A volunteer computing system implemented with peer-to-peer communication optimized for small and limited environments

Caio Santiago[1], **Luciano Antonio Digiampietri**[2]

[1]Bioinformática – Universidade de São Paulo (USP)
05.508-090 – São Paulo – SP – Brazil

[2]Escola de Artes, Ciências e Humanidades – Universidade de São Paulo (USP)
03.828-000 – São Paulo – SP – Brazil

{caio.santiago,digiampietri}@usp.br

Abstract

The computational needs of scientific experiments often require powerful computers. One alternative way to obtain this processing power is taking advantage of the idle processing of personal computers as volunteers. This technique is known as volunteer computing and has great potential in helping scientists. However, there are several issues which can reduce the efficiency of this approach when applied to complex scientific experiments, such as, the ones with long processing time, very large input or output data, etc. In order to face these challenges, we designed a volunteer computing system based on peer-to-peer communication. When compared with the local execution of activities and traditional volunteer computing, the execution time was improved and, in some cases, there was also a reduction of the server upload bandwidth use.

1. Introduction

The computational requirements of scientific experiments often demand powerful computers, which are usually expensive and, probably, they will be idle part of the time [Acharya et al. 1997]. On the other hand, the advance of the personal computers, with multi-core CPUs and GPUs, usually take care easily of the users' needs. Thus, there is a scenario with personal computers idle at part of the time and with scientific computers overloaded in specific moments, during the execution of scientific experiments.

Scientific experiments, in several cases, are organized as bag-of-tasks [Kwan and Jogesh 2010] or scientific workflows [Medeiros et al. 1996]. Bag-of-tasks are composed of a set of completely independent tasks, what is very different from workflows where a task needs to wait for the conclusion of another task. Both, typically, require huge computational power and a way to obtain it is the use of several personal computers, for example, desktop grids [Kondo et al. 2007, Anderson 2004] or volunteer computing [Anderson and Fedak 2006].

Volunteer computing (VC) projects take idle resources from donors: the tasks are sent to volunteers (in general using the Internet), and they send the results back to a server. This approach may provide a lot of computational power [Anderson 2004], but in scientific experiments, there are many issues which can turn this approach inefficient, such as long processing tasks [Dethier et al. 2008], great volumes of data to be transferred [Duan et al. 2012] or instability in volunteer computers [Dias et al. 2010]. The majority of these issues are related to the low-speed communication with donors across the Internet.

In order to solve these issues, some works proposed the use of peer-to-peer (P2P) concepts [Majithia et al. 2004, Zhao et al. 2009] in workflow execution using volunteer computing. This is a dynamic approach and it is able to deal, for example, with heterogeneous environments and faults. However, the majority of the research using P2P in this field aims to create huge and scalable networks. In this work we propose a different approach, dealing with small networks, composed of heterogeneous participants, and with bandwidth limited to speeds similar to the average speed on the Internet. However, it is worth highlighting that P2P communication makes more sense when the data are more frequently reused. Thus, this approach is preferable for experiments with dependent tasks, different from many approaches [Majithia et al. 2004, Gentzsch et al. 2013] in this area that work with bag-of-tasks (sets of independent tasks).

This paper is organized as follows. Section 2 presents related work. Section 3 describes the proposed solution. Section 4 contains the framework evaluation. The results are presented in Section 5. Section 6 presents the final remarks and future work.

2. Related Work

Some works are based only on a server (acting like the source of the tasks) and clients (the volunteers of the system), such as BOINC [Anderson 2004]. However, there are some systems that have others participants, they are not servers or volunteers, that are responsible for the communication among the participants.

Murata et al. 2008 present a work based on BOINC, where the volunteers are clustered in small groups (all of them are disjoint sets). The BOINC continues as the unique server, but each group is responsible to balance by itself. The same was proposed by Wen Dou et al. 2003, using instead of groups, random neighborhood relations.

The works that use intermediate participants in the communication can be divided into two main approaches: trees and directed acyclic graphs (DAG). In the tree-based approaches, the server is the root, the volunteers are, typically, the leaves, and the others are known as supertrees. The tasks travel from the root to the leaves, using decisions taken in each step. The decision could be based on probabilistic models [Kwan and Jogesh 2010] or other factors, such as reputation [Rius et al. 2012].

Mastroianni [Mastroianni et al. 2009] adopted a simpler strategy (from the scheduler algorithm point of view), but with more elements. The volunteers (workers) request tasks to the super-peers and the super-peers request them to the Data Cachers. The Data Cachers receive tasks from the Job Manager. The workers also have to request the input data directly from the Data Source, which receives data from the Job Manager. At last, workers send their results to Job Manager.

