

Big Data Analytics Techniques and Applications _ HW3

310712009 楊家碩 (Nick Yang) GMBA

Environment Setup

Google Cloud Platform (GCP) 中的 Dataproc 項下設置叢集(Cluster)

Step1: 將區域設為 asia-east1 (台灣)，並勾選設定元件(Componets)與選用 jupyter notebook

← Create a Dataproc cluster on Compute Engine

• Set up cluster
Begin by providing basic information.

• Configure nodes (optional)
Change node compute and storage capabilities.

• Customize cluster (optional)
Add cluster properties, features, and actions.

• Manage security (optional)
Change access, encryption, and security settings.

CREATE CANCEL

EQUIVALENT COMMAND LINE ▾

Name

Cluster Name *
cluster-1 ?

Location

Region *
asia-east1 ?

Zone *
asia-east1-b ?

Cluster type

☒ Standard (1 master, N workers)

☐ Single Node (1 master, 0 workers)
Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing

☐ High Availability (3 masters, N workers)
Hadoop High Availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

Autoscaling

Automates cluster resource management based on an autoscaling policy.

Policy
None ▾

← Create a Dataproc cluster on Compute Engine

• Set up cluster
Begin by providing basic information.

• Configure nodes (optional)
Change node compute and storage capabilities.

• Customize cluster (optional)
Add cluster properties, features, and actions.

• Manage security (optional)
Change access, encryption, and security settings.

CREATE CANCEL

EQUIVALENT COMMAND LINE ▾

Versioning

Use a custom image to load pre-installed packages. [Learn more](#)

Image Type and Version
2.0-debian10

Release Date
First released on 1/22/2021.

CHANGE

Components

Component Gateway

☒ Enable component gateway
Provides access to the web interfaces of default and selected optional components on the cluster. [Learn more](#)

Optional components

Select one or multiple components. [Learn more](#)

☐ Anaconda ?

☐ Hive WebHCat ?

☒ Jupyter Notebook ?

☐ Zeppelin Notebook ?

☐ Druid ?

☐ Presto ?

☐ ZooKeeper ?

☐ Ranger ?

☐ HBase ?

Step2: 將主要節點(Master node)進行設定，系列(Series)設為”N1”；機型類型設為 n1-standard-2

Note: Customize cluster 與 Manage security 維持原本預設即可。

← Create a Dataproc cluster on Compute Engine

- Set up cluster
Begin by providing basic information.
- Configure nodes (optional)
Change node compute and storage capabilities.
- Customize cluster (optional)
Add cluster properties, features, and actions.
- Manage security (optional)
Change access, encryption, and security settings.

CREATE CANCEL

EQUIVALENT COMMAND LINE ▾

Master node

Contains the YARN Resource Manager, HDFS NameNode, and all job drivers.

Machine family


GENERAL-PURPOSE COMPUTE-OPTIMIZED MEMORY-OPTIMIZED

Machine types for common workloads, optimized for cost and flexibility

Series
N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type
n1-standard-2 (2 vCPU, 7.5 GB memory)

	vCPU 2	Memory 7.5 GB
---	-----------	------------------

✓ CPU PLATFORM AND GPU

Primary disk size *
250 GB ?

Primary disk type
Standard Persistent Disk ?

Number of local SS...
0 x 375GB ?

Local SSD interface
SCSI ?

Step3: 將資料放在值區(cloud storage)中，之後可以直接呼叫。

Note: 資料放到值區(cloud storage)中會被收費的。

☰

Google Cloud Platform

🔍

搜尋 產品、資源、文件 (/)

▼

☰

Cloud Storage

←

值區詳細資料

📁

瀏覽器

📊

Monitoring

⚙️

設定

yellow_tripdata_2018_10

位置

儲存空間級別

公開存取權

防護措施

asia-east1 (台灣)

Standard

非公開

無

物件

設定

權限

防護措施

生命週期

值區 > yellow_tripdata_2018_10 📁

上傳檔案

上傳資料夾

建立資料夾

管理前設保留

下載

刪除

只依名稱前置字串篩選 ▼

🔍 篩選 篩選物件和資料夾

<input type="checkbox"/>	名稱	大小	類型	建立時間 ⓘ	儲存空間級別	上次修改日期	公開存取權 ⓘ	版本記錄 ⓘ	加密 ⓘ
<input type="checkbox"/>	 yellow_tripdata_2018-10.csv	743.4 MB	text/csv	2022年4月19...	Standard	2022年4月19...	非公開	—	Google-managed key

Step4: 接下來回到 Dataproc 中的 Cluster (叢集) 頁面，點 “Web interfaces (網路介面)” ，再點 Jupyter 。

The screenshot shows the Google Cloud Platform interface for a Dataproc cluster named 'cluster-1'. The cluster is in a 'Stopped' status. The 'Web Interfaces' tab is selected, showing links for SSH tunnel, Component gateway, and various Hadoop ecosystem components like YARN ResourceManager, MapReduce Job History, Spark History Server, HDFS NameNode, YARN Application Timeline, and Tez. The 'Jupyter' link is highlighted in yellow.

