Parallel Processing

Project 1 Report

NASA Kennedy Space Center web server in Florida.

Students:

Al-Anoud Al-Subaie

Raghad Alsuhaibani

Table of Contents:

Introduction To Big Data :

Big Data is the term for a collection of datasets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. "Big Data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze." (McKinsey Global Institute)[1].

Big data has increased the demand of information management specialists so much so that Software AG, Oracle Corporation, IBM, Microsoft, HP and Dell have spent more than $15 billion on software firms specializing in data management and analytics. In 2010, this industry was worth more than $100 billion and was growing at almost 10 percent a year: about twice as fast as the software business as a whole.[2]applications big data such as Education and Insurance , Internet of Things (IoT) ,Government .

Big data can be described by the following characteristics:

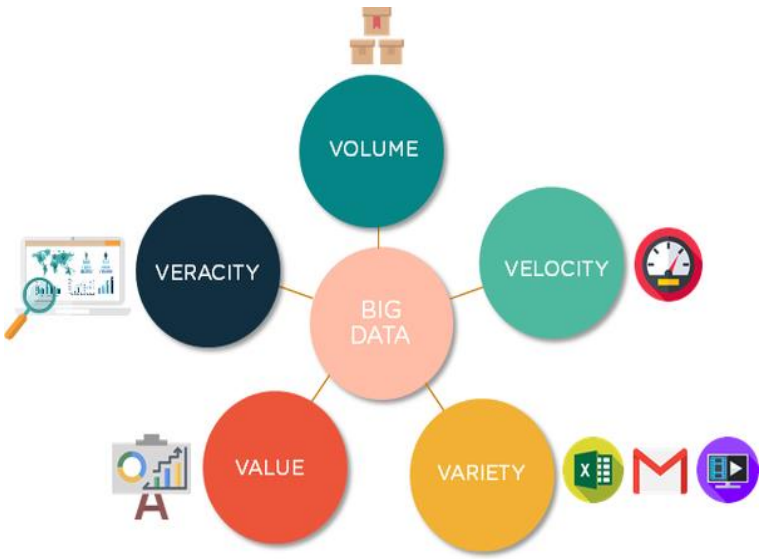
1-**Volume**: The quantity of generated and stored data.

2-**Variety**:The type and nature of the data.

3-**Velocity**:The speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.

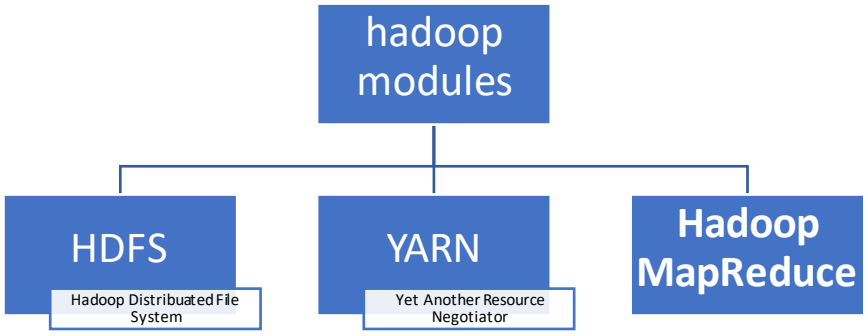
4-**Veracity**: It is the extended definition for big data, which refers to the data quality and the data value.

5-**Value** : insights and impact .



Hadoop introduction:

Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.



Dataset Description:

Typically, server logs are a very common data source in enterprises and log data comes from many sources in an enterprise, such as the web, client and compute servers, and applications. They can be used for monitoring servers, improving business and customer intelligence, building recommendation systems and much more. In this project, we will use log datasets from NASA Kennedy Space Center web server in Florida. These datasets contain two months' worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida.

