Toward an efficient automatic design space exploration frame for multicore optimization

Horia Calborean*, Lucian Vintan*

* *Advanced Computer Architecture & Processing Systems Research Lab Lab -http://acaps.ulbsibiu.ro/research.php, "Lucian Blaga" University of Sibiu, Romania horia.calborean,lucian.vintan@ulbsibiu.ro*

ABSTRACT

During the recent years the computing system complexity has grown (many heterogeneous cores). Finding the best configuration in the extremely huge design space has become a major problem. Many performance indicators have to be considered (IPC, power consumption, area integration, etc.). Mature tools which are able to perform automatically this multiobjective exploration are needed. We present an automatic design space exploration framework called FADSE (Framework for Automatic Design Space Exploration) which comes to meet this need. It includes several multiobjective evolutionary algorithms and it is able to work with most of the existing computing system simulators.

KEYWORDS: Automatic design space exploration, software framework, multicore and manycore architectures

1 Introduction

Today's computing systems are getting more and more complex. Many heterogeneous cores are used as building blocks. The number of configurations that have to be simulated in order to find the best one is enormous. Simulating all the possible instances is impossible due to the time required. For example evaluating a 4 core system where the cores can be selected from a library containing 480 models will require more than 2 billion evaluations [Vin09].

We can identify two approaches (that can be combined) in solving this problem: reducing the simulation time and reducing the number of simulations. This short paper focuses on the last one.

[1]E-mail: {horia.calborean,lucian.vintan}@ulbsibiu.ro

[2]This work was partially supported by CNCSIS no. 485/2008 research grant offered by the Romanian National Council for Academic Research.

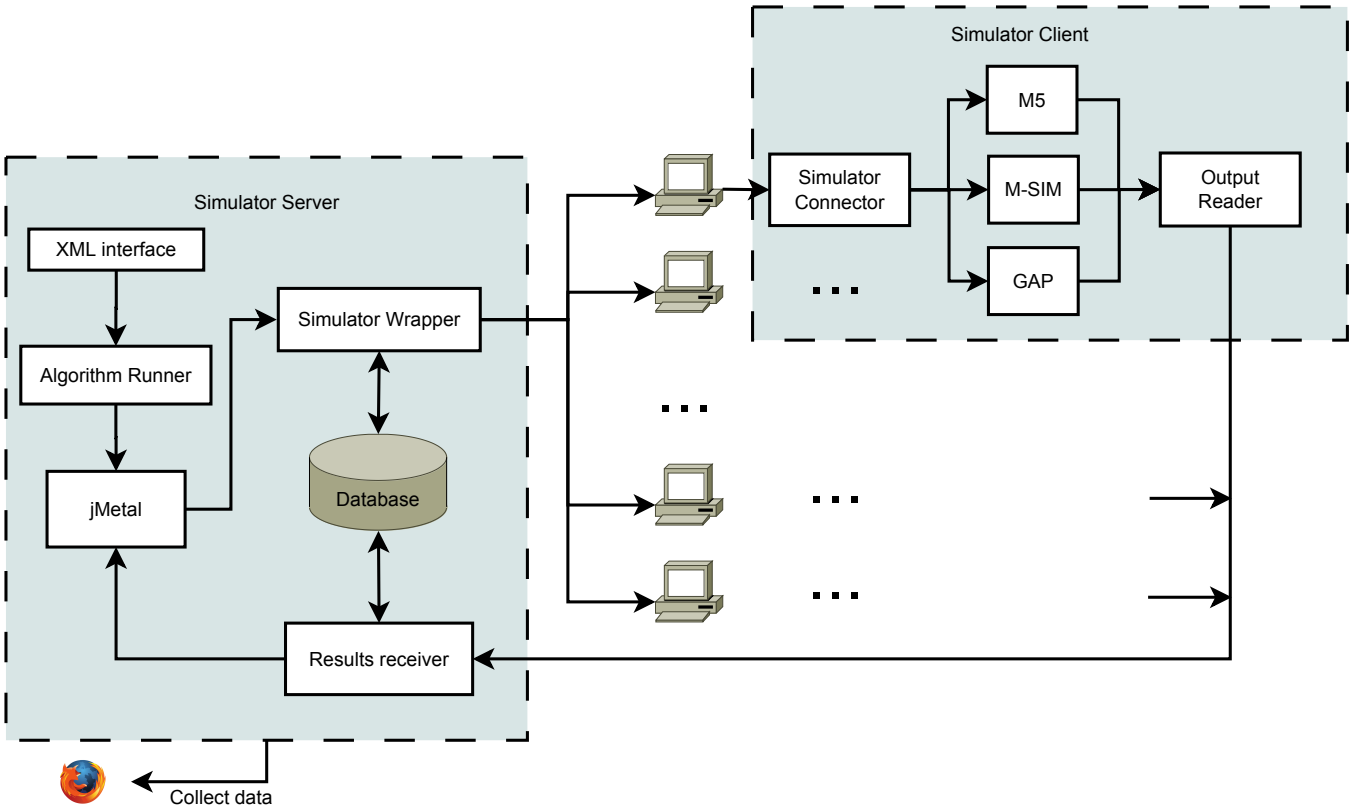

Figure 1: FADSE current architecture

We propose a framework called FADSE which is able to perform automatic design space exploration using many well known evolutionary multiobjective algorithms [CVL02] : NSGA-II, SPEA2, PAES, PESA, etc. To our knowledge there are few available frameworks for design space exploration (e.g. Magellan [KK08], M3Explorer [OS]) and they are bounded to a certain simulator (Magellan) or to a certain operating system (Magellan and M3Explorer). Our tool is implemented in JAVA which makes it portable and it is not bounded to any simulator. This allows us to run almost any existing simulator.

