ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Cloud Computing Architecture

Semester project report

Group XXX
Pitcho Oscar - 17-809-971
Perone Luca - 18-812-388
Oester Robin - 19-932-557

Systems Group
Department of Computer Science
ETH Zurich
March 24, 2023

Instructions

- Please do not modify the template, except for putting your solutions, group number, names and legi-NR.
- Parts 1 and 2 should be answered in maximum six pages (including the questions).
  If you exceed the space, points may be subtracted.

Part 1 [25 points]

Using the instructions provided in the project description, run memcached alone (i.e., no interference), and with each iBench source of interference (cpu, l1d, l1i, l2, llc, membw). For Part 1, you must use the following `mcperf` command, which varies the target QPS from 30000 to 110000 in increments of 5000 (and has a warm-up time of 2 seconds with the addition of `-w 2`):

```
$ ./mcperf -s MEMCACHED_IP --loadonly
$ ./mcperf -s MEMCACHED_IP -a INTERNAL_AGENT_IP  \
          --noload -T 16 -C 4 -D 4 -Q 1000 -c 4 -w 2 -t 5 \
          --scan 30000:110000:5000
```

Repeat the run for each of the 7 configurations (without interference, and the 6 interference types) **at least three times** (three should be sufficient in this case), and collect the performance measurements (i.e., the `client-measure` VM output). Reminder: after you have collected all the measurements, make sure you delete your cluster. Otherwise, you will easily use up the cloud credits. See the project description for instructions how.

(a) [**10 points**] Plot a single line graph with the following stipulations:

- Queries per second (QPS) on the x-axis (the x-axis should range from 0 to 110K). (note: the actual achieved QPS, not the target QPS)
- 95th percentile latency on the y-axis (the y-axis should range from 0 to 8 ms).
- Label your axes.
- 7 lines, one for each configuration. Add a legend.
- State how many runs you averaged across and include error bars at each point in both dimensions.
- The readability of your plot will be part of your grade.

(b) [**6 points**] How is the tail latency and saturation point (the "knee in the curve") of memcached affected by each type of interference? Why? Briefly describe your hypothesis.

(c) [**2 points**] Explain the use of the `taskset` command in the container commands for memcached and iBench in the provided scripts. Why do we run some of the iBench benchmarks on the same core as memcached and others on a different core?

(d) [**2 points**] Assuming a service level objective (SLO) for memcached of up to 1.5 ms 95th percentile latency at 65K QPS, which iBench source of interference can safely be collocated with memcached without violating this SLO? Briefly explain your reasoning.

(e) [**5 points**] In the lectures you have seen queueing theory. Is the project experiment above an open system or a closed system? What is the number of clients in the system? Sketch a diagram of the queueing system and provide an expression for the average response time. Explain each term in the response time expression.

Part 2 [30 points]

1. **Interference behavior [19 points]**

   (a) **[11 points]** Fill in the following table with the normalized execution time of each batch job with each source of interference. The execution time should be normalized to the job's execution time with no interference. Round the normalized execution time to 2 decimal places. Color-code each field in the table as follows: **green** if the normalized execution time is less than or equal to 1.3, **orange** if the normalized execution time is over 1.3 and up to 2, and **red** if the normalized execution time is greater than 2. Briefly summarize in a paragraph the resource interference sensitivity of each batch job.

Workload	`none`	`cpu`	`l1d`	`l1i`	`l2`	`llc`	`memBW`
blackscholes	1.00						
canneal	1.00						
dedup	1.00						
ferret	1.00						
freqmine	1.00						
radix	1.00						
vips	1.00						

   (b) **[8 points]** Explain what the interference profile table tells you about the resource requirements for each application. Which jobs (if any) seem like good candidates to collocate with memcached from Part 1, without violating the SLO of 2 ms P95 latency at 40K QPS?

2. **Parallel behavior [11 points]**

   Plot a single line graph with speedup as the y-axis (normalized time to the single thread config, Time_1 / Time_n) vs. number of threads on the x-axis (1, 2, 4 and 8 threads, see the project description for more details). Briefly discuss the scalability of each application: e.g., linear/sub-linear/super-linear. Do the applications gain significant speedup with the increased number of threads? Explain what you consider to be "significant".