



# Big Data Technologies

Final project

Kim Bang Le- 986999




# Agenda

- Objectives
  - Project introduction
  - Architecture
  - Big data technologies used
  - Instructions
  - Demo
- 



# Project Objectives

- Apply big data technologies learned in this course into real project
  - Research new big data technologies
  - Solve a real problem using big data technologies
- 

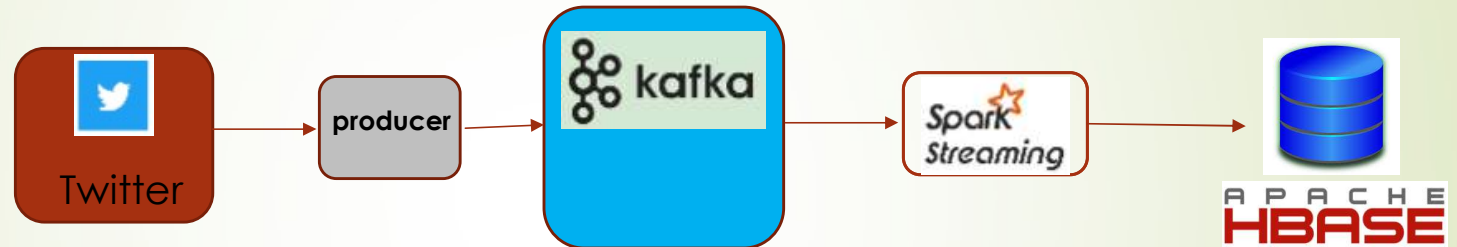


# Project introduction

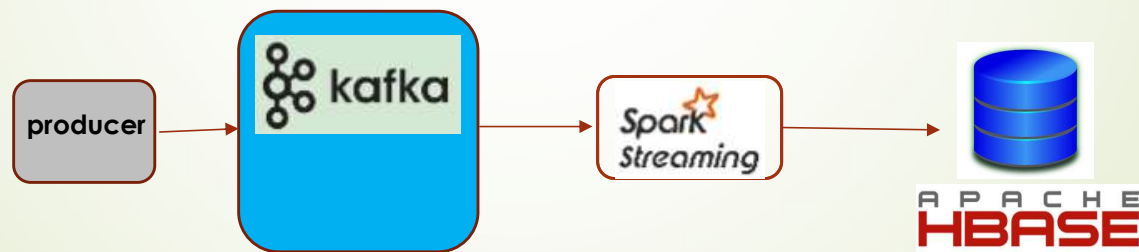
- Original/planned solved problem:
  - Extract data from twitter to see what US people are discussing/thinking about Trump and Biden
  - From the extracted data, there will be another step to analyze them and see Trump or Biden who will have more chance to win the 2020 election.
  - Unsolved issue: there is bug and cannot extract data from Twitter
- Problem is implemented
  - Generate/Read a stream of numbers and analyze/classify which one is even/odd number

# Architecture

- Original/planned solved problem



- Problem is implemented/solved





# Architecture (cont.)

There are 2 Java based projects

➤ Producer project


- Generate/extract data from source and output to Kafka
- Play the role as Kafka producer

➤ Consumer project

- Read data from Kafka using Spark streamming
- Analyze data
- Output to Hbase
- Play the role as Kafka consumer



# Big Data Technologies used

- Spark Streamming
  - Hbase
  - Kafka
  - Others: Twitter API
- 

# Instructions

## ■ Start Kafka

```
KAFKA_HOME=/home/cloudera/kafka_2.12-2.6.0; export KAFKA_HOME  
//Start Zookeeper server  
$KAFKA_HOME/bin/zookeeper-server-start.sh $KAFKA_HOME/config/zookeeper.properties  
//Start Kafka server  
$KAFKA_HOME/bin/kafka-server-start.sh $KAFKA_HOME/config/server.properties
```

## ■ Create a topic

```
//Create a topic  
$KAFKA_HOME/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --  
partitions 1 --topic KafkaData
```

## ■ Test Kafka with Kafka producer and consumer

```
//Send some messages  
$KAFKA_HOME/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic KafkaData  
//Start a consumer  
$KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic KafkaData  
--from-beginning
```





