Parking Lots Real-time Information

R05942103 王以彥

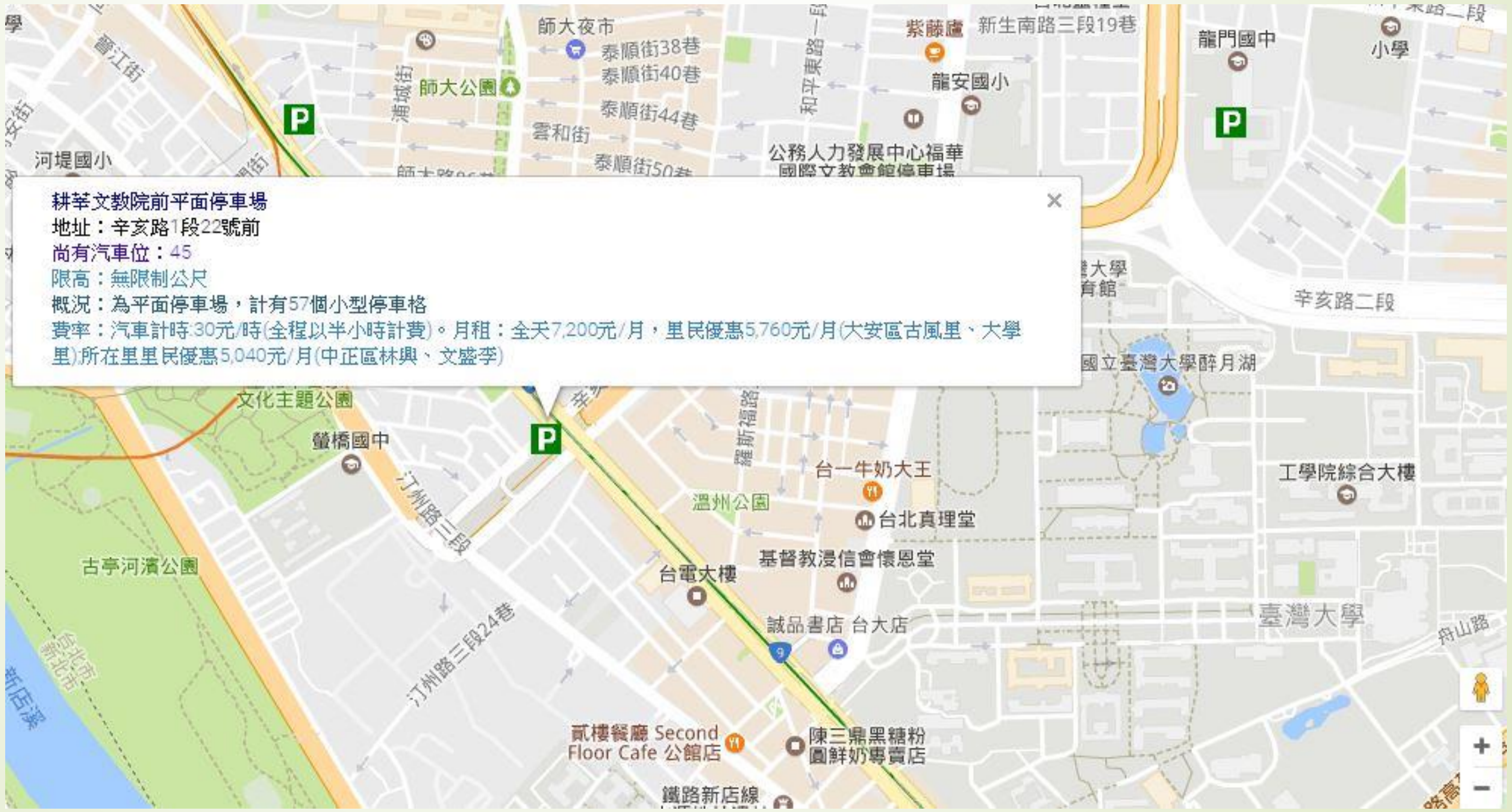
R05942080 鄭理文

Outline

- Final project proposal
- System architecture
- Mongo-spark connector
- Demo

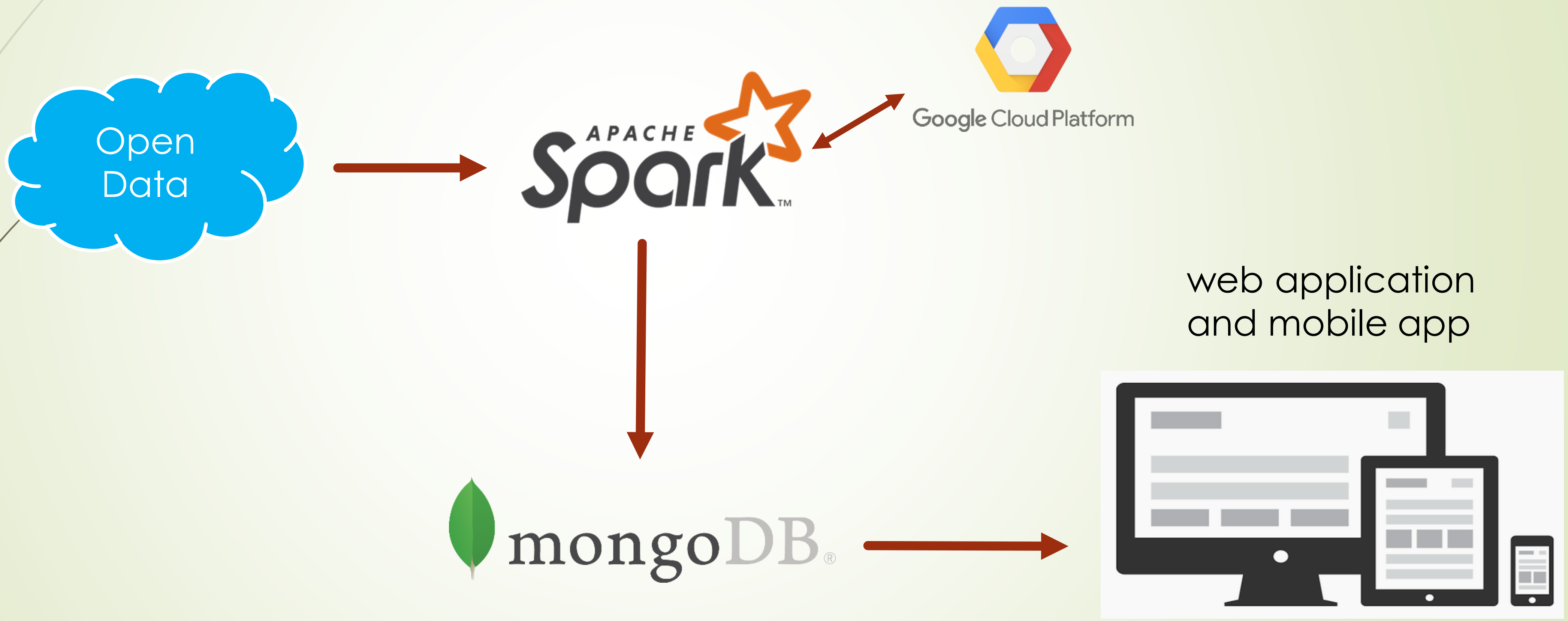
Proposal

- Using real-time parking lots information and combine with google map



耕莘文教院前平面停車場
地址：辛亥路1段22號前
尚有汽車位：45
限高：無限制公尺
概況：為平面停車場，計有57個小型停車格
費率：汽車計時:30元/時(全程以半小時計費)。月租：全天7,200元/月，里民優惠5,760元/月(大安區古風里、大學里);所在里里民優惠5,040元/月(中正區林興、文盛李)

