72

An Empirical Evaluation of Distributed Key/Value Storage Systems

Nikhil Nere (A20354957)
Computer Science Department
Illinois Institute of Technology, Chicago

Abstract

The aim of the project is to evaluate and compare the performance of Distributed hash table (Distributed Key-value store) implemented in programming assignment 2 with different distributed key-value storage systems. The DHT is implemented in java using sockets and multithreading. Each node in the system acts as a server and a client. The performance of DHT is evaluated on Amazon EC2 m3-medium instances. The known systems – MongoDB, CouchDB, Redis are also ran in the same environment and their performance is compared with DHT.
The systems are compared based on the latency and throughput. Same amount of workload is applied on all the systems in same environment. Multiple experiments are run by increasing the number of concurrent clients with same work load.

Test Environment :

Amazon EC2 m3.medium
- High frequency Intel Xeon E5-2670 v2 processor
- SSD-based instance storage for fast I/O performance
- 1 vCPU
- Mem - 3.75 GiB
- SSD Storage (GB) – 1 x 4
- OS – Ubuntu 14.0.4 LTS

Test Settings:

- 16 DHT nodes.
- Each node has one client-server pair
- Random keys of size 10 bytes and values of size 100 bytes

Known Systems:

MongoDB:
- It's a cross platform document-oriented database.
- Classified as a NoSQL database, Retains some friendly properties of SQL.
- Written in C++

Redis:
- It's a data structure server.
- Typically holds dataset in memory.
- Faster access to data
- Written in C

CouchDB:
- Document oriented NoSQL database that completely embraces the web.
- Uses JSON to store data
- Javascript as query language using MapReduce
- Written in Erlang

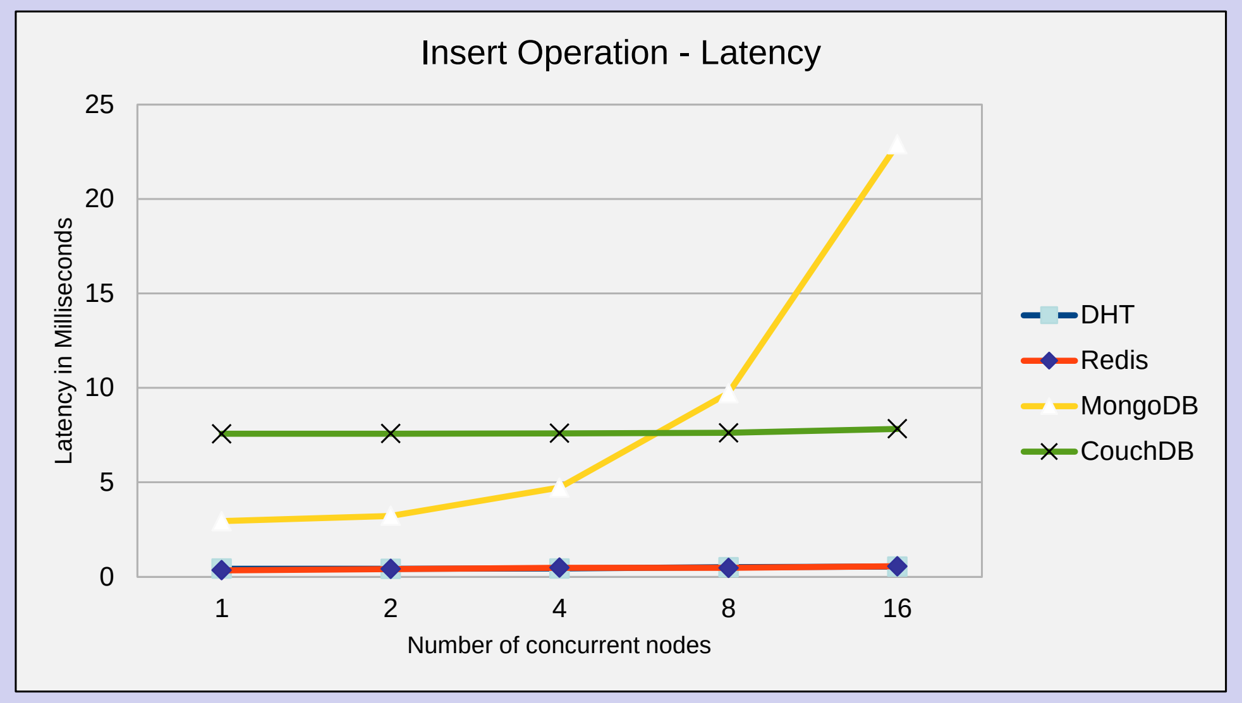
Performance Measures:

Latency: Time per operation taken from a request to be submitted from a client to a response to be received by the client, measured in milliseconds.