Property	Value
Name	cluster-1
Cluster UUID	8a2cc7f9-813a-4472-aa3b-40203f017a8e
Type	Dataproc Cluster
Status	Stopped

MONITORING JOBS VM INSTANCES CONFIGURATION **WEB INTERFACES**

SSH tunnel
[Create an SSH tunnel to connect to a web interface](#)

Component gateway
Provides access to the web interfaces of default and selected optional components on the cluster. [Learn more](#)

[YARN ResourceManager](#)
[MapReduce Job History](#)
[Spark History Server](#)
[HDFS NameNode](#)
[YARN Application Timeline](#)
[Tez](#)
[Jupyter](#)

Step5: 再將程式匯入或是建立一個新的 Notebook，就可以運作了。

The screenshot shows the Jupyter Notebook interface. The top bar indicates the current directory is 'cluster-1'. The 'Files' tab is active, showing a file browser view of the 'Local Disk / HW3' directory. The file list includes '..' (seconds ago), 'HW3_310712009.ipynb' (2 days ago, 19.6 kB), and 'Untitled.ipynb' (9 days ago, 577 B).

silent-matter-347702 > cluster-1

jupyter

Files Running IPython Clusters Nbextensions

Select items to perform actions on them. [Upload] [New] [Refresh]

	Name	Last Modified	File size
<input type="checkbox"/>	..	seconds ago	
<input type="checkbox"/>	HW3_310712009.ipynb	2 days ago	19.6 kB
<input type="checkbox"/>	Untitled.ipynb	9 days ago	577 B

Other approach

另外，也可以使用 “Submit Job” 來執行程式，先把程式上傳到值區，點選 Submit Job 後選擇 Job Type (Pyspark, Python) 檔案。

Note: 路徑為 gs://程式位置/程式名稱

51 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.

My First Project

Search
Products, resources, docs (/)

Cluster details
SUBMIT JOB
REFRESH
START
STOP
DELETE
VIEW LOGS

For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See performance.

Name	cluster-1
Cluster UUID	8a20c7f9-813a-4472-aa3b-40203f017a8e
Type	Dataproc Cluster
Status	Running

MONITORING
JOBS
VM INSTANCES
CONFIGURATION
WEB INTERFACES

Filter
Filter instances

	Name ↑	Role
✓	cluster-1-m	Master

Submit a job

Job ID *
job-ca4ffbfb

Job type *
PySpark

Main python file *
gs://yellow_tripdata_2018_10/HW3_310712009.py

Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix

Additional python files

Jar files

Jar files are included in the CLASSPATH. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Archive files

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

Step1: 讀取已上傳到 cloud storage 的檔案

```
path0 = 'gs://yellow_tripdata_2018_10/Youvegottofindwhatyoulove.txt'
text1 = pd.read_table(path0, header = None)
```

Step2: 將 txt 檔中文章放入 list 中，先查看檔案內容來決定如何將句子做處理，依序分段切出來，並將空值移除，如下圖。

```
rdd = sc.parallelize(text1.loc[:,0].tolist())
lines = rdd.collect()
lines
```

```
['I am honored to be with you today at your commencement from one of the finest universities in the world. I never graduated from college. Truth be told, this is the closest I've ever gotten to a college graduation. Today I want to tell you three stories from my life. That's it. No big deal. I just three stories.',
'I dropped out of Reed College after the first 6 months, but then stayed around as a drop-in for another 18 months or so before I really quit. So why did I drop out?',
'It started before I was born. My biological mother was a young, unwed college graduate student, and she decided to put me up for adoption. She felt very strongly that I should be adopted by coll
```

```
lines1 = []
for l in lines:
    s1 = l.split('.')
    for i in range(len(s1)):
        lines1.append(s1[i])

lines1
```

```
lines_clean = []
for i in lines1:
    if i != '':
        lines_clean.append(i)

lines_clean
```

```
['I am honored to be with you today at your commencement from one of the finest universities in the world',
'I never graduated from college',
'Truth be told, this is the closest I've ever gotten to a college graduation',
'Today I want to tell you three stories from my life',
'That's it',
'No big deal',
'Just three stories',
'I dropped out of Reed College after the first 6 months, but then stayed around as a drop-in for another 18 months or so before I really quit',
'So why did I drop out?',
'It started before I was born',
'My biological mother was a young, unwed college graduate student, and she decided to put me up for adoption',
'She felt very strongly that I should be adopted by college graduates, so everything was all set for me to be adopted at birth by a lawyer and his wife',
'Except that when I popped out they decided at the last minute that they really wanted a girl',
'So my parents, who were on a waiting list, got a call in the middle of the night asking: "We have an unexpected baby boy; do you want him?" They said: "Of course"',
'My biological mother later found out that my mother had never graduated from college and that my father had never graduated from high school',
'She refused to sign the final adoption papers',
'She only relented a few months later when my parents promised that I would someday go to college',
'And 17 years later I did go to college']
```

