ECE 6101 Course Project
Distributed Batch Processing System - Option 1

Abhishek Jain
ajain333@gatech.edu

Akar Shah
akar.j.shah@gatech.edu

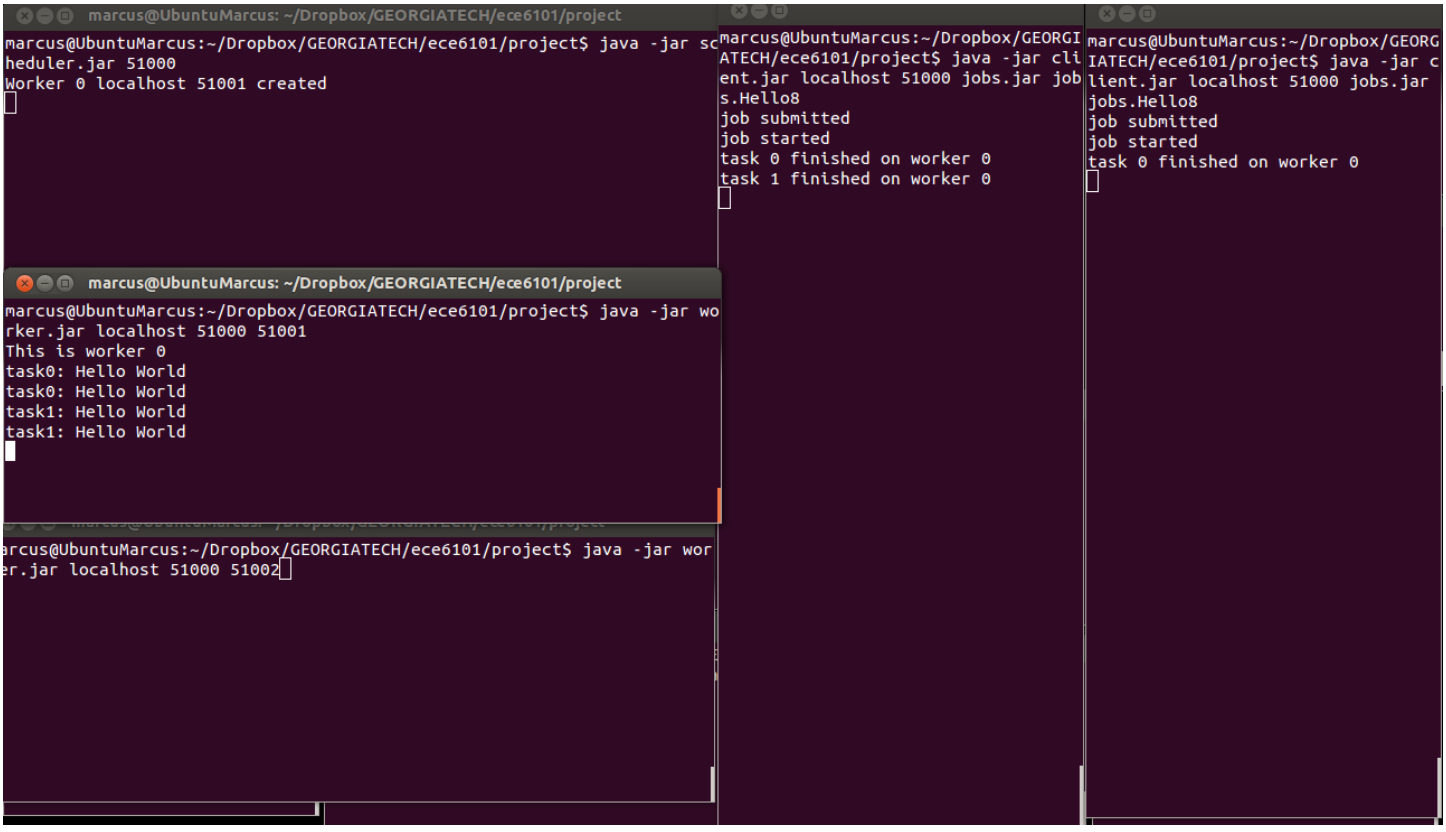
Experiment Screenshots

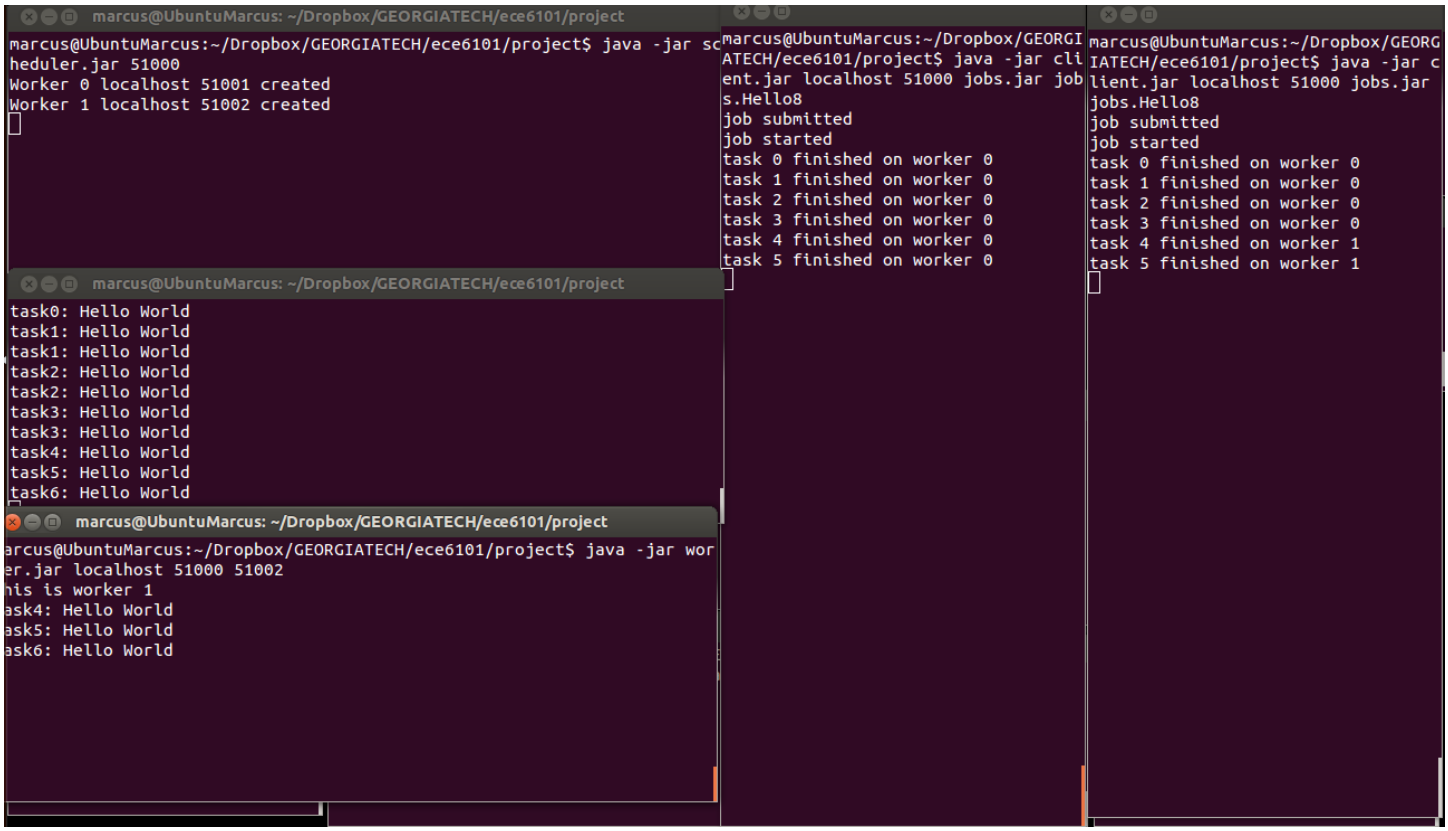
Experiment to test non-blocking request handling:

