Network-Computer for Computer Architecture Education: a Progress Report*

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

†School of ECE
Purdue University

††Department of ECE
Northwestern University

‡Dept. of Mathematics and Computer Science
Chicago State University

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

Abstract

Computer engineers 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 experimentation by students. However, the integration of today's complex tools into curricula poses several challenges to instructors. To deploy a tool-based class experiment, they must (1) obtain access to hardware resources that meet the requirements of the tool, and obtain access to student accounts on these resources; (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. This paper describes how these challenges are addressed by a universally accessible NETwork-computer for Computer Architecture Research and Education - NETCARE - that provides educators with a Web portal to access computing resources, executable tools and educational material. This infrastructure has been used in computer architecture and parallel programming classes for three years; the experience obtained from its extensive usage has motivated several improvements in the infrastructure which are described in this paper. In terms of hardware

*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.

resources, new interfaces to cluster management systems allow access to a large number of processing nodes of supercomputers and networks of workstations. In terms of content, new applications are available on NETCARE, including tools for explicitly-parallel instruction set architectures. In terms of user interfaces, a new specification language allows for increased flexibility in the generation of dynamic Web-based tool interfaces. These features are enabled by a new version (5.0) of the underlying PUNCH network-computing infrastructure.

1 Introduction

As the complexity of computing systems continues to grow, computer-aided design (CAD) software packages have become essential tools to computer architects in both industry and academia. Furthermore, prospective employers expect graduating computer engineering students to be proficient in the usage and development of tools. Therefore, computer architecture educators must promote extensive experimentation by students 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. These tools tend to be tailored to specialized architecture aspects: micro-architecture, memory hierarchy, multi-processors, operating systems and compilers, to name a few. A typical undergraduate- or graduate-level architecture discipline covers many such topics, and is likely to require the availability of several different tools for CAD experiments.

In this scenario, educators face many obstacles in the deployment of an architecture tool for experimentation in a class environment. They must (1) obtain access to hardware resources, including student accounts, that meet the system requirements of a tool; (2) install and maintain the tool (software and documentation); (3) develop education content (tutorials, assignments) to be used in class. 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 that are developed in research projects provide text-based interfaces. Such interfaces, while being efficient for expert users, are an additional overhead to a novice user. If many tools are intended to be used throughout the semester, the overheads are magnified by the number of needed tools.

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 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 [8]. It has been used in computer architecture and parallel programming classes for three years. The experience obtained from its extensive usage has motivated several improvements in the infrastructure which are described in this paper. This paper also reports on experiences of the use of the system for architecture education.

In terms of hardware resources, interfaces to cluster management systems, such as Condor [17] and PBS [2] (Portable Batch System) allow access to a large number of processing nodes of supercomputers and networks of workstations. Currently, a 272-processor IBM SP/2 supercomputer and approximately 1000 workstations at Purdue University and the University of Wisconsin are accessible via Condor and PBS. In terms of content, new applications are available on NETCARE, including tools for explicitly-parallel instruction set architectures, such as DAISY and Trimaran. In terms of user interfaces, a new specification language allows for increased flexibility in the generation of dynamic Web-based tool interfaces. These features are enabled by a new version (5.0) of the underlying PUNCH network-computing infrastructure.

2 Underlying resources

Computer architecture software tools demand powerful computing resources: a typical cycle-accurate, execution-driven architectural simulation of a microprocessor takes hours to complete on a high-end server or workstation. 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 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 underlying network-computing infrastructure used 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 logical user accounts. A logical account is a capability that allows a NETCARE user to "check out" a system account on appropriate computing resources. These system accounts belong to a pool of conventional (e.g. Unix) user accounts in an execution server. After an initial setup of the account pool, no individual user accounts need to be created on the server: a system account is allocated to a NETCARE logical account dynamically for each execution request, and de-allocated 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 [12].

The resource-management subsystem of PUNCH incorporates load-balancing and predictive performance modeling mechanisms [14], and provides access to cluster management software across multiple administrative domains.