Step3: 這次對於字串的處理，將標點符號用空格區分，將字串中的標點符號做清理。

Note: 標點符號相關的清理也可以使用 re 這個 library 來做正規化的處理。

```
# function for cleaning the data
def clean_data(data):
    data.replace("[^a-zA-Z]", " ", regex=True, inplace=True)
    return data
```

Step4: 最後來計算每個字出現的次數，及出線的頻率，由多到少 print 出結果(words、出現次數、出現頻率)，如下圖右側。

Answer:

```
# Create an empty dictionary
d = dict()

# Loop through each line of the file
for line in lines_1:
    # Remove the leading spaces and newline character
    line = line.strip()

    # Convert the characters in line to
    # lowercase to avoid case mismatch
    line = line.lower()

    # Split the line into words
    words = line.split(" ")

    # Iterate over each word in line
    for word in words:
        # Check if the word is already in dictionary
        if word in d:
            # Increment count of word by 1
            d[word] = d[word] + 1
        else:
            # Add the word to dictionary with count 1
            d[word] = 1
```

```
from collections import Counter
d1 = Counter(d)
d1.most_common()
for k, v in d1.most_common(30):
    print('%s: %i: %i' % (k, v, v/len(lines_1)))
```

```
the: 96: 0.67
i: 86: 0.60
to: 71: 0.49
and: 66: 0.46
it: 52: 0.36
was: 48: 0.33
a: 46: 0.32
of: 41: 0.28
that: 38: 0.26
in: 34: 0.24
you: 30: 0.21
my: 30: 0.21
is: 28: 0.19
had: 22: 0.15
with: 18: 0.12
out: 18: 0.12
for: 17: 0.12
so: 17: 0.12
have: 17: 0.12
your: 16: 0.11
all: 16: 0.11
as: 15: 0.10
me: 15: 0.10
on: 15: 0.10
what: 15: 0.10
but: 14: 0.10
be: 13: 0.09
from: 13: 0.09
college: 13: 0.09
life: 13: 0.09
```

Conclusion:

可以從 answer 中得知，需多較常出現的字多為都是介系詞或連接詞，如過要做進一步的 NLP processing 建議要 import nltk 中的 stopwords 可以過濾經常出現但不是那麼重要的字。

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

Step1: 設置 YARN cluster mode , Import Library

```
In [1]: import pandas as pd
        from pyspark.sql import SparkSession
        from pyspark import SparkConf, SparkContext

In [2]: # yarn cluster
        # spark = SparkSession.builder.master("local[*]").appName("HW").getOrCreate()
        spark = SparkSession.builder.master("yarn").appName("user").getOrCreate()
        sparkcontext = spark.sparkContext

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/04/29 04:30:17 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/04/29 04:30:17 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/04/29 04:30:17 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/04/29 04:30:17 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
```

Step2: 讀取 cloud storage 中的檔案，並取所需要的藍未來做後續處理。

```
: path1 = 'gs://yellow_tripdata_2018_10/yellow_tripdata_2018-10.csv'

# data
data_1 = spark.read.csv(path1, header=True, inferSchema=True)

: import pyspark.sql.functions as f

data_q2 = data_1.select("total_amount", "passenger_count", "payment_type")
```

Step3: 計算兩中不同付款方式 credit card(i = 1) 和 cash(i = 2) 下，Passenger_count 中各人數區間 (1~4 人)的 total amount 之平均值分別計算出來。

```
passenger_count = [1,2,3,4]
Credit_card_agv = []
Cash_agv = []

for i in range(1,3):
    for j in range(1, 5):
        d1 = data_q2.filter((data_q2.payment_type == i) & (data_q2.passenger_count == j)).agg({'total_amount': 'avg'}).collect()[0][1]
        if i == 1:
            Credit_card_agv.append(d1)
        else:
            Cash_agv.append(d1)

print(Credit_card_agv)
print(Cash_agv)
```

[Stage 23:=====> (5 + 1) / 6]

[18.095616294838994, 18.82315922602104, 18.397956039882864, 18.679751424181198]
[13.682682121839113, 14.729072956250409, 14.810038897804315, 15.506981138430348]

Step4: 將計算出來的資料作整理放入 dataframe 中展示出來，即可清楚看到不同人數的 Passenger_count 以及所對應的 total amount 之平均值。

```
# specify column names
columns = ['passenger_count', 'Credit_card_agv', 'Cash_agv']

# creating a dataframe by zipping the two lists
dataframe = spark.createDataFrame(zip(passenger_count, Credit_card_agv, Cash_agv), columns)

# show data frame
dataframe.show()
```

