

Optimizing the Data Distribution Layer of BOINC with BitTorrent

Fernando Costa , Luis Silva
{flcosta, luis}@dei.uc.pt
University of Coimbra
3030-290, Coimbra, Portugal

Gilles Fedak
fedak@lri.fr
Laboratoire de Recherche en Informatique (LRI)
INRIA Futurs, Orsay Cedex France

Ian Kelley
I.R.Kelley@cs.cardiff.ac.uk
School of Computer Science
Cardiff University, Cardiff, United Kingdom



CoreGRID Technical Report
Number TR-0139
June 17, 2008

Institute on Architectural issues: scalability,
dependability, adaptability (SA)

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

Optimizing the Data Distribution Layer of BOINC with BitTorrent

Fernando Costa , Luis Silva
{flcosta, luis}@dei.uc.pt
University of Coimbra
3030-290, Coimbra, Portugal

Gilles Fedak
fedak@lri.fr
Laboratoire de Recherche en Informatique (LRI)
INRIA Futurs, Orsay Cedex France

Ian Kelley
I.R.Kelley@cs.cardiff.ac.uk
School of Computer Science
Cardiff University, Cardiff, United Kingdom

CoreGRID TR-0139

June 17, 2008

Abstract

In this paper we show how we applied BitTorrent data distribution techniques to the BOINC middleware. Our goal was to decentralize BOINC's data model to take advantage of client network capabilities. To achieve this, we developed a prototype that adds BitTorrent functionality for task distribution and conducted medium-scale tests of the environment. Additionally, we measured the impact of the BitTorrent components in both the BOINC client and server, and compared it with the original implementation. Our preliminary results indicate that the BitTorrent client had a negligible influence on the BOINC client's computation time, even in the case where it was seeding extensively. The BOINC server, on the contrary, showed an unexpectedly low bandwidth output when seeding the file, as well as spikes on CPU usage. Current results show the BitTorrent scenario allows clients to share the burden of data distribution on BOINC with almost no negative influence on compute time. This paper will discuss the tests that were performed, how they were evaluated, as well as some improvements that could be made to future tests to enhance server-side efficiency.

1 Introduction

The use of personal computers' computational power as a tool for science has steadily increased in popularity. To this end, Desktop Grids have been extremely successful in bringing large numbers of donated compute cycles together to form a large-scale virtual supercomputer. These types of systems are especially well suited for highly parallel number-crunching computations that do not require much, if any, direct communication between network participants. Volunteer computing platforms such as BOINC [1][4] are currently the most successful Desktop Grid systems and primarily rely on donated computer cycles from ordinary citizen communities. BOINC is currently being successfully used by many projects to analyze data, and with a supportive user community can provide compute power to rival that of the world's supercomputers. In the current implementation the network topology is restricted to a strict

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

master/worker scheme, generally with a fixed set of centrally managed project computers distributing and retrieving results from network participants.

As volunteer computing projects gain in popularity and their user-bases expand, network requirements can easily become more demanding, forcing projects to upgrade both their computer hardware and network capacities to cope with increased demand. In these scenarios, the centralized data architecture currently employed by BOINC and other Desktop Grid systems can be potential bottlenecks when tasks require large input files or the central server has limited bandwidth. With new data management technologies it will be possible to explore new types of data-intensive application scenarios - ones that are currently overly prohibitive given their large data transfer needs. The lack of a robust and low-cost data solution for Desktop Grids can force application developers to scale back their applications to problems that do not rely upon large data sets. There are many applications that could either expand their current problem scope or migrate to a Desktop Grid environment if the middleware had support for scalable data management.

Peer-to-Peer (P2P) data sharing techniques [3][11][13] can be used to introduce a new kind of data distribution system for volunteer and Desktop Grid projects - one that takes advantage of client-side network capabilities. This functionality could be implemented in a variety of forms, ranging from BitTorrent-style networks where all participants share relatively equally [8], to more constrained and customizable unstructured P2P networks where more advanced scenarios for data distribution and discovery could be explored [15]. We have chosen to use BitTorrent [6] because it has proven to be both scalable and efficient and could be especially beneficial to projects that have large input files that need to be shared between several independent workers. Furthermore, BitTorrent would be advantageous for projects that have limited or slow outbound connections from the central project server since it would limit their needed bandwidth.

In this paper, we present a new data BOINC model using BitTorrent, discuss the relative advantages and disadvantages of this approach, and present the results we obtained from our first tests conducted with the platform.

