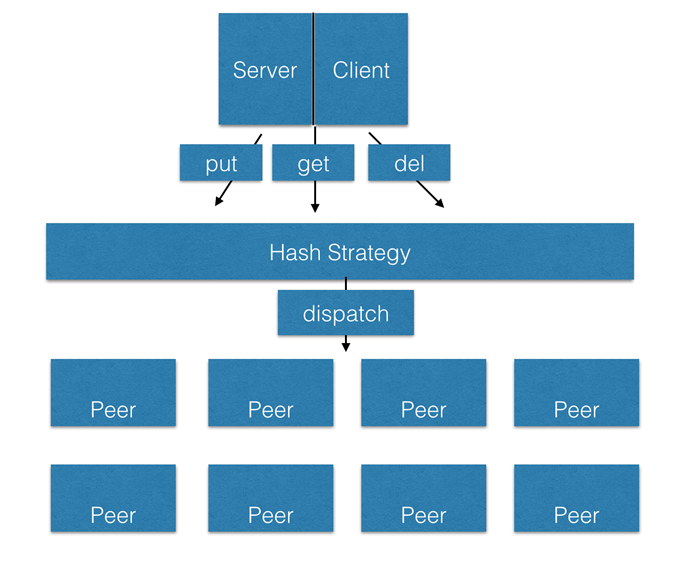
Abstract

My design is a distributed hash table system. It has no central server, each peer is both a server and a client. As a client, it provides interfaces through which users can issue queries and view search results. As a server, it accepts queries from other peers, checks for matches against its local hash table, and responds with corresponding results. In addition, since there’s no central indexing server, search is done through consistent hashing. In this evaluation, it tells it works not bad on cloud environment from latency and throughput.



**Experiment setup**

**Commercial Cloud**

Amazon EC2

M3.medium

**Commodity Cluster**

Up to 16 nodes

**Micro benchmark settings**

N clients to N servers: All-to-All communication pattern

Random generated key–value pairs: 10bytes key, 90 bytes value

**Latency**

**throughput**

**Conclusion**

according to the above picture it shows that my design has a good performance and scalability , also we find out that redis has the best performance through all the four system, it has a latency less than 2ms even in 16 nodes. and mongoDB has a bad scalability, the latency will increase very fast when the nodes numbers increase.

**Related Work and References**

Tonglin Li, Xiaobing Zhou, Ke Wang, Dongfang Zhao,Iman Sadooghi, Zhao Zhang, Ioan Raicu

A Convergence of Key-Value Storage Systems from Clouds to Supercomputers,Wiley InterScience

Tonglin Li, Xiaobing Zhou, Kevin Brandstatter, et al.

ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table, IPDPS, 2013

Redis: <http://redis.io/documentation>

MongoDB:https://[docs.mongodb.org/manual](http://docs.mongodb.org/manual)

Amazon DynamoDB: https://aws.amazon.com/dynamodb/developer-resources/