Cluster management systems, e.g., Condor [17], and PBS [2], typically provide a command and/or library interface to interact with the system. PUNCH accesses clusters resources via this interface [1]. The interface varies from one cluster software to another, and can be divided into the following categories: 1) run-specific information (e.g., tool-input, user-preferences), 2) site-specific information based on the configuration of the cluster management system, and 3) application-specific information (e.g., platforms supported). PUNCH keeps track of and manages site- and application-specific information and obtains the run-specific information from the user at job-submission time. It submits the jobs and monitors them for completion using the appropriate syntax and commands to interface with the cluster resource.

Thus the user does not need to keep track of the available cluster resources, the associated cluster management software, and the corresponding submission syntax and user-commands. Moreover, as PUNCH interacts with the cluster management software at a user-level, no modifications are required to the interface they provide and PUNCH can access existing cluster resources. The ability to interface with cluster management systems transparently from users allows the system to provide access to a large number of hardware resources through a single entry point. Currently, a 272-processor IBM SP/2 supercomputer and approximately 1000 workstations at Purdue University and the University of Wisconsin are accessible via Condor and PBS through NETCARE.

3 Available tools

Research performed in academia and industry 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 and parallel programming disciplines. However, due to their complexity, such tools most often demand time-consuming installation and maintenance procedures.

The NETCARE infrastructure addresses the tool installation, documentation and maintenance overheads by promoting reusability. Once a tool is 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 that allows configuring and executing a tool, as well as browsing its 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 [22]). For tools with text-based interfaces, NETCARE provides a customized interface based on HTML forms (Section 5).

One recent addition to the set of applications available on NETCARE — DAISY — provides an example of how the infrastructure leverages user interfaces and resource management mechanisms to rapidly deliver applications over the Web. The DAISY simulator is a public-domain tool developed by IBM that is used to study dynamic compilation of programs into VLIW architectures [7]. The process of installing this tool onto NETCARE consists of (1) securing hardware resources, (2) installing the software, and (3) developing a user interface.

In terms of hardware resources (1), this application requires PowerPC machines for execution; NETCARE secures access to a large number of PowerPC nodes transparently from

Tool	**User interface**	Topic
CacheSim5 [6]	HTML forms	Caches
DAISY [7]	HTML forms	VLIW, dynamic translation
Dinero-IV	HTML forms	Cache
DLXView [10]	X Window GUI	Pipeline
HPAM_Sim [3]	HTML forms	Multiprocessors
LSU Proteus [4]	HTML forms	Multiprocessors
RSIM [19]	HTML forms	Multiprocessors
Shade [6]	HTML forms	Instruction set, tracing
SimpleScalar [5]	HTML forms	Microarchitecture
WWT-II [18]	HTML forms	Multiprocessors
XSPIM [10]	X Window GUI	Instruction set
Compex	HTML forms	Parallel execution
MaxP [16]	HTML forms	Parallelism profile
Polaris [23]	HTML forms	Parallelizing compilers
Trimaran [11]	X Window GUI	VLIW
Ursa Minor [20, 21]	X Window GUI	Performance optimization

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

its users via a PBS-based cluster interface to an SP/2 supercomputer. In terms of software installation (2), the DAISY tool is installed onto the cluster following its original installation procedures; applications need not be modified or recompiled to execute in NETCARE. In terms of user interfaces (3), the HTML-based dynamic interface for DAISY leverages the configuration files previously developed for a similar architecture simulation tool — Simplescalar. The entire installation process for this application — downloading the tool, obtaining user accounts on SP/2 supercomputer, compiling the tool, and porting the forms interface — took about a week to complete; after its installation, all of the NETCARE users are immediately able to execute it through conventional Web browsers.

Another recently added tool is *Compex*, which is a utility for compiling and executing applications. It is part of the Parallel Programming Hub. In Spring 2001 it is being used by students of a parallel programming class to access otherwise unavailable multiprocessor systems. With Compex it is now possible to develop new programs and run them on Netcare machines. One issue raised by Compex is the need for additional security. Unrestricted program access to the underlying system would enable both unintentional and malicious misuse of Netcare. Because of this, Compex is currently only available to a restricted class of users.

