Technology for Integrating Idle Computing Cluster Resources into Volunteer Computing Projects

A. P. Afanasiev[2], I. V. Bychkov[1], M. O. Manzyuk[1], M. A. Posypkin[2], A. A. Semenov[1], O. S. Zaikin[1]

[1]Institute for System Dynamics and Control Theory SB RAS, Irkutsk, Russia

[2]A. A. Kharkevich Institute for Information Transmission Problems RAS, Moscow, Russia

Keywords: volunteer computing, BOINC, computing cluster, job scheduler, MPI, SAT@home, OPTIMA@home.

Abstract. In this paper we propose the technology for integrating idle computing cluster resources into volunteer computing projects. The main principles of this technology are the following: only standard cluster user rights and only idle computing cluster resources (i.e. the resources that are not employed by other cluster users) are used. We describe the CluBORun tool implementing this technology for BOINC-based volunteer computing projects. The CluBORun tool was successfully applied to boost the performance of volunteer computing projects SAT@home and OPTIMA@home.

Introduction

Nowadays there are many problems, for solving which it is necessary to involve large-scale computational resources for months or even years. Such tasks arise for example during the processing of astronomical data, solving problems from cryptography and cryptanalysis, modelling the synthesis of new chemicals, etc. Unfortunately, the problems of such kind usually conflict with standard administrative rules on the use of modern computing clusters: the chances of obtaining a significant portion of computing cluster resources for solving one particular problem for a prolonged time are very small (unless the computing cluster is designed specifically for solving such problems). That is why in the recent years the concept of volunteer computing is becoming more and more popular for solving the problems of the mentioned type [1]. According to this concept, the private persons called volunteers provide the computational resources for solving large-scale problems. Usually, volunteers offer the resources of their own PCs. A number of important from the practical point of view or challenging problems are being solved today using volunteer computing projects. The most popular projects are SETI@home, Einstein@home, GIMPS, Folding@home, Rosetta@home.

Meanwhile, it is important to note such a significant issue as systematic underutilization of many computing clusters. At some moments the computing cluster can be loaded up to 90%, at some – half of its resources may be idle. This problem is important because even mostly idle computing cluster consumes a lot of electricity. The monitoring of load performed for several clusters showed that an average load per year usually varies between 50% and 70%.

Thus, on one hand we have the problems that can make use of any additional computational resources, on the other hand – underutilization of computing clusters. As a result of the analysis of these two problems we propose the technology that integrates idle computational resources of computing clusters into volunteer computing projects.

Let us give a brief outline of the article. In the next section, we describe the basic ideas of volunteer computing. Then we illustrate the work of such projects on the example of the SAT@home project, developed and maintained by us. After this, we describe the CluBORun system that utilizes the idle computational resources of computing clusters for the needs of volunteer computing projects. Further, we display the results of computational experiments, in which CluBORun was used to boost the performance of volunteer computing projects SAT@home and OPTIMA@home.

Basic elements of the concept of volunteer computing (on the example of the SAT@home project)

As we noted above, in volunteer computing the computational resources are provided by volunteers. Generally it is best to apply volunteer computing to solving problems that can be decomposed into subproblems of lower dimension and processed independently (the so-called embarrassing parallelism). We will refer to the subproblems obtained during the decomposition of the original problem as elementary subproblems. The decomposition should be performed in such a way that each elementary subproblem can be solved even by a weak computing node (in volunteer computing it is usually called host) relatively fast – from several minutes to several hours.

There are special instruments that are used to establish the volunteer computing projects. The most widely used tool for this purpose is the BOINC platform [2]. Below we outline the basic principles of the BOINC computing.

1. The project is managed by the BOINC server that has to work 24/7. The server functionality is the following: it provides client applications for different platforms, generates batches of elementary subproblems and sends them to project participants (volunteers), collects and processes the results obtained.
2. The resources of the project consist of volunteer's hosts. Each volunteer's host gets the client application and batches of elementary subproblems from the server. The results obtained are sent back to the project server. To solve each elementary subproblem only idle computational resources of the host are utilized. The resources of the host are managed by a special program called BOINC manager. Its settings are quite flexible since it should utilize only idle host resources. One can set the BOINC manager in such a way that the solving of corresponding problems is performed only in time intervals when the user does not use the host for solving any other problems. If the user needs the resources, the BOINC manager must interrupt all computations and later return to the point where the process was interrupted. That is why it is important to be able to save intermediate results of the computations.
3. On one host it is possible to work with several projects – the BOINC manager divides idle host resources according to the chosen priorities.

