COMP 7500 Project 3 Report Liangliang Xu

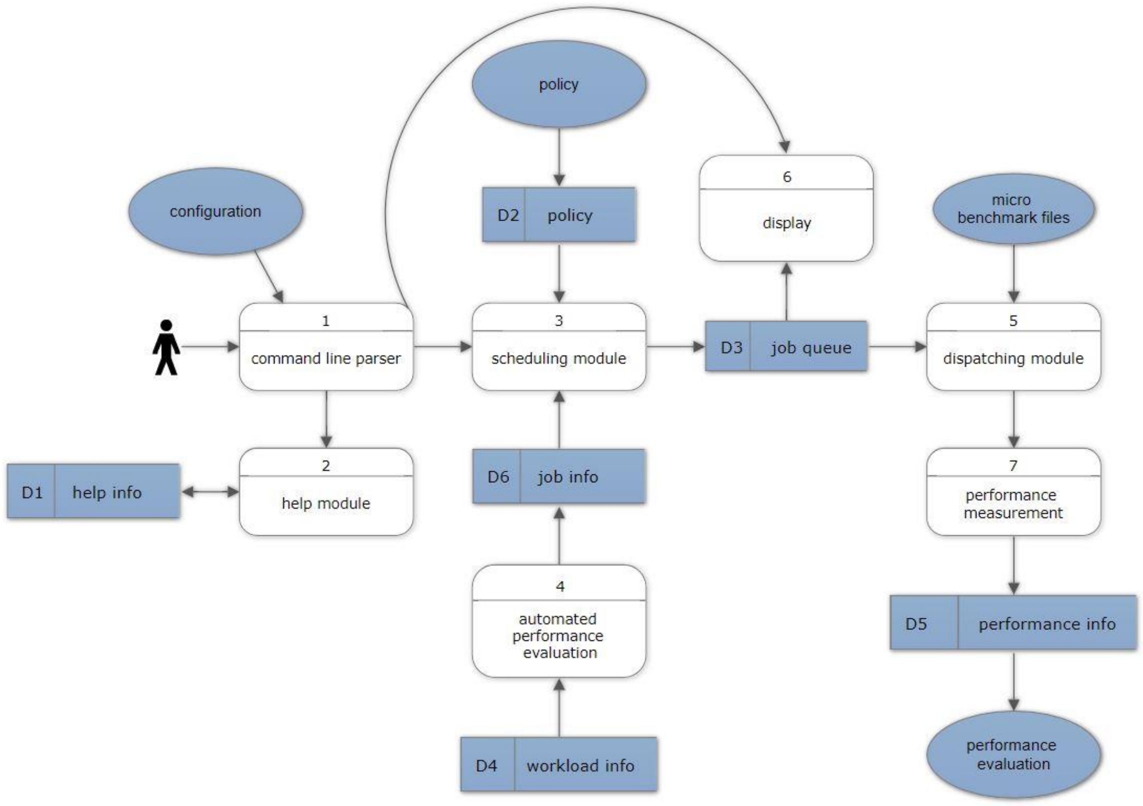
The goal of this project is to design and implement a batch scheduling system using pthread and execv. Implementation details are included in the source code. This project consists of three parts: design document, performance metrics and performance comparison.

1. Design document

Design document will have data flow diagram and program structure. These 2 diagrams gives a clear and intuitive understandability on logic behind system design.