The logs are an ASCII file with one line per request, with the following columns:

1. **host** - making the request. A hostname when possible, otherwise the Internet address if the name could not be looked up.
2. **timestamp** - in the format "DAY MON DD HH:MM:SS YYYY", where DAY is the day of the week, MON is the name of the month, DD is the day of the month, HH:MM:SS is the time of day using a 24-hour clock, and YYYY is the year. The timezone is -0400.
3. **request URL** - given in quotes.
4. **HTTP** reply code.
5. **Bytes returned by the server.**

Figure of the project :

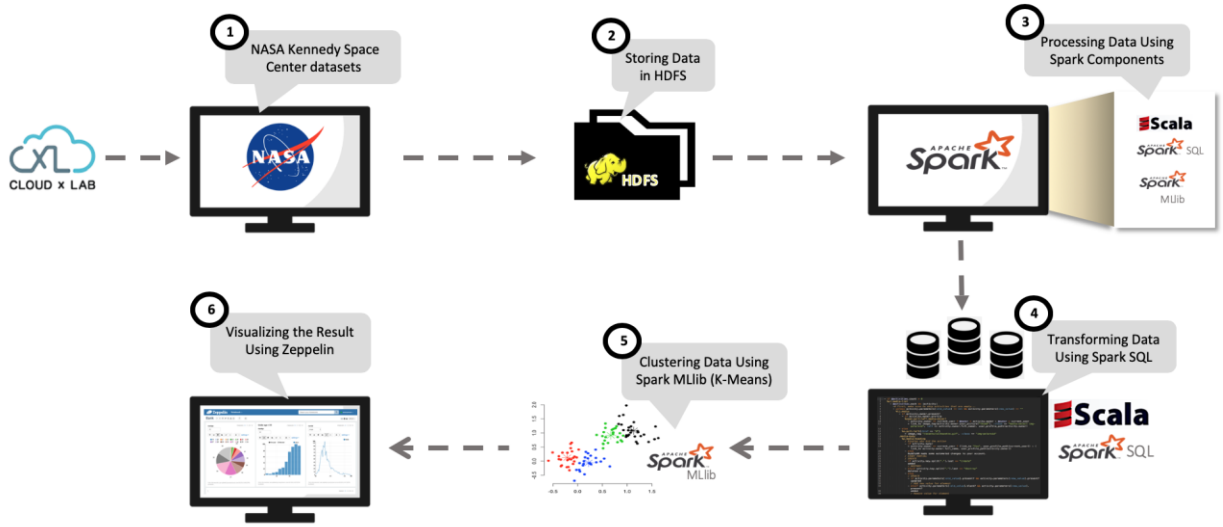


Figure 1

Dataset installing :

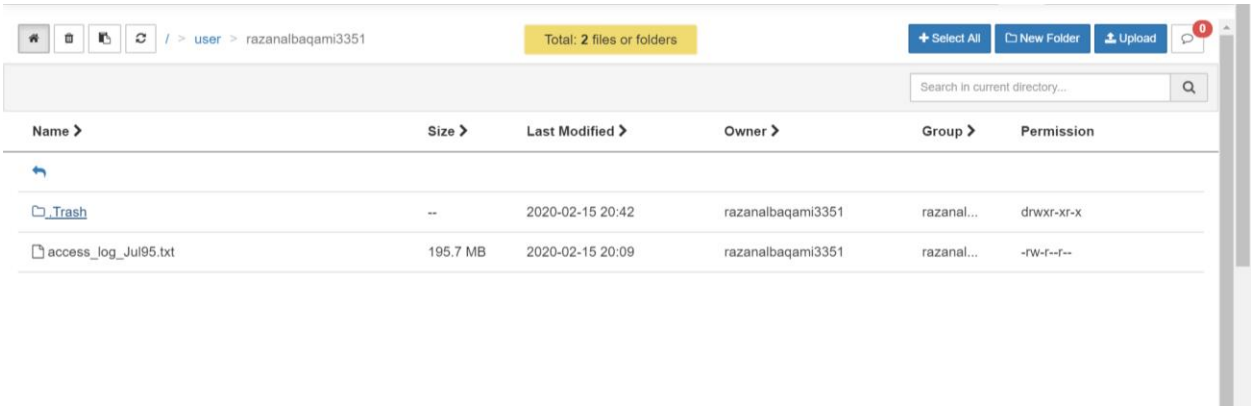


Figure 2. where we want to upload the dataset using ambari UI.

