RSIM x86: A COST-EFFECTIVE PERFORMANCE SIMULATOR

Ricardo Fernández and José M. García
Departamento de Ingeniería y Tecnología de Computadores
Universidad de Murcia, 30071-Murcia (Spain)
E-mail: {r.fernandez,jmgarcia}@ditec.um.es

KEYWORDS
High-performance simulators, multiprocessor architectures, computer architecture, large scale computing platforms.

ABSTRACT

In this paper we present RSIM x86, a port of the widely used RSIM performance simulator for cc-NUMA multiprocessors to GNU/Linux and x86 hardware. Then, we evaluate the simulation throughput obtained by RSIM in several platforms with respect to the hardware cost of each platform. We show that this port of RSIM obtains much better execution times using cheaper and more easily available hardware than the original RSIM, allowing a more efficient usage of our research resources.

INTRODUCTION

Doing research or system design in computer architecture involves deciding among many interrelated tradeoffs. Computer architecture is increasingly driven by quantitative data. Usually, developers can devise analytical models to bound the design space in the very early development stages but the interactions between many design decisions in today increasingly complex systems make impossible to use these analytical models to accurately predict the performance of a finished system. Hence, we need experimental models in order to guess the performance impact of a possible design decision before building a finished system.

Doing direct performance measurements requires a finished model, hence it is not possible to do it during the design phase. Also, building prototypes is too expensive for most research projects. As an alternative, system architects and researches use performance simulators to predict the effect of the ideas and techniques that they need to evaluate.

Performance simulators are complex software systems which accurately model the behavior of a hardware system. Doing a simulation of a hardware model is several orders of magnitude slower than running the simulated system. Developers need fast and accurate simulators to be able to perform as many useful experiments as possible.

There are two main types of performance simulators for processors: trace driven and execution driven. Trace driven simulators use traces obtained from the real execution of programs to drive a performance model while execution driven simulators simulate the actual execution of a program recording detailed performance statistics. The current trend in performance simulation is to use execution driven simulation because it allows much more precise results specially for current processors which exploit instruction level parallelism using out of order execution and speculation.

There are several popular execution driven performance simulators and simulation frameworks like SimOS (Rosenblum et al. 1997), MASE (Larson et al. 2001), Winsconsin Wind Tunnel II (Mukherjee et al. 2000), SimpleScalar (Austin et al. 2002), Simics (Magnusson et al. 2002), Asim (Emer et al. 2002) or RSIM (Hughes et al. 2002).

RSIM (Hughes et al. 2002; Pai et al. 1997a) is a simulator primarily targeted to study shared-memory cache coherent (cc-NUMA) multiprocessor architectures built from processors that aggressively exploit instruction-level parallelism (ILP).

RSIM key advantage is that it models a system comprised by several out-of-order processors which aggressively exploit instruction level parallelism (ILP). The model includes an aggressive memory system and a scalable interconnection network. Using detailed ILP models for the simulated processors provides a realistic approximation to modern and future multiprocessor systems. RSIM provides a great flexibility which allows using it to simulate a range of systems from monoprocessors to different cc-NUMA configurations.

The accurate and flexible model provided by RSIM implies a slower execution speed than other less detailed simulators. Furthermore, although RSIM is supposed to be portable it was not available on common and cheap architectures like Linux/x86, requiring instead Solaris/SPARC, IRIX/MIPS or other big-endian machines. This has proved to be a serious problem to our research group due to the limited access to these kind of machines.

Proceedings 19th European Conference on Modelling and Simulation
Yuri Merkuryev, Richard Zobel, Eugène Kerckhoffs © ECMS, 2005
ISBN 1-84233-112-4 (Set) / ISBN 1-84233-113-2 (CD)

In this work we show how we ported RSIM to Linux/x86 and how that allows us to obtain an increased performance for our simulations at a fraction of the original cost.

In the next section we examine some other performance simulators available, specially those derived from RSIM. Later, we explain some key characteristics of RSIM and the approach we have followed to porting RSIM to Linux/i386. After that section, we evaluate the performance of the ported simulator with respect to the cost of the hardware used to run the simulations. And, finally, in the last section, we present our conclusions from this work.

RELATED WORK

Some performance simulators already mentioned, like Simics or SimpleScalar, already have functional Linux/x86 ports. However, none of them is suitable for our purposes due to the lack of detail of the simulation and the different system architectures that they model.