Throughput: The number of operations the system can handle over a period of time, measured in operations per seconds.

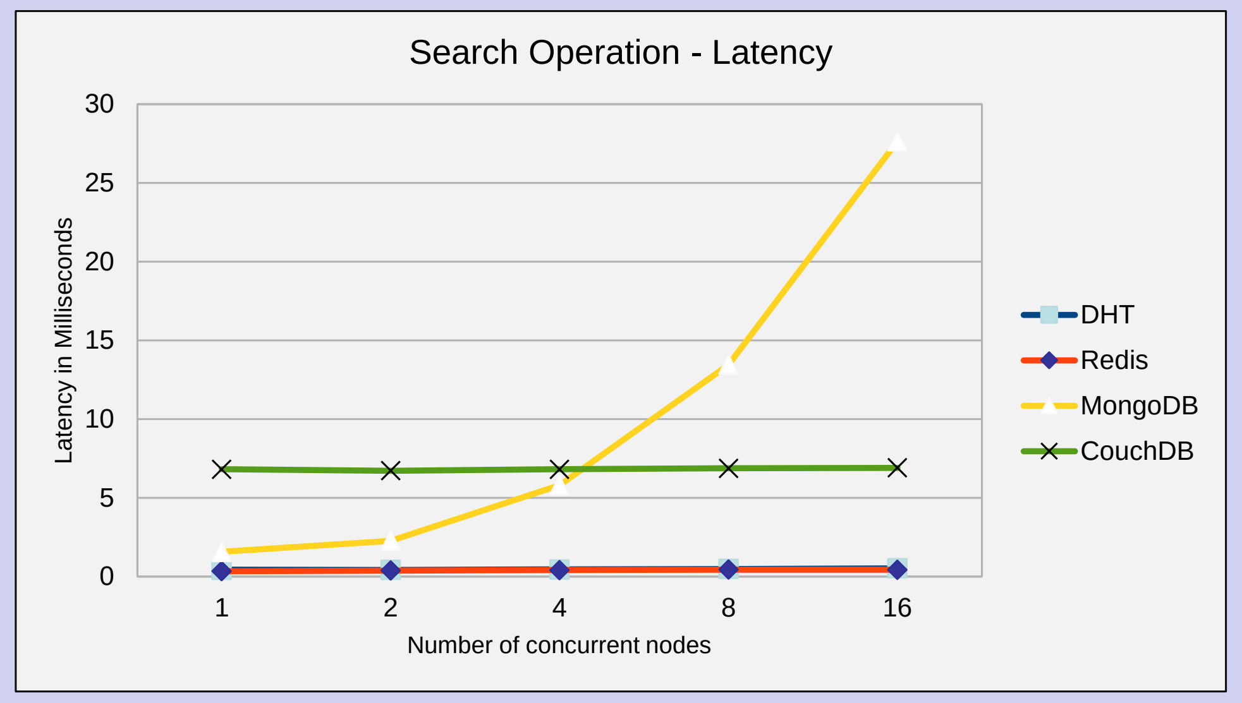
Latency per Operation

Latency of Insert Operation of all Systems:



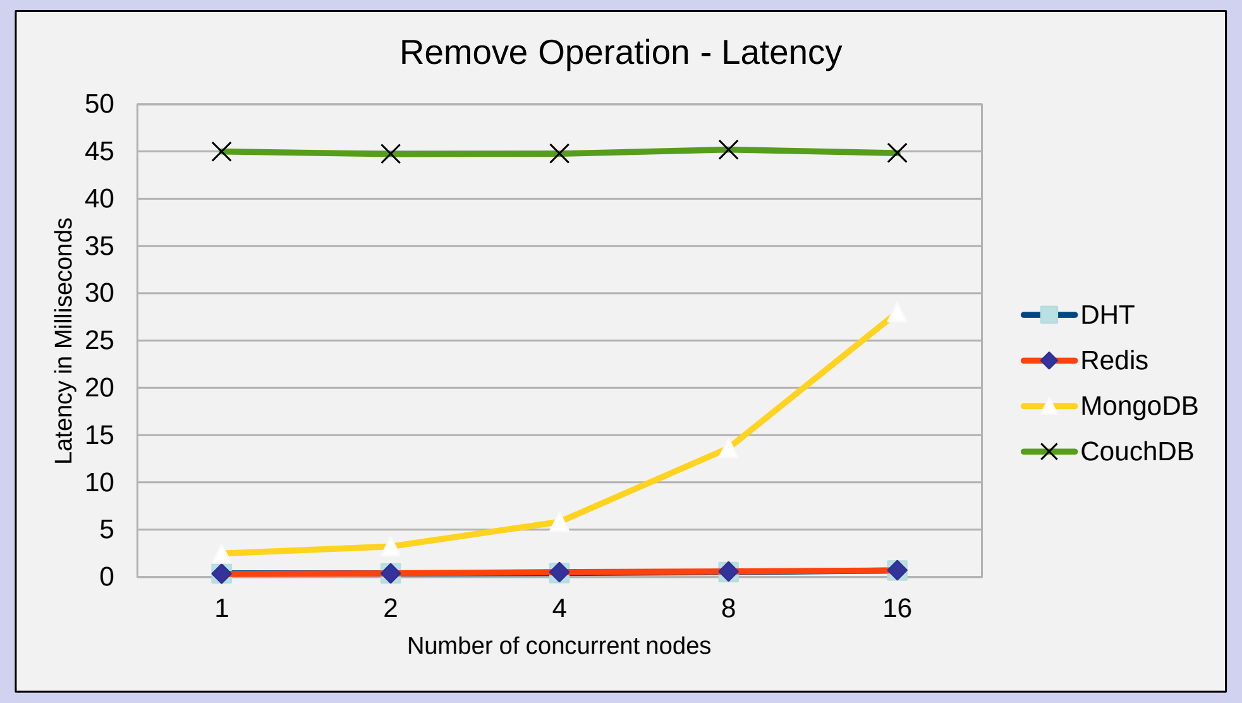
- The latency of DHT, Redis and CouchDB has a same trend with a very negligible change in latency with increased scale.
- The latency of DHT and Redis is very low compared to MongoDB and CouchDB

Latency of Search Operation of all Systems:



- The latency of DHT, Redis and CouchDB has a same trend with a very negligible change in latency with increased scale.
- Overall the latency of DHT and Redis is very low compared to MongoDB and CouchDB

