



Data Science for Wall Street

Cloudera Data Science | Hadoop World NY 2015



Your Hosts



Sean Owen
@sean_r_owen



Juliet Hougland
@j_houg



Sandy Ryza
@SandySifting

A photograph of a person's hands holding three kittens. The person is wearing a grey blazer over a red shirt. The kittens are a tabby, a black, and a black and white. The image is framed by a blue border.

Data

we are
Scientists
WITH LOVE AND SQUALOR

Problem

Predict stock price
movement from
Wikipedia traffic

cloudera

PET SHOP BOYS
OPPORTUNITIES
[let's make lots of money]

Goals

- Hands-on Play
 - Apache [Spark](#), Pyspark
 - [ipython / jupyter](#)
 - [CDH](#), Hue
- [Munge data](#) like a pro with Spark
 - Wikipedia page views
 - S&P 500 ticker data
- Mine for [correlations](#) safely
- Explore [time series](#) analysis

cloudera



STANDARD
&POOR'S **500**



Agenda



Break



+

Break



Intro and Setup

Correlations, Volatility, Outliers, Black Monday

Follow Along with Source

`github.com/srowen/
ds-for-wall-street`

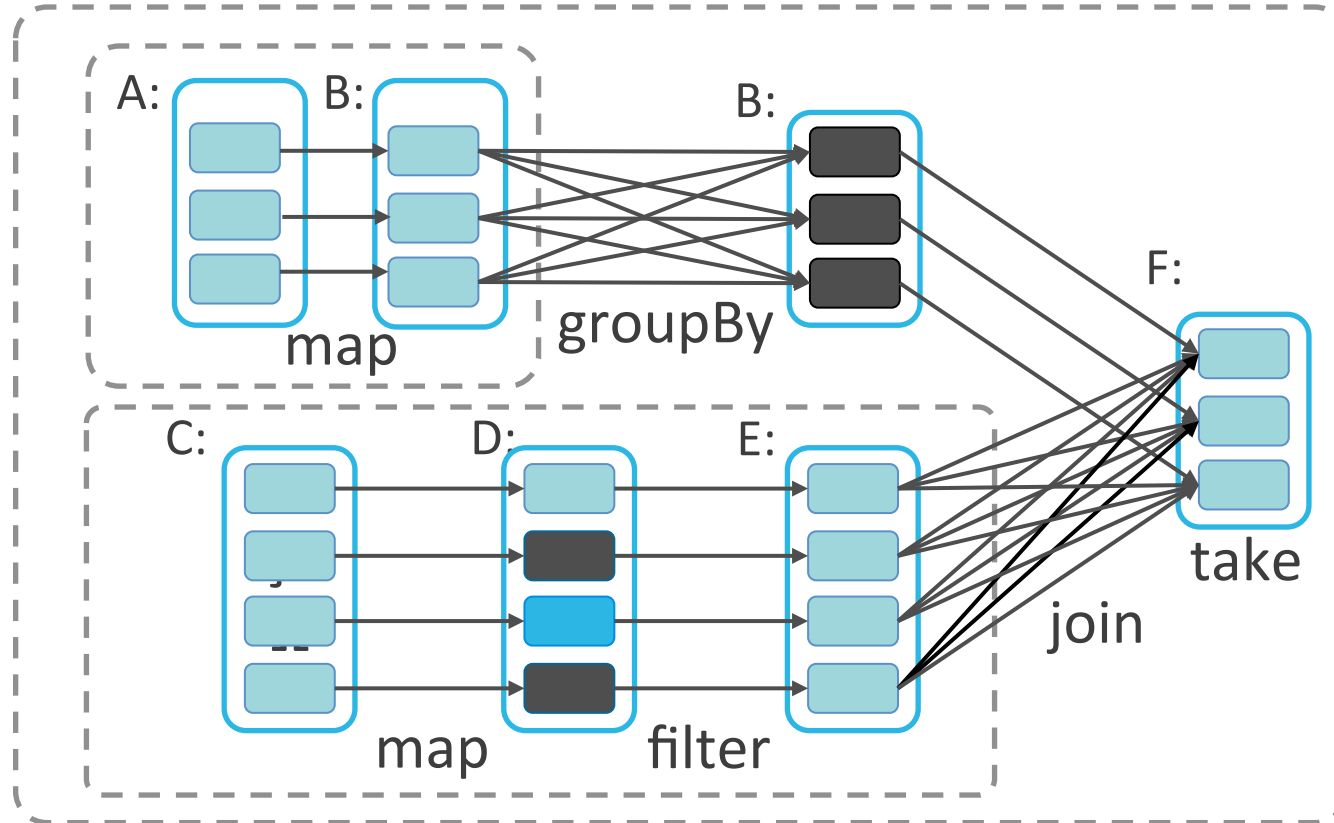
Cluster Tools

Apache Spark

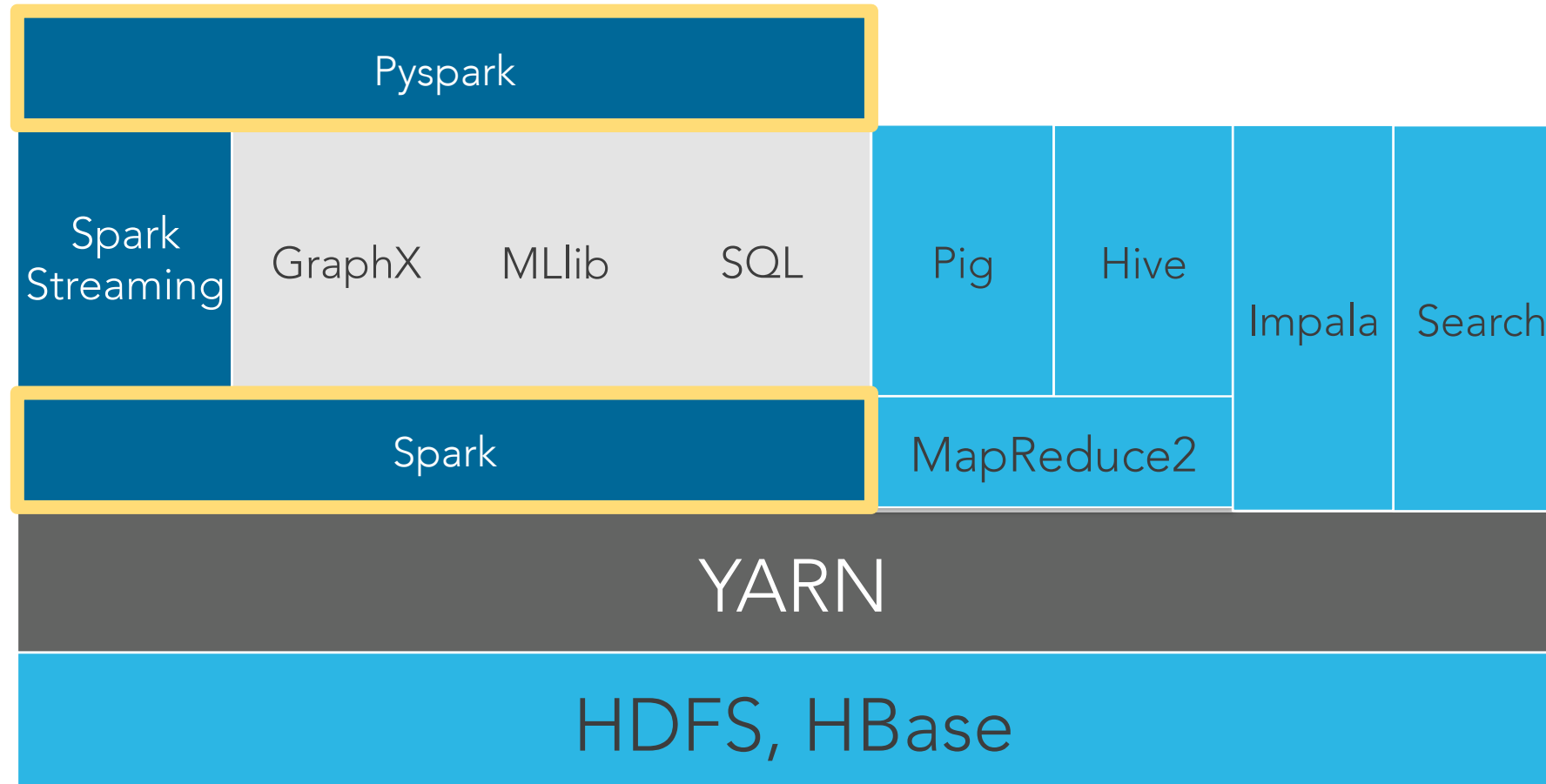
- Fast, in-memory processing
- Ecosystem of related subprojects
 - Streaming
 - ML
 - SQL-like processing
- Language bindings
 - Scala / Java
 - Python
 - R



Complex, In-Memory Processing



Spark in CDH



ipython / jupyter

- ipython
 - Notebook interface for Python
 - Annotate Python-based analysis
 - Web-based, graphics
- jupyter = ipython 4.x
- Can notebook-ize several tools, including Pyspark



CDH Cluster Info

Experiment

Can 40 people in a tutorial share an open cluster without clobbering each other?

cloudera



Cluster Info

Cluster

- CDH 5.4 + Spark 1.3
- Pyspark + Python 2.7
- ipython 4
- 5 nodes + CM management
 - Google Compute engine
 - 160 cores
 - 600GB RAM
 - 10TB disk

