Louisiana Tech University

Integrating Research from Parallel and Distributed Computing Throughout the Undergraduate Curriculum

Principal Investigator:
Barry L. Kurtz Department of Computer Science

Philosophy: The future of computing in the 21st century depends heavily on students acquiring a thorough understanding of the theory and practice of parallel and distributed computing. Most curricula today introduce concurrency in upper division courses such as operating systems, algorithms, programming languages, and architecture. Much of the basic material is duplicated in these courses and after so many courses emphasizing the von Neumann model, these students have difficulty accepting other computational models. To address these problems we proposed a sophomore level course in concurrency that introduces the fundamental concepts of concurrency early, avoids duplication of topics in more advanced courses, and frees up time in the advanced courses to transfer current research results into the undergraduate curriculum. The materials developed are oriented towards undergraduate institutions whose faculty have little research expertise in these areas and whose facilities do not include expensive, special purpose parallel computers. Work on this grant was started in September 1994 and will continue until the end of 1997.

Institutions and Personnel: This grant was awarded to Louisiana Tech University, a Doc III level school with a reputation for quality undergraduate instruction, and to Grambling State University, an historically black university located three miles from Louisiana Tech. The original investigators at Tech were Michael Meehan (distributed systems), Mike O'Neal (algorithms), Barry Kurtz (programming languages), and Mazin Yousif (architecture). Both Meehan and Yousif have left the project to accept employment elsewhere and this has slowed down our timetable for development. We have appointed Chin Kim to work in the area of architecture and Meehan is to continue development of the operating systems module based on earlier commitments. We have requested a no-cost time extension to complete the promised work. The investigators at Grambling are M. Balaram (general concurrency) and Jamal Alsabbagh (database). There have been no changes in personnel at Grambling.

The Introduction to Concurrency Course : The introduction to concurrency course at Louisiana Tech, hereafter referred to by the course number CS240, introduces classical concurrency concepts (semaphores, monitors, rendezvous, remote procedure calls), alternative machine architectures (principally MIMD and SIMD), algorithm development for parallel architectures, and programming language support for concurrency (either built-in or through library extensions). The CS240 course was taught in Fall 1994 and Fall 1995 by Michael Meehan and in Fall 1996 by Barry Kurtz. Meehan's version of the course used the textbook The SR Programming Language: Concurrency in Practice, by Gregory Andrews and Ronald Olsson [1993]. Programming assignments included the Concurrent Pascal System, SR programs, and Parallaxis. Kurtz's version of the course used the textbook Parallel Programming: an Introduction, by Thomas Braunl [1993]. Programming assignments include the Concurrent Pascal System, our own Concurrency Simulator (described below), PVM (Parallel Virtual Machine) and Parallaxis.

Both versions start off with an overview of the field, using Flynn's taxonomy with various subcategories for SIMD and MIMD machines. Architectural support is discussed including various interconnection networks, such as mesh, torus, hypercube, etc. The balance of the course involves an in-depth study of SIMD and MIMD systems. Although SIMD systems are inherently simpler from an advanced viewpoint, for students at the sophomore level they require a substantial paradigm shift from sequential execution on a von Neumann architecture to a data parallel architecture. We have found it easier to extend the von Nuemann model of sequential execution to the MIMD architecture first before we have the paradigm shift necessary to discuss the SIMD model.

The classical concurrency mechanisms for synchronization and communications are covered thoroughly in CS240. The first programming assignment uses the Concurrent Pascal System (CPS) based on Ben-Ari's Pascal interpreter [Ben-Ari, 1982] that provides support for semaphore operations. This

assignment involves translating a monitor solution to a classic problem (e.g., readers-writers or dining philosophers) into the equivalent semaphore version and implementing it using CPS. In this way students gain insight into both the monitor model of concurrency and a semaphore-based implementation. The other two mechanisms of classical concurrency are rendezvous and remote procedure calls. Both versions of the course had assignments in this area: Meehan's assignment used the SR programming language while Kurtz's assignment used our recently developed Concurrency Simulator. Distributed computing based on either a network of UNIX workstations or PCs is introduced next. The programming assignment in Meehan's class used SR while the assignment in Kurtz's class used PVM extensions for C. The coverage of the MIMD model includes discussions of programming languages that have built-in concurrency mechanisms (e.g., Communicating Sequential Processes (CSP), OCCAM, Ada) and library extensions to existing sequential languages (e.g., fork and join techniques, PVM, Linda). Various approaches to algorithm development for coarse-grained problems are also discussed along with techniques for measuring speedup and efficiency.