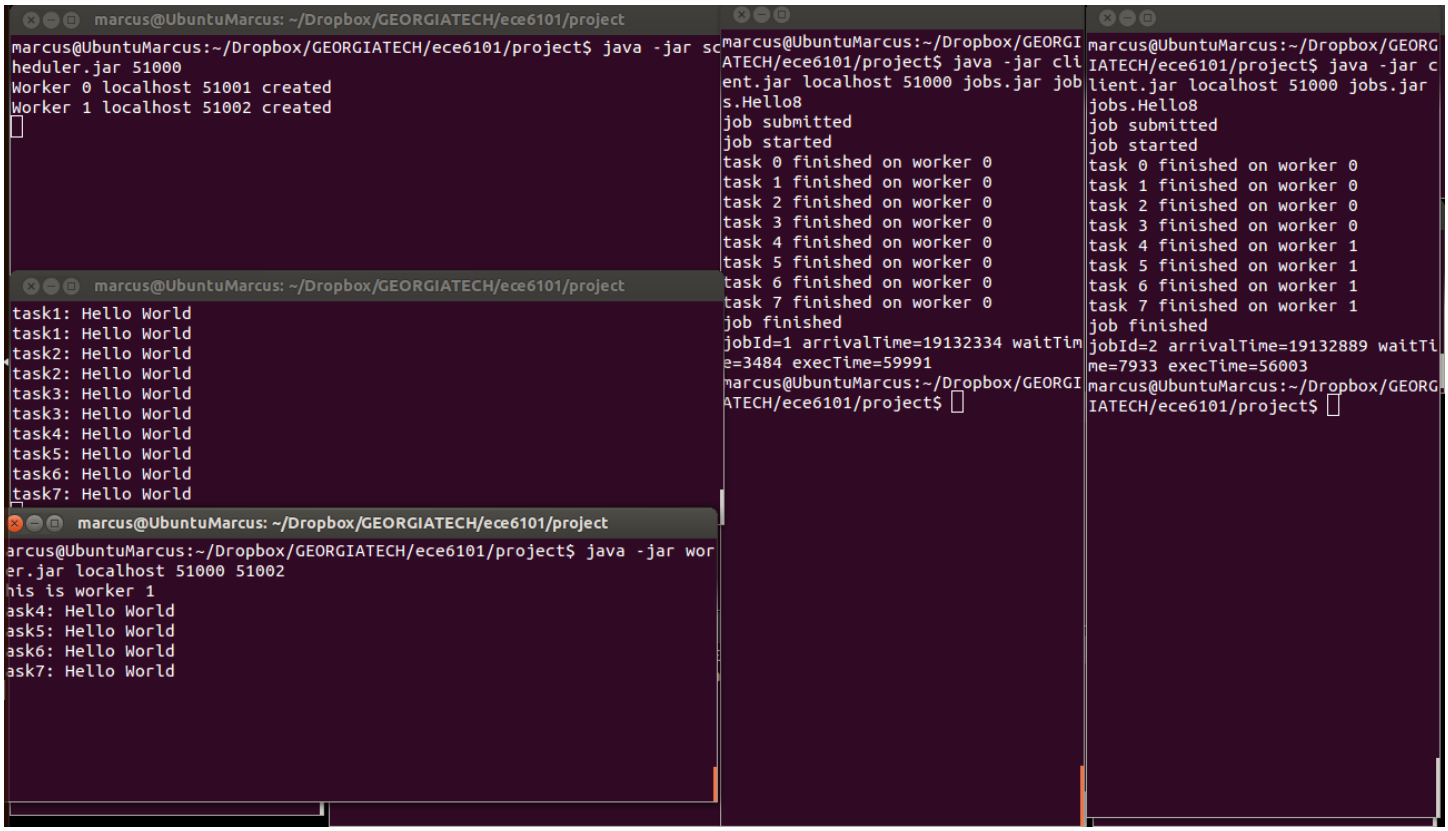
Note: For all experiments, the sleep duration for each task of the *Hello* Job was changed from 1sec to 5secs and for the *Mvm* Job, the sleep duration for each task was changed from 0sec to 1sec. New job files were also created for different experiments such as *Hello8* (8 tasks), *Hello3* (3 tasks), *Hello2* (2 tasks). The only difference between these jobs is the number of tasks in each job.

Part A: Simultaneous registration of multiple workers

To carry out this test, 2 jobs were assigned to the scheduler with no workers initially. Then, one by one 2 workers were added and the jobs were allowed to progress and complete. Initially both jobs start working with the first worker that was added. Upon adding a second worker, the remaining tasks of the jobs were distributed among the workers. This has been shown as follows:

