42

An Empirical Evaluation of Distributed Key/Value Storage Systems on Amazon EC2

Abhijeet Kedari (A20353562)

Department of Computer Science, Illinois Institute of Technology

ILLINOIS INSTITUTE OF TECHNOLOGY

Abstract

A "Simple Distributed Hash" Table was designed to store the Key-value pair across the multiple systems. Distributed Hash Table (DHT) was designed to compete with high-end computing system. We compared it with Centralized P2P system and found improved performance when number of requests increases. Objective for DHT is to low the Latencies, increase the throughputs. Simple DHT performs Put, Get and Delete operations. DHT is tested on 16 nodes of Amazon EC2 cloud and compared with different systems on M3.medium 16 nodes. Objective was to test the system performance under the high collision rate of requests. We performed 10K operations with 0.47ms Latency and 11111 ops/seconds throughput.

Objective

To benchmark throughput and latency of DHT with Mongo DB, Redis and Couch DB on Amazon EC2

Experiment Setup

Commercial Cloud

- M3.medium (1 compute node)
- Amazon EC2
- 16 Ubuntu 14.01
- 16 DHT instances

Benchmark Settings

- One or more DHT client-server pairs per node
- N clients to N servers: N-to-All communication pattern
- Random generated key–value pairs: 10 bytes key, 90 bytes value

Distributed key/value storage systems

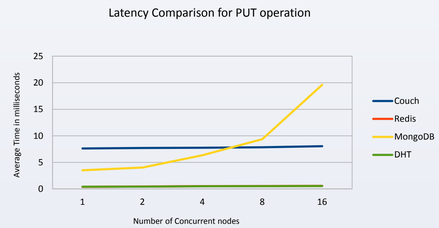
- Mongo DB
- Redis
- Couch

Latency and Throughput comparison

	DHT	Redis	MongoDB	Couch
Latency	Low	Low	High	High
Throughput	High	High	Low	Low

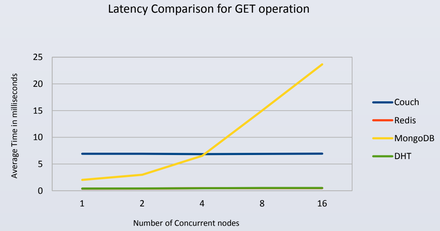
- DHT and Redis performs better when load on server increases
- MongoDB and Couch are having high latency compred to DHT and Redis
- DHT and Redis has high Throughput compared to Mongo DB and Couch

Latency for PUT operation



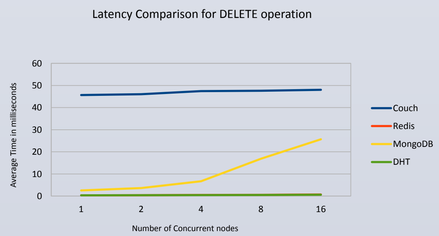
- Couch, Redis and DHT system has constant average latency irrespective of number of concurrent nodes
- Mongo DB's has exponential Latency curve

Latency for GET Operation



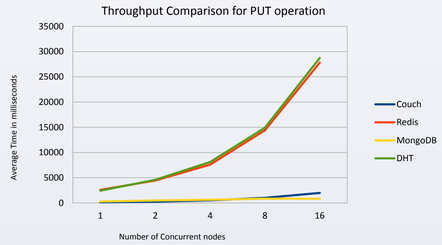
- Couch, Redis and DHT system has constant average latency irrespective of number of concurrent nodes similar to PUT operation
- Mongo DB's has exponential Latency curve same as PUT operation

Latency for DELETE Operation



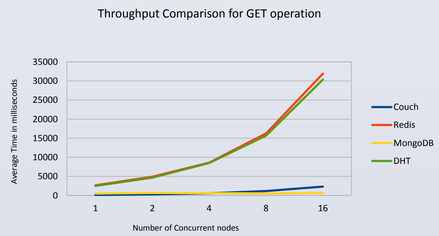
- Couch, Redis and DHT system has constant average latency irrespective of number of concurrent nodes
- Mongo DB's has exponential Latency curve
- But Couch has Highest latency for delete operation, When you delete a document the database will create a new revision which contains the _id and _rev fields as well as a deleted flag.

Throughput for PUT Operation

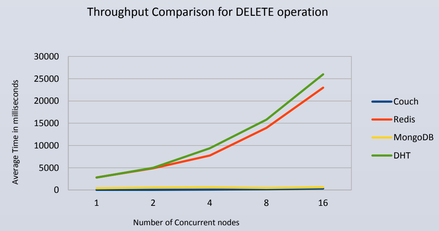


- Redis and DHT system has exponential curve for throughput compared to MongoDB and Couch.
- Mongo DB and Couch has similar throughput but if we compare to with Redis and DHT, they are not increases with increase in number of concurrent nodes

Throughput for GET Operation

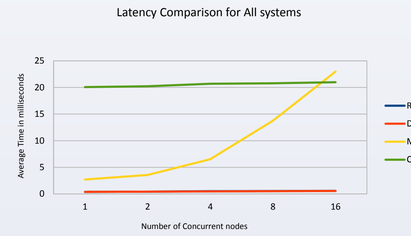


Throughput for DELETE Operation

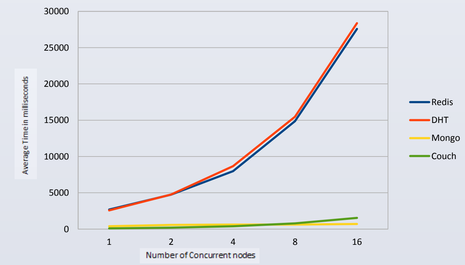


- Similar to Put operation Get and Delete operation has high throughput for DHT, Redis
- Couch and Mongo DB's has constant performance irrespective of number of concurrent operation

Latency of All System



Throughput of All System



- Couch DB has highest latency but it is constant w.r.t number of concurrent nodes.
- But if we keep increase the number of concurrent nodes Mondo DB performed worst than Couch DB.
- All the compared database have there different use case

Conclusion

- On comparing DHT with different distributed hash table system. DHT is having a good throughput and low latency for small number of nodes.
- In future before we evaluate DHT system with higher nodes we will need to update few changes, like we need to keep the client connected to all server all the time, there was no resilience implemented and DHT system need to have fault tolerance.

Related Work and References

- Tonglin Li, Xiaobing Zhou, Kevin Brandstatter, et al. ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table, IPDPS, 2013 B. Fitzpatrick. "Distributed caching with Memcached." Linux Journal, 2004(124):5, 2004
- http://couchdb.apache.org
- http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis
- http://datasys.cs.iit.edu/events/ScienceCloud2013/p02.pdf
- https://en.wikipedia.org/wiki/Redis
- https://docs.mongodb.org/getting-started/java/