Study Guide

Week 1 Monday January 11th 2016 - Sunday January 17th 2016

Author B. Wilkinson Modification date Jan 4, 2016

Study Materials on Moodle

- *PowerPoint Slides*
  - Outline (Spring 2016)
  - Assignment Preliminaries
  - Parallel Computing Demand
  - Parallel Computing Potential
  - Parallel Computers
  - Programming with Shared Memory-1
- *Video*
  - Lecture 1 - 75-minute video of the first class in Fall 2014, covering the outline, the pre-assignment, and an introduction to parallel computing.
  - Lecture 2 was not recorded. (It covered *Parallel Computing Potential, Parallel Computers, and Programming with Shared Memory-1)*

Tasks

- **Mini-Quiz:** No mini-quiz in the first week
- **Pre-Assignment** - Setting up software environment
  - Pre-assignment Instructions
  - Parallel Programming Software
  - Additional Information
  - Pre-Assignment Set: Monday January 11th, 2016
  - **Pre-Assignment Due: *Friday January 22nd, 2016 (Week 2)***

Moodle Saba meeting -- 5 pm, Friday January 15th, 2016

This course is about high performance computing using multiple computers and processor cores.

The *Outline* for the course includes prerequisites, topics to be covered in the course, grading policies, code of student academic integrity, and instructor and TA details. This course is a programming course and basic knowledge of C and Java is expected. Most programming will be in C as this is the language most used in high performance parallel computing. The course will introduce several C-based tools. One tool does use Java. The level of knowledge of C includes pointers and dynamically allocated memory. Those with limited experience of C should be able to pick up what is needed during the course. A lot of sample code is given.

Assignment Preliminaries gives general details on what to submit for assignments. Assignment reports are submitted to Moodle and have late penalties. *Reports must be a single pdf file.* The instructions for each assignment will list specific results and screenshots of program execution, and code you must include. Generally you will not be asked to demonstrate your programs face-to-face with the instructor/TA unless

there is question over the screenshots. *It is critically important that you do not copy code from others or the Internet.*

Parallel Computing Demand briefly introduces the motivation for achieving high performance computing using multiple computers or processor cores. The key idea is to use multiple computers or processors collectively rather than one processor to solve a problem faster. It is a very simple idea (and old idea) -- *N* computers used collectively might create results *N* time faster if we can divide the problem into *N* parts executing at the same time. We will see later that this speedup is generally an upper limit and such speedup may be difficult to achieve. Nevertheless, there are many important problems that need to be solved quicker than is possible on a single computer and using multiple computers or processors offers an attractive approach. Sometimes there is a hard deadline for the result, for example flight control computers. Sometime there is a soft deadline, in that if it takes longer, the work can still proceed but not efficiently, for example computers used in CT scanners. Sometimes the problem cannot be solved in any reasonable time, for example it would take years to produce the result with some scientific calculations. These are called *grand challenge problems*.

A recent motivation is that all computers nowadays use multicore processors. We now need to be able to program these processors to utilize the multiple processor cores, even outside scientific computing.

First in *Parallel Computing Potential*, we establish whether it is possible to get increased execution speed using multiple computers. We have to establish this before moving forward with parallel computing. The principal metric is speed-up factor. Amdahl's law gives the maximum speed up assuming we can divide the computation into parts that can executed at the same time and there is a part that cannot be divided up and must executed sequentially. The maximum speed-up is 1/*f* with an infinite number of processors, where *f* is the fraction of the computation that cannot be divided up. This discouraging result was countered later with Gustafson's law, which indicates almost linear speed up if we scale the problem to a larger size on the multiprocessor. You are expected to be able to derive and use Amdahl's law.

In *Parallel Computers*, the two principle types of parallel computers are outlined – the shared memory system and the distributed memory systems. GPU systems are mentioned. The basic way to program shared memory systems is by using threads, directly or indirectly. Distributed memory systems are usually programmed using message passing libraries. Note multicore processors are shared memory systems.

Programming with Shared Memory-1 start the topic of programming shared memory systems, introducing processes, fork pattern, fork-join pattern, threads, Pthreads example, and the thread pool pattern. Note processes have their own memory and resource allocation. Processes are introduced here to differentiate with threads and because the fork and fork join patterns are seen with processes and appear with low level thread routines. You might have already done some programming with threads in an operating systems class possibly using Pthreads. Here we will focus on the higher level OpenMP and you should not get any questions on Pthreads.

Pre-Assignment: Assignments require certain software to be installed (MPI, OpenMP, etc.). A "Pre-assignment" is set this week to create the necessary software environment on your computer (Windows, Mac or Linux). It is critical that this assignment is completed quickly to be able to continue with future assignments. It should be very straightforward and asks you to download VirtualBox and a prepared course virtual machine appliance (VM) that executes within VirtualBox. Read the assignment instructions for directions.

Parallel Programming Software: Provides the link to the course VM. Notice the VM is large (2GB+) and may take some time to download. Use a fast connection. The VM appliance is 32-bits and does not require any specific VM support on the host machine. It should work with any processor. Always check the *Assignment FAQ* found under *Additional Information* for any issues. The VM also has sample programs pre-installed for each assignment. These are also provided separately should you want to use them in a native installation. The zip file should make it straightforward to import the sample files in that case.

You have the alternative of installing the course software natively on a Linux platform and detailed instructions are given for that for a Ubuntu distribution (see *Additional Information*). This would probably be better if you already have a Ubuntu installation (or another Linux distribution). Note you must use the exact software specified, e.g. OpenMPI not MPICH. Generally if you have Windows system, use the VM. If you have a Mac or Linux system, I would try a native installation first.

Although extremely convenient to use your own computer for program development, executing programs on your own computer may not show any speed up even with a multicore processor as host OS's such as Windows may not use the multiple cores for a single application. (VirtualBox may be seen as a single application). Hence, in addition to doing the programming and testing on your own computer, you will be asked on occasion to demonstrate programs and speedup on the UNC-Charlotte cci-gridgw cluster. Read the notes on this cluster "Using UNC-C cci-gridgw cluster" (*Additional Information*) before continuing with Part 4 of the Pre-assignment. You will have to wait for accounts to be set up before being able to complete Part 4, but do not wait to complete Part 1, Part 2, and Part 3. You will receive an email when your account has been set up. If you have insurmountable problems not covered in the *Assignment FAQs*, please contact me by email (abw@uncc.edu)

Make sure you set up a shared folder as described in the assignment and maintain a backup of all your program files at another location outside the VM (say on a flash drive) as we have seen a situation when Windows automatically updates and negatively affects the installed VirtualBox VM.