The works cited do not consider the data transfer costs because they deal with high-speed communication grids or small tasks. In the other cases, the transfer cost is very relevant.

3. Developed Solution

The development of this work was based on an extension [Digiampietri et al. 2014b] of the Workflow Management System (WfMS) called WOODSS (*A Spatial Decision Support System based on Workflows*) [Seffino et al. 1999], an open source system written in Java extended in this project to deal with P2P communication.

Based on the initial structure of the WfMS and on the extensions developed, a scheduling algorithm (Algorithm 1) that applies P2P techniques was specified and implemented. This algorithm runs on volunteer computers. The aim is to make the volunteers more proactive, i.e., they prepare themselves (downloading inputs) to the next tasks and send inputs to neighbors while they are processing a task. The main role of the server is to respond the volunteers' requests. Algorithm 1 works as follows. Each volunteer establishes a connection with the server in order to obtain information about others volunteers and tasks. After this, the volunteer requests a list of neighbors. With this information, the volunteer will send and receive data. Then the volunteer requests a list of "schematic objects" which are a simplified representation of the workflow that contains only tasks' relationships and inputs' identifications.

```
Connect to the server;
objs ← Request schematic objects;
Priority queue q ← Create priority queue with objs;
while q is not empty do
    Object obj ← Choose best task(q);
    Download inputs not downloaded yet of obj;
    Alert serve about the execution start of obj;
    if obj is able to execute then
        Task a ← Download task(obj);
        Start background thread;
        Execute a;
        Stop background thread;
        Send result;
        Mark result as a parameter not confirmed;
```

Algorithm 1: Scheduling algorithm that apply P2P techniques – Working in the volunteer computers

Then the algorithm decides what task will be processed and what task will be downloaded, with the criteria defined in the rules of the priority queue, based on: Select first tasks that do not depend on other not yet executed ones; the number of parameters; and the number of parameters already downloaded. When a volunteer chooses an input he asks the neighbors if they have it. If none of them have the desired input, it will be downloaded entirely from the server, in the other case, the data is downloaded from the server and the neighbors. Once there is a task ready, the processing is started and a background thread is also started. The background thread is similar to the process to choose a task, with the difference of do not download the task by itself, downloading only parameters of future tasks. This algorithm was designed specifically for small networks. For larger networks, a more complex algorithm with a more robust coordination strategy is necessary.

4. Evaluation

An infrastructure was implemented to measure the performance of the developed solution, which contains two test cases to evaluate three scheduling approaches. In order to do this, four computers were used. The computers' hardware and operating system specifications are presented in Table 1.

Table 1. Features of the computers used.

#	Processor	Clock	Memory	O.S.
1	Intel®Core i5-3230M	4x3.20 GHz	8 GB	Ubuntu 14.04
2	Intel®Core i3-350M	4x2.26 GHz	4 GB	Ubuntu 14.04
3	Intel®Pentium T3400	2x2.26 GHz	3 GB	Ubuntu 12.04
4	Intel®Core 2 Duo T5750	2x2.00 GHz	2 GB	Ubuntu 12.04

In order to simulate the real world connections, all bandwidths were limited via software. We used the average bandwidth connection, measured in Brazil (download at 2.4 Mbps) [Akamai 2013]. The upload bandwidth was considered as 10% of the download bandwidth.

All test cases were compared using two different systems (the traditional VC system and the proposed solution using P2P), based on the developed extension of WOODSS (with three different scheduling approaches). Both used the Computer 4 as the server and the others as volunteers. Also was tested the scenario with and without redundancy required, i.e., all results are confirmed for more than a volunteer to be accepted. The next subsection describes the two test cases used. Each case was performed three times and all presented results correspond to average values.

4.1. Test Case of Social Network Analysis

The first test case was planned to show the performance of the algorithm in a real area of knowledge. We chose a test case based on a workflow of social network analysis [Digiampietri et al. 2014a, Digiampietri et al. 2015]. This test case has approximately 20 high dependent tasks, alternating between light and heavy loads. The data size alternated too, between 20KB and 5MB.

4.2. Test Case of Toy Examples

We create another test case based on structures described by Bharathi et al. 2008. These examples allow a detailed analysis of the strengths and weaknesses of the proposed approach. Bharathi defined that there are four basic structures for workflows, and the combinations of these structures allow the creation of any complex workflows. The structures are: *Data Aggregation* (DA), *Data Distribution* (DD), *Data Redistribution* (DR), and *Pipeline* (P).