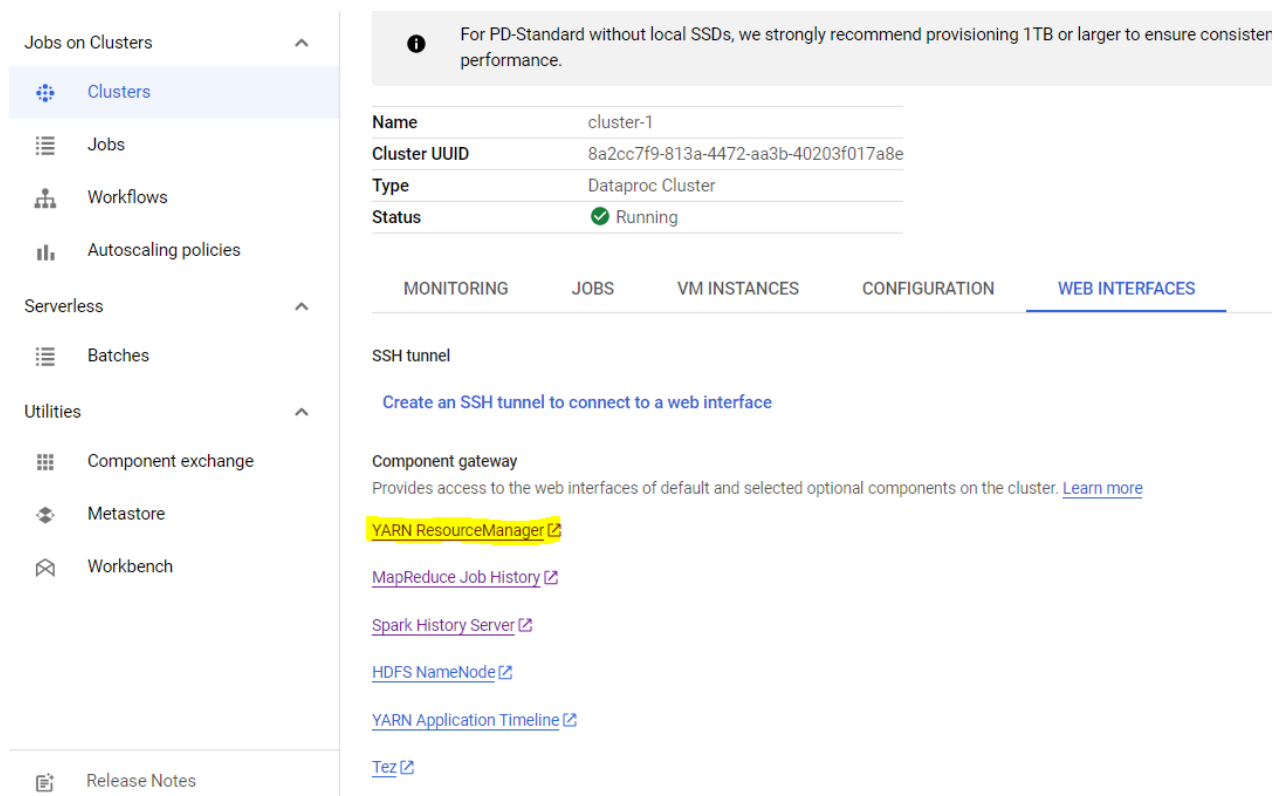
[Stage 27:=====>

(1 + 2) / 3]

	passenger_count	Credit_card_agv	Cash_agv
1	18.09561629483899	13.682682121839113	
2	18.823159226021037	14.729072956250409	
3	18.397956039882864	14.810038897804311	
4	18.679751424181198	15.506981138430348	

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. (Read through this guide to finish the question). Anything else worth mentioning (e.g. other valuable observations, or difficulties encountered in this work and how you resolve them).

Step1: 程式執行完之後，回到 Dataproc 的 cluster 中的 Webinterfaces，並點選“YARN ResourceManager”



Jobs on Clusters

- Clusters
- Jobs
- Workflows
- Autoscaling policies

Serverless

- Batches

Utilities

- Component exchange
- Metastore
- Workbench

Release Notes

For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistent performance.

