Computer Science • 13 (1) 2012 http://dx.doi.org/10.7494/csci.2012.13.1.25

Mikhail Posypkin
Alexander Semenov
Oleg Zaikin

USING BOINC DESKTOP GRID TO SOLVE LARGE SCALE SAT PROBLEMS

Abstract *Many practically important combinatorial problems can be efficiently reduced to a problem of Boolean satisfiability (SAT). Therefore, the implementation of distributed algorithms for solving SAT problems is of great importance. In this article we describe a technology for organizing desktop grid, which is meant for solving SAT problems. This technology was implemented in the form of a volunteer computing project SAT@home based on a popular BOINC platform.*

Keywords desktop grid, Boolean satisfiability problem (SAT), volunteer computing, BOINC

1. Introduction

Many practically important problems (model checking, problems of discrete systems control, information security, etc) can be considered as Boolean satisfiability problems (SAT) [1, 2]. SAT problems are hard and require lots of computational resources. That is why using parallel and distributed computing to solve SAT problems is quite popular nowadays. In this paper we further develop a technique for parallel solving of SAT problems which was proposed in [3, 4, 5]. This approach relies on coarse-grained work distribution and is therefore suitable for desktop grid (DG) systems. Organizing the solving of SAT problems in a DG is challenging. There are currently few studies in this area. Among recent works on SAT solving in DG the paper [6] should be mentioned. In that paper a distributed SAT solver for peer-to-peer DG was proposed. We propose another promising approach based on volunteer DG with client-server architecture. This approach was implemented using BOINC [7] open source platform.

2. Volunteer computing

Volunteer (or desktop grid) computing is a relatively new trend in distributed computing. Unlike service grids or clouds the computational resources are provided by so-called "volunteers". Volunteers are PC users who agree to donate their computing resources for solving scientific problems. The desktop grid technology ensures that only free resources (when the PC is not used for other purposes) are used, so the participation in a volunteer computing project does not interfere with the user's main activity. A comprehensive overview of the desktop grid software can be found in [8]. Below we outline the most popular DG systems.

XtremWeb[9] is an open source software used to build a lightweight Desktop Grid by gathering the unused resources of desktop computers (CPU, storage, network). The XtremWeb architecture is composed of Servers, Workers and Clients. A server (or a group of Servers) host centralized services such as scheduler and result collector. Workers are installed by resource owners on their PCs to donate their computing resources. Clients are installed by resource users on their PCs and permit users to install applications and use distributed resources, submit jobs and retrieve results. Jobs submitted by Client are registered on the Server and scheduled for Workers. Within the XtremWeb architecture any Worker can be a Client.

The OurGrid[10] middleware makes it possible to create the so called peer-to-peer computational grids. In the peer-to-peer grids created by OurGrid, computing and storage resources are provided by a whole community of grid participants. The resources are shared in such a way that those participants who have contributed the most get the most out of the grid resources whenever they need them. The software is written in Java. OurGrid is an open-source software distributed under GPL license.

Condor[11] is a high-throughput distributed batch computing system developed at the University of Wisconsin-Madison, USA. Condor can be used to manage a cluster

of dedicated computing nodes. Condor can be configured to only use desktop machines where the keyboard and mouse are idle. Should the system detect that a machine is no longer available (i.e if a key press detected) Condor is able to transparently produce a checkpoint and migrate a job to a different machine. Condor does not require a shared file system across machines — if no shared file system is available, it can transfer the job's data files on behalf of the user, or be able to transparently redirect all the job's I/O requests back to the submit machine. As a result, Condor can be used to seamlessly combine all of an organisation's computational power into one resource.

BOINC (Berkeley Open Infrastructure for Network Computing [7]) is an open source platform for Desktop Grid computing. It is being developed at U.C. Berkeley Spaces Sciences Laboratory by the group that developed and continues to operate the SETI@home project. The BOINC software consists of two parts: server software that is used to create volunteer computing projects and client software. Each project operates its own server and provides its own web site. Volunteers install and run client software on their computers. The client software is available for all major platforms, including Windows, Linux,and Mac OS X. Volunteers can donate free computing power from their PCs by connecting installed BOINC clients to different BOINC projects. In the last decade BOINC projects helped to obtain several remarkable scientific results e.g. new pulsars discovered by Einstein@home project.

From this big diversity of desktop grid software we selected BOINC as the most reliable platform for volunteer computing that has proved the ability to collect and maintain huge distributed projects. To the best of our knowledge BOINC has not been used yet to solve SAT problems but many BOINC projects demonstrated remarkable efficiency when applied to various combinatorial problems.

