OPTIMAL INFORMATION GATHERING SCHEME OVER A SCALABLE GRID INFORMATION SERVICES ARCHITECTURE

Nianjun Zhou, Dikran Meliksetian, Jean-Pierre Prost, Irwin Boutboul
150 Kettletown Road, Southbury, CT 06488, USA

Keywords: Information gathering, cost optimization, Markov Chain, information reduction, grid computing, dynamic programming.

Abstract: A proactive model for gathering resource attribute values in large scale distributed systems is proposed and analyzed within the context of resource discovery. This model, based on a tree topology, relies on information reduction to limit the amount of information collected at each node of the tree structure and to minimize information update and query cost. A Markov chain is used to model resource attribute value changes. This model is solved using dynamic programming to determine the optimal reduction scheme that minimizes the overall cost of updating and querying resource attributes.

1 INTRODUCTION

Grid computing (Kaufmann, 2004; Foster et al., 2001; Foster et al., 2002) enables the virtualization of distributed computing and data resources such as CPU, network bandwidth and storage including memory to create a single system image, granting users and applications seamless access to vast IT capabilities. One of the key issues is to maintain accurate information about the entities constituting the system. Entity state is typically represented by attributes, exhibiting values, which change over time. As the system size (i.e. the number of resources in the system) grows, maintaining an accurate representation of its state becomes a real challenge, since the number of messages required to maintain this state scales with the number of resources. Typical models used for information gathering, such as the Globus Toolkit MDS (Czajkowski, et al., 2001; Zhang, Schopf, 2004), take a reactive approach. In such models, it is only once a query (explicit or implicit) for a resource attribute value is submitted that the relevant information is fetched. In order to improve the latency incurred in answering queries, caching techniques are used to store the information in aggregator nodes until a pre-defined time-to-live period expires. A tree topology is used, where the leaves represent the system resources and upper level nodes are aggregator nodes. In these models however, the amount of information stored at each aggregator node grows linearly with the number of resources registered to report their attribute values to them, and once cached information becomes stale, querying it may lead to important latencies. This method is efficient when the query to the system is less frequent than the changes of the attribute value. In other grid information systems, peer to peer interactions are used among aggregator nodes to improve scalability (Mastroianni et al., 2005), or filtering techniques (Balaton et al, 2002) and publish-subscribe mechanisms (Jie, 2004; Cooke et al., 2004) allow for the selection of the information to be gathered and thus limit the amount of information aggregated. However, in these systems, depending upon which aggregator is queried, the resulting information can vary greatly.

Ganglia (Massie et al., 2004) is a scalable distributed monitoring system for high performance computing environment. It is based on a hierarchical design targeted at federations of clusters. It relies on a multicast-based listen/announce protocol to monitor state within clusters and uses a tree of point-to-point connections amongst representative cluster nodes to federate clusters and aggregate their state. Although Ganglia is a monitoring system and while the objective of the proposed scheme in this paper is data acquisition and resource discovery; the two share a common proactive approach for data acquisition, and both are concerned with minimizing the network load required for this operation. Since the objectives of the two systems are different, the approaches for load minimization are different. While Ganglia uses standard data compression techniques, we use an information reduction scheme as described in the subsequent sections.

Zhou N., Meliksetian D., Prost J. and Boutboul I. (2006).
OPTIMAL INFORMATION GATHERING SCHEME OVER A SCALABLE GRID INFORMATION SERVICES ARCHITECTURE.
In *Proceedings of WEBIST 2006 - Second International Conference on Web Information Systems and Technologies - Internet Technology / Web Interface and Applications*, pages 184-189
DOI: 10.5220/0001256801840189

In this paper, we propose a proactive model, which triggers information updates up the tree structure each time the attribute value of a resource changes. In addition, we reduce information going up the tree by consolidating at each node the values of a given attribute from all the nodes reporting to it into a set of attribute value intervals. Upon an explicit or implicit user query, the reduction nodes are consulted in a hierarchical fashion until the response can be formulated. The introduction of the information reduction provides us a mechanism of not propagating the un-needed information for the queries to the high levels of the topology. The key issue becomes the determination of the optimal number and ranges of the intervals at each tree level to minimize the cost for updating and querying a resource attribute.