The next major thrust is the full development of the SIMD model with a paradigm shift to data parallelism. The primary focus is on algorithm development with an emphasis on converting fine-grained algorithms in sequential form into parallel versions that can execute on an SIMD machine. The SIMD simulator, Parallaxis, is used in reinforce these concepts. Although we discuss some common problems (e.g. matrix multiplication) that can be implemented on both SIMD and MIMD machines, we also try to emphasize that problem granularity and the degree of data parallelism should be used to determine whether a MIMD or SIMD machine is most appropriate for the problem.

At the end of the course we try to spend two or three lectures discussing parallelism for nonprocedural languages that use the functional and logical paradigms. Our students at Tech have been introduced to these paradigms in our breadth-first overview of computing course [O'Neal, Kurtz, 1995] so they are able to grasp some of the issues involved in parallelizing these languages. Without this initial exposure to the nonprocedural paradigms in an earlier course it is doubtful if these materials would be meaningful at the sophomore level.

The faculty at Grambling were not able to implement the introduction to concurrency course at the sophomore level due to the structuring of their courses with a later introduction to data structures (a required prerequisite before the introduction to concurrency course). The first offering during the Spring 96 was in an existing course: CS425- Introduction to Parallel Processing. The next time this material is offered will be in a new junior level course following data structures. The CS425 course was taught by Jamal Alsabbagh and covered the following topics: the Flynn taxonomy, dynamic and static interconnection networks, embedding other networks onto the hypercube model, basic communications operations, performance and scalability, and various examples of parallel programs. Due to incompatibilities between existing computer systems and available software tools, there was no programming component to this course. These problems will be worked out before the next offering in a junior level course so that students can gain hands-on experience with parallel computing.

Advanced Course Modules: Assuming the introduction of basic concepts in a course like CS240, more advanced courses can introduce research in concurrency that is normally reserved for graduate level courses. Five modules lasting from two to three weeks are being developed to augment upper division courses in the following areas: algorithms, architecture, database, operating systems and programming languages.

The algorithms module is being developed by Mike O'Neal. The component stresses the PRAM model of parallel computing and attempts to develop algorithms in a machine-independent form. O'Neal has also developed an elaborate, Web-based introduction to the Parallaxis system that will prepare students to write programs using the Parallaxis SIMD simulator. This work was originally developed using HTML pages and is now being converted to use Java so that the presentation is more dynamic and not so "snapshot" oriented.

Mazin Yousif initialized work on the architecture module by focusing on instruction level parallelism. A sequence of 35 PowerPoint slides have been prepared to cover this topic in depth. These slides are currently being put into HTML format for viewing over the Internet. Chin Kim has continued the expansion of the architecture module to include high performance parallel architectures such as vector, SIMD, and MIMD machines. At least one representative example of each architecture is presented in detail.

The database module is being developed by Jamal Alsabbagh at Grambling where they have a two course sequence in database: CS451 and CS452. This module was field tested for the first time in the CS452 course during the Spring 1996. The topics include opportunities for parallelism in database processing, a comparison of alternative architectures for database systems, and the similarities and difference for handling data placement, query processing, and concurrency/recovery between parallel database systems (PDBs) and distributed database systems (DDBs). These materials will be refined and used again during the 1996-97 academic year.

The modules in distributed operating systems being developed by Meehan and programming languages being developed by Kurtz are not as well developed as the modules described above. Since Meehan and Kurtz had to devote their time to the entry level course (CS240) and to the software simulators (described below), they have had little time to fully develop these advanced modules. It is hoped that initial versions of these modules can be field tested at Tech during the 1996-97 academic year.

Summer workshops: Another major outcome for this grant proposal was a sequence of two summer workshops to be held in successive summers for faculty from computer science programs at predominantly undergraduate institutions. It is assumed participants in these workshops have little training in parallel computing and no research expertise in this area. It was initially planned to hold a summer workshop in the summer 1996. Announcements were sent to 13 schools in the regional area inviting faculty to a two week workshop scheduled for August 5-16. Only three applications were received. At the same time it was learned that Meehan was leaving Tech, so it was decided not to pursue the summer workshop for this year. After talking with various people, including Ratan Guha who has offered summer workshops in parallel computing at the University of Central Florida, it has been decided to offer one week workshops during the Summer 1997 and to offer multiple sessions to provide more flexibility for the participating faculty from other schools.

