Comparing Couchbase Server with MongoDB: Benchmark Results and Analysis

Composed by Avalon Consulting, LLC

Introduction

The data needs of today's Enterprise require a special set of tools. At the center of these tools lies the NoSQL database. In recent years, NoSQL has become a growing part of the modern Enterprise infrastructure. Knowing how to implement a highly scalable NoSQL database that fits current and future use cases and scales easily and efficiently is critical in satisfying these ever-growing demands.

It's important to consider performance, scalability, consistency, and availability when selecting a NoSQL database. However, this benchmark focuses exclusively on performance. In an era where applications may have to support millions of users and where users expect faster and faster responses, performance can be the deciding factor between success and failure. A high

performance NoSQL database must be able to maintain low latency at high throughput.

In this white paper, we will identify the performance characteristics of two popular NoSQL databases, Couchbase Server and MongoDB. Through the process of benchmarking, we will illustrate which of these two technologies performs best when hit with a balanced workload of reads and updates and there is not enough memory to cache all of the data in memory. By evaluating how both Couchbase Server and MongoDB react to this workload, we will gain a better understanding of which one may be better suited for today's Enterprise data needs.

The reason we chose to do this benchmark at this time was due to the major release enhancements announced for MongoDB. MongoDB 3.0 is a significant release with major improvements, the most notable being the optional storage engine WiredTiger. MongoDB states a 7-10x improvement of write performance with WiredTiger. While we did not compare WiredTiger to the default storage engine, MMAP, we enabled WiredTiger to determine whether or not it addresses MongoDB performance issues. It's important to understand that there is more to performance than the storage engine, but it is important nonetheless.

Benchmarking/Data Specifications

For this benchmark, an equal number of reads and writes were performed on both Couchbase Server and MongoDB. The amount of data utilized for this benchmark meant that not all data would reside in memory. This was an important attribute of this benchmark, as we wanted to see how Couchbase Server and MongoDB would perform outside of memory. Finally, we were looking for latency to be at or below the 5ms mark. To perform this benchmark analysis, we chose to use Yahoo Cloud Serving Benchmark (YCSB).

Testing Methodology

The goal of this benchmark is to show how Couchbase Server and MongoDB respond to an increasing number of concurrent clients until the read or write latency exceeds 5ms. The attributes we used to determine this were latency and throughput. The 95th percentile was used to record latency. The following table shows how we incremented the request load per test run and how we will store data for 3 runs:

Clients / Threads	Run #1 Throughput / Latency	Run #2 Throughput / Latency	Run #3 Throughput / Latency
2 / 70			
3 / 105			
4 / 140			
5 / 175			
6 / 210			
7 / 245			
8 / 280			
9 / 315			
10 / 350			
11 / 385			
12 / 420			
13 / 455			
14 / 490			
15 / 525			

System Infrastructure

Our infrastructure consisted of 9 i2.2xlarge EC2 instances to run the NoSQL databases:

