A Super-Peer Model for Multiple Job Submission on a Grid

Pasquale Cozza[1], Carlo Mastroianni[2], Domenico Talia[1], and Ian Taylor[3]

[1] DEIS University of Calabria, 87036 Rende (CS), Italy
{pcozza,talia}@deis.unical.it
[2] ICAR-CNR, 87036 Rende (CS), Italy
mastroianni@icar.cnr.it
[3] Computer School, Cardiff University, UK
Ian.J.Taylor@cs.cardiff.ac.uk

Abstract. Submission of multiple jobs in a distributed and heterogeneous environment is required by applications that rely on the "public-resource computing" paradigm. We present here a scientific scenario for the analysis of astronomical data, where some nodes are responsible for maintaining and advertising job description files and other so called worker nodes, are dispersed over the Grid to execute the jobs. Job assignment is performed through a mechanism that matches adverts, containing job descriptions, with job queries that are sent by available workers across the Grid exploiting an underlying super-peer topology. With an analogous mechanism, a worker locates the input data file needed to run a job and downloads it from a data center node. This paper presents a super-peer protocol for the submission of a very large number of jobs on a Grid environment. The super-peer architecture enables the replication of data files on multiple data centers, which helps reduce the processing load and speed up the application. A simulation analysis has been performed to evaluate the impact of application and network parameters on performance results.

1 Introduction

Recently, academic and industrial researchers have been promoting the convergence of two paradigms for distributed computing, namely Grid and peer-to-peer (P2P), which in the beginning tended to evolve separately [8]. Super-peer systems have been proposed [7, 9] to achieve a balance between the inherent efficiency of centralized networks, and the autonomy, load balancing and fault-tolerant features offered by P2P networks. A super-peer node can act as a centralized resource for a limited number of regular nodes (peers) of a Grid organization. At the same time, super peers connect among them to form a P2P network at a higher level, thus enabling distributed computing on a very large scale.

This paper reports on a distributed model based on the super-peer paradigm for the support of applications that require the distributed execution of a large number of jobs in a similar fashion to public-resource computing. The term "public resource computing" [1] is used for applications in which jobs are executed by private-owned

W. Lehner et al. (Eds.): Euro-Par 2006 Workshops, LNCS 4375, pp. 116–125, 2007.

computers that use their spare CPU time to support a large scientific computing project. The pioneer project SETI@home [3] has attracted millions of participants wishing to contribute to the digital processing of radio telescope data to search for extra-terrestrial intelligence. A number of similar projects are today supported by the BOINC software system (Berkeley Open Infrastructure for Network Computing [2]), for example: the Einstein@home project [5] aims at detecting certain types of gravitational waves, such as those produced by spinning stars; whereas the Climate@home [4] focuses on long-term climate prediction. The BOINC infrastructure is composed of a scheduling server and a number of clients installed on users' machines. The client software periodically contacts the scheduling server reporting host's hardware and availability, and receives a set of instructions for downloading executable and input files. After that the client runs the assigned job and uploads the resulting output files to the scheduling server.

The BOINC middleware is suited for CPU-intensive applications but it is inappropriate for data-intensive tasks because of its centralized nature. BOINC allows a project to configure a fixed static set of data servers that have to be administered by an entity. Although this scheme enables a number of servers to help load balance the network, the topology is static and is incapable of scaling proportionately as the network grows and more bandwidth is needed for data transfers. In BOINC, an administrator must configure these static machines, which are generally dedicated for a specific project. Such machines are not only costly (to purchase and maintain) but also they are centrally administered and therefore cannot generally be used by other BOINC projects. In the scheme proposed here, the Job initiator is lightweight and sends the data once to the network, which propagates it across the data nodes as and when required. This helps to distribute the data load dynamically in a decentralized fashion, both in topology and administratively, making it far more suitable to the Grid domain. For example, inherent in BOINC-like networks is the need to send a data file needed by several workers several times due to the unreliability of the nodes. This replication represents an evident waste of server-based bandwidth that could be avoided through a caching mechanism that replicates the data across the network when it is first transferred, thereby not only relieving the central bottleneck of the system but also it can place the data in a location closer to where the work is being performed. Further, a number of projects require many nodes to process the same data, with different parameters for example, which can be exploited by such an overlay described here. Our gravitational-wave example described here employs such an algorithm.

