



# EECS E6893 Big Data Analytics

## Spark Dataframe, Spark SQL, Hadoop metrics

Linyang He [linyang.he@columbia.edu](mailto:linyang.he@columbia.edu)

# Agenda

- Spark Dataframe
- Spark SQL
- Hadoop metrics

# Spark Dataframe

- An *abstraction*, an immutable distributed collection of data like RDD
- Data is organized into named columns, like a table in DB
- Create from RDD, Hive table, or other data sources
- Easy conversion with Pandas Dataframe

# Spark Dataframe: read from csv file

```
# read data from csv into Dataframe
```

```
df = spark.read.format("csv").option("header", 'true').load("gs://big_data_ta/data/citibike_stations.csv")
```

```
type(df)
```

```
pyspark.sql.dataframe.DataFrame
```

```
df.show(1)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|station_id|          name|short_name|  latitude|  longitude|region_id|rental_methods|capacity|eightd_has_key
_dispenser|num_bikes_available|num_bikes_disabled|num_docks_available|num_docks_disabled|is_installed|is_renting|is_r
eturning|eightd_has_available_keys|      last_reported|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      382|University Pl & E...|  5905.11|40.73492695|-73.99200509|      71|KEY,CREDITCARD|      0|
false|          0|          0|          0|          0|      false|      false|      fa
lse|      false|1970-01-01 00:00:00|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
only showing top 1 row
```

# Spark Dataframe: common operations

```
df.printSchema()
```

```
root
|-- station_id: string (nullable = true)
|-- name: string (nullable = true)
|-- short_name: string (nullable = true)
|-- latitude: string (nullable = true)
|-- longitude: string (nullable = true)
|-- region_id: string (nullable = true)
|-- rental_methods: string (nullable = true)
|-- capacity: string (nullable = true)
|-- eightd_has_key_dispenser: string (nullable = true)
|-- num_bikes_available: string (nullable = true)
|-- num_bikes_disabled: string (nullable = true)
|-- num_docks_available: string (nullable = true)
|-- num_docks_disabled: string (nullable = true)
|-- is_installed: string (nullable = true)
|-- is_renting: string (nullable = true)
|-- is_returning: string (nullable = true)
|-- eightd_has_available_keys: string (nullable = true)
|-- last_reported: string (nullable = true)
```

```
df.count()
```

# Spark Dataframe: common operations

```
df.columns
```

```
['station_id',  
 'name',  
 'short_name',  
 'latitude',  
 'longitude',  
 'region_id',  
 'rental_methods',  
 'capacity',  
 'eightd_has_key_dispenser',  
 'num_bikes_available',  
 'num_bikes_disabled',  
 'num_docks_available',  
 'num_docks_disabled',  
 'is_installed',  
 'is_renting',  
 'is_returning',  
 'eightd_has_available_keys',  
 'last_reported']
```

# Spark Dataframe: common operations

```
df.describe().show()
```

```
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
|summary|station_id|name|short_name|latitude|longitude|reg
ion_id|rental_methods|capacity|eightd_has_key_dispenser|num_bikes_available|num_bikes_disabled|num_docks_av
ailable|num_docks_disabled|is_installed|is_renting|is_returning|eightd_has_available_keys|last_reported|
+-----+-----+-----+-----+-----+-----+-----+
|count|843|843|843|843|843|843|
843|843|843|843|843|843|843|
843|843|843|843|843|843|843|
|mean|2434.425860023725|null|5806.786515151487|40.73212559944772|-73.9749901186049|70.93950177
935943|null|31.419928825622776|null|14.565836298932384|0.5693950177935944|16.1897983
3926453|0.05219454329774614|null|null|null|null|null|
|stddev|1421.1204113008778|null|1175.6743390458948|0.0387451696148341|0.031207687758326202|0.23854913482
063428|null|12.052012437572532|null|11.188256063195926|0.8613434732614029|13.15807566
4204775|0.6499620307701626|null|null|null|null|null|
|min|116|1 Ave & E 110 St|3460.01|40.655399774478312|-73.9077436|
70|KEY,CREDITCARD|0|false|false|false|0|0|
0|0|false|false|false|false|1970-01-01 00:00:00|
|max|83|York St|JC106|40.814394437915816|-74.0836394|
71|KEY,CREDITCARD|79|true|true|true|9|6|
9|9|true|true|true|true|2019-09-02 00:00:00|
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
```

