IDs: 312497084, 315370551

Operating Systems – Exercise 2

Summary

In this exercise, we have developed a dispatcher-worker system, which utilizes pthreads system calls to create an effective producer-consumer model.

System High Level Design

Our System consists of the following modules:

- Dispatcher Module
- Workers Module which consists of the Queue module
- Statistics Module

The Entire System can be described using the following chart:

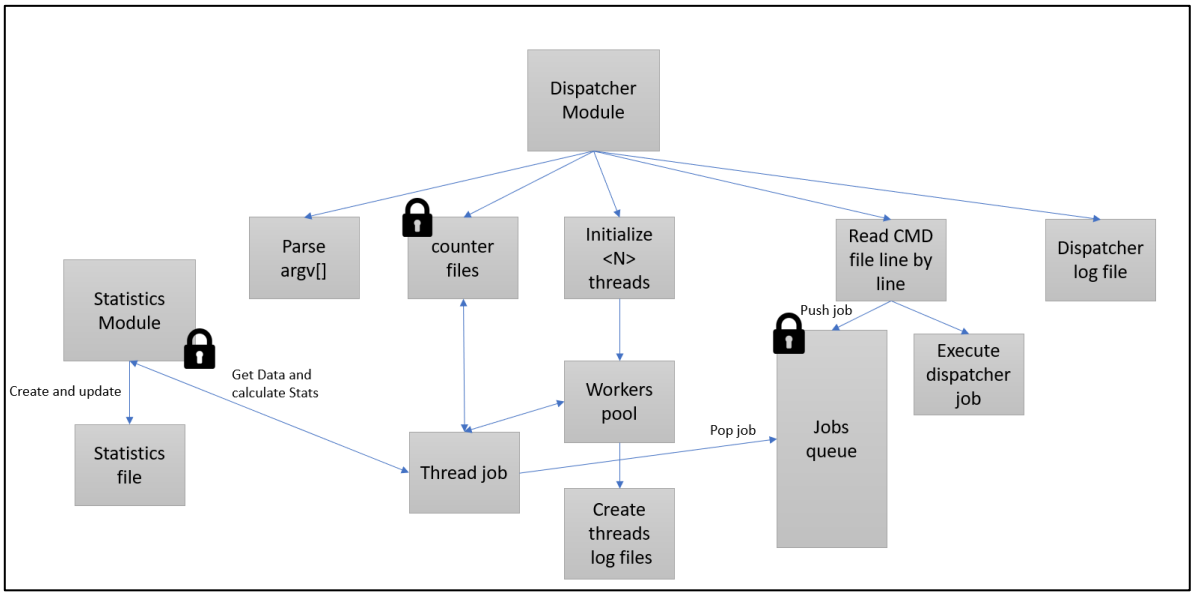

Figure 1 - System High Level Overview

Build and Clean

One can build the system using: *make*

Usage for running the program:

$$hw2 <cmd\_file> <num\_threads> <num\_counters> <trace\_enabled>$$

to clean all files and binaries, one can use: make clean.

System Key elements

We have decided to build the most efficient system possible, using the following measurement:

$$Efficiency = Max\left\{\frac{jobs}{sec}\right\}$$

Moreover, since there are several common resources in this system it was important to keep the access to them safe, to make sure data doesn't get overwritten or miscalculated.

We used the following mutexes to keep all resources synchronized and safe:

- mutexQueue:
  locks the access to the shared working queue. This way we can make sure the queue is indeed a working FIFO model, and that every job is being taken once.
  This mutex is working alongside with a condition variable to make sure all workers that don't have jobs to execute are sleeping and not wasting system CPU time.

- mutexCounters[MAX_NUM_OF_COUNTERS]:
  Since multiple threads can "work" on the same counter file at the same time, we wanted to make sure the access to each counter file is protected, and that the overall calculation is correct.
  Since there are several counter files (according to the user argument when running the software) we decided to use a different mutex for every counter file, to achieve the maximum efficiency: working on the specific counter file should not lock access for all other files.

- mutexStats:
  When a worker finishes its job, it reports back to the statistic module their job execution time. We needed to keep this access safe in order to make sure to take all jobs execution time into account when calculating statistics.

- mutexLiveThreads:
  This mutex whole purpose is to protect a variable which counts all jobs executed in the system, to calculate the average execution time:

$$Avg\ exec\ time = \frac{Sum\ Execution\ time}{\#Executed\ jobs}$$