Grid-enabled Probabilistic Model Checking with PRISM*

Yi Zhang, David Parker, Marta Kwiatkowska
University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK
Email: {yxz, dxp, mzk}@cs.bham.ac.uk

Abstract

In this paper, we present our work on extending the probabilistic model checking tool PRISM with Grid computing capabilities. PRISM is a tool for the verification and analysis of large, probabilistic models, a process which involves numerical solution techniques. Like most verification tools, PRISM can be very resource intensive and will benefit greatly from parallelisation efforts. Grid computing provides an invaluable means through which to invoke parallel numerical solution engines in PRISM, providing support for job distribution, data transfer and task monitoring on large-scale parallel and distributed computing environments. We describe the middleware we are developing to integrate this functionality into PRISM and illustrate its use on large model checking case study.

1 Introduction

The increasing dependence of society on computer systems deployed in safety or business-critical domains requires assurances of software correctness. *Formal verification* techniques, such as *model checking*, have proved very successful in this area. Model checking involves the construction of a precise mathematical model of a real-life system, formal specification of several required properties of this system, and then an automated exhaustive analysis of the model in order to verify that the specified properties are satisfied by the model.

There are many examples of real-life systems for which an accurate analysis must also take into account *stochastic* aspects of the system's behaviour. These include systems that use randomisation, such as communication protocols, as well as those that exhibit uncertainty, for example computer networks and manufacturing systems. In this domain, *probabilistic model checking* [8] and, in particular, the tool PRISM [5, 1] have proved very successful in discovering design flaws and investigating performance characteristics. Like all formal verification techniques, however, developing efficient implementations of these methods is a major challenge.

The PRISM tool already incorporates sophisticated *symbolic* implementation techniques (using data structures based on binary decision diagrams) and is in the process of being extended with parallelisation support. These are used to handle the state space explosion problem faced by all model checking technologies. By applying the two approaches, PRISM has the potential to model and analyse very large real-life systems. However, parallel numerical solution engines have not yet been integrated into the PRISM tool and so manual intervention is required. End users are required to have certain knowledge and experience of the parallel and distributed systems where parallel engines are run. The procedure with manual intervention is tedious and error-prone and hinders the procedure of using PRISM in large and real-life systems. Here, we describe ongoing work to extend the range of real-life systems to which probabilistic model checking can be applied through the application of Grid computing.

The rest of this paper is organised as follows. Section 2 gives a brief introduction to the PRISM probabilistic model checking tool. Section 3 introduces the Globus Toolkit used in this paper. Section 4 presents the structure and details of extending PRISM with Grid computing. We show some real-life case studies

*Supported in part by EPSRC grants GR/S11107 and GR/S27252.

with the middleware presented in Section 5. Section 6 concludes the paper.

2 Probabilistic model checking with PRISM

Probabilistic model checking [8] is a formal verification technique for the analysis of systems which exhibit stochastic behaviour. It has already proven to be useful for studying a very wide range of systems. These include: real-time communication protocols, such as Bluetooth, IEEE 1394 FireWire and Zeroconf; security protocols for anonymity, contract signing and non-repudiation; dynamic power management schemes; fault-tolerant architectures; computer networks; and manufacturing systems. See e.g. [6, 5].

PRISM is an open source probabilistic model checker written in Java and C/C++ being developed at the University of Birmingham. Since its release in 2001, PRISM has been widely adopted for teaching and research worldwide (see the website [1] for downloads, case studies and bibliography). It is designed for the analysis of probabilistic systems and supports discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs). The probabilistic models are specified using the PRISM language, a high-level system description language from which models are automatically constructed. PRISM then performs analysis of a variety of properties of these models, as specified by the user. A graphical user interface allows editing of language descriptions, and facilitates automation of the analysis and collation and visualisation of the results.