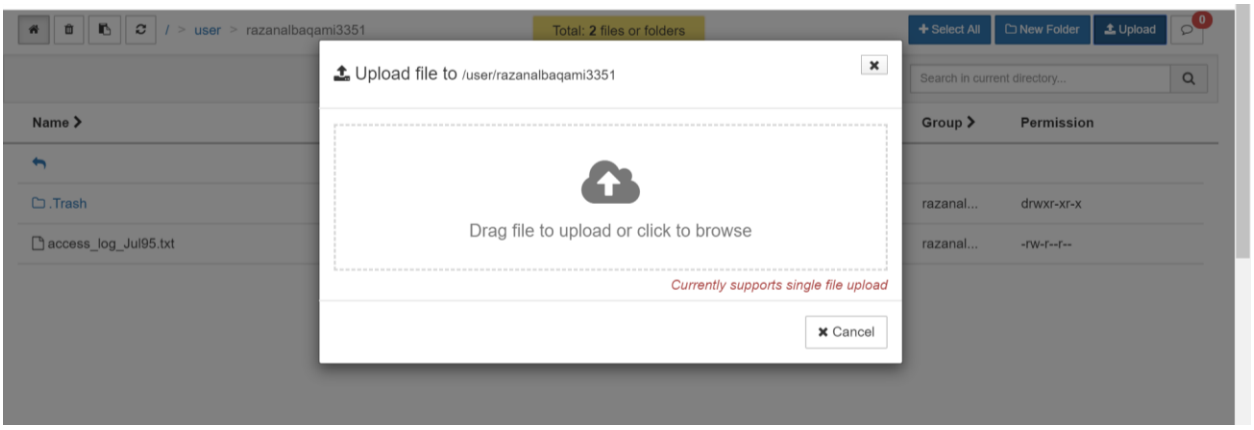


Figure 3. we select the data set.

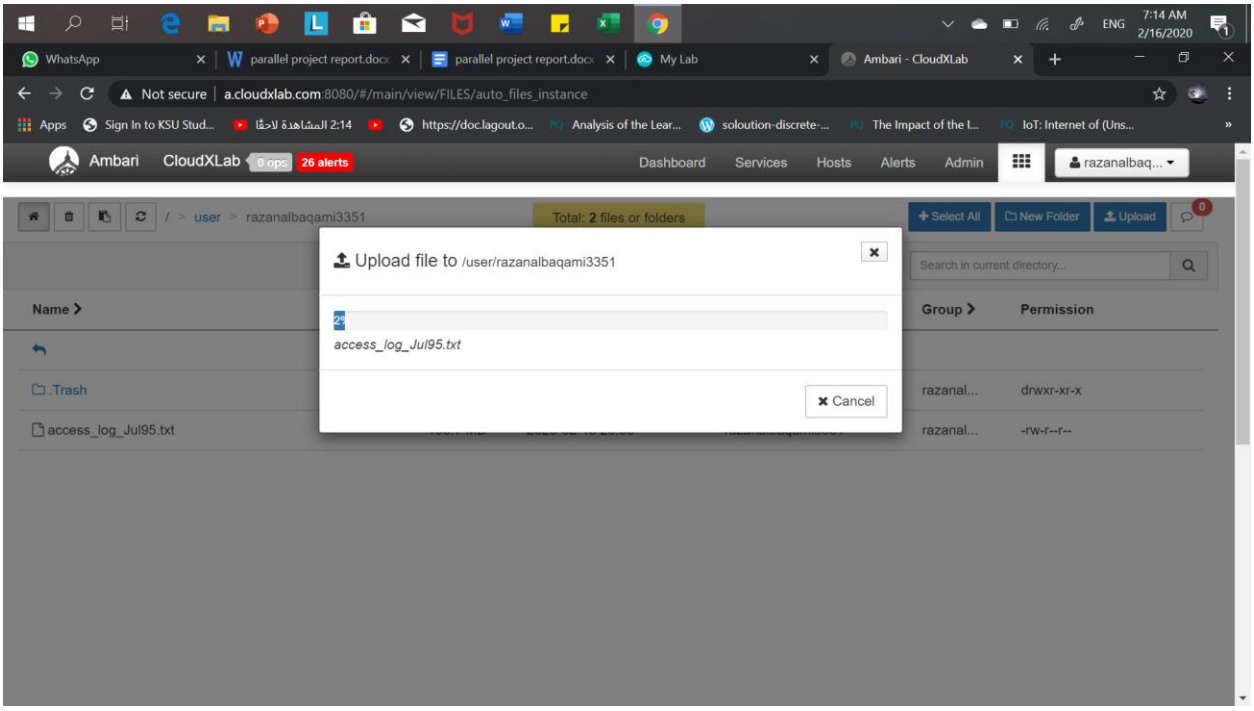


Figure 4. upload it.

