**CS523 - Big Data Technology Project**

**Hoang Phuong Le - 986305**

**Quoc Nghiep Nguyen - 986375**

**Part 1: Spark Streaming**

#Spark installation  
 http://download.nextag.com/apache/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz  
  
tar -xvzf spark-2.3.0-bin-hadoop2.7.tgz  
mv spark-2.3.0-bin-hadoop2.7 /usr/local/spark  
  
  
#Kafka installation  
http://mirrors.sorengard.com/apache/kafka/1.0.1/kafka\_2.12-1.0.1.tgz  
tar -xvf kafka\_2.12-1.0.1.tgz  
  
edit .bash\_profile to add 2 line below  
  
export KAFKA\_HOME=<location of kafka>  
export PATH=$PATH:$KAFKA\_HOME/bin  
  
#Check zookeeper.properties, reconfigure if need --> --clientPort=2181  
#Check zookeeper.properties, dataDir=/tmp/zookeeper  
#start zookeeper first before start kafka  
./bin/zookeeper-server-start.sh config/zookeeper.properties

#start kafka, check broker.id=0 , port etc...  
./bin/kafka-server-start.sh config/server.properties  
  
#create toptic  
./bin/kafka-topics.sh --zookeeper localhost:2181 --create --topic wordcountopic --partitions 1 --replication-factor 1  
  
#list topic on kafka  
./bin/kafka-topics.sh --list --zookeeper --localhost:2181

#Connect to Zookeeper instance:

zookeeper-shell.sh localhost:2181  
ls /brokers/topics

#Remove the topic folder from ZooKeeper

rmr /brokers/topics/yourtopic

* Producer (simple )pulls SELL-BUY data from BITTREX.COM in Json format at <https://bittrex.com/api/v1.1/public/getmarkethistory?market=BTC-BCC> every 30 seconds, then stream what they have crawled to Broker 0 and ‘wordcountopic’ Topic.

* run producer with “simpleproducer” project in the zip file in Eclipse, it will pull data from bittrex every 30 seconds and send to spark, observe the console for what the producer has crawled.
* Consumer receives the result from the ‘wordcountopic’ channel, parse the Json then send the values HBaseDB class which opens the connection to HBase, create ‘tradedata\_hbase’ table and finally ‘put’ the records of TradeLine into HBase
* run consumer with “kafkatest” project in the zip file
* right click on pom.xml file chose Run As > Build Install
* change dir to target folder and start the consumer with

spark-submit --class "cs523.kafkatest.App" '/home/cloudera/Downloads/v0.1/kafkatest/target/kafkatest-0.0.1-SNAPSHOT-jar-with-dependencies.jar'

* observe the shell for what consumer received
* we can use Hbase shell to check if the table was created by using
* $ HBase shell
* $ list

#start consummer from beginning  
./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic wordcountopic --from-beginning

**Part 2: HIVE/ HBase thru SparkSQL**

#Hive on Hbase  
#done with consumer : kafkatest.java

#run this in hive shell to create mapping data  
create external table tradedata\_hbase (rowid String,price float,quantity float,ordertype String,timestmp String,total String,filltype String) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' with serdeproperties ("hbase.columns.mapping"=":key,data\_cf:Price,data\_cf:Quantity,data\_cf:Ordertype,data\_cf:Timestamp,data\_cf:Total,data\_cf:Filltype") tblproperties("hbase.table.name"="TradeData");

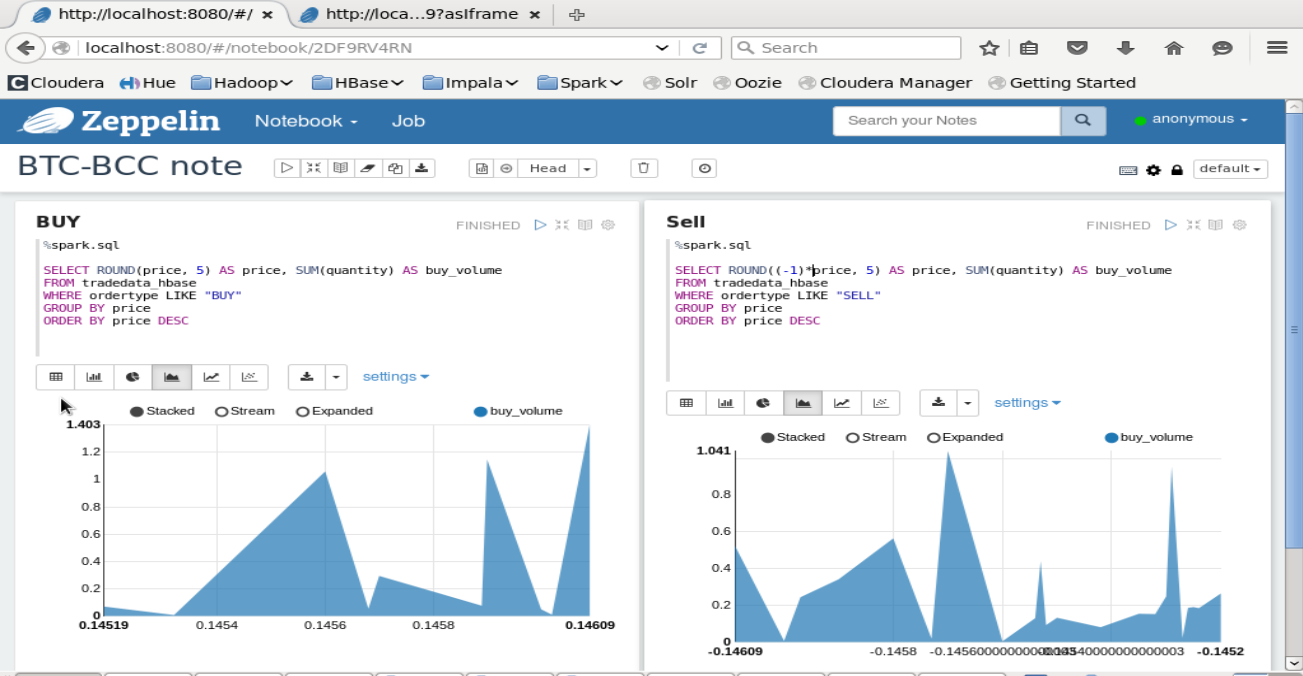
**Part 3: Data Visualization**

#install Zeppelin  
#download Zeppelin at http://www.apache.org/dyn/closer.cgi/zeppelin/zeppelin-0.7.3/zeppelin-0.7.3-bin-all.tgz

# unzip , change dir to bin  
./zeppelin.daemon.sh start  
./zeppelin.daemon.sh stop  
  
# edit permission for hive on zeppelin  
sudo chmod -R 777 /tmp/hive/  
sudo -u hdfs hadoop fs -chmod -R 777 /tmp/hive/

open localhost:8080 will display zeppeline

import chart BTC-BCC.json, run the query and observe the result



**BUY**

%spark.sql  
  
SELECT ROUND(price, 5) AS price, SUM(quantity) AS buy\_volume  
FROM tradedata\_hbase   
WHERE ordertype LIKE "BUY"  
GROUP BY price  
ORDER BY price DESC  
  
  
**SELL**

%spark.sql  
  
SELECT ROUND((-1)\*price, 5) AS price, SUM(quantity) AS buy\_volume  
FROM tradedata\_hbase   
WHERE ordertype LIKE "SELL"  
GROUP BY price  
ORDER BY price DESC