Software Development and Support Materials: A major component of this grant proposal is the development of an MIMD simulator that would make introduction of parallel processing concepts feasible at the sophomore level. This MIMD simulator was to be built on top of Meehan's DICE (Distributed Interpretive Computing Environment) software. After a year's work by Meehan and a full time graduate student, the development of the simulator itself had not started. This time and effort had been devoted to making the DICE software operational so that such a simulator could be developed. With the DICE approach showing little progress, Kurtz initiated an effort to develop a much less ambitious simulator based on Java. These efforts continued in parallel for the first six months of 1996. The DICE-based simulator was still making little progress and so when Meehan resigned from Tech, it was decided to abandon this approach as being overly ambitious. The more modest approach developed by Kurtz was continued and classroom tested for the first time in the Fall 1996.

The Concurrency Simulator is designed to support four types of communications and synchronization: semaphores, rendezvous, asynchronous message passing, and remote procedure calls This is a simulation and does not require multiple machines to execute. It is being developed in Java to insure platform independence. What makes this simulator unique is the coordination of a topology that is developed graphically using point-and-click construction with processes entered separately using a syntax-directed editor. The programming language is modest in complexity but supports expected programming constructs (if statements, while statements, for statements), simple data types (booleans, integers, strings, arrays), and basic mechanisms for concurrency (send, receive, call, conditional rendezvous, and semaphore operations). The coordination of the topology with the processes simplifies algorithm development. For example, for other systems trying to develop a distributed solution to the dining philosophers problem (say with five forks and five philosophers), the student has to number the philosophers and forks carefully so that modulus indexing can be used to insure proper communications. All of this is unnecessary using the Concurrency Simulator since this information is embedded in the graphical topology. This topology is also used to monitor program execution at runtime. The first version of the simulator with rendezvous has been tested with students in CS240 during the Fall 1996. The outcomes are currently being evaluated but observations have shown the simulator to be successful. Students were able to learn the system in a short period of time and use it to implement a parallel programming assignment based on rendezvous. The mechanisms for semaphores and asynchronous message passing have been developed but not field tested yet. The simulation of remote procedure calls

remains to be developed. Given this initial progress in development over a nine month period, we are confident the Concurrency Simulator can be completed successfully by the end of Summer 1997.

The other software aspect of this project is the development of support materials under the guidance of Mike O'Neal. The original proposal envisioned CD-ROM based multimedia software using a standard PC support package such as Macromedia's Director software. With the tremendous interest in the World Wide Web since the original proposal was written and with the development of the Java programming environment to support multimedia on the Web, we have switched approaches to a Java environment with plug-in modules for the most common browsers (Netscape and Explorer) to support multimedia development in a platform independent fashion. This change has focused our attention to the development of a Web site for examination of all materials (support for the entry level course, the advanced modules, and the Concurrency Simulator) rather than only use CD-ROM software for PCs. Physical software distribution will still be based on CD-ROMS, but the software itself will be platform independent and based on using a Web Browser.

Summary of the Current Status: We have designed a sophomore level course introducing concurrency and have taught it successfully three times at Louisiana Tech. A similar course has also been taught at Grambling, but not at the sophomore level. The modules in algorithms and database have been developed and class tested in their initial form (algorithms at Tech and database at Grambling). The modules in architecture, operating systems, and programming languages are under development and will be class tested at Tech during the 1996-97 academic year. The Concurrency Simulator has been class tested and, based on this experience, the outlook for complete development is bright. A fully integrated Web site is being developed and should be operational by the end of Summer 1997. We will also offer several one week workshops during the Summer 1997. Although initial progress on this grant had some setbacks, primarily due to faculty resignations, and progress has been slower than originally expected, the curriculum and software development efforts do appear to be on track and will be successfully completed in fully exportable form by the end of 1997.

References

Gregory Andrews and Ronald Olsson, The SR Programming Language: Concurrency in Practice,Benjamin Cummings, 1993

M. Ben-Ari, Principles of Concurrent Programming, Prentice Hall International, 1982

Thomas Braunl, Parallel Programming: an Introduction, Prentice Hall International, 1993

M. O'Neal, B. Kurtz, "Watson: A Modular Software Environment for Introductory Computer Science Education," SIGCSE Bulletin, Vol. 27, No. 1, March 1995, pp. 87-91