A Network-Computing Infrastructure for Tool Experimentation Applied to Computer Architecture Education*

Renato J. Figueiredo†, José A. B. Fortes†, Rudolf Eigenmann†, Nirav Kapadia†
Sumalatha Adabala†, Jose Miguel-Alonso† ‡‡, Valerie Taylor††, Miron Livny† † †
Luis Vidal‡ and Jan-Jo Chen‡

†School of ECE
Purdue University

††Department of EECS
Northwestern University

‡Dept. of Mathematics and Computer Science
Chicago State University

‡‡Dept. of Computer Architecture and Technology
The University of the Basque Country

† † †Computer Sciences Department
University of Wisconsin-Madison

Abstract

Computer architects increasingly depend on the use of software tools to evaluate and investigate the design of computer systems. It is therefore very important that educators in this field promote extensive tool-based student experimentation in architecture classes. However, the integration of today's complex architecture tools into curricula poses several challenges to an instructor, including management of powerful computing resources, software installation and maintenance, and development of tool-specific educational material. This paper describes how these challenges are addressed by a universally accessible network-computing infrastructure - NETCARE - that provides educators with a Web portal to access computing resources, executable tools and educational material.

1 Introduction

As the complexity of computing systems continues to grow, computer-aided design (CAD) software has become an essential tool to computer architects in both industry and academia. Prospective employers expect graduating computer engineering students to be proficient in the usage (and development) of tools; therefore, computer architecture educators must promote extensive student experimentation with CAD tools in school.

Computer architecture tools, such as simulators and compilers, are complex and often demand powerful computing resources to deliver acceptable performance levels. Furthermore, tools tend to be tailored to specialized architecture aspects, such as micro-architecture, memory hierarchy, multiprocessors, operating systems and compilers. A typical undergraduate- or graduate-level architecture discipline covers many such topics, and is likely to require the availability of several tools for CAD experiments.

An educator faces many obstacles in the deployment of an architecture tool for experimentation in a class environment. They must (1) obtain access to hardware resources that meet a tool's system requirements, including student accounts; (2) install and maintain the tool (software and documentation); (3) develop education content (tutorials, assignments) to be used in class. If many tools are intended to be used throughout the semester, the overheads are magnified in proportion to the number of needed tools. Furthermore, it becomes important to (4) guarantee that tools are presented to students with user-friendly interfaces. While commercial tools often provide rich graphical user interfaces, many leading-edge architecture tools are developed as a result of research efforts and provide text-based interfaces that, while being efficient for expert users, are an additional overhead to a novice user.

At the authors' universities, several tools are being incorporated in courses by using an infrastructure for computer architecture education that leverages network-computing technology to provide a WWW portal to tools. The infrastructure is based on the Purdue University Network Computing Hubs [8] and is currently available as part of the NETwork-computer for Computer Architecture Research and Education (NETCARE), a three-university consortium consisting of Purdue University, Northwestern University and the

*This work was partially funded by the National Science Foundation under grants EIA-9975275 and EIA-9872516, by an academic reinvestment grant from Purdue University, and by the Comision Interministerial de Ciencia y Tecnologia, Spain (TIC98-1162-C02-02). Dr. Miguel-Alonso's stay at Purdue University is supported by the Secretaria de Estado de Universidades, Investigacion y Desarrollo, Spain.

University of Wisconsin.

The infrastructure provides access to large pools of heterogeneous hardware resources, promotes reusability of software installations, documentation and educational content, and provides standard web-based user interfaces. This paper describes how each of the above mentioned obstacles involved in the integration of tools into courses are addressed by the infrastructure, and reports on experiences of the use of the system for architecture education.

2 Underlying Hardware resources

Computer architecture software tools demand powerful computing resources. A single detailed cycle-accurate execution-driven architectural simulation of a microprocessor usually takes hours to complete in a high-end server or workstation, and often many data points are required to draw conclusions from an experiment. In addition, different tools often require different combinations of target architectures and operating systems. Providing access to plentiful computing resources to students adds to the 0 overhead of setting up tool-based experiments in architecture classes. It is also possible that the instructor's institution does not own the high-end or specialized hardware resources that are required to run the desired tools.

The network-computing infrastructure leveraged by NETCARE addresses this administrative overhead by providing (1) resource-management mechanisms for a set of execution servers, which may consist of heterogeneous machines distributed across administrative domains, and (2) access to these resources without the need to set up individual user accounts on each server.

