PDC Education in the BGU ECE Department

Guy Tel-Zur
Ben-Gurion University of the Negev and NRCN
Beer-Sheva, Israel
gtelzur@post.bgu.ac.il

Abstract—This paper describes two undergraduate Parallel and Distributed Computing (PDC) courses which are taught in the Department of Electrical and Computer Engineering (ECE) at Ben-Gurion University (BGU). The first course, An Introduction to Parallel Processing is mandatory and the follow-up course, Parallel and Distributed Computing is an elective. Great efforts have been invested in making these courses attractive and up to date. Significant resources and budget constraints make this goal challenging. It is also a dilemma what material to choose, at what depth, what to ignore and also what should be the right balance between theoretical algorithms and practical tools. There is no single answer to these questions. This paper describes the author’s way of coping with these questions. There are several unique features in the two courses such as the adoption of Virtual Machines, the preference of Open Source Software and the use of on-line material. All these make the PDC courses very accessible to the students. Two unique home assignments using HTCondor, a High-Throughput Computing tool, are described. Finally, an extensive list of references is provided at the end of this paper which may be useful for other educators in similar courses.

Keywords: Parallel Computing, Distributed Computing, Education, Teaching, Grid Computing, Cloud Computing, HTCondor, HTC, HPC.

I. INTRODUCTION

Parallel Computing has expanded in the last 20 years from the niche of giant supercomputing machines, dubbed “Big Iron” to the main stream of programming on commodity hardware. Other significant changes of the last dozen years are the emergence of multi-core and then many-core systems and the emergence of Cloud Computing as the modern way of consuming computing services. These trends require continuous and adaptive changes in the education of our students in CS and ECE departments. Ten years ago there was a single elective course in the ECE department which was mainly focused on teaching Message Passing using the Parallel Virtual Machine (PVM) library. Today, this course is mandatory and many tools are taught (MPI, OpenMP, Hybrid programming, Cilkplus, Scalasca, HTCondor and more). In spring 2014 we opened a new continuation course called “Parallel and Distributed Computing” which deals with advanced MPI and OpenMP, PGAS, Xeon Phi programming, Grid and Cloud Computing. The home pages of the courses are provided in [1] and [2]. Our goal is to teach the introductory course during the autumn (first) semester each year and the advanced course in the following spring (second) semester.

There are a few key concepts to be emphasized:
1. We teach parallel computing and not supercomputing. This means that we use modest systems. We are interested in teaching effective parallel programming of systems with relatively small numbers of cores. We do not have access to PetaFLOP/s machines and it is not our goal to be in the high-end in these courses but rather at the low-end, and perhaps up to the (missing) middle systems.
2. All the paradigms of parallel computing can even be taught on a single core (serial) machine. MPI or OpenMP codes will run on those machines. This is true of the course until performance and scalability issues are involved. Therefore, students can implement their initial practice on modest computers and on Virtual Machines.
3. Modern syllabus. We want to have up-to-date courses and therefore the syllabi must continuously be updated.
4. Broad coverage of topics. We want to give the students a toolbox full of ideas to solve parallel computing challenges. At times this may be at the expense of delving deeper into the theory.
5. The courses are carried out with nearly a zero budget. Therefore: a) We try to minimize the system administration and maintenance by using virtualization and pre-installed images where possible, which also allows us to re-use the installed images course after course. b) We prefer Open Source Software (OSS). This also allows the students to install the same software on their own private computers.
6. The students do not have prior knowledge of Linux and during the first lab we devote time to teach Linux fundamentals, see next section.

II. THE INTRODUCTORY COURSE

A. General

The ECE undergraduate students are exposed to parallel computing for the first time in their 3rd year. Prior to the course they are familiar with serial programming in C and Matlab under the Windows operating system. They know nothing about Linux and they do not possess the concept of using the command line. For them, in order to build a code all that is needed is simply to press the “F5” key. Therefore we must devote time in the course to cover
topics which are not directly related to parallel computing but are needed for the rest of the course. The mandatory course is called *An Introduction to Parallel Processing*. The course has three pillars: Theory (parallel algorithms), Tools (to implement the algorithms, e.g. MPI, OpenMP) and Practice (labs, home assignments). Since a typical semester lasts 12-13 weeks, having three hours a week poses a challenge as far as creating a modern syllabus that on one hand covers a broad spectrum of methods and tools and on the other hand keeps the contents deep enough. During the course the students are required to hand in four home assignments (5% of the final grade for each assignment). In the last two sessions there is a Hackathon of students’ presentations (another 5% of the final grade). Usually there are 60-70 students in the class and they do the assignments in pairs. The overall grade is then determined by the home assignments, the presentations and a final examination with a weight of 75% of the total grade. The weekly course syllabus is described in Table 1.

