

Big Data: Train the trainer

Apache Spark & Hadoop

27 June – 1 July 2016

Dr.Thanachart Numnonda
IMC Institute
thanachart@imcinstitute.com

Outline

- Launch Azure Instance
- Install Docker on Ubuntu
- Pull Cloudera QuickStart to the docker
- HDFS
- HBase
- Hive
- Spark
- Spark SQL
- Spark Streaming
- Spark Mllib
- Kafka

Cloudera Quickstart

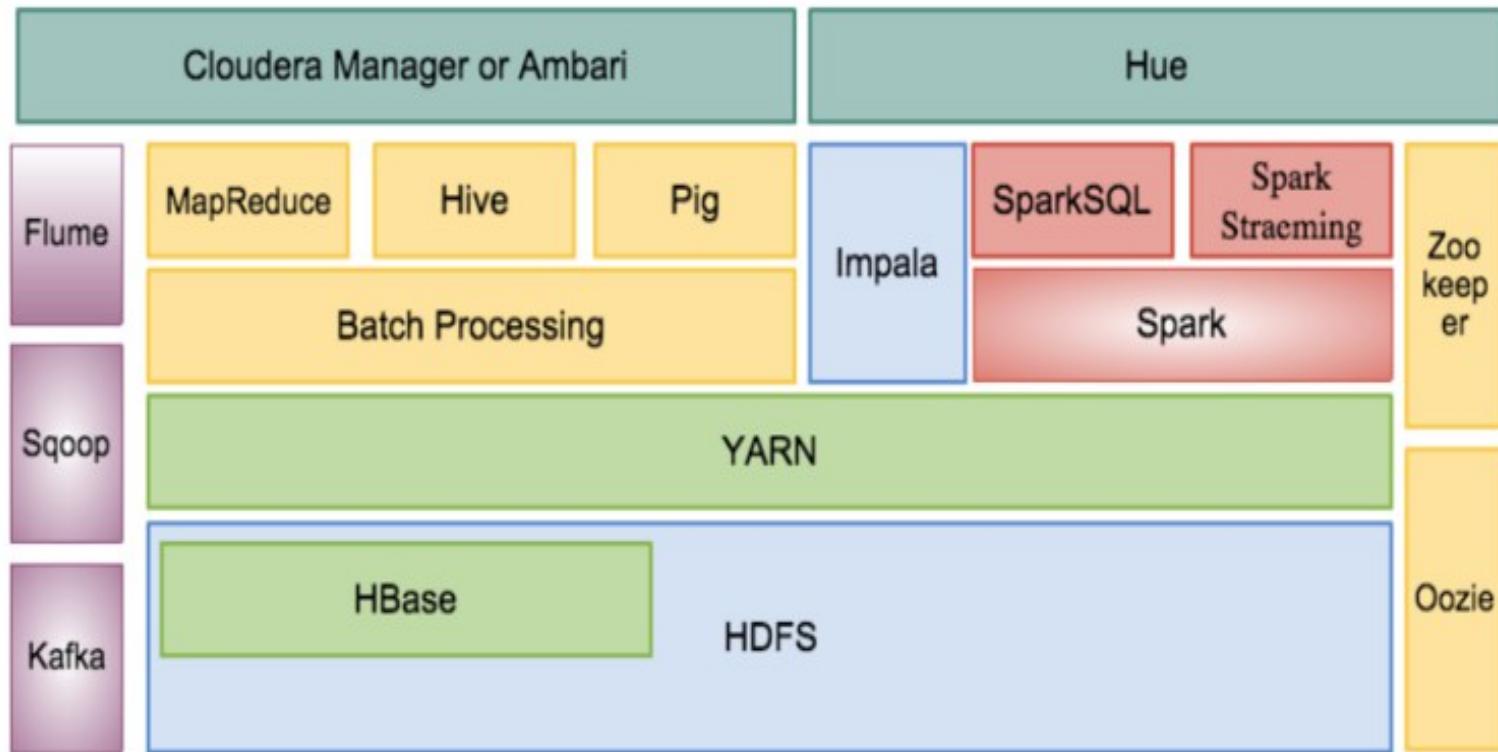
This lab will use an Azure virtual server to run Cloudera QuickStart VM which can be downloaded from:

<http://www.cloudera.com/content/www/en-us/downloads.html>

The screenshot shows a landing page for downloading Cloudera Enterprise software. The background features a sunset or sunrise over clouds. The main title is "Download Cloudera Enterprise" with the subtitle "Local, On Premise, or Cloud-based Apache Hadoop Management". Below this, there are three blue rectangular cards, each representing a different product:

- QuickStart VM**: Features an icon of two monitors. Text: "Get Started on your local machine using a QuickStart VM." Buttons: "DOWNLOAD NOW" and "Learn More".
- Cloudera Manager**: Features an icon of a gear. Text: "A unified interface to manage your enterprise data hub. Express and Enterprise editions available." Button: "DOWNLOAD NOW".
- Cloudera Director**: Features an icon of a cloud. Text: "Self-service, reliable experience for CDH and Cloudera Enterprise in the cloud". Button: "DOWNLOAD NOW".

Hadoop Ecosystem



Hands-On: Launch a virtual server on Microsoft Azure

(Note: You can skip this session if you use your own computer or another cloud service)

Sign up for Visual Studio Dev Essential to get free Azure credit.

Showing: Visual Studio Dev Essentials

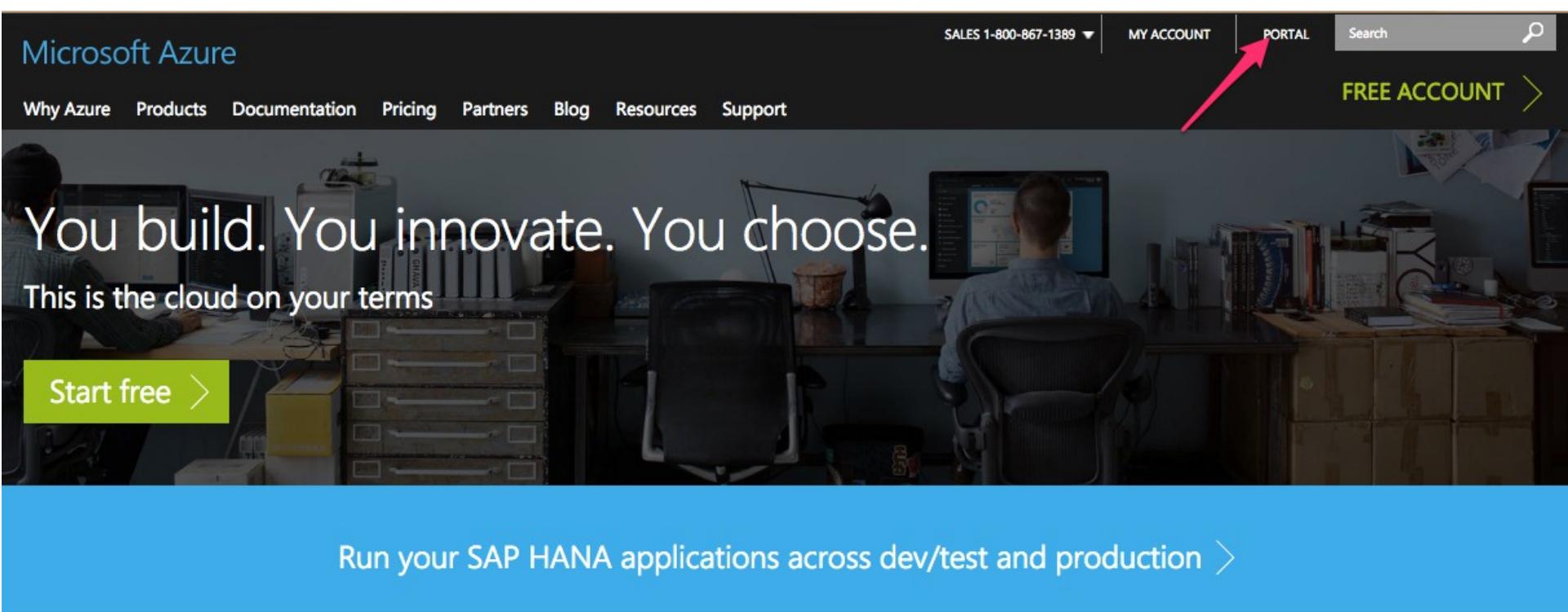
Visual Studio Dev Essentials



Featured (6)

 Visual Studio Community Full-featured, extensible IDE Free for individuals, open source or small teams. Create apps for Windows, iOS, Andro... See more	 Visual Studio Code Modern lightweight editor A powerful, streamlined code editor for your favorite platform - Linux, Mac OS X, and... Windows	 Visual Studio Team Services Basic level Free Git repos, Agile planning tools and hosted builds, for any language – it's the perfect... complement to your IDE	 Azure \$25 monthly credit for 1 year Your own personal sandbox for dev/test! VMs, cloud services, and more. Credit cannot be... applied to existing Azure	 Xamarin University Training Free on-demand access Build native iOS and Android apps in C# with expert getting-started videos (subset of class... videos and materials)	 Pluralsight 3-month subscription World-class training taught by an elite group of industry leaders.
Download 	Download 	Get started 	Activate 	Get Code 	Get Code 

Sign in to Azure Portal



The screenshot shows the Microsoft Azure homepage. At the top, there is a dark header bar with the "Microsoft Azure" logo on the left. On the right side of the header, there are links for "SALES 1-800-867-1389", "MY ACCOUNT", and "PORTAL". A red arrow points from the bottom right towards the "PORTAL" link. Below the header, there is a large banner with the text "You build. You innovate. You choose." and "This is the cloud on your terms". To the left of the banner is a green button labeled "Start free >". In the center of the banner, there is a photograph of a person working at a desk with multiple monitors. Below the banner, there is a blue footer bar with the text "Run your SAP HANA applications across dev/test and production >".

Microsoft Azure ▾

Search resources

New dashboard Edit dashboard Share Fullscreen Clone Delete

contact@imcinstitute....
DEFAULT DIRECTORY

Dashboard

All resources ALL SUBSCRIPTIONS

- hdp41n71a7p
- imclabstorage
- egahdpstorage

Service health MY RESOURCES



Marketplace

Subscriptions Forecast expenses and costs to optimize your apps

Help + support

Feedback

Information icons

The Microsoft Azure dashboard interface. It features a top navigation bar with search, resource management, and account information. Below is a main dashboard area with a sidebar of service icons. The central part includes a 'Service health' section with a world map of resource locations, and four main tiles: 'Marketplace', 'Subscriptions', 'Help + support', and 'Feedback'. At the bottom, there are three large blue circular icons with white letter 'i' symbols.

Virtual Server

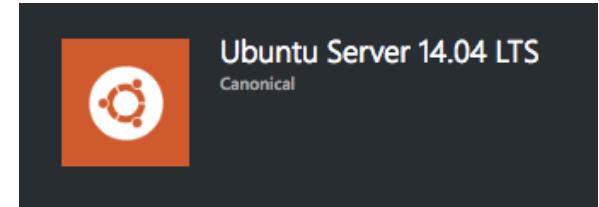
This lab will use an Azure virtual server to install a Cloudera Quickstart Docker using the following features:

Ubuntu Server 14.04 LTS

DS3_V2 Standard 4 Core, 14 GB memory, 28 GB SSD

Select New => Virtual Machines => Virtual Machines

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various icons and a search bar labeled "Search the marketplace". Below the search bar is a "MARKETPLACE" section with a "Virtual Machines" item highlighted by a pink arrow. The main area is titled "Virtual Machines" and contains a "FEATURED APPS" section. It lists four items: "Windows Server 2012 R2 Datacenter", "Ubuntu Server 14.04 LTS", "SQL Server 2014 Enterprise on Windows Server 2012 R2", and "SharePoint 2013 HA Farm". The "Ubuntu Server 14.04 LTS" item is also highlighted with a pink arrow.



Ubuntu Server 14.04.4 LTS (amd64 20160516) for Microsoft popular Linux for cloud environments. Updates and patches until 2019-04-17. Ubuntu Server is the perfect virtual machine for web applications to NoSQL databases and Hadoop. For more information, visit [the Canonical website](#). You can also use Juju to deploy your workloads.

Legal Terms

By clicking the Create button, I acknowledge that I am getting the [legal terms](#) of Canonical apply to it. Microsoft does not accept responsibility for the content of the Canonical site. Also see the [privacy statement](#) from Canonical.

This screenshot shows the "Create" blade for creating a new virtual machine. At the top, there are social media sharing options. Below them is a dropdown menu labeled "Select a deployment model" with "Resource Manager" selected. A pink arrow points to this dropdown. At the bottom is a large blue "Create" button.

On the Basics page, enter:

- a name for the VM
- a username for the Admin User
- the Authentication Type set to password
- a password
- a resource group name

Microsoft Azure New > Virtual Machines > Ubuntu Server 14.04 LTS > Create virtual machine > Basics

Create virtual machine

Basics

1 Basics Configure basic settings

2 Size Choose virtual machine size

3 Settings Configure optional features

4 Summary Ubuntu Server 14.04 LTS

Basics

* Name clouderadocker ✓

* User name imcinstiute ✓

* Authentication type Password SSH public key

* Password ✓

Subscription Developer Program Benefit

* Resource group ⓘ

Create new Use existing

OK

Browse >

Choose DS3_v2 Standard

Microsoft Azure Virtual Machines > Ubuntu Server 14.04 LTS > Create virtual machine > Choose a size

Create virtual machine

Choose a size

Browse the available sizes and their features

world's most available loads from Azure and

al and that software.

1 Basics Done ✓

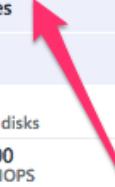
2 Size Choose virtual machine size >

3 Settings Configure optional features >

4 Summary Ubuntu Server 14.04 LTS >

DS1_V2 Standard	DS2_V2 Standard	DS3_V2 Standard
1 Core	2 Cores	4 Cores
3.5 GB	7 GB	14 GB
2 Data disks	4 Data disks	8 Data disks
3200 Max IOPS	6400 Max IOPS	12800 Max IOPS
7 GB Local SSD	14 GB Local SSD	28 GB Local SSD
Load balancing	Load balancing	Load balancing
Auto scale	Auto scale	Auto scale
Premium disk suppo...	Premium disk suppo...	Premium disk suppo...
67.70 USD/MONTH (ESTIMATED)	128.71 USD/MONTH (ESTIMATED)	257.42 USD/MONTH (ESTIMATED)

Select



Microsoft Azure New > Virtual Machines > Ubuntu Server 14.04 LTS > Create virtual machine

Create virtual machine

1 Basics Done ✓

2 Size Done ✓

3 Settings Configure optional features >

4 Summary Ubuntu Server 14.04 LTS >

Settings

Storage

Disk type Standard Premium (SSD)

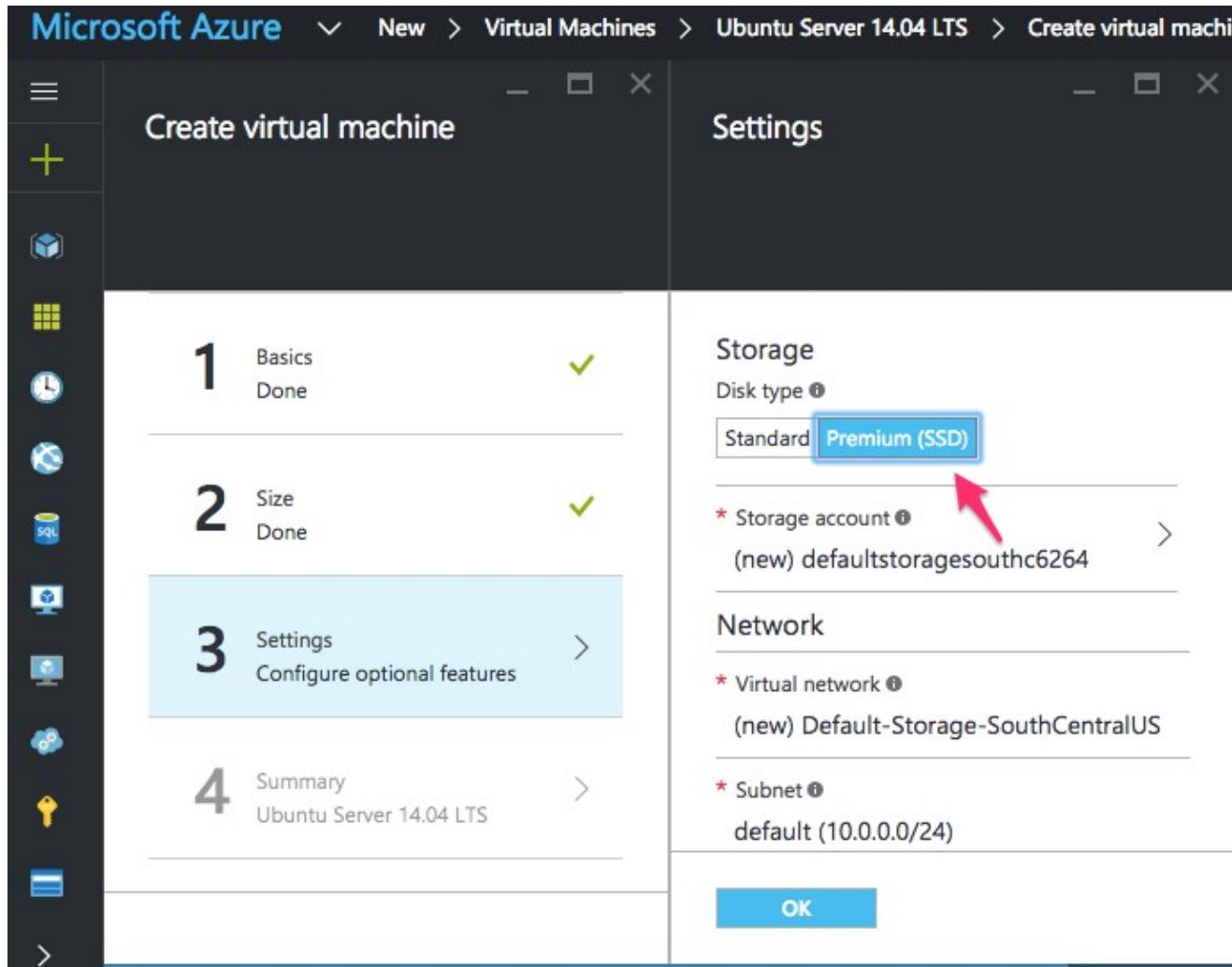
* Storage account (new) defaultstoragesouthc6264 >

