Optimizing Server Placement for QoS Requirements in Hierarchical Grid Environments

Chien-Min Wang[1], Chun-Chen Hsu[2], Pangfeng Liu[2], Hsi-Min Chen[3], and Jan-Jan Wu[1]

[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
{cmwang,wuj}@iis.sinica.edu.tw
[2] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.
{d95006,pangfeng}@csie.ntu.edu.tw
[3] Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan, R.O.C.
seeme@selab.csie.ncu.edu.tw

Abstract. This paper focuses on two problems related to QoS-aware I/O server placement in hierarchical Grid environments. Given a hierarchical network with requests from clients, the network latencies of links, constraints on servers' capabilities and the service quality requirement, the solution to the *minimum server placement problem* attempts to place the minimum number of servers that meet both the constrains on servers' capabilities and the service quality requirement. As our model considers both the different capabilities of servers and the network latencies, it is more general than similar works in the literatures. Instead of using a heuristic approach, we propose an optimal algorithm based on dynamic programming to solve the problem. We also consider the *optimal service quality problem*, which tries to place a given number of servers appropriately so that the maximum expected response time is minimized. We prove that an optimal server placement can be achieved by combining the dynamic programming algorithm with a binary search on the service quality requirement. The simulation results clearly show the improvement in the number of servers and the maximum expected response time.

1 Introduction

Grid technologies enable scientific applications to utilize a wide variety of distributed computing and data resources [1]. A Data Grid is a distributed storage infrastructure that integrates distributed, independently managed data resources. It addresses the problems of storage and data management, data transfers and data access optimization, while maintaining high reliability and availability of the data. In recent years, a number of Data Grid projects [2,3] have emerged in various disciplines.

One of the research issues in Data Grid is the efficiency of data access. One way of efficient data access is to distribute multiple copies of a file across different server sites in the grid system. Researches [4,5,6,7,8,9] have shown that file replication can improve the performance of the applications.

The existing works focus on how to distribute the file replicas in Data Grid in order to optimize different criteria such as I/O operation costs [5], mean access latencies [8] and bandwidth consumption [9]. However, few works use the quality of services as an performance metric of Data Grid. We believe the service quality is also an important performance metric in Data Grid due to the dynamic nature in the grid environment. In [10,11], quality of service is considered. Those works, however, fail to take the heterogeneity of servers' capabilities into consideration. That is, in those works, servers are assumed to be able to serve all I/O requests it received. This assumption omits one of the characteristics in grid computing infrastructure: the heterogeneity of its nature. In an early work by Wang [12], they considered the servers' capabilities when minimizing the number of servers.

In this paper, we focus on two QoS-aware I/O server placement problems in hierarchical Grid environments which consider the service quality requirement, the capabilities of servers and the network latencies. As our model consider both the different capabilities of servers and the network latencies, it is more general than similar works in the literatures. The *minimum server placement problem* asks how to place the minimum number of servers to meet both the constrains on servers' capabilities and the service quality requirement. We propose an optimal algorithm based on dynamic programming to solve this problem. We also consider the *optimal service quality problem*, which tries to place a given number of servers appropriately so that the maximum expected response time is minimized. We prove that such a server placement can be achieved by combining the dynamic programming algorithm with a binary search on the maximum expected response time of servers. The experimental results clearly show the improvement in the number of servers and the maximum expected response time.

2 The System Model

In this paper we use a hierarchical Grid model, one of the most common architectures in current use [7,9,10,11,12,13]. Consider Fig. 1 as an example. Given a tree $T = (V, E)$, V is the set of sites and $E \in V \times V$ represents network links between sites. A distance d_{uv} associated with each edge $(u, v) \in E$ represents the latency of the network link between sites u and v. We may further extend the definition of d_{uv} as the latency of a shortest path between any two sites u and v.

Leaf nodes represent client sites that send out I/O requests. The root node is assumed to be the I/O server that stores the master copies of all files. Without loss of generality, we assume that the root node is the site 0. Intermediate nodes can be either routers for network communications or I/O servers that store file replicas. We assume that, initially, only one copy (i.e., the master copy) of a file exists at the root site, as in [9,10,11,12,13]. Let T_i be the sub-tree rooted at node i.

