Parallel Computing 34 (2008) 593–611

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Designing an information system for Grids: Comparing hierarchical, decentralized P2P and super-peer models

Carlo Mastroianni [a,1], Domenico Talia [b,*], Oreste Verta [b]

[a] ICAR-CNR, 87036 Rende (CS), Italy
[b] DEIS, University of Calabria, 87036 Rende (CS), Italy

ARTICLE INFO

Article history:
Received 22 March 2007
Received in revised form 1 February 2008
Accepted 21 July 2008
Available online 30 July 2008

Keywords:
Grid
Information system
Index Service
Peer-to-peer
Performance evaluation
Resource discovery
Super-peer

ABSTRACT

As deployed Grids increase from 10s to 1000s of nodes, the construction of an efficient and scalable information system is a key issue, as it is vital for providing querying and discovery services. Today most Grids adopt a centralized or hierarchical model for their information system, but this model is characterized by poor scalability, resiliency and load-balancing features. Nowadays the research and development community is heading towards the use of scalable information systems based on distributed models such as the decentralized peer-to-peer (P2P) model and the super-peer model. If the former is adopted, each node can act both as a client and a server as it can generate discovery requests and also respond to requests issued by other peers. Requests and responses are forwarded with a hop-by-hop mechanism by ad hoc Grid Services hosted by Grid nodes. The super-peer model is a recently proposed approach that combines features of centralized and P2P models. A super-peer acts as a server for a single Grid organization, and publishes metadata describing the resources provided by the nodes of that organization. At the same time, super-peers connect to each other to form a P2P network at a higher level. This paper analyzes information systems based on three alternative models: the hierarchical, the decentralized P2P, and the super-peer model. A performance evaluation of such models is reported, and afterwards a performance comparison is discussed in order to analyze the pros and cons of each solution.

1. Introduction

Grid computing is an emerging computing model that provides the ability to perform higher throughput computing by taking advantage of many networked computers and distributing process execution across a parallel infrastructure. Grids use the resources of many separate computers connected by the Internet network to solve large-scale computation problems. In a Grid computing environment, the set of hosts and resources available can change over the time: new resources and services may be added, existing ones can be removed, basic properties of a resource or a service may also change. The *information system* is an important pillar of a Grid framework since it provides information, about static and dynamic Grid resources, which is critical to the operation of the Grid and the construction of applications. In particular, users turn to the information system to discover suitable resources that are needed to design and execute a distributed application, explore the properties of such resources and monitor their availability.

* Corresponding author. Tel.: +39 0984 494726; fax: +39 0984 494713.
E-mail addresses: mastroianni@icar.cnr.it (C. Mastroianni), talia@deis.unical.it (D. Talia), verta@si.deis.unical.it (O. Verta).
[1] Tel.: +39 0984 831725; fax: +39 0984 839054.

0167-8191/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.parco.2008.07.001

A key service in Grid information systems is the *resource discovery* service. The resource discovery problem can be defined as follows: given a set of user/application requirements and constraints, a Grid discovery service must find the resources that match those requirements and constraints. Grid users and applications need to get static and dynamic information about hardware and software 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, the information system is structured according to centralized or hierarchical approaches, mostly because the client/server approach is still used today in the largest part of distributed systems and in Web services frameworks. For example, the Globus Toolkit, the standard de facto Grid framework, in its largely diffused version 2 adopted a strictly hierarchical architecture for its monitoring and discovery system (MDS) [1,2]. In MDS-2, a GRIS (Grid Resource Information Service) gathers and publishes information about the resources offered by each Grid host, while a GIIS (Grid Index Information Service) collects information from a set of GRIS belonging to the same Grid Organization. The hierarchical architecture results from the possibility of having high order GIIS servers which can aggregate information published by lower order GIIS servers.

The service-oriented approach of currently adopted Grid frameworks allows for adopting different paradigms for the construction of an information system. The recently released Globus Toolkit 4 is based on the Web Service Resource Framework (WSRF [3]), which fully exploits the Web services paradigm, thus facilitating the convergence of Grid and Web services worlds. The central component in the GT4 information system [4] is the Index Service, which collects information about Grid resources and makes this information available to users and clients. The Index Service retrieves information from multiple data sources via standard WSRF subscription/notification interfaces. Furthermore, an Index Service can publish information retrieved by other Index Services, giving the possibility to build the information system according to hierarchical, decentralized or hybrid architectures, even though the hierarchical model is still the most frequently used [4,5].

Nowadays, the research and development community agrees that the adoption of the peer-to-peer (P2P) paradigm could favor Grid scalability [6–9]. Indeed, a hierarchical information system can be viable within a single organization or in a small-scale Grid, 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.

Fortunately, Grids and P2P systems share several features and can profitably be integrated, bringing benefits in both fields and resulting in future convergence. The introduction of the Web services resource framework is particularly useful for this convergence trend, since WSRF Grid Services can support P2P interactions between hosts belonging to different domains. Accordingly, the Grid information system can be built according to a *fully decentralized* P2P model [10,11]. A discovery request, issued on a Grid host, travels the Grid by exploiting P2P interconnections among Grid hosts, through the invocation of special-purpose Grid Services.

The decentralized approach can enhance scalability and fault-tolerance with respect to the hierarchical approach, but can also induce a very large network traffic and may limit search effectiveness. Recently, the *super-peer model* has been proposed [12,13] 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 the P2P mechanisms at a higher level. The super-peer model is naturally appropriate for Grids, as a large-scale Grid can be viewed as a network interconnecting small-scale, proprietary Grids, also referred to as Physical Organizations (POs). Within each PO, one or more nodes (those that have the largest capabilities) act as super-peers, while the other nodes use super-peers to access the Grid and forward discovery requests.

This paper discusses and evaluates three Grid resource discovery protocols that are adopted with the three different models discussed so far: hierarchical, decentralized P2P, and super-peer. The goal of such protocols is to let a user issue a query for resources having given characteristics and discover a number of suitable resources matching the specified requirements.

Performance evaluation is carried out by means of an event-based simulation framework, which allows for the evaluation of Grids having a large number of hosts. Performance results can be used to tune the parameters of each protocol, in order to increase search efficiency. Furthermore, results allow to compare the different models and evaluate their pros and cons in different Grid configurations. The outcome of this comparison is of value for Grid designers and administrators. Appropriate design and deploy decisions may be made by taking into account the reported performance results, as well as a number of further considerations such as administrative requirements.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces the three information system models and the corresponding resource discovery algorithms and protocols, and shows how such models can be implemented in a service-oriented Grid framework. Section 4 reports the performance results obtained with the three

resource discovery protocols. Section 5 compares such performance results and analyzes the impact of some important network parameters, such as the time needed to process queries and the average size of Grid Organizations. Section 6 concludes the paper.

2. Related work

The *resource discovery service* is a key component of both Grid and P2P systems. In a Grid the discovery service is invoked by users when they need to discover and use hardware or software resources matching given characteristics. It often occurs that this service must to be invoked several times prior to or during the execution of a complex Grid application, because the latter is generally composed of several sub-tasks that require different resources for their efficient execution.

In most of the Grid frameworks deployed so far, the information system is generally constructed according to centralized or hierarchical approaches, as discussed in the Introduction. The Index Service, which is the central component in the information system of the Globus Toolkit 4, can register to information published by other Index Services, offering the possibility to build an overall information system according to a P2P or hybrid architecture. However, the hierarchical model is still the most frequently used in currently deployed GT4 Grids [4,5], mostly because of the client/server approach used today in the largest part of distributed systems and in Web services frameworks.

The hierarchical model can be efficient when it is necessary to compute aggregate values, that is, global information including average load, amount of available memory, location and description of hotspots, and so on. Perhaps the best-known example of this approach is Astrolabe [14]. In this system a hierarchical architecture is deployed, which reduces the cost of finding the aggregates and enables the execution of complex database queries. On the other hand, maintenance of the hierarchical topology introduces additional overhead, which can be significant if the environment is strongly dynamic.

In the general case, when the discovery service aims to provide specific information about single Grid resources, centralized and hierarchical approaches to Grid information systems do not guarantee scalability and fault-tolerance, in particular as the Grid size increases and Grids become pervasive. Nowadays, the research and development community agrees that the adoption of the P2P paradigm could favor Grid scalability [6–9]. A fully decentralized P2P architecture for resource discovery in Grid environments is proposed in [10]. In this architecture every participant in a Grid Organization publishes information about local resources. Users send their requests to some known (typically local) node, which responds with matching resource descriptions if it has them locally, and forwards the requests to neighbor nodes, and so on, until the TTL of the requests expires. If a node has information matching a forwarded request, it sends the response directly to the node that initiated the forwarding.

