Distributed Web-based Platform for Computer Architecture Simulation

Aleksandar Ilic Frederico Pratas Leonel Sousa

INESC-ID/IST TULisbon
Rua Alves Redol, 9
1000-029 Lisboa, Portugal
E-mail: {ilic,fcpp,las}@inesc-id.pt

Abstract

Computer architecture simulation and modeling require a huge amount of time and resources, not only for the simulation itself but also regarding the configuration and submission procedures. A quite common simulation toolset (SimpleScalar) has been used to model a variety of platforms ranging from simple unpipelined processors to detailed dynamically scheduled microarchitectures with multiple-level memory hierarchies. In this paper we propose a platform for automatically executing a massive number of simulations in parallel, by exploiting a distributed computing approach. We developed a Web-based simulation system consisting in a front-end user interface and a back-end part supported on a Grid system. The front-end is responsible for configuring the simulation and parsing the results, while the back-end distributes the workload by using Condor scheduler. Experimental results show that it is very easy to use the system, even when dealing with a huge number of simulations, and also it provides results in a very suitable format. Moreover, it has been concluded that a significant speedup can be achieved, by exploiting parallelism at the benchmark levels or also by sampling each benchmark with the SimPoint tool.

1. Introduction and Motivation

To accelerate hardware development, designers often employ software models of the hardware they build. Usually these models are implemented in traditional programming languages, such as C, and exercised with the appropriate workloads. Even if simulators execute slower than real hardware, these permit designers not to wait for the several months needed to build real hardware. Besides, the high flexibility of the software permits to modify and test in detail diverse parameters of the model by doing several simulations. In hardware this could not be done due to the inherent restrictions, including time and cost.

In the Computer Architecture field the *SimpleScalar* (SS) toolset [2] provides an infrastructure for simulation and architectural modeling. This toolset can model a variety of platforms ranging from simple unpipelined processors to detailed dynamically scheduled microarchitectures with multiple-level memory hierarchies [1]. The SS simulator is a suite of powerful execution-driven computer simulation tools. For example in 2000 more than one-third of all papers published in top computer architecture conferences used the referred toolset [1] to evaluate their designs.

As stated above, simulators execute slower than real hardware, namely the SS out-of-order superscalar simulator; by using traditional technology, can run target programs with about a 4,000 times slowdown [11], i.e., a target program taking 15 minutes to execute in one machine would take about 1000 hours to execute using SS in the same machine. Moreover, since the simulation space is multidimensional, a large amount of simulations is required to assess the effect of different parameters of the microarchitectures.

Trying to mitigate prohibitively slow simulation speeds some methods where proposed, some more accurate than others, namely: abbreviated instruction execution streams of benchmarks as representative workloads; fewer or smaller input sets in commonly used benchmarks [5,9]; statistical simulation sampling [3, 6, 8, 14]; and profile-driven simulation sampling [4, 7].

Thus, unlike previous works using the referred methods, such as [13], this paper proposes a distributed computing platform that permits to automatically execute a large number of simulations in parallel, reducing the absolute simulation time. Besides, the platform can also combine one of the techniques previously referred. Without loss of generality herein we use *SimPoint* (SP) [4], which is a profile-driven simulation sampling tool. Combining SP with the distributed platform permits to control, and to increase the granularity of each simulation. It allows also to increase the degree of parallelism, because the simulations are divided in several traces.

The proposed platform, that uses the master-worker paradigm, is mainly supported by two different components: a front-end with a Web-based interface and a simulation manager supported by a database which allows the

