Home Exam In Parallel Computing

SIVAN TOLEDO

February 2002

Exam Rules

You must work alone in this exam.

You are not allowed to discuss this exam with anybody except me until the due date is over (even if you submit early, you must not discuss the exam until after everybody else has submitted their solutions).

You may ask me questions.

You must use only the provided code and code that you write yourself.

The solution is due no later than Sunday, April 7, at 17:00, in my mailbox, on paper (no disks or submission by email). Leave the source files in the directory `~/ParallelComputing/HomeExam` with permissions that allow me to read them. Do not touch them after this date; I may check the modification time to ensure that you completed the exam on time.

Solving the exam should not take more than one week of full-time work.

Introduction

Your task in this home exam is to ***optimize*** and ***parallelize*** a sequential code that I provide. The code is in C (except for two Fortran subroutines that you do not need to touch). You should parallelize it using Cilk.

All the code is in a single directory, `/users/courses/368-4064/exam`. It includes two Fortran files, several C files, and one header file. It also includes a makefile. In principle, you should only need to modify the files `direct.c`, which is the main program, and `taucs_sn_llt.c`, in which the algorithm that you need to optimize and parallelize is. The changes to the main program

should be fairly minimal, only to allow it to call a Cilk function and perhaps to make additional performance measurements. Input data files for the exam are stored in another directory `/users/courses/368-4064/data`. Some of these input files may be quite large (so they may cause paging; don't try to optimize paging, concentrate on the cases where no paging occurs). You have to `cd` to these directory to see them.

Copy the source files into a directory and type `make`. The build process should build a library and a binary called `direct`. You can run this binary with a single argument, the name of a file that contains a matrix A. The program reads A, generates a random vector x, computes $b = Ax$, and then solves the linear system $A\hat{x} = b$ for $\hat{x}$. The program then compares x to $\hat{x}$ and computes and prints the size of the residual $A\hat{x} - b$, to ensure that the linear system has been accurately solved.

The function `taucs_ccs_factor_llt_mf` in the file `taucs_sn_llt.c` runs the algorithm that you need to optimize and parallelize. This function accepts a sparse symmetric positive-definite matrix A as input and computes its sparse Cholesky factor L, such that $A = LL^T$. The factor L is lower triangular. You do not have to understand the numerical details of this algorithm to solve the exam. You do need to understand which functions call which other functions and how function arguments are used.

Essentially, this function calls two other functions, `taucs_ccs_symbolic_elimination`, and `recursive_mf_factor`. You should only modify the second function and the function that it calls. Do not touch `taucs_ccs_symbolic_elimination`. The second function, the one that you should optimize, is recursive. It receives as one of its argument a structure that contains a representation of a rooted tree (using a `first_child` and `next_child` integer vectors). Another argument is a vertex v in the tree. The subroutine first calls itself on each child of v. After each recursive call returns, it calls a subroutine, `extend_add`, to merge data from the child with data for v. Once all the children have been processed, it calls a subroutine to process data associated with v itself, `front_factor`. This last subroutine, `front_factor`, calls three subroutines that perform operations on dense matrices associated with v.

What You Should Do

Make `recursive_mf_factor` as fast as you can.

You should focus on three issues:

- Its sequential performance relative to the performance of the original code (that is, how fast it is on a single processor).

- The actual running times on two processors under Cilk.
- The expected running times on machines with more processors.

These three objectives may conflict with each other and you have to use your judgment in order to decide what to do.

Avoid introducing new bugs. In particular, make sure your modified program continues to produce accurate results.

How to Submit the Solution

Your submission should consist of the following parts:

- A short description of what you have achieved (in terms of performance) and why you think that these results are good. This is where you can explain your judgments regarding the three objectives that I listed. At most half a page.
- A short description of the changes that you made to the code. At most half a page.
- A list of at most 3 of the most interesting tricks, techniques, or difficulties that you had to use to solve the exam. Do not include difficulties in installing the code, getting it to compile and run, or finding unused workstations. At most half a page each.
- Performance results. Use only graphs, charts, etc. ***Absolutely no numbers in tables or in text.*** You may add up to 5 lines of text to each graph to explain what it shows and what is the significance of these results. Convince me that you have done a good job. Convince me that you are not hiding any poor results. You must use the `plab-*` machines for the experiments.

Please obey the limits on text length. If you write more, I will not read your entire submission (with the obvious consequences for your grade). Use fonts 11 points or larger and reasonable margins.

Don't forget to leave your sources as instructed in the Introduction.

That's it. Good luck and enjoy the exam.