Simics is a full system functional simulator which attempts to strike a balance between accuracy and performance. The goal of Simics is to allow the simulation of realistic workloads running unmodified operating systems and applications. Simics supports several simulated architectures including x86, UltraSPARC-II, PowerPC and others. It includes accurate device models to simulate I/O intensive applications common in commercial workloads. However, the big performance that it needs and its genericity prevents a detailed enough simulation necessary for many of our tasks.

SimpleScalar is a simulation toolset which provides an infrastructure for simulation and architectural modeling. The toolset can model a variety of platforms ranging from simple unpipelined processors to detailed dynamically scheduled microarchitectures with multiple-level memory systems. However, it simulates only uniprocessor systems.

RSIM has been used by many research groups since its publication and continues to be used nowadays. However, up to our knowledge, no one has published a functional x86 port until now. The original RSIM authors reported initial efforts in this direction, but their x86 port was never published, if it was ever completed.

Schaelicke and Parker ML-RSIM (Schaelicke and Parker 2002) is a derivative of URSIM (Zhang 2001) which is based on the original RSIM. It models the entire Input/Output subsystem and includes a functional operative system kernel called *Lamix* which is System V compatible. ML-RSIM has been ported to Linux/x86 architectures.

Unfortunately, ML-RSIM differs significantly from RSIM and does not model a cc-NUMA architecture, which makes it unfit for our research.

PROBLEMS PORTING RSIM

RSIM is an interpreter for Solaris/SPARC v9 application executables. Internally, RSIM is a discrete event-driven simulator based on the YACSIM (Yet Another C Simulator) library from the Rice Parallel Processing Testbed (RPPT) (Convington et al. 1991; Pai et al. 1997b).

RSIM interprets application executables rather that uses traces, enabling more accurate modeling of the effects of contention and synchronization in multiprocessor simulations as well as speculation in multiprocessor and uniprocessor simulations. For speed, it converts the SPARC v9 instructions into an expanded, loosely encoded instruction set format and internally caches them.

RSIM subsystems include the processor engine, the memory module, the cache module, the directory module and the interconnection network. Each subsystem is mostly independent from each other and they interact through a small number of predefined interfaces.

RSIM is written in a modular fashion using C++ and C for extensibility and portability. Initially, it was developed using Sun systems (Solaris 2.5) on SPARC. It has successfully ported to HP-UX 10 running on a Convex Exemplar and to IRIX running on MIPS. However, porting it to 64-bit or little-endian architectures requires significant additional effort.

We have successfully ported RSIM to GNU/Linux running on x86 architectures. The main problems that we have had to solve were:

- Build issues due to differences in libraries and headers between Solaris and Linux.
- Byte Ordering Issues.
- System call interface differences.
- Floating point incompatibilities.

RSIM was developed using big-endian machines and simulates a big-endian architecture. This configuration is straightforward, but on the other hand our port needs to simulate a big-endian architecture on top of a little-endian machine. This implies that at some places we have to change the order of bytes in words.

We keep the simulated memory always unmodified in the target endianness, so that unaligned accesses or packed arrays are easy to handle. Basically, we swap bytes when performing each simulated memory operation.

The predecoded executable data is generated at benchmark build time. Therefore, it is in the same endianness than the original executable. We swap its bytes after loading it for simulation and cache the byteswapped version.

The endianness differences also subtly affect other parts of the simulator, like partially overlapped forwarding of memory operations and some initialization routines. Some of these cases have been detected only after careful debugging.

Linux and Solaris system call interfaces are not exactly the same, althought they are very similar since they are both based on System V. Hence, some simulator traps that rely on host system calls require a translation of its parameters before they can be executed.

The floating point incompatibilities between SPARC and i386 are caused by the fact that SPARC implements the IEEE 754 floating point standard while i386 processors use the Intel x87 80-bit format for representing floating point numbers in the processor registers, even if they are stored in IEEE 754 format in memory.

In most cases, the extra precision is beneficial, but not for our purposes because it causes different results due to rounding differences. We wish to obtain the exact same results independently of the underlying architecture to be able to compare with our previous results, and the rounding differences would make this impossible.