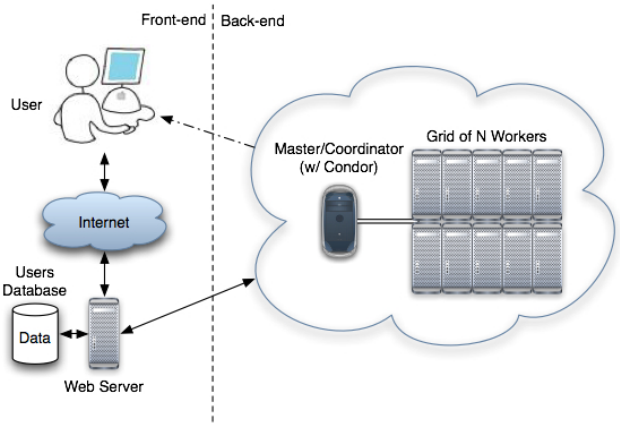


Figure 1. Proposed platform architecture

user to configure, submit and check the status of the simulations; a back-end with a Grid of computers running the simulations (*Workers*), and a scheduler to manage a queue of requested simulations through the Grid system (*Master*). The Web-based interface hides from the user the details of the distributed system for the Computer Architecture simulation. Moreover, it permits to easily select a set of characteristics to be tested, automatically generating all the required simulation scripts. The set of simulations is then scheduled to run in the Grid system using Condor [12], which is a high-throughput distributed batch computing system. With the proposed platform the user is able to execute in parallel in heterogeneous computers several simulations of different SS architectures. It allows the user to easily compare performance results obtained for several configurations of SS architectures, by gathering the results in a database. A set of simulations were performed using the proposed platform and a set of commonly used benchmarks, namely the SPEC CPU2000 [5]. The results obtained were used to evaluate the proposed distributed platform.

The remainder of this paper is organized as follows. Section 2 presents the detailed architecture of the proposed platform. Section 3 describes the implementation details of the Web-based interface. Section 4 presents the evaluation details, namely the setup configuration and relevant details about the case study. Finally, the evaluation results are presented in Section 5 and Section 6 concludes this paper.

2. Platform Architecture

As stated before, the architecture of the proposed platform can be divided in two different parts (shown in Figure 1): the user interface supported by a database (front-end); and a scheduler associated with the Grid of computers (back-end). The following sections describe the referred two parts.

It is worth to note the distinction between the two kinds of *Instruction Set Architectures* (ISA) referred along this paper. The ISA of the Workers, where the simulations are executed, which are designated as *Grid Computer Architectures* (GCA); and the ISA of the different SS simulators, i.e., the simulated ISA, designated as *Simulated Computer Architectures* (SCA). Another important aspect that should be noticed at this point is the distinction between *simulation* or *test-benches* and *job* or *process*. The former two designations are used when referring to the complete program associated to a performance test, i.e., the program executable or the trace ".eio" file; the latter ones refer to the execution of samples, i.e., the blocks of code that are executed in the Grid system. Moreover, a group of *jobs* buildup a *simulation*, and both concepts have the same meaning whenever the *simulation* is not sampled.

2.1. Front-end

The Front-end of the platform supports multi-user operations and is responsible for two main features: it allows the user to configure the simulations to be performed, and it manages the communications with Condor, hosted in the Master/Coordinator machine. An application was created to support this two aspects using PHP and MySQL.

The Web-based interface can be seen as the platforms core, because it simplifies the communication between the user and the system. It allows users to setup all the parameters required to perform simulations, even when they are unfamiliar with the simulator details or the rest of the system. Besides, the interface simplifies the generation and management of a large number of jobs, when using different test-benches, parameters and/or architectures. Moreover, the Web-based interface is associated with a database to support the addition of multiple users, different architectures to simulate, arguments and test-benches. Although this interface simplifies the utilization of the simulator, its design is flexible enough to allow advanced users to upload and exploit their own architectures and test-benches; this is explained in more detail in Section 3.

A database was created to allow a multi-user application, i.e., several users can access to the platform to configure and submit a set of simulations at the same time. This database stores information about user accounts and associates to each user his own architectures, tests and/or parameter preferences. It is also used to store the results of each simulation, allowing the user to have access to them later and/or to compare them with upcoming simulations.

When a user requests for a set of simulations, the application automatically generates the correct arguments to be used within the simulator and creates a group of jobs to perform a Condor request, according to the architecture and test-benches used. It associates the respective request with the user so that the results can be correctly handled later on. This is an important aspect because it allows the user to send at once to the Grid system a very large number of jobs as explained in Section 3.

The jobs results are gathered by the application as-they-

are and parsed to construct intuitive result tables and charts for the user. From the user's point of view, this step facilitates the analysis of the results, e.g. it allows to compare the results between several simulations. Moreover, if the simulations were sampled using the profile-driven tool the application is able to parse automatically the several job results and return the global simulation result to the user.

2.2. Back-end

As it was explained, the PHP application automatically generates the Condor files used in the job submission process. The submitted Condor files contain the information about the jobs to be performed, such as the type of SCAs to use and the parameters, and also the information of the GCAs that Condor can use. The identification of the type of GCA and operating system where each SCA can be run is obtained from the database. After performing the jobs submission the application keeps the information related with each request to allow the users to check the status of their current processes and pick up the respective results as soon as they are completed.

