Benchmarking

HarperDB v. MongoDB

May 2020

HarperDB Inc. | 1001 17th St. Suite 680. Denver, CO 80202 | hello@harperdb.io

Overview

First, let us state that we are grateful for MongoDB and know that it is a product loved and used by many. MongoDB paved the way for companies like HarperDB to exist and pioneered a new way for developers and users to interact with their data. Without awesome products like MongoDB there would be no HarperDB.

We built HarperDB from the ground up to expand and blend the capabilities of SQL, NewSQL, and NoSQL products like MongoDB because we felt there were certain use cases that could be better served with another solution. We believe that providing developers with the ability to choose the right tool for the job empowers developers and spurs innovation. Some examples where we feel that HarperDB is a better fit include cases where you need NoSQL & SQL, rapid application development, integration, edge computing, distributed computing, and real-time operational analytics.

As a result, with the release of HarperDB 2.0, we wanted to see how we stacked up against the most popular NoSQL database in the world, MongoDB. Turns out, we're faster, not just sort of faster, much faster.

We asked an external technology team, Mycos Technologies, to create a simple benchmark for HarperDB compared against a modern, accessible, developer-centric database, MongoDB. We believe HarperDB is easier to use and provides more flexibility than MongoDB. In this benchmark, we intend to prove that HarperDB holds up performance-wise as well.

Highlights

HarperDB **is 37.9 times faster** than MongoDB on average.

HarperDB **is up to 98 times faster** than MongoDB on reads.

HarperDB **is up to 20 times faster** than MongoDB on writes.

HarperDB **scales out-of-the-box** for concurrent operations.

HarperDB Inc. | 1001 17th St. Suite 680. Denver, CO 80202 | hello@harperdb.io

Methodology

This benchmark evaluates data writes, data reads, and a combination of reads and writes on HarperDB and MongoDB. Tests were executed on both databases using their default configurations. Benchmarks were executed based on recommended benchmarking best practices for both databases. The benchmark tests were executed using Apache JMeter for 10 minutes, with a 10-second ramp-up, with a simulated user count of 1, 10, and 50.

The following tests were executed on both databases:

/ Write data with 1 user
/ Read data with 1 user
/ Read/write data with 1 user
/ Write data with 10 users
/ Read data with 10 users
/ Read/write data with 10 users
/ Write data with 50 users
/ Read data with 50 users
/ Read/write data with 50 users

Each test was run two times and using the following order of operations:

1. Create schema/table for HarperDB or database/collection for MongoDB
2. Loop test action for 10 minutes. Test actions:
   a. Write a single, randomly generated record to the database
   b. Read a single, randomly selected record from the database
   c. Write a single, randomly generated record to the database, then read a single randomly generated record form the database
3. Clean environment

HarperDB Inc. | 1001 17th St. Suite 680. Denver, CO 80202 | hello@harperdb.io

Determining Methodolgy

Finding compatible connection patterns for MongoDB is a little tricky, HarperDB uses a REST API, and many other databases rely on product specific drivers and/or an Object Relation Modeling (ORM) library. The initial implementation of this benchmark used Rest Heart, an API front end for MongoDB. The results were not in the competitors favor, therefor it was decided to chose an alternative method provided by BlazeMeter, a leading test automation framework.

HarperDB Evalution Method

A REST API is the primary method to interact with HarperDB. User sessions are not use as each call requires an authorization token. HarperDB was benchmarked via JMeter's default REST API execution option.

MongoDB Evalution Method

JSR 223 scripting, the accepted scripting language for Java, was used to execute MongoDB connectivity. JMeter was updated with a more current Java driver for MongoDB, version 3.12.3.