Associated with each client site i, there is a parameter r_i that represents the arrival rate of read requests for client site i. A data request travels upward from

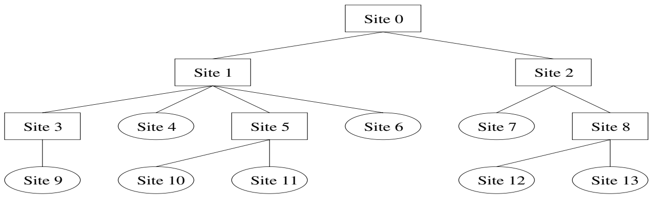

Fig. 1. The hierarchical Grid model

a client site and passes through routers until it reaches an I/O server on the path. Upon receiving the request, the I/O server sends the requested file back to the client site if it owns a copy of the requested file. Otherwise, it forwards the request to its parent server. This process continues up the hierarchy recursively until a node that has the requested file is encountered or the root node is reached. The network latency of a I/O request from a client site to a server site can be computed as the sum of the network latencies of all intermediate links between both sites. The root server might update the contents of a file. For each update, corresponding update requests are sent to the other I/O servers to maintain file consistency. Let u be the arrival rate of update requests from the root server.

For each server site j, μ'_j and λ'_j are represented as the service rate and the arrival rate of I/O requests of server site j respectively. λ'_j can be computed as: $\lambda'_j = \sum_{i \in C_j} r_i + u$, where C_j is the set of clients served by server site j. We assume each server in the grid system is a M/M/1 queueing system. Thus, the expected waiting time at server j will be $1/(\mu'_j - \lambda'_j) = 1/(\mu'_j - u - \sum_{i \in C_j} r_i)$. To simplify the notations, we will use $\mu_j = \mu'_j - u$ and $\lambda_j = \sum_{i \in C_j} r_i$ as the service rate and the arrival rate of server site j throughout this paper.

μ_j and λ_j will be used to decide the expected response times of requests it served. Suppose the I/O requests from site i are served by server j. The expected response time of a request from site i can be defined as the sum of the network latencies in the path and the server j's expected waiting time, i.e., $d_{ij} + \frac{1}{\mu_j - \lambda_j}$.

Given the service quality requirement t, a server site j must satisfy the following conditions: (1) the arrival rate of all requests it served is less than its service rate, i.e., $\lambda_j < \mu_j$ and (2) the expected response times of all requests it served are less than or equal to t, i.e., $max_{i \in C_j}\{d_{ij} + \frac{1}{\mu_j - \lambda_j}\} \leq t$, where C_j is the set of clients served by server site j. Let the expected response time of server$j$ be the maximum expected response time of requests it served.

3 The Minimum Server Placement Problem

In this section, we formally define the minimum server placement problem and introduce our optimal algorithm to this problem. Our first problem is to place the minimum number of I/O servers that will satisfy capability constrains of servers as well as the service quality requirement from clients.

Definition 1. *Given the network topology, network latencies, request arrival rates, I/O service rates and the service quality requirement, the minimum server placement problem tries to place the minimum number of servers such that the expected response time of any request is less than or equal to the service quality requirement.*

Before introducing the optimal algorithm, we first give definitions on three basic functions as follows:

Definition 2. *Let $\lambda(i, m, d, t)$ be the minimum arrival rate of requests that reach node i among all the server placements that meet the following three conditions.*

1. *At most m servers are placed in $T_i - \{i\}$*
2. *The expected response time of any request served by these servers must be less than or equal to t.*
3. *If requests that reach node i exist, the maximum latency of these requests to node i must be less than or equal to d.*

Definition 3. *Let $\omega(i, m, d, t)$ be the minimum arrival rate of leakage requests that pass through node i among all the server placements that meet the following three conditions.*

1. *At most m servers are placed in T_i.*
2. *The expected response time of any request served by these servers must be less than or equal to t.*
3. *If leakage requests that pass through node i exist, the maximum latency of these leakage requests to node i must be less than or equal to d.*

