Evaluation of a Virtual Computer Architecture Lab

Bernhard Fechner; Jörg Keller; Wolfram Schiffmann

Computer Science Department
FernUniversität Hagen
{Bernhard.Fechner, Joerg.Keller, Wolfram.Schiffmann}@fernuni-hagen.de

Abstract

Virtual labs are excellent means to provide students with access to advanced microprocessor hardware. Running the lab, a remote server relieves students from complex installation of simulation software. With the long execution times of microprocessor simulations, a lab server provides results faster than the students' local machines. We report results about the practical use of our Virtual Computer Architecture Lab (VCAL) during a 36 day evaluation. A straightforward designed web interface simplifies the usage of the simulation software, thus allowing the student to concentrate on experiments and parameter settings. In the beginning, the students used the web interface more often, but they conducted experiments only towards the end. We conclude that students first learned to handle the web interface and used it afterwards. Although no student had previous experience with the SimpleScalar simulation software, they had no problems using the web interface, demonstrating the great potential of web interfaces to reduce usability problems compared to complex native user interfaces of simulators. Although only a subset of the simulator's parameter settings is available on the web interface, this still was sufficient for the subjects to be explored.. The students' rating in a questionnaire after the end of the evaluation phase showed acceptable simulation response times and improved understanding of the course content with the help of VCAL. Moreover, the web interface is simple enough to be used from mobile devices, thus improving mobile learning opportunities.

Keywords: Distance learning, evaluation, virtual laboratory, computer architecture

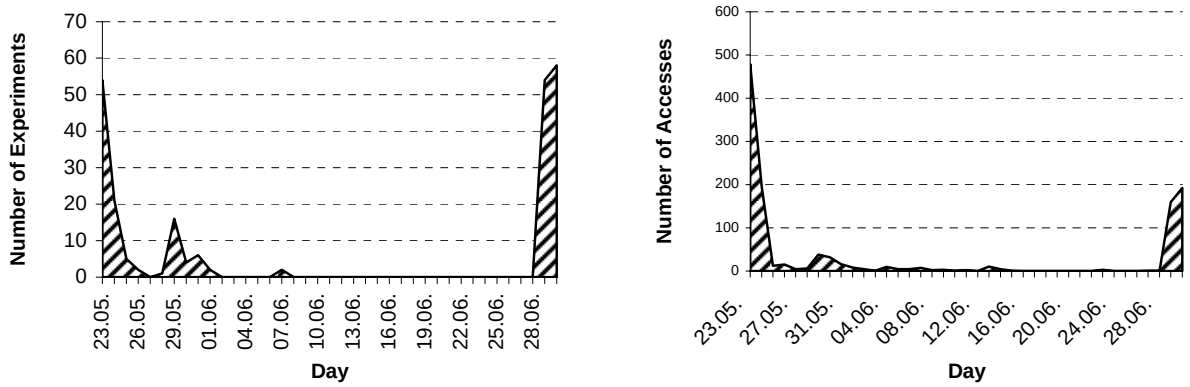
1 Introduction

Network-based multimedia and computer-related technologies improve and enhance the quality of teaching and learning for students. Colleges and universities, especially distance-learning institutions such as open universities are faced with a generation of students eager to learn more. Instructors are confronted with more students particularly at introductory levels. In Hagen, undergraduate courses in computer science serve a large amount of students (over 1000 in 2005) with various backgrounds and abilities. The challenge is to motivate and excite these students so that each performs to their fullest potential. This objective can be best achieved by complementing courses with a laboratory environment in a computer-related, web-based technology because students learn science best by experimenting and gaining hands-on experience, raising questions, solving problems and finding answers to questions through designing and carrying out experiments. In an effort over several years, we have enhanced the basic courses in computer engineering by such facilities [3]. The latest part of these efforts has been the introduction of a virtual computer architecture lab (VCAL) into the respective course, the core being the provision of a microprocessor simulator. While several processor simulation software packages are available for free, their support is restricted to a few operating systems, their installation cumbersome at the best, their usability difficult, and their hardware requirements (processing power, memory size) beyond the level that many students can provide. This strongly called for a remotely provided service with an easier-to-use interface. Last year,

we implemented such a service of the basis of the well-known SimpleScalar [1][2] package. The students can access a web page where they select the benchmark the simulator is going to run, and configure some processor parameters such as cache size and the like. We evaluated this service with the help of the students enrolled in the 2005 computer architecture course. The paper is organized as follows. Section 2 presents the evaluation results. Section 3 concludes the paper.

