Homework #3

(Due: May 10)

Task 1. [**120 Points**] **Distributed-Memory Matrix Multiplication.**

(a) [**50 Points**] Implement the three distributed-memory algorithms for multiplying two square matrices shown in Figures 1–3. Assume the initial distribution of input matrices and the final distribution of the output matrix from lecture 13.

(b) [**10 Points**] Use your implementations from part 1(a) to multiply two $2^k \times 2^k$ matrices (initialized with random integers in $[-100, 100]$) on $2^l \times 2^l$ compute nodes with 1 process/node for $10 \le k \le 14$ and $0 \le l \le 2$. Report the running times and explain your findings.

(c) [**10 Points**] Repeat part 1(b) with t processes/node, where t is the number of cores available on a compute node. Report the running times. Compare with part 1(b) and explain.

(d) [**20 Points**] Suppose a master node initially holds the input matrices and will hold the final output matrix. Augment your fastest implementation from part 1(a) with efficient routines for initial distribution and final collection of matrices. When you measure the running time of this algorithm please include the time needed for these additional distribution/collection steps.

(e) [**10 Points**] Repeat part 1(b) with the algorithm from part 1(d).

(f) [**10 Points**] Repeat part 1(c) with the algorithm from part 1(d).

(g) [**10 Points**] Compare your results from parts 1(b, c) with those from parts 1(e, f). Explain your findings.

Task 2. [**80 Points**] **Distributed-Shared-Memory Matrix Multiplication.**

(a) [**30 Points**] Modify your fastest implementation from part 1(a) and its modified version from part 1(d) to use a shared-memory parallel matrix multiplication algorithm inside each process. Use your fastest shared-memory parallel matrix multiplication routine from HW1.

(b) [**40 Points**] Repeat part 1(b) with the two implementations from part 2(a). Use 1 process/node, but inside each process use all cores available on that node.

(c) [**10 Points**] Compare your results from parts 1(b, c, e, f) with those from part 2(b). Explain your findings.

MM-rotate-A-rotate-B (C, A, B, n, p)

1. decompose each $n \times n$ matrix $X \in \{A, B, C\}$ into $\sqrt{p} \times \sqrt{p}$ blocks of size $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ each and for $0 \leq i, j < \sqrt{p}$, let $X_{i,j}$ be the block on the i^{th} block row and j^{th} block column
2. arrange the p processors into a $\sqrt{p} \times \sqrt{p}$ grid and for $0 \leq i, j < \sqrt{p}$, let $P_{i,j}$ be the processor on the i^{th} row and j^{th} column of the grid
3. for $0 \leq i, j < \sqrt{p}$, initially processor $P_{i,j}$ holds $A_{i,j}$ and $B_{i,j}$
4. parallel: for $0 \leq i, j < \sqrt{p}$, each $P_{i,j}$ does the following:
   (i) sets $C_{i,j} \leftarrow 0$
   (ii) sends $A_{i,j}$ to $P_{i,(\sqrt{p}+j-i) \bmod \sqrt{p}}$ (rotate left by i grid locations)
   (iii) sends $B_{i,j}$ to $P_{(\sqrt{p}+i-j) \bmod \sqrt{p}, j}$ (rotate upward by j grid locations)
5. for $l \leftarrow 1$ to $\sqrt{p}$ do
6. parallel: for $0 \leq i, j < \sqrt{p}$, each $P_{i,j}$ does the following (assuming $k = (j + i + l - 1) \bmod \sqrt{p}$):
   (i) sets $C_{i,j} \leftarrow C_{i,j} + A_{i,k} \times B_{k,j}$
   (ii) if $l < \sqrt{p}$, sends $A_{i,k}$ to $P_{i,(\sqrt{p}+j-1) \bmod \sqrt{p}}$ (rotate left by 1 grid location)
   (iii) if $l < \sqrt{p}$, sends $B_{k,j}$ to $P_{(\sqrt{p}+i-1) \bmod \sqrt{p}, j}$ (rotate upward by 1 grid location)

Figure 1: Distributed matrix multiplications using block rotations for both input matrices.