On the side of Condor, the files containing the requested jobs are parsed and the processes are scheduled according to the constraints requested (types of GCAs) and the available resources (Workers). During the execution process, the PHP application access to the Condor queue and/or history files to collect the status of each job as explained in section 3.1.2. Condor is also responsible for notifying the users when a certain job is finished. When the PHP application detects a completed job it retrieves the respective execution times from Condor and transfers the results to a local area where they are parsed, becoming available to the user. If a simulation is removed, the application stops all the corresponding jobs in Condor. The Condor results consist on a ".out" file containing the output of the *stdout* stream, an ".err" file with the output of the *stderr* stream and a ".log" file containing profile information, such as the time of execution. The actual results of SS are written in the *stderr* stream, so the proposed application parse the results directly from the ".err" file.

It is also worth to mention that the use of different GCAs could be seen as a limitation to the proposed platform, because the default SCA used is compiled for x86 machines. Nevertheless, our platform is robust enough to overcome this issue; the fact that the proposed platform allows the installation of different SCAs can also be used as a way to install several versions of the same SCA but compiled for different GCAs. This allows the user to run jobs in different types of GCAs. Above all, the proposed platform is fully portable and can be used in any Grid system, considering that each SCA is properly compiled for each GCA.

3. Web-based Interface

The Web-based interface is directly related to the database which allows users, for example, to generate simulations and access to the stored information. Two types of users are considered to access the proposed platform: *simulation users* (Us) and *administrators* (Ad). Each of these types has access to an individual interface, the Us interface and the Ad interface, respectively. Both types of interfaces can be divided in two parts: the users data management and the simulations management. The following two sections describe the interface from the point of view of each type of user.

3.1. Simulation Users Interface Design

3.1.1 Users Management

In order to manage the multi-user platform, a system of user accounts was created along with the simulations interface. The user accounts allow the application to store data related with each specific user in a convenient way, not only simple information such as e-mail address or identification, but also information related with the requested simulations, the type of tests uploaded and the obtained results.

As in any platform with user accounts, it is required for each Us to register himself. The information provided during this step is used by the system in several operations, e.g. to send notifications to the Us.

After succeeding with the registration process the Us can login and access to a front menu as the one shown in Figure 2. Options such as "Home", "Account Details", "Change Password", "Contact", and "Logout" allow the Us to access generic information and modify the account details, change the password or logout. The remaining options allow the Us to manage the simulations as explained in the following section.

3.1.2 Simulations Management

As referred before, the proposed platform confers a simulation management area for each user. In this area, users can configure simulations and results format: *i)* the simulation configuration options, namely "My Architecture", "My Simulations", and "Submit Job(s)"; *ii)* the result parsing option "My Submitted Jobs". Each of these options is described in detail in the following subsections.

My Architectures

The purpose of this option is to allow the Us to automatically install and manage his own SCAs. However, due to security reasons, when an Us wants to insert a new SCA a request is sent to the Ad, who is responsible for installing it. The Us request contains the SS file (containing the target SCA), as well as other related information. Once a new

Figure 2. Users menu

SCA is available, it can be used in new simulation configurations. Moreover, the Us can also remove his own SCAs; this action causes Condor to stop processes which are using the respective SCA.

My Arguments

This option allows the Us to add new arguments to each user defined SCA, i.e., the user is allowed to add arguments which are different from the SS defaults. There are two types of arguments: *i)* configuration arguments, which are used to define the parameters of each SCA, e.g. the size of the memory; *ii)* and results arguments, which are used to define new entries in the SS output, e.g. total number of instructions executed.

Both types of user defined arguments are associated with one specific SCA and are only used when this is selected. According to the type of argument, the Us adds a new argument by providing the following information: *i)* associated SCA, name, default value, type (int, float, string), and according to the type the Us needs to provide minimum, maximum and step values if is numeric, or all the possible options in the case of a string type; *ii)* associated SCA, name and at last if the new argument depends or not on the weights, in order to allow the tool to correctly parse the results.

The information requested for the first type of arguments is used by the application to create the correct forms when the Us is configuring new simulations ("Submit Job(s)" option). Regarding the second type, the information provided in this phase allows the application to correctly parse the output of simplescalar when presenting the results ("My Submitted Jobs" option).

