VoDkaV Tool: Model Checking for Extracting Global Scheduler Properties from Local Restrictions

Juan José Sánchez Penas
LFCIA, University of Corunha
Campus de Elvinha S/N. 15071, A Corunha, Spain
juanjo@lfcia.org

Thomas Arts
IT-university of Giöteborg
Box 8718, 402 75 Göteborg, Sweden
thomas.arts@ituniv.se

Abstract

The VoDka server is a Video on Demand system developed using Erlang/OTP. We have developed a tool[1] that, taking directly a simple abstraction of the source code of the system, first translates it into a intermediate process algebra and, later, generates the state space of a given configuration of the system. From this state space, some global properties of the system can be extracted. The tool uses internally different translation and model checking tools, and has a prototype GUI for hiding the internal details of the process.

1 Tool description

The VoDka server [5, 4] is a flexible video-on-demand system, developed using Erlang/OTP [1]. This server can be delivered in many possible configurations with different types of hardware, different network topology and different number of subsystems. The storage subsystem of the server is composed by a hierarchy of different storage systems, i.e., disks, CD players or tapes. These devices all have restrictions of which the process controlling the device is aware of. A second layer of processes controls a set of devices in one machine and has restrictions, for example, the bandwidth of its connection. A third layer may be further out in the network and serve as a cache to store more popular movies. Thus, the system has a complex and flexible architecture.

We have constructed a tool that helps the developers of the system to formally verify scheduling properties of configurations of this system, without the need to build them. The tool takes the source code of all scheduler related modules as input, it lets the user build a configuration by a handy interface and can then be used to formally verify a set of predefined properties of the scheduling behaviour of the chosen configuration.

[1]Partially supported by MCyT, Spain, Project TIC 2002-02859

The tool is a further development of a general tool for the verification of distributed Erlang programs [2] in combination with a specific front-end for this scheduler application.

Every process in the scheduler of the system has a function determining local restrictions based on the configuration and present state of the system. The local properties are restrictions (on bandwidth and number of connections of disk drives, CD players, tape storage devices and such), local scheduling functions (filtering and admission policies) and cost related functions (state of the component and resources still available).

Given only these local restrictions, and the rest of the configuration of the system (number of levels and components in each level), it is far from obvious to extract information about the behaviour and performance of the system. Answering questions such as how many users can watch 'Star Wars' at the same time, is virtually impossible without building the actual configuration and testing this. Answers to such questions, however, are what both the operator of the video-on-demand server and the designers of the system are interested in. The former want to obtain information about the capacity of the system, and the later are more interested in knowing how the different distributed properties of the system influence its performance, in order to be able to know how to improve it (redesign and reconfiguration of the scheduler).

Our tool is made to help in this matter. From the source code of the system, in which the local functions are present, and some configuration parameters, it constructs a complete communication model of that configuration. With techniques from the area of formal methods (in particular model checking) these models are used to determine global properties of the system, such as the maximum number of a certain class of movies that can be served in parallel.

Many global properties of the system can be determined by testing, but testing all possible scenarios of users that request a movie is rather expensive. Moreover, one tests a certain configuration. Performing experiments with new drives, faster network connections and all that, increases the

Proceedings of the Third International Conference on Application of Concurrency to System Design (ACSD'03)
0-7695-1887-7/03 $17.00 © 2003 IEEE

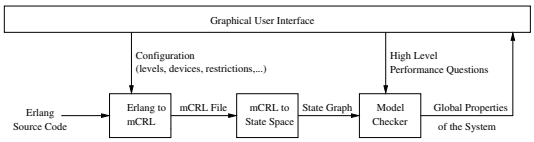

Figure 1. Three steps methodology: from Erlang to global properties

costs even more.

