Assignment no. 4

Solving the Poisson problem in a parallel distributed memory environment

DCAMM PhD Course
Scientific Computing

DTU

January 2008

In this exercise, we consider the solution of a Poisson problem of fundamental interest in many engineering applications. Thus, we can both learn and explore the fundamental MPI routines for solving a partial differential equation (PDE) in a parallel distributed memory environment.

1 Problem formulation

In two spatial dimensions using Cartesian coordinates $(x, y) \in \mathbb{R}^2$ on the unit square it can be stated in the form

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \quad (x, y) \in \Omega, \tag{1}$$

where $f(x, y)$ is a source function, $\Omega = \{(x, y) \in [0, 1]^2\}$ is the domain and $u(x, y)$ is an unknown scalar potential function subject to the boundary conditions

$$u = 0, \quad (x, y) \in \partial\Omega, \tag{2}$$

which ensures that the problem is wellposed (i.e. solution exist and is unique). In the following, the source function is defined as $f(x, y) = 40 \sin(16(2x - 1)y)$. Remark, that if $f(x, y) = 0$ Eq. (1) is the homogenous Laplace equation.
To solve the Poisson equation, we use a second order accurate centered finite difference method to approximate the partial derivatives on a uniform and structured mesh which can be defined as $x_i = i\Delta x$ for $i = 0, ..., N$ and $y_i = i\Delta y$ for $j = 0, ..., M$ with the grid increments $\Delta x = 1/N$ and $\Delta y = 1/M$. On the discrete grid, we then have $u(x_i, y_j) = u_{i,j}$. Thus, we can derive the following numerical scheme

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} + \mathcal{O}(\Delta x^2, \Delta y^2) = f(x_i, y_j), \tag{3}$$

which we can rearrange into the following simple iterative scheme

$$u_{i,j}^{n+1} = \frac{1}{4}\left(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - \Delta x^2 f_{i,j}\right), \tag{4}$$

having assumed $\Delta x = \Delta y$. Remark, that the new function value $u_{i,j}^{n+1}$ is expressed in terms of the function values of the neighbors computed at the last n'th iteration. This method is the classical point Jacobi method and requires an initial guess, which can be set to $u^0(x, y) = 0$ for

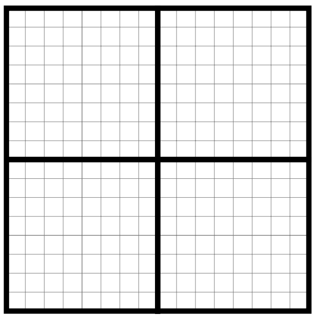
(a)

Figure 1: Example of partition strategy; domain decomposition using 4 blocks.

all $(x, y) \in \Omega$. Also, since we intend to use an iterative method a suitable stop criterion for the algorithm must be chosen, e.g. maximum iterations and an absolute error tolerance measured by a suitable metric. Although, the point Jacobi method can be a convenient method to start with in terms of implementation (it is both simple and fully parallizable), it is wellknown that it may exhibit slow convergence compared to alternative iterative solutions strategies. Therefore, alternative iterative schemes of your own choice may be considered, e.g. the point Gauss-Seidel (GS) scheme given as

$$u_{i,j}^{n+1} = \frac{1}{4}\left(u_{i+1,j}^{n} + u_{i-1,j}^{n+1} + u_{i,j+1}^{n} + u_{i,j-1}^{n+1} - \Delta x^2 f_{i,j}\right), \tag{5}$$

where the last computed values are used whenever available (here grid is assumed swept through by the columns).
The task is to solve the Poisson problem Eq. (1) on a parallel distributed memory environment. Carefully design and implement the algorithm using Fortran/C/C++ and MPI.

2 Parallel strategy

The standard approaches to a parallel solution strategi for Eq. (1) is to either a) use a fast sequential solver and parallelize this solver as efficiently as possible, or start with b) a geometric domain decomposition of the problem into a number of subdomains with overlap. Focus on the latter strategy b) for the exercise problem.
Thus, you need to design a grid partitioning strategy which is based on a domain decomposition into geometric blocks. On each block use some iterative scheme, e.g. start with the Jacobi Eq. (4) or Gauss-Seidel method Eq. (5). You can try alternative iterative schemes to try and improve the performance after you have a working code running.

2.1 Block decomposition with overlap

The global domain Ω is partitioned into a number of subdomains Ω^k, $k = 1, ..., K$, which overlaps in regions of width corresponding to one grid point. The points in the overlapping regions are called ghost points. On each subdomain Ω^k we employ an iterative scheme, e.g. the point Jacobi method Eq. (4), and then we exchange the updated grid points in each overlapping region before continuing the iteration in each subdomain. The number of unknowns in each block or subdomain should be comparable, i.e. roughly $\approx n/K$ for load balancing. To reduce communication overhead the ghost points to each subdomain should be a small fraction of the total points within each subdomain.

3 Guidelines for the numerical experiments

Increase the size of your system by increasing M, N as much as possible to put your code to the test on the computer to its full capacity. You are free to use as many processors as you decide. When you do the timings, submit your jobs to the batch system.

4 Optional task

If time permits it, feel free to test other and perhaps more complex iterative solutions strategies for solving the problem. For example, a very simple domain decomposition strategy combined with a coarse grid correction (basic idea exploited in multigrid solvers). Discuss and seek advice on optional tasks with the teachers of the course before carrying it out on your own. Any solution strategy chosen should be carefully explained in the report.

5 Writing a report on the results

The report has to have the following issues covered:

1. Problem description
2. Theoretical part
   (a) Description of the chosen partitioning strategy - why has this strategy been chosen?
   (b) Derivation of the computational and communication complexity of the algorithm for the chosen data mapping and logical computer geometry. Try and give analytical expressions for parallel runtime T_p, speedup S, efficiency $E = T_s/T_p$ and isoefficiency function for your algorithm.
3. Numerical experiments
   (a) Write a sequential program code in your language of choice (e.g. Matlab, Fortran or C) which can be used for validation and comparison with your parallel code.
   (b) Give a description/discussion of the parallel implementation - algorithms and data structures. What kind of communications have been used and why? Give figures for the memory requirements of your program implementation.
   (c) Include table(s) of results for various problem sizes, varying number of processors, corresponding speedup and efficiency figures.
   (d) Present plots of the speedup (show also the ideal speedup on the plot). The plots cam be done using Matlab, Maple ot any other graphical tool.
   (e) Analyse the speedup and efficiency results in comparison with the theoretical expectations. Does the scalability of your formulation depend on the the architectural characteristics of the machine? Make relevant comparisons with the results you obtained from the experiments.
   (f) Add observations and comment on how the particular computer architecture influences the parallel performance. Does the pair *algorithm - computer system* scale?
4. Conclusions, ideas for possible optimizations.

The report must be written in English. A listing of the program code has to be attached to the report. Standard requirements are put on the design of the code, namely, structure and comments. You are encouraged to work in pairs. However, all topics in the assignment should be covered by each student, and you have to hand in individual reports!

6 Deadlines and credit points

The full report (paper copy!) must be submitted no later than January 30, 2008. Assignments submitted after this date will not be approved.

Allan Engsig-Karup

Any comments on the assignment will be highly appreciated and will be considered for further improvements. Thank you!