Big Data Analytics Techniques and Applications

Homework 3

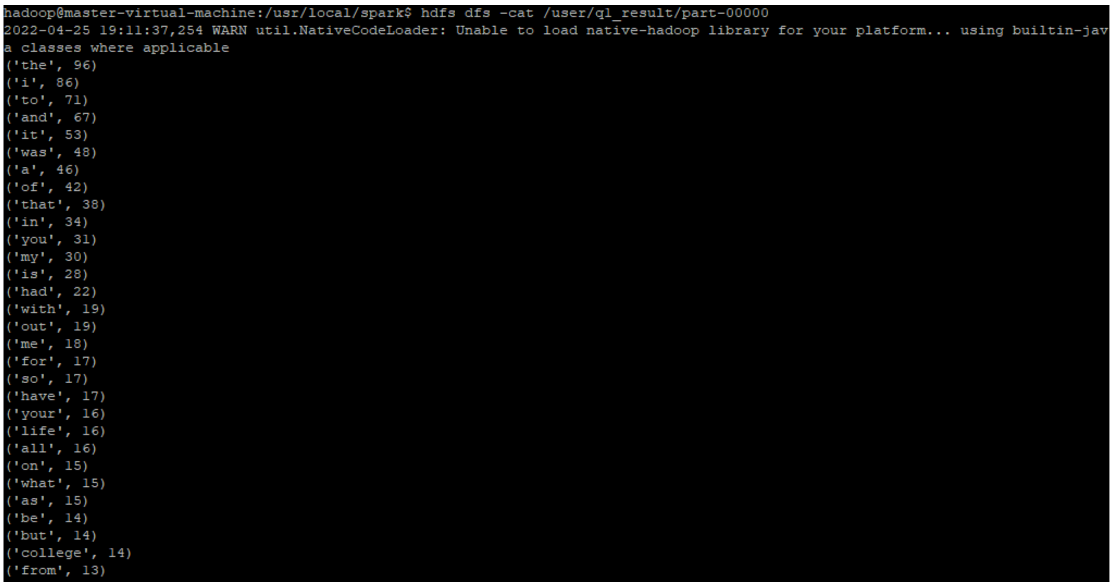
309552063 吳冠潔

使用平台：VM

Q1: Implement a program to calculate the average occurrences of each word in a sentence in the attached article (Youvegottofindwhatyoulove.txt).

A. Show the top 30 most frequent occurring words and their average occurrences in a sentence.

Result A:



```
hadoop@master-virtual-machine:/usr/local/spark$ hdfs dfs -cat /user/q1_result/part-00000
2022-04-25 19:11:37,254 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
a classes where applicable
('the', 96)
('i', 86)
('to', 71)
('and', 67)
('it', 53)
('was', 48)
('a', 46)
('of', 42)
('that', 38)
('in', 34)
('you', 31)
('my', 30)
('is', 28)
('had', 22)
('with', 19)
('out', 19)
('me', 18)
('for', 17)
('so', 17)
('have', 17)
('your', 16)
('life', 16)
('all', 16)
('on', 15)
('what', 15)
('as', 15)
('be', 14)
('but', 14)
('college', 14)
('from', 13)
```

B. According to the result, what are the characteristics of these words?

Result B:

Function word 還有人稱代名詞是最常出現的字

Method:

一開始先把 spark import 進去程式中，接著把 SparkConf 物件，並給定程式名稱，接著建立 SparkContext

```
from pyspark import SparkContext
from pyspark import SparkConf

sparkConf = SparkConf().setAppName("CountWord")

sc = SparkContext(conf = sparkConf)
```

接著把 txt 檔先用 hdfs dfs -put Youvegottofindwhatyoulove.txt /test，放到 hdfs 的/test 中，就可以將 txt 檔用 sc.textFile，讀入成 RDD

```
text = sc.textFile("hdfs://master:9000/test/Youvegottofindwhatyoul
ove.txt")
```