<table>
<thead>
<tr>
<th>Meeting number (week)</th>
<th>Lecture Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to the course: A general lecture that describes what will be taught</td>
</tr>
<tr>
<td>2</td>
<td>Introduction to MPI</td>
</tr>
<tr>
<td>3</td>
<td>MPI and preparation for the hands-on lab. Algorithms: Embarrassingly Parallel Computations</td>
</tr>
<tr>
<td>4</td>
<td>Lab1: Linux fundamentals and hands-on MPI. Home assignment #1. The Scalasca performance analyzer [5]</td>
</tr>
<tr>
<td>5</td>
<td>Parallel Algorithms: Partitioning and Divide-and-Conquer Parallel Algorithms: Synchronous Computations</td>
</tr>
<tr>
<td>6</td>
<td>Parallel Algorithms: Numerical Linear Algebra (I) Home assignment #2</td>
</tr>
<tr>
<td>7</td>
<td>Parallel Programming in shared memory OpenMP Assignment of the topics for the final presentations</td>
</tr>
<tr>
<td>10</td>
<td>Introduction to HTC using HTCondor Lab 2: HTCondor tutorial [29] Home assignment #4</td>
</tr>
<tr>
<td>11</td>
<td>Students’ presentations 1</td>
</tr>
<tr>
<td>12</td>
<td>Students’ presentations 2</td>
</tr>
<tr>
<td></td>
<td>Final Examination</td>
</tr>
</tbody>
</table>

**TABLE I. INTRODUCTION TO PARALLEL PROCESSING SYLLABUS**

**B. Computing Resources**

The students do their labs and home assignments on a dedicated Linux cluster, called *Hobbit*, which is built from physical and virtual machines. The cluster is continuously monitored using Ganglia [3], see Figure 1.

The *Hobbit* educational cluster has 15 nodes and 120 cores in total. It has compilers, an MPI library, a performance analyzer, linear algebra packages and many other tools installed. The students can launch parallel programs using the Torque batch system but, so far, the usage of Torque is not enforced, therefore the students execute parallel MPI programs interactively using *mpirun* from the command line. We are, of course, aware that this degrades the performance and that time measurements, done with *MPI_Wtime()*, are not reliable. In any case we ask the students to repeat their code execution several times in order to average their timing measurements.

Our HTCondor pool is built from non-dedicated resources, mainly students’ lab machines located in several locations on campus. These machines are a mixture of several Windows versions and Linux flavors. There are about 300 cores in the BGU HTCondor pool. The Condorview portal monitors the pool’s activity as shown in Figure 2 [4].

**Figure 1.** The Ganglia monitor of the educational parallel cluster.

**Figure 2.** Condorview monitors the BGU HTCondor pool.

Due to the strict university security policy direct access from outside the campus to the Hobbit cluster is forbidden.

---

1 The figure reflects the HTCondor pool when it is not loaded with jobs.
(except for the Ganglia and Condorview web services). The only way to log in to the cluster from outside the campus is via another server using SSH (PuTTY [7] and Cyberduck [8] are our recommended clients for Windows). Since many ports are blocked by the fire-wall no connection is possible in graphics mode (e.g. TightVNC [9]). To compensate for this inconvenience we recommend that the students install Linux as a guest OS on a Virtual Machine (VM) using Virtualbox [10] or VMware Player [11] on their private laptops and develop their codes on those VMs. In particular, we recommend the use of the pre-installed Vi-HPS live DVD [12] which has all of the necessary software already well configured. Starting from the next semester the Vi-HPS distro will become a standard tool in the course and it will also be installed in the department computer lab. After the students are satisfied with the developed code on the VM they must transfer it, using a secure copy, to the cluster in order to execute it from there. The cluster, of course, gives higher performance and therefore performance measurements must be implemented there. The course text book is *Parallel Programming* by B. Wilkinson and M. Allen [13] and additional material has been prepared by the author. The official programming language of the course is standard C although C++ and Python are also allowed. During class there are demonstrations regarding how to compile and execute MPI and OpenMP on both Linux and Windows operating systems. The Windows demonstrations are based on the DeinoMPI distribution [14] and code development is done using the DevC++ IDE [15] (Code::Blocks [16] is also recommended). The Linux demonstrations are based on the MPICH package [17] and *vi* is recommended for editing (in the future we plan to move to Eclipse PTP [18]). OpenMP and Cilkplus [19] are also demonstrated on both operating systems (in Windows we use Visual Studio for that). By doing so we hope to convince the students that Parallel Computing can be used (compiled and executed) on all common operating systems.

