BOINC-MR: MapReduce in a Volunteer Environment[⋆]

Short Paper

Fernando Costa, Luís Veiga, and Paulo Ferreira

Distributed Systems Group, INESC-ID
Technical University of Lisbon
R. Alves Redol, 9
1000-029 Lisboa, Portugal
fcosta@gsd.inesc-id.pt

Abstract. Volunteer Computing (VC) harnesses computing resources from idle machines around the world to execute independent tasks, following a centralized master/worker model.
We present BOINC-MR, a system able to run MapReduce applications on top of BOINC, the most popular VC middleware in existence. We describe BOINC-MR's architecture and evaluate its performance with a typical MapReduce application. Our results show that BOINC-MR yields a performance increase of 64 % in application turnaround time and close to 50 % reduction in bandwidth usage in the server side, when compared to the unmodified BOINC system.

Keywords: Volunteer Computing, MapReduce, Adaptive Middleware

1 Introduction

The use of personal computers' computational power as a tool for science has steadily increased in popularity. To this end, Volunteer Computing (VC) systems have been extremely successful in bringing large numbers of donated compute cycles together to form a large-scale virtual supercomputer. Applications running on this infrastructure tackle problems from a wide range of scientific subjects, from physics to biology, and are tailored for highly parallel number-crunching computations.

BOINC [2] is a VC middleware that currently supports over 40 projects and bolsters a user base of around 450 thousand active machines, making it the most popular system in the world, rivaling the world's supercomputers in computing power. In its current implementation, the network topology is restricted to a

⋆ This work was partially supported by national funds through FCT – Fundação para a Ciência e Tecnologia, under projects PTDC/EIA-EIA/102250/2008, PTDC/EIA-EIA/108963/2008, PTDC/EIA-EIA/113993/2009 and PEst-OE/EEI/LA0021/2011.

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

Such a centralized architecture is the source of a potential bottleneck in the continuing evolution of Volunteer Computing systems. As projects gain in popularity and their user-bases expand, network and storage requirements can easily become more demanding, thus increasing the load on the server(s). There are worrying signs of stagnation in the number of active users and projects [1], and emerging problems in data distribution and storage [6].

Thus, one must look at alternative computing paradigms that may help Volunteer Computing reach its untapped potential. MapReduce is a widely used computing paradigm, proposed by Google [8], that has obtained considerable support in Cloud Computing communities due to its simplicity, scalability and performance in commodity clusters.

Our goal is to support MapReduce on top an insecure, unreliable VC environment, by taking advantage of the vast improvements in network infrastructure and disk storage in the last mile of the Internet. In this paper, we present BOINC-MR, a BOINC prototype that can run MapReduce jobs, and evaluate its performance in a real-world scenario.

This paper is organized as follows: Section 2 gives background on BOINC and MapReduce; Section 3 discusses the concepts we have just mentioned in more depth; experimental results are presented in Section 4; Section 5 introduces related work; and Section 6 concludes.

2 Background

This section introduces the BOINC system and MapReduce programming paradigm we based our research on.

In order to distribute its work units , each BOINC [2] project has to build its data and executable code as well as setup and maintain their individual servers and databases. Result validation is obtained through the use of task replicas, so that upon task completion, a quorum must be reached by a majority of users before an output can be considered correct.

Most Desktop Grids, such as BOINC or XtremWeb [3], have centralized architectures, in which a server or coordinator is responsible for scheduling task execution. There are exceptions [5], but they are either insignificant in scope or tailored to a different environment.

Such architectures and the limited support for complex applications may have brought on a significant problem: the number of active projects has stagnated. This in turn has lead to a 15 % decrease in the number of active users [1], a number that is expected to dwindle unless new alternatives are presented that may spark the interest of volunteers.

MapReduce is a software framework for parallel data-intensive computations recently popularized by Google [8]; it is able to represent a wide range of applications, by providing an abstraction for parallel execution (”map”) and aggregation of results (”reduce”).

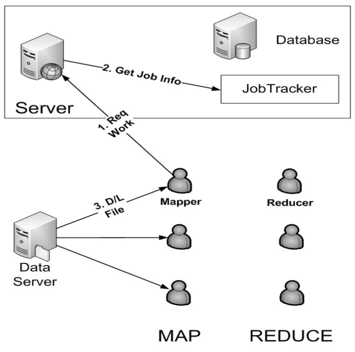

Fig. 1. BOINC-MR Map Phase.

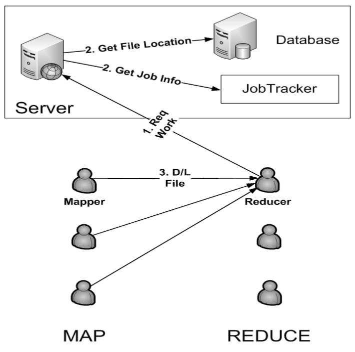

Fig. 2. BOINC-MR Reduce Phase.