This paper is organized as follows: section 2 gives background on the technologies and introduces related work; section 3 introduces how the BitTorrent protocol was used in our prototype; section 4 presents the results obtained from experimentation; and, section 5 concludes the paper.

2 Related Work

This section introduces the technologies we based our research on - BOINC and BitTorrent - and discusses related work on P2P and Desktop Grid systems.

The Berkeley Open Infrastructure for Network Computing (BOINC) [1][4] is a software platform for distributed computation using otherwise idle cycles from volunteered computing resources. BOINC's use is widespread, with many different and varying projects employing the core infrastructure to distribute their data processing jobs. The diverse scientific domains utilizing BOINC range from gravitational wave analysis, to protein folding, to the search for extraterrestrial life. Although these projects are diverse in their scientific nature, each one has something in common with the others: they have work units that can be easily distributed to run autonomously in a highly distributed and volatile environment. To achieve this task, each project must not only prepare its data and executable code to work with the BOINC libraries and client/server infrastructure, but they must also setup and maintain their own individual servers and databases to manage the project's data distribution and result aggregation. BOINC has been highly successful, and to date, over 5 million participants have joined various BOINC projects [2]. There are currently about 40 BOINC-based projects and about 400,000 volunteer computers performing an average of over 500 TeraFlops [18].

XtremWeb [5] is a Desktop Grid project that, like BOINC, follows a centralized architecture, using a three-tier design - Worker, Coordinator, Client -, and runs embarrassingly parallel applications. XtremWeb allows a set of Clients to submit task requests to the system which will execute them on Workers. The role of the third tier, called the Coordinator, is to decouple Clients from Workers and to coordinate tasks execution on Workers. To ease the deployment phase regarding the connection issues raised by firewall and NAT configuration, all the communications are initiated by Clients and Workers toward the Coordinator node.

Both these projects use a data distribution system that has one central point of failure. To distribute the data in a scalable way there are numerous alternatives in the form P2P file sharing systems or data storage systems.

BitTorrent [6] is a popular file distribution protocol based on the P2P paradigm. However, unlike other well-known P2P applications such as Gnutella or KaZaA, which incorporate peer and file discovery algorithms, BitTorrent's focus is more on optimising the distributed of files by enabling multiple download sources through the use of file partitioning, tracking and file swarming techniques. The main idea of BitTorrent is the collaboration between users accessing the

same file by sharing chunks of the file with each other. To obtain information about the file to download, a peer must download a corresponding .torrent file. This file contains the file's length, name and hashing information, and the url of a tracker, which keeps a global registry of all the peers sharing the file. Trackers help peers establish connections between themselves by responding to a user's file request with a partial list of the peers having (parts, or chunks of) the file. A tracker does not participate in the actual file distribution, and each peer decides locally which data to download based on data collected from its neighbours. Therefore, each peer is responsible for maximizing its own download rate. Peers do this by downloading from whoever they can and deciding which peers to upload to via a variant of tit-for-tat policy to prevent parasitic behaviour.

OceanStore [9] is a global, distributed, Internet-based storage infrastructure. It consists of cooperating servers, which work as both server and client. The data is split up in fragments which are stored redundantly on the servers. For search, OceanStore provides the Tapestry [13] subsystem, and updates are performed by using Byzantine consensus protocol. This adds an unnecessary overhead since file search is not a requisite for BOINC, and supporting replication implies the use of a distributed locking service, which incurs further performance penalties.

Collaborative content distribution protocols such as BitTorrent and P2P file sharing protocols such as Kazaa [11] or Gnutella [7] are promising technologies to distribute efficiently and massively large amount of data. In In this study of BitTorrent [8], authors show that the protocol features the following properties : reliability of file transfers even in the context of high volatility and node churn, scalability even when nodes show low to medium bandwidth (Internet) and ability to distribute large files even if the node originally serving the files has low communication capabilities.

As a consequence, there have been several attempts to evaluate the benefit of integrating such protocols into Grid middleware. Cigarra [20] and CompTorrent [22] are two BitTorrent-based Grid System. The former one, CompTorrent is a parallel computing system which relies on an extension of the BitTorrent protocol. In [12] and [21], authors experiment the execution of bag-of task application with large input files distributed with BitTorrent. The both conclusion converge to show that BitTorrent can significantly decrease the makespan of parallel execution if the scheduling strategies is carefully adapted. In [23] authors evaluate the use of the Pastry protocol for data server selection when executing a data-intensive bioinformatics application (BLAST) on a Desktop Grid.

