COMP 7500 Advanced Operating Systems
Project 3

Joshua Boyd
jcb0216@auburn.edu

March 14, 2023

Introduction

The purpose of this project was design and implement a batch scheduling system called AUbatch using the C programming language and the Pthread library.

Design

The design of the AUbatch system begins in the main thread. The main thread spawns three threads. The first thread is the scheduler. The scheduler waits and listens for the user to insert a job. Once the scheduler has a job, it inserts the job into the job queue and re-balances the queue based on a set policy. In order to decouple the policy from the scheduler thread, a scheduler structure was created to house the policy. The scheduler structure also keeps track of a pointer to a slot in the job queue, the current expected wait time for each job, and a cache memory for holding a job before inserting it into the queue.

The second thread is the dispatcher. The dispatcher waits for a job to appear the the job queue. When the dispatcher detects a job in the queue, it grabs the job and executes it. The dispatcher also keeps track of performance metrics. Similar to the scheduler, a dispatcher structure was created to keep track of where the dispatcher grabs from next in the job queue.

The third and final thread is the tester thread. The tester receives a test case from the user through the 'test' command. Using the test command, the tester generates jobs and sends them to the scheduler. The tester requires a thread so that the user can continue to monitor the queue, using the list command, as jobs are submitted from the test case. Once the test is complete, the tester will display metric values for the test to the user.

Data Flow

The direction of data flow can depend on whether the user submits a job to the scheduler or a test case to the tester. If the user submits a job to the scheduler, the job is placed in the schedulers job cache. If the user inserts a test case to the tester, the tester will take the test case and generate jobs. Each job generated is sent to the schedulers job cache. Once a job has reached the scheduler job cache, the scheduler inserts the job into the queue, and rebalances the queue based on the policy set within the scheduler structure.

The dispatcher waits and listens for a job to be in the job queue. Once the dispatcher has detected a job, it grabs the information for that job and executes the job using the 'execv' command. During this time, the dispatcher keeps track of performance metrics. When the job has completed, the dispatcher sends the metric updates to variables within the main thread. A data flow diagram can be found in figure 1 on page 4.

