High Performance Computing

Assignment 6

In this assignment, you will use MPI parallel computation to predict movie ratings using the Netflix data set.

The Netflix matrix is the matrix A, where an entry a_{ij} in the matrix represents the rating that user i assigns to movie j. Obviously not all entries of the matrix are known; it is desired to know what rating a user would assign to a movie that he has not seen.

The input to your program is a sparse matrix corresponding to the known movie ratings collected by Netflix. The matrix has 480,189 rows (users) and 17,770 columns (movies) and contains 99,072,112 ratings (from 1 to 5). The gzipped file for this matrix is located in http://www.edmondchow.com/files/netflix.dat.gz and is 313 MB in size (1.4 GB uncompressed). Each line in the file contains, in order, the user index i, movie index j, and the rating a_{ij}. Indexing starts at 1 (rather than 0). If you are interested in the raw Netflix data (including movie names, but no user names!) you can download http://www.edmondchow.com/files/nf_prize_dataset.tar.gz (698 MB). This data set is actually not very large as "big data" problems go, as it can fit on a single compute node.

Your program should also input a second file, which is located at http://www.edmondchow.com/files/netflix_query (125 bytes). Its contents are

```
427501          8
260749         18
311872         28
 73318         30
182071         30
```

Your program should output the predicted ratings for these five pairs of user and movie indices, i.e., rating of user 427501 on movie 8, etc.

For this assignment, write a program to compute rating predictions. Use the matrix factorization approach discussed in class, which uses a parallel gradient descent algorithm. The approach is composed of two parts. The first part computes vectors p and q and is a computationally expensive part that must be parallelized. The second part uses the vectors computed in the first part to compute the ratings. This second part is very inexpensive unless the query file is very large. For this assignment, you only need to parallelize the first part.

In the algorithm, you may need to experiment with different values of the regularization parameter λ and the learning rate γ. To start, you can try $\lambda = 0.005$ and $\gamma = 0.00125$. Choose the number of features $f = 50$. Use a reasonable way to choose choose the initial guess for the vectors p and q.

To assist in understanding how your program is progressing, you should print a message at each iteration. You could print the value of the least squares objective function, although calculating this quantity could be expensive. You should also decide when the iterations have converged and when you have reached an answer. These are all challenging but important questions.

(a) **10 marks.** Write a sequential version of the rating prediction code. This helps sort out any non-parallel computing issues first. This version should be reasonably well optimized and should be used as the baseline for your parallel version. Explain how you chose the initial guess for the vectors p and q.

(b) **10 marks.** Present the design of your parallel algorithm. Discuss how the ratings data is partitioned among MPI processes. Also discuss how the vectors p and q are partitioned. A good parallelization, for example, is one that allows updates to be performed independently. You might use a "2D" partitioning for this.

(c) **10 marks.** Implement your algorithm. First test your algorithm using smaller data sets that you've created, since it may be a little awkward to use large files for initial testing. Convergence may also be poor for very large problems.

(d) **10 marks.** Present timings and other results that illustrate the performance of your parallel program, e.g., different numbers of nodes. Discuss the parallel performance of your code, and justify the speedup values you achieved. What improvements can be made to make your code even faster in parallel?