Evaluating Resource Discovery Protocols for Hierarchical and Super-Peer Grid Information Systems

Carlo Mastroianni[(1)], Domenico Talia[(2)], Oreste Verta[(2)]
[(1)] *ICAR-CNR 87036 Rende (CS) Italy*
[(2)] *DEIS University of Calabria, 87036 Rende (CS), Italy*
mastroianni@icar.cnr.it, {talia,verta}@deis.unical.it

Abstract

Most currently deployed Grids adopt a hierarchical model for their information system. However, nowadays the research and development community is heading towards the use of scalable models of information services based on decentralized approaches such as the peer-to-peer paradigm. This is mainly due to the poor scalability, resiliency and load-balancing features of the hierarchical model. This paper evaluates a resource discovery protocol exploitable in a hierarchical Grid and compares it with a super-peer based model which has recently been introduced. Performance analysis, carried out through simulation, shows that the hierarchical model is valuable for small and medium sized Grids, while the super-peer model is better suited for very large Grids.

1. Introduction

The information system is an important pillar of a Grid. Users turn to it examine the properties of Grid resources and monitor their availability. The discovery service, provided by the information system, is used to discover the hardware and software resources that are needed to compose and perform a distributed application on a Grid.

Grid users and applications need to get static and dynamic information about available Grid resources. Static information includes CPU speed, operating system, device technology, available compilers and libraries. Dynamic information includes node status such as current CPU load, available disk space, free memory, job queue length, network bandwidth and load, and other similar information. All that information is necessary to efficiently configure and run applications on Grids.

In most Grid frameworks deployed so far, for example in those based on the Globus Toolkit 4 [6], the information system is generally structured according to centralized or hierarchical approaches, mostly because of the client/server approach used today in the largest part of distributed systems and in Web Services frameworks.

Nowadays, the research and development community agrees that the adoption of the peer-to-peer (P2P) paradigm could favour Grid scalability [3, 5, 7]. A hierarchical information system can be viable in a small-scale Grid or within a single organization, but it can become impractical in a large multi-institutional Grid for several reasons, among which:

- fault-tolerance is limited by the presence of a bottleneck at the tree root;
- a significant amount of memory space must be reserved in Index Services to maintain information about a large number of resources, limiting the scalability of the Grid;
- Index Services belonging to different levels must carry very different computation and traffic loads, which leads to challenging problems concerning load imbalance;
- the hierarchical organization can hinder the autonomous administration of different organizations.

Recently, super-peer networks have been proposed [9, 4] to achieve a balance between the inherent efficiency of centralized search, and the autonomy, load balancing and fault-tolerant features offered by distributed search. A super-peer node acts as a centralized resource for a number of regular peers, while super-peers connect to each other to form a network that exploits P2P mechanisms at a higher level. Within each Grid organization, one or more nodes (e.g. those that have the largest capabilities) can act as super-peers, while the other nodes use super-peers to access the Grid and search for resources and services.

The aim of this paper is to discuss the features of the hierarchical model and evaluate the performance of a resource discovery protocol based on this model. Furthermore, a comparison is given with the performance of a super-peer discovery protocol that has been presented and discussed in [4]. A simulation analysis shows that the hierarchical model is valuable for small and medium sized Grids, while the super-peer model is better suited for very large Grids. The outcome of this comparison can be profitably taken into account in designing and deploying information systems of Grids, though the choice of the information system model (hierarchical or peer-to-peer) should be made also according to a number of further considerations such as fault-tolerance and administrative requirements.

The paper is organized as follows. Section 2 discusses the main features of a hierarchical information system based on the GT4 framework. Section 3 evaluates, through a simulation analysis, the performance of the related resource discovery protocol and evaluates the impact that the Grid size and the number of levels that compose the information system hierarchy have on performance results. The performance comparison between the hierarchical and the super-peer discovery protocols is discussed in Section 4. Section 5 discusses related work and Section 6 concludes the paper.

