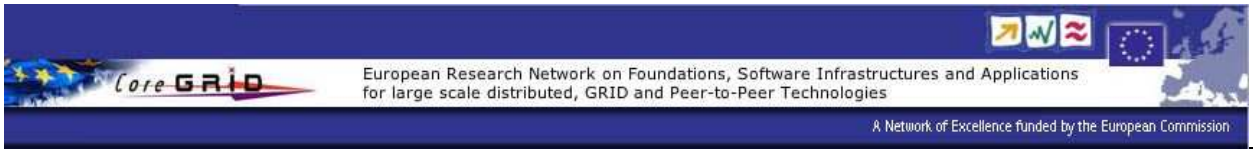

A data sharing protocol for Desktop Grid projects

Daniela Barbalace[1], *Ian Kelley*[2], *Carlo Mastroianni*[3], *Domenico Talia*[1], *Ian Taylor*[2]

[1] *DEIS, University of Calabria, Rende (CS), Italy*
barbalace@si.deis.unical.it,talia@deis.unical.it

[2] *School of Computer Science, Cardiff University, Cardiff, United Kingdom*
{I.R.Kelley|Ian.J.Taylor}@cs.cardiff.ac.uk

[3] *ICAR-CNR, Rende (CS), Italy*
mastroianni@icar.cnr.it

CoreGRID Technical Report
Number TR-0165
August 25, 2008

Institute on Knowledge and Data Management
Institute on Grid Systems, Tools and Environments

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

A data sharing protocol for Desktop Grid projects

Daniela Barbalace[1], Ian Kelley[2], Carlo Mastroianni[3], Domenico Talia[1], Ian Taylor[2]

[1] DEIS, University of Calabria, Rende (CS), Italy
barbalace@si.deis.unical.it,talia@deis.unical.it

[2] School of Computer Science, Cardiff University, Cardiff, United Kingdom
{I.R.Kelley|Ian.J.Taylor}@cs.cardiff.ac.uk

[3] ICAR-CNR, Rende (CS), Italy
mastroianni@icar.cnr.it

CoreGRID TR-0165

August 25, 2008

Abstract

Service grids and desktop grids are great solutions for solving the available compute power problem and helping to balance loads across network systems. However, with the objective to support new scientific communities that need extremely large numbers of resources, the solution could be to interconnect these two kinds of Grid systems into an integrated Service GridDesktop Grid (SG-DG) infrastructure. In this paper, we describe an architecture that exploits technological bridges to facilitate service and desktop grid interoperability, define the adopted data sharing protocol and present a performance evalution.

1 Introduction

The aim of Grid systems was that anyone (donors) could offer resources for a given Grid, and anyone (users) could claim resources dynamically, according to their actual needs, in order to solve a computational or data intensive task.

However, today we can observe two different trends in the development of Grid systems: Service Grids and Desktop Grids. The users of a Service Grid can publish their resource and put them at other users disposal. A resource can become part of the Grid by installing a predefined software set, or middleware. However, the middleware is usually so complex that it often requires extensive expert effort to maintain. It is therefore natural, that individuals do not often offer their resources in this manner, and Service Grids are generally restricted to larger institutions, where professional system administrators take care of the hardware/middleware/software environment and ensure high-availability of the Grid [10] .

Desktop Grids (DGs) on the other hand are commonly known as volunteer computing systems or Public-Resource Computing [1] ,a popular means of providing vast amounts of processing power to scientific applications through the use of personal home computers. To date, major volunteer computing systems, such as the Berkeley Open Infrastructure for Network Computing (BOINC) [6] , have focused solely one exploiting idle CPU cycles of donated computers

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

and have yet to take full advantage of other available resources such as powerful video card processors, hard disk storage capacities, and high-speed network connections. In Desktop Grid systems, anyone can bring resources into the Grid, installation and maintenance of the software is intuitive, requiring no special expertise, thus enabling a large number of donors to contribute into the pool of shared resources. Some of the most well-known Desktop Grid examples are Rosetta@Home [13] , Seti@Home [3] , and Einstein@Home [12] .

