TENTAMEN / EXAM

TDDC78

Programmering av parallelldatorer /
Programming of parallel computers

2018-10-31, 08:00–12:00, G36

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) 013-28-2406; 0703-666687, visiting 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 regulations by Linköping University, we do not give ECTS grades. If you need one, please contact the course secretary after the result has been entered in LADOK.

Tentavisning / Exam review: about 2-3 weeks after the exam, to be announced on the course homepage. Afterwards, the 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.
- 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. (8 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) (2 p.) Describe performance analysis through tracing. (1p)

   What are the advantages and disadvantages of the approach, in comparison to alternative techniques for performance data collection? (1p)

   (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) (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.

   (e) (2p) How is the so-called *peak performance* of a modern cluster-based parallel computer system such as Triolith calculated? *(derive a commented formula).*

   In particular, which assumptions are made for determining the peak performance?

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

   Finally, what is the *unit* of performance typically used in the High-Performance Computing domain?

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

   (a) (1p) Can programs operating on arrays of double-precision numbers with good *spatial* locality but no *temporal* locality in their memory access behavior benefit from a cache architecture with a *cache line size* of 1 double? Justify your answer.

   (b) (1.5p) Many bus-based shared-memory systems use *bus-snooping* for cache coherence. What does that mean, and how does it work? In particular, under what condition can it guarantee sequential consistency?

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

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

Under what condition (on the computations made by these threads) can hardware multithreading help to improve performance (throughput)?

(d) (2p) For each of the following requirements, suggest the most appropriate interconnection network topology/topologies (among those discussed in the lecture):

i. As cheap as possible

ii. As short maximum latency as possible

iii. As low maximum node degree as possible

iv. Good fit for an on-chip network of a many-core CPU

Shortly justify your answers.

4. (6 p.) **OpenMP**

(a) (2.5p) What kind of loop-based computations can benefit from using the `reduction` clause in OpenMP?

Give also one example loop (OpenMP code without `reduction`), explain how the generated code for the loop will work instead when using `reduction` (be thorough), and give 2 main reasons why using `reduction` is likely to improve performance.

(b) In principle, every OpenMP program could likewise be expressed by explicit thread programming, e.g. by using libraries such as Pthreads. What is the main advantage of using OpenMP over such thread programming libraries? Explain your answer. (1p)

(c) (2.5p) OpenMP 3.0 and later versions support the *task* concept.

In what situations (e.g., for what kind of loops) can the use of tasks help to parallelize the execution, while the traditional OpenMP worksharing constructs are not applicable? Give an example (pseudocode, explain your program constructs) of such a case for illustration and explain how the tasks are defined in OpenMP 3+.

And how does OpenMP map the tasks to processors?

5. (7 p.) **Parallel Basic Linear Algebra**

(a) (1.5p) Name a Level-1, a Level-2 and a Level-3 BLAS function and give for each one a short explanation of its (main) operands and computation.

(b) (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)

6. (1 p.) **Parallel Solving of Linear Equation Systems**

When considering the second step in Foster's design method, we discussed as an example the update schemes of iterative solver methods for linear equation systems, including *Jacobi*-style and *Gauss-Seidel*-style update schemes, over the matrix elements in each iteration step. Choose one of them (state which one), and explain its dependence structure between element updates and its maximum degree of parallelism within one iteration of the algorithm.

7. (1 p.) **Transformation and Parallelization of Sequential Loops**

(a) Is the following C loop parallelizable (in this form)?

```
    for (i=1; i<N; i++) {
S1:    a[i] = sin(3.1415 * t[i-1]) + b[i];
S2:    t[i] = 1.0 / a[i] + t[i];
    }
```

Explain why or why not (dependence-based argument). (1p)

8. (7 p.) **MPI**

(a) (1.5p) 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?

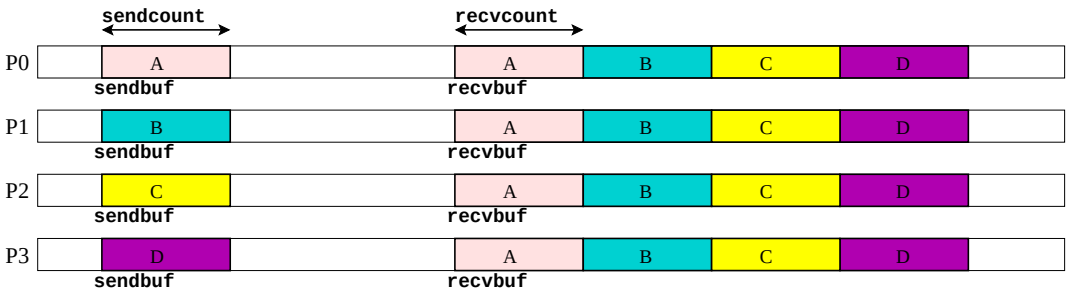

Figure 1: Effect of the `MPI_Allgather` operation on the processes' local memories, here shown for $p = 4$ processes.

(b) (5.5 p.) The `MPI_Allgather` operation is a collective communication operation with the following parameters:

```
int MPI_Allgather ( void *sendbuf, int sendcount, MPI_Datatype sendtype,
                    void *recvbuf, int recvcount, MPI_Datatype recvtype,
                    MPI_Comm comm );
```

`MPI_Allgather` is a generalization of `MPI_Gather`: If executed by a group of p MPI processes each contributing a local array in `sendbuf` with `sendcount` elements of type `sendtype`, each of the p processes will afterwards hold a copy of the entire gathered array in its `recvbuf`, which is composed of the $p \times$`recvcount` elements coming from each send buffer in the order of the processor ranks. See Figure 1.

Usually, the arguments for `sendcount` and `recvcount` and for `sendtype` and `recvtype`, respectively, are equal in calls to `MPI_Allgather`. Also, all participating processes must pass equal argument values for each of these parameters. A typical call could look as follows:

`MPI_Allgather ( Arr1,` n`, MPI_FLOAT, Arr2,` n`, MPI_FLOAT, MPI_COMM_WORLD );`

i. What is the difference in behavior to `MPI_Gather`? (0.5p)

ii. Write an implementation of `MPI_Allgather` using only `MPI_Send` and `MPI_Recv` operations (i.e., no other communication operations). Use C, Fortran, or equivalent pseudocode (explain your language constructs if you are not sure about the right syntax). Explain your code. (2.5p)

iii. Show how your code behaves by drawing an annotated processor-time diagram for $p = 4$ showing *when* messages are sent from *which* source to *which* destination, and *what* they contain. (1p)

iv. Let n denote the value of `sendcount` and `recvcount`, and assume that the `send`/`recvtype` denotes normal floatingpoint numbers. Analyze asymptotically the (worst-case) *parallel execution time* of your implementation (for arbitrary values of p and n) as a function in p and n. You may use the delay model, the BSP model or the LogP / LogGP model for this purpose (state which model you use). If you need to make further assumptions, state them clearly. (1.5p)