Int. J. Engng Ed. Vol. 17, No. 2, pp. 207–219, 2001
Printed in Great Britain.

0949-149X/91 $3.00+0.00

PUNCH: A Computing Portal for the Virtual University*

NIRAV H. KAPADIA, JOSÉ A. B. FORTES, MARK S. LUNDSTROM
School of Electrical and Computer Engineering, Purdue University, USA

DOLORS ROYO VALLES
Universitat Politècnica de Catalunya, Barcelona, Spain. E-mail: kapadia@purdue.edu

Given that modeling, simulation and computer-aided design are an integral component of the modern practice of science and engineering, virtual universities are faced with the problem of providing students with access to simulation programs required for their education. This paper describes the Purdue University Network Computing Hubs (PUNCH), an extensively-used Web-based computing portal that allows geographically distributed users to share and run unmodified tools via standard web browsers. The PUNCH infrastructure has been operational for five years, and has been successfully used to share and disseminate specialized tools and educational materials, and to offer simulation-intensive classes in distance education environments.

INTRODUCTION

MODELING, simulation, and computer-aided design are an integral component of the modern practice of science and engineering. Educators make extensive use of such programs in the classroom, both to illustrate concepts and to teach students how to effectively use computers for engineering design and analysis. Modeling and simulation continue to grow in importance, and this raises a number of serious issues regarding the education of scientists and engineers [1]. It also raises a practical question for virtual universities: how are students to be provided with access to simulation programs required for their education? Commercial programs used in classes are often only licensed to run on a small number of high-end workstations. Student versions that run on personal computers are sometimes available, but these versions typically offer limited functionality that may not be adequate for advanced classes. Research codes developed by faculty and used in education may only run on one type of machine, whereas students in a virtual university will be using a variety of computing platforms. Moreover, research codes often depend on site-specific libraries and configurations, making it difficult to use them at other locations.

This paper describes PUNCH (Purdue University Network Computing Hubs), a software infrastructure that allows unmodified programs to be accessed and operated from remote locations via standard WWW browsers running on the students' preferred platforms. The first version of PUNCH went on-line in April 1995. Today, version 4 is used on a regular basis for education and research by more than eight hundred students and researchers from ten countries. Usage has doubled each year over the past four years, with over one million hits in 1999. PUNCH currently provides access to specialized tools for semiconductor technology, VLSI design, computer architecture, and parallel programming; fifty tools developed by six vendors and thirteen universities are available. PUNCH can be accessed at www.punch.purdue.edu.

This paper focuses on the capabilities and applications of PUNCH as perceived by its users, and on the applicability of the infrastructure in the context of the Virtual University. It describes PUNCH's dynamically generated Web-based interface, and explains the tool execution services available to PUNCH users. It provides a network-computing perspective into the PUNCH infrastructure and describes some of the applications of PUNCH, along with concrete examples of our experiences over the past few years. Finally, the advantages of PUNCH over 'point solutions' are outlined.

PUNCH: A USER'S VIEW

From a user's perspective, PUNCH is a WWW-accessible collection of simulation tools and related information (see Fig. 1). It allows geographically dispersed tools to be indexed and cross-referenced, and makes them available to users world-wide. The infrastructure hides all details associated with the remote invocation of tools from its users. Functionally, PUNCH allows users to: 1) upload and manipulate input-files, 2) run programs, and 3) view and download output—all via standard WWW browsers.

* Accepted 5 May 2000.

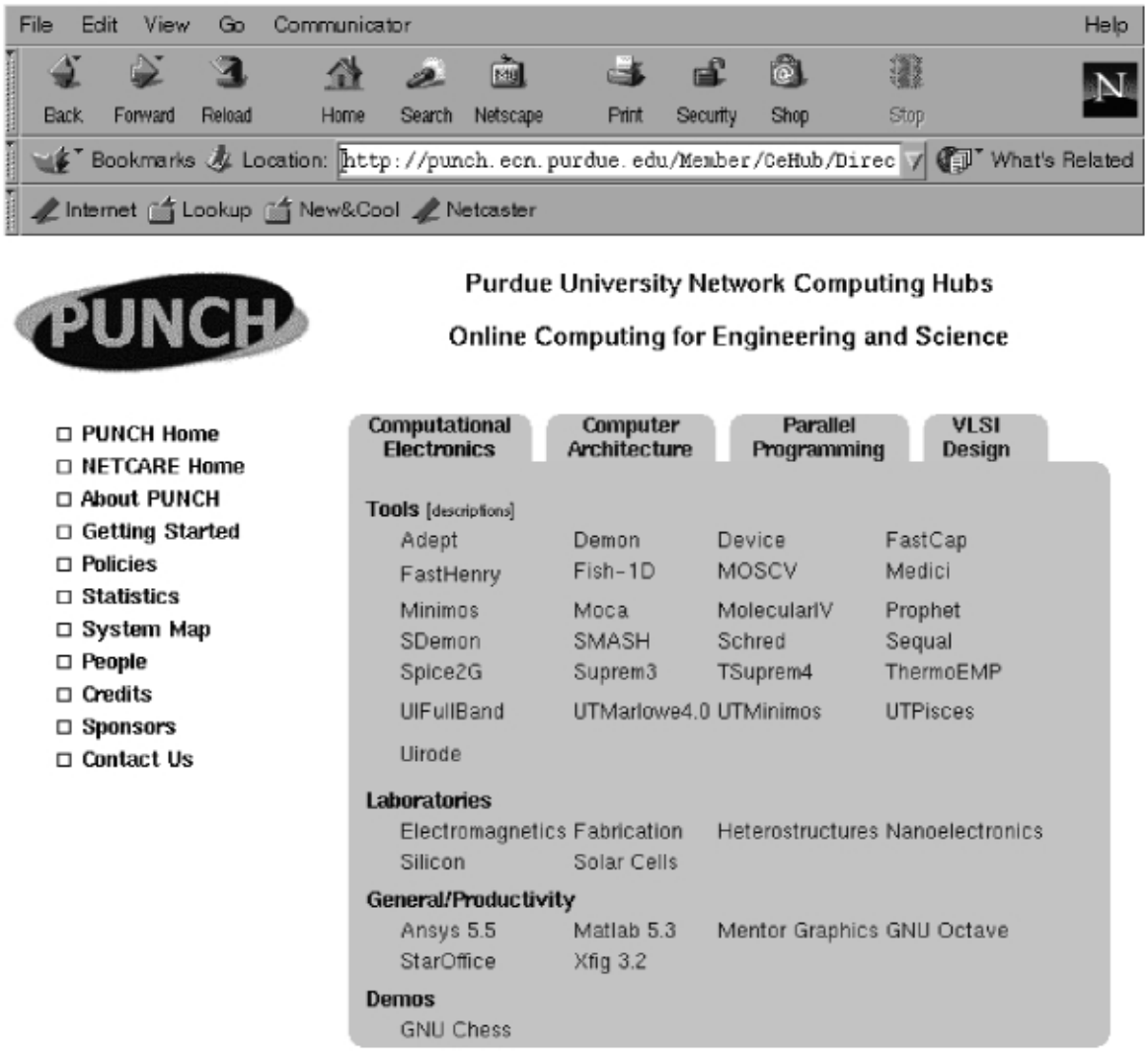

Fig. 1. A view of the Computational Electronics Hub of PUNCH, along with the programs available in that hub.