下一步就可以算字出現的頻率，一開始先將標點符號替換掉空白，以免有字被切成和標點符號一起，以致於沒有被算到正確的字數，替換好後，用 split 以空白將字做切割，接下來將每個字先用 map 對應到 1，代表有 1 次出現，然後用 reduceByKey 計算相同字次數，最後因為題目要 30 個最常出現的字，所以將其以次數做排序

```
wordcount = text.map(lambda x: x.replace(',', ' ').replace('.', '
') \
           .replace('"', ' ').replace('“', ' ').replace('”', '
').lower()) \
           .flatMap(lambda x: x.split()) \
           .map(lambda x: (x, 1)) \
           .reduceByKey(lambda x,y: x+y) \
           .sortBy(lambda x: x[1], ascending=False)
```

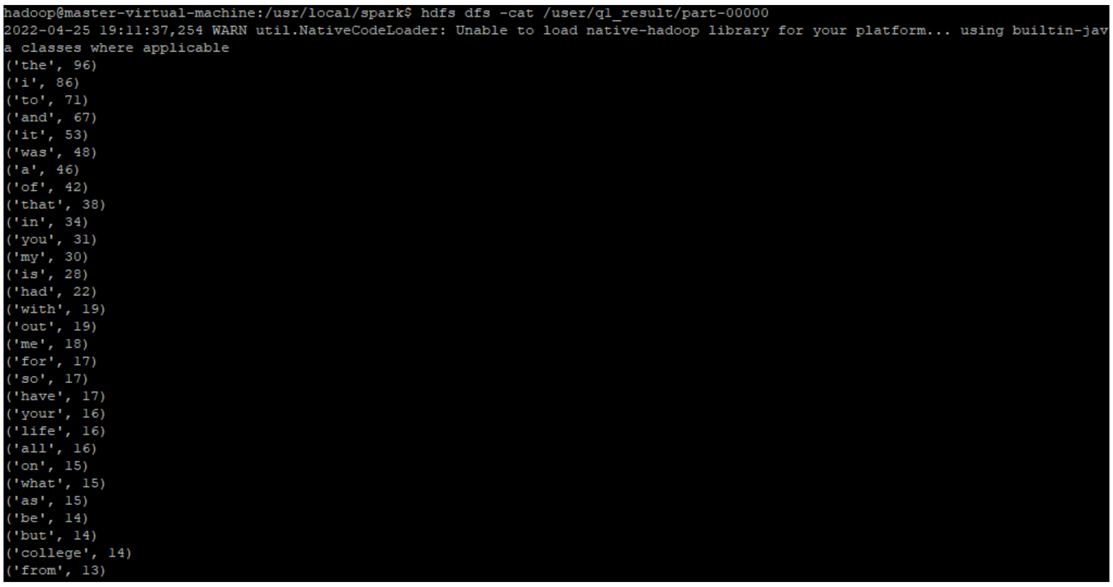
最後將結果存到 hdfs 的 user/q1_output 中

```
wordcount.coalesce(1).saveAsTextFile("hdfs://master:9000/user/q1_o
utput/")
```

執行程式

```
./bin/spark-submit --master yarn --deploy-mode cluster ~/hw3/Q1_co
untword.py
```

最後將 user/q1_output 中儲存的結果用 cat 印出來，結果如下圖，就可以得到前 30 個出現次數最多的字



```
hadoop@master-virtual-machine:/usr/local/spark$ hdfs dfs -cat /user/q1_result/part-00000
2022-04-25 19:11:37,254 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
a classes where applicable
('the', 96)
('i', 86)
('to', 71)
('and', 67)
('it', 53)
('was', 48)
('a', 46)
('of', 42)
('that', 38)
('in', 34)
('you', 31)
('my', 30)
('is', 28)
('had', 22)
('with', 19)
('out', 19)
('me', 18)
('for', 17)
('so', 17)
('have', 17)
('your', 16)
('life', 16)
('all', 16)
('on', 15)
('what', 15)
('as', 15)
('be', 14)
('but', 14)
('college', 14)
('from', 13)
```

Q2: In **YARN cluster mode**, implement a program to calculate the average amount ("Total_amount") in credit card trip and cash trip for different numbers of passengers, which are from one to four passengers in 2018/10 NYC Yellow Taxi trip data. In NYC Taxi data, the "Passenger_count" is a driver-entered value. Explain also how you deal with the data loss issue.

