Big Data Management

Project Report : Landing Zone - P1

Tejaswini Dhupad
Chun Han Li

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

March, 2022

TABLE OF CONTENTS

1. Scope of the Project

As a part of our joint project of - Viability of Business Project, Big Data Management and Semantic Data Management we are trying to provide a holistic solution for on-demand computational powers and have named our business idea as - IdleCompute. The project is mainly divided into 2 subparts :

1. *Tenant*[1] & *Renter*[2] data processing
2. *Task processing*[3]

The project scope of our team is mainly implementing the tenant and renter data processing. Here, we are gathering the data from both the tenant and the renter and landing it into our identified document storage.

2. Data Source

The data is received from the *Registration form*[4] filled by the Tenant and the Renter on the IdleCompute webpage. When the tenant/renter submits the form, the web page is going to launch an API which contains the information as below :

Tenant JSON:

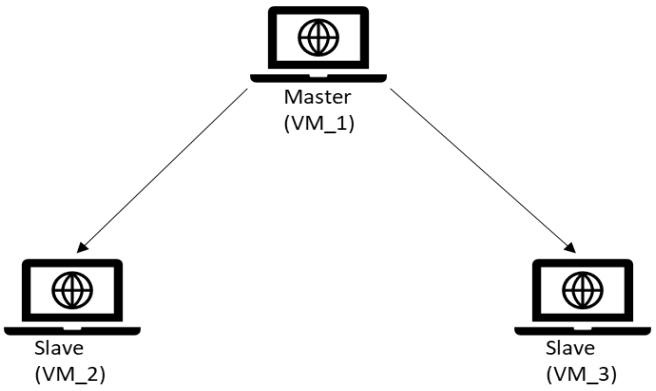

```
{"ID":{"0":"T0"},"name":{"0":"Albert"},"deadline":{"0":"2022-10-01 10
    :00:00"},"data_size_GB":{"0":2},"type":{"0":"classification"}
    ,"get_paid_per_hour":{"0":5}}
```


Renter JSON:


```
{"ID":{"0":"R0"},"name":{"0":"Adam"},"available_start_time":{"0"
    :"2022-01-01 10:00:00"},"available_end_time":{"0":"2022-01-01 20
    :00:00"},"pay_per_hour":{"0":1},"CPU_name":{"0":"Intel_Xeon"}
    ,"CPU_core":{"0":8},"CPU_RAM_GB":{"0":32},"GPU_name":{"0"
    :"GeForce_RTX_3060"},"GPU_core":{"0":5000},"GPU_RAM_GB":{"0":6}}
```


2.1. System Architecture

Here, we tried to implement the Master-Slave Architecture for connecting different VM's. The main reason to implement this architecture is that the user data (i.e. tenant and renter data) is critical in the working of our business idea and to safeguard the data we wanted to replicate and store the data safely over the different nodes. The architecture of our system as shown in the below diagram.



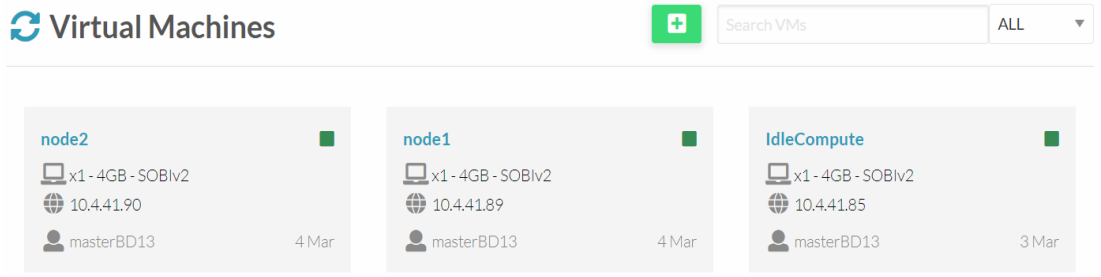
[1] Tenants can be individuals or organisations business, non-profit, etc.
[2] Renter can be individual computer owners or big organisations with large computation powers.
[3] Proof of concept (POC) for task processing is done by the Team BDMA11-B1 (Adam Broniewski and Vlada Kylynnyk).
[4] Registration forms are created using Google forms to collect the tenant/supplier details.