- 8 vCPU
- 61 GB Memory
- CentOS 6

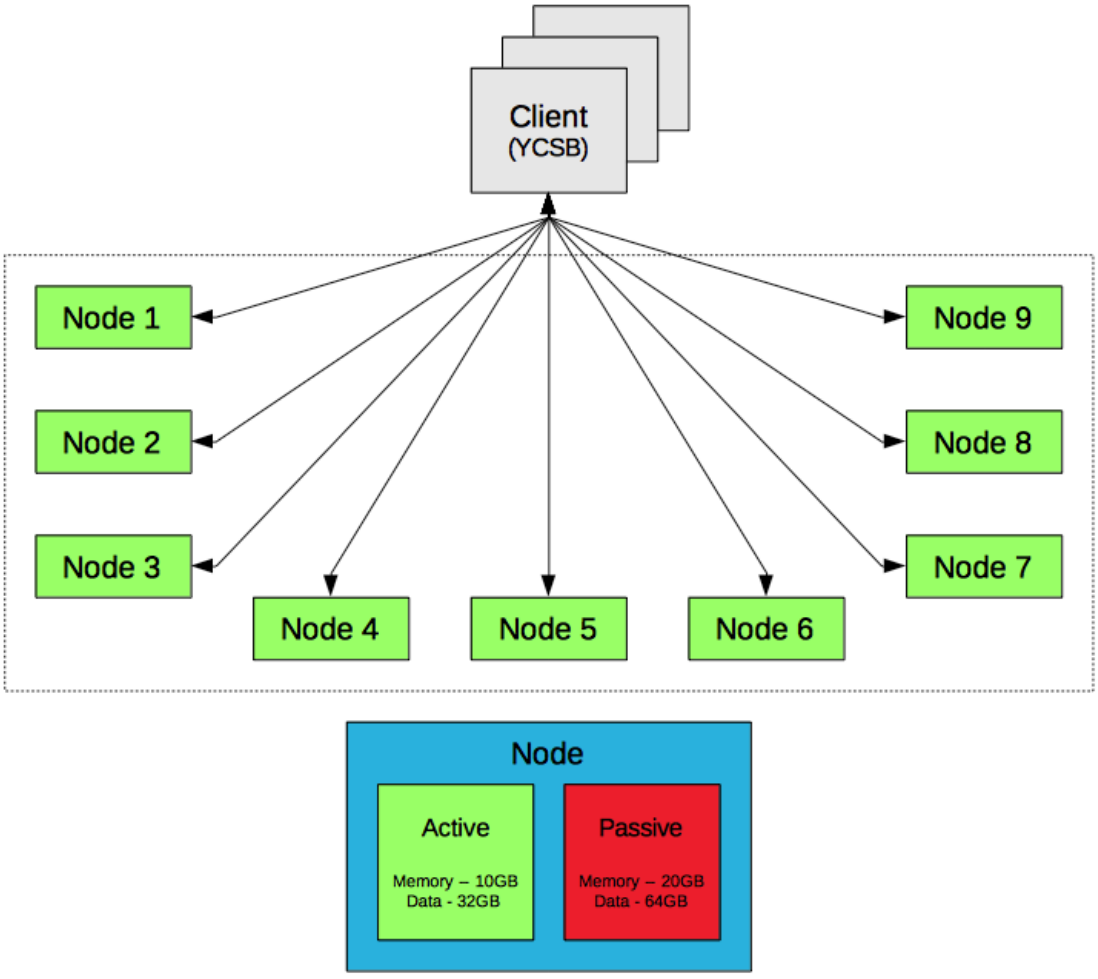
For running the YCSB client threads we used r3.8xlarge for each client instance:

- 32 vCPU
- 244 GB Memory
- Amazon Linux

Other System Configurations

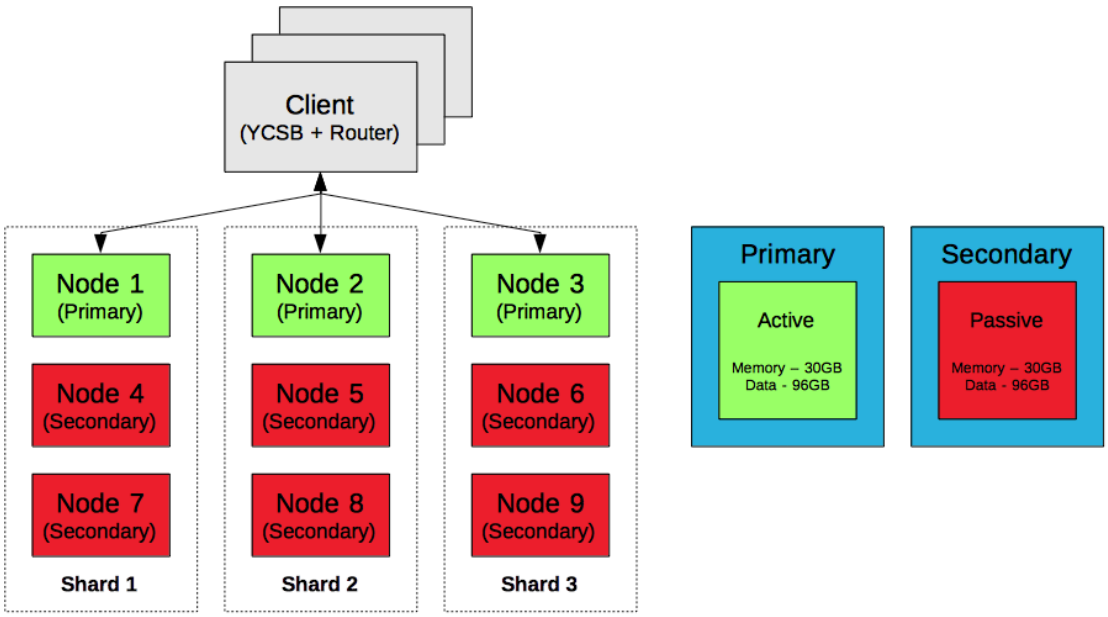
- In order to avoid potential performance issues, numa was disabled on the NoSQL EC2 instances.
- Memory utilization was set for each NoSQL instance in order to capture how Couchbase Server and MongoDB perform outside of RAM.
    - 10GB of memory was used for primary data on all 9 Couchbase Server nodes.
    - 30GB of memory was used for primary data on the 3 MongoDB primary nodes.

