HPCE : Coursework 1

February 11, 2013

Issue date: Mon 4th Feb 2013.
Submission date: Fri 1st Mar 2013.

1 Overview

The first coursework will use Threaded Building Blocks to accelerate three algorithms. The assessment will be based on your uploaded code, and a ten minute oral.

2 Deliverables

This course-work relies on a set of source files which is available from the course web-site as either a zip file or a tarball:

`http://cas.ee.ic.ac.uk/people/dt10/teaching/2012/hpce/cw/hpce_cw1_sources.zip`

`http://cas.ee.ic.ac.uk/people/dt10/teaching/2012/hpce/cw/hpce_cw1_sources.tar.gz`

The **src** directory contains code for three algorithms:

- **mat_mat_mul** : Multiply two $n \times n$ matrices using recursive matrix partitioning.
- **fft** : Fast-Fourier Transform of a length n vector.
- **graph_distance** : Given a size n directed cyclic graph and a starting node, find the minimum number of edge traversals required to reach all other nodes.

Each algorithm X has two parts: a header file "X.hpp", which contains the function "X", and any other helper functions; and a source file "X_run.cpp" which can generate input data of size n and run the function. We are only interested in the run-time of the function "X", and the time taken to generate test data is not important. Your goal is to develop TBB accelerated versions of each of the algorithms, resulting in algorithms which are both efficient and scale to multiple processors.

For each algorithm X ∈ {mat_mat_mul,fft,graph_distance} you must:

1. Develop a basic TBB enabled version called "X_tbb' with *exactly* the same signature (function prototype) as X, located in a file called "X_tbb.hpp". This version should be a straightforward application of TBB constructs, introducing parallelism without attempting to modify or optimise the structure of the algorithm too much.

2. Optimise and tune your TBB version to produce another function called "X_opt" located in a file called "X_opt.hpp". In this version you should apply any techniques you think appropriate to maximise scalability and performance on a multi-core CPU.

3. Strip out any parallelism related constructs from "X_opt" to produce a serial elision called "X_seq", located in a file called "X_seq.hpp".

4. Create a driver program "X_time.cpp", which for a given function takes exactly two command line arguments. The first argument should be the problem size (n), and the second argument should specify how many processors TBB should use (or -1 for automatic selection). The output should be four lines, each of which is an accurate estimate of the wall-clock time for one execution with problem size n. The first line is for the original program, the next is for the "tbb" version, the next is for the "opt" version, and the final for the serial elision.

You are free to create other files, and may choose to include them if you want to talk about them in the oral, or they show good engineering practise, but the listed files must exist.

Your code *will* be compiled and executed by me, so it must fit the filenames and signatures precisely. [1] Feel free to break things into header files, but your code must compile in a standard C++ compiler, and if I include "X_tbb.hpp" then that should bring in all other header files needed to compile the "X_tbb" function.

Do not use any platform specific headers or libraries, as I may be compiling this on linux, windows, or something else. The only thing you need to know is that it is C++ compliant, and the TBB library and headers are available. [2] Also note that the compiler I am using is not necessarily C++11 compliant, so *don't* use lambdas - for those who haven't encountered them, they are the "[=]" syntax that occurs in some TBB examples. Source code does not need to be beautiful, but it should be consistently formatted and readable, and anything non-obvious or interesting should be commented.

After your code is submitted, you will each have a ten minute oral. The first five minutes will be based around a presentation, which should be *no longer* than five minutes long, followed by questions about your code and solution. During your presentation you should briefly explain your approach to parallelising the

[1] Otherwise I will have to spend time renaming everything, and you will lose marks.

[2] Unfortunately this rules out SSE optimisations, as there is no easy way of doing it portably.

algorithms, and also how well you think it worked (ideally backed up by quantitative data). Focus on information I wouldn't get from the code (methodology, high-level design, evaluation).

Your presentation must be uploaded as a pdf in the same zip you use to submit the source code, and you will be required to use the uploaded pdf during your presentation. [3] Name your presentation "LOGIN.pdf", where LOGIN is your login name (so that I can manage them during presentations). Remember that five minutes is not a long time - if you have more than ten slides you probably have way too many.

When complete, zip up all the code (including both the original files and the files you created) along with your presentation pdf, and upload it at: `https://intranet.ee.ic.ac.uk/scripts2/UploadCW.aspx`

3 Marks Schema

This coursework will be assessed as:

10% : Deliverables : Were all the deliverables correctly named, with the right function signatures, as the right type of file, etc.