C. Home Assignments

As mentioned above there are four home assignments during the course. The assignments that were given in autumn 2013-2014 are described in Table II and in [21].

In each assignment the students are asked to measure the execution time of the relevant blocks in the code for different numbers of tasks (or threads) and to derive conclusions based on the speed-up and parallel efficiency.

It is not easy to find ideas for home assignments that make use of HTCondor. We developed two such assignments which shall now to be discussed\(^2\). Let us first explain what is meant in home assignment #4 in Table II. The goal of the assignment is to compute the binding energy of nuclei using the semi-empirical mass formula [22] on a distributed High-Throughput computing environment, i.e. using HTCondor.

**TABLE II. AUTUMN 2013-2014 HOME ASSIGNMENTS**

<table>
<thead>
<tr>
<th>No.</th>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MPI</td>
<td>Simple image processing (an Embarrassingly Parallel Computation). The students are asked to take a photo of themselves and do parallel low level image processing on the photo.</td>
</tr>
<tr>
<td>2</td>
<td>MPI</td>
<td>Parallel Matrix-Matrix Multiplication</td>
</tr>
<tr>
<td>3</td>
<td>OpenMP</td>
<td>Time independent 2D Heat Equation in OpenMP</td>
</tr>
<tr>
<td>4</td>
<td>HTCondor</td>
<td>Computing the Binding Energy of nuclei using the Semi-Empirical Mass Formula on the BGU HTCondor pool</td>
</tr>
</tbody>
</table>

The students are asked to make a plot similar to the one in Figure 3 where 200 x 200 computations are needed, i.e. for each number of neutrons between 1 and 200 and for each number of protons, again between 1 and 200. The students must write an HTCondor *submit file* and to submit 200 jobs each containing 200 computations. The plot itself is generated by an external tool such as Matlab after all the HTCondor jobs are completed.

![Figure 3. HTCondor home assignment. Computation of the nuclei binding energy using HTCondor and the semi-empirical mass formula [22].](image)

Another example of an HTCondor home assignment, which was given last year, deals with HTCondor ability to handle DAGs (Directed Acyclic Graphs). In this home assignment the students are asked to compute the value of \(e^x\pi^x\) via Monte Carlo methods by implementing the DAG of Figure 4.

---

\(^2\) The two assignments will be uploaded to the CDER web site (in preparation).
HTCondor handles such and much more complex DAGs easily. The instructions of the assignments are as follows:

A1 Compute the natural logarithm, e, via a Monte Carlo method based on the Salt-Shaker algorithm [23] and write the approximated result into a file (submit 100 independent such jobs).

A2 Average all the results of step A1 into a single number (a single job).

B1 Compute π via a Monte Carlo method, see [24], and write the approximated result into another file (again submit 100 such jobs).

B2 Average all the results of B1 into a single number (a single job).

C Compute e⁻π² based on A2 and B2. Compare your result with the expected numerical value, −0.68153491441822, to verify the whole DAG computation.

D. Final Presentations

Usually there are 30 or more different students’ presentations at the end of the course that enrich the students’ knowledge beyond the topics covered in the regular lectures. In the middle of the course the students must select topics for their final presentations. They are encouraged to bring their own ideas or choose topics from a pool available on the course web site. Each presentation takes about 10 minutes. The topics presented in the 2013-2014 course are described in Table III. The presentations may be viewed and downloaded from the course web site [25].

