# Incorporating PDC Modules into Computer Science Courses at Jackson State University

Ali Abu El Humos, Sungbum Hong, Jacqueline Jackson, Xuejun Liang, Tzusheng Pei and Bernard Aldrich
Department of Computer Science
Jackson State University
Jackson, MS 39217, USA

# Agenda

- Background
- Curriculum Changes
- Early Adopting Courses
- Students Feedback
- PDC Modules Inclusion
- Assessment Tools and Results
- Conclusions and Future Work

# Background

- Jackson State University (JSU) has a student population that is over 90% from under-represented groups
- About 25% of our students are females
- This NSF/IEEE-TCPP Curriculum Initiative award will therefore have a direct impact on minorities specifically in the Computer Science field

# Curriculum Changes

- The computer science department at Jackson State University is updating its curriculum according to the new ABET guidelines.
- The computer science new curriculum at JSU is a 125 credit-hour program of which 57 in computer science.
- Some advanced computer architecture topics are moved to elective courses. This may prevent students from learning about important PDC topics such as pipelining, superscalar architectures, Multiprocessors and Multi-core processors, and GPU programming.

# Early Adopting Courses

1) CSC 119 Object Oriented Programming (core)
2) CSC 216 Computer Architecture and Organization (core)
3) CSC 312 Advanced Computer Architecture (core)
4) CSC 325 Operating Systems (core)
5) CSC 350 Organization of Programming Languages (core)
6) CSC 425 Parallel Computing (elective)

# CSC 119 Object Oriented Programming in Java

- Covers inheritance, polymorphism, interfaces, exception handling, streams and file input/output, recursion, dynamic data structures (linked lists, stacks, queues, hash tables, graphs, trees) and associated algorithms
- For supporting parallel computing, Java has the Thread class and the Runnable interface, and it also provides rich primitives with the java.util.concurrent packages, which include the fork/join framework. Students explored these features in Java for parallel computing.

# CSC 216 Computer Architecture and Organization

- Covers the basic concepts of computer architecture which includes :
  - machine level representations of data
  - computer arithmetic
  - instruction set architecture and assembly language,
  - datapath and control
  - memory system
  - and bus architectures and I/O devices
- A new PDC module was added to this course to introduce students to multi core processors and GPU hardware. Also, throughout the course, parallelism at different levels were discussed.

# CSC 312 Advanced Computer Architecture

- This course may become an elective under the new curriculum.
- Covers various advanced topics of PDC curriculum such as:
  - instruction level parallelism: pipelining and superscalar architectures
  - processor level parallelism: array processors, multi-processor and multi-computer systems.
  - Techniques to reduce instruction pipeline stalls,
  - Direct map and set associative caches are analyzed
  - Quantitative approaches of computer performance are emphasized.
- A new PDC module was added to this course to introduce students to bench marks and how they can be used to differentiate between various parallel systems performance.

# CSC 325 Operating Systems

- This course introduces the major concepts of process communication and synchronization, protection, performance measurement, and causes and evaluations of the problems associated with mutual exclusions and process synchronization among concurrent processes. It also, introduces and analyzes various operating systems in terms of processor management, memory management, device management, information management, and distributed systems management.
- A PDC module was added to this course to extend process synchronization issues to parallel programming concepts. With this module, the course will provide students with parallel thread programming opportunities.

# CSC 350 Organization of Programming Languages

- Covers several issues in language design, including typing regimens, data structure models, control structure models, abstraction, virtual machines, language translation, interpreters, compiler design, lexical analysis, parsing, symbol tables, declaration and storage management, code generation; and optimization techniques.
- In this course, after a brief review Java features for supporting parallel computing (taught in CSC 119), parallel programming assignments were given for gaining hands-on experience. Generic concepts in parallel computing were also introduced.

# CSC 425 Parallel Computing

- This is a newly developed elective.
- Only one student registered for the Parallel Computing class offered last fall semester, which resulted in cancelling the class.
- A study of hardware and software issues in parallel computing. Theoretical and practical survey of parallel processing, including a discussion of parallel architectures, parallel programming languages, and parallel algorithms. Programming on multiple parallel platforms in a higher-level parallel language.
- In this course, students will learn how to write parallel programs on three different parallel architectures: i) shared memory model- thread programming; ii) Cluster- Message passing Computing; and iii) Multicore- GPU Programming.