The PUNCH infrastructure allows NETCARE users to access a large pool of computers via "shadow" accounts. Shadow accounts are conventional (e.g. Unix) user accounts that are created on behalf of PUNCH on an execution server when the server is added to the system's resource pool. After this initial setup, no individual user accounts need to be created on the server: a shadow account is allocated to a NETCARE user dynamically for each tool execution request, and deallocated upon completion. This mechanism provides immediate access to computing resources for new NETCARE users, and automatic access to newly added computing resources to the existing user base.

The resource-management subsystem of PUNCH incorporates load-balancing and usage prediction mechanisms [9], and provides access to cluster management software such as Condor [12]. This allows the system to provide access to a large number of hardware resources through a single entry point. Currently, the NETCARE infrastructure provides access to 5 dedicated servers at Purdue, and approximately 600 Condor nodes at Wisconsin and 100 at Purdue.

3 Available Computer Architecture Tools

Research performed in many universities and industries continues to provide public-domain and commercial tools for the design, evaluation and programming of computing systems. These tools provide valuable material for experimentation in architecture disciplines. However, due to their nature and complexity, such tools most often demand complex installation and maintenance procedures.

The NETCARE infrastructure addresses the tool installation, documentation and maintenance overheads by promoting reusability. The addition of a tool to NETCARE still may demand complex installation procedures; however, once installed, it automatically becomes available to users across different administrative domains. Furthermore, tool and/or documentation maintenance needs to be performed only for the single installation site.

The installation of a tool onto NETCARE involves two steps. First, the tool is installed onto suitable resources following its original installation procedures. NETCARE provides access to *unmodified* program binaries; hence, tools can be installed without requiring access to their source code or object files. The second step involves the creation of a Web user interface to the tool's execution and documentation. Section 5 describes how the underlying PUNCH architecture supports both native (graphical user-interface - GUI) and customized (HTML-based) interfaces to tools.

Many representative research tools in computer architecture and parallel programming have been installed onto NETCARE as part of this project. These tools, corresponding documentation, and educational material are being integrated into computer education curricula.

Table 1 shows the set of computer architecture and parallel programming tools currently available on NETCARE. For each tool, the table lists its user interface: for tools with native graphical/interactive user interfaces, NETCARE provides the original interface to the user through remote display technologies such as Virtual Network Computing (VNC [16]). For tools with text-based interfaces, NETCARE provides a customized interface based on HTML forms (Section 5).

Tool	User interface
CacheSim5 [4]	HTML forms
Dinero-IV	HTML forms
DLXView [6]	XWindow GUI
HPAM_Sim [1]	HTML forms
LSU Proteus [2]	HTML forms
RSIM [14]	HTML forms
Shade [4]	HTML forms
SimpleScalar [3]	HTML forms
WWT-II [13]	HTML forms
XSPIM [6]	XWindow GUI
MaxP [11]	HTML forms
Polaris [17]	HTML forms
Trimaran [7]	XWindow GUI
Ursa Minor [15]	XWindow GUI

Table 1: Computer architecture and parallel programming tools currently available on NETCARE. Tools in *italics* are currently integrated with Condor.

4 Educational content

In addition to providing access to hardware and software resources, integrating tools to existing computer

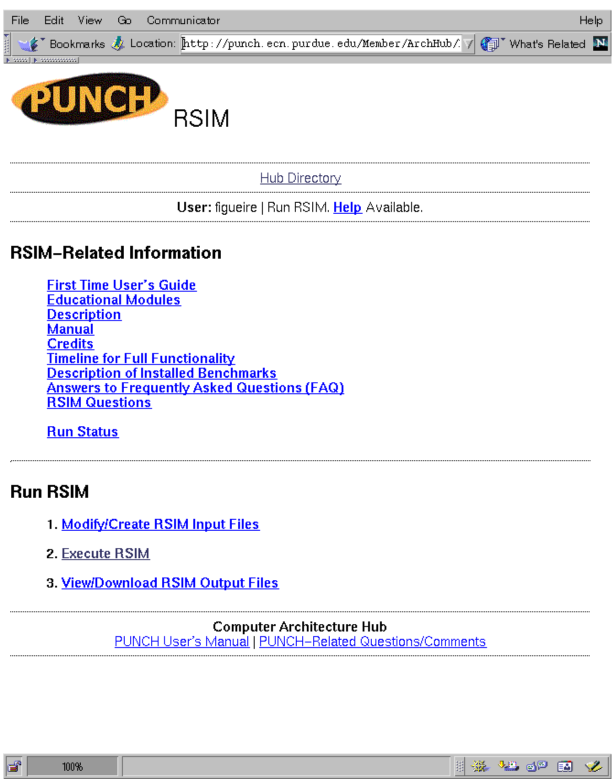

Figure 1: Access to RSIM's execution, documentation and educational content.