Definition 4. *$\Omega(i, m, d, t)$ is an optimal server placement that meets all the requirements for $\omega(i, m, d, t)$.*

Leakage requests that pass through node i are those requests generated by leaf nodes in the sub-tree rooted at node i, but not served by servers in that sub-tree. Such requests must be served by a server above node i in the hierarchy. Hence, it is desirable to minimize the arrival rate of these leakage requests. Depending on the server placement, the arrival rate of leakage requests may changes. $\omega(i, m, d, t)$ represents the minimum arrival rate of leakage requests among all possible server placements that satisfy the above three conditions while $\Omega(i, m, d, t)$ represents an optimal server placement. If no server placement satisfy the above three conditions, $\omega(i, m, d, t)$ simply returns null. Let n be the number of nodes in the grid system. By definition, we can derive the following lemmas.

Lemma 1. *$\omega(i, m_1, d, t) \leq \omega(i, m_2, d, t)$ for any node i, $m_1 \geq m_2 \geq 0, d \geq 0$ and $t \geq 0$.*

Lemma 2. *$\omega(i, m, d, t_1) \leq \omega(i, m, d, t_2)$ for any node i, $m \geq 0, d \geq 0$ and $t_1 \geq t_2 \geq 0$.*

Lemma 3. *$\omega(i, m, d_1, t) \leq \omega(i, m, d_2, t)$ for any node i, $m \geq 0, d_1 \geq d_2 \geq 0$ and $t \geq 0$.*

Lemma 4. *If $\omega(i, m, d_1, t) = 0$ for some d_1, then $\omega(i, m, d, t) = 0$ for any $d \geq 0$.*

Based on the above lemmas, theorems for computing the minimum arrival rate of leakage requests can be derived. We show that it can be computed in a recursive manner.

Theorem 1. *If node i is a leaf node, then $\omega(i, m, d, t) = \lambda_i$ and $\Omega(i, m, d, t)$ is an empty set for $0 \leq m \leq n$, $d \geq 0$ and $t \geq 0$.*

Proof. Since a leaf node cannot be a server, all requests generated by a client site will travel up the tree toward the leaf node's parent. In addition, the latency to node i must be 0. By definition, $\omega(i, m, d, t) = \lambda_i$ and $\Omega(i, m, d, t)$ is an empty set for $0 \leq m \leq n$, $d \geq 0$ and $t \geq 0$. ■

Theorem 2. *For an intermediate node i with two child nodes, j and k, we can derive:*

$$\begin{aligned}
\lambda(i, m, d, t) &= min_{0 \leq r \leq m}\{\omega(j, r, d - d_{ji}, t) + \omega(k, m - r, d - d_{ki}, t)\} \\
\omega(i, m, d, t) &= 0 \textit{ if there exists } 0 \leq d' \leq t \textit{ such that} \\
&\qquad \lambda(i, m - 1, d', t) + 1/(t - d') \leq \mu_i. \\
\omega(i, m, d, t) &= \lambda(i, m, d, t), \textit{ otherwise.}
\end{aligned}$$

Proof. For node i, there are two possibilities for an optimal placement of at most m servers:

Case 1: A server is placed on node i. At most $m - 1$ servers can be placed on T_j and T_k. Suppose that, in an optimal server placement, there are p servers on T_j and q servers on T_k, as shown in Fig. 2(a). Obviously, we have $0 \leq p, q \leq m - 1$

