TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer / Programming of parallel computers

2018-08-21, 14:00–18:00, TER1

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) 013-28-2406; 0703-666687, visiting 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 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: about 2-3 weeks after the exam, to be announced on the course homepage. Afterwards, the 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.
- 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) (1.5p) 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?

   (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) 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?

   (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) (1.5p) 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. (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 p.) **Parallel computer architecture**

   (a) (1p) Can programs operating on arrays of double-precision numbers with good *spatial* locality but no *temporal* locality in their memory access behavior benefit from a cache architecture with a *cache line size* of 1 double? Justify your answer.

   (b) (1.5p) Many bus-based shared-memory systems use *bus-snooping* for cache coherence. What does that mean, and how does it work? In particular, under what condition can it guarantee sequential consistency?

   (c) (1.5p) What does *hardware multithreading* mean?

   And what is the main difference between a hardware-multithreaded processor and a *multi-core* processor?

   Under what condition (on the computations made by these threads) can hardware multithreading help to improve performance (throughput)?

   (d) (2p) For each of the following requirements, suggest the most appropriate interconnection network topology/topologies (among those discussed in the lecture):

i. As cheap as possible
ii. As short maximum latency as possible
iii. As low maximum node degree as possible
iv. Good fit for an on-chip network of a many-core CPU

Shortly justify your answers.

4. (7 p.) **OpenMP**

(a) (1p) Consider the following parallel loop:

```
#pragma omp parallel shared(N)
{
#pragma omp for
        for (i=0; i<N; i++)
            calculate_strange_function( i );
}
```

We know that the execution time of `calculate_strange_function` varies considerably depending on its argument, and that it cannot be predicted at program design time; some calls may take several seconds, others return already after a millisecond. We also know that the value of `N` is about 10 times larger than the available number of cores. Which OpenMP scheduling clause do you recommend for this loop? Motivate your decision.

(b) (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.

(c) (2.5p) 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, while the traditional OpenMP worksharing constructs are not applicable? Give an example (pseudocode, explain your program constructs) of such a case for illustration and explain how the tasks are defined in OpenMP 3+.

And how does OpenMP map the tasks to processors?

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

5. (8 p.) **Parallel Basic Linear Algebra**

   (a) (3p) Given a $n \times n$ single-precision floatingpoint matrix A and a n-element single-precision floatingpoint vector b. Assume that A is already distributed row-wise across the p nodes of a distributed-memory parallel system (i.e., rows $0...n/p$ on node 0, rows $n/p+1, ...2n/p$ on node 1 etc.) and vector b resides entirely on node 0.

   What is the most efficient way of computing the matrix-vector product $y = Ab$ using all p nodes? Show the pseudocode with the structure of communication, and analyze the parallel execution time using the Delay model (i.e., communicating a block of N elements takes linear time $\alpha + \beta N$) as a formula in n, p, α, β.

   (b) (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)

   (c) (2p) We learned about two recursive algorithms for matrix-matrix multiplication (a simpler one and a more intricate one). What is the fundamental (linear algebra) property of matrix multiplication that makes such recursive approaches applicable? Which fundamental parallel algorithmic design pattern can be leveraged in these algorithms to expose parallelism? Pick one of these two algorithms (state which one) and explain how it works.

6. (1 p.) **Parallel Solving of Linear Equation Systems**

   When considering the second step in Foster's design method, we discussed as an example the update schemes of iterative solver methods for linear equation systems, including *Jacobi*-style and *Gauss-Seidel*-style update schemes, over the matrix elements in each iteration step. Choose one of them (state which one), and explain its dependence structure between element updates and its maximum degree of parallelism within one iteration of the algorithm.

7. (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:    t[i] = 1.0 / a[i] + t[i];
    }
```

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

8. (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) (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) (1 p.) What is the purpose of the `MPI_Win_fence` operation in one-sided communication?