However, none of this work address the issue of evaluating the integration of BitTorrent into the most popular volunteer computing system, the BOINC middleware. Such evaluation would impact the whole BOINC community and potentially enlarge the range of application supported by BOINC to application with large input files which, up to now, cannot be efficiently executed on BOINC.

3 Applying BitTorrent to BOINC

The BOINC architecture is based on a strict master/worker model, with a central server responsible for dividing applications in thousands of small independent tasks and then distributing the tasks to the worker nodes as they request the work units. To simplify network communication and bypass any NAT problems that might arise with bidirectional communication, the centralized server never initiates communication with worker nodes: all communication is instantiated from the worker when more work is needed or results are ready for submission. In the current implementation of BOINC, data distribution is achieved though the use of multiple centralized HTTP servers that share data with the entire network.

The centralized architecture of BOINC may lead to some potential bottlenecks and fails to take advantage of the client-side network bandwidth. If client-side network bandwidth could be used to distribute the data sets, not only would it allow for larger data files to be distributed, but it would also minimize the needed network capabilities of BOINC projects, thereby substantially lowering down the operation costs. To decentralize the current model as it relates to data, we propose the use of BitTorrent to optimize the data distribution.

There are some advantages and disadvantages when implementing a pure BitTorrent solution. The advantages are many, for example, BitTorrent has proven itself to be an efficient and low-overhead means of distributing data; it can scale easily to large numbers of participants; and it has built-in functionality to ensure relatively equal sharing ratios [8]. Some of these advantages however turn into disadvantages when trying to apply BitTorrent to a volunteer computing platform. For example, because of its flat topology, BitTorrent only works if enough nodes in its network are listening for incoming connections, which can be problematic when confronted with firewalls and NAT systems. Another potential disadvantage is related with the "tit-for-tat" sharing requirement, which forces most participants to share on a relatively equal scale to what they are receiving. Although this proves quite effective for preventing selfish file-sharing on traditional home networking systems, it is not necessarily a requirement when applying P2P

technologies to volunteer computing. For example, in the volunteer computing case, not everyone may wish to be a BitTorrent node but they may wish to offer their CPU time to a project. So, in the pure tit-for-tat BitTorrent world, this would not be possible.

3.1 BitTorrent Scenario

To apply this new scenario, both BOINC server and client software had to be modified. The BitTorrent protocol requires the integration of components such as a tracker, and a BitTorrent client. BOINC clients that had BT capability would be able to use the BT protocol to download by connecting to a tracker while others would continue using HTTP downloads as usual. When the BitTorrent tracker is installed on the central server, a port is defined to receive client requests (normally 6881). We decided to use a centralized tracker because the decentralized alternative is very recent, and the maintenance of the DHT requires each peer to maintain an orthogonal set of neighbours within the DHT and pay the communication costs in the face of high rates of churn [10]. For every input file that should be downloaded through BitTorrent, a *.torrent* file is created, pointing to the tracker in the central server. The torrent file is named by adding the extension *.torrent* to the original file's name: *file.data* - *file.data.torrent*. Both files are hosted on a project data server. A tracker, however, is not enough to allow BitTorrent transfers. The original data file has to be spread across the network from the central server. This means that a BitTorrent client has to be running along the other components of the server (Figure 1). To start sharing the file, the BOINC server must start the BitTorrent client to act as a seed and announce itself to the tracker.

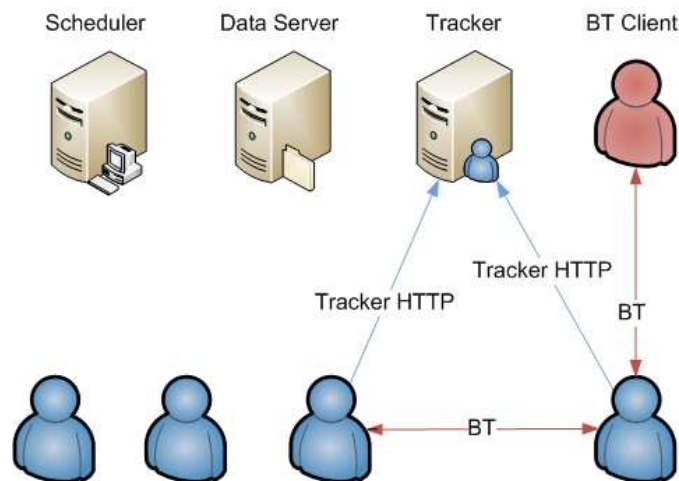


Figure 1: BT BOINC Server

The *.torrent* file is related to the data file through the work unit. When creating work, a tag `<bittorrent/>` is added to the file info of the data file in the work unit template and the *.torrent* file itself is added as an input file.