Until now, these two kinds of Grid systems have been completely separated and hence there has not been a mechanism to be able exploit their individual advantageous features in a unified environment. Enabling Desktop Grids for e-Science (EDGeS)[11] is an EU FP7 project that is setting up infrastructure and building software to enable integration of Service Grids, or traditional Grid environments generally composed of clusters and supercomputers, and Desktop Grid systems, such as the popular volunteer computing project BOINC. When moving jobs between these two environments, and specifically when transferring a job from a Service Grid to a Desktop Grid, there is a need to manage the phase of data transfer, due to security reasons. Desktop Grids in fact keep the user in a state of anonymity, while a Service Grid requires the authentication of the user and high level of security. Anonymous access is generally not an issue for most BOINC projects, as they are able to have dedicated and network isolated data servers, but it could quickly become problematic, both technically and politically, if one tried to somehow bootstrap a BOINC data server onto a cluster or supercomputer to enable access to users' files. The data distribution mechanism is the principal topic in this work and we try to define some ways to implement a decentralized data distribution system among workers.

The paper is organized as follows: Section 2 gives background on the tools and related technologies involved; Section 3 describes the protocol's working and the scenario; Section 4 presents some results obtained using an event-driven simulator, and, Section 5, concludes.

2 Related Work

Volunteer computing systems have become extremely popular as a means to garnish many resources for a low cost in terms of both hardware and manpower. The most popular volunteer computing platform currently available, the BOINC infrastructure [2] is composed of a scheduling server and a number of clients installed on users' machines. The client software periodically contacts the scheduling server to report its hardware and availability, and then receives a given set of instructions for downloading and executing a job. After a client completes the given task, it then uploads resulting output files to the scheduling server and requests more work.

For data distribution, BOINC projects generally use a single centralized server or a set of mirrors. This centralized architecture, although very effective, can be a potential bottleneck when tasks share input files or the central server has limited bandwidth. Increasing the number of mirrors to accommodate puts extra administrative burden on the project organizer and can prove very time consuming to manage. Popular and proven P2P technologies such as BitTorrent [8] [5] , or commercial solutions like Amazon's S3 or Googles GFS, could be fairly effectively applied to provide for the data needs of BOINC, at least as they relate strictly to distribution. However, in the case of commercial products, there is a monetary cost involved, and for P2P systems like BitTorrent, the facility to secure or limit who is able to receive, cache, or propagate different pieces of information is generally limited or nonexistent. For example, BitTorrent, like many other P2P systems, has focused on ensuring conventional file-sharing features, such as efficient transfers, equal sharing and file integrity.

However, Desktop Grid environments have different requirements to general file-sharing P2P communities because security becomes more of a complex issue than solely guaranteeing data validity. It can be a requisite that only certain amounts of data are shared with an individual peer, or communities are reluctant to introduce a system that would have peers directly sharing with one another, as it might have the potential (or perceived potential) to have security implications for clients as ports are opened for outside connectivity. It is therefore important not only for data integrity and reliability to be ensured, but also to have available safeguards that can limit peer nodes' exposure to malicious attacks. It is these types of requirements that has prompted our work to create a hybrid centralized/ decentralized P2P network that allows for data distribution while providing client safeguards and stricter controls for project administrators as to what network participants receive and distribute data [9].

3 The EDGeS protocol

In this section we illustrate how the protocol works and the sequence of messages exchanged between the nodes of the system. There are several kinds of nodes, which can play different *roles*. So we would like to give a brief overview of

what role each node on the network plays, and the functionalities associated with that role.

- the *Network Manager* manages the worker requests, assigning them the Work Units (WUs) and creating new work packages (DART). The Network Manager keeps all the data items and, at the beginning, it transfers these items on the cacher nodes in order to make the download phase of the workers faster. It has a cache of Data Center addresses that is always kept up to date, and that contains, for each cacher, the data items stored on it.
- the *Data Lookup Service*: a node that plays this role should provide the worker with a list of Data Centers which store the data item needed to execute the assigned Work Unit. So the worker can directly contact these cachers to retrieve the data item. The exact procedure to retrieve data is described in the following. In this work we assume that the Data Lookup Service is included in the Network Manager role, but in general these nodes could be different.
- the *Data Center* receives some data items from the Network Manager and stores them for worker requests. When a worker asks a cacher for some data, it sends this data through a direct connection.
- the *Worker Node* is a node able to execute a Work Unit. It periodically queries the Network Manger for a new job and then retrieves the data item needed to carry out that job. When it finishes to process a Work Unit (WU), it produces some results, but in this work we do not take into account the results and the procedure to collect them.
- *Aggregator (optional)*: as mentioned above, the results are not important in this phase. Anyway, an Aggregator node should be provided for future work in order to collect the outcome of Work Unit executions.

