**PROJECT 3 REPORT**

**Sujasri Vijaichandran**

**AU Batch Design:**

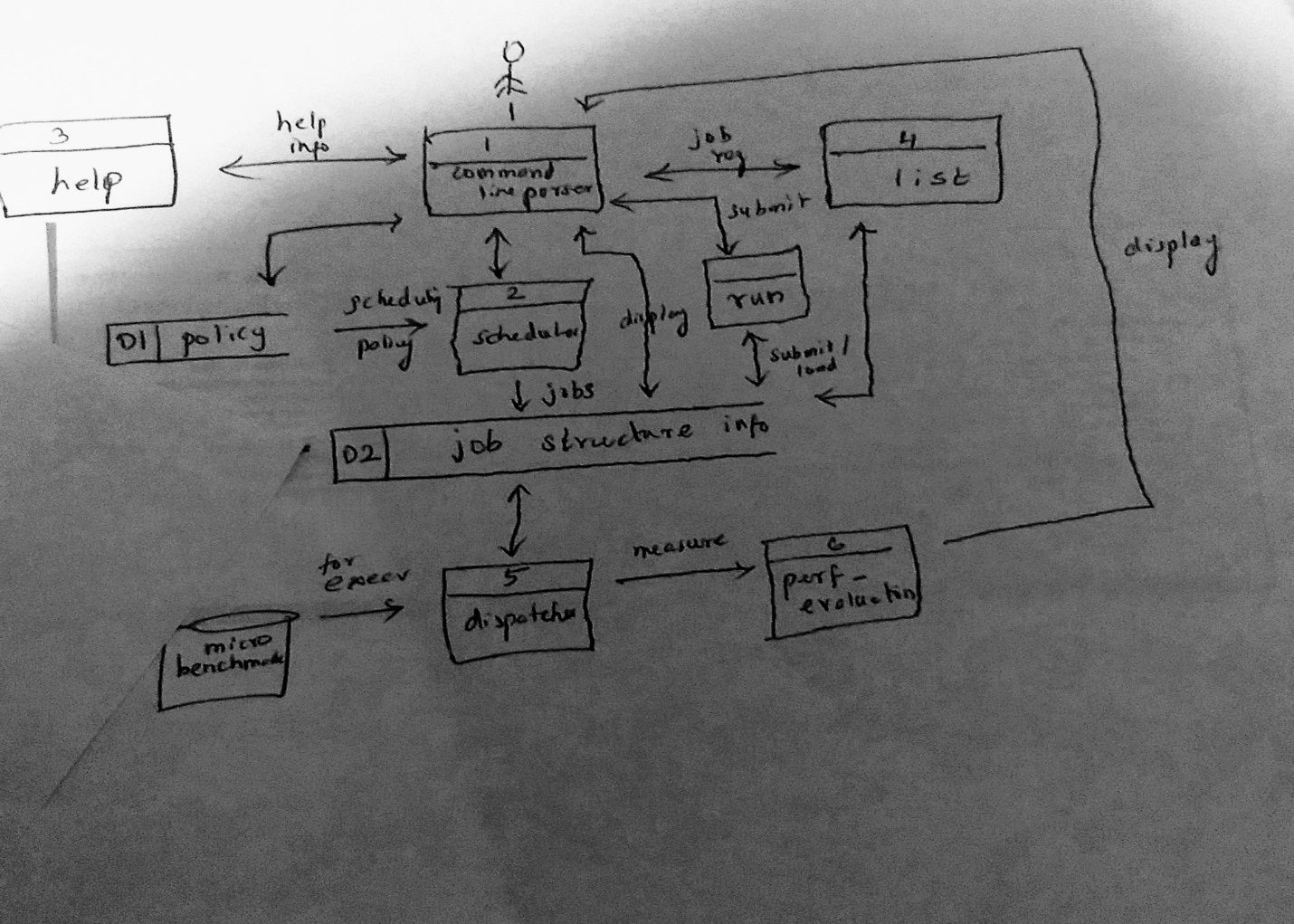


Fig 1: Data Flow Diagram

**AU Batch Implementation:**

‘AU Batch’ demonstrates a batching system. Here, jobs are submitted by the user and the system is designed to execute those jobs in an orderly fashion with two modules (scheduler and dispatcher), where the scheduler schedules the jobs input from the user (using the 3 scheduling algorithms as specified) and loads them into the buffer and the dispatcher takes the scheduled jobs present in the buffer and executes them. This system is made possible by the help of condition variables and mutexes, thus establishing synchronization. Also, performance evaluation is done for the jobs executed in the system.

First, two threads(being the scheduler and dispatcher) are created using the pthread library.

In scheduler, user interaction with the system is made possible through commands such as:

Help: Gives a list of commands that can be used to interact with the system and it says how to use them

Run: It is to submit the jobs into the buffer. It takes the 2 arguments to run the microbenchmark along with the burst time and priority of the job

List: Displays the submitted jobs and their details.

FCFS: It schedules the jobs in the buffer with FCFS policy

SJF: It schedules the jobs in the buffer with SJF policy

Priority: It schedules the jobs in the buffer with this policy

Quit : quits the system with the performance evaluated results.

In dispatcher, the jobs loaded into the buffer are taken one by one and executes the microbenchmark specified in the program, through the execv().

Both the threads make use of the buffer. So here comes the concept of mutexes. Whenever a module wants to access the buffer(an array of structure in the program),it attains the lock available in the pthread library and access the buffer. When it completes accessing the buffer, it releases the lock so that the other module can access the buffer. The two threads communicate through the condition variables accomplished through wait(..) and signal(..) system calls. Both the threads after execution, join in the main function.

**Performance metrics**:

In dispatcher, a function is called to evaluate the execution of jobs submitted by the user taking into account the arrival time, waiting time, the time at which the job gets executed and the completion time through timestamps, to calculate the average turn around time, average waiting time and throughput of the job execution.

Through the evaluation, it is seen that SJF contributes to the minimum average waiting time. FCFS will hold good with the jobs executed first have lesser burst time. Priority will make the less prior jobs wait for some time to get executed. If the high priority jobs have lesser burst time, then it holds good.

**Lessons from this project:**

Working of threads – Implementation of mutexes and condition variables – microbenchmarks

**Future work:**

Test module is to be implemented.

The automated performance evaluation is also to be implemented