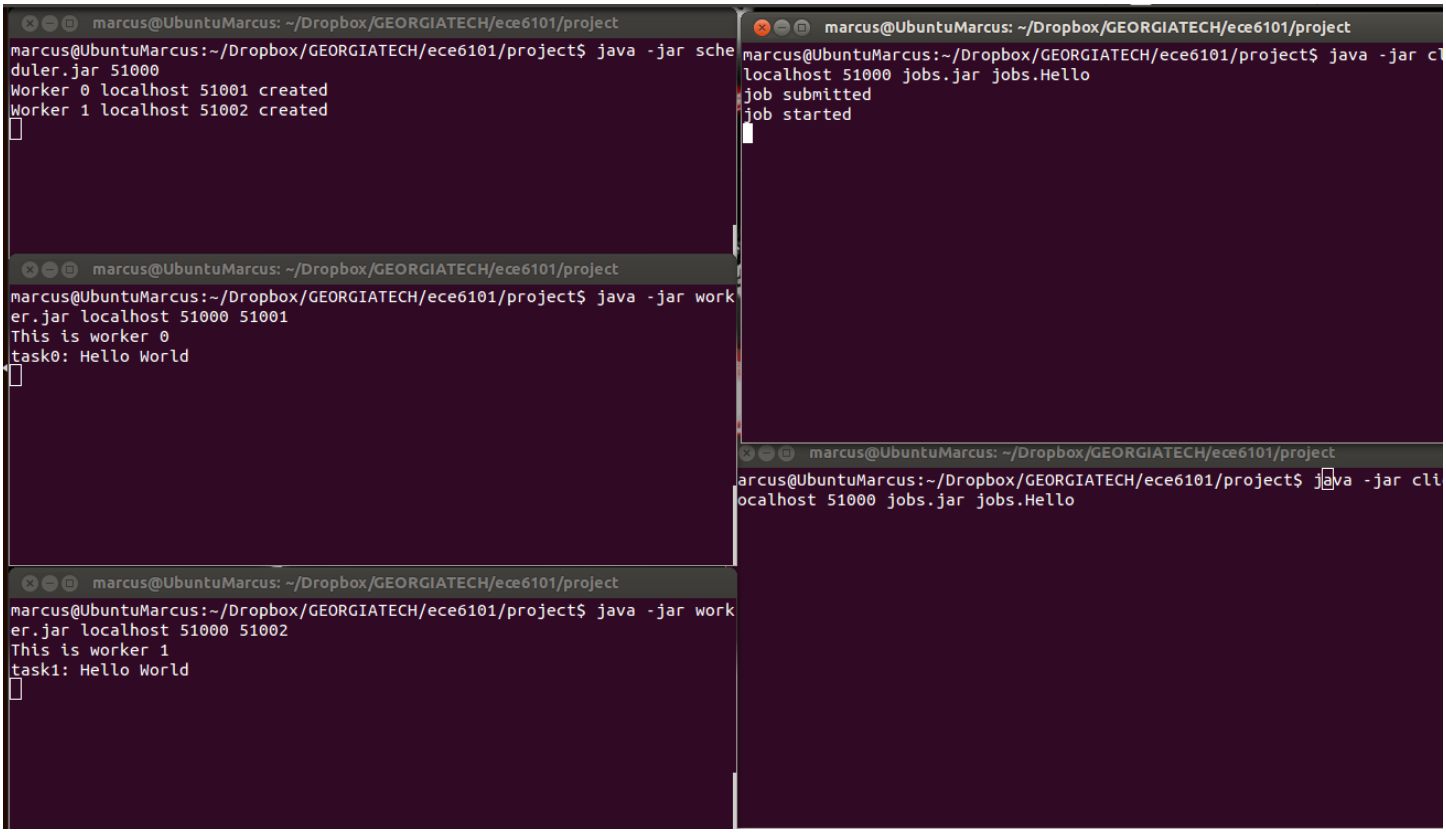


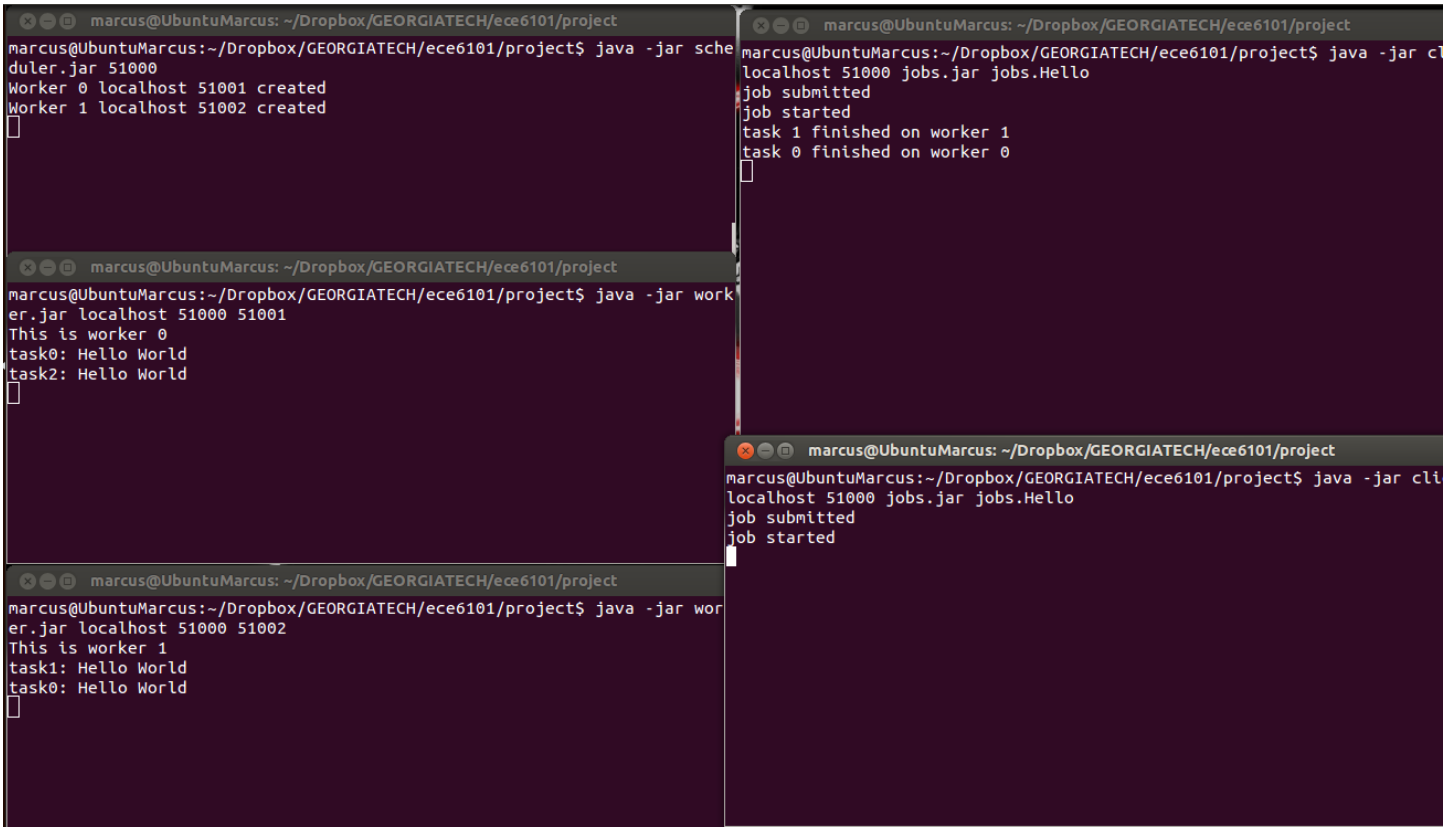

marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar scheduler.jar 51000
Worker 0 localhost 51001 created
Worker 1 localhost 51002 created
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
task0: Hello World
task1: Hello World
task1: Hello World
task2: Hello World
task2: Hello World
task3: Hello World
task3: Hello World
task4: Hello World
task5: Hello World
task6: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51002
This is worker 1
task4: Hello World
task5: Hello World
task6: Hello World
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello8
job submitted
job started
task 0 finished on worker 0
task 1 finished on worker 0
task 2 finished on worker 0
task 3 finished on worker 0
task 4 finished on worker 0
task 5 finished on worker 0
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello8
job submitted
job started
task 0 finished on worker 0
task 1 finished on worker 0
task 2 finished on worker 0
task 3 finished on worker 0
task 4 finished on worker 1
task 5 finished on worker 1

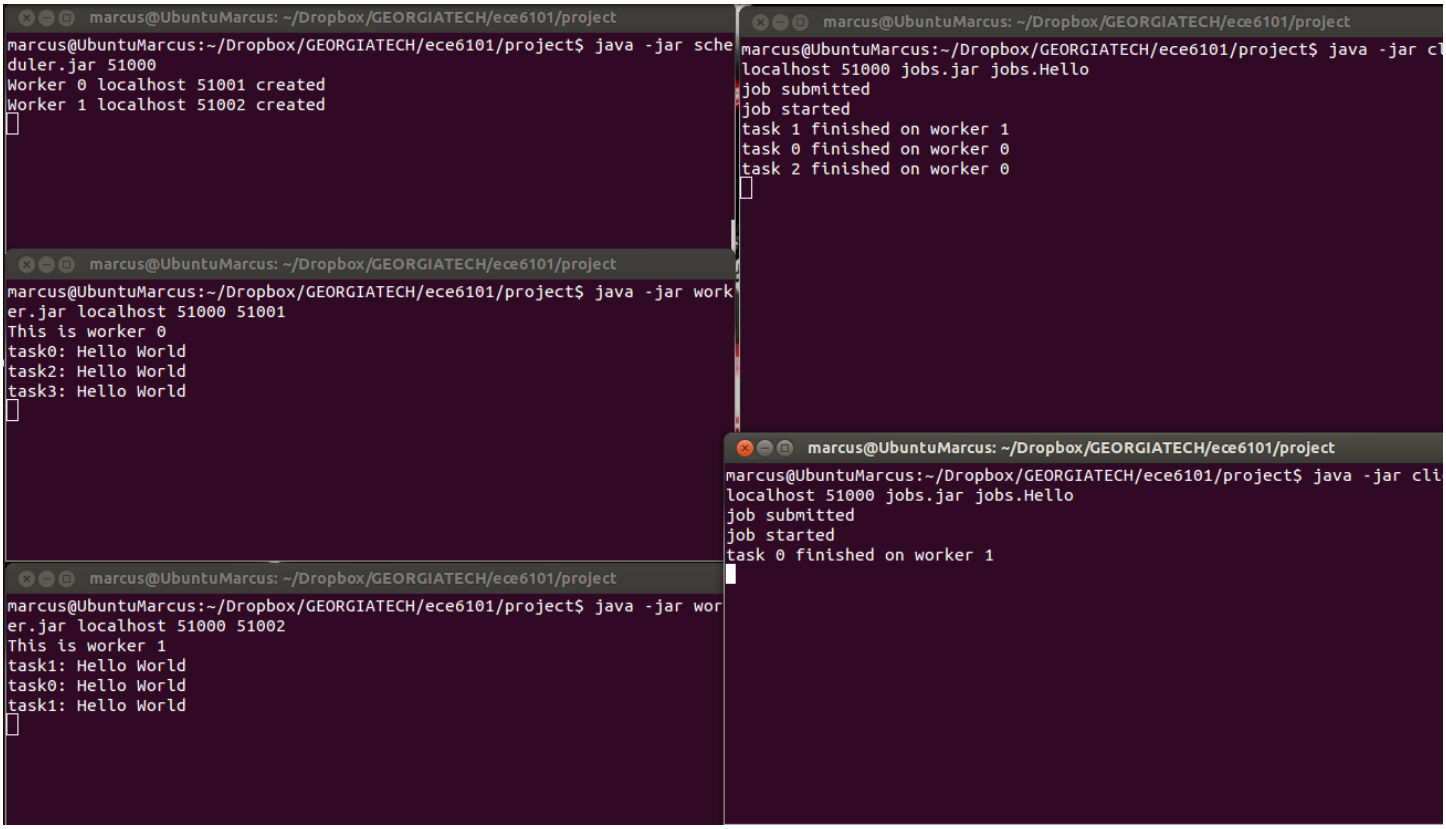

marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar scheduler.jar 51000
Worker 0 localhost 51001 created
Worker 1 localhost 51002 created
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
task1: Hello World
task1: Hello World
task2: Hello World
task2: Hello World
task3: Hello World
task3: Hello World
task4: Hello World
task5: Hello World
task6: Hello World
task7: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51002
This is worker 1
task4: Hello World
task5: Hello World
task6: Hello World
task7: Hello World
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello8
job submitted
job started
task 0 finished on worker 0
task 1 finished on worker 0
task 2 finished on worker 0
task 3 finished on worker 0
task 4 finished on worker 0
task 5 finished on worker 0
task 6 finished on worker 0
task 7 finished on worker 0
job finished
jobId=1 arrivalTime=19132334 waitTime=3484 execTime=59991
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello8
job submitted
job started
task 0 finished on worker 0
task 1 finished on worker 0
task 2 finished on worker 0
task 3 finished on worker 0
task 4 finished on worker 1
task 5 finished on worker 1
task 6 finished on worker 1
task 7 finished on worker 1
job finished
jobId=2 arrivalTime=19132889 waitTime=7933 execTime=56003
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$

