ASU and Intel Collaboration in Parallel and Distributed Computation

Violet R. Syrotiuk[1], Yinong Chen[1], Eric Kostelich[2], Yann-Hang Lee[1], Alex Mahalov[2], and Gil Speyer[3]
[1]School of Computing, Informatics, and Decision Systems Engineering
[2]School of Mathematical and Statistical Sciences
[3]ASU Advanced Computing Center
Arizona State University, Tempe, AZ

***Abstract*—Arizona State University (ASU) is working with Intel and the Intel Academic Community to integrate topics in Parallel and Distributed Computing (PDC) into the Computer Science (CS), the Computer Systems Engineering (CSE), and the Mathematics and Statistical Sciences (MAT) programs at the Undergraduate and Master's degree levels, leveraging ASU's initiative in high performance computing (HPC) in the Advanced Computing Center.**

I. Introduction

As more multicore processors are manufactured, their expected use in future computing has generated a demand in industry for developers to create software that leverages concepts in parallel and distributed computing (PDC) to take advantage of the processing power available. This increased applicability of PDC necessitates that our undergraduate and master's degree programs be restructured accordingly.

To begin to achieve this goal we have restructured standard curricula in some courses in our Computer Science (CS), Computer Systems Engineering (CSE), and Mathematical and Statistical Sciences (MAT) programs to integrate modules related to PDC (see §II). However, we already see that there are far more opportunities to restructure courses to integrate topics of PDC into the curriculum [1].

An ad hoc committee is examining all courses in the curriculum looking for opportunities to integrate "thinking in parallel" without the need for removing material. In Spring 2011 the committee has proposed updates to our 100-200 level course syllabi, with approval pending by the undergraduate curriculum committee and program faculty. In Summer 2011, work on integrating topics in PDC into our higher level (300-400) courses will take place. As well, a new 400-level CSE course on "Parallel and Distributed Computing" is being developed. In addition, a "data bank" of course materials, including lectures, sample programming projects and solutions, will be prepared to help instructors adopt the new syllabi. This work will be shared with the Intel Academic Community and highlighted on ASU web pages.

II. Courses in Early Adoption

1. ASU 101-CSE: The ASU Experience

Description: ASU 101 is the first course all ASU students take in their freshman year. The CSE version consists of a sequence of modules to introduce the discipline of Computer Science and Engineering. To illustrate multicore architectures and parallel and distributed computing, Intel games demonstrating multicore performance, such as "Horsepower," will be shown to the students. In this game, the performance of a multithreaded ambient animation process is enhanced when run on a multicore CPU. A performance metric of "Horsepower" is the horse count for a fixed frame rate. The CPU utilization history also shows the load balance on each core when the application is threaded.

Evaluation plan: A lab environment will be set up for students to play with Intel games. Students will configure the computer to run on a different number of cores and record the performance.

2. CSE 310: Data Structures and Algorithms

Description: This is a course in advanced data structures and algorithms, and algorithmic analysis. It includes heaps, hash tables, red-black trees, and graphs among other data structures. Algorithms for sorting, selection, elementary graph algorithms are studied. Algorithmic techniques, e.g., divide-and-conquer, dynamic programming, and greedy, are applied. The impact of the choice of data structure on the algorithmic complexity (time and space) is discussed.

To date, all the algorithms discussed are sequential. We plan to follow the suggestions in the NSF/TCPP curriculum initiative report [2] for the course Data Structures and Algorithms (DS/A) in CSE 310; the augmented course will pilot in Fall 2011.

Evaluation plan: To date, programming projects in C/C++ are given in CSE 310 to give students experience implementing various algorithmic techniques and see, first hand, the impact of the choice of data structure on the run time. Programming projects can be augmented to include the implementation of a parallel formulation of a problem, e.g., merge sort, depth-first search, graph algorithms (e.g., Dijkstra's single-source shortest path and Floyd's all-pairs shortest path algorithms).

3. CSE 430 : Operating Systems

Description: This course studies operating system structure and services, emphasizing concurrent processes (e.g., critical sections, race conditions, threads, mutual exclusion and synchronization, semaphores, concurrent programming paradigms) and deadlocks using Intel's Parallel Studio, game demos, and tools. Other topics include processor scheduling, main and virtual memory, file system interface and implementation, mass-storage structure and I/O systems, and protection.

We previously gave projects using the POSIX threads library. However, there were no effective tools to support students in developing or debugging their multithreaded programs. In the restructured course, a new module is introduced to review basic software tools for use on multicore systems, such as performance analyzers, with demonstrations in class. A sample program is developed to explore how to implement parallelism through threading, and how to check the validity of the code with software tools to avoid problems such as data races.

