Contact during exam:
Einar M. Rønquist (73593547 or 93404778)

FINAL EXAM FOR COURSE TMA4280

INTRODUCTION TO SUPERCOMPUTING

Friday, May 27, 2011
Time: 09:00–13:00

Permitted aids:	Approved calculator.
	All printed and hand written aids.

In general:

- Final grades will be announced by June 17.
- Short answers are fine as long as they are justified and explained.
- There are 15 questions and 4 points to be earned on each one.

Problem 1

Let $\underline{a}$ and $\underline{b}$ be real vectors of length n, and $\underline{A}$, $\underline{B}$ and $\underline{C}$ be real $n \times n$ matrices.
Consider the innerproduct (or dot product)

$$\sigma = \underline{a}^T\underline{b} = \underline{a} \cdot \underline{b} \tag{1}$$

and the matrix-matrix multiplication

$$\underline{C} = \underline{A}\,\underline{B}. \tag{2}$$

We recall that each matrix element c_{ij} of $\underline{C}$ can be expressed as the innerproduct between the i-th row of $\underline{A}$ and the j-th column of $\underline{B}$, i.e., one innerproduct per matrix element.

a) First, we implement the matrix-matrix multiplication (2) on a single processor using a fast implementation of the innerproduct operation (1) (e.g., using Level 1 BLAS). Timing of (2) with $n = 1000$ yields an average time per innerproduct equal to $1.0 \cdot 10^{-5}$ seconds. What is the time to perform (2) for this implementation? What is the corresponding performance (in terms of the number of floating point operations per second)?

Fasit: The matrix $\underline{C}$ can be computed using n^2 innerproducts since there are n^2 elements in $\underline{C}$. Hence, the time to perform (2) is $10^{-5} \cdot 1000^2$ seconds $= 10$ seconds.

Operation (2) requires $2n^3$ floating point operations. Hence, the performance of (2) using this implementation is $(2 \cdot 10^9/10)$ flops $= 200$ Mflops.

This is also the same as the performance of a single innerproduct: operation (1) requires $2n$ operations, and the associated performance is $(2 \cdot 1000/10^{-5})$ flops $= 200$ Mflops.

b) Next, we implement (2) using Level 3 BLAS. The performance is measured to be 6 Gflops for $n = 1000$. What is the time to perform (2) for this implementation? Does this seem reasonable compared to the result in a)?

Fasit: The performance is 30 times higher than in a). Hence, the time to perform (2) is 30 times smaller than in a), or 1/3 second. Since operation (2) requires $2n^3$ floating point operations, we could also find the time as $(2 \cdot 1000^3/6 \cdot 10^9)$ seconds $= 1/3$ second.

Yes, the results seem reasonable. For example, tests on `njord` has shown that we are able to obtain a performance of approximately 6 Gflops for operation (2) using Level 3 BLAS (with $n = 1000$). However, a `daxpy` operation (also a vector-vector operation or Level 1 BLAS operation similar to the innerproduct operation) only yields approximately 200 Mflops on `njord`. The Level 3 BLAS implementation is able to exploit the computational resources in a better way and is typically able to get fairly close to maximum theoretical performance.

Problem 2

We would like to solve the Poisson problem on the unit square $\Omega = (0,1) \times (0,1)$ with boundary conditions $u = 0$ specified along the boundary $\partial\Omega$. We discretize the problem using finite difference techniques; the Laplace operator is approximated using the standard 5-point stencil. The grid spacing in each spatial direction is $h = 1/n$, where $n \gg 1$.

The conjugate gradient method is used to solve the system of algebraic equations. The implementation is done in double precision. In the multi-processor case, the original grid is decomposed into P approximately equal subgrids, each of size $m \times m$, i.e., $m^2 \approx n^2/P$ (i.e., we "slice" the grid both in the x- and y-direction). We assign one subgrid to each processor.

In the following, you can assume that the time it takes to send a message of k bytes over the network can be expressed as $\tau_C(k) = \tau_S + \gamma \cdot k$, where τ_S is the startup time and γ is the inverse bandwidth.

a) Give an expression for the communication cost per conjugate gradient iteration.

Fasit: From the lecture notes (and earlier exams), the communication is associated with the exchange of surface data between adjacent subdomains and the two innerproducts per iteration. Hence, the communication cost per iteration is $T_{comm} = 8\tau_C(8m) + 2\tau_S \log_2 P$.

b) Consider the case with $n = 8000$ and $P = 64$. What is the communication time per iteration (in seconds)? You can assume that $\tau_S = 10^{-6}$ s, and $\gamma = 1.25 \cdot 10^{-9}$ s/byte.