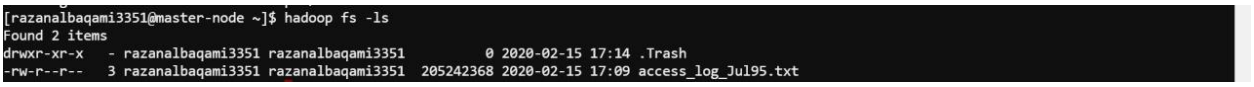


Figure 5. check if the process done correctly.

Figure 6. set the replication factor.

```
[razanalbaqami3351@master-node ~]$ hdfs fsck -blocks -locations -racks  -files /user/razanalbaqami3351/access_log_Jul95.txt
Connecting to namenode via http://cxln1.c.thelab-240901.internal:50070/fsck?ugi=razanalbaqami3351&blocks=1&locations=1&racks=1&files=1&path=%2Fuser%2Fraz
analbaqami3351%2Faccess_log_Jul95.txt
FSCK started by razanalbaqami3351 (auth:SIMPLE) from /10.142.1.4 for path /user/razanalbaqami3351/access_log_Jul95.txt at Sat Feb 15 18:48:57 UTC 2020
/user/razanalbaqami3351/access_log_Jul95.txt 205242368 bytes, 2 block(s):  OK
0. BP-726716649-10.142.1.1-1559900602525:blk_1075697148_1963057 len=134217728 repl=4 [/default-rack/10.142.1.3:50010, /default-rack/10.142.1.1:50010, /de
fault-rack/10.142.1.4:50010, /default-rack/10.142.1.2:50010]
1. BP-726716649-10.142.1.1-1559900602525:blk_1075697149_1963058 len=71024640 repl=4 [/default-rack/10.142.1.1:50010, /default-rack/10.142.1.2:50010, /def
ault-rack/10.142.1.4:50010, /default-rack/10.142.1.3:50010]

Status: HEALTHY
 Total size:    205242368 B
 Total dirs:    0
 Total files:   1
 Total symlinks:                0
 Total blocks (validated):      2 (avg. block size 102621184 B)
 Minimally replicated blocks:   2 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     4.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          4
 Number of racks:               1
FSCK ended at Sat Feb 15 18:48:57 UTC 2020 in 1 milliseconds


The filesystem under path '/user/razanalbaqami3351/access_log_Jul95.txt' is HEALTHY
```

Figure 7. Check the replication process and the data blocks

Processing Data Using Spark Components :

```
import os
import sys
import re

os.environ["SPARK_HOME"] = "/usr/spark2.4.3"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"
# In below two lines, use /usr/bin/python2.7 if you want to use Python 2
os.environ["PYSPARK_PYTHON"] = "/usr/local/anaconda/bin/python"
os.environ["PYSPARK_DRIVER_PYTHON"] = "/usr/local/anaconda/bin/python"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")


from pyspark import SparkContext, SparkConf
from pyspark.sql import Row
from pyspark.sql import SparkSession


PATTERN = '^(\S+) (\S+) (\S+) \[([\w:/]+\s[+\-]\d{4})\] "(\S+) (\S+)(.*)" (\d{3}) (\S+)'

#Example line
line = 'slppp6.intermind.net - - [01/Aug/1995:00:00:39 -0400] "GET /history/skylab/skylab-logo.gif HTTP/1.0" 200 3274'
#line = 'slppp6.intermind.net - - [01/Aug/1995:00:00:39 -0400] "GET /history/skylab/skylab-logo.gif HTTP/1.0" XXX 3274

def parseLogLine(log):
    m = re.match(PATTERN, log)
    if m:
        return [Row(host=m.group(1), timeStamp=m.group(4),url=m.group(6), httpCode=int(m.group(8)))]
    else:
        return []

#Test if it is working
parseLogLine(line)
```

Figure 8

```
root
 |-- host: string (nullable = true)
 |-- httpCode: long (nullable = true)
 |-- timeStamp: string (nullable = true)
 |-- url: string (nullable = true)
```

Figure 9 : Test if ParseLogLine method it's working .

