



Försättsblad till skriftlig

tentamen vid Linköpings Universitet

(fylls i av ansvarig)

Datum för tentamen	2011 – 08 – 16
Sal	TER1
Tid	14:00 – 18:00
Kurskod	TDDC78
Provkod	TEN1
Kursnamn/benämning	Programmering av paralleldatorer – metoder och verktyg
Institution	IDA
Antal uppgifter som ingår i tentamen	8
Antal sidor på tentamen (inkl. försättsbladet)	5
Jour/Kursansvarig	Christoph Kessler, Henrik Branden
Telefon under skrivtid	Se tackbladet (sida 1) av tentan
Besöker salen ca kl.	Se tackbladet (sida 1) av tentan
Kursadministratör (namn + tfnr + mailadress)	Gunilla Mellheden, IDA, 2297, gunme @ ida.liu.se
Tillåtna hjälpmedel	Engelsk ordbok
Övrigt (exempel när resultat kan ses på webben, betygsgränser, visning, övriga salar tentan går i m.m.)	Se tackbladet (sida 1) av tentan

TENTAMEN / EXAM
TDDC78/TANA77
Programmering av paralleldatorer /
Programming of parallel computers
2011-08-16, 14:00–18:00, TER1

Christoph Kessler and Henrik Brandén
Dept. of Computer Science (IDA), Dept. of Mathematics (MAI)
Linköping University

Hjälpmedel / Allowed aids: Engelsk ordbok /
dictionary from English to your native language

Examinator: Christoph Kessler (TDDC78), Henrik Brandén (TANA77)

Jourhavande lärare:

Christoph Kessler (IDA) 013-28-2406; 0703-666687; visiting at ca. 16:00
Henrik Brandén (MAI) 013-28-5759; 0706-011969; visiting at ca. 16:00

Maximalt antal poäng / Max. #points: 40

Betyg / Grading (prel.):	MatNat		C, D, Y, DI	ECTS-graded students ^a
< 20	U	< 20	U	FX
20 – 30	G	20 – 27	3	C
31 – 40	VG	28 – 33	4	B
		34 – 40	5	A

^aSwedish grades will be automatically translated to the ECTS marks for exchange and international master program students as given above, according to a decision by the LiU rector in 2008.

General instructions

- Please use a new sheet for each assignment. Number all your sheets, and mark each sheet on top with your exam ID number and the course code.
- 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.

Note: In the case of questions regarding the assignments 4 to 5, please contact Henrik Brandén in the first hand; for questions regarding the other assignments, contact Christoph Kessler in the first hand.

1. (5 p.) Performance tools and analysis

- (a) (0.5p) Give one typical example for performance-related data about *message passing* programs that can be collected using *software counters*.
- (b) (1 p.) Which performance data collection method is required in order to be able to draw a processor-time diagram (also known as a *Gantt chart*)? Justify your answer (technical reasons).
- (c) (1p) There exist several possible causes for *speedup anomalies* in the performance behavior of parallel programs. Name and explain one of them.
- (d) (2.5 p.) Derive Gustafsson's law and give its interpretation. Explain how it differs from Amdahl's law, and for what kind of parallel computations it is more appropriate.

2. (2 p.) Describe the PRAM (parallel random access machine) computation model (choose any variant) of parallel computation, including its cost model. Which aspect of parallel computation is modeled well by PRAM, and which important property / properties of real parallel machines does it abstract from?

3. (4 p.) Parallel program design methodology

Foster's design methodology consists of four stages. Name and explain them. Give details about their goals. What are the important properties of the result of each stage? Be thorough!

4. The sum

$$s = \sum_{k=1}^p a_k$$

is to be computed using message passing on a distributed memory machine with p processors organized as a ring. A naive approach is to use the algorithm

```
do k=1,p-1
  transfer a(k) from node k to node k+1
  let node k+1 compute a(k+1) = a(k) + a(k+1)
end do
```

- (a) What is the speed-up S_p and efficiency E_p of this algorithm? (2p)
- (b) Describe the Cascade summation algorithm. (2p)
- (c) Show that the execution time of the cascade summation algorithm is

$$T_p = (p - 1)(\alpha + \beta) + \gamma \log_2 p$$

if p is a power of 2. (2p)

5. Consider matrix-matrix multiplication $C = AB$ on a

- (a) distributed memory machine using data parallel programming, (3p)
- (b) distributed memory machine using message passing, and on a (3p)
- (c) shared memory machine using shared memory programming. (3p)

For each case, explain one algorithm of your own choice, including the partitioning and distribution of data.