Fasit: In this case $m^2 = 64 \cdot 10^6/64 = 10^6$, implying that $m = 1000$. Hence, the communication time is $T_{comm} = 8\tau_C(8000) + 2\tau_S \log_2(64) = 8(\tau_S + 1.25 \cdot 10^{-9} \cdot 8000\,\mathrm{s}) + 12\tau_S = 8(1+10)\tau_S + 12\tau_S = 100\tau_S = 10^{-4}$ s.

c) Consider now the case with $n = 16000$ and $P = 256$. Will the speedup increase or decrease compared to the case in b)?

Fasit: From the lecture notes the speedup can be expressed as

$$S_p = P\left(\frac{1}{1 + P\frac{T_{comm}}{T_1}}\right),$$

where T_{comm} is the communication cost per iteration and T_1 is the solution time on a single processor per iteration. We know that T_1 is proportional to n^2 (an estimate from the lecture notes is $T_1 = 19n^2\tau_A$). Since n is twice as large in case c) compared to case b), T_1 is 4 times larger in case c) compared to case b). On the other hand, P is 4 times as large in case c) compared to case b). Hence, the change in speedup only depends on T_{comm}. Since m is the same in both case b) and case c), the communication cost for the exchange of surface data is the same. However, the communication cost for the innerproducts is larger in case c) than in case b) since P is larger. Hence, T_{comm} is larger in case c) than in case b), implying that the speedup in case c) will decrease compared to case b).

d) What is the total memory requirement per processor for the conjugate gradient method (in number of bytes)? Consider the cases in b) and c).

Fasit: The memory requirement for the conjugate gradient method on a single processor is approximately equal to the memory needed to store 4 vectors of length n^2. Here, we assume a matrix-free implementation (i.e., we do not store the system matrix, $\underline{A}$). The memory requirement per processor is thus approximately equal to $8 \cdot 4\,m^2$ bytes $= 32\,m^2$ bytes. Since $m = 1000$ in both case b) and in case c), the memory requirement per processor is approximately equal to 32 Mbytes.

e) The parallel efficiency for case b) is $1000/1001 \approx 0.999$. What is the average time for a floating point operation (τ_A) in this case?

Fasit: From the expression for the speedup S_p given above we conclude that $P\frac{T_{comm}}{T_1} = 0.001$, where $T_1 = 19n^2\tau_A$, $T_{comm} = 10^{-4}$ s, $n = 8000$, and $P = 64$. Inserting the numbers gives $\tau_A = \frac{100}{19} \cdot 10^{-9}$ s ~ 5 ns.

Problem 3

For each point in this problem, please choose the correct alternative.

a) Consider the following MPI-function (here given in C, but this is not important):

```
MPI_Send(buffer, 1024, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
```

The amount of data sent corresponds to 128 floating point numbers in double precision.
Answer: true or false.

Fasit: False. In this case 1024 floating point numbers are sent, i.e., 8192 bytes.

b) The most efficient implementation of the `MPI_Allreduce` operation on P processors completes in a certain number of communication stages. Which of the 4 alternatives is correct: (i) 1 stage; (ii) $\log_2(P)$ stages; (iii) $2\log_2(P)$ stages; (iv) P stages ?

Fasit: (ii) $\log_2(P)$ stages.

c) The functions `MPI_Send` and `MPI_Recv` are appropriate to use for the exchange of surface data in Problem 2. Is it possible to experience "deadlock" when using these functions?
Answer: yes or no.

Fasit: Yes. If the program does not honor matching send and receive, deadlock can occur.

d) If the single-processor speed is decreased from 1 Gflops to 200 Mflops, the speedup of a multi-processor program will be closer to ideal speedup.
Answer: true or false.

Fasit: True The computation/communication ratio will increase, making the relative impact of communication less noticable.

e) If $(1/k)$th of the time spent executing an algorithm involves operations that must be performed sequentially, then the maximum speedup achievable by a parallel form of the algorithm is k.
Answer: true or false.

Fasit: True; this is Amdahl's law.

f) Assume that P floating point numbers are distributed across P processors. Finding the maximum of all these numbers can be done using the recursive doubling algorithm.
Answer: true or false.

Fasit: True. This algorithm is used in `MPI` to find the global sum, the maximum, the minimum, etc.

g) We decide to tighten the stopping criterion (i.e., reduce the tolerance) in the conjugate gradient iteration. We observe that twice the number of iterations is now required to reach convergence. Will this change the expected speedup on P processors?
Answer: yes or no.

Fasit: No. The computation/communication ratio per iteration is the same. Hence, the overall speedup stays the same.

h) An efficient parallel implementation of a Monte Carlo method requires the same sequence of random numbers generated on each processor.
Answer: true or false.

Fasit: False; on the contrary.

Good luck!

Einar M. Rønquist