Teaching Parallel Programming using Java

Aamir Shafi¹, Aleem Akhtar¹, Ansar Javed¹, and Bryan Carpenter²

Presenter: Faizan Zahid¹

¹ National University of Sciences & Technology (NUST), Pakistan
² University of Portsmouth, UK
Introduction

• This paper presents an overview of the “Applied Parallel Computing” course taught to final-year Software Engineering (SE) students at NUST, Pakistan in Spring 2014:
  – 2+1 hours elective course in the 8\textsuperscript{th} semester:
    • 5 (2+3) contact hours, 18 weeks/semester
  – The SE program spans 4 years, 8 semesters, and 136 credit hours

• A unique aspect of the course was that Java was used as the principle programming language
Course Contents

• After introducing parallel programming fundamental concepts, the course was divided into three sections:

1. Programming Shared Memory (SM) systems including SMPs and multicore processors

2. Programming Distributed Memory (DM) systems including clusters/network of computers

3. Advanced Topics:
   • GPU Computing
   • MapReduce computations
Motivation for using Java

• Portability—“Write once, run anywhere”

• A popular language in colleges and software industry:
  – Large pool of software developers

• Technical Features:
  – OO programming abstractions
  – Improved compile and runtime checking of the code
  – Automatic garbage collection
  – Support for multithreading
  – Rich collection of support libraries
Sample Parallel Application Codes

- Few *embarrassingly parallel* and *synchronous* applications were used throughout the course to practice shared/distributed memory parallel programming

Diagram:

- **Embarrassingly Parallel**
  - Tridiagonal Calculation
  - Mandelbrot Set
  - Matrix-Matrix Multiplication
- **Synchronous**
  - Conway’s Game of Life
  - Laplace Equation Solver
- **MapReduce**
  - WordCount
A typical teaching week consisted of:
- Two 50-mins lectures & One 3-hours lab

Weekly lectures covered:
- Intro of the serial sample app code (including Mandelbrot Set, Matrix Mult, Game of Life etc.)
- Thoughts on parallelizing the sample app on target systems (SM/DM)

Weekly labs:
- Invited students to develop a parallel version of the sample application on the target platform

For e.g. Week-3:

<table>
<thead>
<tr>
<th>W-#</th>
<th>Topic</th>
</tr>
</thead>
</table>
| W-3 | *Lect*: Embarrassingly Parallel Computations (Mandelbrot)  
      | *Lect*: Shared Memory Programming using threads  
      | *Lab*: Mandelbrot Set Calculation using threads |
Section 1: SM Programming

• Target systems included SMPs and multicore processors

• The course reviewed:
  – Java threads
  – Cilk and OpenMP

• Sample embarrassingly parallel and synchronous apps were discussed in the context of SM systems:
  – *Java threads* were utilized for development
  – SM parallelization issues including domain decomposition, load balancing, synchronization, race conditions, deadlocks etc. were discussed
Section 2: DM Programming

• Target systems included clusters and network of computers

• **MPJ Express** for DM app development:
  – It is an MPI-like library for Java programmers
  – Developed and maintained by authors
  – Implements the mpiJava 1.2 API defined by the Java Grande Forum
  – [http://mpjexpress.org](http://mpjexpress.org)

• Other Java MPI-like libraries:
  – Open MPI Java—Java bindings in the Open MPI library
MPJ Express ..

- **Cluster** mode
  - Supports executing MPJ apps on cluster/network of computers

- **Multicore** mode:
  - Supports executing MPJ apps on a shared memory systems
  - MPI processes are represented as Java threads inside a single OS process
  - *Very easy* to install, develop, and run MPJ apps

- MPJ app code remains the same for both execution modes
- Students initially developed and tested their MPJ apps using multicore mode
MPJ Express Runtime

- DM programming was conducted in a lab with PCs
  - No dedicated HPC platform was used
- A custom version of MPJ Express runtime was developed, which allowed execution without relying on the shared file system
Course Summary

• Offered for the first time as part of the SE curriculum:
  – 45/77 students opted for this elective course

• The students found the course practical and useful:
  – Several students got motivated to pursue PDC/HPC for graduate studies

• Practical experience:
  – Students found the first two sections (SM & DM programming) useful because of practical & extensive lab scripts
  – The third & final section on Advanced Topics (GPGPU & MapReduce w/Hadoop) had very basic lab component that needs to be improved in future offerings

• Feedback:
  – 1st: 86.9% & 2nd: 89.8% (considered excellent)
Future Offerings

• For the next offering (Spring ‘15), we plan to:
  – Include Java *Fork/Join* API as an option for programming SM systems
  – Introduce Big Data application codes including *PageRank, K-means*, etc.
  – Increase coverage of GPGPU topics:
    • Programming exercises using *JCUDA* ([http://www.jcuda.org/](http://www.jcuda.org/))

• We plan to keep the course *Java*-centric
Thank you. Questions?

• Please feel free to contact the main author:
  – Aamir Shafi:
    • W: http://hpc.seecs.nust.edu.pk/~aamir
    • E: aamir.shafi@seecs.nust.edu.pk