2 Evaluation Results

As a base for the simulation environment, we used SimpleScalar [1][2] which is used as a simulation tool in approximately 30% of all publications in computer architecture. Figure 1.a shows the number of actual experiments carried out per day on the lab. The length of the evaluation phase comprised 36 days. For comparison, Figure 1.b (right) shows the number of accesses on the webserver.



(a) (b)

Figure 1: Number of experiments/ accesses per day on the lab

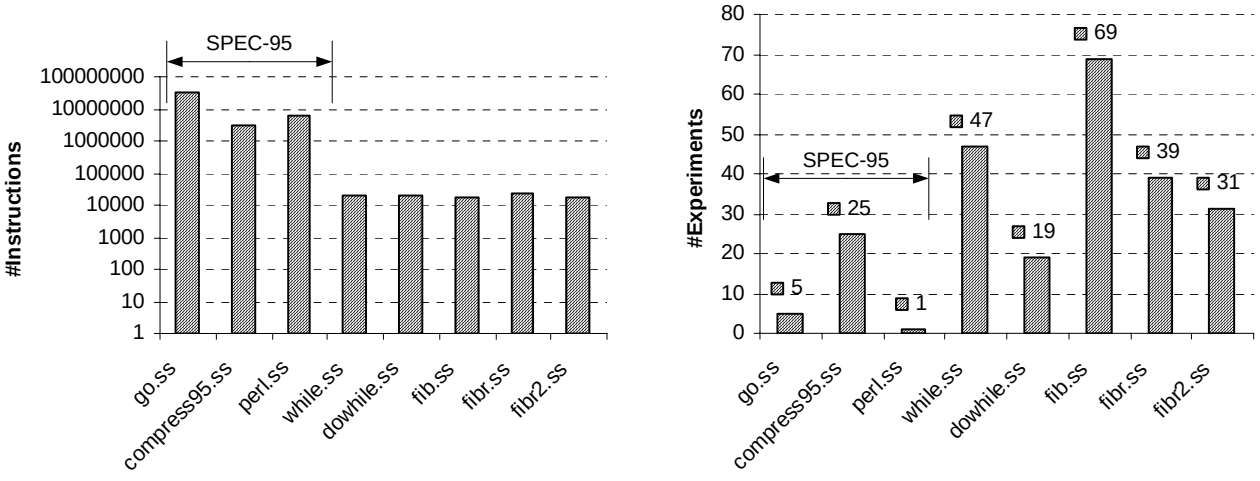
We notice a very high web-server access rate at the beginning of the evaluation phase. This shows that students were curious to see the web interface, but were not as much experimenting as at the end of the evaluation phase. Here we have a lower web server access rate than in the beginning, but a high number of experiments carried out. This means that students learned to manage the web interface and were not just taking a first look because of curiosity. We offered two main program categories as input for the simulator. The first part were three programs (go, compress9, perl) from the SPEC95 benchmark suite (CPU95 and CFP95)[1]. The other category was doing simple tasks like going through loops, which gives the students a better understanding of branch prediction and cache performance. Other programs calculated a set of fibonacci numbers, illustrating one iterative and two recursive methods. Table 1 shows the intended learning-effect and the function of each program.