This approach is based on an *unstructured* P2P system, since resource descriptors are published by peers without a global pattern or structure. Conversely, in *structured* P2P systems [15], resources are assigned to hosts with a well-specified strategy, for example through a distributed hash table (DHT), by which the correct location of a resource on the network is obtained as the result of a hash function. Gnutella and KaZaA are examples of unstructured P2P networks, while CAN [16], Chord [17] and Pastry [18] are examples of structured P2P networks. The structured approach generally speeds up discovery operations but also presents important drawbacks [19,15], such as: (i) limited fault-tolerance and scalability, since the disconnection of a peer requires an immediate reorganization of resources and of peer index tables; (ii) poor load balancing, because peers that are assigned the descriptors of the most popular resources can be easily overloaded. Overall, structured algorithms are usually efficient in file sharing P2P networks, but structure management can be cumbersome and poorly scalable in large and dynamic Grids, especially when the churn rate – the frequency of peer disconnections – is high. Moreover, structured algorithms may hinder the discovery of resources when they are not specified with a name or a code (for example the name of a file) but are defined with looser constraints, for example when the CPU speed of a computer or the response time of a Web service are required to be within a given interval. The latter case is much more frequent in Grids than in file sharing systems. Therefore, unstructured algorithms are valuable in Grids, which is the reason why they are considered in this paper.

The super-peer model was proposed in [12] to achieve a balance between the inherent efficiency of centralized search, and the autonomy, load-balancing and fault-tolerant features offered by distributed search. In [12] performance of super-peer networks is evaluated, and rules of thumb are given for an efficient design of such networks: the objective is to enhance the performance of search operations and at the same time to limit bandwidth and processing load. In [20] a general mechanism for the construction and the maintenance of a super-peer network is proposed and evaluated. In this work, a gossip paradigm is used to exchange information among peers and dynamically decide how many and which peers can efficiently act as super-peers. The super-peer protocol adopted by the KaZaA system http://www.kazaa.com designates the more available and powerful peers as supernodes. In KaZaA, when a new peer wants to join the network, it searches for the existing supernodes, and establishes an overlay connection with the supernode that has the shortest RTT. In [21] a framework that combines the structural DHT design with a multi-level architecture based on super-peers is proposed. Peers are organized in disjoint groups, and lookup messages are first routed to the destination group, then to the destination peer using an intra-group overlay. In [22] both resources and the content stored at peers are described by means of RDF metadata. Routing indices located at super-peers use such metadata to perform the routing of queries expressed through the RDF-QEL query language. The super-peer based Grid Information Service proposed in [23] uses an Hop Counting Routing Index algorithm to exchange queries among the super-peers and, in particular, to select the neighbor super-peers that offer the highest probability of success.

Strictly related to resource discovery is the *resource matchmaking* issue, which is a key feature in job submission systems, such as Condor [24]. The Condor framework receives job requests from users and then finds suitable, available resources on the network where these jobs can be executed. The scheduling task is performed by a *Central Manager* (CM). The CM collects information about the state of resources and receives the users' requests. The CM then matches the resource description and usage policies, specified by the resource owners, with the jobs needs and decides where to schedule them. Both resource and job descriptions are expressed using the *ClassAd* specification language. This language enables the specification of job and resource attributes along with users and resource owner policies and preferences. Users often need to find several 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, for example of *ClassAd* properties. This paper assumes that a given classification of resources is available and known to the user. This assumption is common in similar works. For example, in [25], performance of a discovery technique is evaluated by assuming that resources have been previously classified in four disjoint classes. Classification can be done by characterizing the resources on the basis of a set of parameters that can have discrete or continuous values. Classes can be determined with the use of Hilbert curves that represent the different parameters on a single dimension [26]. Alternatively, an n-dimension distance metric may be exploited [27] to determine the similarity among resources, and then assign similar resources to the same class.

3. Models for the Grid information system: hierarchical, decentralized and super-peer

This section introduces the three different models of Grid information systems which are examined and discussed in this paper. For each of these, the envisaged architecture is discussed and the related resource discovery algorithm is described.

3.1. Hierarchical information system

The hierarchical model is the most frequently adopted in Grid frameworks. For example, the information system of GT4, today the most used Grid middleware, 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 the Grid hosts of a Grid Organization. In a large organization, as well as in a Grid that spans different organizations, several Index Services can be configured and organized in a hierarchy. Fig. 1 shows a hierarchical information system having a tree height equal to H. Regular Grid hosts belong to level 0, while each level 1 Index Service aggregates and publishes information about all or some of the resources provided by a group of regular Grid nodes. An Index Service can subscribe to metadata information 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 retrieved from a group of level $n-1$ Index Services, up to the *root* Index Service, located at level $H-1$, which aggregates metadata about the resources of the entire organization. In most cases, an Index Service cannot maintain metadata information related to all the resources of descendant Index Services, at least for the following reasons:

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

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

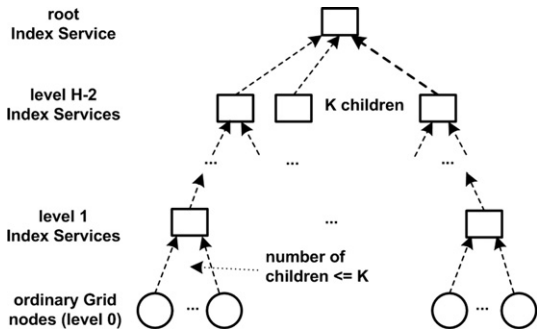

Fig. 1. Architecture of a hierarchical information system based on Index Services.

A query can be issued by a regular Grid node to search for resources belonging to a particular class of resources. The query is first delivered to the local Index Service, which can subsequently propagate it towards higher level Index Services, up to the root Index Service, in order to find more results. A resource discovery algorithm that exploits the features of the GT4 hierarchical information system is reported in Fig. 2. The pseudo-code shown in this figure is executed by a generic Index Service when it receives a query from a regular 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 of the information system. Finally, a queryHit including information about the discovered resources is sent to the node from which the query has been received.

3.2. P2P decentralized information system

As discussed in previous sections, an information system based on the P2P paradigm can be more effective and fault-tolerant than a centralized or hierarchical one. In order to maximize the benefits related to the adoption of the P2P approach in a Grid information system, it is useful to compare the characteristics of Grids and commonly used (e.g. file sharing) P2P networks. In particular, two main differences emerge:

(1) Grids are less dynamic than P2P networks, since Grid nodes and resources often belong to large enterprizes or public institutions and security reasons generally require mutual authentication of Grid nodes before accessing external resources.
(2) Whereas in a P2P network tailored to file sharing applications users usually search for well defined resources (e.g. MP3 or MPEG files), in Grid systems they often need to discover software or hardware resources that belong to a particular resource class, where a resource class is defined as the set of resources that satisfy a number of given constraints on resource properties.

In a decentralized P2P system hosts are arranged in a P2P overlay network. Adjacent peers, i.e. peers that can directly communicate through an overlay P2P connection, can belong to the same Grid organization or to different organizations. To model the network topology and approximate the behavior of a real Grid as much as possible, the well known power-law algorithm defined by Barabási and Albert [28] is adopted in this study. This model incorporates the characteristic of preferential attachment that was proved to exist widely in real networks. Specifically, the network is constructed starting from a few disconnected nodes, then one node is added at each step, and this node is connected, through P2P links, to a number of existing nodes. These nodes are chosen according to the "rich get richer" paradigm: the more connected a node is, the more likely it is to receive new links from new nodes. A very useful feature of this model is that it is *scale-free*, since the probability distribution of the number of ingoing and out-going links (the *in-degree* and *out-degree* of a node) does not depend on the size of the network.

The *discovery service* is based on the exchange of *query* and *queryHit* messages among hosts. Whenever a user initiates a search procedure, the client process of the corresponding peer sends a query message to a selected number of neighbors, which subsequently forward it to their own neighbors. As a node receives a query message, it checks whether it possesses one or more resources belonging to the requested class; if so, it sends a queryHit message that will follow the same path back to the node that issued the query.

