Resource Discovery in a Dynamic Grid Environment*

Moreno Marzolla[1†] Matteo Mordacchini[1,2] Salvatore Orlando[1,3]
[1] Dip. di Informatica, Università Ca' Foscari di Venezia, via Torino 155, 30172 Mestre, Italy
[2] INFN Sezione di Padova, via Marzolo 8, 35100 Padova, Italy
[3] ISTI Area della Ricerca CNR, via G. Moruzzi 1, 56124 PISA, Italy
e-mail: {marzolla|mordacchini|orlando}@dsi.unive.it

Abstract

Resource discovery in a Grid environment is a critical problem, as a typical Grid system includes a very large number of resources, which must be readily identified and accessed to run applications. Traditional Grid discovery algorithms perform poorly, as they do not scale, nor allow Grid-enabled applications to transparently query the whole set of Grid resources. Peer-to-Peer (P2P) has been argued as a suitable distributed paradigm that not only overcomes the issues of scalability of such discovery systems, but also better supports the discovery of resources in a context of dynamicity of resources and associated information. In this paper we propose a P2P system for indexing and discovering Grid resources. We assume that Grid resources are characterized by a set of attributes, and our system can be queried for resources satisfying arbitrary range conditions on these attributes. Note that traditional P2P searching techniques can not be directly applied in this case, since they work well mostly for static content and exact queries. Simulation results show that the system provides an adequate degree of scalability.

1. Introduction

Matching the needs of an application with available resources is one of the basic and key aspects of a Grid system. In order to run Grid applications, we have to look for suitable resources satisfying a given set of constraints, and for which we have the access permission.

Several actual Grid implementations adopt a trivial solution to resource location and selection. Given a Virtual Organization (VO) [5], the information regarding the distributed resources of the VO are centralized. Such information service, associated with the specific VO, is accessed by a Grid component called Resource Broker (RB) or Workload Management System (WMS) [3], which is in charge of choosing the best VO's resource(s) for executing the job submitted by any VO's user. Hereinafter we will use the acronym WMS, which is the name adopted in the EDG and EGEE EU Projects [1] for this Grid component.

This simple solution can be viable if we consider a Grid VO as a static entity, where each WMS is responsible for all the resources of a static VO, and every VO's user has to submit her/his job queries through this VO WMS (or a replica of this WMS). Next generation Grid have to address the needs of dynamic virtual enterprises, so it is needed to build the infrastructure supporting the formation of dynamic VOs. Since VOs should be dynamically created, the resource set accessible from a user becomes larger.

In order to permit searching and selection within a large amount of resources, we can think about the creation of a single, centralized index of Grid resources, which should be used by any WMS contacted by a user. Despite common beliefs, the approach of providing a large centralized search service has been shown viable and scalable by modern distributed Web Search Engines, whose indexed information are, however, mostly static. Unfortunately we have to solve this problem in a context where information about resources are dynamic, by achieving both scalability and efficiency in the actual implementation.

P2P systems seem to have the characteristics required to overcome the above-mentioned problems. Recent studies in the field of resource discovery pointed out the possible synergies and convergence that can be exploited between the Grid and P2P worlds [4, 10]. In general, resource discovery in P2P networks is a relatively well studied problem, and many solutions have been proposed in the literature [2]. However, most of these algorithms work well for locating static data, i.e., whose content which does not change over time, or data that can be exactly identified by a key. This is

*This work has been partially supported by the EU Project EGEE (Enabling Grids for E-sciencE), the European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies (CoreGRID), and by MIUR Research Project FIRB/PERF.

†Author's current address: INFN Sezione di Padova, via Marzolo 8, 35100 Padova, Italy.

not the case for a Grid environment, since users may request resources with characteristics that do change over time (e.g., CPU utilization, free disk space, and so on), or may perform range queries over these resource features (e.g. memory size greater than 512 MB). Some modern P2P systems, such as those proposed in [8, 9], are not suited for this problem either because the managed data are static, or because they work well for exact queries.

In this paper we propose a P2P system for resource location in Grid environments where resource attributes may change value over time. We consider a tree-structured overlay network over the set of WMSes, where each node has an exact knowledge on the set of resources directly managed by it, but also a condensed description of the resources present in every neighboring WMS with respect to the overlay tree. This condensed description consists of bitmap indexes of the values of resources attributes. We use the bitmap indexes to route queries towards the location of resources possibly satisfying the query. We also describe how the indexes can be updated if some attribute value changes. We provide simulation results to show the potential benefits of this approach.