# Spark Dataframe: common operations

```
df.describe('capacity').show()
```

```
+-----+-----+
|summary|      capacity|
+-----+-----+
|   count|           843|
|   mean|31.419928825622776|
| stddev|12.052012437572532|
|   min|              0|
|   max|              79|
+-----+-----+
```

```
df.select('station_id').distinct().count()
```

```
843
```



# Spark Dataframe: conversion with Pandas

```
# conversion with Pandas
```

```
import pandas as pd
```

```
pandaDf = df.toPandas()
```

```
pandaDf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 843 entries, 0 to 842
```

```
Data columns (total 18 columns):
```

```
station_id          843 non-null object
```

```
name                843 non-null object
```

```
short_name         843 non-null object
```

```
latitude           843 non-null object
```

```
longitude          843 non-null object
```

```
region_id          843 non-null object
```

```
rental_methods     843 non-null object
```

```
capacity           843 non-null object
```

```
eightd_has_key_dispenser 843 non-null object
```

```
num_bikes_available 843 non-null object
```

```
num_bikes_disabled 843 non-null object
```

```
num_docks_available 843 non-null object
```

```
num_docks_disabled 843 non-null object
```

```
is_installed       843 non-null object
```

```
is_renting         843 non-null object
```

```
is_returning       843 non-null object
```

```
eightd_has_available_keys 843 non-null object
```

```
last_reported     843 non-null object
```

```
dtypes: object(18)
```

```
memory usage: 118.6+ KB
```

# Work with Spark SQL

```
# Play with Spark SQL  
# Register the DataFrame as a SQL temporary view  
df.createOrReplaceTempView("citibike")
```

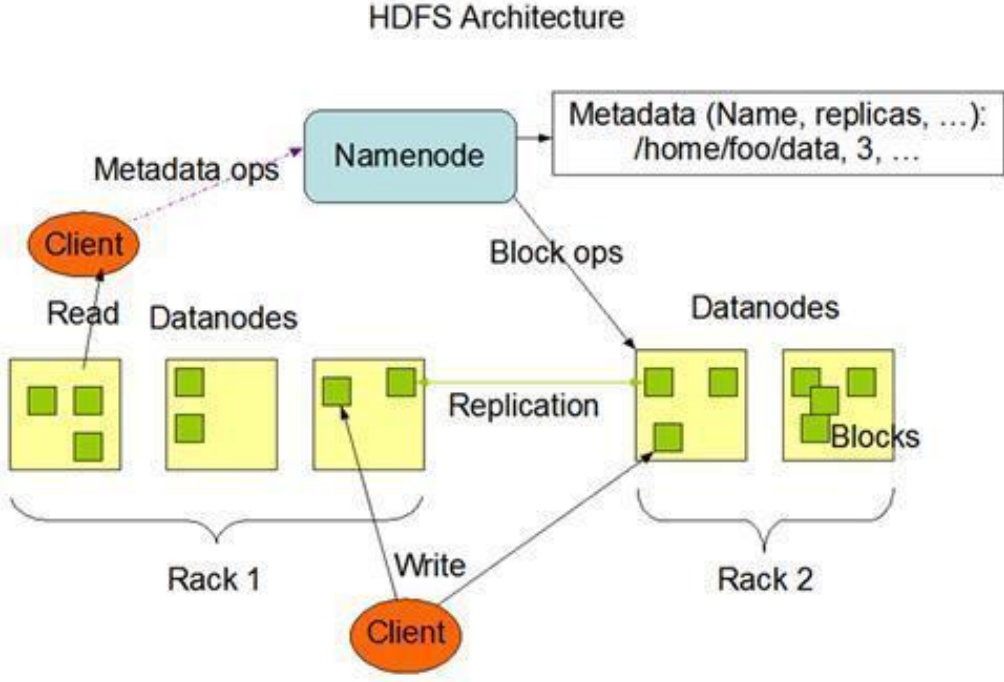
```
sqlDF = spark.sql("""  
    SELECT COUNT (DISTINCT station_id)  
    FROM citibike  
    """)  
sqlDF.show()
```

```
+-----+  
|count(DISTINCT station_id)|  
+-----+  
|                          843|  
+-----+
```

```
# get data out of df  
sqlDF.select("count(DISTINCT station_id)").collect()[0]["count(DISTINCT station_id)"]
```

```
843
```

