CS4204 – Concurrency and Multi-core Architectures

Practical 2: Parallel Patterns

Chris Brown
cmb21@st-andrews.ac.uk

2019

Weighting: 50% of coursework

Deadline: 24th April 2019 at 21:00

Purpose

This practical will help develop your skills in:

- Implementing parallel patterns
- Implementing queues
- Use of PThreads to create parallelism
- Evaluation and experimentation of parallel execution

Overview and Background

You are a newly employed software developer of *ParallelSoft Ltd.*, who specialise in writing parallel software. You have been employed to develop a new parallel pattern library for C using PThreads called *para-pat*.

You are required to design the library so that it is easily callable by the user, with well-defined interfaces. The user must be able to use the library to parallelise their C code. This might require passing a function pointer to a worker function, for example. You will need to make use of PThreads, locking mechanisms and queues to implement the parallel pattern library, and provide a high-level interface so that the pattern is easily callable by passing in some parameters. *Providing an implementation that simply wraps an already existing parallel framework, such as FastFlow or TBB will not be permitted.*

`parpat.c` contains some stubs of functions you should complete. You will need to create your own functions around these that deal with, for example, the locking of the queues, getting and putting tasks, etc.

para-pat will be sold to customers as a library that they can interface with in their programs. Potential customers should be able to call the library with

```
#include "parapat.h"
```

And compile their code with:

```
gcc mycode.c -lpthread -lparapat -o mycode-exe
```

Tasks

1. (Easy) Implement the parallel farm pattern. The farm must allow the user to specify the number of workers and the worker function to be executed in parallel.
2. (Medium) Implement the parallel pipeline pattern. The pipeline must allow the user to declare a number of stages that themselves call user-defined pipeline functions.
3. (Hard) Implement nested parallel patterns. Modify your farm and pipeline patterns so that they can themselves take patterns. For example, a farm may take a pipeline as a worker; or, similarly, a pipeline may itself take instances of farms as workers.
4. Evaluate your parallel patterns against the supplied examples. Record your performance results for different configurations or nesting of patterns if applicable to the example. Explain in your report which configuration gives the best results and why.

Extension

5. (Extra hard) *Extension*: Implement further parallel patterns that you think useful.

Submission

You are required to hand in, on MMS by **9pm on the 24th April**, a zip file containing the sources of your *para-pat* implementations, your parallelised examples and a report of no more than 1500 words. The submission should contain clear and easy instructions on how to install, compile and run your implementations.

In your report, you must explain your key design decisions, explain how your library can be used, and how it fulfils the above tasks. The report must include an evaluation section, which shows the performance results of your implementations against the examples.

Marking

I am looking for:

- Good design and understandable code, which is well documented, with major design decisions explained;
- A study of the performance characteristics of your parallel pattern implementations;

- A report showing an evaluation section giving the trade-offs of different pattern configurations.

The standard mark descriptors in the School Student Handbook will apply:
https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors
Completing all the easy questions with a basic report will allow you to gain a mark of 12. Completing all the easy and medium questions with a good report will allow you to gain a mark of 17. To get over a 17, you will need to implement the hard questions and attempt all or some of the extra hard questions.

Lateness

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good Academic Practice

The University policy on Good Academic Practice applies:
https://www.st-andrews.ac.uk/students/rules/academicpractice/