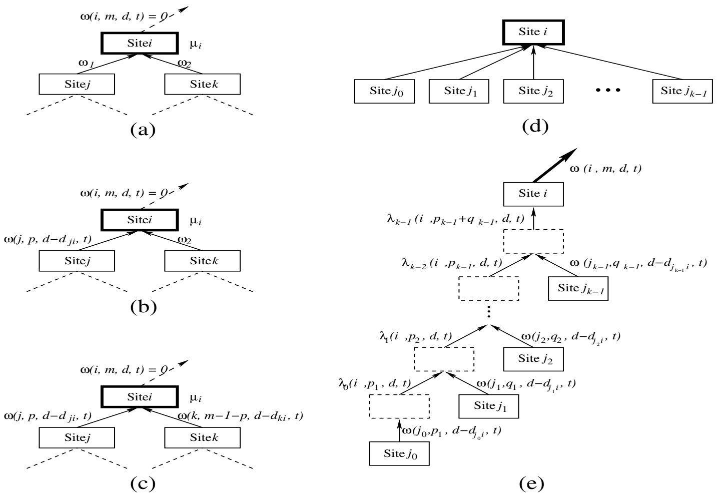

Fig. 2. (a), (b) and (c) illustrate the concept of Theorem 2. (d) and (e) illustrate the basic concept of Theorem 3.

and $p + q \leq m - 1$. Without loss of generality, we may assume the arrival rates of leakage requests from node j and node k are ω_1 and ω_2 and the maximum latencies of their leakage requests are d_1 and d_2, respectively. The maximum latency of requests that reach node i is assumed to be d'.

Next, we show that another optimal server placement can be generated by substituting the placement of p servers on T_j with $\Omega(j, p, d' - d_{ji}, t)$ as shown in Fig. 2(b). If $\omega_1 \neq 0$, then $d' \geq d_1 + d_{ji}$. We can derive

$$\omega_1 \geq \omega(j, p, d_1, t) \geq \omega(j, p, d' - d_{ji}, t)$$

After the substitution, the arrival rate of requests that reach node i can be reduced while the maximum latency of requests remains unchanged. Thus, it is also an optimal server placement. On the other hand, if $\omega_1 = 0$, we can derive

$$0 = \omega_1 = \omega(j, p, d_1, t) = \omega(j, p, d' - d_{ji}, t)$$

In this case, it is also an optimal server placement. Therefore, another optimal server placement can be generated by substituting the placement of p servers on T_j with $\Omega(j, p, d' - d_{ji}, t)$. Similarly, we can show that another optimal server placement can be generated by replacing the placement of q servers on T_k with $\Omega(k, m - 1 - p, d' - d_{ki}, t)$ as shown in Fig. 2(c).

$$\omega_2 \geq \omega(k, q, d_2, t) \geq \omega(k, q, d' - d_{ki}, t) \geq \omega(k, m - 1 - p, d' - d_{ki}, t) \text{ if } \omega_2 \neq 0$$

$$0 = \omega_2 = \omega(k, q, d_2, t) = \omega(k, q, d' - d_{ki}, t) = \omega(k, m - 1 - p, d' - d_{ki}, t) \text{ if } \omega_2 = 0$$

By assumption, the maximum expected response time of leakage requests that reach node i is less than or equal to t. In other words, $d' + 1/(\mu_i - \omega_1 - \omega_2) \leq t$. Accordingly, we an derive

$$\begin{aligned}\mu_i &\geq \omega_1 + \omega_2 + 1/(t - d') \\ &\geq \omega(j, p, d' - d_{ji}, t) + \omega(k, m - 1 - p, d' - d_{ki}, t) + 1/(t - d') \\ &\geq \lambda(i, m - 1, d', t) + 1/(t - d')\end{aligned}$$

Therefore, there exists $0 \leq d' \leq t$ such that $\lambda(i, m - 1, d', t) + 1/(t - d') \leq \mu_i$. In this case, Fig. 2(c) is an optimal server placement and $\omega(i, m, d, t) = 0$. This completes the proof of Case 1.