Each tool in PUNCH is accompanied by a brief description of its capabilities, a manual (where available), example input files and pre-computed output, and the e-mail address of the support site for that tool. If the tool has a home-page on the Web, a link to it is also provided (see Fig. 2).

Running a typical batch simulation on PUNCH is a three-step process (Fig. 2). The first step involves the creation of the input file (Fig. 3) required for the relevant simulation. To facilitate this, PUNCH provides an editor interface (not shown) which allows users to create and modify text files. Also, with a view to assisting new users, PUNCH provides an 'examples folder' (part 'c' in the figure) with sample input files for each tool. A new user can select a sample input file that best matches his/her current needs, copy it over to his/her PUNCH account, and then modify it as necessary. Alternatively, users who prefer to do the editing on their own computers can *upload* files to their PUNCH accounts (part 'e' in the figure). PUNCH's input interface (Fig. 3) also supports general file-manipulation and utility functions such as copying, renaming, deleting, and compressing files; frequently-used commands are displayed as check-boxes in Fig. 3, while other commands may be accessed via the text-based command-line (part 'd' in the figure).

In the second step, users define the input parameters (e.g. command-line arguments, etc.) for the program and start the simulation. For the MINIMOS [2] program, users must select the input file that is to be used for the simulation, name the directory in which the output is to be stored, and name the file to which the output of the program is to be redirected (Fig. 4). Users can also enter an e-mail address at which they wish to be notified of the completion of the simulation. This feature is typically used for simulations that are expected to run for several hours or days. Once all this information has been entered, users can submit the request to PUNCH for processing. Before initiating the run, PUNCH will verify that all the necessary parameters were entered and supply default values where appropriate. The run is typically initiated on a remote compute-server; a

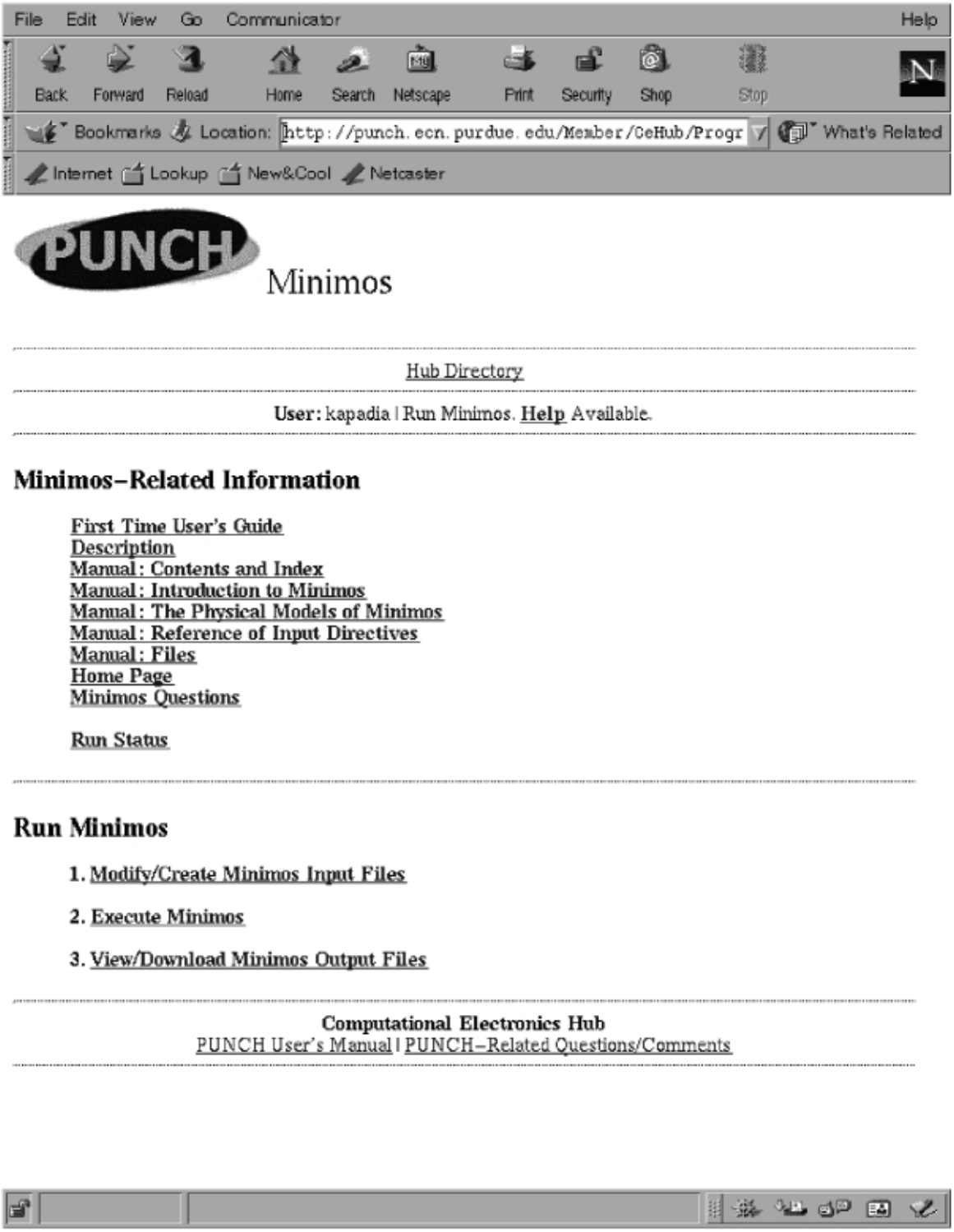

Fig. 2. The entry point to the MINIMOS 6.0 user-interface. This page is dynamically generated by PUNCH.

run-status page (not shown) allows the user to check on the status of the simulation and abort it if necessary.

Finally, after the simulation is complete, the user can see and download the results of the simulation via PUNCH's output interface. The output interface is very similar to the input interface (Fig. 3) in that it allows the user to view files and provides general file-manipulation and utility functions. In addition to this functionality, the output interface also facilitates post-processing of results. The post-processing interface allows users to manipulate the output generated by the simulator. Typically, this involves generating postscript plots interactively; in a few cases, the post-processor further manipulates the results of the simulation. Like the execution interface, the post-processing interface is programmable; its organization and content are