My Simulations

In this configuration option the Us can add or remove his own test-benches. The interface provides two different methods to add new tests: manually or in a SP style. In both cases the Us has to provide the executable ".out" or the trace ".eio" files. When using the manual style the Us is able to define samples for each test-bench by manually introducing constraints for the number of instructions to skip, number of instructions to execute, and weights associated with each sample.

When using the SP style, the Us does not need to insert the constraints manually, instead he only has to provide the samples and weights files of SP and the block size used. If no samples are specified, by default, the system assumes that the new test-bench consists of only one job.

In the manual method the application automatically parses the data inserted by the Us and generates two SP style files, one with the weights and the other with the samples. This technique reduces the complexity of the application, because the way of parsing the jobs is the same for every test-bench.

Submit Job(s)

The procedure to submit new jobs depends on a chain of steps that allow the Us to select several simulation parameters, as described bellow

- in the first step, the Us selects the SCAs from two lists: the first one lists the available global SCAs (available for all the users), the second one allows the Us to select between his own SCAs, previously installed using the option "My Architectures"; in this step several SCAs can be selected to be tested at the same time;
- steps 2, 3 and 4 allow the Us to configure the default arguments for the processors core, the memory hierarchy and the branch prediction respectively, i.e., it allows to configure the default parameters of the previously selected SCAs; each attribute of SS is defined according to [2]: as shown in the example of Figure 3 the Us can select a value for each attribute - "Single value" - or a range of values to be tested - "Specify range". This is useful for instance if the Us wants to assess the effects of one parameter in the SCA without the need to submit several individual simulations; some attributes are not numerical, these attributes are specified in a multi-option style, the Us can also select several options at the same time if required, this situation is also exemplified in Figure 3 for the "Replacement policy" parameter; the application automatically combines the different selected attributes generating all the possible combination of simulations;
- the Us is only prompted with the fifth step if any of the SCAs chosen during the first step has user defined arguments of the first type, previously defined using op-

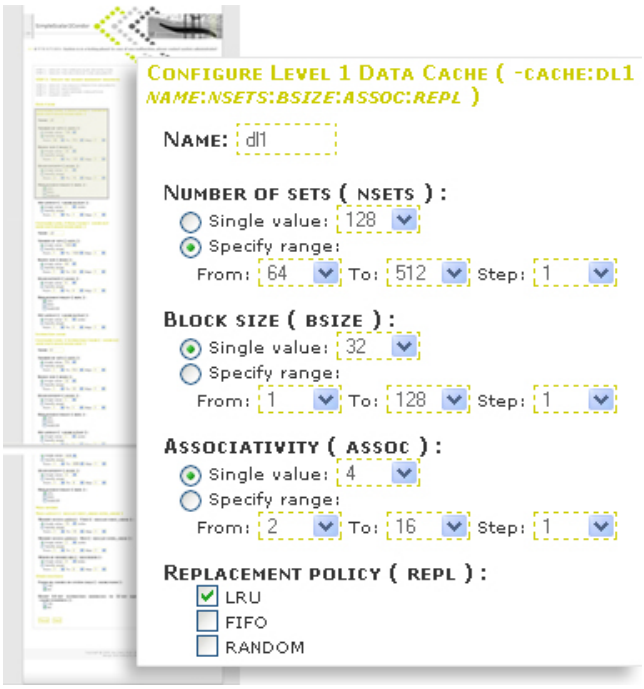


Figure 3. Example of how to configure simulation parameters

tion "My Arguments"; this arguments are only combined with the configurations of the associated SCA, so if the user selects multiple architectures he has to configure the user defined arguments for each of them during this step;

- finally, the last two steps are used to select the test-benches to run in the previously configured SCAs; the sixth step lists the test-benches available globally, while the seventh allows the Us to select from his own simulations, previously installed with the option "My Simulations"; the chosen simulations are executed in all the SCAs previously selected and configured.

At last, according to the selections performed in each step the application automatically generates all the possible combinations of the architecture parameters, which are then combined with the samples of every test-bench. Moreover, the resultant jobs are combined with each SCA and used to create the Condor files, which are transferred to the Grid system and submitted to Condor for execution. Besides, during this last phase the application fills the database with the information about each simulation and job, and the Us is redirected to the option "My Submitted Jobs", where he can check the status of the submitted jobs.