The super peer job submission protocol described in this paper enables caching of the input data files in multiple *data centers*, i.e. in super-peers, which have sufficient data storage facilities. Benefits of this replication strategy range from a larger degree of reliability and fault-tolerance to a more efficient use of bandwidth and CPU resources. The job submission protocol requires that job execution is preceded by two *matching* phases, the first one for job assignment and the second one for downloading of input data. Since input data files can be very large, focus is especially on the download phase which is the most bandwidth consuming. A set of simulation runs have been performed to evaluate the impact of the caching and replication mechanism on a set of performance indices, such as overall time to execute all the jobs,

throughput, mean time to download a data file, and load experienced by data centers and worker nodes. In particular, we simulated the behavior of the super-peer protocol in a Grid containing 25 super-peers and 250 ordinary peers. The experimental results show that the use of several data center can bring benefits to the Grid applications in terms of lower total execution times, higher throughout and load balancing among worker nodes. The study can also be used to determine the number of data centers that, for a given number of jobs, maximizes the utilization of data center nodes.

In section 2 the super-peer model and the related protocol are presented in more detail, whereas performance is analyzed in section 3. Conclusions and future work are discussed in section 4.

2 Job Assignment and Data Download

A data-intensive Grid application can require the distributed execution of a large number of jobs with the goal to analyze a set of data files. One sample application scenario defined for the GridOneD project [6] shows how one might conduct a massively distributed search for gravitational waveforms produced by binary stars orbiting one around the other. In this scenario, a data file of about 7.2 MB of data is produced every 15 minutes and it must be compared with a large number of templates (between 5,000 and 10,000) by performing fast correlation. Data can be analyzed in parallel by a number of Grid nodes to speed up computation and keep the pace with data production.

The scenario evaluated in this paper assumed the existence of a Grid network in which nodes are organized in a super-peer topology. The *job manager* node (i) receives data from a detector, (ii) produces the job description files (or *job adverts*), and (iii) collects output results. Simple peers, or *workers*, are available for job execution: they issue a job query to get a job description and then a data query to collect the corresponding input data file to be analyzed. Super-peer interconnections are exploited to make job and data queries travel the network rapidly; super peers play the role of *rendezvous nodes*, since they can store job and data adverts (and potentially the data files themselves), and compare these files with queries issued to discover them; thereby acting as a meeting place for both job or data providers and consumers. Since input data files can require a large amount of storage memory, it is assumed that only some of the peers in the network will cache such files. Such peers are referred to as *data centers* (*DC*) nodes and can be located on super peers or worker peers. We envisaged that the same user-driven process is used to configure a peer; that is, each user decides if they want to be a super peer and/or data center, as well as a worker. In the BOINC scenario, the existing dedicated machines would form the obvious data-center backbone and other peers (with high storage and network capacity) would also make themselves available in this mode.

Figure 1 shows a sample topology with 5 super-peers (2 of which are also data centers), and the sequence of messages exchanged among workers, super-peers and data centers to perform the job submission protocol. These messages are related to the execution of a job by a single worker, labeled as `W0`. Note that here, we do not use normal peers as data centers but we will be comparing this approach in later studies.

