11

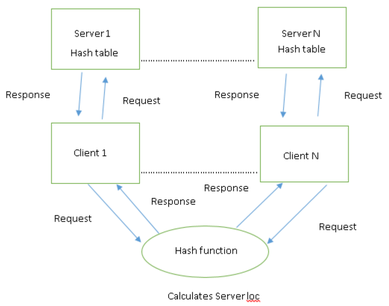
Evaluation of a Decentralized Peer to Peer File Transfer System – ACDHT

Chaudhary Anuradha S A20361279

Abstract:

ACDHT is a decentralized p2p fts implemented using DHTs. It supports operations like put, get, delete and obtain. The goals of ACDHT are delivering high availability, good fault tolerance, high throughput and low latencies at large number of nodes. ACDHT has some important properties like allowing nodes to join or leave dynamically, fault tolerance thru replication and scalable. On evaluating the system on a small scale it was found that ACDHT easily scaled up to 16 Nodes on Amazon EC2 cloud with 620 ms latency and 1.565 kops/sec throughput.

Architecture and Design:



Features:

Fault tolerance: replication support.
Obtain operation: allows user to download a file from respective peer.

Experiment Set Up:

Commercial Cloud

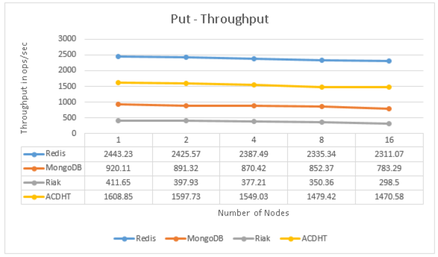
- Amazon EC2
- M3 medium instances

Commodity Cluster

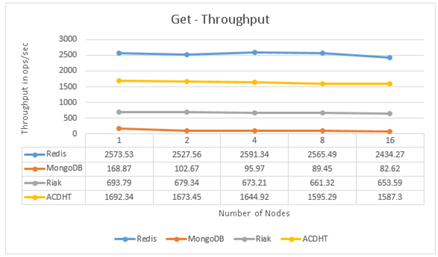
- 16 nodes

Micro benchmark settings

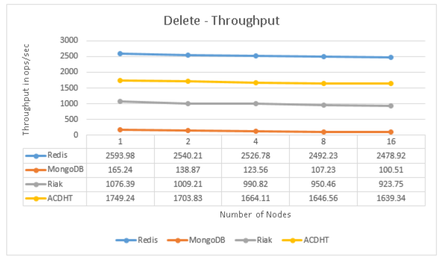
- One or more ACDHT client-server pairs per node.
- Random key-value pairs- 10 byte key and 90 byte value.



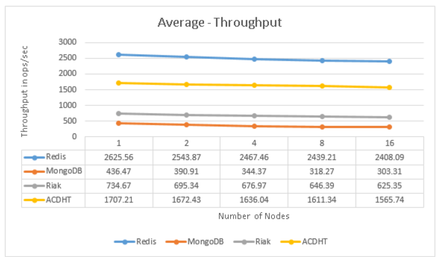
	1	2	4	8	16
Redis	2443.23	2425.57	2387.49	2335.34	2311.07
MongoDB	920.11	891.32	870.42	852.37	783.29
Riak	411.65	397.93	377.21	350.36	298.5
ACDHT	1608.85	1597.73	1549.03	1479.42	1470.58



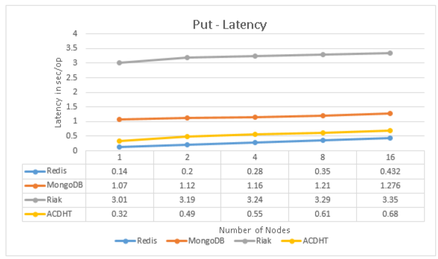
	1	2	4	8	16
Redis	2573.53	2527.56	2591.34	2565.49	2434.27
MongoDB	168.87	102.67	95.97	89.45	82.62
Riak	693.79	679.34	673.21	661.32	653.59
ACDHT	1692.34	1673.45	1644.92	1595.29	1587.3



