Study Guide

Week 8 Monday February 29th, 2016 – Sunday March 6th, 2016

Author B. Wilkinson Modification date February 27, 2016

Study Materials on Moodle

- *PowerPoint Slides*
    - Seeds framework
    - Stencil pattern
- *Videos*
    - Demo of *N*-body problem (mp4 file)
    - Lecture 13 video: 75-minute video of Lecture 13 in Fall 2014 finishing Paraguin and outlining the Seeds framework.
    - Lecture 14 video: 75-minute video of Lecture 14 in Fall 2014 on the synchronous all-to-all pattern and the stencil pattern.
- *Sample Quiz Questions*
    - Seeds quiz questions
    - Iterative synchronous pattern quiz questions
    - Stencil quiz questions

Tasks

- **Mini-Quiz:** Answer the short posted quiz before 11:55 pm Sunday March 6th, 2016.
- **Continue working on Assignment 4** Second MPI Assignment - Monte Carlo π workpool
    - Assignment 4 Due: *Sunday March 20th, 2016 (Week 9)*

Moodle Saba meeting – Friday March 4th, 5 pm

Seeds Framework describes a high-level Java-based pattern based parallel programming framework developed at UNC-Charlotte called Seeds, which enables a programmer to create and execute a fully distributed computation without any explicit message-passing. Several patterns are already implemented including the ubiquitous workpool. A neat feature of this framework is that it will deploy on any platform, local computer, a cluster, or a geographically distributed computers without first loading any software on the computers (except Java). You are expected to know some Java. The code given uses generics. The Seeds framework has the highest level of abstraction of the tools described. You will get a change to try Seeds in Assignment 6 in Week 11.

Next the course moves onto various patterns and applications.

Synchronous All-to-All Pattern describes the iterative synchronous all-to-all pattern, and two applications that can be implemented using this pattern, the *N*-body problem and solving a system of linear equations. The objective of the *N*-body problem is to determine the movement of bodies subject to forces between them. The specific *N*-body problem described in the slides is the gravitational *N*-Body problem where the bodies are astronomical bodies, but there are many *N*-body problems, for example where the bodies are molecules, atoms, sub-atomic particles, etc. The equations governing the movement of the bodies will depend upon the type of bodies. Astronomical bodies are attracted to each other by gravitational forces. If the bodies were electrons, the forces would repel the bodies, not attract them. You are expected to understand basic laws of Physics (Newton's laws from the 17th century). The gravitational force between two bodies is computed using Newton's law of universal gravitation. Each body will have forces from all the other bodies. These forces have to be added together in the *x*, *y*, and *z* directions. The acceleration that these accumulated forces cause is computed using Newton's 2nd law, again in the *x*, *y*, and *z* directions. This leads to a new velocity for each body in the *x*, *y*, and *z* directions. "Velocity" is speed in particular direction. Finally the new positions are determined. Everything done in three dimensions if 3-*D* space using discrete approximations for the continuous variables. The whole process is repeated over a number of time intervals. For more information on Newton's law of universal gravitation, check out Wikipedia http://en.wikipedia.org/wiki/Newton%27s_law_of_universal_gravitation.

The material then moves onto the quite recent Barnes Hut algorithm (1986) for reducing the time complexity of each time step from $O(N^2)$ to $O(N \log N)$, which then makes the calculation practical for very large values of *N*. You are expected to know the Barnes Hut algorithm although not to code it in an assignment. Coding the basic $O(N^2)$ algorithm is straightforward and appears in Assignment 5 (Hybrid Suzaku/OpenMP/MPI programming). The material also mentions an important term, *Jacobi* iteration, the name given to a computation that uses the previous iteration values to compute the next values. We have already seen this in solving the Heat equation. Later in the stencil pattern, we will see an iteration method that uses not only values from the previous iteration but also from the present iteration and how to parallelize it. Also presented is a discussion on the accuracy of the result. Simply stopping the iterations when all the values do not change by some fixed value, *e*, does not create an answer to that precision.

The demo video shows the *N*-body problem using our VM to display X11 graphics. The actual motion presented in the graphics will depend on the constants used. Note if the bodies become very close, the forces will approach infinity. In program, colliding bodies are destroyed and removed from further consideration. You will be writing the code for this in Assignment 5.

Stencil Pattern introduces the message-passing pattern where each process only communicates only with its neighbors in a 2- or 3-dimensional mesh. This pattern matches many physical problems. Normally the computation repeats, so the stencil pattern is actually an "iterative synchronous" stencil. We have already seen the stencil pattern in the assignment to solve the steady state or static heat equation (heat distribution problem) and this is described in the slides also.