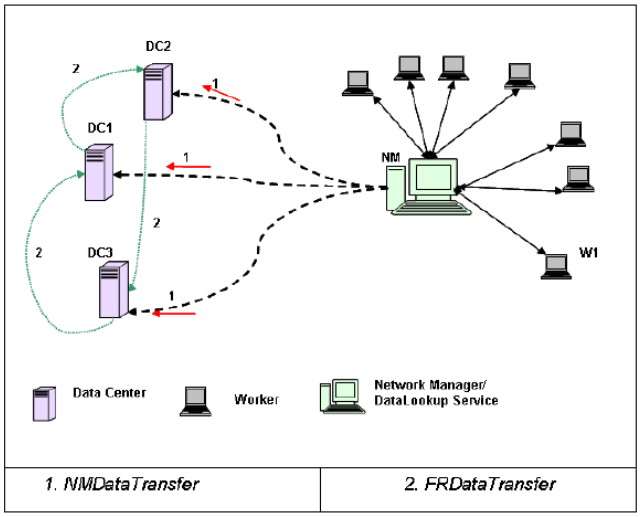

Figure 1: Caching algorithm in a sample network.

Figures 1 and 2 depict the sequence of messages exchanged among the Network Manager, the Data Centers and the workers. In particular, this sample network contains three Data Centers, seven workers and the Network Manager (which includes the functionality of the DataLookup Service).

At the beginning, the Network Manager sends its data items to the cachers. The particular caching strategy adopted in this work is the one described in [7] . Basically, what we aim to do is support the FastReplica method, to have an efficient and reliable replication of large files among Data Cachers.

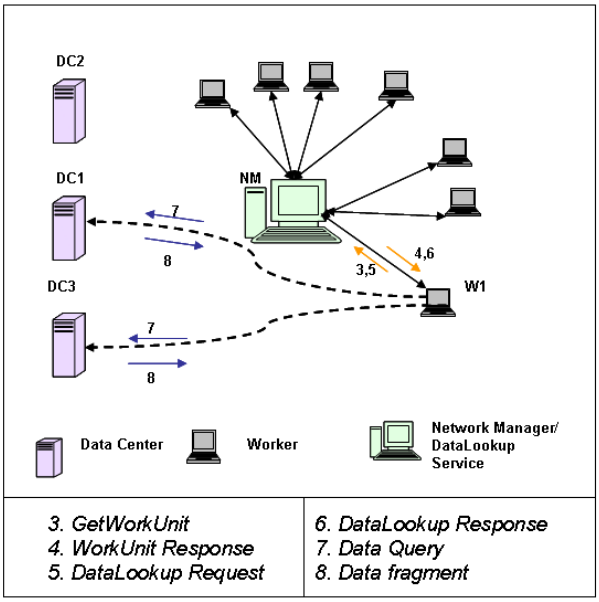

Figure 2: Work Unit's execution in a sample network.

There are a few basic ideas exploited in FastReplica. In order to replicate a large file among n nodes, the original file is partitioned into n subfiles of equal size and each subfile is transferred to a different node in the group. After that, each node propagates its subfile to the remaining nodes in the group. Thus, instead of the typical replication of an entire file to n nodes by using n Internet paths, by connecting the original node to the replication group, FastReplica exploits $n * n$ Internet paths within the replication group, and each path is used for transferring a $1/nth$ part of the file, so as to avoid that the bandwidth of the unique Data Source (the Network Manager) is saturated and therefore that the speed of Data Cachers in the retrieval of data is too slow.