Your Piece of the Cluster

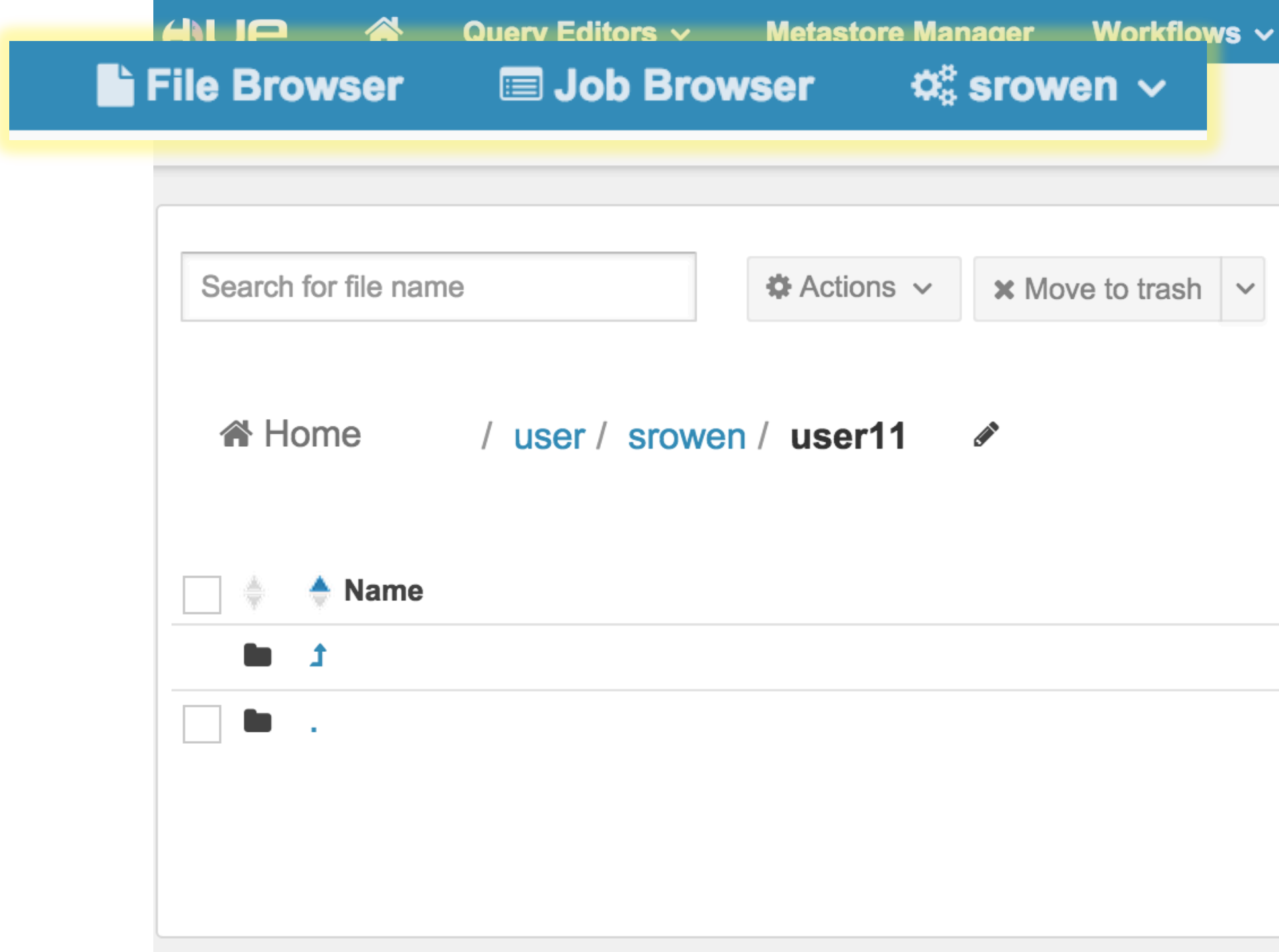
- 40 slots available
(4 cores, 12GB RAM)
- Claim a card which gives you:
 - Host to access
 - ipython notebook name
 - Directory
 - Spark UI port
- **Free Play** – no need to follow along

Your Piece of the Cluster

Hue: cdh4.srowen.com:8888 srowen/srowen
ipython: cdh1.srowen.com:8880 user11.ipynb
Spark: cdh1.srowen.com:4040 (redirects)
HDFS: /user/srowen/user11


Hue
HDFS

cloudera



ipython /
Jupyter

cloudera

 **Jupyter** user11 Last Checkpoint: a


File Edit View Insert Cell Kernel

Save Add Cut Copy Paste Undo Redo

In [1]: `print sc`

`<pyspark.context.SparkContext`

In []:

 **Jupyter**

Files Running Clusters

Select items to perform actions on them.

- ☐ user11.ipynb
- ☐ user12.ipynb
- ☐ user13.ipynb
- ☐ user14.ipynb
- ☐ user15.ipynb
- ☐ user16.ipynb
- ☐ user17.ipynb
- ☐ user18.ipynb

cdh1.srowen.com:4040



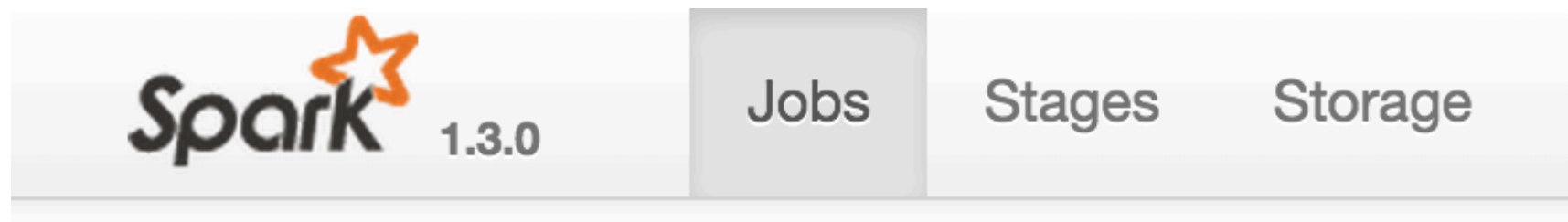
cdh4.c.ds-wallstreet.internal:....



cdh4.srowen.com:....

Spark on YARN

cloudera



Spark Jobs (?)

