CS 550 Project

An Empirical Evaluation of Distributed Key/Value Storage Systems

Instructions:

- ***Assigned: Wednesday 11/13/19***
- ***Due date: 11:59PM on Tuesday, 12/03/19***
- ***Maximum Score: 100 points***
- ***Extra Credit: 10 points***
- *This project can be done in groups up to 3 students*
- *Please post your questions to the Piazza forum*
- *Only a softcopy submission is required; it will automatically be collected through GIT after the deadline; email confirmation will be sent to your HAWK email address*
- *Late submission will be penalized at 20% per day; an email to the TA with the subject "CS550: late homework submission" must be sent*

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 m1.medium instances.

1.2 Systems to Evaluate

You must choose 1 of the following 2 distributed key/value stores to evaluate:

- Cassandra (http://cassandra.apache.org)
- Memcached (https://memcached.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 chosen systems to ZHT

Write a paragraph about each of your 3 systems you chose, 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 chosen 3 systems to ZHT. Create a table of features that clearly shows how these systems are different. This section should be about 2 pages long (+/- 0.5 pages). 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?

EXTRA CREDIT (10 points)

Conduct additional experiments for each of your chosen systems to be able to generate CDF (cumulative distribution function) plots. An example of such results can be found in a follow-up paper on ZHT (http://datasys.cs.iit.edu/publications/2015_CCPE-zht.pdf), in figure 10. Please make a plot that shows the CDF of each system at 8 node scales (there should be 3 lines, 1 per system).

1.4 Oral Presentation

You must present your project results in an oral presentation. Presentations should be self-contained, have a title, author list, motivation, proposed work, evaluation, conclusions, and references. You will be giving a live oral presentation of your poster in 7 min and participate in a Q&A session for 3 min on November 25th during class (with an extension into the lunch break until); expect to stay from 11:25am-1:40pm. For online students, you must record a 7-minute video presenting your project. Absence from these live oral presentations will result in a 0 for the oral presentation score. There will be no makeup oral presentation.

2 Where you will submit

You will have to submit your project to a private git repository created for you at git@matrix.cs.iit.edu:cs550-2019/group-<id>.git. A directory structure for all the repositories will be created that looks like this: pa1/, pa2/, proj/, ... You will have to firstly clone the repository. Then you will have to add or update your source code, documentation and report. Your solution will be collected automatically after the deadline. If you want to submit your homework later, you will have to push your final version to your GIT repository and you will have let the TA know of it through email. There is no need to submit anything on BB for this assignment. 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.
3. **Oral Presentation:** You will have to present your project through a live Q&A session on November 25th during class; for online students, you must record a video of up to 7-minutes long presenting your project.
4. **Extra Credit (10 points):** Include CDF plot for comparing systems

Submit code/report through GIT.

Grades for late programs will be lowered 20% per day late.