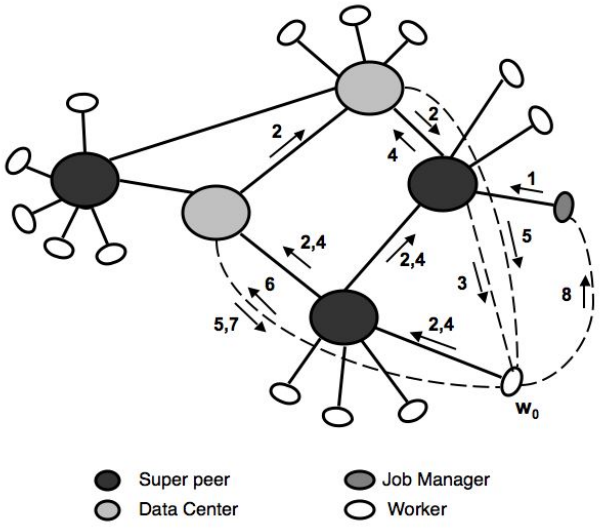

Fig. 1. Super-peer job submission protocol: sample network topology and sequence of exchanged messages to execute one job

The proposed protocol requires that job execution is preceded by two matching phases that exploit the features of the super-peer network: the *job-assignment* phase and the *data-download* phase.

In the *job-assignment* phase the *job manager* generates a number of *job adverts*, which are XML documents describing the properties of the jobs to be executed (job parameters, characteristics of the platforms on which they must be executed, information about required input data files, etc.), and sends them to the local rendezvous super-peer, which stores the adverts (step 1 in figure 1). Each worker, when ready to offer a fraction of its CPU time (e.g., worker `W0` in the figure), sends a *job query* that travels the Grid through the super-peer interconnections (step 2). In particular, a query message is sent to the directly connected super-peer, which in turn forwards it to its neighbor super-peers and so on, until the message `TTL` parameter is decremented to 0 or the job query finds a matching job advert. A job query is expressed by an XML document and can contain the main hardware and software features of the requesting node and CPU time and memory amount that the node offers. A job query matches a job advert when the job query parameters are compatible with the information contained in the job advert, for example concerning the characteristics required for the host that will execute the job. Whenever the job query gets to a rendezvous super-peer that maintains a matching job advert, such a rendezvous assigns the related job to the requesting worker by directly sending it a *job assignment* message (step 3).

In the *data-download* phase, the worker that has been assigned a job inspects the job advert, which contains information about the job and the required input data file (e.g. size and type of data). Then the worker sends a *data query* message to discover the input file (step 4). In a similar fashion to the job assignment phase, the data query travels the super-peer network searching for a matching input data file stored by a data center. Since the same file can be maintained by different data centers, the data center that receives a data query, in order to avoid multiple transmissions of the same file, does not send data directly to the worker. Conversely, the data center sends only a small *data advert* to the super peer connected to the worker and then to the worker itself (step 5). The worker initiates the download operation after receiving the first data advert (steps 6 and 7), and discards the subsequent adverts. After receiving the input data, the worker executes the job, reports the results to the job manager (step 8) and immediately issues a query for another job.

Replication of input data files on multiple data centers allows for a significant saving of time in the querying phase and enables the concurrent retrieving of files from different data centers. In the simulated scenario, it is assumed that all the data centers possess the data files before starting the job submission process. In a more dynamic scenario, the data file is initially maintained by only one data center, and the other data centers could cache the file during the download phase. For example, if in the network depicted in figure 1 the super-peer connected to the worker W_0 becomes capable of playing the data center role, it can store the data file downloaded by that worker and provide it for successive requests issued by other workers.

In the job assignment phase the protocol works in a way similar to the BOINC software, except that job queries are not sent directly to the job manager, as in BOINC, but travel the super-peer network hop by hop. Conversely, the data download phase differs from BOINC in that it exploits the presence of multiple data centers in order to replicate input data files across the Grid network.

3 Performance Evaluation

A simulation analysis has been performed by means of an ad hoc event-based simulator, written in C++, to evaluate the performance of the proposed super-peer protocol. The parameters of the astronomical application mentioned in Section 2 were used for the test case (e.g., file size, single job execution time, etc.). It is assumed that all the jobs have similar characteristics and can be executed by any worker.