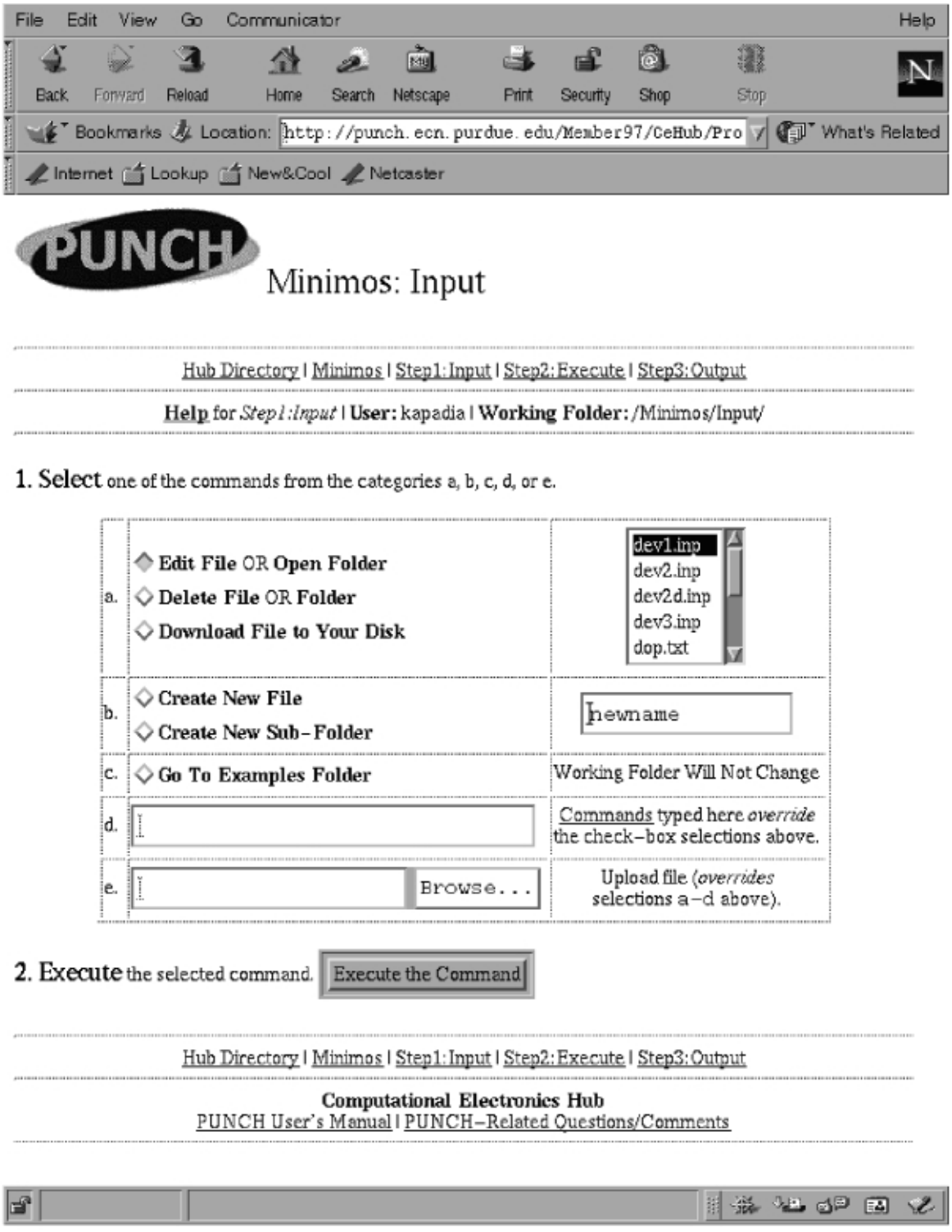

Fig. 3. The input interface for MINIMOS 6.0. This page is also dynamically generated by PUNCH.

defined by the tool administrator when he/she installs a simulation tool on PUNCH. The post-processing interface for MINIMOS 6.0 is shown in Fig. 5.

PUNCH also supports tools with graphical user interfaces. It accomplishes this by leveraging remote display-management technologies such as Broadway [3] and VNC [4] (see Fig. 6). One issue that arises with the use of tools with graphical interfaces is the need to manage user-accounts on remote computer-servers. PUNCH addresses this problem by employing logical user accounts. Although physical accounts are still required on machines, they all belong to PUNCH, and can be recycled among PUNCH users as necessary (based on specified usage policies).

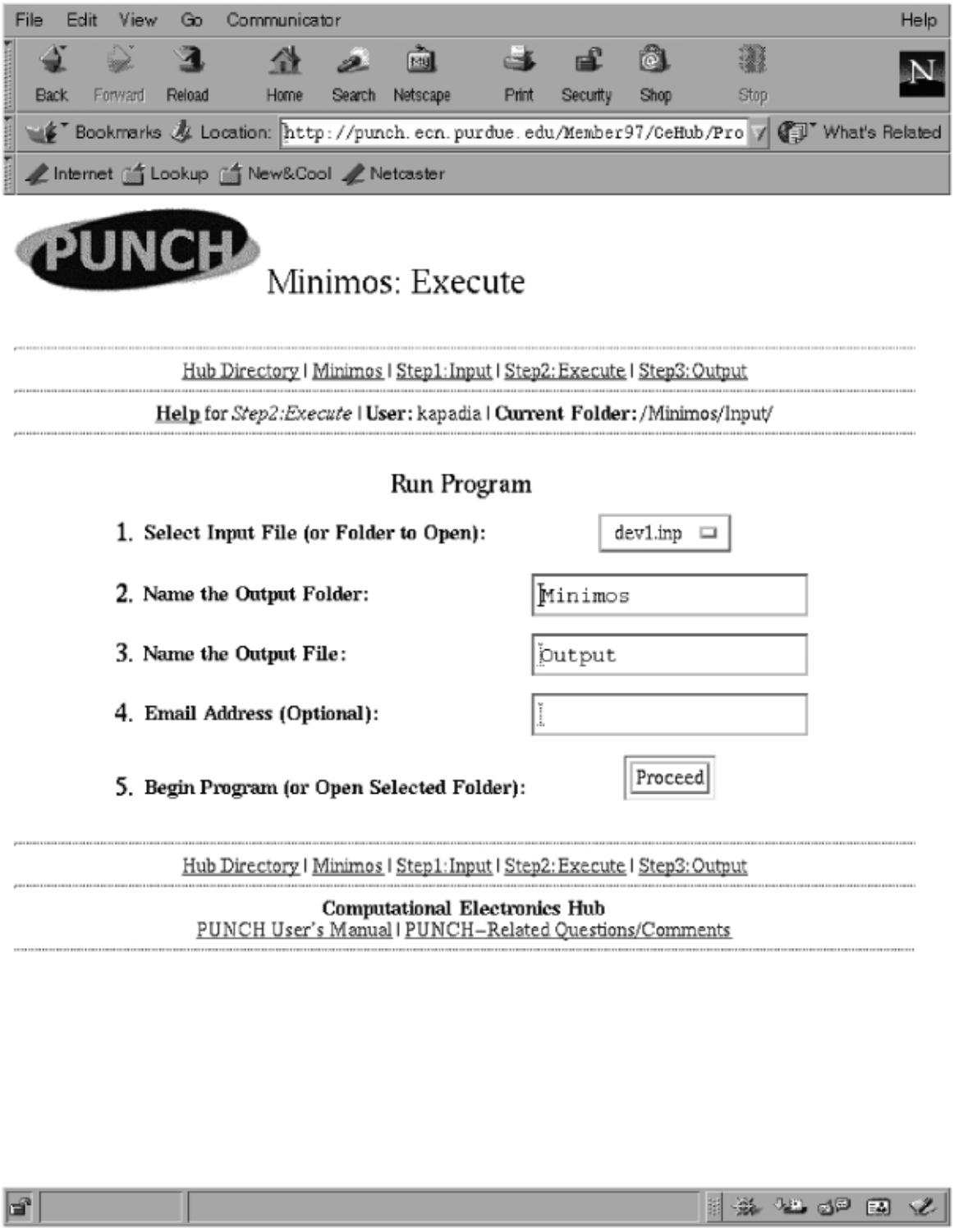

Fig. 4. The execution interface for MINIMOS 6.0. This page is dynamically generated.