To limit the traffic load, a query is only forwarded to a maximum number of neighbors. If the out-degree of a node is higher that this number, the neighbors are chosen through a statistical approach: a node forwards a query to the neighbor nodes from which the largest numbers of queryHits were received in the past [23]. A number of further techniques are adopted to reduce the network load. (i) The number of hops is limited by a time-to-live (TTL) parameter. (ii) Each query message contains a field used to annotate the nodes that the query traverses along its path. A peer does not forward a query to a neighbor peer that has already received it. (iii) Each peer maintains a cache memory where it annotates the IDs of the last received

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

Fig. 2. The resource discovery algorithm executed by an Index Service in the hierarchical model.

query messages. A peer discards the queries that it has already received. Techniques (ii) and (iii) are used to avoid the formation of cycles in the query path, and are complementary, since technique (ii) can *prevent* cycles in particular cases (i.e. when a query, forwarded by a peer, is subsequently delivered to the same peer), whereas technique (iii) can *remove* cycles in other cases (e.g. when two copies of a query, sent by a peer A to two distinct neighbor peers B and C, are both subsequently delivered to the remote peer D).

Fig. 3 depicts how the decentralized P2P model can be implemented in the GT4 framework. Each peer contains a GT4 Index Service, which aggregates metadata information (*Resource Properties*) related to a number of Grid resources (in GT4 referred to as *WS-Resources*), and a Peer Service. A Peer Service is a static Grid Service that processes query messages coming from remote peers; when receiving a query, it asks the local Index Service to find resources matching the query constraints. A Peer Service forwards query and queryHit messages through the Network Module.

3.3. Super-peer information system

As discussed in Section 1, the super-peer model can be advantageously exploited in Grid systems for the deployment of information and discovery services. In each Grid Organization (or PO, Physical Organization), a subset of powerful nodes having high availability properties can be chosen and elected as super-peers. Resource discovery efficiency is enhanced if compared with the decentralized P2P approach because a larger number of resources can be inspected in the same amount of time.

For the sake of simplicity, it is assumed that only one super-peer is selected in each organization, even if it can change with time. A super-peer accomplishes two main tasks: it is responsible for the communications with the other POs, and it maintains metadata about all the nodes of the local PO. The set of nodes belonging to a PO (i.e. the super-peer and the regular nodes) is also referred to as a *cluster* in the following.

As is the decentralized model described in Section 3.2, the super-peer topology is constructed according to the Albert–Barabási power-law algorithm [28]. The algorithm first selects a small number of disconnected POs, then and at each step it connects a new PO to a number of others that are already connected, by linking the respective super-peers.

The choice of constructing clusters on the basis of the existing Grid Organizations is a peculiar feature of the super-peer model described in this paper. This is not the norm for super-peer networks, in which the selection of super-peers and the assignment of simple nodes to them is generally done randomly or according to a different strategy. For example, in some systems nodes are assigned to super-peers so as to gather nodes of similar characteristics [22] in the same cluster. In other systems, selection and assignment are made to achieve some predetermined objective, for example a *target topology* that is composed by the minimum set of super-peers whose total capacity is sufficient to cover all other nodes as clients [20].

In a Grid, however, a super-peer topology that relies on previously formed organizations has at least the following advantages:

- Security and authorization tasks are facilitated, since intra-cluster communications are always within the same organization. For example, each super-peer maintains information only about local nodes;
- It is possible to exploit the presence of heterogeneous information systems that can be available in different Grid Organizations, as shown in Fig. 4: in this example, the MDS-2 service of GT2, the Index Service of GT4, and the Unicore information service. Indeed it is not necessary that the same Grid middleware is used in all the organizations: it is only required that super-peers are able to communicate with each other using a standard protocol and that each super-peer knows how to interact with the information service of the local organization;

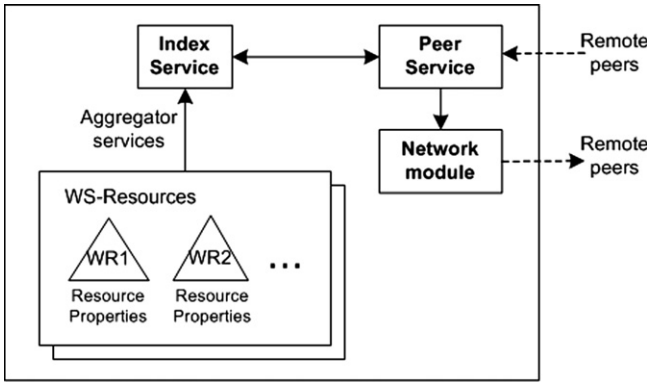

Fig. 3. Implementation of the decentralized P2P model using the GT4 framework.

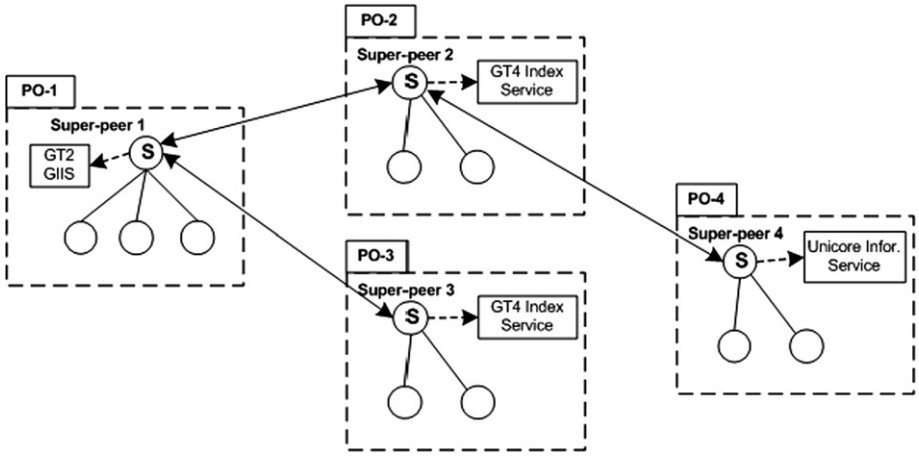

Fig. 4. A Grid network configuration based on the super-peer model.

```
// Nh = max number of neighbours
// q.list: list of hosts traversed by the query q
// q.sender: neighbour super-peer from which q has been received
// q.id: query identifier
// q.ttl: current value of ttl
For each incoming query q:
    if <q.id is in the cache> then queryInCache:=true;
    else <put q.id in the cache>;
    q.ttl -= 1;
    if ((q.ttl>0) and not queryInCache)
    {
        select at most Nh best neighbours;
        for each selected neighbour n:
            if <n is not in q.list> {
                add this super-peer to q.list;
                forward a copy of q to n;
            }
    }
    <ask the local information service for resources matching q>
    if <there are such resources> {
        send to q.sender a queryHit containing information about
        the discovered resources;
        send notifications to the hosts owning the resources;
    }
```

Fig. 5. The resource discovery algorithm executed by the Super-peer Service.

- The administrators of each Grid Organization can choose the nodes that can play the role of super-peer, according to locally specified criteria.

The resource discovery protocol, exploited by the discovery service, is defined as follows. Query messages generated by a Grid node are forwarded to the local super-peer. The super-peer examines the local information service to verify if one or more useful resources are present in some of the nodes belonging to the local PO. Furthermore, as in the decentralized model, the super-peer forwards a copy of the query to a number of neighbor super-peers, which in turn contact the respective information systems and so on. Whenever a resource, matching the criteria specified in the query, is found in a PO, a queryHit is generated and is forwarded along the same path back to the requesting node. To decrease the network load, the super-peer protocol exploits the same techniques (i.e. selection of neighbors, TTL and caching) that were discussed in Section 3.2 for the decentralized model.

The super-peer model is integrated with the information service of the WSRF-based GT4 framework, in a fashion similar to that shown for the fully decentralized model (Fig. 3). The GT4 Index Service subscribes to the Resource Properties of Grid Services hosted by the nodes of the local PO. The *Super-peer Service* processes requests coming from remote POs, queries the local Index Service to find resources matching the query constraints, and forwards query and queryHit messages through the Network Module. A sketch of the resource discovery algorithm, executed by a Super-peer Service when receiving a query from an external PO, is reported in Fig. 5.

4. Performance evaluation of the resource discovery protocols

This section reports and discusses the performance results obtained by using the three resource discovery models and protocols introduced in Section 3. For each of the three models, network and protocol parameters were set to values that are coherent with Grid frameworks currently deployed and with performance studies available in literature [29]. Moreover, parameter settings are consistent for the three models, so they allow for a fair comparison of the discovery protocols. Section 4.1 describes the simulation environment and introduces general assumptions that are valid for all the three protocols. The performance indices are also defined in this section. Afterwards, protocols are separately evaluated in Sections 4.2–4.4. Comparative analysis is given in Section 5.

