Blue Waters Educational Allocation

Esteban Meneses
University of Pittsburgh
emeneses@pitt.edu

Final Report

Summary

The Computer Science Department of the University of Pittsburgh offered the class *CS1645: Introduction to High Performance Computing Systems* during Spring 2014. This course covers the fundamentals of parallel computing and programming languages for supercomputing. It represents a unique opportunity for students at the undergraduate level to get first-hand exposure to the main technologies in the high performance computing (HPC) world and to the most prominent problems faced by the HPC community. Blue Waters was an ideal resource for this class because it provides in one single system the most advanced HPC features in terms of architecture, programming languages and HPC libraries. The class included 3 programming assignments that explored OpenMP, OpenAcc, and MPI.

Team Members

Principal Investigator
Name: Esteban Meneses, PhD
Title: Research Assistant Professor
Institution: University of Pittsburgh
Email: emeneses@pitt.edu

Secondary Instructor
Name: Albert DeFusco, PhD
Title: Research Assistant Professor
Institution: University of Pittsburgh
Email: defusco@pitt.edu

Teaching Assistant
Name: Angen Zheng
Title: Research Assistant
Institution: University of Pittsburgh
Email: anz28@pitt.edu

Student List

The course was officially registered by 33 students, coming mostly from the departments of Computer Science (CS), and Electrical and Computer Engineering (ECE) at the University of Pittsburgh. Table 1 presents the list of all the registered students.

Table 1: Registered Student in *CS1645: Introduction to HPC Systems*

Name	Major	Level	Institution
Alexander Blanck	ECE	Undergraduate	University of Pittsburgh
Alexander Quinn	CS	Undergraduate	University of Pittsburgh
Anthony Jack	CS	Undergraduate	University of Pittsburgh
Barry Aarons	CS	Undergraduate	University of Pittsburgh
Britanny Barry	ECE	Undergraduate	University of Pittsburgh
Christian Guerrero	CS	Undergraduate	University of Pittsburgh
Christopher Rogers	CS	Undergraduate	University of Pittsburgh
Conor Freeland	ECE	Undergraduate	University of Pittsburgh
Corey Furmanski	CS	Undergraduate	University of Pittsburgh
David Lang	ECE	Undergraduate	University of Pittsburgh
Edward McCluskey	CS	Undergraduate	University of Pittsburgh
Eric Gratta	CS	Undergraduate	University of Pittsburgh
Eric Wiegandt	CS	Undergraduate	University of Pittsburgh
Haleigh Wright	CS	Undergraduate	University of Pittsburgh
Huang Junyang	ECE	Undergraduate	University of Pittsburgh
James Castiglione	CS	Undergraduate	University of Pittsburgh
James McMillian	CS	Undergraduate	University of Pittsburgh
James Vento	ECE	Undergraduate	University of Pittsburgh
John Abraham	ECE	Undergraduate	University of Pittsburgh
Jonathan Doron	CS	Undergraduate	University of Pittsburgh
Joshua Zwolan	CS	Undergraduate	University of Pittsburgh
Justin Loutsenhizer	CS	Undergraduate	University of Pittsburgh
Kevin Ireland	CS	Undergraduate	University of Pittsburgh
Kevin Yealy	ECE	Undergraduate	University of Pittsburgh
Kyle Walters	CS	Undergraduate	University of Pittsburgh
Michael Appel	CS	Undergraduate	University of Pittsburgh
Oscar Prom	ECE	Undergraduate	University of Pittsburgh
Steven Comer	ECE	Undergraduate	University of Pittsburgh
Thomas Molinari	ECE	Undergraduate	University of Pittsburgh
Tyler Wolf	ECE	Undergraduate	University of Pittsburgh
Tyler Raborn	CS	Undergraduate	University of Pittsburgh
Xing Yin	CHEM	Graduate	University of Pittsburgh
Zachary Koopmans	ECE	Undergraduate	University of Pittsburgh

The students were all impressed by the sheer magnitude of a supercomputer like Blue Waters: *What type of applications do they run on that machine?* was a common question at the beginning of the semester. The other fundamental question was *How do you write programs for such machine? Do you use sockets?*. The course tried to answer both questions, but we really focused on the latter. Using the online Blue Waters tutorial, the user help desk (when strictly necessary), and the publicly available documentation on traditional HPC tools, the students had a smooth experience running on Blue Waters. They were excited that a single specialized system could provide a wide range of programming tools. But, what was even more exhilarating was the type of speedup it is possible on Blue Waters. *Using accelerators (GPUs in this case) can really execute certain codes in a snap*, was a common opinion in the class. The other source of speedup

was the size of jobs that could be submitted to Blue Waters. When a well-written parallel program scale to hundreds of processors, then it is possible to substantially decrease the execution time.

Learning Outcomes

One particular feature of Blue Waters that was useful for the goals of the course is the hybrid nature of the machine. Since multicore nodes coexist (at least in one portion of the machine) with accelerators (GPUs), then it is possible to teach the main programming languages in HPC on a single system. The class covered OpenMP, OpenAcc, and MPI, all over the same system and using the same software configuration. This features greatly simplifies the design of the class, because avoids a second learning curve on a different supercomputer.

We present a list of the lessons learned throughout the semester-long course:

- The asynchronous execution model of supercomputers requires a change in the mindset of the students. A job does not necessarily start execution immediately after submitted. It all depends on the scheduling policy and the current circumstances of the machine at the time of submission. There are two direct implications of that fact. First, the student has to plan ahead to develop, test, and execute his code. Second, having interactive allocations simplifies the debugging process.

- Compilers are a wonderful piece of technology, but they can not match the ability of a skilled parallel programmer (yet?). What directive-based programming languages do is to provide hints for the compiler on how to parallelize the code. In the case of OpenAcc, the compiler carries out a very intricate analysis on data dependencies to provide a safe code transformation for parallelization. However, it is fundamental to have a good understanding of the basic parallel programming principles to come up with a good parallel program.

- Parallel programming is an art. As much as instructors teach principles, algorithms, and theorems, the practice makes the master. All parallel programming tools offer a rich set of possibilities to implement different algorithms to solve the same problem. Even for the same algorithm, there are several variants and implementation alternatives. Therefore, it is imperative to be exposed to as many examples as possible to get a good grip on the techniques to optimize a parallel code.

- Scalability comes at a price. The MPI standard was, in general, more involved than OpenMP when the task at hand was to parallelize a serial algorithm. Often times, it requires to rethink the whole solution procedure to adjust the algorithm to the workings of message passing. But, persistence pays off. In one of the machine problem homeworks, students were asked to implement a particle-interaction algorithm using MPI and scale the program to 1,024 cores. Although there is some satisfaction in running such large jobs, the lesson is usually clear, scaling programs requires effort and the use of appropriate tools.

Highlights

Parallel programming skills are becoming more important in the information technology industry. From large database applications to sensor network systems, dealing with concurrency is fundamental. This course provided the students with a tool chest of algorithms, techniques, and tools to face a wide range to problems. The class emphasized the use of mature programming languages that enjoy wide acceptance in the HPC community. Those tools allow programmers to exploit massive supercomputing platforms such as Blue Waters.