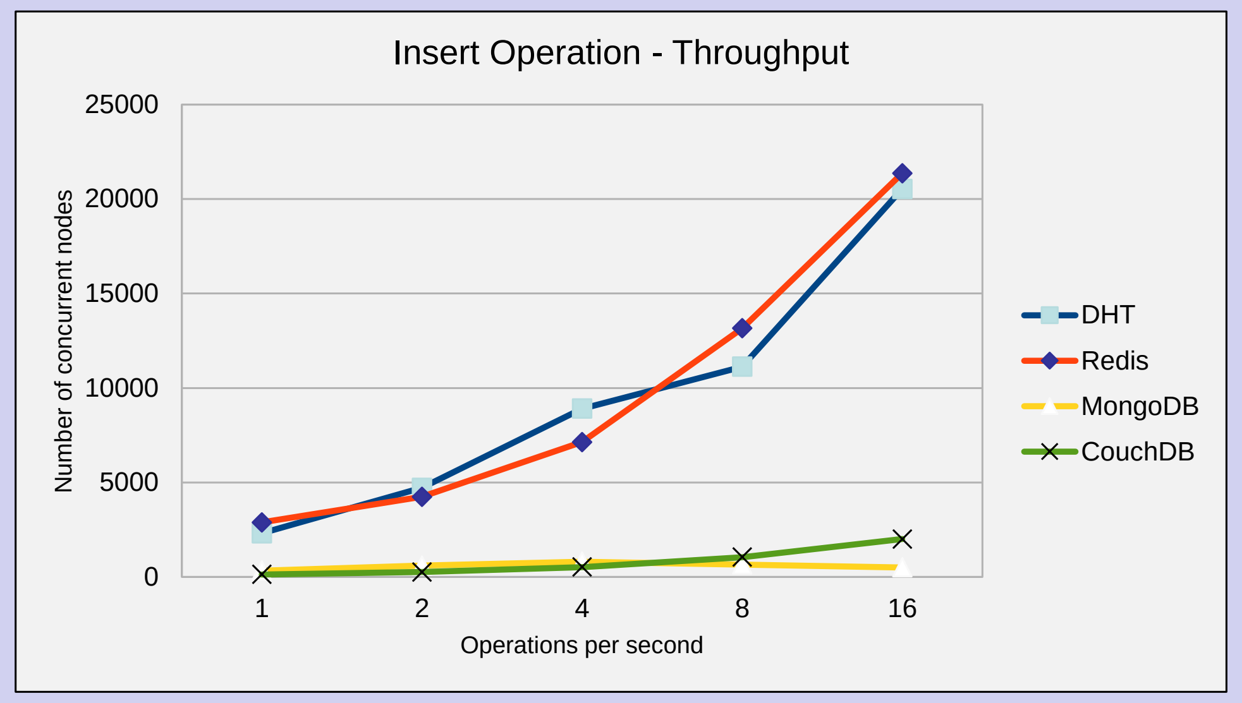
Latency of Delete Operation of all Systems:



- For Search operation DHT and Redis has lowest latency same as in case of insert and search.
- The latency of CouchDB is a way higher than other system this is because the delete request requires two http calls to be made. One for search and other for delete.