PUNCH: A NETWORK-COMPUTING PERSPECTIVE

The PUNCH infrastructure can be divided into two parts (see Fig. 7). The network desktop (or front-end) primarily deals with content management and user-interface issues. It allows users to interact with the underlying network-computing infrastructure (SCION) via standard Web browsers, and generates customized views of available resources for each class of users. SCION serves as PUNCH's user-transparent middleware. It consists of hierarchically distributed servers that co-operate to provide on-demand network-computing. This part of the infrastructure manages the run-time environment

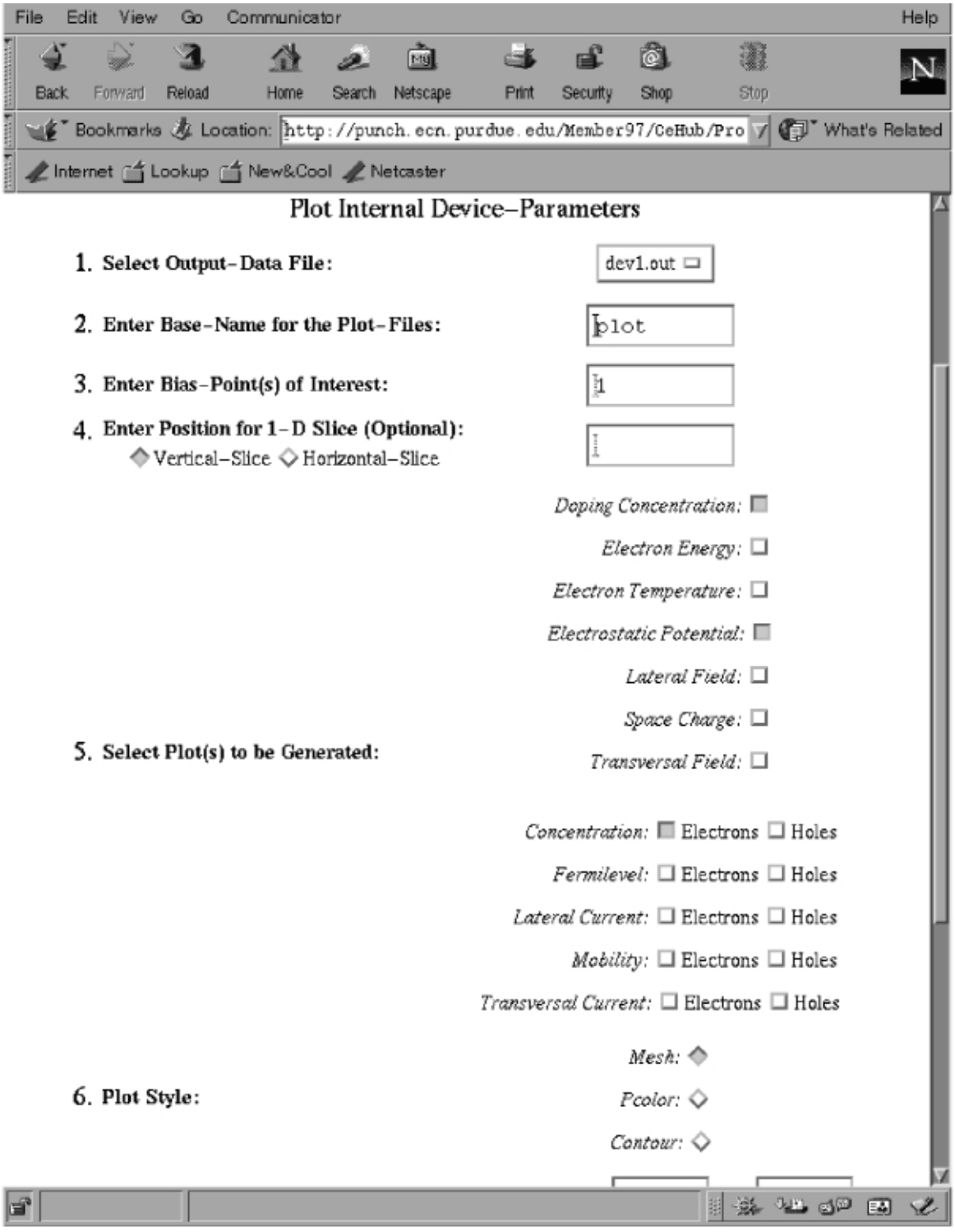

Fig. 5. The post-processing interface for MINIMOS 6.0.

for tools, addresses security issues, controls resource access and visibility, and facilitates cross-enterprise scheduling of available resources.

PUNCH allows tools to be organized and cross-referenced according to their discipline. Resources can be added incrementally using a resource-description language specifically designed to facilitate the specification of tool and machine characteristics. For example, a machine can be incorporated into PUNCH simply by specifying its architecture (make, model, operating system, etc.) and starting a server on it. Similarly, a tool can be added by 'telling' PUNCH the tool's location, its input behavior (e.g. command-line arguments), what machines it can run on (e.g. Ultra-Sparc), and how it fits into the logical organization of software resources (e.g. device simulation tool). The Web pages needed for

