GridSAT Portal: A Grid Portal for Solving Satisfiability Problems On a Computational Grid *†

Wahid Chrabakh
University of California Santa Barbara
Department of Computer Science
Santa Barbara, CA
chrabakh@cs.ucsb.edu

Rich Wolski
University of California Santa Barbara
Department of Computer Science
Santa Barbara, CA
rich@cs.ucsb.edu

Abstract

We present a Grid Portal for solving satisfiability problems: **http://orca.cs.ucsb.edu/sat_portal** . *The portal provides a trivial interface which allows the use of a sophisticated and complex grid application GridSAT [6] running a large set of distributed computational resources hosted in different national computing centers. In this paper we describe the design goals of the portal and how it has influenced some of the application features. We also describe how a true grid application makes portal development easier and less complex.*

1 Introduction

Grid Computing is a research area with the vision of making computational power as easy and seamless to use as the electrical power grid. The grid paradigm allows for the aggregation of computational power to be used to solver problems in science and engineering. Just like its electrical counterpart, the computational grid consists of many sophisticated and powerful components. The complexity of Grid applications is due to their distributed nature and the "hostile" computing environments where they execute. It is a complex task to manage and coordinate a large set of components each performing a sub-task. The complexity is multiplied when the underlying sets of resources is dynamic. Therefore inorder to make the computational grid simple to use, the final user interface should be as simple as possible just like the electric socket. The final interface should isolate the users from as much internal complexity as possible. One way grid users are presented with an easy to use interface to complex grid applications is through grid portals. For example, the Cactus [3] portal is used to simulate blackholes and the Lattice [12] Portal specializes in high energy physics.

A computationally challenging problem with numerous application is boolean satisfiability. As satisfiability solvers became more efficient, they have been adopted as a powerful tool in many areas of science and engineering. In practice, many engineering disciplines require the solution of domain-specific large

*This work was supported by grants from the National Science Foundation, numbered CAREER-0093166, EIA-9975020, ACI-0103759, and CCR-0331654.

†This research was supported by an LRAC grant from the National Science Foundation through TeraGrid resources provided by SDSC, NCSA,PSC,TACC.

satisfiability instances. Such disciplines include scheduling [5], model checking [4], security [2], Artificial Intelligence [9] and software verification [8]. Satisfiability is especially important in the area of Electronic Design Automation (EDA). EDA encompasses a variety of problems such as circuit design [18], Field-Programmable Gate Arrays (FPGA) detailed routing [13], combinational equivalence checking [10, 15] and, automatic test and pattern generation [11].

Since solving satisfiability instances is very important in many fields of science and engineering, an annual competition is held [16] for satisfiability solvers. The solvers are rated using several criteria using a set of problems. The problems are are submitted by the SAT community [17] and belong to a large variety of fields. All the problem are represented in a community standard canonical CNF format. Figure 1 shows a short example where the first line defines successively the number of variables and the number of subexpressions in the logical formula. Each successive line defines a subexpression and terminates with a zero.

```
p cnf 3 2
-1 2 3 0
3 -1 -2 0
```

Figure 1. A short CNF example with three variables and a two part expression

However, most solvers used in practice are sequential. We have developed a parallel solver GriDSAT, as a true grid application. GridSAT is capable of out-performing current state of the art solvers on problems that these solvers can solve. Moreover, it has been able to use a large set of computing resources inorder to solve problems that current solvers could not solve.

In order to make this solving power available to interested users we have developed a GridSAT portal. The portal allows users to submit and solve their specific problem instances. The portal presents a simple interface for users where they just enter a few parameters. The portal then launches the problem on the available resources and collects the final result. The portal is designed so that the user does not interact directly with resources. It is the GridSAT application which automatically selects the resources and schedules dynamically the parallel components inorder to provide the best performance.

2 GridSAT

A propositional satisfiability problem takes as input an arbitrary logical expression about a set of variables. The problem is solved by determining whether there exists or there does not exist a set of variable assignments so that the given expression evaluates logically to “true”.

GridSAT is a complete and parallel distributed satisfiability solver. GridSAT can combine many heterogeneous resources to solve a given satisfiability instance. Each resource used by the application executes a client. The client consists of a sequential solver and can also asynchronously process incoming messages without interrupting the local solver. This method of communication allows clients to cooperate and share intermediate results with the other resources throughout the duration of the execution.

Because the number of resources required by a given problem is unknown, GridSAT adjusts the number of resources it uses according to the current problem. At the start of execution, only one resource is used. When the CPU or memory load exceeds a given threshold one more resources is added. New resources as assigned work by splitting the problem’s search space. This process is applied to all resources which become part of the execution. Eventually some of the resources are released as soon as they have finished investigating their assigned search space.