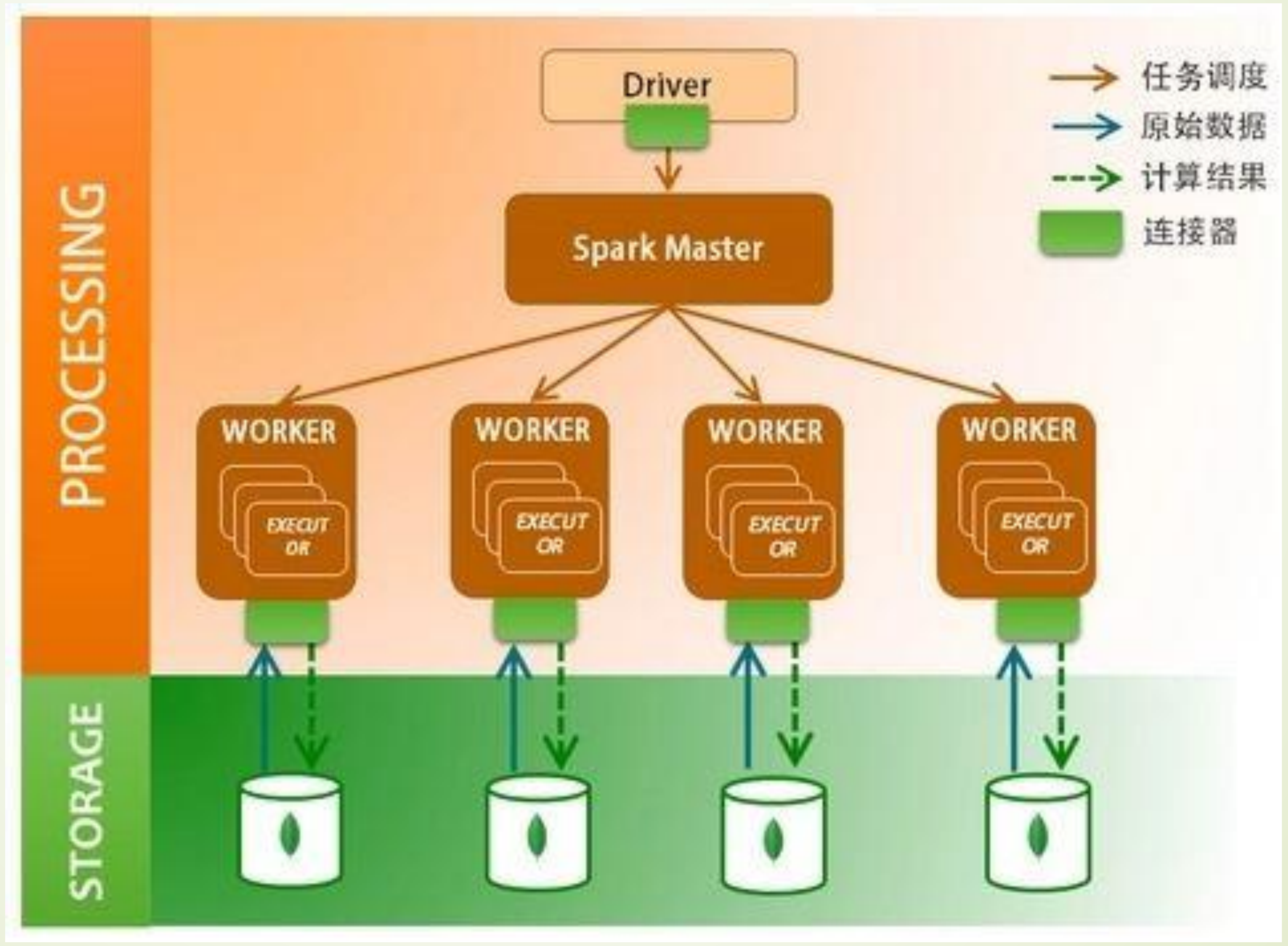
System Architecture



Open Data Resource

- Real-time information : http://data.gov.tw/node/26701
- Parking lots location information : http://data.gov.tw/node/26653
- Using python requests to get information
- Using linux crontab to update information every 3~5 minutes

Mongo-Spark Connector



Mongo-Spark Connector

- Official mongo-spark connector :

  https://docs.mongodb.com/spark-connector/master/
- Stratio spark-mongo connector :

  https://github.com/Stratio/Spark-MongoDB
- Support for Scala、Java、R、Python
- Provides integration between MongoDB and Apache Spark
- In this project, we use python version

