Lab 4
Parallel and Distributed Algorithms

Dense matrix multiplication

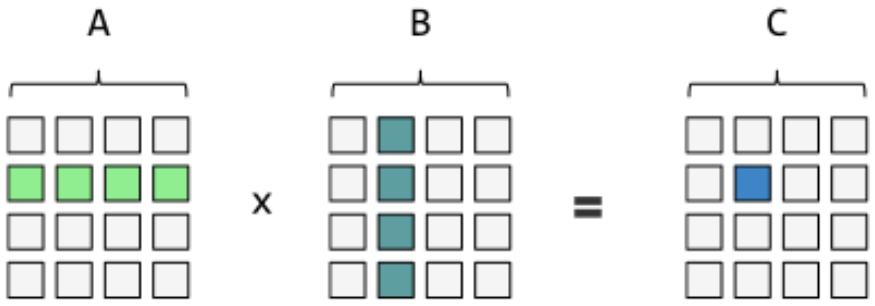
Author: Adam Brzeski
Date: 21.04.2016

1 Aims of the laboratory

The aim is to practice parallel and distributed programming skills on a problem of matrix multiplication problem, which is a common task in HPC (High Performance Computing). The algorithm is to be implemented both in shared memory model and in distributed model. The implementation will utilize the provided java framework for simulating distributed system, which was already used in the previous classes.

The code for this lab is in Lab04.zip file. Extracted folder contain a project for Netbeans IDE. The source can be however easily imported to other IDEs (Idea, Eclipse).

2 Reminder - matrix multiplication




```
C[i][j] = sum(A[i][k] * B[k][j]) for k = 0 ... n
```

```
In our case:
C[1][1] =>
A[1][0]*B[0][1] + A[1][1]*B[1][1] + A[1][2]*B[2][1] + A[1][3]*B[3][1]
```

Figure 1: Matrix multiplication operation example.
Source: http://www.stoimen.com/blog/2012/11/26/computer-algorithms-strassens-matrix-multiplication/

3 Shared memory model

The basic approach for parallel matrix multiplication using multiple threads is to partition the resulting C matrix into blocks (sub-matrices), each to be evaluated by different thread. Each of the threads has access to full A and B matrices and write to it's own block of C matrix, so the computations are performed independently.

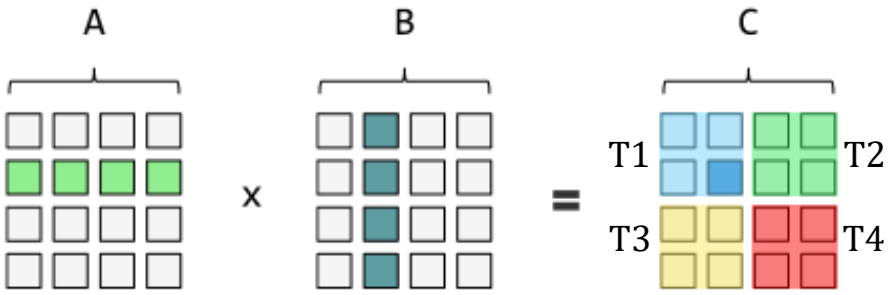


Figure 2: Matrix multiplication in shared memory model example. 4x4 C matrix is divided into four 2x2 blocks and assigned to 4 threads. Each of the threads computes its sub-matrix independently.
Source: http://www.stoimen.com/blog/2012/11/26/computer-algorithms-strassens-matrix-multiplication/

4 Distributed memory model – Cannon's algorithm

In distributed memory model the basic approach also assumes partitioning of C matrix into blocks, which are assigned to different nodes. In this case, however, A and B matrices are not stored in one memory, but are scattered throughout the nodes.

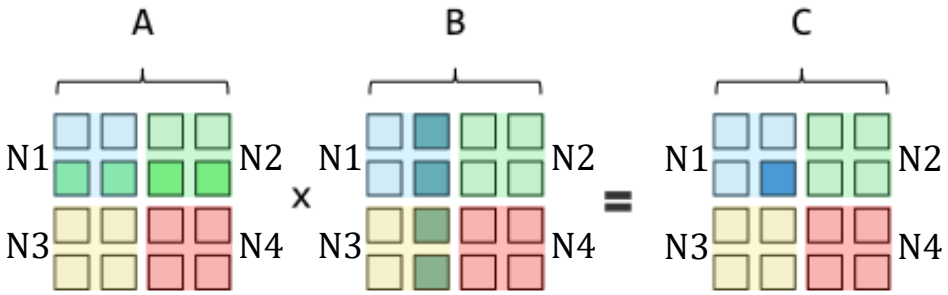


Figure 3: Matrix multiplication in distributed memory model example. 4x4 C matrix is divided into four 2x2 blocks and assigned to 4 nodes. Each of the nodes also stores the proper block of A and B matrices.
Source: http://www.stoimen.com/blog/2012/11/26/computer-algorithms-strassens-matrix-multiplication/

The multiplication of matrices divided into blocks has a property allowing to compute the result using exactly the same formula as before, but operating on matrices instead of scalars. Therefore, in the above example C[N1] sub-matrix can be computed as:

```
C[N1] = A[N1] × B[N1] + A[N2] × B[N3]
```

As can be in the above formula, before computing their C sub-matrices, the nodes need to exchange their A and B sub-matrices they store in order to acquire the full row and column of A and B matrices, required for the given sub-matrix of C.

A simple algorithm for distributed matrix multiplication with block partitioning is described at http://parallelcomp.uw.hu/ch08lev1sec2.html. Another presented method is Cannon's algorithm, which is a memory efficient version of the simple algorithm.

5 Student's task

The tasks are evaluated with 5 tests provided in the testAll() method of labs.Lab04 class. The tests need to be uncommented along with completing subsequent steps. The tasks to complete are:

1. **Implement shared memory matrix multiplication – 2 pts**
   - requires implementing the following method of the algorithms.shared.MatrixOperationsSM class:

     multiplyMtx(Matrix a, Matrix b, int nThreadsPerRow, int nThreadsPerCol)
   - requires passing Tests 1 and 2

2. **Implement distributed memory matrix multiplication using the simple algorithm or Cannon's algorithm – 6 pts**
   - requires implementing the following method of the algorithms.distributed.MatrixOperations class:

     multiplySquareMtxSimple(Node node)
   - requires passing Tests 3-5