3. Parallelization of SAT problems encoding some combinatorial problems

Boolean satisfiability problem (SAT [1, 2]) is to find a satisfying assignment for a Boolean formula represented by a Conjunctive Normal Form (CNF). SAT is a NP-hard problem, i.e. it cannot be solved by any known polynomial time algorithm. However, this problem is extremely important for various practical applications: verification problems in microelectronics, discrete optimization, cryptography, analysis of discrete automaton models, etc. Many of the problems from these areas can be efficiently (in polynomial time) reduced to SAT. Over the past 10 years, an interest in the construction of computational algorithms for solving SAT problems has significantly increased. Since 2002, specialized SAT solver competitions are regularly held (see [2]).

The vast majority of efficient sequential SAT solvers are based on non-chronological version of algorithm DPLL [12, 13]. Parallel SAT solvers started to massively appear quite recently, despite the fact that the first theoretical works on the

parallelization of algorithm DPLL had been published already in the 1990s [14]. Parallel SAT solver competitions have been held regularly since 2008 [2]. Many modern parallel SAT solvers ([15, 16, 17], etc.) involve intensive exchanges of Boolean constraints (so-called "conflict clauses") accumulated in parallel on different computing nodes. That is why the use of such solvers in grids is quite problematic. Nevertheless there are some examples of distributed SAT solvers. The paper [6] describes a distributed SAT solver that uses peer-to-peer protocol to exchange constraints among DG nodes. To the best of our knowledge there are no published results on the application of client-server desktop grids to SAT problems.

In our paper we propose an approach that is designed for volunteer computing and implies no data exchange among computing nodes. The approach is based on the coarse-grained parallelization of a SAT problem. The original problem is decomposed by assigning values to a set of selected variables. Since all variables are binary we obtain 2^n subproblems for n selected variables. Then the resulting subproblems are processed independently on different nodes of a distributed system. The key issue affecting the efficiency of the proposed approach is the proper selection of variables for assignment. For this purpose a meticulous research of the original combinatorial problem is performed. This research is carried out in the preprocessing stage and its result is a list of tasks. In particular for the original CNF we construct some family of subsets of the set of its Boolean variables. Each subset is associated with a value of a special predictive function. The argument of this function is a random sample of assignments of variables from the subset. The value of a predictive function gives an estimation of the total time for solving the original SAT-problem in a distributed environment. We perform the optimization of the predictive function to obtain the subset with minimal prediction value. This subset we call the decomposition set and we use it to construct the list of tasks. Note that when searching for the decomposition set it makes sense to analyze in detail the features of the original problem and use this information to improve the efficiency of predictive function optimization.

In [3, 4, 5] this approach was applied to the cryptanalysis of some keystream generators. In most of these generators so-called linear feedback shift registers (LFSRs [18]) are used as primitives. Including to decomposition set a variables which encode whole initial state of some LFSR greatly simplifies tasks in a list. In [5] it was shown that the use of this parallelization technique for the solving of the SAT problems encoding cryptanalysis problems in some cases can lead to superlinear speedup.

The preprocessing stage may require the resources of a parallel computer (usually the use of low performance cluster is sufficient for this purpose). However, the total computational cost of this step is significantly less than that would later be recognized for processing the constructed list of tasks. As a result of processing of this list we obtain the solution of the original problem. The processing of the list is performed by independent of each other's hosts of DG (usually individual PCs). To control the processing of the list (dispatch tasks, receive and analyze the results) a dedicated server is used.

In 2009 this approach was implemented in a distributed environment (comprising of several supercomputers) and applied to the cryptanalysis of a widely known keystream generator A5/1. Corresponding results can be found in [4].

It should be specifically noted that we consider cryptanalysis problems only as hard tests. We believe that the successful testing of computing technology on cryptographic tests means principal applicability of this technology for solving practically important complex combinatorial problems (in the form of SAT problems) that are not artificially designed to be hard, for example: discrete optimization problems (i.e. QAP [19]), the search for some interesting combinatorial structures (i.e. mutually orthogonal latin squares [20]), bioinformatics [21], etc.

The above results and considerations stimulated our research towards the construction of the volunteer computing project for solving SAT problems.

4. Volunteer computing project SAT@home

We created a special BOINC project SAT@home [22] aimed at solving various SAT problems. This project was launched in September 29, 2011 and now has nearly 3000 volunteer PCs connected. The project was created with the help of SZTAKI Desktop Grid package [23] which is a featured BOINC distribution. Both server and client parts of a distributed SAT solver were implemented using DC-API library [24]. The server part is responsible for creating tasks in the project database as well as for processing results collected from client PCs. Sending tasks to the client PCs and collecting results is performed by standard BOINC daemons (see Fig. 1).