4 Educational content

In addition to providing access to hardware and software resources, integrating tools to existing computer engineering classes requires the availability of extensive educational material.

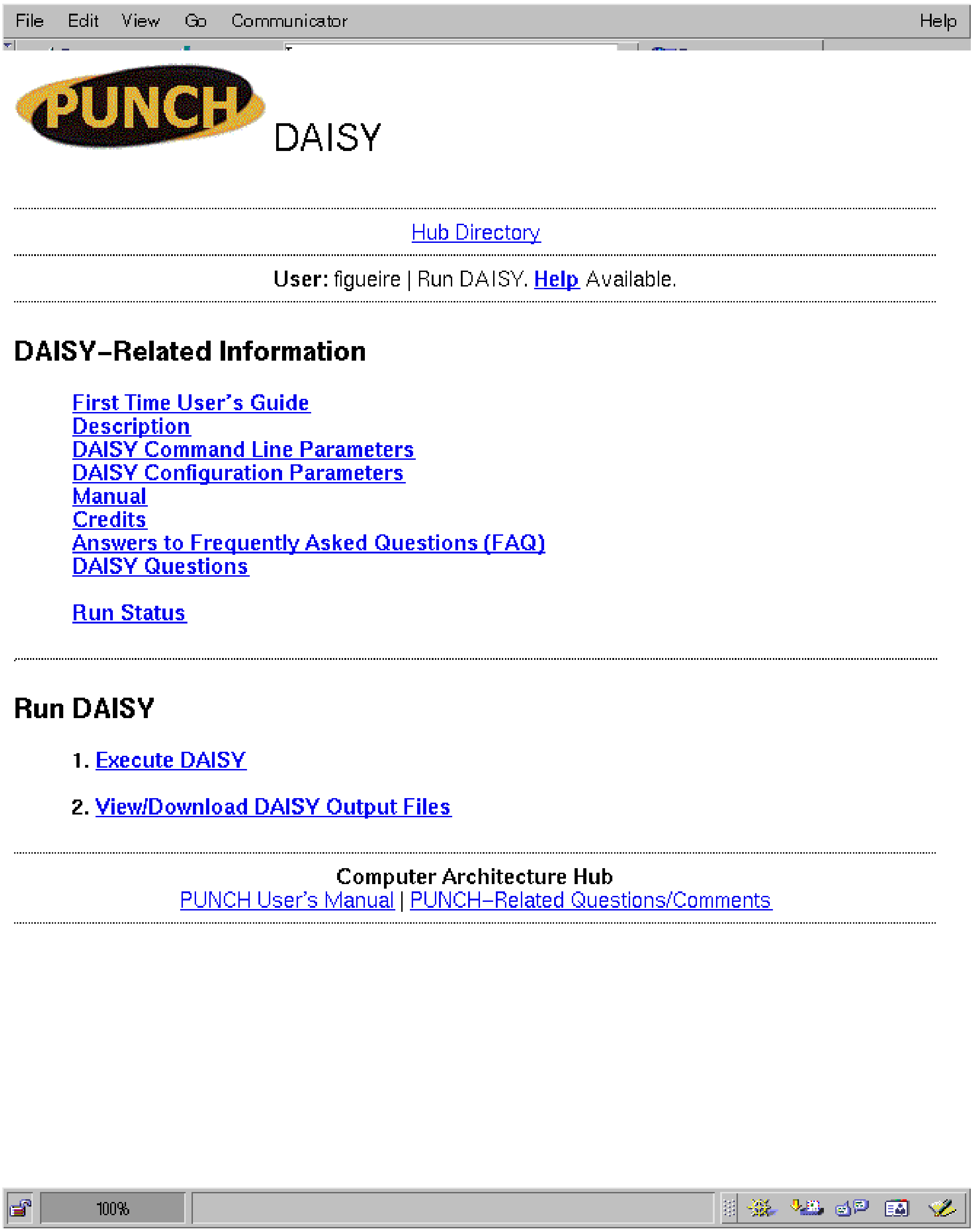


Figure 1: Access to DAISY's documentation and educational content.