25% : Correctness : Does the solution work? Does it try to use multiple processors? Is the code well written and commented?

25% : Scalability/Performance : Do the solutions scale with increasing numbers of processors? Have the solutions been well tuned?

20% : Presentation : Did the presentation explain how the parallelisation worked (or didn't work)? Was the evaluation objective?

20% : Questions : Can you explain the results you see on your machine? Can you explain the results seen on my machines? Do you understand how your code works? How could your solution work better?

4 Clarifications and Additional Details

There will inevitably be clarifications needed in the spec, or possibly bugs in my source programs. If you find inconsistencies or errors in this spec or the code, then email me at `dt10@imperial.ac.uk`. As they arise I will document them here, and upload updated code+spec to the web-site.

4.1 2013/02/11 : Cross-platform functions and Makefiles

Someone asked in the lecture about being cross-platform, and needing to use timers. I grudgingly said that '#ifdef WIN32' was ok, and that I would call makefiles if they existed.

[3] This is so that people who end up with later orals don't have an advantage over those who go first.

I have since changed my mind - I will simply remove a (very minor) challenge you faced, which is to notice that TBB includes a cross-platform timer class called "tbb::timer". This makes for cleaner submissions, as no extra libraries are needed beyond TBB, and so I don't need to worry about sometimes calling makefiles. Note that you still need to work out how to use "tbb::timer" accurately...

4.2 2013/02/10 : Signatures of functions

By function signature, I mean the function prototype. So for example, for "mat_mat_mul" the three functions you create should be:

- "void mat_mat_mul_tbb (mat_t dst, const mat_t a, const mat_t b)" in the file "mat_mat_mul_tbb.hpp".
- "void mat_mat_mul_opt(mat_t dst, const mat_t a, const mat_t b)" in the file "mat_mat_mul_opt.hpp".
- "void mat_mat_mul_seq(mat_t dst, const mat_t a, const mat_t b)" in the file "mat_mat_mul_seq.hpp".

and similarly for "graph_distance" and "fft".

4.3 2013/02/05 : Updated Submission Date

The original submission date was supposed to be 25th of Feb, but on the website I accidentally put the fall-back date of 1st Mar, which was for if a significant fraction of students have problems or there is a major technical hiccough.

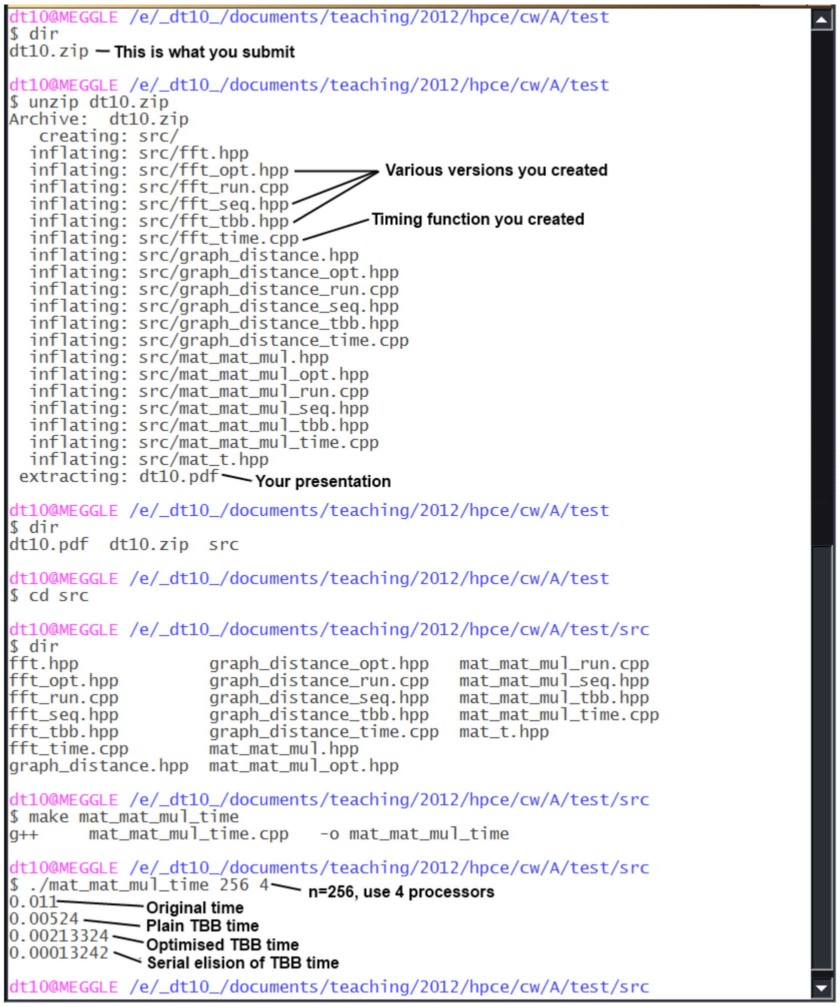
It was there for long enough that a number of people noticed it (not such a bad thing it means people read the website), so to avoid ambiguity I've reset the whole thing to 1st Mar. That means a little longer to do it, but it also means there is now no chance for extensions on this part. The GPU coursework will be issued before this coursework is finished, but the amount of time for it is still the same.