Simulation parameters, and corresponding values, are reported in Table 1. The Grid network is composed of 25 Grid organizations, each containing one super-peer node and 10 regular nodes on average. The super-peer overlay network is organized so that each super-peer is connected to at most 4 neighbor super-peers. It is assumed that local connections (i.e. between a super-peer and a local simple peer) have a larger bandwidth and a shorter latency than remote connections. To compute download times with a proper accuracy, a data file is split in 100 KB segments, and for each segment the download time is calculated assuming that the downstream bandwidth available at a data center is equally shared among all the download connections that are simultaneous active from the data center to different workers.

In this preliminary study, it is assumed that the data centers download input data files before the workers join the system and issue their job queries. In future work,

analysis will focus on a more complex scenario in which data files are replicated, as a whole or in parts, during the execution of jobs.

The last two rows of Table 1 are related to parameters that were given varying values in the simulation runs, specifically the number of jobs `Njob` and the number of data centers `Ndc`, i.e. the number of super-peer nodes able to cache data files.

Table 1. Simulation parameters

Parameters	Values
Grid size: overall number of nodes = super-peers + workers	275= 25 + 250
Maximum number of neighbors for a super-peer	4
Size of input data files	7.2 MB
Latency between two adjacent super-peers (or between two remote peers in a direct connection)	100 ms
Latency between a super-peer and a local simple peer (worker)	10 ms
Bandwidth between two adjacent super-peers (or between two remote peers in a direct connection)	1 Mbps
Bandwidth between a super-peer and a local simple peer	10 Mbps
TTL parameter for job and data queries	4
Mean Job execution time	500 s (±10%)
Number of jobs, N_{job}	from 250 to 10000
Number of data centers, N_{dc}	1, 2, 3, 5, 9, 13

Performance indices are listed in Table 2. The index `Texec`, the overall time to execute all the jobs, is crucial to determine the rate at which data files can be retrieved from an astronomic telescope while guaranteeing that the workers are able to keep the pace with data. By the throughput index `Thr` it is possible to evaluate the efficiency of the job submission system. The remaining performance indices help determine the load that is experienced by data centers and by workers in different scenarios.

Table 2. Performance indices

Performance index		Definition
Overall execution time	T_{exec}	Time to execute all the jobs (s)
Throughput	T_{hr}	Average number of jobs completed per time unit (jobs/s)
Percentage of activity time	P_{act}	Average percentage of time in which a data center is active, i.e. has at least one download connection in progress
Mean download time	T_{dl}	Average time that it takes for a worker to download a data file from a data center (s)
Max number of executed jobs	J_{max}	Maximum number of jobs executed by a single worker

Figure 2 shows that the overall execution time decreases as more data centers are made available in the network, for two main reasons: (i) data centers are less heavily loaded and therefore data download time decreases, (ii) workers can exploit a higher

parallelism both in the downloading phase and during the execution of jobs. However, depending on the number of jobs to be executed, it is possible to determine a suitable number of data centers, beyond which the insertion of a further data center produces a performance increase which does not justify the related cost. For example, if 10,000 jobs are to be executed, a significant reduction of `Texec` is perceived as the number of data centers is increased up to a value of 9, whereas if the number of jobs is not greater than 1,000 two or three data centers are sufficient to achieve a good performance level. Analogous comments can be made about the throughput index, reported in Figure 3. A further consideration is that the throughput increases with the number of jobs because download and execution periods are alternated more efficiently if workers execute a larger number of jobs. But this increase tends to be negligible as the number of jobs is so large that the job submission system begins to approach a stable working condition.