To solve this, we instruct the compiler to produce floating point code using SSE2 instructions and registers present in newer x86 processors (Intel Corporation 2004). These instructions use eight 128-bit registers which can hold two double precision (64-bit) or four single precision (32-bit) numbers using the IEEE 754 format and perform short vectorial operations with them or scalar operations.

Another problem related with floating point differences is the different bit representation of the "Not a Number" (NaN) value in SPARC and i386 using SSE. Both representations are correct according to the IEEE 754 standard, but a benchmark could behave sightly differently if it tried to interpret the in memory representation of a NaN value.

Out port can optionally normalize the NaN representation ensuring that the SPARC representation is always used (this normalization process may affect simulation performance for our port. This option was enabled in the simulations used to measure times for this work). Most benchmark results are unaffected by this difference, but it makes debugging the port harder due to the different values stored in the simulated registers and memory.

Finally, the functions used to implement the simulated SPARC instructions to change the rounding modes are different between Solaris and Linux.

We have ensured that our port obtains exactly the same simulation results in both Solaris/SPARC and Linux/i386 architectures. This allows us to use all our machines to perform simulations and, more importantly, meaningfully compare the results of benchmarks and compare new results with old results from experiments performed prior to the port.

During the development of the port, we extended RSIM to produce extensive trace information detailing the content of every register and the instructions being executed at every moment in an easy to parse and compare format. This tracing support has proved useful beyond its initial purpose to diagnose problems during the development of new experiments.

EVALUATION

The purpose of our evaluation is to check if using the ported version of RSIM on off-the-shelf x86 machines is a cost-effective solution to perform the great number of long running simulations needed for our research.

Firstly, we compare the execution speed of RSIM running in several different architectures. Secondly, since the execution time of a single benchmark is not the most valuable metric for our purposes, we define a better indicator of the usefulness of each simulation platform and version of RSIM.

Usually, simulations are run in batches using a queue management system like Condor (Thain et al. 2004). We will measure which version allows us to utilize our computing resources more efficiently in terms of hardware cost and execution time. We will use a metric based in the normalized number of simulations per hour per thousand euros.

We have measured the impact that the actual benchmark being simulated has in the speedup obtained by our port and have found that it is small, but not inexistent. Hence, we have chosen a small set of representative benchmarks from the SPLASH suite to perform our experiments. The chosen benchmarks are:

fft: Complex unidimensional version of the radix-$\sqrt{n}$ six-step FFT algorithm optimized to minimize interprocessor communication. It is one of the fastest benchmarks with less memory requirements.

em3d: Models the propagation of electromagnetic waves through objects in three dimensions.

ocean: Studies large-scale ocean movements based on eddy and boundary currents.

We have also measured the impact that varying the problem size of the simulated benchmarks has in the achieved

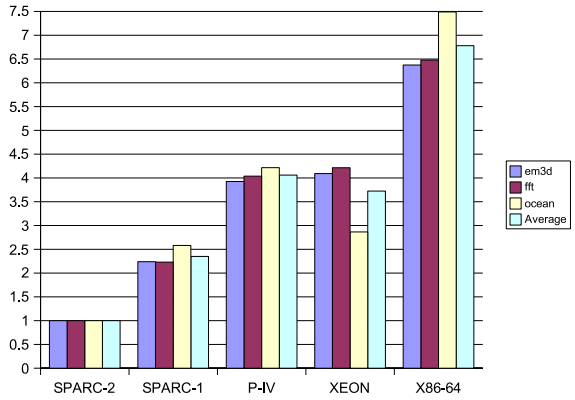

Figure 1: Normalized Throughput Per Processor for Each Architecture

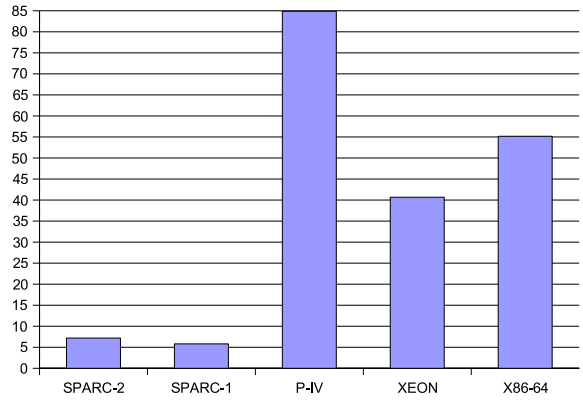

Figure 2: Average Number of Simulations Per Hour Per Thousand Euro

speedup and have found that it is very small once a certain threshold is reached. Other simulator parameters, like the number of processors, have a very small influence too.