# Hadoop metrics



# Collecting HDFS metrics

- Collecting NameNode metrics via API
- Collecting DataNode metrics via API
- Collecting HDFS metrics via JMX

# Collecting HDFS metrics

- **NameNode HTTP API**

The NameNode offers a summary of health and performance metrics through an easy-to-use web UI. By default, the UI is accessible via port 50070, so point a web browser at: `http://<namenodehost>:50070`

## Overview 'evan.hadoop' (active)

Started:	Mon Jun 6 17:58:36 UTC 2016
Version:	2.7.1.2.4.0-D-100, (2610498ad33884c87764738230071768542db)
Compiled:	2016-02-10T06:18Z by jenkins from (HEAD detached at 26104d8)
Cluster ID:	CID-211a5822-a1d0-497f-a85d-08996cbe65f1
Block Pool ID:	BP-706476385-10.0.2.15-1457965111091

## Summary

Security is off.

Safemode is off.

832 files and directories, 606 blocks = 1438 total filesystem object(s).

Heap Memory used 90.67 MB of 240 MB Heap Memory. Max Heap Memory is 240 MB.

Non Heap Memory used 54.7 MB of 130.63 MB Committed Non Heap Memory. Max Non Heap Memory is 304 MB.

Configured Capacity:	41.65 GB
DFS Used:	2.29 GB (5.5%)
Non DFS Used:	13.08 GB
DFS Remaining:	26.28 GB (63.1%)
Block Pool Used:	2.29 GB (5.5%)
DataNodes usages% (Min/Median/Max/stdDev):	5.50% / 5.50% / 5.50% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total DataNode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	602
Number of Blocks Pending Deletion	10
Block Deletion Start Time	6/23/2016, 2:58:38 PM

# Collecting HDFS metrics

- **DataNode HTTP API**

A high-level overview of the health of your DataNodes is available in the NameNode dashboard, under the Datanodes tab  
(<http://localhost:50070/dfshealth.html#tab-datanode>)