We use formal methods in a rather unconventional way. The combination of the front-end that we constructed for the scheduler and the slightly modified verification tool for Erlang [2], we construct a graph that represents the system load when users request all possible sequences of classes of movies. This graph is constructed by automatically translating the Erlang source code of a certain configuration into a process algebraic model (μCRL [7]). This model serves as an intermediate step for generating the performance graph of the system. This graph is a reduction of the state space generated (by existing tools [8, 9]) from the μCRL model.

In this graph the failures of requests are visible and therefore, the shortest path to a failure. This answers the question on how many users are guaranteed to be able to be served in parallel. Other questions, such as 'How many people can watch the movie A such that the system can still serve B?' or 'Where should we store the movie A for being able to serve it to N users?', can also be expressed as properties of the graph.

We designed a user interface to guide the whole process: choosing the parameters of the configuration, generating the model, constructing the graph and translating human understandable global properties of the system into a temporal logic formula. This formula is checked by model checking techniques using the Caesar/Aldebaran Development Package [6] (using the formula as a declarative way of asking for knowledge of the system and the checking techniques primarily as efficient graph search techniques).

2 Conclusions

The tool we have developed uses the scheduler of the VoDka server as a case-study for our methodology to verify global performance properties of a distributed system. The methodology is based on three main steps that are all performed in a completely automatic way, managed by the user with a high level graphical interface, developed with the goal of hiding the technical details.

With the current version of the tool, we are able to handle configurations of the system that are as complex as the ones that are being used in the VoDka prototypes. The main bottleneck is the step from the process algebra to the state space of the system, that can take quite a lot of time for complex system configurations. As an example: if we take a system with two level configuration, without cache, and almost without restrictions, but with four storage devices, and all the possible media object combinations, the resulting state space contains up to a few million states.

The methodology implemented in this tool and future work have been discussed more detailed in [3].

3 Acknowledgements

The authors would like to thank Víctor M. Gulías, member of the VoDka development team, for his help with the technical details of the server implementation. We also would like to thank Clara Benac Earle for her contributions to the development and testing of the tools we use as underlying framework.

References

[1] J. Armstrong, S. Virding, M. Williams, and C. Wikström. *Concurrent Programming in Erlang, 2nd edition*. Prentice Hall International, 1996.

[2] T. Arts and C. Benac Earle. Verifying Erlang code: a resource locker case-study. In *Int. Symposium on Formal Methods Europe*, volume 2391 of *LNCS*, pages 183–202. Springer-Verlag, July 2002.

[3] T. Arts and J. J. Sanchez-Penas. Global scheduler properties derived from local restrictions. In *Proceedings of Erlang Workshop at PLI2002*, Workshops in Computing Series. ACM, 2002.

[4] M. Barreiro, V. M. Gulías, J. L. Freire, and J. J. Sánchez. An Erlang-based hierarchical distributed VoD. In *7th Int. Erlang/OTP User Conference*. Ericsson Utvecklings AB, September 2001.

[5] M. Barreiro, V. M. Gulías, J. J. Sánchez, and S. Jorge. The tertiary level in a functional cluster-based hierarchical VoD system. In *Functional Programming and λ-Calculus Workshop*, volume 2178 of *LNCS*, pages 540–554. Springer-Verlag, February 2001.

[6] J. Fernández, H. Garavel, A. Kerbrat, and R. Mateesc. Caesar/Aldébaran development package: A protocol validation and verification toolbox. In *11th Int. Conf. on Computer-Aided Verification*, volume 1102 of *LNCS*, pages 437–440. Springer-Verlag, August 1996.

[7] J. Groote. The syntax and semantics of timed μCRL. Technical Report SEN-R9709, CWI, Amsterdam, The Netherlands, June 1997.

[8] SEN group. A language and tool set to study communicating processes with data. Technical report, CWI, `http://www.cwi.nl/~mcrl`, February 1999.

[9] A. G. Wouters. Manual for the μCRL tool set (version 2.8.2). Technical Report SEN-R0130, CWI, Amsterdam, The Netherlands, 2001.