FutureGrid Education: Using Case Studies to Develop A Curriculum for Communicating Parallel and Distributed Computing Concepts

Jerome Mitchell
Indiana University
Bloomington, Indiana
jeromitc@indiana.edu

Judy Qiu
Indiana University
Bloomington, Indiana
judy.qiu09@gmail.com

Massimo Canonico
University of Piemonte Orientale
Vercelli, Italy
mex@di.unipmn.it

Shantenu Jha
Lousiana State University
Baton Rouge, LA
sjha@cct.lsu.edu

Linda Hadyen
Elizabeth City State University
Elizabeth City, North Carolina
haydenl@mindspring.org

Barbara Ann O'Leary
Indiana University
Bloomington, Indiana
baoleary@indiana.edu

Renato Figueiredo
University of Florida
Gainesville, Florida
renato@acis.ufl.edu

Geoffrey Fox
Indiana University
Bloomington,Indiana
gcf@indiana.edu

ABSTRACT

The shift to parallel computing—including multi-core computer architectures, cloud distributed computing, and general-purpose GPU programming—leads to fundamental changes in the design of software and systems. As a result, learning parallel computing techniques in order to allow software to take advantage of the shift toward parallelism is of important significance. To this end, FutureGrid, an experimental testbed for cloud, grids, and high performance computing, provides a resource for anyone to find, share, and discuss modular teaching materials and computational platform supports.

This paper presents a series of case studies for experiences in parallel and distributed education using the FutureGrid testbed. Building on previous experiences from courses, workshops, and summer schools associated with FutureGrid, we present a viable solution to developing a curriculum by leveraging collaboration with organizations. Our approach to developing a successful guide stems from the idea of anyone interested in learning parallel and distributing computing can do so with minimum assistance from a domain expert, and it addresses the educational goals and objectives to help meet many challenges, which lie ahead in the discipline.

We validate our approach to developing a community driven curriculum by providing use cases and their experiences with the teaching modules. Examples of some use cases include the following: hosting a workshop for faulty members of historically black colleges and universities, courses in distributed and cloud computing at universities, such as Indiana University, Lousiana State University, and the University of Piemonte Orientale.

1. INTRODUCTION

In order to prepare students for the manycore era, computer science educators must urgently increase the prominence of parallel computational concepts and programming in their curriculum. The concurrency evolution is not the only reason to teach parallelism to interested students. Current avenues to teach parallel and distributed computing focus on smaller- sized problems and datasets, which can be processed on a student's personal computer, making them ill-prepared to cope with the vast quantities of data in operational environments. Even when larger datasets are leveraged in a control medium, they are mostly used as static resources. Thus, students experience a disconnect as they transition from a learning environment to where they work on real-world problems. Some of the most exciting emerging applications of computing involve data-intensive scalable computing, which allow computational frameworks, such as, Google's proprietary, MapReduce and the open-source Apache Hadoop to be applied to datasets of web scale.

Because computer scientists and software developers can no longer take advantage of Moore's dividend, where software developers rely on increasingly faster CPUs for faster software - has expired, educators must become pioneers, learn from their efforts, and work together to infuse a curriculum with parallelism. Some computer science educators have started by suggesting ways to introduce parallel concepts and problems into an educational context.

As a step towards enabling interest in parallel and distributed

computing, we have produced flexible, customizable teaching modules to aid multiple courses and institutional learning with parallel platforms to support hands-on assignments.

Our goals for the community of educators include feedback and supplementation of the initial modules, contribution of new modules and platform packages from the community, and mentoring and support for new users of these materials. The combination of modular materials and a supportive community of peers is designed to make it practical for courses, workshops, and summer schools to introduce parallel computing.

In this paper, we discuss our rationale, describe educational modules, and present case studies of a supportive community of educators.