Each test case was limited to a single type of structure, with the same shape and the same number of tasks (four tasks DA and DD, seven tasks for DR and three for P), but independent from each other. The tests are composed of 10 structures (i.e., a workflow with ten copies of the same structure, but each copy was able to run independently from the others). The intention is to verify the impact when the volunteer has, or not, many tasks to be chosen.

The tasks from this case are all of them toy examples. All the tasks process the classic problem $3n + 1$to the interval from 1 to 5.10^8. Therefore, there will be a runtime variation in the execution of each individual task caused only by differences in the performance of the computers. Furthermore, the input and output data were defined with constant size (it is important to highlight the selected problem need only a number as input and a boolean as output, but, to serve the purposes of this evaluation, we introduced an input and output parameter with constant size).

5. Results

This section describes the results from each test case.

5.1. Test Case of Social Network Analysis

The results from the application of P2P technique in the Social Network Analysis test case were very positive. The developed solution (P2P) was faster than the others (up to 1,11 of speedup) and the time spent only transferring date was smaller than the traditional VC approach. Table 2 presents these results.

Table 2. Execution and transfer time from the test case of social network analysis

Execution		Run Time	Speedup	Transfer Time
Without Redundancy	VC	4:57:00	1.11	0:28:19
	P2P	4:27:45		0:19:48
With Redundancy	VC	8:33:30	1.04	1:05:28
	P2P	8:15:02		0:29:14

There are two main reasons to explain the improvement in the speedup. The first one is that the execution of the background thread did not increase runtime. It was observed that the volunteers have less variation in processing since the time between one task and the next was smaller than the traditional VC system. The second reason is the proactive behavior of the volunteers in preparing tasks to be processed before they are really necessary and, therefore, reducing the time expended exclusively with the inputs download. On average, the time spent with transfer corresponds to 9.5% (and 12.7% with redundancy) of the total execution time in the traditional volunteer computing system, and in the developed solution corresponds to 7.9% (and 5.9% with redundancy).

The volunteers get more parameters than they really need (Table 3). It happens because, when a volunteer is choosing the next parameter to be downloaded, it does not know yet if he will really execute the respective task. Between the moment of the download of the parameters and the execution, the task could be concluded by other volunteer or just could have enough volunteers processing the task.

Table 3. Data transferred by each volunteer in MBs on the social network analysis (*Download* - D.; *Upload* - U.)

Execution		Server		Volunt. 1		Volunt. 2		Volunt. 3	
		U.	D.	U.	D.	U.	D.	U.	D.
With Redundancy	VC	75.54	15.01	4.14	31.63	5.59	25.04	5.66	23.21
	P2P	115.74	17.92	6.30	38.29	6.74	37.70	5.00	38.82
Without Redundancy	VC	153.31	30.18	9.41	64.15	11.26	54.65	10.22	38.38
	P2P	99.39	31.33	7.12	23.71	14.76	42.62	10.17	29.85

5.2. Test Case of Toy Examples

In the cases with the toy examples structures (equals and independents), the results were more promising. This kind of test help to illustrate the behavior of the system imagining the execution of complex workflows.

The volunteer computing with P2P was faster than the traditional VC (Table 4). The improvements in execution time were significant, in both cases: with or without redundancy. The processing time of the two solutions proves that the P2P approach has a speed increase between 1.14 and 1.33 (and with redundancy between 1.22 and 1.33).

Table 4. Time of runtime/transfer of the test case with toy examples

Execution		Without Redundancy			With Redundancy		
		Run Time	Speedup	Transfer Time	Run Time	Speedup	Transfer Time
Data Aggregation	VC	3:09:50	1.33	0:57:55	7:17:01	1.23	1:18:55
	P2P	2:22:37		0:21:53	5:56:38		0:36:51
Data Distribution	VC	2:51:00	1.14	0:41:37	6:58:02	1.33	1:55:24
	P2P	2:29:13		0:20:04	5:13:22		0:37:58
Data Redistribution	VC	5:04:07	1.20	1:25:21	11:14:16	1.28	2:54:57
	P2P	4:12:43		0:31:49	8:44:02		1:01:52
Pipeline	VC	2:20:37	1.30	0:33:24	5:13:39	1.22	1:08:03
	P2P	1:48:28		0:15:09	4:15:22		0:28:53

The time spent exclusively with data transfers was reduced at least in the half in almost all cases. The volunteers (from the volunteer computing with P2P) expended most of the transfer time sending results, because the majority of the tasks did not spend extra time downloading parameters (they were downloaded in the background thread), but the upload of results is an activity without background threads (in order to optimize the total execution time).