The master process, shown in figure 2, manages the entire application state. It uses existing grid services such as NWS [19] and Globus MDS [7] to discover and monitor grid resources. The GridSAT master process is designed to tolerate resource failures of remote clients at any stage of the execution. The application uses checkpointing to restart tasks after resource or network failures. Moreover, GridSAT adapts its work load balancing depending on the connectivity exhibited between groups of resources. For instance, work assigned to scattered resources is migrated to larger clusters of resources with higher network connectivity.

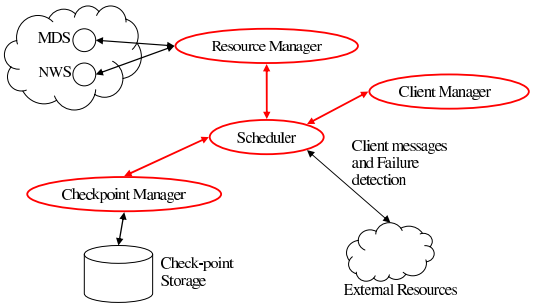

Figure 2. GridSAT components and their interactions. Grid resources are shown in clouds.

3 GridSAT Portal

In general, there exists two classes of portals. The first class is *user portals* which provides an interface for grid infrastructure to users. An example user portal is the NPACI portal [1]. The other class of portals is *application portals*. The GridSAT portal belongs to the latter category. There exists tools to help build user portals such as [14]. Application portals, however, are more specific and there have been attempts at building tools for such portals. For example, GridSpeed [14] abstracts portals into a mechanism for specifying parameters and a predefined set of templates for task execution. Under this abstraction, the purpose of the portal is to initiate commands on remote systems.

The GridSAT portal is different in two aspects. First, it is the GridSAT application that is responsible for launching and monitoring the tasks on the remote resources. Second, the number of tasks, the number of resources and duration of each task is not predefined. The GridSAT scheduler dynamically chooses the job characteristics (i.e. location, size and duration) depending on the observed resource load and problem behavior from previous jobs.

The GridSAT portal runs the GridSAT master component locally (or on a trusted host) to guard against any remote resource failure. Since the master process is long lived, the probability of a remote resource failing becomes higher. In fact, in GriDSAT we assume that any remote resource may fail at any moment. This may happen because of the resource's own failure or because the resource becomes unreachable through the network. According to our experience, all resources even those which are professionally maintained can become unresponsive from the application's perspective. Those resources that do not experience hardware and software failures usually have scheduled routine preventive maintenance periods or a combination of software and hardware upgrades. From the point of view of the application these are "scheduled" or "anticipated" failures.

Thus the GridSAT portal starts the master process locally. It is the responsibility of this process to insure the continual execution of the application in-spite of resource failures or performance degradations.

3.1 User Environment

The portal allows users to securely login and create their private accounts. After logging-in a user can submit satisfiability problems using the form shown in figure 3. The user submits problems as files in the standard CNF format. A set of test problems of variable sizes are available for download and can be submitted to the portal at **http://orca.cs.ucsb.edu/sat_portal/test_problems.htm**.

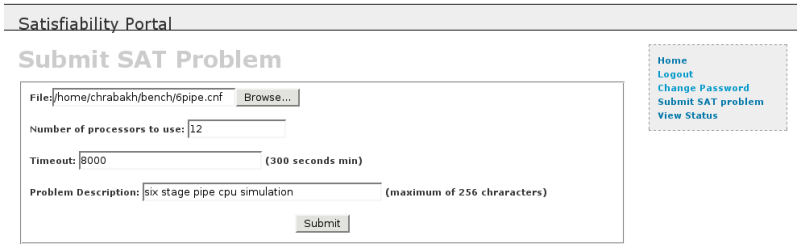

Figure 3. Screen shot of submission form used by GridSAT portal to submit satisfiability problems.

The user also specifies the maximum number of processors to use and the duration. The GridSAT application can take more parameters to control the rate of sharing intermediate results and other aspects of the scheduling procedure. The portal hides all these details because most users cannot determine which values to use for these parameters. Instead the GridSAT application uses heuristics to assign values to these parameters.

Currently the GridSAT portal can use many TeraGrid sites at NCSA, SDSC and DataStar. We are also in the process of integrating more computing resources such as LoneStar and PSC. Additional resources can be incorporated by simply installing the application and updating a configuration file. The user is involved in selecting which set of resources will be used. Instead the GridSAT application selects the resources automatically.

The user can query and manage his own set of submissions while being provided with continuous feedback. Th user can query the portal about all the sets of problems which he has previously submitted. Moreover, the user can view a detailed status for each problem. The status of each problem is continuously updated. The detailed view of a problem shows the CPU*Hours consumed, the number of active clients are running and the number of total splits which occurred during the elapsed execution time. Moreover, the portal displays which resources have been used and the submission,start and end time of each job. The combination of all this information presents the user with the progress rate at which a given problem is being solved.

