



Apache Spark and Zeppelin on Local

Cluster Kit Co.,Ltd.
www.clusterkit.co.th
May 23, 2018

สารบัญ

1. คุยกันก่อน.....	3
2. ซอฟต์แวร์ที่เกี่ยวข้อง.....	3
3. Apache Spark.....	3
3.1 ติดตั้ง Hadoop.....	3
4. Apache Zeppelin.....	5
5. Enable shiro for zeppelin authentication.....	7
6. อ้ออิง.....	8

1. คุยกันก่อน

เอกสารนี้เป็นคู่มือแนะนำการติดตั้งใช้งานซอฟต์แวร์ Apache Spark ร่วมกับ Apache Zeppelin แบบทำงานเครื่องเดียว (ใน spark เรียกว่าแบบ local) การติดตั้งแบบนี้จะใช้ทรัพยากรน้อย เหมาะกับงานทดลองศึกษาในบ้านเรา เพราะหลายงานไม่ได้มีข้อมูลขนาดใหญ่ถึงกับต้องใช้ Hadoop ฉะนั้นการที่เราใช้งาน Spark แบบ local นี้ ก็จะเป็นประโยชน์สำหรับผู้ที่ใช้งาน Spark ไม่ว่าจะเป็นการใช้กราฟอัลกอริทึม หรือ อัลกอริทึม Machine Learning และเมื่อติดตั้งใช้งานร่วมกับ Zeppelin จะทำให้สามารถใช้งาน Spark ผ่านหน้าเว็บ ทำให้สะดวกต่อการใช้งานมากขึ้น พวกเราชาวคลัสเตอร์คิดหวังว่าเอกสารนี้ จะเป็นประโยชน์กับท่านผู้สนใจตามสมควร เชิญทรรศนาและแบ่งปันกันตามสบาย

2. ซอฟต์แวร์ที่เกี่ยวข้อง

ณ วันที่ 17 เม.ย. 2561 Apache zeppelin รุ่นล่าสุดคือ 7.3 รองรับแค่ spark 2.1 (ซึ่งรุ่นล่าสุดคือ 2.3)

- Oracle JDK 1.8
- Apache Spark 2.1.2 (spark-2.1.2-bin-hadoop2.7.tgz) <https://spark.apache.org/downloads.html>
- Apache Zeppelin (zeppelin-0.7.3-bin-all.tgz) <https://zeppelin.apache.org/download.html>
- การทดลองตามเอกสารนี้ ได้ลองกับ Ubuntu Desktop 16.04.4 LTS และ CentOS-7

3. Apache Spark

Spark เป็นซอฟต์แวร์สำหรับประมวลผลข้อมูล ได้รับความนิยมมากกับงานประมวลผลข้อมูลขนาดใหญ่ (Big Data) ทำงานใน 3 โหมด หลัก ๆ

1. Local คือ การทำงานที่เครื่องเดียว อ่านและประมวลผลข้อมูลในเครื่องนั้น ๆ
2. Standalone คือ การทำงานเป็นคลัสเตอร์ใช้เครื่องหลายเครื่องช่วยกันประมวลผล ในลักษณะเป็นคลัสเตอร์สำหรับ spark โดยตรง ถ้าติดตั้งร่วมกับ Hadoop ก็จะมาอ่านไฟล์ข้อมูลจาก HDFS แต่ไม่ได้ทำงานร่วมกับ Yarn
3. Yarn Client คือ การทำงานร่วมกับตัวจัดการทรัพยากรและจัดลำดับงานของ Hadoop ที่ชื่อว่า Yarn ซึ่ง Yarn จะบริหารจัดการทรัพยากรที่มีอยู่ของระบบ เพื่อให้การใช้งานร่วมงานของทั้ง MapReduce / spark และซอฟต์แวร์ประมวลผลอื่น ๆ เช่น Impala, Hbase เป็นไปด้วยดี ไม่แย่งกันใช้ทรัพยากร

