Parallel Computing for Partial Differential Equations — Matthias K. Gobbert
Sommersemester 2012 — Universität Kassel
Homeworks 5, 6, and 7 — due on Wednesdays, May 09, 16, and 23, 2012, respectively

This homework is set up in a style that is similar to a typical project. That is, fairly little structure is given in this assignment. The stress of this text is on explaining the goals of the investigation, some concrete tasks that are to be accomplished, and to specify the issue on which you are supposed to provide guidance at the end.
You should structure this unstructured problem yourself. Based on your experience with previous homework, you know what steps are typically needed to work a problem. Some suggestions will be made below. Ideas might also be gleaned from reading research papers or technical reports and noting their outline. Do not hesitate to talk to me to work out the outline of your attack on the problem.

Background and Idea This homework is designed to learn about the advanced point-to-point communication commands that are available in MPI. One Reference is Chapter 13 in Pacheco, our textbook, as stated in the syllabus. However, the example used in this chapter is irrelevant and artificial on clusters with switched networks such as ours, and Pacheco does not give any numerical results. Compared to that, the Poisson problem in the tech. report HPCF–2010–2 is a classical example for using advanced point-to-point communcation commands. The immediate goal of this project is to reproduce (some of) the numerical results in tech. report HPCF–2010–2, and the eventual goal is to compare the performance of blocking and non-blocking point-to-point communication commands.

Problem and Basic Task The classical problem presented in the tech. report HPCF–2010–2 is the Poisson equation with Dirichlet boundary conditions in two spatial dimensions. This problem is discretized by the finite difference method using the conventional five-point stencil, resulting in a system of linear equations. We use the famous conjugate gradient (CG) method to solve the linear system.

So, your *basic task* is to implement the CG method for the Poisson problem. We want to we use a matrix-free implementation of the algorithm, that is, the system matrix will not be set up in memory. Rather, you need to program a function that computes the result of the matrix-vector product of the system matrix with an input vector. Implementing the CG method is then not a difficult task by using this function and some of the utility functions from the previous homeworks. We will discuss more details on these methods and issues in class. Use a one-dimensional split of the two-dimensional domain, as they suggest. Note that the idea is to compare the communication commands; we will discuss some choices that I believe make things simpler in regard to data structures.

To accomplish this *basic task*, I suggest that you stick with the blocking `MPI_Send`/`MPI_Recv` that you are familiar with. To ensure that there is no 'blocking,' I recommend that you have all even-numbered processes (`id%2 == 0`) send first and then receive, and vice versa for all odd-numbered processes.

Part of this *basic task* is to ensure that the solution is correct (correct-looking plot for some moderately fine grid) and performs as predicted by theory (compute and print proper norm as well as numerical performance parameters like iteration count, etc.). You will have to pick a suitable sample problem; you might want to use my technical report HPCF–2010–2.

Homework 5: I *highly urge* you to program a serial version of this code first. Clearly, you should do this in the framework of a parallel code (i.e., many parts of your code would be parallel by having `MPI_Init`, etc. and you can use `MPI_Wtime` to time), but *program and test* convergence of the finite difference method with a *serial matrix-vector product* first and make sure that this code behaves like your Matlab test from previous homework!

Advanced Tasks and Requested Guidance Once your code works flawlessly, replace the basic blocking `MPI_Send/MPI_Recv` commands with non-blocking `MPI_Isend/MPI_Irecv` commands; you may also use other available functions; see Chapter 13 or Appendix A in Pacheco. Describe how you implement your code. The *advanced tasks* include running and timing all versions of your code for several problem sizes, up to the largest problem you can solve. Design sensible tables to summarize all results for each method of communications and provide speedup and efficiency plots. Calculate and report the predicte memory usage for your code to determine the largest problem you can solve.

Your *final task* is provide guidance to the reader on the following question: **Which communication command(s) would you recommend to use on our system for a problem of this type?** Specify the type of problem that your guidance applies to. State your recommendation as part of the abstract of your report as well as explain it in the Introduction; you should base this on a summary table of results, derived from the results for the individual methods, that compares them and explains your recommendation.

Homework 6 is a report that contains all figures and tables; we will discuss these for feedback on correctness as well as on effectiveness of presentation; this homework should also contain a test of correctness by a convergence study of the finite difference method, but this time using your final parallel code.

Homework 7 is the full report in final form with all studies explained fully and accompanied by publication-grade figures and tables.

Some Other Notes, Hints, and Suggestions Your report should read like a paper. That is, fully explain what problem you are solving, specify the method of solution with suitable references (which you can use to avoid stating all details on background material on the problem and its numerical method). Explain in detail those aspects of your implementation that are material to parallel computing (example: the split of the domain is crucial to the later timings), then provide sufficient detail on how others can reproduce your results (i.e., specify all parameters used in your method), discuss and explain your results, before drawing a conclusion on the question on which guidance was requested above. Provide a detailed description of those components of the hardware you use that are relevant. Provide some specifics on software used and their version numbers.

Some details that might come up and should be mentioned in your paper are the following: Does your code assume that the number of processes divide some quantity? — How did you perform the timings? Which part of your code exactly is timed and how? — Consider all available mathematical and numerical theory to decide which quantities to compute (you might need to write some auxiliary routine, e.g., computation of the function norm of the error here) and print out. This allows you to debug your code and present the solution once at the beginning, before focusing on the numerical or parallel performance, respectively. — And for purposes of a self-contained report, specify the hardware used as well as all relevant software used (e.g., operating system, compiler, libraries; including version numbers where relevant);

Formal Note on Code Submission Submit your code to me in the form of a tgz-file. In order to ensure that I can distinguish the directories from different students, please incorporate your name into the name of the directory. Include a README file that lists the files that make up your code; list which functions are contained in each file and what their purpose is. Include basic usage instructions for your code.