This paper is organized as follows. Section 2 contains a description of the proposed algorithm. In 2.1 we introduce the notation and we give an abstract representation of the problem we are considering. In 2.2 we give the algorithm used to locate resources, and in 2.3 the algorithm used to propagate changes over the network. Some simulation results are presented in Section 3, and conclusions and future works are illustrated in Section 4.

2. Discovery of Dynamic Resources

2.1. The high-level structure

We assume the system is made of N resources $\{R_1, R_2, \ldots R_N\}$, where disjoint subsets of them are managed by distinct WMSs. Each resource has a set of attributes with corresponding values. We denote with $A[R]$ the value of attribute A for resource R, and we assume that attribute values are real numbers. In actual Grid deployments, we may have at least two types of resources, namely Computing Element (CE) and Storage Element (SE). Each CE C may have attributes *CpuSpeed*$[C]$ (speed of the CPUs), *NumCpus*$[C]$ (number of CPUs), *RamSize*$[C]$ (maximum available physical memory), *WaitingJobs10*$[C]$ (average number of waiting jobs over the last 10 minutes), *Utilization10*$[C]$ (utilization of the CE over the last 10 minutes), and others. Each SE resource S may have attributes *Capacity*$[S]$ (total capacity of the SE), *FreeSpace*$[S]$ (free space left on the device), and others.

For each resource, some of the attributes may have constant value, while others may vary over time. We assume that variations may be arbitrary, even if in practice the magnitude of such variations is usually limited, i.e. the relative difference between two successive values is small.

Users need to locate and acquire resources in order to execute jobs. To find the most suitable resources for a given task, users query the system for one or more resources satisfying certain criteria. For example, one of such queries may look like:

$$\begin{aligned} Q = \{R \in \{R_1, \ldots R_N\} \mid & \textit{CpuSpeed}[R] \geq 2.0\textit{GHz} \\ \textbf{and} \quad & \textit{RamSize}[R] \geq 512\textit{MB} \\ \textbf{and} \quad & \textit{Utilization10}[R] \leq 0.3\} \end{aligned}$$

This query looks for a computational resource with CPU speed at least 2.0*GHz*, at least 512 MB of RAM and with utilization over the last 10 minutes of at most 0.3. We assume that a generic query predicate is a boolean composition of range conditions on some of the attributes [7]. The query strategy described in this paper can be applied to range queries on any attribute type on which it is possible to define a total ordering.

In order to efficiently locate the resources matching a given query, we propose the following search algorithm, over a P2P network connecting all the WMSs. Suppose that the value of $A[R]$ is in the range $[a, b]$. We choose a set of k pivot elements $a = a_0 < a_1 < \ldots a_{k-1} < a_k = b$ and we encode the value of $A[R]$ with a k bits binary string, such that the i-th bit is set to 1 if and only if $A[R] \in [a_i, a_{i+1})$. This representation, called *bitmap index*, was first described in [6]. We define *BitIdx*(v) the bitmap index corresponding to value v. Note that *BitIdx*(v) has exactly one bit set to 1.

A bitmap index is a simplified version of a histogram. A histogram is based on partitioning one of the relation attributes into buckets, and then storing, for each bucket, of a few summary information in place of the detailed one. Information compression within buckets allows fast approximate answers to be obtained, by evaluating queries on reduced data in place of original ones. We may define different domain partitions for different types of attributes, so that the bitmap indexes may have different lengths.

We suppose that the WMS are connected with a tree overlay network. Each WMS W has complete knowledge over the values of the resources it directly manages. We denote with *Nb*(W) the set of neighbors of W, that is, the set of WMSes directly connected with W on the overlay network. For each $W' \in$ *Nb*(W), let $\mathcal{R}_W(W')$ be the set of resources on the subtree rooted at W' which does not contain W. W knows the bitmap index for the attributes in $\mathcal{R}_W(W')$. More precisely, let us denote with $W' \rightarrow W$ the link from W' to W in the overlay network. For each $W' \in$ *Nb*(W), for each attribute A of resources in $\mathcal{R}_W(W')$, W associates with the link $W \rightarrow W'$ the fol-

lowing quantity:

$$\textit{LinkBitIdx}\,(W \rightarrow W', A) \equiv \bigvee_{R \in \mathcal{R}_W(W')} \textit{BitIdx}\,(A[R]) \quad (1)$$

which is the bitwise intersection of all the bitmap indexes $\textit{BitIdx}\,(A)$R associated with the resources in $\mathcal{R}_W(W')$. We use the values $\textit{BitIdx}\,(W \rightarrow W', A)$ to route the query on the overlay network, according to algorithms described in 2.2.