In order to perform model analysis, PRISM combines graph-theoretic algorithms with numerical computation, such as solution of linear equation systems. As is typically the case with formal verification approaches, one of the main practical problems in this area is the state space explosion problem: the fact that models of real-life systems to grow prohibitively large, and hence require excessive amounts of memory and/or time to analyse. One way that PRISM counteracts these effects is through the use of novel *symbolic* implementation techniques. These employ sophisticated data structures based on binary decision diagrams (BDDs) which allow compact storage and manipulation of large probabilistic models by exploiting high-level structure in the model. With these techniques models as large as 10^{20} states can be stored and analysed.

Symbolic approaches, however, are unable to ensure efficient computation on their own. To this end, we use *parallel computing* in combination with symbolic approaches (see e.g. [10]), an approach currently being implemented in PRISM. This allows us to distribute both storage requirements and computation time across several processes or computers running in parallel. Symbolic methods can be beneficial in this setting because they can reduce the total amount of communication required between parallel components.

There are at least two ways to apply *parallel computing* to PRISM. Firstly, end users can perform parallel computing manually: using PRISM to generate and export linear equation systems on a desktop machine; then transferring the data to a parallel computing resource; starting a parallel numerical engine to solve the linear equations; and, finally, transferring the results back to their desktops and importing the results into PRISM. Another approach is to develop middleware to connect PRISM running on end users' desktops and parallel numerical engines running on remote parallel computation resources. This approach will free end users from learning the configurations of the remote parallel systems and reduce the chances of user errors. The whole procedure of transferring files and submitting jobs is tedious and error-prone. Furthermore, the middleware provides a platform for future parallel processing for model checking. For example, numerical results generated by parallel numerical engines can be processed at remote parallel computing resources before being transferred back to end users' desktops. The aim of this work is to provide an efficient middleware for end users to use parallel computation. The Globus Toolkit [4] developed by the Globus Alliance provides a good base for developing such middleware.

3 The Globus Toolkit

The Globus Toolkit [4] is an open source software toolkit that supports Grids and Grid applications and has been using to build Grid systems worldwide. It provides essential services for solving basic issues during the construction of such systems, including security, resource access and management, data movement and management, and resource discovery.

The last two years have witnessed a dramatic change in the Globus Toolkit, which has been changing to a service-oriented infrastructure based on Web services. The Globus Toolkit 3 follows version 1.0 of the Open Grid Services Infrastructure specification (OGSI) [9], released in July 2003. In Globus Toolkit 3, Grid services are stateful Web services. Globus Toolkit 3 uses its own conventions and extensions of WSDL (Web Service Definition Language). As the Web services architecture evolves rapidly, OGSI has become outdated. Now, the standard for Grid computing is the WS-Resource Framework [2].

The Globus Toolkit 4, GT4, is the latest release from the Globus Alliance which follows the WS-Resource Framework standard. It was released in April 2005. GT4 has a service-oriented architecture and follows the industry-standard Web services protocols and mechanisms for service description, discovery, access, authentication and authorisation.

GT4 provides a set of predefined services and components for building service-oriented distributed systems. Among them, the following services and components are essential for the key issues involved in this paper: Grid Resource Allocation and Management (GRAM) for job management, the Grid Security Infrastructure (GSI) for security, GridFTP and GSI-OpenSSH.

4 PRISM with Parallel Numerical Engines

Our aim is to use Grid computing to enhance the capabilities and performance of the PRISM tool. We will first describe the structure of PRISM and how end users currently invoke parallel numerical engine with PRISM manually. Then, we will discuss what needs to be done for the middleware which connects PRISM and its parallel numerical engines.

To model check a system using PRISM, a number of tasks need to be executed. The whole procedure is depicted in Figure 1. First, end users need to construct a model specification of the system with the high-level PRISM description language. They must also specify properties of the system to be analysed. These specifications are based on temporal logic: the logic PCTL for properties of DTMCs and MDPs, and the logic CSL for properties of CTMCs. This can either be done using the PRISM GUI or independently. Then, PRISM takes the model description, parses it and constructs the corresponding probabilistic model. This process includes determining the set of all reachable states of the model. Next, PRISM will generate the matrices and initial solution vectors which define the linear equation systems which must be solved to analyse the model. Finally, it will use one of its three numerical solution engine to solve the linear equation system and, using the solution, return the results of the model analysis to the user.