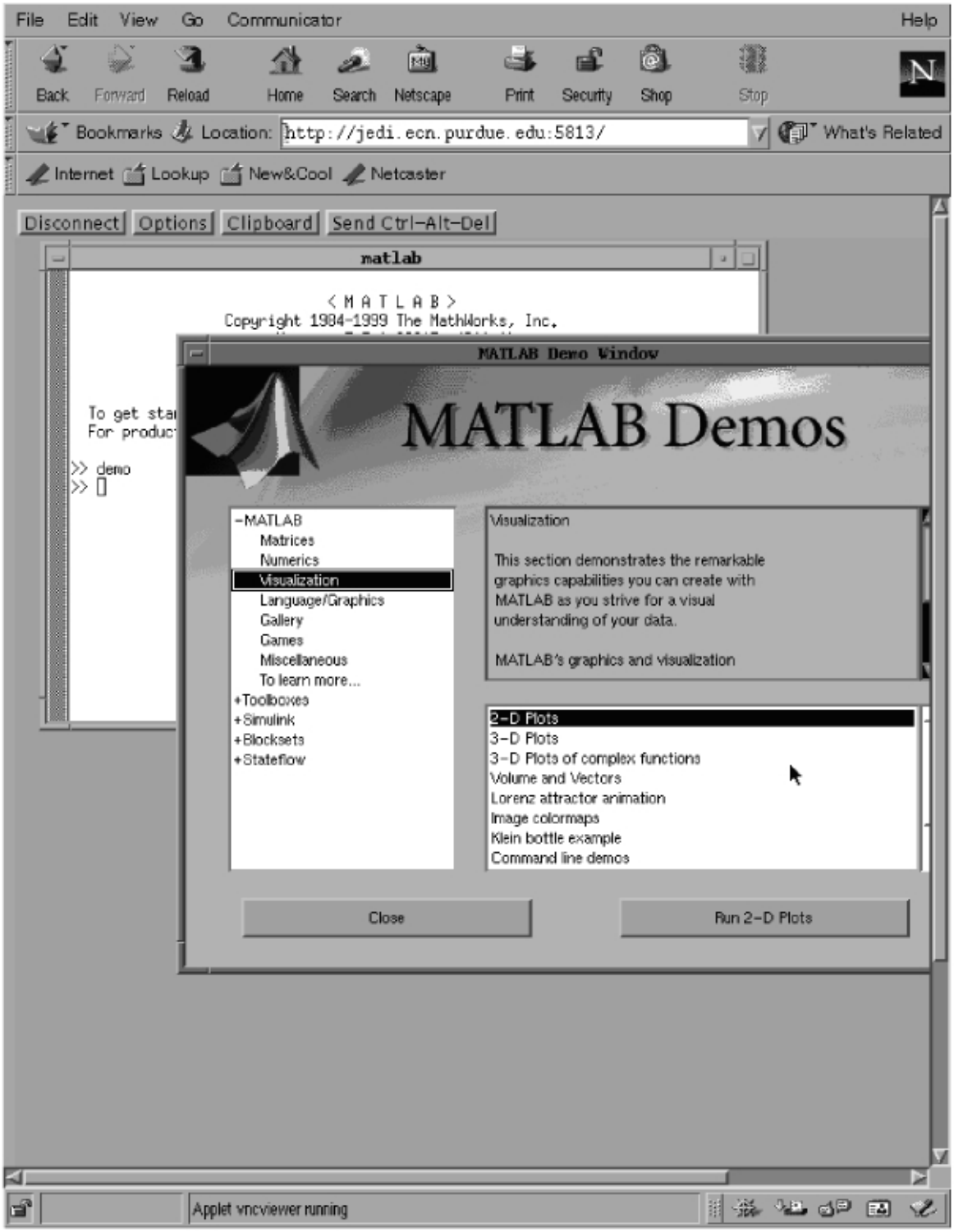

Fig. 6. An example use of a tool with a graphical user interface. The figure shows Matlab from MathWorks running within a browser. The display is mapped to a browser by way of VNC.

the tool are generated automatically from HTML templates [5]. These tasks are typically accomplished in less than thirty minutes.

One of the key aspects of the PUNCH infrastructure is that it is designed to manage resources across administrative domains [6]. In this context, the Virtual University presents a particularly interesting environment because of its decentralized structure. Students enrolled in any given class are spread across states, countries, and even continents. Moreover, hardware and software resources available to the students are likely to be provided and/or hosted by different organizations, depending on the geographical location of the student and the nature of the class. In such environments, usage constraints and costs can vary

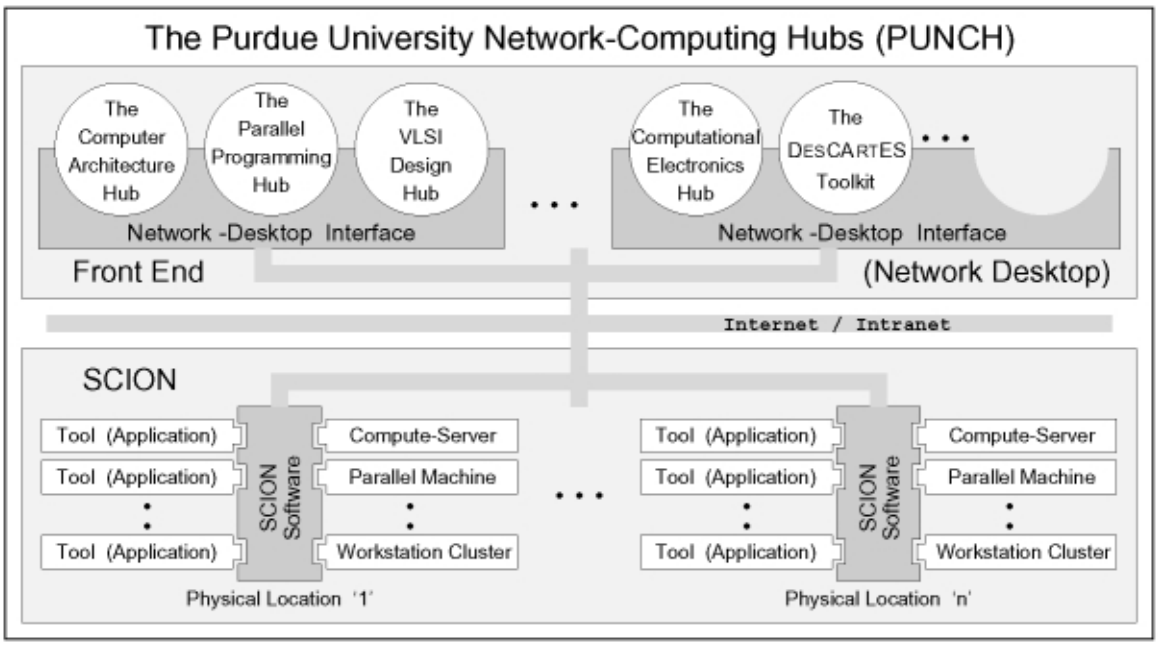

Fig. 7. The main components of the PUNCH infrastructure: the front-end manages data and interfaces, and SCION manages resources.

significantly with the nature of the software and the provider of the resources. This, in turn, makes intelligent resource management critical to the scalability and economic viability of any network-computing infrastructure. The following discussion highlights the suitability of PUNCH's pipelined resource management architecture for computing across institutional boundaries; related information on enterprise-wide computing with PUNCH can be found in [7].

From a resource management standpoint, PUNCH can be divided into three main components (see Fig. 8): the network desktop, the application management component, and the resource management pipeline. With reference to the figure, users interact with PUNCH via its Web-accessible network desktop (event 1 in the figure). The network desktop processes file- and data-manipulation requests locally, and forwards requests for tool execution to SCION (event 2 in the figure). The application management component of SCION (see Fig. 9) parses the user input,

