TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer /
Programming of parallel computers

2013-01-11, 14:00–18:00, TER1

Christoph Kessler Dept. of Computer Science (IDA)
Linköping University

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

Examinator: Christoph Kessler

Jourhavande lärare:
Christoph Kessler (IDA) 013-28-2406; 0703-666687; visiting at ca. 16:00

Maximalt antal poäng / Max. #points: 40

Betyg / Grading (prel.): The preliminary threshold for passing (grade 3) is at 20p, for grade 4 at 28p, for grade 5 at 34p.
Because of new regulations by Linköping University, we can no longer give ECTS grades.

General instructions

- Please use a new sheet of paper for each assignment. Order your sheets by assignments, number them, and mark them on top with your exam ID number and the course code.
- You may answer in either English or Swedish. **English is preferred** because not all correcting assistants understand 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.

1. (5.5 p.) **Performance tools and analysis**

   (a) (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).

   (b) (1p) Modern tool-suites for performance analysis of parallel programs consist of a collection of several kinds of tools. Give four different kinds of such tools. *(I.e., no concrete tool names, but a short term saying what each kind of tool does.)*

   (c) (1p) When is a parallel algorithm for some problem (asymptotically) *work-optimal*? *(give a formal definition)*

   (d) (2 p.) (*i*) Derive Amdahl's law and (*ii*) give its interpretation.
   (Note: a picture is nice but is not a proof; a calculation is expected for the derivation of Amdahl's Law.)

   (e) (0.5 p.) How are the supercomputers on the TOP500 list ranked?

2. (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!

3. (6.5 p.) **Parallel computer architecture**

   (a) Many cache-based shared-memory systems use *bus-snooping*. What does that mean, what is its purpose, and how does it work? (1.5p)

   (b) What does 'hardware multithreading' mean?
   What is the main difference between a hardware-multithreaded processor and a multicore processor?
   For what kind of code (type of instructions) can hardware multithreading help to improve performance (throughput)? (2p)

   (c) (2 p.) What is 'false sharing'? In what kind of parallel computers and in what situations does it occur, and how does it affect performance? Suggest one possible way how the problem could be reduced.

   (d) Give an example of a scalable interconnection network topology (name and sketch) where the node degree is constant, independent of the number of nodes. What is the advantage of a constant node degree? (1p)

4. (3.5 p.) **OpenMP Example**

A novice OpenMP programmer has written the following erroneous program for calculating $\sum_{i=1}^{N}(1/i)$:

```
#include <omp.h>
#include <stdio.h>

#define N 100000000

double s = 0.0;

int main()
{
 int i;
#pragma omp parallel private(i)
 {
#pragma omp for schedule(static)
   for (i=1; i<=N; i++)
      s = s + 1.0/(double)(i);
 }
 printf("Sum: %lf\n", s );
 return 0;
}
```

(a) The programmer uses static scheduling of the parallel loop iterations. Explain how the iterations are mapped to the executing threads. (0.5p)

Taking the program as it is (ignoring the correctness problem below), is static scheduling of the loop appropriate here? Explain why or why not. (0.5p)

(b) The program prints an incorrect result if executed on more than one processor. Where is the bug, and what goes wrong? (1p)

(c) There are several possible fixes to this problem, which differ considerably in their effect on performance. Suggest a correct solution (code) that should lead to best achievable performance with multiple processors, and *explain* why.
(1.5p; a correct but suboptimal solution with explanation gives 0.5p)

5. (6 p.) A matrix-vector multiplication $y = Ax$, where A is $n \times n$, is to be computed using message passing on a distributed memory machine with p processors, organized as a ring. The matrix A and the vector x have been partitioned into row blocks A_i and X_i of size $n/p \times n/p$ and $n/p \times 1$ respectively,

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{pmatrix}, \qquad x = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix},$$

and distributed in such a way that processor i has A_i and X_i.

(a) Propose a suitable algorithm for computing y. (2p)

(b) Show that the execution time of this algorithm is roughly

$$T_p = \frac{2n^2}{p}\gamma + p\left(\alpha + \frac{n}{p}\beta\right)$$

where α is the message latency, β^{-1} is the data transfer speed, and γ is the time of one arithmetic operation. (2p)

(c) Assuming $n = p$, what is the (relative, asymptotic) efficiency of this algorithm? (2p)

6. (9 p.) Consider the following column version of the LU factorization algorithm:

```
do k=1,n-1
   a(k+1:n,k) = a(k+1:n,k)/a(k,k)
   do j=k+1
      a(k+1:n,j) = a(k+1:n,j) - a(k,j)*a(k+1:n,k)
   end do
end do
```

Transcription for C programmers:

```
for (k=0; k<n-1; k++) {
   for (i=k+1; i<n; i++)
      a[k][i] = a[k][i] / a[k][k];
   for (j=k+1; j<n; j++)
      for (i=k+1; i<n; i++)
         a[j][i] = a[j][i] - a[j][k] * a[k][i];
}
```

(a) Describe a parallel version of the algorithm, including the partitioning and distribution of data, for a distributed memory machine using data parallel programming. (3p)

(b) The following is an OpenMP parallelization of the algorithm for a shared memory machine.

```
   do k=1,n-1
      a(k+1:n,k) = a(k+1:n,k)/a(k,k)
      !$omp parallel do schedule (static)
      do j=k+1,n
         a(k+1:n,j) = a(k+1:n,j)-a(k,j)*a(k+1:n,k)
      end do
      !$omp end parallel do
   end do
```

Transcription for C programmers:

```
   for (k=0; k<n-1; k++) {
      for (i=k+1; i<n; i++)
         a[k][i] = a[k][i] / a[k][k];
#pragma omp parallel for schedule (static)
      for (j=k+1; j<n; j++)
         for (i=k+1; i<n; i++)
            a[j][i] = a[j][i] - a[j][k] * a[k][i];
   }
```

Explain the algorithm, including the partitioning of data. (3p)

(c) What kind of modification of the column LU algorithm and what kind of partitioning and distribution of data are required in order to get a scalable algorithm when using message passing on a distributed memory machine? (3p)

7. (4 p.) **MPI**

(a) (2 p.) MPI supports *nonblocking* (also called *incomplete*) point-to-point communication. What does that mean?

Give an example scenario demonstrating how using a nonblocking send (`MPI_Isend`) or receive (`MPI_Irecv`) routine instead of their blocking counterpart could speed up program execution. What kind of routine is necessary with nonblocking communication if we need to make sure that data dependences are preserved?

(b) (2 p.) *One-sided communication in MPI*

i. Explain the principle of one-sided communication in MPI-2. (1p)
Hint: You might want to illustrate your answer with a pseudocode example and/or an annotated picture.

ii. Why is one-sided communication considered being "closer" to the shared-memory programming model than ordinary two-sided message passing? (0.5p)

iii. How does one-sided communication still differ from the shared-memory programming model? (0.5p)

8. (1.5 p.) **OpenMP**

(a) What is the memory consistency model guaranteed by OpenMP implementations? (short answer) (0.5p)

(b) Why is the design of OpenMP helpful for *incremental* parallelization of sequential codes? (1p)