Case 2: No server is placed on node i. Consequently, at most m servers are placed on T_j and T_k. Obviously, we have $0 \leq p, q \leq m$ and $p + q \leq m$. Suppose that, in an optimal server placement, there are p servers on T_j and q servers on T_k. Without loss of generality, we may assume the arrival rates of leakage requests from node j and node k are ω_1 and ω_2 and their maximum latencies are d_1 and d_2, respectively. The maximum latency of requests that reach node i is assumed to be d. Similar to the proof of Case 1, the optimal arrival rate of leakage requests can be computed as

$$\begin{aligned}\omega(i, m, d, t) &= \omega_1 && + \omega_2 \\ &\geq \omega(j, p, d_1, t) && + \omega(k, q, d2, t) \\ &\geq \omega(j, p, d - d_{ji}, t) && + \omega(k, q, d - d_{ki}, t) \\ &\geq \omega(j, p, d - d_{ji}, t) && + \omega(k, m - 1 - p, d - d_{ki}, t) \\ &\geq \lambda(i, m, d, t)\end{aligned}$$

Since it is an optimal server placement, all the equalities must hold. Therefore, this theorem holds for Case 2. Since an optimal server placement must be one of the two cases, this completes the proof of this theorem. ■

Theorem 3. *For an intermediate node i with k child nodes $j_0, \ldots, j_{k-1}$, the minimum arrival rate of leakage requests that pass through node i can be computed iteratively as follows:*

$$\begin{aligned}
\lambda_0(i,m,d,t) &= \omega(j_0, m, d - d_{j_0 i}, t) \\
\lambda_q(i,m,d,t) &= min_{0 \le r \le m}\{\lambda_{q-1}(i,r,d,t) + \omega(j_q, m-r, d-d_{j_q i}, t)\}, \\
&\quad 1 \le q \le k-1, \\
\omega(i,m,d,t) &= 0 \textit{ if there exists } 0 \le d' \le t \textit{ such that} \\
&\quad \lambda_{k-1}(i, m-1, d', t) + 1/(t-d') \le \mu_i \\
\omega(i,m,d,t) &= \lambda_{k-1}(i,m,d,t), \textit{otherwise}
\end{aligned}$$

Proof. Fig. 2(d) and 2(e) illustrate the basic concept of this theorem. To find an optimal server placement, we can view an intermediate node with k child nodes in Fig. 2(d) as the sub-tree in Fig. 2(e). Then, the minimum arrival rate of leakage requests can be computed recursively along the sub-tree. As the detailed proof of this theorem is similar to that of Theorem 3, it is omitted here. ■

Theorem 4. *The minimum number of I/O servers that meet their constraints can be obtained by finding the minimum m such that $\omega(0, m, 0, t) = 0$.*

Corollary 1. *Let m' be the minimum number of servers found by the dynamic programming algorithm. m' grows nondecreasingly when the service quality requirement t decreases.*

Based on Theorems 1 to 3, we can compute the minimum arrival rates of leakage requests that start from leaf nodes and work toward the root node. After the minimum arrival rate of leakage requests that reach the root node has been computed, the minimum number of I/O servers that meet their constraints can be computed according to Theorem 4. The proposed algorithm is presented in Fig. 3.

In the first line of the algorithm, we sort all nodes according to their distances to the root node in decreasing order. This ensures that child nodes will be computed before their parents so that Theorems 1 to 3 can be correctly applied. The execution time of this step is $O(n \log n)$. The loop in line 2 iterates over every node in the system. Note that there are at most n values on the maximum latency to some node i. Thus, for each leaf node, it takes $O(n^2)$ execution time in line 4. For an intermediate node that has k child nodes, it takes $O(n^3)$ execution time in line 9, and iterates $k-1$ times in line 8. This results in $O(kn^3)$ execution time for lines 8 to 10. Lines 11 to 16 also take $O(n^2)$ execution time. Consequently, the complexity of lines 3 to 16 is $O(kn^3)$ and the complexity of the whole algorithm is $O(n^4)$, where n is the number of nodes in the Grid system. The complexity can be further reduced to $O(p^2n^2)$, where p is the minimum number of servers, by computing $\omega(i,m,d,t)$ incrementally from $m = 0$ to $m = p$.

```
Algorithm Minimum_Leakage
Input:   1. the arrival rate $\lambda_i$ for all leaf nodes.
         2. the service rate $\mu_i$ for all intermediate nodes.
         3. the network latency $d_{ji}$
         4. the service quality requirement $t$.
Output:  the minimum arrival rate $\omega(i, m, d, t)$ for $0 \leq i, m \leq n$.
Procedure:
 1. sort all nodes according to their distance to the root node in decreasing order.
 2. for each node $i$ do
 3.   if node $i$ is a leaf node then
 4.     compute $\omega(i, m, d, t) = \lambda_i$ for $0 \leq m \leq n$
 5.   else
 6.     let the child nodes of node $i$ be nodes $j_0, \ldots, j_{k-1}$
 7.     compute $\lambda_0(i, m, d, t) = \omega(j_0, m, d - d_{j_0 i}, t)$, $0 \leq m \leq n$
 8.     for $q$ from 1 to $k - 1$ do
 9.       $\lambda_q(i, m, d, t) = min_{0 \leq r \leq m}\{\lambda_{q-1}(i, r, d, t) + \omega(j_q, m - r, d - d_{j_q i}, t)\}$, $0 \leq m \leq n$
10.     endfor
11.     for $m$ from 0 to $n$ do
12.       if exists $d'$, $0 \leq d' \leq t$, such that $\lambda_{k-1}(i, m - 1, d', t) + 1/(t - d') \leq \mu_i$
13.         $\omega(i, m, d, t) = 0$
14.       else
15.         $\omega(i, m, d, t) = \lambda_{k-1}(i, m, d, t)$
16.     endfor
17.   endif
18. endfor
```