My Submitted Jobs

The jobs stored in the database and previously submitted by the Us, using the "Submit Job(s)" option, are accessible

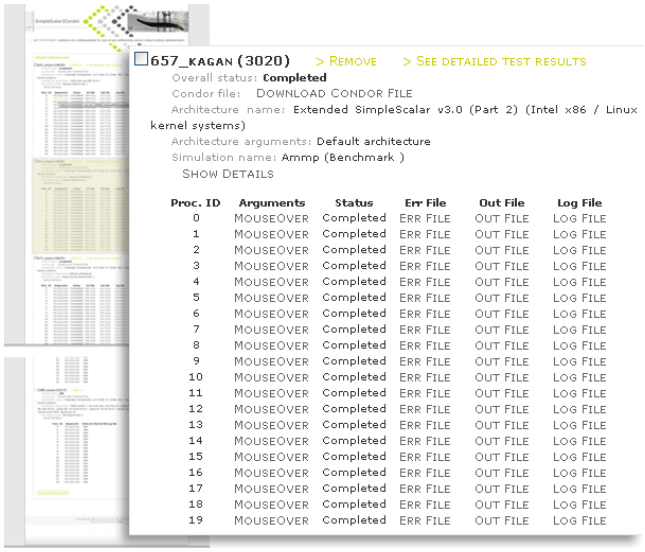
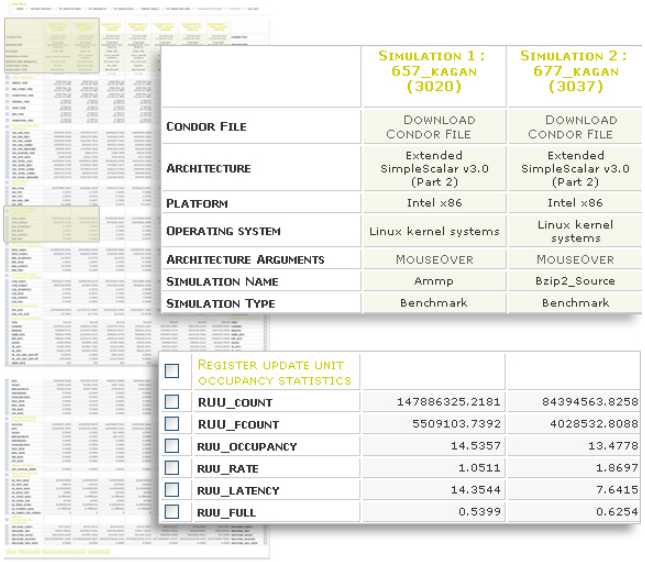


Figure 4. "My Submitted Jobs" layout

in this part of the system. Specifically, it shows to the Us all the information related to each submitted simulation, as shown in the example of Figure 4 for the *ammp* simulation. This information includes the architecture parameters, the SCAs, benchmarks, the overall status of the simulation and of the jobs that it includes. It is also possible for the Us to download the Condor file with the simulation script at any time. Moreover, this part of the system is responsible for managing the simulations as well as all the Condor related operations.

Each job has a status depending on its progress and the overall status of a simulation depends on the state of the jobs that compose it. Besides the status directly related to Condor, namely: *Idle, Running, Done, Unexpanded, Held* an additional set of status was defined, associated with a group of actions, to correctly manage the simulations. The referred status are divided in two types: *i) transitional* status that allow the jobs to perform actions for a undetermined period of time, such as: *Waiting, Submitting, Transferring and Completing*; and *ii) terminal* status which are used by the simulation manager to invoke predefined actions, e.g. *Submitted, Transfered and Completed*. As an example, when a job arrives to the "Completing" state the application connects to Condor and transfers the ".out", ".err", and ".log" files with the results to a local area, becoming available for the Us. When the simulation is finished and arrives to the "Completed" status the Us is allowed to access to the results parsed from the SS files using the Web interface.

The proposed application is set to automatically analyze the SS outputs, by creating tables with the raw results of each job. Due to the sampling method used, each job has a different impact in the overall results of the associated simulation, which is characterized by the weights provided during the test-benches submission. In order to simplify the



	Simulation 1 : 657_kagan (3020)	Simulation 2 : 677_kagan (3037)
Condor File	Download Condor File	Download Condor File
Architecture	Extended SimpleScalar v3.0 (Part 2)	Extended SimpleScalar v3.0 (Part 2)
Platform	Intel x86	Intel x86
Operating System	Linux kernel systems	Linux kernel systems
Architecture Arguments	MouseOver	MouseOver
Simulation Name	Ammp	Bzip2_Source
Simulation Type	Benchmark	Benchmark

Register Update Unit Occupancy Statistics		
RUU_count	147886325.2181	84394563.8258
RUU_fcount	5509103.7392	4028532.8088
RUU_occupancy	14.5357	13.4778
RUU_rate	1.0511	1.8697
RUU_latency	14.3544	7.6415
RUU_full	0.5399	0.6254

Figure 5. Comparable simulation results

interpretation of the results by the user, the PHP application uses the weights to generate the partial results for each job and assembles these into an overall simulation result.