4.4 Including extra source files

Feel free to break things up into seperate headers if it makes things clearer, and create additional .cpp files for testing if you want. However, your extra files should follow some sort of naming convention (don't call them test1.cpp), and make sure that your submission still meets the requirements for files that I want. Also make sure that if I "#include" one of the required header files (e.g. "fft_tbb.hpp"), then I should not need to link in any other C++ source files or libraries (apart from TBB).

4.5 Zip file for submission

When your zip file is unzipped it should produce a directory structure that is the same as the original coursework files. The various header files for algorithm implementations (e.g. ”src/fft_tbb.hpp”) should appear alongside the originals, and it should be possible to compile from within the source directory. So assuming that I have set things up to provide TBB headers and library paths automatically, then *my* session with your submission might look like Figure 1.



```
dt10@MEGGLE /e/_dt10_/documents/teaching/2012/hpce/cw/A/test
$ dir
dt10.zip

dt10@MEGGLE /e/_dt10_/documents/teaching/2012/hpce/cw/A/test
$ unzip dt10.zip
Archive:  dt10.zip
   creating: src/
  inflating: src/fft.hpp
  inflating: src/fft_opt.hpp
  inflating: src/fft_run.cpp
  inflating: src/fft_seq.hpp
  inflating: src/fft_tbb.hpp
  inflating: src/fft_time.cpp
  inflating: src/graph_distance.hpp
  inflating: src/graph_distance_opt.hpp
  inflating: src/graph_distance_run.cpp
  inflating: src/graph_distance_seq.hpp
  inflating: src/graph_distance_tbb.hpp
  inflating: src/graph_distance_time.cpp
  inflating: src/mat_mat_mul.hpp
  inflating: src/mat_mat_mul_opt.hpp
  inflating: src/mat_mat_mul_run.cpp
  inflating: src/mat_mat_mul_seq.hpp
  inflating: src/mat_mat_mul_tbb.hpp
  inflating: src/mat_mat_mul_time.cpp
  inflating: src/mat_t.hpp
 extracting: dt10.pdf

dt10@MEGGLE /e/_dt10_/documents/teaching/2012/hpce/cw/A/test
$ dir
dt10.pdf  dt10.zip  src

dt10@MEGGLE /e/_dt10_/documents/teaching/2012/hpce/cw/A/test
$ cd src

dt10@MEGGLE /e/_dt10_/documents/teaching/2012/hpce/cw/A/test/src
$ dir
fft.hpp             graph_distance_opt.hpp   mat_mat_mul_run.cpp
fft_opt.hpp         graph_distance_run.cpp   mat_mat_mul_seq.hpp
fft_run.cpp         graph_distance_seq.hpp   mat_mat_mul_tbb.hpp
fft_seq.hpp         graph_distance_tbb.hpp   mat_mat_mul_time.cpp
fft_tbb.hpp         graph_distance_time.cpp  mat_t.hpp
fft_time.cpp        mat_mat_mul.hpp
graph_distance.hpp  mat_mat_mul_opt.hpp

dt10@MEGGLE /e/_dt10_/documents/teaching/2012/hpce/cw/A/test/src
$ make mat_mat_mul_time
g++     mat_mat_mul_time.cpp   -o mat_mat_mul_time

dt10@MEGGLE /e/_dt10_/documents/teaching/2012/hpce/cw/A/test/src
$ ./mat_mat_mul_time 256 4
0.011
0.00524
0.00213324
0.00013242
dt10@MEGGLE /e/_dt10_/documents/teaching/2012/hpce/cw/A/test/src
```

Figure 1: Example interaction with your zip file. Note that the output numbers are made up, and have no relationship to what you should expect.