Transforming Data Using Spark SQL :

```
spark = SparkSession.builder.appName("PythonWordCount").getOrCreate()
sc = spark.sparkContext

logFile = sc.textFile("/user/razanalbaqami3351/access_log_Jul95.txt")

accessLog = logFile.flatMap(parseLogLine)
accessDf = spark.createDataFrame(accessLog)
accessDf.printSchema()
accessDf.createOrReplaceTempView("nasalog")
output = spark.sql("select * from nasalog")
output.createOrReplaceTempView("nasa_log")

# Uncomment this for optimizations only
#spark.sql("cache TABLE nasa_log")

spark.sql("select url,count(*) as req_cnt from nasa_log where upper(url) like '%HTML%' group by url order by req_cnt d

spark.sql("select host,count(*) as req_cnt from nasa_log group by host order by req_cnt desc LIMIT 5").show()

spark.sql("select substr(timeStamp,1,14) as timeFrame,count(*) as req_cnt from nasa_log group by substr(timeStamp,1,14

spark.sql("select substr(timeStamp,1,14) as timeFrame,count(*) as req_cnt from nasa_log group by substr(timeStamp,1,14

spark.sql("select httpCode,count(*) as req_cnt from nasa_log group by httpCode ").show()
```

Figure 10

Output :

```
+--------------------+--------+
|                 url|req_cnt|
+--------------------+--------+
|           /ksc.html|   40231|
|/shuttle/missions...|   24865|
|/shuttle/countdow...|   22000|
|/shuttle/missions...|   16717|
|/shuttle/missions...|   16123|
|/shuttle/missions...|   15897|
|/history/apollo/a...|   14472|
|/history/apollo/a...|   13768|
|/history/history....|   11818|
|/shuttle/countdow...|    8572|
+--------------------+--------+
```

Figure 11

We write spark code to find out top 5 hosts / IP making the request along with count (This information will help to find out locations where website is popular or to figure out potential DDoS attacks)

```
+--------------------+-------+
|                host|req_cnt|
+--------------------+-------+
|piweba3y.prodigy.com|  17572|
|piweba4y.prodigy.com|  11591|
|piweba1y.prodigy.com|   9868|
|  alyssa.prodigy.com|   7852|
| siltb10.orl.mmc.com|   7573|
+--------------------+-------+
```

Figure 12

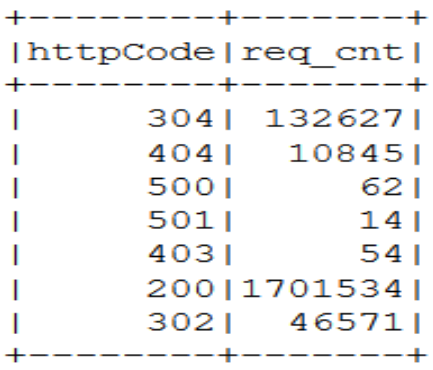
We write spark code to find out top 5 time frame for high traffic (which day of the week or hour of the day receives peak traffic, this information will help to manage resources for handling peak traffic load)

```
+--------------+-------+
|     timeFrame|req_cnt|
+--------------+-------+
|13/Jul/1995:09|  14926|
|13/Jul/1995:08|  11567|
|13/Jul/1995:10|  10214|
|13/Jul/1995:11|   9748|
|13/Jul/1995:12|   8533|
+--------------+-------+

+--------------+-------+
|     timeFrame|req_cnt|
+--------------+-------+
|18/Jul/1995:04|    316|
|18/Jul/1995:06|    371|
|09/Jul/1995:05|    397|
|18/Jul/1995:05|    409|
|09/Jul/1995:06|    442|
+--------------+-------+
```

Figure 13

We write spark code to find out 5 time frames of least traffic (which day of the week or hour of the day receives least traffic, this information will help to do production deployment in that time frame so that less number of users will be affected if something goes wrong during deployment.



httpCode	req_cnt
304	132627
404	10845
500	62
501	14
403	54
200	1701534
302	46571

Figure 14

We write spark code to find out unique HTTP codes returned by the server along with count (this information is helpful for devops team to find out how many requests are failing so that appropriate action can be taken to fix the issue).

Reference :

1-Mckinsey.com. (2020). [online] Available at: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Big%20data%20The%20next%20frontier%20for%20innovation/MGI_big_data_exec_summary.ashx [Accessed 5 Feb. 2020].

2-"Data, data everywhere". The Economist. 25 February 2010. Retrieved 9 December 2012.