COMP 7500/7506 Advanced Operating Systems

Project 3: AUbatch - A *Pthread*-based Batch Scheduling System

Frequently Asked Questions

Document History: Created on Feb. 19, 2018. Revised on March 12, 2020. **Version 2.0**

1. Should we use fork() or pthread_create() to create two threads?

Answer: Please use pthread_create() to create two threads (i.e., the scheduler and disptacher). Please check the sample code here:
http://timmurphy.org/2010/05/04/pthreads-in-c-a-minimal-working-example

2. Are we executing actual programs (e.g., hello_world) or simulating them?

Answer: You must actually execute the programs (a.k.a., benchmarks).

3. Do we ignore the actual execution time and use the execution time given by the run command?

Answer: The execution times should be measured on the fly based on actual execution times rather than the estimated ones submitted by users.

4. If you add a job with a higher priority and change the policy to priority, the running job will still remain the same, right?

Answer: Once your change your policy, all the jobs waiting in the queue must be rescheduled based on the new policy. The only job that should be affected by the new policy is the current running job (i.e., the one sitting in the head of the job queue).

5. Can I use any 3rd party class library?

Answer: You don't have to develop all the functions from the ground up. Please feel free to use any third-party libraries dealing with basic functions like data/times, queues, lists, file I/Os. It is important to give credits to the original authors of these libraries.

6. Why users must specify a job's CPU time priori to the job submission?

Answer: All the existing batching systems require users to provide estimated the CPU time for each submitted job. Schedulers will make scheduling decisions based on required amount of resources (e.g., CPU time, memory). Estimated CPU times should be close to the real measured ones. Prior to job submissions, users (i.e., you in this project) must estimate the CPU times of jobs to be submitted to AUbatch.

7. Should "quit" exit immediately or after all jobs have completed?

Answer: Ideally, you should choose to let the parent process wait for the dispatcher thread to finish executing all the pending jobs. A simple solution is "immediate termination" using the exit system call, which is demonstrated in the sample source code - commandline_parser.c

8. Do we include the running job's expected execution time to compute expected waiting times?

Answer: Because "list" can occur while a job is running, you must calculate the running job's remaining time to estimate the expected waiting times of all the other waiting jobs. After computing the expected time left (i.e., remaining_time) for the current running job, all the expected wait time for pending jobs can be easily evaluated. The remaining_time of the running job can be expressed as:

```
remaining_time = expected_exe_time - (current_time - start_time);
```

9. Can I test my program using built-in test command?

Answer: Yes. You may implement your performance evaluation module using the test command. Please feel free to change the command format when it comes to performance evaluation. You are encouraged to propose an improved user interface with respect to performance evaluation.

10. **Test Command.** For the Test command in Project 3, after entering the **test** command, should the user be able to interact with the program while the jobs are executing? Or, should the test run until it is complete, and then print out the report for the test? (Spring'20)

Answer: '`test`' is a command furnishing an automatic performance testing process; therefore, users don't interact with the AUBatch while jobs (a.k.a., microbenchmarks) are executing. All the jobs should be batching jobs rather than interactive jobs.

Users may issue "quit -i" to immediately terminate testing. Alternatively, the users may type "quit -d" to exit AUBatch after all the submitted jobs are completed.

11. **Test Command - Job Arrival Rate.** There is no parameter describing arrival rate for jobs. Should all the jobs be submitted immediately (at approximately the same time) or should there be some sort of delay to their submission. (Spring'20)

Answer: You raised an excellent question related to performance testing. The format of the test command is:

```
test <benchmark> <policy> <num_of_jobs> <priority_levels>
<min_CPU_time> <max_CPU_time>
```

Ideally, we should add an input parameter to specify the arrival rate. The updated format should be:

```
test <benchmark> <policy> <num_of_jobs> <arrival_rate>
<priority_levels> <min_CPU_time> <max_CPU_time>
```

The workload conditions can be specified as an input parameter list of the test command (see the above format).

12. **Test when the initial queue is non-empty.** If several jobs are already in the queue, should we still be able to run the test function knowing that the queue is not empty? (Spring'20)

Answer: When we issue the test function, we assume that the queue is initialized to be empty. If the queue is non-empty prior to running the test function, your system should report an error.

13. **Priority Settings.** In the test module do we assign priorities randomly to the jobs (if yes what distribution we use for randomly generating the priorities) or they all have to carry the same priorities? (Spring'20)

Answer: You may choose to assign priorities to the jobs or randomly assign priority levels based on a uniform distribution in a range.

14. **Micro Benchmarks.** Do these micro-benchmarks actually have to use the CPU (by doing garbage computations) or can a call to "sleep" or "usleep" be used to wait the appropriate amount of time? (Spring'20)