3.1 ติดตั้ง Spark

การติดตั้ง spark แบบ Local สำหรับใช้ในเครื่อง ๆ เดียวนั้นแทบจะไม่ต้องทำอะไร แค่แตก tar แล้วรันได้ก็ใช้ได้ มีวิธีการดังต่อไปนี้

ในตัวอย่างนี้เป็นแตกไฟล์ .tgz ไปยัง /opt (ท่านอาจเปลี่ยนเป็นตำแหน่งอื่นได้)

```
tar xvf spark-2.1.2-bin-hadoop2.7.tgz -C /opt
```

รันโปรแกรมตัวอย่าง คำนวณค่าพายเพื่อทดสอบการทำงานดังต่อไปนี้

```
/opt/spark-2.1.2-bin-hadoop2.7/bin/run-example SparkPi 10
```

จะได้คำตอบในลักษณะนี้ “Pi is roughly 3.14017514017514” แทรกอยู่ใน log (ประมาณบรรทัดที่ 10 นับจากท้าย)

3.2 ตัวอย่างการใช้งาน pyspark shell

1. เตรียมไฟล์ salary.csv มีเนื้อหาดังต่อไปนี้
2. รันคำสั่ง pyspark

```
01, john, 50000  
02, Jim, 25000  
03, Jame, 80000  
04, Jazz, 18000
```

```
/opt/spark-2.1.2-bin-hadoop2.7/bin/pyspark
```

3. รันโค้ดตัวอย่างใน pyspark shell (เมื่อรันเสร็จแล้ว กด Ctrl+D เพื่อออกจาก spark-shell)

```
raw_data = sc.textFile('salary.csv').cache()

from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

from pyspark.sql import Row
csv_data = raw_data.map(lambda l: l.split(","))
row_data = csv_data.map(lambda p: Row(id=p[0],name=p[1],salary=p[2]))

intdf = sqlContext.createDataFrame(row_data)
row_data.count()

intdf.registerTempTable("salary")
salary = sqlContext.sql("select * from salary")
salary.show()
salary.count()
intdf.filter(salary.salary > 25000).show()

salaryAvg = sqlContext.sql("select avg(salary) from salary")
salaryAvg.show()
```

3.3 Spark-Summit

ต่อไปนี้เป็นตัวอย่างการรันอัลกอริทึม Kmean กับชุดข้อมูลตัวอย่าง Iris ผ่านคำสั่ง spark-submit

1. ดาวน์โหลดไฟล์ kmean.py ตามลิงค์ต่อไปนี้

<https://raw.githubusercontent.com/apache/spark/branch-1.6/examples/src/main/python/mllib/kmeans.py>

2. ดาวน์โหลดไฟล์ข้อมูล Iris จากลิงค์ต่อไปนี้ <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>

3. สำหรับระบบปฏิบัติการลินุกซ์ที่ทดสอบต้องติดตั้ง python numpy เพิ่มเติม ต่อไปเป็นตัวอย่างสำหรับ CentOS / RHEL

```
yum install numpy
```

4. เตรียมข้อมูลสำหรับรัน

```
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
```

จากข้อมูลในไฟล์ iris.data นั้น มีคลาสอยู่ด้วย (คอลัมน์สุดท้าย) เราจะตัดคลาसออก (เพราะเราจะใช้เฉพาะข้อมูลกลีบเลี้ยงกลีบดอกในการแบ่งกลุ่ม) และบันทึกในชื่อไฟล์ iris.csv ด้วยคำสั่งต่อไปนี้

```
cut -f 1-4 -d , iris.data > iris.csv
```

5. ปรับแก้โค้ดโปรแกรม kmean.py บรรทัดที่ 33 จากการแบ่งข้อมูลด้วยช่องว่าง เป็นการแบ่งข้อมูลด้วยจุลภาค

```
return np.array([float(x) for x in line.split(' ')])
```

เป็น

```
return np.array([float(x) for x in line.split(',')])
```

6. รันโปรแกรม kmean.py โดยมีอินพุตเป็นไฟล์ iris.csv และกำหนดค่า k เป็น 3 คือแบ่งสามกลุ่ม ดังนี้

```
/opt/spark-2.1.2-bin-hadoop2.7/bin/spark-submit kmeans.py iris.csv 3
```