Total Duration: 1.9 min

Scheduling Mode: FIFO

Completed Jobs: 1

Completed Jobs (1)

Job Id	Description
0	runJob at PythonRDD.scala:356

The Data

Wikipedia Pagecount Data Set

- Clicks per page per hour
- `dumps.wikimedia.org/other/pagecounts-raw`
- site page hits bytes
- Aug 3 – Sep 22 2015
 - 566GB, ~11GB per day
 - 8.2B lines, ~164M per day
- Date in *file name*
 - `pagecounts-20150701-*`

```
...
en Mars 92 12952918
en Mars,_Incorporated 21 596441
en Mars,_Pennsylvania 2 49320
en Mars,_the_Bringer_of_War 1 28986
en Mars-Grunt 1 13851
en Mars-la-Tour 1 19013
en Mars-sous-Bourcq 1 18636
en Mars:_War_Logs 1 9535
en MarsDial 4 22276
en Mars_%28chocolate_bar%29 1 23738
en Mars_(1968_film) 1 9977
en Mars_(Gackt_album) 1 14180
en Mars_(Swedish_ship) 1 10781
en Mars_(TV_series) 3 32973
en Mars_(astrology) 1 44190
en Mars_(chocolate_bar) 15 356060
...
```

Preprocessing and Sampling Wikipedia (1)

```
import org.apache.hadoop.fs.{FileSystem,Path}

def pagecountsFileToRDD(path: Path) = {
  val mhdMatch =
    "pagecounts-(\\d{4})(\\d{2})(\\d{2})-(\\d{2})0000".r.
    findFirstMatchIn(path.getName).get
  val List(year, month, day, hour) =
    mhdMatch.subgroups.map(_.toInt)

  sc.textFile(path.toString).map { line =>
    val Array(site, page, hits, _) = line.split(" ")
    (site, page, hits.toInt)
  }.filter {
    case (site, _, hits) => site == "en" && hits >= 20
  }.map { case (_, page, hits) =>
    (year, month, day, hour, page, hits)
  }
}
```

- Function from file (Path) to RDD of tuples of 6 strings
- Year/month/day/hour from name with regex
- Page/hits from line
- Drop non-en, hits < 20
- 350x smaller!

Preprocessing and Sampling Wikipedia (2)

```
val paths =  
  FileSystem.get(sc.hadoopConfiguration).  
  listStatus(new Path("/user/.../Wikipedia")).  
  map(_.getPath)  
  
val hourlyRDDs = paths.map(pagecountsFileToRDD)  
  
sc.union(hourlyRDDs).  
  repartition(20).  
  map(_.productIterator.mkString("\t")).  
  saveAsTextFile("/user/admin/Wikipedia")
```

- List all files (Paths)
- Parse as RDD
- Union and repartition
- Convert to TSV and save
- 566GB -> 1.6GB

S&P 500 Ticker Data Set

- From Google Finance
- 500 S&P 500 companies
- Price / volume per ticker per ~3 minutes
- `time,close,...,volume`
- 1 file per company
- Aug 3 – Sep 22 2015
 - 79MB
 - 2.3M lines

```
...
a1438608600,626.13,626.21,625.34,625.34,21786
1,627.58,628.47,625.51,625.94,13038
2,628.095,628.28,627.07,627.852,11602
3,628.45,629.608,627.21,627.91,13638
4,628.84,629.6,628.35,628.72,8068
5,627.93,629.1,627.41,628.84,8886
6,628.25,628.97,627.64,627.96,6208
7,628.7,628.7,627.85,628.21,3755
8,629.4,629.4,628.59,628.89,10191
9,629.28,629.88,628.42,629.02,12075
10,629.685,629.9799,629.02,629.505,12290
11,629.42,629.98,629.27,629.89,7416
12,629,629.62,628.4,629.42,7074
13,630.19,630.4,628.82,628.82,19672
14,629.24,629.8,629.01,629.79,3200
15,628.95,629.63,628.586,629.3035,10259
...
```

Preprocessing and Sampling Ticker (1)

```
val tickerFiles = sc.wholeTextFiles("/.../")

val ticks = tickerFiles.flatMap { case (file, csv) =>
  val ticker = file.split('/').last.split('.').head
  var baseTime = 0L
  csv.split("\n").map { line =>
    val Array(time, close, _, _, _, vol) =
      line.split(',')
    val tickTime =
      if (time.head == 'a') {
        baseTime = time.tail.toLong
        baseTime
      } else {
        baseTime + (180 * time.toLong)
      }
    (tickTime, ticker, close.toDouble, vol.toLong)
  }
}
```

- Parse whole ~150K files at a time
- Convert absolute aXXX and relative offsets to Unix timestamp
- Output tuples with time, ticker, close, volume

Preprocessing and Sampling Ticker (2)

```
import java.util.{Calendar,Locale,TimeZone}

val closeVolByHourTicker = ticks.map {
  case (tickTime, ticker, close, vol) =>
    val cal = Calendar.getInstance(
      TimeZone.getTimeZone("GMT"), Locale.ENGLISH)
    cal.setTimeInMillis(tickTime * 1000)
    val year = cal.get(Calendar.YEAR)
    val month = cal.get(Calendar.MONTH) + 1
    val day = cal.get(Calendar.DAY_OF_MONTH)
    val hour = cal.get(Calendar.HOUR_OF_DAY)
    ((year, month, day, hour, ticker), (close, vol))
}
```

- Convert timestamp to year/month/day/hour GMT
- Key = time and ticker
- Value = close and volume

Preprocessing and Sampling Ticker (3)

```
val tickerVolVWAP = closeVolByHourTicker.  
  groupByKey().  
  mapValues { closeVols =>  
    val totalVol = closeVols.map {  
      case (_, vol) => vol  
    }.sum  
    val vwap = closeVols.map {  
      case (close, vol) => close * vol  
    }.sum / totalVol  
    (totalVol, vwap)  
  }
```

```
tickerVolVWAP.map {  
  case ((y, m, d, h, ticker), (vol, vwap)) =>  
    Seq(y, m, d, h, ticker, vol, vwap).mkString("\t")  
}.saveAsTextFile("/user/admin/Ticker/")
```

