Aalto University
School of Science

First Assignment

Author:

Omer Ahmed Khan

802062

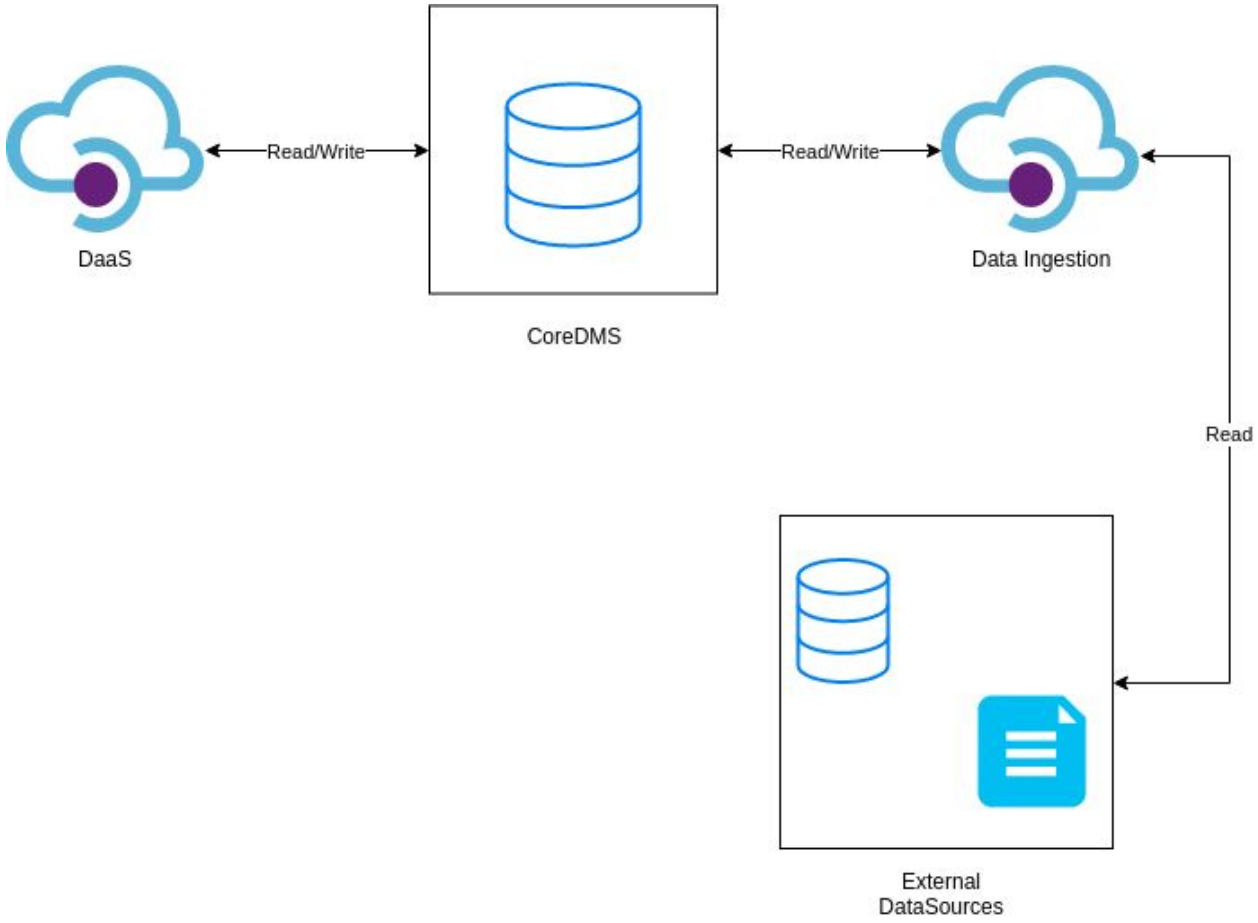
omer.khan@aalto.fi

Big Data Platform
Autumn 2019

07 October 2019

PART 1:

1) There are three main components of this architecture which are coredms, daas and data ingestion. While coredms is a central component which will be consist of a database. Daas will be serve as a Function as a service module which can write and read directly from coredms in a scaleable fashion. While dataIngestion will work through data stream platform like Kafka to read/write data to coredms and external data sources respectively.



Some points for above design:

- CoreDms is storage and should be elastic fault-intolerance and scalable
- Daas is a service from which we can read/write data directly into CoreDms
- DataIngest is a to copy external data source from/to Coredms

2) For the whole system to work, there should be single instances of service be up every time which could be done by applying load balancer before Daas and DataIngest service. For CoreDms we need 2 nodes which work as Master-slave nodes. In Mongo Db, there is a concept of replication which provides us fault intolerance and scalability which is critical for this component.