Taking into consideration the transfer issues, the results were very positive, which is different from the ones achieved in the social network analysis cases. Table 5 shows the application of P2P obtained results better than the traditional volunteer computing, with or without redundancy. In some cases, the server did not download more data than in the traditional computing approach. It creates a situation in which the server downloaded more data than uploaded. Receiving more data than sending is typically considered a very positive thing in many real scenarios (for example, most of the Internet providers provide a much higher download rate than the upload one), favoring the proposed approach again.

The difference of this test case with the social network analysis one is due to the difference of output size of the tasks. In this case, the inputs and outputs have the same size, but in social network analysis case, the size of the first task input was bigger than the output of any task.

Table 5. Data transferred by each volunteer in MBs on test case with toy examples (*Download* - D.; *Upload* - U.)

Execution		Without Redundancy								With Redundancy							
		Server		Volunt. 1		Volunt. 2		Volunt. 3		Server		Volunt. 1		Volunt. 2		Volunt. 3	
		U.	D.	U.	D.	U.	D.	U.	D.	U.	D.	U.	D.	U.	D.	U.	D.
Data Aggregation	VC	120.68	82.38	37.27	60.17	25.66	30.57	20.74	34.10	243.28	167.59	75.72	113.37	49.73	70.50	43.52	66.35
	P2P	75.92	85.04	40.20	18.73	22.98	26.14	16.49	26.22	80.29	165.71	65.30	20.80	44.60	23.73	44.99	31.59
Data Distribution	VC	84.64	84.28	37.48	37.57	25.72	26.63	18.37	19.14	168.12	165.18	76.21	75.33	46.86	47.34	41.01	44.19
	P2P	33.97	83.37	38.37	11.16	25.41	13.00	18.31	8.66	35.77	164.07	69.75	10.32	46.21	11.90	42.00	11.75
Data Redistribution	VC	185.21	147.01	62.49	92.37	44.27	43.99	36.20	45.41	372.88	293.27	132.15	168.38	81.50	98.83	71.88	101.33
	P2P	71.02	143.72	63.14	19.98	42.48	24.51	34.86	27.28	86.46	288.25	127.12	28.60	80.16	23.84	70.74	31.54
Pipeline	VC	61.79	61.29	26.85	26.75	18.91	19.41	14.21	15.25	121.99	124.28	55.21	54.73	33.55	33.93	32.48	34.27
	P2P	28.13	59.52	30.32	8.36	14.46	8.57	14.35	10.51	26.21	123.21	58.20	7.97	35.74	8.24	26.36	9.40

6. Final Remarks and Future Work

In this work, we presented an alternative solution to the problem of executing scientific workflows in a distributed way, in small networks. It combines the use of volunteer computing with peer-to-peer techniques. The current approach is not designed to deal with big networks because the communication process is not scalable. The comparison of the proposed approach with local execution of tasks and the traditional volunteer computing

showed promising results. The developed solution was tested with two cases: the first is a real case of social network analysis and the second is a test case with toy examples (created considering the basic workflow structures). In both cases, the developed solution obtained positive results to reduce the total run time and the bandwidth needs of the server (except in one specific case). The results are more expressive when considering the scenarios which require redundancy of task execution. Among the main contributions, we highlight the algorithm presented in Section 3. It corresponds to improvements in the execution time of scientific experiments for small networks and with limited bandwidth.Another contribution of this approach is that a volunteer can help the execution of the workflows not only running tasks but also helping in the sharing of input data.

As future work, we intend to improve the volunteer communication. We also intend to make the scheduling algorithm able to work with massive quantities of volunteers. Finally, we will improve the algorithm with a robust fault tolerance mechanism.

Acknowledgment

The work presented in this paper was partially funded by CAPES, CNPq, and FAPESP.

References

Acharya, A., Edjlali, G., and Saltz, J. (1997). The utility of exploiting idle workstations for parallel computation. *ACM SIGMETRICS Performance Evaluation Review*, 25(1):225–234.

Akamai (2013). The state of the internet. Technical Report Vol 6, Num 2, Akamai Faster Forward.

Anderson, D. (2004). BOINC: a system for public-resource computing and storage. In *Fifth IEEE/ACM International Workshop on Grid Computing*, pages 4–10.

