TENTAMEN / EXAM

TDDC78/TANA77

Programmering av parallelldatorer /
Programming of parallel computers

2012-08-14, 14:00–18:00, G32

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
Marco Kupiainen (MAI) 011-4958703. **In case of questions call before 16:30.**

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.

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

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

   (a) (0.5p) Modern processors provide several built-in *hardware counters* that allow to collect certain kinds of performance data. Give one typical example for performance-related information (about sequential code) that can be obtained from such counters.

   (b) (0.5p) Give one typical example for performance-related data about *message passing* programs that can be collected using *software counters*.

   (c) (1p) Compare performance data collection with hardware counters vs. software counters. What has, in each case, to be done with the program (technically) to enable this data collection?

   What is the main advantage of using hardware counters (where applicable) compared to software counters?

   (d) (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.)*

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

   (f) (1.5p) Give a simple example of some (not necessarily elegant) parallel algorithm (commented pseudocode and analysis) for a problem of your choice, which is *not* cost-optimal. Justify your answer.

   Keep it really simple. Shared-memory will be most appropriate.

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. (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)

(In case of questions about the following two assignments please ask Marco Kupiainen in the first hand.)

4. (6 p.) The problem

$$\frac{du_j}{dt} + D_0 u_j(t) = 0 \quad j = 2, 3, 4, \ldots, N-1$$

with boundary data $u_1(t) = 1,\ u_N(t) = u_{N-1}(t)$, is an approximation of the partial differential equation $u_t + u_x = 0$, with boundary data $u(0,t) = 1$. The centered difference operator is given by

$$D_0 u_j = \frac{u_{j+1} - u_{j-1}}{2\Delta x}$$

Use message-passing for your solution.

(a) Draw a picture and explain how the array `u[N]`, representing the solution u_j can be distributed onto p processors. (2 p.)

(b) Write down the algorithm to evaluate $D_0 u_j$ at all points, $j = 2, 3, \ldots, N-1$, for the array distributed on the parallel computer. (2 p.)

(c) What changes would be necessary in (a) and (b) if we replace the boundary conditions by the periodic $u_1 = u_N$? Think especially about the type of send-receive needed in (b). (2 p.)

5. (9 p.) Assume that we are given two grids covering a domain of computation, and that we have a computer with two processors. Grid 1 has $n_i^1 \times n_j^1 \times n_k^1$ points, and grid 2 has $n_i^2 \times n_j^2 \times n_k^2$ points. *(Hint: these 1, 2 are upper indexes, no exponents).*

We have the array $u[g]^n_{i,j,k} \approx u(t_0 + (n-1)\Delta t, x_0 + (i-1)\Delta x, y_0 + (j-1)\Delta y, z_0 + (k-1)\Delta x)$ defined on the grid $g = 1, 2$, which is the solution of a partial differential equation. Taking one time step involves only direct neighbors in all three directions,

$$u[g]^{n+1}_{i,j,k} = S(u[g]^n_{i+1,j,k}, u[g]^n_{i,j,k}, u[g]^n_{i-1,j,k}, u[g]^n_{i,j+1,k}, \ldots)$$

We consider two different schemes to distribute the domain, i.e. the two grids and thus the two arrays $u[g]$, $g = 1, 2$, across the two processors. Assume that the time to evaluate the difference scheme S in one grid point is given by Ct_a, where C is a constant and t_a the time to do one arithmetic operation on the computer. Write down a formula for the time it takes to do one time step for the following cases (a) and (b):

(a) The two grids (arrays) are each split into two equal-sized blocks that are distributed on the machine, i.e., each processor holds $n_i^1/2 \times n_j^1 \times n_k^1$ points (elements) of Grid 1 (array $u[1]$) and $n_i^2/2 \times n_j^2 \times n_k^2$ points (elements) of Grid 2 (array $u[2]$) (3 p.)
[Hint: here you should include the time it takes to communicate the overlap boundary between processors. We do not consider the possible communication between the two grids (arrays) and assume for simplicity that they are processed independently of each other.]

(b) Grid 1 (entire array $u[1]$) is assigned to processor 1 unsplit, and Grid 2 (entire array $u[2]$) is assigned to processor 2 unsplit. (3 p.)

(c) Which approach is best if grid 1 holds $100 \times 100 \times 100$ points and grid 2 holds $150 \times 100 \times 100$ points? Assume $C = 200$, $t_a = 10^{-9}$, $t_s = 10^{-6}$, $t_w = 10^{-8}$. (3 p.)

6. (5 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) (1 p.) Give two good reasons for using collective communication operations instead of equivalent combinations of point-to-point communication (MPI_send and MPI_receive) operations.

   (c) (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)

7. (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)

8. (5.5 p.) **Parallel computer architecture**

   (a) (0.5p) What does *Moore's Law* really say? Be precise!

   (b) How is the (theoretical) *peak performance* of a parallel computer system defined? (0.5p)

   (c) Identify two important trends in supercomputer architectures that can be derived from the (recent) TOP-500 lists. (1p)

   (d) 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)

   (e) Explain the *write-invalidate protocol* used to achieve sequential consistency in a cache-based shared-memory system with a bus as interconnection network. (1.5p)

   (f) 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)