# Instructions (cont.)

- Start consumer application

```
spark-submit --class "bangle.cs523.SparkStreamProj.App" --master yarn  
/home/cloudera/FinalProject/SparkStreamProj/target/SparkStreamProj-0.0.1-SNAPSHOT.jar
```

- Start producer application

```
spark-submit --class "bangle.cs523.KafkaProducer.App" --master yarn  
/home/cloudera/FinalProject/KafkaProducer/target/KafkaProducer-0.0.1-SNAPSHOT.jar
```

- Check Hbase for result

# Demo

- Start Kafka server and test Kafka with Kafka producer and consumer

The screenshot shows a VMware Workstation 15 Player window titled "cloudera-quickstart-vm-5.13.0-0-vmware - VMware Workstation 15 Player (Non-commercial use only)". The main window contains a terminal window titled "cloudera@quickstart:~" showing Kafka logs. The logs include messages about Kafka partitions, a test message being sent, and a group rebalance.

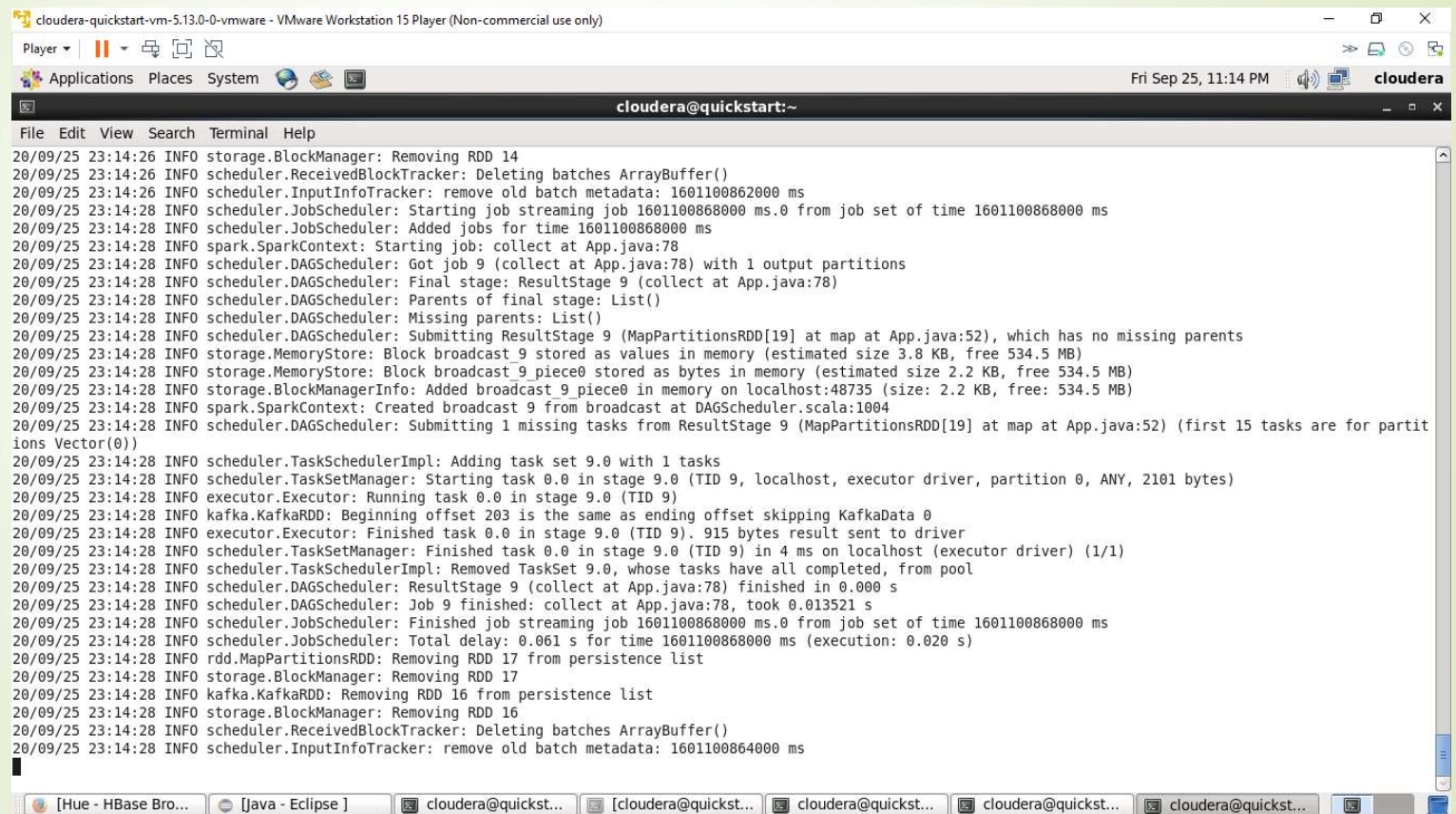
A smaller terminal window is open in the foreground, also titled "cloudera@quickstart:~". It shows the command prompt and the test message being sent to Kafka.

```
cloudera@quickstart:~$ test Kafka message
to make sure it is working
```

```
cloudera@quickstart:~$
```