MM-rotate-A-broadcast-B (C, A, B, n, p)

1. decompose each $n \times n$ matrix $X \in \{A, B, C\}$ into $\sqrt{p} \times \sqrt{p}$ blocks of size $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ each and for $0 \leq i, j < \sqrt{p}$, let $X_{i,j}$ be the block on the i^{th} block row and j^{th} block column
2. arrange the p processors into a $\sqrt{p} \times \sqrt{p}$ grid and for $0 \leq i, j < \sqrt{p}$, let $P_{i,j}$ be the processor on the i^{th} row and j^{th} column of the grid
3. for $0 \leq i, j < \sqrt{p}$, initially processor $P_{i,j}$ holds $A_{i,j}$ and $B_{i,j}$
4. parallel: for $0 \leq i, j < \sqrt{p}$, each $P_{i,j}$ sets $C_{i,j} \leftarrow 0$
5. for $l \leftarrow 1$ to $\sqrt{p}$ do
6. parallel: for $0 \leq i, j < \sqrt{p}$, each $P_{i,j}$ does the following (assuming $k = (j + l - 1) \bmod \sqrt{p}$):
   (i) if $k = i$, broadcasts $B_{i,j}$ to $P_{0,j}, P_{1,j}, \ldots, P_{\sqrt{p}-1,j}$ (broadcast to grid column j)
   (ii) sets $C_{i,j} \leftarrow C_{i,j} + A_{i,k} \times B_{k,j}$
   (iii) if $l < \sqrt{p}$, sends $A_{i,k}$ to $P_{i,(\sqrt{p}+j-1) \bmod \sqrt{p}}$ (rotate left by 1 grid location)

Figure 2: Distributed matrix multiplications using block rotations for one input matrices and block broadcasts for the other.

APPENDIX 1: What to Turn in

One compressed archive file (e.g., zip, tar.gz) containing the following items.

– Source code, makefiles and job scripts for both tasks.

– A PDF document containing all answers.

MM-broadcast-A-broadcast-B (C, A, B, n, p)

1. decompose each $n \times n$ matrix $X \in \{A, B, C\}$ into $\sqrt{p} \times \sqrt{p}$ blocks of size $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ each and for $0 \leq i, j < \sqrt{p}$, let $X_{i,j}$ be the block on the i^{th} block row and j^{th} block column
2. arrange the p processors into a $\sqrt{p} \times \sqrt{p}$ grid and for $0 \leq i, j < \sqrt{p}$, let $P_{i,j}$ be the processor on the i^{th} row and j^{th} column of the grid
3. for $0 \leq i, j < \sqrt{p}$, initially processor $P_{i,j}$ holds $A_{i,j}$ and $B_{i,j}$
4. parallel: for $0 \leq i, j < \sqrt{p}$, each $P_{i,j}$ sets $C_{i,j} \leftarrow 0$
5. for $l \leftarrow 1$ to $\sqrt{p}$ do
6. parallel: for $0 \leq i, j < \sqrt{p}$, each $P_{i,j}$ does the following (assuming $k = l - 1$):
   - (i) if $k = j$, broadcasts $A_{i,j}$ to $P_{i,0}, P_{i,1}, \ldots, P_{i,\sqrt{p}-1}$ (broadcast to grid row i)
   - (ii) if $k = i$, broadcasts $B_{i,j}$ to $P_{0,j}, P_{1,j}, \ldots, P_{\sqrt{p}-1,j}$ (broadcast to grid column j)
   - (iii) sets $C_{i,j} \leftarrow C_{i,j} + A_{i,k} \times B_{k,j}$

Figure 3: Distributed matrix multiplications using block broadcasts for both input matrices.

APPENDIX 2: Things to Remember

– **Please never run anything that takes more than a minute or uses multiple cores on login nodes.** Doing so may result in account suspension. All runs must be submitted as jobs to compute nodes (even when you use Cilkview or PAPI).

– Please store all data in your work folder ($WORK), and not in your home folder ($HOME).