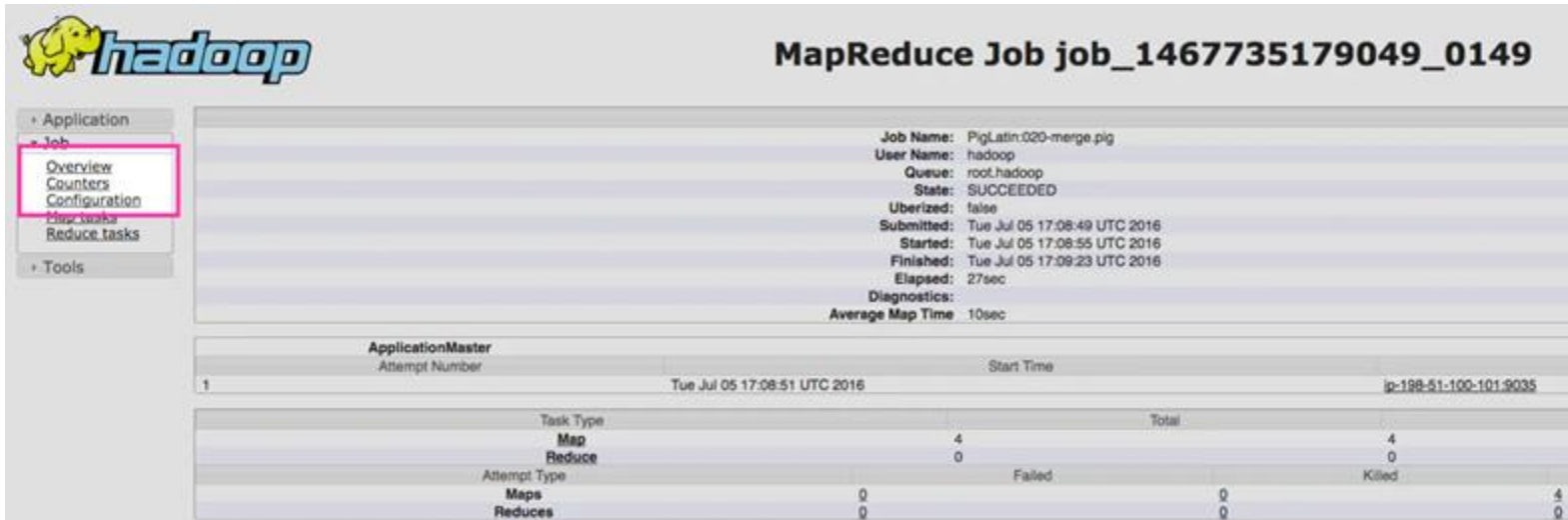
Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-198-51-100-100.ec2.internal (198.51.100.101:9200)	0	In Service	581.08 GB	7.2 GB	2.16 GB	571.72 GB	1908	7.2 GB (1.24%)	0	2.4.0-aman-6
ip-198-51-100-101.ec2.internal (198.51.100.102:9200)	2	In Service	581.08 GB	8.43 GB	1.72 GB	572.93 GB	1777	8.43 GB (1.11%)	0	2.4.0-aman-6

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks in files under construction
------	--------------	-------------------------	------------------------------	---

# Collecting MapReduce counters

MapReduce counters provide information on MapReduce task execution, like CPU time and memory used. They are dumped to the console when invoking Hadoop jobs from the command line, which is great for spot-checking as jobs run, but more detailed analysis requires monitoring counters over time.



The screenshot displays the Hadoop JobTracker web interface for a specific MapReduce job. The job name is "PigLatin.020-merge.pig" and it has successfully completed. The interface includes a navigation menu on the left with "Counters" highlighted, and a main area showing job metadata and a summary table of task execution.

**MapReduce Job job\_1467735179049\_0149**

Job Name: PigLatin.020-merge.pig  
User Name: hadoop  
Queue: root.hadoop  
State: SUCCEEDED  
Uberized: false  
Submitted: Tue Jul 05 17:08:49 UTC 2016  
Started: Tue Jul 05 17:08:55 UTC 2016  
Finished: Tue Jul 05 17:09:23 UTC 2016  
Elapsed: 27sec  
Diagnostics:  
Average Map Time: 10sec

ApplicationMaster	Attempt Number	Start Time	ip-198-51-100-101-9035	
	1	Tue Jul 05 17:08:51 UTC 2016		
Task Type			Total	
Map		4	4	
Reduce		0	0	
Attempt Type		Failed	Killed	
Maps	0	0	4	
Reduces	0	0	0	

# Collecting MapReduce counters



Logged in as: evan

## MapReduce Job job\_1467735179049\_0149

Job Overview

Job Name: PigLatin:020-merge.pig  
User Name: hadoop  
Queue: root.hadoop  
State: SUCCEEDED  
Uberized: false  
Submitted: Tue Jul 05 17:08:49 UTC 2016  
Started: Tue Jul 05 17:08:55 UTC 2016  
Finished: Tue Jul 05 17:09:23 UTC 2016  
Elapsed: 27sec  
Diagnostics:  
Average Map Time: 10sec

### ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Jul 05 17:08:51 UTC 2016	ip-188-51-100-101-9035	logs
Task Type			
Map	4	Total	4
Reduce	0		0
Attempt Type			
Maps	0	Failed	0
Reduces	0	Killed	4
			Successful