MapReduce input is initially split into several chunks, each to be executed by an independent "map" task, assigned to each worker by a centralized master node. Each worker node processes the map task it was given, and reports its completion to the master. For the "reduce" step, a predetermined number of reduce tasks are created, whose goal is to perform join operations on the map outputs. All reduce inputs are therefore outputs from the previous map task. Throughout the rest of the paper, we will be referring to them as map outputs. Once all map outputs have been downloaded, the reduce task is executed and its final result is saved.

3 BOINC-MR Architecture

BOINC-MR supports the MapReduce paradigm in an unreliable, unsecured Internet environment. One of our main goals was to improve performance when adapting MapReduce to BOINC. In order to achieve this, BOINC-MR decentralizes data distribution and removes unnecessary overhead from the central server by leveraging inter-client transfers. MapReduce is an ideal framework to evaluate the impact of our proposition, since the map stage produces intermediate results that are used as input by reduce tasks.

Map tasks are embarrassingly parallel, with no dependencies or any shared data between them, which allowed us to use the traditional scheduling mechanism when dealing with this step. The BOINC-MR map stage is shown in Fig. 1: (1) A user (mapper) starts by requesting work from the projects central server; (2) The server takes into account the workload of each mapper, as well as its hardware and availability information, and dispatches work units that fit the request; (3) Each mapper downloads input and executable files from the data server, and runs the application. The computation results are then returned to the central server. The server keeps track of which mapper is holding each output file by storing that information in the database.

The reduce stage is depicted in Fig. 2: (1) A user (reducer) requests work from the projects central server; (2) The scheduler appends to each reduce task information the address (IP and port) of mappers holding output for the same job; (3) The reducer then has the possibility of downloading the required input files directly from the mappers. The server also stores a copy, thus providing a failover mechanism in case of error and guaranteeing data availability. After downloading all required files, each reducer executes its task and returns the final result to the server.

3.1 BOINC-MR Client

A BOINC-MR client requests work by sending the server a message with host characteristics and other information necessary for task scheduling. If there is work available, the server reply includes information on the task to be executed (mentioned in step (2) of Fig. 1 and 2). This task information allows clients to identify which tasks belong to MapReduce jobs.

Once a map task is obtained, the BOINC-MR client acts as mapper and runs the executable to produce the results. Mappers who have finished their task make their output available for reducers to download. We consider mappers to be hosting map outputs for as long as the files are available for download. A BOINC-MR client only accepts incoming requests for its output files, while rejecting messages that do not comply to a predefined file request template. Each mapper will stop accepting connections if one of the following situations occur: the BOINC-MR client is shut down; the MapReduce job has completed successfully; or the mapper has reached a timeout in total hosting time.

If a BOINC-MR client obtains a reduce task, it becomes a reducer. After parsing the task information (sent in step (2) of Fig. 2), the reducer is able to identify which map output files can be downloaded directly from mappers and which files are only available in the server. The BOINC-MR reducer always attempts to download from a mapper before resorting to the server. Each map output file may have multiple mappers hosting it, so the reducer goes through each mapper in the list in order. The mapper address list is ordered randomly at the server side, to prevent the overloading of a single BOINC-MR mapper.

We use a fall back mechanism for failed inter-client downloads. After n failed attempts to download an output file directly from mappers, the reducer resorts to downloading all missing files from the server.

3.2 MapReduce in BOINC Server

The BOINC-MR server must ensure a timely and valid transition between map and reduce steps. It must be able to deal with both BOINC-MR and BOINC clients, and provide information that allows each client to handle each task according to is characteristics. In order to differentiate map tasks from ”normal” (non MapReduce) ones, we modify their templates by adding ”<*mapreduce*>” tags with additional information such as job id and stage.

The BOINC-MR server uses an additional general configuration file to co-ordinate between stages and handle task creation. This file is responsible for defining global MapReduce parameters for each job, such as the number of map

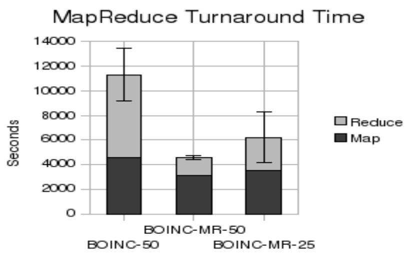

Fig. 3. Weight of Map and Reduce stage in MapReduce job.

