Technische Universität München
Institut für Informatik
Dr. rer. nat. Tobias Neckel
Matous Sedlacek

WT 2012/2013

Parallel Numerics

Exercise 4: Matrix–Matrix Operations & P2P Communication III

1) DAGs for Function Evaluation

Given is $f(x_1, x_2, x_3, x_4) = x_1^2 + x_2x_3 - x_1x_3 - x_1x_2 = x_2x_3 + x_1(x_1 - x_3 - x_2)$.
Give the DAGs for both evaluation schemes in parallel and compare the number of required time steps.

2) Amdahl's Law

Given is an algorithm working on a fixed amount of data. f is the fraction of a calculation that is sequential and, thus, cannot benefit from parallelisation. The algorithm has execution time t_1.

i) What is the execution time t_p, if exactly this problem is ported to a parallel machine with p processors.

ii) What is the speedup?

iii) What is the efficiency?

iv) What happens if the number of processors is increased ($\lim_{p \mapsto \infty}$)?

3) Gustafson's Law (Gustafson-Barsis' Law)

You are given a parallel algorithm running on p nodes that needs the time t_p to finish. Again, f is the fraction of a calculation that is sequential and, thus, cannot benefit from parallelisation.

i) Derive the time t_1, one processor would need to solve this problem.

ii) What is the speedup?

iii) What is the efficiency?

iv) What happens, if the number of processors is increased ($\lim_{p \mapsto \infty}$)?

v) Compare these formulas to Amdahl's law. In today's problems, often not the execution but the pure amount of data is the limiting factor for computations. Thus, an algorithm

is parallelised because one wants to compute bigger problems, and a speedup is only a (nice) side–effect. What speedup law is the appropriate one for this scenario?

4) P2P Communication III: Synchronised Operations

Given is the following pseudocode fragment:

```
# node 0:
int a = 0;
send a to node 1
a = 1;
send a to node 1

#node 1:
int b;
receive b from node 0
printf("%i", b);
receive b from node 0
printf("%i", b);
```

i) Implement the send and receive operations using blocking and non–blocking operations. What might be the difference in the results?

ii) Define the MPI term *synchronised*.

iii) Implement the send and receive operations using synchronised non–blocking operations. Insert a wait operation directly after the next integer assignment on node 0. What might be the difference in the results?

iv) Implement the send and receive operations using synchronised blocking operations. What might be the drawback?

v) Make yourself familiar with the MPI terms *buffered send* and *ready send*.

5) Matrix–Matrix Multiplication

The product $A \cdot B$ of two $N \times N$-matrices A and B is calculated as follows:

$$C = A \cdot B \qquad \text{where} \qquad (C)_{ij} = \sum_{k=1}^{N} (A)_{ik}(B)_{kj}.$$

A program that calculates this product has to use three independent loops (over i, j, k). The execution speed of the program depends on the arrangement of these loops.

i) Give a scetch of the data dependency graph for one element of this problem.

ii) Assume N is a multiple of $\sqrt{p}$. All the three matrices are split up into p quadratic submatrices. Every node has to compute one submatrix of the result matrix C. Illustrate this fact!

iii) Pick out three different nodes. What parts of A and B are required by them to compute the result?

iv) Have a look at the node computing the upper left submatrix of C. Interprete the multiplication formular $(C)_{ij} = \sum_{k=1}^{\sqrt{p}} (A)_{ik}(B)_{kj}$ block–wise. Let's assume every summand of the result is computed in one time step. What different parts of A and B are required subsequently? How many time steps are required?

v) Every node is allowed to communicate with the node that computes the left, right, top or bottom submatrix of C only. Assume this cartesian topology is cyclic. Furthermore, every node is allowed to hold only one submatrix of A and B at one time. Derive a communication scheme.