There are various benefits of Hadoop cluster :

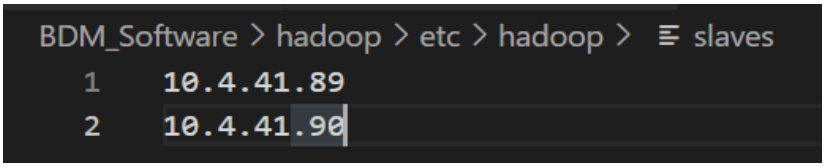
1. Hadoop clusters replicate a data set across the distributed file system, making them resilient to data loss and cluster failure.
2. Also as a part of the quality of service, Hadoop clusters keep performing tasks even if some of its components fail. Hence, the data is always available and reliable.
3. Given the scalability aspect of Hadoop is able to continuously evolve and support a growing amount of data/tasks.

2.2. Implementation steps

1. Create the VM's on https://virtech.fib.upc.edu/ which will be our nodes. Here, *IdleCompute* node acts as a master node. The *node1* and *node2* acts as a slave nodes.



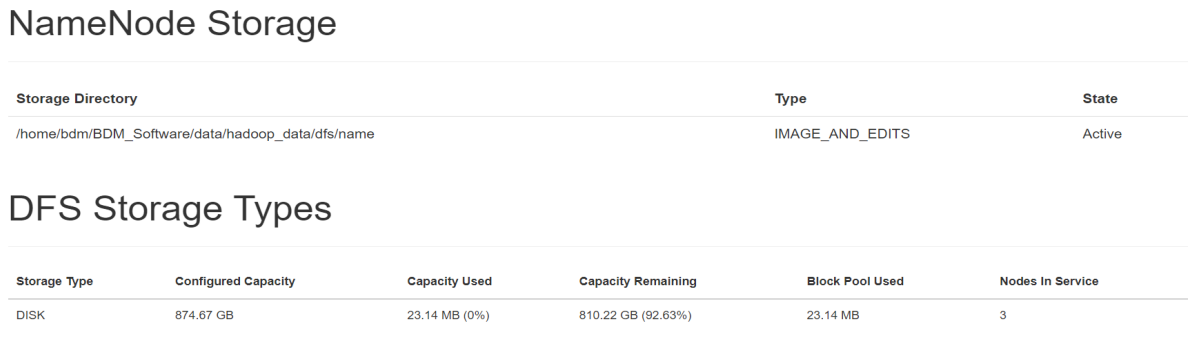
2. Go to folder : *hadoop→etc→slaves* file and add the IP addresses of the slave nodes.



3. Then we copied core-site.xml to the slave nodes to let HDFS know which node is their master node (i.e. *IdleCompute* node with IP - 10.4.41.85)

```
(base) bdm@bellsprout:~/BDM_Software/hadoop/etc/hadoop$ scp core-site.xml
bdm@10.4.41.85:~/BDM_Software/hadoop/etc/hadoop/
core-site.xml
100% 991 1.4MB/s 00:00
(base) bdm@bellsprout:~/BDM_Software/hadoop/etc/hadoop$ scp core-site.xml
bdm@10.4.41.89:~/BDM_Software/hadoop/etc/hadoop/
core-site.xml
100% 991 863.1KB/s 00:00
(base) bdm@bellsprout:~/BDM_Software/hadoop/etc/hadoop$ scp core-site.xml
bdm@10.4.41.90:~/BDM_Software/hadoop/etc/hadoop/
core-site.xml
100% 991 708.6KB/s 00:00
```

4. Now our Master-slave architecture setup is ready as shown below :



3. Data collectors

In *PyHdfsConnect.py* file, we have implemented our code for the data collector. When the web page calls its API, this python file in our service will also receive an *api_url* to get the information of the user's input (JSON file).
Below is the code for get api_url, where we get/collect the user (i.e. tenant/supplier) information from the Registration form.