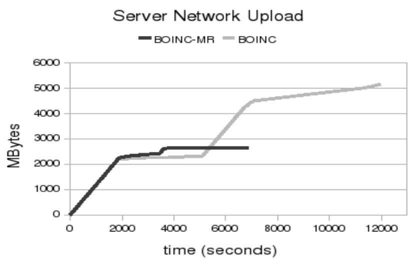

Fig. 4. Network Upload Bandwidth on server with 50 nodes.

and reduce tasks. The server uses a dynamic work unit creation mechanism, which is activated as soon as all map tasks have been returned and validated. This mechanism takes all the information provided by the mappers hosting file outputs to edit the necessary templates and insert reduce work units into the server's database.

Therefore, all reduce tasks sent to BOINC-MR clients (reducers) have the location of the required input data, as IP addresses of mappers and as the projects data server address (URL). It is worth noting that providing the server URL allowed us to guarantee retro-compatibility with unmodified BOINC clients.

4 Experiments

To evaluate our prototype, we use PlanetLab [4], a distributed testbed designed for applications deployed over the Internet. We present the results and implementation details in this section.

We use either 25 or 50 PlanetLab nodes as clients, and one node to act as server. To evaluate our scenarios, we create a BOINC project to run the word count MapReduce application. In word count, each map task receives a text document as input, counts the number of words in it and outputs an intermediate file with "word 1" pairs for each word found. The reduce step collects all the map intermediate outputs and aggregates them into one final output. In our experiments, we use an initial input file of 1 GB, divided into 100 chunks (10MB), each to be handled by a different map task.

Our goal is to measure the performance of BOINC-MR when running MapReduce applications, especially in two axis: application turnaround and server bandwidth usage.

4.1 Network Bandwidth and Application Turnaround

We use either 25 or 50 client nodes, while the BOINC-MR server replicates each task twice. Figure 3 shows the application turnaround time results. The BOINC column corresponds to a scenario with 50 unmodified BOINC clients.

BOINC-MR-50 and BOINC-MR-25 represent 50 and 25 nodes, respectively, running BOINC-MR. While the difference in the Map stage is not significant, with BOINC-MR doing slightly better, the Reduce stage shows remarkable improvements. This speedup is due to the fact that BOINC-MR employs inter-client transfers, and because the server spends more time uploading files with BOINC clients. Therefore, it has a higher chance of experiencing higher load due to other images running on its PlanetLab node. With respect to BOINC-MR clients, 25 nodes (BOINC-MR-25) performed worse than 50 nodes (BOINC-MR-50) in the reduce stage. This was due to a smaller number of nodes hosting the map output files. Overall, BOINC-MR takes less than half the time (46%) needed by the unmodified BOINC to complete the MapReduce job.

In order to more accurately evaluate the overhead on the server, we measure its bandwidth usage when running BOINC-MR clients. We do not present the results of network traffic from clients to the server since the server downloads the same amount of data from either BOINC-MR or BOINC clients. In both cases, the server has to download the map and reduce output from each client.

Figure 4 presents the data uploaded by the central data server, when using either BOINC-MR or BOINC clients. As BOINC-MR is faster than BOINC, its experiment ends earlier, after 7000 seconds, while BOINC clients only finish at the 12.000 second mark. We can observe an initial increase in uploaded data in both scenarios, which corresponds to the distribution of map inputs from the server to the clients. Around second 2000, the map tasks seem to have been deployed since we reach a plateau in both scenarios, which is only interrupted when the reduce step begins. The server running with BOINC-MR clients has a slight increase around second 4000 when it starts uploading the reduce task executable file to clients. On the scenario with BOINC clients, however, we can witness a steep slope in upload bandwidth from the central server to clients around second 5000. The server, being the sole owner of reduce input files, must upload all the data to clients. The server in the BOINC-MR scenario ends up with around 2600 MBytes of uploaded data. On the other hand, using BOINC clients forces the server to upload close to 5200 MBytes. This means that the BOINC-MR client can cut the server's upload bandwidth consumption in half.

4.2 Replication Factor

At this point, we evaluate the impact of the replication factor on the map task. Figure 5 shows the results for our experiments with 2 and 3 replicas for the map task. BOINC-2 and BOINC-3 represent the original BOINC client, with a replication factor of 2 and 3, respectively. BOINC-MR2 and BOINC-MR3 are the corresponding BOINC-MR clients. For the unmodified BOINC, using a higher replication factor helped speedup both the map and reduce stage. This can be explained by the lower impact of a slower node. With only 2 replicated tasks, both results are needed to validate the work unit, which means that any node holding a task can slow down the whole computation by not returning it in time. Having 3 nodes makes the slower one redundant.