To summarize, the notation used in this paper is the following:

Nb (W) The set of WMS directly connected to W on the overlay network

BitIdx (v) The bitmap index for value v

$\boldsymbol{A}[R]$ The value of attribute A for resource R

LinkBitIdx $(W \rightarrow W', A)$ The bitmap index for attribute A associated with the link from W to W'; it denotes the or-value of all bitmap indexes for all instances of A in the tree rooted in W' which does not contain W

2.2. Query Processing

We consider resource queries as partial range queries, i.e. the user may look for resources with (a subset of) attribute values within given ranges. User requests are submitted to one of the WMS, which routes the query and collects responses from other nodes in the network. Query routing is performed according to the strategy described below.

Each WMS W receives a query Q from one of its neighbors on the overlay network. First, W checks whether it has some resources satisfying the request; if so, a query hit is reported. The query Q is always forwarded to each neighbor WMS (excluding the one which originally sent it) using a BFS (Breadth First Search) algorithm. In order to avoid flooding the entire network, queries are forwarded only to a subset of neighbors, excluding those paths which surely will not contain any useful resource. This technique is similar to the Directed BFS visit described in [11]; however, we cannot rely on statistics from previous query results to select the neighbors (as in [11]), because the resource attributes may change value, so that past query responses do not provide any meaningful information on the actual state of the system. Each WMS performs a DBFS by checking the the bitmap indexes associated with each neighbor. The idea is as follows: let us consider attribute $A[R]$ with domain $[a, b]$, such that the domain is partitioned into k disjoint intervals $[a_i, a_{i+1}), i = 0, \ldots k-1$. As already seen in Section 2.1, if $A[R] = v$, then the bitmap index $B = (b_0, b_1, \ldots b_{k-1})$ for this attribute is defined as $b_i = 1 \Leftrightarrow v \in [a_i, a_{i+1})$. Considering a query $Q \equiv$ "$v_1 \leq A[R] \leq v_2''$. We build the bitmap representation of the query as $B_Q = (b_{Q,0}, b_{Q,1}, \ldots b_{Q,k-1})$ such that, for every $i = 0, 1, \ldots k-1$

$$b_{Q,i} = \begin{cases} 1 & \text{if } [a_i, a_{i+1}) \cap [v_1, v_2] \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

If $B \wedge B_Q = (0, 0, \ldots 0)$, then R does not match query Q. On the other hand, if $B \wedge B_Q \neq (0, 0, \ldots 0)$, R *may* satisfy query Q; it is however necessary to compare the exact value $A[R] = v$ of the attribute of R with the query interval $[v_1, v_2]$ in order to know whether R satisfies Q or not. We observe that an exact query $Q \equiv$ "$A[R] = v''$ can be expressed as $Q \equiv$ "$v \leq A[R] \leq v''$, and the corresponding bitmap representation B_Q can be computed as in Eq. 2. Note that this approach is trivially extended to queries represented as boolean combinations of range predicates.

In the search algorithm we propose, each node W forwards a query Q only to neighbors $W' \in \textit{Nb}\,(W)$ if the bitmap indexes $\textit{LinkBitIdx}\,(W \rightarrow W', A)$ satisfy the bitmap representation of Q. This ensures that the query eventually reaches all the resources satisfying it. Also, requests are *not* routed to those WMS whose resources surely won't match the query.

Algorithm 1 describes how queries are routed and processed by each node W. Replies are routed in the opposite direction with respect to the one of query messages.

Algorithm 1 Process Query

Require: W is the WMS executing this program

```
1:  loop
2:    Wait for query Q from WMS W_in
3:    for all W_out ∈ Nb(W) − W_in do
4:      if bitmap representation of Q satisfied by
        LinkBitIdx(W → W_out, A) then
5:        Relay Q to W_out
6:    Wait for all neighbors to reply
7:    if any neighbor reported a match, or there is a local
      match then
8:      Report matches to W_in
9:    else
10:     Report query failed to W_in
```

2.3. Updating the bitmap indexes

Let us suppose that the value of attribute A for resource R changes from v (old value) to v' (new value). The WMS W which is the owner of resource R executes Algorithm 2

Basically, W computes the new bitmap index for the updated value of $A[R]$. If the new bitmap index is equal to the old one, the update is not propagated. If the new bitmap index differs, W sends update messages to all its neighbors.

