Parallel Programming Assignment-3: Page rank algorithm

Swayambhu Nath Ray
CDS M Tech
SR No. - 13288

28 March 2017

Method (Program Outline): A very simplistic version of the page rank has been implemented in MPI as well as in C++ for comparing the time required for the given input graph file. The steps followed for implementing the page rank algorithm as described in the given paper is as follows:

1. All the processors is reading the entire input file and then from the file select its own portion of the adjacency matrix that it will work on.
2. After forming the adjacency matrix for its own set of nodes, it is calculating the weights each node will send to all its neighbouring nodes and populating a weight array.
3. An allreduce is performed on the weight array to find the resultant weights for all the nodes and thus the variable part of the tentative page rank is calculated.
4. Steps 2 and 3 are performed for 30 iterations and the final weights for each node is calculated by all reduce in each iteration.
5. At the end of the iteration each processor has the entire tentative page rank algorithm with itself to find the rank of the node it holds.

The same algorithm is implemented in C++ for the sequential version to judge the performance of the parallel algorithm and to be fair with both the implementation.

Time required for the sequental algorithm - 1.211284 seconds
The results of the parallel algorithm for 8, 16, 32, 64 and 96 processors are shown below:

No. of processors	Time required in seconds	Speed-up obtained
8	0.226892	5.3386
16	0.192517	6.2918
32	0.155341	7.7976
64	0.145925	8.3007
96	0.167582	7.2280

Table 0.1: Speedup table.

The speed-up obtained is plotted in the following graph to get an idea of the scalability of the page rank algorithm with the number of processors:-

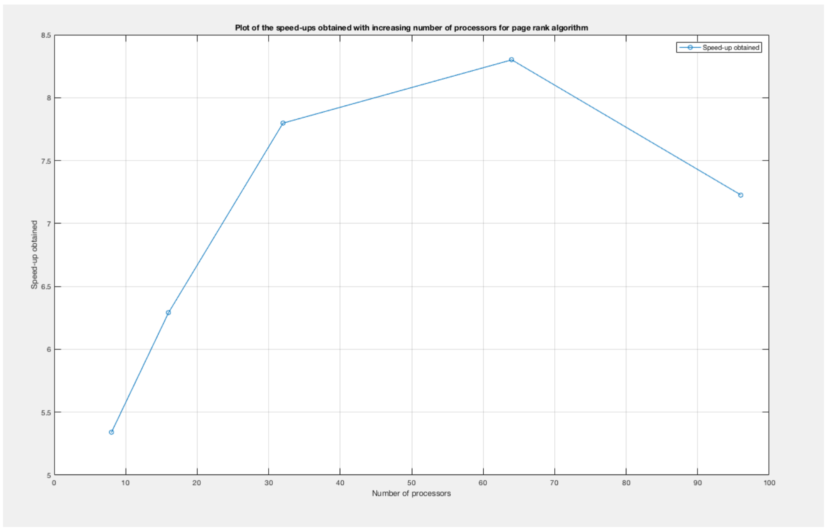

Figure 0.1: Plot for speed-up.

Observations: From the above graph we can see that the speed-up obtained increases initially but decreases after the number of processors is increased beyond 64. The speed-up obtained is actually determined by the computations and communications required for the algorithm and their efficient overlapping due to parallelism. In case of lower number of processors the computations play the main role and hence the delay due to communication is covered up by the computations done. Now when he number of processors is increased the communication begins to play the main role due to the all reduce performed. Thus we experience a lower speed up in case of 96 processors. Here the communication plays the important role as the number of processor is higher. Hence with more number of processor the dominant factor will be communication. Thus implementing this algorithm will be best executed with 64 processors so far our experiments proves and no better result will be obtained with increasing number of processor.

The parallel and sequential code is attached with this report.

_ _