TDDC78 Programming of Parallel Computers — Methods and Tools

Prof. C. Kessler, IDA, Linköping University, 2011

Optional Theory Exercises in Parallel Computing

These are optional exercises for self-assessment about some of the theory parts of our course.

No supervision, no submission, no correction, no points.

1 Parallel Computer Architecture Concepts

Exercise I.1

Show that a Butterfly-network connecting 2^k inputs to 2^k outputs uses $k2^{k-1}$ switches, for all $k \geq 1$.

Exercise I.2

Give formulas (depending on the number of processors) for the node degree and the diameter for the following network topologies:

- Bus
- Ring
- 2D-grid
- 2D-torus
- 3D-grid
- 3D-torus,
- complete binary tree (processors as tree nodes),
- complete binary tree (processors as leaves, switches as inner nodes)
- hypercube,
- butterfly
- crossbar.

What about node-connectivity and edge-connectivity for these topologies?

Exercise I.3

(a) Give an example for communication with indirect addressing.

(b) What are the advantages of a blocking send() operation?

(c) What are the advantages of asynchronous communication (non-blocking send() operation)?

(d) What are the advantages of rendezvous communication (blocking send and blocking receive)? What are the disadvantages?

(e) What are the advantages and disadvantages of buffered communication?

Exercise I.4

Consider a simple uniprocessor system with no caches. How does register allocation (applied by the compiler) affect memory consistency? Which language feature of C allows you to enforce sequential consistency for a variable?

Exercise I.5

A parallel program for a cache-based SMP contains a (asynchronous) loop with n iterations where processor P_1 reads shared variable x in each iteration and processor P_2 writes shared variable y in each iteration. Assume that x and y happen to be placed in the same memory block, so they will end up in the same cache line. How many messages are to be sent if sequential consistency is to be enforced?

Exercise I.6

Consider a superscalar RISC processor running at 1 GHz. Assume that the average CPI (clock cycles per instruction) is 1. Assume that 15% of all instructions are stores, and that each store writes 8 bytes of data. How many processors will a 2.5 GB/s bus be able to support without becoming saturated?

2 Design and analysis of parallel algorithms

Exercise II.1

Give high-level CREW and EREW PRAM algorithms for copying the value of memory location $M[1]$ to memory locations $M[2],...,M[n+1]$.

Exercise II.2

On a RAM the maximum element in an array of n real numbers can be found in $O(n)$ time. We assume for simplicity that all n elements are pairwise different.

(a) Give an EREW PRAM algorithm that finds the maximum element in time $\Theta(\log n)$. How many processors do you need at least to achieve this time bound?

What is the work and the cost of this algorithm? Is this algorithm cost-effective with n processors? With $n/\log n$ processors?

(b) Give an algorithm for a Common CRCW PRAM with n^2 processors that computes the maximum element in constant time.

(*Hint:* Arrange the processors conceptually as a $n \times n$ grid to compare all n^2 comparisons of pairs of elements simultaneously. An element that is smaller in such a comparison cannot be the maximum. Use the concurrent write feature to update the entries in an auxiliary boolean array m of size n appropriately, such that finally holds $m[i] = 1$ iff array element i is the maximum element. Given m, the maximum location i can be determined in parallel using n processors.)

What is the work and the cost of this algorithm? Is this algorithm cost-effective?

Further reading on the maximum problem:

Any CREW PRAM algorithm for the maximum of n elements takes $\Omega(\log n)$ time.
See [Cook/Dwork/Reischuk *SIAM J. Comput.* 1986]

There exist CRCW PRAM algorithms for n processors that take $O(\log\log n)$ time.
See [Valiant *SIAM J. Comput.* 1975, Shiloach/Vishkin *J. Algorithms* 1981]

Exercise II.3

Show that the cost of a cost-optimal parallel algorithm A is of the same order of magnitude as the work of the optimal sequential algorithm S: $c_A(n) = \Theta(t_S(n))$.

Exercise II.4

Give a $O(\log n)$ time algorithm for computing parallel prefix sums on a parallel list. (Hint: Use the pointer doubling technique.)

Exercise II.5

A program contains a function m that cannot be parallelized and that accounts for 40% of the execution time. What is the speedup limit for this program on a p-processor machine?

Exercise II.6

A program contains a function m' that that accounts for x % of the execution time. What should be the speedup of m' in order to speed up the overall program by a factor of 2?

Exercise II.7

A program contains a function m'' that can be parallelized to achieve speedup 3. What fraction of the overall execution time must m'' account for in order to double the overall speedup of the

program?

Exercise II.8

Given a message passing parallel computer whose interconnection network is a d-dimensional hypercube. Design a deterministic parallel algorithm (using send and receive operations) for broadcasting a block of data from a single source node s to all other nodes. Using the delay model or the LogP model, what is the worst-case execution time of your algorithm?

Exercise II.9

Derive the speedup for cascade summation on p processors on a ring.

3 MPI

Exercise III.1

Consider the finite–differences MPI program from the MPI lecture.

Why is it important to use *different* tags (10, 20) in the exchange phase?

Exercise III.2

Consider the finite–differences MPI program from the MPI lecture.

How could this program be improved in efficiency?

(*Hint:* try to overlap communication phases with local computation where possible.)

Exercise III.3

Consider the finite–differences MPI program from the MPI lecture.

Try to construct a deadlock situation by reordering the `send` / `recv` operations.

Exercise III.4

Write a MPI program for p processes, using send and receive operations only. Each process contributes in its local memory a block of data of statically unknown size $S_i > 0$, where the sizes $S_0, \ldots, S_{p-1}$ may differ. The p blocks should be concatenated in a single array of size $\sum_{i=0}^{p-1} S_i$, without holes in between, to be stored in the local memory of process P_0.

(Remark: This is a special case of the `MPI_Gatherv` function.)

4 Automatic Parallelization

Exercise IV.1

Automatic parallelization of sequential code into multi-threaded code requires the compiler to perform a data dependence analysis and to estimate whether or not it is worthwhile to parallelize the code. Give an example of a loop that can be safely auto-parallelized with high speedup and one example where auto-parallelization is difficult or not beneficial.

5 Parallel Linear Algebra Algorithms

Exercise V.1

Both LINPACK and LAPACK uses a standard for basic linear algebra subroutines, called BLAS. Describe the three different levels BLAS1, BLAS2, and BLAS3 in terms of the amount of data, the arithmetic complexity and the number of nested loops.

6 Data-Parallel Programming

Exercise VI.1

Describe the basic principles of parallelization, synchronization, and communication in a data parallel programming language such as Fortran 90.

(To be continued...)