Result:

	Credit Card	Cash
Average amount	18.231360325967003	6.980922144486943

Method:

一開始先將把 SparkConf 以及 SpaekContext 用好

```
import csv
from pyspark import SparkContext
from pyspark import SparkConf

sparkConf = SparkConf().setAppName("TaxiAvg")

sc = SparkContext(conf = sparkConf)
```

接著將資料用 textFile 讀出來，並將其用','做分割，分割好每一筆資料；再將 header 從資料中移除，在 rawdata.filter 這一步，除了將 header 移除，再用一個 filter 將沒有資料的空白行刪除

```
csvdata =
sc.textFile("hdfs://master:9000/test/yellow_tripdata_2018-10.csv")
```

```
rawdata = csvdata.map(lambda x: x.split(','))

header = rawdata.first()
tdata = rawdata.filter(lambda x: x!=header).filter(lambda x: len
(x)>1)
```

因為”Passenger_count”可能會有資料缺失的問題，並且在題目中寫道 different numbers of passengers, which are from one to four passengers，所以我用 filter 將 Passenger_count 不在 1~4 人的範圍中的資料都刪除，只保留符合條件的

```
data = tdata.filter(lambda x: (int(x[3]) > 0) & (int(x[3]) < 5))
```

分別對 credit card 和 cash 的資料做處理，在”Payment_type”中，根據 NYC Taxi data 網站中 csv 檔的 data dictionary describes yellow taxi trip data

Payment_type	A numeric code signifying how the passenger paid for the trip. **1= Credit card** **2= Cash** **3= No charge** **4= Dispute** **5= Unknown** **6= Voided trip**

”Payment_type”中 1 代表 credit card，2 代表 cash，所以分別用 filter 以不同的 Payment_type 抓取資料，接著將算出總乘客人數以及總金額，算出平均

```
# Credit Card
creditData = data.filter(lambda x: x[9] == '1')

cntCredit = creditData.map(lambda x: int(x[9])).sum()
amountCredit = creditData.map(lambda x: float(x[16])).sum()

avgCredit = amountCredit / cntCredit

# Cash
cashData = data.filter(lambda x: x[9] == '2')

cntCash = cashData.map(lambda x: int(x[9])).sum()
amountCash = cashData.map(lambda x: float(x[16])).sum()

avgCash = amountCash / cntCash
```

將結果存成 RDD，在將其放到 hdfs 的/user/q2_output/中

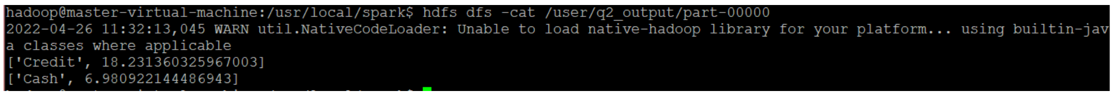
```
result = [['Credit', avgCredit], ['Cash', avgCash]]
resultrdd = sc.parallelize(result)

resultrdd.coalesce(1).saveAsTextFile("hdfs://master:9000/user/q2_o
utput/")
```

執行程式

```
./bin/spark-submit --master yarn --deploy-mode cluster ~/hw3/Q2_ta
xi.py
```