In our work, the file to divide among cachers is represented by the set of all data items, and each data item is a subfile. Moreover, we introduce a new parameter, *DReplicationFactor*, that indicates the number of Data Centers on which each data item should be replicated. Substantially, it could not be useful to replace the same data item on all cachers. In Figure 1, for example, it can be noticed that there are three data items and a DReplicationFactor equals to two. Therefore the Network Manager sends a Data Item to each cacher (step 1, NMDataTransfer), and then the cacher propagates its Data Item to another cacher (step 2, FRDataTransfer), in order to replicate the data item on two node. The cacher sends the item to $DReplicationFactor - 1$ other cachers that are randomly chosen.

In our scenario, we have a total number of jobs or Work Units (*NJob*) and a number of different data items (*NDataItem*). The same data item can be given as input for different jobs. The number of times a data item needs to be assigned to workers is on average: $NdataAssign = Njob/NDataItem$. Moreover, we have N data cachers (*Ndatacachers*) and each data item is replicated on DReplicationFactor of these. In the absence of swarming, DReplicationFactor should be lower than NdataAssign, so that each of these DReplicationFactor data cachers can serve several job requests.

The FastReplica method avoids the Network Manager to become a bottleneck. In fact, instead of sending NDataItem multiplied by DReplicationFactor data items, it transfers NDataItem files, and then each cacher is responsible for the completion of the caching algorithm.

In Figure 2, we show the steps needed for a Work Unit execution. The worker W_1 joins the network and connects directly to the Network Manager NM asking for a job with a GetWorkUnit request (3).

In the step (4), the Network Manager answers to the worker, using a WorkUnit Response, and assigns a Work Unit to the worker. The message also contains other information, which the worker can then use to ask the manager for components the worker is missing, as for example, the required data item.

After the assignment phase, the worker should retrieve the data item concerning the assigned Work Unit. The Worker must search for a DataCenter that stores the required data package. The Network Manager keeps a cache of all DataCenter locations. The worker sends the DataLookup Request (step 5) using the information of the previous message (WorkUnit Response), to the DataLookupService (in this version of the protocol prototype, this is the Network Manager). The DataLookup Service/Network Manager gives a DataLookup Response (step 6) back to the worker, which contains the location of one or more DataCenters that have the data needed by the worker. The worker will retrieve the data item fragments simultaneously from different sources (steps 7 and 8), as in BitTorrent [8] , to minimize the download time.

The query sent by the worker to download data is DataQuery, and the cacher that receives this query responds with a DataFragment, sending a packet to the worker node for local processing.

This way, when the Network Manager receives a query for a WU, it returns to the worker a list of cachers that have that data item, and the worker can download the fragments of this data item in parallel from different sources. If the number of the fragments (*NFragment*) is lower than the number of Data Centers that have that data item, the Network Manager will choose $NFragment$ Data Centers to put in the list, among those that are less loaded. The Data packet sent to the worker contains the data itself (the payload) and other support information.

As mentioned, this phase of the protocol only copes with data/package distribution/delivery over the network. The results publication stages are not taken into account.

The phase in which the Data Cachers acquire data items and the one in which workers requires these data items are executed in parallel; when a worker asks for some data, the Network Manager gives to it a list of addresses of Data Cachers that have already retrieved that data item.

A simulation analysis was performed with an event-driven simulator, written in java and already presented in [4] , to test the performance of the protocol. The simulator was modified to support the different network topology needed in this work. In the following, we list some substantial differences and modifications:

- In EDGeS, the Data Center and the Network Manager are contacted directly, not through a P2P search. In the previous version of the simulator, a P2P search was performed each time a worker had to contact the Network Manager or a Data Center for the first time.
- In EDGeS, the Network Manager is the only Super-Peer of the network, to which all the workers must be connected (the workers are not connected among them). To be able to contact a Date Center, a worker must contact the Network Manager that has a list of available Data Centers and it communicates the addresses of one ore more of these to the worker. Conversely, in the previous work the workers could be connected to a Data Center or to a Rendevouz node.