To incorporate a BT client's capability in BOINC, one could use one of the many BitTorrent clients available, or use a library to create a client. The idea in this research was to obtain a client that could be used on any platform, with the original, unaltered version of the BitTorrent protocol, and that was not on an experimental phase but had preferably been widely used. Experiments with BitTorrent had already been conducted successfully before, on XtremWeb [12], using the BitTorrent client Azureus [17], further pointing us towards that solution. We decided to use the original BitTorrent client because it features the original BitTorrent protocol, and it can be used in all platforms (Linux, Mac and Windows). The Azureus client, on the other hand, is written in Java, which would add an extra dependency for the JRE, and has extra functionalities.

Figure 2 shows the architecture and highlights the steps of a file transfer:

1. The client contacts the scheduler and asks for work. The scheduler then replies with a given work unit and a reference to a *.torrent* file that represents an input file made available via BitTorrent;
2. The client then downloads the *.torrent* file through HTTP from the specified Data Server;

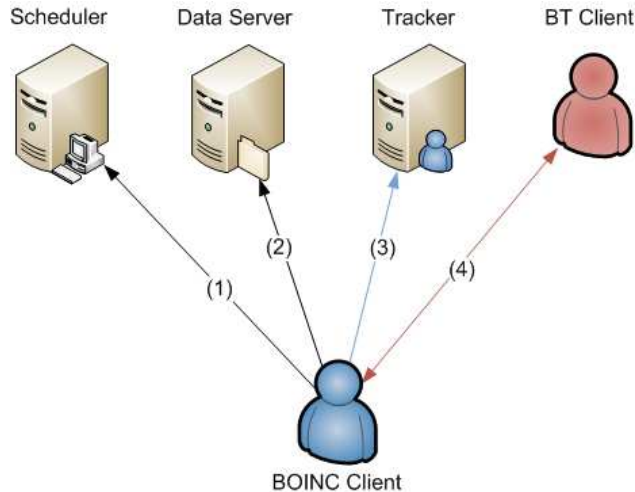


Figure 2: BOINC File Transfer

3. After downloading the .torrent file, the BOINC client initiates the local BitTorrent client with the .torrent as an argument. The BitTorrent library then contacts the tracker defined on the file and receives list of peers;
4. The client contacts the chosen peers and the BitTorrent protocol is used to download the subsequent file chunks and re-assemble the input file for processing by the local BOINC client.

The downloaded input file is then checked for integrity through its hash and size. After being verified, it is used for the processing of its workunit. The rest of the process is unchanged from the original BOINC.

4 Experimental Results

To test this new architecture, experiments of medium scale were performed, and various parameters were considered when trying new scenarios. The results of these tests are presented in this section, as well as information on the testing infrastructure.

4.1 Experiments Setup

The area of this research requires many machines to achieve meaningful results without having to resort to a simulator. This requirement was met by Grid'5000 [16], a project that serves as an experimental testbed for research in Grid Computing. Experiments were conducted on the Orsay site that was composed of 312 IBM eServer 326m machines, with dual-core AMD Opteron (246 or 250), and 2GB of RAM. Nodes are interconnected with a PCI-X Gigabit Ethernet card. The BitTorrent client used was the original one with version 5.0.7. The BOINC client version used was 5.8.8 and server version 5.9.3.

To evaluate this new scenario against the original one, a base of comparison must be used. Therefore, we had to develop a BOINC project that would be used by all the scenarios, creating a standard environment, and reducing the potential disparities in tests.

The application benchmark that was used in the project was based on an example provided together with BOINC's source code. The application has a single loop that performs some numeric calculations to keep the client busy for approximately 2 minutes.

We then proceeded with the creation of a work unit and a result template. This template was the default provided with the code for an example application. The work unit template, however, was slightly changed, since we had to identify the input files there. The input file, whose size was variable, was created with a simple text file. It was then referenced in the work unit template, along with the .torrent file, when testing the BitTorrent version. To analyze the

performance of the scenario, compared to the original BOINC, we used the Ganglia tool [14] and the monitoring of BOINC logs.

