TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer / Programming of parallel computers

2016-06-01, 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 (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 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. (7.5 p.) **Performance tools and analysis**

   (a) (1 p.) Which performance data collection method is required in order to be able to draw a communication statistics diagram (displaying the amount of bytes communicated between every pair of processes)? Justify your answer (technical reasons).

   (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) (2p) Why should the PRAM (Parallel Random Access Machine) model of parallel computing be used *early* in the process of designing and analyzing parallel algorithms?

   And why should it better be replaced by some other cost model in the *later* phases of parallel program design? Give one *example* of such a parallel cost model that could be used for analysis when targeting MPI programs on a cluster system like Triolith.

   (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 each,
//    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) (0.5 p.) How (by which metric) are the supercomputers on the TOP500 list ranked?

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

   (a) Given an application with a fixed performance requirement (in GFlops). Why can, provided that the application can be parallelized (work-)efficiently, the transition from a single-core to a multicore execution platform be advantageous from a power efficiency point of view? (1p)

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

   (c) (2 p.) What is 'false sharing'? In what kind of parallel computers and in what situations does it occur, and how does it affect performance? Suggest one possible way how the problem could be reduced.

   (d) (1p) The Infiniband network used to interconnect the nodes of Triolith is based on a so-called *fat-tree* network. Explain the topology (structure) of a *fat-tree* interconnection network, and how it improves the scalability of communication bandwidth over ordinary (leaf-oriented) tree networks.

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) Why is the design of OpenMP helpful for *incremental* parallelization of sequential codes? (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. (5.5 p.) **MPI**

(a) What does the operation `MPI_Allreduce` do? And how does it differ from the operation `MPI_Reduce`? (1p)
Hint: The parameter types are as follows:

```
int MPI_Allreduce ( void *sendbuf, void *recvbuf,
                    int count, MPI_Datatype elemtype,
                    MPI_Op op, MPI_Comm comm );
```

(b) How does the *nonblocking* (also called *incomplete*) send routine `MPI_Isend` differ from the ordinary blocking `MPI_Send`?
Under what condition is it safe to replace a `MPI_Send` call by `MPI_Isend`? (1.5p)

(c) (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.

(d) (2 p.) Explain the Communicator concept in MPI. How does it support the construction of parallel software components?