Name: cluster-1

Cluster UUID: 8a2cc7f9-813a-4472-aa3b-40203f017a8e

Type: Dataproc Cluster

Status: ✔ Running

MONITORING JOBS VM INSTANCES CONFIGURATION **WEB INTERFACES**

SSH tunnel

[Create an SSH tunnel to connect to a web interface](#)

Component gateway

Provides access to the web interfaces of default and selected optional components on the cluster. [Learn more](#)

YARN ResourceManager

[MapReduce Job History](#)

[Spark History Server](#)

[HDFS NameNode](#)

[YARN Application Timeline](#)

[Tez](#)

Step2: 這裡有你執行的所有紀錄，可以點選 History 了解更多詳細內容。

loop

All Applications

Cluster Metrics

7

Apps Submitted

0

Apps Pending

1

Apps Running

6

Apps Completed

1

Containers Running

Used Resources

<memory:3072, vCores:1>

Total Resources

<memory:12288, vCores:4>

Reserved Resources

<memory:0, vCores:0>

Physical Mem Used %

32

Physical VCores Used %

0

Cluster Nodes Metrics

2

Active Nodes

0

Decommissioning Nodes

0

Decommissioned Nodes

0

Lost Nodes

0

Unhealthy Nodes

0

Rebooted Nodes

0

Shutdown Nodes

0

Scheduler Metrics

Scheduler Type

Capacity Scheduler

Scheduling Resource Type

[memory-mb (unit=M), vcores]

Minimum Allocation

<memory:1, vCores:1>

Maximum Allocation

<memory:6144, vCores:2>

Maximum Cluster Application Priority

0

Show 20 entries

Search

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1651134968653_0001	root	user	SPARK	default	0	Thu Apr 28 21:23:38 +0800 2022	N/A	Thu Apr 28 22:08:58 +0800 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	<div></div>	History	0
application_1651134968653_0004	root	user	SPARK	default	0	Thu Apr 28 23:07:02 +0800	N/A	Fri Apr 29 11:50:13 +0800 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0	<div></div>	History	0

Step3: 這裡會進到 Spark History Service(也可以直接點選 Web Interfaces 項下的 Spark History Service 進入)，找到你對應的 App id 可以查看各項執行的數據。

Event log directory: gs://dataproc-temp-asia-east1-969376720804-lazhx3mm/8a2cc7f9-813a-4472-aa3b-40203f017a8e/spark-job-history

Last updated: 2022-04-29 14:08:58

Client local time zone: Etc/GMT-8

Search:

Version	App ID	App Name	Driver Host	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.1.2	application_1651134968653_0004	user	cluster-1-w-1.asia-east1-b.c.silent-matter-347702.internal	2022-04-28 23:06:57	2022-04-29 11:50:13	12.7 h	root	2022-04-29 11:50:14	Download
3.1.2	application_1651134968653_0001	user	cluster-1-w-0.asia-east1-b.c.silent-matter-347702.internal	2022-04-28 21:23:32	2022-04-28 22:08:58	45 min	root	2022-04-28 22:08:59	Download
3.1.2	local-1650970567472	HW	cluster-1-m.asia-east1-b.c.silent-matter-347702.internal	2022-04-26 18:56:05	2022-04-26 19:02:43	6.6 min	root	2022-04-26 19:02:44	Download

Step4: 你可以藉由 Event Timeline 中圖表了解到你哪個階段、什麼時間點、做了什麼事情。另外，你也可以從 Complete Jobs 中了解到每個步驟所時間。

[silent-matter-347702](#) > [cluster-1](#)

[Sign out](#)

Spark Jobs (?)

User: root

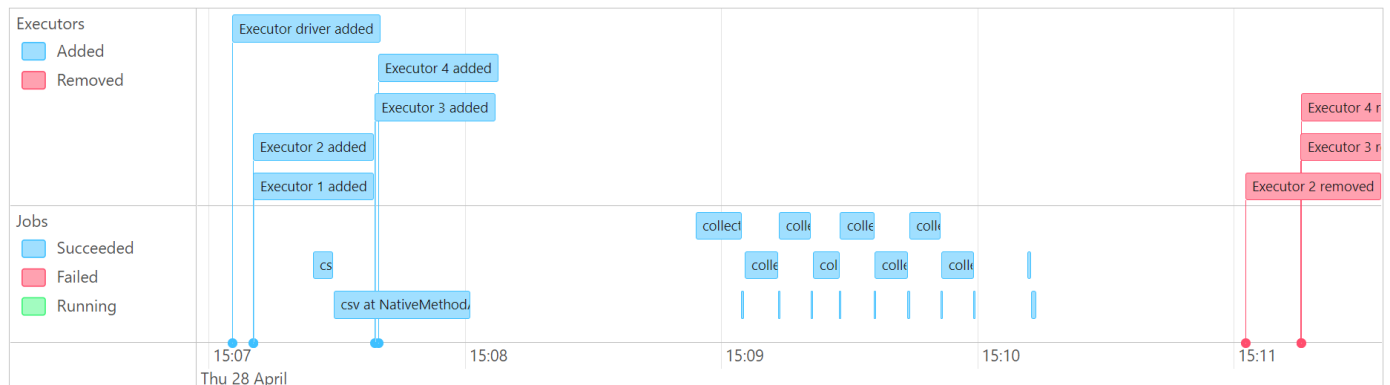
Total Uptime: 12.7 h

Scheduling Mode: FAIR

Completed Jobs: 20

▼ Event Timeline

☐ Enable zooming



[silent-matter-347702](#) > [cluster-1](#)

[Sign out](#)

Spark Jobs (?)