To evaluate the BitTorrent architecture we conducted several experiments while considering two variables: input file size and the number of clients.

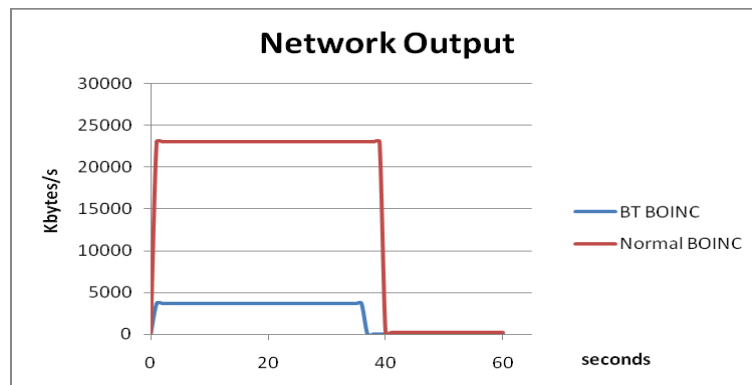


Figure 3: Server Network Output for 25 nodes and 30MB files

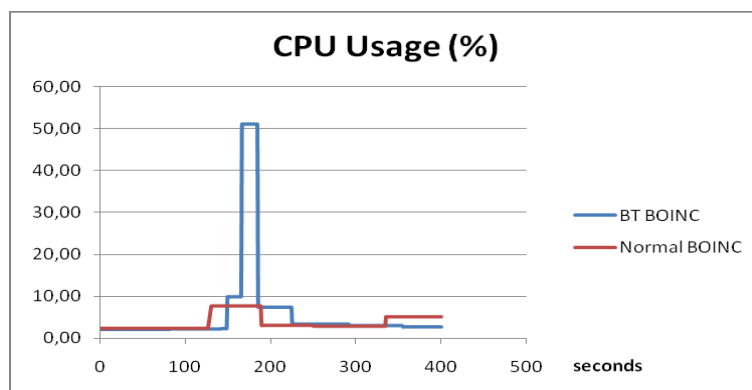


Figure 4: Server CPU usage for 25 nodes and 30MB files

4.2 Server Load

To evaluate the load on the central server we used two measures: CPU usage, and network output. Both were obtained from Ganglia. CPU usage helped us to measure the overhead caused by the BitTorrent tracker, while the server is serving requests for a varying number of clients. Network Output shows us how much contribution the clients have in the distribution of the file.

In our first experience we used 25 nodes and a 30MB file. The results are presented in Figure 4. The network output for BT BOINC was a little over 10% of the value for the original BOINC. The time spent uploading the file was also 3 seconds lower for BT BOINC.

For the same configuration, we analyzed the CPU Usage. Results are presented in Figure 5. In this test, we observed a spike in the servers CPU usage just before starting the upload of the file. With exception of that spike, the values were similar.

We proceeded with tests with a different number of nodes and different file size. In Figure 6 we present the server network output with 50 nodes. The Figure shows there was a considerable difference in outgoing bandwidth on the central server, with BT BOINC using a little more than 5 MB/s whereas Normal BOINC uses almost 30 MB/s. In this case, BT BOINC stopped distributing files a little later.

The CPU usage is presented on Figure 7. We observed a spike during the first seconds of the file transfer on BT BOINC, where transfer rates reached a little over 50%. After this initial stage, the value decreased to 5%, stabilizing alongside Normal BOINC.

In the next experiment we also used 50 nodes but we increased the file size to 40 MB. Figure 8 shows the network output for this case, where we can see a much lower network output from BT BOINC, after a slow start probably due to the slower boot of some of the nodes. On the second phase, the difference in bandwidth used is over 90% of Normal BOINC's output, which is extremely exaggerated, even considering the BitTorrent premise of sharing the data distribution between clients. It would be expected that the server used up more of its output bandwidth, so there may be a problem with the file transfer from the central server.

In Figure 9 we can conclude that the peak is very similar regardless of the number of nodes or file size, with values of up to 55%. These values are attributed to the BitTorrent extra components: the BitTorrent client and tracker. It is, however, important to state that the peak values do not have a direct correspondence to the servers upload period. We can see, for instance, that on Figure 3, the server seeded the file for nearly 80 seconds, whereas, as shown on Figure 4, the peaks in CPU usage happened in a 50 second period. Therefore, it is possible that the peak is caused by an initial period in the BitTorrent protocol where the servers BT client is connecting to the other peers. To determine this with certainty, tests on a larger scale, with longer upload periods should be conducted (with the internet as a testbed, preferably).

