MongoDB vs DocumentDB

Cosmin Pintoiu
Solution Architect at Bigstep
Cosmin Pintoiu
Solution Architect at Bigstep

Designed and implemented critical message processing projects in financial sector and real-time analytics in retail sector.

Currently focused on large-scale real-time implementations, Data lakes and machine learning using Tensorflow.
Agenda:

- Intro to Mongo and DocumentDB
- Setup Methodology
  - Node types
  - Network and AZ
- Benchmark using
  - Jmeter and custom sampler
  - YCSB
  - Mongo Socialite
- Price consideration
- Conclusions
- Q&A

Duration: 25m – 30m
In this study, we will take a look at performance and cost aspects of running a MongoDB database environment on Bigstep Metal Cloud versus DocumentDB from AWS. To make it a fair comparison we will use similar resources and identical load tests.

- **MongoDB** is a cross platform document oriented database, was released 10 years ago and offers a multitude of features: indexing, replication, load balancing, aggregation, transactions.

- **Amazon DocumentDB** (with MongoDB compatibility) is a fast, scalable, highly available, and fully managed document database service that supports MongoDB workloads.

Our scope is to make this study impartial and easily reproducible, in this regard all the steps involved in setting up the environment and the test are available on github (https://github.com/ccpintoiu?tab=repositories).
Benchmarks

Considerations when performing benchmarks*:

- **Relevant** (for users of the benchmark: engineering, marketing, buyers etc)
- **Repeatable** (results)
- **Fairness** (to both hw and sw involved)
- **Verifiability** (in case of audit)
- **Economical** (to set up, run and publish)

* key aspects according to: Performance Evaluation and Benchmarking for the Era of Artificial Intelligence TPCTC 2018

Authors: Raghunath Nambiar and Meikel Poess

## Node types

<table>
<thead>
<tr>
<th>Model</th>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>Network Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigstep MongoDB</td>
<td>FMCI 8.32</td>
<td>8*</td>
<td>32 GB ECC</td>
<td>BSA</td>
</tr>
<tr>
<td>AWS DocumentDB</td>
<td>db.r4.2xlarge</td>
<td>8 (vCPU)</td>
<td>61 GB</td>
<td>EBS-only**</td>
</tr>
<tr>
<td>AWS MongoDB EC2</td>
<td>m5.2x.large</td>
<td>8 (vCPU)</td>
<td>32 GB</td>
<td>EBS-only**</td>
</tr>
</tbody>
</table>

### Diagram

- **Jmeter GUI**
  - Local Station
  - Control
  - Jmeter Server
    - Load node for YCSB and Socialite
    - read
    - write
    - replication

- **Mongo Server**
  - IP: 10.0.0.x
    - Primary
    - Secondary

- **Cluster under test**
  - read
  - replication
Setup Mongo on Bigstep

- 1 Load node + 3 Mongo nodes
- Version 4.0.1
Setup DocumentDB

- 1 Load node (EC2) + 3 Mongo nodes (db.r4.2xlarge)
- API version 3.6

- setup EC2 instance
- ssh
- add repo for mongo shell
- sudo yum install -y mongodb-org-shell
- mongo --ssl --host docdb

NOTE:
- both docdb and ec2 instance should be in the same security group
- Connect from outside to Document DB using port forwarding: nc -zv host port
**Benchmark using**

**Jmeter** is a load testing tool used mostly on web apps but it can be used very well on databases. It is Java-based and supports variable parametrization. This version uses ReactiveStreams 1.10 and the 3.9 Java MongoDB Driver and is tested with Jmeter version 5.0. It supports the following operations: read/write and readMany/writeMany.

**YCSB** is a popular tool when comparing relative performance on NoSQL databases. Developed at Yahoo! for the specific purpose of comparative studies of various databases systems, YCSB is highly customizable. Workload files with 50/50 reads/writes, 75/25 and 95/5 to have a valid comparison.