engineering classes requires the availability of extensive educational material. This includes manuals, tool documentation, answers to frequently-asked questions, examples, tutorials and homeworks.

Educational content in NETCARE is available as a collection of on-line documents accessible to students and instructors via the web-based interface (Figure 1), including:

1. *Original tool documentation.* All documentation that is provided in the tool's distribution package.
2. *Local tool documentation.* Includes description of extra features and/or limitations that are specific to the interface of the tool provided by NETCARE.
3. *Questions and Answers.* Answers to frequently-asked questions (FAQs).
4. *Educational modules and examples.* Include example files, first-time user's guide (tutorials introducing the tool's interface), and educational modules (tutorials introducing the tool's capabilities and functionality).
5. *Exercises.* Documents with tool-based experiments (exercises and open-ended projects) that can be assigned in classes.

Items 4 and 5 are designed to facilitate the integration of tools with computer engineering courses. Educational modules are introductory tutorials on the usage of a tool; their purpose is to guide the student through experiments that illustrate the tool's functionality. Educational modules are self-contained documents that allow students to become familiar with a tool by independently experimenting with it, and checking results and conclusions against the expected ones described in the module. Students are not expected to turn in assignments based on educational modules, but rather to familiarize themselves with the tool(s) that are used in exercises and course projects.

Exercises and open-ended projects are documents that are intended to be used in (graded) class assignments. They assume that students have acquired the necessary background knowledge of the subject and related tool(s) from class material and the educational modules, and typically involve more challenging experiments than the ones of introductory educational modules.

The structure of educational modules and exercises/projects is similar; they differ on the scope/complexity of the experiments and on the availability of results, answers and conclusions to students.

These documents contain five sections, as follows:

1. Introduction. Brief introduction to the problem being addressed by the module or exercise.

2. Experimental methods and models. It defines which aspects of the material discussed in the *Introduction* will be covered by the experiment (and are thus modeled by the tool used in the experiment), and explains the experimental methodology. In open-ended projects, the definition of methods and models may be left as part of the student's assignment.

3. Experiment setup. This section determines the model parameters that will be varied/observed by the student during the experiment and the specific tool options to be used. As in *Experimental methods and models*, the definition of these parameters may be left as part of the student's assignment for open-ended projects.

4. Analysis. This section provides guidelines for the analysis of the data gathered after the experiments have been conducted. The purpose of this section is to aid in the interpretation and understanding of the results obtained from the experiment. For educational modules, results, answers and conclusions are provided in this section to allow students to check their work. For exercises and projects, results and answers are provided in a separate document whose access is granted by PUNCH to registered instructors only.

5. References. This section contains bibliographical references to all published material that is cited throughout the entire document.

5 User interfaces

The PUNCH infrastructure leveraged by NETCARE currently supports two types of user interfaces: XWindow GUI and HTML forms. The former is used for tools that have native graphical interfaces, while the latter is used for text-based tools. Both interfaces are provided to NETCARE users via conventional Web browsers: the graphical XWindow interface is handled via a Java VNC [16] browser applet, while the text-based interface uses standard HTML language.

Many tools that can be used in computer architecture education are text-based (Table 1); often, these tools are configured via command-line parameters and files that are unfriendly to a novice user. PUNCH supports the definition of metaprograms that generate a dynamic HTML interface to these tools.

The metaprograms generate customized HTML interfaces and are specified via a high-level scripting language [10]. Metaprograms allow the translation of data obtained from the HTML interface (e.g. text boxes and radio buttons) into a format that is suitable for the tool's native interface (e.g. command-line parameters). Figure 2 shows an example of one of the dynamic HTML pages generated for the interface to the Dinero-IV simulator.