Let us illustrate these principles on the example of the BOINC-based volunteer computing project SAT@home [3], that was developed and is being maintained by us. The project was launched on September 29, 2011. On February 7, 2012 SAT@home was added to the official list of BOINC projects[1] with alpha status. Recently its status was improved to beta. SAT@home is aimed at solving hard combinatorial problems that can be effectively reduced to Boolean satisfiability problem (SAT) [4]. Such problems can be found in many areas, for example, verification, cryptography, combinatorics and bioinformatics. SAT is usually considered as the problem of search for solution of a Boolean equation in the form of CNF=1, where CNF is a conjunctive normal form. If the solution exists, then it is called the satisfying assignment. Otherwise the CNF is called unsatisfiable. All known SAT solving algorithms are exponential in the worst case (SAT is NP-hard). Nevertheless, modern SAT solvers successfully cope with different instances from problem areas mentioned above. The improvement of the effectiveness of SAT solving algorithms, including the development of algorithms that are able to work in parallel and distributed computing environments is a very important direction of research.

Let us briefly describe basic features of the SAT@home project. The SAT@home server uses a number of standard BOINC daemons responsible for sending and processing tasks (transitioner, feeder, scheduler, etc.). Such daemons as work generator, validator and assimilator were implemented taking into account the specificity of the project. The work generator decomposes the original SAT problem to subproblems based on the previously found decomposition parameters. In SAT@home we use the method for finding such parameters that was proposed in [5]. The work generator creates 2 copies of each task in accordance with the concept of redundant calculations

[1] http://boinc.berkeley.edu/projects.php

used in BOINC. The validator checks the correctness of the results, and the assimilator processes correct results. When the assimilator finds satisfying assignment in obtained results, it checks this assignment. If the assignment is correct, then the original problem is marked as solved and the generation of tasks for this problem stops. The SAT@home client application is based on the SAT solver MiniSat 2.2 [6], which was slightly modified to use less RAM.

The characteristics of the SAT@home project as of 10 of December 2014 are the following (according to BOINCstats[2]):

- 3246 active hosts (in volunteer computing the host is considered to be active if it has sent at least one result in the last 30 days) about 80% of them use Microsoft Windows OS;
- 1440 active users (active user is a user that has at least one active host);
- versions of the client application for platforms: Windows x86, Windows x86-64, Linux x86, Linux x86-64;
- average real performance: 4,5 teraflops, maximal performance: 7,9 teraflops (achieved during the competition held by BOINCstats in October, 2014).

The dynamics of the real performance of SAT@home can be seen at the SAT@home performance page.

The CluBORun Tool for Integrating Idle Computing Cluster Resources into Volunteer Computing Projects

As we noted above, the development of the system that would integrate idle computing cluster resources into volunteer computing projects is highly relevant. It is important to note, that such system should manage the cluster resources in the same way the BOINC manager does it for the PC resources, i.e. to not interfere with processes launched by other users of the cluster. In addition, a significant limitation consists in the fact that such system should rely on standard user rights, and should not need any assistance from the cluster administrator. Below we will describe the CluBORun (Cluster for BOINC Run) tool developed according to these conditions.

The final purpose of the CluBORun is to use the BOINC manager to launch client applications of BOINC projects utilizing the computing cluster computational resources. The use of the BOINC manager on a particular computing node makes it possible to perform computations for several different projects simultaneously. It should be noted, that usually all the tasks within the computing cluster are launched as MPI applications, while the BOINC manager is not designed as one. At the same time, the BOINC manager has its own functions for grabbing and freeing available resources of the computing node, on which it is launched (usually, the volunteer computing project host). To launch the BOINC manager correctly, using standard tools for communicating with the cluster queue, we developed a special MPI-program start_boinc that is launched on the cluster according to standard rules (via the mpirun).

Let us describe how the start_boinc works on an arbitrary cluster node. Suppose that the considered node has n processor cores (hereinafter by these we mean only CPU cores). Since the start_boinc program is a standard MPI application, it means that MPI environment tools allow it to use n MPI processes. On one MPI process, to which we will refer as master process, it starts the BOINC manager. Remaining n-1 processes are not employed by the start_boinc program. We will refer to these n-1 processes as sleeping processes. After having been launched the BOINC manager "sees" all the available resources of the node: both those that correspond to the sleeping MPI processes and the one it is launched on. After this the BOINC manager employs all the available node resources as if it were the idle resources of the volunteer computing host. If it is necessary to interrupt the computations, the start_boinc program sends a corresponding command to the BOINC manager. The BOINC manager processes the command and performs the required actions: saves intermediate computations results, sends results for processed subproblems back to project server and stops.