Evaluation plan: The learning outcomes will be evaluated in homework assignments. As well, one or two programming projects using tools in Intel's Parallel Studio to use pragmas, examine data locality, load balancing, acceleration, profiling, and similar, will be given.

4. CSE 445/598: Distributed Software Development

Description: In a service-oriented distributed system, applications at a server may be invoked by multiple clients. Such applications can utilize multicore architectures and threading techniques to improve service response time and to reduce resource contention. Intel's Thread Building Blocks (TBB) are discussed in class for server application design. TBB provides a straightforward way of introducing performance threading, by turning synchronous calls into asynchronous calls and converting large methods (threads) into smaller ones. To demonstrate the performance impact, a program that validates the Collatz conjecture has been used to evaluate the performance in a single core up through 32 cores using Intel Manycore Testing Lab (MTL). Figure 1 shows the measured speed-up and the usage efficiency for 4, 8, 16, and 32 cores with respect to a single core.

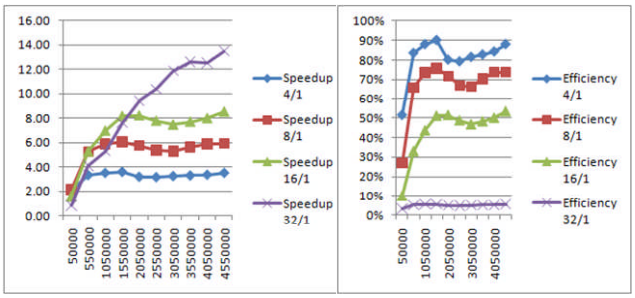

Figure 1. Speedup and efficiency

Evaluation plan: A number of assignments are planned. In the multithreading programming assignment, students will test their program in both single core and multicore environments. Students will also revise the program using Intel's TBB library and measure the speed-up. In the service hosting assignment, students will explore parallelism on the server side and determine the performance improvement based on the service model and the multicore efficiency.

5. CSE 494/598: Introduction to Parallel Programming

Description: CSE 494/598 aims to introduce students to the fundamentals of parallel computing and to important techniques and practices in parallel programming. Various models and applications will be discussed. In addition, a sampling of current topics in HPC will be covered.

Students will gain skills in OpenMP, MPI, parallel I/O, as well as some exposure to GPUs through CUDA. Students will develop code implementing various paradigms and learn how to debug, benchmark, and tune code on a parallel system using software tools. Finally, students will become familiar with the latest approaches and strategies used in research and industry through a research project, guest lectures, and application-specific discussions.

Evaluation Plan: The midterm will evaluate understanding of many basic concepts. Four projects will ensure application: (1) OpenMP. Students will implement pragmas and examine data locality, load balancing, acceleration, profiling, etc. (2) MPI. Students will apply collective communication and data decomposition. (3) MPI. Students will apply parallel I/O, local communication and a parallel library. (4) CUDA. GPU programming.

6. MAT 420: Scientific Computing

Description: This course is directed at majors in applied mathematics and the physical sciences for students who plan to pursue internships or research projects at the undergraduate or graduate levels that involve a significant amount of numerical computing. The emphasis is on programming tools and paradigms for high-performance computing.

Topics have included Fortran 95 and C++ (with an emphasis on the Standard Template Library (STL)) and major scientific libraries, such as LAPACK. In the restructured course, first piloted in Fall 2010, an introduction to OpenMP and MPI were included. Further refinements are anticipated in Fall 2011, including the implementation of LAPACK in the Intel Math Kernel Library (MKL) and coarrays in Fortran 2008.

Evaluation Plan: Students are required to demonstrate mastery of each topic in a suitable programming exercise. Examples include the use of Fortran 95/2008 vector constructs to compute the Mandelbrot set; OpenMP to parallelize a partial differential equation solver using finite differences; and the basic blocking MPI_Send and MPI_Recv calls to implement a PDE solver on a distributed memory cluster.

III. SUMMARY

The introduction of PDC concepts into select courses has had encouraging results. We are now working through our curriculum systematically to establish new learning outcomes, develop content, and evaluation methods.

IV. ACKNOWLEDGEMENTS

Thanks to Intel Corp, for their support and collaboration.

V. REFERENCES

[1] ASU SCIDSE, http://engineering.asu.edu/cidse/Curriculum

[2] S.K. Prasad et al., "NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing: Core Topics for Undergraduates," Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE'11).