the probability that k jobs will use the link between S_h and S_g at the same time.

$$R(J_j, S_h, S_g) = \begin{cases} w_{i,h} + tc_i + td_j \cdot \sum_{k=1}^{p_h} (pd_{h,k} \cdot k) + tp_{h,j} + \theta & \text{if } h = g, \\ w_{i,h} + tc_i + td_j \cdot \sum_{k=1}^{p_h} (pd_{h,k} \cdot k) + tp_{h,j} & \\ \quad + \theta + \frac{d_j}{b_{hg}} \cdot \sum_{k=1}^{p_{hg}} (pn_{hg,k} \cdot k) & \text{if } h \neq g. \end{cases} \tag{6}$$

The first five terms of both the upper and the bottom line of Eq. 6 characterize the waiting time, computational time, data access time, paging time, and fixed cost of remote execution. The last term of the bottom line of Eq. 6 is the time spent on transmitting data over the network link between S_h and S_g. Let $\tau_{i,k}$ be the earliest available time of the kth computational node at site S_i, meaning that the node is used without any waste prior to $\tau_{i,k}$. We assume that $\tau_{i,1} \leq \tau_{i,2} \leq ... \leq \tau_{i,u_i}$. The waiting time $w_{i,j}$ of J_j at S_i is approximated by $w_{i,j} = \tau_{q_j}$. It is worth noting that $R(J_j, S_i)$ and $R(J_j, S_h, S_g)$ are computed by the DLB strategy at runtime. Since the calculations of Eqs. 5 and 6 are efficient, they can be obtained without imposing too much overhead on a load manager.

Response times $R(J_j, S_i)$ and $R(J_j, S_h, S_g)$ derived above rely on the probability $pd_{i,k}$ that k jobs are accessing the storage resources at site S_i at the same time. In what follows, we derive the probability $pd_{i,k}$. Let α_v be the probability that job J_v is storing or retrieving data. α_v is measured by the percentage of time spent in accessing the storage subsystem. Thus, we have

$$\alpha_v = \frac{td_v}{tc_v + td_v + tn_v}. \tag{7}$$

The probability that J_v is using computational and communication resources can be expressed as $1 - \alpha_v$. Let $\Phi_m = \{J_{\gamma_1}, J_{\gamma_2}, ..., J_{\gamma_m}\}$ be m jobs accessing the storage subsystem of site S_i during the execution of job J_j. To facilitate the derivation of $pd_{i,k}$, we have to calculate the probability that $k-1$ jobs in Φ_m are using the storage resource at the same time. As such, we first introduce an array of subsets of Φ_m, i.e., $\Phi_{m,k,1}, \Phi_{m,k,2}, ..., \Phi_{m,k,f}$, where $f = \binom{m}{k-1}$, each subset contains $k-1$ jobs in set Φ_m, and any pair of two subsets are not identical to each other. Let $p_{m,k,x}$ be the probability that jobs in $\Phi_{m,k,x}$ are accessing data at the same time while jobs in $\bar{\Phi}_{m,k,x} = \Phi_m - \Phi_{m,k,x}$ are either computing or communicating. Given a subset $\Phi_{m,k,x} (1 \leq x \leq f)$, we can obtain the probability $p_{m,k,x}$ as

$$p_{m,k,x} = \prod_{J_v \in \Phi_{m,k,x}} \alpha_v \cdot \prod_{J_v \in \bar{\Phi}_{m,k,x}} (1 - \alpha_v). \tag{8}$$

Based on the above probability for each subset of jobs, one can estimate the probability $pd_{i,k}$ using Eq. 10 below.

$$\begin{aligned} pd_{i,k} &= \sum_{x=1}^{f} p_{m,k,x} \\ &= \sum_{x=1}^{f} \left(\prod_{J_v \in \Phi_{m,k,x}} \alpha_v \cdot \prod_{J_v \in \bar{\Phi}_{m,k,x}} (1 - \alpha_v) \right) \\ &= \sum_{x=1}^{f} \left(\prod_{J_v \in \Phi_{m,k,x}} \frac{td_v}{tc_v + td_v + tn_v} \cdot \prod_{J_v \in \bar{\Phi}_{m,k,x}} \left(1 - \frac{td_v}{tc_v + td_v + tn_v} \right) \right). \end{aligned} \tag{9}$$