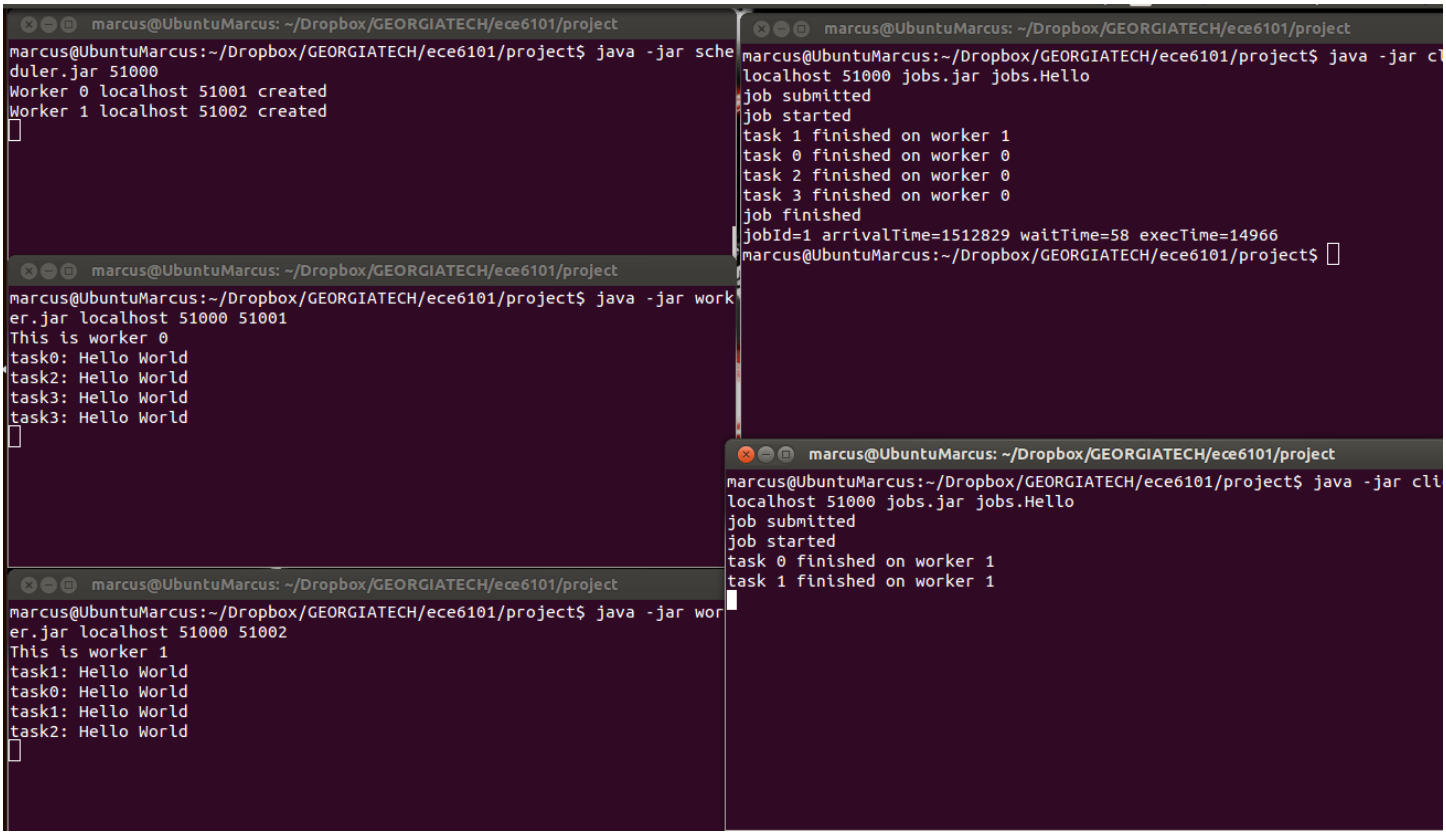
Part B: Simultaneous job submission

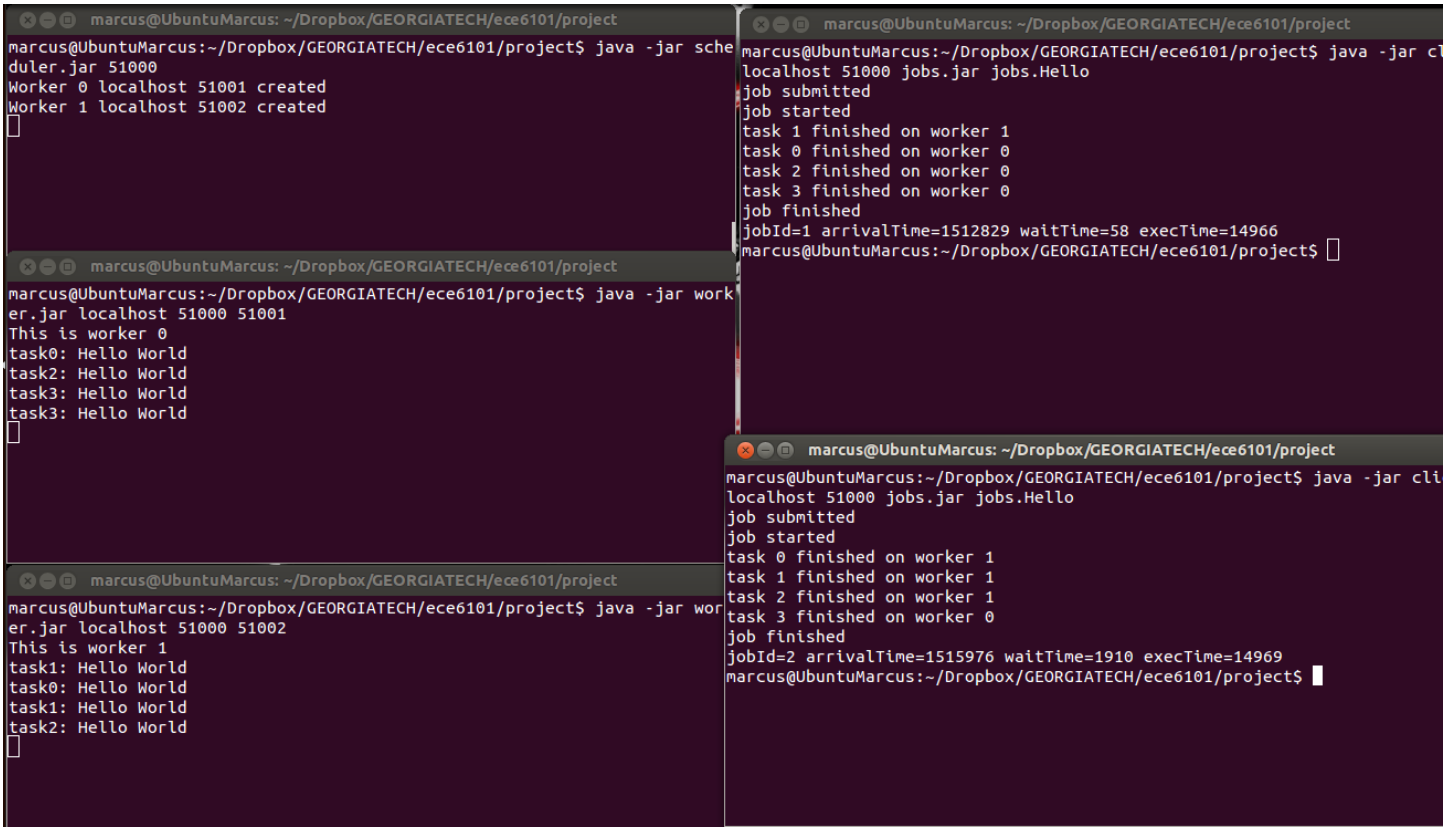
To carry out this test, 2 workers were initially asked to carry out a single job. During the execution of the first job, another job was added which was then handled by both the existing workers. This is shown below:





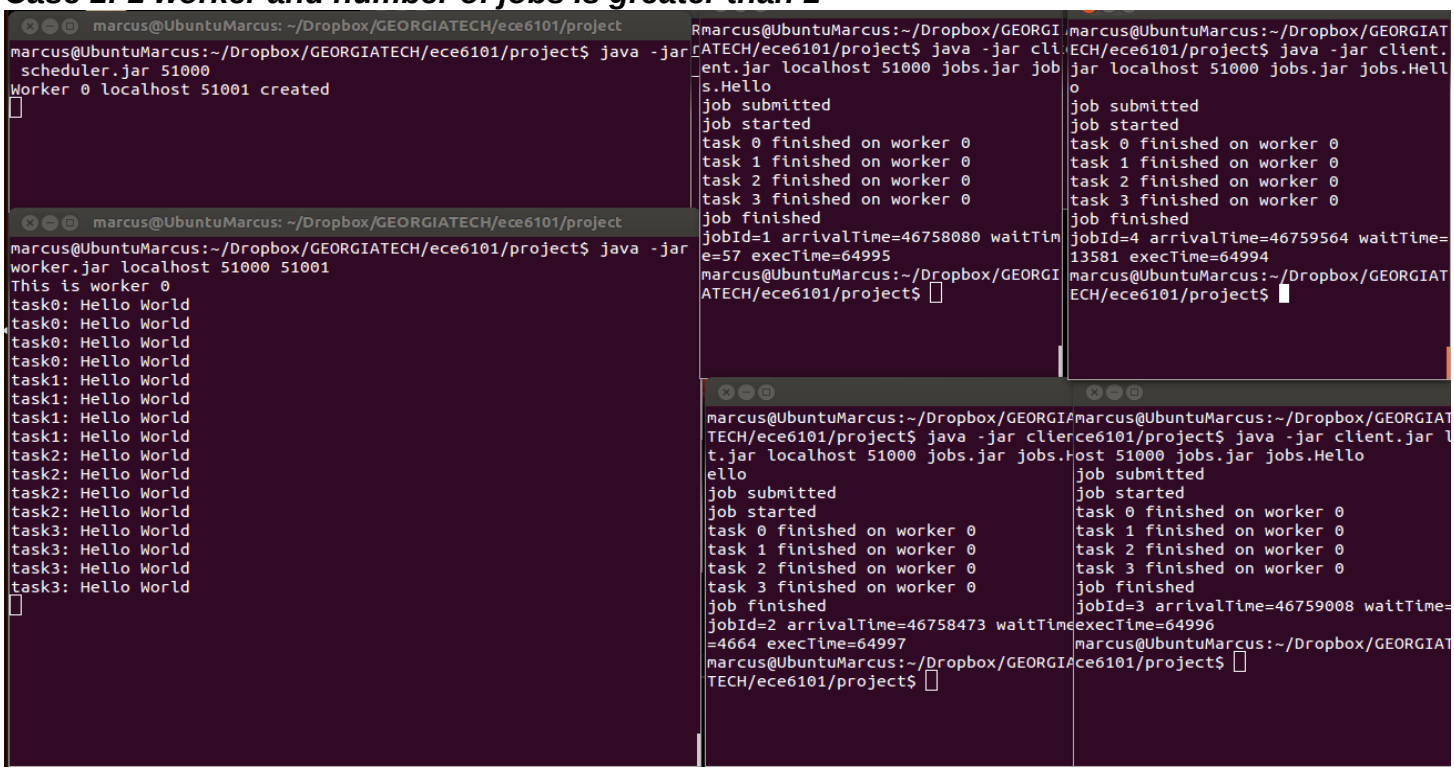

marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar scheduler.jar 51000
Worker 0 localhost 51001 created
Worker 1 localhost 51002 created
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar cl
localhost 51000 jobs.jar jobs.Hello
job submitted
job started
task 1 finished on worker 1
task 0 finished on worker 0
task 2 finished on worker 0
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51001
This is worker 0
task0: Hello World
task2: Hello World
task3: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar cli
localhost 51000 jobs.jar jobs.Hello
job submitted
job started
task 0 finished on worker 1
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51002
This is worker 1
task1: Hello World
task0: Hello World
task1: Hello World


marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar scheduler.jar 51000
Worker 0 localhost 51001 created
Worker 1 localhost 51002 created
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar cl
localhost 51000 jobs.jar jobs.Hello
job submitted
job started
task 1 finished on worker 1
task 0 finished on worker 0
task 2 finished on worker 0
task 3 finished on worker 0
job finished
jobId=1 arrivalTime=1512829 waitTime=58 execTime=14966
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51001
This is worker 0
task0: Hello World
task2: Hello World
task3: Hello World
task3: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar cli
localhost 51000 jobs.jar jobs.Hello
job submitted
job started
task 0 finished on worker 1
task 1 finished on worker 1
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51002
This is worker 1
task1: Hello World
task0: Hello World
task1: Hello World
task2: Hello World

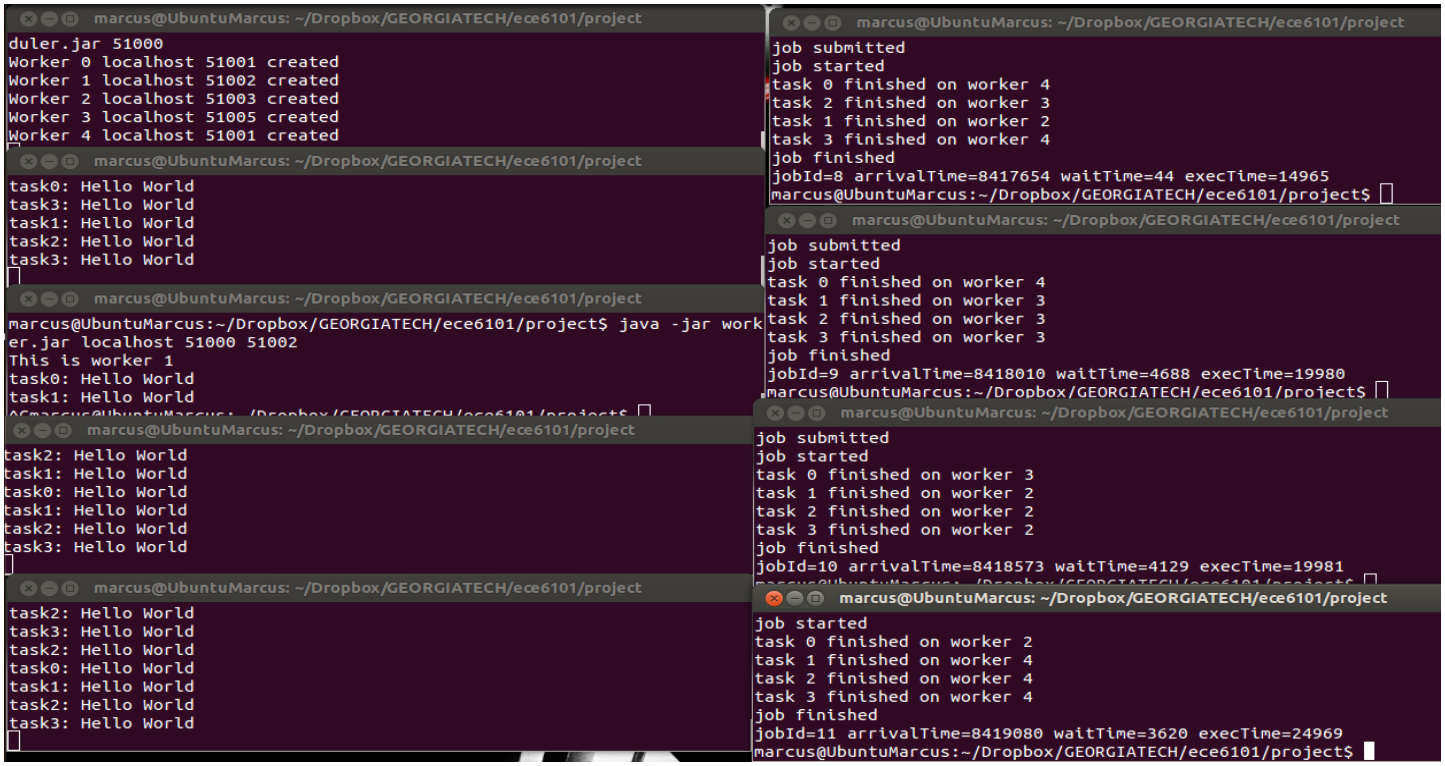


Experiment to test fair scheduling:

Case 1: 1 worker and number of jobs is greater than 1



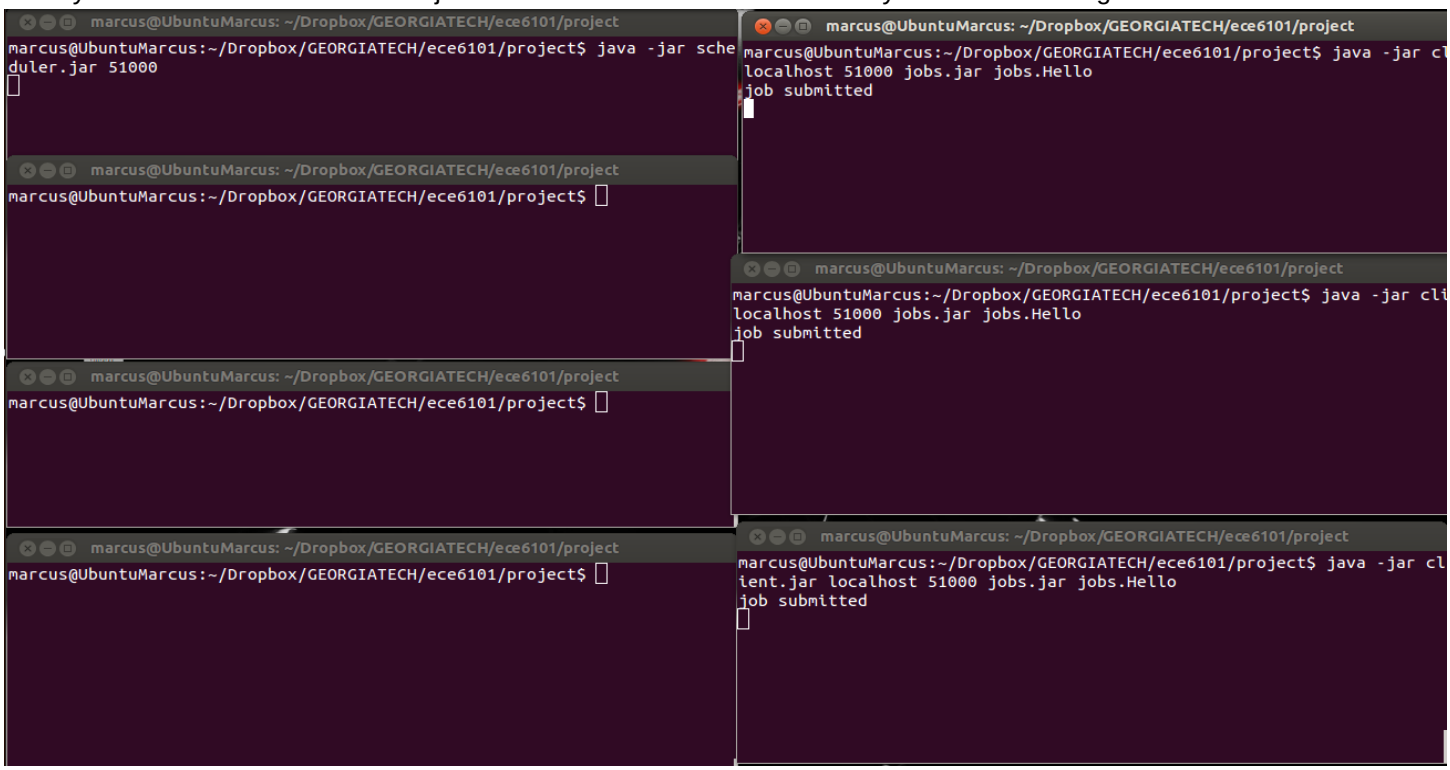
Case 2: Number of workers is less than the number of tasks



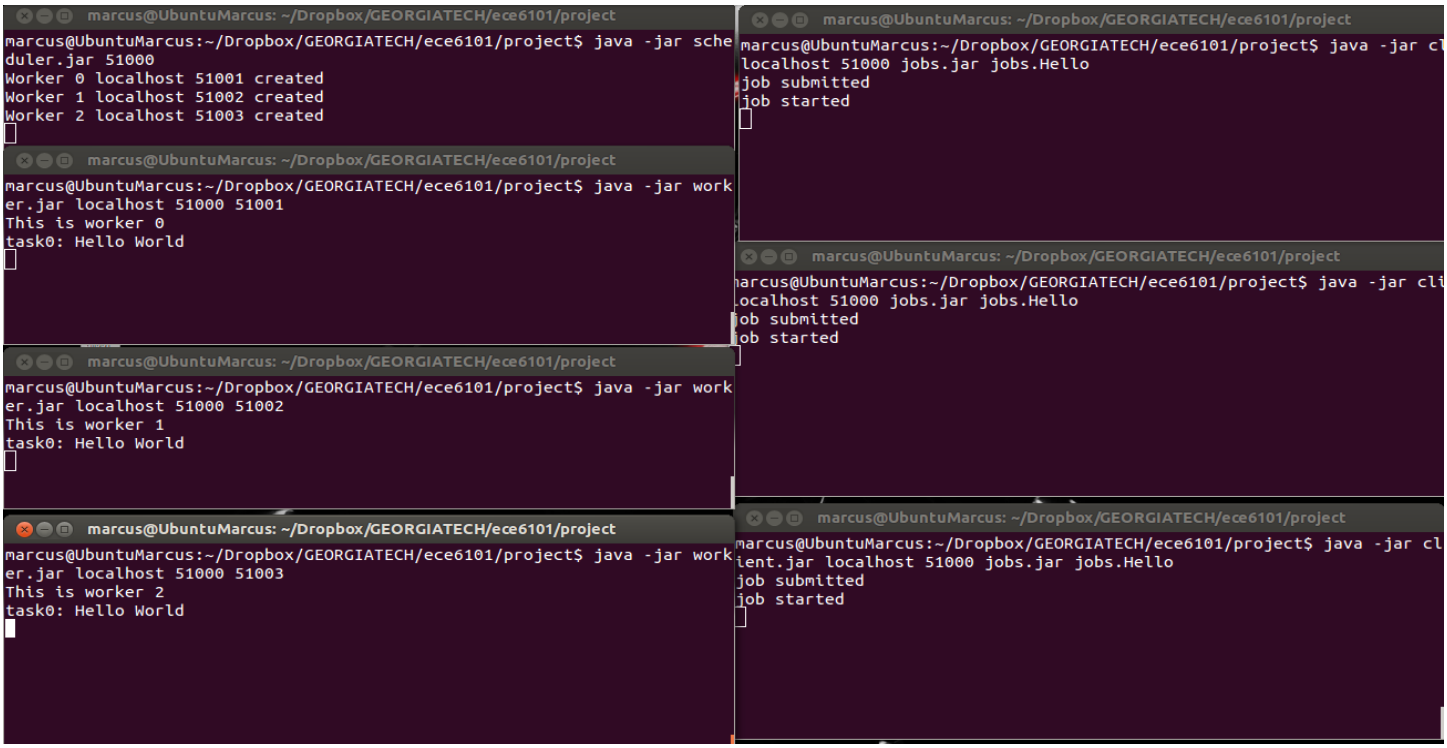
Experiment to test fault tolerance:

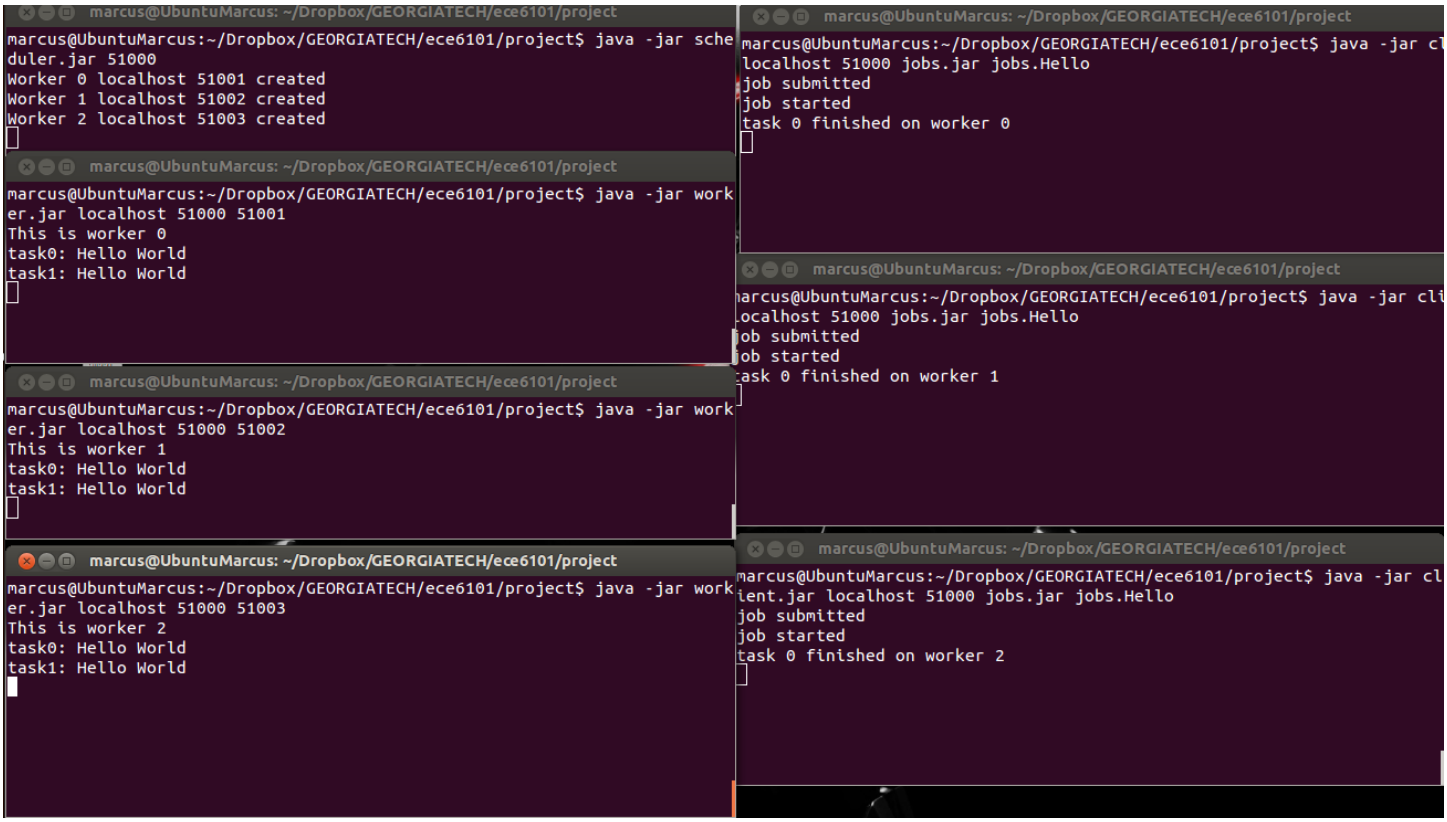
The following experiment shows how our system exhibits fault tolerance:

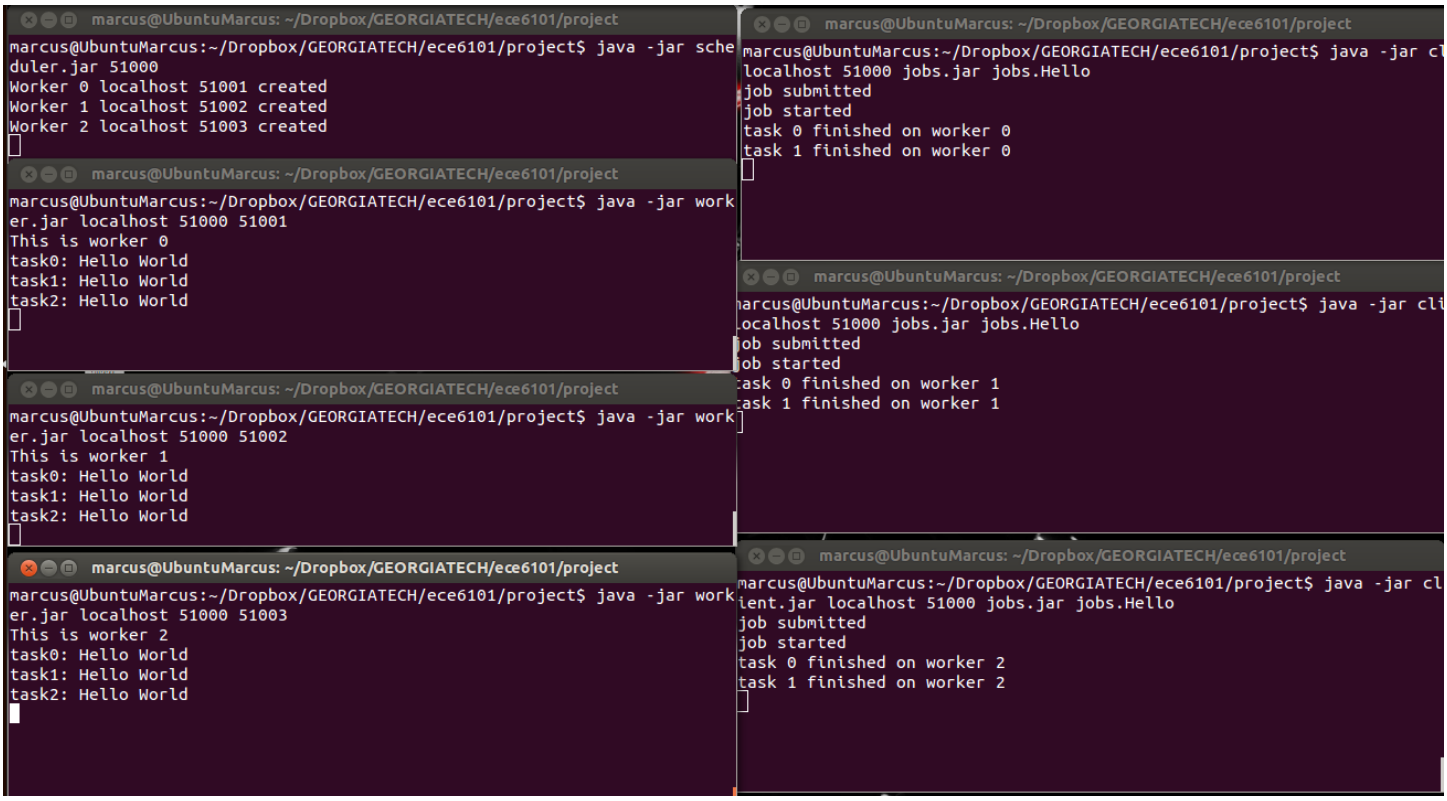
Initially there are no workers but the jobs are still submitted and wait for any free worker to register itself.



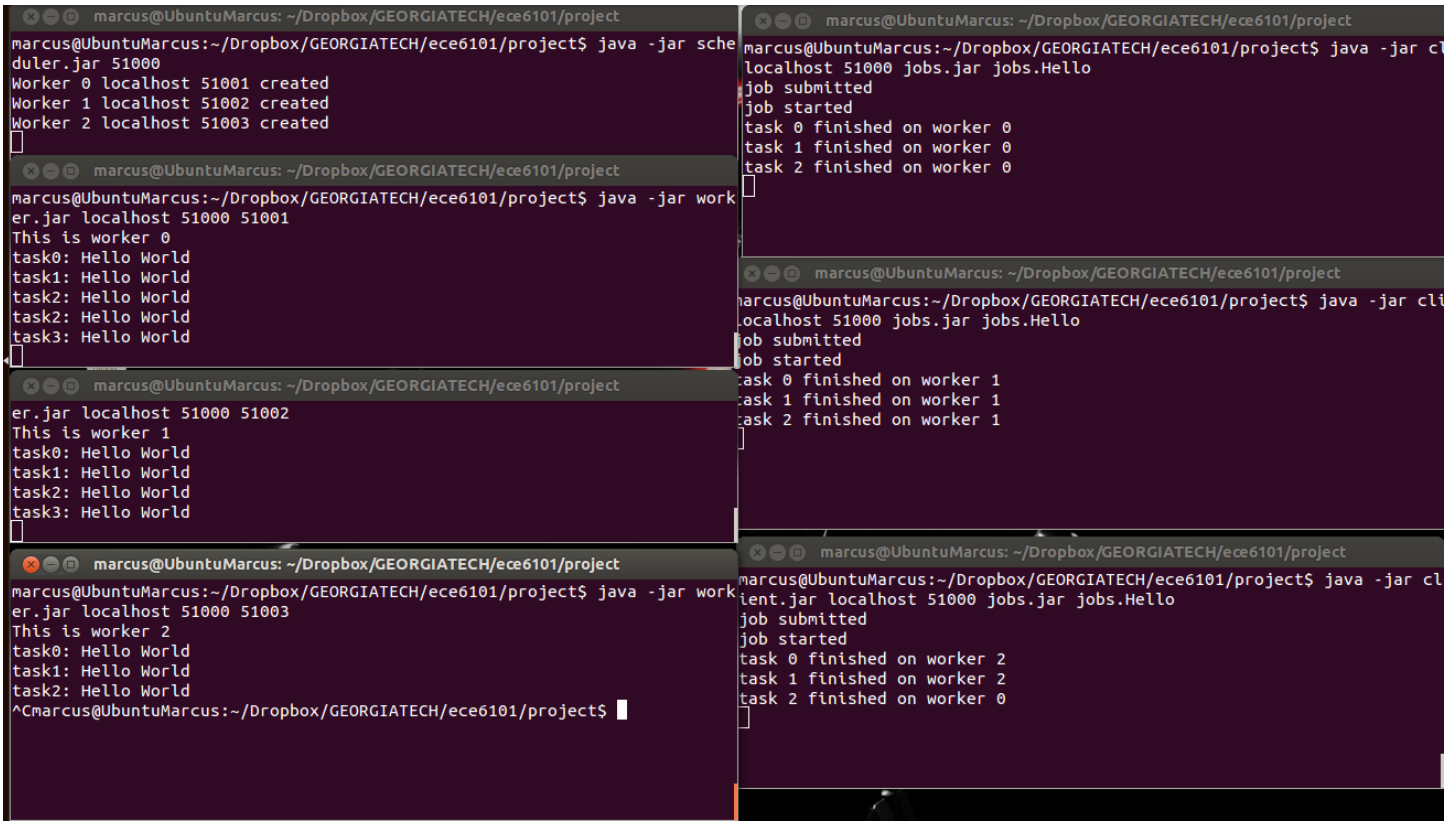
Here we see that as workers are added, tasks from different jobs are distributed fairly between the workers and execution of different jobs starts.


marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar scheduler.jar 51000
Worker 0 localhost 51001 created
Worker 1 localhost 51002 created
Worker 2 localhost 51003 created
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51001
This is worker 0
task0: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51002
This is worker 1
task0: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51003
This is worker 2
task0: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello
job submitted
job started
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello
job submitted
job started
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello
job submitted
job started