Network

* Virtual network (new) Default-Storage-SouthCentralUS

* Subnet default (10.0.0.0/24)

OK



Microsoft Azure New > Virtual Machines > Ubuntu Server 14.04 LTS > Create virtual machine > Summary

Create virtual machine

Summary

Validation passed

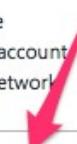
Basics

Subscription	Developer Program Benefit
Resource group	Default-MachineLearning-SouthCentralUS
Location	South Central US

Settings

Computer name	clouderadocker
User name	imcinstitute
Size	Standard DS11 v2
Disk type	Premium (SSD)
Storage account	(new) defaultmachinelearn2038
Virtual network	(new) Default-MachineLearning-SouthCentralUS
Subnet	(new) default (10.1.0.0/24)

OK



Microsoft Azure cluderadocker > Settings

cluderadocker

Virtual machine

Settings Connect Start Restart Stop Delete

Essentials

Resource group
[Default-MachineLearning-SouthCentralU...](#)

Status
Running

Location
South Central US

Subscription name
[Developer Program Benefit](#)

Subscription ID
59cbb519-5261-48e5-9825-9df96f8302a9

Computer name
cluderadocker

Operating system
Linux

Size
Standard DS11 v2 (2 cores, 14 GB memory)

Public IP address/DNS name label
[104.210.146.182/<none>](#)

Virtual network/subnet
[Default-MachineLearning-SouthCentralUS/...](#)

All settings →

Monitoring

Add tiles +

CPU percentage

100%

Settings

cluderadocker

Filter settings

SUPPORT + TROUBLESHOOTING

Troubleshoot

Audit logs

Resource health

Boot diagnostics

Reset password

Redeploy

New support request

GENERAL

Setting the inbound port for Hue (8888)

The screenshot shows two windows side-by-side in the Azure portal.

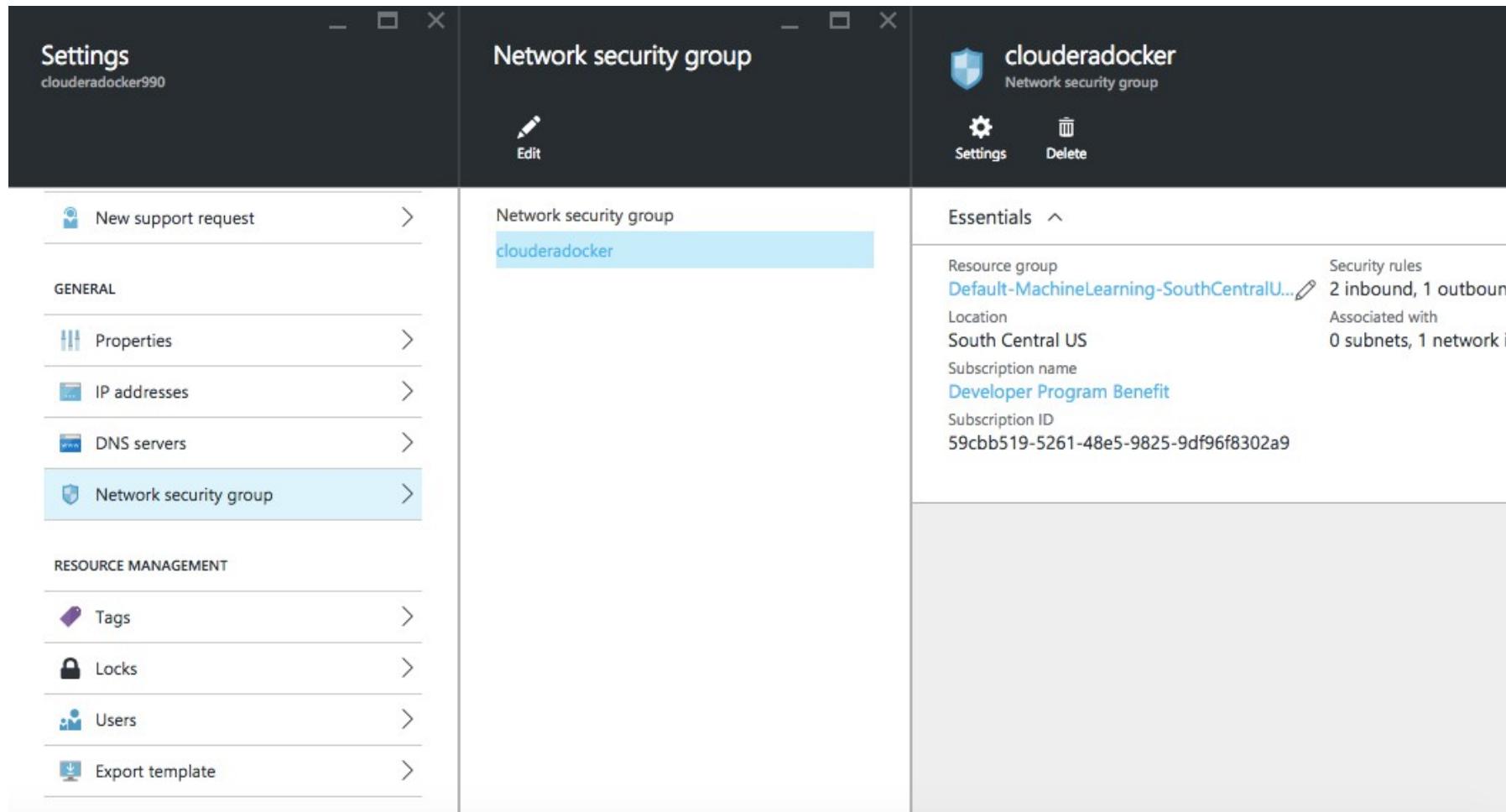
Left Window: Settings

- GENERAL**
 - Properties
 - Disks
 - Network interfaces** (highlighted in blue)
 - Availability set
 - Extensions
 - Size
- MONITORING**
 - Alert rules
 - Diagnostics
- RESOURCE MANAGEMENT**

Right Window: Network interfaces

Search bar: Search network interfaces

NAME	PUBLIC IP ADDRE...	PRIVATE IP ADDR...	SECURITY GROUP	...
clouderadocker990	104.210.146.182	10.1.0.4	clouderadocker	...



The screenshot shows three separate windows from the Microsoft Azure portal:

- Settings** window (left): Shows a list of options under "GENERAL" and "RESOURCE MANAGEMENT". The "Network security group" option is highlighted.
- Network security group** window (middle): Displays a list of network security groups, with "clouderadocker" selected. An "Edit" button is visible at the top.
- clouderadocker** window (right): A detailed view of the selected network security group. It includes:
 - Essentials** section: Resource group (Default-MachineLearning-SouthCentralU...), Location (South Central US), Subscription name (Developer Program Benefit), and Subscription ID (59cbb519-5261-48e5-9825-9df96f8302a9).
 - Security rules** section: 2 inbound, 1 outbound.
 - Associated with**: 0 subnets, 1 network interface.

DEPLOY DIRECTORY

Inbound security rules

clouderadocker

Add **Default rules**

Filter settings

SUPPORT + TROUBLESHOOTING

- Audit logs >
- New support request >

GENERAL

- Properties >
- Inbound security rules** > (highlighted)
- Outbound security rules >
- Network interfaces >
- Subnets >

Search inbound security rules

PRIORITY	NAME	SOURCE	DESTINATION	SERVICE
1000	default-allow-ssh	Any	Any	SSH (TCP/22)

Inbound security rules

Default rules

Add

Search inbound security rules

PRIORITY	NAME	SOURCE	DESTINATION	SERVICE
1000	default-allow-ssh	Any	hue Default-MachineLearning-SouthCentralUS	

Name: hue

Priority: 1010

Source: Any

Protocol: Any

Source port range: *

Destination: Any

Destination port range: 8888

Action:

Get the IP address

The screenshot displays two side-by-side views in the Azure portal:

Left View: Virtual Machine Details

- Resource group:** Default-MachineLearning-SouthCentralU...
- Status:** Running
- Location:** South Central US
- Subscription name:** Developer Program Benefit
- Subscription ID:** 59ccb519-5261-48e5-9825-9df96f8302a9
- Computer name:** cluderadocker
- Operating system:** Linux
- Size:** Standard DS11 v2 (2 cores, 14 GB memory)
- Public IP address/DNS name label:** 104.210.146.182/<none>
- Virtual network/subnet:** Default-MachineLearning-SouthCentralUS/...

Right View: Public IP Address Configuration

- Resource group:** Default-MachineLearning-SouthCentralU...
- Location:** South Central US
- Subscription name:** Developer Program Benefit
- Subscription ID:** 59ccb519-5261-48e5-9825-9df96f8302a9
- IP address:** 104.210.146.182
- DNS name:** -
- Associated to:** cluderadocker990
- Virtual machine:** cluderadocker

Connect to an instance from Mac/Linux

```
ssh -i ~/.ssh/id_rsa imcinstigate@104.210.146.182
```

WARNING! Your environment specifies an invalid locale.

This can affect your user experience significantly, including the ability to manage packages. You may install the locales by running:

```
sudo apt-get install language-pack-UTF-8  
or  
sudo locale-gen UTF-8
```

To see all available language packs, run:

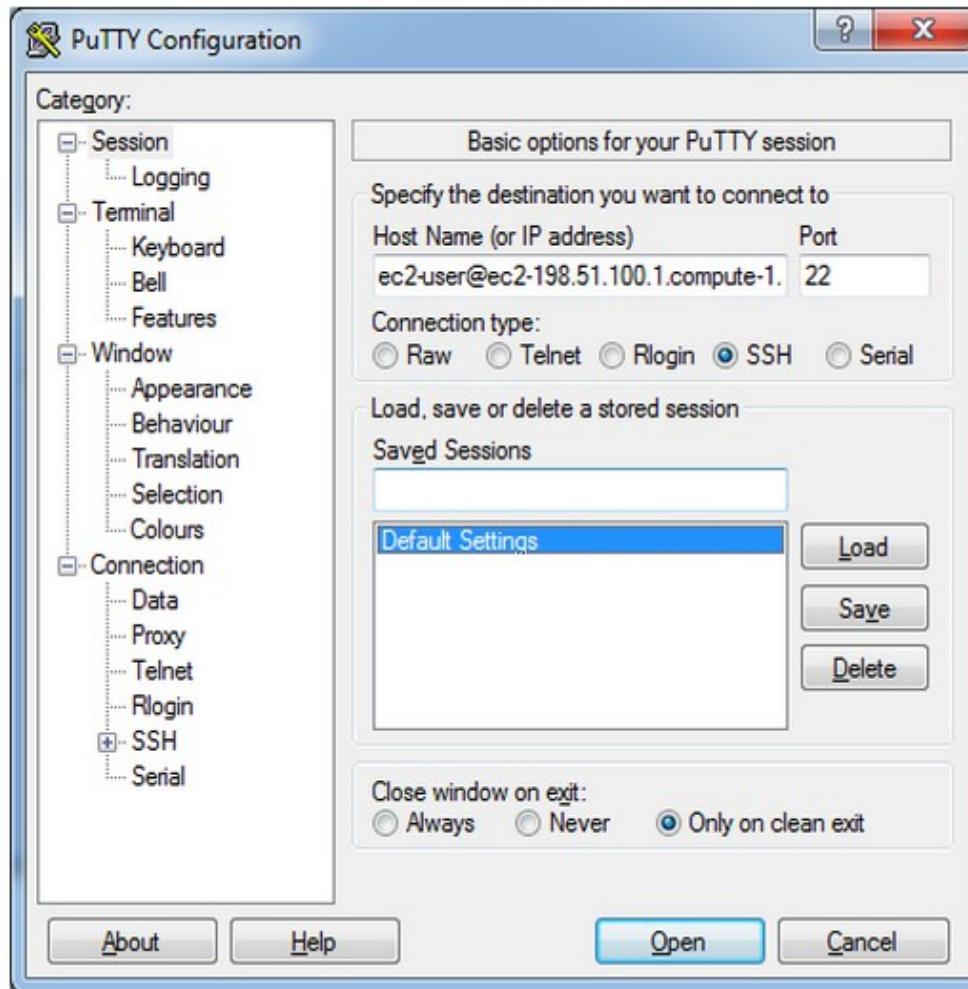
```
apt-cache search "^language-pack-[a-z][a-z]$"
```

To disable this message for all users, run:

```
sudo touch /var/lib/cloud/instance/locale-check.skip
```

```
imcinstigate@clouderadocker:~$ █
```

Connect to an instance from Windows using Putty



Hands-On: Installing Cloudera Quickstart on Docker Container

Installation Steps

- Update OS
- Install Docker
- Pull Cloudera Quickstart
- Run Cloudera Quickstart
- Run Cloudera Manager

Update OS (Ubuntu)

- Command: sudo apt-get update

```
ubuntu@ip-172-31-30-238:~$ sudo apt-get update
Ign http://us-east-1.ec2.archive.ubuntu.com trusty InRelease
Get:1 http://us-east-1.ec2.archive.ubuntu.com trusty-updates InRelease [65.9 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com trusty-backports InRelease [65.9 kB]
Hit http://us-east-1.ec2.archive.ubuntu.com trusty Release.gpg
Hit http://us-east-1.ec2.archive.ubuntu.com trusty Release
Get:3 http://security.ubuntu.com trusty-security InRelease [65.9 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/main Sources [277 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/restricted Sources [535
2 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/universe Sources [156 k
B]
Get:7 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/multiverse Sources [593
9 B]
Get:8 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/main amd64 Packages [78
1 kB]
```

Docker Installation

- Command: sudo apt-get install docker.io

```
ubuntu@ip-172-31-30-238:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  aufs-tools cgroup-lite git git-man liberror-perl
Suggested packages:
  btrfs-tools debootstrap lxc rinse git-daemon-run git-daemon-sysvinit git-doc
  git-el git-email git-gui gitk gitweb git-arch git-bzr git-cvs git-mediawiki
  git-svn
The following NEW packages will be installed:
  aufs-tools cgroup-lite docker.io git git-man liberror-perl
0 upgraded, 6 newly installed, 0 to remove and 84 not upgraded.
Need to get 8150 kB of archives.
After this operation, 51.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/ trusty/universe aufs-tools amd
64 1:3.2+20130722-1.1 [92.3 kB]
```

Pull Cloudera Quickstart

- Command: sudo docker pull cloudera/quickstart:latest

```
ubuntu@ip-172-31-30-238:~$ sudo docker pull cloudera/quickstart:latest
latest: Pulling from cloudera/quickstart
2cda82941cb7: Already exists
Digest: sha256:f91bee4cdfa2c92ea3652929a22f729d4d13fc838b00f120e630f91c941acb63
Status: Downloaded newer image for cloudera/quickstart:latest
ubuntu@ip-172-31-30-238:~$ █
```

Show docker images

- Command: sudo docker images

```
ubuntu@ip-172-31-30-238:~$ sudo docker images
REPOSITORY          TAG      IMAGE ID      CREATED
VIRTUAL SIZE
cloudera/quickstart    latest  2cda82941cb7  9 weeks ago
6.336 GB
```

Run Cloudera quickstart

- Command: `sudo docker run --hostname=quickstart.cloudera --privileged=true -t -i [OPTIONS] [IMAGE] /usr/bin/docker-quickstart`

Example: `sudo docker run --hostname=quickstart.cloudera --privileged=true -t -i -p 8888:8888 cloudera/quickstart /usr/bin/docker-quickstart`

