TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer / Programming of parallel computers

2016-08-16, 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), on travel, 0703-666687
Lu Li (course assistant) 0704-759692; visiting at ca. 16:00 (if possible)

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

   (e) (1p) There exist several possible causes for *speedup anomalies* in the performance behavior of parallel programs. Name and explain one of them.

   (f) (1.5 p.) How is the so-called *peak performance* of a parallel computer system defined?

   And how does the performance criterion used for the *TOP500 ranking* differ from the *peak performance* of a parallel computer system?

   Which of these two performance metrics would you consider more helpful if you had to decide about which type of new supercomputer system to purchase for a supercomputing center, and why?

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.) **Parallel computer architecture**

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

   (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) (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)

4. (5 p.) **OpenMP**

   (a) Given the following OpenMP parallel loop:

```
#pragma omp parallel for
 for (i=0; i<N; i++)
    while (not_converged(i))
       iterate_again(i);
```

   Assume that the number of iterations of the inner `while` loop is not known beforehand and differ for each value of `i`, i.e., that the boolean condition `not_converged` depends on the value of `i`; for some `i` a few iterations of the inner loop suffice while for others it can take several hundred iterations. Assume also that N is much larger than the available number of processors. Which of the OpenMP loop scheduling clauses do you consider most appropriate for the parallel `i` loop, and why? (1.5p)

   (b) What kind of loops can benefit from using the `reduction` clause in OpenMP? Give one example (code), and explain why using `reduction` is likely to improve performance. (2p)

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

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

5. (9 p.) **Parallel Basic Linear Algebra**

   (a) (5.5p) 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) Assume that N is large. 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. Which loop in your tiled loop nest is the most suitable one for vectorization using SIMD instructions, and why? Suggest a preparing loop 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, and a suitable scheduling policy (motivate your choices). If necessary, transform the code. Show the resulting pseudocode. (1.5p)

(b) (3.5p) For a $N \times N$ matrix A of floatingpoint elements, explain how *matrix-vector multiplication* $b = Ax$ is parallelized for a distributed memory (message passing) system. (We discussed two variants based on loop ordering, choose one of them that is likely to have better spatial data locality, explain why.)

Explain how the operand and result arrays are partitioned and distributed, how data is communicated, which communication is done by each node, and what kind of collective communication operations are used. Show (pseudo)code and explain it, preferably with an annotated figure.

Analyze the parallel execution time using the Delay model (sending a message of k floatingpoint words takes time $\alpha + \beta k$; assume that there are no network contention problems among messages and that nodes are fully connected but each node can send or receive only one message at a time). Assume that a floatingpoint operation takes 1 time unit, and ignore all other operations for simplicity. Assume that the machine has $P > 1$ single-processor nodes.

6. (2 p.) **Parallel Solving of Linear Equation Systems**

Straightforward message passing implementations for Gaussian Elimination (and similarly, LU decomposition) expose a *load balancing problem* for row-block-wise and column-block-wise distributions of the system matrix A. Explain the cause of the load balancing problem (be thorough!) and how it could be solved (to most degree). (2p)

7. (2 p.) **Transformation and Parallelization of Sequential Loops**

Given the following C loop:

```
s = a[0] * b[0];
for (i=1; i<N; i++) {
   s = s + a[i] * b[i];
}
```

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

(b) What kind of computation does this loop actually do? (0.5p)

Suggest a *parallel algorithm* for the same problem that could utilize up to N processors in parallel (give the basic idea and asymptotic parallel time complexity in the EREW PRAM model, but no details). (1p)

8. (7.5 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) (5.5 p.) **Simple String Matching in MPI**

Given is a huge array A of N characters stored in the main memory of one node of a cluster computer with P (for simplicity, fully connected) nodes ($P > 1$).

Given is also a short constant array S of M (with $M \ll N$, and M can be considered constant) characters that is available on node 0 at program start.

i. Write a message-passing parallel program (MPI-C or message passing pseudocode is fine, explain your code) that uses all P nodes in order to count the *total number* of (contiguous) occurrences of the string S in A, i.e., the number of all positions $i < N - M$ of A where, for all $j = i, ..., i + M - 1$, $S[j] == A[j]$. Use a simple brute-force parallel algorithm for string matching like the one that we discussed in the lecture. Be thorough, and explain your code carefully. (3p)

ii. Which type of parallelism did you use in your program in (i)? (0.5p)

iii. Make sure to explain all communication operations in (i). Draw a figure for $P = 4$ nodes showing the distribution of A and the flow of messages between the nodes over time. Which of these communication operations are point-to-point communications and which ones are collective communication operations? (1p)

iv. Derive the asymptotic worst-case parallel execution time for your algorithm as an expression (use big-O notation where appropriate) in N and P. (1p)