Couchbase Server Topology



The Couchbase Server topology is simple. Each client responsible for running YCSB communicated directly with the Couchbase Server nodes. The range of clients that Couchbase Server was able to handle before exceeding the 5ms latency threshold was 2 – 23.

MongoDB Topology



This image shows the MongoDB topology for the benchmark. For running the benchmark, we had YCSB located on the same node as the router. Each client/router node communicates via the configuration server nodes, which contains metadata pertaining to each shard. The range of clients that MongoDB was able to handle before exceeding the 5ms latency threshold was 2 – 7.

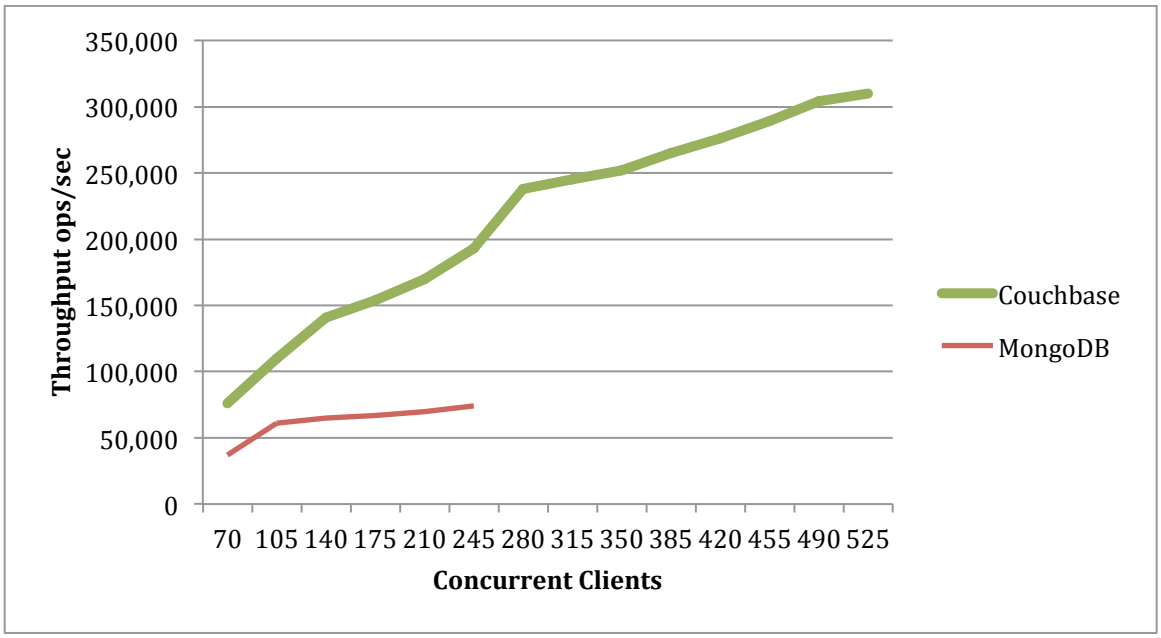
As shown in the topology diagrams for Couchbase Server and MongoDB, Couchbase Server has 3x as many active nodes as MongoDB. In order to get MongoDB to have the 9 active nodes that Couchbase Server has, we would have had to provision 3x the number of servers for MongoDB. When you consider hardware and subscription costs, it would not be fair to do this, as cost to

implement is a very real factor to consider here. This is a clear disadvantage that you must deal with when implementing MongoDB.