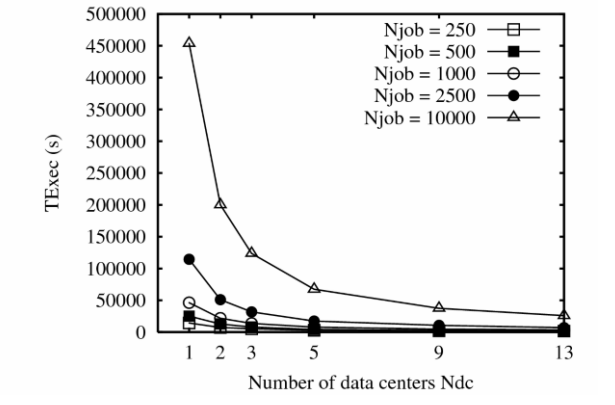

Fig. 2. Performance of the job submission super-peer protocol: overall execution time w.r.t. the number of data centers, for different numbers of jobs

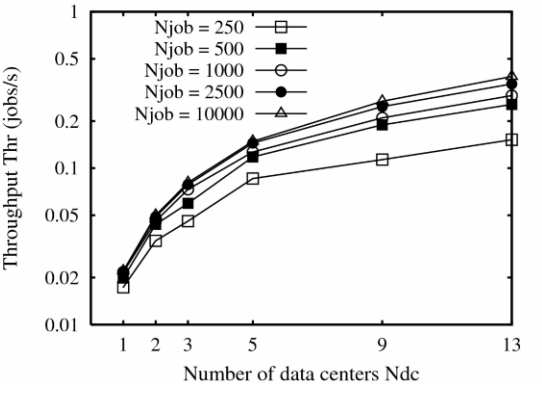

Fig. 3. Performance of the job submission super-peer protocol: throughput w.r.t. the number of data centers, for different numbers of jobs

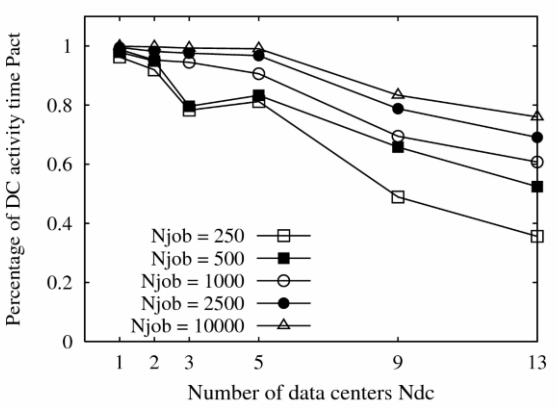

Fig. 4. Percentage of activity time of data centers, for different numbers of jobs

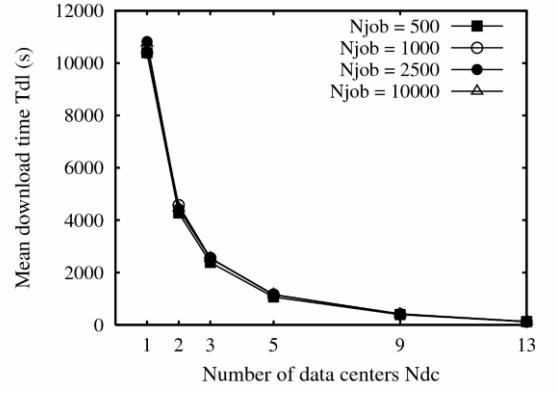

Fig. 5. Mean time to download an input data file w.r.t. the number of data centers, for different numbers of jobs

Figure 4 reports the average percentage of time in which a generic data center supports at least one download connection. Results confirm that the presence of an excessive number of data centers can be inappropriate, especially if the number of jobs is not very large. Indeed when the percentage of activity time decreases below 60%, machine utilization is very low resulting in a poor return of investment (ROI).

Figures 5 and 6 show performance results related to workers. Figure 5 proves that the download time decreases as the number of data centers is increased, resulting in smaller overall execution time. On the other hand, the download time hardly depends on the number of jobs because the simultaneous number of connections that a data center must serve is only related to the number of workers (250), not to the number of jobs. Finally, figure 6 compares number of jobs executed by a worker on average (obtained as `Njob`/250) to the maximum number of jobs executed by a single worker.