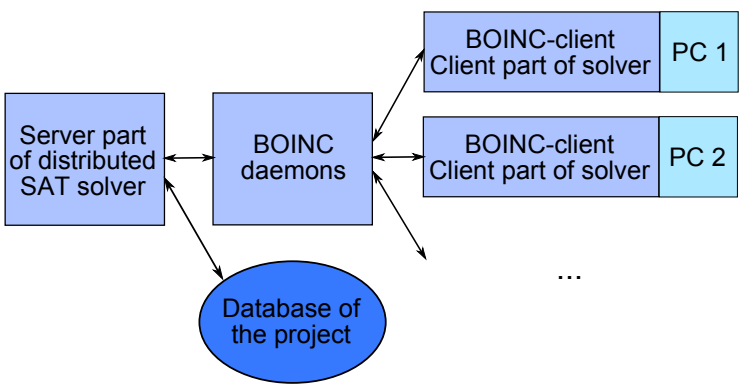

Figure 1. The scheme of the distributed solver in SAT@home

Client part is based on publicly available SAT solver minisat 1.14.1 [25] modified to take into account the peculiarities of CNFs encoding the original problems. The client part is executed on volunteers' PCs

The scheme of solving SAT problems in SAT@home is shown in Figure 2. For a particular SAT problem (CNF in DIMACS format [2]) we find "good" decomposition

parameters with the help of the predictive function technique described above. The parameters include: a type of SAT solver, a variables selection method and a number of subproblems. We use ISDCT RAS cluster Blackford [26] to find the decomposition parameters since these computations involve intensive interprocessor exchanges. This step takes quite a little amount of time (about several hours). The obtained parameters are used by the server part of the SAT@home project for decomposing the original problem into a number of independent subproblems. These subproblems are then submitted as new BOINC workunits (tasks) in the project database.

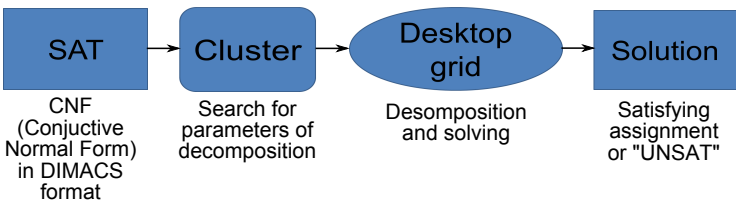

Figure 2. The scheme of solving SAT problems in SAT@home

On January 26, 2012 SAT@home had the following characteristics:

- 1002 participants;
- 2891 PCs, in total 11281 processor cores, 78% with Windows OS
- client parts of the application for windows x86, linux x86, linux x64;
- average real performance 1.5 TFLOPs, peak performance 4.3 TFLOPs.

Figure 3 shows the dynamics of the number of PCs connected to the project from September 29 to January 26 2011. Each column displays the total number of PCs connected to the project from its start to a specific date. It is easy to see that the number of PCs has significantly increased from October 10. The reason is that on this day the project was added to the statistical site Free-DC [27] (it contains information about BOINC projects), and that an export of statistics was enabled, allowing other statistical sites to add the project to their lists. Availability of information about the project on major statistical sites is an important factor for attracting new users. In Figure 4 a dynamics of the real performance of the project in GFLOPs is shown. The increase in performance from November 24 to December 1 is due to the fact that the site BOINCStats ran a competition on SAT@home so a lot of users from all over the world desired to participate in it.

Currently processing of one task on client PC takes about 3 hours. The main resource needed by application is CPU, at the same time only 20 Mb of RAM and 100 Mb of disk storage are used. The deadline for every task is 14 days. Every 2–5 minutes checkpointing is performed. It prevents the loss of intermediate results caused by extraordinary shutdown of a client's PC.

