CS 594 HW 3 Yuping Lu

I run all the programs on EECS hydra clusters.

MPI Datatype

diagonal : diagonal.c
a datatype describing matrix diagonal.
Command: mpirun -np 2 diagonal (The process number is fixed)
Input:

1, 2, 3, 4, 5, 6, 7, 8,
9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54, 55, 56,
57, 58, 59, 60, 61, 62, 63, 64

Output: rank= 1 b= 1 10 19 28 37 46 55 64

C arrays are stored in row-major order while FORTRAN arrays are stored in column-major order.

transpose : transpose.c
a datatype for matrix transpose
Command: mpirun -np 2 transpose (The process number is fixed)
Input:

1, 2, 3, 4, 5, 6, 7, 8,
9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54, 55, 56,
57, 58, 59, 60, 61, 62, 63, 64

Output:

rank= 1
1 9 17 25 33 41 49 57
2 10 18 26 34 42 50 58
3 11 19 27 35 43 51 59
4 12 20 28 36 44 52 60

5 13 21 29 37 45 53 61
6 14 22 30 38 46 54 62
7 15 23 31 39 47 55 63
8 16 24 32 40 48 56 64

Ring Application

ring : ring.c
a simple application that passes a token between all the processes in one communicator.
Command: mpirun -np 4 ring
Output:

rank= 1, received token 0.
rank= 2, received token 1.
rank= 3, received token 2.

Process Grids & Multi-pipeline communications

I put these two tasks together. I first create the row and column communicators and then do the iterations to pass tokens.
grids : grids.c
Command: mpirun -np 9 grids (The process number is fixed)
Input: These are the original tokens.

1, 2, 3, 4, 5, 6,
7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17, 18,

19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36

P=3, Q=3, so in this example above, process 0 has 1, 4, 19, 22. The tokens will be passed through rows and columns and back to their original position in the end. I only output the first token and second token on the process 0 after the iterations. If they are the same with the original one, then the program is correct.

P*Q is the grid size. So we need P+Q-2 steps to have a full exchange of tokens.

Output:

Row iteration finishes.
rank[0], 1st token: 1, 2nd token: 4
Column iteration finishes.
rank[0], 1st token: 1, 2nd token: 4

PDGEMM

I only implemented a simple matrix matrix multiplication.
PDGEMM("No transpose", "No transpose", N, N, N, α = 1, A, B, β = 0, C)
And I haven't successfully implemented the task with all the requirements, e.g. a user supplied argument k. But this parallel version works and the result is accurate.

Command: mpirun -np 6 a1.txt b1.txt (a1.txt and b1.txt are filenames)
Output: (a matrix C)
Elapsed time: 0.000057 seconds.
The Matrix C.
2.119985 2.773072 1.938637 2.246718 1.949592 3.105905 2.937909 2.226654 2.549402 …
…… (The detailed output of the matrix is omitted here.)