# Demo (cont.)

## Start Consumer application

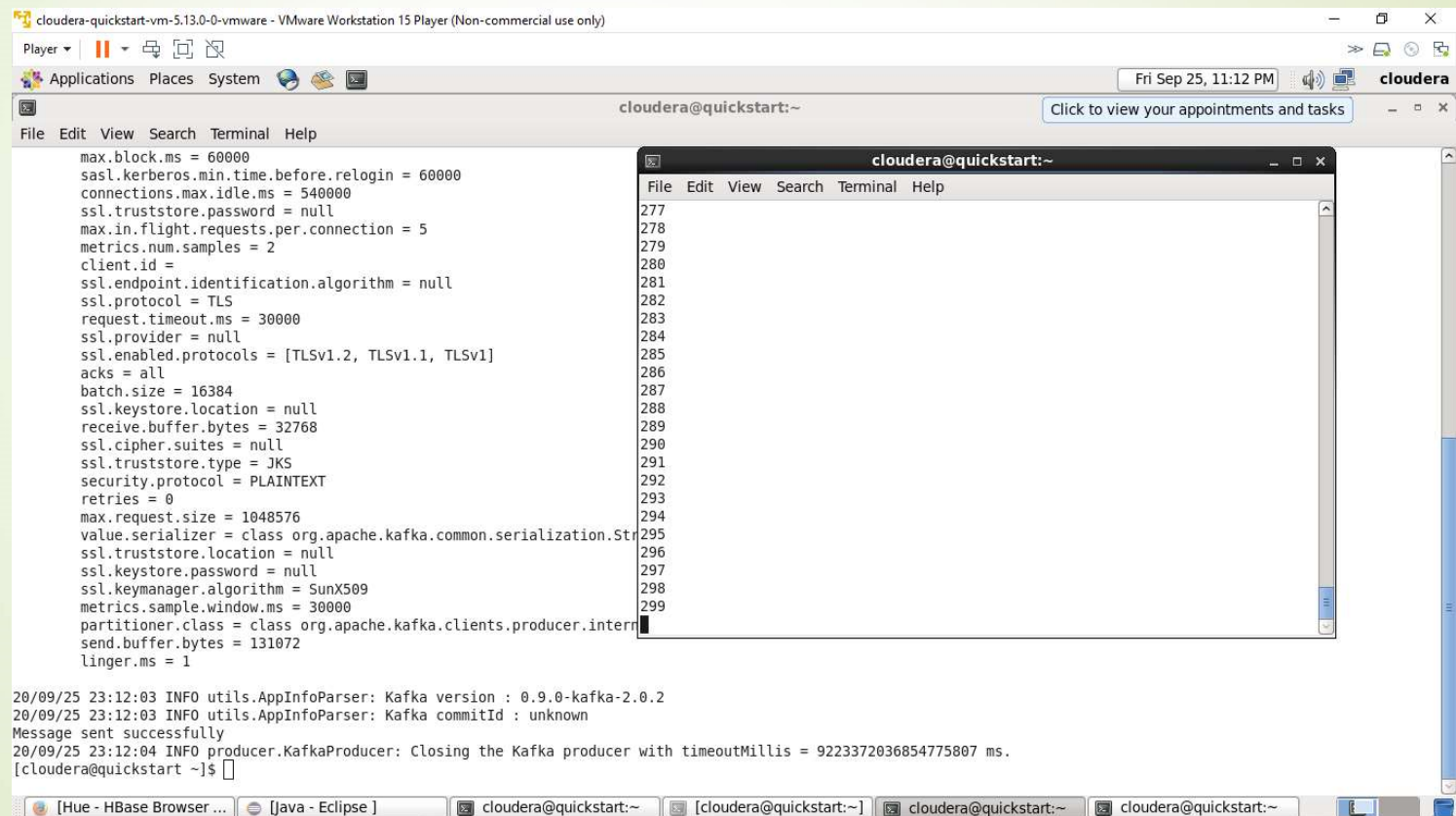


The screenshot shows a terminal window titled "cloudera-quickstart-vm-5.13.0-0-vmware - VMware Workstation 15 Player (Non-commercial use only)". The terminal displays a series of log messages from the Spark scheduler and executors. The logs indicate the start of a job streaming, the submission of a task set, and the completion of the job. The final message shows the job finished with a total delay of 0.061 s.

```
cloudera-quickstart-vm-5.13.0-0-vmware - VMware Workstation 15 Player (Non-commercial use only)
Player
Applications Places System
cloudera@quickstart:~
File Edit View Search Terminal Help
20/09/25 23:14:26 INFO storage.BlockManager: Removing RDD 14
20/09/25 23:14:26 INFO scheduler.ReceivedBlockTracker: Deleting batches ArrayBuffer()
20/09/25 23:14:26 INFO scheduler.InputInfoTracker: remove old batch metadata: 1601100862000 ms
20/09/25 23:14:28 INFO scheduler.JobScheduler: Starting job streaming job 1601100868000 ms.0 from job set of time 1601100868000 ms
20/09/25 23:14:28 INFO scheduler.JobScheduler: Added jobs for time 1601100868000 ms
20/09/25 23:14:28 INFO spark.SparkContext: Starting job: collect at App.java:78
20/09/25 23:14:28 INFO scheduler.DAGScheduler: Got job 9 (collect at App.java:78) with 1 output partitions
20/09/25 23:14:28 INFO scheduler.DAGScheduler: Final stage: ResultStage 9 (collect at App.java:78)
20/09/25 23:14:28 INFO scheduler.DAGScheduler: Parents of final stage: List()
20/09/25 23:14:28 INFO scheduler.DAGScheduler: Missing parents: List()
20/09/25 23:14:28 INFO scheduler.DAGScheduler: Submitting ResultStage 9 (MapPartitionsRDD[19] at map at App.java:52), which has no missing parents
20/09/25 23:14:28 INFO storage.MemoryStore: Block broadcast 9 stored as values in memory (estimated size 3.8 KB, free 534.5 MB)
20/09/25 23:14:28 INFO storage.MemoryStore: Block broadcast 9 piece0 stored as bytes in memory (estimated size 2.2 KB, free 534.5 MB)