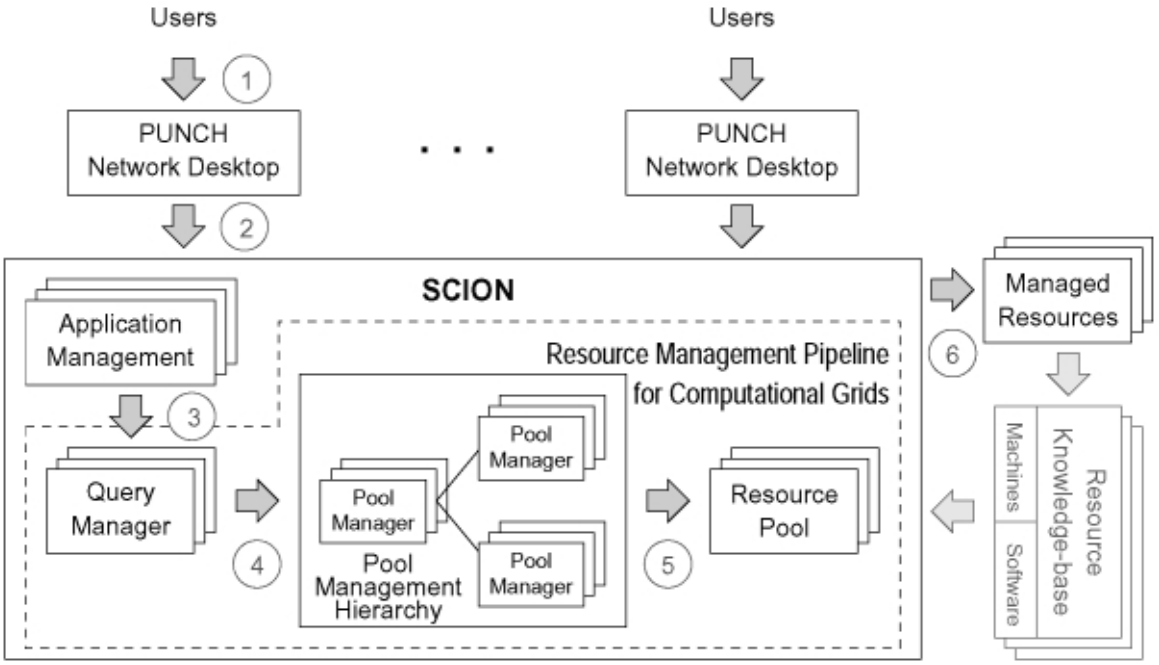

Fig. 8. The components of the PUNCH infrastructure from a resource management perspective. The numbers 1–6 in the figure show the sequence of events that occur in the process of scheduling and initiating a run on PUNCH. Details are provided in the text.

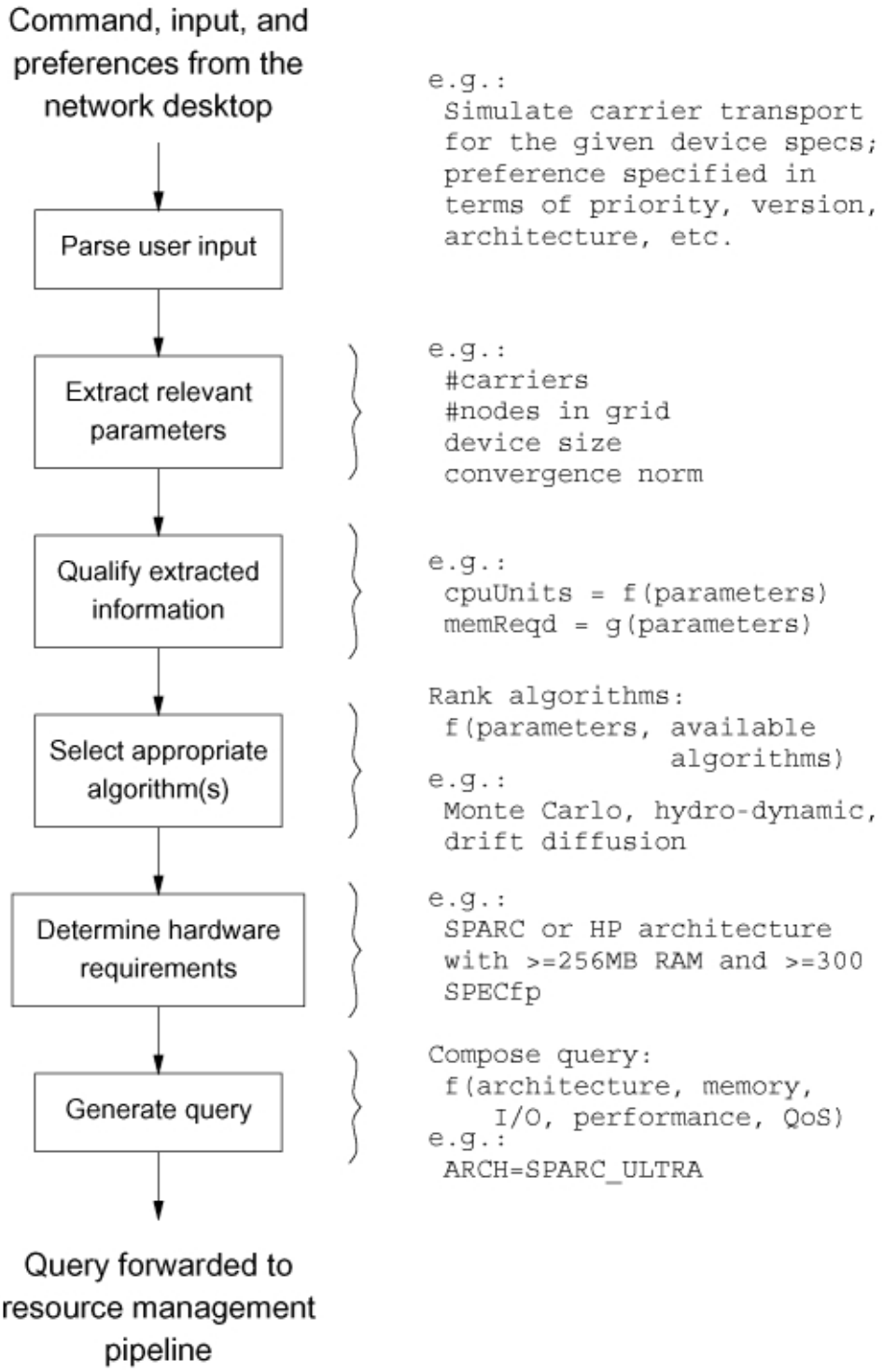

Fig. 9. An overview of the scheduling events that occur within the application management component shown in Fig. 8.

extracts relevant parameters based in information in a knowledge base, estimates the run-time for the application (via a performance modeling system; see [8, 9] for details), determines software and hardware requirements, and constructs a query for the resource management pipeline from the available data. The generated query is subsequently forwarded to the resource management pipeline (event 3 in Fig. 8).

The resource management pipeline provides two capabilities that are key to managing machines and software in a cross-enterprise computing environment: 1) it allows individual components of the pipeline to be distributed and replicated as necessary, and 2) it dynamically adapts to new requirements specified within queries and to changing constraints imposed by resource owners. The first capability allows administrators to tailor the configuration of the resource management system to the demands imposed by the observed mix of

users, jobs, and available resources in a given environment. It also allows the system to grow with increasing numbers of jobs and resources. The second capability allows resources to be dynamically aggregated in a manner that minimizes scheduling overheads for the kinds of jobs that are being initiated at any given time. This aggregation is based on the idea of an *active yellow pages directory*. The traditional yellow pages directory allows easy access to 'resources' by indexing and cross-referencing them within predefined categories according to a fixed set of classification criteria. The active yellow pages directory employs the same basic principle, but allows the categories and the classification criteria to be created and specified on the fly.

The query received by the resource management pipeline is first forwarded to a query manager (event 3 in Fig. 8). If the query is complex (e.g. when multiple machines with different architectures are requested), the query manager decomposes it into its individual components. It then selects an appropriate pool manager for each component and forwards the components to the selected pool managers (event 4 in the figure). The pool manager maps a given query to the corresponding resource pool according to the classification criteria contained within the query (event 5 in the figure). In addition to mapping queries, the pool manager creates resource pools if they do not exist, and destroys pools that have been inactive; these steps are analogous to choosing, creating, and destroying specific categories within a yellow pages directory. The information required for aggregating resources into pools is obtained via a distributed knowledge base that contains data about available resources (see Fig. 8). Once a query has been mapped to a resource pool, scheduling agents associated with that pool select the requested number of the machines on the basis of performance-related criteria. Finally, SCION initiates the requested run on the selected machines (event 6 in the figure).