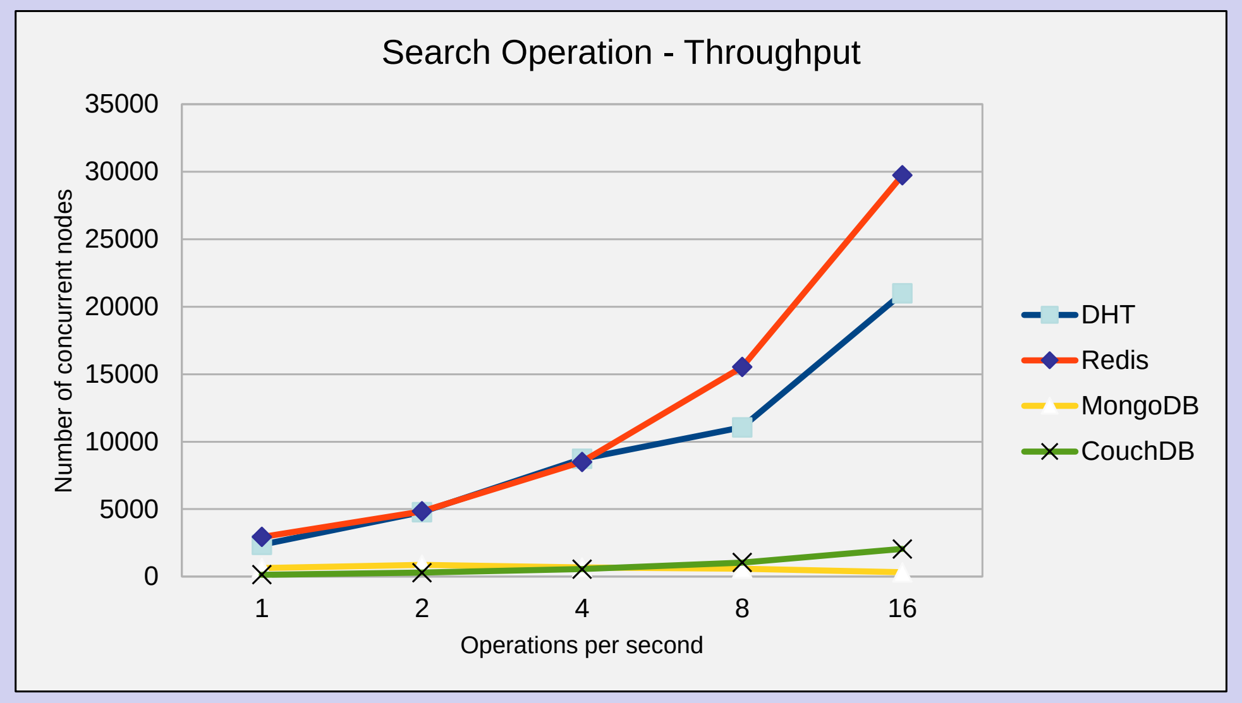
Throughput per Operation

Throughput of Insert Operation of all Systems:



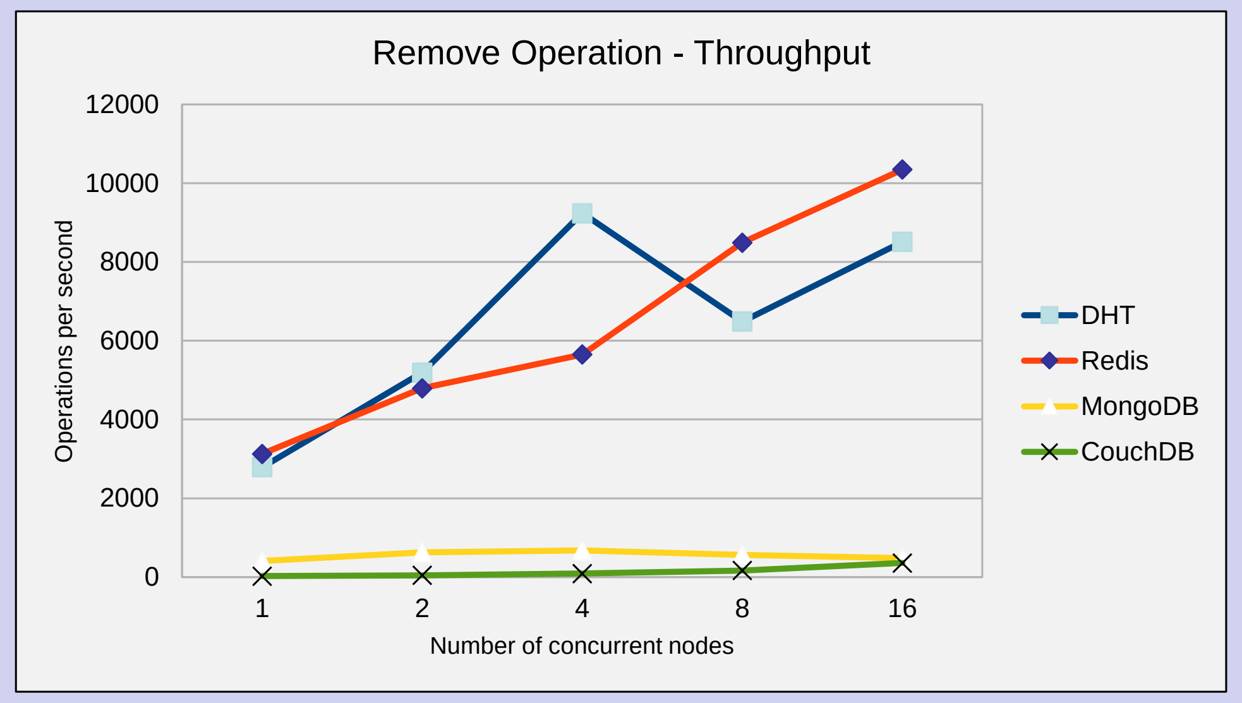
- The throughput of CouchDB and mongoDB has a same trend with a small change in throughput with increased scale.
- The throughput of DHT and Redis increases with increase in scale.

Throughput of Search Operation of all Systems:



- The throughput of CouchDB and mongoDB has a same trend with a small change in throughput with increased scale.
- The throughput of DHT and Redis increases with increase in scale.