2. Hierarchical Information System

The information system of GT4, based on the Web Service Resource Framework [8], exploits the functionalities of Index Services. An Index Service is a special-purpose Grid service that aggregates and indexes metadata related to the resources provided by a set of Grid hosts belonging to a Virtual Organization (VO). While in a small VO a single Index Service can be sufficient, in a large VO several Index Services can be configured and organized in a hierarchy at different levels.

Figure 1 shows the model discussed in this paper. It assumes that ordinary Grid hosts belong to level 0, while each level 1 Index Service aggregates and publishes all or some of the resources hosted by a group of ordinary Grid nodes. In particular, a GT4 Index Service can subscribe to metadata information (*resource properties*) related to Grid resources by means of proper software mechanisms such as *information providers* and *aggregators*. In general, an Index Service belonging to level *n* aggregates information managed by a group of level *n-1* Index Services, up to the *root* Index Service which aggregates metadata about the resources of the entire organization. If *H* is the height of the tree, it is assumed that the root Index Service belongs to level *H-1*, while ordinary Grid nodes belong to level 0.

In general an Index Service cannot subscribe to all the resources, at least for the following reasons:

i) Since most resources are owned by external organizations, administrative and security reasons may require proper authorization procedures;
ii) It is not convenient to publish very dynamic information which would not be reliable and up-to-date if retrieved by other Index Services instead of Grid nodes directly;
iii) Limits in the memory space and access.

It can therefore be assumed that each Index Service publishes information about a portion of the Grid resources which are provided by lower level Index Services and simple Grid nodes.

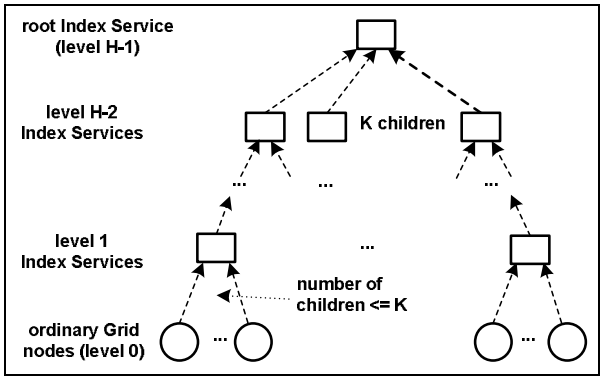

Fig. 1. Architecture of a hierarchical information system based on GT4 Index Services.

A query can be issued by an ordinary Grid node to search for resources belonging to a particular class of resources. Hosts and Index Services should be grouped with the purpose of improving the efficiency of resource discovery queries. In particular, the organization of Index Services should maximize the probability that a query will find useful resources within the local group, i.e. querying the local Index Service. However, the local Index Service can propagate the query towards higher level Index Services, up to the root Index Service, to find more results.

A resource discovery algorithm that exploits the features of the GT4 hierarchical information system is reported in Figure 2. The shown pseudo-code is executed by a generic Index Service when it receives a query from an ordinary node or from a child Index Service. As soon as the query message is received, it is forwarded to the parent Index Service. Then, a local search is performed to find useful resources in the local

sub-tree. Finally, a queryHit including information about the discovered resources is sent to the node from which the query had been received.

```
// MyIndex = parent Index Service
// q.sender: child node from which the query q has been
// received
For each incoming query q:
  forward a copy of q to MyIndex
  <ask local information service for resources matching q>(*)
  if  <there are such resources> {
    send to q.sender a queryHit with information about
      the discovered resources;
    send notifications to the nodes owning the resources;
  }
(*) to avoid duplications, resources owned by the nodes
belonging to the subtree rooted by q.sender are not
considered
```

Fig. 2. Resource discovery algorithm executed by an Index Service.

3. Performance of the Hierarchical Model

Analysis of the hierarchical resource discovery was performed with an object-oriented event-driven simulator written in C++. Simulator objects model the behavior of Grid components and are able to exchange messages among them. Every time an object receives a message/event, it performs a related procedure, according to a finite state automaton, and possibly sends new messages to other objects. The objects defined in the simulator are the following:

- `IndexService`, which executes the resource discovery algorithm described in Section 2;
- `UserAgent`, which generates queries on behalf of users. Each `UserAgent` object is connected to a `Node` object;
- `Node`, which models an ordinary Grid node, transmits user queries to the parent Index Service and receives the related queryHits from it;
- `Event`, which embodies a message exchanged between other objects. An `Event` object is characterized by its source and destination objects, its message delivery time and its type (i.e. query, queryHit, notification);
- `Event Dispatcher`, which manages events, stores them in a queue ordered by message delivery times, and dispatches them to destination objects.