This paper is organized as follows. In Section 2, we formulate our optimization problem in the case of a single attribute. In Section 3, we describe a stochastic model where the process of the attribute value change is represented as a Markov chain. In Section 4, we solve the model to determine the update and the query costs associated with a single attribute. In Section 5, we demonstrate that the determination of the optimal set of attribute value intervals at each tree level to achieve least cost is a dynamic programming problem, and we propose an algorithm to determine the optimal solution. In Section 6, we conclude our paper with future plans, including the extension of this research into multiple attribute scenarios and a comparison with the reactive model.

2 PROBLEM FORMULATION

As described above, we have a multi-level tree topology, in which leaves are system resources and upper level nodes are reduction nodes. We assume a fully balanced tree, in which each node has a limited fixed number of children. The leaf nodes of our tree structure are the system resources and are monitored regularly for attribute value changes. Each time an attribute value changes, some of the reduced information maintained for this attribute by the reduction nodes it reports to (directly or indirectly – we will refer to these nodes as its reporting branch), may have to be updated. In other words, some of the numbers of resources in each attribute value interval of the reporting nodes may have to be incremented or decremented. We want to minimize the number of updates up the reporting branch. Therefore, if we were to only look at the update cost, the best solution is to use coarse attribute value intervals.

We must also take in consideration the query cost. Unlike updates, which are triggered from leaf nodes to the root, queries start at the root. If we only use coarse intervals, depending on the attribute value query (assumed to be of the form "find all the resources for which *attr* ≤ *value*"), we may end up traversing most of the tree, since most top level reduction nodes will have resources reported in each attribute value interval. Therefore, the query cost may be expensive.

The whole problem is to find the optimal set of attribute value intervals that achieves the best trade-off between the update cost and the query cost. We define the cost as the number of messages exchanged between nodes (either between a resource and a reduction node, or between two reduction nodes).

3 OUR MODEL

Each node has a fixed number of nodes reporting directly to it, called n. K represents the number of levels in the tree and N the number of leaf nodes (i.e. system resources), Level 0 always represents the leaf nodes and level K the root. Finally,

$$K = \log_n N \tag{1}$$

3.1 Attribute Change Model

We assume that the attribute can only have discrete numerical values. We let M represent the number of attribute values and the set $\{a_1, a_2, ..., a_M\}$ represent the possible values of the attributes. We model the attribute value change process using Markov chain and the changes of the attribute at different leaf nodes are identical independent distributions. We further assume that the attribute value can either stay the same, or increase and decrease by one unit within a time interval as a birth-and-death process (Leon-Garcia, 1994). This assumption is valid if we limit the time interval to an arbitrary small value. We denote by $p_0(i)$ the equilibrium probability that the value of the attribute is a_i , and by $f_0(i)$ and $r_0(i)$ the forward and backward transition probabilities from a_i to a_{i+1} and a_{i-1} respectively. The equilibrium probability $p_0(i)$ can be deduced from the transition probabilities. For our analysis, we assume that these transition probabilities are known as prior knowledge.

3.2 Update, Query and Cost Models

Updates occur at the same regular time intervals as attribute changes. Updates are triggered up the tree, starting from the system resource whose attribute value either increased or decreased, along its

reporting branch. The number and identities of resources falling into each attribute value interval of each reduction node are adjusted when appropriate. An update is generated by a leaf node only when the change of the attribute would cause a change in the attribute value interval where the value falls at the parent node. Similarly, an update is generated by a reduction node only if the change will cause a change of the attribute value interval of its parent node.