We are working on parallel and distributed numerical engines for PRISM. In our experience, the numerical solution part is typically a bottleneck for model checking. We have developed parallel and distributed numerical engines for shared memory systems [7] and message passing systems [10].

End users can presently use the parallel and distributed numerical engines manually with PRISM. First, end users will input the model and properties for verification. After that, PRISM will generate a matrix and initial vectors for the corresponding numerical solution problem. The matrix is represented by a BDD (binary decision diagram) based data structure. The data of the matrix and vectors will be exported into files by PRISM. The end user will then transfer the files to the remote parallel computing resources and invoke a parallel numerical engine to read in the files and solve the linear equation system. After the equation is solved, a file contained the solution vector will be generated by the numerical engine. The user will transfer the file back to the desktop and import it back to PRISM. Finally, PRISM can use the solution to present the results to the user.

The above usage of the parallel numerical engine with PRISM has several drawbacks. First, end users have to manage passwords on remote parallel resources by themselves. When there are several resources available, password management adds complexity for end users. Second, users have to interact with job scheduling systems on remote parallel resources. There are many different kinds of job scheduling systems and users have to spend time becoming familiar with these. When a new computation resource becomes available, end users may have to learn a new job scheduling system. Then, users have

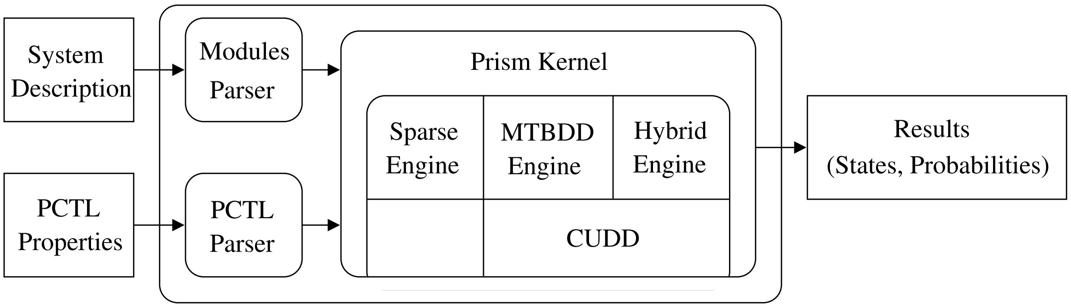

Figure 1. The structure of PRISM

to manage the transfer of data and solution files themselves. This file management becomes even worse when users want to handle several models at the same time. Finally, users have to monitor the progress of their jobs on remote resources. As job scheduling systems run on most resources, the finish time of a job depends not only on the computation time of the job but also on the load of the parallel resources and the configuration of the job scheduling system on the resource.

4.1 The structure of Grid-enabled PRISM

We are developing middleware based on the Globus Toolkit 4 (GT4) for the above procedure in order to free end users from this tedious and error-prone methodology. GT4 provides components based on Web services and components that are compatible with the Globus Toolkit 2. Our middleware uses the components based on Web services.

The middleware has the following three main components: job submission, data transfer and job monitoring. The structure of Grid-enabled PRISM is shown in Figure 2. The structure shows how the three components fit into PRISM. After the matrix and initial vectors are created, PRISM exports the matrix and vectors to files and uses the data transfer component to put these files on remote systems. After that, the job submission component will submit a job for the model checking task to the remote system. Then, users can monitor their jobs through the job monitoring component. After the job has finished, the result vector will be transferred back by the data transfer component.

All three components share the same the Grid Security Infrastructure (GSI) provided by the Globus Toolkit. The GSI is based on Public Key Infrastructure (PKI). It gives end users an effective single sign-on in a Grid environment and allows users to access parallel resources in a safe and secure manner.