2. FUTUREGRID EDUCATION

FutureGrid provides unique capabilities by enabling researchers to deploy customized environments for their experiments in grid and cloud computing, and it allows educators and students to use virtual private clusters for hands-on education and training activities. A core technology supporting FutureGrid's training and education activities is a virtual Grid appliance [4] [6]: a system which leverages virtualization, cloud computing and self-configuring capabilities to create self-contained, flexible, plug-and-play executable modules; this capability isolates virtual clusters, so individual students, student groups, or classes can acquire resources on demand. Within the virtual cluster, an entire distributed computing middleware stack is self-configured, allowing activities to focus on the application of distributed/cloud computing platforms rather than the configuration of middleware. The provisioning time of an isolated, educational virtual cluster is of the order of a few minutes. It requires no configuration from instructors or students other than joining a Web 2.0 site to create and manage a group for their class.

In terms of design and implementation, the educational virtual cluster on-demand builds upon Grid virtual appliances, which encapsulate pre-configured software environments. Appliances are deployed on FutureGrid using either Nimbus or Eucalyptus. Once appliances are deployed, a group virtual private network (GroupVPN [5]) is self-configured to provide a seamless IP-layer connectivity among members of a group and support unmodified middleware and applications. Within deployed appliances, Condor is used as the core underlying scheduler to dispatch tasks. In our system, these tasks could be jobs students schedule directly with Condor, as well as tasks, which are used to bootstrap MPI , Hadoop, or Twister pools on demand. The GroupVPN virtual network and the Condor middleware are both self-configured by using a peer-to-peer Distributed Hash Table (DHT) as an information system to publish and query information about available job scheduler(s) and assign virtual IP addresses. Users create on-demand MPI, Hadoop, and Twister virtual clusters by submitting Condor jobs, which are "wrappers"for dispatching and configuring the respective run-time systemsc. MPI tasks are submitted together with the job that creates an MPI ring, while Hadoop and Twister tasks are submitted to the virtual cluster using standard tools, respectively. The entire system can be seamlessly deployed on a managed IaaS infrastructure, but it is also easily installable on end-user resources the same virtual appliance image is used in both environments. On a managed cloud infrastructure, IaaS middleware is used to deploy appliances, while in desktop/user environments, appliances are deployed with the native user interface of a virtual machine monitor. The appliance has been tested on widely-used open-source and commercial desktop and server virtualization technologies; the same image can be instantiated on VMware, KVM, and VirtualBox on x86-based Windows, MacOS and Linux systems. Academic institutions with restrictive computational resources can use virtual appliances providing students the opportunity to easily experiment with cloud technology.

3. THE NEED FOR COMMUNITY MODULES

To handle the rapid changes in computer architecture, we will benefit from community-based modules offering additional services and features to support efforts in anyone willing to learn parallel computing. We envision support resources for providing open and inviting environments, in which prior parallel computing knowledge or experience is not a prerequisite.

4. MODULES AND PLATFORM PACKAGES

4.1 Modular Approach to Parallelism

The FutureGrid modular curriculum have the following characteristics: seamless with respect to the "host "course, and it reinforces rather than replace existing material. Our modules provide examples and exercises to present a context for standard topics covered in a course, workshop or summer school. By focusing on examples, we increase the student's awareness and sophistication with respect to this area

4.2 Platform Packages

Every organization dedicated to advancing computer science education has its own interests and is usually implemented in the context of the organization's faculty with their individual area of expertise. These attributes led us to create customizable teaching materials and "hands-on"experiential learning exercises. If such teaching modules incorporate different programming models and presume minimal prerequisite assumptions, they will have wide compatibility with a student 's knowledge and flexibility for use in multiple educational contexts. Platform-specific supports, such as the "virtual clusters"combined with community support tutorials, will expedite and simplify the use of parallel computation resources students will need in order to learn about parallelism by experiencing it first-hand. The following are modules supported by our virtual clusters and available to the computing community:

4.2.1 Twister

Twister [1] is an extended MapReduce implementation for supporting iterative computing. In its implementation, Twister reads data from local disks of worker nodes and handles the intermediate data in the distributed memory of those worker nodes. All communication and data transfers are performed by a pub/sub broker network, which is a distributed messaging infrastructure. As in other MapReduce runtimes, a master (MRDriver) controls other workers according to instructions given by the user program. Twister uses a pub/sub broker network to handle four types of communication needs: send and receive control events, send data

from the client-side MapReduce driver to the Twister daemons, transfer intermediate data between map and reduce tasks, and send the outputs of the reduce tasks to the client side driver.

