Curriculum Initiative on Parallel and Distributed Computing at the University of Western Ontario

This proposal describes the on-going integration of high-performance computing courses in the undergraduate curriculum of the computer science department at the University of Western Ontario (UWO) Canada. [http://www.uwo.ca/]

1 Background.

The UWO Computer Science Department offers bachelor's, mater's and PhD degrees in computer science. [http://www.csd.uwo.ca/] The department is home to over 100 undergraduate students and 120 graduate students. The research interests of the 28 Faculty members cover a large range of topics in HPC and their industrial partners include companies like IBM, Intel, Microsoft, Maplesoft. The university of Western Ontario is home of SHARCNET, a consortium of Canadian academic institutions who share a network of high performance computers. [www.sharcnet.ca/]

2 Integration Plans.

As many other computer science departments, we are implementing the necessary transformations of our curriculum that the democratization of High Performance Computing (HPC) requires.

Algorithms are often taught with algebraic complexity as the main complexity measure and with serial running time as the main performance counter. On modern computers minimizing the number of arithmetic operations is no longer the primary factor affecting performance. Effective utilization of the underlying architecture can be much more important. Therefore, many traditional courses need to evolve and integrate the different and related aspects of HPC, among them, parallel processing, distributed computing, computer architecture, models of computation, code optimization, performance analysis, etc.

This transformation of our curriculum will certainly go through several iterations. We first describe the courses of our new *Minor in High Performance*, which will be offered for the first time this academic year. [http://www.westerncalendar.uwo.ca/2011/pg1602.html]

Then, we present two other courses that shall extend our HPC course offer targeting the needs of our university. In many ways, the design of the aforementioned minor and planned courses follow the *NSF/TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduate.* [http://www.cs.gsu.edu/~tcpp/curriculum/sites/default/files/NSF-TCPP-curriculum-Dec23.pdf]

Our current HPC courses reflect three types of transformation.

- Integration of an HPC component into pre-existing non-HPC courses: This is the case of our *CS 2101 – Foundations of Programming for High Performance Computing* which used to be an introductory course to C-UNIX programming and which will now include an emphasis on performance issues, in particular data locality, together with an initial exposure to multithreaded programming.

- Creation of new HPC courses: Our *CS 3101 – Theory of High Performance Computing* is the first instance of such courses. The motivation is two-fold. First, making efficient use of computers implies to understand how to measure and reach efficiency, which naturally requires a theoretical framework. On the other hand, measuring and reaching performance on modern architectures also requires a deep understanding of the hardware together with an ability to use complex performance analysis software tools. Thus, this course will combine these two aspects, relying on the programming experience that students have acquired with CS 2101.

- Natural evolution of pre-existing HPC courses: This is the case of our *CS 4402 – Distributed and Parallel Systems* which used to be primarily oriented toward the *message passing* distributed computing model and which is now including the *fork-Join* multithreaded programming model (targeting multicores) and the *single instruction multiple data (SIMD)* programming model (targeting GPGPUs).

These courses are also adapted to the diverse student populations that we need to serve. CS 2101 is designed in a way that non-CS students can take this course while CS 3101 and CS 4402 are primarily for CS students with the appropriate background knowledge in C/C++ programming, data-structures and algorithms. Based on this, taking CS 2101 will not be sufficient for a student to be able to write optimized code; this is only a first exposure to the idea of HPC.

The other two planned courses in our HPC offering for undergraduate students are listed below. Among other objectives, the first one aims at preparing students to the UWO Graduate Program in Computational Science (GPCS)
[http://www.apmaths.uwo.ca/gpsc/]

- High Performance Scientific Computing: This course will help students with interest in scientific computing (say students from our departments of Applied Mathematics, Biology, Chemistry, Engineering, Physics and Astronomy) to learn how to design algorithms and write code, which is efficient in terms of data locality and parallelism. Students will also study fundamental scientific library software.

- Advanced topics in High Performance Scientific Computing: This course will provide a comprehensive overview of the hot technologies in parallel and distributed computing. Thus, this course would probably serve mostly CS students looking for jobs in the IT Sector.

2.1 CS 2101 – Foundations of Programming for High Performance Computing. Following the notations of the *NSF/TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduate*, this course will be a one-term course in between CS1 and CS2. Indeed, we will assume that students have already taken an introductory CS course and have experience with an imperative programming language. However, the course will start with a 4-week overview of C-UNIX. At the same time, students will be introduced to performance issues, in particular memory traffic. The next 6 weeks will be dedicated to an introduction to multithreaded programming, using `Cilk` [http://supertech.csail.mit.edu/cilk/]. Indeed, this language is an extension of C (with essentially three additional keywords) and its running system comprises a dynamic scheduler. This will allow students to focus on exposing parallelism in their programs. However, notions from scheduling (dependencies, task graphs, work and span, work stealing principle) together with performance metrics (speedup, efficiency, scalability) will be presented to the students. Matrix multiplication, matrix transposition, counting sort and merge sort will illustrate the topics of the course. The last two weeks, students will be taught how SIMD programs are written and how they execute.

2.2 CS 3101 – Theory of High Performance Computing. This course is built on the principle that, in parallelism, the models of computation that have a practical value are those which bear a relationship to what happens in hardware. Thus this course will interleave the following topics from computer architecture and algorithms.