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 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 use 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: *Introduction*, *Experimental methods and models*, *Experiment setup*, *Analysis* and *References* [9].

5 User interfaces

The underlying PUNCH network-computing infrastructure used by NETCARE currently supports two types of user interfaces: X Window 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 X Window interface is handled via a Java VNC [22] 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

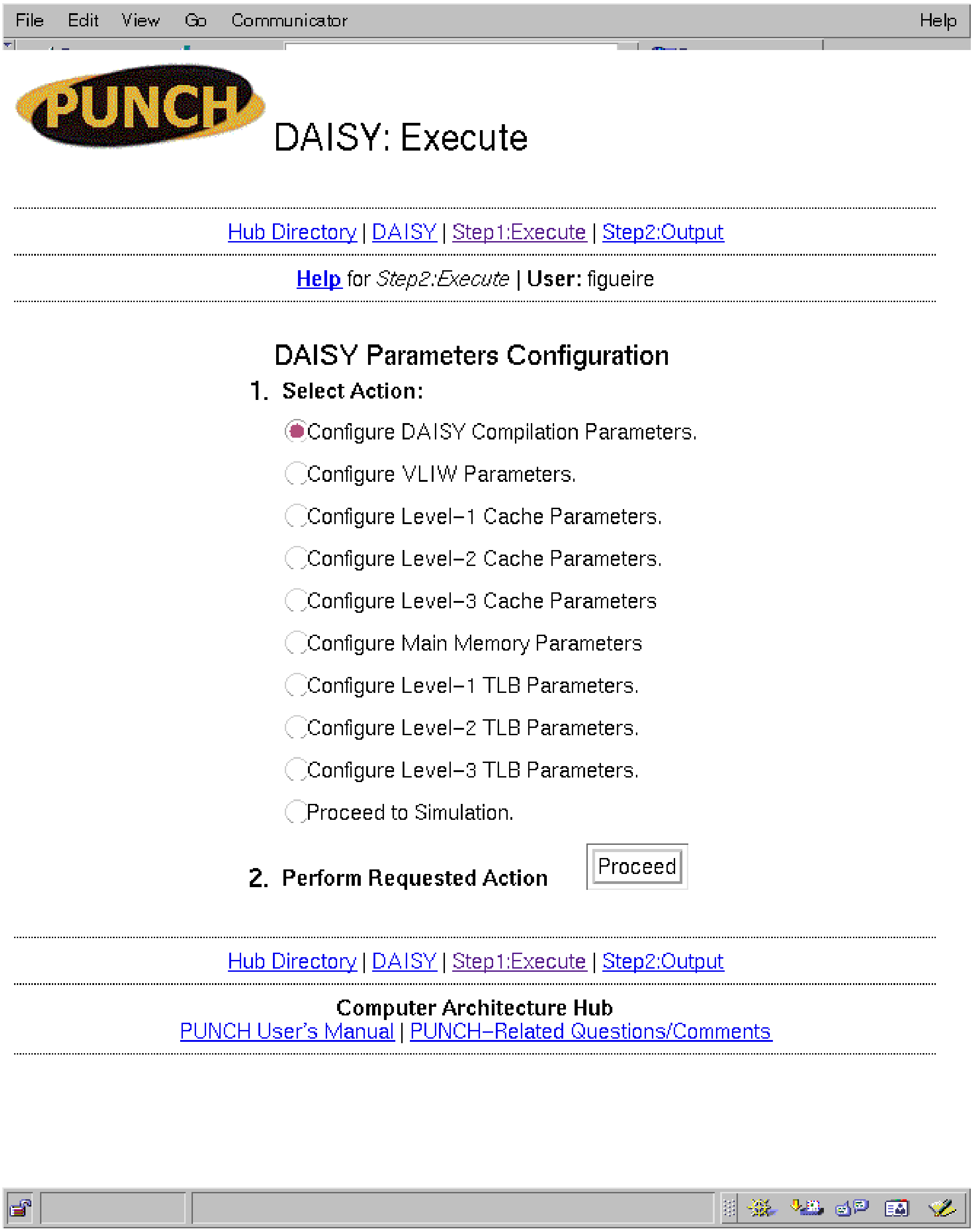


Figure 2: Entry-point HTML interface to execution of a DAISY simulation.

scripting language [15]. 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 DAISY 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 at different sites and used as a basis for the installation of newer versions/variations of the tool, or tools that exhibit similar functionality. For example, the metaprogram developed for the Simplescalar tool has been used as a basis for the installation of WWT-2, RSIM and DAISY onto NETCARE.

Furthermore, the interface provided by metaprograms allows tailoring of the interface complexity based on the needs of target users: the state machine defined by metaprograms can be programmed to display different dynamic Web pages based on user inputs. 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 parameters 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 Window GUI tools can be performed with little effort by reusing existing templates. Ongoing efforts 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, MaxP and Compex) and instruction-level parallelism (SimpleScalar) [9].

The following two subsections present examples of an educational module and an experimentation-based exercise using the NETCARE computing infrastructure. These educational documents are summarized in this paper; their complete versions are accessible from the Cachesim5 and WWT main pages in NETCARE, available via the URL `http://www.ece.purdue.edu/NETCARE`.

6.1 Educational module example