# Students Feedback

- A Survey of six questions was distributed among ACM computer science students. Results reported as follows with Min=1 and Max=4:

| Question | Average Score |
|---|---|
| Please, rate your current knowledge of PDC. | 1.4 |
| Please, rate the breadth of PDC topics covered in the computer science curriculum at JSU | 1.4 |
| Please, rate the depth of PDC topics covered in the computer science curriculum at JSU. | 1.5 |
| Please, rate your overall learning experience of PDC at the computer science department at JSU. | 1.5 |
| Will you be interested in pursuing a career that requires PDC knowledge? | 2.5 |
| Will you be interested in registering for Advanced PDC classes if offered during fall or spring semesters? | 3.0 |

# Students Comments

- “I do not have knowledge of PDC. It would be helpful if we learned about this topic in our classes. This could help with strengthening our programming skills. Activities or projects during class would be helpful”.
- “I hope that there would be mentoring sessions with faculty to help students address the issue of being able to learn on our own different languages and principles of Computer Science”.
- “One could provide real world Applications of PDC. If it is implemented in classes, make sure that you provide real world application as well as theory. The combination of both solidifies its importance and builds interest”.
- “I think it is very relevant, but at the same time, I do not know anything about it. I just know that our curriculum is already filled with classes, and it is tough trying to manage those classes”.
- “The topic should be touched in classes at every level. For a topic like this, the students should start learning about it as freshmen so that they can increase understanding of it over time”.
- “PDC careers should be discussed more and more. More hands on activities”.
- “We really did not talk about PDC in my undergraduate program. I would love to see it offered for students”.

# PDC Modules Inclusion

1) CSC 119 Object Oriented Programming: PDC topics like JAVA threads were emphasized in class. Problems were developed for students to work on outside classroom.
2) CSC 216 Computer Architecture and Organization: This course was taught twice a week. Since the class size was small (only 6 students), the instructor agreed with the students to add 5 minutes to each lecture. That provided the instructor with about 3 extra hours to cover PDC topics like multi core processors and GPU hardware.

# PDC Modules Inclusion (continue)

3) CSC 312 Advanced Computer Architecture : The time to cover pipeline hazards was reduced to accommodate the inclusion of benchmarks (around 1.5 hours). Students were asked to submit 2 research papers:
   - A comprehensive survey of various benchmarks used to report performance of Desktop, Server and Embedded Systems.
   - Multi Core Processors issues such as caching, pipelining, energy consumption, Operating Systems, and performance.
4) CSC 325 Operating Systems: In order to create space to cover PDC topics of multi core programming and multi threading model, the following actions were taken:
   - Skip some subtopics not dependent on the important topics that ACM/IEEE 2013 suggested. Students can understand these subtopics by just reading them.
   - Problem sets were developed for students to study in groups outside classroom.

# PDC Modules Inclusion (continue)

5) CSC 350 Organization of Programming Languages: The time to cover the course topics is already limited. Students were given extra curriculum parallel programming assignment.

# Assessment Tools:
## CSC 119 Object Oriented Programming

- Final Exam Questions Set:
  - How do you specify concurrent activities in JAVA?
  - What activities benefit from concurrent programming?
  - Describe the life cycle of a thread.
  - Write a program that demonstrates the use of a thread.
- Programming Assignment:
  - Write a program the uses multiple threads to print at random intervals.

# Assessment Tools:
## CSC 216 Computer Architecture and Organization

- Bonus Test on GPU Architecture and Computing:
  - What is heterogeneous system? Please provide an example?
  - How GPUs have evolved from special graphics processors to programmable general-purpose parallel processors?
  - What is GPU computing?
  - What is Basic Unified GPU Architecture?
  - What is CUDA?
  - What are the three CUDA Paradigms?
  - Please draw a block diagram to show the structure of a contemporary PC with Intel CPU.

