Eric Faurie
CS 546
04-MAR-2013

Project

Algorithm:

The general algorithm used is as follows. First the root process initializes the data by reading from the file. It then splits the rows in a Block Distribution and transmits the data to the other processes. They then compute the 1D-FFT on each row they receive. After which, they transmit the result back to Root. The Root transposes the matrix, re-distributes (Row Block Distribution, which is now essentially columns because of the transpose), and the processes complete the computation with another 1D-FFT on their rows. Then the processes each holding the final result in the rows of both matrices, compute the multiplication step, and then the first inverse 1D-FFT on the resulting matrix. All the processes transmit their resultant matrix to the Root. It then transposes it, redistributes, and the final inverse 1D-FFT is computed. Each process sends the data to the Root one last time, where it is written to a file by the Root process.

In 2D-Conv-P1, all communication is done with explicit Sends and Receives. In 2D-Conv-P2, all communication is done with Scatter and Gather.

2D-Conv-P3 is Task parallel. That is, the processors are split into groups. For Tasks 1 and 2 (doing FFT's on the two initial matrices), the processors are split into two separate communicators each containing half of the processors. One set of processors does the 2D-FFT on matrix A, while the second set does the 2D-FFT on matrix B. This is task parallelism as both Task 1 and 2 are being computed concurrently. Then all the processors send their data to the Global Root to consolidate. Then Task 3 and 4 carry on much as they did in the original algorithm because these two Tasks are dependent on one another. All processors collectively act in Task 3 and 4. Again, in this approach, all communication is done with Scatter and Gather collective calls.

Times:

	Time (1P)	Time (2P)	Time (4P)	Time (8P)
Serial 2D-Conv	0.409172	N/A	N/A	N/A
2D-Conv-P1 (Send/Recv)	0.411333	0.231616	0.587782	0.811868
2D-Conv-P2 (Scatter/Gather)	0.426544	0.238719	0.472829	1.05952
2D-Conv-P3 (Multi-Comm)	N/A	N/A	N/A	N/A

Speedup:

	Speedup (1P)	Speedup (2P)	Speedup (4P)	Speedup (8P)
2D-Conv-P1 (Send/Recv)	0.994746349	1.766596435	0.696128837	0.503988333
2D-Conv-P2 (Scatter/Gather)	0.959272666	1.714031979	0.865369933	0.386186198
2D-Conv-P3 (Multi-Comm)	N/A	N/A	N/A	N/A

Problems:

I cannot figure out why the third approach, that is using Multiple Communicators for task parallelism is not completing. It Segmentation Faults at some point in the program, but it does compile.

It's also concerning that the only cases that the speedup was positive was when there were only 2 processors. This shows that the amount of communication is too high for the higher number of processors to be beneficial.