In order to validate the protocol on real study cases, we retrieved some stats about three of the most important BOINC projects: Rosetta@Home, Einstein@Home and Seti@Home. Here a list of the most interesting parameters for these projects is reported:

- Size of a Work Unit (WU)
- Processing Time of a WU
- Size of Initial Data (the first WU that a worker performs requires more data, because, for example, it needs the source code of the application, etc.)
- Results/Day. The number of results products in one day corresponds to the number of WUs correctly performed by all the workers within one day
- Participants (workers)
- Replication factor. The replication factor defines how many times a WU must be performed in order to check results coming from different workers. It must not be confused with DReplicationFactor, that represents the number of times a data item should be replicated on the cachers.
- %NewWorkers: it indicates the percentage of new users in the network, that is, how many workers ask for a WU for the first time.

The tables 1, 2 and 3 show the stats mentioned above for these three projects.

In the simulation analysis we reduced the number of workers and consequently scaled the size of this projects in order to obtain the results more quickly. Finally, we give an overview of the performance indexes that we computed

Table 1: Rosetta@Home stats

Size of a Work Unit	3MB
Processing Time of a Work Unit	3h
Size of Initial Data	17MB
Results in a Day	115.000
Number of Data Centers	variable from 3 to 20
Participants	100.000

Table 2: Seti@Home stats

Size of a Work Unit	340KB
Processing Time of a Work Unit	2h
Size of Initial Data	2,5MB
Results in a Day	1 Million
Number of Data Centers	variable from 3 to 20
Participants	500.000
%NewWorkers	96 sec
Replication factor	2

through the simulation tests and analyzed. Some of these were already presented in [4] , some others are used here for the first time.

- the *Percentage of bandwidth utilization*: while in the previous works we calculated the utilization of the Data Centers as the percentage of time in which the Data Centers have at least one active connection (there is at least one worker that is retrieving data), here we calculate the percentage of bandwidth of the Data Centers that is used by the workers in a time unit. Since in BOINC it is not very realistic to assume that the bandwidth of a Data Center is equal to that of a worker, the Data Center is assumed to be a more powerful calculator with a larger available bandwidth (for example, 100 Mbps) than the workers, which usually have a domestic connection, 10 Mbps for example. Therefore we want to figure out how many connections are active in a time unit and what percentage of the bandwidth of a Data Center is utilized by the connections with workers. So, for example, if the number of connections to the Data Center, multiplied for the bandwidth that each worker requires (10 Mbps), is lower than the bandwidth of the Data Center (100Mbps), then each worker will have, as available bandwidth, exactly 10Mbps; otherwise the bandwidth of the Data Center is fairly divided among the connected workers (so they will have less than 10 Mbps). The same strategy of bandwidth sharing has been implemented between the Network Manager and the cachers.
- the *Average speed of download*: this index is defined as the average speed of a worker (in Mbps) in downloading the data items used to process the assigned Work Units.
- the *Percentage of new users*: due the vastness of the BOINC projects, which involve thousands of computers and in which new clients continuously join, in our tests we simulated the continuous arrival of new workers. The substantial difference among old and new workers is that the new ones download more data: for example, in Einstein@Home, the first WU asks for 40MB instead of 3.2. To simulate this, each worker requires to download 40MB of data not only for the first Work Unit that it must execute, but also for each successive Work Unit with a certain probability (which was set to 20 %). This way we wanted to simulate the joining of new workers in the network.
- the *Average download time* is the average needed time to download an entire data item.
- the *Overall execution time* is the time needed to retrieve and process all the Work Units.

Table 3: Einstein@Home stats

Size of a Work Unit	3,2MB
Processing Time of a Work Unit	5h
Size of Initial Data	40MB
Results in a Day	50.000
Number of Data Centers	variable from 3 to 20
Participants	200.000
Replication factor	2

4 Performance Evaluation

A simulation analysis was performed by means of an event-based simulator, in order to evaluate the performance of the EDGeS protocol described in the previous section. The simulation scenario, and the related network and protocol parameters, are set to assess the representative BOINC applications mentioned in Section 3. The DReplicationFactor was fixed to three in this set of experiments. Moreover, as we said above, we scaled the size of each project to quickly run simulations. A table containing the values of the number of workers and of results/day is reported below.