[2] http://boincstats.com/en/stats/123/project/detail/

CluBORun minimizes the impact on the work of other users by means of automatic system that distributes the computing cluster idle resources (to which we will refer as distribution system). This system interconnects with the computing cluster job scheduler using standard user rights in the following manner: it periodically monitors the current cluster queue and takes only the idle resources as a usual user would. In the process of monitoring, one has to take several features into account. For example, even if there are a lot of idle resources, the distribution system should not take it for a long time since in this case it is possible that some users will be "scared" when they see that the cluster is fully loaded. So if the distribution system takes idle resources only for a short time and corresponding information is displayed in the queue, then such situation can be avoided – the user puts its job into the queue knowing that soon the resources will be freed. If the distribution system puts the jobs for several hours and there appear the jobs from other users in the cluster queue, then the distribution system should be able to withdraw its jobs, saving the intermediate results and freeing the computational resources. The distribution system in CluBORun is organized in the form of two scripts – one monitors the cluster queue and another launches the start_boinc program on the idle nodes.

Structurally the CluBORun tool consists of the following components.

1. The folders corresponding to MPI tasks start_boinc that should be solved on the cluster (each of the tasks launches one or several instances of the BOINC manager).
2. Indicator files corresponding to MPI tasks start_boinc: if it is possible to launch a particular MPI task then a start flag is created and if it is necessary to interrupt it then the stop flag is created.
3. Text file all_tasks.txt – the list of MPI tasks that need to be launched. For each task in this file the following information is written: the start flag file name, the stop flag file name and the path to the folder corresponding to this task.
4. The script catch_node.sh which monitors the cluster queue and creates the start/stop flags to communicate with the start_boinc.sh script.
5. The script start_boinc.sh that starts and stops MPI tasks start_boinc.

Scheme of launching BOINC computations on computing cluster with the help of CluBOrun tool is shown in the Figure 1.

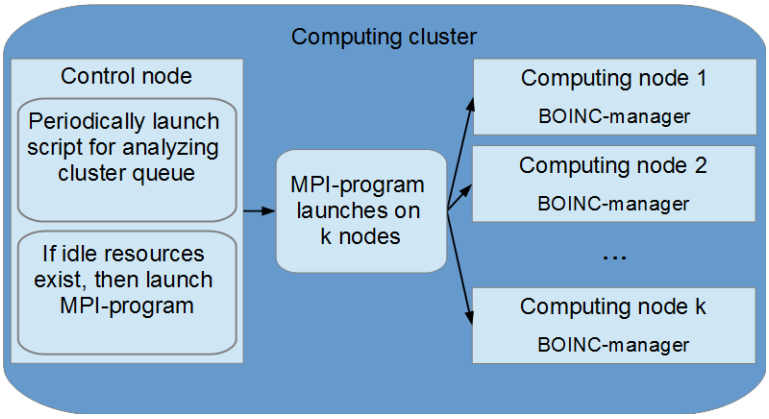

Fig. 1. Scheme of launching BOINC computations by CluBORun

Further, let us briefly describe how the CluBORun tool works. It performs all actions under the name of the user with standard rights, i.e. which can put its MPI tasks to the cluster queue. Using the standard cron scheduler the rule is created: to run the catch_node.sh script regularly with short time interval (for example, every minute). This script analyzes the cluster queue using standard commands available to any cluster user. In particular, the catch_node.sh script requests the list of all launched tasks and the list of tasks in queue (if any). Analyzing this information the script, based on the number of idle nodes and the composition of cluster queue, determines how many BOINC tasks are already running and how many tasks can be started. The result of this analysis is the list of MPI tasks start_boinc and the list of cluster nodes on which these tasks can be started at the moment. All these tasks are assigned the start flag. Similarly if the catch_node.sh determines that in the queue

there are tasks from other users that await their turn then it interrupts several (possibly all) running start_boinc tasks. The stop flag is put in the folders of the corresponding problems.

The start_boinc.sh periodically scans the folders of MPI tasks start_boinc. If in some folder there is a start flag then it starts the corresponding task on an available cluster node in a usual manner (via the mpirun). If start_boinc.sh finds in the folder of some problem that there is a stop flag then it interrupts the corresponding task. After receiving from the start_boinc.sh the command to interrupt the work all the master processes of start_boinc send similar commands to the BOINC managers launched within them. The latter then save the intermediate results and free the cluster resources.

Computational Experiments

The main technical issue that arises when one wants to connect a new cluster to some volunteer computing project using the CluBORun tool consists in the fact that different clusters use different job schedulers. Therefore, the algorithms described above have to be implemented individually for each scheduler. At the present moment the CluBORun tool is able to work with three cluster job schedulers: SUPPZ[3], Cleo[4] and SLURM[5]. In this list the SUPPZ and the Cleo systems are developed in Russia and the SLURM system is an international product.