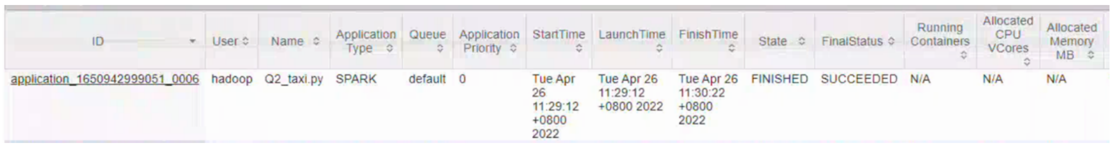
最後查看/user/q2_output/中的結果，就可以得到答案



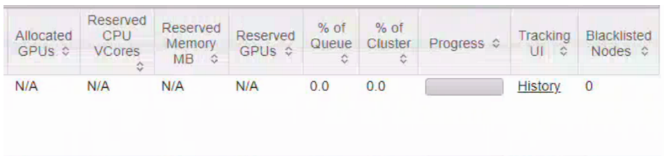
```
hadoop@master-virtual-machine:/usr/local/spark$ hdfs dfs -cat /user/q2_output/part-00000
2022-04-26 11:32:13,045 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
a classes where applicable
['Credit', 18.231360325967003]
['Cash', 6.980922144486943]
```

Other:
在處理 Passenger_count 時，我有印出來，發現有些資料是 5 人、6 人，但是因為在題目中提到乘客人數是 1~4 人，我就將超過 4 人的資料和空白資料一起刪除。

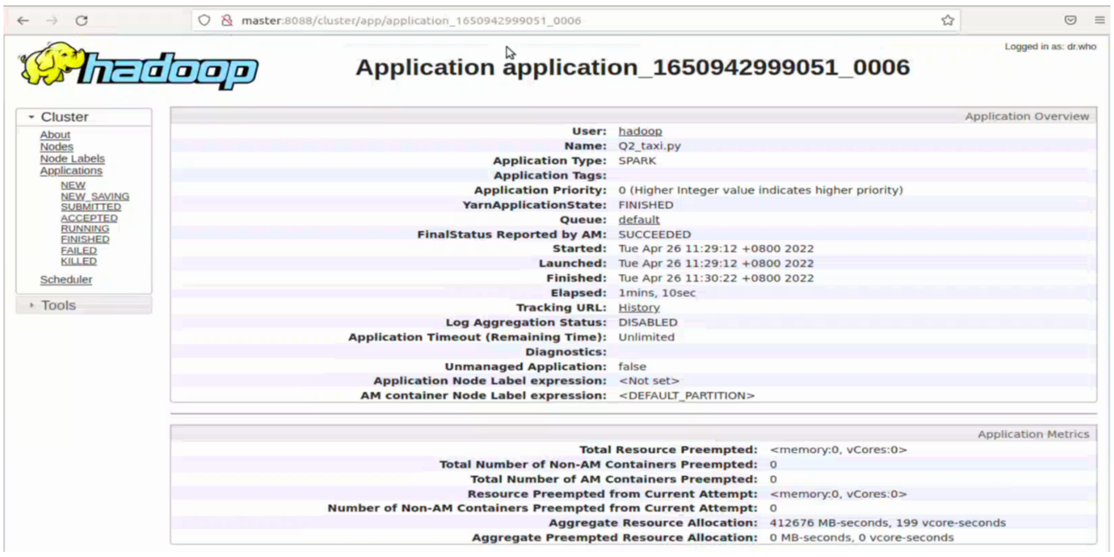
Q3: Referring to Q2, monitor HDFS and YARN metrics through HTTP API; collect MapReduce counters-related information through the web UI. Please provide screenshots and observations regarding the metrics in your report.
Result:



ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB
application_1650942999051_0006	hadoop	Q2_taxi.py	SPARK	default	0	Tue Apr 26 11:29:12 +0800 2022	Tue Apr 26 11:29:12 +0800 2022	Tue Apr 26 11:30:22 +0800 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A



Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
N/A	N/A	N/A	N/A	0.0	0.0		History	0