The number of attribute value intervals at a given level k, is denoted by l_k, and the intervals are indexed with $i \in \{1, \cdots, l_k\}$. The attribute value of a given leaf node will belong to one and only one attribute value interval at level k. We use I_k to represent the random variable corresponding to the attribute value interval index that the attribute value of a given resource belongs. It is obvious that the changes of I_k are Markov birth-and-death processes. We use $p_k(i)$ to denote the equilibrium probability that $(I_k = i)$. The transition probabilities of I_k from interval i to interval $(i+1)$ and $(i-1)$ during a time interval are denoted by $f_k(i)$ and $r_k(i)$ respectively. Based on our earlier assumption, all other transition probabilities are null.

We consider only simple queries of the form "find all the system resources for which value of attribute x is less or equal to value a_j", where a_j represents the jth discrete numerical value that x can have.

In this study, we only consider the cost related to network traffic. We consider the message payload cost to be negligible compared to the message assembly, transfer, and de-assembly, hence the cost will be expressed as the total number of messages exchanged between nodes, while accounting for all potential attribute value updates and queries.

4 UPDATE AND QUERY COSTS

The total number of leaf nodes that are in the subtree rooted by a node at level k is n^k. The number of nodes at level k is equal to $\left(N/n^k\right)$. The total number of nodes at level k and above is denoted by m_k and is equal to:

$$m_k = \sum_{j=k}^{K} \frac{N}{n^j} = \sum_{j=k}^{K} n^{K-j} = \sum_{j=0}^{K-k} n^j = \frac{n^{K-k+1}-1}{n-1} \qquad (2)$$

Given the attribute value intervals at all levels, and a the Markov chain model for the attribute values specified by $p_0(i)$, $f_0(i)$ and $r_0(i)$, the equilibrium probability distribution and the transition probabilities at all levels can be computed as follows. Consider the i^{th} attribute value interval $\left[a_{x_i}, a_{y_i}\right]$ at level k. The probability of the attribute value belonging to the i^{th} interval is equal to the sum of the probabilities that the attribute has values a_{x_i} through a_{y_i}:

$$p_k(i) = \sum_{j=x_i}^{y_i} p_0(j) \qquad (3)$$

The transition probabilities are determined similarly:

$$f_k(i) = \frac{p_0(y_i) f_0(y_i)}{p_k(i)} \qquad (4)$$ and

$$r_k(i) = \frac{p_0(x_i) r_0(x_i)}{p_k(i)} \qquad (5)$$

Each update generates one message, while a query to a particular node consists of a request and a response. We will take this factor into consideration in the following subsections.

For the purpose of this analysis we are assuming a given fixed reduction scheme that specifies the attribute value intervals at all levels, a given attribute value distribution by $p_0(i)$, $f_0(i)$ and $r_0(i)$, a query activity model specified by F, the frequency of query requests within a given time interval, and q_j, the probability that the query of the form "find all leaf nodes that have the attribute value less than or equal to a_j" refer to the particular value a_j.

Update Cost

Given $p_k(i)$, $f_k(i)$ and $r_k(i)$, we can determine the update cost by determining the expected number of messages exchanged for an update. A given leaf node will cause an update to be generated from a node at level k to a node at level $(k+1)$, if the attribute value crosses attribute value interval boundaries at level $(k+1)$. The probability of this event is denoted as u_k, where $0 \le k \le K-1$, and is equal to:

$$u_k = \sum_{i=1}^{l_{k+1}} p_{k+1}(i)\left[f_{k+1}(i) + r_{k+1}(i)\right] \qquad (6)$$

The probability that a node at level k, $0 \le k \le K-1$, will actually generate an update is equal to the probability that at least one of the n^k leaf nodes in the subtree rooted at this node causes an update; this probability, U_k is:

$$U_k = 1 - (1-u_k)^{n^k} \qquad (7)$$

The expected number of updates, i.e. the update cost CU, is determined by adding these probabilities at all nodes:

$$CU = \sum_{k=0}^{K-1} \frac{N}{n^k} U_k = \sum_{k=0}^{K-1} \frac{N}{n^k}\left[1 - (1-u_k)^{n^k}\right] \qquad (8)$$

Query Cost

The query cost is equal to twice the number of nodes that need to be visited to respond to the query. In the following sections, we consider a simple query of

the form "find all leaf nodes that have the attribute value less than or equal to a_j", where a_j is one of the numerical values of the attribute. We consider two different variations for counting the visited nodes.