4.2 Job Submission

The job submission component uses the Grid Resource Allocation and Management (GRAM) interface to submit job to remote computing resources. This interface is a key component in GT4 for monitoring, managing, scheduling and coordinating remote computing resources. GT4 provides two different GRAM implementations: WS GRAM, based on Web services, and Pre-WS GRAM, which is from GT2. At the same time, GT4 supplies APIs for C, Java, and Python clients. We use the C APIs for WS GRAM.

The job submission component works as follows. Information about remote computing resources is stored in a XML configuration file for PRISM. This information includes the end points of WS GRAM services on remote computing resources, home directories of end users on remote computing resources, and paths of parallel numerical engines. After PRISM creates the matrix and initial vectors, the job submission component will take over and export the matrix and initial vectors into several files. The file names, which contain process id, model name, and time stamp, are unique for each model checking session. In this way, users can analyse several models simultaneously or work on the same model with different parameters without concerns about conflicts of file names. The files will be transferred to the remote computing resource with the component which will be described later. The job submission component will create a

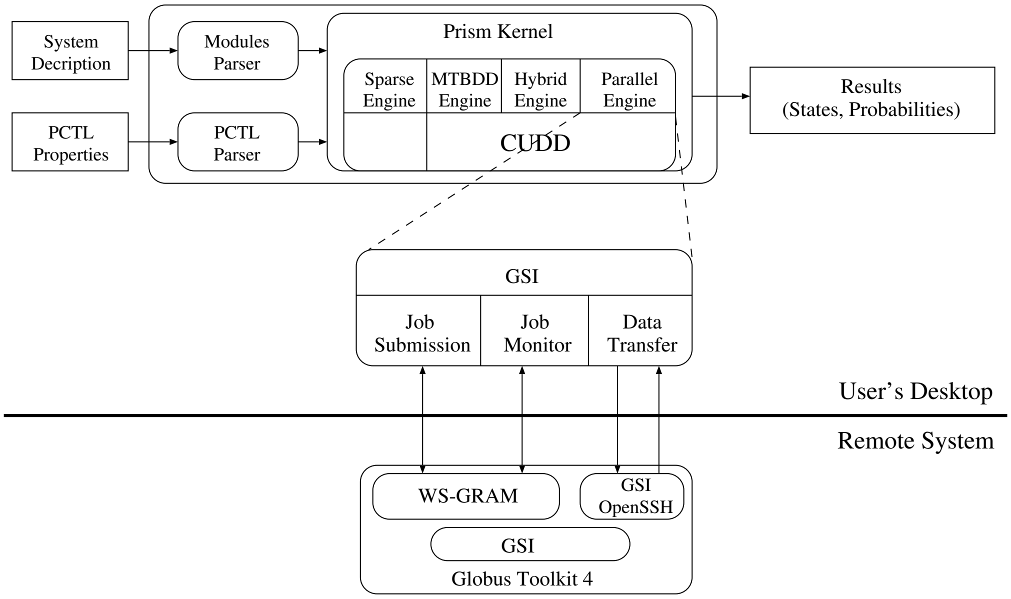

Figure 2. The structure of Grid-enabled PRISM

job description file in XML. This file includes details of which parallel numerical engine will be used, how long the program should be run for, where the data files are, and where the result file should be written. Then, the job submission component will invoke the WS GRAM service on the remote computation resource with the XML file. The WS GRAM resource on the remote site will parse the XML file and submit a job to the local job scheduling system. With a successful submission, a handle will be created and returned to PRISM. The job monitoring component described later can use this handle to monitor the status of the job.

4.3 Data Transfer

Numerical solution in PRISM often requires huge amounts of data. Matrices may be as large as several hundred megabytes. The initial vectors and the result vector may also require several gigabytes. Some examples will be given in Section 5. Efficient transfer of such huge amounts of data is very important.

