TENTAMEN / EXAM
TDDC78
Programmering av parallelldatorer / Programming of parallel computers

2014-06-02, 14:00–18:00,

Christoph Kessler Dept. of Computer 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 should need one, please contact the course secretary after the result has been entered in LADOK.

Tentavisning / Exam review: in August, to be announced on the course homepage.

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) Give one typical example for performance-related data about *message passing* programs that can be collected using *software counters*.

   (b) (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?

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

   (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? *(give a commented formula)* (1p)

   And why does the peak performance generally differ significantly from the R_{max} performance obtained for the LINPACK benchmark? (0.5p)

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

   (a) How does the MSI *write-invalidate protocol* for cache coherence in a bus-based shared memory system work? *(Be thorough)* (2p)

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

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

   (c) (3p) Explain the *2D-Mesh* interconnection network topology. (0.5p)

   Determine its maximum node degree, the maximum distance of any two nodes, and the maximum aggregated bandwidth (given by the maximum possible number of communications proceeding simultaneously in the network). (1.5p)

   Shortly describe one kind of computation problem (e.g., from your labs) for execution on a message passing environment, for which a 2D-Mesh network would be more efficient than its 1D equivalent, and why. (1p)

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

   (a) OpenMP provides different scheduling methods for parallel loops. Characterize the kind of loops that are expected to perform best with *static scheduling*, and explain why. (1.5p)

   (b) (2.5p) Given the following example loop:

```
 s = 0.0;
#pragma omp parallel for shared(s)
 for (i=0; i<N; i++)
    s = s + A[i];
```

   (i) There is a correctness problem with this code. Explain, and suggest a simple fix that however leads to a scalability problem.

   (ii) Apply the `reduction` clause properly to the example loop and show what kind of code (commented pseudocode) the compiler generates for this loop. Explain why it fixes both the correctness issue and the scalability problem.

6. (5 p.) **Parallel Basic Linear Algebra**

   (a) (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)

   (b) (2p) 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];
```

   Apply *tiling* with tile size $S \times S$ (with $1 < S \ll N$) to the second loop nest. Show the resulting pseudocode.

   Why is tiling (with a suitably chosen S) beneficial for the performance of this code on modern processor architectures?

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

   (a) 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)

   (b) (5p) *Column pivoting* searches in each step k of Gaussian Elimination within column k for an element $a_{i,k}$ with maximum absolute value and then exchanges entire row i with row k (unless just $i = k$) before the updating of submatrix $A_{k..n-1,k..n-1}$ of step k is being performed.

   For a *row-block-wise* distribution of the system matrix A, describe the required additional computation and communication operations for column pivoting, and give its parallel time cost as a formula in k and n for the delay model.

   For the above analysis, use the delay model, assuming for simplicity that the underlying interconnection network is fully connected (i.e., each processor has a direct network link to every other processor) with message startup time α (clock cycles) and average (float) word transfer time β (clock cycles), and that a processor can only send or receive one message at a time. If you need to make further assumptions, state them carefully.

8. (1.5 p.) **Transformation and Parallelization of Sequential Loops**

   Given the following C loop:

```
for (i=0; i<N; i++)
   a[i] = 2.0 * a[i+1] / b[i] + c[i];
```

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

   (b) Suggest a (semantics-preserving) transformation of the loop (show the new code) such that the loop iterations now could be executed in parallel. Explain why (dependence-based argument). (1p)

9. (3.5 p.) **MPI**

   (a) 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)

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

10. (0.5 p.) **Grid Computing**

   Name one task performed by *grid middleware*.