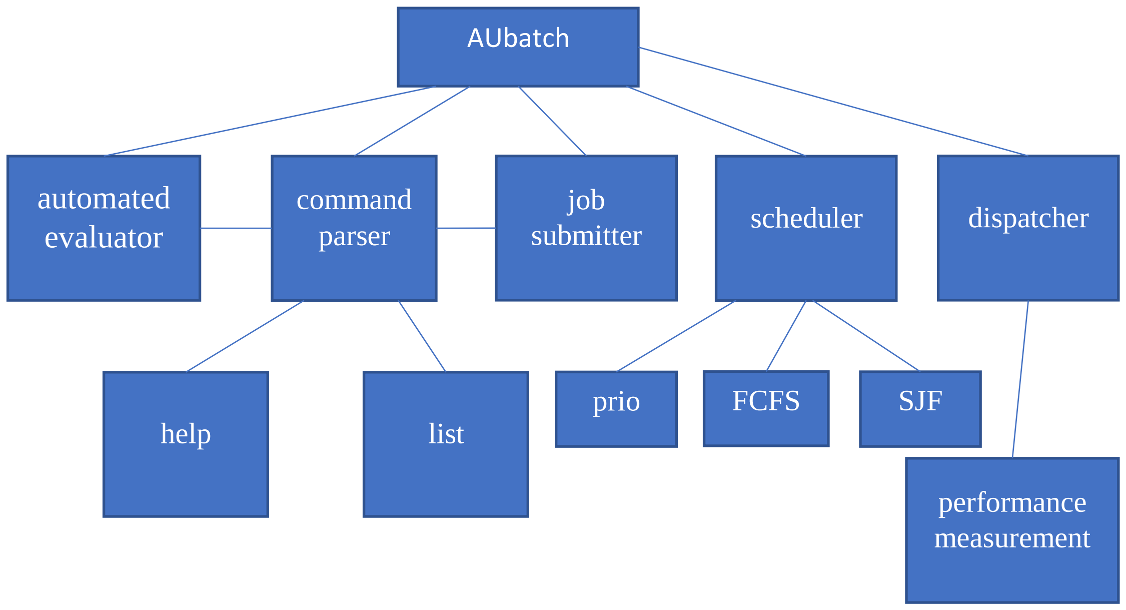
1.1 Data Flow Diagram

data flow diagram illustrates how data is processed by Aubatch. This is from the early stage of the project, so the real implementation may be a little bit different, but it still explains a lot about the system nonetheless. The core is that scheduling module takes job information, sorts the job queue based on policy and passes them on to dispatching module to be processed.



1.2 Program Structure

Program structure illustrates the modules included by Aubatch. Job submitter is separated from scheduler.



2. Performance metrics and workload information

2.1 Performance Metrics

In this project the following performance metrics will be applied.

- Average turnaround time
- Average CPU time
- Average waiting time
- Maximum waiting time
- Througput

Minimum waiting time is considered as well, but in this case it will always be 0. Because first coming job will always be executed upon entering Aubatch.

2.2 Workload Information

In this project the following workload information will be applied.

- Number of submitted job: 5, 10
- Arrival Interval: 2, 5, 10 seconds, which will be denoted in test by high, medium and low arrival rate.
- Load Distribution(uniform): [10,20]

3. Performance comparison

For the purpose of comparison, the random seed used in generating jobs will be the same. Every workload condition combinations will be run on all policies. Priority of jobs will be randomized in the range of 0 and 3.

policy	**job num**	**arrival rate**	**load**	**avg turnaroud**	**avg CPU**	**avg wait**	**max wait**	**throughput**
FCFS	5	low	[10,20]	26.611	14.604	12.007	22.013	0.038
SJF	5	low	[10,20]	25.610	14.603	11.007	27.013	0.039
prio	5	low	[10,20]	27.010	14.604	12.407	28.010	0.037
FCFS	10	low	[10,20]	37.718	14.703	23.015	47.029	0.027
SJF	10	low	[10,20]	34.819	14.703	20.115	60.027	0.029
prio	10	low	[10,20]	39.118	14.703	24.415	66.023	0.026
FCFS	5	med	[10,20]	36.611	14.604	22.007	42.014	0.027
SJF	5	med	[10,20]	34.810	14.603	20.207	42.014	0.029
prio	5	med	[10,20]	37.810	14.604	23.207	42.014	0.026
FCFS	10	med	[10,20]	60.218	14.703	45.514	92.029	0.017
SJF	10	med	[10,20]	54.918	14.703	40.214	87.029	0.018
prio	10	med	[10,20]	64.418	14.703	59.015	119.029	0.014
FCFS	5	high	[10,20]	42.610	14.603	28.007	54.013	0.023
SJF	5	high	[10,20]	40.810	14.603	26.206	54.013	0.025
prio	5	high	[10,20]	43.810	14.603	29.207	54.013	0.023
FCFS	10	high	[10,20]	73.718	14.703	59.015	119.029	0.014
SJF	10	high	[10,20]	67.817	14.703	53.114	111.029	0.015
prio	10	high	[10,20]	76.018	14.703	61.315	119.029	0.013

Some conclusions can be drawn from the result.

1. For the same sequence of jobs, SJF always has the smallest average waiting time, which leads to the smallest average turnaround time. FCFS and priority should have larger, if not the same, average waiting time. Which one is larger depends on the actual jobs.
2. For some workload information, policies don't have effects on max waiting time. In this experiment it happened when job number is 5 and didn't happen when job number is 10 at all. It seems that more jobs may cause a bigger change in schedule.
3. A bigger max waiting time doesn't necessarily mean a bigger average waiting time. This suggests it may not be a good performance metric.

For future experiments, some improvement can be made.

1. More performance metrics can be added such as waiting time standard deviation.
2. Different job sequences can be generated to compare their differences under the same policy.