Linköpings universitet
IDA Department of Computer and Information Sciences
Doc. Christoph Kessler

EXAM

FDA125 / TDDC85
Advanced Parallel Programming: Models, Languages, Algorithms

2007-03-30, 14:00–18:00

Name: _____ **Personnummer:** _____

Please mark solved problems with X:

Question	1	2	3	4	5	6	7	8	9
answered (X)									
score (points)									

Total score: _____ **Grade (U/3/4/5):** _____

Sign. Examinator: _____

Examinator: Christoph Kessler, Welf Löwe

Jour / tentavakt (Linköping): Christoph Kessler (0703-666687)

via telephone: Welf Löwe (0470-708495)

External supervisors for exams written outside Linköping,

please confirm with your signature below that you have checked the identity of the candidate and supervised this exam before you send it by mail to Christoph Kessler, IDA, Linköpings universitet, 58183 Linköping.

Supervisor (name, signature): _____ _____

Hjälpmedel / Admitted material:

None.

General instructions

- Read all assignments carefully and completely before you begin.
- It is recommended that you use a new sheet for each assignment. Number all your sheets, and mark each sheet on top with your name, personnummer, and a page number.
- You may answer in either English or Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- Motivate clearly all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 45 points; you should aim at solving for a total worth of about 40 points within 4 hours. You may thus plan about 5 minutes per point.

  The preliminary threshold for passing is 20 points.
- Grading: U, 3, 4, 5.

Assignments

1. (4 p.) **Theory of parallel computing**

   (a) Describe *Brent's Theorem* and give its interpretation. Explain all notation used. (2 p)

   (b) Motivate carefully why Brent's Theorem holds. (2 p)
   If possible, give a formal proof (+2p).

2. (6 p.) **PRAM Emulation**

   (a) Explain the main ideas of using hashing and a combining network to emulate a CRCW PRAM on a set of processors and memory modules. In particular, what happens for read and for write accesses to shared memory? (3 p)

   (b) Which properties should a combining network have in order to allow for scalability, efficient combining and short memory access latency? Give an example for a combining network and explain how it supports these requirements. (2 p)

   (c) Given a combining network for CRCW PRAM emulation of sufficiently high bandwidth. Which hardware technique could we use (and has been used, e.g., in the SB-PRAM) to make the emulation *cost-effective* if shared memory access latency takes longer than a local computation step by one or two orders of magnitude and caches are not an option? (Just give the technical term, no details.) (1 p)

3. (6 p.) **Parallel prefix sums**

   Recall: The prefix sums vector $ps = \langle ps_1, ..., ps_n \rangle$ of a vector $s = \langle s_1, ..., s_n \rangle$ is defined as

   $$ps_i = \sum_{j=1}^{i} s_j, \text{ for } i = 1, ..., n.$$

   Recall: A *parallel list* is an array of n list elements that are linked by `next` pointers in some unknown order.

   (a) Give a $O(\log n)$ time EREW PRAM algorithm (pseudocode and analysis of worst-case execution time) for computing parallel prefix sums on a parallel list. (Hint: Use the pointer doubling technique.) (4p)

   (If you do not recall how to handle parallel lists, do this assignment instead for prefix sums of an ordinary array, 3 p).

   (b) What is the *work* performed by your algorithm (as a function of n)? Is your algorithm work-optimal? (Explanation) (1p)

   (c) What is the *cost* function of your algorithm? Is your algorithm cost-optimal? (Explanation) (1p)

4. (5 p.) **Distributed shared memory**

   (a) What is *false sharing*? (2p)

   (b) Name and explain any relaxed memory consistency model of your choice. In how far is it more flexible than strict or sequential consistency? How could this improve performance of distributed shared memory computations? (3p)

5. (3 p.) **Transaction-based parallel programming**

   Explain the main idea of programming with *atomic transactions* (transactional memory).

   Why should one prefer this over monitor or lock-based solutions to achieve atomic execution of critical sections? (3 p)

6. (5 p.) **Run-time parallelization**

   (a) What type of loops cannot be parallelized statically? Give a small example with an explanation. (2p)

   (b) Sketch the main ideas of *speculative parallelization* (thread-level speculation). (2p)

   (c) What is the main difference between dynamic loop parallelization techniques (such as *doacross* and *inspector-executor*) and speculative loop parallelization? (1p)

Please make sure to solve the following exercises on a separate set of sheets; they will be mailed to Växjö for correction.

7. (8 p.) **Message passing models: BSP and LogP**

   (a) Describe the architecture of LogP machines. What do the parameters L, o, g, and P model?

   Why is it unrealistic in general to assume that L, o, g are constants - and under which assumption does it make sense again? (4p)

   (b) Define the notion of an h-relation in the BSP model.

   Describe the schedule of a statically known h-relation on a LogP machine. (4p)

8. (4 p.) **Data distribution**
   Programming languages allow annotating (shared, distributed) arrays with distribution functions defining the mapping of array elements to processors. Define two array element distribution strategies, which go beyond such functions and could even further reduce communication costs. Explain the potential benefit of your distribution strategies.

9. (4 p.) **Scheduling**
   Define an algorithm for scheduling send graphs for the delay model optimally. Sketch the proof for optimality.