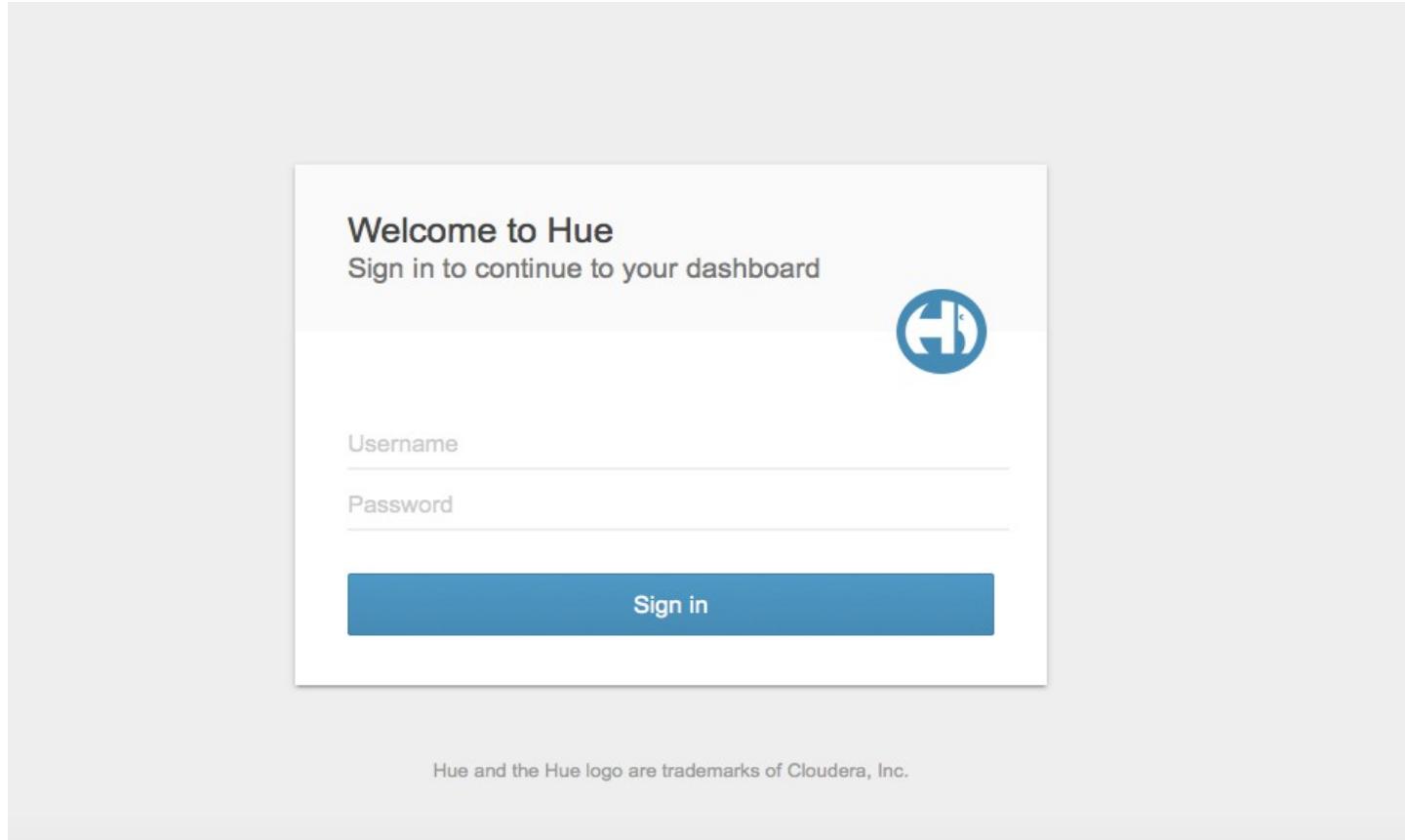
```
ubuntu@ip-172-31-30-238:~$ sudo docker run --hostname=quickstart.cloudera --privileged=true -t -i -p 8888:8888 -p 7180:7180 cloudera/quickstart /usr/bin/docker-quickstart
Starting mysqld: [ OK ]  
  
if [ "$1" == "start" ] ; then
  if [ "${EC2}" == 'true' ] ; then
    FIRST_BOOT_FLAG=/var/lib/cloudera-quickstart/.ec2-key-installed
    if [ ! -f "${FIRST_BOOT_FLAG}" ] ; then
      METADATA_API=http://169.254.169.254/latest/meta-data
      KEY_URL=${METADATA API}/public-keys/0/openssh-key
```

Docker commands:

- *docker images*
- *docker ps*
- *docker attach id*
- *docker kill id*
- *Exit from container*
 - *exit (exit & kill the running image)*
 - *Ctrl-P, Ctrl-Q (exit without killing the running image)*

Login to Hue

`http://104.210.146.182:8888`
(username = cloudera; password = cloudera)



Quick Start Wizard - Hue™ 3.9.0 - The Hadoop UI

Step 1: Check Configuration

Step 2: Examples

Step 3: Users

Step 4: Go!

Checking current configuration

Configuration files located in [`/etc/hue/conf.empty`](#)

All OK. Configuration check passed.

[Back](#)[Next](#)

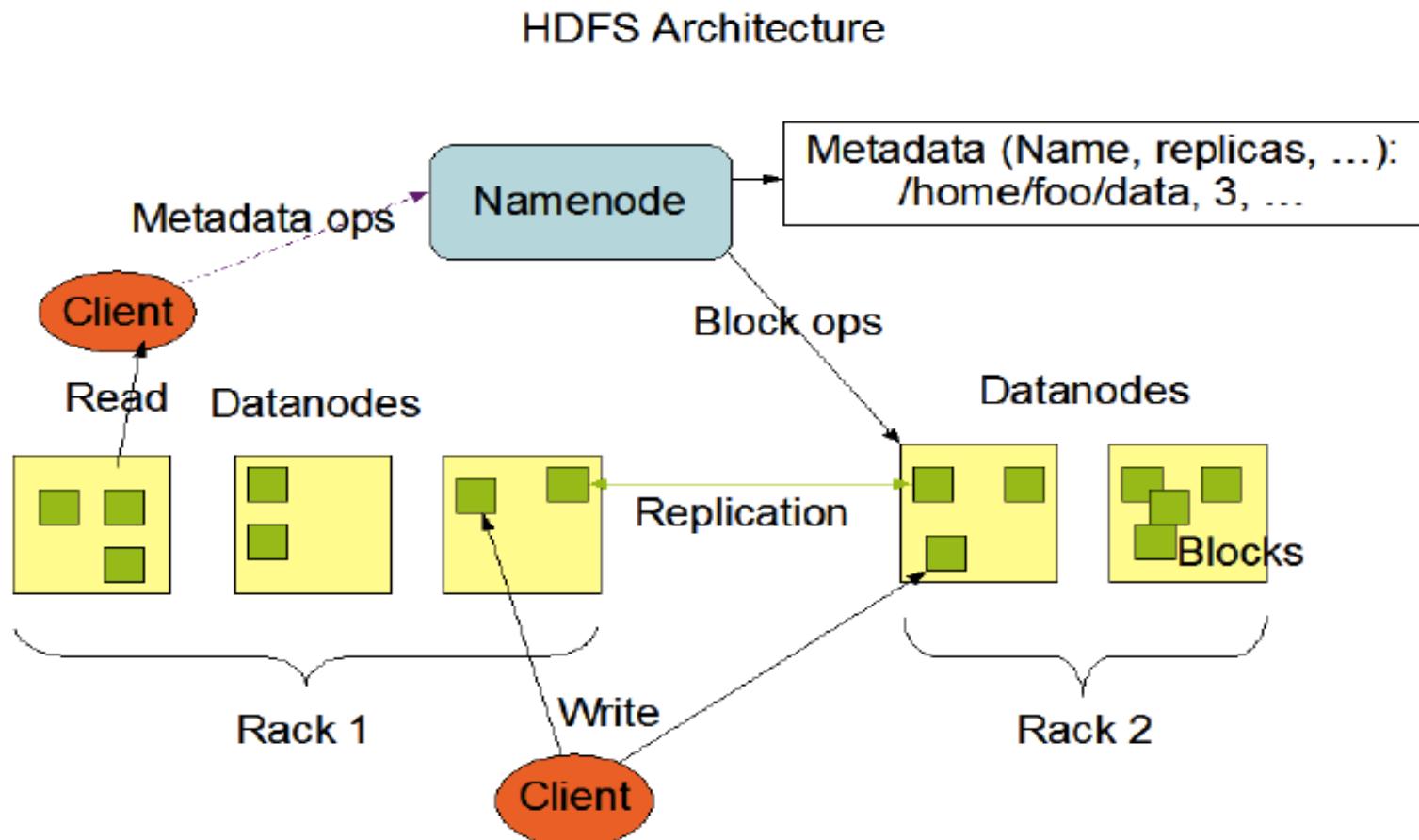
Hue and the Hue logo are trademarks of Cloudera, Inc.

Hands-On: Importing/Exporting Data to HDFS

HDFS

- Default storage for the Hadoop cluster
- Data is distributed and replicated over multiple machines
- Designed to handle very large files with streaming data access patterns.
- NameNode/DataNode
- Master/slave architecture (1 master 'n' slaves)
- Designed for large files (64 MB default, but configurable) across all the nodes

HDFS Architecture



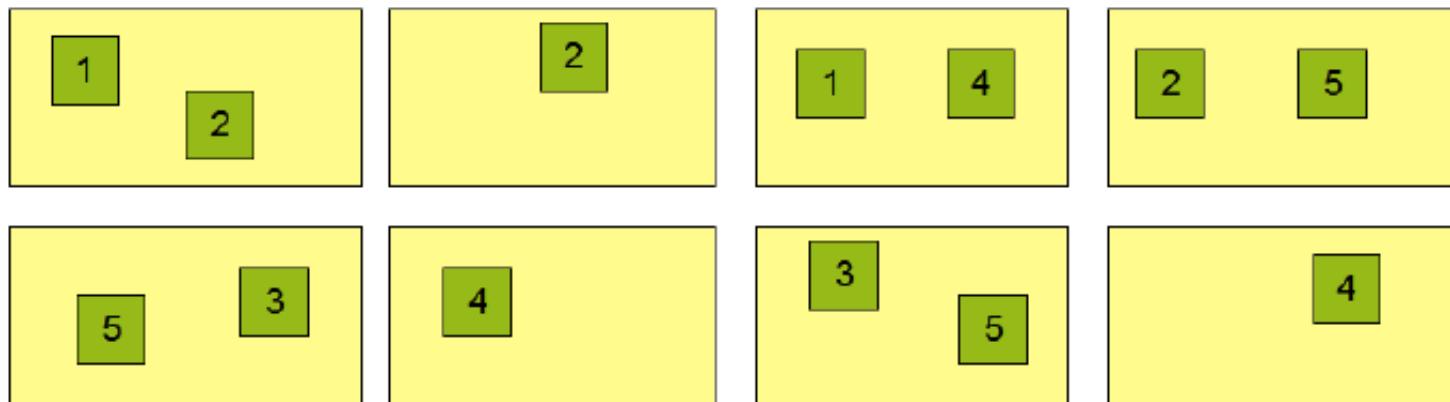
Source Hadoop: Shashwat Shriparv

Data Replication in HDFS

Block Replication

```
Namenode (Filename, numReplicas, block-ids, ...)  
/users/sameerp/data/part-0, r:2, {1,3}, ...  
/users/sameerp/data/part-1, r:3, {2,4,5}, ...
```

Datanodes



Source Hadoop: Shashwat Shriparv

How does HDFS work?

A file we want to store on HDFS ...

600 MB

We're raising the question because no one else wants to, because no one else wants to say what needs to be said.

And let's be real, it's the two-ton elephant in the room with nearly every other star's name on the trade rumor radar these days.

We've read over and over again about Nash refusing to ask for a trade, refusing to play the game that so many others have late in their careers.

Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

How does HDFS work?

HDFS Splits file into **blocks** ...

256 MB

We're raising the question because no one else wants to, because no one else wants to say what needs to be said.

256 MB

And let's be real, it's the two-ton elephant in the room with nearly every other star's name on the trade rumor radar these days.

88 MB

We've read over and over again about Nash refusing to play the game that so many others have late in their careers.

Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

How does HDFS work?

HDFS will create **3replicas** of each block ...

3 copies

We're raising the question because no one else wants to, because no one else wants to say what needs to be said.

3 copies

And let's be real, it's the two-ton elephant in the room with nearly every other star's name on the trade rumor radar these days.

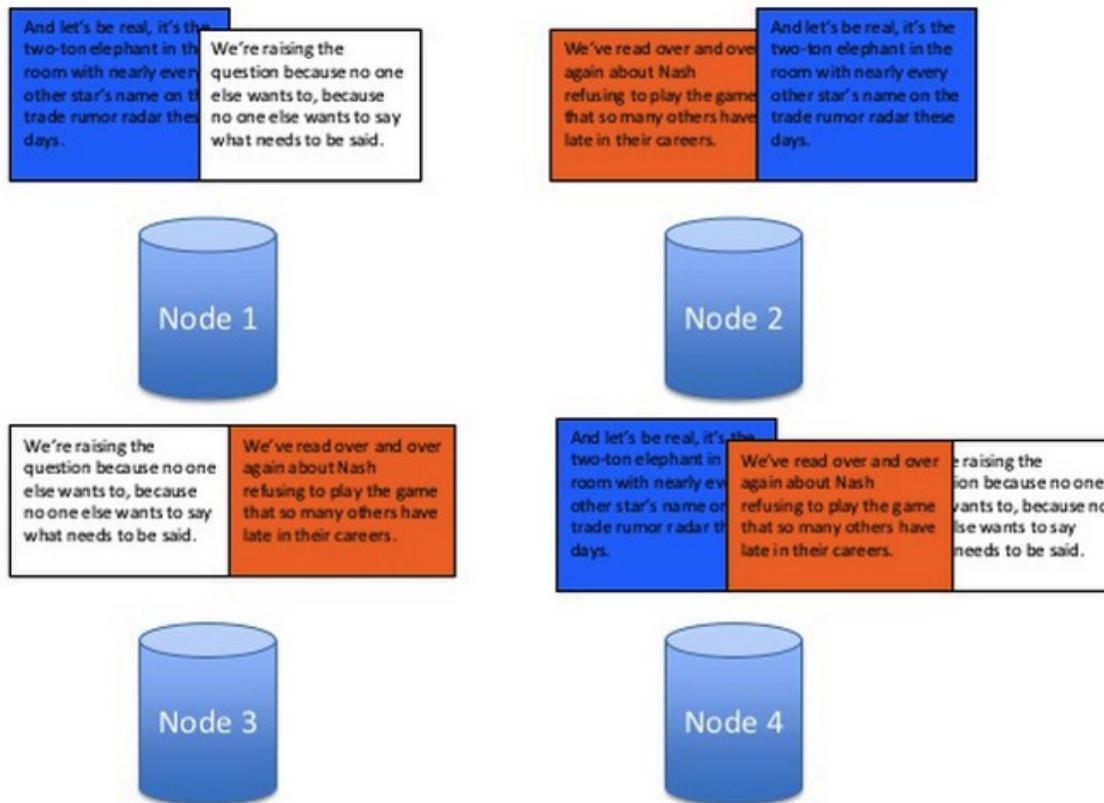
3 copies

We've read over and over again about Nash refusing to play the game that so many others have late in their careers.

Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

How does HDFS work?

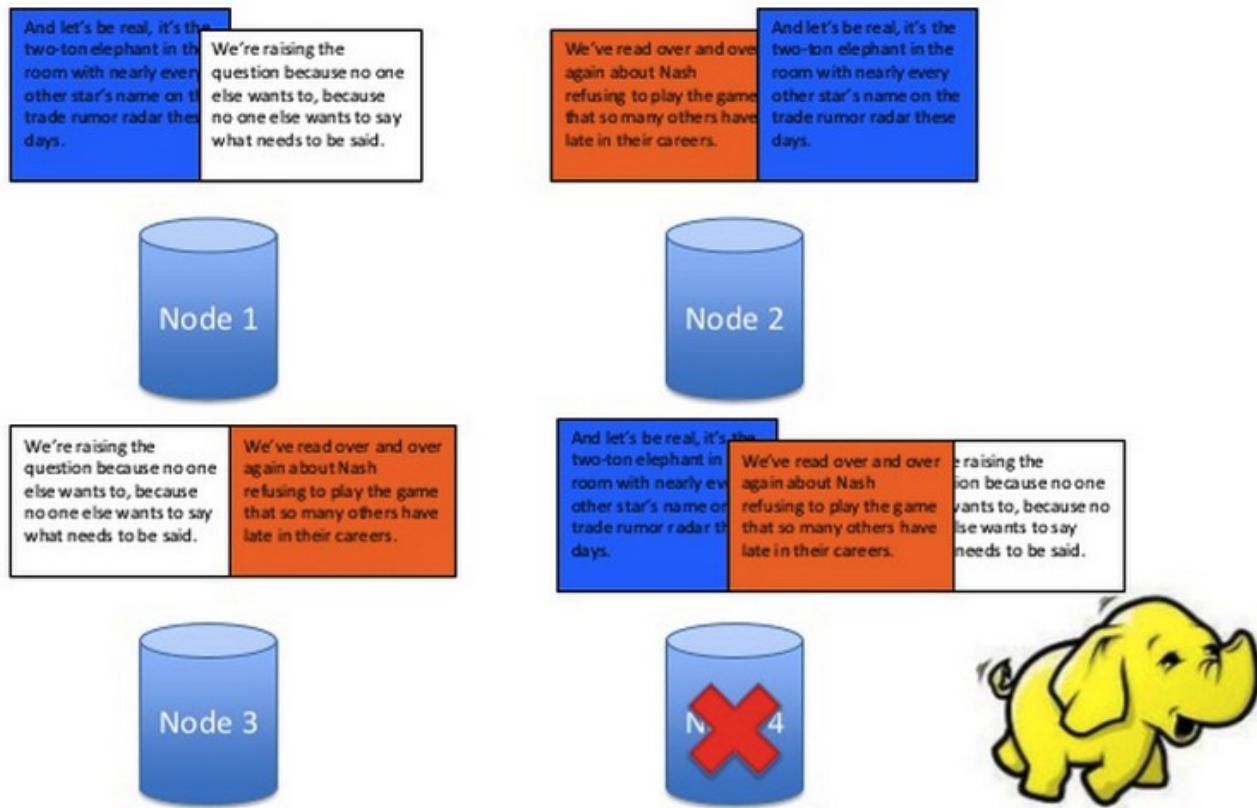
HDFS distributes these replicas across the cluster ...



Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

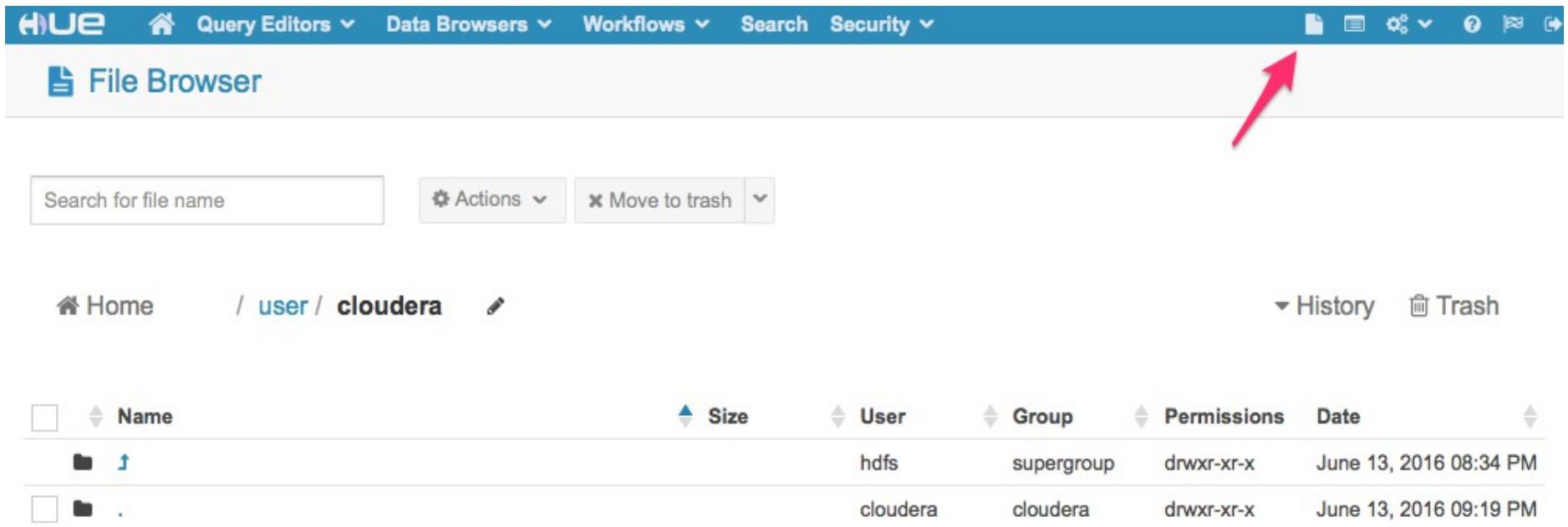
How does HDFS work?

If a node goes down, we have copies elsewhere



Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

Review file in Hadoop HDFS using File Browse



The screenshot shows the Hue File Browser interface. At the top, there is a navigation bar with links for Home, Query Editors, Data Browsers, Workflows, Search, Security, and various system icons. Below the navigation bar, the title "File Browser" is displayed next to a folder icon. On the left, there is a search bar labeled "Search for file name" and a set of buttons for "Actions" and "Move to trash". The main area shows a file listing for the path "/user/cloudera". The columns in the table are Name, Size, User, Group, Permissions, and Date. Two entries are listed:

Name	Size	User	Group	Permissions	Date
..		hdfs	supergroup	drwxr-xr-x	June 13, 2016 08:34 PM
.		cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:19 PM

Create a new directory name as: **input & output**

The screenshot shows the Hue File Browser interface. At the top, there is a navigation bar with links for Home, Query Editors, Data Browsers, Workflows, Search, and Security. Below the navigation bar, the title "File Browser" is displayed. On the left, there is a sidebar with "Actions" and "Move to trash" dropdown menus, and "Upload" and "New" buttons. The main area shows a list of files and directories under the path "/user/cloudera". The list includes two entries: "hdfs" (User: supergroup, Group: supergroup, Permissions: drwxr-xr-x, Date: June 13, 2016 08:34 PM) and "cloudera" (User: cloudera, Group: cloudera, Permissions: drwxr-xr-x, Date: June 13, 2016 09:19 PM). To the right of the list, there are "History" and "Trash" buttons. A red arrow points to the "New" button, which has a dropdown menu open. The dropdown menu contains two options: "File" and "Directory". The "Directory" option is highlighted with a red box.

Size	User	Group	Permissions	Date
	hdfs	supergroup	drwxr-xr-x	June 13, 2016 08:34 PM
	cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:19 PM

Directory Name

Cancel Create

HUE  Query Editors ▾ Data Browsers ▾ Workflows ▾ Search Security ▾       

File Browser

Search for file name Actions ▾  Move to trash ▾

 Home / user / cloudera  History  Trash

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	 		hdfs	supergroup	drwxr-xr-x	June 13, 2016 08:34 PM
<input type="checkbox"/>	 .		cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:21 PM
<input type="checkbox"/>	 input		cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:20 PM
<input type="checkbox"/>	 output		cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:21 PM

Upload a local file to HDFS

The screenshot shows the Hue File Browser interface. At the top, there's a navigation bar with links for Query Editors, Data Browsers, Workflows, Search, and Security. Below the navigation bar, the title "File Browser" is displayed. On the left, there's a search bar and some actions like "Actions" and "Move to trash". The main area shows a directory structure under "/user/cloudera/input". On the right, there's a toolbar with an "Upload" button, which has a dropdown menu. The dropdown menu is open, showing two options: "Files" (with a file icon) and "Zip/Tgz/Bz2 fil" (with a zip file icon). A red arrow points from the text "Select files" in the upload dialog to the "Files" option in the dropdown menu. Below the toolbar, there's a table listing files with columns for Size, User, Group, Permissions, and Date.

Size	User	Group	Permissions	Date
	cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:21 PM
	cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:20 PM

Upload to /user/cloudera/input

Select files

or drag and drop them here

03_Suitability test.pdf

99% from 0.3MB x

HUE Home Query Editors Data Browsers Workflows Search Security

File Browser

Search for file name Actions Move to trash

Home / user / cloudera / input

History Trash

	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	..		cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:21 PM
<input type="checkbox"/>	.		cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:22 PM
<input type="checkbox"/>	03_Suitability test.pdf	336.8 KB	cloudera	cloudera	-rw-r--r--	June 13, 2016 09:22 PM

Hands-On: Connect to a master node via SSH

SSH Login to a master node

```
THANACHARTs-MacBook-Air:elastic-mapreduce-cli THANACHART$ ssh -i "imchadoop.pem" ub  
untu@ec2-54-201-147-59.us-west-2.compute.amazonaws.com  
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-74-generic x86_64)
```

* Documentation: <https://help.ubuntu.com/>

System information as of Sun Mar 27 09:08:01 UTC 2016

System load: 0.0	Processes: 135
Usage of /: 27.6% of 29.40GB	Users logged in: 0
Memory usage: 24%	IP address for eth0: 172.31.10.53
Swap usage: 0%	

Graph this data and manage this system at:
<https://landscape.canonical.com/>

Get cloud support with Ubuntu Advantage Cloud Guest:
<http://www.ubuntu.com/business/services/cloud>

*** System restart required ***

```
Last login: Sun Mar 27 09:08:01 2016 from node-io5.pool-125-24.dynamic.totbb.net  
ubuntu@ip-172-31-10-53:~$ █
```

Hadoop syntax for HDFS

Command	Syntax
Listing of files in a directory	<code>hadoop fs -ls /user</code>
Create a new directory	<code>hadoop fs -mkdir /user/guest/newdirectory</code>
Copy a file from a local machine to Hadoop	<code>hadoop fs -put C:\Users\Administrator\Downloads\localfile.csv /user/rajn/newdirectory/hadoopfile.txt</code>
Copy a file from Hadoop to a local machine	<code>hadoop fs -get /user/rajn/newdirectory/hadoopfile.txt C:\Users\Administrator\Desktop\</code>
Tail last few lines of a large file in Hadoop	<code>hadoop fs -tail /user/rajn/newdirectory/hadoopfile.txt</code>
View the complete contents of a file in Hadoop	<code>hadoop fs -cat /user/rajn/newdirectory/hadoopfile.txt</code>
Remove a complete directory from Hadoop	<code>hadoop fs -rm -r /user/rajn/newdirectory</code>
Check the Hadoop filesystem space utilization	<code>hadoop fs -du /</code>

Install wget

- Command: yum install wget

```
[root@quickstart /]# yum install wget
Loaded plugins: fastestmirror
Setting up Install Process
Determining fastest mirrors
epel/metalink | 13 kB     00:00
 * base: mirrors.evowise.com
 * epel: mirror.cogentco.com
 * extras: mirror.us.leaseweb.net
 * updates: mirror.cs.pitt.edu
base | 3.7 kB     00:00
base/primary_db | 4.7 MB     00:06
```

Download an example text file

Make your own directory at a master node to avoid mixing with others

```
$mkdir guest1
$cd guest1
$wget https://s3.amazonaws.com/imcbucket/input/pg2600.txt
```

```
--2016-03-27 09:58:48-- https://s3.amazonaws.com/imcbucket/input/pg2600.txt
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.19.187
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.19.187|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3291648 (3.1M) [text/plain]
Saving to: 'pg2600.txt'

100%[=====>] 3,291,648 3.14MB/s in 1.0s

2016-03-27 09:58:50 (3.14 MB/s) - 'pg2600.txt' saved [3291648/3291648]
```

Upload Data to Hadoop

```
$hadoop fs -ls /user/cloudera/input  
$hadoop fs -rm /user/cloudera/input/*  
$hadoop fs -put pg2600.txt /user/cloudera/input/  
$hadoop fs -ls /user/cloudera/input
```

```
[root@quickstart guest1]# hadoop fs -ls /user/cloudera/input  
Found 1 items  
-rw-r--r-- 1 root cloudera 3291648 2016-06-14 04:29 /user/cloudera/input/pg2600.txt  
[root@quickstart guest1]#
```



Lecture

Understanding HBase

Introduction

An open source, non-relational, distributed database



HBase is an open source, non-relational, distributed database modeled after Google's BigTable and is written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS (, providing BigTable-like capabilities for Hadoop. That is, it provides a fault-tolerant way of storing large quantities of sparse data.

HBase Features

- Hadoop database modelled after Google's Bigtable
- Column oriented data store, known as Hadoop Database
- Support random realtime CRUD operations (unlike HDFS)
- No SQL Database
- Opensource, written in Java
- Run on a cluster of commodity hardware

When to use HBase?

- When you need high volume data to be stored
- Un-structured data
- Sparse data
- Column-oriented data
- Versioned data (same data template, captured at various time, time-elapse data)
- When you need high scalability

Which one to use?

- HDFS
 - Only append dataset (no random write)
 - Read the whole dataset (no random read)
- HBase
 - Need random write and/or read
 - Has thousands of operation per second on TB+ of data
- RDBMS
 - Data fits on one big node
 - Need full transaction support
 - Need real-time query capabilities

HBase vs. RDBMS

	HBase	RDBMS
Hardware architecture	Similar to Hadoop. Clustered commodity hardware. Very affordable.	Typically large scalable multiprocessor systems. Very expensive.
Fault Tolerance	Built into the architecture. Lots of nodes means each is relatively insignificant. No need to worry about individual node downtime.	Requires configuration of the HW and the RDBMS with the appropriate high availability options.
Typical Database Size	Terabytes to Petabytes - hundred of millions to billions of rows.	Gigabytes to Terabytes – hundred of thousands to millions of rows.
Data Layout	A sparse, distributed, persistent, multidimensional sorted map.	Rows or column oriented.
Data Types	Bytes only.	Rich data type support.
Transactions	ACID support on a single row only	Full ACID compliance across rows and tables
Query Language	API primitive commands only, unless combined with Hive or other technology	SQL
Indexes	Row-Key only unless combined with other technologies such as Hive or IBM's BigSQL	Yes
Throughput	Millions of queries per second	Thousands of queries per second

- Given this RDBMS:

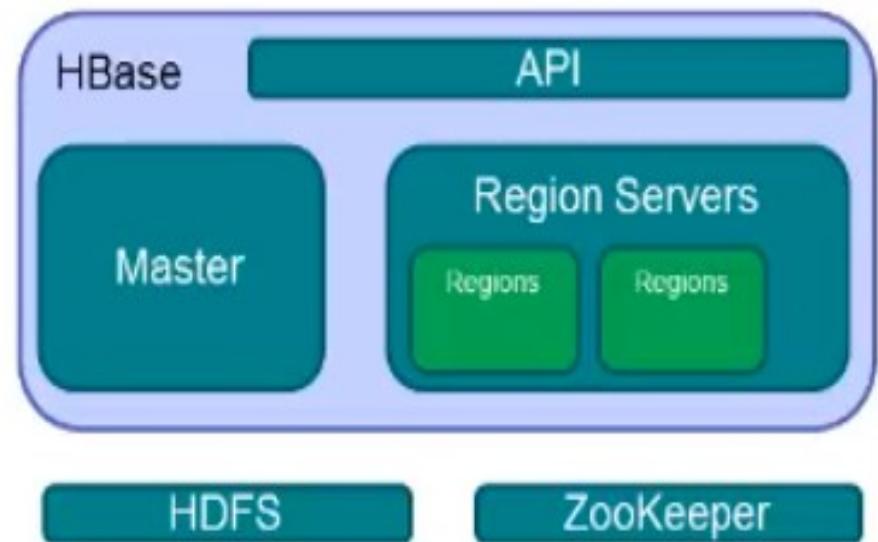
ID (Primary key)	Last name	First name	Password	Timestamp
1234	Smith	John	Hello, world!	20130710
5678	Cooper	Joyce	wysiwyg	20120825
5678	Cooper	Joyce	wisiwig	20130916

- Logical view in HBase:

Row-Key	Value (CF, Qualifier, Version)
1234	info {'lastName': 'Smith', 'firstName': 'John'} pwd {'password': 'Hello, world!'}
5678	info {'lastName': 'Cooper', 'firstName': 'Joyce'} pwd {'password': 'wysiwyg'@ts 20130916, 'password': 'wisiwig'@ts 20120825}

HBase Components

- Region
 - Row of table are stores
- Region Server
 - Hosts the tables
- Master
 - Coordinating the Region Servers
- ZooKeeper
- HDFS
- API
 - The Java Client API



HBase Shell Commands

- See the list of the tables

```
list
```

- Create a table:

```
create 'testTable', 'cf'
```

- Insert data into a table:

Insert at rowA, column "cf:columnName" with a value of "val1"

```
put 'testTable', 'rowA', 'cf:columnName', 'val1'
```

- Retrieve data from a table:

Retrive "rowA" from the table "testTable"

```
get 'testTable', 'rowA'
```

- Iterate through a table:

```
- scan 'testTable'
```

- Delete a table:

```
enable 'testTable'  
drop 'testTable'
```

Hands-On: Running HBase

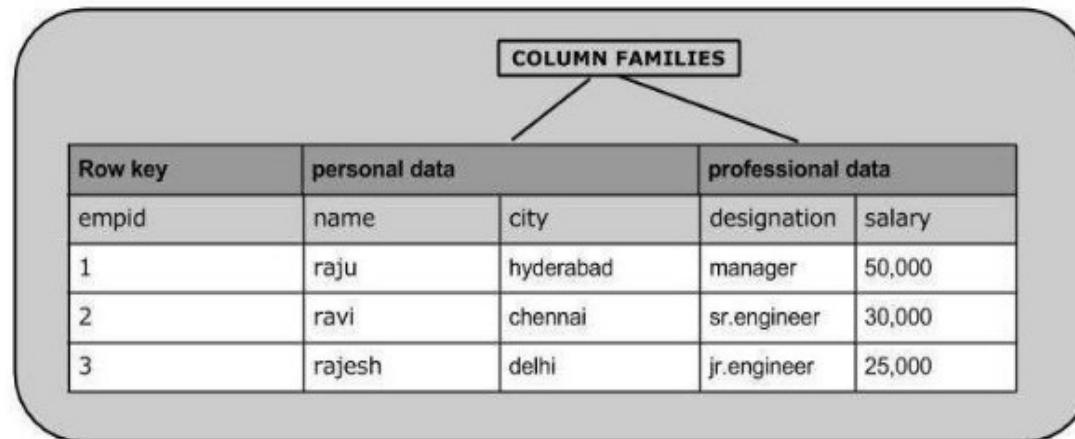
Hbase shell

Row key	personal data	professional data

```
$hbase shell  
hbase(main):001:0> create 'employee', 'personal data',  
'professional data'  
hbase(main):002:0> list
```

```
TABLE  
employee  
1 row(s) in 0.0310 seconds
```

Create Data



```
hbase(main):010:0> put 'employee','1','personal data:name','raju'  
0 row(s) in 0.1720 seconds
```

```
hbase(main):011:0> put 'employee','1','personal data:city','hyderabad'  
0 row(s) in 0.0140 seconds
```

```
hbase(main):018:0> put 'employee','1','professional data:designation','manager'  
0 row(s) in 0.0110 seconds
```

```
hbase(main):019:0> put 'employee','1','professional data:salary','50000'  
0 row(s) in 0.0070 seconds
```

Running HBase Browser

The screenshot shows the Apache Hue web interface. At the top, there is a navigation bar with links for 'Query Editors', 'Data Browsers', 'Workflows', 'Search', 'File Browser', 'Job Browser', and a user session 'guest1'. The 'Data Browsers' menu is open, showing 'Metastore Tables' and 'HBase'. Below the navigation bar, the main area is titled 'Home - HBase'. It features a search bar for 'Table Name', checkboxes for 'Enable', 'Disable', and 'Drop', and a 'New Table' button. The status of the table is shown as 'Enabled'. A message at the bottom states 'No data available in table'. A red arrow points to the 'HBase' link in the dropdown menu.