For a special case where jobs in Φ_m are identical, (i.e., $\forall J_v \in \Phi_m : \alpha_v = \alpha$), $pd_{i,k}$ can be simplified as

$$pd_{i,k} = \binom{m}{k-1} \alpha^{k-1} (1 - \alpha)^{m-k+1}. \tag{10}$$

Now we are positioned to derive the probability $pn_{hg,k}$ that k jobs will use the link between S_h and S_g at the same time. Let β_v be the probability that job J_v tries to communicate. We can obtain β_v by the percentage of time spent in communication. Thus,

$$\beta_v = \frac{tn_v}{tc_v + td_v + tn_v}. \tag{11}$$

Let $\Phi^*_{m^*} = \left\{J_{\gamma^*_1}, J_{\gamma^*_2}, ..., J_{\gamma^*_{m^*}}\right\}$ be m^* jobs communicating at the same time through the link between S_h and S_g. To determine the probability that $k-1$ jobs in Φ will try to communicate at the same time, we define f^* subsets of $\Phi^*_{m^*}$, i.e., $\Phi^*_{m^*,k,1}, \Phi^*_{m^*,k,2}, ..., \Phi^*_{m^*,k,f^*}$, where $f^* = \binom{m^*}{k-1}$. Each subset contains $k-1$ jobs in $\Phi^*_{m^*}$, and any pair of two subsets are not identical. We denote $p^*_{m^*,k,x}$ be the probability that jobs in $\Phi^*_{m^*,k,x}$ are communicating at the same time while other jobs in $\bar{\Phi}^*_{m^*,k,x} = \Phi^*_{m^*} - \Phi^*_{m^*,k,x}$ are either computing or accessing data. Thus, one can obtain $p^*_{m^*,k,x}$ as

$$p^*_{m^*,k,x} = \prod_{J_v \in \Phi^*_{m^*,k,x}} \beta_v \cdot \prod_{J_v \in \bar{\Phi}^*_{m^*,k,x}} (1-\beta_v). \tag{12}$$

In light of Eq. 12, we quantify the probability $pn_{hg,k}$ as follows.

$$\begin{aligned} pn_{hg,k} &= \sum_{x=1}^{f^*} p^*_{m^*,k,x} \\ &= \sum_{x=1}^{f^*} \left(\prod_{J_v \in \Phi^*_{m^*,k,x}} \beta_v \cdot \prod_{J_v \in \bar{\Phi}^*_{m^*,k,x}} (1-\beta_v) \right) \\ &= \sum_{x=1}^{f^*} \left(\prod_{J_v \in \Phi^*_{m^*,k,x}} \frac{tn_v}{tc_v + td_v + tn_v} \cdot \prod_{J_v \in \bar{\Phi}^*_{m^*,k,x}} \left(1 - \frac{tn_v}{tc_v + td_v + tn_v}\right) \right). \end{aligned} \tag{13}$$

If all the jobs in $\Phi^*_{m^*}$ are identical in terms of communication patterns, (i.e., $\forall J_v \in \Phi^*_{m^*} : \beta_v = \beta$), $pn_{hg,k}$ in Eq. 6 can be expressed as

$$pd_{i,k} = \binom{m^*}{k-1} \beta^{k-1} (1-\beta)^{m^*-k+1}. \tag{14}$$

5 Performance Evaluation

Now we evaluate the performance of the load-balancing strategy presented in Section 4 by conducting trace-driven simulations. We experimentally compare our DLB strategy with two existing schemes: CM [25][26], and WAL [27]. The CM policy is focused on effective usage of global CPU and memory resources, whereas the WAL scheme balances load using a weighted average of required resource loads to quantify load at each site.

We simulated a Data Grid of six sites by extending a distributed system simulator developed by Harchol-Balter and Downey [14]. In the simulated Data Grid, each site consists of 128 computational nodes and a storage subsystem. We modified the traces used in [14][26] by adding storage access requests to each job, whose number of requested nodes is randomly generated with a uniform distribution in $[1, 64]$. The sizes of files accessed by jobs are generated uniformly at random from 10MB to 100MB. Each file has three replicas evenly distributed among the sites in the Data Grid. When any one of the disks in a storage subsystem is full, the least recently used files are automatically removed by the storage manager.

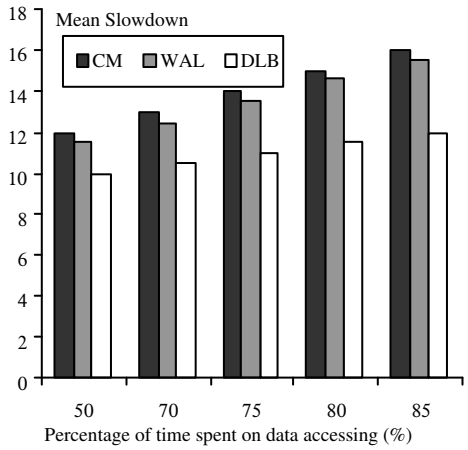

Fig. 3. Mean slowdowns as a function of the percentage of time spent on data accessing.