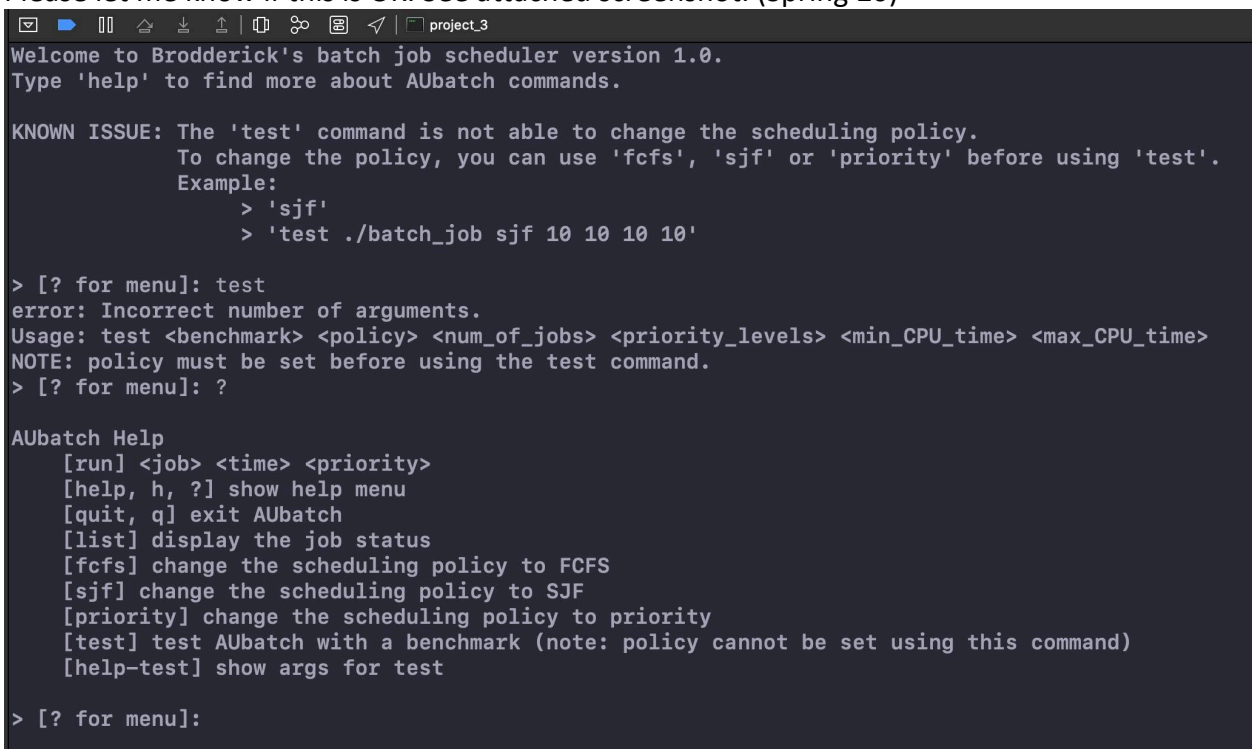
Answer: These micro benchmarks should consume CPU utilization (e.g., dealing with garbage computations). Please don't use "sleep" or "usleep" in the micro benchmarks.

15. **Slight UI change?** I know the assignment document says not to deviate from the user interface shown. However, I am having an issue where the 'test' command cannot change

the policy. To get around this, you can first change the policy using ‘fcfs’, ‘sjf’, or ‘priority’, then call ‘test’. I cannot figure out why this is happening. The project is complete besides this one issue. I plan on submitting it as is and document the issue.

I would like to add:
(1) a “known issue” statement to the “welcome” printout.
(2) add a note to the “correct use” of ’test’.
(3) add a note to the “help” menu.

Please let me know if this is OK. See attached screenshot. (Spring’20)



Answer: Your proposed user interface was well designed and; therefore, I endorse the new user interface developed by you.

16. **Extra Work.** If we wanted to implement extra metrics and also print each metric per job in addition to overall metrics when quitting would it be permissible? Additionally I have it right now so when you ask to list the jobs it also lists the jobs that have finished, just to show the user a complete view of jobs running, jobs waiting and jobs that have finished. I just wanted to know if it was okay to do the extra work, as I do not want to get docked for not having my output exactly like the specifications show? (Spring’20)

Answer: You are encouraged to do the extra work. It is a brilliant idea to list finished jobs in addition to pending jobs.

17. **File Names.** If have three files, aubatch.c (contain main and sets up the threads and all the condition variables), modules.c (contains the scheduler and dispatcher modules), and commandline.c (contains code that interacts with the users). Additionally, I have a header file for both commandline.c (commandline.h) and for modules.c (modules.h) to allow aubatch.c to call them. There is a note in the specifications that says the files must adhere to the file names specified. Is it fine if I have these files? (Spring'20)

Answer: These three source code files look good to me. I recommend you to change modules.c into scheduler.c, because modules.c sounds too generic.

18. **Waiting time.** After the run command is given and the job is inserted, we need to display the total waiting time. Should this be the sum of the execution time of jobs waiting in the queue only or the remaining execution time of the currently running process also needs to be added? (Spring'20)

Answer: A simple way of computing the expected waiting time of a newly submitted job is the summation of the execution time of pending jobs that are scheduled ahead of this submitted job. It might be hard to calculate the remaining execution time of the currently running job, so you may count the entire execution time of the running job toward the waiting time of the submitted job.

19. **Commandline Parser and Scheduler Threads.** My understanding is that we have two threads - dispatcher thread and the scheduler thread. Dispatcher thread will take the jobs from queue (bufTail) and send it to execv/system. And the scheduler thread will be continuously checking for user input and working accordingly. Now here is my question. A new job arrives and we call `run` command when our job_queue is full, we would use `cv_wait' on the scheduler thread (as in sample code). I.e. our scheduler thread is sleeping and user can no longer enter any more commands (`list`, `help` etc). Is this how it is supposed to be? (Spring'20)

Answer: If the buffer is full, the scheduler will sleep. If the scheduler and commander line parser are embedded within the same thread, you will be unable to enter another command until your scheduler is waked up by the executor.