Advancing science through education in high performance computing

Greg Walker and Alan Tackett
Department of Mechanical Engineering
Physics & Astronomy Department
Vanderbilt University

Abstract

High-performance computing (HPC) platforms are becoming increasingly accessible to scientists and engineers due the remarkable decrease in commodity hardware costs. The promise of HPC allows engineers to perform more detailed analysis of complex systems in shorter times. Ultimately, design cycle times can be reduced and reliability can be increased by utilizing new HPC facilities. However, barriers to effective use of existing and emerging HPC technologies remain. In fact, few researchers and engineers possess the knowledge to benefit from the current computing capabilities. In response to this unheralded demand, a pilot course for exposing engineering students to new technologies and capabilities in the computing world has been developed. As a result, not only have student participants become HPC savvy, but also the research community as a whole has expressed intense interest in the continuation and expansion of the initial project. This surge in interest is derived from the fact that student participants have been able to solve problems that were previously not considered because of their computational requirements. In other words, science has been advanced because of this single class offering.

Introduction

Until recently, high-performance computing was the exclusive purview of highly specialized research programs with large government grants.[1] Further, the applications deemed worthy of such large-scale facilities and resources were usually defense related. As a result, a cloud of mystery has surrounded scientific communities involved in development and implementation of both hardware and software devoted to solving computationally intense problems. Because of the tremendous expense of building and operating high-performance computing facilities, resources were scarce, and many researchers did not have the luxury of being able to consider scaling up their own projects.

Currently, high-performance computing centers are appearing in almost every university and government lab. In fact, many individual researchers and small research groups are acquiring their own mini-cluster. While the dramatic drop in the cost of commodity hardware has made the migration of high-performance computing equipment to smaller programs possible, the real enabling technology is largely software.[2] For example, legacy systems consist primarily of proprietary hardware and software, each with its own set of strengths and weaknesses. Development on such monoliths requires a very specialized knowledge of the particular system, so technology was slow to disseminate through the community. The eventual development and maturity of open standards has allowed the utilization of high-performance hardware without substantial investment in narrowly focused austere training on proprietary systems.

This rapidly changing landscape of high-performance computing presents its own challenges. Specifically, the abundance of available hardware does not necessarily translate to widespread or effective utilization of resources. In fact, experience has shown that few computational researchers understand the capabilities and limitations of modern high-performance equipment. Further, the expertise to take full advantage of existing resources is poor. This deficit of HPC-literate researchers stems directly from the fact that researchers in high-performance computing fields have been historically isolated from mainstream research.

The present work describes development of a course that is designed to address the computational researcher's needs. While the course is not unique, its approach differs from pure computer science instruction or programming class in that the student researcher and tools used by that researcher to leverage parallel facilities are considered essential. In other words, the direction of the class was driven largely by research needs of the student participants through externally derived projects. Due to the remarkable success of the inaugural offering of this course, we believe that the structure can be used as a model for future course development.

The course was designed to introduce and to instruct students from a wide variety of backgrounds in the fundamentals and (sometimes) art of high-performance programming. Because computing touches researchers from all disciplines, it was important to open the course to students from engineering as well as the school of arts and science, and the medical school. While teaching to a disparate group is often tedious, the diversity proves to be an important element. Ironically, the variety of students that participated in the class contributed significantly to the overall success of the course.

We should note that in the present context, high-performance computing refers to programming of parallel machines (usually massively parallel clusters) for the solution of scientific models. This definition implicitly excludes vector programming, high-performance storage and other forms of large-scale computer-based research not directly associated with computations. While these subjects are incredibly important for many areas of science and engineering, the utility to a varied audience is limited.

Course Structure

The published course description provides a general direction and broad topics that were covered in the classes.

> The fundamentals of high-performance computing will be introduced including design, development and deployment. Important language constructs such as garbage collection, parallelization (MPI/PVM), and computational libraries (LaPack/BLAS) will be discussed. Further, the effects of network architectures and nature of computational tasks on program design will be discussed. Emphasis will be placed on reusability and good programming practices.

In addition to these topical areas discussed in class, the thrust of the instruction revolved around student projects. Each participant was expected to devise a project related to their thesis research that could be augmented by applying high-performance computing facilities.