Case I: It is obvious that if the query is addressed to a node where a_j is equal to the upper bound of one of the attribute value intervals $[a_x, a_y]$, i.e. $a_j = a_y$, the node can definitely respond to the query. Since the attribute value intervals are the same for all nodes at a given level, the simplest way of counting the nodes visited is to count all the nodes down to the level where all the nodes have an attribute value interval $[a_x, a_y]$. Let $k(a_j)$ denote this level, then the total number of nodes visited is equal to the total number of nodes from the root, level K, down to and including level $k(a_j)$. Using equation (2), this number is equal to:

$$m_{k(a_j)} = \frac{n^{K-k(a_j)+1} - 1}{n-1} \tag{9}$$

Consequently, the query cost for searching the leaf nodes satisfying $x \le a_j$ is:

$$cq(j) = 2m_{k(a_j)} = 2\frac{n^{K-k(a_j)+1} - 1}{n-1} \tag{10}$$

and the weighted total cost for all possible values of a_j is

$$CQ = \frac{2F}{n-1}\sum_{j=1}^{M} q_j\left(n^{K-k(a_j)+1} - 1\right) \tag{11}$$

where F is the frequency of query requests within a given time interval and q_j is the probability that the query refers to a particular a_j.

Case II: A more interesting and challenging case occurs when we consider closely the conditions under which we can prune the search space even before we reach level $k(a_j)$. Obviously, if we encounter a node g at level $k(g) > k(a_j)$ such that the attribute value interval $[a_x, a_y]$ that contains a_j has no resources, node g can respond to the query without consulting its children. Let the index of the attribute value interval $[a_x, a_y]$ at this node be denoted by $i_{k(g)}(j)$. Note that this index is a function of only the level $k(g)$ of the node and not of the node itself because all the nodes at a given level have the same attribute intervals. The probability of this attribute value interval having no resources is equal to the probability that none of the $n^{k(g)}$ leaf nodes in the sub-tree rooted by g have an attribute in this interval. In our notation this is equal to

$$\left(1 - p_{k(g)}\left(i_{k(g)}(j)\right)\right)^{n^{k(g)}} \tag{12}$$

where $p_k(i)$ is the probability that the attribute value of a leaf node falls in the i^{th} interval of level k.

In this case, we want to determine the expected number of visited nodes given the $p_k(i)$ distributions at all levels. With the current notation, the probability that the children of g are visited is:

$$1 - \left(1 - p_{k(g)}\left(i_{k(g)}(j)\right)\right)^{n^{k(g)}} \tag{13}$$

and consequently, the expected number of the children of g that are visited is

$$n\left(1 - \left(1 - p_{k(g)}\left(i_{k(g)}(j)\right)\right)^{n^{k(g)}}\right) \tag{14}$$

Since there are $\frac{N}{n^{k(g)}}$ nodes at level $k(g)$ and there are n children nodes for node g, and the probability distributions of these nodes are independent from each other, the expected number of visited nodes at level $(k(g)-1)$ is:

$$\frac{N}{n^{k-1}}\left(1 - \left(1 - p_k\left(i_k(j)\right)\right)^{n^k}\right) \tag{15}$$

Note that we have dropped the explicit dependence of $k(g)$ on g since all nodes at level $k = k(g)$ are identical.