For a complex tool, the specification of metaprograms often require that its installers have good knowledge of its native interface and functionality, and hence requires extra programming effort in the process of web-enabling a tool. However, once a metaprogram is developed for a complex tool, it can be re-used for the installation of the tool in different sites and used as a basis for the installation of newer versions/variations of the tool, or tools that exhibit similar functionality. Furthermore, the interface provided by metaprograms allows tailoring of the interface's complexity based on the needs of target users. For instance, the NETCARE metaprogram for the cache simulator Dinero-IV provides interfaces to both novice and advanced users. The former presents only basic cache simulation pa-

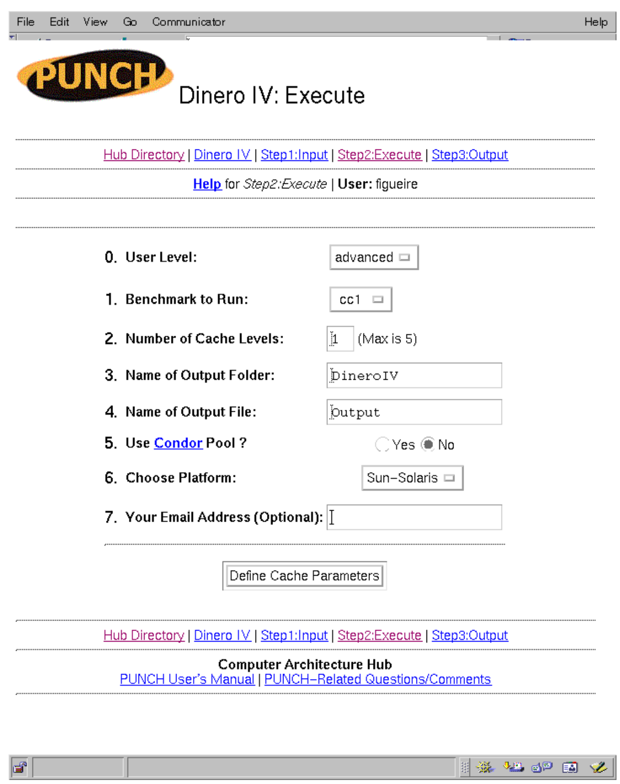

Figure 2: Entry-point HTML interface to execution of a Dinero-IV simulation.

rameters to the student (e.g. size and associativity), while the latter includes parameters that may not be covered in an introductory memory hierarchy class (e.g. prefetch distance and fetch policies).

The definition of metaprograms for text-based tools that have simple interfaces and for X-Windows GUI tools can be performed with little overhead by reusing existing templates. Current efforts in user interfaces include automatic generation of metaprograms for such types of tools.

6 Integration with curricula

The computer architecture and programming tools that are currently available via NETCARE have been used in several undergraduate and graduate classes across Purdue University, Northwestern University and Chicago State University since 1998. Tools have been used in experiments involving memory hierarchies (Cachesim5 and Dinero-IV), pipelining (DLXview), instruction sets and execution-based simulation techniques (Shade), parallel programming (Polaris and MaxP) and instruction-level parallelism (SimpleScalar) [5].

Feedback received from student evaluations indicates that the network-computing interface is easy to learn and an effective aid to understanding architecture concepts [5]. Most criticism was received from advanced users in graduate-level courses with regard to the HTML-based file manipulation; for this class of users, a Unix-like terminal interface can enhance productivity. We are currently looking into the integration of a shell interface to the network-computing infrastructure to address the needs of expert users that use tools for both research and education.

7 Using NETCARE

The educational infrastructure provided by NETCARE described in this paper is publicly accessible and available for use by educators, upon request. The system is accessible through any Web browser via the URL `http://www.ece.purdue.edu/NETCARE`. This entry point to the infrastructure allows educators and students to request accounts, access documentation, execute tools and contact the personnel involved in the project.

In addition to reusing the currently available infrastructure, NETCARE also allows educators to add and share tools and documents currently used in their courses to the infrastructure. New tools can be added via the the specification of metaprograms; templates for XWindow-based tools and installation guidelines for text-based tools are available for use.

8 Conclusions

A large percentage of computer engineering graduates will need experience with computer-based simulation and design tools in their jobs. This paper presents a network-computing infrastructure - NETCARE - that provides universal access to tools and simulation-based experimentation in computer architecture courses.

