Study Guide

Week 5 February 8th, 2016 – February 14th, 2016

Author B. Wilkinson   Modification date February 6, 2016

Study Materials on Moodle

- *PowerPoint Slides*
  - MPI Collective Routines
  - MPI Synchronization
- *Video*
  - Lecture 8 video: 75-minute video of Lecture 8 in Fall 2014 on collective data transfer routines.
  - Lecture 9 video: 75-minute video of Lecture 7 in Fall 2014 on synchronization.
- *Sample Quiz Questions*
  - Collective MPI routines
  - Asynchronous MPI

Tasks

- **Mini-Quiz:** Answer the short posted quiz before 11:55 pm Sunday February 14th, 2016.
- **Complete Assignment 2** OpenMP heat distribution problem, graphics
  - Assignment 2 Due: *Sunday February 14th, 2016* (Week 5)
- **Assignment 3 MPI tutorial using command line and Eclipse-PTP**
  - Assignment 3 Instructions
  - Assignment 3 Due: *Sunday February 28th, 2016* (Week 7)

Moodle Saba meeting – *Friday February 12th*, 2015, 4 pm

MPI Collective Routines continues the material on MPI and introduces collective data transfer patterns - broadcast, scatter, gather, reduce, and all-to-all. An important aspect of these routines is that all processes must call the routines with the same (or equivalent) parameters. This is actually very convenient with MPI usually consisting of a single program. Also although collective operations could be achieved by the programmer with multiple point-to-point MPI routines, it expected that the collective MPI routines are more efficient. The MPI standard does not specifically say how they are to be implemented although the semantics are to be same as using locally blocking point-to-point MPI routines in the manner they return. Unfortunately that is not a precise statement as it will depend upon how they are implemented. We can expect MPI data transfer collective routines to be implemented using a tree construction that leads to logarithmic time complexity. Generally the sources of data will return when all the data has been sent and each destination will return when they receive the data they are waiting for, i.e. processes do not wait for each other. If you need for all processes to wait for each other before continuing you will need to add an explicit barrier, see the next section. Other things to note about collective data transfer routines is that one process is called the *root* (not necessarily rank = 0), which is either is the destination collecting all the data or the source sending all the data out to other processes. The data itself can be a single data item such as an integer or floating point number or a group of items of the same type stored in consecutive locations (i.e., parts of an array). In an exam, you are not expected to memorize the parameter list of MPI routines, as that will be given, although you will be expected to write MPI code using these routines.

The Powerpoint slides match those use in the Fall 2014 videos. However I have doubts about using scatter where the number of blocks does not equally divide into data array, leaving some processes without any data. The MPI standard seems to be silent on this matter and I think it is better not to do this. Assignment 3 Part 4 actually addresses the possibility in a different way.

MPI Synchronization explores how to synchronize MPI processes, covering synchronous point-to-point send routine MPI_SSend(), three-way protocol implementation, asynchronous routines MPI_ISend() and MPI_IRecv() and how they might convert to synchronous routines, and MPI_Barrier() routine and its implementation. Finally safety and deadlock are discussed and the use of deadlock-free routines including MPI_Sendrecv(). Note MPI_Recv() is defined as a locally blocking routine and used with MPI_Send() but it is also synchronous as it also only returns when the transfer is complete and hence is also used with MPI_SSend(), i.e. there is no "MPI_SRecv()" routine.

Sample Quiz Questions: Review the sample questions on collective and asynchronous routines.

Assignment 3 Instructions: Assignment 3 is a tutorial on MPI with most of the MPI code given. The focus is on matrix multiplication. The code is executed on the command line and through Eclipse-PTP on the VM (or native installation) and on the UNCC cluster. For the UNCC cluster, review the notes on using the cluster carefully first. The last part, Part 4, is the only part that needs a little new code to be devised. The purpose of Assignment 3 is to become proficient in compiling and executing MPI code on a variety of setting and very detailed instructions are given. Although you have two weeks to complete this assignment, start early.

Note: If some of the nodes of the cluster are unavailable in the last few days before the assignment is due (Week 7) use those nodes that are available. If all the nodes the cluster are unavailable, use your own computer to complete Part 3. Of course you will not be able to use multiple computers. Watch for announcements. Document in your report about the availability of nodes. Best to start early to avoid this possibility.

There will be an online test in Week 6. Watch for announcements.