Fig. 3. An optimal algorithm for the minimum server placement problem.

4 The Optimal Service Quality Problem

In this section, we try to place a given number of servers appropriately so that the maximum expected response time of servers is minimized. We call this the *optimal service quality problem.*

Definition 5. *Given the network topology, request arrival rates, service rates and network latencies of links, the optimal service quality problem aims at placing a given number of I/O servers so that the maximum expected response time of the Grid system is minimized.*

Let m be the number of servers to be placed. We aim to place m servers such that the maximum expected response time is minimized. To achieve this goal, we can perform a binary search on the service quality requirement t. Given a service quality requirement t, we can use the dynamic programming algorithm described in Section 3 to find an optimal server placement such that the maximum expected response time of servers is less or eqaul to t. Let the minimum number of servers be m'. If $m' > m$, according to Corollary 1, we cannot find a placement of m servers whose maximum expected response time is less than or equal to t. Therefore, when $m' > m$, we need to increase t to find a server placement with m servers and, when $m' < m$, we may decrease t to find if a better server placement exists.

Before applying a binary search, we have to determine an upper bound and a lower bound. It is rather easy to get an upper bound and a lower bound on the maximum expected response time. We can use $1/(\mu_{max} - \lambda_{min})$ as a proper lower bound, where μ_{max} is the maximum server capability of servers and λ_{min} is the minimum requests of clients. A upper bound can be computed by the following steps. First, we set t to a sufficient large value and find a server placement. According to Corollary 1, the number of used servers must be smaller than or equal to m. Then we can use the maximum expected response time of servers as a proper upper bound. Next, we can combine a binary search of the maximum expected response time and the dynamic programming algorithm for the minimum server placement problem to find the optimal value of the maximum expected response time. Because the lower bound and the upper bound of the binary search are both functions of the input parameters, the algorithm is strongly polynomial.

5 Experimental Results

In this section we conduct several experiments to evaluate the proposed algorithms. Test cases are generated based on the proposed Grid model. The height of each case is at most 8. Each node has at most 4 children. The number of nodes in each test case is between 1250 and 1500. The request arrival rates for the leaf nodes and the service rates for intermediate nodes are generated from a uniform distribution. There are four testing groups. Each group has a different range of network latencies: 0.00005∼0.00015, 0.0005∼0.0015, 0.005∼0.015, and 0.05∼0.15. We will refer them as group 1, 2, 3 and 4, respectively. There are 1000 test cases in each group. Table 1 shows the summary of these parameters.