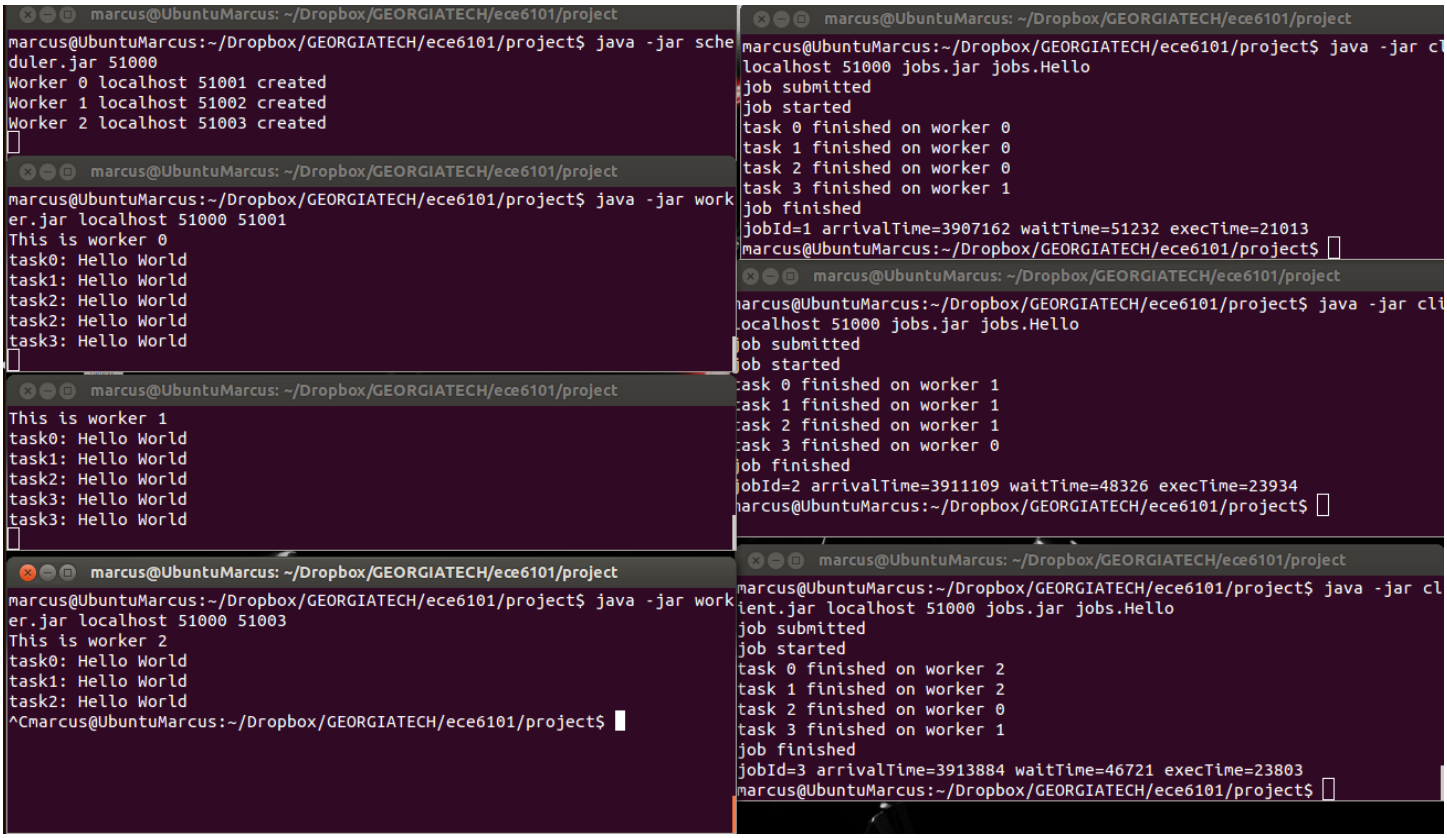

marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar scheduler.jar 51000
Worker 0 localhost 51001 created
Worker 1 localhost 51002 created
Worker 2 localhost 51003 created
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51001
This is worker 0
task0: Hello World
task1: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51002
This is worker 1
task0: Hello World
task1: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar worker.jar localhost 51000 51003
This is worker 2
task0: Hello World
task1: Hello World
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello
job submitted
job started
task 0 finished on worker 0
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello
job submitted
job started
task 0 finished on worker 1
marcus@UbuntuMarcus: ~/Dropbox/GEORGIATECH/ece6101/project
marcus@UbuntuMarcus:~/Dropbox/GEORGIATECH/ece6101/project$ java -jar client.jar localhost 51000 jobs.jar jobs.Hello
job submitted
job started
task 0 finished on worker 2

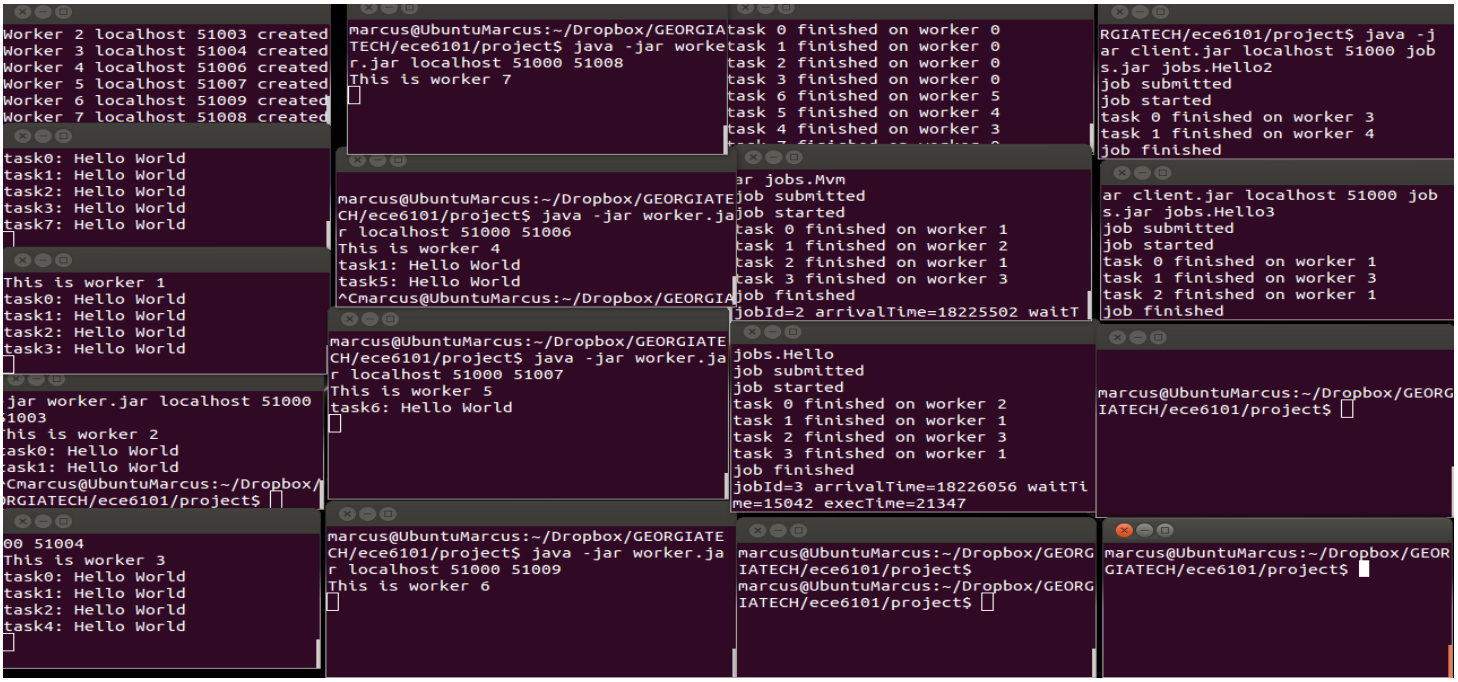


The diagrams below shows worker 2 failing and then shows how the remaining active workers namely worker 1 and 0 manage the remainder of the tasks from different jobs. Here worker 2 fails.



Here the remaining workers handle the remaining tasks and jobs.



As can be seen from the above experiments, it is seen that our system continues to execute jobs even after there is worker failure. Repeated testing with different numbers of node failures still yields the same results and as long as there is even a single worker alive, jobs will continue to get executed, irrespective of the complexity , duration or the number of tasks of the job.

The diagram below shows the result of an execution where several jobs are given to the scheduler and not only are several workers made to fail during the runtime of the jobs, several new workers are also added during the runtime of different jobs. Thus it can be seen that no matter what the scenario, our system is able to complete all jobs that are assigned to it.