TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer / Programming of parallel computers

2016-10-18, 08:00–12:00

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 in case of emergency
August Ernstsson (assistant) 076-7710605; visiting at ca. 10:00

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

   (d) (2p) Given is an idealized processor with a fully associative last-level cache of size M memory words, with perfect LRU replacement policy and a cache line size of 1 memory word. Estimate the total number of last-level cache misses (including cold misses) for the following sequential program snippet, depending on N, M and K:

```
// A is an array of at least N+K memory words,
//    initially not in cache.
// K > 1, K < N is fixed for this loop but not statically known.
// Accumulator variable s is kept in a register during the loop.

s = 0.0;
for (i=K; i<N+K; i++)
   s += A[i] * A[i-K];
```

   Hint: You need to distinguish between several cases.

   (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 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. (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) (2p) Cache-based architectures usually have cache line sizes that contain more than one memory word.

   (i) Describe, in general, the benefit(s) of having large cache line sizes instead of small ones (quantitative argument). What kind of memory access patterns can make best use of caches with large cache line sizes? (1p)

   (ii) For a cache-based shared memory parallel system with coherent caches, in what situation(s) would a small cache line size probably be better from a performance point of view? (1p)

   (b) (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. (6 p.) **OpenMP**

   (a) (2 p.) What is *guided self-scheduling* for loops (GUIDED clause in OpenMP)? (1p)

   Contrast its strengths and weaknesses to those of dynamic scheduling and of static scheduling of loops. (1p)

   (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) (2 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.

5. (10 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.

   Show the pseudocode of both variants (state whether you assume C or Fortran memory layout for 2D arrays), and also show for each variant which level-1 BLAS function(s) it could use as subroutine(s). (2p)

   Which of the two variants is better suited for cache performance, and which one is better suited for vectorization? Explain your answer. (2p)

(c) (4 p.) How does the SUMMA algorithm for distributed-memory parallel matrix-matrix multiplication work? (2p)

Derive its time complexity (computation and communication time) as a formula in n (number of rows, columns) and in p (number of nodes and of processors). You may assume for simplicity that p is a square number, that transmitting a message of M elements takes time $\beta M + \gamma$, that only one message can be sent or received at a time, that there is no additional delay due to contention for network connections, and that a broadcast of M elements across q nodes takes time $(\beta M + \gamma) \log_2 q$. (2p)

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. (6.5 p.) **MPI**

(a) (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?

(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)