3.1. Parameters and Performance Indices

Table 1 summarizes the simulation parameters used in the simulation analysis. The fist parameter is the Grid size N (i.e. the number of nodes including ordinary Grid hosts and Index Services), which ranges from 10 to 10,000 to take into account small, medium and large Grids. The next three parameters of Table 1 are related to the distribution of resources among Grid hosts and the categorization of such resources.

Table 1. Simulation Parameters.

Parameters	Values
Grid size (number of nodes) ***N***	10 to 10000
Overall number of resources vs. the Grid size, ***Ntot***	$5N$
Overall number of resource classes vs. the Grid size, ***Ncl***	$5(\log_2 N)^2$
Percentage of resource published by Index Services, ***Pres***	25%, 50% and 100%
Mean Query Generation Time ***MQGT***	300 sec
Tree height (number of levels) ***H***	3,4
Tree order ***K***	2 to 100
Time to live ***TTL***	*H–1*
Mean hop time between hosts and Index Services	10 ms
Mean hop time between Index Services	50 ms
Mean time for processing a resource	0.2 ms

Grid users generally need to discover resources that belong to a *class* of resources, rather than a specific single resource. A class of resources is defined as the set of resources that satisfy a number of given constraints on resource properties. In this paper, it is assumed, as in [3], that the average number of elementary resources offered by each single Grid host remains constant as the Grid size increases. This average value was set to *5*, and a gamma stochastic function was used to determine the actual number of resources owned by each node. However, as the Grid size increases, it becomes more and more unlikely that a new node connecting to the Grid provides resources belonging to a new resource class. Therefore, it is assumed [4] that the overall number of distinct resource classes offered by the Grid does not increase linearly with the Grid size. A logarithmic distribution is adopted: the number of resource classes offered by a Grid network with N nodes (where N ranges from 10 to 10000) is set to $5*(\log_2 N)^2$. As an example, a Grid having 1024 nodes provides 5120 resources belonging to 500 different classes. The percentage of Grid resources published in Index Services, named *Pres*, is set to 25%, 50% and 100% in different simulation runs. During a simulation run, each Grid host generates a set of queries. The mean query generation time MQGT, i.e. the average interarrival time between two successive query requests issued by the same node, is set to 300 seconds. For each generated query, the

simulator randomly selects the class of the resources that the user needs to discover.

The value of the tree height H (see Figure 1) was set to 3 and 4. This choice comes from the observation that a value of H equal to 2 would reduce the Grid to a simple Grid Organization, while values of H larger than 4 would mean the presence of more than 3 hierarchical layers of Index Services, which is very unlikely in current Grids. The tree order K is chosen as follows: once the Grid size N and the tree height H are given, K is set to the minimum value that satisfies the equation (1) reported below (in which the first member calculates the overall number of nodes in a complete tree with height H and order K).

$$(1)\ \frac{K^H - 1}{K - 1} > N$$

Furthermore, it is assumed that the load imbalance among the Index Services belonging to the same level is limited. This assumption has been made to evaluate the performance of an Index Service with a strong statistical accuracy. However, processing and traffic loads carried by Index Services belonging to different levels can differ. In particular, it is assumed that Index Services have K children, except for level 1 Index Services that have as many children as are necessary to let the number of nodes reach the value N. For example, in a Grid with 2000 nodes and three levels of Index Services (i.e. with $H=4$), K is set to 13, since with this value the first member of equation (1) is equal to 2380. The upper three levels contain 183 Index Services; to have exactly 2000 nodes, level 0 must contain 1817 Grid hosts. Therefore, the 169 level 1 Index Services are connected to 10.75 hosts on average. Table 2 reports the values of the parameters N, H and K that were computed through the tree building approach described so far. The time to live of query messages TTL is set to $H-1$, thus allowing queries to explore the entire hierarchy**.**

Finally, it is assumed that the mean hop time (i.e., the mean amount of time necessary to send a message from one node to another) is equal to 10 ms for a hop between an ordinary node and the parent Index Service, and to 50 ms for a hop between two connected Index Services. The mean time needed to compare the query constraints with the properties of a single resource stored in the Index Service, hence to verify if such a resource belongs to the class specified within the query message, is set to 0.2 ms, in accordance with the values obtained in real Grids [2].