- Architecture topics:
  - Highly data parallel models (superscalar, pipeline, stream, vector, and dataflow)
  - Control parallelism models (multithreading, multiprocessing, multicore, cluster, and grid/cloud)
  - Memory hierarchy (cache organization, etc.) and issues with data traffic (latency, bandwidth, false sharing, memory contention, etc.)
  - Performance analysis (cache misses, cycles per instruction, arithmetic intensity, effective memory bandwidth, burden parallelism, speedup, etc.).
- Algorithm topics:
  - Models (BSP, Cilk) and complexity (cache complexity, work and span)
  - Paradigms (divide & conquer, series-parallel composition)
  - Patterns (broadcast, scatter/gather, MapReduce, stencil computation)
  - Concurrent data structures
  - Synchronization (mutual exclusion mechanisms, lock-free synchronization techniques, termination detection leader election, etc.) and scheduling
  - Fundamental applications (linear algebra, sorting, graph algorithms).

2.3 CS 4402 – Distributed and Parallel Systems. This course is dedicated to the programming aspects of HPC. Students will study in depth three concurrency platforms that illustrate the main stream programming paradigms in distributed and parallel computing:

- CilkPlus for fork-join multithreaded programming targeting multicores
- CUDA for data parallelism on GPUs
- MPI for message passing distributed computing.

The course also gives an overview of other popular concurrency platforms such as PThreads, OpenMP and OpenCL.

2.4 High Performance Scientific Computing. This course will extend CS 3101 and CS 4402 with traditional topics in scientific computing with a focus on HPC aspects. In addition, we will assume that

students have a background knowledge in mathematical sciences (linear algebra, numerical computing) and wish to learn how to efficiently implement their favorite algorithms on modern parallel architectures. This course will explore implementation techniques for optimizing code in terms of data locality and parallelism. It will also present the software architecture of fundamental libraries in high performance scientific computing, among them:

- *ATLAS (Automatically Tuned Linear Algebra Software)* [http://math-atlas.sourceforge.net/]
- *ScaLAPACK* [http://www.netlib.org/scalapack/scalapack_home.html]
- *CULA, a set of GPU-accelerated linear algebra libraries* [http://www.culatools.com/]
- *FFTW* [http://www.fftw.org/]
- *SPIRAL* [http://www.spiral.net/index.html]
- *The Pochoir Stencil Compiler* [http://people.csail.mit.edu/yuantang/]
- *Maple Grid Computing* [http://www.maplesoft.com/products/toolboxes/gridcomputing/index.aspx]
- *MATLAB-PCT, MATLAB and Parallel Computing Toolbox* [http://www.mathworks.com/products/parallel-computing/index.html]

2.5 Advanced topics in High Performance Computing. This course will extend CS 3101 and CS 4402 with advanced topics (mainly in the area of distributed computing) that are of interest in the IT Industry. These topics are: concurrency and non-determinism, power consumption, cluster, cloud and grid computing, fault tolerance, security in distributed systems, embedded systems, distributed transactions, web search.

3 Evaluation Plans.

For our Minor in High Performance Computing, we present our evaluation plans, at both the student level and the instructor level.

1. First of all, and in addition to the traditional instructor evaluations and course surveys, we value greatly the spontaneous feedback letters from the students. For its Winter 2011 edition (the first one with the contents listed above) the instructor of CS 4402 received several such letters.
2. Secondly, for a course like CS 4402, where the final exam is replaced by individual and pairwise different projects, the student-instructor interaction during the project development provides another way to measure the course impact.
3. For courses, with several programming assignments (CS 2101, CS 4402) one can also easily analyze the students' progress (for instance in terms of code performance) by running appropriate software tools.
4. Evaluating the impact of CS 3101 (for which the portion of programming exercises in the assignments will be limited) will be done differently. Indeed, the first edition of CS 3101 will be taught during the Winter 2013. Thus, until that term, CS 4402 integrates part of the CS 3101 materials. After that, we hope to see that the students will be much more at ease at programming CilkPlus, CUDA or MPI, when they have acquired the appropriate background knowledge in architecture and algorithms during a previous course, namely CS 3101.
5. We are also looking forward to see the impact of these courses on students from other departments than the Computer Science Department. We hope that the number of such students will increase regularly and that our Department could become a *Service Department* for teaching HPC on our campus.

Appendix A HPC courses taught in our department

- *CS 2101 – Foundations of Programming for High Performance Computing*
  [http://www.csd.uwo.ca/~moreno/CS2101-1112.html]
- *CS 9535/ CS 4402 – Distributed and Parallel Systems*
  [http://www.csd.uwo.ca/~moreno/CS9535-4402-1112.html]
- *CS 9610 – Topics in Distributed Systems*
  [http://www.csd.uwo.ca/faculty/hanan/610/index.html]
- *CS 9624 – High Performance Computing: From Models of Computation to Applications*
  [http://www.csd.uwo.ca/~moreno/CS98xx-HPC-1011.html]

Appendix B Other courses related to this proposal taught in our department

- *CS 2208 – Fundamentals of Computer Organization*
  [http://markdaley.org/cs2208/]

- *CS 2210 – Data Structures and Algorithms*
  [http://www.csd.uwo.ca/courses/CS2210a/]
- *CS 2211 – Software Tools and Systems Programming*
  [http://www.csd.uwo.ca/courses/CS2211a/]
- *CS 3340 – Analysis of Algorithms I*
  [http://www.csd.uwo.ca/courses/CS3340b/]
- *CS 3305 – Operating Systems*
  [http://www.csd.uwo.ca/courses/CS3305a/]
- *CS 3350 – Computer Architecture*
  [http://www.csd.uwo.ca/courses/CS3350b/]
- *CS 3357 – Computer Networks I*
  [http://www.foxhollow.ca/cs3357/]
- *CS 3388 – Computer Graphics I*
  [http://www.csd.uwo.ca/courses/CS3388a/]
- *CS 4457 – Computer Networks II*
  [http://www.csd.uwo.ca/courses/CS4457b/]