The educational modules described in this subsection illustrate the use of a multiprocessor simulator (WWT-2) to study the parallel performance of both bus-based symmetric multiprocessors (SMPs) and distributed shared-memory multiprocessors (DSMs). The educational modules consist of two HTML documents, formatted as described in Section 4.

The *Introduction* section of each module briefly describes the multiprocessor architecture under study (SMP or DSM), the simulator (WWT-2) and the parallel benchmarks that are used in the experiment.

The *Experimental models* section describes the multiprocessor machine models supported by the simulator. The SMP module describes the bus-based multiprocessor with single-level snooping caches supported by WWT-2, while the DSM module shows that WWT-2 allows the modeling of distributed multiprocessors consisting of SMP nodes and supporting Simple-COMA cache coherence. The *Experimental setup* section describes the benchmarks used in the experiments (FFT, LU, Radix) and determines the simulation parameters that the student will control (e.g. number of processors varying from 1 to 16) and those that are fixed (e.g. cache size and processor speed).

The *Analysis* section provides the student with questions to guide their analysis of the experimental data gathered from the simulations (e.g. "What are the average parallel speedups and efficiencies for 4, 8 and 16 processors?"). Since educational modules are non-graded self-contained documents that are used to illustrate the usage of a tool, this section also contains answers to the proposed questions. The questions and answers provided in the module allow students to check their experimental data and conclusions. Figure 3 shows a screenshot of the answers provided in this section, including simulation results and a parallel speedup plot.

6.2 Exercise example

The example exercise described in this subsection has been used as a graded assignment in an undergraduate-level computer architecture class (CPTR-303) at Chicago State University. The topic covered by the assignment is memory hierarchy, and the tool used in the simulation experiments is Cachesim5. The assignment consists of two documents formatted as described in Section 4; the first exercise deals with hierarchies with a single cache level, while the second exercise deals with two-level cache hierarchies.