# Assessment Tools:
## CSC 216 Computer Architecture and Organization

- Programming Assignments:

**1.** **The following code is computing y = ax + y with a serial loop, where x and y are arrays of length n and a is a scalar constant.**

```
void saxpy_serial (int n, float alpha, float *x, float *y)
{
    for (int i=0; i<n; i++)
        y[i]  = alpha*x[i] + y[i];
}
```

**The code in the next line will invoke the serial function saxpy_serial.**

```
    saxpy_serial(n, 2.0, x, y)
```

**Please write a CUDA kernel function to compute the same y = ax +y in parallel. Please also show how to invoke the CUDA kernel.**

**2.** **The following CUDA kernel function utilizes three different levels of memory (global, shared and local). Please fill in the following blanks with a digit from 1 to 4 to indicate which is faster (1: fastest, 4 slowest).**

```
--global–
void foo(float *x, float *y, float *z)
{                                                    //x, y, z are global variables
    --shared--  float a, b, c;          //a, b, c are shared memory
    float               s, t, u;                              //s, t, u are local memory

    s = *x;                                                  //_____
    t = s;                                                   //_____
    a = b;                                                   //_____
    *y = *z;                                                 //_____
}
```

# Assessment Tools:
## CSC 312 Advanced Computer Architecture

- **Exam1 Q1**

  Define the following terms briefly:
  - Flynn's Taxonomy:
  - Processor Level Parallelism:
  - Instruction Pipelining:
- **Exam3 Q1**

  An MIMD machine has m processors and m memory modules interconnected by means of a single shared-bus. Suppose that a processor tries to use the bus in a given cycle with probability, p. what is the probability that:
  - The bus is idle (zero requests)?
  - Exactly one request is made?
  - More than on request is made
- **Exam3 Q2**

  what are the advantages and disadvantages of using a single shared-bus as an interconnection network in a multiprocessor system?
- **Project1 :** Provide a comprehensive survey of various benchmarks used to report performance of Desktop, Server and Embedded Systems. Show some comparative results.
- **Project2 :** Research the topic of Dual and Multi Core Processors; provide information about caching, pipelining, energy consumption and performance?

# Assessment Tools:
## CSC 325 Operating Systems

- **Exam 2- 2. Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single-processor system?**
- **Exam 2- 4. Write a structure of process synchronization solution with "TestAndSet."**

```
boolean TestAndSet (boolean *target) {
        boolean rv = *target;
    *target = TRUE;
    return rv:  }
```

**Shared boolean variable lock., initialized to false:**

# Assessment Tools:

## CSC 325 Operating Systems (continue)

- **Final 3. Fill in the following blanks to solve the Producer-Consumer Problem with the below condition: (i)Producer process produces information that is consumed by a consumer process; (ii) Bounded-buffer assumes that there is a fixed buffer size.**

```
Shared data
semaphore full, empty, mutex;
Initially: full = 0, empty = n, mutex = 1

Producer Process_i:
 do {
        …..
   produce an item in nextp
            …..
(                                                          );          );
        wait(mutex);
            ….      add nextp to buffer     …..
         signal(mutex);
(                                                          );          s)
      } while (1);

Consumer Process_j:
   do {
        wait(full)
(                                                          );   x);
          …..     remove an item from buffer to nextc
           …..
(                                                  );          si;
         signal(empty);
           ……
         consume the item in nextc
           ….
      } while (1);
```

# Assessment Tools:

## CSC 325 Operating Systems (continue)

- **Exam 1- 2 Write three primary thread libraries.**
- **Exam 1-III-1**. Including the initial parent process, how many processes are created by the program shown below? To verify your answer, you need to draw a graph to show the spawning process.

```
#include <stdio.h>
#include <unistdio.h>
int main (){
/* for a child process */
fork();
/* for a child process */
fork();
/* for a child process */
fork();
/* for a child process */
fork();
/* and fork another */
fork();
return 0; }
```

# Assessment Tools:

## CSC 325 Operating Systems (continue)

- **Exam 1-III-2. Using the program below, identify the value of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2400 and 2700, respectively)**

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
   int main (){
pid_t pid, pid1;
pid=fork();
if(pid< 0){ /* error occurred */
fprintf(stderr, "Fork Failed"); return 1;
} else if (pid == 0) { /* child process */
pid1 = getpid();
printf("child: pid = %d", pid); /* line A, pid= */
printf("child: pid1 = %d", pid1); /*line B, pid1= */
} else { /* Parent process */
pid1 = getpid();
printf("Parent: pid = %d", pid); /* line C, pid = */
printf("Parent: pid1 = %d", pid1); /*line D, pid1= */ }
return 0; }
```

# Assessment Tools:

## CSC 350 Organization of Programming Languages

Read the following tutorial for concurrency and answer the following questions:

http://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html

1.There are two ways to implement threads. Describe them. Also describe what the following 2 program segments do.