2 The developed framework

The framework is designed to be flexible so that new algorithms, simulators and evaluation metrics can be easily added.

FADSE makes use of the implemented mutiobjective algorithms found in the jMetal [DNL+06] library. The user configures the framework through an XML file (the XML file is similar with the one used in M3Explorer[DNL+06]).

The first step is to specify the name of the simulator and the path to the executable file. The desired design space exploration (DSE) algorithm is selected and its parameters (e.g. maximum number of generations, crossover operator, etc.) can be set.

Next the simulator parameters are described. All the parameters that need to be varied have to be written in the XML and also their possible values (the range in which they can be varied). The parameters can be integer, float and lists of strings (e.g. possible benchmarks, types of branch predictors). The integer parameters can be varied in arithmetic or geometric

progression.

Then the objectives are described: their names and whether the objective has to be minimized (e.g. power consumption) or maximized (e.g. IPC).

A set of constrains (rules) is defined in the XML configuration file. These constrains limit the design space by not allowing the tool to search for impossible/trivial configurations (e.g. level 2 data cache smaller than level 1 data cache). We have already implemented three types of rules (similar with M3Explorer): *relational rule*, *and rule* and *if rule* [CV10].

The *relational rule* can specify relations between the parameters (e.g. parameter 1 must be greater than parameter 2). The relations can be: greater, greater or equal, equal, less, less or equal, not-equal.

The *and rule* can specify a set of rules that must be obeyed at the same time.

The *if rule* is used when the user wants to apply a rule only when a certain condition is met.

FADSE server is able to start several clients, on different computers, to run the simulation in a parallel-distributed manner. The user has to specify in a separate XML file the structure of the network (a list of IPs and other information about each client). In this way FADSE is able to discover all the clients and their properties. A client can also be configured to run multiple simulations in parallel. In the XML configuration file, to each client a number of slots is assigned. This represents the number of individuals a client can evaluate in parallel. This feature is very important since most simulators are not threaded applications and do not benefit of the multicore processors found today in most desktop machines.

After the XML is read the Application Runner (see Figure 1) configures the jMetal library: it selects the DSE algorithm, configures its properties accordingly with the XML file requirements, converts the parameters and the objectives to the jMetal format, selects the problem to be solved and starts the algorithm.

jMetal starts generating individuals and sends them to the Simulator Wrapper for evaluation. The Simulator Wrapper receives the individual to be evaluated. It searches for the individual in the database and if the same individual has not been evaluated before it sends it to the Simulator Connector.

A Simulator Connector has to be implemented for every simulator, to make it compatible with FADSE. Up to this moment FADSE is able to run and interpret the results of the following simulators: GAP[USJU09], M-SIM (http://www.cs.binghamton.edu/ msim/) and M5 (http://www.m5sim.org) . The Simulator Connector starts the selected simulator with the parameters extracted from the received individual. After the simulation is done, the results are extracted by the Output Reader and sent back to the Results Receiver.

The Results Receiver saves the results in the database. The database is implemented using Apache Derby (http://db.apache.org/derby/). This helps us to distribute the application without the need to install other software on the host.

The Results Receiver converts the results to the jMetal format and passes them to the DSE algorithm. The algorithm generates new individuals and the process is continued until a stop condition is met. Then the current Pareto optimal set [CV10] is saved and presented to the user.

The algorithm progresses can be followed from a web interface. This feature has been implemented using the Apache Tomcat server (http://tomcat.apache.org/), JSP and AJAX (through DWR - http://directwebremoting.org/). FADSE does not need this module to run. On systems where security is a problem or installing the Tomcat server is not possible the web application can be omitted.

3 Conclusion and further work

We have presented a framework that is able to perform automatic design space exploration on most of the existing simulators. The portability offered by JAVA allows us to run simulators implemented on any platform supported by the virtual machine.

We have used the jMetal library which has been developed for over three years, it is well tested and the confidence in the correct implementation of the algorithms is high.

FADSE offers the capability to run parallel simulations which leads to a great improvement when it comes to the time required to find the Pareto optimal set.

Our current focus is to develop connectors to other simulators like GEMS, ns3, etc.

We are planning to perform a comparison between the DSE algorithms on how well they perform on the problem of design space exploration of a computing system.

As a future development we want to make it possible to configure/control the framework from a web browser directly. This will allow the user to perform configurations more easily using a simple GUI.

References

[CV10] Horia Calborean and Lucian Vintan. An automatic design space exploration framework for multicore architecture optimizations. In *Proceedings of The 9-th IEEE RoEduNet International Conference*, Sibiu, Romania, June 2010. IEEE Xplore Digital Library.

[CVL02] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 1 edition, June 2002.

[DNL+06] Juan J. Durillo, Antonio J. Nebro, Francisco Luna, Bernabe Dorronsoro, and Enrique Alba. jMetal: a java framework for developing Multi-Objective optimization metaheuristics. Technical Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computacion, University of Malaga, E.T.S.I. Informatica, Campus de Teatinos, December 2006.

[KK08] Sukhun Kang and Rakesh Kumar. Magellan: a search and machine learning-based framework for fast multi-core design space exploration and optimization. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1432–1437, Munich, Germany, 2008. ACM.

[OS] M. O.D.S.E OF and M. P SOC. FP7-216693-MULTICUBE project.

[USJU09] Sascha Uhrig, Basher Shehan, Ralf Jahr, and Theo Ungerer. A two-dimensional superscalar processor architecture. In *The First International Conference on Future Computational Technologies and Applications, Athens, Greece*, 2009.

[Vin09] Lucian Vintan. Directii de cercetare in domeniul sistemelor multicore / main challenges in multicore architecture research. *Revista Romana de Informatica si Automatica*, 19(3), 2009.