- Group (close, volume) by time and ticker key
- Compute total volume and VWAP (volume-weighted average price)
- Output as TSV

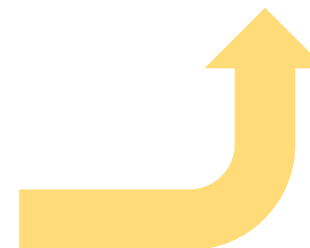
Data Frames

DataFrame = Tabular RDD + Schema

- Like R, Python DataFrame concept
- Specific type of RDD containing row-like objects
- Columns, with names and types
- Enables SQL-like operations

Sepal Length	Sepal Width	Species
5.1	3.5	setosa
4.9	3.0	setosa
4.7	3.2	setosa
6.3	3.3	virginica
5.8	2.7	virginica
7.1	3.0	virginica

```
"5.1,3.5,setosa"  
"4.9,3.0,setosa"  
"4.7,3.2,setosa"  
"6.3,3.3,virginica"  
"5.8,2.7,virginica"  
"7.1,3.0,virginica"
```



To DataFrames and to Spark SQL

```
from pyspark.sql import SQLContext, Row

lines = sc.textFile("/user/admin/Wikipedia/*")
def parse_line(line):
    tokens = line.split('\t')
    return Row(page=tokens[4], hits=int(tokens[5]))
data = lines.map(parse_line)

sqlContext = SQLContext(sc)
wtDataFrame = sqlContext.createDataFrame(data)
wtDataFrame.registerTempTable("wt")

hitCountsRDD = sqlContext.sql(
    "SELECT hits, COUNT(*) AS c FROM wt "
    "GROUP BY hits ORDER BY hits").cache()
hitCounts = hitCountsRDD.collect()
```

- Parse line and create Row
- Infer DataFrame[
 hits: bigint,
 page: string]
- Count hits using SQL
- Get Rows of (hits, count)

Minutes Later...

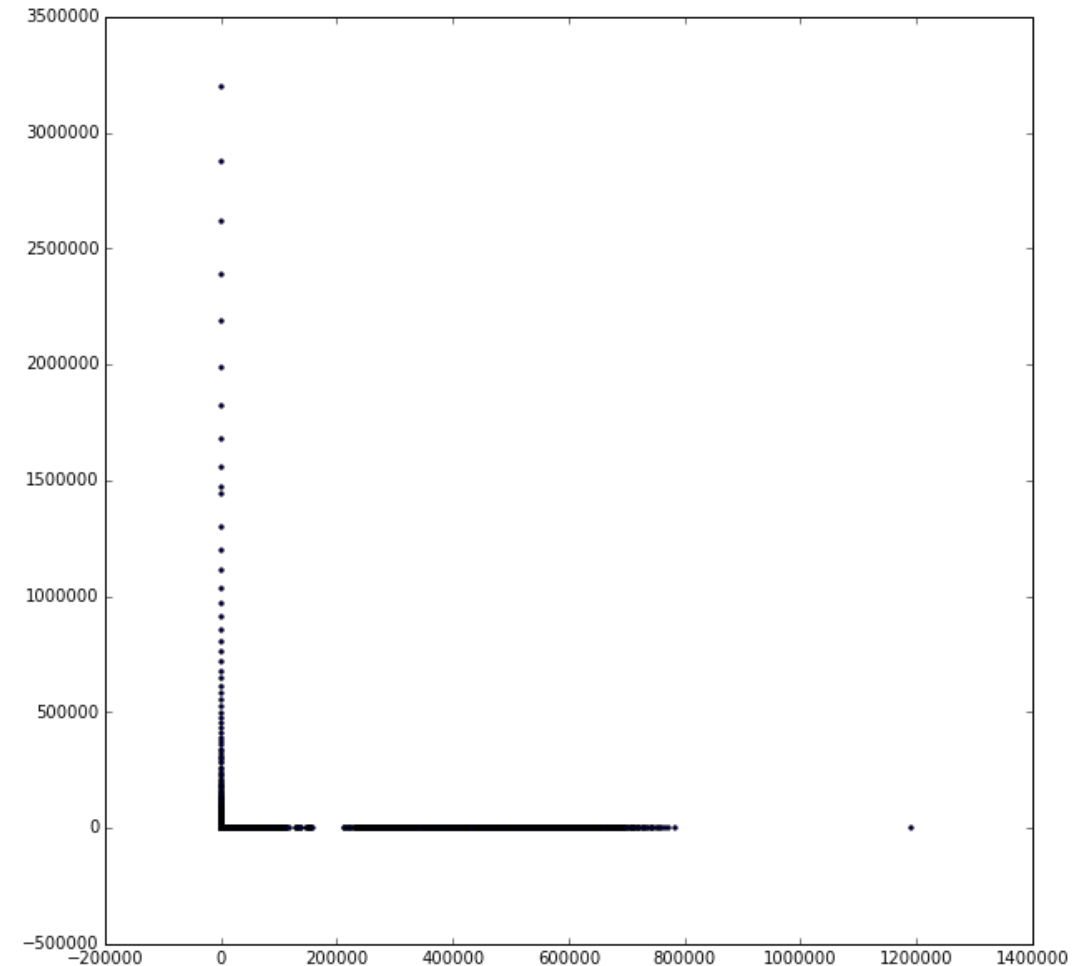
How (Not) to Plot Power Law Distributions

```
import matplotlib.pyplot as plt

%matplotlib inline

hits = map(lambda r: r.hits, hitCounts)
counts = map(lambda r: r.c, hitCounts)

plt.figure(figsize=(10,10))
plt.scatter(hits, counts, marker=".")
plt.show()
```