Viewing Employee Table

HUE Home Query Editors Data Browsers Workflows Search Security

H HBase Browser

Home - Cluster Switch Cluster

Search for Table Name Enable Disable Drop New Table

<input type="checkbox"/> Table Name	<input checked="" type="checkbox"/> Enabled
employee	<input checked="" type="checkbox"/>

H HBase Browser

Home - Cluster / employee Switch Cluster

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix] personal data: professional data:

Filter Columns/Families All Sort By ASC

1			
personal data: city	personal data: name	professional data: designation	professional data: salary
hyderabad	raju	manager	50000

Create a table in HBase

H HBase Browser

Home - HBase Switch Cluster ▾

Search for Table Name Enable Disable Drop

Table Name Enabled

 New Table

Create New Table

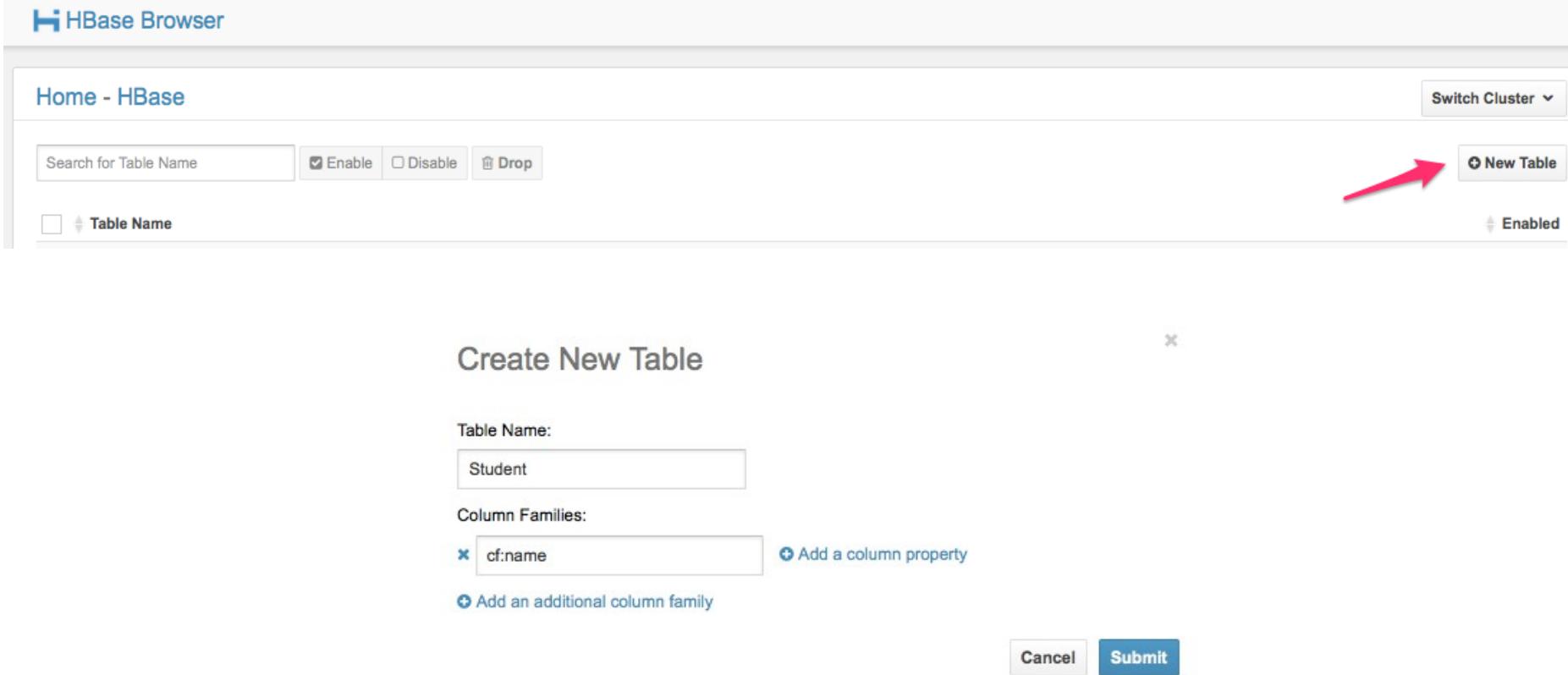
Table Name:

Column Families:

Add a column property

Add an additional column family

Cancel Submit



Insert a new row in a table

HBase Browser

Home - HBase / Student

Switch Cluster ▾

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix* +3, fam:] Filter Columns/Families All ▾

No rows to display.

Fetched 123 in 0.374 seconds.



Add field into a new row

Insert New Row

Row Key

123

cf:firstname

Thanachart

cf:lastname

Numnonda

 Add Field

Cancel

Submit

Insert New Row

Row Key

124

cf:firstname

Somchai

 Add Field

Cancel

Submit

Home - HBase / Student

Switch Cluster ▾

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix* +3, fam:



cf:



Filter Columns/Families

All

Sort By DESC ▾

124

cf: firstname

Somchai

123

cf: firstname

cf: lastname

Thanachart

Numnonda

Introduction

A Petabyte Scale Data Warehouse Using Hadoop



Hive is developed by Facebook, designed to enable easy data summarization, ad-hoc querying and analysis of large volumes of data. It provides a simple query language called Hive QL, which is based on SQL

What Hive is NOT

Hive is not designed for online transaction processing and does not offer real-time queries and row level updates. It is best used for batch jobs over large sets of immutable data (like web logs, etc.).

Sample HiveQL

The Query compiler uses the information stored in the metastore to convert SQL queries into a sequence of map/reduce jobs, e.g. the following query

```
SELECT * FROM t where t.c = 'xyz'
```

```
SELECT t1.c2 FROM t1 JOIN t2 ON (t1.c1 = t2.c1)
```

```
SELECT t1.c1, count(1) from t1 group by t1.c1
```

Sample HiveQL

The Query compiler uses the information stored in the metastore to convert SQL queries into a sequence of map/reduce jobs, e.g. the following query

```
SELECT * FROM t where t.c = 'xyz'
```

```
SELECT t1.c2 FROM t1 JOIN t2 ON (t1.c1 = t2.c1)
```

```
SELECT t1.c1, count(1) from t1 group by t1.c1
```

System Architecture and Components

Metastore: To store the meta data.

Query compiler and execution engine: To convert SQL queries to a sequence of map/reduce jobs that are then executed on Hadoop.

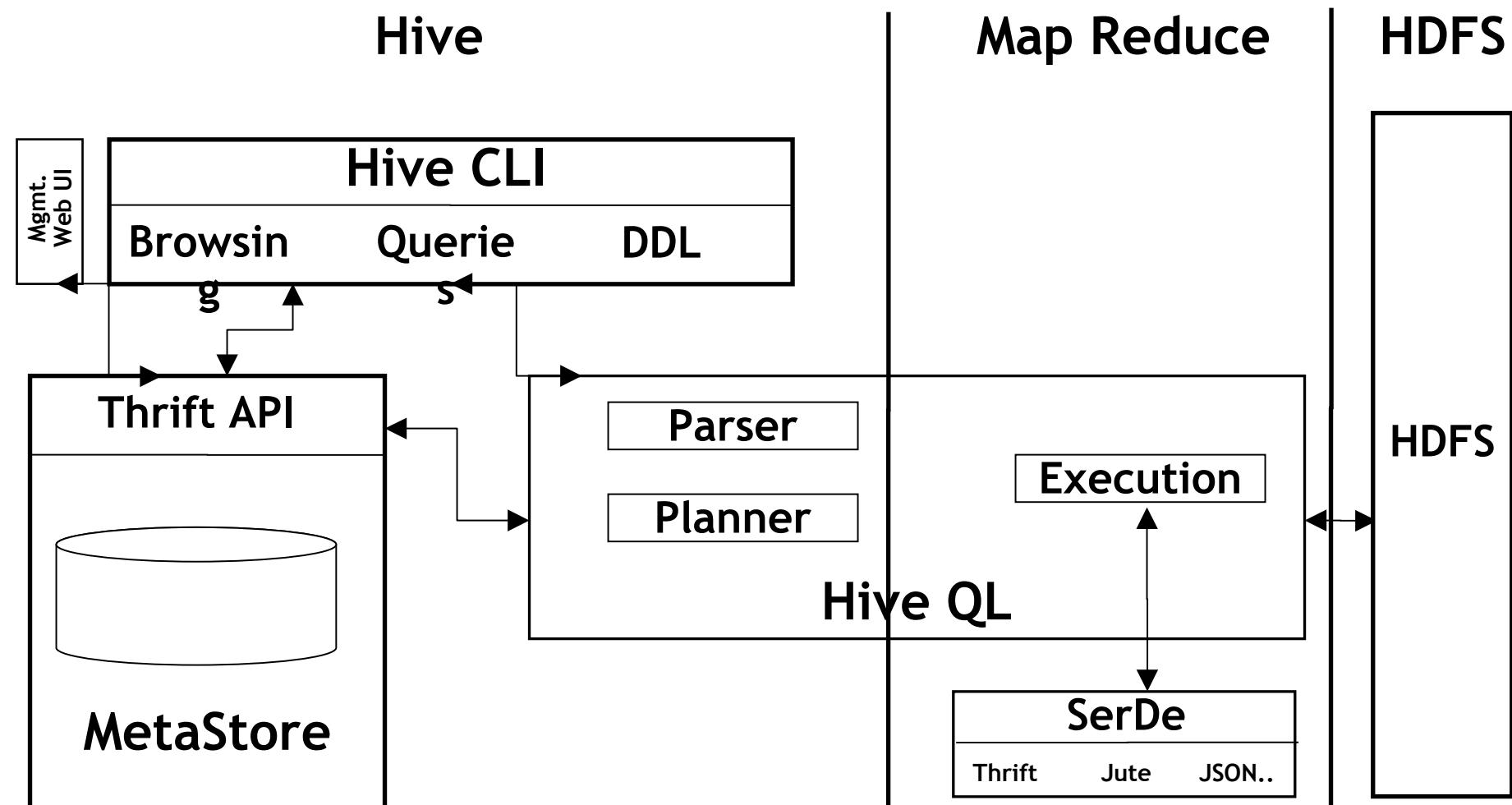
SerDe and ObjectInspectors: Programmable interfaces and implementations of common data formats and types.

A SerDe is a combination of a Serializer and a Deserializer (hence, Ser-De). The Deserializer interface takes a string or binary representation of a record, and translates it into a Java object that Hive can manipulate. The Serializer, however, will take a Java object that Hive has been working with, and turn it into something that Hive can write to HDFS or another supported system.

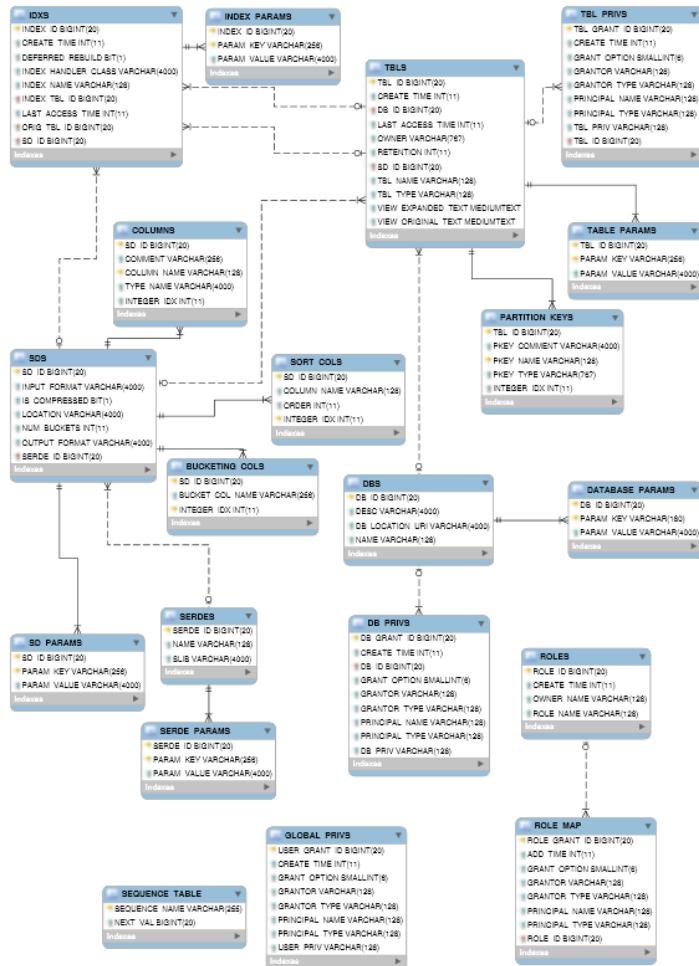
UDF and UDAF: Programmable interfaces and implementations for user defined functions (scalar and aggregate functions).

Clients: Command line client similar to Mysql command line.

Architecture Overview



Hive Metastore



Hive Metastore is a repository to keep all Hive metadata; Tables and Partitions definition.

By default, Hive will store its metadata in Derby DB

Hive Built in Functions

Return Type	Function Name (Signature)	Description
BIGINT	round(double a)	returns the rounded BIGINT value of the double
BIGINT	floor(double a)	returns the maximum BIGINT value that is equal or less than the double
BIGINT	ceil(double a)	returns the minimum BIGINT value that is equal or greater than the double
double	rand(), rand(int seed)	returns a random number (that changes from row to row). Specifying the seed will make sure the generated random number sequence is deterministic.
string	concat(string A, string B,...)	returns the string resulting from concatenating B after A. For example, concat('foo', 'bar') results in 'foobar'. This function accepts arbitrary number of arguments and return the concatenation of all of them.
string	substr(string A, int start)	returns the substring of A starting from start position till the end of string A. For example, substr('foobar', 4) results in 'bar'
string	substr(string A, int start, int length)	returns the substring of A starting from start position with the given length e.g. substr('foobar', 4, 2) results in 'ba'
string	upper(string A)	returns the string resulting from converting all characters of A to upper case e.g. upper('fOoBaR') results in 'FOOBAR'
string	ucase(string A)	Same as upper
string	lower(string A)	returns the string resulting from converting all characters of B to lower case e.g. lower('fOoBaR') results in 'foobar'
string	lcase(string A)	Same as lower
string	trim(string A)	returns the string resulting from trimming spaces from both ends of A e.g. trim(' foobar ') results in 'foobar'
string	ltrim(string A)	returns the string resulting from trimming spaces from the beginning(left hand side) of A. For example, ltrim(' foobar ') results in 'foobar '
string	rtrim(string A)	returns the string resulting from trimming spaces from the end(right hand side) of A. For example, rtrim(' foobar ') results in ' foobar'
string	regexp_replace(string A, string B, string C)	returns the string resulting from replacing all substrings in B that match the Java regular expression syntax(See Java regular expressions syntax) with C. For example, regexp_replace('foobar', 'oo ar',) returns 'fb'
string	from_unixtime(int unixtime)	convert the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00"
string	to_date(string timestamp)	Return the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01"
int	year(string date)	Return the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970
int	month(string date)	Return the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11
int	day(string date)	Return the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1
string	get_json_object(string json_string, string path)	Extract json object from a json string based on json path specified, and return json string of the extracted json object. It will return null if the input json string is invalid

Hive Aggregate Functions

Return Type	Aggregation Function Name (Signature)	Description
BIGINT	count(*), count(expr), count(DISTINCT expr[, expr_.])	count(*) - Returns the total number of retrieved rows, including rows containing NULL values; count(expr) - Returns the number of rows for which the supplied expression is non- NULL; count(DISTINCT expr[, expr]) - Returns the number of rows for which the supplied expression(s) are unique and non-NULL.
DOUBLE	sum(col), sum(DISTINCT col)	returns the sum of the elements in the group or the sum of the distinct values of the column in the group
DOUBLE	avg(col), avg(DISTINCT col)	returns the average of the elements in the group or the average of the distinct values of the column in the group
DOUBLE	min(col)	returns the minimum value of the column in the group
DOUBLE	max(col)	returns the maximum value of the column in the group

Running Hive

Hive Shell

Interactive

hive

Script

hive -f myscript

Inline

*hive -e 'SELECT * FROM mytable'*

Hive Commands

Command Line

Function	Hive
Run query	hive -e 'select a.col from tab1 a'
Run query silent mode	hive -S -e 'select a.col from tab1 a'
Set hive config variables	hive -e 'select a.col from tab1 a' -hiveconf hive.root.logger=DEBUG,console
Use initialization script	hive -i initialize.sql
Run non-interactive script	hive -f script.sql

Hive Shell

Function	Hive
Run script inside shell	source file_name
Run ls (dfs) commands	dfs -ls /user
Run ls (bash command) from shell	!ls
Set configuration variables	set mapred.reduce.tasks=32
TAB auto completion	set hive.<TAB>
Show all variables starting with hive	set
Revert all variables	reset
Add jar to distributed cache	add jar jar_path
Show all jars in distributed cache	list jars
Delete jar from distributed cache	delete jar jar_name

Hive Tables

- Managed- CREATE TABLE
 - LOAD- File moved into Hive's data warehouse directory
 - DROP- Both data and metadata are deleted.
- External- CREATE EXTERNAL TABLE
 - LOAD- No file moved
 - DROP- Only metadata deleted
 - Use when sharing data between Hive and Hadoop applications or you want to use multiple schema on the same data

Hive External Table

- `CREATE EXTERNAL TABLE external_Table (dummy STRING)`
- `LOCATION '/user/notroot/external_table';`