The query starts from the root. It is obvious that the root has to be visited with probability one, while for the lower levels the expected number of visited nodes is given by the equation derived earlier. Consequently the total expected number of visited nodes is:

$$1 + \sum_{k=k(a_j)+1}^{K} \frac{N}{n^{k-1}}\left(1 - \left(1 - p_k\left(i_k(j)\right)\right)^{n^k}\right) \tag{16}$$

The corresponding expected cost for a given a_j is:

$$cq(j) = 2 + 2\sum_{k=k(a_j)+1}^{K} \frac{N}{n^{k-1}}\left(1 - \left(1 - p_k\left(i_k(j)\right)\right)^{n^k}\right) \tag{17}$$

and

$$CQ = 2F + 2F\sum_{j=1}^{M} q_j \sum_{k=k(a_j)+1}^{K} \frac{N}{n^{k-1}}\left(1 - \left(1 - p_k\left(i_k(j)\right)\right)^{n^k}\right) \tag{18}$$

Total Cost

Case I: From (8) and (11), we derive:

$$C = CU + CQ = \sum_{k=0}^{K-1} \frac{N}{n^k}\left[1 - \left(1 - u_k\right)^{n^k}\right] + \frac{2F}{n-1}\sum_{j=1}^{M} q_j\left(n^{K-k(a_j)+1} - 1\right) \tag{19}$$

Case II: From (8) and (18), we derive:

$$C = CU + CQ = \sum_{k=0}^{K-1} \frac{N}{n^k}\left(1-\left(1-u_k\right)^{n^k}\right) + 2F + 2F\sum_{j=1}^{M} q_j \sum_{k=k(a_j)+1}^{K} \frac{N}{n^{k-1}}\left(1-\left(1-p_k\left(i_k(j)\right)\right)^{n^k}\right) \quad (20)$$

5 ANALYSIS OF THE OPTIMIZATION PROBLEM

As described earlier, the optimization problem considered in this paper is to determine the attribute value intervals at all levels of the hierarchy given the attribute value distribution and the query probabilities. First, let us consider the possible attribute value intervals, called partitions hereafter. The total number of possible partitions of an attribute with M discrete values is:

$$\sum_{i=0}^{M} \frac{(M-1)!}{(M-i-1)!\,i!} = 2^{M-1} . \quad (21)$$

Let Π denote the set of all partitions and let us define a "coarser or equal to" relationship between the partitions. A partition $P \in \Pi$ is coarser or equal to another partition $P' \in \Pi$, denoted $P \succ P'$, if the intervals of P are unions of one or more of consecutive intervals of P'.

A particular feasible solution consists of a sequence of partitions $\{P_K, P_{K-1}, \cdots, P_1, P_0\}$ where P_k represents the partition of level k, P_0 is always the partition consisting of singleton intervals ${}^1P$, and $P_i \succ P_j$ for all $i \geq j$.

The cost functions $C(\{P_K, P_{K-1}, \cdots, P_1, P_0\})$ that were calculated in the previous sections are for a particular feasible solution $\{P_K, P_{K-1}, \cdots, P_1, P_0\}$, and the optimization problem is to determine a solution $\{P_K^*, P_{K-1}^*, \cdots, P_1^*, P_0^*\}$ that minimizes C.

5.1 Dynamic Programming Formulation

In the following, we rewrite the cost equations to highlight the dynamic programming (Ecker and Kupferschmid, 1998) nature of the problem. We will consider the results for Case II of the previous section; however the same approach is also valid for case I.

The cost for a particular feasible solution $\{P_K, P_{K-1}, \cdots, P_1, P_0\}$ was derived as in equation (20). The constant term in that equation has no consequence on the solution. Let us consider the last term CQ and change the order of summations:

$$CQ = 2F\sum_{j=1}^{M} q_j \sum_{k=k(a_j)+1}^{K} \frac{N}{n^{k-1}}\left(1-\left(1-p_k\left(i_k(j)\right)\right)^{n^k}\right) \quad (22)$$

$$CQ = 2F\sum_{k=1}^{K} \frac{N}{n^{k-1}} \times \sum_{j=1}^{M} q_j s\left(k-\left(k(a_j)+1\right)\right) \times \left(1-\left(1-p_k\left(i_k(j)\right)\right)^{n^k}\right) \quad (23)$$

where $s\left(k-\left(k(a_j)+1\right)\right)$ is the step function, i.e.:

$$s\left(k-\left(k(a_j)+1\right)\right) = \begin{cases} 1 & \text{if } k \geq k(a_j)+1 \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