Algorithm 2 Generate Update Message

Require: W is the WMS executing this program
1: Let v be the old value of $A[R]$
2: Let v' be the new value of $A[R]$
3: **if** $\mathit{BitIdx}(v) \neq \mathit{BitIdx}(v')$ **then**
4: **for all** $W_{out} \in \mathit{Nb}(W)$ **do**
5: Let $B := \mathit{BitIdx}(v')$
6: **for all** $W' \in \mathit{Nb}(W) - W_{out}$ **do**
7: Let $B := B \vee \mathit{LinkBitIdx}(W \rightarrow W', A)$
8: Send B to W_{out}

These updates are computed as to preserve Property 1: for each neighbor W_{out}, the new update index to be sent to W_{out} is computed as:

$$\mathit{LinkBitIdx}(W_{out} \rightarrow W, A) = \mathit{BitIdx}(A[R]) \vee \left(\bigvee_{W' \in \mathit{Nb}(W) - W_{out}} \mathit{LinkBitIdx}(W \rightarrow W', A) \right) \quad (3)$$

Each WMS executes Algorithm 3 when receiving an update message from one of its neighbors W_{in}. Algorithm 3 computes according to Eq. 3 the new bitmap index to be sent to neighbors.

Algorithm 3 Process Update Message

Require: W is the WMS executing this program
1: **loop**
2: Wait for bitmap index B from W_{in} for $A[R]$
3: **if** $B \neq \mathit{LinkBitIdx}(W \rightarrow W_{in}, A)$ **then**
4: Let $\mathit{LinkBitIdx}(W \rightarrow W_{in}, A) := B$
5: **if** $\mathit{BitIdx}(A[R]) \vee B \neq B$ **then**
6: **for all** $W_{out} \in \mathit{Nb}(W) - W_{in}$ **do**
7: Let $B' := (0, 0, \ldots 0)$
8: **for all** $W' \in \mathit{Nb}(W) - W_{out}$ **do**
9: Let $B' := B' \vee \mathit{LinkBitIdx}(W \rightarrow W', A)$
10: Send B' to W_{out}

3. Experimental Results

We now describe some performance measurements on the algorithm described in the previous section. All results were obtained via simulation on a randomly generated tree with $N = 500$ resources; all resources are of the same kind, and each one is bound to a different WMS. Resources have a single attribute which assumes values in the range $[0, 1]$. Initially, all resources are assigned uniformly distributed random values. In our experiments, we consider bitmap indexes of k bits corresponding to partitioning the $[0, 1]$ interval into k disjoint intervals of width $1/k$ each.

First, we performed 100 random updates of the resources and computed the mean number of WMSes affected by the updates. We considered different sizes of the bitmap index, ranging from 8 bits to 64 bits. New values for the attributes are chosen uniformly in $[0, 1]$. Fig. 1 represents the 90% confidence intervals for the mean number of affected nodes as a function of the size of the bitmap index. We see that the number of affected nodes increases as the bitmap size increases. This is due to the fact that longer bitmap indexes imply that each node maintains more accurate information on its neighbor subtrees; if one value gets modified, then the update message is more likely to propagate to a bigger part of the network. Note that the situation we simulated in Fig. 1 can be considered as a worst-case scenario for updates. Variations are likely to be relatively small for most attributes commonly used in Grid systems. Small variations are more likely not to change the bitmap index, so that the update will not be propagated outside the WMS where the update is generated.

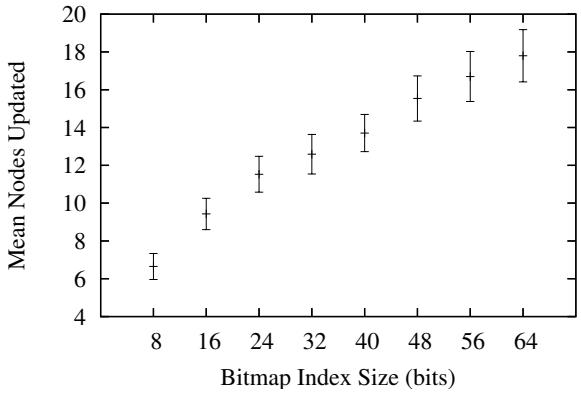

Figure 1. Mean number of nodes affected by 100 consecutive updates (lower is better)

In order to measure query performances, we executed 100 consecutive range queries of the form $v_1 \leq A[R] \leq v_2$, for uniformly chosen $v_1, v_2 \in [0, 1]$. Queries originated from a randomly chosen node. Fig. 2 shows 90% confidence intervals of the average number of nodes which received a query message. Query propagation is reduced by increasing the size of the bitmap indexes. The reason is that larger indexes are more precise, so that a larger number of queries can be be filtered out. In the same figure we also plot the number of query matches, which is the number of nodes whose bitmap indexes match the query. The number of matches on the bitmap indexes will always be greater than or equal to the number of matches to the exact query, because an exact match always implies a match on the bitmap index.

