Parallel Programming
HW4

CHANG LIU
chang.liu@jhu.edu

1. Using the nomeclature of Chapters 3 and 4 in *Patterns for Parallel Programming*, how would you characterize (describe and or justify your answers):

1.1 The decomposition of the problem (task or data)

The algorithm is based on the data geometric decomposition. We partition the whole array into blocks and each block has its own process created. All the processes have the same task: updating its own block. In our problem, the matrix is divides into blocks by row, the decomposition is linear in row dimension.

1.2 The dependencies between the decomposed parts of the problem

If two blocks are adjacent in the matrix, the corresponding process only has to synchronize those values on the boundary between them. In our problem, the boundary is the first and last rows for each block.

1.3 The algorithms structure of your solution

My algorithm is based on geometric data decomposition. The algorithm equally partition the matrix (d by d, and assume d = M x p) into M blocks by row. Then, it creates M processes as well as M small arrays for each p by d block. Each process updates its own block during the iterations. When updating the first row and the last row, the algorithm creates 2 surrogate arrays: top and bottom. For the i-th process, the top surrogate array stores the last row in the i-1 th block and the bottom surrogate array stores the first row in the i+1 th block. These surrogate arrays need to be synchronized after all blocks are updated. The synchronization has 2 steps:

1. Even processes send the first row and last row in their blocks to the surrogate arrays in their adjacent odd process. For example, the second process need to send the first row in the second block to the bottom surrogate array for the first process and send the last row to the top surrogate array for the third process.

2. Similarly even processes send their first row and last row in their blocks to surrogate arrays in their adjacent odd processes.

After the synchronization, data from all the blocks will be merged into a single array and printed out by the master process.

2. Consider an alternative spatial decomposition in which we divide the grid recursively into smaller squares.

2.1 What are the relative advantages and disadvantages of this decomposition when compared with the horizontal slicing of the space in your implementation?

Advantages:

This decomposition could get at most n2 partition whereas the horizontal slicing could get at most n partitions. So this new decomposition could get a better partition of the data and thus increase the overall parallelism.

Disadvantages:

In this decomposition each process will have 8 adjacent processes, whereas in the horizontal slicing each process only has 2 adjacent processes. Thus the new decomposition may bring more communication between processes as well as more interference.

2.2 On the number and size of MPI messages in the simulation?

In the new decomposition:

The decomposition using K by K partition

The number of MPI message: $8K^2$

The size of each MPI message: 1 or $\frac{n}{K}$

The total size of MPI message: $4nK + 4K^2$

In the horizontal slicing:

The decomposition using K by K partition

The number of MPI message: $2K^2$

The size of each MPI message: n.

The total size of MPI message: $2nK^2$

2.3 On the memory overhead at each processor?

In the new decomposition:

Memory overhead at each process: $4 \cdot \frac{n}{K} + 4$

In the horizontal slicing:

For K^2 partition, the memory over head at each process: $2n$

2.4 On the flexibility of decomposition?

In the new decomposition using K by K partition, the total size of MPI message is $4nK + 4K^2$,

the total memory overhead is $4nK + 4K^2$.

In the horizontal slicing using K^2 partition, the total size of MPI message is $2nK^2$,

the total memory overhead is $2nK^2$.

We can see that when K is large than 2 and n is large, the horizontal slicing will have bigger MPI messages and memory overhead, which means more cost. Thus the new decomposition is a trade-off between communication load and communication frequency.

2.5 For the new decomposition, describe a deadlock free messaging discipline for transmitting the top, bottom, left, and right sides and the top left, top right, bottom left and bottom right corners among neighboring partitions.

Assume we usa a K by K partition, and each process $P_{i,j}$ is assigned a data block, i is the row index and j is the column index.

The synchronization has 8 steps:

1. for $P_{i,j,}$ if i is even, then send its first row to $P_{i-1,j}$ and last row to $P_{i+1,j}$
2. for $P_{i,j,}$ if i is odd, then send its first row to $P_{i-1,j}$ and last row to $P_{i+1,j}$
3. for $P_{i,j,}$ if j is even, then send its first column to $P_{i,j-1}$ and last column to $P_{i,j+1}$
4. for $P_{i,j,}$ if j is odd, then send its first column to $P_{i,j-1}$ and last column to $P_{i,j+1}$
5. for $P_{i,j,}$ if i is even, then send its element in first row and first column to $P_{i-1,j-1}$ and the element in last row and last column to $P_{i+1,j+1}$
6. for $P_{i,j,}$ if i is odd, then send its element in last row and first column to $P_{i+1,j-1}$ and the element in first row and last column to $P_{i-1,j+1}$
7. for $P_{i,j,}$ if j is even, then send its element in first row and first column to $P_{i-1,j-1}$ and the element in last row and last column to $P_{i+1,j+1}$
8. for $P_{i,j,}$ if j is odd, then send its element in last row and first column to $P_{i+1,j-1}$ and the element in first row and last column to $P_{i-1,j+1}$