We have chosen medium problem sizes and we use two processors and default values for the rest of the parameters for our experiments.

We have evaluated the speed of running our port of RSIM in the following machines:

- A high-end Solaris/SPARC Sunblade-2000 system: *SPARC-1*.
- A low-end Solaris/SPARC Sunblade-100 system: *SPARC-2*.
- A high-end Linux/Athlon64 SMP system (running in legacy IA-32 mode): *X86-64*.
- A high-end Linux/Xeon SMP system: *XEON*.
- A low-end Linux/Pentium-IV system: *P-IV*.

The relevant characteristics and price of each machine is shown in table 1. When testing both SPARC machines, we have used a version of RSIM targeted for that platform. The prices indicated for the machines are necessarily approximate. These are the approximate prices that those systems would cost as of January 2005 in Spain.

In figure 1 we show the normalized throughput time of our set of benchmarks for each architecture. In other words, we show the throughput speedup of each machine compared with the slowest one (*SPARC-2*). For now, we only use one processor per machine even on those machines which have two processors, to allow a fair comparison between them.

We see that Linux/x86 based machines outperform the more expensive Solaris/SPARC based ones in raw simulation speed per processor. The fastest platform is the Linux/Athlon64 based machine, which is 81% faster than the Xeon based machine and 67% faster than the Pentium-IV based machine (despite the much higher clock frequency of the Pentium-IV).

Some of our machines are SMP systems with two processors. In those cases, we can run two instances of RSIM simultaneously effectively doubling the throughput. Since the simulation work is CPU limited with very little IO and modest memory requirements there is no contention between the two independent processes.

In figure 2 we show the average number of simulations per hour per thousand euros achieved for each platform. When we account for the price of each machine, we see that the cheapest platform, the *Pentium-IV*, is the best option for efficiently take advantage of any given budget. Also, the easy availability of these kind of machines make them an even more attractive alternative to the expensive Solaris/SPARC machines used until now to run simulations based on RSIM. The two Solaris/SPARC based machines are much less cost-effective than the other platforms despite their higher prices.

CONCLUSIONS

The purpose of our port of RSIM is to allow us to use our research resources more efficiently. Prior to the port, the small number of available machines to develop and run our simulations created long waiting queues and serious organizational problems.

Using a RSIM version which runs on cheap and readily available x86 hardware allows us to provide each researcher with its own workstation to comfortably develop and test his experiments and use an inexpensive cluster of Linux/x86 machines to execute the longest simulations. The x86 version not only executes each benchmark faster, but more importantly, it is easier to provide more resources to increase the throughput of the whole team.

Table 1: Characteristics of Evaluated Configurations

	SPARC-1	SPARC-2	X86-64	XEON	P-IV
Processor	UltraSPARC-III	UltraSPARC-IIi	AMD Opteron	Intel Xeon	Intel Pentium-IV
No. of processors	1	1	2	2	1
Frequency	1015 MHz	650 MHz	1791 MHz	2 GHz	3 GHz
RAM Memory	2GB	256 MB	1GB	1GB	1GB
L2 Cache	8MB	512 KB	1024 MB	512 KB	1024 KB
Price	5000€	1800€	3000€	2600€	600€

The work required to port RSIM to x86 resulted greater than initially expected. RSIM code makes too many assumptions about low level details that are not portable and debugging problems caused by the different endianness proved to be difficult and time-consuming.

The code resulting from the port still has many portability problems that prevent its use in 64-bit architectures, like AMD x86-64 in native mode.

We are looking forward into making RSIM code more portable. In particular, we will make it 64-bit clean so it can be compiled natively for x86-64 architectures.

AVAILABILITY

The source of our port is publicly available at:
`http://www.ditec.um.es/gacop/tools/RSIM-x86`

ACKNOWLEDGEMENTS

This work has been supported by the Spanish Ministry of Ciencia y Tecnología and the European Union (Feder Funds) under grant TIC2003-08154-C06-03, and by fellowship 01090/FPI/04 from the Comunidad Autónoma de la Región de Murcia (Fundación Séneca, Agencia Regional de Ciencia y Tecnología).