Benchmark Results

Throughput

The following are the throughput results for Couchbase Server and MongoDB

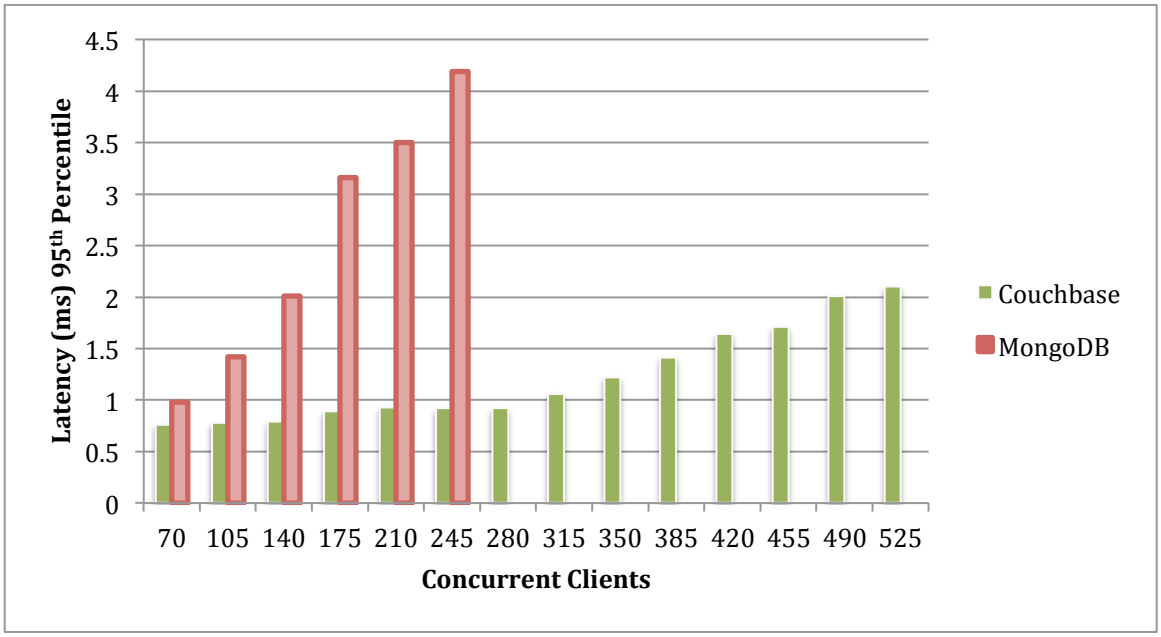


Couchbase Server provided 2.5x the throughput of MongoDB with the same number of concurrent clients - 245. This is where MongoDB exceeded the maximum latency of 5ms. While scalability is important, so is concurrency - the ability for a database to accommodate a high number of concurrent clients before scaling is required. MongoDB was overwhelmed by a 2x increase in the number of concurrent clients, and latency suffered. Couchbase Server, with a 13x increase, showed increased throughput and latency well below the 5ms limit.

	MongoDB	Couchbase Server
245 Concurrent Clients	72K Ops / Sec	186K Ops / Sec

Read Latency (Lower is Better)