Table 2. Values of Grid size, tree height and tree order used in the simulations.

Grid size N	Values of K with H=3	Values of K with H=4
10	3	2
20	4	3
50	7	4
100	10	5
500	22	8
1000	35	10
2000	45	13
5000	75	17
10000	100	22

The performance indices used to evaluate the performance of the resource discovery protocol are shown in Table 3. The mean number of results *Nres* is particularly significant since it is generally argued that the *satisfaction of the query* depends on the number of discovered resources sent back to the user that issued the query: for example, in [10] a resource discovery operation is considered *satisfactory* only if the number of results exceeds a given threshold.

Table 3. Performance indices.

Performance index	Definition
Mean number of results, ***Nres***	Mean number of useful resources that a node discovers after its query.
Message load, ***L***	Frequency of messages received by a node (messages/sec).
Response time ***Tr***, ***Tr(1)***	Mean amount of time (sec) that elapses between query generation and reception of a generic result and of the first result.
Memory space ***Sm***	Amount of memory necessary to maintain information about published resources

Since every query can reach the root Index Service, the *Nres* index can be obtained with formula (2), as reported below. Note that *Nres* increases with the Grid size, since the overall number of resources *Ntot*, which is a linear function of N, increases more rapidly than the number of resource classes *Ncl*, which is a logarithmic function of N.

$$(2)\ \mathrm{Nres} = \frac{\mathrm{Ntot} * \mathrm{Pres}}{\mathrm{Ncl}}$$

The message load L, defined as the frequency of messages received by a single Grid node, should obviously be kept as low as possible. This performance index often counterbalances the number of results, in

the sense that high success probabilities are achievable at the cost of having high elaboration loads.

Response times are related to the *time to satisfaction* experienced by a user. Finally, the memory space *Sm* is the amount of memory that must be reserved by a node to maintain information about the published Grid resources. Here, a further assumption is that, for each resource, the related metadata information requires 10 KB of data on average.

3.2. Performance Results

This section focuses on two performance indices, load and response times, whereas the other two indices (number of results and memory size) are discussed in section 4, where their values are compared to those obtained with the super-peer model.

Figures 3 and 4 report the load experienced by hosts belonging to all the different levels of the Grid hierarchy. Figures are related, respectively, to Grids having 3 and 4 levels. While the message load experienced by ordinary hosts is relatively low and approaches a saturation level, the message load at Index Services increases about linearly with the Grid size, so confirming the poor scalability properties of the hierarchical architecture. It is also interesting to notice that the root node suffers the largest load for small- and medium-sized Grids but, in very large Grids, the load experienced by the Index Services located immediately below the root node (i.e. belonging to level H-2) approaches or even exceeds (with H =4) the load of the root node. The reason is that the H-2 level Index Services receive query messages from an increasing number of child nodes, as well as queryHit messages from the root Index Service, whereas the root node receives only query messages. In conclusion, intermediate Index Services undergo a very heavy load that can be even worse than that of the root node, which is an interesting and perhaps non predictable outcome of the simulation analysis.

Figure 5 reports the average response times, versus the Grid size, obtained with hierarchical Grids having a tree height equal to 4, for different values of the percentage of resources published by Index Services, *Pres*. As the Grid size increases, and consequently the number of resources that must be considered when processing a query, response times become longer, up to almost 20 seconds for a Grid with 10,000 nodes. In large Grids, the value of *Pres* has a strong impact on response times since processing times are the major component of the response time, whereas delay times are negligible being in the order of milliseconds.