III. THE CONTINUATION COURSE

A. General

In spring 2014 we opened for the first time a continuation course called Parallel and Distributed Computing.

There are a few unique features of this course which are worth mentioning:

1. We purchased two servers, each equipped with two Intel Xeon Phi 5110P [26]. The overall number of cores from both systems is 244 from the Xeon Phi and additional 24 cores from Xeon Ivy Bridge E5-2620 processors. Together the estimated theoretical peak performance is almost 4.5TFLOP/s. These servers are used for research, lab exercises and for final projects by the students.

2. The above machines are interconnected via two 100Gbps MCB194A-FCAT Connect-IB™ Host Channel Adapter cards by Mellanox Technologies [27].

---

3 E5-2620 peak performance is calculated from: 6 cores x 8 ops/clock x 2.1GHz which is ~100GFLOPs, The Xeon Phi 5110P has 1.01TFLOPs peak. We have four pieces of each in total.
3. In this course we also use Allinea’s DDT + MAP parallel debugger and profiler [20] with a license that allows tracing 8 processes (on the Xeon and/or on the Xeon Phi).

4. The HPC Advisory Council [28] allowed us to use two of its clusters for demonstrations. This gave us access to several Nvidia 2090 GPGPUs accelerators.

5. The Open Science Data Cloud (OSDC) [30] gave the author access to one of its cloud resources, Nebula, also for demonstrations. This system has the OpenStack cloud software and the Gluster file system.

6. Isragrid, the Israeli national grid initiative [31] which is part of the European Grid Infrastructure (EGI) prepared five additional nodes for the course which were installed with CentOS 6.5 for experiments with the Globus Toolkit grid middleware [32] which eventually were not carried out due to lack of preparation time.

7. In the courses we occasionally invite external distinguished guest lectures. This semester we had an introductory talk on cloud computing by Mr. Avner Algorn, CEO of the IGTCldCloud [33].

8. We selected the Amazon Web Services (AWS) as the main cloud computing platform for the course. During the third lab the students practiced AWS SQS (following an AWS tutorial which was given at the SC/ISC conferences [34]) and the construction of an HPC cluster on Amazon’s cloud using the MIT Star Cluster AWS image [35][36].

9. The OpenNebula [37] was selected as an additional (open) cloud platform. OpenStack [38] is also a fine option and it is left as a topic for one of the final projects. An easy way to demonstrate a private cloud based on OpenNebula can be performed in minutes using a VM with a pre-installed Virtualbox sandbox image [39].

The duties of the students in this course are: reading material prior to class, mostly available on-line, participation in class, handing in three home assignments and doing final projects, see next section.

The weekly course syllabus is described in Table IV.

<table>
<thead>
<tr>
<th>No.</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High-Performance Linpack (HPL) [45] on Xeon Phi</td>
</tr>
<tr>
<td>2</td>
<td>N-Body simulation on Xeon Phi [46]</td>
</tr>
<tr>
<td>3</td>
<td>Sweep3D [47] on Xeon Phi</td>
</tr>
<tr>
<td>4</td>
<td>FFTW [48] on Xeon Phi</td>
</tr>
<tr>
<td>5</td>
<td>Other optional project ideas for next PDC courses</td>
</tr>
<tr>
<td>6</td>
<td>Paas using Cloud Foundry [49] or Cloudify [50] or OpenShift [51]</td>
</tr>
<tr>
<td>7</td>
<td>Cloud inter-operability using HTCondor [52]</td>
</tr>
<tr>
<td>8</td>
<td>Data mining on public data bases using Hadoop or Elastic MR or Cloud9 [53]</td>
</tr>
<tr>
<td>9</td>
<td>Foring the NAS parallel benchmarks [54] to the Xeon Phi</td>
</tr>
<tr>
<td>10</td>
<td>Studies of the ConnectIB throughput and latency</td>
</tr>
<tr>
<td>11</td>
<td>OpenStack as a private cloud</td>
</tr>
<tr>
<td>12</td>
<td>A complex workflow of dependent tasks on a grid using the Kepler project [55]</td>
</tr>
<tr>
<td>13</td>
<td>Data mining using Apache Spark [56]</td>
</tr>
<tr>
<td>14</td>
<td>N-Body simulation on a cluster of GPGPUs</td>
</tr>
</tbody>
</table>