In order to quantify the precision of the search algorithm,

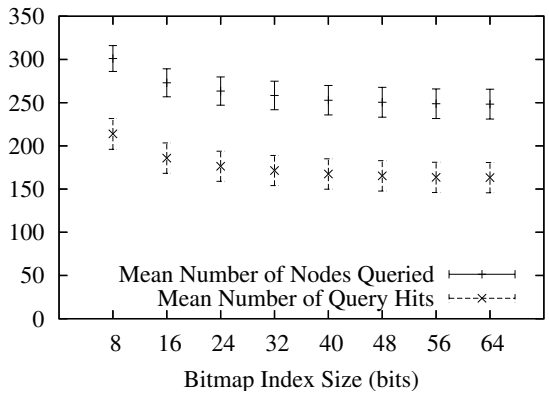

Figure 2. Mean number of nodes affected by 100 consecutive queries (lower is better)

we also considered the ratio:

$$precision = \frac{Number\ of\ exact\ matches}{Number\ of\ matches\ on\ the\ BI}$$

The precision is always less than or equal to one; greater values implies that the match candidates are more likely to be also exact query matches. As we can see in Fig. 3, larger bitmap indexes imply better precision.

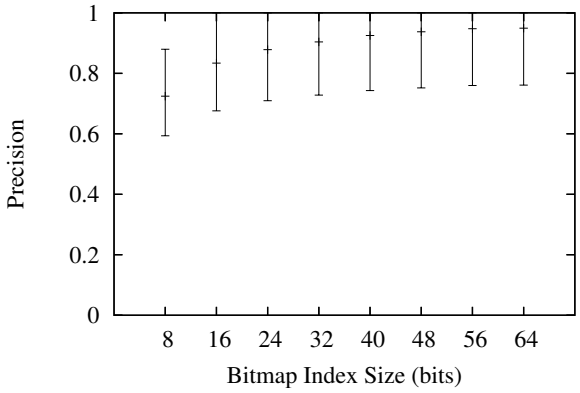

Figure 3. Average precision of 100 consecutive queries (higher is better)

4. Conclusions

In this paper we presented a distributed algorithm for resource location in a dynamic Grid environment. The algorithm uses simple data structures in order to efficiently route resource queries without flooding the network. We performed some simulation studies in order to show the effectiveness of the proposed approach.

Future research will include a more detailed simulation study of the performance of the proposed algorithm with respect to the overlay network topology, the location of resources and the change pattern for their attributes. In particular, as the propagation of queries and updates is likely to be influenced by the topology of the overlay network, we will investigate how nodes can join and leave the P2P network without altering its topological structure. Another open problem which will be investigated is related to limiting the number of links a message is allowed to traverse before being destroyed. In this case users may be unable to get the full list of resources satisfying a query, as potentially useful resources may be beyond the horizon of messages. Clearly, a tradeoff between the value of the time-to-live counter and ability to recall a significant fraction of resources needs to be identified.

References

[1] P. Andreetto et al. Practical approaches to grid workload and resource management in the EGEE project. In *Proc. of CHEP'04*, Interlaken, CH, Sept. 2004.

[2] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.

[3] G. Avellino et al. The datagrid workload management system: Challenges and results. *J. Grid Comput.*, 2(4):353–367, 2004.

[4] I. Foster and A. Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), February 2003, Berkeley, CA*, 2003.

[5] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 15, 2001.

[6] P. O'Neil. Model 204 architecture and performance. In *Proc. of the 2nd lnlernarional Workshop on High Performance Transactions Systems*, number 359 in Lecture Notes in Computer Science, pages 40–59. Springer-Verlag, Asilomar, CA, 1987.

[7] F. Pacini. JDL attributes specification. EGEE Document EGEE-JRA1-TEX-555796-JDL-Attributes-v0-1, 3 Feb. 2005.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. SIGCOMM '01*, pages 161–172. ACM Press, 2001.

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[10] D. Talia and P. Trunfio. Toward a synergy between p2p and grids. *IEEE Internet Computing*, 7(4):94–96, 2003.

[11] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *Proc. of the 22 nd Int. Conference on Distributed Computing Systems (ICDCS'02)*, page 5. IEEE Computer Society, 2002.