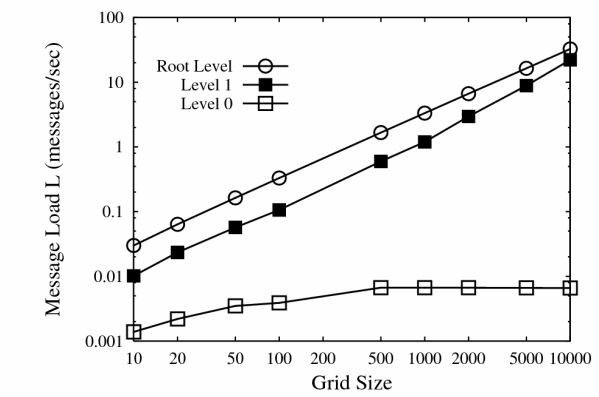

Fig. 3. Message load of Index Services belonging to different levels in a Grid with tree height H=3.

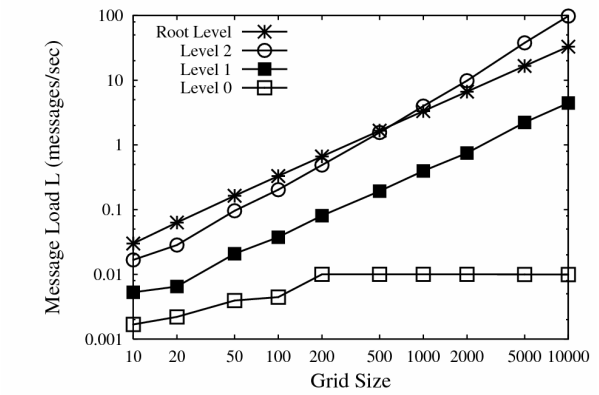

Fig. 4. Message load of Index Services belonging to different levels in a Grid with tree height H=4.

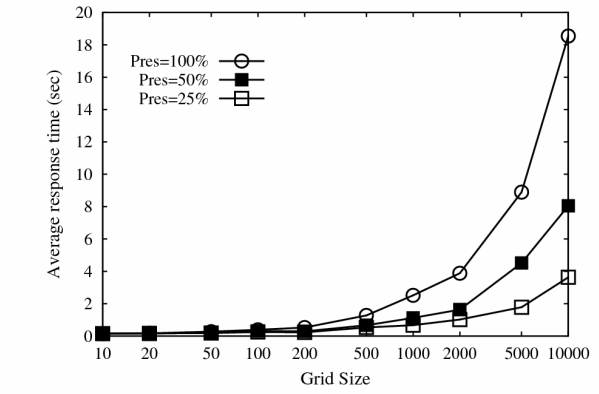

Fig. 5. Average response times in a Grid with tree height H=4.

The response time of the first result is shown in Figure 6. This trend is similar to that of the average response time only for small Grids; in fact, when the overall number of resources is small, it is very likely that the

first result is found in the root Index Service. However, as the Grid size increases, a query issued by an ordinary host has more and more chances to find results in its parent Index Service, which makes the response time of the first result decrease to a very small value. In fact, a very abrupt curve drop is observed between values of the Grid size comprised between 100 and 200. Finally, for even larger Grids, the response time of the first result increases again, the consequence of larger processing times experienced by the Index Services located at level 1. The impact of processing time is also the reason for which the response time of the first result increases with the value of *Pres*.

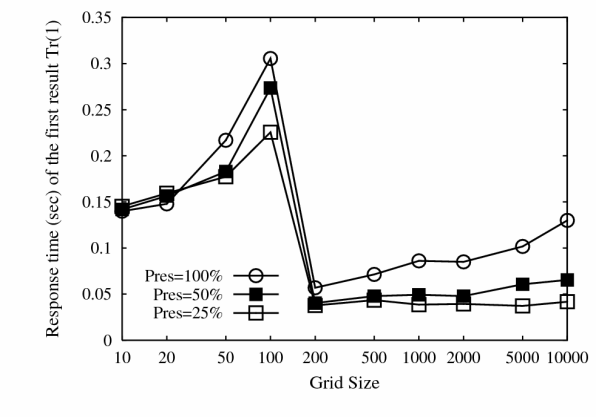

Fig. 6. Response times of the first result in a Grid with tree height H=4.