Dropping External Table using Hive:-

Hive will delete metadata from metastore

Hive will NOT delete the HDFS file

You need to manually delete the HDFS file

Java JDBC for Hive

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;

public class HiveJdbcClient {
    private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

    public static void main(String[] args) throws SQLException {
        try {
            Class.forName(driverName);
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.exit(1);
        }
        Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/default", "", "");
        Statement stmt = con.createStatement();
        String tableName = "testHiveDriverTable";
        stmt.executeQuery("drop table " + tableName);
        ResultSet res = stmt.executeQuery("create table " + tableName + " (key int, value string)");
        // show tables
        String sql = "show tables '" + tableName + "'";
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        if (res.next()) {
            System.out.println(res.getString(1));
        }
        // describe table
        sql = "describe " + tableName;
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        while (res.next()) {
            System.out.println(res.getString(1) + "\t" + res.getString(2));
        }
    }
}
```

Java JDBC for Hive

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;

public class HiveJdbcClient {
    private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

    public static void main(String[] args) throws SQLException {
        try {
            Class.forName(driverName);
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.exit(1);
        }
        Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/default", "", "");
        Statement stmt = con.createStatement();
        String tableName = "testHiveDriverTable";
        stmt.executeQuery("drop table " + tableName);
        ResultSet res = stmt.executeQuery("create table " + tableName + " (key int, value string)");
        // show tables
        String sql = "show tables '" + tableName + "'";
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        if (res.next()) {
            System.out.println(res.getString(1));
        }
        // describe table
        sql = "describe " + tableName;
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        while (res.next()) {
            System.out.println(res.getString(1) + "\t" + res.getString(2));
        }
    }
}
```

HiveQL and MySQL Comparison

Metadata

Function	MySQL	HiveQL
Selecting a database	USE database;	USE database;
Listing databases	SHOW DATABASES;	SHOW DATABASES;
Listing tables in a database	SHOW TABLES;	SHOW TABLES;
Describing the format of a table	DESCRIBE table;	DESCRIBE (FORMATTED EXTENDED) table;
Creating a database	CREATE DATABASE db_name;	CREATE DATABASE db_name;
Dropping a database	DROP DATABASE db_name;	DROP DATABASE db_name (CASCADE);

HiveQL and MySQL Query Comparison

Query

Function	MySQL	HiveQL
Retrieving information	<code>SELECT from_columns FROM table WHERE conditions;</code>	<code>SELECT from_columns FROM table WHERE conditions;</code>
All values	<code>SELECT * FROM table;</code>	<code>SELECT * FROM table;</code>
Some values	<code>SELECT * FROM table WHERE rec_name = "value";</code>	<code>SELECT * FROM table WHERE rec_name = "value";</code>
Multiple criteria	<code>SELECT * FROM table WHERE rec1="value1" AND rec2="value2";</code>	<code>SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";</code>
Selecting specific columns	<code>SELECT column_name FROM table;</code>	<code>SELECT column_name FROM table;</code>
Retrieving unique output records	<code>SELECT DISTINCT column_name FROM table;</code>	<code>SELECT DISTINCT column_name FROM table;</code>
Sorting	<code>SELECT col1, col2 FROM table ORDER BY col2;</code>	<code>SELECT col1, col2 FROM table ORDER BY col2;</code>
Sorting backward	<code>SELECT col1, col2 FROM table ORDER BY col2 DESC;</code>	<code>SELECT col1, col2 FROM table ORDER BY col2 DESC;</code>
Counting rows	<code>SELECT COUNT(*) FROM table;</code>	<code>SELECT COUNT(*) FROM table;</code>
Grouping with counting	<code>SELECT owner, COUNT(*) FROM table GROUP BY owner;</code>	<code>SELECT owner, COUNT(*) FROM table GROUP BY owner;</code>
Maximum value	<code>SELECT MAX(col_name) AS label FROM table;</code>	<code>SELECT MAX(col_name) AS label FROM table;</code>
Selecting from multiple tables (Join same table using alias w/"AS")	<code>SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;</code>	<code>SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);</code>

Hands-On: Loading Data using Hive

Start Hive

```
[root@quickstart guest1]# hive
2016-06-14 07:48:56,273 WARN  [main] mapreduce.TableMapReduceUtil: The
hbase-prefix-tree module jar containing PrefixTreeCodec is not present.
Continuing without it.

Logging initialized using configuration in file:/etc/hive/conf.dist/hiv
e-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended
.
hive> █
```

Quit from Hive

```
hive> quit;
```

Create Hive Table

```
hive> CREATE TABLE test_tbl(id INT, country STRING) ROW FORMAT DELIMITED
      FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
OK
Time taken: 0.886 seconds
hive> show tables;
OK
test_tbl
Time taken: 0.125 seconds, Fetched: 1 row(s)
hive> describe test_tbl;
OK
id          int
country      string
Time taken: 0.115 seconds, Fetched: 2 row(s)
hive> █
```

See also: <https://cwiki.apache.org/Hive/languagemanual-ddl.html>

Reviewing Hive Table in HDFS

The screenshot shows the Hue File Browser interface. The top navigation bar includes links for Home, Query Editors, Data Browsers, Workflows, Search, and Security. Below the navigation is a search bar labeled "Search for file name" and a "Actions" dropdown menu. The main area displays a list of files and directories under the path "/user/hive/warehouse". The list includes three entries: a folder named ".", a folder named "test_tbl", and a file named "t". The columns in the table are Name, Size, User, Group, Permissions, and Date.

Name	Size	User	Group	Permissions	Date
..		hive	supergroup	drwxrwxrwx	April 05, 2016 07:27 PM
test_tbl		hive	supergroup	drwxrwxrwx	June 14, 2016 12:50 AM
t		root	supergroup	drwxrwxrwx	June 14, 2016 12:50 AM

Alter and Drop Hive Table

```
Hive > alter table test_tbl add columns (remarks STRING);
```

```
hive > describe test_tbl;
```

```
OK
```

```
id int
```

```
country string
```

```
remarks string
```

```
Time taken: 0.077 seconds
```

```
hive > drop table test_tbl;
```

```
OK
```

```
Time taken: 0.9 seconds
```

See also: <https://cwiki.apache.org/Hive/adminmanual-metastoreadmin.html>

Preparing Large Dataset

<http://grouplens.org/datasets/movielens/>



[about](#) [datasets](#) [publications](#) [blog](#)

MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (<http://movielens.org>). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

Help our research lab: Please [take a short survey](#) about the MovieLens datasets

MovieLens 100k

100,000 ratings from 1000 users on 1700 movies.

- [README.txt](#)
- [ml-100k.zip](#)
- [Index of unzipped files](#)

MovieLens 1M

1 million ratings from 6000 users on 4000 movies.

- [README.txt](#)

Datasets

[MovieLens](#)

[HetRec 2011](#)

[WikiLens](#)

[Book-Crossing](#)

[Jester](#)

[EachMovie](#)

MovieLen Dataset

1) Type command > `wget`

`http://files.grouplens.org/datasets/movielens/ml-100k.zip`

2) Type command > `yum install unzip`

3) Type command > `unzip ml-100k.zip`

4) Type command > `more ml-100k/u.user`

```
[root@quickstart guest1]# more ml-100k/u.user
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
11|39|F|other|30329
```

Moving dataset to HDFS

- 1) Type command > `cd ml-100k`
- 2) Type command > `hadoop fs -mkdir /user/cloudera/movielens`
- 3) Type command > `hadoop fs -put u.user /user/cloudera/movielens`
- 4) Type command > `hadoop fs -ls /user/cloudera/movielens`

```
[root@quickstart ml-100k]# hadoop fs -ls /user/cloudera/movielens
Found 1 items
-rw-r--r--  1 root cloudera      22628 2016-06-14 08:04 /user/cloudera/
movielens/u.user
[root@quickstart ml-100k]#
```

CREATE & SELECT Table

```
hive> CREATE EXTERNAL TABLE users (userid INT, age INT,  
> gender STRING, occupation STRING, zipcode STRING) ROW FORMAT  
> DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE  
> LOCATION '/user/cloudera/movielens';  
OK  
Time taken: 0.646 seconds  
hive> SELECT * FROM users;  
OK  
1      24      M      technician      85711  
2      53      F      other      94043  
3      23      M      writer      32067  
4      24      M      technician      43537  
5      33      F      other      15213  
6      42      M      executive      98101  
7      57      M      administrator      91344  
8      36      M      administrator      05201
```

Bay Area Bike Share (BABS)

<http://www.bayareabikeshare.com/open-data>

The screenshot shows the top navigation bar of the Bay Area Bike Share website. On the left is the logo featuring a gear and the text "BAY AREA BikeShare". To the right are links for "OPEN DATA", "GIFT STORE", "ABOUT", "RESOURCES", "APP", "CONTACT", and "LOGIN". Below this, there are links for "SIGN UP", "HOW IT WORKS", "SUGGEST A STATION", "STATION MAP", and "PRICING", followed by social media icons for Facebook, Twitter, and Instagram.

OPEN DATA

Here you'll find Bay Area Bike Share's trip data for public use. So whether you're a designer, developer, or just plain curious, feel free to download it and bring it to life!

THE DATA

Each trip is anonymized and includes:

- Bike number
- Trip start day and time
- Trip end day and time

YEAR 1 DATA

(August 2013 - August 2014)

YEAR 2 DATA

(September 2014 - August 2015)



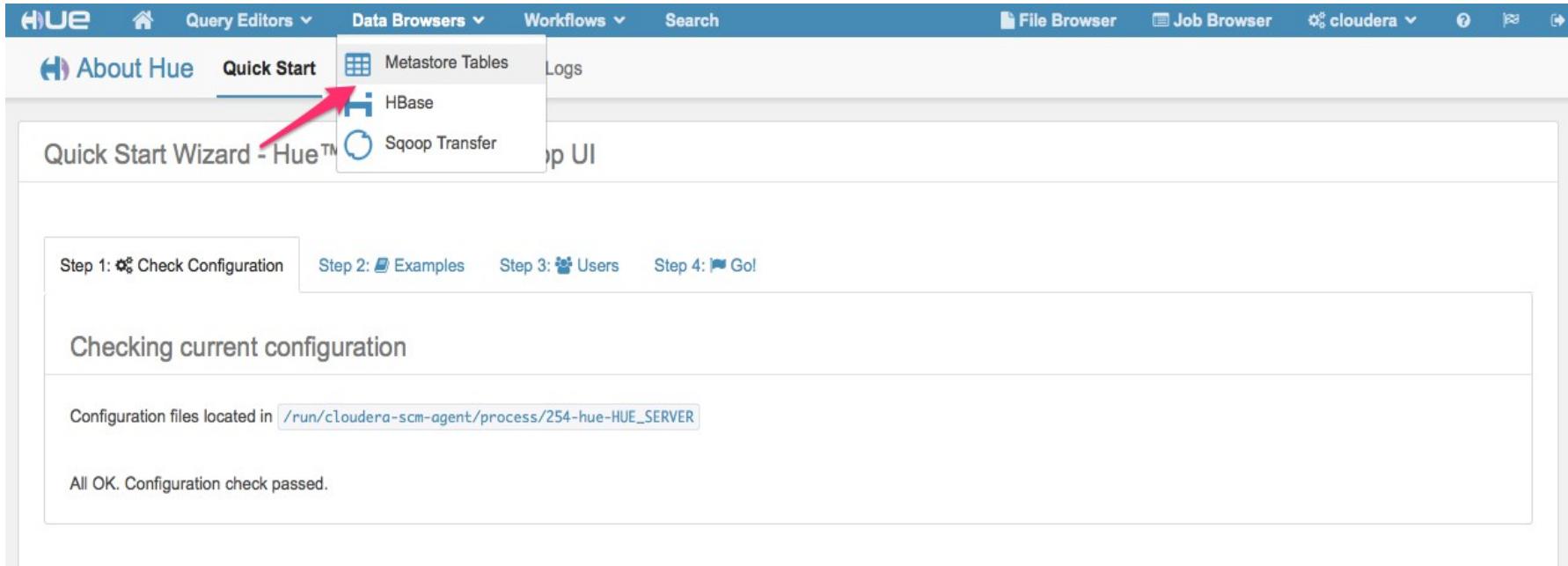
Preparing a bike data

```
$wget https://s3.amazonaws.com/babs-open-data/  
babs_open_data_year_1.zip  
$unzip babs_open_data_year_1.zip  
$cd 201402_babs_open_data/  
$hadoop fs -put 201402_trip_data.csv  
/user/cloudera  
$ hadoop fs -ls /user/cloudera
```

```
[root@quickstart 201402_babs_open_data]# hadoop fs -ls /user/cloudera  
Found 4 items  
-rw-r--r-- 1 root cloudera 17219022 2016-06-14 08:13 /user/cloudera/201402_t  
rip_data.csv  
drwxr-xr-x - cloudera cloudera 0 2016-06-14 04:29 /user/cloudera/input  
drwxr-xr-x - root cloudera 0 2016-06-14 08:04 /user/cloudera/movielen  
s  
drwxr-xr-x - cloudera cloudera 0 2016-06-14 04:32 /user/cloudera/output  
[root@quickstart 201402_babs_open_data]# █
```

Importing CSV Data with the Metastore App

The BABS data set contains 4 CSVs that contain data for stations, trips, rebalancing (availability), and weather. We will import **trips** dataset using Metastore Tables



The screenshot shows the Hue web interface with the following details:

- Top Navigation Bar:** Includes links for Home, Query Editors, Data Browsers, Workflows, Search, File Browser, Job Browser, cloudera, and various help and settings icons.
- Left Sidebar:** Features a "Quick Start Wizard - Hue™" section with four steps: Step 1: Check Configuration, Step 2: Examples, Step 3: Users, and Step 4: Go! Below this, it says "Checking current configuration".
- Metastore Tables Link:** A red arrow points to the "Metastore Tables" link in the "Data Browsers" dropdown menu.
- Content Area:** Displays the results of the configuration check, stating "Configuration files located in /run/cloudera-scm-agent/process/254-hue-HUE_SERVER" and "All OK. Configuration check passed."

Select: Create a new table from a file

The screenshot shows the Hue Metastore Manager interface. The top navigation bar includes links for Home, Query Editors, Data Browsers, Workflows, Search, and Security. The main title is "Metastore Manager". On the left sidebar, there are icons for Tables and SQL, and a list showing "default" and "sample_07". The main content area displays the "Databases > default" view. It includes sections for "STATS" (Default Hive database, public (ROLE), Location) and "TABLES". A search bar at the top of the tables section contains the placeholder "Search for a table...". Below the search bar are buttons for "View", "Browse Data", and "Drop". A table lists the existing table "sample_07". The table has columns for "Table Name" (with a checkbox), "Comment", and "Type". The "sample_07" row has a checkbox checked. In the top right corner of the main content area, there are three icons: a refresh symbol, a file with a plus sign, and a plus sign. A red arrow points towards the file-plus icon, indicating where to click to create a new table.

Table Name	Comment	Type
sample_07		

Name a table and select a file

Databases > default > Create a new table from a file

Step 1: Choose File

Step 2: Choose Delimiter

Step 3: Define Columns

Name Your Table and Choose A File

Table Name

trip

Name of the new table. Table names must be globally unique. Table names tend to correspond to the column names.

Description

Bay Area Bike Share

Use a table comment to describe the table. For example, note the data's provenance and any other relevant information.

Input File

/user/cloudera/201408_trip_data.csv

The HDFS path to the file on which to base this new table definition. It can be compressed (.zip) or not.

Import data from file



Check this box to import the data in this file after creating the table definition. Leave it unchecked to define an empty table.

Warning: The selected file is going to be moved during the import.

Choose a file

Home

/user / cloudera

..

201402_trip_data.csv

input

movielens

output



Upload a file



Choose Delimiter

Databases > default > Create a new table from a file

Step 1: Choose File

Step 2: Choose Delimiter

Step 3: Define Columns

Choose a Delimiter

Beeswax has determined that this file is delimited by **commas**.

Delimiter

Comma (,)



Preview

Enter the column delimiter which must be a single character. Use syntax like "\001" or "\t" for special characters.

Table preview

col_1	col_2	col_3	col_4	col_5	col_6	col_7	col_8	col_9	col_10	col_11
Trip ID	Duration	Start Date	Start Station	Start Terminal	End Date	End Station	End Terminal	Bike #	Subscriber Type	Zip Code
432946	406	8/31/2014 22:31	Mountain View Caltrain St...	28	8/31/2014 22:38	Castro Street and El Cami...	32	17	Subscriber	94040
432945	468	8/31/2014 22:07	Beale at Market	56	8/31/2014 22:15	Market at 4th	76	509	Customer	11231

Define Column Types

Databases > default > Create a new table from a file

Step 1: Choose File Step 2: Choose Delimiter Step 3: Define Columns

Define your columns

Use first row as column names Bulk edit column names

Column name	Column Type	Sample Row #1	Sample Row #2
TripID	int	432946	432945
Duration	int	406	468
StartDate	string	8/31/2014 22:31	8/31/2014 22:07
StartStation	string	Mountain View Caltrain Station	Beale at Market
StartTerminal	tinyint	28	56
EndDate	string	8/31/2014 22:38	8/31/2014 22:15



Create Table : Done

Databases > default > trip

Comment: Bay Area Bike Share			
	Columns	Sample	Properties
0	Name	Type	Comment
0	tripid	int	
1	duration	int	
2	startdate	string	
3	startstation	string	
4	startterminal	tinyint	
5	enddate	string	
6	endstation	string	
7	endterminal	tinyint	
8	bike	smallint	
9	subscribertype	string	
10	zipcode	smallint	

HUE Home Query Editors Data Browsers Workflows Search Security

Metastore Manager

Databases > default

SQL

< default

Tables (3)

test_tbl
trip
users

STATS

Default Hive database public (ROLE) Location

TABLES

Search for a table...