20/09/25 23:14:28 INFO storage.BlockManagerInfo: Added broadcast 9 piece0 in memory on localhost:48735 (size: 2.2 KB, free: 534.5 MB)
20/09/25 23:14:28 INFO spark.SparkContext: Created broadcast 9 from broadcast at DAGScheduler.scala:1004
20/09/25 23:14:28 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 9 (MapPartitionsRDD[19] at map at App.java:52) (first 15 tasks are for partitions Vector(0))
20/09/25 23:14:28 INFO scheduler.TaskSchedulerImpl: Adding task set 9.0 with 1 tasks
20/09/25 23:14:28 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 9.0 (TID 9, localhost, executor driver, partition 0, ANY, 2101 bytes)
20/09/25 23:14:28 INFO executor.Executor: Running task 0.0 in stage 9.0 (TID 9)
20/09/25 23:14:28 INFO kafka.KafkaRDD: Beginning offset 203 is the same as ending offset skipping KafkaData 0
20/09/25 23:14:28 INFO executor.Executor: Finished task 0.0 in stage 9.0 (TID 9). 915 bytes result sent to driver
20/09/25 23:14:28 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 9.0 (TID 9) in 4 ms on localhost (executor driver) (1/1)
20/09/25 23:14:28 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 9.0, whose tasks have all completed, from pool
20/09/25 23:14:28 INFO scheduler.DAGScheduler: ResultStage 9 (collect at App.java:78) finished in 0.000 s
20/09/25 23:14:28 INFO scheduler.DAGScheduler: Job 9 finished: collect at App.java:78, took 0.013521 s
20/09/25 23:14:28 INFO scheduler.JobScheduler: Finished job streaming job 1601100868000 ms.0 from job set of time 1601100868000 ms
20/09/25 23:14:28 INFO scheduler.JobScheduler: Total delay: 0.061 s for time 1601100868000 ms (execution: 0.020 s)
20/09/25 23:14:28 INFO rdd.MapPartitionsRDD: Removing RDD 17 from persistence list
20/09/25 23:14:28 INFO storage.BlockManager: Removing RDD 17
20/09/25 23:14:28 INFO kafka.KafkaRDD: Removing RDD 16 from persistence list
20/09/25 23:14:28 INFO storage.BlockManager: Removing RDD 16
20/09/25 23:14:28 INFO scheduler.ReceivedBlockTracker: Deleting batches ArrayBuffer()
20/09/25 23:14:28 INFO scheduler.InputInfoTracker: remove old batch metadata: 1601100864000 ms
```