4.2.2 Hadoop

Hadoop [3] is an open source implementation of the MapReduce programming model. A MapReduce job usually consists of three phases: map, copy, and reduce. In the map phase, a user defined function operates on every chunk of input data producing intermediate key-value pairs, which are stored on local disk. A map process is invoked to process one chunk of input data while in the copy phase, the intermediate key-value pairs are transferred to the location where a reduce process would operate on the intermediate data. In reduce phase, a user defined reduce function operates on the intermediate key-value pairs and generates the output. One reduce process is invoked to process a range of keys.

4.2.3 MPI

MPI [2] is a message passing interface used for parallel processing in distributed memory systems. MPI is a library of routines that can be called from C, C++, FORTRAN77 and FORTRAN90 programs. A single user program is prepared, but is run on multiple processes. Each instance of the program is assigned a unique process identifier, so that it is labelled and known which process it is. This allows the same program to be executed, but different tasks are performed in each process. By convention, the user sets up one process as a master, and the others as workers, but this is not necessary. Each process has its own set of data, and can communicate directly with other processes by passing data around. Because the data is distributed, it is likely that a computation on one process will require that a data value be copied from another process.

5. CASE STUDIES

5.1 A Cloudy View on Computing workshop

A hands-on workshop for faculty from historically black colleges and universities (HBCUs) was conducted on June 6-10, 2011 at Elizabeth City State University. Participants were immersed in a MapReduce boot camp, where faculty members sought introduction to the MapReduce programming framework. An overview of parallel and distributed processing provided a transition into the abstractions of functional programming, which introduces the context of MapReduce along with its distributed file system. Lectures focused on specific case studies of MapReduce, such as graph analysis and information retrieval. The workshop concluded with a programming exercise (PageRank or All-Pairs problem) to ensure faculty members have a substantial knowledge of MapReduce concepts and the Twister/Hadoop API. The following were themes for five boot camp sessions:

- Module 1 Introduction to parallel and distributed processing
- Module 2 From functional programming to MapReduce and the Google File System (GFS)
- Module 3 “Hello World”MapReduce Lab
- Module 4 Graph Algorithms with MapReduce
- Module 5 Information Retrieval with MapReduce

FutureGrid was leveraged to create a self-contained, flexible, plug-and-play educational ”virtual appliance”. The virtual appliances support multiple virtualization technologies allowing them to run on a variety of resources, including user workstations and desktop grids. The “Cloudy View on Computing”workshop highlighted two specific educational virtual appliances detailing different middleware stacks used actively in clouds: Hadoop and Twister, each with varying data-intensive applications.

Figure 1: Cloudy View on Computing participants and presenter

5.2 Courses in Distributed Systems and Cloud Computing

5.2.1 Indiana University

CSCI-B534 (Distributed Systems) is offered to PhD and Masters students who are interested in the evolutional changes in computing landscape characterized by parallel, distributed, and cloud computing systems. The following modules were intended to provide an understanding of the technical issues involved in the design of modern distributed systems. Besides conveying the central principles involved in designing distributed systems, these modules also aims to present some of the major current paradigms.

- Module 1: Computer clusters for Scalable Parallel Computing
- Module 2: Introduction to Distributed Systems, Architectures, and Communication
- Module 3: Processes, Performance Issues, and Synchronization
- Module 4: Naming
- Module 5: Data Centers, Clouds Platform and Infrastructure, Energy Efficiency, and Virtualization Technologies and tools
- Module 6: MapReduce and data parallel applications; Hadoop, Dryad/DryadLINQ, and Twister

- Module 7: Security and Distributed File Systems

FutureGrid was used to build a prototype system and acquire an in-depth study of essential issues in practice, such as scalability, performance, availability, security, energy-efficiency, and workload balancing. Students took advantage of FutureGrid's dynamic provisioning infrastructure to switch between bare metal and virtual machine environments and enabled a message broker monitored the CPU and memory usage of a MPI PageRank application the dynamic cluster.