The server probes have shown that BT BOINC spends approximately the same time as Normal BOINC when uploading the files. It was not expected for the biggest difference in distribution time to be achieved with the smaller number of nodes, on relatively large files: 25 nodes and 30 MB file. Since it was uploading a relatively small file to few users, the server did not have to use a big portion of its output bandwidth, therefore reducing the difference between its output and BT BOINC's output. On bigger files, there is a larger difference since the Normal BOINC server uses up much more bandwidth, while BT BOINC shows only a slight increase in output.

That is another point worth noting: the server in BT BOINC never used more than 10 MB/s of its output bandwidth. This potential throttle may be caused by the BitTorrent client himself, which can be extremely limiting in the capacity to obtain better results. This pushes us to try new clients, or use torrent libraries to increase the servers bandwidth contribution. If the problem is not caused by the client, but rather the protocol itself, the server should be stressed, by placing it further away from the clients, to limit its output. The use of libraries could prove to be even more beneficial since it should reduce the CPU usage in the server, with particular emphasis on the spikes encountered.

4.3 Computational Time

Figure 10 presents a graph with the computational time for various file sizes and number of clients. This allows us to identify more precisely just how much BOINC's main objective (to perform computation) is hindered by the BitTorrent client/protocol.

The BT BOINC client was configured to seed the file for 1 minute after its download. We decided to use this value since the computation time was normally slightly over 2 minutes, and seeding for 50% of the computation time

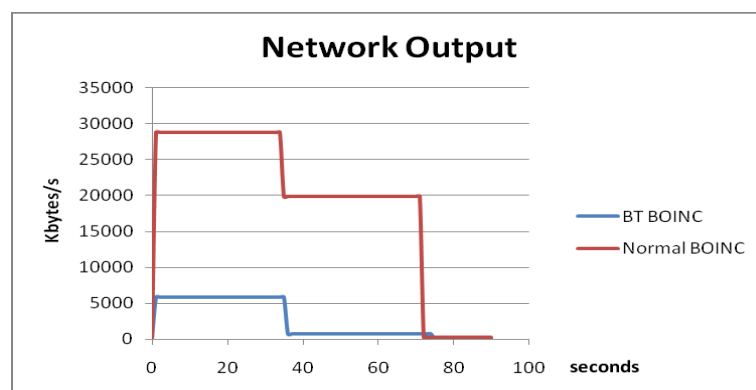


Figure 5: Server Network Output for 50 nodes and 30MB files

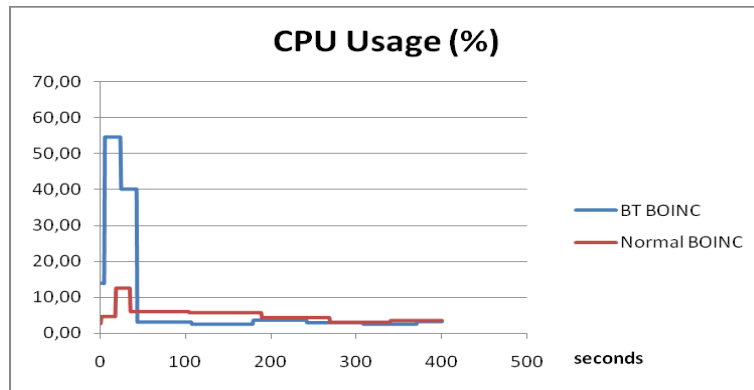


Figure 6: Server CPU usage for 50 nodes and 30MB files

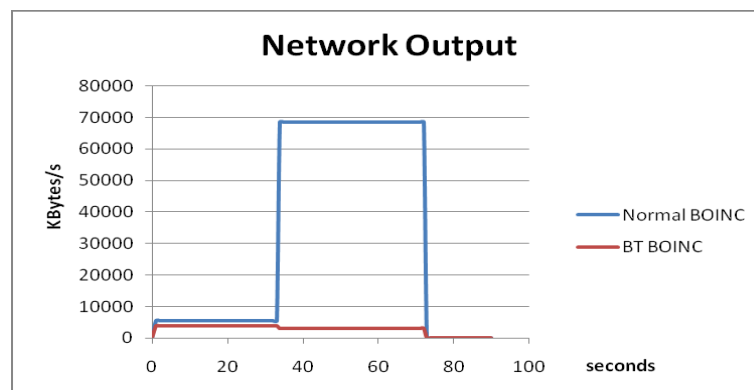


Figure 7: Server network output for 50 nodes and 40MB files

seemed a reasonable compromise.