# Third-party tools

- Apache Ambari
- Cloudera Manager

# References

- <https://spark.apache.org/docs/latest/sql-getting-started.html>
- <https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/>
- <https://spark.apache.org/docs/latest/ml-guide.html>
- <https://towardsdatascience.com/machine-learning-with-pyspark-and-mllib-solving-a-binary-classification-problem-96396065d2aa>
- <https://www.datadoghq.com/blog/collecting-hadoop-metrics/#namenode-and-datanode-metrics-via-jmx>
- <https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/Metrics.html>

# Third-party tools

- **Apache Ambari**

ambari-server setup

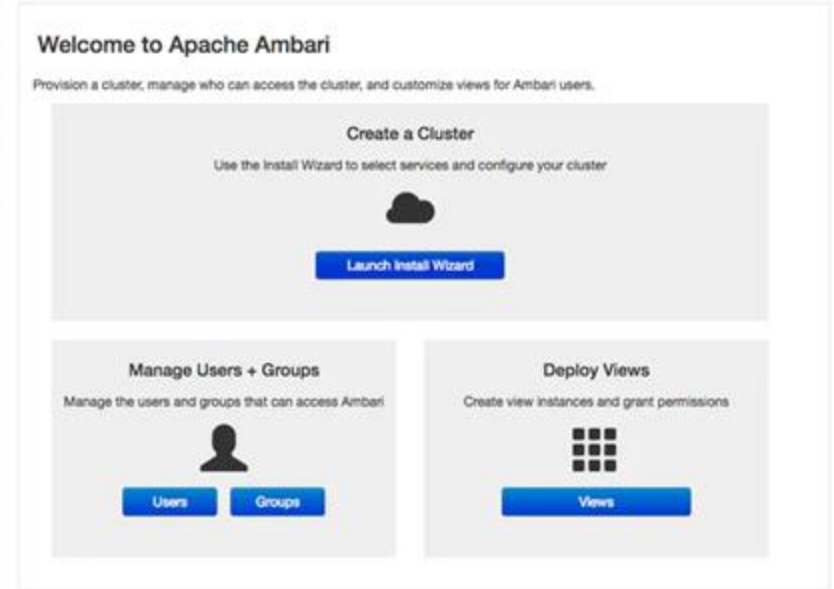
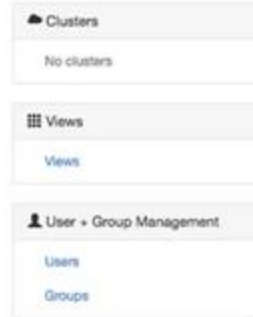
service ambari-server start

