Remote Job Management in the Möbius Modeling Framework

Ken Keefe, Quincy Mitchell, Eric Rozier, William H. Sanders

Coordinated Science Laboratory, Information Trust Institute,
Department of Electrical and Computer Engineering, and Department of Computer Science
University of Illinois at Urbana-Champaign
mobius-info@crhc.illinois.edu

Abstract

Large and complex models can often benefit from parallel execution on multiple machines. In the Möbius modeling environment, this is especially true of models for which the user wants to examine several independent configurations or parameterizations. A Remote Job Server with the capability to securely forward jobs to remote workstations or securely submit jobs via a standard batch submission system, manage these remote jobs, and collect results from the remote processes has been implemented in the tool. With the Remote Job Server, Möbius users can painlessly parallelize the execution of their models across many platforms and network topologies. This paper describes the Remote Job Server and summarizes other recent extensions to the Möbius tool.

1. Introduction

Möbius is a discrete-event system modeling and analysis tool that has been widely used for dependability, reliability, performability, and security analysis of systems. The Möbius tool builds models using the Möbius framework, which supports a wide range of modeling formalisms and solution techniques [2, 4, 7] via the Abstract Functional Interface (AFI). The AFI enables Möbius modules to interact in a unified manner.

The AFI is implemented by a set of C++ base classes [3], which represent common elements of discrete-event system models and allow models to interact without knowing details of the implementation for each model. Two distinct AFIs exist in Möbius: a "model-level" AFI that specifies state transitions on events, independent of the formalism in which the model is expressed, and a "state-level" AFI that allows access to the state of models and the state transition information without knowledge of how the stochastic process is implemented [3, 4].

The rich set of model formalisms and solution techniques available in the Möbius framework has made it well-suited to the analysis of large and complex systems [9, 8, 5]. In addition to implemented advances in representation and solution of such models in Möbius, the framework has also been expanded to include a Remote Job Server that provides sophisticated management of model solvers, executed in parallel. The Remote Job Server enables users to easily and securely manage the solution of several independent configurations or parameterizations of a model on multiple computing resources, whether they are multiple processors on a single workstation, multiple workstations on a network, or specialized nodes on an advanced computer cluster.

2. Remote Job Server

Möbius has long had the capability to execute independent simulations on multiple machines [1]. However, the previous method of distributing processing load required a significant amount of configuration, which was unsuitable for many users. The original system only supported the simulator, meaning that state space generators and analytical solvers could only be executed locally. The scheduling algorithm did not provide the ability to distribute work at a granularity finer than a per-experiment basis and had no way to alter the schedule on-the-fly in the event that a simulation process terminated early.

Users frequently had to choose between running processes on homogeneous system configurations (identical OS and library versions, etc.) and setting up a networked file system. In addition, all remote machines had to be directly reachable from the network and configured for password-free remote shell access. Many corporate or academic security policies do not allow for this type of configuration, and some requirements are simply not possible for many common compute cluster configurations (e.g., some do not allow individual nodes to communicate directly with systems outside of the cluster).

2.1 Capabilities

The Remote Job Server provides users of Möbius with the ability to schedule and manage processes on remote machines. It also allows users to indicate the appropriate priority of a job to the scheduler and to halt remote jobs and remove them from the scheduler, allowing solver execution to be cleanly halted and resumed later without losing existing progress.

General Möbius models (which include state space generators and solvers) are transferred as minimal archives to remote machines and built remotely in order to ensure compatibility of the models with the remote environment and to minimize traffic on the network when distributing models. Remote workstations' progress is constantly monitored by the head Remote Job Server node, which can, in turn, provide detailed status information and control to the user via the Möbius Java client.

For security reasons, all network traffic is encrypted using OpenSSL to ensure secure transmission of model components and results, to provide secure authorization of job launch requests, and to restrict requests to authorized Möbius users.

2.2 Architecture

The Remote Job Server operates in one of three modes: Manager, Worker, and Forward. The RJS Manager mode is typically a unique process in a Möbius remote job configuration and manages job scheduling, monitoring, and balancing of computation. The manager node also provides an interface between all the worker nodes and the user's Möbius client. The manager spawns automatically when the user runs a new job from the Möbius client or can be configured to run continuously as a service so multiple users can share the RJS configuration.

The Worker mode receives a minimal archive of a model, spawns processes that compile transformer or solver binaries, executes the binaries, reports status information back to the manager node at a regular interval, and reports the final results when it has finished processing a batch. The worker node provides a rich feature set to the manager node, which allows the manager node to optimize workload distributions across the available resources in a fine-grain manner.

The Forward mode seeks to provide flexibility in network topology by serving as a gateway between the manager node and other forward or worker nodes that the manager cannot directly contact. This enables the use of many configurations of clusters and network configurations that partition the network with firewalls in ways that the previous remote features of Möbius were unable to. The Forward mode is lightweight so that it can run on head nodes and network gateway machines, both of which often have stiff limits on processing resources.

3. Conclusions

The Remote Job Server was designed to be secure, flexible, and easy to use. It uses the OpenSSL public key infrastructure to handle all inter-process communication in a secure manner. It complements the existing modeling and solution capabilities of the Möbius framework by allowing users to evaluate metrics of interest for a given model in parallel. The RJS provides a rich job management interface for parallelizing the analysis of independent configurations or parameterizations of models via an intuitive and simple user interface, with minimal additional system configuration required.

4. Acknowledgments

The authors would like to acknowledge the contributions to the Möbius tool by the former members of the Möbius group and others from groups throughout the world. The authors would also like to thank Jenny Applequist for her editorial assistance.

References

[1] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster. The Möbius modeling tool. In *Proc. of the 9th Int. Workshop on Petri Nets and Perf. Models*, pages 241–250, Sept. 2001.

[2] D. Deavours and W. H. Sanders. Möbius: Framework and atomic models. In *Proceedings of the 9th International Workshop on Petri Nets and Performance Models*, pages 251–260, Sept. 2001.

[3] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The Möbius framework and its implementation. *IEEE Transactions on Software Engineering*, 28(10):956–969, Oct. 2002.

[4] S. Derisavi, P. Kemper, W. H. Sanders, and T. Courtney. The Möbius state-level abstract functional interface. *Perform. Eval.*, 54(2):105–128, 2003.

[5] S. Gaonkar, E. Rozier, A. Tong, and W. H. Sanders. Scaling file systems to support petascale clusters: A dependability analysis to support informed design choices. In *Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 386–391, 2008.

[6] Möbius Team. *The Möbius Manual, www.mobius.illinois.edu*. University of Illinois at Urbana-Champaign, 2008.

[7] W. H. Sanders. Integrated frameworks for multi-level and multi-formalism modeling. In *Proc. of 8th International Workshop on Petri Nets and Performance Models*, pages 2–9, Sept. 1999.

[8] S. Singh, A. Agbaria, F. Stevens, T. Courtney, J. F. Meyer, W. H. Sanders, and P. Pal. Validation of a survivable publish-subscribe system. *Int. Sci. Jour. of Comp.*, 4(2), 2005.

[9] F. Stevens, T. Courtney, S. Singh, A. Agbaria, J. Meyer, W. H. Sanders, and P. Pal. Model-based validation of an intrusion-tolerant information system. *23rd IEEE International Symposium on Reliable Distributed Systems*, pages 184–194, Oct. 2004.