Early Adopter: Integration of Parallel Topics in the Undergraduate Curriculum at Calvin College

Joel C. Adams
Department of Computer Science, Calvin College,
Grand Rapids, MI 49546 USA
adams@calvin.edu

***Abstract* - Since 1989, all Calvin College computer science students have learned about concurrency constructs and distributed systems, and they have had the option of learning about parallelism since 1997. In 2006, manufacturers began releasing processors with multiple cores instead of faster clock speeds, making knowledge of shared-memory parallelism a necessity for all computer science students. In 2008, the department began integrating shared-memory parallel topics into its *Data Structures* course (aka CS2) and the *Operating Systems and Networking* course. Thanks to the NSF/IEEE TCCP 2011 Early Adopters Program, additional parallel topics are now being integrated into the *Algorithms and Data Structures* course, the *Intro to Computer Architecture* course, the *Programming Language Concepts* course, and the *High Performance Computing* course. This work provides an overview of the department's curriculum, and the precise courses in which specific parallel topics and technologies are covered.**

I. Background

Calvin College is a 4-year comprehensive liberal arts college of about 4,000 undergraduate students located in Grand Rapids, Michigan. In addition to strong liberal arts programs, the college offers several professional programs, including ABET-accredited programs in Computer Science (BCS) and Engineering (BSE). Its Department of Computer Science (CS) offers degrees in computer science and information systems, has roughly 100 students and seven full-time professors. All students complete a large general education requirement and at least one major program.

In semester 1, computer science majors take *Introduction to Computing*, a CS1 course taught in Java; and *Calculus I*. In semester 2, they take *Data Structures*, a CS2 course taught in C++; and *Calculus II*. In semester 3, they take *Data Structures and Algorithms*, a CS3 course in which students can use either Java or C++; *Intro to Computer Architecture*; and *Discrete Mathematics I*. In semester 4, they take *Programming Language Concepts*, and *Discrete Mathematics II*. In semester 5, they take *Software Engineering*; and the first of their advanced electives. In semester 6, they take *Operating Systems and Networking*; *Statistics*; and perhaps another advanced elective. In semester 7, they choose another advanced elective; and begin their *Senior Practicum*, which may be a 2-semester senior project or a 1-semester internship. In their final semester, they complete their final advanced elective; their *Senior Practicum*; and a *Perspectives on Computing* capstone course dealing with social and ethical issues.

II. Concurrent/Distributed Coverage At Calvin

Calvin's CS department has been teaching undergraduates about concurrent and distributed computing for many years. Department chair Joel Adams' dissertation area was distributed systems, and as the first PhD computer scientist at Calvin, he influenced the early CS curriculum development.

As a result, Calvin's *Programming Language Concepts* course (CS 214) has since 1989 covered the use of tasks, semaphores, locks, condition variables, and monitors; threads were added in 1998. Likewise, Calvin's *Operating Systems and Networking* course (CS 232) has since 1989 covered the implementation of processes, threads, semaphores, locks, condition variables, and monitors; plus distributed systems topics like temporal logic, global clocks, consensus, and so on.

Both of these courses have been updated regularly; for example, CS 232 was updated with coverage of TCP/IP networking and client-server systems in 1996 and POSIX multithreading in 1998. In each case, topics were prioritized and coverage of a lower-priority topic was reduced.

Both of these courses are part of the department's core curriculum (i.e., all CS majors must take these courses), so all of Calvin's CS majors have learned about concurrency, multithreading, and distributed systems for many years.

III. Parallel Coverage At Calvin

In the summer of 1996, Dr. Adams attended an NSF parallel computing workshop at Colgate University, where he learned about different aspects of parallelism, including the message passing interface (MPI). In 1997, he offered an advanced elective course *High Performance Computing* (CS 374, aka HPC), which has since then been offered every other year. The course initially featured coverage of multiprocessor architectures, parallel algorithms, scalability, Amdahl's and Gustafson's Laws, with hands-on lab and homework exercises using Parallaxis [6] in Modula-2 and MPI in C/C++.

Over the years, coverage of Parallaxis was gradually eliminated to make room for other topics, including Beowulf clusters (1999), grids (2001), OpenMP (2003), and Google's MapReduce technology (2009).

Initially, students worked their MPI exercises on a network of workstations (NOW), but timing results were unreliable. To provide reliable timing, Dr. Adams and his students have built a series of Beowulf clusters, including *MBH'99* [1], *Ohm.calvin.edu* [2], *Sleipnir*, *Microwulf* [3,4], and *Dahl.calvin.edu* [5]. Some of these clusters were built using grants from the NSF *Major Research Instrumentation* (MRI) program; others were built using internal funds.

Calvin's CS majors have thus had the option of learning about parallelism since 1997.

