TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer /
Programming of parallel computers

2014-10-21, 08:00–12:00, TER2

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 (examiner), on travel, 0703-666687
Lu Li (course assistant) 0704-759692; visiting at ca. 09:00

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.

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) (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) (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).

   (c) (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.)*

   (d) (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?

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

   (f) (2.5 p.) Derive Gustafsson's law and give its interpretation. Explain how it differs from Amdahl's law and for what kind of parallel computations it is more appropriate.

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. (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) (3p) Explain the *2D-Torus* 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-Torus network would be more efficient than its 1D equivalent, and why. (1p)

5. (4.5 p.) **OpenMP**

   (a) What is *guided self-scheduling*? How does it differ from the other scheduling schemes for parallel loops defined in OpenMP? What kind of parallel loops are suitable candidates for guided self-scheduling? Why? (2p)

   (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+1] = 2.0 * a[i] / 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. (4 p.) **MPI**

   (a) (4 p.) The `MPI_Barrier` operation is a collective communication operation with the following signature:

```
int MPI_Barrier ( MPI_Comm comm );
```

   where the current communicator is passed as `comm`.

   Once called, the function does not return control to the caller before all p MPI processes belonging to the process group of communicator `comm` have called `MPI_Barrier` (with this communicator).

i. Write an implementation of `MPI_Barrier` using only `MPI_Send` and `MPI_Recv` operations. Use C, Fortran, or equivalent pseudocode (explain your language constructs if you are not sure about the right syntax). *Explain your code.* (2p)
(Note: There exist several possibilities. An algorithm that is both time-optimal and *work-optimal for large p gives a 1p bonus, provided that the analysis is correct and the optimality is properly motivated.)*

ii. Show how your code behaves over time by drawing a processor-time diagram for $p = 8$ showing when messages are sent from which source to which destination, and what they contain. (0.5p)

iii. Analyze asymptotically the (worst-case) *parallel execution time*, the *parallel work* and the *parallel cost* of your implementation (for arbitrary values of p) as a function in p. You may use the delay model or the LogP model for this purpose (state which model you use). If you need to make further assumptions, state them clearly. (1.5p)
Hint: Communication contributes to the work done by a parallel algorithm.