PUNCH IN EDUCATION

PUNCH allows users to run tools and access educational materials developed by educators and researchers from academia and industry; many representative tools from disciplines such as solid-state devices and materials, VLSI design, computer architecture, and parallel programming are already available. The emphasis has been on exposing the students to the functionality and nature of the tools, while eliminating the need for time spent in securing access to machines, accounts, documentation, and learning unfriendly interfaces. Results show that PUNCH supports the integration of a large number of tools in undergraduate and graduate classes, while minimizing the overheads of installing tools and learning their interfaces, and finding resources to run them. PUNCH is publicly accessible at www.ece.purdue.edu/punch, and, upon request, available for use by universities and educators. The following discussion highlights the utility of PUNCH in educational settings.

PUNCH is currently being used in several undergraduate and graduate courses at Purdue and elsewhere. Experience shows that PUNCH facilitates the use of simulation in courses by providing an easy, platform-independent means to access simulation programs, run simulations, and view printed and graphical output. Many students use PUNCH through a modem from their residences rather than making use of on-campus computer labs where seats are frequently hard to find. Results from user-surveys indicate that the system performs well under the highly peaked usage patterns (very high usage in the hours before homeworks and projects are due) characteristic of an academic environment.

Enriching traditional curricula

In a classroom setting, students often require access to a variety of software tools during the course of engineering design projects. In such situations, the learning experience of students would be greatly enhanced if they could be given access to a broad range of state-of-the-art research software packages. However, this is rarely seen in practice because of three problems:

1. the instructor (or the students) must first locate and identify relevant software packages;
2. they must then install the packages on local machines;
3. the students must subsequently learn the software packages' native user interfaces and understand idiosyncrasies typical of research tools.

PUNCH helps students and instructors to locate and identify appropriate tools by providing a single point of access to a wide range of software. PUNCH obviates the need to install tools locally because it allows users to run remotely-installed tools directly from their Web browsers. Finally, PUNCH provides tool developers and installers with a mechanism to quickly and easily improve the user-interfaces to their codes. For example, command-line arguments (e.g. '-O' to indicate optimization) can be presented to users as sets of check-boxes or menus with appropriate captions.

These benefits of PUNCH are being exploited by instructors at Purdue University and the University of Illinois at Chicago to enhance integrated circuit (IC) processing courses. At Purdue, the course is taught in the School of Electrical and Computer Engineering, and makes use of a commercial simulation tool for IC process simulation. AT UIC, the course is taught in the Department of Chemical Engineering, and makes use of custom-developed software to simulate the chemical processes in IC manufacturing. Students in these two courses share the same on-line computer

laboratory, where they operate the necessary tools and access common homework problems and example files. In a separate but similar project, educators at Chicago State University, Northwestern University, and Purdue University have been leveraging PUNCH to integrate computer architecture tools into the undergraduate curriculum.

Sharing and disseminating content

Many university projects result in new educational content or software tools—or both. The educational content and software tools are often directed at future-generation technologies and have extended development times. Once they are available, however, there is no quick and easy way for the developers to make other educators and researchers aware of their results.

From an educator's perspective, after a seemingly appropriate tool is located, it must be acquired, installed, and integrated into the local environment before it can be evaluated—a time-consuming and costly task at best. Moreover, it is often very difficult to find example input and data files, first-time users' guides, and tutorials that describe the tool's capabilities and functionality—even though such educational aids do exist in many cases. This reality has lead to considerable duplication of efforts, waste of resources, and delays in education and research work.

PUNCH provides tool and content developers with a quick and easy way to 'publish' their results to their communities. In the same vein, PUNCH allows educators and researchers to access and try out university-developed software tools and the associated educational content with a minimal investment of time and effort. Experience shows that, by facilitating interactions between content developers and end-users, both parties benefit, and research is accelerated.

DesCArtES, a NSF-funded Distributed Center for Advanced Electronics Simulations consisting of teams at the University of Illinois, Stanford University, Purdue University, and Arizona State University utilizes PUNCH to make advanced simulation tools available to experimentalists and students worldwide. NETCARE, a three-university NSF-funded project involving Purdue University, Northwestern University, and the University of Wisconsin at Madison, is using PUNCH to establish a universally accessible pool of software tools and computing resources for the computer architecture community.

Distance learning

Currently, it is difficult for universities to offer simulation-intensive courses to off-campus students because of the problems associated with providing all students with access to a common tool-set (e.g. a particular version from a specific vendor). Access to these tools is also required to be through the users' preferred computing platforms (e.g. some users may not be familiar with a UNIX environment). Industry, however, continues to express interest in such courses to assist engineers in becoming effective users of sophisticated simulation programs and in the use of simulation for computational prototyping.

PUNCH facilitates education of consistent quality across institutions by providing universal access to a uniform computing environment. Indeed, instructors from several universities (e.g. Arizona State University, Purdue University, Technion in Israel, University of California at Berkeley, University of Illinois at Chicago, University of Louisville in Kentucky) already utilize PUNCH to allow their students to access and run specialized simulation tools. The School of Electrical and Computer Engineering at Purdue University has offered television courses on computer architecture, advanced C-MOS device technology, digital computational techniques, and electronic instrumentation by leveraging PUNCH. The Indianapolis and Calumet campuses of Purdue University are using PUNCH to facilitate off-campus access to computing resources.

RELATED WORK

The work on PUNCH was motivated by the fact that many of the systems and technologies that currently allow computing on the Web target a single or a relatively small set of tools and/or work within controlled environments in which many of the issues that arise in production environments can be ignored.

Solutions that target individual tools tend to be non-reusable in spite of the fact that they involve a significant amount of duplicated effort. (Reusability, in this context, implies an ability to reuse the computing system with other applications without making any modifications to the system itself.) For example, a large number of systems (e.g. the Exploratorium [10], JSPICE [11], and others) are based on scripts that need to be modified in order to add any new application to the system. Other designs are more flexible. The MOL [12] prototype, for example, employs static Web interfaces that can be adapted for individual tools. The NetSolve [13], Ninf [14], and RCS [15] systems are based on structured designs that target numerical software libraries. However, static interfaces are not adequate for all tools, and structured approaches cannot be easily applied to general-purpose applications. Another problem is that the majority of these designs assume the availability of the source and/or object code for the applications—which effectively precludes the installation of most commercial tools.

Solutions that address individual issues are generally reusable, but the tasks of adapting them for a production environment and integrating them into a complete computing infrastructure are non-trivial. For example, VNC

[4] and WinFrame [16] provide very flexible mechanisms for exporting graphical displays to remote consoles in a platform-independent manner—but, by themselves, the technologies do not help address other issues (e.g. access control and management) that arise in a wide-area distributed computing environment.