There are two ways of previewing the referred results, per simulation or multi-simulation: in the former case the Us selects one simulation and has access to the overall result, to the results of each associated job (raw and partial), as well as the respective time statistics; in the latter case it presents, side-by-side, the global results of a preselected group of completed simulations.

In both cases the results are presented in a user friendly manner: the arguments are grouped by types and the Us can select only the ones he needs to check. If any SCA has associated user defined arguments, the application presents an additional group with this results. The results are shown in a table format and are easily comparable, as it can be observed in the example for two simulations provided in Figure 5. Moreover, in the multi-simulation preview, the selected parameters are also shown in a chart format.

3.2. Administrator User Interface Design

The Ad interface was built with the purpose to maintain the correct functionality of the system, by allowing the administrators to analyze and modify parameters that are crucial for it. The major available procedures are referred in the following sections.

3.2.1 Users Management

It is possible for the administrator to access important information related with the user accounts, namely information that allows the Ad to contact the users. Moreover, the Ad is able to restrict the access of Us's or send warnings, for example, in the case of an abuse that compromises the system's functionality.

3.2.2 Simulations Management

The Ad is not only responsible for the users but also for all the simulation environment. Furthermore, there is a group of basic actions associated with the Ad, namely: editing, disabling or even deleting user defined SCAs, arguments or test-benches, which were previously defined by the Us's with the respective options "My Architectures", "My Arguments" and "My Simulations". This is specially useful if the Ad detects a conflict situation in Condor due to possible errors in the simulations. In order to detect such events the Ad is also able to inspect the overall status of the Condor, i.e., he can track the jobs submitted by any user.

Moreover, the Ad responsibility is to install the new SCAs and to decide which SCAs and test-benches should be available globally or only locally for a specific user. The installed SCAs became available to the Us only when the Ad fills a specific permission page with the installation parameters.

The above cited options have a major influence in the stability of the proposed application, because these are specified by the users.

4. Evaluation of the Platform

The following section (Section 4.1) describes the hardware and software resources used in this work to support the proposed platform. Section 4.2 presents the preferences and parameters used in the simulations performed to evaluate the platform.

4.1. Execution Environment

The hardware and software resources presented in Table 1 were used to experimentally assess the proposed platform. As explained in Section 2.1, the front-end of this platform requires the use of a web-server incorporating the Apache, PHP and MySQL tools, while the back-end is mainly supported by the Condor scheduler and the distributed computing system. The Grid system consists on a group of 14 machines, each with a Pentium 4, 3.2GHz Intel x86 processor and the Linux Operating System.

4.2. Simulation Environment

Given that different microarchitectures can be incorporated in the proposed platform, we used the original SimpleScalar v3.0 microarchitecture to obtain the results presented in Section 5. Two configurations were considered during the simulations: one uses the SS default parameters, while in the second some parameters were modified to simulate a more accurate modern microarchitecture. Both configurations are depicted in Table 2.

Table 1. Execution Environment Attributes

Hardware Resources	
Number of Workers	14
Number of CPUs per Worker	1
Available memory per Worker	1GB
Swap per Worker	4GB
Machine type	x86
CPU speed	3,207 MHz
Software Resources	
Condor	v.6.9.3 June 12, 2007
PHP	v.5.2.5
Apache	v.2.2.4 Linux/SUSE
MySQL	v.5.0.45

Table 2. Configuration overview

SimpleScalar Parameters default (modified)	
BTB	512 sets, 4 way associativity
Branch Predictor	Bimodal: 1024, 8 entries
(Branch Predictor)	Combined Predictor Meta-table: 4096 RAS: 32 entries
Branch misprediction penalty	3(2) cycles
L1 I-cache L1 D-cache L2 unified cache	512 sets, 32B blocks, 1 way 128(256) sets, 32B blocks, 4(8) way 1024 sets, 64B blocks, 4(8) way
TLB	ITLB: 16 sets, 4-way DTLB: 32 sets, 4-way 4KB page size
Fetch/Decode/Issue and Commit Width	4 instructions
LSQ size	8(64) entries
RUU size	16(variable) entries