Log(Power Law)= Linear

$$counts = a hits^{-b}$$

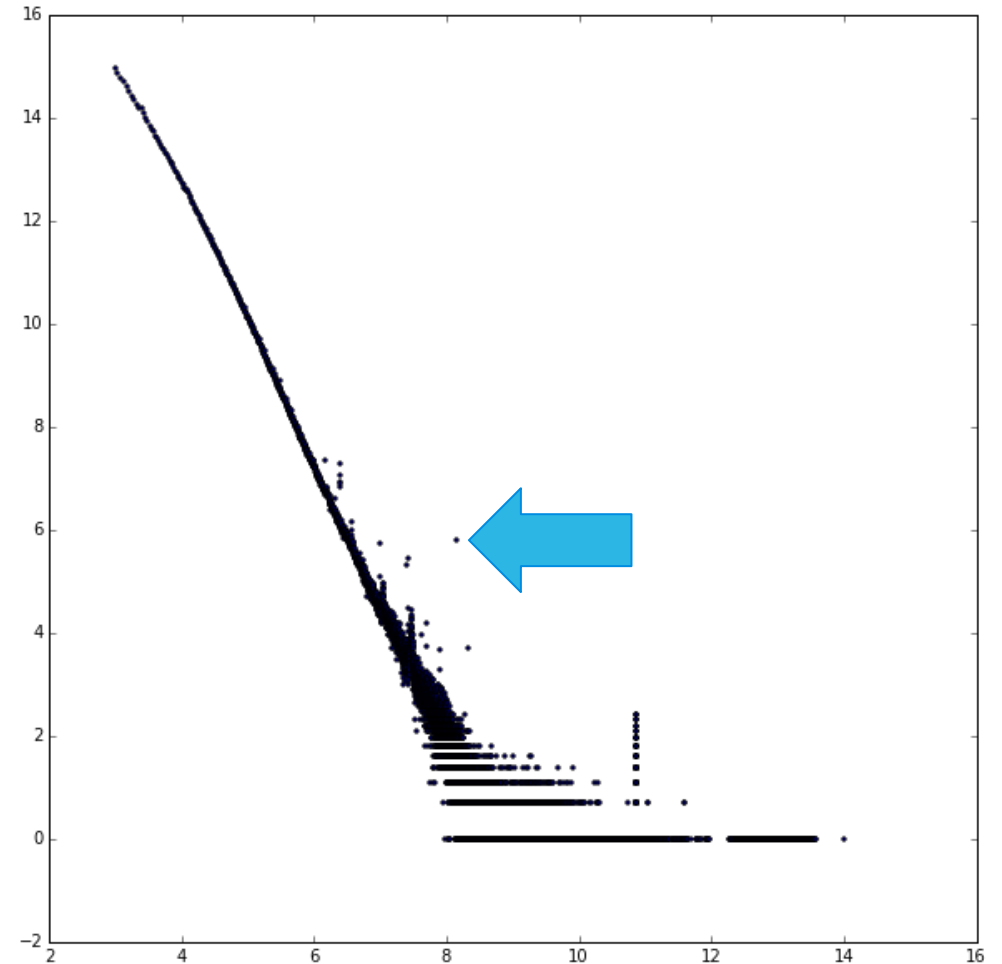
$$\log(counts) = \log(a) - b \log(hits)$$

Plotting log-log

```
import math

logHits = map(lambda r:
    math.log(r.hits), hitCounts)
logCounts = map(lambda r:
    math.log(r.c), hitCounts)

plt.figure(figsize=(10,10))
plt.scatter(
    logHits, logCounts, marker=".")
plt.show()
```



Anomaly Hunting: Back to RDDs

```
hitCountsRDD.rdd.filter(lambda r:  
    math.log(r.hits) >= 8.0 and  
    math.log(r.hits) <= 8.3 and  
    math.log(r.c) >= 5.0).collect()
```