Figure 10 presents the computation times observed after running experiments with 25 nodes and varying file sizes. We can observe that BT BOINC has very similar computation times, with a slight overhead in mean time of 1 to 2 seconds. We can deduce from this that, in this scenario, for every 2 minutes of computation, with half the time seeding, there was an average overhead of 1,15 seconds, which is a little under 1% of overhead. This is an acceptable value: in a two-hour computation the overhead was around 72 seconds.

5 Conclusions

This study allowed us to establish a few bridges to the expected theoretical results but also presented us with some interesting lessons.

The first lesson was that the current BitTorrent implementation could save the central server an enormous amount of bandwidth when distributing files. In the preliminary tests explored in this paper, using BT BOINC reduced network requirements imposed upon the central server by over 90%. These results came as a pleasant surprise, even with the premise of clients sharing data distribution among them. This positive outcome can suggest a potential limitation on the normal BitTorrent client to reach high bandwidth outputs. This potential throttle can prevent BitTorrent from faster distribution of the file in an initial phase. Future work should comparatively test the use of different BitTorrent clients, or a BitTorrent library such as libtorrent [19], which reportedly can seed up to three times faster than a normal client. Regarding the overhead caused by the BitTorrent components added to BOINC, the results showed that the overall client computation time is practically unaffected by the addition of the BitTorrent client. With a seeding time

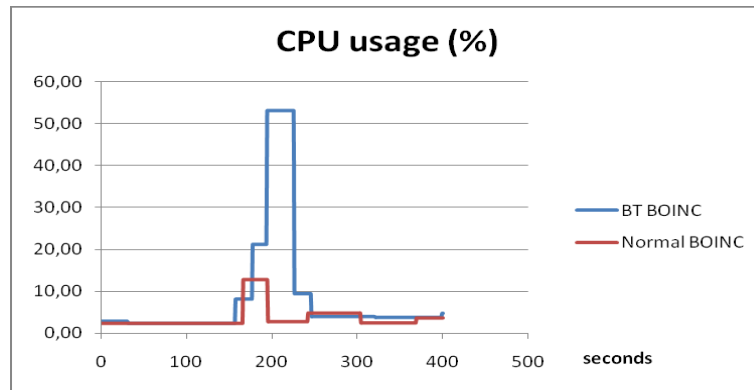


Figure 8: Server CPU usage for 50 nodes and 40MB files

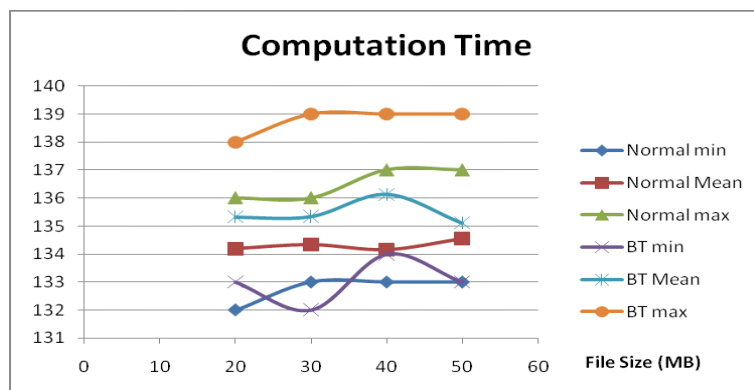


Figure 9: Computational time: max, mean and min for BT and Normal BOINC

of approximately half the computation time, the overhead caused was a little under 1% of the total computing time, a quite acceptable value in our estimation. Results in [12] showed that BitTorrent was not efficient for small files, and that FTP should therefore be used instead. In this case, the overhead was low regardless of file size, which would allow for an easier integration, as we would not have to manage two protocols but only BitTorrent in place of the HTTP protocol.

On the server side, the joint use of both the BitTorrent client and a tracker caused spikes in CPU usage, that reached a 50% value, which can be problematic should the server require more computing power for other tasks. However, the spikes were not long lasted, suggests an initial startup time when establishing connections to the other peers, followed by less demanding slowing down period.