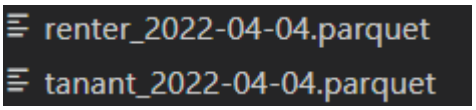
```
api_url = "https://Idlecompute.com/renter_submit/1"
response = requests.get(api_url)
daily_renter = response.json()

api_url = "https://Idlecompute.com/tanant_submit/1"
response = requests.get(api_url)
daily_tanant = response.json()
```

4. Landing Zone

4.1. Temporal landing

In the *PyHdfsConnect.py*, while we get a new JSON file from API, we save it into our dataframe and append it to the existing data we got on that particular date in *tenant|renter_YYYY-MM-DD* format, this is our temporal landing part. At the end of each day, we convert the whole file of tenants and renters into two separate *Parquet*[5] files, and then put it into our HDFS, after that, we delete all the files from the temporal landing zone.
For example, these two files are what we get at the end of 2022-04-04.



In other words, our temporal landing zone only saves the data for that particular date and at the end of the day, it moves the data to the persistent landing zone (HDFS), and then flush all the data to prepare for the next day.

4.2. Data format

After the data conversion from JSON to Parquet in the temporal landing zone, the data format is as shown below:

A. Tenant data format

ID	Name	Deadline	Data_size	Type	Get_paid_per_hour
T0	Albert	01-10-2022 10:00:00	2	classification	5
T1	Byron	24-10-2022 09:05:26	61	classification	9
T2	Emma	29-05-2022 06:11:19	34	reinforcement_learning	5

The unit for the Data_size is GB.

[5] Parquet is a column-oriented data file format designed for efficient data storage and retrieval. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk.

The tenant dataset is stored as tenant_YYYY-MM-DD.parquet.

B. Renter data format

ID	Name	Available_start_time	Available_end_time	Pay_per_hour	CPU_name	CPU_core	CPU_RAM	GPU_name	GPU_core	GPU_RAM_GB
R0	Alex	01-01-2022 10:00:00	01-01-2022 20:00:00	1	Intel_Xeon	8	32	GeForce_RTX_3060	5000	6
R1	Brown	06-04-2022 00:54:38	23-08-2022 03:04:16	4	AMD_Rome_Epyc	1	1024	GeForce_RTX_3070	3550	8
R3	Craig	02-01-2022 20:35:52	17-12-2022 00:16:35	2	AMD_Milan_Epyc	4	32	Nvidia_TITAN_V	4877	32
R4	Kelley	19-01-2022 14:42:37	19-11-2022 02:40:06	3	AMD_Milan_Epyc	2	256	Nvidia_TITAN_X	8269	16

The supplier dataset is stored as renter_YYYY-MM-DD.parquet.

4.3. Persistent landing

At the end of each day after temporal landing, we receive two Parquet files as- renter_YYYY-MM-DD.parquet and tenant_YYYY-MM-DD.parquet in HDFS. These files are then stored in HDFS as their persistent landing.

We run our code from 2022-03-15 to 2022-04-04, and this is how it looks like in the HDFS as shown below :

☐	-rw-r--r--	bdm	supergroup	11.75 KB	Apr 04 14:39	3	128 MB	renter_2022-04-01.parquet
☐	-rw-r--r--	bdm	supergroup	12.64 KB	Apr 04 14:39	3	128 MB	renter_2022-04-02.parquet
☐	-rw-r--r--	bdm	supergroup	11.91 KB	Apr 04 14:39	3	128 MB	renter_2022-04-03.parquet
☐	-rw-r--r--	bdm	supergroup	10.05 KB	Apr 04 14:56	3	128 MB	renter_2022-04-04.parquet
☐	-rw-r--r--	bdm	supergroup	6.79 KB	Apr 04 14:39	3	128 MB	tanant_2022-03-15.parquet
☐	-rw-r--r--	bdm	supergroup	6.57 KB	Apr 04 14:39	3	128 MB	tanant_2022-03-16.parquet
☐	-rw-r--r--	bdm	supergroup	5.92 KB	Apr 04 14:39	3	128 MB	tanant_2022-03-17.parquet
☐	-rw-r--r--	bdm	supergroup	6.23 KB	Apr 04 14:39	3	128 MB	tanant_2022-03-18.parquet

Why HDFS?

