Norwegian University of Science and Technology
Department of Mathematical Sciences

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

FINAL EXAM FOR COURSE TMA4280

INTRODUCTION TO SUPERCOMPUTING

Friday, May 31, 2010
Time: 09:00–13:00

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

In general:

- Final grades will be announced by June 21.
- 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 $\underline{A}$, $\underline{B}$ and $\underline{C}$ be real $n \times n$ matrices. Consider the following operations:

$$\begin{aligned} \underline{C} &= \underline{A} + \underline{B} \qquad (1) \\ \underline{C} &= \underline{A}\,\underline{B} \qquad (2) \end{aligned}$$

i.e., matrix addition and matrix multiplication, respectively. We are interested in the performance (in terms of the number of floating point operations per second) in the single processor case on `njord` (the current supercomputer at NTNU). The program is compiled using the highest level of optimization.

a) Will the performance of (1) and (2) be the same? If not, which operation will yield the highest performance? Explain your answer.

Fasit: Operation (2) will be faster than (1). The memory traffic for both operations is the same: we need to fetch $\underline{A}$ and $\underline{B}$ (each matrix has n^2 floating point numbers) , and we need to store $\underline{C}$ (n^2 floating point numbers). Operation (1) involves n^2 flops (additions), while operation (2) involves approximately $2n^3$ flops (multiplications and additions). Hence, the number of floating point operations per floating point number fetched or stored is $\mathcal{O}(1)$ for operation (1) and $\mathcal{O}(n)$ for operation (2). This yields a higher performance for operation (2), especially since we can also exploit the superscalar feature (chaining) available on each processor.

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 $\frac{n}{P} \times n$ (i.e., we "slice" the grid only in the x-direction), and we associate one subgrid with 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) Exactly how many unknowns do we have in this problem?

Fasit: The number of unknowns is precisely $N = (n-1)^2$ (the number of internal points).

b) There are two contributions to the communication cost:
(i) the exchange of surface data between neighboring subgrids, and
(ii) the computation of global innerproducts.
For which value of P is the communication cost of (i) twice that of (ii)?
You can assume that $n = 1000$, $\tau_S = 10^{-6}$ s, and $\gamma = 10^{-9}$ s/byte.

Fasit: From the lecture notes (and earlier exams), the communication cost per iteration for (i) is $T_{comm,i} = 4\tau_C(8n)$, while the communication cost per iteration for (ii) is $T_{comm,ii} = 2\tau_S \log_2 P$. For the first expression we have assumed that $P > 2$. Setting $T_{comm,i} = 2T_{comm,ii}$ and using the given network model, we get

$$4(\tau_S + \gamma \cdot 8n) = 4\tau_S \log_2 P.$$

Inserting the given numbers yields $\log_2 P = 9$ or $P = 512$. (Our earlier assumption that $P > 2$ is thus valid.)

c) Assume that for a particular value of n, and for $P = 128$, the total communication cost per conjugate gradient iteration is 10^{-2} s. The parallel efficiency is equal to 0.5. The solution is obtained after 100 iterations. What is the total solution time (in seconds) on a single processor?

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. Since the parallel efficiency is equal to 0.5, we have $P\frac{T_{comm}^*}{T_1^*} = 1$. Hence, $T_1^* = PT_{comm}^* = 128 \cdot 0.01s = 1.28s$. With 100 iterations, the total solution time on a single processor is $T_1 = 128s$.

d) How would you check whether the conjugate gradient iteration has converged?

Fasit: The residual vector $\underline{r}$ is the difference between the right hand side and the left hand side of our system of equations. A small residual generally means that we have a good approximation of the solution. The length of the residual vector is automatically available during the conjugate gradient iteration (since we compute $s = \underline{r}^T\underline{r}$). Hence, one alternative is to check if $\sqrt{s} < tol$, where tol is prescribed tolerance, e.g., $tol = 10^{-6}$.

Problem 3

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

a) Let a, b, and c be three floating point numbers in double precision.
Assume that $a = 1 \cdot 10^{-12}$ and $b = 2 \cdot 10^{-12}$. We compute $c = a + b$.
The answer c is accurate to about 4 digits.
Answer: true or false.

Fasit: False. We have about 16 digits of accuracy.

b) A recommended way to implement matrix-matrix multiplication on a single processor is to use an appropriate Level-1 BLAS routine.
Answer: true or false.

Fasit: False. Level 1 BLAS include only vector operations. We should use Level 3 BLAS.

c) It is faster to send one long message over the network rather than many small messages.
Answer: true or false.

Fasit: True. This is because of the overhead associated with the startup of a message.

d) The spectral bisection algorithm is useful in order to automatically partition a computational grid into subgrids. For two-dimensional problems, the subgrids will always cover approximately the same area.
Answer: true or false.

Fasit: False. The spectral bisection algorithm will give us subgrids with approximately the same number of gridpoints, not necessarily the same area.

e) In the multi-processor case, a global sum can be performed using `MPI_Allreduce` with the `MPI_SUM` option. The answer will be identical on all the processors, i.e., all the bits in the global sum will be the same on all the processors.
Answer: true or false.

Fasit: True.

f) Assume that P floating point numbers are evenly distributed across P processors. Using `MPI`, computing the product of all these numbers can be done equally fast as computing the sum.
Answer: true or false.

Fasit: True.

g) In the multi-processor case, the function `MPI_Scatter` is sometimes convenient to use. Do you expect to experience "deadlock" when using this function?
Answer: yes or no.

Fasit: No. We expect the functions in the MPI library to be properly implemented and tested.

h) It is typically easier to achieve good speedup for a code compiled with the highest level of compiler optimization compared to a code using no compiler optimization at all.
Answer: true or false.

Fasit: False. See lecture notes, exercises and earlier exams.

i) On `njord` (the current supercomputer at NTNU), it is recommended to exclusively use OpenMP if you need to run your program on more than 32 processors.
Answer: true or false.

Fasit: False. We can only use shared memory programming within a single node on `njord`, i.e., using a maximum of 16 processors. We need to use MPI when we need to communicate across nodes.

j) The first part of a parallel program involves reading input data from a single file stored on a single hard drive. In the single-processor case, the program spends about 1% of the time on this task. The maximum speedup for this program is 100.
Answer: true or false.

Fasit: True. Reading from a single file is a sequential operation, even in the multiple processor case. Hence, the sequential part of this program will at least be 1%. According to Amdahl's law, the maximum speedup is 100.

Good luck!

Einar M. Rønquist