The following are the read latency results for Couchbase Server and MongoDB

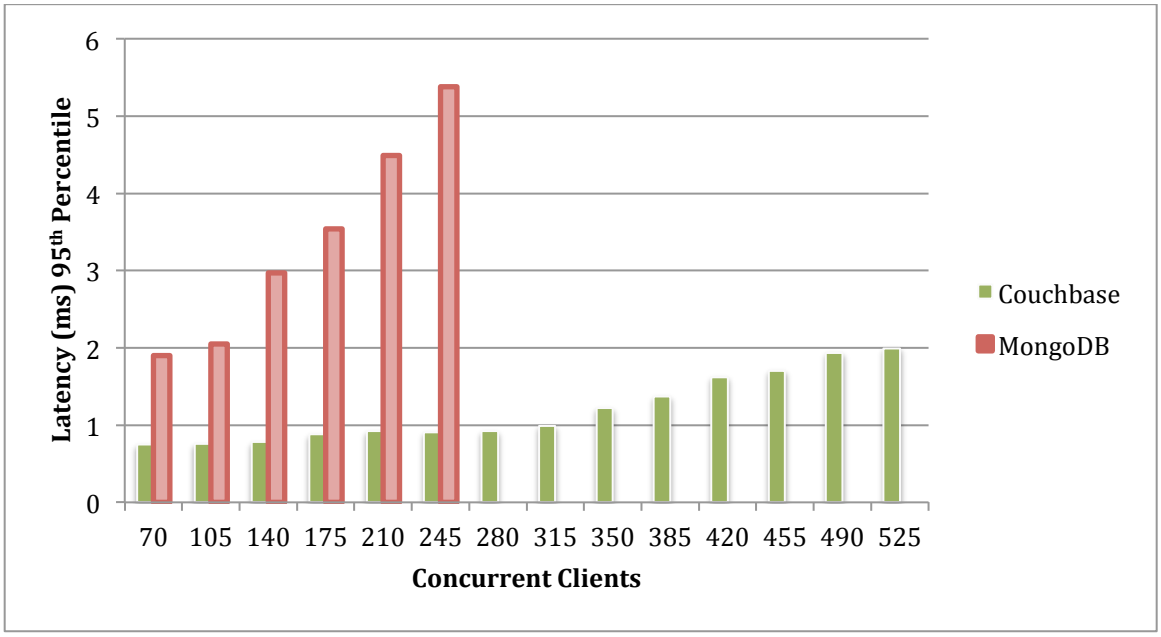


Couchbase Server provided 4x better read latency than MongoDB with the same number of concurrent clients - 245. Like throughput, concurrency is important. MongoDB latency increased by over 50% as the number of concurrent clients was increased by 50%. However, Couchbase Server latency increased by much smaller margins - as little as 10%.

	MongoDB	Couchbase Server
245 Concurrent Clients	4.19ms	.96ms

Update Latency (Lower is Better)

The following are the update latency results for Couchbase Server and MongoDB

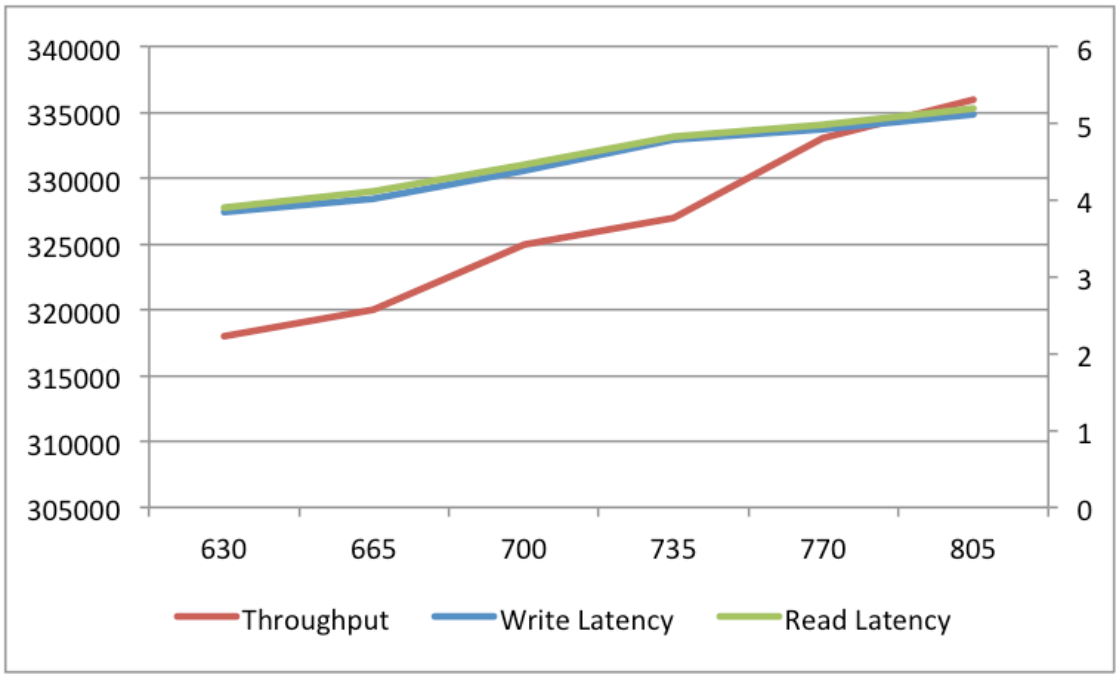


Couchbase Server provided 5x better update latency than MongoDB with the same number of concurrent clients - 245. Update latency quickly increased as we increased the number of concurrent clients. MongoDB latency continued to increase at levels much higher than that of Couchbase, until reaching the latency threshold of 5ms. At this point, with MongoDB you would need to consider adding additional nodes to handle additional concurrent clients.