```
public class HelloRunnable implements Runnable {
   public void run() {
      System.out.println("Hello from a thread!");
   }

   public static void main(String args[]) {
      (new Thread(new HelloRunnable())).start();
   }
}
```

```
public class HelloThread extends Thread {

        public void run() {
        System.out.println("Hello from a thread!");
        }

        public static void main(String args[]) {
        (new HelloThread()).start();
        }
 }
```

# Assessment Tools:

## CSC 350 Organization of Programming Languages (continue)

2.Describe what the following program segment does:

```
public class SleepMessages {
   public static void main(String args[])
      throws InterruptedException {
      String importantInfo[] = {
         "Mares eat oats",
         "Does eat oats",
         "Little lambs eat ivy",
         "A kid will eat ivy too"
      };

      for (int i = 0;
          i < importantInfo.length;
          i++) {
         //Pause for 4 seconds
         Thread.sleep(4000);
         //Print a message
         System.out.println(importantInfo[i]);
      }
   }
}
```

# Assessment Topics

| Course Name | PDC Topics |
|---|---|
| CSC 119 Object Oriented Programming | Concurrent activities in Java, Java thread. |
| CSC 216 Computer Architecture and Organization | GPU Architecture and Computing, Simple CUDA programs. |
| CSC 312 Advanced Computer Architecture | Flynn's Taxonomy, Instruction Level and Processor Level Parallelism, Benchmarks. |
| CSC 325 Operating Systems | Algorithms for mutual exclusion, Parallel Computing and Algorithms. |
| CSC 350 Organization of Programming Languages | Understanding fundamental concepts in threads, Reading programs with threads. |

# Assessment Results

| Course Name | [EX, EF, M, U] Vector | Weighted Average |
|---|---|---|
| CSC 119 Object Oriented Programming | [3, 0, 0, 1] | 2.87 |
| CSC 216 Computer Architecture and Organization | [3, 0, 1, 1] | 3.00 |
| CSC 312 Advanced Computer Architecture | [2, 4, 3, 1] | 2.70 |
| CSC 325 Operating Systems | [8, 5, 1, 24] | 1.92 |
| CSC 350 Organization of Programming Languages | [2, 1, 1, 0] | 3.25 |

# Conclusions and Future Work

- PDC modules were implemented in the aforementioned courses.
- A presentation about PDC education for ACM computer science students was organized early in the fall semester. Students shared their views on how to integrate PDC into the computer science curriculum and had constructive feedback from students and faculty.
- Two graduate students were motivated about PDC education and attended the EduHPC 14 workshop, where we presented some of our early experience of PDC education at JSU [3].
- Assessment data was collected at the end of the fall semester. The data showed that students were comfortably able to learn PDC concepts and were motivated by these topics to pursue a career or do research in the area of PDC.
- To support our students with adequate PDC resources, a Tesla K40C, GPU hardware granted by NVIDIA Inc. [4] has been added to the Distributed Computing Laboratory. The GPU contains 2880 CUDA cores with 12 Giga Bytes of memory on the Tesla K40 and 400 GB HDD and another 192 CUDA cores on a Quadro 2000 GPU card. Combined, they provide adequate power to support simulations requiring high-power computing capacity.
- Matlab [5] Parallel Computing Tool box will soon be available for our students. Internal university funding is also sought to continue this project in the coming semesters.
- Course Continuous Improvement is a requirement that ABET accreditation teams always look for. Our Department FCARs (Faculty Course Assessment Reports) will include PDC modules to satisfy this requirement.

# References


[1] “NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing – Core Topics for Undergraduates.” [Online]. Available at: http://www.cs.gsu.edu/~tcpp/curriculum/.

[2] http://www.cs.gsu.edu/~tcpp/curriculum/?q=node/21183.

[3] A. Abu El Humos, S. Hong, J. Jackson, X. Liang and T. Pei, “NSF/TCPP Early Adopter Experience at Jackson State University Department of Computer Science”. Workshop on Education for High-Performance Computing EduHPC-14, in conjunction with SC-14: The International Conference for High Performance Computing, Networking, Storage, and Analysis, New Orleans, LA, November 16-21, 2014.

[4] http://www.nvidia.com.

[5] http://www.mathworks.com.