NTNU – Trondheim
Norwegian University of
Science and Technology

Department of Mathematical Sciences

Examination paper for
TMA4280 Introduction to Supercomputing

Academic contact during examination: Arne Morten Kvarving

Phone: 97544792

Examination date: Aug, 2015

Examination time (from–to): 09:00-13:00

Permitted examination support material: Code: C

All lecture notes, slides, codes, exercises and suggested solutions from course material.
Rottmann: Mathematical formulas.
Earlier exams+suggested solutions in TMA4280.
LINPACK specification and FAQ.
All handwritten notes, including annotations on notes/slides/codes.
Wikipedia printouts.
Introduction to parallel programming from llnl.gov
Simple, approved calculator.

Other information:

- There are 14 questions and 4 (8) points to be earned on each one.
- Make sure to always state any assumptions you make.

Language: English

Number of pages: 4

Number pages enclosed: 0

Checked by:

Date Signature

Problem 1 We have considered two ways of parallelizing a program in the course.

a) Give an overview of the two programming models. In particular, focus on typical issues encountered when using them separately as well as additional things to watch out for when using them in combination.

b) Consider the following code,

```
double do_some_funky_stuff(int N, int rank, int size, MPI_Comm* comm)
{
  int i;
  Vector result = createVector(N, comm, 1, 0);
  Vector result2 = createVector(N, comm, 1, 0);
  for (i=0;i<result.len;++i)
    result[i] = rand() / RAND_MAX;
  MPI_Allreduce(result.data, result2.data, result.len,
                MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
  return dotproduct(result2, result);
}

int main(int argc, char** argv)
{
  MPI_Init(&argc,&argv);
  MPI_Comm comm;
  int size, rank;
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  MPI_Comm_size(MPI_COMM_WORLD,&size);
  if (size%2 != 0) {
    MPI_Finalize();
    return 1;
  }
  MPI_Comm_split(MPI_COMM_WORLD, rank/2, rank%2, &comm);
  double val = do_some_funky_stuff(1000,rank,size,&comm);
  MPI_Comm_free(&comm);
  MPI_Finalize();
}
```

Turns out this program crashes. Explain why.

c) Explain how domain decomposition is implemented within the PETSc framework.

d) Outline how you would perform dense matrix-vector products for row and column split matrices in a distributed memory context. Which of these methods do you think is the most efficient? Consider both the number of FLOP required and the communcation time (you can use the standard linear network model). Is any of the partitioning approaches preferable with a hybrid (MPI+OpenMP) model?

Problem 2 Consider the solution of the linear system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

where $\mathbf{A}$ is a $(n-1)^3 \times (n-1)^3$ matrix and $\mathbf{b}$ is a vector of length $(n-1)^3$.

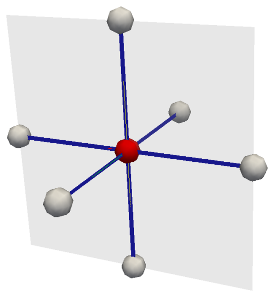

Figure 1: The stencil used to form (1). The resulting matrix will have a bandwith of $b = m^2$.

Here $\mathbf{A}$ originates from the discretization of a 3D Poisson problem with homogenous Dirichlet boundary conditions, using a centered difference approach on a grid with $n+1$ grid points in each direction; see Figure 1. The unknowns are numbered in a natural order running fastest along the x direction. Sue, a researcher without any background in Supercomputing and numerical algorithms, comes to you with the problem (1). This problem is at the core of her work, which is a forecasting application. She needs an estimate of the maximum problem size she can solve, while still keeping within time limits.

The machine she has access to is a cluster with a rather slow network, and it only allows for a pure distributed memory programming model. You can assume a simple network model, where $\tau_c(k)$, the time to send k bytes over the network can be estimated as

$$\tau_c(k) = \tau_s + \gamma \cdot k,$$

where τ_s is a startup time and γ is the inverse bandwith. For the particular computer in use, you can use $\tau_s = 1 \times 10^{-4}\,\mathrm{s}$ and $\gamma = 1 \times 10^{-7}\,\mathrm{m\,s^{-1}}$.

We also assume a very simple model for the computation times, and assume that we get 50% of the full superscalar behavior of the processors, which runs at 2.4GHz.

The algorithm in use is a preconditioned CG approach, in combination with domain decomposition. The preconditioner is diagonal, i.e., it requires a single flop per unknown.

a) Give an estimate of the flop count, the memory usage and the parallel speedup.

b) The time restriction is 10 seconds, and Sue can use 64 processors. What is your recommendation with regards to the problem size? We need 10 iterations to reach an acceptable tolerance. You can ignore any boundary effects.

Note! You will end up with an expression that is cubic in m. With the limited calculator you are allowed, solving this may be a challenge. The symbolic expression and an estimate of m is enough to get full score.

You can earn up to 8 points on this task.

c) For some problems, it is possible to observe superlinear speedup, i.e., $S_P > P$. What is the reason for this? Do you think it is possible to observe for a inner product (dot product)?

Problem 3 For each task in this problem, please choose the correct alternative and give your reasoning. You get 0 points for a wrong answer, 1 point for a correct alternative with the wrong/lacking explanation and 4 points for the correct alternative with a correct explanation.

a) An n-way associative cache is much more prone to cache trashing.

Answer: true or false

b) Dense *LU* factorization is useless for large systems due to the low FLOPS.

Answer: true of false

c) The BLAS and LAPACK libraries generally work with row-major matrix formats.

Answer: true or false

d) If we do parallel Monte-Carlo simulations, it is imperative to know the period of the PRNG to avoid doing the same simulations several times within a single process.

Answer: true or false

e) Using MPI-IO individual I/O operations are preferred for maximizing speed.

Answer: true or false

f) A SIMD processor can perform several multiplications simultanously.

Answer: true or false

g) OpenMP can be used with all C/Fortran compilers.

Answer: true or false

h) MPI can be used with all compilers.

Answer: true or false