In order to increase the performance of the SAT@home project CluBORun was launched in December 2013 on the MVS-100k[6] cluster. This cluster uses the SUPPZ job scheduler. The CluBORun tool made it possible to employ a significant portion of idle computing nodes. So, from 5 to 12 March, 2014, on average 1824 cluster cores were idle and 512 of them were employed to perform computations for the SAT@home via CluBORun. At some periods of time the contribution of the MVS-100k cluster to the performance of the SAT@home project reached 40%. Using these resources it was possible to carry out the following experiments in the SAT@home.

- Solved 10 cryptanalysis instances for the A5/1 keystream generator [7]. In each instance only the information about first 114 bits of keystream (one burst) was used.
- Solved 3 weakened cryptanalysis instances for the Bivium cipher. The instances were weakened by setting the known values to the variables corresponding to the last 10 of 177 bits of the registers initial values. In this series of experiments the keystream fragment of length 200 bits was used.
- Found 17 new pairs of orthogonal diagonal Latin squares of order 10 (previously only 3 such pairs were published in [8]).

Since the October 2014 the CluBORun tool is used to increase the performance of the volunteer computing project OPTIMA@home[7] (it too uses the idle resources of the MVS-100k cluster). This project was developed specifically to implement modern numerical optimization algorithms. At the present moment the OPTIMA@home project is used to solve several minimization problems for functions that arise in molecular dynamics when modeling the properties of various materials. Using the CluBORun tool it was possible to carry out the research of the precision of parameters identification for multiatom Tersoff potential for the crystalline silicon.

Conclusions

In the present paper we described the technology that can be used to integrate idle computing cluster resources into volunteer computing projects and the implementation of this technology in the form of the CluBORun tool. In the nearest future, we plan to extend CluBORun with an ability to work with other with PBS TORQUE[8]. The CluBORun source code can be found online[9].

[3] http://suppz.jscc.ru/
[4] http://parcon.parallel.ru/cleo.html
[5] http://slurm.schedmd.com/
[6] http://www.jscc.ru/hard/mvs100k.shtml
[7] http://boinc.isa.ru/dcsdg/
[8] http://www.adaptivecomputing.com/products/open-source/torque/
[9] https://github.com/Nauchnik/CluBORun

Acknowledgements

Authors thank Stepan Kochemazov for numerous valuable comments that allowed us to significantly improve the quality of the paper. This work was partly supported by Russian Foundation for Basic Research (grants № 13-07-00291-a, 14-07-00403-a, 15-07-07891-a), the Council at the President of the Russian Federation for the State Maintenance of the Leading Scientific Schools (project NSh–5007.2014.9) and the Special Program of Russian Academy of Sciences № 14.

References

[1] Durrani M.N. & Shamsi J.A.: Volunteer computing: requirements, challenges, and solutions. Journal of Network and Computer Applications. Vol. 39, pp. 369-380. 2014.

[2] Anderson D.P. BOINC: A System for Public-Resource Computing and Storage. In: Buyya, R. (ed.) GRID. IEEE Computer Society. pp. 4-10. 2004.

[3] Posypkin M.A., Semenov A.A. & Zaikin O.S. Using BOINC desktop grid to solve large scale SAT problems. Computer Science. Vol. 13, no 1. pp. 25-34. 2012.

[4] Biere A., Heule M., van Maaren & H., Walsh T. (eds.). Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications. Vol. 185. IOS Press. 2009.

[5] Semenov A.A. & Zaikin O.S. On estimating total time to solve SAT in distributed computing environments: Application to the SAT@home project. arXiv:1308.0761 [cs.AI].

[6] Een N. & Sorensson N. An Extensible SAT-solver. Lecture Notes in Computer Science. Vol. 2919. pp. 502–518. 2003.

[7] Semenov A.A., Zaikin O.S & Otpuschennikov I.V. Using Volunteer Computing for Mounting SAT-based Cryptographic Attacks. arXiv:1411.5433 [cs.DC].

[8] Brown J.W. et al. Completion of the Spectrum of Orthogonal Diagonal Latin Squares. Lecture notes in pure and applied mathematics. Vol. 139. pp. 43–49. 1992.

[9] Semenov A.A., Zaikin O.S., Bespalov D.V. & Posypkin M.A. Parallel Logical Cryptanalysis of the Generator A5/1 in BNB-Grid System // Lecture Notes in Computer Science. Vol. 6873. pp 473-483. 2011.

[10] Farkas Z., Kacsuk P., Balaton Z. & Gombas G. Interoperability of BOINC and EGEE // Future Generation Computer Systems. Vol. 26, no. 8. pp. 1092-1103. 2010.