Let us define, for a given k, the set $J(k)$ of all indices j such that a_j does not appear as a right boundary at level k or above. $J(k)$ is equal to:

$$J(k) = \left\{ j \,\middle|\, 1 \leq j \leq M \text{ and } k > k(a_j) \right\} \quad (25)$$

CQ can then be written as:

$$CQ = 2F\sum_{k=1}^{K} \frac{N}{n^{k-1}} \sum_{j \in J(k)} q_j \left(1-\left(1-p_k\left(i_k(j)\right)\right)^{n^k}\right) \quad (26)$$

With these transformations, the cost function can be written as:

$$C = \sum_{k=1}^{K} \frac{N}{n^{k-1}}\left(1-\left(1-u_{k-1}\right)^{n^{k-1}}\right) + 2F + 2F\sum_{k=1}^{K} \frac{N}{n^{k-1}} \sum_{j \in J(k)} q_j \left(1-\left(1-p_k\left(i_k(j)\right)\right)^{n^k}\right) \quad (27)$$

Finally:

$$C = 2F + \sum_{k=1}^{K} \frac{N}{n^{k-1}} \times \left(\left(1-\left(1-u_{k-1}\right)^{n^{k-1}}\right) + 2F \sum_{j \in J(k)} q_j \left(1-\left(1-p_k\left(i_k(j)\right)\right)^{n^k}\right) \right) \quad (28)$$

By definition, $J(k)$ can be uniquely determined knowing the partitions in the sub-sequence $\{P_K, P_{K-1}, \cdots, P_k\}$, $i_k(j)$ and other terms in the second part are uniquely determined by the partition P_k. Finally, since:

$$u_k = \sum_{i=1}^{l_{k+1}} p_{k+1}(i)\left[f_{k+1}(i) + r_{k+1}(i)\right] \quad (6)$$

u_{k-1} is uniquely determined by the partition P_k.

These considerations drive us to define the partial cost function $C(\{P_K, P_{K-1}, \cdots, P_i\})$, for $1 \leq i \leq K$, as:

$$C(\{P_K, P_{K-1}, \cdots, P_{i+1}, P_i\}) = 2F + \sum_{k=i}^{K} \frac{N}{n^{k-1}} \times \left(\left(1-\left(1-u_{k-1}\right)^{n^{k-1}}\right) + 2F \sum_{j \in J(k)} q_j \left(1-\left(1-p_k\left(i_k(j)\right)\right)^{n^k}\right) \right) \quad (29)$$

or equivalently:

$$C(\{P_K, P_{K-1}, \cdots, P_{i+1}, P_i\}) = C(\{P_K, P_{K-1}, \cdots, P_{i+1}\}) + \frac{N}{n^{i-1}}\left(\left(1-(1-u_{i-1})^{n^{i-1}}\right) + 2F\sum_{j\in J(k)} q_j\left(1-(1-p_i(i_i(j)))^{n^i}\right)\right) \quad (30)$$

with:

$$C(\{P_K\}) = 2F + \frac{N}{n^{K-1}} \times \left(\begin{array}{l}\left(1-(1-u_{K-1})^{n^{K-1}}\right)+ \\ 2F\sum_{j\in J(K)} q_j\left(1-(1-p_K(i_K(j)))^{n^K}\right)\end{array}\right) \quad (31)$$

where the second term is a function of $\{P_K, P_{K-1}, \cdots P_{i+1}, P_i\}$ and none of the partitions P_k for $k < i$.

The following algorithm formalizes the derivation in the previous subsections.

