**Part 3**

**Q1:**

Mean time(s) Std

Dedup 37.33 1.70

Blackscholes 348.33 4.50

Ferret 174.33 0.47

Freqmine 96.67 0.47

Canneal 140.67 3.30

Fft 80.33 1.25

Total time 348.33 4.50

SLO violation ratio: 0％ for all three runs.

**Three plots：**

See plot2.pdf, plot3.pdf, plot4.pdf

**Q2**.

**Scheduling Policy:**

For Memcached: We run it on “node-a-2core”, with 1st thread.

For 6 PARSEC jobs:

node-a-2core: At T0, we deploy job “blackscholes”, using 2nd thread.

node-b-4core: At T0, we deploy job “canneal”, using all 4 threads; After “canneal” completes, we run “fft” immediately, using 4 threads; After “fft” completes, deploying “dedup” immediately, with 4 threads.

node-c-8core: At T0, we deploy job “ferret”, using all 8 threads; After “ferret” completes, we run “freqmine” immediately, using 8 threads;

We modified the yaml file for each job, using the “NodeSelect” and “CoreSelect” features of Kubernetes to achieve node selection and core assignment for jobs. For example, we modify this part “cca-project-nodetype: "node-b-4core"” in yaml to select which node we want to deploy the job on; Use the command” "taskset -c 0-3 ./bin/parsecmgmt -a run -p dedup -i native -n 4"]”, specifying the arguments after “-c”, to assign the job with specified cores, and the arguments after “-n”, to appoint the number of threads for the specific job.

**The reason of using this policy:**

We designed the policy considering the analysis from part1 and part2. From part1 Question 2, we know that memchaed is significantly sensitive to cpu, l1i, moderately and lightly sensitive to llc, membw; From part2 Question 1, dedup, ferret and freqmine are highly l1i-intensive, and dedup, ferret, canneal and fft are highly llc-intensive. Among all 6 PARSEC jobs, only blackscholes will not have too intense resource competition with memcached. Therefore, we choose to deploy blackscholes together with memcached on node with 2 cores to avoid too intense resource competition between memcached and other PARSEC jobs, which might lead to SLO violations. To further avoid cpu competition, assign these two jobs with different cores.

From part2 question2, ferret and freqmine are the 2 jobs that are most time-consuming. Thus, we deploy them on the node with 8 cores. Furthermore, these 2 jobs are both l1i-intensive, so we decide to run them in sequence, instead of in parallel, to avoid drastic competence on l1i.

For the remaining three jobs: canneal, fft and dedup, their time costs are mediate among 6 PARSEC jobs. Also, the benefit of assigning them with 8 cores is not very larger than with 4 cores. Thus, we choose to put them on node-b with 4 cores. For the same reason, we run these three jobs serially rather than in parallel.