User: root

Total Uptime: 12.7 h

Scheduling Mode: FAIR

Completed Jobs: 20

► Event Timeline

▼ Completed Jobs (20)

Page:

1 Pages. Jump to . Show items in a page. [Go](#)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
19	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2022/04/28 15:10:12	1 s	1/1	3/3
18	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2022/04/28 15:10:11	0.8 s	1/1	1/1
17	collect at /tmp/ipykernel_11492/3342397675.py:7 collect at /tmp/ipykernel_11492/3342397675.py:7	2022/04/28 15:09:58	46 ms	1/1 (1 skipped)	1/1 (6 skipped)

Other valuable observations, or difficulties encountered in this work and how you resolve them

- (1) 你可以點 Description 來查看更細部的訊息，這個裡面除了剛剛上面說的 Event Timeline 與 Complete Jobs 外，還有“DAG Visualization”，google 用圖形化的方式呈現執行步驟讓你更加清楚與了解 Spark 下的運作。

Details for Job 19

Status: SUCCEEDED

Submitted: 2022/04/29 07:47:55

Duration: 0.9 s

Associated SQL Query: 9

Completed Stages: 1

▶ Event Timeline

▶ DAG Visualization

Completed Stages (1)

Page: 1

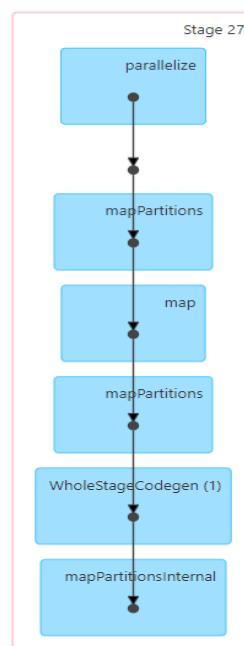
1 Pages. Jump to 1

Stage Id ▾	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Ir
27	default	showString at NativeMethodAccessorImpl.java:0 +details	2022/04/29 07:47:55	0.9 s	3/3	

Page: 1

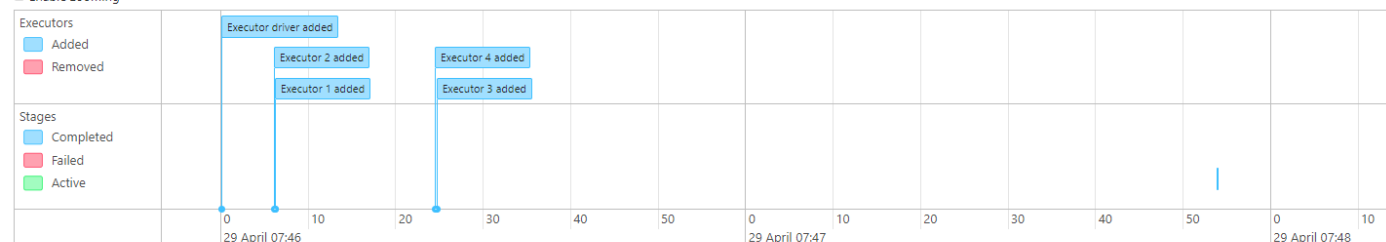
1 Pages. Jump to 1

▼ DAG Visualization

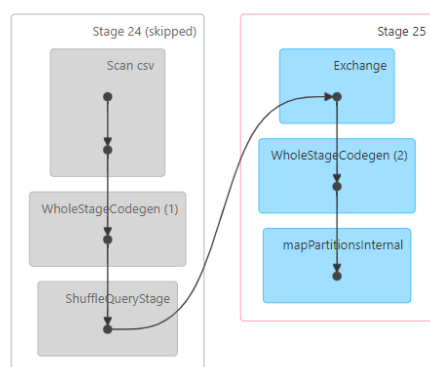


silent-matter-347702 > cluster-1

Enable zooming



▼ DAG Visualization



Completed Stages (1)

Page: 1

Stage Id ▾	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Ir
25	default	collect at /tmp/ipykernel_23389/3342397675.py:7 +details	2022/04/29 07:47:53	28 ms	1/1	