# Demo (cont.)

- Start Producer application and check Kafka consumer for generated numbers



```
cloudera-quickstart-vm-5.13.0-0-vmware - VMware Workstation 15 Player (Non-commercial use only)
Player | [Icons]
Applications Places System [Icons]
cloudera@quickstart:~
Click to view your appointments and tasks
File Edit View Search Terminal Help
max.block.ms = 60000
sas.l.kerberos.min.time.before.relogin = 60000
connections.max.idle.ms = 540000
ssl.truststore.password = null
max.in.flight.requests.per.connection = 5
metrics.num.samples = 2
client.id =
ssl.endpoint.identification.algorithm = null
ssl.protocol = TLS
request.timeout.ms = 30000
ssl.provider = null
ssl.enabled.protocols = [TLSv1.2, TLSv1.1, TLSv1]
acks = all
batch.size = 16384
ssl.keystore.location = null
receive.buffer.bytes = 32768
ssl.cipher.suites = null
ssl.truststore.type = JKS
security.protocol = PLAINTEXT
retries = 0
max.request.size = 1048576
value.serializer = class org.apache.kafka.common.serialization.StringSerializer
ssl.truststore.location = null
ssl.keystore.password = null
ssl.keymanager.algorithm = SunX509
metrics.sample.window.ms = 30000
partitioner.class = class org.apache.kafka.clients.producer.internals.DefaultPartitioner
send.buffer.bytes = 131072
linger.ms = 1
20/09/25 23:12:03 INFO utils.AppInfoParser: Kafka version : 0.9.0-kafka-2.0.2
20/09/25 23:12:03 INFO utils.AppInfoParser: Kafka commitId : unknown
Message sent successfully
20/09/25 23:12:04 INFO producer.KafkaProducer: Closing the Kafka producer with timeoutMillis = 9223372036854775807 ms.
[cloudera@quickstart ~]$
```

```
cloudera@quickstart:~
File Edit View Search Terminal Help
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
```

# Demo (cont.)

## Check result in HBase

The screenshot displays the Hue HBase Browser interface within a Mozilla Firefox browser window. The browser's address bar shows the URL `quickstart.cloudera:8888/hue/hbase#Cluster/numbers`. The interface includes a top navigation bar with various Cloudera services like Hadoop, HBase, Impala, Spark, Solr, Oozie, and Cloudera Manager. Below this, a search bar and a 'Query' dropdown are visible. The main content area shows a table with the following structure:

number	even_odd
0	even number
1	odd number
2	even number
3	odd number

The table is displayed in a grid-like format with alternating row colors. The left sidebar shows the 'Cluster' view with the 'numbers' table selected. The bottom of the screen shows the VMware Workstation taskbar with several open windows, including 'Hue - HBase Browser', '[Java - Eclipse]', and multiple terminal windows with the prompt `cloudera@quickst...`.



# Coveraged Requirements

Project Parts	Short Description	Notes
1	Spark Streaming Project	SparkStreamProj project
2	Spark and HBase Integration	SparkStreamProj project
3	Additional small research project	Integrated with Kafka in both projects
4	Video recording of project demo	Link to video will be provided



**Thank You**