Experience of Applying NSF/IEEE TCPP Curriculum Initiative on Parallel and Distributed Computing at College of Staten Island

Yumei Huo and Feng Gu
Department of Computer Science, College of Staten Island, City University of New York
Yumei.Huo@csi.cuny.edu, and Feng.Gu@csi.cuny.edu

Abstract

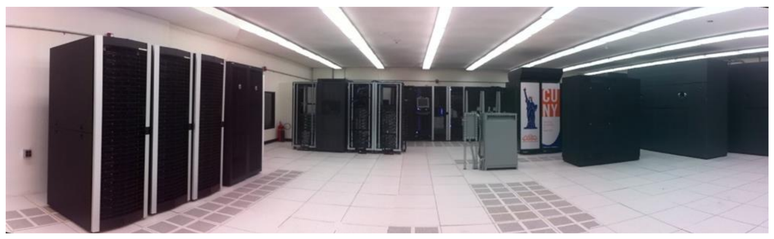
Department of Computer Science at College of Staten Island (CSI) has been offering three different levels of parallel and distributed computing (PDC) courses, mainly covering parallel programming and parallel algorithm analysis and design in addition to the basic concepts of parallel architectures. We applied early adopter award to improve the teaching of PDC by adapting the current core course of data structures and adding additional topics into existing two undergraduate PDC courses. In addition, a new course of shared memory programming has been proposed to enhance the PDC curriculum.

Background

- One of twenty-three institutions of The City University of New York (CUNY)
- Offers B.S. program, M.S. program, and Ph.D. program in computer science
- The CUNY High Performance Computing Center (http://www.csi.cuny.edu/cunyhpc) is housed on the campus of CSI, and financially supported by several NSF grants and from state and local funds

HPC Facilities

System	Type	Job Mix	Nodes	Cores /node	Memory /node	Memory /core	Chip type
ANDY	Cluster	64 or fewer cores	93	8	24 Gbytes	3 Gbytes	Nehalem (2.93 GHz)
APPEL	SMP		1	384	12 Tbytes	NA	Ivy Bridge (3.0 GHz)
BOB	Cluster	Gaussian	28	8	16 Gbytes	2 Gbytes	AMD Barcelona
KARLE	SMP	Interactive and batch for some applications	1	24	96 Gbytes	NA	Penryn (2.4 GHz)
PENZIAS	Cluster	128 or fewer cores	60	12	48 Gbytes	4 Gbytes	Sandy Bridge (2.2 GHz)
		4 cores/GPU	60	4 cores and 2 GPUs	16 Gbytes		NVIDIA K20 GPUs
SALK	Cluster	1024 or fewer cores	176	16	32 Gbytes	2 Gbytes	AMD Magny-cour (2.3 GHz)



Proposed Adopted Courses

CSC 229
Introduction to High Performance Computing

- 2 credit course: 1 lecture hour and 3 lab hours
- An elective course to early expose PDC topics to students

CSC 326
Data Structures

- Core course
- Introduces simply parallel/distributed models
- Parallel-sum and even-odd transposition sorting

CSC429
Advanced High Performance Computing

- 4 credit course: 3 lecture hours and 2 lab hours
- An elective course to focus on parallel algorithms design and analysis
- Both theoretical component and programming component
- Added BSP/CILK models
- Added the inter-influence of various cost measures
- Added cluster/cloud/grid/ computing
- Added peer-to-peer/collaborative/ pervasive/mobile computing

CSC 4XX
Shared Memory Parallel Computing

- 4 credit course: 3 lecture hour and 2 lab hours
- An elective course to focus on shared memory model
- Multithreading, OpenMP, and GPU programming

Activities and Results:

- ❖ Course modifications and development
- ❖ Faculty training: training seminars & workshops
  - training seminar of parallel programming and optimization with Intel Xeon Phi coprocessors by Intel
  - NVIDIA GPU Programming Workshop by NVIDIA
- ❖ Students participation
  - class projects
  - Join in faculty research and publish papers
  - Internships (CUNY HPCC) and NSF REU program at University of Texas at Austin

Evaluation & Dissemination

- ❖ Projects, labs and exams
- ❖ Student evaluation forms
- ❖ Posters and papers

Sample Student Projects:

Topic 1: Stock Market

Given the price of a stock at each day for n days, we want to determine the biggest profit we can make by buying one day and selling on a later day.

Input from the input file:
1) The number of days which are considered;
2) The price for each day.

Output: The biggest profit.

Write a parallel program to solve the stock market problem with work complexity $O(n)$ and depth $O(\log n)$.

Topic 2: Evaluate Polynomials

Input from the input file:
1) The degree of the polynomial (the highest degree of all the terms in the polynomial);
2) The coefficient for each term from the highest degree to the lowest degree;
3) The value of x.

Output: Evaluate the polynomial and print out the value of the polynomial.

Topic 3: Matrix-Vector Multiplication

Part One: Use rowwise 1-D Partitioning or columnwise 1-D Partitioning to implement Matric-Vector Multiplication. (Refer to textbook Chapter 8.1.1)

Part Two: You will use 2-D Partitioning to implement Matric-Vector Multiplication. (Refer to textbook Chapter 8.1.2)

Input from the input file:
1) n which is the total number of integers which will be operated;
2) matrix A (n2 integers) and vector x (n integers).

Output: vector y (n integers).

Compare Part I and Part II, what number is n ranged in when the running time of Part I is better than Part II?