Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Cloud Computing Architecture

Semester project report

Group 076
Dennis Jüni - 19-935-493
Eric Nothum - Legi-NR
Alex Thillen - 19-946-953

Systems Group
Department of Computer Science
ETH Zurich
March 14, 2024

Instructions

- **Please do not modify the template!** Except for adding your solutions in the labeled places, and inputting your group number, names and legi-NR on the title page.

  Divergence from the template can lead to subtraction of points.

- Parts 1 and 2 of the project should be answered in **maximum 6 pages** (including the questions, and excluding the title page and this page, containing the instructions - the maximum total number of pages is 8).

  If you exceed the allowed 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 5000 to 55000 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 5000:55000:5000
```

Repeat the run for each of the 7 configurations (without interference, and the 6 interference types) **at least 3 times** (3 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 to make sure your cluster is deleted.

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

- Queries per second (QPS) on the x-axis (the x-axis should range from 0 to 55K). (**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.

Answer:

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

Answer:

- `ibench-cpu`:

  Explanation:

- `ibench-l1d`:

  Explanation:

- `ibench-l1i`:

  Explanation:

- `ibench-l2`:

  Explanation:

- `ibench-llc`:

  Explanation:

- `ibench-membw`:

  Explanation:

(c) [**2 points**]

- Explain the use of the `taskset` command in the container commands for memcached and iBench in the provided scripts.

  Answer:

- Why do we run some of the iBench benchmarks on the same core as memcached and others on a different core? Give an explanation for each iBench benchmark.

  Answer:

(d) [**2 points**] Assuming a service level objective (SLO) for memcached of up to 2 ms 95th percentile latency at 35K QPS, which iBench source of interference can safely be collocated with memcached without violating this SLO? Briefly explain your reasoning.

Answer:

(e) [**5 points**] In the lectures you have seen queueing theory.

- Is the project experiment above an open system or a closed system? Explain why.

  Answer:

- What is the number of clients in the system? How did you find this number?

  Answer:

- Sketch a diagram of the queueing system modeling the experiment above. Give a brief explanation of the sketch.

  Answer:

- Provide an expression for the average response time of this system. Explain each term in this expression and match it to the parameters of the project experiment.

  Answer:

Part 2 [31 points]

1. **Interference behavior [20 points]**

   (a) **[10.5 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 cell 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 less or equal to 2, and **red** if the normalized execution time is greater than 2. The coloring of the first three cells is given as an example of how to use the cell coloring command. **Do not change the structure of the table. Only input the values inside the cells and color the cells properly.**

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) **[7 points]** Explain what the interference profile table tells you about the resource requirements for each job. Give your reasoning behind these resource requirements based on the functionality of each job.

   Answer:

   - `blackscholes:`
     Explanation:
   - `canneal:`
     Explanation:
   - `dedup:`
     Explanation:
   - `ferret:`
     Explanation:
   - `freqmine:`
     Explanation:
   - `radix:`
     Explanation:
   - `vips:`
     Explanation:

(c) [**2.5 points**] 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? Explain why.
Answer:

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

(a) [**7.5 points**] Plot a 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). Pay attention to the readability of your graph, it will be a part of your grade.
Answer:

(b) [**3.5 points**] Briefly discuss the scalability of each job: 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".
Answer:

- `blackscholes:`
- `canneal:`
- `dedup:`
- `ferret:`
- `freqmine:`
- `radix:`
- `vips:`