4.1. Simulation settings and performance indices

The simulation framework includes two components: a network generator and a discrete-event continuous-time simulator. The network generator used for the decentralized and the super-peer model follows the Albert–Barabási model, as discussed in Sections 3.2 and 3.3. On the other hand, the topology used for the hierarchical model is constructed as described later in Section 4.2.

The discrete-event simulator, written in C++, is fully object-oriented. The different objects simulate the behavior of Grid/P2P components and are able to exchange messages. Every time an object receives a message/event, it performs a related procedure, according to its finite state automaton, and possibly sends new messages to other objects.

As mentioned in Section 2, in Grid systems users often need to discover resources that belong to a class of resources, rather than a specific single resource. Resource categorization is an efficient method to restrict the search space and improve the efficiency of resource discovery services. For example, in building a distributed data mining application [30], a user may need to find a software tool that performs a clustering task on a given type of source data. In this case, a query containing the appropriate constraints is generated to find such resources on the Grid; afterwards, one of the discovered resources will be chosen by the user (or by an automatic Grid scheduler) for execution. This is a common problem in Grids where resources and/or services must be used in complex applications.

To fairly compare the three models discussed in this paper, the same distribution of resources is assumed, in Grids having from 10 to 10,000 nodes. In particular, it is assumed, as in [10], that the average number of elementary resources offered by a single Grid host remains constant as the Grid size increases. This average value is set to 5, and a gamma stochastic function is used to determine the number of resources owned by each node. As the Grid size increases, it becomes increasingly unlikely that a new node connecting to the Grid provides resources belonging to a new resource class [10]. Therefore, 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 is comprised between 10 and 10,000) 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 logarithmic distribution of resources was motivated and analyzed with more details in [11], where comparison is given with other kinds of resource distribution patterns.

Other parameters of the simulation scenario are appropriately set to achieve a fair comparison of the three models. Both in the decentralized and the super-peer, the average degree of nodes is set to 4 that is also the maximum number of neighbors to which a query can be forwarded by a peer or a super-peer. Communication and processing delays are also set to consistent values. The processing time, needed to check a resource against a query, is set to 0.2 ms for all the three models (this value was estimated in [29] and is in accordance with our experiments on Grids). The average link delay is set to 10 ms for connections which can be assumed to be local in most cases (link between a regular node and its parent Index Service in the hierarchical model, link between a peer and the local super-peer in the super-peer model), and to 50 ms for the other types of connections (link between two peers in the decentralized model or between two super-peers in the super-peer model, link between two Index Services in the hierarchical model).

During a simulation run, each Grid host generates a set of queries. The mean interarrival time between two successive query requests issued by the same node is modeled as a stochastic variable having a gamma probability distribution function, with a shape parameter equal to 2 and an average value equal to 300 s. For each generated query, the simulator randomly selects the class of the resources that the user needs to discover. The set of performance indices is shown in Table 1. The index N_{res} is computed as the average number of discovered resources. This index is particularly important, since it is

Table 1
Performance indices and their definition

Performance index	Definition
Mean number of results N_{res}	Mean number of resources discovered by a query
Message load L	Frequency of messages received by a node (messages/s)
queryHits/messages ratio R	Number of queryHits divided by the overall number of messages received by a node
Response time T_r, $T_r(K)$, $T_r(L)$	Mean time interval (s) between the generation of a query and the reception of a generic result (mean response time), of the kth result, of the last result
Memory space S_m	Amount of memory necessary to maintain information about published resources

often argued that the *satisfaction of the query* depends on the number of results (i.e. the number of discovered resources) returned to the user that issued the query. For example, in [31] a resource discovery operation is considered *satisfactory* only if the number of results exceeds a given threshold. 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 a high elaboration load. The ratio R is an index of efficiency: if an increase of R is achieved, a higher relative fraction of queryHit messages, containing useful results, is generated or forwarded with respect to the overall number of messages.

Response times are used to estimate the *time to satisfaction* experienced by a user, and are obtained as the sum of two contributions: the communication delays between nodes and the processing time needed to compare a query with the resources stored at each node. In general, the second component is negligible when the nodes store a small number of resources (in the decentralized model), but it becomes significant when the number of stored resources is large (in the super-peer model and to a larger extent in the high layers of the hierarchical model). Accordingly, different settings of the average processing time or the communication delays affect the response times, perceived in the three models, in different ways. This issue will be discussed in Section 5. We measured different response times, that is, the average response time T_r, the response time of the last result received for a query $T_r(L)$, and the response time of the kth result $T_r(K)$, to account for the case in which a query needs to find at least k results. Specifically, we measured the response times of the first and the 10th results, $T_r(1)$ and $T_r(10)$.

Finally, the memory space S_m is the amount of memory used by a node to store information about the published Grid resources; the assumption here is that, for each resource, the related metadata document requires 10 KB of data on average.

4.2. Performance of the hierarchical model

This section evaluates the performance of the hierarchical model based on GT4 Index Services, described in Section 3.1, and specifically of the resource discovery algorithm reported in Fig. 2. Table 2 summarizes the simulation scenario for this model.

P_{res} is defined as the percentage of Grid resources that an Index Services collects from the descendant hosts and Index Services and publishes, and it is set to 25%, 50% and 100% in different simulation runs.

The value of the tree height H (see Fig. 1) was set to 3 and 4 in two different simulation sets. This choice comes from the observation that a value of H equal to 2 would correspond to a Grid having only one Index Service and a number of regular Grid hosts (level 0), thus reducing the Grid to a simple organization. Conversely, values of H larger than 4 would mean the presence of more than 3 hierarchical layers of Index Services, which is very unlikely in 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 Eq. (1), in which the first member calculates the overall number of nodes in a complete tree with height H and order K:

$$\frac{K^H - 1}{K - 1} > N. \tag{1}$$

Furthermore, it is assumed that the load imbalance among the Index Services belonging to the same level is limited. This assumption was 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 $H = 4$), the value of K is set to 13 since, with this value, the first member of Eq. (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. Finally, the time-to-live TTL of query messages is set to $H - 1$, thus allowing queries to explore the entire hierarchy.

Since every query can reach the root Index Service, the N_{res} index can be obtained with the formula (2) reported below, in which N_{tot} is the overall number of resources published on the Grid:

$$N_{res} = \frac{N_{tot} * P_{res}}{N_{cl}}. \tag{2}$$

Note that N_{res} increases with the Grid size, since N_{tot} is a linear function of N (see Section 4.1), and therefore increases more rapidly than the number of resource classes, which is a logarithmic function of N, as discussed in Section 4.1.

Table 2
Simulation scenario for the hierarchical model

Parameter	Value
Percentage of resources published by Index Services, P_{res}	25%, 50% and 100%
Tree height (number of levels), H	3, 4
Tree order (maximum number of children of an Index Service), K	2–100
Time-to-live, TTL	$H - 1$

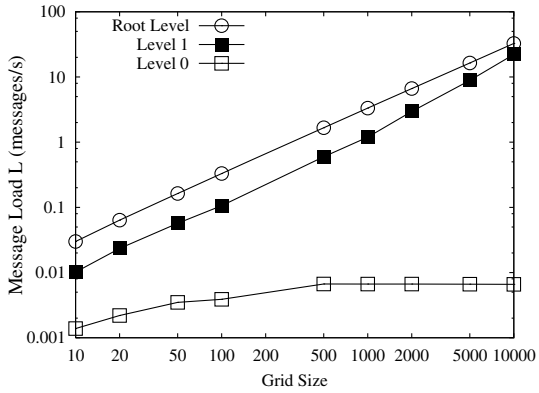

Fig. 6. Hierarchical model: message load vs. the Grid size, in a Grid with tree height $H = 3$, at hosts belonging to different levels.

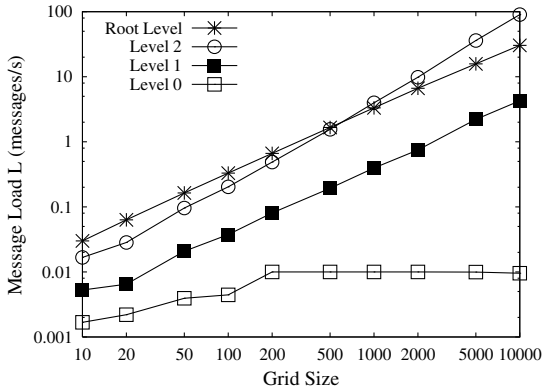

Fig. 7. Hierarchical model: message load vs. the Grid size, in a Grid with tree height $H = 4$, at hosts belonging to different levels.