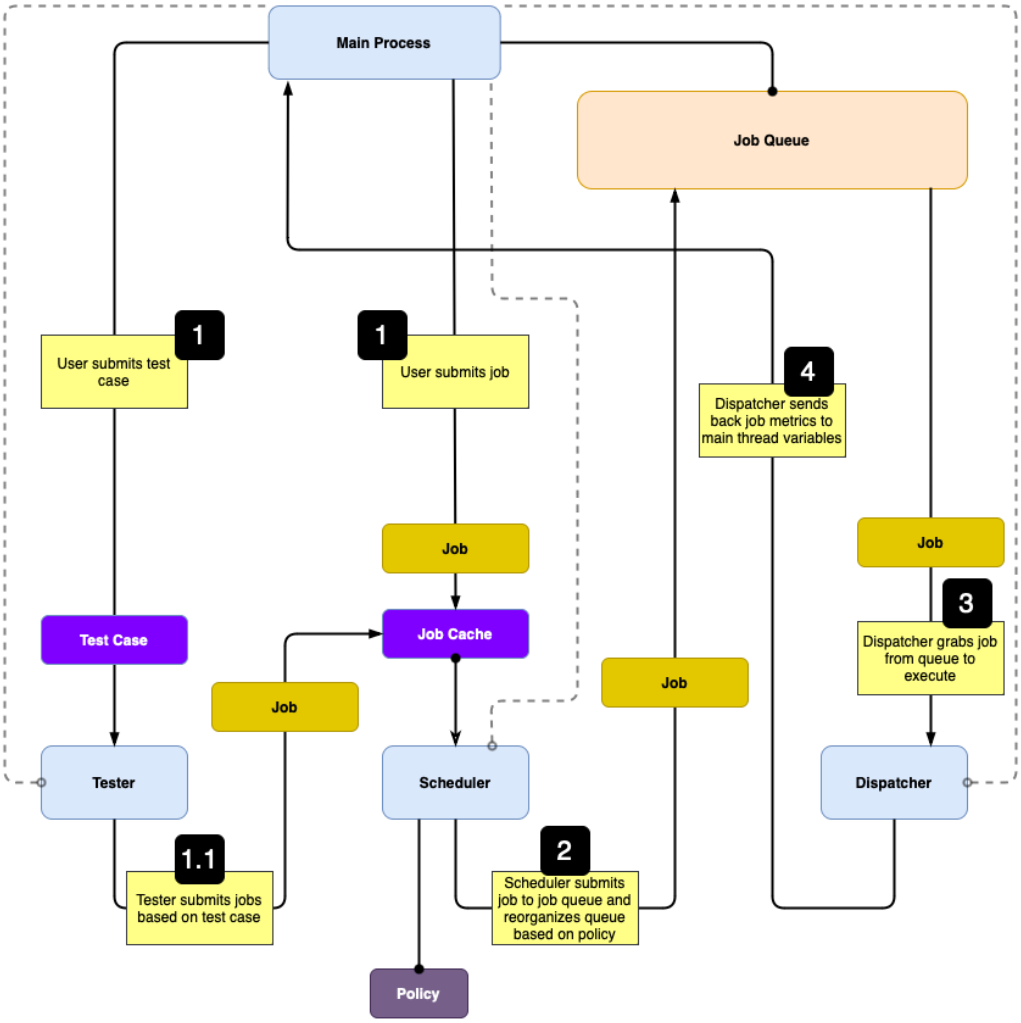

Figure 1: AUbatch Data Flow

Performance Metrics and Workload Conditions

There are four metrics utilized for performance evaluation. The first metric is average turnaround time. Turnaround time is calculated from difference between the arrival time of the job and the completion time of the job. The second metric is average CPU time. The CPU time is directly coordinated with the user CPU time inputted by the user, since this is the first parameter into the batch job program. The third performance metric is average wait time. Wait time is calculated as the difference between the arrival time of the job and the start time. The final performance metrics is the throughput, which is calculated as 1 over the average turnaround time.

The following workload condition was used to test performance.

Workload Parameters	Values
Number of Submitted Jobs	30
Arrival Rate	0.01
Priority Levels	15
Load Distribution	[1, 10]

Table 1: AUbatch Workload Condition

Performance Evaluation

FCFS

After running the workload condition with the First Come First Served policy, the metrics obtained are shown in table 2 on page 6. A screenshot of the performance evaluation for First Come First Served can be found in figure 2 on page 7.

Performance Metric	Value
Average Turnaround Time	72.30 seconds
Average CPU Time	5.70 seconds
Average Wait Time	66.60 seconds
Throughput	0.01 No./second

Table 2: First Come First Served Performance Metric Values.

```
>test batch_job fcfs 30 0.01 15 1 10
---- Test Metrics ----
Benchmark name: batch_job
Policy: fcfs
Number of Jobs: 30
Arrival rate: 0.010000
Priority Levels: 15
Min Cpu Time: 1
Max Cpu Time: 10

>Test starting...
Adding Jobs...
Feel free to add additional jobs or view jobs with the 'list' command
Finished Adding Jobs
list
Total number of jobs in the queue: 23
Name            CPU_TIME        PRI     Arrival_time    Progress
batch_jo        5               14      15:23:16        Running
batch_jo        7               4       15:23:17
batch_jo        6               6       15:23:18
batch_jo        10              15      15:23:19
batch_jo        9               10      15:23:20
batch_jo        3               14      15:23:21
batch_jo        1               14      15:23:22
batch_jo        8               13      15:23:23
batch_jo        4               13      15:23:24
batch_jo        4               7       15:23:25
batch_jo        10              15      15:23:26
batch_jo        10              9       15:23:27
batch_jo        4               12      15:23:28
batch_jo        3               14      15:23:29
batch_jo        3               11      15:23:30
batch_jo        8               13      15:23:31
batch_jo        6               6       15:23:32
batch_jo        6               11      15:23:33
batch_jo        8               1       15:23:34
batch_jo        8               6       15:23:35
batch_jo        7               5       15:23:36
batch_jo        2               10      15:23:37
batch_jo        5               4       15:23:38
Scheduling Policy: fcfs

>Total number of job submitted: 30
Average turnaround time: 72.30 seconds
Average CPU time: 5.70 seconds
Average wait time: 66.60 seconds
Throughput: 0.01 No./second
```


Figure 2: First Come First Served Performance Evaluation

SJF

After running the workload condition with the Shortest Job First policy, the metrics obtained are shown in table 3 on page 8. A screenshot of the performance evaluation for Shortest Job First can be found in figure 3 on page 9.