ผลลัพธ์จะออก 2 บรรทัด โดยปะปนมากับข้อความ Log ลักษณะดังต่อไปนี้

```
Final centers: [array([ 5.9016129 ,  2.7483871 ,  4.39354839,  1.43387097]),
                array([ 5.006,  3.418,  1.464,  0.244]),
                array([ 6.85      ,  3.07368421,  5.74210526,  2.07105263])]
```

และ

```
Total Cost: 78.9408414261
```

3.4 การเซต Path สำหรับ Spark

ในกรณีที่ต้องการกำหนด Path ให้กับโปรแกรมที่ติดตั้ง สามารถทำได้โดยรันคำสั่งต่อไปนี้

```
export SPARK_HOME=/opt/spark-2.1.2-bin-hadoop2.7
export PATH=$PATH:/$SPARK_HOME/bin
```

คำสั่ง export ข้างต้น สามารถนำไปใส่ใน .bash_profile หรือ .bashrc หรือ ใน module file ก็ได้ตามแต่ถนัด

3.5 สำหรับค่าเตือนเรื่องการ Set SPARK_LOCAL_IP

ถ้าเราไม่ได้กำหนดชื่อเครื่องจะมีค่าเตือนในลักษณะดังกล่าวออกมาเวลาเรียกใช้งาน spark สามารถแก้ไขได้โดยการกำหนดชื่อเครื่อง หรือ กำหนดค่าไอพีแอดเดรสที่จะให้เข้าใช้งาน spark ได้โดยกำหนดที่ไฟล์ conf/spark-env.sh ในลักษณะต่อไปนี้

```
SPARK_LOCAL_IP="<IP address>"
```

4. Apache Zeppelin

Zeppelin เป็นเว็บสำหรับใช้เขียนโปรแกรมที่ออกแบบสำหรับ Spark หรือจะเรียกว่า Spark Notebook พอเป็นโปรแกรมที่อยู่บนเว็บก็จะทำให้สามารถแสดงผลเป็นกราฟต่าง ๆ ได้ ใช้งานสะดวกมากยิ่งขึ้น

4.1 ติดตั้ง Zeppelin

1. แดกไฟล์

```
tar xvf zeppelin-0.7.3-bin-all.tgz -C /opt
```

2. คัดลอกไฟล์ /opt/zeppelin-0.7.3-bin-all/conf/zeppelin-env.sh.template เป็น zeppelin-env.sh

```
cd /opt/zeppelin-0.7.3-bin-all/conf/  
cp zeppelin-env.sh.template zeppelin-env.sh
```

3. กำหนดค่าในไฟล์ zeppelin-env.sh

แก้ไขบรรทัดที่ 52 จาก

```
#export SPARK_HOME # (required) When it is defined ...
```

เป็น

```
export SPARK_HOME=/opt/spark-2.1.2-bin-hadoop2.7
```

4. เริ่มการทำงาน Zeppelin

```
/opt/zeppelin-0.7.3-bin-all/bin/zeppelin-daemon.sh start
```