Figs. 6 and 7 report the message load experienced by hosts belonging to the different levels of the Grid hierarchy. Figures are related, respectively, to Grids having 3 and 4 levels. While the message load experienced by regular hosts is relatively low and approaches a saturation level as the Grid size increases, 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 receives the largest number of messages per second 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 descendant nodes, and 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 outcome of the simulation analysis.

Fig. 8 reports the average response time for hierarchical Grids having a tree height equal to 4. Comparison is made among the performances obtained with different values of P_{res}, the percentage of resources published by Index Services. As the Grid size increases, the average response time becomes longer, up to almost 20 s for a Grid with 10,000 nodes. The value of P_{res} has a strong impact on the response time: in fact, owing to the large number of resources maintained in the Index Services, the processing time is the major component of the response time, whereas communication delays, particularly in large Grids, are relatively small, being in the order of milliseconds. Response time measured with Grids having H equal to 3 is similar and is not reported.

Analogous trends were observed for the response times of the 10th and the last result, $T_r(10)$ and $T_r(L)$. Conversely, the trend of the response time of the first result $T_r(1)$, depicted in Fig. 9, is different and is worth being analyzed. This trend is similar to the others only for small Grids; in fact, when the overall number of resources is small, the first result is frequently found in the root Index Service. However, in large Grids, a query issued by a regular host has more chances to find useful results in the local Index Service, which makes the response time of the first result decrease to much smaller values.

A final consideration concerns the memory space. The major load is obviously carried by the root Index Service, which must store, in its main memory, information about resources belonging to the entire Grid. With the discussed assumptions (mean size of a metadata document equal to 1 KB and mean number of resources published by each peer equal to 5), S_m is equal to $5 * N * P_{res} * 10$ KB for the root Index Service. For example, with $N = 10{,}000$ and $P_{res} = 0.50$, S_m is equal to 250 Mbytes.

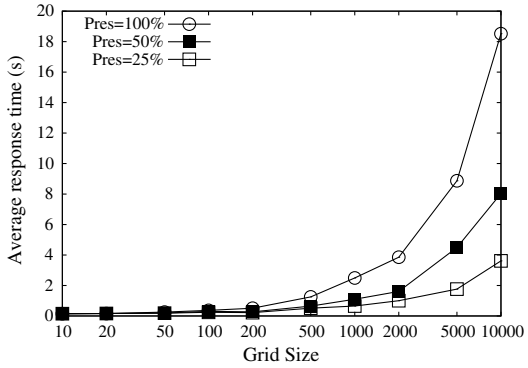

Fig. 8. Hierarchical model: average response time vs. the Grid size, in a Grid with tree height $H = 4$, for different values of the percentage of resources published by Index Services, P_{res}.

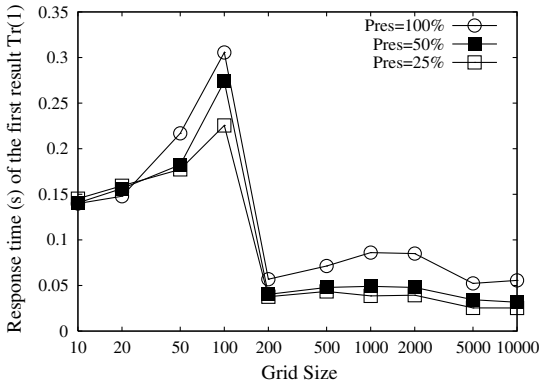

Fig. 9. Hierarchical model: response time of the first result vs. the Grid size, in a Grid with tree height $H = 4$, for different values of the percentage of resources published by Index Services, P_{res}.

4.3. Performance of the decentralized P2P model

This section evaluates the performance of the P2P decentralized model discussed in Section 3.2. Different values of TTL (the maximum number of hops that a query message can perform) are tested, in order to evaluate how this parameter can be tuned to improve performance for a given Grid size.

The mean number of results is reported in Fig. 10. For small networks, this number is low because the Grid is not able to offer a significant number of resources for all the different resource classes. An increase in the TTL value results in a corresponding increase in the mean number of results that is negligible for small networks, but becomes significant for larger

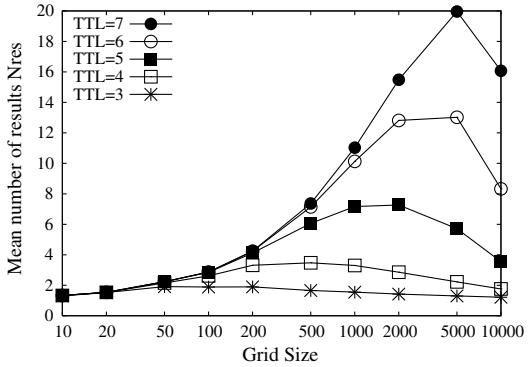

Fig. 10. Decentralized P2P model: mean number of results vs. the Grid size, for different values of TTL.

networks. It is also noticed that, with a fixed value of TTL, the number of results decreases when the network size exceeds a threshold value. Indeed, beyond this threshold, the TTL value limits the chances to discover the required resources, because they are more and more dispersed over the network.

In Fig. 11, it is observed that a high message load is a toll to pay if a large number of results is desired. Indeed, curves of message load show a trend comparable to the number of results. A trade-off should be reached between maximizing the number of results and minimizing the processing load; to this aim, the efficiency index R was evaluated. Fig. 12 allows to identify the TTL value that maximizes the efficiency of the network for a given value of the Grid size. For Grids having less than 400 nodes, the highest values of R are obtained with a TTL value equal to 3, but for larger networks the optimum TTL value gradually increases. For example, a TTL equal to 4 is preferable for Grid sizes ranging from about 400 nodes to about 2000 nodes. With 10,000 nodes, a TTL equal to 6 outperforms other TTL values, and curve trends seem to suggest that, for even larger networks, a TTL equal to 7 is the best choice.

Fig. 13 shows that the average response time increases with the TTL and with the Grid size because a larger fraction of queryHit messages come from more distant peers. However, it is also noticed that the response time gets to a saturation level in very large Grids because the tested TTL values allow to explore a limited region of the Grid. Fig. 14 compares the different response times measured with a TTL value of 4. The response times of the first, the average and the last result have similar trends and comparable values for both very small and very large Grids. The reason is that in small Grids results can only be discovered in neighbor peers, while in large Grids a high fraction of results are discovered in peers located on the boundary of the Grid region covered by the given TTL. Conversely, a higher variability of response times is experienced in medium-scale Grids, i.e. having a number of nodes comprised between 50 and 2000: in such Grids, results are discovered in nodes that are more fairly partitioned between close and remote ones.

The trend of the 10th response is peculiar with respect to the other ones. First, it can be noticed that such a curve has not been evaluated for Grids having less than 200 nodes, since it is very rare to discover at least 10 useful results in those Grids. For larger Grids, it is observed that $T_r(10)$ is higher with respect to the other indices, included the response time of the last result. The reason is that only a fraction of the queries are followed by at least 10 results, since the average number of results

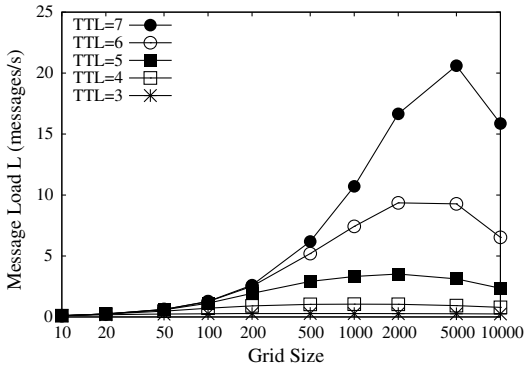

Fig. 11. Decentralized P2P model: message load vs. the Grid size, for different values of TTL.

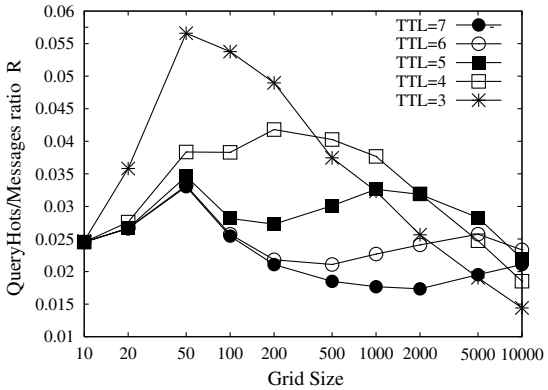

Fig. 12. Decentralized P2P model: queryHits/messages ratio vs. the Grid size, for different values of TTL.

