TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer /
Programming of parallel computers

2017-08-15, 14:00–18:00, KÅRA

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

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

Examinator: Christoph Kessler

Jourhavande lärare:
Christoph Kessler (examiner), visiting ca. 15:30, then on travel, reachable via 0703-666687

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 regulations by Linköping University, we do not give ECTS grades. If you need one, please contact the course secretary after the result has been entered in LADOK.

Tentavisning / Exam review: None for re-exams. 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. (6 p.) **Performance tools and analysis**

   (a) (1p) What is *false sharing*, and why and how does it affect performance?

   (b) (1p) What kind of performance data and performance data collection method could be helpful to get indications for a possible *false sharing* issue in a multi-threaded shared-memory parallel program running on a standard multicore CPU? Motivate your answer.

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

   (d) (1 p.) Amdahl's Law and Gustafson's Law differ only in a single assumption about the application's performance behavior. Which one, and how?

   (e) (2p) How is the so-called *peak performance* of a parallel computer system such as Triolith calculated?

   In particular, which assumptions are made for determining the peak performance?

   And why does the peak performance generally differ (often, significantly) from the (R_{max}) performance obtained for the LINPACK benchmark?

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

   (b) Explain the difference between a *hardware-multithreaded* processor and a *multi-core* processor. (1p)

   (c) (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)

   (d) (1.5 p.) Describe the d-dimensional Hypercube network architecture (1p).
   How does the maximum communication distance between any two nodes grow with the number p of nodes? (give a formula in p, 0.5p)

5. (6.5 p.) **OpenMP**

   (a) (2.5p) What kind of loop-based computations can benefit from using the `reduction` clause in OpenMP?

   Give also one example loop (OpenMP code without `reduction`), explain how the generated code for the loop will work instead when using `reduction` (be thorough), and give 2 main reasons why using `reduction` is likely to improve performance.

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

   (c) OpenMP allows so-called *orphaning* of some directives. What does that mean? Give also a simple example. (1p)

   (d) OpenMP 3.0 and later versions support the *task* concept. In what situations (e.g., for what kind of loops) can the use of tasks help to parallelize the execution? And how does OpenMP map the tasks to processors? (1.5p)

   (e) What is the main difference between a *PGAS programming language* and a shared memory programming language like OpenMP? (1p)

6. (8.5 p.) **Parallel Basic Linear Algebra**

   (a) (2p) Name 2 different Level-1 BLAS functions and explain for each of them how it could be parallelized on a shared-memory parallel system.

   (b) (4p) We discussed 2 variants of (sequential) matrix-vector multiplication, the ij variant and the ji variant.

   (i) Choose one of the two variants, show its sequential pseudocode (state whether you assume C or Fortran memory layout for 2D arrays), and show for this variant which level-1 BLAS function(s) it could use as subroutine(s). (2p)

   (ii) Describe the most appropriate way of parallelizing this variant for a message-passing parallel system. In particular, how should the main data structures be distributed? (2p)

   (c) (2.5p) We discussed two *systolic* parallel algorithms for matrix-matrix multiplication in the lecture (Kung-Leiserson and Cannon's algorithm).

   (i) What is a systolic parallel algorithm, in general? (1p)

   (ii) How does the communication structure in the systolic algorithms for matrix-matrix multiplication (e.g., Cannon's algorithm) look like, and how does it differ from that of non-systolic algorithms such as SUMMA? (1p)

   (iii) What kind of interconnection network topology would be suitable for these systolic algorithms, and why? (0.5p)

7. (2 p.) **Parallel Solving of Linear Equation Systems**

   Consider the message-passing parallel implementation of Gaussian Elimination / LU decomposition as discussed in the lecture. Which data distribution(s) for the matrix A is/are most appropriate, and why? (2p) *(Hint: If you need to make certain assumptions, e.g. about the used programming language, state them carefully.)*

8. (3 p.) **Transformation and Parallelization of Sequential Loops**

   (a) What is *tiling* of a loop nest, in general? (1p)

   Explain why tiling can considerably improve the performance of sequential matrix-matrix multiplication on today's computer architectures. (1p)

   (b) Is the following C loop parallelizable (in this form)?

```
    for (i=1; i<N; i++) {
S1:    a[i] = sin(3.1415 * t[i-1]) + b[i];
S2:    b[i] = 1.0 / a[i] + t[i];
    }
```

   Explain why or why not (dependence-based argument). (1p)

9. (3 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.