Table 4: Values used in the simulations

Rosetta@Home	
Number of Jobs	12000
Number of Workers	10000
Seti@Home	
Number of Jobs	10000
Number of Workers	5000
Einstein@Home	
Number of Jobs	5000
Number of Workers	10000

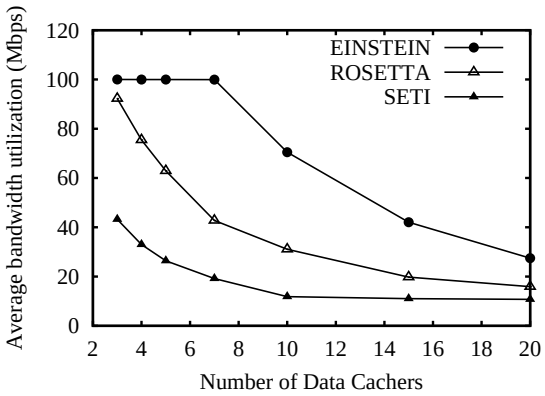

Figure 3: Average bandwidth utilization vs. the number of data centers, for the three different projects.

Figure 3 shows the average bandwidth utilization vs. the number of data centers, for the three different projects. It can be notice that, as the number of Data Centers increases, their bandwidth utilization decreases, because the worker can download data from more data sources. However, this trend has an optimum: if the number of cachers increases over a certain threshold, the bandwidth utilization does not decrease further, because the new cachers are not actually employed.

In figure 3 it can be seen that the different size of the data items in each project impacts heavily on the optimum

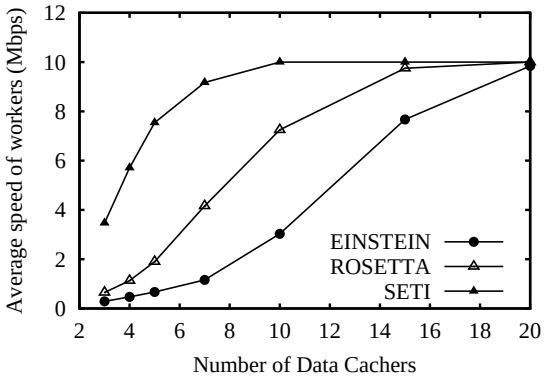

Figure 4: Average download speed of workers vs. the number of data centers, for the three different projects.

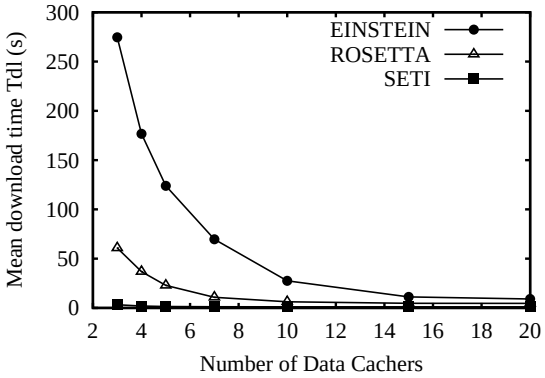

Figure 5: Average download time vs. the number of data centers, for the three different projects.

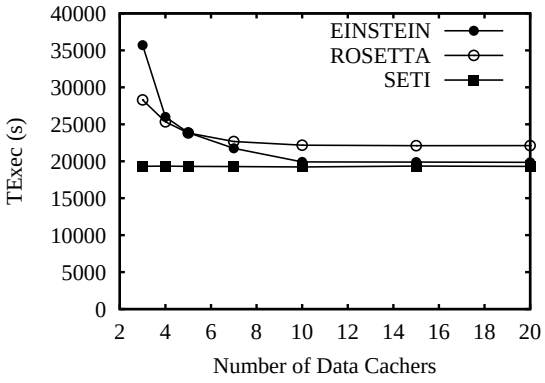

Figure 6: Overall execution time vs. the number of data centers, for the three different projects.

number of the Data Centers that should be used: SETI@Home, for example, uses small data items (about 300KB), and the optimum number of Data Center is about 10; in the case of Einstein@Home, on the other hand, this value is over 20, due to the larger size of a data item in this project.