Because the course was intended as an introduction to high-performance computing, it was assumed that the students brought virtually zero knowledge to the class. While not universally valid, the expectation was justified. As a result, some relatively basic information was covered initially, followed by progressively more technical content. The semester was divided into three primary topical areas.

1. "How to" type of instruction to acclimate the participants to a cluster facility.
2. CPU speed and associated bottlenecks to fast computations.
3. Essential elements and tools used for parallelization of codes.

The first component is crucial to allow students access to the instructional HPC facilities. While progress in architectures of HPC facilities is migrating away from proprietary systems, rapid changes in technology and unique requirements mean that each system has particular modes of operation. Size, design and intended use also determine how people interface with the hardware. Therefore, a user must become acclimated to a particular system, and this first initiation into high-performance computing facilitates this learning curve. Fortunately, the primary components of different systems are usually not terribly different. The following non-inclusive list identifies several key questions a typical user might face.

- How does a user access the facility (log in)?
- Where does the user store user data?
- Where are the compilers and libraries and how are they invoked?
- What software tools are available and how are they accessed?
- How are jobs submitted?
- Are there any limitations imposed on use (disk space, CPU usage, etc.)?
- Are there any (what are the) rules of conduct?

Because of the flexibility and capabilities of the Linux operating system, which appears to be the operating system of choice for commodity-hardware parallel systems, most clusters are Unix based, and a general familiarity with Unix-like operating systems will ease this introductory material. This introduction also includes a pre-built parallel code that can be downloaded, compiled and run without modification. This step helps to solidify concepts presented in class. Finally, the user is asked to modify the pre-built code in directed ways, to highlight certain features of high-performance code. While the student is instructed on the operation and use of the cluster, he is expected to know a procedural programming language (preferably C or FORTRAN) as a prerequisite. The intent of the class is to instruct how high-performance computing is achieved and issues in obtaining significant speedup, not how to program in a particular language.

The second component of the class introduces the student to issues involved in getting optimal performance from computational hardware. Discussions revolve around the interplay between hardware and software. Optimal performance can often only be achieved through an intimate knowledge of hardware features in software. By definition, this precludes the use of platform-independent software (one reason Java is not a viable language in high-performance applications). However, this does not preclude the development of code that is specific to a single resource. Instead, an understanding of how performance can be enhanced by leveraging hardware features and avoiding hardware bottlenecks provides useful design constraints for code development. For example, using BLAS (Basic Linear Algebra Subroutines) libraries, which are tuned to specific hardware, demonstrates the incredible power of balancing memory reads and writes with data organization in memory and utilization of on-board cache. While detailed design of the BLAS library code is well beyond the scope of the class, students receive an important appreciation for hardware configurations and utilizing external libraries.

In addition to the discussion of direct hardware and software interactions, differing parallel architectures were presented including concepts such as shared-memory, clusters, MOMS classification of hardware and a brief history of the development of high-performance computing. Further, the importance of understanding the problem at hand was emphasized. For example, many problems can be classified as task parallel or data parallel. These are usually solved using completely different approaches. Therefore, patterns in parallelized problems and programs helps reduce code development time.

The third and final component of the class introduces many tools that have been found useful in preparing parallel codes. While MPI (Message Passing Interface) is perhaps the most common and most general approach to parallelization, there are many tools that alleviate the student from learning the subtle nuances of exchanging data between processors. Recall that the focus of the course is to instruct student researchers in how to solve problems using high-performance facilities. Because of the strong researcher directive, the computer science is often downplayed in preference to enabling tools. Of course, the basics of MPI constructs are also discussed because many problems do not fit well within existing software solutions. Examples of the tools that were explored during the class include the following.

Dakota An iterator toolkit designed and developed at Sandia labs for parallelization of

design space studies.

Scalapack Parallelized subset of the popular LaPack libraries used for linear algebra operations.

Global Array Libraries developed at Pacific Northwest Labs to perform large-scale matrix and vector operations across multiple processors.

PETSc Suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.

The course is unique among high-performance computing classes because of the three component structure and the research-focused goal. Listed below is a far from complete sample of class offerings at universities around the country. However, it is believed that these are typical of the types of programs found.

Penn State A one hour class is offered for students and faculty to teach them how to use the cluster and related tools. The lectures provide an overview of pre- and post-processing tools, available compilers and libraries and system particulars by "watching experts."