1. For every $P \in \Pi$, calculate $C(\{P\})$.
2. For $k = K-1$ to 1 do:
   2.a For every $P \in \Pi$, identify all partial solutions $\{P_K, P_{K-1}, \cdots, P_{k+1}\}$ such that $P_{k+1} \succ P$.
   2.b For every partial solution in step 2.a do:
      2.b.i Calculate $C(\{P_K, P_{K-1}, \cdots P_{k+1}, P\})$.
      2.b.ii Determine the $\{P_K^*, P_{K-1}^*, \cdots, P_{k+1}^*\}$ that minimizes $C(\{P_K, P_{K-1}, \cdots P_{k+1}, P\})$.
      2.b.iii Add $\{P_K^*, P_{K-1}^*, \cdots, P_{k+1}^*, P\}$ to the set of potential solutions of level k.

6 FUTURE WORK

In summary, we have introduced a new architecture for a scalable grid information service and modeled it in order to obtain the optimal parameters of the system in the particular case of a single attribute with known attribute model and query distribution model. We are working on extending this approach to multiple resource attributes. By assuming that multiple attributes share the same index topology for information update and query, and that the update and query messages and their responses for all attributes are merged together, we can calculate the cost given the aggregation intervals for each attribute. The aggregation intervals themselves can be found by using the method given in Section 5 as a first order of approximation.

Another avenue for future research is the overall cost vs. accuracy comparison with reactive information gathering scheme. We believe that there is a threshold beyond which the proactive information gathering scheme is better than the reactive information gathering scheme. This threshold depends on 1) how quickly the attribute changes; 2) how often queries occur and 3) the time to live values of the reactive caching.

REFERENCES

Kaufmann, M., 2004. *The Grid: Blueprint for a new computing infrastructure,* 2nd ed., ISBN: 1-55860-933-4K.

Foster, I.; Kesselman, C.; and Tuecke, S., 2001, *"The anatomy of the grid: Enabling scalable virtual organizations,"* International J. Supercomputer Applications, 15(3).

Foster, I; Kesselman, C.; Nick, J; Tuecke, S. June 22, 2002, *"The physiology of the grid: An open grid services architecture for distributed systems integration,"* Open Grid Service Infrastructure WG, Global Grid Forum, http://www.globus.org/alliance/publications/papers/ogsa.pdf.

Czajkowski, K.; Fitzgerald, S.; Foster, I.; and Kesselman, C., August 2001,*"Grid information services for distributed resource sharing,"* Tenth IEEE International Symposium on High-Performance Distributed Computing, IEEE Press.

Zhang, X.; Schopf, J., April 2004. *"Performance analysis of the Globus Toolkit monitoring and discovery service, MDS2,"* Proc. International Workshop on Middleware Performance (MP 2004), 23rd International Performance Computing and Communications Workshop (IPCCC).

Mastroianni, C.; D. Talia, D.; and Verta, O., 2005. *"A superpeer model for building resource discovery services in grids: Design and simulation analysis,"* Advances in Grid Computing - EGC : European Grid Conference, Amsterdam, The Netherlands, P.M.A. Sloot et al. Eds., Springer-Verlag.

Balaton, Z.; Gombás, G.; and Németh, Zs., 2002. *"Information system Architecture for brokering in large scale grids,"* Parallel and Distributed Systems: Cluster and Grid Computing (Proceedings of DAPSYS 2002, Linz), Kluwer, pp. 57-65.

Jie, C., February 2004. *"Index grid services using Globus Toolkit 3.0,"* IBM developerWorks, http://www-128.ibm.com/developerworks/grid/library/gr-indexgrid/.

Cooke A. W. et al., December 2004. *"The relational grid monitoring architecture: Mediating information about the grid"*, Journal of Grid Computing, Vol. 2 No. 4.

Massie, M.; Chun, B; and Culler, D., 2004. *"The Ganglia distributed monitoring system: Design, implementation, and experience"*, Parallel Computing Vol. 30, pp.817–840.

Leon-Garcia, A., 1994. *"Probability and random processes for electrical engineering"*, 2nd ed., Addison-Wesley Publishing Company, pp. 459-498.

Ecker J. and Kupferschmid, M., 1998. *"Introduction to operations research",* Krieger Publishing Company, 1988, pp. 347-374.