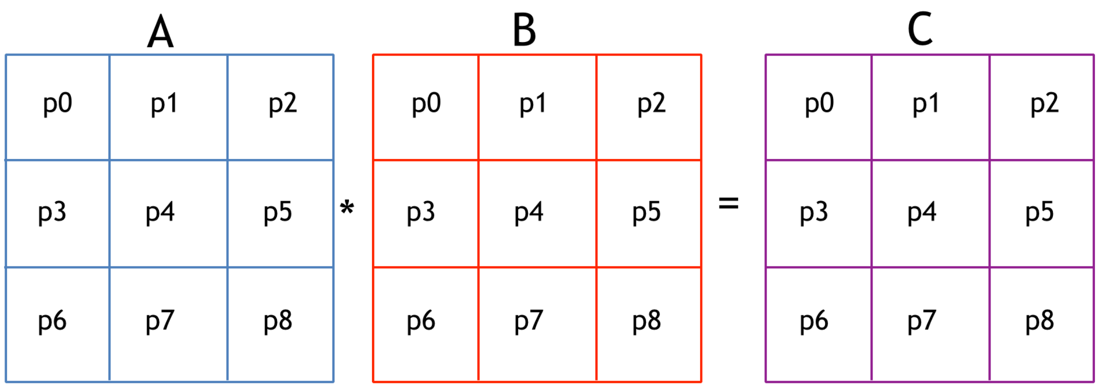
CSE 891-Section 1
Parallel Computing: Fundamentals and Applications

Homework 2

Due 5 pm, Friday, October 10, 2014

1. (40 points) Recall the reduction algorithms, Alg 2 and Alg 3, discussed in Lecture 7.
   A. Implement the (sum) reduction for a vector of length N using only point-to-point MPI primitives (MPI_Send, MPI_Recv and variants). You can assume $p = 2^m$.
   B. Implement the vector (sum) reduction using MPI collectives (i.e. MPI_Reduce).
   C. Plot the running times of the three implementations (Alg 2, Alg 3 and MPI_Reduce) vs. number of cores used (p = 1, 4, 16, 64, 256) for N=10, N=100, N=5000. Produce 3 separate plots for clarity.
   D. Interpret your results in C. Which implementation scales best, which one performs worst, why?

2. (60 points) Consider three square matrices of dimensions N. Recall the matrix multiplication algorithms that we discussed in class.
   A. Implement the (sequential) naive and blocked matrix multiplication algorithms. For blocked matrix multiply, use only one level of blocking (6 for loops), and experiment with block sizes of 16, 32 and 64 for best performance.
   B. For 100 <= N <= 2000, compare the performance (in Gflop/s) of your implementations against DGEMM (http://www.math.utah.edu/software/lapack/lapack-blas/dgemm.html). Comment on your observations. In particular, what kind of other optimization methods could the vendor DGEMM be using for achieving high performance?
   C. Implement a parallel matrix multiplication based on a 2D partitioning (see the figure below). For local matrix multiplications on each processor, use the best performing implementation based on your results in parts A and B above.
   D. Plot the running time of your MPI implementation vs. the number of cores used (p = 1, 4, 16, 64, 256) for N = 6144.



Turn-in Instructions

Please turn in a hard-copy of your homework at my office (1126 Engineering Building) on the due date (slide it under the door, if I am not there). If you will not be on campus that day, you can email me a pdf.

We will use the CSE Handin system for turning in the source code, instructions will follow. In case CSE Handin system does not work for students in other departments, we will resort to emailing.

Suggestions

- Start early!
- Familiarize yourself with the HPCC computing environment, documentation and help resources:
- https://wiki.hpcc.msu.edu/
- You can contact HPCC for technical questions.
- Note that you can consolidate several runs into a single job. For example, when you request 256 cores, you can test the Alg2, Alg3 and MPI_Reduce for N=1, 50 and N=2500.
- We will discuss how to take timings in class. Also note that for small jobs (i.e. N=1 on p=4), you need to do several runs (10 or potentially more) and take the average for precise timings.
- Use development nodes to make sure that your code runs correctly. You have up to 20 cores on a single development node.
- In your job scripts, to reduce the queue wait times, make sure not to request a long reservation. For most tests you will need to do, 5-10 minutes should be plenty.