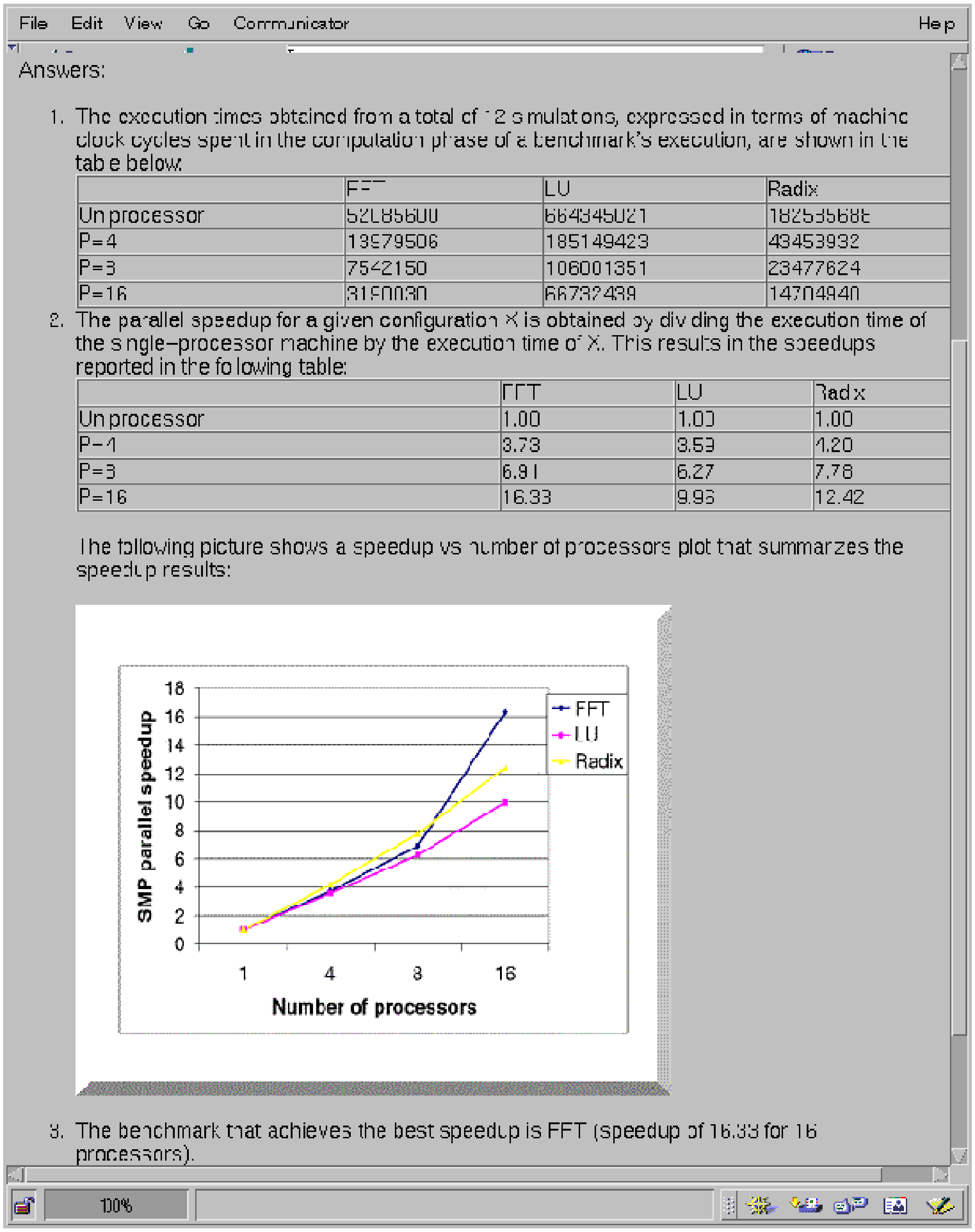
The *Introduction* section of the exercise documents contain a brief description of caches, referencing the chapter on memory hierarchy of the class textbook. It also contains a description of the selected tool (Cachesim5), including a brief explanation of the execution-based simulation technique employed by it, and of the three Spec92 benchmarks (Ora, Doduc, Su2cor) used in the experiments.

The *Experimental methods* and *Experimental setup* sections describe the experiments that students are required to perform in the assignment: they must select three different cache organizations, simulate the execution of all benchmarks for each organization, and identify and summarize cache statistics (such as miss rates). The choices of cache organizations are constrained by a set of acceptable parameters (such as cache and block sizes and associativity) defined in the document.