	Table Name	Comment	Type
<input type="checkbox"/>	test_tbl		
<input type="checkbox"/>	trip		
<input type="checkbox"/>	users		

Starting Hive Editor

The screenshot shows the Hue Query Editor interface. At the top, there's a navigation bar with links for 'HUE', 'Query Editors', 'Data Browsers', 'Workflows', 'Search', 'File Browser', 'Job Browser', 'cloudera', and help icons. Below the navigation bar, the main area has tabs for 'Hive Editor' (which is selected), 'Query Editor', 'My Queries', 'Saved Queries', and 'History'. On the left, there's a sidebar titled 'DATABASE' with a dropdown set to 'default'. Below it, a 'Table name...' input field and a list of tables: 'airline_data' and 'trip'. The 'trip' table is expanded, showing columns: 'tripid (int)', 'duration (int)', 'startdate (string)', 'startstation (string)', 'startterminal (tinyint)', 'enddate (string)', 'endstation (string)', 'endterminal (tinyint)', 'bike (smallint)', 'subscribertype (string)', and 'zipcode (smallint)'. In the center, there's a query editor area with a text input field containing a placeholder: 'Example: SELECT * FROM tablename, or press CTRL + space'. Below the input field are buttons for 'Execute', 'Save as...', 'Explain', and 'or create a New query'. At the bottom, there's a 'Recent queries' section with tabs for 'Recent queries', 'Query', 'Log', 'Columns', 'Results', and 'Chart'. This section lists three recent queries with their execution time, query text, and a 'See results...' link:

Time	Query	Result
11/04/2015 2:49:28 PM	DROP TABLE `default`.`babs`	See results...
11/04/2015 2:46:08 PM	SELECT startterminal, startstation, COUNT(1) AS count FROM babs GROUP BY startterminal, startstation ORDER BY count DESC LIMIT 10	See results...
11/04/2015 2:45:42 PM	SELECT startterminal, startstation, COUNT(1) AS count FROM bikeshare.trips GROUP BY startterminal, startstation ORDER BY count	

Find the top 10 most popular start stations based on the trip data

```
SELECT startterminal, startstation, COUNT(1) AS count FROM trip
GROUP BY startterminal, startstation ORDER BY count DESC LIMIT 10
```

The screenshot shows the Hue web interface. At the top, there's a navigation bar with links for Query Editors, Data Browsers, Workflows, Search, and Security. Below the navigation bar, the main area has tabs for Hive Editor (selected), Query Editor, My Queries, Saved Queries, and History. On the left, there's a sidebar for the 'default' database showing tables: test_tbl, trip, and users. The main content area contains a code editor with the following SQL query:

```
1 SELECT startterminal, startstation, COUNT(1) AS count FROM trip
GROUP BY startterminal, startstation ORDER BY count DESC LIMIT 10
```

Below the code editor are buttons for Execute, Save as..., Explain, Format, or create a New query. The results are displayed in a table under the 'Results' tab:

	startterminal	startstation	count
1	70	San Francisco Caltrain (Townsend at 4th)	9838
2	50	Harry Bridges Plaza (Ferry Building)	7343
3	60	Embarcadero at Sansome	6545
4	77	Market at Sansome	5922
5	55	Temporary Transbay Terminal (Howard at Beale)	5113
6	76	Market at 4th	5030
7	61	2nd at Townsend	4987
8	69	San Francisco Caltrain 2 (330 Townsend)	4976

```
1 SELECT startterminal, startstation, COUNT(1) AS count FROM trip GROUP BY startte
```

Execute

Save as...

Explain

Format

or create a

New query



Recent queries

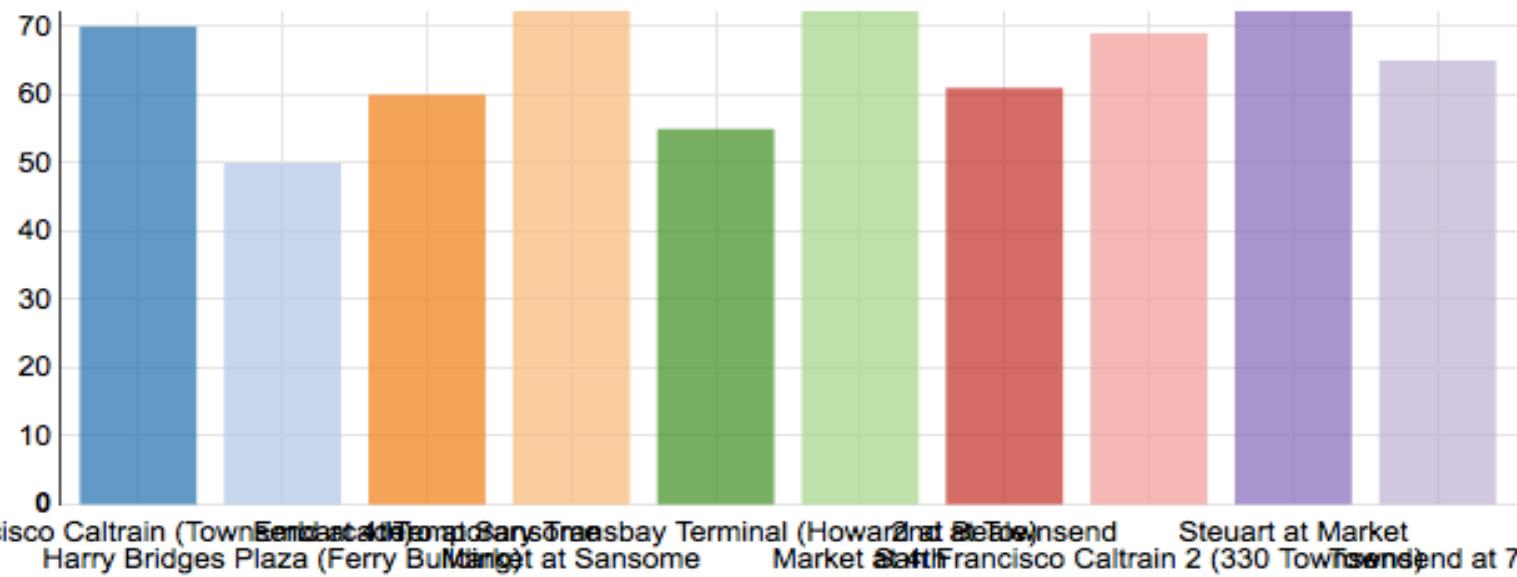
Query

Log

Columns

Results

Chart



1 Francisco Caltrain (Townsend)
2 San Jose Terminal
3 San Jose/Santa Clara Terminal
4 Harry Bridges Plaza (Ferry Building)
5 Market at Sansome
6 Market at 2nd Street
7 Market at 5th Street
8 Steuart at Market
9 Steuart at 2nd Street
10 Steuart at 7th Street

Find the total number of trips and average duration (in minutes) of those trips, grouped by hour

```
SELECT
    hour,
    COUNT(1) AS trips,
    ROUND(AVG(duration) / 60) AS avg_duration
FROM (
    SELECT
        CAST(SPLIT(SPLIT(t.startdate, ' ') [1], ':') [0] AS INT) AS
hour,
        t.duration AS duration
    FROM 'trip' t
    WHERE
        t.startterminal = 70
        AND
        t.duration IS NOT NULL
    ) r
GROUP BY hour
ORDER BY hour ASC;
```



Lecture

Understanding Spark

Introduction

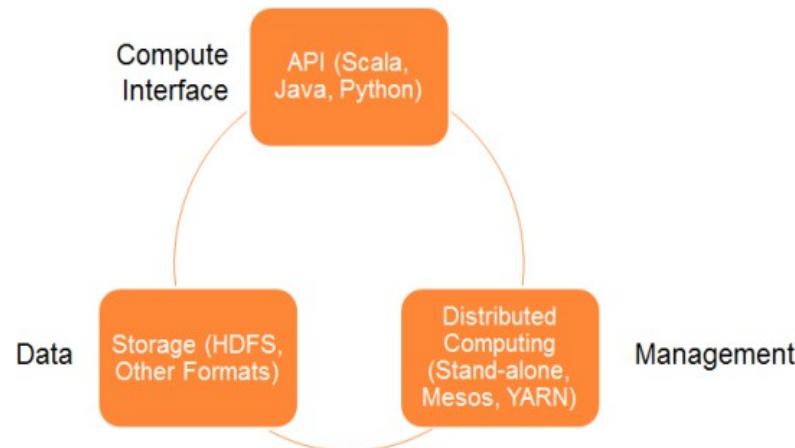
A fast and general engine for large scale data processing



An open source big data processing framework built around speed, ease of use, and sophisticated analytics. Spark enables applications in Hadoop clusters to run up to 100 times faster in memory and 10 times faster even when running on disk.

What is Spark?

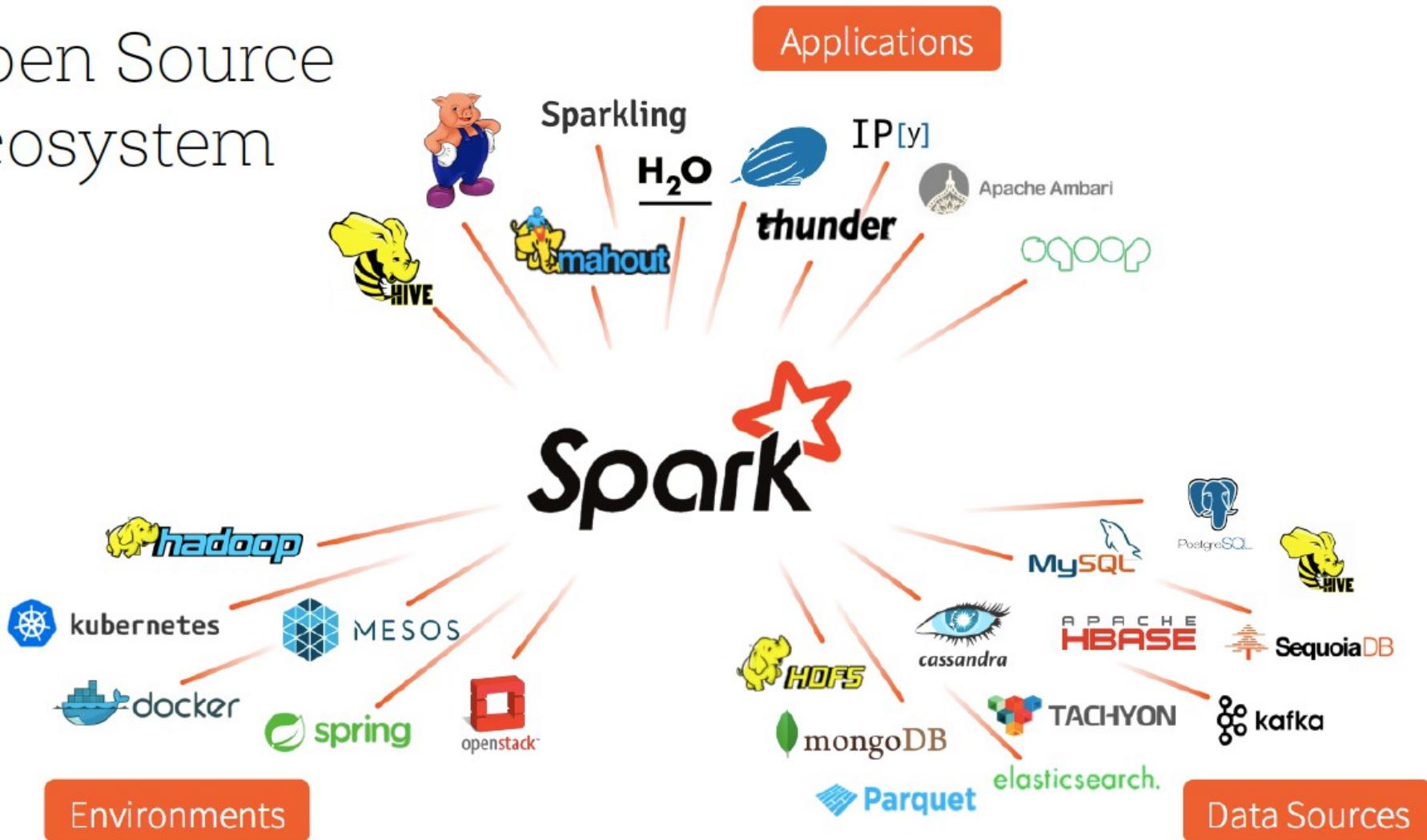
- Framework for distributed processing.
- In-memory, fault tolerant data structures
- Flexible APIs in Scala, Java, Python, SQL, R
- Open source



Why Spark?

- Handle Petabytes of data
- Significant faster than MapReduce
- Simple and intuitive APIs
- General framework
 - Runs anywhere
 - Handles (most) any I/O
 - Interoperable libraries for specific use-cases

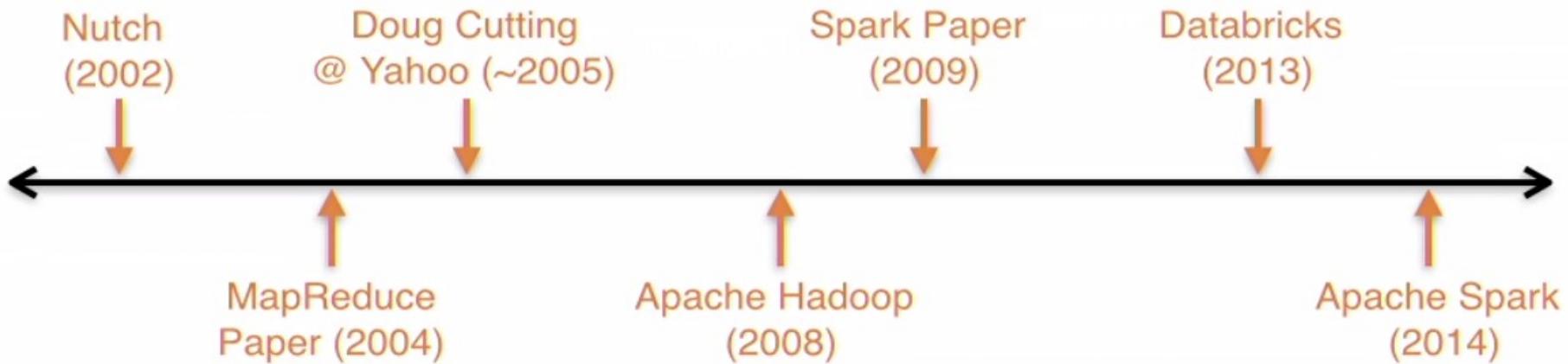
Open Source Ecosystem



Source: Jump start into Apache Spark and Databricks

Spark: History

- Founded by AMPIlab, UC Berkeley
- Created by Matei Zaharia (PhD Thesis)
- Maintained by Apache Software Foundation
- Commercial support by Databricks





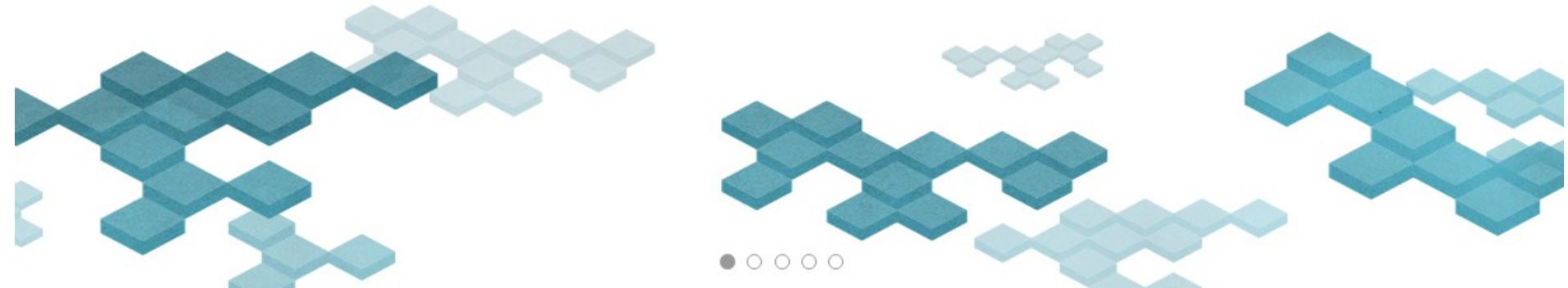
PRODUCT SPARK SOLUTIONS CUSTOMERS COMPANY BLOG RESOURCES

Partners Training [Sign Up](#) 



Data Science made easy, from ingest to production. Powered by Apache Spark™.