4. Comparison between the Hierarchical and the Super-Peer Model

This section discusses and compares the results obtained with the hierarchical model and the results obtained with an information model based on the super-peer approach.

In a super-peer network, for each Grid organization, a subset of powerful nodes having high availability properties can be identified; these nodes can be elected as super-peers, so exploiting their advanced computational performance. A super-peer network can be seen as a P2P network that interconnects super-peers, where each super-peer acts also as a server for a number of ordinary peers. A super-peer accomplishes two main tasks: it is responsible for the communications with the other Grid organizations, and it maintains metadata about all the nodes of the local organization. When a peer needs to explore the network to find useful resources, it sends a query message to the local super-peer, which performs the following operations: (i) it searches the information system of the local organization and returns discovered results; (ii) it forwards the query to neighbour super-peers which in turn will perform similar operations. Whenever a resource matching the query criteria is found in a remote Grid organization, a queryHit is generated and is forwarded along the same path to the requesting node.

Here it is assumed that each super-peer is connected, on average, to 9 ordinary peers (thus forming clusters of 10 nodes on average), and to 4 adjacent super-peers. The TTL value is used as a parameter: larger values of TTL allow for a wider exploration of the network. Further details about the super-peer protocol can be found in [4].

Figure 7 compares the number of results obtained after issuing a query, with the hierarchical model (labelled with `HR`) and with the super-peer model (labelled with `SP`). For the hierarchical model, results were obtained with different values of *Pres*, the percentage of resources published by an Index Service. With the super-peer model, the number of reachable resources depends on the network topology and the TTL value. The values of `Nres` obtained in the hierarchical model with *Pres* =100% can be considered as an upper bound for the values achievable with both the architectures, since all the useful resources can be directly found in the root Index Service. It is worth mentioning that a *Pres* equal to 100% is not likely in real Grids, while values of 50% or 25% are more realistic. Indeed the Index Services generally belong to different administrative domains, and an Index Service cannot publish all the information maintained by another Index Service. Conversely, this is not an issue in super-peer networks, because each super-peer only publishes information related to the resources of the local Grid organization.

With the super-peer model, the number of results can be increased by increasing the TTL value, thus permitting to explore a larger number of super-peers. It is interesting to note that if a proper value of the TTL parameter (i.e., 5) is set in the super-peer model, the average number of results exceeds the one achieved by using the hierarchical model with *Pres*=25%, for all the considered values of the Grid size. Of course if *Pres* is even lower than 25%, the use of the super-peer model becomes more and more effective.

In Figure 8, the load of a generic super-peer is compared with the load experienced by the Index Services located at the two highest layers of the hierarchy. The figure shows that the super-peer load increases with the TTL value and in most cases it is higher than the load at the root Index Service. The reason is that the root Index Service receives only query messages, while a super-peer receives also queryHit messages coming from the neighbour super-

peers. However, as the Grid size increases, the super-peer load reaches a saturation level whereas the root Index Service load increases with a notably slope and exceeds the super-peer load for Grids having 10000 nodes or more. Figure 8 also shows the very high load that is carried - in very large Grids - by the Index Services located below the root node of the hierarchy (level 2), which confirms the better scalability features of the super-peer model.

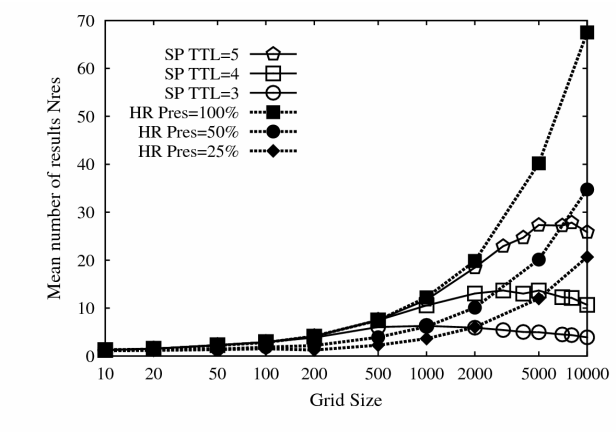

Fig. 7. Mean number of results obtained with the hierarchical model (HR) and the super-peer model (SP).