The workload selected to be used with the simplescalar tool was a set of 10 benchmarks from the SPEC CPU2000 suite [5], since these are widely used in general purpose computer benchmarking and also because they represent real applications. The SPEC CPU2000 is divided in two components, Integer and Floating Point, the *bzip, gcc, gzip, mcf, parser, twolf, vortex* and *vpr* programs were selected from the Integer while the *equake* and *ammp* are from the Floating Point. The actual number of simulations created is 22 because the benchmarks *bzip, gcc, gzip, vortex* and *vpr* were used with different input configurations, respectively *bzip2_source, bzip2_graphic, bzip2_program, gcc_166, gcc_200, gcc_expr, gcc_integrate, gcc_scilab, gxip_source, gzip_log, gzip_graphic, gzip_program, vortex_zero, vortex_one, vortex_two, vpr_place* and *vpr_route*.

According to the description in Section 1 the *SimPoint v3.0* tool [4] was used to create samples for each test with at most 30 samples of 10 millions of instructions each. The execution of the chosen tests, for one static set of parameters, corresponds to a group of 455 jobs.

The experimental results presented in this paper are part of the simulations that have been performed to produce the results obtained in our research work published in [10].

5. Experimental Results

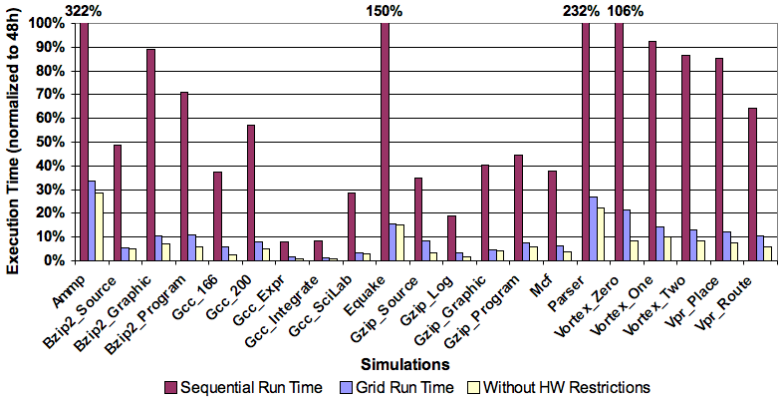
To show the major attributes of the proposed platform two types of simulations were performed and the results are analyzed in this section. The first type of simulations were executed to evaluate the execution time and it was performed using the default parameters of the SS architecture (see Table 2), having as inputs the 22 test-benches described in Section 4.2, consisting in a total of 455 jobs.

A second type of simulations was executed to reproduce a real situation in which an user wants to assess the effects of a parameter in an SCA. It was performed using the default SimpleScalar SCA with the modified parameters defined in Table 2, and the 22 test-benches described in Section 4.2. Without loss of generality, we decided to assess the effects of the *Register Update Unit (RUU)* size in the SCA. This was accomplished by choosing a variation of the RUU size parameter with power of two values in a range between 16 and 128. After automatically combining the referred preferences, the application generates and submits a total of 1820 jobs (4×455), and as a consequence the proposed application automatically parses the results gathered from 5460 Condor files.

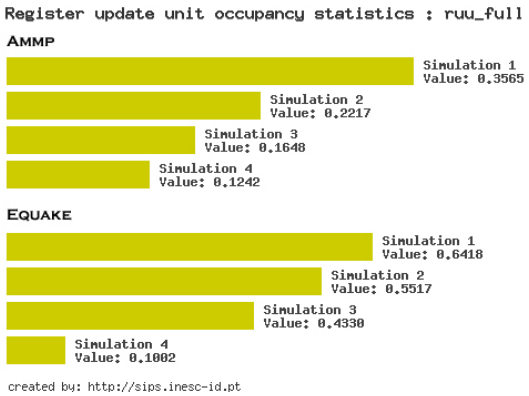
The results concerning the first simulations are presented in Figure 6(a). The chart presents three types of execution times, all of them are normalized to a period of 48h. The "Sequential Run Time" type shows the time that the simulation takes if it is executed only in one machine, i.e., if the jobs associated with the simulation are executed sequentially. The second type "Grid Run Time" displays the results obtained from the execution of all the 455 jobs in the environment defined in Table 1 (14 computers). Finally, the third type of results "Without HW Restrictions", represent the best results that could be obtained with the same environment but with an unrestricted number of computers.