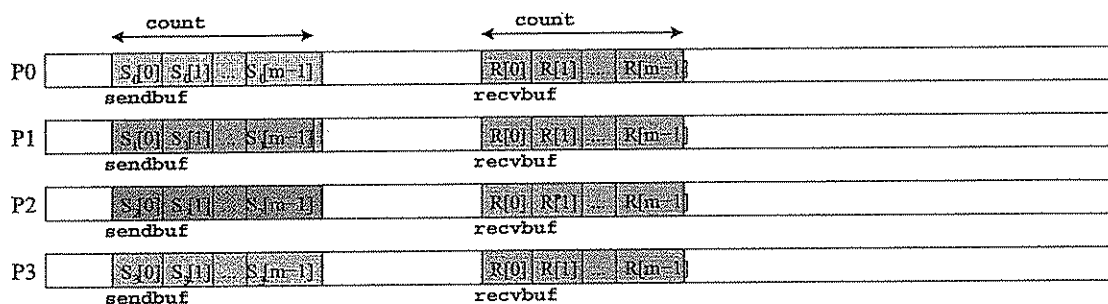


Figure 1: The MPI_Allreduce operation, here shown for $p = 4$ processes.

6. (6.5 p.) MPI

- (a) (5.5 p.) The MPI_Allreduce operation is a collective communication operation with the following parameters:

```
int MPI_Allreduce ( void *sendbuf, void *recvbuf,
                   int count, MPI_Datatype elemtype,
                   MPI_Op op, MPI_Comm comm );
```

where op is a pointer to a function combining two elements of type $elemtype$ into a single element of the same type. Examples of predefined combine functions in MPI are MPI_MAX (maximum), MPI_SUM (sum) or MPI_LAND (logical AND).

MPI_Allreduce works as follows: If executed by a group of p MPI processes, with each process i contributing a local array $s_i[0 : m - 1]$ in sendbuf with $count = m$ elements of type $elemtype$, for $i = 0, \dots, p - 1$, each of the p processes will afterwards hold, in its array recvbuf with m elements of same type $elemtype$, an array $r_i[0 : m - 1]$ that is the elementwise op -combination of the operand arrays s_0, \dots, s_{p-1} , i.e.,

$$r_i[j] = s_0[j] \text{ op } s_1[j] \text{ op } \dots \text{ op } s_{p-1}[j] \text{ for } j = 0, \dots, count - 1 \text{ and } i = 0, \dots, p - 1.$$

All processes of the group executing a call to MPI_Allreduce must pass in equal argument values for count, elemtype and op, respectively. A typical call could look as follows:

```
MPI_Allreduce ( S, R, m, MPI_FLOAT, MPI_SUM, MPI_COMM_WORLD );
```

- i. What is the difference in behavior to MPI_Reduce? (0.5p)
- ii. Write an implementation of MPI_Allreduce using only MPI_Send and MPI_Recv operations. Use C, Fortran, or equivalent pseudocode (explain your language constructs if you are not sure about the right syntax). Explain your code. (2.5p)
(Note: There exist several possibilities. An algorithm that is time-optimal for large p gives a 1p bonus, provided that the analysis is correct and the optimality is properly motivated.)
- iii. Show how your code behaves over time by drawing a processor-time diagram for $p = 4$ showing when messages are sent from which source to which destination, and what they contain. (0.5p)
- iv. Let m denote the value of count, and assume that the elemtype is MPI_FLOAT (normal floatingpoint numbers) and that any call to function op executes in time t_{op} , where $t_{op} \in O(1)$. Analyze asymptotically the (worst-case) parallel execution time and the parallel work of your implementation (for arbitrary values of p and m) as a function in p and m . You may use the delay model, the BSP model or the LogP / LogGP model for this purpose (state which model you use). If you need to make further assumptions, state them clearly. (2p)

Hint: Communication contributes to the work done by a parallel algorithm.

- (b) (1.5 p.) Explain the Communicator concept in MPI. How does it support the construction of parallel software components?

7. (2.5 p.) **OpenMP**

- (a) What kind of parallel loops are suitable candidates for dynamic scheduling? Why? (1p)
- (b) What is the purpose of the reduction clause in OpenMP parallel loops? Be thorough! (1.5p)

8. (5 p.) **Parallel computer architecture**

- (a) How is the *peak performance* of a parallel computer system defined? (0.5p)
- (b) The TOP-500 list of the 500 world-wide "most powerful" supercomputer installations is published twice a year. How is it calculated? I.e., how are these machines ranked, what kind of performance is measured for comparison, and how is it measured? And for what kind of workload (applications) could these measurements be representative? (1.5p)
- (c) Given an application with a fixed performance requirement (in GFlops). Why can, provided that the application can be parallelized (work-)efficiently, the transition from a single-core to a multicore execution platform be advantageous from a power efficiency point of view? (1p)
- (d) For a shared memory architecture, does sequential memory consistency imply a deterministic total ordering of all memory accesses? Justify your answer. (1p)
- (e) (1p) Give an example of an interconnection network topology (name and sketch) where the maximum distance between any two nodes grows logarithmically in the number of nodes.