University of St Andrews

CS4204 Coursework 1

Parallel Patterns

Author:
150008022

April 26, 2019

Goal

To implement and evaluate a library for parallelising C programs, using PThreads, locks, and queues.

1 Blocking Queue

The implementation of parallel task handlers would have to accumulate their outputs and buffer their inputs in a thread safe way, and so a blocking queue was necessary. queue.h provides an interface for a queue to be used by other components, and queue.c provides an implementation.

The queue is a linked list structure, where each node in the list has a reference to the element next in line. The queue structure tracks the last in line for adding new nodes, and the first in line for removing nodes. A conditional variable is used to notify threads waiting to add or remove elements that the size of the queue has changed, and a normal mutex is used to ensure that only one node is able to perform the addition or removal at a time. Figure 1 shows a queue struct with a series of node structs.

For convenience, the destroyer function for the queue also takes the destroyer function for its elements as a parameter so that it can be assumed that destroying a queue will free all memory that it is responsible for.
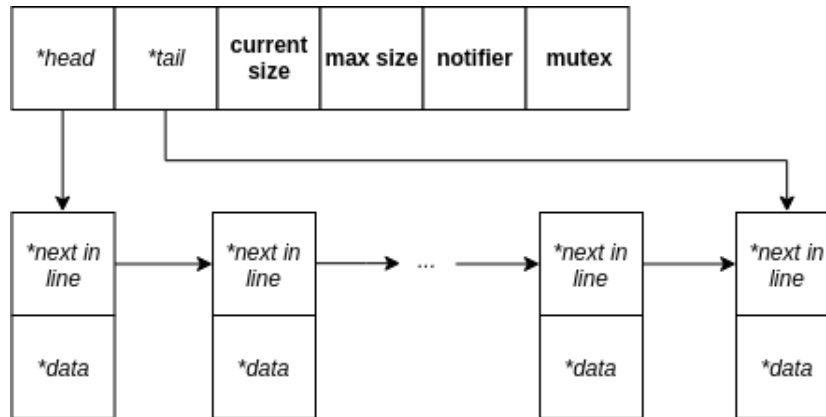

Figure 1: Queue structure, showing head and tail pointers, and each member of queue referencing the next.

2   Pipeline

The pipeline pattern is where a series of functions are applied to some input. Often the analogy of a conveyor belt is used, as inputs can flow continuously and functions can be applied simultaneously to inputs that are at differenet stages of the pipeline.

Figure 2 shows how an atomic pipeline was implemented for this submission. An array of functions are submitted on creation of the pipeline, along with the number of worker threads that should run for each stage in the pipeline. The pipeline interface allows for inputs to be added to a queue, where they will wait to be processed, and for outputs to be polled from the outgoing queue.
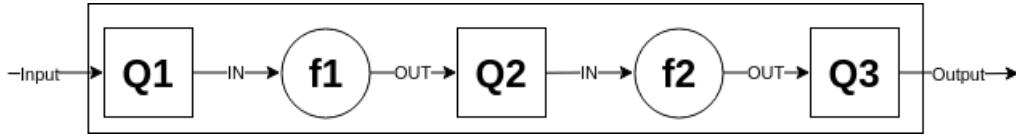

Figure 2: Pipeline abstraction, where Q signifies a blocking queue, and f signifigies a function being applied.

The pipeline was defined to consist of a series of steps which would be managed by a thread. The thread for step i would poll queue Q_i for an input x to process, compute $f(x)$, then add the output to queue Q_{i+1}. Q_0 and Q_{n+1} are the input and output queues made accessible by the pipeline interface, where n is the number of functions in the pipeline.
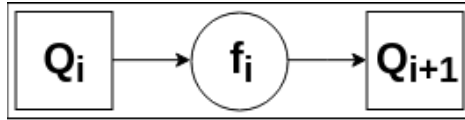

Figure 3: Step of pipeline.

3   Parallel Farm

The parallel farm pattern involves a pool of worker threads that remain idle until assigned a task by a coordinator. Once a worker thread completes its

task, it then returns to being idle until a new task is assigned. Parallel farms are useful as they avoid the overhead involved .
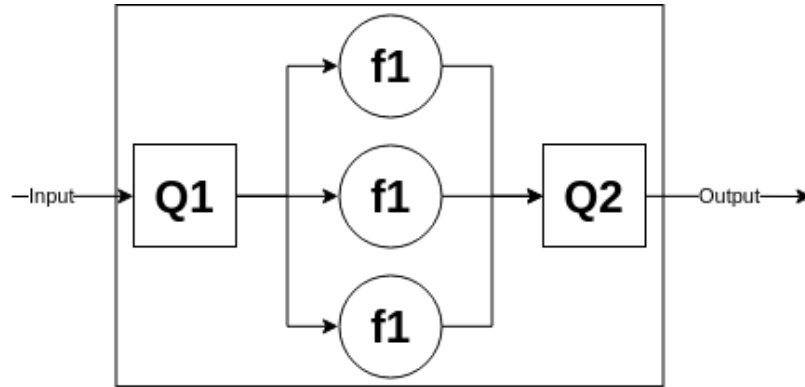

Figure 4: Farm abstraction

Conclusion