The test-bench *twolf* is not displayed in the chart because the obtained results are meaningless, the execution time of this simulation was of about minutes. On the other hand, the sequential results for the *ammp, equake, parser* and *vortex_zero* test-benches are larger than 48h. The results presented show an average improvement in the execution time of 85%, which is near the 90% obtained in an environment without hardware restrictions.

Figure 6(b) presents two charts generated by the Web-based application as a result of the second simulations. Due to the lack of space, only the results of *ammp* and *equake* are presented. As expected, increasing the RUU size results in a reduction of the RUU occupancy.



(a) Execution times of each benchmark



(b) Interface results chart (Simulation 1, 2, 3 and 4 correspond to the RUU sizes 16, 32, 64, 128, respectively)

6. Conclusions

This work proposes a Web-based Computer Architecture simulation system, which simplifies the configuration and management of a large number of simulations, for instance when assessing the effects of changing parameters in microarchitectures. The proposed application has been used to get results for research works already published in the area of computer architecture, namely in [10]. It was proved that this tool is very flexible and easy to employ, allowing users with different experience level to completely configure and perform large scale simulations, without being concerned with the background execution system. Moreover, the application parses a large number of output files, calculates the results and presents them in an intuitive format.

The experimental results obtained using a pool of 14 machines have shown an average speedup up to about 10 comparing to the sequential execution. It is also important to note that the SimPoint tool can be used to increase the parallelism at the simulation level while exploiting the global parallelism provided by the Grid system itself. Nevertheless, the user should take into consideration that increasing the number of samples can affect the accuracy of the obtained results.

References

[1] T. Austin, E. Larson, and D. Ernst. Simplescalar: an infrastructure for computer system modeling. *Computer*, 35(2):59–67, Feb 2002.

[2] D. Burger and T. M. Austin. The simplescalar tool set, version 2.0. *SIGARCH Comput. Archit. News*, 25(3):13–25, 1997.

[3] T. Conte, M. Hirsch, and K. Menezes. Reducing state loss for effective trace sampling of superscalar processors. *Computer Design: VLSI in Computers and Processors, 1996. ICCD '96. Proceedings., 1996 IEEE International Conference on*, pages 468–477, 7-9 Oct 1996.

[4] G. Hamerly, E. Perelman, J. Lau, and B. Calder. Simpoint 3.0: Faster and more flexible program analysis. *The Journal of Instruction-Level Parallelism*, 7, Sep 2005.

[5] J. L. Henning. SPEC CPU2000: Measuring CPU Performance in the New Millennium. *Computer*, 33(7):28–35, 2000.

[6] J. W. H. Jr and K. Skadron. Minimal subset evaluation : Rapid warm-up for simulated hardware state. *iccd*, 00:0032, 2001.

[7] T. Lafage and A. Seznec. Choosing representative slices of program execution for microarchitecture simulations: a preliminary application to the data stream. pages 145–163, 2001.

[8] G. Lauterbach. Accelerating architectural simulation by parallel execution of trace samples. Technical report, Mountain View, CA, USA, 1993.

[9] C. Lee, M. Potkonjak, and W. Mangione-Smith. Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. *Microarchitecture, 1997. Proceedings. Thirtieth Annual IEEE/ACM International Symposium on*, pages 330–335, 1-3 Dec 1997.

[10] F. Pratas, G. N. Gaydadjiev, M. Berekovic, L. A. Sousa, and S. Kaxiras. Low power microarchitecture with instruction reuse. In *proceedings of Computing Frontiers 2008*, May 2008.

[11] E. Schnarr and J. R. Larus. Fast out-of-order processor simulation using memoization. *SIGOPS Oper. Syst. Rev.*, 32(5):283–294, 1998.

[12] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[13] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe. Simflex: Statistical sampling of computer system simulation. *IEEE Micro*, 26(4):18–31, 2006.

[14] R. Wunderlich, T. Wenisch, B. Falsafi, and J. Hoe. Smarts: accelerating microarchitecture simulation via rigorous statistical sampling. *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pages 84–95, 9-11 June 2003.