UCSD Aside from standard computer-science oriented classes The University of California at San Diego also offers seminar series to discuss trends in HPC. While interesting and possibly useful, the courses are not directed toward researchers.

Colorado State A class on the high-level manipulation of data for advanced algorithms is offered. This course is intended for elite computer science students.

Mississippi State A series of computer science oriented classes that cover fundamentals such as scheduling and load balancing is offered.

In general we have found that courses are either directed toward computer science students or only present the most basic "how-to" type of instruction. The success of the described course is primarily attributed to the emphasis placed on researchers.

Project Overviews

Because the course is designed around the implementation of ongoing research projects, the content is not limited to topics of interest to computer scientists only. Therefore, the course contains an inherent multidisciplinary element, and participants were exposed to a variety of technologies. To demonstrate the breadth of projects pursued during the class, a one sentence description of each is provided along with an estimated speedup that was obtained as a direct result of the course.

- *Finite element model for intra-operative updating of the liver* – High-performance computing coupled to diagnostic equipment gives surgeons real-time information on non-visible organ conformation.
  - Research Group: Biomedical Engineering

  - Estimated Speedup: 8

- *Elastography using a mutual information metric for the detection of breast cancer* – New methods for reliable detection of tumors are leveraging high-performance computing for quick diagnostics.

  - Research Group: Biomedical Engineering
  - Estimated Speedup: 8

- *Destructive particle interaction in semiconductors* – Using the GEANT4 high-energy particle physics libraries to simulate particle strikes on multi-layer planar targets representative of semiconductor devices, researchers are predicting failure in devices due to extraterrestrial radiation.

  - Research Group: Electrical Engineering and Computer Science
  - Estimated Speedup: 32

- *Improving web server reliability during flash crowds* – The handling of sudden and extreme volumes of web traffic is being modeled and studied using advanced computing facilities.

  - Research Group: Electrical Engineering and Computer Science
  - Estimated Speedup: 8+

- *Parallel Preprocessing of functional brain images based on statistical parametric mapping* – Using high-performance computing, doctors are able to receive MRI data analysis in minutes instead of days.

  - Research Group: Institute of Imaging Science
  - Estimated Speedup: 32

- *Ab Initio calculation of properties of neutron rich nuclei far from the valley of stability* – Modeling of elementary interactions with high-performance computing enables scientists to study the origins of the universe.

  - Research Group: High-energy and Nuclear Physics
  - Estimated Speedup: 32

- *Use of symbolic discriminant analysis to identify genes associated with the onset and maturity of rheumatoid arthritis based on their expression levels as measured by microarray experiments* – The widespread use of data collected from the human genome project relies heavily on the utilization of high-performance computing facilities.

  - Research Group: Program in Human Genetics
  - Estimated Speedup: 100+

- *Simulation of single event burnout of diodes* – High performance computing is facilitating the study of microelectronic failure in harsh environments.

  - Research Group: Mechanical Engineering

  - Estimated Speedup: 8

- *The effects of chemical boundary defects in microelectronic performance* – Continued scaling of electronic devices will require detailed understanding of new small-scale effects that can only be modeled using high-performance computing.

  - Research Group: Mechanical Engineering
  - Estimated Speedup: 8

Conclusions

The course offering enabled student researchers to achieve significant levels of computational speedup in the thesis projects. This newfound ability has allowed more complex models to be considered and science to be advanced. The key to the overwhelming success of the course is directly related to the strong emphasis placed on the student projects and to the adaptation of the topical material. We feel this course has helped bridge a gap between the capability of high-performance computing facilities and the ability of researchers to utilize the resources.

References

[1] R. Buyya, editor. *High Performance Cluster Computing: Architectures and Systems*, volume 1. Prentice Hall, 1999.

[2] D. Grigoras. Programming models for cluster computing. *Lecture Notes in Computer Science*, 2326:26–35, 2002.

Biography

GREG WALKER is an Assistant Professor in Mechanical Engineering at Vanderbilt University. His computationally intensive research and lack of training in graduate students has lead him to create the course discussed in the manuscript.

ALAN TACKETT is a Research Assistant Professor in the Physics and Astronomy Department at Vanderbilt University. He is also responsible for the design and development of the high-performance computing resources used during the course.