Anderson, D. P. and Fedak, G. (2006). The Computational and Storage Potential of Volunteer Computing. In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, volume 1, pages 73–80.

Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.-H., and Vahi, K. (2008). Characterization of scientific workflows. In *2008 Third Workshop on Workflows in Support of Large-Scale Science*, pages 1–10. IEEE.

Dethier, G., Briquet, C., Marchot, P., and de Marneffe, P.-A. (2008). LBG-SQUARE Fault-Tolerant, Locality-Aware Co-Allocation in P2P Grids. In *2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 252–258. IEEE.

Dias, J., Ogasawara, E., de Oliveira, D., Pacitti, E., and Mattoso, M. (2010). Improving Many-Task computing in scientific workflows using P2P techniques. In *2010 3rd Workshop on Many-Task Computing on Grids and Supercomputers*, pages 1–10. IEEE.

Digiampietri, L., Alves, C., Trucolo, C., and Oliveira, R. (2014a). Análise da rede dos doutores que atuam em computação no brasil. In *CSBC-BraSNAM 2014*.

Digiampietri, L., de Jsus Prez-Alczar, J., Santiago, C., Oliveira, G., Khouri, A., and Arajo, J. (2014b). A framework for automatic composition of scientific experiments: Achievements, lessons learned and challenges. In *VIII Brazilian e-Science Workshop*.

Digiampietri, L. A., Maruyama, W. T., Santiago, C. R. N., and da Silva Lima, J. J. (2015). Um Sistema de Predio de Relacionamentos em Redes Sociais. In *Simpsio Brasileiro de Sistemas de Informao (SBSI 2015)*, pages 139–146.

Duan, K., Padget, J., Kim, H. A., and Hosobe, H. (2012). Composition of engineering web services with universal distributed data-flows framework based on ROA. In *Proceedings of the Third International Workshop on RESTful Design - WS-REST '12*, page 41, New York, New York, USA. ACM Press.

Gentzsch, W., Grandinetti, L., Joubert, G., Ricci, L., Baraglia, R., Ghafarian, T., Deldari, H., Javadi, B., Yaghmaee, M. H., and Buyya, R. (2013). CycloidGrid: A proximity-aware P2P-based resource discovery architecture in volunteer computing systems. *Future Generation Computer Systems*, 29(6):1583–1595.

Kondo, D., Fedak, G., Cappello, F., Chien, A. A., and Casanova, H. (2007). Characterizing resource availability in enterprise desktop grids. *Future Generation Computer Systems*, 23(7):888–903.

Kwan, S. K. and Jogesh, K. M. (2010). Bag-of-Tasks applications scheduling on volunteer desktop grids with adaptive information dissemination. In *IEEE Local Computer Network Conference*, pages 544–551. IEEE.

Majithia, S., Shields, M., Taylor, I., and Wang, I. (2004). Triana: a graphical Web service composition and execution toolkit. In *Proceedings. IEEE International Conference on Web Services, 2004.*, pages 514–521. IEEE.

Mastroianni, C., Cozza, P., Talia, D., Kelley, I., and Taylor, I. (2009). A scalable super-peer approach for public scientific computation. *Future Generation Computer Systems*, 25(3):213–223.

Medeiros, J. W., Weske, M., Vossen, G., and Bauzer, C. (1996). Scientific workflow systems. *NSF Workshop on Workflow and Process Automation: State-of-the-art and Future Directions*.

Murata, Y., Inaba, T., Takizawa, H., and Kobayashi, H. (2008). Implementation and evaluation of a distributed and cooperative load-balancing mechanism for dependable volunteer computing. In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pages 316–325. IEEE.

Rius, J., Estrada, S., Cores, F., and Solsona, F. (2012). Incentive mechanism for scheduling jobs in a peer-to-peer computing system. *Simulation Modelling Practice and Theory*, 25:36–55.

Seffino, L., Medeiros, C., Rocha, J., and Yi, B. (1999). WOODSS - A Spatial Decision Support System based on Workflows. *Decision Support Systems*, 27(1-2):105–123.

Wen Dou, Yan Jia, Huai Ming Wang, Wen Qiang Song, and Peng Zou (2003). A P2P approach for global computing. In *Proceedings International Parallel and Distributed Processing Symposium*, page 6. IEEE Comput. Soc.

Zhao, Z., Yang, F., and Xu, Y. (2009). PPVC: A P2P volunteer computing system. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 51–55. IEEE.