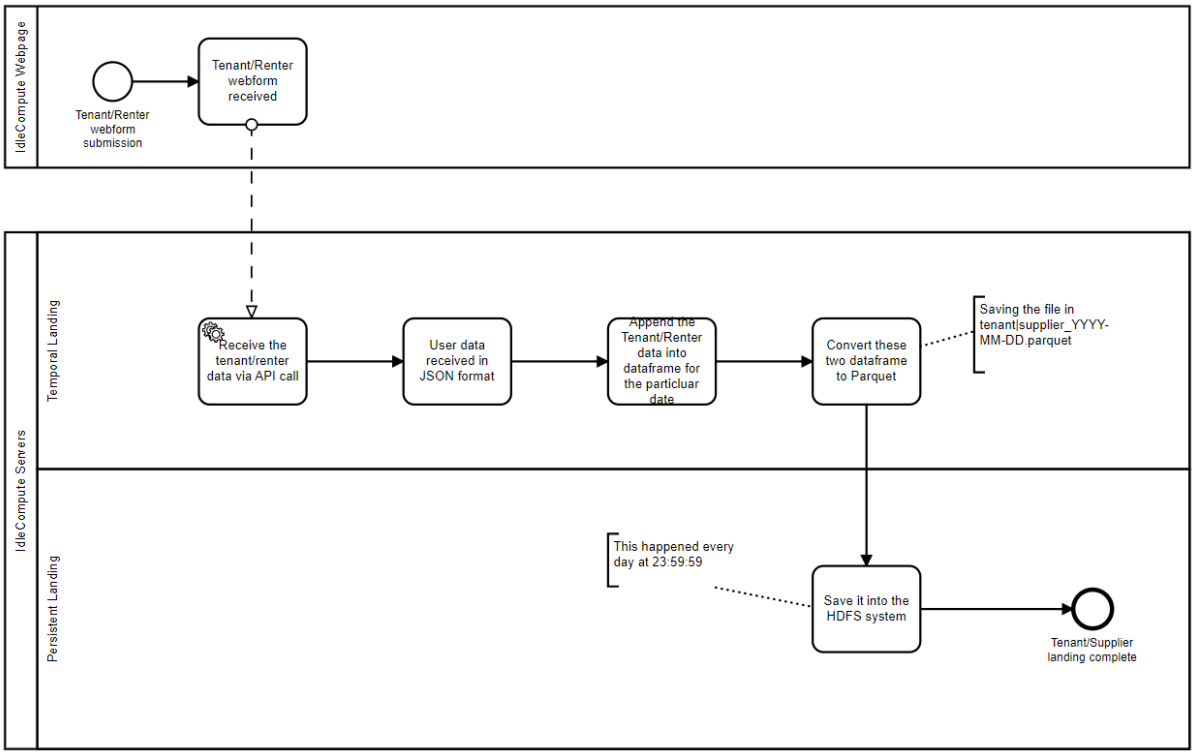
1. HDFS can store large amounts of data, it is also scalable and has fast access to the information, so we can scale up easily if we have more and more customers in the future.
2. In our business idea, we need to distribute the data, HDFS allows us to do it in a very simple way.
3. We will have a lot of clients, HDFS can serve a substantial number of clients by adding more machines to the cluster.
4. High Reliability by automatically maintaining multiple copies of data and automatically redeploying processing logic in the event of failures. This feature makes our system more stable.
5. To work with the other group in the VBP course, it is a good idea if we use the same system.

6. In the next lab, we are planning to implement the MapReduce framework using Hadoop, and HDFS is their file system. We would be using the MapReduce framework to calculate all the matching resources and tasks in a single time unit.[6]

Why Parquet?

1. It can be compressed while storing if we have large data in the future.
2. Query time is faster because of the column-based storage and presence of metadata.
3. As mentioned before, we are going to use Hadoop in the future and Parquet is a columnar storage format in the Hadoop ecosystem.[7]

5. The BPMN diagram - IdleCompute



[6] *Hadoop HDFS Overview*. (2022). Hadoop HDFS. https://www.tutorialscampus.com/hadoop/hdfs-overview.htm

[7] Thoughts, B. (2021, November 25). *Probably The Best Alternative to CSV Storage: Parquet Data*. Geekflare. https://geekflare.com/parquet-csv-data-storage/