The *Analysis* section contains several questions to be answered based on the experimental data collected from the simulations. As opposed to the educational modules of the previous subsection, no answers are provided to the students: their work will be graded based on their results and analysis. The students are also asked to make a recommendation as to which of the chosen cache organizations yields the best performance for the benchmarks under study (in terms of average miss rates) based on the simulation analysis.

6.3 Evaluations

The authors have surveyed students that had been exposed to Web-based experiments with tools in computer architecture classes. The survey consists of two questionnaires handed out


File Edit View Go Communicator Help

Answers:

1. The execution times obtained from a total of 12 simulations, expressed in terms of machine clock cycles spent in the computation phase of a benchmark's execution, are shown in the table below:

	FFT	LU	Radix
Uniprocessor	52085600	664345021	182535686
P=4	13979506	185149423	43453932
P=8	7542150	106001351	23477624
P=16	3190030	66732439	14704940

2. The parallel speedup for a given configuration X is obtained by dividing the execution time of the single-processor machine by the execution time of X. This results in the speedups reported in the following table:

	FFT	LU	Radix
Uniprocessor	1.00	1.00	1.00
P=4	3.73	3.59	4.20
P=8	6.91	6.27	7.78
P=16	16.33	9.96	12.42

The following picture shows a speedup vs number of processors plot that summarizes the speedup results:

3. The benchmark that achieves the best speedup is FFT (speedup of 16.33 for 16 processors).

100%

Figure 3: Parallel speedup plot of the SMP educational module for WWT-2.

to students in class: the first survey is distributed prior to the use of NETCARE and asks students about previous experiences with conventional simulators; the second survey is distributed at the end of the semester, and asks students about their experience with Web-based tool experimentation. The motivation for the use of two surveys is to identify whether tools with Web-based interfaces are easier to learn and use than their native, command-line counterparts, and to provide feedback on how to improve the system's usability.

Feedback received from student evaluations indicates that the network-computing interface is easy to learn and an effective aid to understanding architecture concepts [9]. 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 who use tools for both research and education.

In another experiment the students of a parallel programming class had the choice between using a multiprocessor system directly through Unix commands and using a system through the Compex utility, described earlier. While several students preferred the familiar Unix command line interface, other students chose the Netcare option for the following reasons: (1) Netcare allows access to systems other than the directly accessible ones. Some of these systems were only lightly loaded and thus provided better response time. (2) The Compex utility provides the option of running the program from 1 through 4 processors and plotting a speedup curve at the end. Although such functionality can be written with a simple Unix script, many students preferred the Compex option.

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.

The infrastructure can also be customized for different educators by supporting class-specific accounts. In addition to reusing the currently available infrastructure, NETCARE also allows educators to add and share tools and documents (e.g. tutorials and exercises) currently used in their courses to the infrastructure. New tools can be added via the the specification of metaprograms; templates for X Window-based tools and installation guidelines for text-based tools are available for reuse.

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 progress report on a Web-based 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] Adabala, S., Kapadia, N. H., and Fortes, J. A. B. Performance and interoperability issues in incorporating cluster management systems within a wide-area network-computing environment. In *SC-2000: High Performance Networking and Computing*, November 2000.

[2] Bayucan, A., Henderson, R.L., Lesiak, C., Mann, B., Proett, T., and Twenten, D. *Portable Batch System: External reference specification.* MRJ Technology Solutions, November 1999.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] Ebcioglu, K. and Altman, E. R. DAISY: Dynamic compilation for 100% architectural compatibility. In 24^{th} *Annual Internation Symposium on Computer Architecture*, pages 26–37, June 1997.

[8] Figueiredo, . J., Fortes, J. A. B., Eigenman, R., Kapadia, N., Adabala, S., Miguel-Alonso, J., Taylor, V., Livny, M., Vidal, L., and Chen, J-J. A network-computing infrastructure for tool experimentation applied to computer architecture education. In *IEEE TCCA Newsletter*, September 2000.