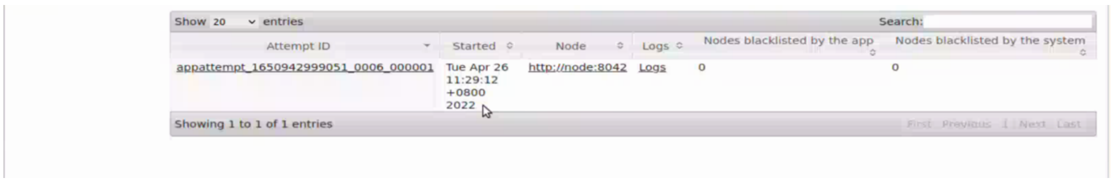
Application application_1650942999051_0006

Application Overview

User:	hadoop
Name:	Q2_taxi.py
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Tue Apr 26 11:29:12 +0800 2022
Launched:	Tue Apr 26 11:29:12 +0800 2022
Finished:	Tue Apr 26 11:30:22 +0800 2022
Elapsed:	1mins, 10sec
Tracking URL:	History
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

Application Metrics

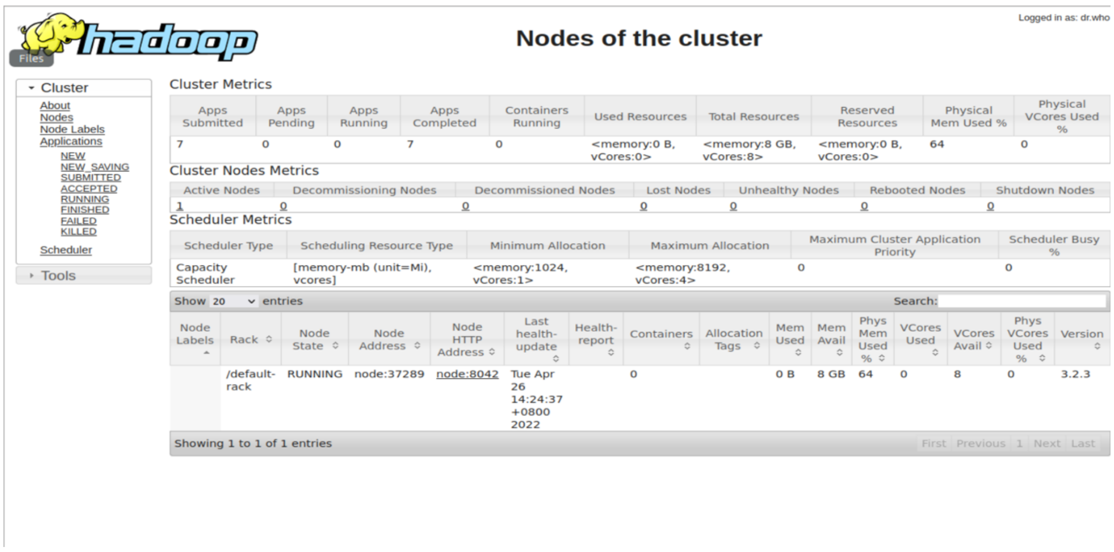
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	412676 MB-seconds, 199 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds



Show 20 entries Search:

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1650942999051_0006_000001	Tue Apr 26 11:29:12 +0800 2022	http://node:8042	Logs	0	0

Showing 1 to 1 of 1 entries



Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical VCores Used %
7	0	0	7	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:8>	<memory:0 B, vCores:0>	64	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

