Benchmarking Availability of Large Scale Data Storage Applications

CS848 Project Proposal

Wei Sun and Alexander Pokluda

October 28, 2013

Introduction and Motivation

In the past several years, a new generation of applications like Social networking, Web 2.0 and Business Intelligence developed rapidly. The traditional relational database (RDBMS) design has been found to be inadequate for handling the huge amounts of unstructured data generated and analyzed by these applications. A growing number of users have turned to a new class of large scale data storage applications that relax the ACID guarantees of RDBMSs and instead provide only the basic functions of persistent storage: Create, Read, Update and Delete (CRUD). These systems have come be be known as NoSQL systems. Compared to RDBMSs, NoSQL systems are designed to handle huge amounts of data with high availability, performance and scalability. Over 100 well-known NoSQL systems exist including Cassandra, MongoDB, CouchDB, Redis, Riak, Membase, Neo4j and HBase [9]. All of these systems offer overlapping features but differ in their design and implementation. System benchmarks are essential for evaluating different designs and implementations and ensuring that these systems meet their stated goals. System benchmarks also enable engineers and system architects to select the most appropriate storage application for a given project.

Benchmarking large scale distributed databases is challenging and an active area of research. The open-source Yahoo! Cloud Serving Systems Benchmark (YCSB) has become the industry standard benchmarking tool for NoSQL systems. YCSB was introduced in a paper from Yahoo! in which they presented benchmark results for four widely-used systems: Apache Cassandra, Apache HBase, Yahoo!'s PNUTS and a sharded MySQL implementation [6]. YCSB is an extensible and generic framework for the evaluation of key-value stores. It allows synthetic workloads to be generated that consist of a configurable distribution of CRUD operations on a set of records. Many other industry and academic groups have used the the YCSB to compare the performance of NoSQL systems including Aerospike, Cassandra, Couchbase, HBase, MongoDB and Riak [2, 3, 4, 5]. An influential paper presented at the 2012 VLDB conference by researchers from the University of Toronto provided an up-to-date comparison of the throughput of Cassandra, HBase, Voldemort, Redis, VoltDB, and a MySQL cluster, which were chosen as a representative set covering a broad area of modern storage architectures [1]. This study found that Voldemort had the lowest latency while Cassandra had the highest throughput. Despite the fact that these systems have all been designed for high availability, existing benchmark comparisons focus almost exclusively on latency and throughput under blue-sky scenarios and do not evaluate metrics for availability directly. In this project, we will benchmark several NoSQL systems in the presence of simulated network and node failures.

Problem Description

Objectives

Our main objective is to provide a measure of the level of fault tolerance provided by NoSQL systems and quantitative comparison between them. To the best of our knowledge, a thorough evaluation and comparison of the availability of each of these systems has not been done. Evaluating the availability of each system is a challenging problem and not straightforward. To begin with, Brewer's Theorem, also known as the CAP Theorem, states that any distributed system must make tradeoffs between consistency, availability and performance. Only two of these metrics can be achieved at any one time and at the expense of the third. Most of the systems mentioned previously emphasise performance and either consistency or availability at the expense of the other. In order to ensure that we are making an apples-to-apples comparison, we will start by evaluating Cassandra and Voldemort, as both of these are AP systems in the CAP parlance. If time permits, we will evaluate HBase using our framework, although we will likely not be able to compare these results to the Cassandra and Voldemort results because HBase is a CP system. Our Methodology, Milestones and Metrics for Success follow.

Methodology

We will complete both a research phase and an experimental phase before summarizing our results and completing our write up. Our research phase will involve investigating how our chosen NoSQL systems attempt to ensure availability of data in the presence of physical equipment failures and planning for the experimental phase. In the experimental phase, we will use our $100 AWS credits to setup and testbed and carry out experiments using the YCSB.

Milestones

1. Investigate how the different NoSQL systems (attempt to) ensure data availability in the presence of equipment failures. The goals of this step are to gain insight into how NoSQL systems cope with failures in order to tune our workload and failure scenarios as well as interpret and compare our results.