	MongoDB	Couchbase Server
245 Concurrent Clients	5.38ms	.91ms

Couchbase Server Max Load Testing

These additional tests were performed to identify how many concurrent clients were necessary to saturate Couchbase Server. While MongoDB exceeded the 5ms limit at 245 concurrent clients, Couchbase Server was well below the limit at 525. We wanted to find out just how many concurrent clients Couchbase Server could support.



Couchbase Server did not exceed the maximum latency of 5ms until 805 concurrent clients. These last tests indicate Couchbase Server can reach up to 4.5x the throughput of MongoDB while maintaining latency of 5ms or less. Assuming MongoDB scales linearly, it would have required 4-5x the number of nodes to provide the same performance as Couchbase Server.

Conclusion

The workload we used for this benchmark represents a standard Enterprise scenario of some reads and some updates - common in web and mobile applications. There are scenarios where a use case may have called for heavy reads and light updates, reporting, or heavy updates and light reads, sensor data. We did not cover these scenarios in this benchmark. Overall, we felt that the balanced workload would cover the broadest range of potential use cases for enterprise applications.

Based on the results of this benchmark, Couchbase Server was clearly more capable of handling the workload we threw at it. Couchbase Server displayed the ability to handle requests and maintain a higher throughput with the low latency demanded by today's enterprise web and mobile applications.

The basic clustered architecture of Couchbase Server vs. MongoDB was also a disadvantage for MongoDB in this case. With Couchbase Server, each of the 9 nodes was an active node. MongoDB, on the other hand, was limited to only 3 active nodes due to having only 1 active node per replica set. In addition, extra servers were required for MongoDB to fit into this benchmark. For example, as stated in MongoDB documentation, production instances should have 3 configuration servers. In order to maintain the same setup as the Couchbase Server configuration, we still needed 9 servers for the 3 shards with 2 replicas in addition to the configuration server instances.

The ability to have pluggable storage engines with MongoDB is a potentially useful attribute of the NoSQL database. This capability to have pluggable storage engines will allow it to meet more specific use cases that have specific data needs and requirements. With WiredTiger, however, we did not see the efficiency improvements we were hoping to see. MongoDB did showed signs of stress as we increased the request load. However, MongoDB read latency was comparable to Couchbase Server under the lighter load cases.

Couchbase outperformed MongoDB in the following areas:

- Concurrency
  - Couchbase Server demonstrated better concurrency. It was able to handle over 3x as many concurrent clients as MongoDB.

- Throughput
  - Couchbase Server demonstrated high throughput. Even with the same number of concurrent clients, Couchbase Server was able to provide 2.5x the throughput of MongoDB.

- Latency
  - Couchbase Server demonstrated lower latency. Even with the same number of concurrent clients, Couchbase Server was able to provide 4-5x lower latency than MongoDB.

- Price / Performance Ratio
  - Couchbase Server was able to provide 2.5x, potentially 4.5x, the throughput of MongoDB with the same hardware while meeting the

same latency requirements. The cost per operation for Couchbase Server would be 22-40% of that for MongoDB.

YCSB Setup

Specifications for YCSB workload

- Nodes: 9
- Workload A: 50% reads, 50% updates
- 858GB of Data (Includes Replicas)
- Key Size - 32 bytes
- Value Size - 1K
- Entries - 300,000,000
- Memory Per Node - 30GB
- Primary Data Resident in Memory - 32%
- Request Distribution - Uniform

Results/Data

Couchbase Server Benchmark Results

Clients / Threads	Run #1 Throughput / Latency	Run #2 Throughput / Latency	Run #3 Throughput / Latency
2 / 70	76,000 ops/sec Update: .75ms Read: .76ms	73,000 ops/sec Update: .77ms Read: .77ms	73,000 ops/sec Update: .77ms Read: .79ms
3 / 105	110,000 ops/sec Update: .76ms Read: .78ms	109,000 ops/sec Update: .77ms Read: .78ms	108,000 ops/sec Update: .78ms Read: .78ms
4 / 140	141,000 ops/sec	136,000 ops/sec	132,000 ops/sec