point your browser to

<AmbariHost>:8080 and login

with the default user admin and

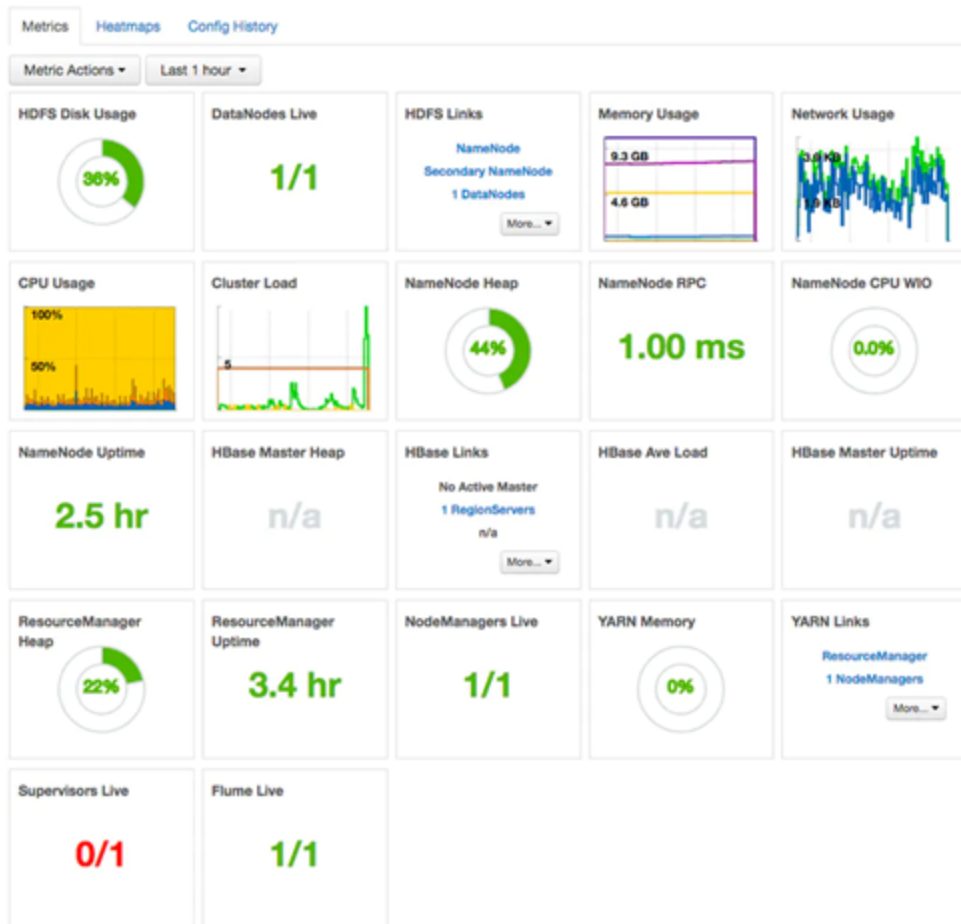
password admin



# Third-party tools

- **Apache Ambari**

Select “Launch Install Wizard”, the series of screens that follow, you will be prompted for hosts to be monitored and credentials to connect to each host in your cluster, then you’ll be prompted to configure application-specific settings.



# Third-party tools

- **Cloudera Manager**

service cloudera-scm-server start

point your browser to <ClouderaHost>:7180 and login with the default user admin and password admin.



# Hadoop metrics: take wordcount as an example

files	content	task
aa.txt	this is data and intelligence	Sort by alphabet, count the number of occurrences of each word in the three files.
bb.txt	hello everyone	
cc.txt	welcome	

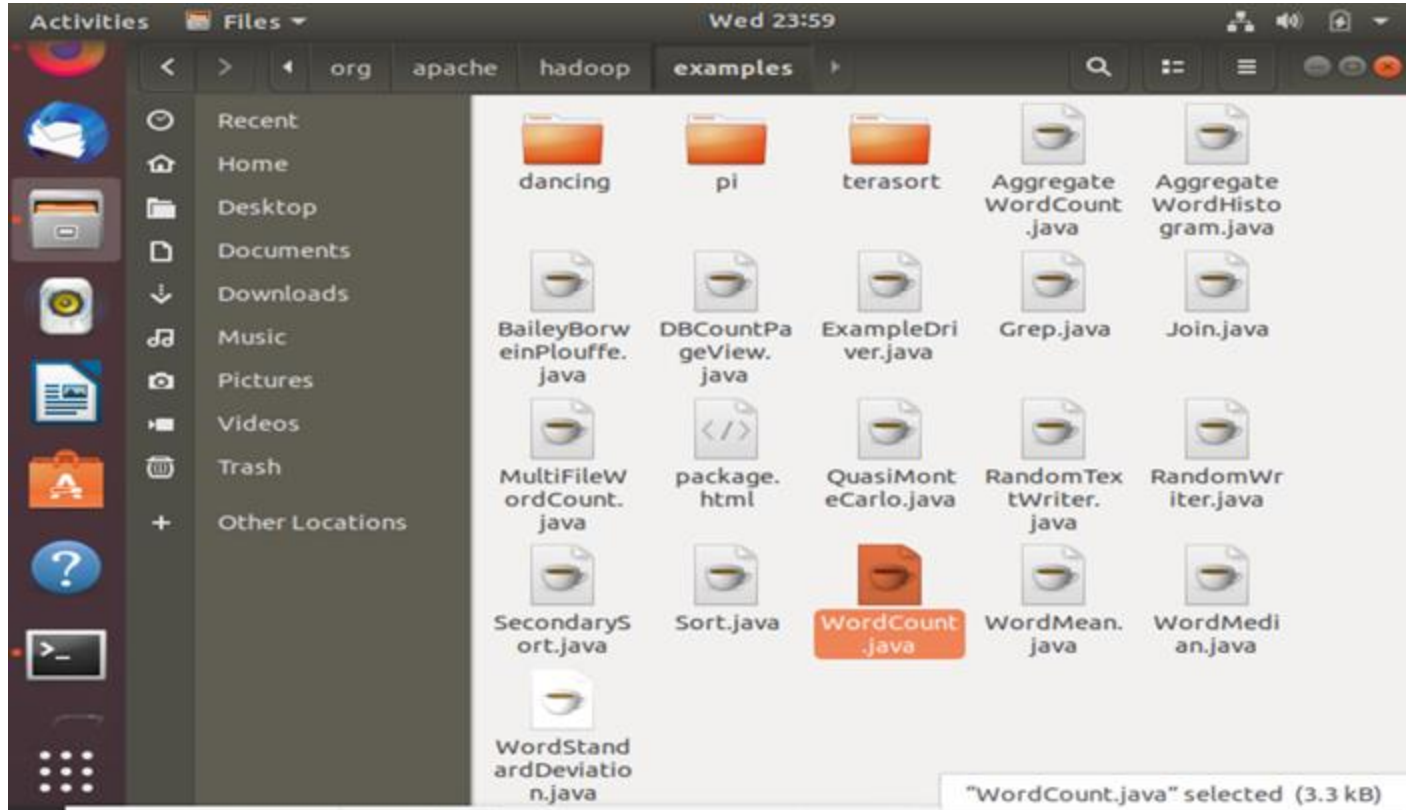
input	output
This is data and intelligence  Hello everyone  welcome	and 1
	data 1
	everyone 1
	hello 1
	intelligence 1
	is 1
	this 1
	welcome 1