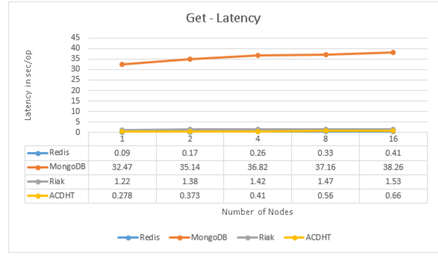
	1	2	4	8	16
Redis	2593.98	2540.21	2526.78	2492.23	2478.92
MongoDB	165.24	138.87	123.56	107.23	100.51
Riak	1076.39	1009.21	990.82	950.46	923.75
ACDHT	1749.24	1703.83	1664.11	1646.56	1639.34



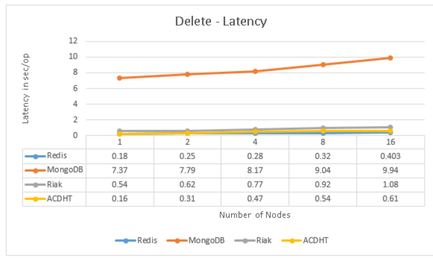
	1	2	4	8	16
Redis	2625.56	2543.87	2467.46	2439.21	2408.09
MongoDB	436.47	390.91	344.37	318.27	303.31
Riak	734.67	695.34	676.97	646.39	625.35
ACDHT	1707.21	1672.43	1636.04	1611.34	1565.74



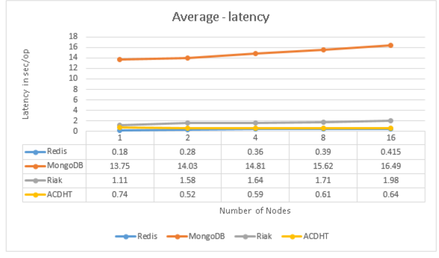
	1	2	4	8	16
Redis	0.14	0.2	0.28	0.35	0.432
MongoDB	1.07	1.12	1.16	1.21	1.276
Riak	3.01	3.19	3.24	3.29	3.35
ACDHT	0.32	0.49	0.55	0.61	0.68



	1	2	4	8	16
Redis	0.09	0.17	0.26	0.33	0.41
MongoDB	32.47	35.14	36.82	37.16	38.26
Riak	1.22	1.38	1.42	1.47	1.53
ACDHT	0.278	0.373	0.41	0.56	0.66



	1	2	4	8	16
Redis	0.18	0.25	0.28	0.32	0.403
MongoDB	7.37	7.79	8.17	9.04	9.94
Riak	0.54	0.62	0.77	0.92	1.08
ACDHT	0.16	0.31	0.47	0.54	0.61



	1	2	4	8	16
Redis	0.18	0.28	0.36	0.39	0.415
MongoDB	13.75	14.03	14.81	15.62	16.49
Riak	1.11	1.58	1.64	1.71	1.98
ACDHT	0.74	0.52	0.59	0.61	0.64

Graph Explanations:

ACDHT is evaluated against MongoDB, Redis and Riak. Graphs show the results for each put, get and del function.

Throughputs

- Throughtput is measured in ops/s that is the no of operations that can be processed in one second.
- It can be seen from the graphs that maximum throughput is obtained for Redis system. This makes sense because Redis bundles the values into hashes which more memory efficient and performant than discrete string storage.
- Also, for each put, get, del and avg throughput it can be seen that with increase in number of nodes the throughput of the system decreases.
- It makes sense because with increase in number of nodes the response time increases as a result of which throughput (No of Operations/second) decreases.

Latency

- Latency is the combined delay between input and desired output. So here latency is measured in Responsetime/no of operations
- From the graphs it can be seen that maximum latency is recorded by MongoDB amongst the 4 systems.
- This implies that MongoDB is the slowest amongst the 4 datasystems evaluated.

Conclusions:

ACDHT has shown pretty good performance and hence can be used for building blocks of several distributed systems.