Nowadays the most effective way of cryptanalysis of this generator is the so-called "rainbow method" considered in [28]. The advantage of this method is its realtively low computational cost that allows to perform cryptanalysis on an ordinary PC (it

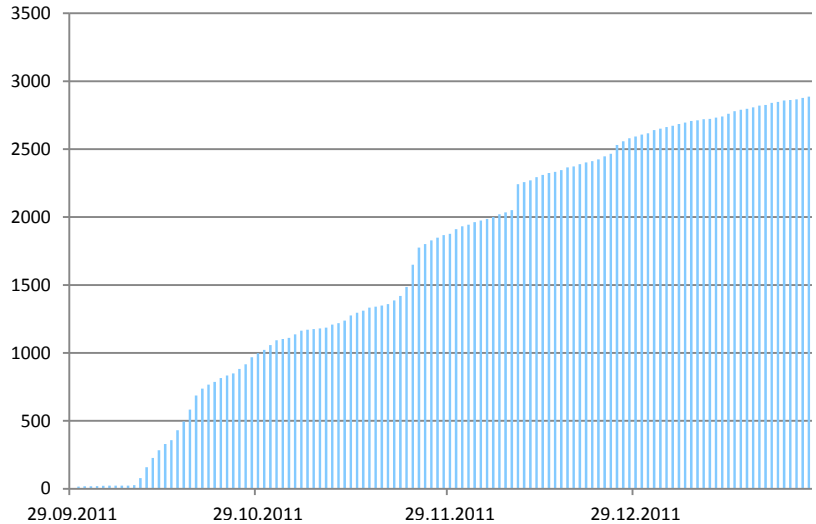

Figure 3. The dynamics of the number of PCs connected to SAT@home

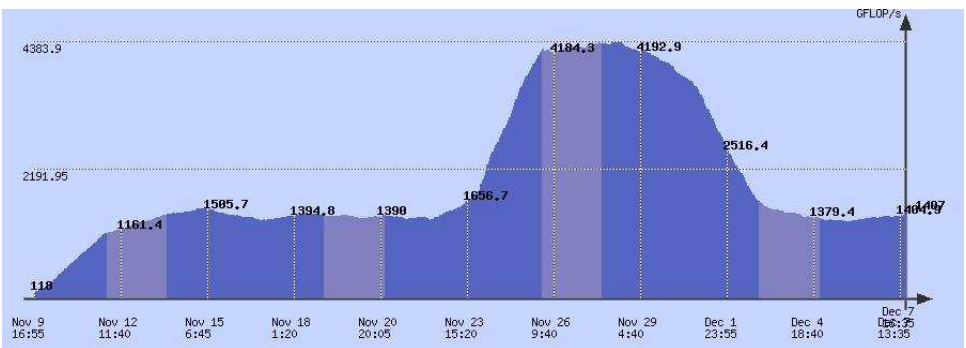

Figure 4. The dynamics of real performance of SAT@home (GFLOPs)

requires downloading 1.6 Tb of rainbow tables from [28]). However, the main disadvantage of this method is that rainbow tables do not cover the whole key space. These tables cover around 88 % of the key space. For testing the SAT@home project we selected 10 problems for which rainbow method does not give any result. At the moment 4 of 10 problems have been successfully solved in SAT@home [29]. On average, the solving of each test in the project took 10 days.

5. Conclusion

In this article we described general principles of coarse grained parallelization of SAT problems aimed at large-scale distributed systems. This approach was implemented

in SAT@home volunteer computing project. This project can be used for solving various computationally difficult combinatorial problems reduced to SAT problems. The project has successfully solved several instances of the inversion problem of the keystream generator A5/1 for which the well-known rainbow-method [28] did not yield any results. We hope that SAT@home will be useful to researchers who work with computationally difficult combinatorial problems in such areas as discrete optimization, cryptography, combinatorics [20], and bioinformatics [21].

Acknowledgements

This work was supported by Russian Foundation for Basic Research (Grants No. 11-07-00377-a and No. 10-07-00301-a) and European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement No. 261561 (DEGISCO). We would like to thank all the volunteers who participated in the project.

References

[1] Eds. Biere A., Heule M., van Maaren H., Walsh T.: *Handbook of Satisfiability.* IOS Press, 2009.

[2] *Up-to-date links for the SATisfiability Problem.* `http://www.satlive.org/`

[3] Zaikin O., Semenov A.: *Large-block parallelism technology in SAT problems.* Control Sciences, No. 1, 2008, pp. 43–50 (in Russian).

[4] Semenov A., Zaikin O., Bespalov D., Posypkin M.: *Parallel logical cryptanalysis of the generator A5/1 in BNB-Grid system.* Lecture Notes in Computer Science, vol. 6873, 2011, pp. 473–483.

[5] Semenov A., Zaikin O., Bespalov D., Posypkin M.: *Parallel algorithms for SAT in application to inversion problems of some discrete functions.* arXiv:1102.3563v1 [cs.DC].

[6] Schulz S., Blochinger W.: *Parallel SAT Solving on Peer-to-Peer Desktop Grids.* Journal Of Grid Computing, vol. 8, No. 3, 2010, pp. 443–471.