- (2) 藉由時間消耗可以知道 showString 所使用的時間較 collect 多出許多。

- ```
dataframe.show()
```
- |   | passenger_count   | Credit card_agv   | Cash_agv |
|---|-------------------|-------------------|----------|
| 1 | 16.8150032579668  | 16.8150032579668  |          |
| 2 | 17.55266010762928 | 17.55266010762928 |          |
| 3 | 17.23603242470929 | 17.23603242470929 |          |
| 4 | 17.52751244328926 | 17.52751244328926 |          |

```
dataframe.show()
```

| passenger_count | Credit_card_agv    | Cash_agv           |
|-----------------|--------------------|--------------------|
| 1               | 18.09561629483899  | 13.682682121839113 |
| 2               | 18.82315922602104  | 14.72907295625041  |
| 3               | 18.397956039882864 | 14.810038897804311 |
| 4               | 18.679751424181198 | 15.506981138430348 |

- silent-matter-347702 > cluster-1



## FINISHED Applications

+ Cluster

---

+ Tools

  - [Configuration](#)
  - [Local logs](#)
  - [Server stacks](#)
  - [Server metrics](#)

**Cluster Metrics**

|                |              |              |                |                    |                         |                          |
|----------------|--------------|--------------|----------------|--------------------|-------------------------|--------------------------|
| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Used Resources          | Total Resources          |
| 8              | 0            | 1            | 7              | 1                  | <memory:3072, vCores:1> | <memory:12288, vCores:4> |

**Cluster Nodes Metrics**

|              |                       |                      |            |                 |
|--------------|-----------------------|----------------------|------------|-----------------|
| Active Nodes | Decommissioning Nodes | Decommissioned Nodes | Lost Nodes | Unhealthy Nodes |
| 2            | 0                     | 0                    | 0          | 0               |

**Scheduler Metrics**

| Scheduler Type     | Scheduling Resource Type     | Minimum Allocation   | Maximum Allocation      |
|--------------------|------------------------------|----------------------|-------------------------|
| Capacity Scheduler | [memory-mb (unit=M), vcores] | <memory:1, vCores:1> | <memory:6144, vCores:2> |

Show 20 ▾ entries

| ID                                             | User | Name | Application Type | Queue   | Application Priority | StartTime                      | LaunchTime | FinishTime                     | State    | FinalStatus | Running Containers | Allocated CPU V-Cores | Allocated Memory MB | Allocated GPUs |
|------------------------------------------------|------|------|------------------|---------|----------------------|--------------------------------|------------|--------------------------------|----------|-------------|--------------------|-----------------------|---------------------|----------------|
| <a href="#">application_1651134968653_0007</a> | root | user | SPARK            | default | 0                    | Fri Apr 29 12:30:22 +0800 2022 | N/A        | Fri Apr 29 15:45:46 +0800 2022 | FINISHED | SUCCEEDED   | N/A                | N/A                   | N/A                 | N/A            |
| <a href="#">application_1651134968653_0004</a> | root | user | SPARK            | default | 0                    | Thu Apr 28 23:07:02 +0800 2022 | N/A        | Fri Apr 29 11:50:13 +0800 2022 | FINISHED | SUCCEEDED   | N/A                | N/A                   | N/A                 | N/A            |
| <a href="#">application_1651134968653_0001</a> | root | user | SPARK            | default | 0                    | Thu Apr 28 21:23:38 +0800 2022 | N/A        | Thu Apr 28 22:08:58 +0800 2022 | FINISHED | SUCCEEDED   | N/A                | N/A                   | N/A                 | N/A            |

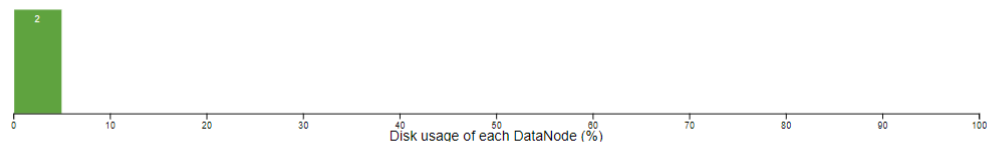
Showing 1 to 3 of 3 entries

```
{
 "beans" : [{
 "name" : "Hadoop:service=ResourceManager,name=RMNMInfo",
 "modelerType" : "org.apache.hadoop.yarn.server.resourcemanager.RMNMInfo",
 "LiveNodeManagers" : [{"\HostName\":"cluster-1-w-0.asia-east1-b.c.silent-matter-347702.internal","\Rack\":"b.c.silent-matter-347702.internal:8042","\LastHealthUpdate\":"1651223347567","\HealthReport\":"","\NodeManagerVersion\":"3.2.2","\Rack\":"default-rack","\State\":"RUNNING","\NodeId\":"cluster-1-w-1.asia-east1-b.c.silent-347702.internal:8042","\LastHealthUpdate\":"1651223346852","\HealthReport\":"","\NodeManagerVersion\":"3.2.2","\N"}, {
 "name" : "Hadoop:service=ResourceManager,name=RpcActivityForPort8033",
 "modelerType" : "RpcActivityForPort8033",
 "tag.port" : "8033",
 "tag.serverName" : "ResourceManagerAdministrationProtocolService",
 "tag.Context" : "rpc",
 "tag.NumOpenConnectionsPerUser" : "{}",
 "tag.Hostname" : "cluster-1-m",
 "ReceivedBytes" : 0,
 "SentBytes" : 0,
 "RpcQueueTimeNumOps" : 0,
 "RpcQueueTimeAvgTime" : 0.0,
 "RpcLockWaitTimeNumOps" : 0,
 "RpcLockWaitTimeAvgTime" : 0.0,
 "RpcProcessingTimeNumOps" : 0,
 "RpcProcessingTimeAvgTime" : 0.0,