# Hadoop metrics: take wordcount as an example



# Hadoop metrics: take wordcount as an example



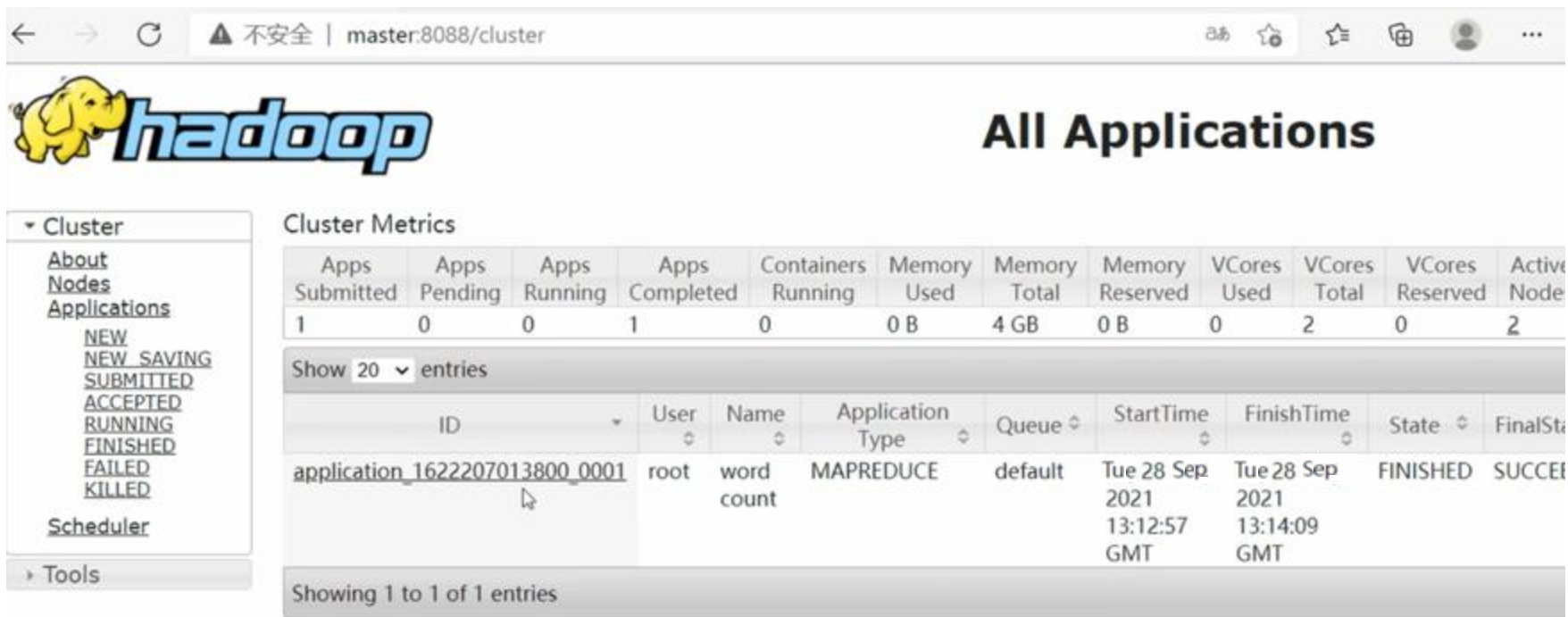
# Hadoop metrics: take wordcount as an example

