Technische Universität München
Chair of Scientific Computing
Dr. Ralf-Peter Mundani
Dipl.Inf. Atanas Atanasov

SS 2010

Parallel Programming and HPC
Exercise Sheet 6: Programming with MPI

7th July 2010

1 Matrix-Matrix Multiplication

1.1 Definition

The product of two rectangular matrices is only well-defined if their dimensionalities match certain criteria:

$$AB = C = \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mk} \end{pmatrix} \times \begin{pmatrix} b_{11} & \dots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{k1} & \dots & b_{kn} \end{pmatrix} = \begin{pmatrix} c_{11} & \dots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \dots & c_{mn} \end{pmatrix}$$

Each element c_{ij} of the product matrix C can be calculated as the scalar product $c_{ij} = \vec{a_i}^T \cdot \vec{b_j}$ with the row vector $\vec{a_i} = (a_{i1}, a_{i2}, \dots, a_{in})$ and column vector $\vec{b_j} = (a_{1j}, a_{2j}, \dots, a_{nj})$.

1.2 Parallelization Strategy

Assume there are 2 processes, p_1 and p_2, available to multiply two matrices $A, B \in \mathbb{R}^{4\times 4}$. Firstly, to execute the multiplication in parallel, the data has to be split up into two equal parts. p_1 holds the first two rows of A as well as the first two columns of B in memory. Accordingly, p_2 stores row 3 and 4 of A and column 3 and 4 of B. The resulting matrix $C \in \mathbb{R}^{4\times 4}$ can be computed in 2 steps (see figure 1). Obviously, if the data is initially distributed as described above, there has to be some communication between step 1 and step 2. The two pairs of rows have to be exchanged in order to continue the execution. In MPI, the appropriate routines would be `MPI_Send` and `MPI_Recv`.

However, the data exchange must be possible for an arbitrary number of processes n_p. A communication pattern which can be applied here is the so-called *round robin* strategy, which is depicted in figure 2. In each step, each process shifts its columns to the following process and receives new columns from the precedent process. After n_p steps, each process has received each part of A exactly once and has computed its resulting columns of C independently of the other processes.

$$A_{p_1}B_{p_1} = C_{p_1} = \begin{pmatrix} {\color{red}a_{11}} & {\color{red}a_{12}} & {\color{red}a_{13}} & {\color{red}a_{14}} \\ {\color{red}a_{21}} & {\color{red}a_{22}} & {\color{red}a_{23}} & {\color{red}a_{24}} \\ {\color{green}a_{31}} & {\color{green}a_{32}} & {\color{green}a_{33}} & {\color{green}a_{34}} \\ {\color{green}a_{41}} & {\color{green}a_{42}} & {\color{green}a_{43}} & {\color{green}a_{44}} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & & \\ b_{21} & b_{22} & & \\ b_{31} & b_{32} & & \\ b_{41} & b_{42} & & \end{pmatrix} = \begin{pmatrix} {\color{red}b_{11}} & {\color{red}b_{12}} & & \\ {\color{red}b_{21}} & {\color{red}b_{22}} & & \\ {\color{green}b_{31}} & {\color{green}b_{32}} & & \\ {\color{green}b_{41}} & {\color{green}b_{42}} & & \end{pmatrix}$$

$$A_{p_2}B_{p_2} = C_{p_2} = \begin{pmatrix} {\color{green}a_{11}} & {\color{green}a_{12}} & {\color{green}a_{13}} & {\color{green}a_{14}} \\ {\color{green}a_{21}} & {\color{green}a_{22}} & {\color{green}a_{23}} & {\color{green}a_{24}} \\ {\color{red}a_{31}} & {\color{red}a_{32}} & {\color{red}a_{33}} & {\color{red}a_{34}} \\ {\color{red}a_{41}} & {\color{red}a_{42}} & {\color{red}a_{43}} & {\color{red}a_{44}} \end{pmatrix} \times \begin{pmatrix} & & b_{13} & b_{14} \\ & & b_{23} & b_{24} \\ & & b_{33} & b_{34} \\ & & b_{43} & b_{44} \end{pmatrix} = \begin{pmatrix} & & {\color{green}b_{13}} & {\color{green}b_{14}} \\ & & {\color{green}b_{23}} & {\color{green}b_{24}} \\ & & {\color{red}b_{33}} & {\color{red}b_{34}} \\ & & {\color{red}b_{43}} & {\color{red}b_{44}} \end{pmatrix}$$

Figure 1: Parallel matrix multiplication in two steps: step 1 and step 2 by processes p_1 and p_2

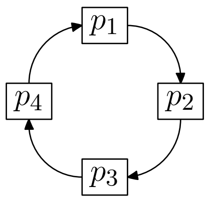

Figure 2: The round robin strategy for $n_p = 4$ processes

1.3 Assignment

a) Write a C program, using MPI for message passing, which performs the following steps:

   1. Read two $n \times n$ matrices of type *int* from one/two text file(s)
   2. Distribute the data according to the communication pattern
   3. Compute the matrix product of the two matrices as described in 1.2
   4. Collect the resulting matrix and store it to a text file

b) Multiply matrices of different sizes, e.g. 250×250, 500×500, 1000×1000 and 2000×2000, and perform the computations on different numbers of processors, e.g. 1, 2, 4, ... Measure the run time. Draw speedup- and efficiency curves. Interpret your results with respect to Amdahl's and Gustafson's Law.

2 Load balancing in static network topologies

Different strategies are known for the fair distribution of the load by the execution of parallel applications on n nodes. One of this strategies is the diffusion model. According to this model the load $w_i^{(t)}$ of process P_i for the t-th iteration is defined as:

$$w_i^{(t+1)} = w_i^{(t)} - \sum_{j \in N(i)} a_{ij} * (w_i^{(t)} - w_j^{(t)}),$$

with $1 \le i \le n$, $-1 < a_{ij} < 1$ and $N(i)$ the neighbors of this process. The parameter a_{ij} represents the rate of load, which should be exchanged between P_i and P_j. There are different

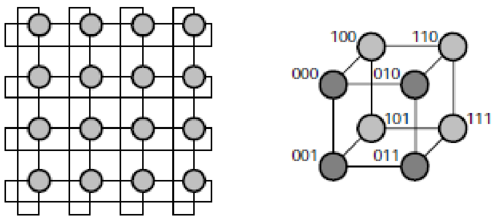

Figure 3: 4-2 Torus and 3-dimensional Hypercube

methods to determine appropriate value for a_{ij}, which will not be discussed any further. In this exercise the method should be used on two types of static networks: the $k - d$network of dimension d with k nodes in each direction and d-dimensional Hypercube-network. The following pictures show 4-2 cyclic-network on the left and 3D-hypercube on the right.

2.1 Assignment

Write a C/C++ program, which computes the load by the means of the diffusion method for 8-3 network without cycle and for the 9-dimensional Hypercube. All nodes have load $w_i^(0) = 0$ at the beginning. Add the load $w_k(0) = 1000$ to arbitrary node k and measure the number of iterations to distribute the load! What effect does the choice of the node have on the number of iterations? Make the measurements for $a_{ij} = 0.15$, $a_{ij} = 0.25$ and $a_{ij} = 0.35$! Plot your results!