Data Format

In all tests, object data was generated with the following 7 attributes:

```
{
  "id": "${__UUID}",
  "timestamp": "${__time()}",
  "photosensor": "${__Random(1,999)}.${__Random(1,100)}",
  "sensor_uuid": "probe-${__UUID}",
  "radiation_level": "${__Random(1,999)}",
  "ambient_temperature": "${__Random(1,100)}.${__Random(1,99)}",
  "humidity": "${__Random(1,999)}.${__Random(1,999)}"
}
```

Read tests requested a single record from a query request filtering on a random id value

```
where id = _Random(0,<number of written records>)
```

Test Results

This benchmark shows that HarperDB is faster than MongoDB in every benchmark test. While HarperDB shows significant advantages in all tests, two cases show a clear distinction: data reads and concurrent interactions.

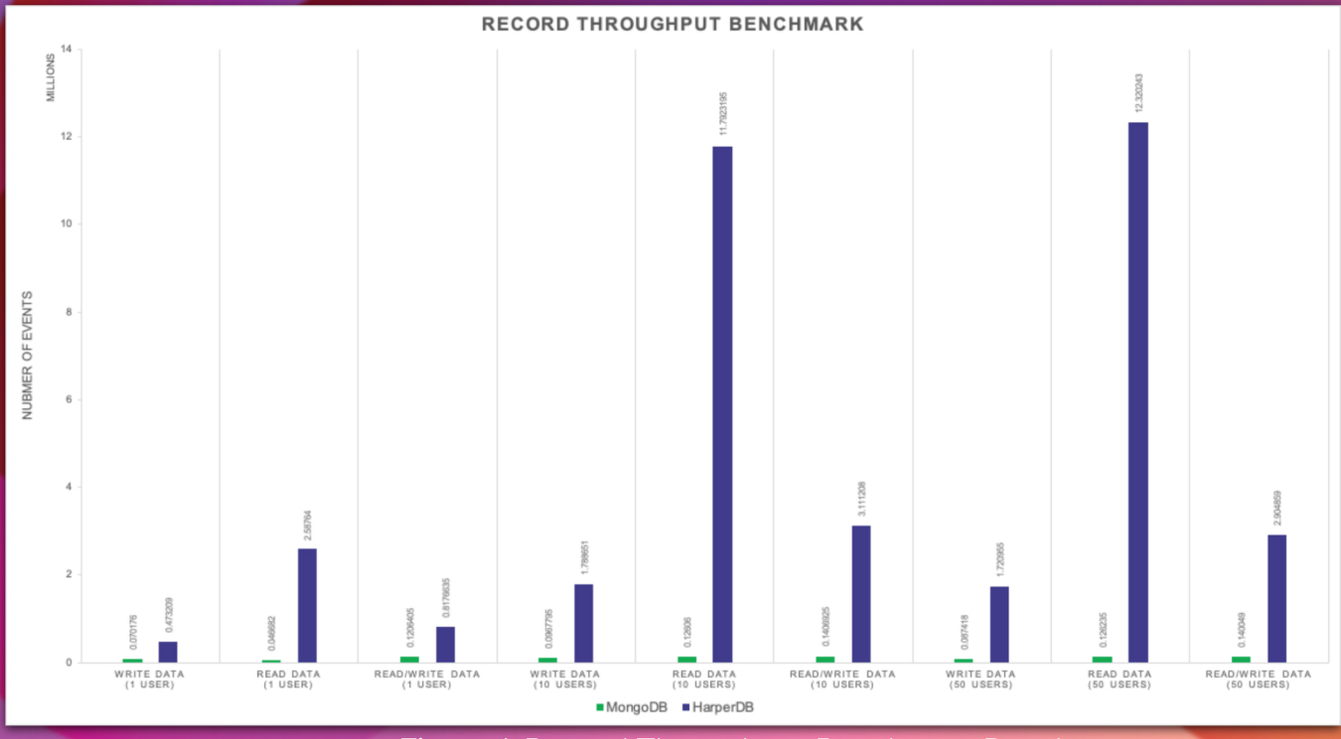

Figure 1: Record Throughput Results per Database

HarperDB Inc. | 1001 17th St. Suite 680. Denver, CO 80202 | hello@harperdb.io

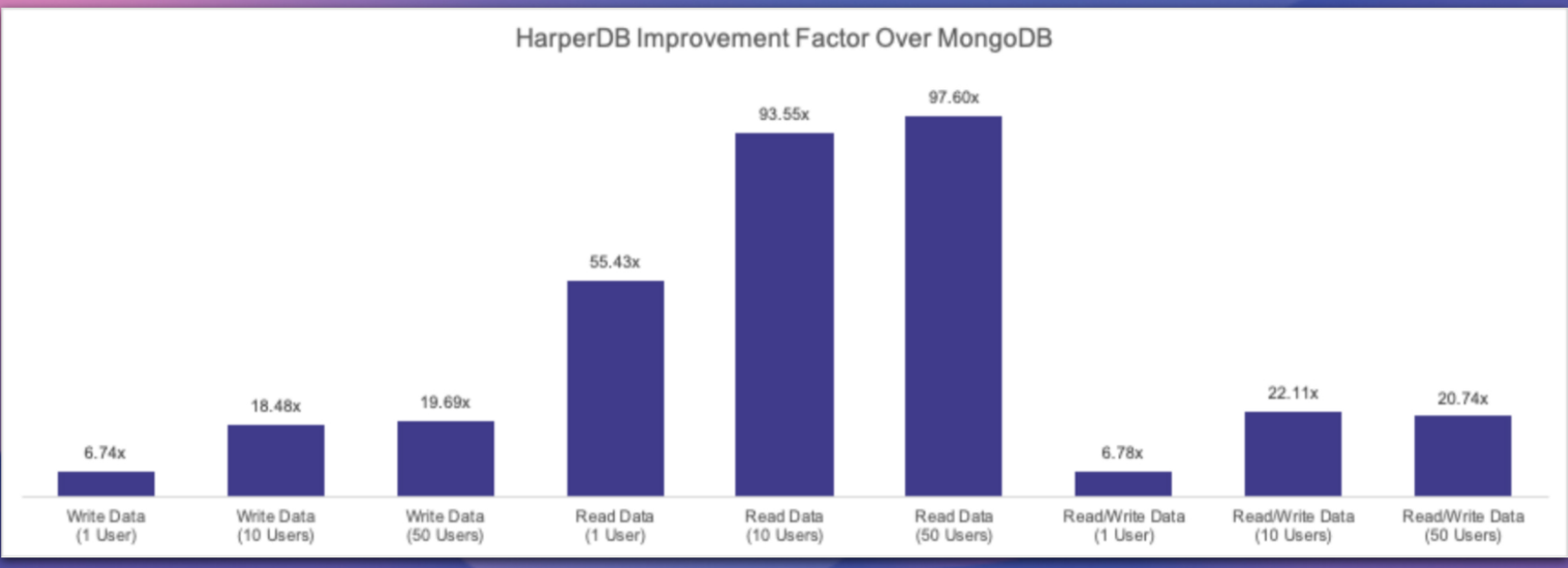

Figure 2: HarperDB Performance Multiplication Factor Compared to MongoDB

Conclusion

Our goal in developing HarperDB was to build a product that would empower developers through ease of use while remaining highly scalable, which we have demonstrably shown in the results of this benchmark. As a result, it was our decision that both databases were benchmarked using default configurations and best practices for both products. HarperDB is built to be simple to install and use, without requiring costly product experts for tuning. We feel that this test conclusively demonstrates that at scale with numerous clients, HarperDB is more performant and can be scaled out more simply and cost-effectively.

Questions, comments, or feedback? Email us at benchmarks@harperdb.io

Ready to ditch MongoDB? Use code BYEMONGO when signing up for HarperDB Cloud and receive $300 in credits towards our paid tiers

Evaluation Machine Specs

CPU	CPU Intel(R) Core(TM) i7-8700 6 Core 3.20GHz
Memory	32 GiB
Disks	2x 250GB SSD