The course, Cloud Computing for Data Intensive Sciences, offers graduate students cloud computing programming models and tools to support data-intensive science applications. These include virtual machine-based utility computing environments, such as Amazon AWS and Microsoft Azure. The following modules supported this course:

- Module 1: Data Intensive Sciences, Data Center Model, Current Clouds with Infrastructure, and Platforms and Software as a Service
- Module 2: Parallel Programming/MPI vs. Mapreduce/Hadoop
- Module 3: Virtualization Technologies and Tools
- Module 4: MapReduce and Data Parallel Applications, Building All-to-All Blast Using Hadoop
- Module 5: Iterative Mapreduce and EM Algorithms
- Module 6: Mapreduce on Multicore/GPU
- Module 7: Cloud Storage

Project topics included: Matrix Multiplication with DryadLINQ; Implementing PhyloD application with DryadLINQ; Parallelism for Latent Dirichlet allocation (LDA); Memcached Integration with Twister; Improving Twister Messaging System Using Apache Avro; Large Scale PageRank with DryadLINQ; A Survey of Open-Source Cloud Infrastructure; A Survey on Cloud Storage Systems; Performance Analysis of HPC Virtualization Technologies.

The projects were performed on FutureGrid and carefully chosen to include different aspects of the cloud architecture stack from top-level biology and large-scale graphics applications, optimization of MapReduce runtimes, cloud storage, to low-level virtualization technologies.

5.2.2 *University of Piemonte Orientale*

A course in cloud computing was offered to Master's and Ph.D students interested in learning and experiencing with the most important cloud solutions (Eucalyptus, Nimbus, and OpeNebula).

- Module 1: Introduction to Cloud Computing
- Module 2: Introduction to Eucalyptus, Nimbus and OpenNebula

Figure 2: the Cloud Computing for Data Intensitve Sciences course at Indiana University

- Module 3: Eucalyptus: Image Management; Monitoring and Cloning, HybridFox
- Module 4: Nimbus LAMP + CMS installation
- Module 5: One-click Cluster

FutureGrid allowed students to implement a service, such as a web server, and to monitor its response time. A scheduler, considering the response time, decided where/when to switch on/off virtual machines.

5.3 Scientific Computing

5.3.1 *Louisiana State University*

A graduate course covering the basics of practical scientific computing for mathematicians, computer scientists, physical scientists, and finance. Topics ranged from basic mathematical principles and algorithms of numerical analysis to practical issues in software reliability to performance on modern computing hardware. Modules supported for this course is outlined in the following:

- Module 1: Preliminaries
- Module 2: Introduction to Numerical Methods
- Module 3: Vector Algebra, Basic Visualization Programming
- Module 4: Advanced Secure Shell Usage
- Module 5: Best Coding Practices
- Software Development, Revision Control
- Compiling, Debugging, Profiling

Through this course, students used FutureGrid and were required to face a number of small issues which complicated computational research, such problems with compilation and deployment etc., and it further provided students with the confidence to work through these problems and to be aware of the options for assistance through documentation and help desk facilities.

6. CONCLUSION

We have adopted an incremental, modular approach to introducing parallel computing concepts and experiential exercises as a strategy for addressing the urgent need for anyone interested in learning parallelism. Modules and support materials were designed for flexible use in diverse course and institutional settings, and for ease of adoption. We sought the creation of a supportive community of educators who share and encourage efforts to teach more parallelism through this modular approach, including instructors without particular expertise in parallelism.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.H. Bae, J. Qiu, and G. Fox. Twister: a runtime for iterative mapreduce. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 810–818. ACM, 2010.

[2] P.S. Pacheco. *Parallel programming with MPI*. Morgan Kaufmann, 1997.

[3] T. White. *Hadoop: The definitive guide*. Yahoo Press, 2010.

[4] D.I. Wolinsky, P. Chuchaisri, K. Lee, and R. Figueiredo. Experiences with self-organizing, decentralized grids using the grid appliance. *Cluster Computing*, pages 1–19, 2011.

[5] D.I. Wolinsky, K. Lee, P.O. Boykin, and R. Figueiredo. On the design of autonomic, decentralized vpns. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*, pages 1–10. IEEE, 2010.

[6] D.I. Wolinsky, A. Prakash, and R. Figueiredo. Grid applianceÑon the design of self-organizing, decentralized grids. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 563–567. IEEE, 2010.