Table 1. Parameters of experiments

Parameter	Description
Height of tree	≤ 8
Number of child nodes	≤ 4
Number of nodes in each case	≈ 1300
Range of arrival rates	1∼4
Range of service rates	50∼350
Range of network latencies	0.00005∼0.00015, 0.0005∼0.0015, 0.005∼0.015 and 0.05∼0.15

First, the experiments for the minimum server placement problem are conducted. We use a greedy heuristic algorithm as a performance comparison with our dynamic programming algorithm since, to the best of our knowledge, there are no similar studies on QoS server placement problems that both consider the server's capacity and the network latency. The Greedy algorithm works as follows: in each iteration, it first selects all candidate servers that can satisfy the service quality requirement t, i.e., the expected response time of requests it served will less than t. Then it selects a site who has the maximum arrival rate of I/O requests. The process is repeated until all requests are served.

As the experiments with the four testing groups show similar results , we will present only the result with group 4. The performance metric is the difference in the number of servers used by Greedy and DP, i.e., the extra number of servers used by Greedy. The experimental results for the minimum server placement problem is shown in Fig. 4. The vertical axis shows the number of test cases, while the horizontal axis shows the difference in the number of servers used by these two algorithms.

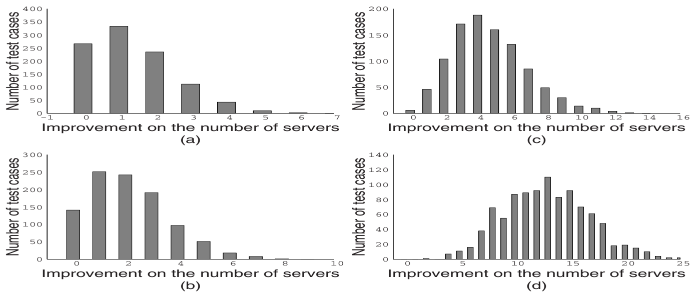

Fig. 4. Performance comparison for the minimum server problem. (a), (b), (c) and (d) are experimental results when t is set to 1, 0.75, 0.6 and 0.45 respectively.

In Fig. 4, it is clear that the difference in the number of servers used becomes significant as t decreases, i.e., as the service quality requirement becomes crucial. In Fig. 4(a), Greedy generates optimal solutions in 23.9% of the test cases and, in 84.4% of the test cases, the differences are between 0 and 2. However, in Fig. 4(d), Greedy generates no optimal solution and over 80% of test cases, the differences are between 10 and 28 when t is set to 0.45. Although Greedy is rather fast and easy to implement, the results show that it cannot generate acceptable solutions when the service quality requirement becomes crucial.

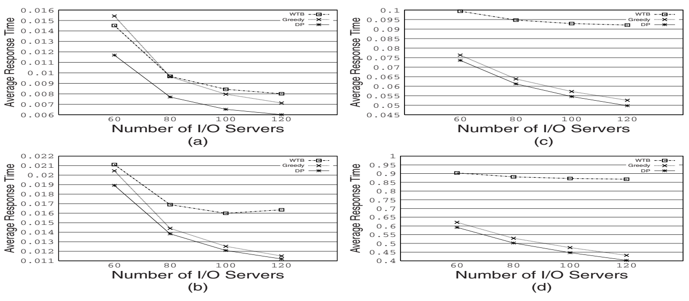

Fig. 5. Performance comparison for the optimal service quality problem. (a), (b), (c) and (d) are experimental results for group 1, 2, 3, and 4 respectively.

We next conduct the following experiments for the optimal service quality problem. We compare three algorithms: (1) the DP algorithm combined a binary search as described in Section 4, (2) the Greedy algorithm combined a binary search and (3) a waiting-time based(WTB) server placement algorithm described in [12]. Note that there is no guarantee of performance for the Greedy algorithm combined a binary search since the Greedy algorithm does not have the property of Corollary 1. A binary search is only used to adjust t such that Greedy can generate a placement with m servers. The WTB algotithm is similar to the algorithm described in Section 4 except it only tries to minimize the maximum waiting time of servers.

In the experiments, for each group of test cases, we use 4 different values of server numbers m: 60, 80, 100 and 120. The performance metric is the average of maximum expected response times of test cases. For each test case, there will be a maximum expected response time among those m servers. We use the average of maximum expected response times in 1000 test cases as our performance metric. The experimental results are shown in Fig. 5. The vertical axis shows the average expected response time, while the horizontal axis shows the number of servers m.