The paper describes the current status of NETCARE, and presents a summary of the experiences obtained from its application to architecture education. The use of the infrastructure in the universities participating in the consortium has shown that it is able to reduce the overheads in hardware, software and documentation management, facilitating the integration of tools into existing computer architecture courses.

References

[1] Ben-Miled, Z., Fortes, J.A.B., Eigenmann, R., and Taylor, V. A Simulation-based Cost-efficiency Study of Hierarchical Heterogeneous Machines for Compiler and Hand Parallelized Applications. *9th Int. Conf. on Par. and Dist. Computing and Systems*, Oct 1997.

[2] Brewer, E.A., Dellarocas, C.N., Colbrook, A., and Weihl, W.E. Proteus: A high-performance parallel architecture simulator. Technical Report mit/lcs/tr-516, Massachusetts Institute of Technology, Sep. 1991.

[3] Burger, D.C., Austin, T. M., and Bennett, S. Evaluating future microprocessors-the simplescalar tool set. Technical Report #1308, University of Wisconsin, Computer Science Dept., July 1996.

[4] Cmelik, B. and Keppel, D. Shade: A fast instruction-set simulator for execution profiling. In *Proceedings of the 1994 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 1994.

[5] Figueiredo, . J., Fortes, J. A. B., Eigenman, R., Kapadia, N., Taylor, V., Choudhary, A., Vidal, L., and Chen, J-J. On the use of simulation and parallelization tools in camputer architecture and programming courses. In *Proceedings of the American Society for Engineering Education Annual Conference*, Nune 2000.

[6] Hennessy, J. L. and Patterson, D. A. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, 1996.

[7] J. C. Gyllenhaal, W. -m. W. Hwu, and B. R. Rau. Hmdes version 2.0 specification. Technical Report Technical Report IMPACT-96-3, University of Illinois at Urbana-Champaign, 1996.

[8] Kapadia, N. H. and Fortes, J. A. B. PUNCH: An architecture for web-enabled wide-area network-computing. *Cluster Computing: The Journal of Networks, Software Tools and Applications*, 2(2):153–164, Sep. 1999. In special issue on High Performance Distributed Computing.

[9] Kapadia, N. H., Fortes, J. A. B., and Brodley, C. E. Predictive application-performance modeling in a computational grid environment. In *Proceedings of the 8^{th} International Symposium on High Performance Distributed Computing (HPDC'99)*, pages 47–54, Aug. 1999.

[10] Kapadia, N. H., Robertson, J. P., and Fortes, J. A. B. Interface issues in running computer architecture tools via the world wide web. In *Workshop on Computer Architecture Education (held in conjunction with the Intl. Symposium on Computer Architecture)*, 1998. Barcelona, Spain.

[11] Seon Wook Kim. MaxP: Maximum parallelism detection tool in loop-based programs. Technical Report ECE-HPCLab-99206, HPCLAB, Purdue University, School of Electrical and Comp uter Engineering, 1999.

[12] Litzkow, M., Livny, M., and Mutka, M. W. Condor - a hunter of idle workstations. In *Proceedings of the 8^{th} International Conference on Distributed Computing Systems*, pages 104–111, June 1988.

[13] Mukherjee, S. S., Reinhardt, S. K., Falsafi, B., Litzkow, M., Huss-Lederman, S., Hill, M. D., Larus, J. R., and Wood, D. A. Wisconsin Wind Tunnel II: A Fast and Portable Parallel Architecture Simulator. In *Workshop on Perf. Analysis and Its Impact on Design (PAID)*, June 1997.

[14] Pai, V. S., Ranganathan, P., and Adve, S. V. The impact of instruction-level parallelism on multiprocessor performance and simulation methodology. In *Proc. 3^{rd} International Symposium on High-Performance Computer Architecture*, Feb 1997.

[15] Insung Park, Michael J. Voss, Brian Armstrong, and Rudolf Eigenmann. Parallel programming and performance evaluation with the Ursa tool family. *International Journal of Parallel Programming*, 26(5):541–561, 1998.

[16] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, January-February 1998.

[17] W. Blume, R. Doallo, R. Eigenmann, J. Grout, J. Hoeflinger, T. Lawrence, J. Lee, D. Padua, Y. Paek, B. Pottenger, L. Rauchwerger and P. Tu. Parallel Programming with Polaris. *IEEE Computer*, Dec 1996.