IV. The Multicore Challenge

In 2006, manufacturers began releasing multicore processors instead of processors with faster clock speeds. With these changes, it became apparent that *all* of our CS majors needed to learn how to achieve speedup through parallelism (especially shared-memory parallelism), not just those choosing our HPC elective. Moreover, the radically different thought process needed to design parallel software made it apparent to us that this material should be integrated early and at appropriate points throughout our curriculum, to prevent our students from becoming locked into a sequential mindset.

We decided to begin by encouraging faculty members to start adding parallel content to their courses. This brought about two changes:

- 2008: In our *Intro to Data Structures* course (core, year 1, semester 2), we added: (i) a lab exercise where students use OpenMP threads to solve embarrassingly parallel problems (matrix addition and transpose), and time their computations using differing numbers of cores; and (ii) lectures on simple race conditions, and how algorithms such as linear search and merge sort can be parallelized.
- 2009: In our *Operating Systems and Networking* course (core, year 3, semester 2), we added: (i) approaches to scheduling on systems with multicore processors; and (ii) a comparison of the POSIX and OpenMP libraries, in terms of thread creation, synchronization constructs, etc.

These changes were sufficient to provide all of our students with a minimal exposure to shared-memory parallelism. However, Dr. Adams taught both of these courses; other faculty members had not yet integrated parallel topics into their courses, mainly because their textbooks were not doing so.

V. Going Further

In 2010, the NSF/IEEE TCCP group began their Early Adopter Program. Dr. Adams applied for and received one of these grants on behalf of Calvin's CS department, to provide financial incentives for faculty to integrate more parallelism into their courses. Thanks to this funding, the following changes are occurring during 2011:

- In our *Algorithms and Data Structures* course (core, year 2, semester 1), we are adding coverage of parallel algorithm design (e.g., data vs task decomposition), specific parallel algorithms (e.g., parallel maximum), distributed algorithms (e.g., graphs), parallel speedup, parallel efficiency, and asymptotic analysis.
- In our *Intro to Computer Architecture* course (core, year 2, semester 1), we are adding coverage of multicore processors and their implications for main memory, caching, bus bandwidth, memory controllers, etc.
- In our *Programming Language Constructs* course (core, year 2, semester 2), we have expanded our coverage of shared-memory parallelism (OpenMP, Ada multitasking, synchronization mechanisms, race conditions) and added an introduction to distributed memory parallelism (MPI). We have also added a lab exercise comparing synchronization mechanisms (OpenMP's reduction, Ada's entry procedures, and Ruby threads' join method); in which students measure scalability by varying the problem size and the number of threads/cores being used.
- In our *High Performance Computing* course (elective, year 3 | 4, semester 1), we are adding a module and lab exercise to introduce OpenCL/GPGPU computing, to supplement our coverage of MPI, OpenMP, and MapReduce.

By spreading shared-memory parallel topics throughout the undergraduate core curriculum, our students will be exposed to parallelism early and frequently. We believe that by combining conceptual foundations with hands-on experiences, our students will gain the knowledge and experience they need to take advantage of today's multicore processors and tomorrow's manycore processors.

VI. Discussion

Even in a relatively small department like ours, it can be difficult for busy faculty to change what they teach. We are grateful to the NSF/IEEE TCCP Early Adopter Program for providing funding we can use as financial incentives to drive such change. We anticipate that as faculty members become more familiar with parallelism and gain confidence in it, they will naturally increase their use of it in the courses they teach.

VII.

Acknowledgment

Calvin's Department of Computer Science thanks the NSF/IEEE TCCP group for making this work possible.

References

[1] J. Adams, W. D. Laverell and M. Ryken, "MBH'99: A Beowulf Cluster Capstone Project", *14th Annual Midwest Computer Conference*, Whitewater, WI, March 2000.

[2] J. Adams and D. Vos, "Small College Supercomputing: Building a Beowulf Cluster at a Comprehensive College", *33rd SIGCSE Technical Symposium on Computer Science Education*, Covington, KY, February 2002, pp. 411-415.

[3] J. Adams and T. Brom, "Microwulf: A Beowulf Cluster For Every Desk", *39th SIGCSE Technical Symposium on Computer Science Education*, Portland, OR, March 2008, pp. 121-125.

[4] J. Adams, T. Brom*, and J. Layton, "Microwulf: Breaking the $100/GFLOP Barrier, *Cluster Monkey*, Aug 2007, Online: http://www.clustermonkey.net/content/view/211/1/, accessed April 2011.

[5] J. Adams, K. Hoogeboom, J. Walz, "A Cluster for CS Education in the Multicore Era", *42nd SIGCSE Technical Symposium on Computer Science Education*, Dallas, TX, March 2011, in press.

[6] T. Braunl, "Parallaxis-III: A Structured Data-Parallel Language", Online: http://robotics.ee.uwa.edu.au/parallaxis/, accessed April 2011.