Mongo-Spark Connector

- What can Mongo-Spark connector do
  - Support both read and write between from Spark to MongoDB
  - Support condition push

    search condition in Spark will be pushed to MongoDB, and execute in MongoDB server
  - Support building Spark and MongoDB on each worker node

MongoDB server address

- spark.mongodb.input.uri : MongoDB server address to read data

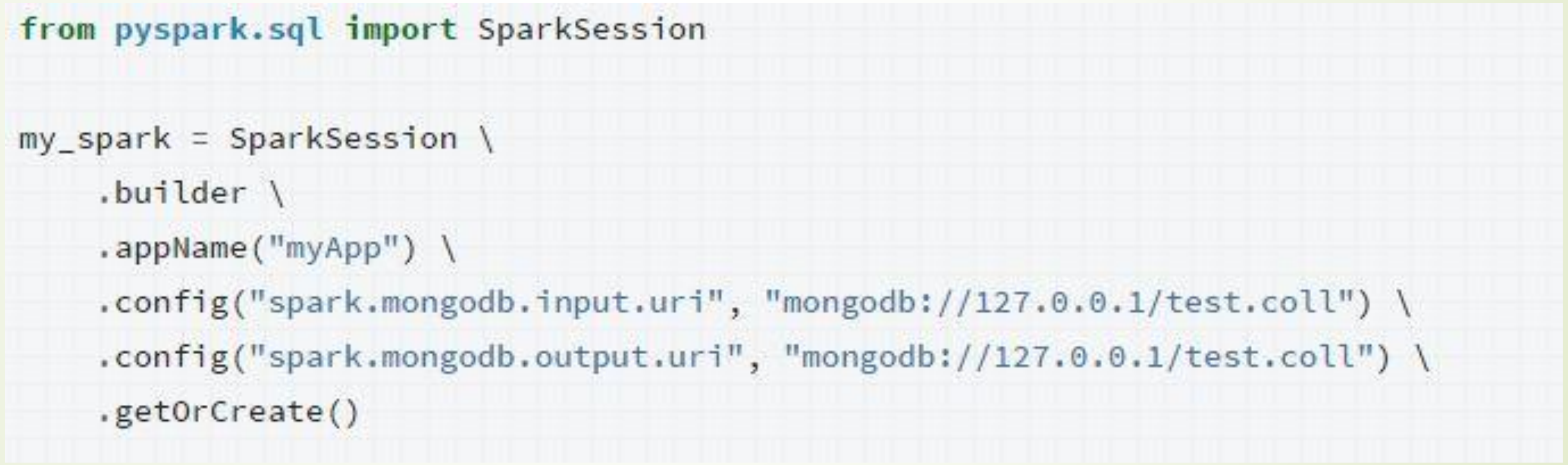
--conf "spark.mongodb.input.uri=mongodb://127.0.0.1/test.myCollection?readPreference=primaryPreferred“

- spark.mongodb.output.uri : MongoDB server address to write data

--conf "spark.mongodb.output.uri=mongodb://127.0.0.1/test.myCollection"

PySpark SparkSession

- Can also set configuration options in pyspark SparkSession
- Use SparkSession object to write/read data to MongoDB, create DataFrames, and perform SQL operations



```
from pyspark.sql import SparkSession

my_spark = SparkSession \
    .builder \
    .appName("myApp") \
    .config("spark.mongodb.input.uri", "mongodb://127.0.0.1/test.coll") \
    .config("spark.mongodb.output.uri", "mongodb://127.0.0.1/test.coll") \
    .getOrCreate()
```

Write/Read Data to MongoDB

- Write

```
people.write.format("com.mongodb.spark.sql.DefaultSource").mode("append"
). save()
```