[7] Anderson D. P.: *Boinc: A system for public-resource computing and storage.* [in:] Fifth IEEE/ACM International Workshop on Grid Computing, 2004, pp. 4–10.

[8] *Desktop Grids for eScience*, – A Road map. Produced by DEGISCO. `http://desktopgridfederation.org/documents/10508/57919/RoadMapD.pdf`

[9] Cappello F., Djilali S., Fedak G., Herault T., Magniette F., Neri V., Lodygensky O.: *Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models,Security, Tests and Convergence with Grid* Future Generation Computer Science, 2005, pp. 417–437.

[10] Cirne W., Brasileiro F., Andrade N, Costa L.B., Andrade A., Novaes R., Mowbray M.: *Labs of the World, Unite!!!* Journal of Grid Computing, vol. 4, issue 3, 2006, pp. 225–246.

[11] Litzkow M., Livny M., Mutka M.: *Condor — A Hunter of Idle Workstations*. [in:] Proc. The 8th International Conference of Distributed Computing Systems, San Jose, California, June, 1988, pp. 204–111.

[12] Davis M., Logemann G., Loveland D.: *A machine program for theorem proving*. Communication of the ACM, vol. 5, 1962, pp. 394–397.

[13] Marqeus-Silva J. P., Sakallah K. A.: *GRASP: A search algorithm for propositional satisfiability*. IEEE Transactions on Computers, vol. 48, No. 5, 1999, pp. 506–521.

[14] Bohm M., Speckenmeyer E.: *A fast parallel SAT solver — efficient workload balancing*. Annals of Mathematics and Artificial Intelligence, vol. 17, No. 2, 1996, pp. 381–400.

[15] Hamadi Y., Jabbour S., Sais L.: *ManySAT: a Parallel SAT Solver*. Journal on Satisfiability, Boolean Modeling and Computation, Special Issue on Parallel SAT Solving, vol. 6, 2009, pp. 245–262.

[16] Schubert T., Lewis M., Becker B.: *PaMiraXT: Parallel SAT Solving with Threads and Message Passing*. Journal on Satisfiability, Boolean Modeling and Computation, vol. 6, 2009, pp. 203–222.

[17] Soos M., Nohl K., Castelluccia C.: *Extending SAT Solvers to Cryptographic Problems*. Lecture Notes in Computer Science, vol. 5584, 2009, pp. 244–257.

[18] Menezes A., Van Oorschot P., Vanstone S.: *Handbook of Applied Cryptography*. USA, CRC Press, 1996.

[19] *Quadratic Assignment Problem Library* `http://www.seas.upenn.edu/qaplib/`

[20] Colbourn C. J., Dinitz J. H.: *Handbook of Combinatorial Designs*. Chapman & Hall / Taylor & Francis, 2007.

[21] Eds. Bower J. M., Bolouri H.: *Computational Modeling of Genetic and Biochemical Networks*, MITPress, 2004.

[22] *Volunteer computing project SAT@home* `http://sat.isa.ru/pdsat/`

[23] Kacsuk P., Kovacs J., Farkas Z., Marosi A. C., Gombas G., Balaton Z.: *SZTAKI Desktop Grid (SZDG): A Flexible and Scalable Desktop Grid System*. Journal of Grid Computing, vol. 7, No. 4, 2009, pp. 439–461.

[24] Balaton Z., Gombas G., Kacsuk P., Kornafeld A., Kovacs J., Marosi A. C., Vida G., Podhorszki N., Kiss T.: *Sztaki desktop grid: a modular and scalable way of building large computing grids*. [in:] Proc. of the 21th Int. Parallel and Distributed Processing Symposium, Long Beach, California, USA, 2007, pp. 1–8.

[25] *The MiniSat page* `http://minisat.se/MiniSat.html`

[26] *Supercomputer center of ISDCT SB RAS* `http://www.mvs.icc.ru/`

[27] *Distributed computing stats system Free-DC* `http://stats.free-dc.org/`

[28] *A5/1 Cracking project* `http://reflextor.com/trac/a51/wiki`

[29] *Solutions found in SAT@home* `http://sat.isa.ru/pdsat/solutions.php`

Affiliations

Mikhail Posypkin
Institute for Systems Analysis of RAS, Moscow, Russia, posypkin@isa.ru

Alexander Semenov
Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia, biclop@rambler.ru

Oleg Zaikin
Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia, zaikin.icc@gmail.com

Received: 9.12.2011
Revised: 29.01.2012
Accepted: 30.01.2012