Assignment of The A*B matrix:

I have written two version of it, the first one is the basic one and the second one is the advanced one.

The basic one is we have N cores and calculate the result of A*B, using “cout” command to see the result. The main point is on the transform and reception through different cores.

The result when using 10 cores calculation N=10:

```
3400 3645 3760 3745 3600 3325 2920 2385 1720 925
8100 8640 8880 8820 8460 7800 6840 5580 4020 2160
14100 14985 15360 15225 14580 13425 11760 9585 6900 3705
21400 22680 23200 22960 21960 20200 17680 14400 10360 5560
30000 31725 32400 32025 30600 28125 24600 20025 14400 7725
39900 42120 42960 42420 40500 37200 32520 26460 19020 10200
51100 53865 54880 54145 51660 47425 41440 33705 24220 12985
63600 66960 68160 67200 64080 58800 51360 41760 30000 16080
77400 81405 82800 81585 77760 71325 62280 50625 36360 19485
92500 97200 98800 97300 92700 85000 74200 60300 43300 23200
~
~
~
```

The advanced one is using determined like 4 cores to calculate whatever N is.

The key step is using “mod” to determined how many rows each cores shall calculate, using these:

```
int caltimes,fz,kk,Acol;//Acol is Arow, wrong name! XD

// know the calculation times of this core:
caltimes = 0;
caltimes = N / size;
if ((N % size - 1) >= rank) { caltimes += 1;  }
```

And the communication between core 0 and other cores, using:

```
    // other core send rowAB
    if (rank != 0) {
        MPI_Send(&(rowAB[0]), N, MPI_INT, 0, Acol, MPI_COMM_WORLD);
    }
```

And core 0 receive:

```
if ((i) % size != 0) {      // receive from others but not himself part.
    MPI_Recv(&(rowAB[0]), N, MPI_INT, (i) % size, i, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    //cout <<"Receive col"<< i <<" from "<<i%size<<" FN "<< rowAB[0] << " "<<endl;
```

The test result of this code, when 4cores, N=10:

```
3400 3645 3760 3745 3600 3325 2920 2385 1720 925
8100 8640 8880 8820 8460 7800 6840 5580 4020 2160
14100 14985 15360 15225 14580 13425 11760 9585 6900 3705
21400 22680 23200 22960 21960 20200 17680 14400 10360 5560
30000 31725 32400 32025 30600 28125 24600 20025 14400 7725
39900 42120 42960 42420 40500 37200 32520 26460 19020 10200
51100 53865 54880 54145 51660 47425 41440 33705 24220 12985
63600 66960 68160 67200 64080 58800 51360 41760 30000 16080
77400 81405 82800 81585 77760 71325 62280 50625 36360 19485
92500 97200 98800 97300 92700 85000 74200 60300 43300 23200
~
~
```

Same as the basic version result.

The test result of this code, when 4cores, N=3, case: N less than amount of cores.

```
75 68 43
204 184 116
387 348 219
~
~
```

This is also correct.

The code has satisfied what is expected.

Thank you! Love your course.

Xingyuan Chen

S2016500@ed.ac.uk