The parameters after wordcount is input files except last one.

The last one “newwordcount” is output folder.

```
[root@master ~]# hadoop jar /usr/local/hadoop-2.6.5/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.5.jar wordcount /usr/root/txt/aa.txt /usr/root/txt/bb.txt /usr/root/txt/cc.txt /usr/root/txt/newwordcount
21/09/28 21:12:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
21/09/28 21:12:50 INFO client.RMPProxy: Connecting to ResourceManager at master/192.168.65.100:8032
21/09/28 21:12:54 INFO input.FileInputFormat: Total input paths to process : 3
21/09/28 21:12:55 INFO mapreduce.JobSubmitter: number of splits:3
21/09/28 21:12:56 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1622207013800_0001
21/09/28 21:12:58 INFO impl.YarnClientImpl: Submitted application application_1622207013800_0001
21/09/28 21:12:58 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1622207013800_0001/
21/09/28 21:12:58 INFO mapreduce.Job: Running job: job_1622207013800_0001
21/09/28 21:13:23 INFO mapreduce.Job: Job job_1622207013800_0001 running in uber mode : false
21/09/28 21:13:23 INFO mapreduce.Job:  map 0% reduce 0%
21/09/28 21:13:42 INFO mapreduce.Job:  map 33% reduce 0%
21/09/28 21:13:51 INFO mapreduce.Job:  map 67% reduce 0%
```

# Hadoop metrics: take wordcount as an example



The screenshot shows the Hadoop web interface at the URL `master:8088/cluster`. The page title is "All Applications". On the left, there is a navigation menu under "Cluster" with links for "About", "Nodes", "Applications", "Scheduler", and "Tools". The "Applications" link is selected, showing a list of application states: NEW, NEW SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, and KILLED.

The main content area displays "Cluster Metrics" and a table of application entries. The "Cluster Metrics" table shows the following data:

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes
1	0	0	1	0	0 B	4 GB	0 B	0	2	0	2

Below the metrics, there is a "Show 20 entries" dropdown and a table of application details. The table has columns for ID, User, Name, Application Type, Queue, StartTime, FinishTime, State, and FinalState. One entry is shown:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalState
<a href="#">application_1622207013800_0001</a>	root	word count	MAPREDUCE	default	Tue 28 Sep 2021 13:12:57 GMT	Tue 28 Sep 2021 13:14:09 GMT	FINISHED	SUCCESS

At the bottom of the application list, it says "Showing 1 to 1 of 1 entries".

# Hadoop metrics: take wordcount as an example

File: [/usr/root/txtdir/newwordcount/part-r-00000](#)

Goto :

[Go back to dir listing](#)

[Advanced view/download options](#)

```
and      1
data     1
everyone      1
hello     1
intelligence 1
is        1
this      1
wellcome   1
```