An interesting consideration concerns the memory space. In the super-peer model the amount of memory that must be reserved on a super-peer to maintain information about Grid resources does not depend on the Grid size but exclusively on the number of peers connected to the super-peer. With the discussed assumptions (cluster size equal to 10, mean size of a metadata document equal to 10 KB, and mean number of resources published by each peer equal to 5), *Sm* is equal to about 500 Kbytes at a super-peer, while it is equal to $Sm = 5 * N * Pres * 10$ KB at the root Index Service of the hierarchical architecture. For example, with N=10,000 and *Pres*=0.50, *Sm* is equal to 250 Mbytes. Therefore, the hierarchical model requires a much larger amount of main memory. Even if such an amount of memory space can be available in the server machine that runs the Index Service, the analysis of metadata information may require a large amount of time. Having assumed that the time to process a single resource is set to 0.2 ms, Figure 9 reports the response times experienced with the two models, with *Pres*=50% in the hierarchical model. As the Grid size increases it is clear that the Index Services of the hierarchical framework spend a lot of time in processing a large number of resources, leading to average response times which are much longer that those experienced in the super-peer model.

In conclusion, the hierarchical model can be deemed suitable only for small- and medium-sized Grids. However in a large Grid the advantage related to the larger number of results that can be obtained with the hierarchical model is often surmounted by its larger costs in terms of processing load and response times with respect to the super-peer model.

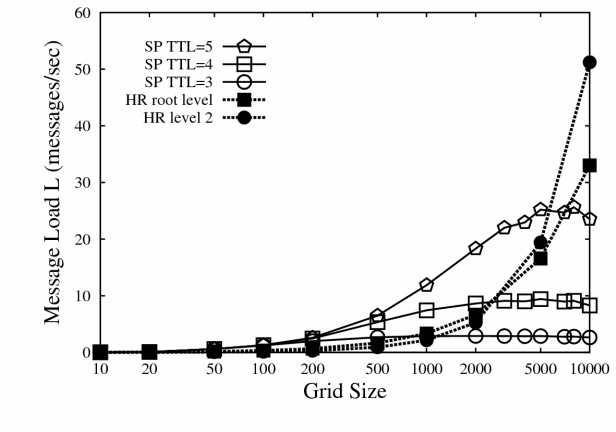

Fig. 8. Message load experienced by the Index Services belonging to the two highest levels of the hierarchical model (HR) and by super-peers (SP).

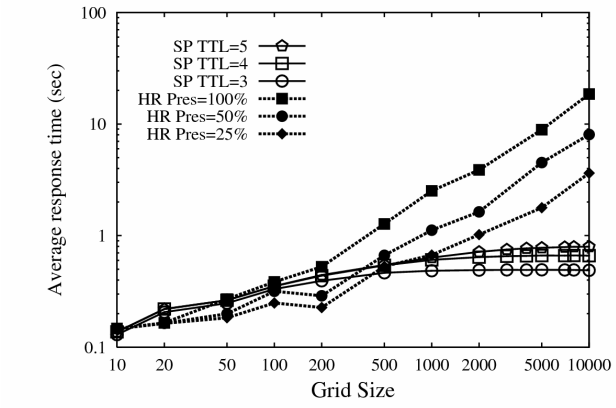

Fig. 9. Average response times obtained with the hierarchical model (HR) and the super-peer model (SP).

5. Related Work

In most of the Grid frameworks deployed so far, the information system is generally structured according to centralized or hierarchical approaches. For example, the recently released Globus Toolkit 4 is based on the Web Service Resource Framework (WSRF) [8], which fully exploits the Web services paradigm. The central

component in the GT4 information system [6] is the Index Service, which collects information about Grid resources and makes this information available to users and applications. An Index Service can register to information published by other Index Services, offering the possibility to build an overall information system according to a hierarchical, peer-to-peer or hybrid architecture. However, the hierarchical model is still the most frequently used in currently deployed GT4 Grids, mostly because of the client/server approach used today in the largest part of distributed systems and in Web services frameworks.