	Update: .78ms Read: .79ms	Update: .81ms Read: .83ms	Update: .87ms Read: .89ms
5 / 175	154,000 ops/sec Update: .88ms Read: .89ms	147,000 ops/sec Update: .89ms Read: .89ms	145,000 ops/sec Update: .9ms Read: .98ms
6 / 210	170,000 ops/sec Update: .92ms Read: .93ms	159,000 ops/sec Update: .95ms Read: .97ms	160,000 ops/sec Update: .99ms Read: .99ms
7 / 245	193,000 ops/sec Update: .91ms Read: .92ms	189,000 ops/sec Update: .96ms Read: .97ms	178,000 ops/sec Update: 1.19ms Read: 1.21ms
8 / 280	238,000 ops/sec Update: .92ms Read: .92ms	230,000 ops/sec Update: .96ms Read: .99ms	201,000 ops/sec Update: 1.21ms Read: 1.3ms
9 / 315	245,000 ops/sec Update: .99ms Read: 1.06ms	235,000 ops/sec Update: 1.04ms Read: 1.10ms	229,000 ops/sec Update: 1.22ms Read: 1.32ms
10 / 350	252,000 ops/sec Update: 1.22ms Read: 1.22ms	244,000 ops/sec Update: 1.3ms Read: 1.31ms	233,000 ops/sec Update: 1.22ms Read: 1.35ms
11 / 385	265,000 ops/sec Update: 1.37ms Read: 1.41ms	251,000 ops/sec Update: 1.59ms Read: 1.65ms	246,000 ops/sec Update: 1.7ms Read: 1.42ms
12 / 420	276,000 ops/sec Update: 1.62ms Read: 1.64ms	268,000 ops/sec Update: 1.75ms Read: 1.8ms	251,000 ops/sec Update: 1.87ms Read: 1.93ms
13 / 455	289,000 ops/sec Update: 1.7ms Read: 1.71ms	277,000 ops/sec Update: 1.79ms Read: 1.87ms	264,000 ops/sec Update: 1.88ms Read: 1.88ms
14 / 490	304,000 ops/sec Update: 1.93ms Read: 2.01ms	289,000 ops/sec Update: 2ms Read: 2.04ms	270,000 ops/sec Update: 1.94ms Read: 1.97ms
15 / 525	310,000 ops/sec Update: 1.99ms Read: 2.1ms	297,000 ops/sec Update: 2.11ms Read: 2.17ms	289,000 ops/sec Update: 2.10ms Read: 2.11ms

Couchbase Server Max Load Benchmark Results

18 / 630	318,000 ops/sec Update: 3.84ms Read: 3.9ms
19 / 665	320,000 ops/sec Update: 4.01ms Read: 4.12ms
20 / 700	325,000 ops/sec Update: 4.39ms Read: 4.47ms
21 / 735	327,000 ops/sec Update: 4.79ms Read: 4.82ms
22 / 770	333,000 ops/sec

	Update: 4.93ms Read: 4.99ms
23 / 805	336,000 ops/sec Update: 5.12ms Read: 5.2ms

MongoDB Benchmark Results

Clients / Threads	Run #1 Throughput / Latency	Run #2 Throughput / Latency	Run #3 Throughput / Latency
2 / 70	37,000 ops/sec Update: 1.9ms Read: .98ms	38,000 ops/sec Update: 1.98ms Read: 1.12ms	35,000 ops/sec Update: 1.86ms Read: 1.10ms
3 / 105	61,000 ops/sec Update: 2.05ms Read: 1.42ms	60,000 ops/sec Update: 2.14ms Read: 1.62ms	58,000 ops/sec Update: 1.99ms Read: 1.72ms
4 / 140	65,000 ops/sec Update: 2.97ms Read: 2.01ms	68,000 ops/sec Update: 3.01ms Read: 2.64ms	68,000 ops/sec Update: 3.12ms Read: 2.71ms
5 / 175	67,000 ops/sec Update: 3.54ms Read: 3.16ms	67,000 ops/sec Update: 3.47ms Read: 3.04ms	66,000 ops/sec Update: 3.63ms Read: 3.41ms
6 / 210	70,000 ops/sec Update: 4.49ms Read: 3.5ms	69,000 ops/sec Update: 4.41ms Read: 3.39ms	67,000 ops/sec Update: 4.7ms Read: 4.01ms
7 / 245	74,000 ops/sec Update: 5.38ms Read: 4.19ms	73,000 ops/sec Update: 5.21ms Read: 4.12ms	71,000 ops/sec Update: 5.39ms Read: 4.49ms