From figure 4, we can derive the optimum number of Data Centers. The figure shows the mean speed of the workers in the download of data vs. the number of data centers, for the three different projects. In fact, using 10 Data Centers, workers can download at their maximum allowed speed, 10Mbps in Seti@Home, while 14 and over 20 cachers, respectively, are needed to obtain the same result in Rosetta@Home and Einstein@Home.

The figure 5 depicts the mean time that a worker takes to download a data item from the cachers. As the number of Data Cachers increases, the download time decreases, but the threshold mentioned above is confirmed here too: if the number of cachers increases over this threshold, these nodes could not be correctly employed.

Finally, in figure 6 we see the trend of the overall execution time vs. the number of cachers. It is interesting

to notice that, for Seti@Home, this value is almost constant. This behavior could be explained by looking at the execution time of a Work Unit. While in Einstein@Home and Rosetta@Home, the process time and the download time are comparable, in Seti@Home the download time is much smaller than the execution time, so an increase in the number of cachers does not affect the overall time to complete all the Work Units.

5 Conclusions

In this paper we have reported on the ongoing work and results of an architecture based on EDGeS protocol, for the execution of scientific applications according to the "public resource computing" paradigm, where resources are distributed and generally donated by network volunteers, and a large number of independent jobs are executed in parallel by dispersed worker nodes.

We showed one way to provide the needed bandwidth and data storage to support this scenario, exploiting the client-side network capacities in a P2P-type system for distributed data sharing and propagation. We showed a valid data distribution algorithm and a caching strategy based on the FastReplica method to enhance performances and considerably reduce the download time. Moreover, we implemented a technique to download a data item in parallel by multiple data sources, as in BitTorrent.

Future work in this area will investigate interesting research avenues, such as: (i) vary the $DReplicationFactor$ and analyze the advantages/disatvantages; and, (ii) comparing this data distribution algorithm with other kind of strategies.

6 Acknowledgements

This work has been partially supported by the European project EDGeS - enabling Desktop Grids for e-Science

References

[1] David P. Anderson. Public computing: Reconnecting people to science. In *Proceedings of Conference on Shared Knowledge and the Web*, pages 17–19, Madrid, Spain, November 2003.

[2] David P. Anderson. Boinc: A system for public-resource computing and storage. In *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 4–10, 2004.

[3] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11), 2002.

[4] Daniela Barbalace, Pasquale Cozza, Domenico Talia, and Carlo Mastroianni. A p2p job assignment protocol for volunteer computing systems. Technical Report TR-0117, Institute on Knowledge and Data Management and Institute on Architectural Issues: Scalability, Dependability, Adaptability, CoreGRID - Network of Excellence, December 2007.

[5] BitTorrent. See web site at http://www.bittorrent.com/.

[6] BOINC. Berkeley open infrastructure for network computing. See Web site at http://boinc.berkeley.edu/.

[7] Ludmila Cherkasova and Jangwon Lee. Fastreplica: efficient large file distribution within content delivery networks. In *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, pages 7–7, Berkeley, CA, USA, 2003. USENIX Association.

[8] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proceedings of the First Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[9] Fernando Costa, Luis Silva, Ian Kelley, and Ian Taylor. Peer-to-peer techniques for data distribution in desktop grid computing platforms. Technical Report TR-0095, Institute on Architectural issues: scalability, dependability, adaptability, CoreGRID - Network of Excellence, July 2007.

[10] Miguel Càrdenas-Montes, Ad Emmen, Attila Csaba Marosi, filipe Araujo, Gàbor Gombàs, Gabor Terstyanszky, Gilles Fedak, Ian Kelley, Ian Taylor, Oleg Lodygensky, Pèter Kacsuk, Ròbert Lovas, Tamas Kiss, Zoltàn Balaton, and Zoltàn Farkas. Edges: bridging desktop and service grids. In *2nd Iberian Grid Infrastructure Conference*, pages 2–4, Porto, Portugal, May 2008.

[11] EDGeS. Enabling desktop grids for e-science. See Web site at http://www.edges-grid.eu/.

[12] Einstein@home. See web site at http://einstein.phys.uwm.edu/.

[13] Rosetta@HOME. See web site at http://boinc.bakerlab.org/rosetta/.