5. เปิดเว็บไปที่ <http://localhost:8080/>

4.2 รัน Kmean บน Zeppelin

เลือก New Notebook แล้วลองนำโค้ด kmean วางและรัน

1. ดาวน์โหลดโค้ดจากลิงค์ต่อไปนี้ <https://spark.apache.org/docs/latest/mllib-clustering.html#k-means> โดยปรับพารามิเตอร์ให้เหมาะกับชุดข้อมูล (ตำแหน่งไฟล์ iris.csv, การแบ่งข้อมูลด้วยจุลภาค (,) จำนวนกลุ่ม จำนวนรอบสูงสุด และเพิ่มสองบรรทัดสุดท้ายโดยประยุกต์จากไฟล์ kmean.py ก่อนหน้า รวมถึงตำแหน่งการบันทึกโมเดลที่ได้)

```
%pyspark

from numpy import array
from math import sqrt

from pyspark.mllib.clustering import KMeans, KMeansModel

# Load and parse the data
data = sc.textFile("/tmp/iris.csv")
parsedData = data.map(lambda line: array([float(x) for x in line.split(',')]))

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, 3, maxIterations=1000, initializationMode="random")

# Evaluate clustering by computing Within Set Sum of Squared Errors
def error(point):
    center = clusters.centers[clusters.predict(point)]
    return sqrt(sum([x**2 for x in (point - center)]))

WSSSE = parsedData.map(lambda point: error(point)).reduce(lambda x, y: x + y)
print("Within Set Sum of Squared Error = " + str(WSSSE))
print("Final centers: " + str(clusters.clusterCenters))
print("Total Cost: " + str(clusters.computeCost(parsedData)))

# Save and load model
clusters.save(sc, "/tmp/KMeansModel")
sameModel = KMeansModel.load(sc, "/tmp/KMeansModel")
```

2. ทดลองรันทำนายผล (prediction) ในขั้นนี้เป็นการหาว่าจุดข้อมูลอยู่ที่ใกล้จุดศูนย์กลางของกลุ่มไหนก็เป็นสมาชิกของกลุ่มนั้น

```
%pyspark

from pyspark.mllib.clustering import KMeans, KMeansModel

sameModel = KMeansModel.load(sc, "/tmp/KmeansModel")
print("Final centers: " + str(sameModel.clusterCenters))

print(sameModel.predict(array([6.9, 3.1, 5.4, 2.1])))
print(sameModel.predict(array([4.9, 3.1, 1.5, 0.1])))
```


5. Enable shiro for zeppelin authentication

จะสังเกตเห็นว่า Apache Zeppelin ที่ติดตั้งและใช้งานตามข้างต้นไม่มีหน้าให้ Login ใช้งาน ในขั้นตอนต่อไปนี้จะทำให้ zeppelin มีระบบตรวจสอบตัวตน

1. คัดลอกไฟล์ /opt/zeppelin-0.7.3-bin-all/conf/shiro.ini.template เป็น shiro.ini

```
cd /opt/zeppelin-0.7.3-bin-all/conf/  
cp shiro.ini.template shiro.ini
```

2. กำหนดค่าในไฟล์ shiro.ini โดยการแก้ไขไฟล์ โดยการใส่หมายเหตุ และนำหมายเหตุออก จาก

```
# anon means the access is anonymous.  
# authc means Form based Auth Security  
# To enforce security, comment the line below and uncomment the next one  
/api/version = anon  
#/api/interpreter/** = authc, roles[admin]  
#/api/configurations/** = authc, roles[admin]  
#/api/credential/** = authc, roles[admin]  
#/** = anon  
/** = authc
```

เป็น

```
# anon means the access is anonymous.  
# authc means Form based Auth Security  
# To enforce security, comment the line below and uncomment the next one  
/api/version = anon  
/api/interpreter/** = authc, roles[admin]  
/api/configurations/** = authc, roles[admin]  
/api/credential/** = authc, roles[admin]  
#/** = anon  
/** = authc
```

5.1 กำหนดบัญชีผู้ใช้ในการยืนยันตัวตน

ระบบจะใช้บัญชีผู้ใช้ที่อยู่ในส่วน [users] ในไฟล์ shiro.ini นี้ในการยืนยันตัวตน

```
[users]  
# List of users with their password allowed to access Zeppelin.  
# To use a different strategy (LDAP / Database / ...) check the shiro doc  
  at http://shiro.apache.org/configuration.html#Configuration-INISections  
admin = password1, admin  
user1 = password2, role1, role2  
user2 = password3, role3  
user3 = password4, role2
```

แต่ถ้าเราไม่ประสงค์จะใช้วิธีนี้ ยังสามารถยืนยันตัวตนกับระบบอื่น เช่น LDAP, AD ได้อีกด้วย รวมถึงการยืนยันตัวตนกับบัญชีผู้ใช้ที่อื่นๆที่สาธิตในขั้นถัด