```

(5) Interfaces 中的 HDFS NameNode 可以查看 Node 與 HDFS metric 的資訊。

Datanode usage histogram



In operation

Show  entries

Search:

| Node                                                                                | Http Address                                                           | Last contact | Last Block Report | Capacity  | Blocks | Block pool used | Version |
|-------------------------------------------------------------------------------------|------------------------------------------------------------------------|--------------|-------------------|-----------|--------|-----------------|---------|
| ✓ cluster-1-w-0.asia-east1-b.c.silent-matter-347702.internal:9866 (10.140.0.3:9866) | http://cluster-1-w-0.asia-east1-b.c.silent-matter-347702.internal:9864 | 1s           | 152m              | 245.89 GB | 15     | 5.86 MB (0%)    | 3.2.2   |
| ✓ cluster-1-w-1.asia-east1-b.c.silent-matter-347702.internal:9866 (10.140.0.2:9866) | http://cluster-1-w-1.asia-east1-b.c.silent-matter-347702.internal:9864 | 1s           | 40m               | 245.89 GB | 15     | 5.86 MB (0%)    | 3.2.2   |


Showing 1 to 2 of 2 entries

Previous **1** Next

```
{
 "beans" : [{
 "name" : "Hadoop:service=NameNode,name=JvmMetrics",
 "modelerType" : "JvmMetrics",
 "tag.Context" : "jvm",
 "tag.ProcessName" : "NameNode",
 "tag.SessionId" : null,
 "tag.Hostname" : "cluster-1-m",
 "MemNonHeapUsedM" : 70.65913,
 "MemNonHeapCommittedM" : 72.25,
 "MemNonHeapMaxM" : -1.0,
 "MemHeapUsedM" : 45.64061,
 "MemHeapCommittedM" : 114.125,
 "MemHeapMaxM" : 1473.375,
 "MemMaxM" : 1473.375,
 "GcCountParNew" : 47,
 "GcTimeMillisParNew" : 1181,
 "GcCountConcurrentMarkSweep" : 2,
 "GcTimeMillisConcurrentMarkSweep" : 142,
 "GcCount" : 49,
 "GcTimeMillis" : 1323,
 "GcNumWarnThresholdExceeded" : 0,
 "GcNumInfoThresholdExceeded" : 4,
 "GcTotalExtraSleepTime" : 11399,
 "ThreadsNew" : 0,
 "ThreadsRunnable" : 13,
 "ThreadsBlocked" : 0,
 "ThreadsWaiting" : 7,
 "ThreadsTimedWaiting" : 55,
 "ThreadsTerminated" : 0,
 "LogFatal" : 0,
 "LogError" : 0,
 "LogWarn" : 2,
 "LogInfo" : 519
 }, {
 "name" : "Hadoop:service=NameNode,name=RpcActivityForPort8051"

```

(6) 此外你也可以在 Interfaces 中查看 YARN Application Timeline，了解 Applications 的完成狀況。



FINISHED Applications

Logged in as: anonymous

Application History

About Applications  
FINISHED  
FAILED  
KILLED

Tools

Show 20 entries

| ID                             | User | Name | Application Type | Queue   | Application Priority | StartTime                      | LaunchTime | FinishTime                     | State    | FinalStatus | Progress    | Tracking UI |
|--------------------------------|------|------|------------------|---------|----------------------|--------------------------------|------------|--------------------------------|----------|-------------|-------------|-------------|
| application_1551134968653_0007 | root | user | SPARK            | default | 0                    | Fri Apr 29 12:30:22 +0800 2022 | N/A        | Fri Apr 29 15:45:46 +0800 2022 | FINISHED | SUCCEEDED   | <div></div> | History     |
| application_1551134968653_0004 | root | user | SPARK            | default | 0                    | Thu Apr 28 23:07:02 +0800 2022 | N/A        | Fri Apr 29 11:50:13 +0800 2022 | FINISHED | SUCCEEDED   | <div></div> | History     |
| application_1551134968653_0001 | root | user | SPARK            | default | 0                    | Thu Apr 28 21:23:38 +0800 2022 | N/A        | Thu Apr 28 22:08:58 +0800 2022 | FINISHED | SUCCEEDED   | <div></div> | History     |

Showing 1 to 3 of 3 entries

First Previous 1 Next Last