The Globus Toolkit 4 provides two tools for data movement: GridFTP and SCP in GSI-OpenSSH. The implementation of GridFTP in GT4 is based on the protocol defined by the Global Grid Forum (Recommendation GFD.020). GridFTP provides secure, robust, fast and efficient transfer of data. Grid supports parallel transferring and TCP buffer size control. With the above techniques, GridFTP is extremely efficient for transferring very large quantities of data. The Globus Toolkit 4 also provides a set of web services around GridFTP to enhance its functionality. These include: Reliable File Transfer (RFT), Replica Location Service (RLS), and Data Replication Service (DRS).

GSI-OpenSSH is a new function provided in GT4. It is an extension of OpenSSH with support for GSI authentication and delegation. It provides a single sign-on remote login to remote systems and file transfer service without entering a password. GSI in GT4 provides a valid proxy credential for authentication for GSI-OpenSSH. At the client side, SSH/SCP of GSI-OpenSSH forwards proxy credentials to the remote system on login. The remote systems can obtain the information and certificate of the user through the Gridmapfile on the systems.

For experiments on the Grid system in our e-Science centre, we chose to use GSI-OpenSSH in our middleware for PRISM. To use parallel numerical engines with PRISM, we just need to transfer files between an end users' desktop machine and remote parallel systems. The advanced features of GridFTP, such as parallel transferring and Replica Location Service,

Table 1. The Kanban Model's parameters and statistics.

Model Size	States	Transitions	Size (MB)	
			Matrix	Vector
K=8	133,865,325	1,507,898,700	43	1,021
K=9	384,392,800	4,474,555,800	95	2,933
K=10	1,005,927,208	12,032,229,352	195	7,675

will not be used in our case. Furthermore, GridFTP imposes too many requirements on firewalls. In our environment, we have to deal with several firewalls. On the other hand, GSI-OpenSSH provides efficient and secure peer-to-peer file transfer functions. It uses the standard TCP port for SSH which is enabled in most firewalls. Based on our experience, the GSI-OpenSSH is easier to configure and maintain than GridFTP in an environment with many firewalls which is a common case nowadays.

4.4 Job Monitoring

Job monitoring is an important function for PRISM to let users know about the progress of their computation tasks. The total time for running a model checking job on a Grid is not easy to determine. First, the time at which a model checking task will be started in a Grid environment is not simple to determine: it is not the case that we run jobs on a standalone desktop where the job will start immediately. In a Grid environment, the queueing time is determined by the job scheduling system on the remote parallel system. Many factors will affect the time, such as the number of jobs in the queue and the priority of each job. We need to provide information to end users regarding approximately how long their jobs will be in the queue and the current load of the job scheduling system. Second, the execution time of the parallel numerical engine on a model checking task is also difficult to determine. Usually, the model checking tasks put on a Grid are too large to solve on a single desktop. We have no idea how much time is required to solve them on a standalone machine. Furthermore, the configuration of a remote parallel machine usually is quite different from a standalone desktop machine. The times required on a standalone machine gives little clue to the times required on a parallel computing resources.

Job monitoring give users information about the progress of their jobs at different stages. Such information will allow users to make prompt decisions. When users find that there are too many jobs in a queue on a remote system, they may switch to another queue or system. Users can check the convergence rate of their jobs. If they find that the jobs cannot finish within the allocated time slot, they can stop the job or request more time from the system.

At this moment, we use the default job monitoring function from the WS-GRAM services. It supports querying of the status of jobs and monitoring of the output and error streams of running jobs. We plan to create our own job monitoring Grid service which provides more detailed information about parallel numerical engines.