The goal of the PUNCH project is to design and deploy a production wide-area computing framework in which the computing infrastructure is not tied to the characteristics of individual applications. Functionally, this is equivalent to designing a multi-user *operating system* for networked resources that provides user-transparent file, process, and resource management functions, handles security and access control across multiple administrative domains, and manages state information (session management). The need for such a system and the associated benefits in the context of the World-Wide-Web have also been recognized elsewhere [17, 18].

CONCLUSIONS

Over the past five years, PUNCH has served as a virtual computing center for thousands of users across more than ten countries. The infrastructure has allowed educators to share and disseminate specialized tools and educational materials, to enrich the curriculum in existing courses, and to offer simulation-intensive classes in distance education environments. These experiences have shown that Web-based computing can be used very effectively in education and research. An infrastructure such as PUNCH will allow this new computing paradigm to become an integral part of the services offered by the Virtual University.

Acknowledgements—This work was partially funded by the National Science Foundation under grants EEC-9700762, ECS-9809520, EIA-9872516, and EIA-9975275, by an academic reinvestment grant from Purdue University, and by a grant from the Commission for Cultural, Educational and Scientific Exchange between the United States of America and Spain.

REFERENCES

1. Eugene S. Ferguson, *Engineering and the Mind's Eye*, M.I.T. Press (1992).
2. S. Selberherr, A. Schutz and H. W. Potzl, Minimos—a two-dimensional MOS transistor analyzer, *IEEE Transactions on Electron Devices*, **27** (8) (August 1980) pp. 1540–1550.
3. X Consortium, Inc. X Window System Version 11 Release 6.3: Release Notes (1996).
4. Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood and Andy Hopper, Virtual network computing, *IEEE Internet Computing*, **2** (1) (January-February 1998) pp. 33–38.
5. Nirav H. Kapadia, José A. B. Fortes and Mark S. Lundstrom, The Purdue University Network-Computing Hubs: Running unmodified simulation tools via the WWW, *ACM Trans. Modeling and Computer Simulation*, Vol. 10, No. 1, Jan 2000, p. 39–57.
6. Nirav H. Kapadia and José A. B. Fortes, PUNCH: An architecture for web-enabled wide-area network-computing, *Cluster Computing: J. Networks, Software Tools and Applications*, **2** (2) (September 1999) pp. 153–164, in special issue on High Performance Distributed Computing.
7. Nirav H. Kapadia, José A. B. Fortes and Mark S. Lundstrom, Statewide enterprise computing with the purdue university network computing hubs, in *Proc. 1st Int. Conf. Enterprise Information Systems (ICEIS'99)*, Setúbal, Portugal, March 1999.
8. Nirav H. Kapadia, Carla E. Brodley, José A. B. Fortes and Mark S. Lundstrom, Resource-usage prediction for demand-based network-computing, in *Proc. Workshop on Advances in Parallel and Distributed Systems (APADS)* West Lafayette, Indiana, October 1998, IEEE Computer Society, pp. 372–377.
9. Nirav H. Kapadia, José A. B. Fortes and Carla E. Brodley, Predictive application-performance modeling in a computational grid environment, in *Proc. 8th IEEE Int. Symp. High Performance Distributed Computing (HPDC'99)* Redondo Beach, California, August 1999, pp. 47–54.
10. Christopher Adasiewicz, Exploratorium: User friendly science and engineering. *NCSA Access*, **9** (2) (1995) pp. 10–11.
11. Dan Souder, Morgan Herrington, Rajat P. Garg and Dennis DeRyke, JSPICE: A component-based distributed Java front-end for SPICE, in *Proc. 1998 Workshop on Java for High-Performance Network Computing*, 1998.
12. A. Reinefeld, R. Baraglia, T. Decker, J. Gehring, D. Laforenza, F. Ramme, T. Romke and J. Simon, The MOL project: An open, extensible metacomputer, in *Proc. 1997 IEEE Heterogeneous Computing Workshop (HCW97)* (1997) pp. 17–31.
13. Henri Casanova and Jack Dongarra, NetSolve: A network solver for solving computational science problems, in *Proc. Supercomputing Conference*, 1996. Also Technical Report #CS-95–313, University of Tennessee.
14. Mitsuhisa Sato, Hidemoto Nakada, Satoshi Sekiguchi, Satoshi Matsuoka, Umpei Nagashima and Hiromitsu Takagi, NINF: A network-based information library for global world-wide computing infrastructure, in *High-Performance Computing and Networking (Lecture Notes in Computer Science, 1225)* Springer-Verlag, Berlin (1997) pp. 491–502.
15. Peter Arbenz, Walter Gander and Michael Oettli, The Remote Computation System, *Parallel Computing*, **23** (1997) pp. 1421–1428.
16. ICA technical paper, WWW document at www.citrix.com/technology/ (March 1996).
17. Slim Ben Lamine, John Plaice and Peter Kropf, Problems of computing on the Web, in *Proc. 1997 High Performance Computing Symposium* (1997) pp. 296–301.
18. Franklin D. Reynolds, Evolving an operating system for the Web, *Computer*, **29** (September 1996).

Nirav Kapadia is currently a Senior Research Scientist in the School of Electrical and Computer Engineering at Purdue University. His research interests are in the areas of network-based and wide-area distributed computing, WWW-based computing portals, predictive application-performance modeling, and resource management across institutional boundaries. He received the BE degree in Electronics and Telecommunications from Maharashtra Institute of Technology (India) in 1990, the MS degree in Electrical Engineering from Purdue University in 1994, and the Ph.D. degree in Computational Engineering from Purdue University in 1999. He is a member of Phi Beta Delta, an honor society for international scholars. E-mail: kapadia@purdue.edu.

José Fortes is a Professor and Assistant Head for Education in the School of Electrical and Computer Engineering at Purdue University. His research interests are in the areas of parallel processing, computer architecture, network-computing, and fault-tolerant computing. He received the BS degree in Electrical Engineering (Licenciatura em Engenharia Electrotécnica) from the Universidade de Angola in 1978, the MS degree in Electrical Engineering from the Colorado State University, Fort Collins in 1981, and the Ph.D. degree in Electrical Engineering from the University of Southern California, Los Angeles in 1984. He is a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) professional society, and was a Distinguished Visitor of the IEEE Computer Society from 1991 till 1995. E-mail: fortes@purdue.edu.

Mark Lundstrom is currently a Professor of Electrical and Computer Engineering at Purdue University where he has also served as Director of the Optoelectronics Research Center and as Assistant Dean of Engineering. His research interests center on the physics of semiconductor devices and include carrier transport, ultrasmall transistors and computational electronics. Previous work included studies of heavy doping effects, heterostructure devices, solar cells, heterojunction bipolar transistors, and semiconductor lasers. He received the BEE and MSEE degrees from the University of Minnesota in 1973 and 1974 and his Ph.D. from Purdue University in 1980. He is an IEEE Fellow ('for contributions to heterostructure device physics and simulation'), the recipient of the ASEE Frederick Emmons Terman Award, and the recipient of two teaching awards from Purdue University. E-mail: lundstro@purdue.edu.

Dolors Royo Valles is an Associate Professor at the department of computer architecture at UPC (Polytechnic University of Barcelona), Spain. Her research interests include parallel algorithms, distributed systems and metacomputing. She received her Ph.D. in computer science from UPC. She is a member of the IEEE. E-mail: dolors@ac.upc.es.