Fig. 3 plots slowdowns of the three load-balancing strategies when the percentage of time spent on data accessing is varied from 50% to 85%. To fairly compare the performance of the three strategies, we generate traces with a good-mix of memory-intensive and data-intensive jobs. Fig. 3 demonstratively shows that regardless of which load-balancing strategy is employed, the mean slowdowns increase with the increasing percentage of time spent on accessing storage subsystems. This result is expected because high data access load leads to longer waiting times on storage resources due to a high utilization of storage subsystems. The result further shows that the proposed DLB strategy

substantially outperforms the two existing load-balancing schemes. For example, DLB reduces the slowdowns by up to 24.9% and 29.2% (with averages of 21.1% and 22.4%). This is because DLB improves the global usage of the storage resources, which dominate the Data Grid's performance in cases where the data access load is high. In addition, we measured the overall response times of the jobs. Our experimental results show that DLB is able to significant reduce response times of data-intensive jobs. Due to the space limit, we omit the results for the response times.

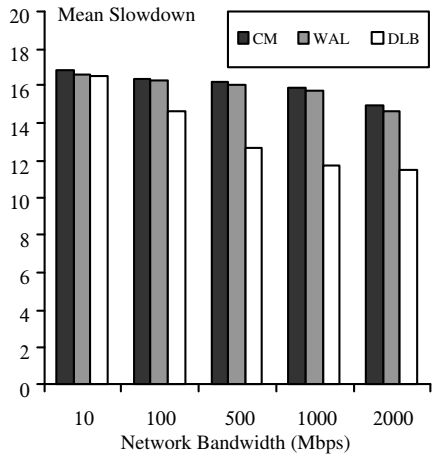

Fig. 4. Mean slowdowns as a function of the network bandwidth.

Network bandwidth plays a critical role in the overall performance of a Data Grid. Fig. 4 reveals impact of network bandwidth on the mean slowdowns of the three evaluated load-balancing schemes. In this experiment, we vary the network bandwidth from 10Mbps to 2Gbps. The first observation made from Fig. 4 is that the mean slowdown decreases as the network bandwidth increases. We attribute this result to the fact that the Data Grid with a high network bandwidth gives rise to low load-balancing overhead, which in turn makes it possible for data-intensive jobs to benefit substantially from load balancing. A second observation drawn from Fig. 4 is that the DLB strategy is more sensitive to the network bandwidth than the other two alternatives. The reason behind this result is that the DLB strategy leverage the high bandwidth network to noticeably reducing remote data access cost, which contributes a significant fraction of response times of data-intensive jobs. The implication of this result is that our DLB load-balancing strategy can greatly improve the overall performance of a Data Grid with high-speed network interconnections.

6 Conclusion

In the past five years, Data Grids have increasingly become an attractive computing platform for a variety of data-intensive applications. In recognition that storage subsystems of Data Grids are most likely to be a performance bottleneck, we developed a data-aware load-balancing strategy to improve the global usage of storage resources in the Data Grids. We built a model to estimate the response time of job running at a local site or remote site. With this model in place, we can calculate slowdowns imposed on jobs in a Data Grid environment. The proposed strategy aims at balancing load among sites of a Data Grid in a judicious way to improve the global usage of CPU, memory, and storage resources. To evaluate effectiveness of our novel load-balancing strategy, we compared it with two existing approaches that ignore storage resources in the process of load balancing. Our empirical results confirm that the proposed strategy can achieve high performance for data-intensive jobs under diverse workload conditions.

There are, of course, several open issues that need to be addressed in load balancing for Data Grids. First, the performance of our load-balancing strategy in Data Grids depends to some extent on data movements. Reducing overhead incurred by data movements can ultimately improve performance of Data Grids. We plan to investigate a new approach to moving data at a time when source and destination sites are lightly loaded. It is expected that the new approach can achieve improved performance without dramatically affecting response times of data-intensive jobs running on Data Grids. Second, data placement in the context of load-balancing in Data Grids has received little attention in the past years. In the future work we will develop a framework that seamlessly integrates data placement techniques with our data-aware load-balancing strategy.

7 Acknowledgments

The work reported in this paper was supported in part by the New Mexico Institute of Mining and Technology under Grant 103295 and by Intel Corporation under Grant 2005-04-070.

References

[1] B. Tierney, W. Johnston, J. Lee, M. Thompson, A data intensive distributed computing architecture for grid applications, Future Generation Computer

Systems 16 (5) (2000) 473–481.

[2] J.-P. Martin-Flatin, P. V.-B. Primet, High-speed network and services for data-intensive grids: The datatag project, Future Generation Computer Systems 21 (4) (2005) 439–442.

[3] M. Tang, B.-S. Lee, X. Tang, C.-K. Yeo, The impact of data replication on job scheduling performance in the data grid, Future Generation Computer Systems 22 (3) (2006) 254–268.