4.5 Middleware Usage

When using the PRISM Grid middleware, the first step for end users is to authenticate themselves with the Grid Security Infrastructure (GSI). To use GSI, end users must have valid GSI certificates. The certificates should be accepted by parallel resources where end users want to use. To sign onto the Grid and use remote resources, users need to use the "grid-proxy-init" command to create valid proxies of certificates. As described before, the information about available remote resources and their endpoints has been stored in a XML configuration file. Now, end users will use PRISM the same way as they use a standalone version. All tasks, such as authentication, file transfer and job monitoring, will be handled by the middleware.

5 An Example of Usage of the Middleware

In this section, we demonstrate the usage of our middleware for invoking parallel numerical engines in

PRISM. The case study we use is an analysis of a Kanban flexible manufacturing system [3]. Information about the probabilistic models we construct for this case study is given in Table 1. We performed analysis of these models using our PRISM Grid middleware on the Midlands e-science cluster. Table 2 provides performance results: the total computation time for several different models executed on different numbers of parallel nodes. The time given includes both model construction and numerical solution on the cluster. The time for queueing on the remote system is not included. Entries marked as "O/M" denote that there was insufficient memory to complete the analysis.

Table 2. Total computation time (seconds) for model checking of Kanban models.

Num. nodes	Kanban		
	K=8	K=9	K=10
1	19,868	O/M	O/M
4	8,666	38,451	O/M
8	4,698	18,120	O/M
16	2612	10,172	32,459
24	2,006	7,123	22,971

6 Conclusion

In this paper, we have described Grid-based middleware for integrating parallel computing resources within the PRISM probabilistic model checking tool. This provides an easy way for end users to use PRISM together with parallel computing resources on a Grid, free from the tasks of password management, file organisation and job monitoring. Users can easily switch from one parallel resource to another without considering about the different job scheduling systems on them.

The middleware provides a foundation for future parallelisation work in PRISM. At this stage, PRISM only uses parallel computing resources for numerical solution. As models become larger and more complex, so do the the result vectors and it becomes infeasible to store and analyse the vectors on a single machine. We plan to extend the middleware to handle the manipulation of result vectors.

The Globus Toolkit provides reliable and efficient components for extending existing software with parallel and distributed computing capacity. We use the latest release of GT4 which is well structured and documented. The run-time overhead of GT4 is very small compared with the total time required by our problems.

In the future, we plan to develop new Grid services for job monitoring and vector processing. The current job monitoring function we use is a part of the WS GRAM component GT4. It provides basic information about the status of a process. We are aiming to create a job monitoring service which provides detailed information about our parallel numerical engines. We also plan to develop a Grid service which processes the solution vector on the parallel computing resources before sending results back to users' desktop machines. This will reduce the amount of data which is needed to be send back.

References

[1] PRISM web site. www.cs.bham.ac.uk/˜dxp/prism.

[2] The WS-Resource framework. www.globus.org/wsrf.

[3] G. Ciardo and M. Tilgner. On the use of Kronecker operators for the solution of generalized stocastic Petri nets. ICASE Report 96-35, Institute for Computer Applications in Science and Engineering, 1996.

[4] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[5] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 2.0: A tool for probabilistic model checking. In *Proc. 1st International Conference on Quantitative Evaluation of Systems (QEST'04)*, pages 322–323. IEEE Computer Society Press, 2004.

[6] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic model checking in practice: Case studies with PRISM. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):16–21, 2005.

[7] M. Kwiatkowska, D. Parker, Y. Zhang, and R. Mehmood. Dual-processor parallelisation of symbolic probabilistic model checking. In D. DeGroot and P. Harrison, editors, *Proc. 12th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'04)*, pages 123–130. IEEE Computer Society Press, 2004.

[8] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden

and F. van Breugel (eds.), volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.

[9] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D. Snelling, and P. Vanderbilt. Open Grid Services Infrastructure. The Global Grid Forum, www.gridforum.org/documents/GFD.15.pdf.

[10] Y. Zhang, D. Parker, and M. Kwiatkowska. A wavefront parallelisation of CTMC solution using MTBDDs. In *Proc. International Conference on Dependable Systems and Networks (DSN'05)*, 2005. To appear.