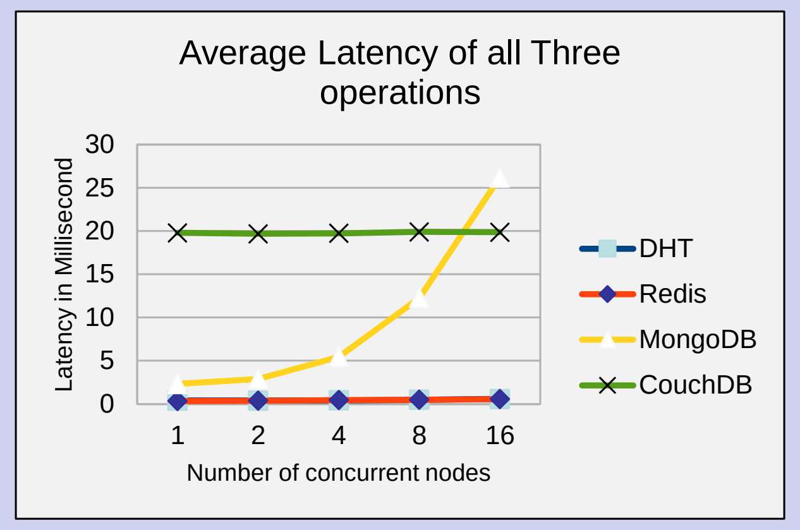
Throughput of Delete Operation of all Systems:



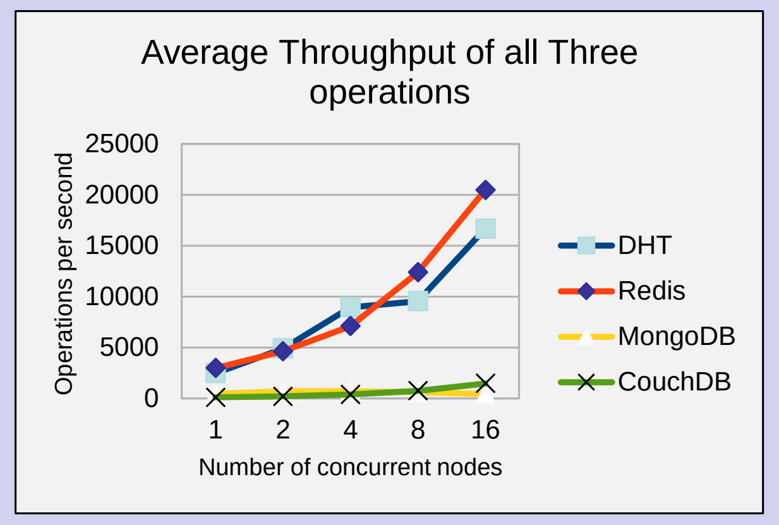
- The throughput of CouchDB and mongoDB has a same trend with a small change in throughput with increased scale.
- DHT has a non uniform behavior at some point in the experiment. But overall the throughput of DHT is increasing with increasing in scale.

Avg Latency and Avg Through

Avg Latency of all Systems:



Avg Throughput of all Systems:



- Overall DHT and Redis has low latency compared with MongoDB and Couch.
- The throughput of DHT and Redis increases with increase in number of concurrent clients.

Conclusion

- DHT has a similar behavior as Radis. DHT has low latency compared with MongoDB and CouchDb
- Throughput of DHT increases same as Radis with increase in number of concurrent clients. The distributed nature of DHT performs better by handling concurrent requests which are processed by different servers at the same time.
- CouchDB accepts the requests over http. For every operation the http request is made. For Delete operation CouchDB has maximum latency this is because before delete a search request is made. Hence for a delete requests it requires two http requests. The delete requests is sent after the response of search request is received.
- MongoDB shows exponential increase in latency with increase in number of concurrent clients because the mongoDB client fetches all the key-values before every operation. So with increased load more and more data is fetched for a simple get/put/delete operation

Reference

- http://datasys.cs.iit.edu/reports/index.html
- https://www.mongodb.org
- http://couchdb.apache.org
- http://redis.io
- http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis
- https://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/
- http://api.mongodb.org/c/current/tutorial.html#starting-mongod