3) I will be using VM for the following reasons:

- VM is more secure than container
- It can be a Horizontal Scale which is always better than vertical scaling in case of containers
- Cloud Services provides click and deploy feature which could help me to spend less time in configuration

4) I will be using Apache Kafka which is basically a pub-sub queue, which helps to remove the bottleneck from both CoreDms and DataIngestion. Moreover, with larger request, CoreDms will scale automatically with new instances, and with Kafka queue, we will not have a load in between connection of these two components.

Note:
For development in question 2, I just use a simple script to do dataIngestion

5) For current project following are the motivations to choose Industrial cloud Infrastructure:

- Not enough expertise to manage the network operations of the systems
- It is cost-effective rather than building it on-premise
- Easy to deploy and manage
- The dataset is open which doesn't require any security which in-premise provides.
- Don't have to worry about scalability and elasticity of the services

Part 2:

1) As we are using NoSql as a data source for our forms, there will be no proper schema. However, the columns of the dataset will be the fields of documents in a single collection.

Deploying CoreDMS

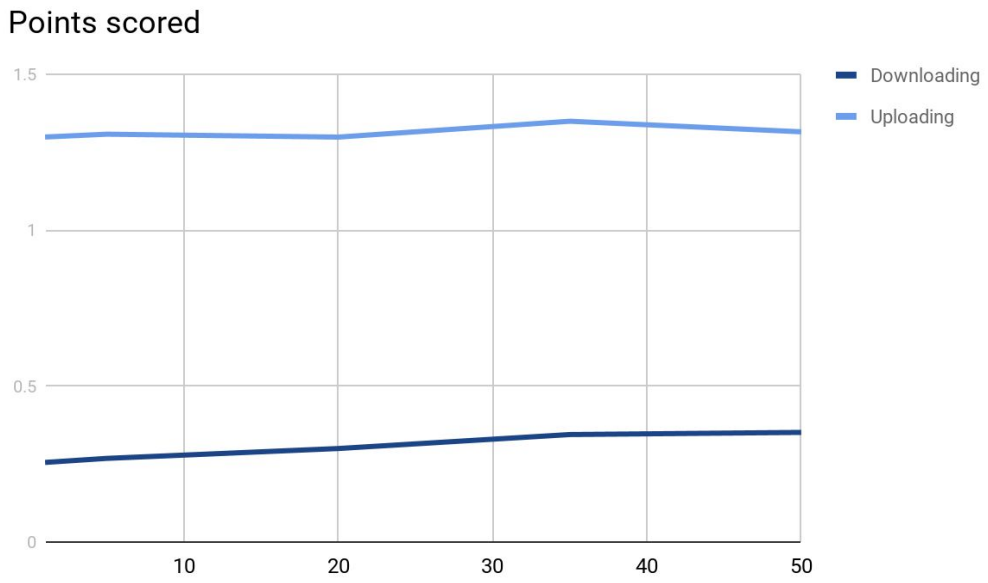
1) Create a project on GCP
2) Spin Mongo cluster
3) Fix port settings to connect Mongo from outside either by all or add sources and apply setting to all networks.

2) The dataset I used is “Google Play Store Apps”, in which there are two fields which could be used to shard data. One is “Type” which simply divides the data into two parts “Free” and “paid”, while other is “Content Rating” which provide information of the target audience group. This way we can boost our read while having a sharded database for targeted needs.

3) DataIngest is a developed with Python Script with the use of MongoDB API to ingest data from an external source(CSV). To execute this script do the following:

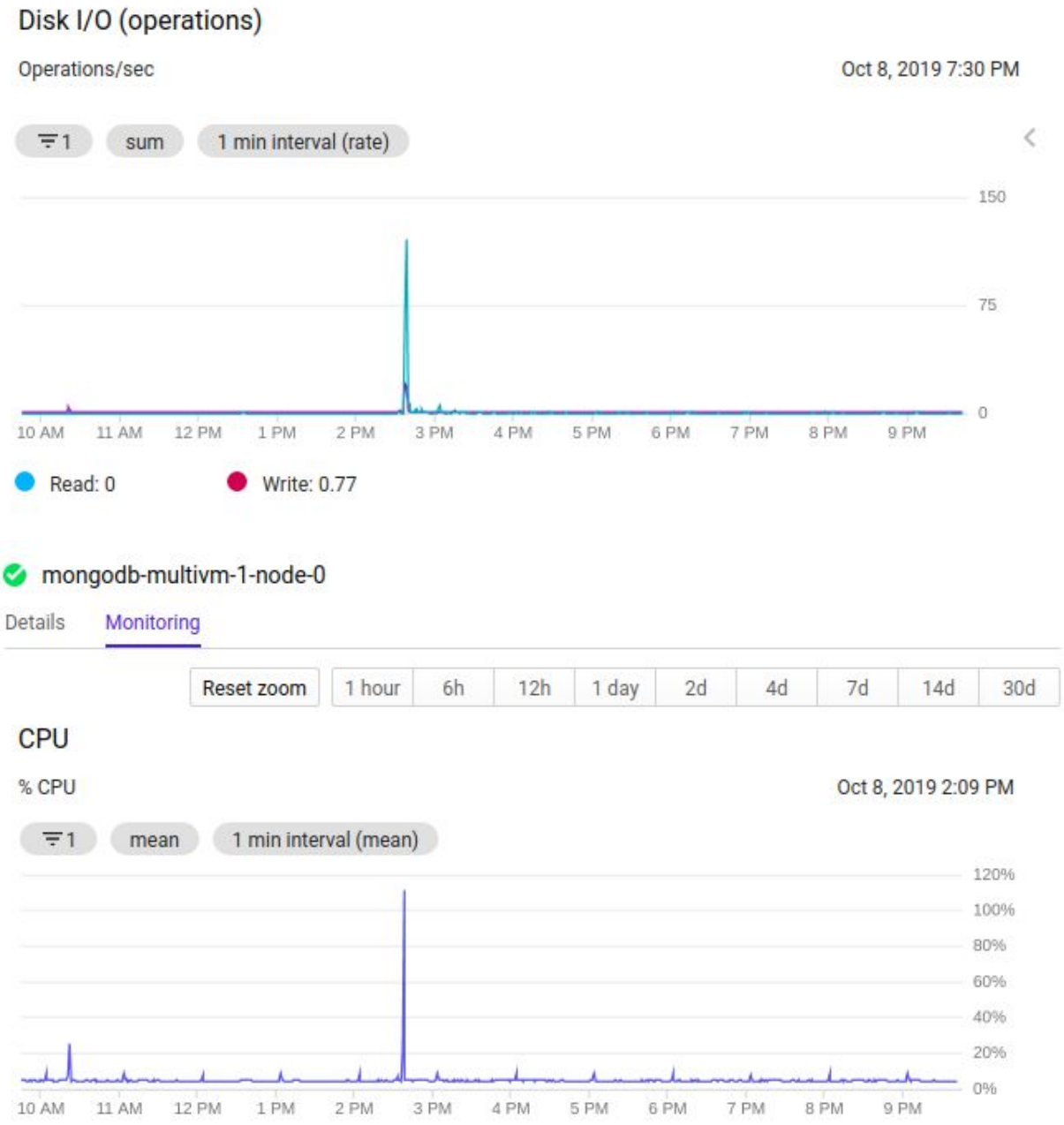
- Run requirement.txt to add dependencies need to run dataIngest.
- Run with command “python mydataIngest -p <path of csv path> -o <read/write>”

4) Following are the chart results for uploading and downloading data with “n” number of process.



The graph shows flat results because of the parallel process and usage of batch processing within a DataIngest. It doesn't matter who many processes you spawn the results will be the same unless the CoreDms remain stable that is doesn’t run out of resources and scale accordingly.

The following graphs will show the load on CPU and Disk of CoreDms:



With more than 100 processes for writing grom DataIngest to forms, some of the process failed due to the following warning:

```
The master 0 ran at 100% CPU utilization.
```

Which results in 500 error from CoreDms for several DataIngest processes.

5) There are many reasons for downgrading performance due to the current environment, but mostly because of the resources in hand.

- As I am using my own personal machine to run data ingest thus the number of processes is limited due to resource constraint.
- Secondly, the mongo instance is running on the least instance configuration by GCP which has very limited resources thus the load is much more for the resources in hand.

The main problem is with resources, which can be done by scaling CoreDms both horizontally and vertically to handle the load. For the CPU utilization issue, we need to have a load balancer before CoreDMS which can easily scale our CoreDMS on metrics like CPU and memory utilization. Moreover, the DataIngest script should also be deployed to the cloud to control its scalability and boost its performance

Bonus Question:

I develop myDaas API with use of the Flask framework of Python, which can easily be delpy by just copying the project folder to /home/$USER/ and running the script called “playit.sh”. It will install every dependency needed and deploy the project locally. MyDaas will have 2 API, one is to read and other is to write. Documentation for API are as follows:

Prerequisites
1) Lunix OS
2) Python3
3) Curl (if executing web service endpoint from terminal)
4) Terminal or web browser

Please follow below instruction to create a local server for web services:

1) Open Lunix terminal
2) Copy bigdata directory to /home/<user> (current user of the lunix) path.
3) Go to bigdata directory
4) Execute bash script "playIt.sh"
5) And Boom your web server is deployed on localhost
6) To check the status of web services(API) execute "curl http://127.0.0.1:5000/status"

Following are the endpoints and their arguments:

1)/status/
To check the status of a web server

2)/read/<recordCount>
Get records w.r.t count, to get all provide “0”

3)/write/
Create a record, allowed fields are as follows:
['app', 'category', 'rating', 'reviews', 'size', 'installs', 'type', 'price', 'content_rating', 'genres', 'last_updated', ‘'current_ver', 'android _ver']