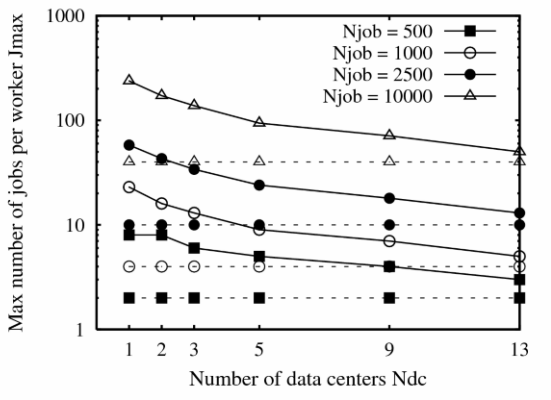

Fig. 6. Maximum number of jobs executed by a single worker w.r.t. the number of data centers, for different numbers of jobs. This index is compared to the average number of jobs executed by a single worker (dotted lines).

It is interesting to note that the two indices approach one another as the number of data centers is increased, leading to a fairer load balancing among workers.

4 Conclusions

This paper reports the first results of a work in progress research on a decentralized architecture for data intensive scientific computing on Grids according to the "public-resource computing" paradigm. We presented a super-peer protocol for the submission of a very large number of jobs in a Grid environment. In the discussed scenario some Grid nodes maintain and advertise job description files, whereas a number of worker nodes, dispersed over the Grid, execute single jobs. Job assignment is performed by matching job descriptions with the job queries that when issued by available workers, travel the Grid by exploiting an underlying super-peer topology.

Simulation analysis has been performed to evaluate the impact of application (the number of jobs) and network parameters (the number of data centers) on performance indices such as the overall time to execute all the jobs, throughput, efficiency of data centers, load experienced by workers. Results show that the use of several data centers can bring benefits to Grid applications in terms of lower total execution times, higher throughput and load balancing among worker nodes. However, since a large number of data centers also causes a smaller utilization of a single data center, the study can also be used to determine the number of data centers that can maximize the return of investment related to the deployment of new data centers.

Future work will move along a number of interesting research avenues, such as: the analysis of (1) redundant computing for applications that require multiple executions of each job; (2) caching of data file fragments on the P2P network, instead of storing entire files, to improve data download performance; (3) performance of the super-peer protocol in the case that input data is progressively fed as a data stream by an external source.

Acknowledgements

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265). We would also like to thank Eddie Al-Shakarchi, Tom Goodale, Andrew Harrison, Ian Kelley Matthew Shields and Ian Wang for their help defining the distributed architecture presented here.

References

1. Anderson, D.: Public computing: Reconnecting people to science, Proc. of Conference on Shared Knowledge and the Web, Madrid, Spain, November 2003, pp. 17-19
2. Anderson, D. P.: BOINC: A System for Public-Resource Computing and Storage, 5th IEEE/ACM International Workshop on Grid Computing, November 2004, Pittsburgh, PA, pp. 365-372
3. D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D.Werthimer: SETI@home: An experiment in public resource computing, Communications of the ACM, November 2002, Vol. 45 No. 11, pp. 56-61.
4. http://climateprediction.net/
5. http://einstein.phys.uwm.edu/
6. http://www.gridoned.org/
7. Mastroianni, C., Talia, D., Verta, O.: A Super-Peer Model for Resource Discovery Services in Large-Scale Grids, Future Generation Computer Systems, Elsevier Science, Vol. 21, No. 8 (2005) 1235-1456.
8. Talia, D., Trunfio, P.: Towards a Synergy between P2P and Grids, IEEE Internet Computing 7(4) (2003) 94-96
9. Yang, B., Garcia-Molina, H.: Designing a Super-Peer Network, 19th Int'l Conf. on Data Engineering, IEEE Computer Society Press, Los Alamitos, CA, USA (2003)