In Fig. 5, it is clear that the difference in performance between DP and WTB becomes larger as the network latency increases. When the network latency is small with respect to the server's waiting time, the difference of the average expected response time is less significant. However, as the network latency increases, the difference becomes larger because the expected response time is dominated by the network latency and WTB does not take network latencies into consideration. This result explains the advantage of DP algorithm: it takes both the server's waiting time and the network latency into consideration. Thus, DP can always get the best performance no matter the expected response time is dominated by either server's waiting time as the result shown in Fig. 5(a) or the network latency as the result shown in Fig. 5(d).

In Fig. 5(c) and 5(d), Greedy has a good performance when the number of I/O servers increases and the network latency dominates the expected response time. This is mainly due to the power of the binary search. However, as the expected waiting time dominates the expected response time, Greedy performs worse than WTB as shown in Fig. 5(a). Therefore, Greedy does not perform well in all kind of situations like DP does.

6 Conclusions

In this paper, we focus on two QoS I/O server placement problems in Data Grid environments. We consider the minimum server placement problem which asks how to place the minimum number of servers that meet both the constrains on servers' capabilities and the service quality requirement. Instead of using a heuristic approach, we propose an optimal algorithm based on dynamic programming as a solution to this problem.

The optimal service quality problem is also considered, which tries to place a given number of servers appropriately so that the maximum expected response time of servers can be minimized. By combining the dynamic programming algorithm with a binary search on the service quality requirement, we can find an optimal server placement. Several experiments are also conducted, whose results clearly show the improvement on the number of servers and the maximum expected response time compared with other algorithms.

Acknowledgments

The authors acknowledge the National Center for High-performance Computing in providing resources under the national project, "Taiwan Knowledge Innovation National Grid". This research is supported in part by the National Science Council, Taiwan, under Grant NSC NSC95-2221-E-001-002.

References

1. Johnston, W.E.: Computational and data Grids in large-scale science and engineering. Future Generation Computer Systems. **18**(8) (2002) 1085–1100
2. Grid Physics Network (GriphyN). (http://www.griphyn.org)
3. TeraGrid. (http://www.teragrid.org)
4. Wang, C.M., Hsu, C.C., Chen, H.M., Wu, J.J.: Efficient multi-source data transfer in data grids. In: CCGRID '06. (2006) 421–424
5. Lamehamedi, H., Shentu, Z., Szymanski, B.K., Deelman, E.: Simulation of Dynamic Data Replication Strategies in Data Grids. In: IPDPS 2003. (2003) 100
6. Deris, M.M., Abawajy, J.H., Suzuri, H.M.: An efficient replicated data access approach for large-scale distributed systems. In: CCGRID. (2004) 588–594
7. Hoschek, W., Jaén-Martínez, F.J., Samar, A., Stockinger, H., Stockinger, K.: Data Management in an International Data Grid Project. In: GRID 2000. (2000) 77–90
8. Krishnan, P., Raz, D., Shavitt, Y.: The cache location problem. IEEE/ACM Transactions on Networking **8**(5) (2000) 568–582
9. Ranganathan, K., Foster, I.T.: Identifying Dynamic Replication Strategies for a High-Performance Data Grid. In: GRID 2001. (2001) 75–86
10. Tang, M.X., Xu, M.J.: QoS-aware replica placement for content distribution. IEEE Trans. Parallel Distrib. Syst. **16**(10) (2005) 921–932
11. Wang, H., Liu, P., Wu, J.J.: A QoS-aware heuristic algorithm for replica placement. In: International Conference on Grid Computing. (2006) 96–103
12. Wang, C.M., Hsu, C.C., Liu, P., Chen, H.M., Wu, J.J.: Optimizing server placement in hierarchical grid environments. In: GPC. (2006) 1–11
13. Abawajy, J.H.: Placement of File Replicas in Data Grid Environments. In: International Conference on Computational Science. (2004) 66–73