**Socialite** is a test developed by Mongo team part of their regression testing for mongo product. This test simulates a social media platform with a number of users, followers and articles per user. The run command reads the first 100 iterations and writes the results into a file. The output file is quite rich, most important field is the mean_rate, which shows the average ops/sec.
Jmeter test

https://github.com/bigstepinc/jmeter-mongo-db-custom-sampler
https://github.com/bigstepinc/jmeter-mongo-db-custom-sampler/releases/latest
Jmeter test

Jmeter custom sampler read/write 50 threads
Avg Ops/sec

Bigstep MongoDB
19737.72222

AWS DocDB
12662.01951
14825.54737

Jmeter config file used:

50 threads (simulates users)
loop count: 40000 (how many times a thread group gets executed)

Run command and time:
./jmeter.sh -n -t /tmp/Jmeter-Bigstep_1.3_WRSingle4M.jmx -l /tmp/output_jmxWRSingle4M.csv
Jmeter test

3rd test including Mongo Db on AWS EC instance (all instances in one availability zone)

![Graph showing Jmeter custom sampler read/write 50 threads]

**Avg Ops/sec**

- Bigstep MongoDB: 50124.86667
- AWS DocDB: 19737.72222
- AWS Mongo on EC2: 66813.92222
Jmeter distributed test

Next steps: Stress test using Jmeter Distributed testing
We can use our Custom Mongo Sampler (one other option is: https://github.com/johnlpage/POCDriver)
The goal of YCSB project is to develop a framework and common set of workloads for evaluating the performance of different "key-value" and "cloud" serving stores.
YCSB test

Load command and time:

```
./bin/ycsb load mongodb -s -P workloads/workload_small -threads 32 -p
mongodurl=mongodb://10.0.0.31:27017/?replicaSet=mongo_rs&w=majority
```

<table>
<thead>
<tr>
<th>Load time:</th>
<th>Bigstep MongoDB</th>
<th>AWS DocDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>load 4M</td>
<td>9m44.324s</td>
<td>45m44.28s</td>
</tr>
</tbody>
</table>

Example workload file used:
requestdistribution=zipfian
recordcount=4096000
operationcount=20000000
readallfields=true
readproportion=0.5
updateproportion=0.5
YCSB test

- db.usertable.count()
- 20GB
- W=majority
- 32 threads

writeConcern:
- Allowed values are:
  - errors_ignored
  - unacknowledged
  - acknowledged
  - journaled
  - replica_acknowledged
  - majority

readPreference:
- Allowed values are:
  - primary
  - primary_preferred
  - secondary
  - secondary_preferred
  - nearest
YCSB test

The first step is to load 4 million records using 32 threads and count the time that each environment needs to complete the task.
YCSB test

YCSB Ops/sec

- Bigstep MongoDB
- AWS DocDB
- AWS EC2 mongoDB

Run 95 read / 05 write:
- Bigstep MongoDB: 35303.76111
- AWS DocDB: 31495.03278
- AWS EC2 mongoDB: 17015.22105

Run 75 read / 25 write:
- Bigstep MongoDB: 14096.64095
- AWS DocDB: 11929.41
- AWS EC2 mongoDB: 8944.402222

Run 50 read / 50 write:
- Bigstep MongoDB: 11625.95944
- AWS DocDB: 5521.108333
- AWS EC2 mongoDB: 5864.294118
Socialite

As with the YCSB tool, Socialite is quite complex and offers various load tests: benchmark, timeline-read-follower-ramp, send-ramp-followers.
The Socialite implementation uses 3 MongoDB collections by default called users, followers, following.

Load command and time:
java -jar ./target/socialite-0.0.1-SNAPSHOT.jar load --users 100000 --maxfollows 5000 --messages 20 --threads 32 sample-config.yml

Run command:
java -jar ./target/socialite-0.0.1-SNAPSHOT.jar timeline-read-follower-ramp --out output1 --start 1 --stop 100 sample-config.yml

Example config file used:

totalUsers=10000
activeUsers=1000
duration=3600
sessionDuration=30
concurrency=512
maxFollows=5000
messages=20

![Diagram of MongoDB collections](image)
Socialite

timeline-read-follower-ramp

Ops/sec

Bigstep MongoDB

AWS DocDB
Ops/sec for: timeline-read-follower-ramp

<table>
<thead>
<tr>
<th>Service</th>
<th>Ops/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigstep MongoDB</td>
<td>394.0030235</td>
</tr>
<tr>
<td>AWS DocDB</td>
<td>268.0969634</td>
</tr>
<tr>
<td>AWS EC2 MongoDB</td>
<td>717.1475015</td>
</tr>
</tbody>
</table>
Cluster deployment and scalability

For the moment **Bigstep** does not provide managed services on MongoDB. You can install the software off the shelf on the bare metal instances. The main advantage is that you can configure the cluster accordingly to your needs. Being a self-manage platform you can also setup up a Sharded Cluster which will offer you better performance on large amount of data.

**AWS DocumentDB** is ready for production and you can start loading data as soon as the instances are up. You don’t have access on the management side of the services so the disadvantage is that you cannot customize. You use as it is.