[4] A. Breckenridge, L. Pierson, S. Sanielevici, J. Welling, R. Keller, U. Woessner, J. Schulze, Distributed, on-demand, data-intensive and collaborative simulation analysis, Future Generation Computer Systems 19 (6) (2003) 849–859.

[5] M. S. Prez, J. Carretero, F. Garca, J. M. Pea, V. Robles, Mapfs: A flexible multiagent parallel file system for clusters, Future Generation Computer Systems 22 (5) (2006) 620–632.

[6] W. Allcock, et al., Grid-enabled particle physics event analysis: experiences using a 10 gb, high-latency network for a high-energy physics application, Future Generation Computer Systems 19 (6) (2003) 983–997.

[7] X. Qin, H. Jiang, Data grids: Supporting data-intensive applications in wide area networks, in: High Performance Computing: Paradigm and Infrastructure, 2005.

[8] J. Hennessy, D. Patterson, in: Computer Architecture: A Quantitative Approach, 2000.

[9] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, in: Proc. Network Storage Symp., 2000.

[10] A. Samar, H. Stockinger, Grid data management pilot (gdmp): a tool for wide area replication, in: Proc. Int'l Conf. Applied Informatics, 2000.

[11] K. Ranganathan, I. Foster, Decoupling computation and data scheduling in distributed data-intensive applications, in: Proc. 11th IEEE Int'l Conf. High Performance Distibuted Computing, 2002, p. 352.

[12] J. Cao, D. P. Spooner, S. A. Jarvis, G. R. Nudd, Grid load balancing using intelligent agents, Future Generation Computer Systems 21 (1) (2005) 135–149.

[13] F. Zhang, A. Mawardi, E. Santos, R. Pitchumani, L. E. Achenie, Examination of load-balancing methods to improve efficiency of a composite materials manufacturing process simulation under uncertainty using distributed computing, Future Generation Computer Systems 22 (5) (2006) 571–587.

[14] M. Harchol-Balter, A. Downey, Exploiting process lifetime distributions for load balancing, ACM Transactions on Computer Systems 15 (3) (1997) 253–285.

[15] C. Hui, S. Chanson, Improved strategies for dynamic load sharing, IEEE Concurrency 7 (3).

[16] P. D. Michailidis, K. G. Margaritis, Performance evaluation of load balancing strategies for approximate string matching application on an mpi cluster of heterogeneous workstations, Future Generation Computer Systems 19 (7) (2003) 1075–1104.

[17] A. Acharva, S. Setia, Availability and utility of idle memory in workstation clusters, in: Proc. ACM SIGMETRICS Conf. Measuring and Modeling of Computer Systems, 1999.

[18] G. Voelker, Managing server load in global memory systems, in: Proc. ACM SIGMETRICS Conf. Measuring and Modeling of Computer Systems, 1997.

[19] X. Ma, M. Winslett, J. Lee, S. Yu, Faster collective output through active buffering, in: Proc. Int'l Symp. on Parallel and Distributed Processing, 2002.

[20] X. Qin, H. Jiang, Y. Zhu, D. Swanson, Dynamic load balancing for I/O- and memory-intensive workload in clusters using a feedback control mechanism, in: Proc. 9th Int'l Euro-Par Conf. Parallel Processing, Klagenfurt, Austria, 2003.

[21] B. Forney, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, Storage-aware caching: Revisiting caching for heterogeneous storage systems, in: Proc. Symp. File and Storage Technology, Monterey, California, USA, 2002.

[22] P. Scheuermann, G. Weikum, P. Zabback, Data partitioning and load balancing in parallel disk systems, The VLDB Journal (1998) 48–66.

[23] L. Lee, P. Scheauermann, R. Vingralek, File assignment in parallel I/O systems with minimal variance of service time, IEEE Trans. Computers 49 (2) (2000) 127–140.

[24] T. Xie, X. Qin, Saha: A scheduling algorithm for security-sensitive jobs on data grids, in: Proc. IEEE/ACM 6th Int'l Symp. Cluster Computing and the Grid, 2006.

[25] L. Xiao, X. Zhang, Y. Qu, Effective load sharing on heterogeneous networks of workstations, in: Proc. Int'l Symp. Parallel and Distributed Processing, 2000.

[26] X. Zhang, Y. Qu, L. Xiao, Improving distributed workload performance by sharing both cpu and memory resources, in: Proc. 20th Int'l Conf. Distributed Computing Systems, 2000.

[27] M. Surdeanu, D. Modovan, S. Harabagiu, Performance analysis of a distributed question/answering system, IEEE Trans. on Parallel and Distributed Systems 13 (6) (2006) 579–596.