### TABLE IV. PARALLEL AND DISTRIBUTED PROCESSING SYLLABUS (SPRING 2014)

<table>
<thead>
<tr>
<th>Meeting number (week)</th>
<th>Lecture Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to the course. Advanced MPI (MPI-2, MPI-3)</td>
</tr>
<tr>
<td>2</td>
<td>Advanced MPI (MPI-2, MPI-3)</td>
</tr>
<tr>
<td>3</td>
<td>Advanced OpenMP</td>
</tr>
<tr>
<td>4</td>
<td>Lab 1: Introduction to Xeon Phi and Intel Cluster Studio 2013</td>
</tr>
<tr>
<td>5</td>
<td>PGAS and UPC [42], Allinea DDT</td>
</tr>
<tr>
<td>6</td>
<td>Introduction to Grid Computing</td>
</tr>
<tr>
<td>7</td>
<td>Lab 2: Grid Computing hands-on, UNICORE [43], Kepler [55]</td>
</tr>
<tr>
<td>8</td>
<td>Guest lecture: Introduction to Cloud Computing AWS, EC2 Google Compute Engine</td>
</tr>
<tr>
<td>9</td>
<td>Lab 3: AWS, Cloud Computing, Building HPC Cluster on the Cloud using StarCluster</td>
</tr>
<tr>
<td>10</td>
<td>OpenNebula and OpenStack</td>
</tr>
<tr>
<td>11</td>
<td>Big Data, Hadoop, Map-Reduce</td>
</tr>
<tr>
<td>12</td>
<td>Cloud Security (ISO27001, NIST guideline 800-144 [44]), Students’ presentations</td>
</tr>
</tbody>
</table>

The main text book for the course is *Distributed and Cloud Computing* by Hwang, Fox and Dongarra [40] and Barry Wilkinson’s *Grid Computing: Techniques and Applications* [41]. Additional on-line resources can be accessed from the course web site and from the university’s internal Moodle learning system. Another source of excellent material can be compiled from the Supercomputing conferences’ tutorials.

### B. Final Projects

The final projects include 15-minute presentations in front of the class which take place during the last lecture. The goal of the presentations is to give an overview of the final projects and at this stage the students still do not need to show results. The students then must develop code, do the necessary installations, execute the code, take measurements and derive conclusions. Finally they must write summary reports in a paper format. Table V describes the list of topics that were implemented during the spring 2014 course, where the emphasize was to work on the new Xeon Phi servers, and possible additional topics for future courses.

### SUMMARY AND RECOMMENDATIONS

In this paper we reviewed two PDC courses given in the BGU ECE department. The courses are oriented toward scientific computing. We are aware of the tension between teaching theoretical algorithms and teaching practical tools and the need for hands-on experience. The way we relaxed this tension is by opening the continuation course. This allows us to cover the same material but at a deeper level and to include additional new topics. The situation is somewhat similar to the way people fill a wardrobe. They add more and more clothing until they need to buy another wardrobe. However, the similarity stops when it comes to throwing away old clothing. In PDC there is far less material to give up than new material that should be added. It is the author’s belief that a third course in PDC is needed but this probably will not happen any time soon. To take this idea further, the author believes that there is a need for a full track toward a degree
in Computational Science and Engineering which is not yet offered by any Israeli academic institution. Continual improvement in the courses: The course never stagnates. Material is updated all the time and similar courses worldwide are checked in order to get inspiration for new ideas [57].

ACKNOWLEDGMENT

This work was partially funded by donations from Intel Corporation and from Mellanox Technologies. The author would like to acknowledge the following organizations, companies and people for their support of the PDC course:

Intel Corporation, Mellanox Technologies, the HPC Advisory Council, the Open Science Data Cloud and Israagrid.

Finally, special thanks to the ECE department computing manager Ms. Tami Chuchem and her team. Without the kind support and assistance of all these people and organizations the courses would not have become a reality.

REFERENCES

[31] Isragrid http://www.isragrid.org.il
[38] OpenStack, http://www.openstack.org/
[57] Introduction to Parallel Processing, References: http://tel-zur.net/teaching/bgu/pp/references2014A.html