[1] http://www.spec.org/cpu95/press.html

Table 1: Programs, their function and intended learning-effect

Program	Function	Learning effect
While.ss	Simple loop with condition checking at the beginning.	Branch Prediction/ Cache
Dowhile.ss	Simple loop with condition checking at the end.	Branch Prediction/ Cache
Fib.ss	Fibonacci numbers (iterative).	Return Address Stack, Branch Prediction
Fibr.ss	Fibonacci numbers (recursive) .	Return Address Stack, Branch Prediction
Fibr2.ss	Fibonacci numbers (recursive, out of Lucas).	Return Address Stack, Branch Prediction

Figure 2.a shows the number of instructions produced by each benchmark (experiment) the students could carry out, using a logarithmic scale. In this context the number of instructions is a measurement for the duration of an experiment. The more instructions are to be simulated, the longer the time to execute the simulation. In correlation with the number of experiments, shown in Figure 2.b, this indicates the preference of students to shorter over longer simulations. Furthermore, we included the source code of the programs from Table 1 on the web page, so students could figure out, what the function of the program was. For the SPEC95 benchmarks we could not include the source code, because of copyright reasons. One further conclusion from Figure 2.b is that simple programs, which the students could easily understand, gave a much higher impact on learning than the much more time consuming programs used in SPEC95.



(a) (b)

Figure 2: Load and usage of experiments

To get the students' opinions concerning the lab and their experience, the students could fill out a questionnaire in the evaluation phase on a voluntary basis. The rating was from 1 (very good) to 5 (did not match the requirements). The evaluation showed that no students did have any previous experience with SimpleScalar and but none had problems using the web interface, showing that there is a great potential for such interfaces, because they avoid installation problems, and simplify the access and usage. Furthermore, the command-line interface of SimpleScalar requires more than 20 options, each with a different meaning. Wrong usage of these parameters leads to a different result or even an error. The web interface omits this error prone usage. The students reported gaining good understanding of branch prediction strategies, pipelining and cache hierarchies because of the simulator (average rating of 1.75), which was

the intention of the course and the content of the exercises. Hence, although the web interface only allows to modify a few of the simulation parameters, the choice was sufficient for the goal in mind. Even though the underlying simulation environment was only a 400MHz Pentium II with 512 MB of main memory, the students judged the reaction and simulation times as good (average rating of 2.75). For higher student loads, a more powerful server could easily (and transparently for the users) replace the current one.

3 Conclusion

This paper presented the results of a 36 day evaluation phase of our virtual computer architecture lab VCAL. The students frequently used the simulator. They tended to use the simpler benchmark programs mainly because of the tighter correlation between cause and effect. The students reported the computer architecture lab as helpful for understanding the course contents. Hence, the restricted set of options that the web interface provides seems to be sufficient, while considerably simplifying the handling of the simulator. As a side effect, the straightforward design of the web interface makes access from wireless devices possible, since it uses standard HTML which is readable by most internet browsers running on modern mobile devices. In the beginning of the evaluation, the students were curious to see the web interface, but were not experimenting as much as at the end, meaning that students first learned how to handle the web interface and were not just taking a first look because of curiosity. Although no student had previous experience with the simulation software, they did not have problems using the web interface, showing that there is a great potential for such interfaces, because they reduce software problems, simplify the access and the use. Students tended to use the time-consuming SPEC-benchmark programs less than the simpler programs because they could see what the programs were actually doing by looking at the source code. The benchmark programs were precompiled versions, so the sources were not available.
Possible enhancements could be a set of predefined architectures, which could be selected at the beginning of the experiment. The set of architectures could imply commercial architecture setups, so that students could experiment with different processors actually available to the market, thus increasing the student's interest.

References

[1] D. Burger, T.M. Austin, The SimpleScalar Tool Set, Version 2.0, University of Wisconsin-Madison Computer Sciences Department Technical Report #1342, June 1997.

[2] T. Austin, E. Larson, D. Ernst. "SimpleScalar: An Infrastructure for Computer System Modeling", *IEEE Computer*, vol. 35, no. 2, pp. 59-67, February 2002.

[3] H. Bähring, J. Keller, W. Schiffmann. Web-Based Support for Computer Engineering. In *Proc. 5th Conference Virtual University: VU 2005*, Warzaw, June 2005.