Nowadays, the research and development community agrees that the adoption of the P2P paradigm could favour Grid scalability [5, 7]. The super-peer model has been originally proposed in [9] to achieve a balance between the inherent efficiency of centralized search, and the autonomy, load balancing and fault-tolerant features offered by distributed search. This model was adopted in [4] to design a P2P-based Grid information service. The super-peer model is advantageously exploited in the Grid context because it is naturally appropriate for large scale Grid environments. In fact, a large-scale Grid can be viewed as a network interconnecting in a P2P fashion a number of small-scale, proprietary Grid organizations.

In a Grid, users often need to find a number of resources belonging to a given class, so that they can subsequently select the best resource for their job. A resource class can be seen as a set of resources satisfying a given set of constraints on the values of resource parameters. The work reported in this paper assumes that a given classification of resources is available and known to the user. Classes can be determined with the use of Hilbert curves that represent the different parameters on a single dimension [1].

6. Conclusions

This paper discusses and evaluates a resource discovery protocol for a Grid information system designed according to a hierarchical approach. Comparison has been done with a decentralized discovery protocol suited for the recently introduced super-peer model. Performance comparison shows that the hierarchical model is valuable for small and medium sized Grids, while the super-peer model is more effective in very large Grids. The reported analysis, in terms of average number of results, processing and memory load and response times, can be profitably used by designers and developers of the information system of Grids. In addition to these results, a number of further considerations (concerning fault-tolerance, scalability, data registry size, load balancing, administrative features, etc.), generally favour the use of the super-peer paradigm, especially for very large multi-institutional Grids.

Acknowledgements

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265)

References

[1] A. Andrzejak and Z. Xu, “Scalable, efficient range queries for grid information services”, Proc. of the 2nd IEEE International Conference on Peer-to-Peer Computing, Linkping University, Sweden, 2002.

[2] P. Hasselmeyer, “Dynamic Distributed Registries and Protocols”, the NextGRID Project: Architecture for Next Generation Grid, Work Package 5, Grid Dynamics, Document P.5.2.1, September 2005.

[3] A. Iamnitchi, I. Foster, J. Weglarz, J. Nabrzyski, J. Schopf and M. Stroinski, “A Peer-to-Peer Approach to Resource Location in Grid Environments”, eds. Grid Resource Management, Kluwer Publishing , 2003.

[4] C. Mastroianni, D. Talia and O. Verta, “A Super-Peer Model for Resource Discovery Services in Large-Scale Grids”, Future Generation Computer Systems, Elsevier Science, Vol. 21, No. 8, 2005, pp. 1235-1456.

[5] D. Puppin, S. Moncelli, R. Baraglia, N. Tonellotto and F. Silvestri, “A Grid information Service Based on Peer-to-Peer”, Proc. of the 11th International EuroPar Conference, Lisbon, Portugal, August-September 2005, pp. 454-464.

[6] J. M. Schopf, M. D'Arcy, N. Miller, L. Pearlman, I. Foster and C. Kesselman, “Monitoring and Discovery in a Web Services Framework: Functionality and Performance of the Globus Toolkit's MDS4”, Argonne National Laboratory Technical Report ANL/MCS-P1248-0405, April 2005.

[7] D. Talia and P. Trunfio, “Towards a Synergy between P2P and Grids”, IEEE Internet Computing 7(4), 2003, pp. 94-96.

[8] The Web Services Resource Framework, http://www.globus.org/wsrf.

[9] B. Yang and H. Garcia-Molina, “Designing a Super-Peer Network”, Proc. of the 19th International Conference on Data Engineering, Los Alamitos, CA, USA, March 2003.

[10] B. Yang and H. Garcia-Molina, “Efficient Search in Peer-to-Peer Networks”, Proc. of the 22nd International Conference on Distributed Computing Systems, Wien, Austria, July 2002.