When a problem is solved the satisfiable solution found is displayed. Otherwise the problem is marked as unsatisfiable. In some cases, the problem might timeout because the specified period has expired before a solution can be determined. The user can also cancel a given problem at any time even when it already started running. A user may also choose to delete the problem altogether and it will not be displayed in the history of his submissions. Also any disk resources used by the problem will be purged. In fact, all files associated with a given file will be deleted after a given time period inorder to alleviate disk space consumption.

3.2 Budget Based Scheduling

As shown in figure 3, when submitting a problem the user specifies two additional parameters which are the maximal number of processes and a maximal duration for trying to solve the satisfiability problem.

The GridSAT scheduler uses these parameters as guidelines to submitting resource requests because

it is not always possible fulfill exactly the user requests through a single allocation. For example, if the user asks for a number of processes greater than the number that can be provided by the resources available, GridSAT uses the maximal number of processes available within the resources instead. Also when a user specifies a small time duration lower than a minimal predetermined value, then all jobs requested will use the minimal time value instead. The portal enforces a minimal time for using a CPU. Using a CPU for a very short time period does not allow the solver enough time to make progress in solving the problem it is assigned. Therefore, the same problem will at exactly the same state when restarted later. Hence, a minimal duration for each job is enforced to avoid wasteful use of resources.

Effectively, the GridSAT scheduler tries to satisfy the user requests within the constraints of the resources available. If it is not possible then the scheduler submits a series of usually smaller jobs with equivalent total computational budget (cpu*hours). The scheduler keeps count of how much of the budget has been consumed by the application. The remaining portion of the initial budget is decremented continuously as processes are executing on behalf of the application. When a job terminates, the scheduler uses the remaining budget and resource specific parameters to submit a new job.

In the future we will replace both parameters with a single one. This new parameter will represent the total CPU*hours which will be dedicated to solving the submitted problem. The GridSAT scheduler will issue successively larger jobs inorder to solve the given problem.

4 Conclusion

In this paper we have presented the GridSAT portal based on the complete and parallel distributed satisfiability solver GridSAT. The GridSAT portal presents a simple user interface to a complex grid application running on a set of volatile but powerful resources.

The GridSAT portal hides all complexities involved in managing a grid environment because the underlying application manages all resources and failures seamlessly. The robustness of the GridSAT application has facilitated developing the portal and made it less complex.

References

[1] Npaci hotpage.

[2] L. C. Alessandro Armando. Abstraction-driven sat-based analysis of security protocols. In *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003*, pages 257–271, May 2003.

[3] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf. The Cactus Worm: Experiments with dynamic resource discovery and allocation in a Grid environment. *The International Journal of High Performance Computing Applications*, 15(4):345–358, 2001.

[4] C. W. B. Li and F. Somenzi. Abstraction refinement in symbolic model checking using satisfiability as the only decision procedure. December 2003.

[5] R. Bjar and F. Many. Solving the Round Robin Problem Using Propositional Logic. AAAI/IAAI, 2000.

[6] W. Chrabakh and R. Wolski. GridSAT: A chaff-based Distributed SAT solver for the Grid. In *Supercomputing Conference, Phoenix, AZ*, November 2003.

[7] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proc. 10th IEEE Symp. on High Performance Distributed Computing*, 2001.

[8] D. Jackson and M. Vaziri. Finding bugs with a constraint solver. *International Symposium on Software Testing and Analysis*, 2000.

[9] H. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence*, pages 359–379, August 1992.

[10] W. Kunz and D. Stoffel. *Reasoning in Boolean Networks: Logic Synthesis and Verification Using Techniques*. Kluwer Academic Publishers, Boston, 1997.

[11] T. Larrabee. Test pattern generation using boolean satisfiability. In *IEEE Transactions on Computer-Aided Design*, pages 4–15, January 1992.

[12] Lattice Portal. http://lqcd.jlab.org/.

[13] G. Nam, F. Aloul, K. Sakallah, and R. Rutenbar. A Comparative Study of Two Boolean Formulations of FPGA Detailed Routing Constraints. *International Symposium on Physical Design (ISPD), Sonoma Wine County, California*, pages 222–227, 2001.

[14] J. Novotny. The grid portal development kit, 2001.

[15] S. Reda and A. Salem. Combinational equivalence checking using boolean satisfiability and binary decision diagrams. In *Proceedings of the conference on Design, automation and test in Europe*, pages 122–126. IEEE Press, 2001.

[16] SAT Competitions. `http://www.satlive.org/SATCompetition/`.

[17] SAT Live. `http://www.satlive.org/`.

[18] J. P. M. Silva. Search Algorithms for Satisfiability Problems in Combinational Switching Circuits. Ph.D. Thesis, The University of Michigan, 1995.

[19] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 1999.