To conclude, BitTorrent shows substantial promise to be used as an effective tool in Desktop Grid systems. Further and more exhaustive testing is needed to determine more specific advantages and drawbacks of this protocol in a real and production Desktop Grid environment.

Acknowledgements

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research. This research work is carried out in part under the European Union FP6 CoreGRID Network of Excellence (Contract IST-2002- 004265).

References

- [1] David Anderson, BOINC: A System for Public-Resource Computing and Storage, *In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, November 2004.
- [2] David Anderson, Volunteer Computing: Planting the Flag, *PCGrid 2007 Workshop*, Long Beach, March 30 2007.
- [3] H. Balakrishnan, F. Dabek, M.F. Kaashoek, D.R. Karger, D. Liben-Nowell, R. Morris, and I. Stoica, Chord: a scalable peer-to-peer lookup protocol for Internet applications, *Networking, IEEE/ACM Transactions on*, Volume **11**, February 2003.
- [4] Berkeley Open Infrastructure for Network Computing (BOINC).
See web site at: <http://boinc.berkeley.edu/>.
- [5] F. Cappello, S.Djilali, G.Fedak, T.Herault, F.Magniette, V.Neri, and O.Lodygensky, Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with Grid, *FGCS Future Generation Computer Science*, 2004.
- [6] Bram Cohen, Incentives build robustness in BitTorrent, *Proceedings of IPTPS*, 2003
- [7] Gnutella Project.
See web site at: <http://www.gnutella.com/>
- [8] M. Izal, G. Urvoy-Keller, E.W. Biersack, P. A. Felber, A. A. Hamra, and L. Garces-Erice, Dissecting BitTorrent: Five Months in a Torrents Lifetime, *In Proceedings of Passive and Active Measurements (PAM)*, 2004.
- [9] J. Kubiawicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells and B. Zhao, Oceanstore: An architecture for global-scale persistent storage, *9th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2000.
- [10] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil, A performance vs. cost framework for evaluating DHT design tradeoffs under churn, *IEEE Conference on Computer Communications (INFOCOM)*, 2005.
- [11] J. Liang, R. Kumar, K. W. Ross, The KaZaA Overlay: A Measurement Study, *Computer Networks Journal*, Oct. 2005.
- [12] Baohua Wei, G. Fedak, and F. Cappello, Scheduling independent tasks sharing large data distributed with BitTorrent, *Grid Computing, 2005. The 6th IEEE/ACM International Workshop on*, IEEE Computer Society, 2005, pages 219-226.
- [13] B. Y. Zhao, J. Kubiawicz, and A. Joseph, Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing, *UCB Tech Report UCB/CSD-01-1141*, University of California, Berkeley, 2001.
- [14] M. L. Massie, B. N. Chun, and D. E. Culter, The Ganglia Distributed Monitoring System: Design, Implementation, and Experience, *Parallel Computing*, vol. 30, pp.817-840, July 2004
- [15] Peer-to-Peer Architecture for Data-Intensive Cycle Sharing (P2P-ADICS).
See web site at: <http://www.p2p-adics.org/>
- [16] Grid5000.
See web site at: <http://www.grid5000.fr/>
- [17] Azureus.
See web site at: <http://azureus.sourceforge.net/>
- [18] Derrick Kondo, David P. Anderson and John McLeod VII, Performance Evaluation of Scheduling Policies for Volunteer Computing, *3rd IEEE International Conference on e-Science and Grid Computing*, Bangalore, India, December 10-13 2007.
- [19] libtorrent.
See web site at: <http://libtorrent.sourceforge.net/>

- [20] Alexandre Freire da Silva, Francisco Gatto, and Fabio Kon, Cigarra - A Peer-to-Peer Cultural Grid, *Proceedings of the FISL Workshop on Free Software*, . pp. 177-183. Porto Alegre, 2005.
- [21] C. Briquet, X. Dalem, S. Jodogne and P.A. de Marneffe, Scheduling data-intensive bags of tasks in P2P Grids with bittorrent-enabled data distribution, *In Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, Monterey, California, 2007.
- [22] B. Goldsmith, Enabling Grassroots Distributed Computing with CompTorrent, *Sixth International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2007)*, Honolulu, Hawaii, USA.
- [23] Jinoh Kim, Abhishek Chandra, and Jon B. Weissman, Exploiting Heterogeneity for Collective Data Downloading in Volunteer-based Networks, *in Proceedings of the 7th IEEE International Symposium on CLuster Computing and the Grid (CCGRID'07)*, pp. 275-282, 2007.