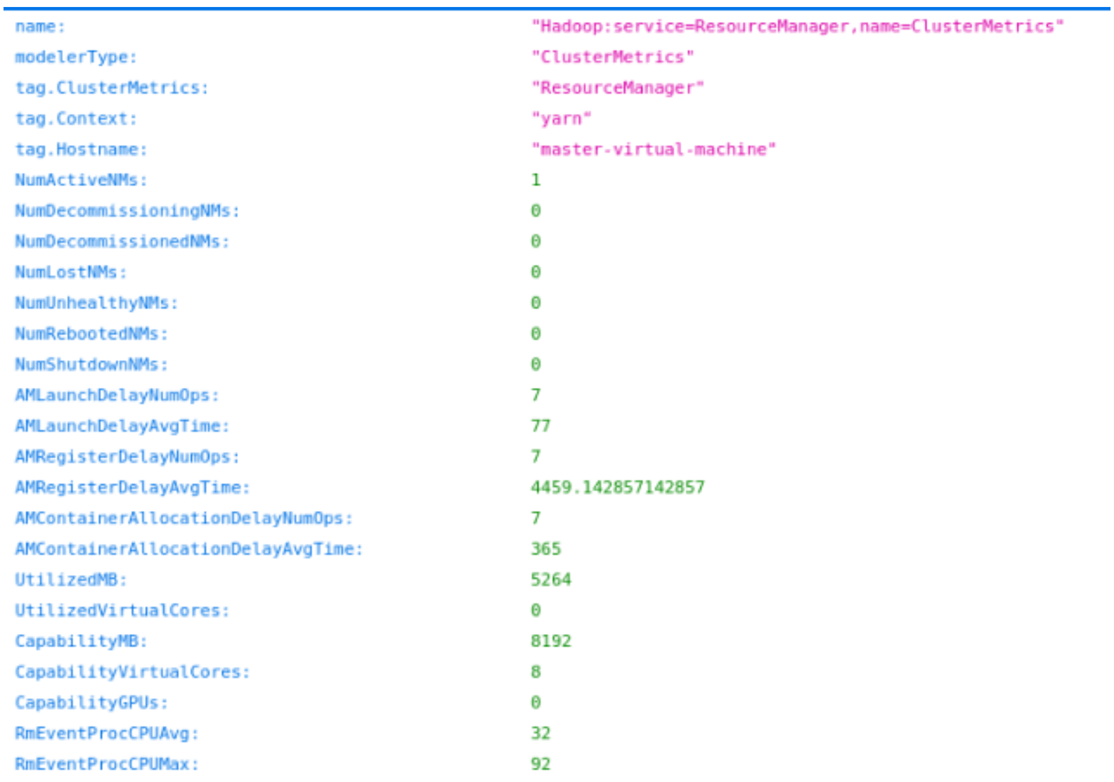
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority	Scheduler Busy %
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0	0

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	Phys Mem Used %	VCores Used	VCores Avail	Phys VCores Used %	Version
	/default-rack	RUNNING	node:37289	node:8042	Tue Apr 26 14:24:37 +0800 2022		0		0 B	8 GB	64	0	8	0	3.2.3