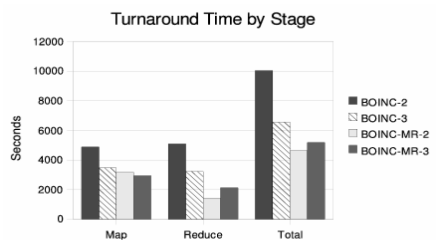

Fig. 5. Turnaround time of BOINC and BOINC-MR clients.

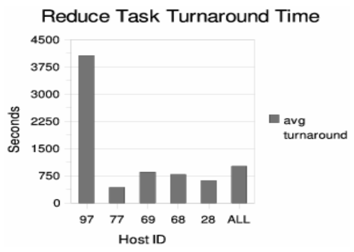

Fig. 6. Reduce task execution time in different hosts.

In the BOINC-MR client scenarios, we can see a slight speedup in the map stage which is attributed to the aforementioned lower impact of slower nodes. However, using 3 map task replicas produced worse results in the reduce step. This was unexpected since having 1 more mapper to download map outputs from should improve inter-client transfer speed. There are two explanations for these results. First, the current version of our prototype does not use any heuristic or complex algorithm when choosing which node to download each map output from. Secondly, we can witness a recurring event that is presented in Fig. 6. In cases with low replication such as this (2 reduce task replicas), one reducer's output cannot be discarded as there is no third or fourth reducer running. If a slower reducer is able to obtain several tasks it will reduce the application turnaround time. In Fig. 6, we can see that node 97 is 5 times slower than any other node. This means that, even after all the other reducers have returned their output, the MapReduce job will only end when this node returns its results.

5 Related Work

Combining the concepts of Cloud and Volunteer Computing has already been proposed in [9], in which the authors studied the cost and benefits of using clouds as a substitute for volunteers or servers.

There are two projects that have adapted MapReduce to a desktop grid. MOON (MapReduce On opportunistic eNvironments) [10] is a Hadoop[1] extension that adds adaptive task and data scheduling mechanisms for an enterprise desktop grid.

The work that most closely resembles ours was presented in [11], and, as MOON, introduces MapReduce to desktop grids. XtremWeb has been used in much smaller scale than BOINC, however, and its typical use scenario consists of a federation of research labs.

[1] Apache Hadoop. `http://hadoop.apache.org/`

6 Conclusion

We have presented BOINC-MR, a working prototype that allows MapReduce applications to run on top of a VC system, BOINC. Our results have shown that we can have a significant improvement in both performance and server bandwidth efficiency if we tailor this paradigm to our wide area environment. We have shown that BOINC-MR takes less than half the time (46%) needed by the unmodified BOINC to complete a word count MapReduce job. Furthermore, using BOINC-MR clients can cut bandwidth consumption in half on the server side, by successfully leveraging client's resources. We have also detected an excessive impact of slower nodes on application turnaround, when clients with limited bandwidth obtain a large number of tasks.

References

1. Anderson, D.: Boinc status report. In: The 7th BOINC Workshop. (2011)
2. Anderson, D.P.: Boinc: A system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing. GRID '04, Washington, DC, USA, IEEE Computer Society (2004) 4–10
3. Cappello, F., Djilali, S., Fedak, G., Herault, T., Magniette, F., Néri, V., Lodygensky, O.: Computing on large-scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid. Future Gener. Comput. Syst. **21** (March 2005) 417–437
4. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: Planetlab: an overlay testbed for broad-coverage services. SIGCOMM Comput. Commun. Rev. **33** (July 2003) 3–12
5. Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., Mowbray, M.: Labs of the world, unite!!! Journal of Grid Computing **4** (2006) 225–246
6. Costa, F., Kelley, I., Silva, L., Fedak, G.: Optimizing data distribution in desktop grid platforms. Parallel Processing Letters (PPL) **18**(3) (September 2008) 391–410
7. Cunsolo, V.D., Distefano, S., Puliafito, A., Scarpa, M.: Volunteer computing and desktop cloud: The Cloud@Home paradigm. In: Eighth IEEE International Symposium on Network Computing and Applications, IEEE (July 2009) 134–139
8. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51** (January 2008) 107–113
9. Kondo, D., Javadi, B., Malecot, P., Cappello, F., Anderson, D.P.: Cost-benefit analysis of cloud computing versus desktop grids. In: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing. IPDPS '09, Washington, DC, USA, IEEE Computer Society (2009) 1–12
10. Lin, H., Ma, X., Archuleta, J., Feng, W.c., Gardner, M., Zhang, Z.: Moon: Mapreduce on opportunistic environments. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. HPDC '10, New York, NY, USA, ACM (2010) 95–106
11. Tang, B., Moca, M., Chevalier, S., He, H., Fedak, G.: Towards mapreduce for desktop grid computing. In: Proceedings of the 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. 3PGCIC '10, Washington, DC, USA, IEEE Computer Society (2010) 193–200