TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer /
Programming of parallel computers

2015-06-02, 14:00–18:00 KÅRA

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. If you should need one, please contact the course secretary after the result has been entered in LADOK.

Tentavisning / Exam review: to be announced on the course homepage. Afterwards, exams will be archived in the IDA student expedition.

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. (7 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) (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) 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.)*

   (d) (1p) Which aspect of parallel computation is modeled well by the PRAM (Parallel Random Access Machine) model, and which important property / properties of real parallel machines does it abstract from?

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

   (f) (1.5p) What is the difference between relative and absolute parallel speed-up? Which of these is expected to be higher, and why?

   (g) (1 p.) A long-running program is known to have a perfectly parallelizable part that accounts for 90 percent of its sequential execution time. The rest is inherently sequential. If we parallelize the program, how much parallel speedup can we expect with 9 processors? Explain your calculation.

2. (2 p.) **Parallel programming models**

   Explain the parallel program execution styles *fork-join* and *SPMD*, and explain the main difference. (1p)

   Which one is more comfortable for the programmer, and why? (0.5p)

   Which one is used in MPI? (0.5p)

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

   (a) How is the (theoretical) *peak performance* of a parallel computer system defined? (1p)

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

   (c) For a shared memory architecture, does sequential memory consistency imply a deterministic total ordering of all memory accesses? Justify your answer. (1p)

   (d) (1.5p) Simple tree-shaped interconnection networks are convenient for global aggregation of data (e.g., parallel reductions), but lead to a scalability problem when used as the main interconnection network of a parallel computer architecture. Why? (0.5p)

   Name and sketch an advanced tree-based interconnection network that does not suffer from this problem, and explain why. (1p)

5. (4.5 p.) **OpenMP**

   (a) What is *guided self-scheduling*? How does it differ from the other scheduling schemes for parallel loops defined in OpenMP? What kind of parallel loops are suitable candidates for guided self-scheduling? Why? (2p)

   (b) (1.5 p.) What is the purpose of the `flush` directive in OpenMP? Give a short example to illustrate how it is used. Name at least one technical cause that makes the explicit use of `flush` in the program necessary to guarantee a correct program execution.

   (c) In principle, every OpenMP program could likewise be expressed by explicit thread programming, e.g. by using libraries such as Pthreads. What is the main advantage of using OpenMP over such thread programming libraries? Explain your answer. (1p)

6. (8 p.) **Parallel Basic Linear Algebra**

   (a) (3p) The BLAS (Basic Linear Algebra Subroutines) library API comes in three different levels (BLAS Level 1, 2, 3).

   (i) Name one representative function from each level and give a short description. (1.5p)

   (ii) Why is it very important for a supercomputing center to have efficient implementations of BLAS installed? (0.5p)

   (iii) Which of the levels is most likely to have efficient parallel implementations even where interprocessor communication is expensive, and why? (1p)

(b) (5p) Consider the following straightforward sequential code for the multiplication of two $N \times N$ square matrices A and B:

```
for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      C[i][j] = 0.0;
for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      for (k=0; k<N; k++)
         C[i][j] = C[i][j] + A[i][k] * B[k][j];
```

(i) Apply *tiling* with tile size $S \times S$ (with $1 < S \ll N$) to the second loop nest. Show the resulting pseudocode. (1p)

(ii) Why is tiling (with a suitably chosen S) beneficial for the performance of this code on modern processor architectures? (1p)

(iii) Assume that your processors offer SIMD instructions that can load, store, multiply or add four floatingpoint words together in one clock cycle if they are adjacent in memory resp. in (SIMD) registers. Which loop in your tiled loop nest is the most suitable one for vectorization using SIMD instructions, and why? Suggest a preparing code transformation that helps with efficient vectorization, and show the resulting pseudocode. (2p)

(iv) Now parallelize the tiled code for a shared-memory parallel system, using OpenMP loop parallelization. Choose a suitable loop to parallelize. If necessary, transform the code. Show the resulting pseudocode. (1p)

7. (4 p.) **Parallel Solving of Linear Equation Systems**

(a) (4p) Consider the problem of solving a triangular linear equation system, e.g. a system $Ux = b$ where U is a $n \times n$ upper triangular matrix (i.e., all elements below the main diagonal are zero) by *backward substitution*, with the following sequential pseudocode:

$x_{n-1} \leftarrow b_{n-1}/u_{n-1,n-1}$;
for $i = n-2$ **downto** 0 **do**
$$x_i \leftarrow \left(b_i - \sum_{j=i+1}^{n-1} u_{i,j} x_j\right) / u_{i,i};$$

i. Derive the asymptotic execution time of the above sequential algorithm as an expression in n. (0.5p)

ii. Suggest a possible parallelization for a shared memory system. What kind(s) of parallelism can you exploit here? Show the resulting pseudocode (using e.g. OpenMP or pthreads), and derive the parallel execution time as an expression in n and p. You can ignore cache issues for simplicity.
Calculate the parallel efficiency, and discuss possible performance issues where the number p of parallel threads is large. (3.5p)

8. (1.5 p.) **Transformation and Parallelization of Sequential Loops**

Given the following C loop:

```
for (i=0; i<N-1; i++) {
   a[i+1] = 2.0 * a[i] / b[i] + c[i];
}
```

(a) Explain why the loop iterations cannot be executed in parallel in this form (dependence-based argument). (0.5p)

(b) Suggest a (semantics-preserving) transformation of the loop (show the new code) such that the loop iterations now could be executed in parallel. Explain why (dependence-based argument). (1p)

9. (4 p.) **MPI**

(a) (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.

(b) (1p) Today's HPC clusters have multi-core nodes. How can, in general, MPI and OpenMP parallelization be suitably combined to leverage hybrid (MPI+OpenMP) parallelism in the same application?

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

i. Explain the principle of one-sided communication in MPI-2. (1.5p)
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)