<table>
<thead>
<tr>
<th></th>
<th>Create Cluster</th>
<th>Install Mongo</th>
<th>Scale (1 node)</th>
<th>Attach Workers</th>
<th>Total time:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigstep Platform</td>
<td>12 min</td>
<td>10 min</td>
<td>-</td>
<td>-</td>
<td>22 min</td>
</tr>
<tr>
<td>AWS DocDB</td>
<td>12 min</td>
<td>-</td>
<td>4 min</td>
<td>2 min</td>
<td>18 min</td>
</tr>
</tbody>
</table>
Cluster deployment and scalability

Steps to take for scalability Bigstep or AWS EC cluster:
- deploy new instance
- configure private IP
- install mongodb
- add the new node in cluster

The work time is higher than DocumentDB as at the moment Bigstep does not provide MongoDB as an integrated service, yet some steps can be automated. DocumentDB is way easier to scale up to 15 replica nodes and grows the size of your storage volume automatically

<table>
<thead>
<tr>
<th>Scale Cluster</th>
<th>Add Instance</th>
<th>Conf IP</th>
<th>Install Mongo</th>
<th>Add node in cluster</th>
<th>Total time:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigstep Platform</td>
<td>5 min</td>
<td>2 min</td>
<td>5 min</td>
<td>5 min</td>
<td>17 min</td>
</tr>
<tr>
<td>AWS DocDB</td>
<td>5 min</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5 min</td>
</tr>
</tbody>
</table>
### Costs for entire cluster:

<table>
<thead>
<tr>
<th>FULL METAL CLOUD INSTANCES</th>
<th>Description</th>
<th>Quantity</th>
<th>Reserved price (EUR/mth)</th>
<th>On demand price (EUR/h)</th>
<th>Reserved price (USD/mth)</th>
<th>On demand price (USD/h)</th>
<th>Total Monthly Reserved price (EUR)</th>
<th>Total Hourly On demand price (EUR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMCI 8.32v2 - General Purpose</td>
<td>CPU: 1 x Intel Xeon E3-1585L v5 - 4 cores 8 threads @ 3.3Ghz</td>
<td>1</td>
<td>189</td>
<td>0.30</td>
<td>€ 189.00</td>
<td>€ 225.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAM: 32 GB DDR4 ECC Network: 2 x 10 GbE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>SSD Storage (100 GB)</td>
<td>100</td>
<td>0.15</td>
<td>0.00</td>
<td>€ 14.62</td>
<td>€ 19.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bigstep MongoDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>€ 203.62</td>
<td>€ 244.50</td>
</tr>
<tr>
<td>AWS r4.2xlarge</td>
<td>8 (vCPU) 61 GB EBS-only*</td>
<td>1</td>
<td>256.51</td>
<td>0.56</td>
<td>288.2</td>
<td>0.62</td>
<td>€ 256.51</td>
<td>€ 416.52</td>
</tr>
<tr>
<td>Storage</td>
<td>First 50 TB / Month - price per GB</td>
<td>100</td>
<td>0.02</td>
<td>$0.02</td>
<td>€ 10.32</td>
<td>€ 10.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>€ 266.83</td>
<td>€ 426.84</td>
</tr>
<tr>
<td>AWS m5.2x.large</td>
<td>8 (vCPU) 32 GB EBS-only*</td>
<td>1</td>
<td>202.06</td>
<td>0.40</td>
<td>227</td>
<td>0.44</td>
<td>€ 202.06</td>
<td>€ 296.37</td>
</tr>
<tr>
<td>Storage</td>
<td>First 50 TB / Month - price per GB</td>
<td>100</td>
<td>0.02</td>
<td>$0.02</td>
<td>€ 10.32</td>
<td>€ 10.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>€ 212.38</td>
<td>€ 306.69</td>
</tr>
</tbody>
</table>
Costs for entire cluster:

Price comparison

<table>
<thead>
<tr>
<th>Service</th>
<th>Euro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigstep MongoDB</td>
<td>814.48</td>
</tr>
<tr>
<td>AWS DocDB</td>
<td>1587.21</td>
</tr>
<tr>
<td>AWS EC2 MongoDB</td>
<td>1226.76</td>
</tr>
</tbody>
</table>

Bigstep MongoDB, AWS DocDB, AWS EC2 MongoDB

Cluster reserved vs cluster on demand
General Takeaways

Pick the right type of node

Perform custom tests for your problem

Take into account scalability and flexibility
I’m all ears!

@bigstepinc

@ cosmin.pintoiu@bigstep.com