Yahoo Cloud Serving Benchmark (YCSB) Redis, MongoDB and MySQL

Introduction

Twitter data is represented in JSON format as available from the data source, this is semi-structured data including text content, user information, and other meta-information like hashtags, geolocations, and timestamps (not all this metadata is available). Given the shape of the data in scope for this research, the following storage database (which includes support for this data type) will be evaluated using Yahoo Cloud Serving Benchmark.

**MongoDB:** MongoDB is a natural fit for handling JSON data, as it's a document-oriented database designed to store, query, and process large amounts of data in JSON-like formats, BSON, in the case of MongoDB. The fact that this DB is schema-less model makes it easy to manage and process the variable structure of tweets. MongoDB is a distributed system that allows for scaling horizontally (add more servers) and vertically (CPU and memory).

**Redis**: Redis, traditionally it's a key-value store for in-memory data well fitting for cache; it has evolved to support data persistent on distributed systems and with support for semi-structured data types. It offers data structures such as lists, sets, and hashes that could be used to manage tweet data effectively. Note: for Redis to withstand millions of tweets, persistence (EDUCB, 2023) must be configured on the distributed store with Redis Cluster.

**MySQL:** MySQL is a relational database which traditionally works with structured data and can require complex schemas for semi-structured data like JSON. Since version 5.7.8 (Oracle Ltd, 2015) MySQL includes JSON data type support, allowing for efficient storage and querying of JSON data.

For this research's scope, Benchmarking process is required to provide insights into the performance of various databases; this will help to compare the strengths, weaknesses, and bottlenecks of different database management systems (DBMS).

# Benchmarking

Bencharmark plan

The objective is to benchmark Redis, MongoDB, and MySQL using the following workloads.

|  |  |
| --- | --- |
| **Workload** | **Description** |
| A | Update heavy workload: 50% read, 50% update operations. |
| B | Read mostly workload: 95% read, 5% update operations. |
| C | Read only: 100% read operations. |
| D | Read latest workload: New records are read more frequently. |
| E | Short ranges: Range queries over the latest records. |
| F | Read-modify-write: Client read a record, modifies it, and writes back the changes. |

*\*Workload E uses the Zipfian distribution to choose the first key in the range, and the Uniform distribution to choose the number of records to scan* (Cooper *et al.*, 2010)

The testing strategy will be applied to the different workload loads on these databases with operation counts of 10,000, 100,000, 1,000,000 per iteration.

Reproducibility and test conditions:

The test environment will be docker container to ensure isolation and reproducibility of the test environment. The host computer is a Dell XPS 13, Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz (4 cores, 8 logical processors), 16 Gb RAM @2133MHT and 500 Gb SSD.

Docker images:

|  |  |  |  |
| --- | --- | --- | --- |
| **Container Name** | **Docker Image** | **Base Image** | **Repo / Documentation** |
| YCBS | ycbs-loadgen:latest | maven:3.6.1-jdk-8-slim | [BlueMedoraPublic/docker-ycsb: YCSB in docker (github.com)](https://github.com/BlueMedoraPublic/docker-ycsb) |
| REDIS | redis/redis-stack-server:6.2.6-v7 |  | [Run Redis Stack on Docker | Redis](https://redis.io/docs/stack/get-started/install/docker/) |
| MONGO | mongodb/mongodb-community-server:6.0.4-ubuntu2204 |  | [Docker & MongoDB | Containers & Compatibility | MongoDB](https://www.mongodb.com/compatibility/docker) |
| MySQL |  |  | [mysql - Official Image | Docker Hub](https://hub.docker.com/_/mysql) |
| Cassandra |  |  |  |

For each database, we'll start with setting up the database and ensuring it's running optimally. We'll then load the data into the database using YCSB's load functionality. Each test will then be executed, and the results will be recorded.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **WORKLOAD** | **A** | | | **B** | | | **C** | | | **D** | | |
|  | **10k** | **100k** | **1M** | **10k** | **100k** | **1M** | **10k** | **100k** | **1M** | **10k** | **100k** | **1M** |
| **Redis** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Mongo** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Cassandra** |  |  |  |  |  |  |  |  |  |  |  |  |
| **WORKLOAD** | **E** | | | **F** | | |  | | | | | |
|  | **10k** | **100k** | **1M** | **10k** | **100k** | **1M** |
| **Redis** |  |  |  |  |  |  |
| **Mongo** |  |  |  |  |  |  |
| **MySQL** |  |  |  |  |  |  |

References

Cooper, B.F. *et al.* (2010) ‘Benchmarking cloud serving systems with YCSB’, in *Proceedings of the 1st ACM symposium on Cloud computing*, pp. 143–154. Available at: https://courses.cs.duke.edu/fall13/cps296.4/838-CloudPapers/ycsb.pdf.

EDUCB (2023) ‘EDUCB - Redis Persistence’, *EDUCB - Redis Persistence*. Available at: https://www.educba.com/redis-persistence/ (Accessed: 14 May 2023).

Oracle Ltd (2015) ‘MySql: The JSON Data Type’. Available at: https://dev.mysql.com/doc/refman/5.7/en/json.html (Accessed: 14 May 2023).