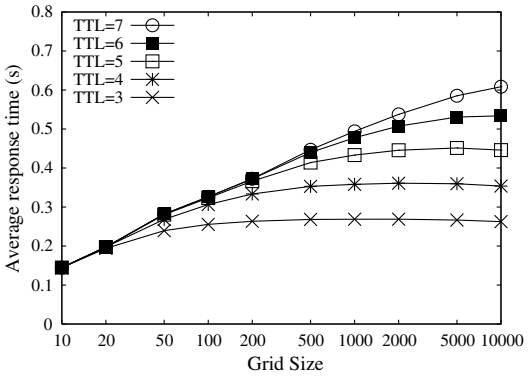

Fig. 13. Decentralized P2P model: average response time vs. the Grid size for different values of TTL.

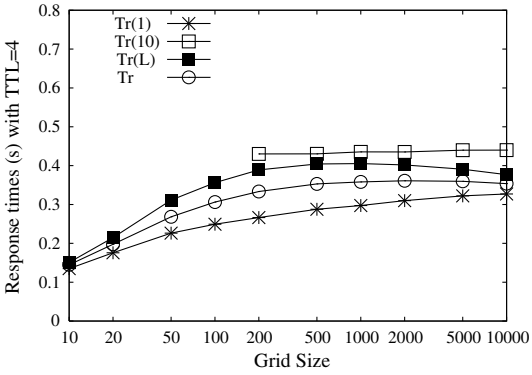

Fig. 14. Decentralized P2P model: response times vs. the Grid size, with TTL = 4. Comparison between T_r, $T_r(1)$, $T_r(10)$ and $T_r(L)$.

is lower than 10 (as appears in Fig. 10). Hence $T_r(10)$, which is evaluated and averaged only for such queries, is found to be higher than the response time of the last result averaged on all the queries.

4.4. Performance of the super-peer model

This section evaluates the performance of the super-peer model and the resource discovery algorithm shown in Fig. 5. The mean cluster size is set to 10: this assumption, as well as the impact of the cluster size on results, are discussed in Section 5.

It appears from Fig. 15 that an increase in the TTL value allows for discovering a notably larger number of resources only with networks having more than 1000 nodes. A similar effect is observed on the message load of super-peers (not shown): it increases with the TTL and the Grid size. As in the fully decentralized network, a trade-off may be obtained by analyzing the

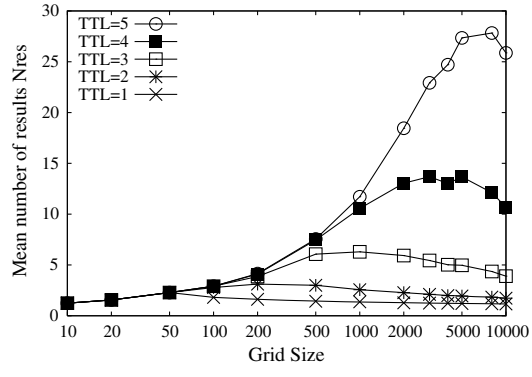

Fig. 15. Super-peer model: mean number of results vs. the Grid size, for different values of TTL.

efficiency index *R*. Fig. 16 shows that the optimum TTL value, i.e. the value that maximizes *R*, increases with the Grid size. For example, in a Grid with 1000 nodes the maximum value of *R* is obtained with a TTL equal to 3, whereas, in a Grid with 10,000 nodes, *R* is maximized with a TTL equal to 5. As a consequence, TTL should be set to a value equal or greater than 5 only if the number of nodes exceeds 5000; for smaller networks, tuning decisions should take into account that a high value of TTL can severely decrease the overall efficiency.

The trend of the average response time is observed in Fig. 17. It can be noticed that increasing the TTL value permits to explore a wider region of the Grid, causing a corresponding increase in the average response time. This effect is limited in small Grids, but it is increasingly more evident as the Grid size increases. Finally, Fig. 18 compares the values of different response times with a TTL fixed to 4.

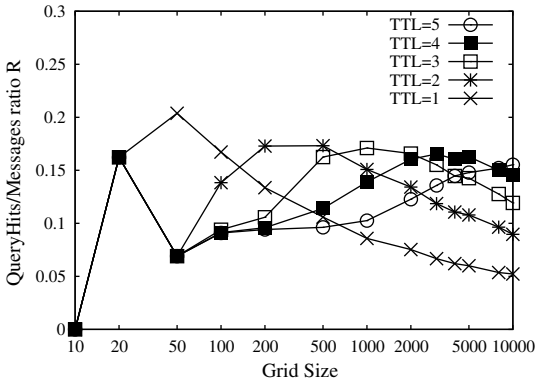

Fig. 16. Super-peer model: queryHits/messages ratio at super-peers vs. the Grid size, for different values of TTL.

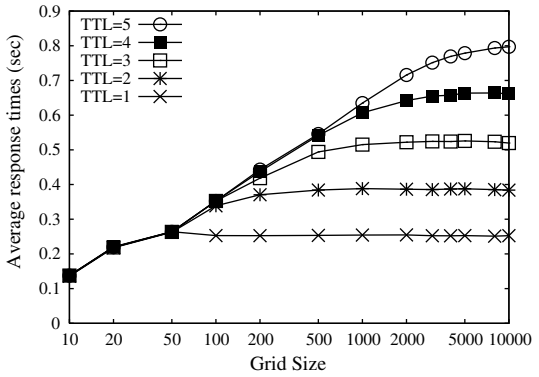

Fig. 17. Super-peer model: average response time vs. the Grid size, for different values of TTL.

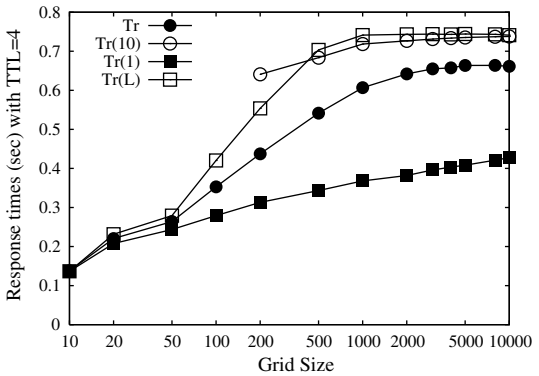

Fig. 18. Super-peer model: response times vs. the Grid size, with TTL = 4. Comparison between T_r, $T_r(1)$, $T_r(10)$ and $T_r(L)$.

5. Comparison and discussion of results

This section discusses and compares the results obtained with the three discussed models. The TTL value is used as a variable parameter for the decentralized and the super-peer model, whereas it is set to 3 for the hierarchical model, thus allowing queries to reach the root Index Service. For the super-peer model, it is assumed that each super-peer serves, on average, 9 regular nodes (thus forming clusters of 10 nodes on average). Resource distribution, topology, communication delays and processing times are set so as to permit a fair comparison among the models, as discussed in Section 4.1. Comparison is first made between the two peer-to-peer models. Since results suggest that the super-peer model is more scalable and efficient than the decentralized model, performance comparison is then made between the super-peer and the hierarchical model.

Figs. 19–21 compare, respectively, the average number of results, the average load and the average response time achieved with the decentralized model (denoted as D) and with the super-peer model (labelled as SP), in Grids having different sizes, and for different values of the TTL parameter. Since a super-peer manages metadata related to the resources maintained by all the peers of the local cluster, the average number of results per query is remarkably larger than that obtained with the decentralized model. Actually this improvement is obtained at the cost of having a higher load and a longer average response time with respect to the corresponding values measured with the fully decentralized model.

Overall, the super-peer model, especially for large-scale Grids, proves to be more efficient than the decentralized model, if it is considered that:

(1) The super-peer message load shown in Fig. 20 is the load carried by super-peers, not by simple hosts. This load (at most 25 messages per second with TTL = 5) can be considered acceptable, since a super-peer is usually chosen among the hosts of a Grid Organization specifically because it offers higher performance capabilities.
(2) The average response time experienced with the super-peer model is anyhow appropriate, being always lower than 1 second and, if compared to the decentralized model, the increase in response time is much lower than the corresponding increase in the number of results.

