CITS3402 High Performance Computing Assignment 3

Sparse matrix-matrix multiplication using MPI and OpenMP

The total marks for this assignment is 20, the assignment can be done in groups of two, or individually.

The matrix-matrixi multiplication algorithm using MPI has been discussed thoroughly in the lecture slides. The task in this project is to implement the same algorithm using a two-level parallelization, MPI at the higher level and OpenMP at a lower level. Hence you have to introduce OpenMP directives for execution at each node.

Sparse matrices are common in many scientific applications. A sparse matrix is usually a very large matrix with only a small fraction of its entries non-zero, and most entries are zero. For example, let us assume we have a 1000X1000 (a matrix with one thousand rows and one thousand columns) matrix. In a sparse matrix, the typical percentage of non-zero elements is about 10% or less. Hence the number of non-zero elements in our example matrix could be 100,000 out of 10,000,000 possible entries.

As can be easily seen, storing a sparse matrix as a regular matrix is very wasteful, and also will incur a huge communication cost if such sparse matrices are sent between nodes in a MPI program. Hence the input to our program will be sparse matrices.

There are many different formats for representing sparse matrices, one of the simplest is the *matrix market* format. Every line of a file in this format has three entries, the row number, the column number and the entry at that row and column position. For example

```
0  0  0
0  34  0
0  0  0
```

This matrix can be stored as just one line in the matrix market format, namely : 2, 2, 34
There is no need to store all the other entries as they are all zero.

Tasks: Your task is to implement the matrix-matrix multiplication algorithm using MPI and OpenMP taking two sparse matrices in the matrix market representation as the input.

- Implement a correct matrix-matrix multiplication algorithm using MPI and OpenMP. The master node should distribute the matrices to the other nodes. - 10 marks
- Thorough performance analysis with different number of nodes and different number of cores in each node (at least three configurations for number of nodes and three choices of number of cores for each of the three configurations, in total 9 experiments). - 5 marks
- A report presenting your algorithms and performance analysis. The report must have your UWA official name and number as recorded in Calista. I will not record marks for wrong names and student numbers. The report must be in pdf format. - 5 marks

Data:

- a small 10X10 sparse matrix that you can use to test your code. You should write a sequential program to verify that your code is correct.
- an 1138X1138 large sparse matrix and another 1138X1138 large sparse matrix. You should use these two matrices for performance evaluation of your parallel code.

Some hints: You can sort the matrix files according to the first column by using this shell command in linux:

```
sort -k1 -n 1138_bus.mtx > out
```

`k1` means sort based on the first column, `n` means according to numerical values (not lexicographic order), `>out` will store the output in a file called `out`. One can sort according to the second column by changing `k1` to `k2`.

Deadline: The submission deadline is 11:59 pm on November 2, through cssubmit. This project carries 15% of the total marks for this unit.

Amitava Datta
October 2018