REFERENCES

Austin, T.; E. Larson and D. Ernst. February 2002. "Simplescalar: An infrastructure for computer system modeling". *IEEE Computer 35*(2), 59–67.

Convington, R. G.; S. Dwarkadas; J. R. Jump; J. B. Sinclair and S. Madala. 1991. "Efficient simulation of parallel computer systems.". *International Journal in Computer Simulation 1*(1).

Emer, J.; P. Ahuja; E. Borch; A. Klauser; C.-K. Luk; S. Manne; S. S. Mukherjee; H. Patil; S. Wallace; N. Binkert; R. Espasa and T. Juan. February 2002. "Asim: A performance model framework". *IEEE Computer 35*(2), 68–76.

Hughes, C.; V. Pai; P. Ranganathan and S. Adve. February 2002. "RSIM: Simulating shared-memory multiprocessors with ILP processors". *IEEE Computer 35*(2), 40–49.

Intel Corporation. 2004. *IA-32 Intel architecture software developer's manual*. Intel Corporation. Available at http://developer.intel.com/.

Larson, E.; S. Chatterjee and T. Austin. 2001. "MASE: A novel infrastructure for detailed microarchitectural modeling". In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 1–9.

Magnusson, P. S.; M. Christensson; J. Eskilson; D. Forsgren; G. Hållberg; J. Högberg; F. Larsson; A. Moestedt and B. Werner. February 2002. "Simics: A full system simulation platform". *IEEE Computer 35*(2), 50–58.

Mukherjee, S. S.; S. K. Reinhardt; B. Falsafi; M. Litzkow; M. D. Hill; D. A. Wood; S. Huss-Lederman and J. R. Larus. October 2000. "Wisconsin wind tunnel II: A fast, portable parallel architecture simulator". *IEEE Concurrency 8*(4), 12–20.

Pai, V. S.; P. Ranganathan and S. V. Adve. 1997a. "RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors". In *Proceedings of the Third Workshop on Computer Architecture Education*. Also appears in IEEE TCCA Newsletter, October 1997.

Pai, V. S.; P. Ranganathan and S. V. Adve. 1997b. "RSIM reference manual. version 1.0". Technical Report 9705, Electrical and Computer Engineering Department, Rice University.

Rosenblum, M.; E. Bugnion; S. Devine and S. Herrod. January 1997. "Using the SimOS machine simulator to study complex computer systems". *ACM Transactions on Modeling and Computer Simulation 7*(1), 78–103.

Schaelicke, L. and M. Parker. 2002. "ML-RSIM reference manual". Technical Report 02-10, Department of Computer Science and Engineering, University of Notre Dame.

Thain, D.; T. Tannenbaum and M. Livny. 2004. “Distributed computing in practice: The condor experience”. *Concurrency and Computation: Practice and Experience 17*, 323–356.

Zhang, L.. 2001. “URSIM reference manual”. Technical Report UUCS-00-015, University of Utah.

AUTHOR BIOGRAPHIES

RICARDO FERNÁNDEZ PASCUAL was born in Madrid, Spain and went to the University of Murcia (Spain), where he studied Ingeniería Informática and obtained his master degree in 2004. He is currently researcher in the Computer Engineering Department at the Universidad de Murcia. His main research interests are cc-NUMA architectures, fault tolerance and performance simulation. His e-mail address is `r.fernandez@ditec.um.es`.

JOSÉ M. GARCÍA received the MS and the PhD degrees in electrical engineering from the Technical University of Valencia (Spain), in 1987 and 1991, respectively. Currently, Dr. García is a professor in the Computer Engineering Department at the Universidad de Murcia (Spain), and also the head of the research group on parallel computing architecture. He specializes in computer architecture, parallel processing, and interconnection networks. Dr. García served as vicedean of the School of Computer Science from 1995 to 1997, and also as director of the Computer Engineering Department from 1998 to 2004. His current research interests lie in high-performance coherence protocols for shared-memory multiprocessor systems, and high-speed interconnection networks. Dr. García is member of several international associations such as the IEEE and ACM, and also member of some European associations (Euromicro and ATI). His e-mail address is `jmgarcia@ditec.um.es`.