CS 553 HW7

An Empirical Evaluation of Distributed Key/Value Storage Systems

Instructions:

- ***Assigned: Tuesday 11/15/22***
- ***Due date: 11:59PM on Monday 11/28/22***
- *Maximum Points: 100%*
- *This homework can be done in teams of up to 3 students*
- *Please post your questions to BB*
- *Only a softcopy submission is required; submission is a 2-step process: 1) push changes to GIT repository, and email confirmation will be sent to your HAWK email address at the deadline; a confirmation document with all team member names and A# must be submitted through BlackBoard for your submission to be graded; only 1 student must submit the assignment, and only the submitting student will receive the confirmation email*
- *Late submission will be penalized at 10% per day (students working alone can get 1 late-day submission without penalty)*

1 The problem

In this project, you are going to evaluate various distributed key/value storage systems, and compare them to existing literature.

1.1 Read paper

Read paper on ZHT [1]. Pay special attention to Figure 8 and Figure 10, as you are going to conduct similar experiments in this project.

[1] Tonglin Li, Xiaobing Zhou, Kevin Brandstatter, Dongfang Zhao, Ke Wang, Anupam Rajendran, Zhao Zhang, Ioan Raicu. "ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table", IEEE International Parallel & Distributed Processing Symposium (IPDPS) 2013; http://datasys.cs.iit.edu/publications/2013_IPDPS13_ZHT.pdf

The 3 systems that were compared in this paper were:

- ZHT (see above paper)
- Cassandra (http://cassandra.apache.org)
- Memcached (https://memcached.org)

Please use the following data (that came from the ZHT paper [1]) to plug into your own results comparison:

System / Scale	1	2	4	8
ZHT	0.243	0.362	0.408	0.428
Cassandra	1.199	1.870	1.875	1.994
Memcached	0.122	0.324	0.272	0.278

Figure 8: Performance evaluation of ZHT and Memcached plotting latency vs. scale (1 to 64 nodes on the HEC-Cluster); unit of measurement is milliseconds per operation

System / Scale	1	2	4	8
ZHT	4117	5524	9813	18680
Cassandra	926	1131	2189	4322
Memcached	7385	7961	14480	40995

Figure 10: Performance evaluation of ZHT, Memcached and Casandra plotting throughput vs. scale (1 to 64 nodes on the HEC-Cluster); unit of measurement is operations per second

1.2 Testbed: Chameleon Cloud

You are to use the Chameleon cloud (https://www.chameleoncloud.org) to conduct your evaluation. Evaluate your system on up to 8 instances. You must deploy Ubuntu Linux 22.04 using "compute-haswell" nodes, at the CHI@IIT site (although other sites would work as well). Once you create a lease (up to 7 days are allowed), and start your 1 physical node, and Linux boots, you will find yourself with a physical node with 24 CPU cores, 48 hardware threads, 128GB of memory, and 250GB SSD hard drive. You will install your favorite virtualization tools (e.g. virtualbox, LXD/KVM, qemu), and use it to deploy 8 VMs with the following sizes: small.instance (3-cores, 12gb ram, 24gb disk).

1.2 Systems to Evaluate

You must evaluate the following distributed key/value stores:

- Cassandra (http://cassandra.apache.org)

You must also choose 2 of the following 7 distributed key/value stores to evaluate:

- MongoDB (https://www.mongodb.org)
- CouchDB (http://couchdb.apache.org)
- HBase (http://hbase.apache.org)
- Riak (http://basho.com/products/#riak)
- Redis (http://redis.io)
- HyperTable (http://www.hypertable.com)
- Oracle NoSQL Database (https://www.oracle.com/database/technologies/related/nosql.html)

You are to compare these 3 systems to the data presented in paper [1].

1.3 Compare and contrast your 3 evaluated systems to ZHT

Write a paragraph about each of your 3 systems, as well as about ZHT summarizing what the systems are, and what the key features are. Follow this writeup with a comparison that discusses the similarities and differences between your 3 systems to ZHT. Create a table of features that clearly shows how these systems are different. Evaluate these systems not just on performance, but scalability, and ease of use. This section should be 2 to 3 pages long. Too short, and you will lose points for not enough detail. Too long, and you will also lose points for being verbose.

1.4 Evaluation Scale and Metrics

Read this paper on ZHT [1]. Particularly, Figure 8 and Figure 10 are very important. You will have to conduct enough experiments to generate 2 figures, 1 for latency and 1 for throughput, with each data point representing the average across all 3 operations (insert, lookup, and remove).

On each instance/node, a client-server pair is deployed. Test workload is a set of key-value pairs where the key is 10 bytes and value is 90 bytes. Clients sequentially send all of the key-value pairs through a client API for insert, then lookup, and then remove. Your keys should be randomly generated, which will produce an All-to-All communication pattern, with the same number of servers and clients.

The metrics you will measure and report are:

• **Latency**: Latency presents the time per operation (insert/lookup/remove) taken from a request to be submitted from a client to a response to be received by the client, measured in milliseconds (ms). Note that the latency consists of round trip network communication, system processing, and storage access time

• **Throughput**: The number of operations (insert/lookup/remove) the system can handle over some period of time, measured in Ops/s

Make the necessary plots to visualize your data. Explain why your results make sense; what explicit things did you do to verify that your performance as you expected? If there are differences in your results compared to those published in [1], discuss why these differences might make sense. What changes would you have to do to your experiments to validate your theories about the differences in your results?

2 Where you will submit

Submit code/report through GIT. To submit your work, simply commit your PDF file and push your work to Github. You will have to submit your solution to a private git repository created for you at git@github.com:datasys-classrooms/cs553-fall2022-hw7-<team name>.git. All repositories will be collected automatically every day at midnight. The repository is created through GitHub Classroom and you will need to accept the assignment before you can clone it at https://classroom.github.com/a/kEOe9k0o. You will also need to create a new team or join an existing team. Your submission will not be graded unless you submit a confirmation document through BlackBoard (BB) that clearly shows the pushing of your final homework to your GIT repository. This confirmation document can simply be a screen shot of your final commands to push your repository to GIT. The timestamp on the BB submission will be used to determine if the submission is on-time. ***You must also include the names and A# of all your team members in this confirmation document.*** If you cannot access your repository contact the TAs. You can find a git cheat sheet here: https://www.git-tower.com/blog/git-cheat-sheet/.

3 What you will submit & Grading

When you have finished implementing the complete assignment as described above, you should submit your solution to your private git repository. Each program must work correctly and be detailed in-line documented. You should hand in:

1. **Source Code:** You must hand in all your source code of any scripts you used to automate your performance evaluation. If you wrote any programs specific to each system in order to perform the evaluation, include these evaluation drivers.
2. **Report:** You must write a project report to cover all the requirements in this assignment; you will be scored on completeness, correctness, organization, and visual appeal.

Submit code/report through GIT.

Grades for late programs will be lowered 10% per day late.