2. Research, develop and evaluate different metrics for measuring availability such as the percentage of reads and writes that fail in the presence of network and node failures.

3. Decide on network and node failure scenarios and how to simulate them. This is a major issue that will require a lot of careful attention. The scenarios chosen must accurately reflect the types of real-life failures observed in production data centres. A simple node failure can be simulated by killing a server while a workload is running. However, in a real deployed system, a larger variety of errors are likely to occur, including disk, network, and data centre failures. As suggested by the YCSB paper, network failures could be simulated by implementing fault injection in the workload generator or by using a simulated network layer such as Modelnet [8] or Emulab [7].

4. Establish workload parameters. We will use the latency and throughput benchmarks cited previously as guides.

5. Set up a testbed on AWS EC2 instances using the Datastax benchmark configuration as a guide [3]. As done for the datastax benchmark, we will also run our experiments at 3 different times on 3 different days to minimize the effect of CPU and I/O variability. To further reduce the impact of any “lame instance” or “noisy neighbour” effect, the Datastax benchmark use freshly created instances for each test run. We will also follow this approach if time permits.

6. Run experiments for each NoSQL system in each failure scenario using the YCSB. Organise the experimental data into plots and/or tables.

7. Summarize the results and complete our write up.

Metrics for Success

The YCSB paper states that benchmarking availability is challenging and that is one of the reasons why it has been left as “future work.” Although the procedure outlined above appears straight forward, we expect to encounter many difficulties along the way. In addition to completing research, we must learn how to setup, configure, and tune several different complex software systems including the YCSB tools. At this point, we are not even sure if the YCSB will give us the availability metrics that we desire, or whether we will need to implement our own way of evaluating availability. Given that we have very tight time constraints (approximately one month) and that we both have other coursework, research and TA obligations, we will consider our project successful if we are able to provide some availability metrics for one NoSQL system.

References

Academia Performed Benchmarks

[1] Rabl, T., Gómez-Villamor, S., Sadoghi, M., Muntés-Mulero, V., Jacobsen, H. A., & Mankovskii, S. (2012). Solving big data challenges for enterprise application performance management. *Proceedings of the VLDB Endowment*,*5*(12), 1724-1735.

Industry Performed Benchmarks

[2] Bushik, Sergey. (2012, October). A vendor-independent comparison of NoSQL databases: Cassandra, HBase, MongoDB, Riak. *Network World*.
http://www.networkworld.com/news/tech/2012/102212-nosql-263595.html?page=1

[3] DataStax Corporation. (2013, February). Benchmarking top NoSQL databases.
http://www.datastax.com/resources/whitepapers/benchmarking-top-nosql-databases

[4] Thumbtack Technology. (2013, March). NoSQL failover Characteristics: Aerospike, Cassandra, Couchbase, and MongoDB. http://thumbtack.net/whitepapers.html

[5] Thumbtack Technology. (2013, January). Ultra-high performance NoSQL benchmarking: analyzing durability and performance tradeoffs. http://thumbtack.net/whitepapers.html

Benchmarking and Network Simulation Tools

[6] Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., & Sears, R. (2010, June). Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM symposium on Cloud computing* (pp. 143-154). ACM.

[7] Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., ... & Lepreau, J. (2008, June). Large-scale Virtualization in the Emulab Network Testbed. In *USENIX Annual Technical Conference* (pp. 113-128).

[8] Vishwanath, K. V., Gupta, D., Vahdat, A., & Yocum, K. (2009, September). Modelnet: Towards a datacenter emulation environment. In *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on* (pp. 81-82). IEEE.

NoSQL Storage Systems

[9] NoSQL. (2013, October 24). In *Wikipedia, The Free Encyclopedia*. Retrieved 22:19, October 25, 2013, from http://en.wikipedia.org/w/index.php?title=NoSQL&oldid=578498311.