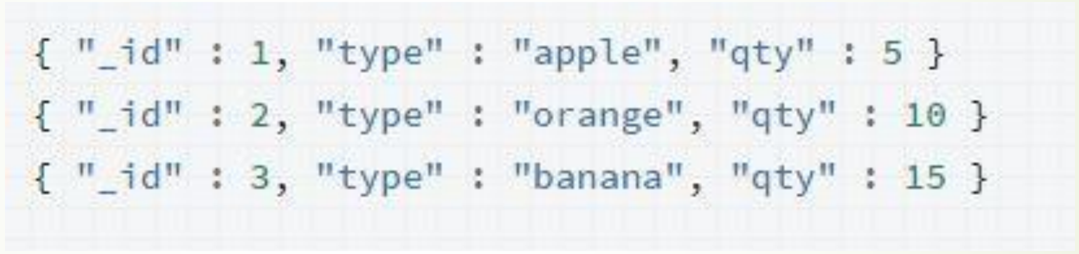
  - Mode : append 、 overwrite

- Read

```
df = spark.read.format("com.mongodb.spark.sql.DefaultSource").load()
```

Aggregation Pipeline

- Apply filtering rules and perform aggregation operations when reading data from MongoDB into Spark



```
{ "_id" : 1, "type" : "apple", "qty" : 5 }
{ "_id" : 2, "type" : "orange", "qty" : 10 }
{ "_id" : 3, "type" : "banana", "qty" : 15 }
```

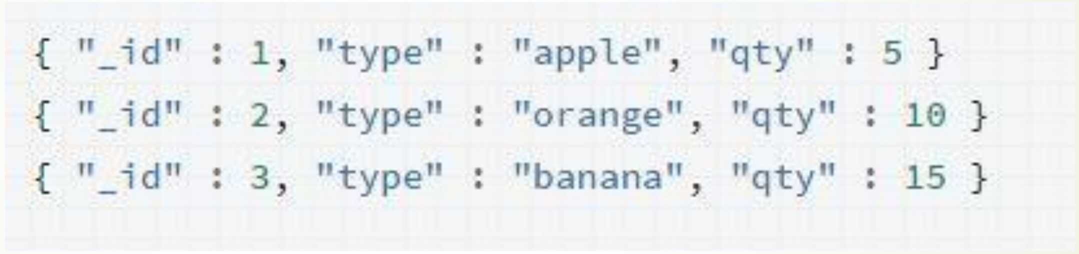
pipeline = "{'$match': {'type': 'apple'}}"

df=spark.read.format("com.mongodb.spark.sql.DefaultSource").option("pipeline", pipeline).load()

df.show()

Filtering

- Filtering with python dataframe function



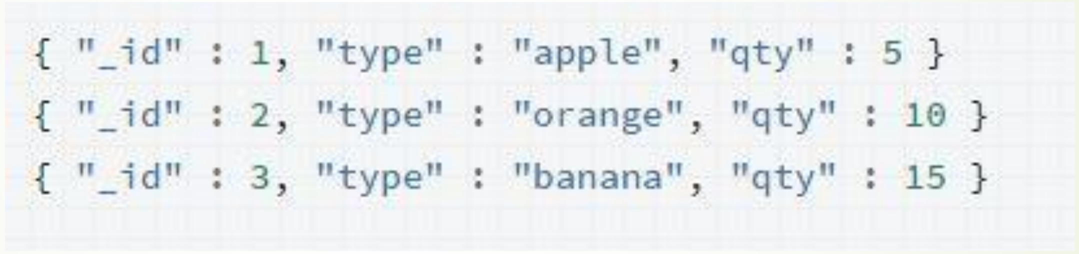
```
{ "_id" : 1, "type" : "apple", "qty" : 5 }
{ "_id" : 2, "type" : "orange", "qty" : 10 }
{ "_id" : 3, "type" : "banana", "qty" : 15 }
```

```
df = spark.read.format("com.mongodb.spark.sql.DefaultSource").load()
df.filter(df['qty'] >= 10).show()
```

- When using filters with DataFrames, the underlying Mongo Connector code constructs an aggregation pipeline to filter the data in MongoDB before sending it to Spark.

SQL

- Using SQL command to filter data



```
{ "_id" : 1, "type" : "apple", "qty" : 5 }
{ "_id" : 2, "type" : "orange", "qty" : 10 }
{ "_id" : 3, "type" : "banana", "qty" : 15 }
```

```
df.createOrReplaceTempView("temp")
some_fruit = spark.sql("SELECT type, qty FROM temp WHERE type LIKE '%e%'")
some_fruit.show()
```

DEMO