Performance Metric	Value
Average Turnaround Time	42.40 seconds
Average CPU Time	5.03 seconds
Average Wait Time	37.37 seconds
Throughput	0.02 No./second

Table 3: Shortest Job First Performance Metric Values.

```
>test batch_job sjf 30 0.01 15 1 10
---- Test Metrics ----
Benchmark name: batch_job
Policy: sjf
Number of Jobs: 30
Arrival rate: 0.010000
Priority Levels: 15
Min Cpu Time: 1
Max Cpu Time: 10

>Test starting...
Adding Jobs...
Feel free to add additional jobs or view jobs with the 'list' command
Finished Adding Jobs
list
Total number of jobs in the queue: 18
Name            CPU_TIME        PRI     Arrival_time    Progress
batch_jo        3               14      15:45:25        Running
batch_jo        3               15      15:45:29
batch_jo        5               2       15:45:09
batch_jo        5               14      15:45:10
batch_jo        5               2       15:45:12
batch_jo        6               4       15:45:02
batch_jo        6               14      15:45:07
batch_jo        6               5       15:45:08
batch_jo        6               2       15:45:21
batch_jo        6               5       15:45:15
batch_jo        7               7       15:45:11
batch_jo        7               13      15:45:19
batch_jo        8               6       15:45:14
batch_jo        9               8       15:45:13
batch_jo        9               5       15:45:30
batch_jo        10              15      15:45:17
batch_jo        10              11      15:45:26
batch_jo        10              1       15:45:24
Scheduling Policy: sjf

>Total number of job submitted: 30
Average turnaround time: 42.40 seconds
Average CPU time: 5.03 seconds
Average wait time: 37.37 seconds
Throughput: 0.02 No./second
```

Figure 3: Shortest Job First Performance Evaluation

Priority

After running the workload condition with the Priority policy, the metrics obtained are shown in table 4 on page 10. A screenshot of the performance evaluation for Priority can be found in figure 4 on page 11.

Performance Metric	Value
Average Turnaround Time	68.10 seconds
Average CPU Time	5.60 seconds
Average Wait Time	62.50 seconds
Throughput	0.01 No./second

Table 4: Priority Performance Metric Values.

```
>test batch_job priority 30 0.01 15 1 10
---- Test Metrics ----
Benchmark name: batch_job
Policy: priority
Number of Jobs: 30
Arrival rate: 0.010000
Priority Levels: 15
Min Cpu Time: 1
Max Cpu Time: 10

>Test starting...
Adding Jobs...
Feel free to add additional jobs or view jobs with the 'list' command
Finished Adding Jobs
list
Total number of jobs in the queue: 24
Name            CPU_TIME        PRI     Arrival_time    Progress
batch_jo        8               12      15:54:38        Running
batch_jo        3               11      15:54:47
batch_jo        3               11      15:54:41
batch_jo        2               11      15:54:55
batch_jo        1               10      15:54:49
batch_jo        2               9       15:54:33
batch_jo        9               9       15:54:48
batch_jo        9               9       15:54:43
batch_jo        7               9       15:54:45
batch_jo        10              9       15:54:46
batch_jo        1               8       15:54:31
batch_jo        10              8       15:54:34
batch_jo        4               7       15:54:39
batch_jo        4               7       15:54:29
batch_jo        7               7       15:54:52
batch_jo        6               7       15:54:27
batch_jo        3               6       15:54:54
batch_jo        8               6       15:54:53
batch_jo        10              5       15:54:35
batch_jo        8               3       15:54:36
batch_jo        3               3       15:54:44
batch_jo        7               1       15:54:30
batch_jo        9               1       15:54:51
batch_jo        2               1       15:54:37
Scheduling Policy: priority

>Total number of job submitted: 30
Average turnaround time: 68.10 seconds
Average CPU time: 5.60 seconds
Average wait time: 62.50 seconds
Throughput: 0.01 No./second
```


Figure 4: Priority Performance Evaluation

Lessons Learned

In comparing the three scheduling policies, it is clear that Shortest Job First provides the best average waiting time and average turnaround time between jobs. However, SJF can be difficult to implement in practice, due the inability to know the exact run time of future jobs. Priority based scheduling did a bit better than First Come First Served in this experiment. However, this may not always be the best case in practice if jobs with higer prioirty keep coming into the queue.

Additionally, this project gave me tons of practice using mutex and conditional locks between threads in C. By the end of the project I really felt that I had a firm control on when I wanted each thread to do an action. I accomplished this by having them wait when there was nothing to do and signal to them when it was time to perform. In the future I plan on using this new skill in my personal projects to achieve better parallelism.