...

```
[Row(hits=3510, c=334)]
```

- Select part of data by eyeballing the graph
- Spike near 3510 page views

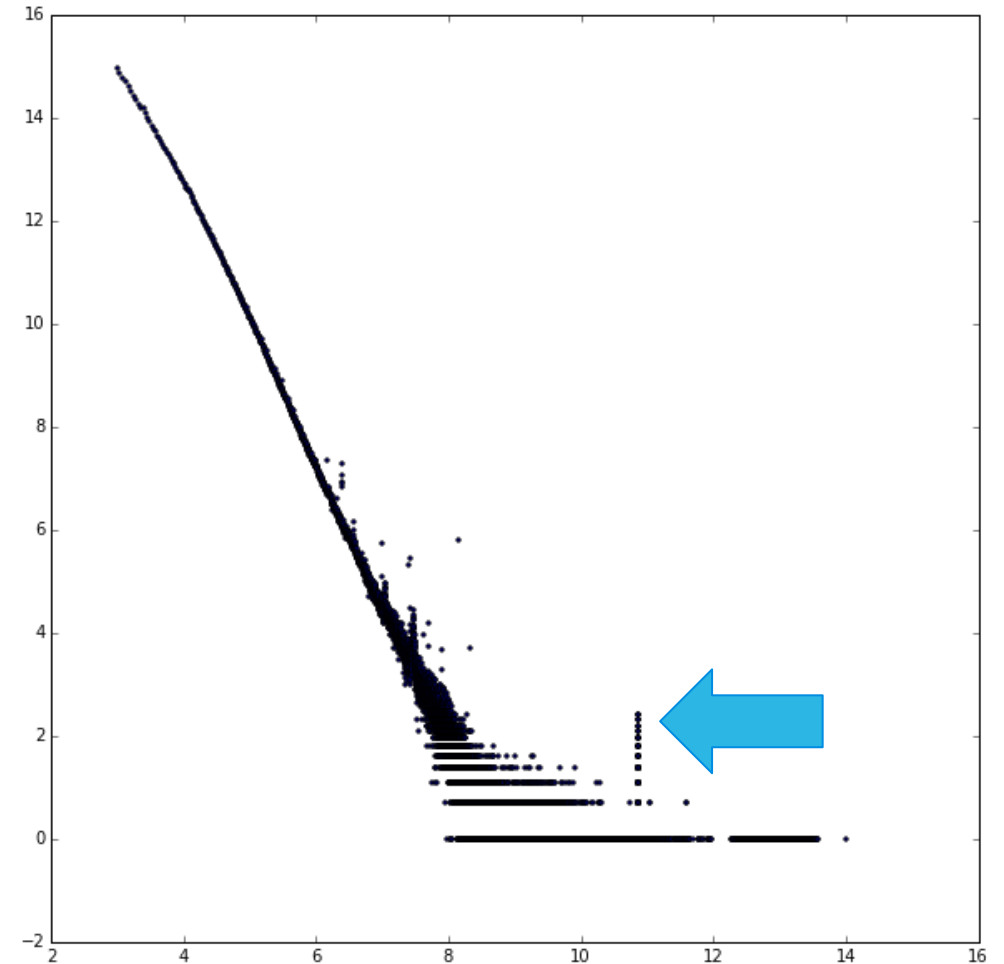
???

Exercise: What are these pages getting exactly 3510 views?

```
[Row(day=18, hits=3510, hour=4, month=9, page=u'https://en.wikipedia.org/wiki/F%C3%A9d%C3%A9ration_Internationale_de_l%27Automobile', year=2015),
 Row(day=18, hits=3510, hour=6, month=9, page=u'https://en.wikipedia.org/wiki/F%C3%A9d%C3%A9ration_Internationale_de_l%27Automobile', year=2015),
 Row(day=10, hits=3510, hour=7, month=9, page=u'https://en.wikipedia.org/wiki/F%C3%A9d%C3%A9ration_Internationale_de_l%27Automobile', year=2015),
 Row(day=11, hits=3510, hour=0, month=9, page=u'https://en.wikipedia.org/wiki/F%C3%A9d%C3%A9ration_Internationale_de_l%27Automobile', year=2015),
 Row(day=20, hits=3510, hour=9, month=9, page=u'https://en.wikipedia.org/wiki/F%C3%A9d%C3%A9ration_Internationale_de_l%27Automobile', year=2015),
 Row(day=20, hits=3510, hour=13, month=9, page=u'https://en.wikipedia.org/wiki/F%C3%A9d%C3%A9ration_Internationale_de_l%27Automobile', year=2015),
 Row(day=14, hits=3510, hour=4, month=9, page=u'https://en.wikipedia.org/wiki/F%C3%A9d%C3%A9ration_Internationale_de_l%27Automobile', year=2015),
 ...]
```

Anomaly Hunting 2

```
[Row(hits=52524, c=9),  
 Row(hits=52526, c=8),  
 Row(hits=52529, c=10),  
 Row(hits=52620, c=11),  
 Row(hits=52627, c=8),  
 Row(hits=52632, c=11),  
 Row(hits=52634, c=9),  
 Row(hits=52641, c=10),  
 Row(hits=52643, c=11),  
 Row(hits=52652, c=9),  
 Row(hits=52655, c=10),  
 Row(hits=52656, c=8)]
```



```
[Row(day=5, hits=52645, hour=11, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=13, hits=52634, hour=3, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=17, hits=52610, hour=2, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=29, hits=52543, hour=13, month=8, page=u'Special:BlankPage', year=2015),  
Row(day=1, hits=52600, hour=19, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=3, hits=52574, hour=3, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=4, hits=52614, hour=23, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=14, hits=52644, hour=3, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=16, hits=52565, hour=17, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=18, hits=52634, hour=14, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=21, hits=52650, hour=22, month=9, page=u'Special:BlankPage', year=2015),  
Row(day=29, hits=52529, hour=18, month=8, page=u'Special:BlankPage', year=2015),  
Row(day=31, hits=52526, hour=19, month=8, page=u'Special:BlankPage', year=2015),  
...
```



cloudera Intermission

Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh1.srowen.com:8880 user11.ipynb Spark: cdh1.srowen.com:4040 (redirects) HDFS: /user/srowen/user11	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh1.srowen.com:8880 user12.ipynb Spark: cdh1.srowen.com:4041 (redirects) HDFS: /user/srowen/user12
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh1.srowen.com:8880 user13.ipynb Spark: cdh1.srowen.com:4042 (redirects) HDFS: /user/srowen/user13	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh1.srowen.com:8880 user14.ipynb Spark: cdh1.srowen.com:4043 (redirects) HDFS: /user/srowen/user14
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh1.srowen.com:8880 user15.ipynb Spark: cdh1.srowen.com:4044 (redirects) HDFS: /user/srowen/user15	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh1.srowen.com:8880 user16.ipynb Spark: cdh1.srowen.com:4045 (redirects) HDFS: /user/srowen/user16
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh1.srowen.com:8880 user17.ipynb Spark: cdh1.srowen.com:4046 (redirects) HDFS: /user/srowen/user17	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh1.srowen.com:8880 user18.ipynb Spark: cdh1.srowen.com:4047 (redirects) HDFS: /user/srowen/user18

Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh2.srowen.com:8880 user21.ipynb Spark: cdh2.srowen.com:4040 (redirects) HDFS: /user/srowen/user21	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh2.srowen.com:8880 user22.ipynb Spark: cdh2.srowen.com:4041 (redirects) HDFS: /user/srowen/user22
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh2.srowen.com:8880 user23.ipynb Spark: cdh2.srowen.com:4042 (redirects) HDFS: /user/srowen/user23	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh2.srowen.com:8880 user24.ipynb Spark: cdh2.srowen.com:4043 (redirects) HDFS: /user/srowen/user24
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh2.srowen.com:8880 user25.ipynb Spark: cdh2.srowen.com:4044 (redirects) HDFS: /user/srowen/user25	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh2.srowen.com:8880 user26.ipynb Spark: cdh2.srowen.com:4045 (redirects) HDFS: /user/srowen/user26
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh2.srowen.com:8880 user27.ipynb Spark: cdh2.srowen.com:4046 (redirects) HDFS: /user/srowen/user27	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh2.srowen.com:8880 user28.ipynb Spark: cdh2.srowen.com:4047 (redirects) HDFS: /user/srowen/user28

Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh3.srowen.com:8880 user31.ipynb Spark: cdh3.srowen.com:4040 (redirects) HDFS: /user/srowen/user31	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh3.srowen.com:8880 user32.ipynb Spark: cdh3.srowen.com:4041 (redirects) HDFS: /user/srowen/user32
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh3.srowen.com:8880 user33.ipynb Spark: cdh3.srowen.com:4042 (redirects) HDFS: /user/srowen/user33	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh3.srowen.com:8880 user34.ipynb Spark: cdh3.srowen.com:4043 (redirects) HDFS: /user/srowen/user34
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh3.srowen.com:8880 user35.ipynb Spark: cdh3.srowen.com:4044 (redirects) HDFS: /user/srowen/user35	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh3.srowen.com:8880 user36.ipynb Spark: cdh3.srowen.com:4045 (redirects) HDFS: /user/srowen/user36
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh3.srowen.com:8880 user37.ipynb Spark: cdh3.srowen.com:4046 (redirects) HDFS: /user/srowen/user37	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh3.srowen.com:8880 user38.ipynb Spark: cdh3.srowen.com:4047 (redirects) HDFS: /user/srowen/user38

Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh4.srowen.com:8880 user41.ipynb Spark: cdh4.srowen.com:4040 (redirects) HDFS: /user/srowen/user41	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh4.srowen.com:8880 user42.ipynb Spark: cdh4.srowen.com:4041 (redirects) HDFS: /user/srowen/user42
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh4.srowen.com:8880 user43.ipynb Spark: cdh4.srowen.com:4042 (redirects) HDFS: /user/srowen/user43	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh4.srowen.com:8880 user44.ipynb Spark: cdh4.srowen.com:4043 (redirects) HDFS: /user/srowen/user44
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh4.srowen.com:8880 user45.ipynb Spark: cdh4.srowen.com:4044 (redirects) HDFS: /user/srowen/user45	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh4.srowen.com:8880 user46.ipynb Spark: cdh4.srowen.com:4045 (redirects) HDFS: /user/srowen/user46
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh4.srowen.com:8880 user47.ipynb Spark: cdh4.srowen.com:4046 (redirects) HDFS: /user/srowen/user47	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh4.srowen.com:8880 user48.ipynb Spark: cdh4.srowen.com:4047 (redirects) HDFS: /user/srowen/user48

Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh5.srowen.com:8880 user51.ipynb Spark: cdh5.srowen.com:4040 (redirects) HDFS: /user/srowen/user51	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh5.srowen.com:8880 user52.ipynb Spark: cdh5.srowen.com:4041 (redirects) HDFS: /user/srowen/user52
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh5.srowen.com:8880 user53.ipynb Spark: cdh5.srowen.com:4042 (redirects) HDFS: /user/srowen/user53	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh5.srowen.com:8880 user54.ipynb Spark: cdh5.srowen.com:4043 (redirects) HDFS: /user/srowen/user54
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh5.srowen.com:8880 user55.ipynb Spark: cdh5.srowen.com:4044 (redirects) HDFS: /user/srowen/user55	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh5.srowen.com:8880 user56.ipynb Spark: cdh5.srowen.com:4045 (redirects) HDFS: /user/srowen/user56
Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh5.srowen.com:8880 user57.ipynb Spark: cdh5.srowen.com:4046 (redirects) HDFS: /user/srowen/user57	Hue: cdh4.srowen.com:8888 srowen/srowen ipython: cdh5.srowen.com:8880 user58.ipynb Spark: cdh5.srowen.com:4047 (redirects) HDFS: /user/srowen/user58