Parallel implementations of matrix multiplication

This is the second mandatory assignment of INF3380. Each student should work independently and submit her/his parallel programs to the group teacher before the deadline.

Matrix multiplication

As described in Chapter 11.2 of the textbook, *Michael J. Quinn*, **Parallel Programming in C with MPI and OpenMP**, the product between matrix A (of dimension $l \times m$) and matrix B (of dimension $m \times n$) will be a matrix C (of dimension $l \times n$), whose elements are defined by

$$C_{i,j} = \sum_{k=0}^{m-1} a_{i,k} b_{k,j}, \qquad \text{for } 0 \leq i \leq l-1, \quad 0 \leq j \leq n-1.$$

Remark. Students who have trouble understanding the above mathematical formula should familiarize with simple examples of matrix multiplications, for example, given on the following webpage:

`http://www.intmath.com/matrices-determinants/4-multiplying-matrices.php`

Task 1: MPI implementation

The student can choose either the rowwise block-striped parallel algorithm described in Chapter 11.3, or the checkerboard Cannon's algorithm described in Chapter 11.4. An MPI program should be implemented such that it can

- accept two file names at run-time,

- let process 0 read from file the A and B matrices,
- let process 0 distribute the pieces of A and B to all the other processes,
- involve all the processes to carry out the the chosen parallel algorithm for matrix multiplication $C = A * B$,
- let process 0 gather, from all the other processes, the different pieces of C,
- let process 0 write out the entire C matrix to a file.

Task 2: OpenMP-MPI implementation

The student should extend her/his MPI program from Task 1, so that OpenMP is used within each MPI process for the computation-intensive parts.

Input of matrix

For the sake of I/O efficiency, it is assumed that the A and B matrices are stored in binary formatted data files. More specifically, the following function can be used to read in a matrix stored in a binary file:

```
void read_matrix_binaryformat (char* filename, double*** matrix,
                               int* num_rows, int* num_cols)
{
  int i;
  FILE* fp = fopen (filename,"rb");
  fread (num_rows, sizeof(int), 1, fp);
  fread (num_cols, sizeof(int), 1, fp);

  /* storage allocation of the matrix */
  *matrix = (double**)malloc((*num_rows)*sizeof(double*));
  (*matrix)[0] = (double*)malloc((*num_rows)*(*num_cols)*sizeof(double));
  for (i=1; i<(*num_rows); i++)
    (*matrix)[i] = (*matrix)[i-1]+(*num_cols);

  /* read in the entire matrix */
  fread ((*matrix)[0], sizeof(double), (*num_rows)*(*num_cols), fp);
  fclose (fp);
}
```

For example, suppose the following three variables are declared:

```
double **matrix;
int num_rows;
int num_cols;
```

Then, a matrix stored in file `mat.bin` can be read in by calling `read_matrix_binaryformat` as follows:

```
read_matrix_binaryformat ("mat.bin'', &matrix, &num_rows, &num_cols);
```

Output of matrix

Similarly, the multiplication result matrix C should be written to file in binary format by using the following function:

```
void write_matrix_binaryformat (char* filename, double** matrix,
                                int num_rows, int num_cols)
{
  FILE *fp = fopen (filename,"wb");
  fwrite (&num_rows, sizeof(int), 1, fp);
  fwrite (&num_cols, sizeof(int), 1, fp);
  fwrite (matrix[0], sizeof(double), num_rows*num_cols, fp);
  fclose (fp);
}
```

Examples of A and B matrices

From the website of INF3380, the following matrices (in binary format) can be downloaded for code debugging and testing:

`small_matrix_A.bin`	of dimension 100×50
`small_matrix_B.bin`	of dimension 50×100
`large_matrix_A.bin`	of dimension 1000×500
`large_matrix_B.bin`	of dimension 500×1000