[9] 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*, June 2000.

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

[11] 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.

[12] Kapadia, N. H., Figueiredo, R. J., and Fortes, J. A. B. Enhancing the scalability and usability of computational grids via logical user accounts and virtual file systems. In *Heterogeneous Computing Workshop (to appear)*, 2001.

[13] 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.

[14] 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.

[15] 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*, 1998. Barcelona, Spain.

[16] 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.

[17] 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.

[18] 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.

[19] 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} Intl. Symposium on High-Performance Computer Architecture*, Feb 1997.

[20] 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.

[21] Park, I., Kapadia, N. H., Figueiredo, R. J., Eigenmann, R., and Fortes, J. A. B. Towards an integrated, web-executable parallel programming tool environment. In *SC-2000: High Performance Networking and Computing*, November 2000.

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

[23] 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.

RENATO FIGUEIREDO is currently a doctoral candidate at Purdue University. Renato Figueiredo received both B.S.E.E. and M.S.E.E. degrees from the State University of Campinas (UNICAMP), Brazil, in 1994 and 1995.

Dr. JOSÉ A. B. FORTES is currently a Professor and Assistant Head for Education of the School of Electrical and Computer Engineering at Purdue University. Dr. Fortes received a B.S.E.E. degree from Univ. de Angola in 1978, a M.S.E.E. degree from Colorado State University in 1981, and a Ph.D. in Electrical Engineering and Systems from the University of Southern California in 1984.

Dr. RUDOLF EIGENMANN is currently an Associate Professor of Electrical and Computer Engineering at Purdue University. Dr. Eigenmann received a Diploma in Electrical engineering from ETH Zürich, Switzerland in 1980, and a Ph.D. in Computer Science from ETH Zürich in 1988.

Dr. NIRAV KAPADIA is currently a researcher at Purdue University. Nirav Kapadia received his B.E. degree in Electronics and Telecommunications from Maharashtra Institute of Technology (India) in 1990. He subsequently worked at the institution for a year as a lecturer. He received his M.S.E.E. and Ph.D. degrees from Purdue University in 1994 and 1999, respectively.

SUMALATHA ADABALA is currently a doctoral student at Purdue University. Sumalatha Adabala. She received B.E. and M.S.E.E. degrees from Bangalore University, India, and Purdue University, respectively.

Dr JOSE MIGUEL-ALONSO is currently an Associate Professor at The University of the Basque Country. Dr Miguel-Alonso received the M.S. and Ph.D. degrees in Computer Science from The University of the Basque Country in 1989 and 1996, respectively.

Dr. VALERIE TAYLOR is currently an Associate Professor of Electrical and Computer Engineering at Northwestern University. Dr. Taylor received a B.S.C.E.E. degree with highest honors from Purdue University in 1985, a M.S.E.E. degree from Purdue University in 1986, and a Ph.D. degree in Electrical Engineering and Computer Science from the University of California at Berkeley in 1991.

Dr. LUIS VIDAL is currently a Professor of Mathematics and Computer Science at Chicago State University. Dr. Vidal received a B.S. degree in Mathematics from the University of Trujillo in 1970, M.S. degrees in Computer Science and Mathematics from Washington State University in 1981 and 1983, and a Ph.D. degree in Computer Science from the Illinois Institute of Technology in 1990.

Dr. JAN-JO CHEN is currently an Associate Professor of Mathematics and Computer Science at Chicago State University. Dr. Chen received a B.S. degree in Architecture from Tunghai University, Taiwan in 1985, a M.S. degree in Computer Science from the Illinois Institute of Technology in 1990, a M.S. degree in Mathematics, Statistics and Computer Science from the University of Illinois at Chicago in 1996, a M.S. degree in Management Information Systems from the University of Illinois at Chicago in 1997, and a Ph.D. degree in Mathematics, Statistics and Computer Science from the University of Illinois at Chicago in 1997.