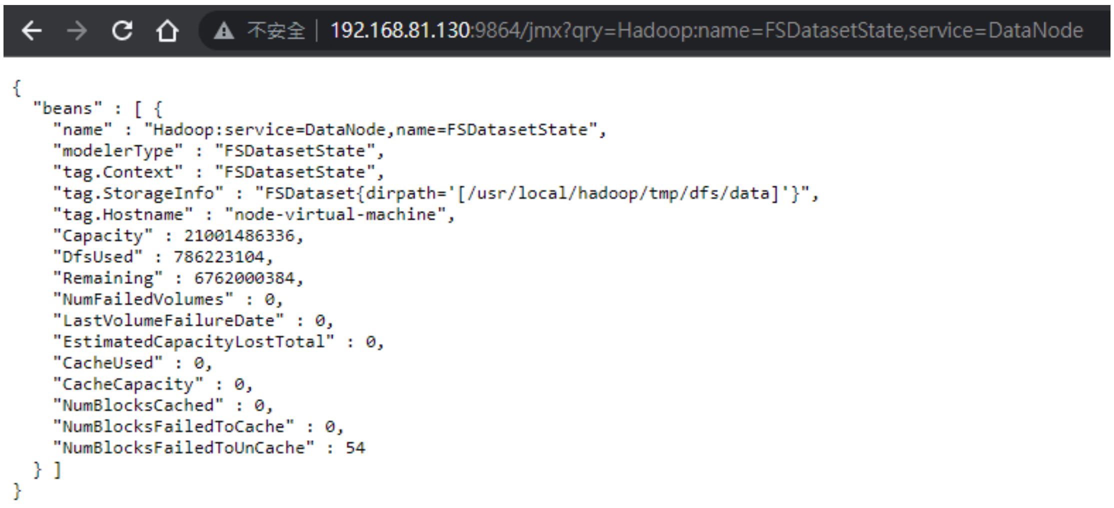
Showing 1 to 1 of 1 entries

192.168.81.129:8088/jmx



```
name:                                   "Hadoop:service=ResourceManager,name=ClusterMetrics"
modelerType:                            "ClusterMetrics"
tag.ClusterMetrics:                     "ResourceManager"
tag.Context:                            "yarn"
tag.Hostname:                           "master-virtual-machine"
NumActiveNMs:                           1
NumDecommissioningNMs:                  0
NumDecommissionedNMs:                   0
NumLostNMs:                             0
NumUnhealthyNMs:                        0
NumRebootedNMs:                         0
NumShutdownNMs:                         0
AMLaunchDelayNumOps:                    7
AMLaunchDelayAvgTime:                   77
AMRegisterDelayNumOps:                  7
AMRegisterDelayAvgTime:                 4459.142857142857
AMContainerAllocationDelayNumOps:       7
AMContainerAllocationDelayAvgTime:      365
UtilizedMB:                             5264
UtilizedVirtualCores:                   0
CapabilityMB:                           8192
CapabilityVirtualCores:                 8
CapabilityGPUs:                         0
RmEventProcCPUAvg:                      32
RmEventProcCPUMax:                      92
```

192.168.81.130:9864/jmx?qry=Hadoop:name=FSDatasetState,service=DataNode


不安全
192.168.81.130:9864/jmx?qry=Hadoop:name=FSDatasetState,service=DataNode

```
{
  "beans" : [ {
    "name" : "Hadoop:service=DataNode,name=FSDatasetState",
    "modelerType" : "FSDatasetState",
    "tag.Context" : "FSDatasetState",
    "tag.StorageInfo" : "FSDataset{dirpath='[/usr/local/hadoop/tmp/dfs/data]'}",
    "tag.Hostname" : "node-virtual-machine",
    "Capacity" : 21001486336,
    "DfsUsed" : 786223104,
    "Remaining" : 6762000384,
    "NumFailedVolumes" : 0,
    "LastVolumeFailureDate" : 0,
    "EstimatedCapacityLostTotal" : 0,
    "CacheUsed" : 0,
    "CacheCapacity" : 0,
    "NumBlocksCached" : 0,
    "NumBlocksFailedToCache" : 0,
    "NumBlocksFailedToUnCache" : 54
  } ]
}
```

Difficulties Encountered:

在這次作業前我從來沒有用過 Hadoop，所以一開始遭遇到很多困難，像是不知道如何在 hadoop 中執行 python 程式，還有如何將檔案匯入到 hdfs 中，這些在透過查詢網路上的教學後，有成功解決。

在如何將結果存到 hdfs 的時候嘗試了很久，明明已經使用正確的指令但還是沒有結果儲存到給予的路徑中，後來發現是要放到一個全新還不存在的資料夾中，

並且要在隨著程式執行時生成的在 hdfs 的資料夾/user 中，終於成功將結果存到資料夾中。因為在一開始第一題時試了很久沒辦法將結果存到路徑中，就只好將結果用 print 印出來，但是如果在 cluster mode 中，print 的執行結果沒有被印出來，就先用 client mode 看結果，在最後找到方法後，將其結果和用 client mode 的結果做相比。
在執行程式時，還有遇到一直在 Application report for application_ (state: ACCEPTED)這行一直執行不停止的狀況，後來重新啟用 hadoop 才解決問題。

Source Code:
Q1_countword.py
Q2_taxi.py