5.2 กำหนดให้ยืนยันตัวตนกับบัญชีผู้ใช้ลินุกซ์

1. ให้แก้ไขไฟล์ shiro.ini เอาคอมเมนต์ บรรทัดที่ 49-50 ออก

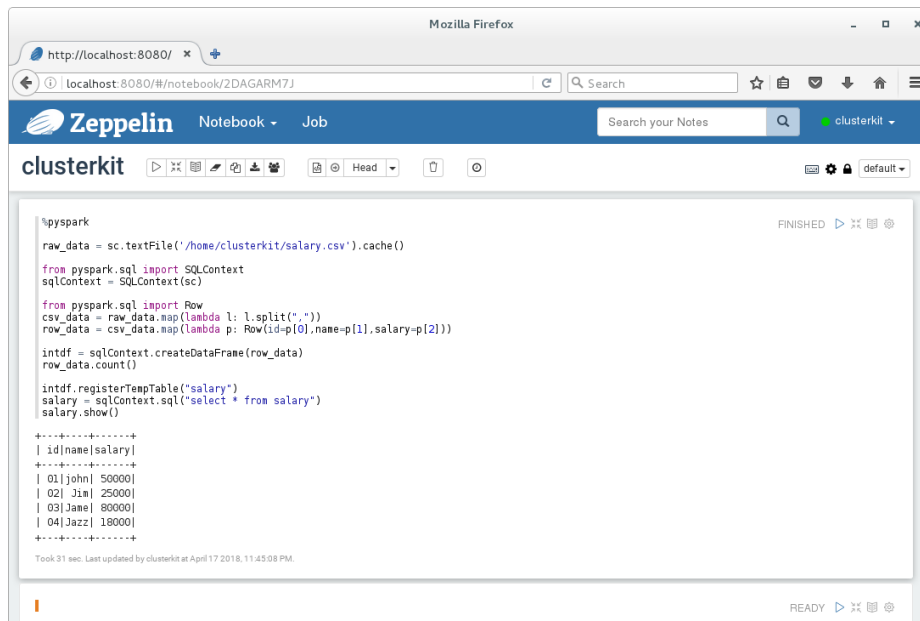
```
### A sample PAM configuration
pamRealm=org.apache.zepplin.realm.PamRealm
pamRealm.service=sshd
```

และถ้าจะไม่ใช้ผู้ใช้ในส่วน [users] ให้ไปใส่หมายเหตุไว้หน้าบรรทัดชื่อผู้ใช้ทุกรายการ

2. เริ่มการทำงาน zeppelin ใหม่

```
/opt/zeppelin-0.7.3-bin-all/bin/zeppelin-daemon.sh restart
```

ทดลองเข้าเว็บ <http://localhost:8080> อีกครั้ง ใช้ Login/password ของลินุกซ์



```
%pyspark
raw_data = sc.textFile('/home/clusterkit/salary.csv').cache()
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
from pyspark.sql import Row
csv_data = raw_data.map(lambda l: l.split(", "))
row_data = csv_data.map(lambda p: Row(id=p[0],name=p[1],salary=p[2]))
intdf = sqlContext.createDataFrame(row_data)
row_data.count()

intdf.registerTempTable("salary")
salary = sqlContext.sql("select * from salary")
salary.show()

+-----+-----+
| id|name|salary|
+-----+-----+
| 01|john| 50000|
| 02|jim| 25000|
| 03|jame| 80000|
| 04|jazz| 18000|
+-----+-----+

Took 31 sec. Last updated by clusterkit at April 17 2018, 11:45:08 PM.
```

นอกจากการยืนยันตัวตนกับบัญชีผู้ใช้ลินุกซ์ ก็ยังสามารถยืนยันกับบริการอื่นอย่าง LDAP, AD ได้โปรดอ่านเพิ่มเติมในเอกสารอ้างอิงลำดับที่ 2

6. อ้างอิง

- <https://dziganto.github.io/anaconda/shiro/spark/zeppelin/zeppelinhub/How-To-Locally-Install-Apache-Spark-And-Zeppelin/>
- Shiro <https://zeppelin.apache.org/docs/0.7.0/security/shiroauthentication.html>