Figs. 22–24 compare the super-peer model with the today largely used hierarchical model (labelled as HR), in which the tree height H is set to 4. Fig. 22 shows the number of results obtained with the two models. With the hierarchical model, the

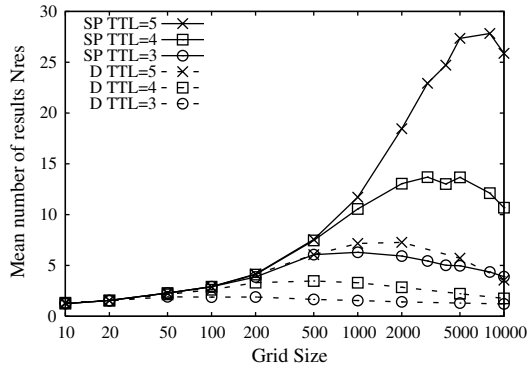

Fig. 19. Mean number of results obtained with the fully decentralized model (D) and the super-peer model (SP).

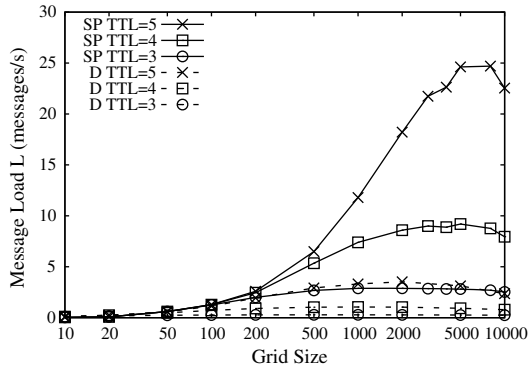

Fig. 20. Message load experienced by the peers of the fully decentralized model (D) and by super-peers (SP).

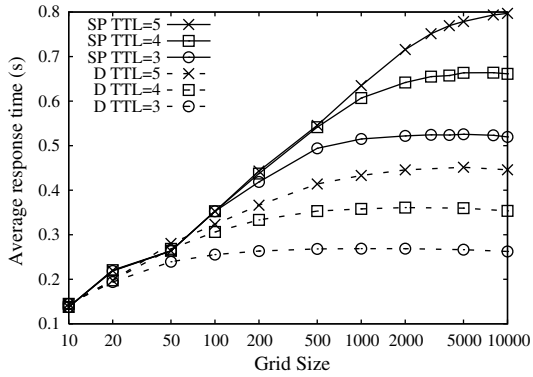

Fig. 21. Average response time with the fully decentralized model (D) and the super-peer model (SP).

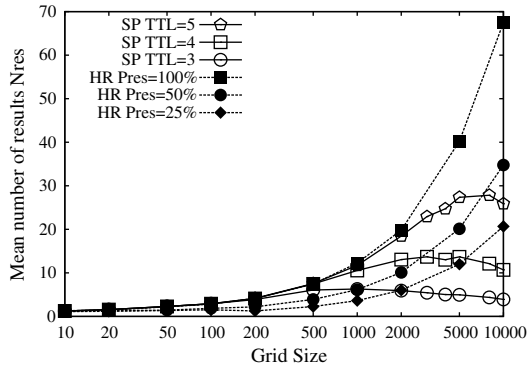

Fig. 22. Mean number of results obtained with the hierarchical model (HR) and the super-peer model (SP).

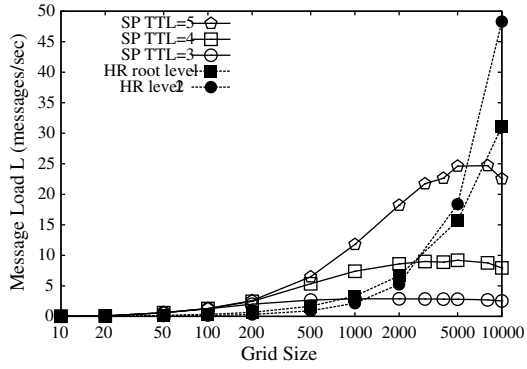

Fig. 23. Message load experienced by the Index Services belonging to the two highest levels of the hierarchical model (HR) and by super-peers (SP).

number of results depends on the Grid size and the value of P_{res}, the percentage of resources published by an Index Service. The number of results obtained with P_{res} = 100% can be considered as an upper bound, because in this case all the resources belonging to the class of interest can be found in the root Index Service. However, a P_{res} equal to 100% is not frequent in real Grids, while values of 50% or 25% are more realistic. One of the reasons is that Index Services located at different hierarchy layers may belong to different administrative domains, and an Index Service does not always have the possibility or the authorization to 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 data related to the resources of the local Grid Organization.

With the super-peer model, the number of results can be improved by increasing the TTL value, thus permitting to explore a larger region of the network. If a proper value of the TTL parameter (i.e. 5) is set, the average number of results

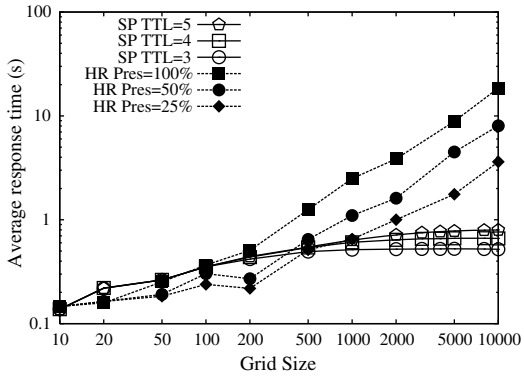

Fig. 24. Average response time with the hierarchical model (HR) and the super-peer model (SP).

exceeds that obtained with the hierarchical model in which P_{res} = 25%, for all the considered values of the Grid size. Of course, if P_{res} is even lower than 25%, the use of the super-peer model becomes even more effective.

In Fig. 23, 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, with P_{res} = 50%. This figure shows that the super-peer load is generally higher than the load at the root Index Service, for small- and medium-sized Grids. The reason is that the root Index Service receives only query messages, while a super-peer receives also queryHit messages from the neighbor 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 in Grids having 10,000 nodes or more. Fig. 23 also shows the very high load that is carried – in large Grids – by the Index Services located just below the root node of the hierarchy (at level 2), which confirms the better scalability features of the super-peer model.

Another 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 hosts of the local cluster. 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 host equal to 5), S_m is equal to about 500 Kbytes at a super-peer, while it is equal to $S_m = 5 * N * P_{res} * 10$ KB at the root Index Service of the hierarchical architecture. For example, with N = 10,000 and P_{res} = 0.5, S_m is equal to 250 Mbytes. Therefore, the hierarchical model requires a much larger amount of main memory. Even if such a memory can be available in the server machine that runs the Index Service, the analysis of metadata information may require a large amount of time. In fact, for every Index Service entry, the resource characteristics must be compared with the constraints specified in the query. In particular, resource matchmaking can be quite cumbersome when it is based on the evaluation of multiple parameters that have discrete or continuous values, as is the case for example of the Condor matchmaking [24]. With the assumption that the time to process a single resource is 0.2 ms, Fig. 24 reports the average response time experienced with the two models, with P_{res} = 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 Grids with more than 1000 hosts.

As mentioned in Section 4.1, the response time of queries has two main components: the processing time of resources and the communication delays. A modification in the average processing time can obviously alter the weight of these two

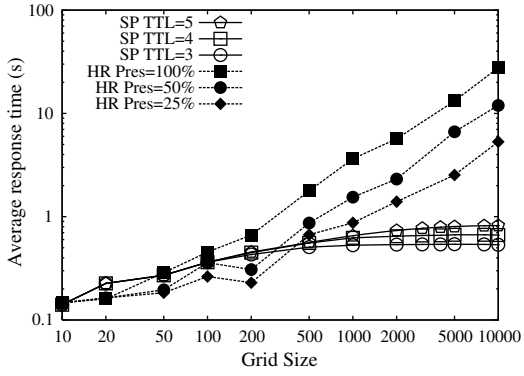

Fig. 25. Average response time with the hierarchical model (HR) and the super-peer model (SP); the average processing time is set to 0.3 ms.

components. To analyze this issue, further simulations were run, for the three models with different values of the average processing time. The impact of this variation increases with the average number of resources stored at each node. Therefore, it has no perceivable effect on the decentralized model (because the processing time is negligible with respect to the communication delays), but it has a moderate effect on the super-peer model and a much more evident impact on the hierarchical model.

Fig. 25 compares the response time obtained in the hierarchical and super-peer models, with the average processing time set to 0.3 ms instead of 0.2 ms. Note that the qualitative trend of figures is not altered. However, the response time in the hierarchical model increases more rapidly with the Grid size, and it exceeds that of the super-peer model for lower values of the Grid size.

In conclusion, the hierarchical model can be deemed suitable only for small- and medium-sized Grids. Conversely, 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 time with respect to the super-peer model. As an example, let us consider a 10,000-nodes Grid and compare the performance results obtained with the hierarchical model with $P_{res} = 50\%$, and with the super-peer model with TTL = 5. It can be observed that the ratio between the corresponding numbers of results is about 1.25, in favor of the hierarchical model (Fig. 22), but the Index Services located at level 2 must undergo a load which is more than twice the load sustained by a generic super-peer (Fig. 23), while the average response time (Fig. 24) is even 10 times longer than the response time experienced in the super-peer model (about 8 s against 0.7 s), assuming an average processing time of 0.2 ms per resource.

Another interesting issue is worth being analyzed, because it is related to current and future trends of Grid systems: the effect of the average size of clusters on the performance of the super-peer model. In fact, in file sharing super-peer networks, clusters are usual large, with as many as 100 nodes per cluster. On the other hand, in today Grids this value is unusual. Most organizations possess only a few nodes (even if often they are multi-processor nodes), so an average cluster size of 10 is much more common. However, the cluster size in Grids will probably increase in the future: to anticipate this trend, simulation experiments with larger clusters were performed, and results were again compared with those of the hierarchical model, in which the height is equal to 4.

As clusters become larger, the following considerations can be made:

- The super-peer model is able to retrieve more results, as expected. With an average cluster size equal to 100, and TTL larger than 3, the number of results approaches that obtained with the hierarchical model;
- However, the load at super-peers significantly increases and, with a TTL larger than 3, it exceeds that of Index Services. This happens because every query makes a larger number of hops than in the hierarchical model (in which it is equal to 3) and, for each further hop, the number of messages is multiplied by a factor that can be as high as 4, which is the maximum number of neighbors to which a query can be forwarded;
- The response time slightly increases in the super-peer model, but in the hierarchical model it is still much longer, because the root Index Service has to process a much larger number of resources than super-peers. In fact, in the hierarchical model all the resources, or a percentage P_{res} of them, are replicated and published by the root Index Service, whereas in the super-peer model the resources are distributed on the different super-peers.

6. Conclusions

Three models for building a Grid information system, namely the hierarchical model, the decentralized P2P model and the super-peer model, have been analyzed by means of a simulation framework. The three models have also been compared in order to determine which one is the most effective in different Grid configurations and sizes. In particular, performance comparison showed that the super-peer model is generally more convenient then the decentralized model because the higher load that must be carried by super-peers is repaid by a notable improvement in terms of response time and expected number of results. Moreover, comparison between the hierarchical and the super-peer model showed that the hierarchical model is valuable for small and medium-sized Grids, but the super-peer model is more effective in very large Grids and therefore is more scalable. In addition to the results reported in this paper, a number of further considerations (concerning fault-tolerance, load balancing, administrative features, etc.), generally favor the use of the super-peer paradigm, especially for very large multi-institutional Grids. The reported analysis can be profitably used by designers and developers of Grid information systems also because it gives hints about appropriate strategies tailored to the tuning of protocols in different network configurations.

Acknowledgement

This work has been partially supported by the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

References

[1] The Globus Alliance, Globus toolkit information services: monitoring and discovery system (MDS), 2006. <http://www.globus.org/toolkit/mds>.

[2] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid information services for distributed resource sharing, in: Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing, 2001.
[3] The Globus Alliance, The Web services resource framework (WSRF), 2006. <http://www.globus.org/wsrf>.
[4] J.M. Schopf, M. D'Arcy, N. Miller, L. Pearlman, I. Foster, C. Kesselman, Monitoring and discovery in a Web services framework: functionality and performance of the globus toolkit's MDS4, Tech. Rep. ANL/MCS-P1248-0405, Argonne National Laboratory, April 2005.
[5] V. Silva, Globus toolkit 4 early access, Tech. Rep., October 2004. <http://www-128.ibm.com/developerworks/grid/library/gr-gt4early>.
[6] A. Iamnitchi, I. Foster, On death, taxes, and the convergence of peer-to-peer and grid computing, in: Proceedings of the Second International Workshop on Peer-to-Peer Systems IPTPS'03, 2003.
[7] D. Talia, P. Trunfio, Toward a synergy between P2P and grids, IEEE Internet Computing 7 (4) (2003) 96-95; doi:10.1109/MIC.2003.1215667.
[8] I.J. Taylor, From P2P to Web Services and Grids: Peers in a Client/Server World, Springer, 2004.
[9] M. Marzolla, M. Mordacchini, S. Orlando, Peer-to-peer systems for discovering resources in a dynamic grid, Parallel Computing 33 (4–5) (2007) 339–358.
[10] A. Iamnitchi, I. Foster, D.C. Nurmi, A peer-to-peer approach to resource location in grid environments, in: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing HPDC'02, 2002, p. 419.
[11] C. Mastroianni, D. Talia, O. Verta, A P2P approach for membership management and resource discovery in grids, in: Proceedings of the International Conference on Information Technology: Coding and Computing ITCC'05, 2005, pp. 168–174.
[12] B. Yang, H. Garcia-Molina, Designing a super-peer network, in: 19th International Conference on Data Engineering ICDE, 2003.
[13] C. Mastroianni, D. Talia, O. Verta, A super-peer model for resource discovery services in large-scale grids, Future Generation Computer Systems 21 (8) (2005) 1235–1248.
[14] R. van Renesse, The importance of aggregation, in: Future Directions in Distributed Computing, Lecture Notes in Computer Science, vol. 2584, Springer, 2003, pp. 87–92.
[15] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, S. Haridi, Peer-to-peer resource discovery in grids: models and systems, Future Generation Computer Systems 23 (7) (2007) 864–878.
[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, A scalable content-addressable network, in: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications SIGCOMM'01, 2001, pp. 161–172.
[17] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for internet applications, in: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications SIGCOMM'01, 2001, pp. 149–160.
[18] A.I.T. Rowstron, P. Druschel, Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms, 2001, pp. 329–350.
[19] G. Sakaryan, M. Wulff, H. Unger, Design and implementation tradeoffs for wide-area resource discovery, in: Proceedings of the Innovative Internet Computing Conference, IICS'04, Springer LNCS, vol. 3473, Guadalajara, Mexico, 2004.
[20] A. Montresor, A robust protocol for building superpeer overlay topologies, in: Proceedings of the Fourth International Conference on Peer-to-Peer Computing P2P'04, 2004, pp. 202–209.
[21] L. Garces-Erice, E. Biersack, P. Felber, K. Ross, G. Urvoy-Keller, Hierarchical peer-to-peer systems, in: Proceedings of the ACM/IFIP International Conference on Parallel and Distributed Computing Euro-Par 2003, 2003.
[22] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, A. Loser, Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks, in: Proceedings of the 12th International Conference on World Wide Web WWW'03, 2003, pp. 536–543.
[23] D. Puppin, S. Moncelli, R. Baraglia, N. Tonellotto, F. Silvestri, A peer-to-peer information service for the grid, in: Proceedings of the International Conference on Broadband Networks, 2004.
[24] D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: the condor experience, Concurrency – Practice and Experience 17 (2–4) (2005) 323–356.
[25] A. Crespo, H. Garcia-Molina, Routing indices for peer-to-peer systems, in: Proceedings of the 28th Conference on Distributed Computing Systems ICDCS, 2002.
[26] A. Andrzejak, Z. Xu, Scalable, efficient range queries for grid information services, in: Proceedings of the Second International Conference on Peer-to-Peer Computing P2P'02, 2002, p. 33.
[27] A.Z. Kronfol, FASD: a fault-tolerant, adaptive, scalable, distributed search engine, Ph.D. Thesis, 2002.
[28] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[29] P. Hasselmeyer, The nextgrid project: architecture for next generation grid, work package 5, grid dynamics, document p.5.2.1, Tech. Rep., 2005.
[30] C. Mastroianni, D. Talia, P. Trunfio, Managing heterogeneous resources in data mining applications on grids using XML-based metadata, in: Proceedings of the International Parallel and Distributed Processing Symposium IPDPS 2003, 2003.
[31] B. Yang, H. Garcia-Molina, Improving search in peer-to-peer networks, in: Proceedings of the 22nd International Conference on Distributed Computing Systems ICDCS'02, 2002.