[SIGN UP FOR A 14-DAY FREE TRIAL](#)

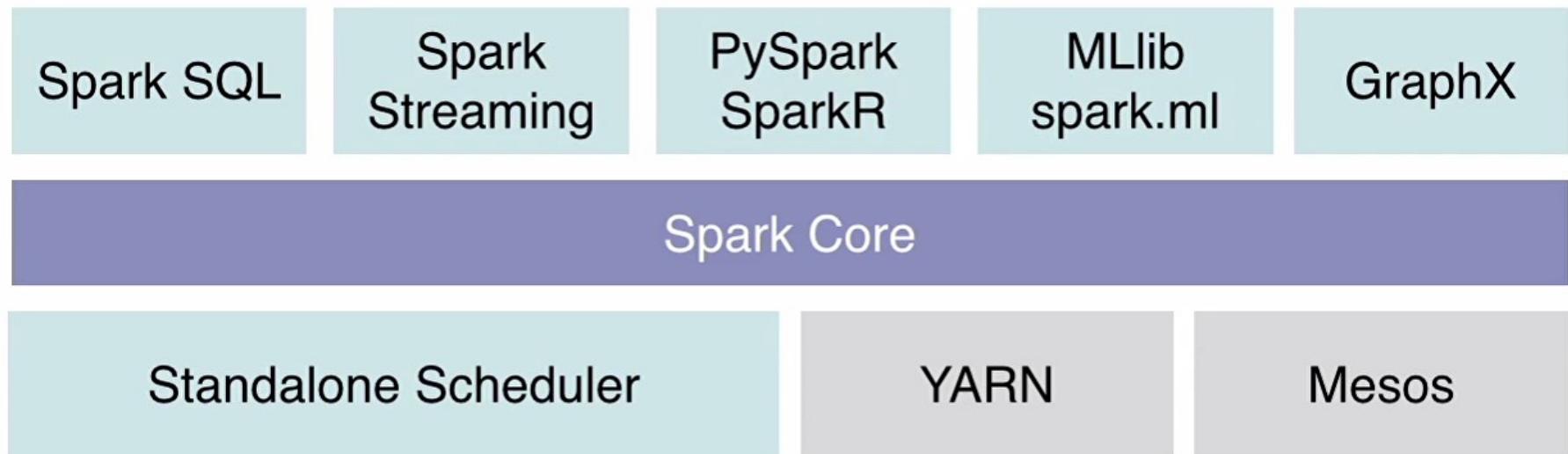


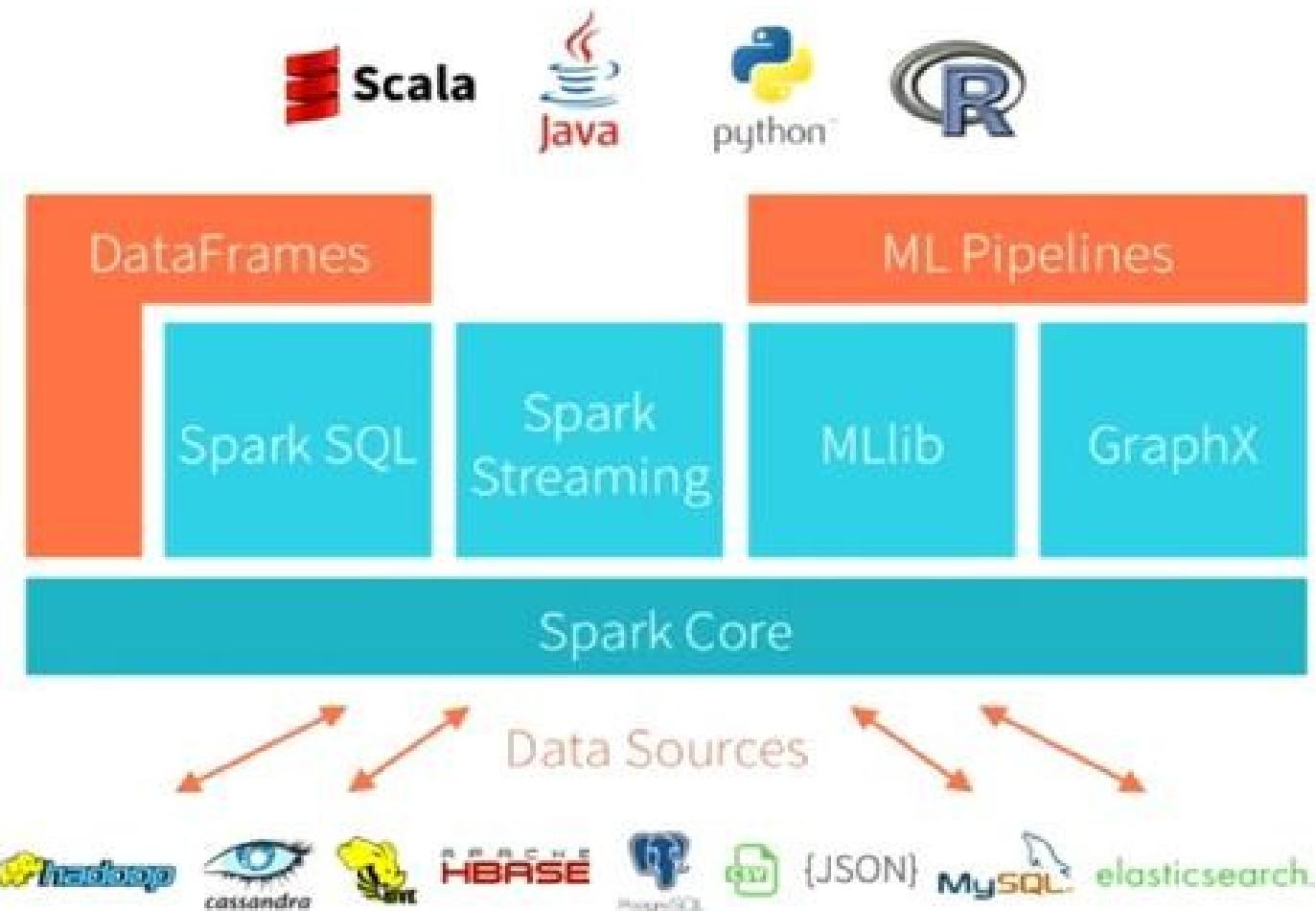
LEARN SPARK

Join the Community Edition Beta waitlist >

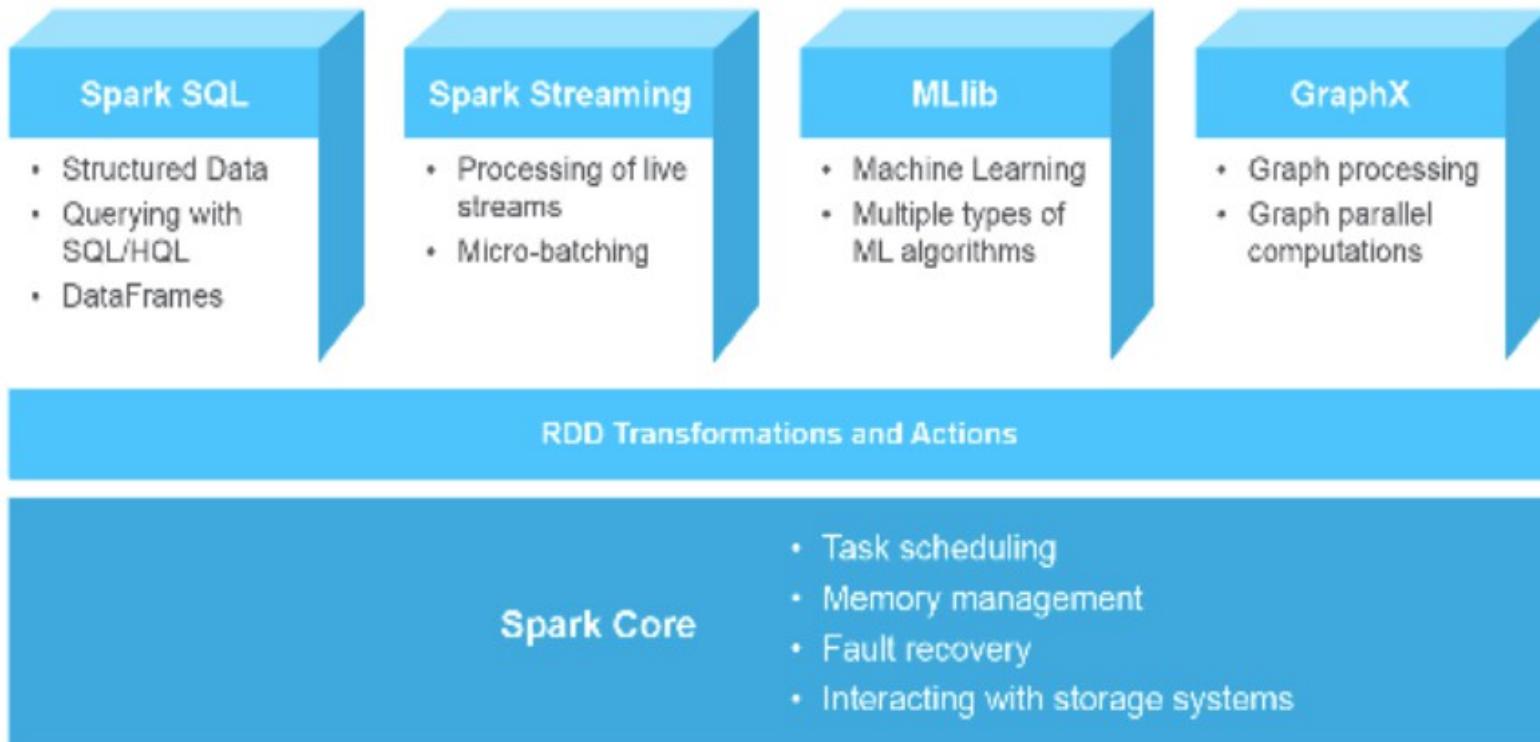
[Community Edition](#) [Spark 1.6](#) [Apache Spark 1.6](#)

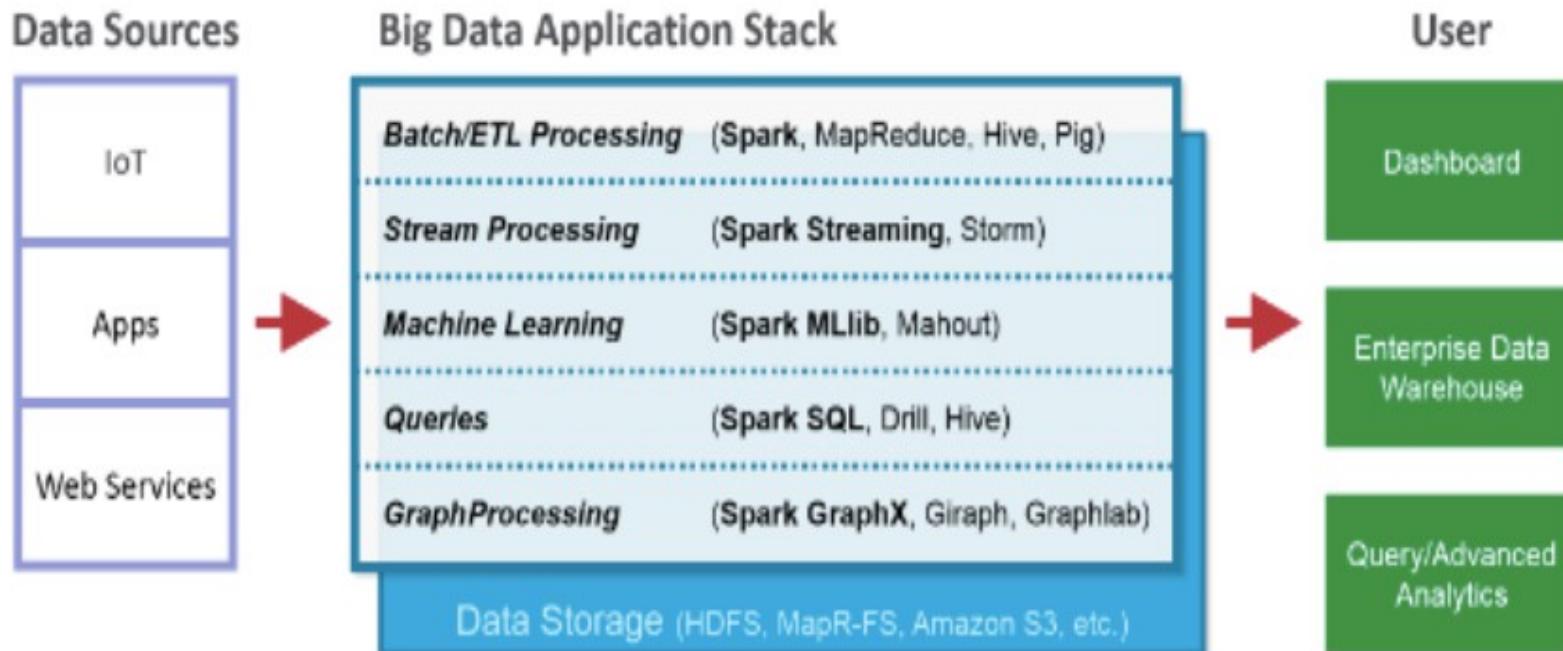
Spark Platform

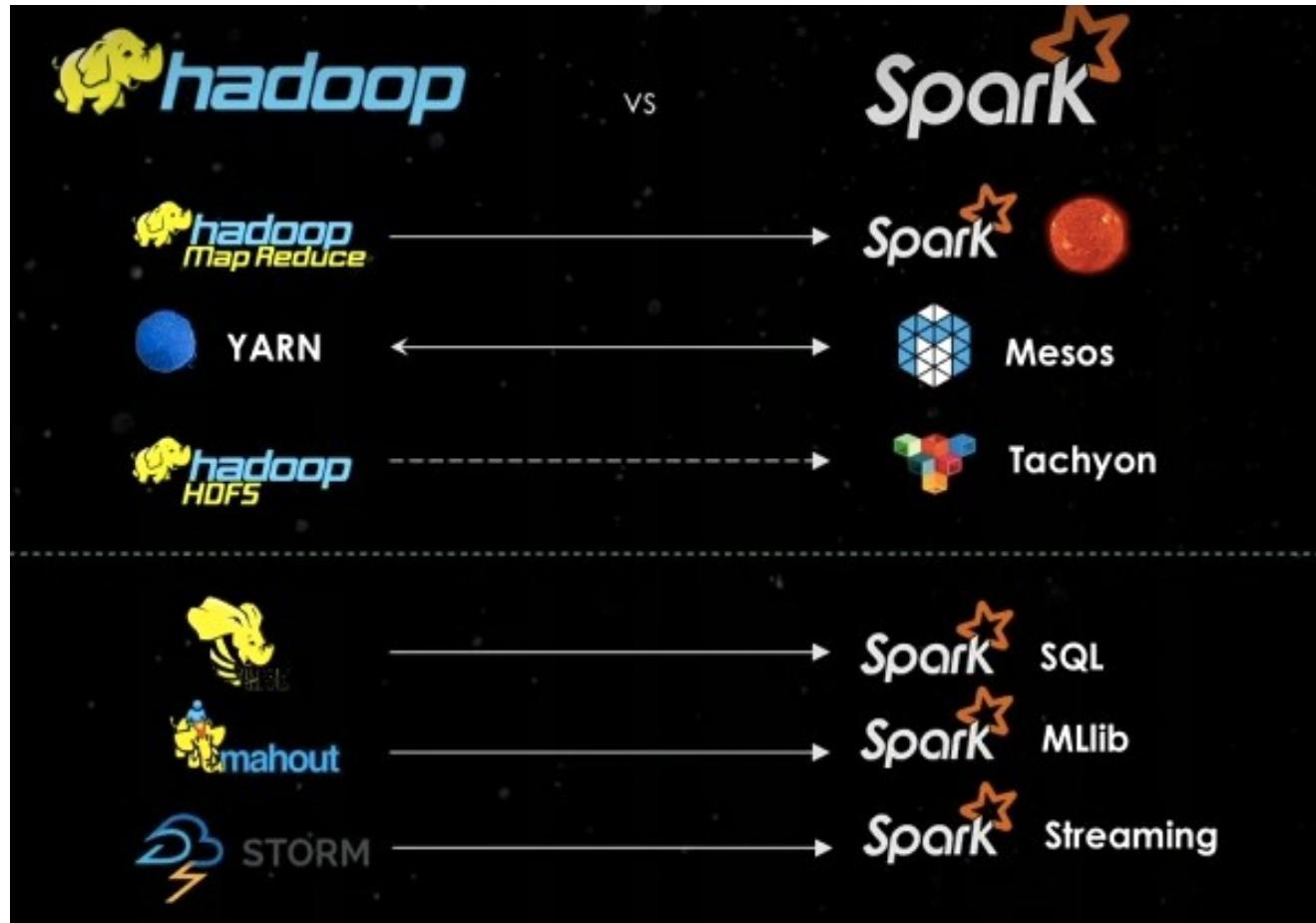




Spark Platform







Source: TRAINING Intro to Apache Spark - Brian Clapper

Large-Scale Usage



Largest cluster
8000 Nodes (Tencent)



Largest single job
1 PB (Alibaba, Databricks)



Top Streaming Intake
1 TB/hour (HHMI
Janelia Farm)



2014 On-Disk Sort Record
Fastest Open Source Engine
for sorting a PB

Source: Jump start into Apache Spark and Databricks

Notable Users

Companies That Presented at Spark Summit 2015 in San Francisco

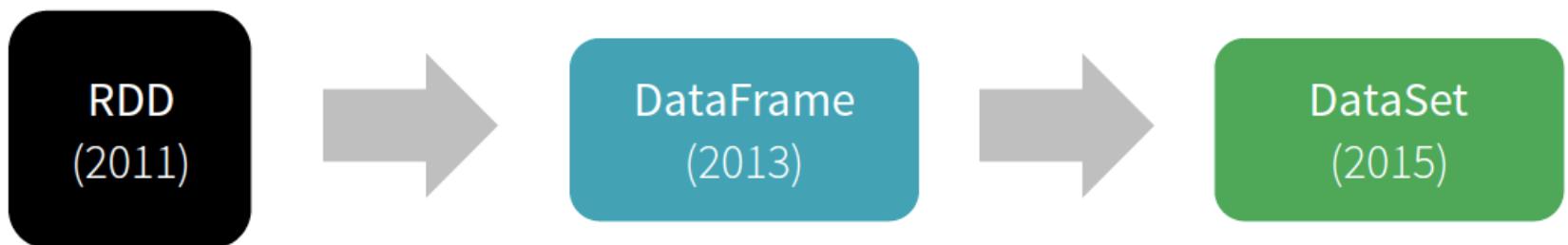


Source: Jump start into Apache Spark and Databricks

Do we still need Hadoop?

- Yes, why Hadoop?
 - HDFS
 - YARN
 - MapReduce is mature and still be appropriate for certain workloads
 - Other services: Sqoop, Flume, etc.
- But you can still use other resource management, storages
 - Spark Standalone
 - Amazon S3
 - Mesos

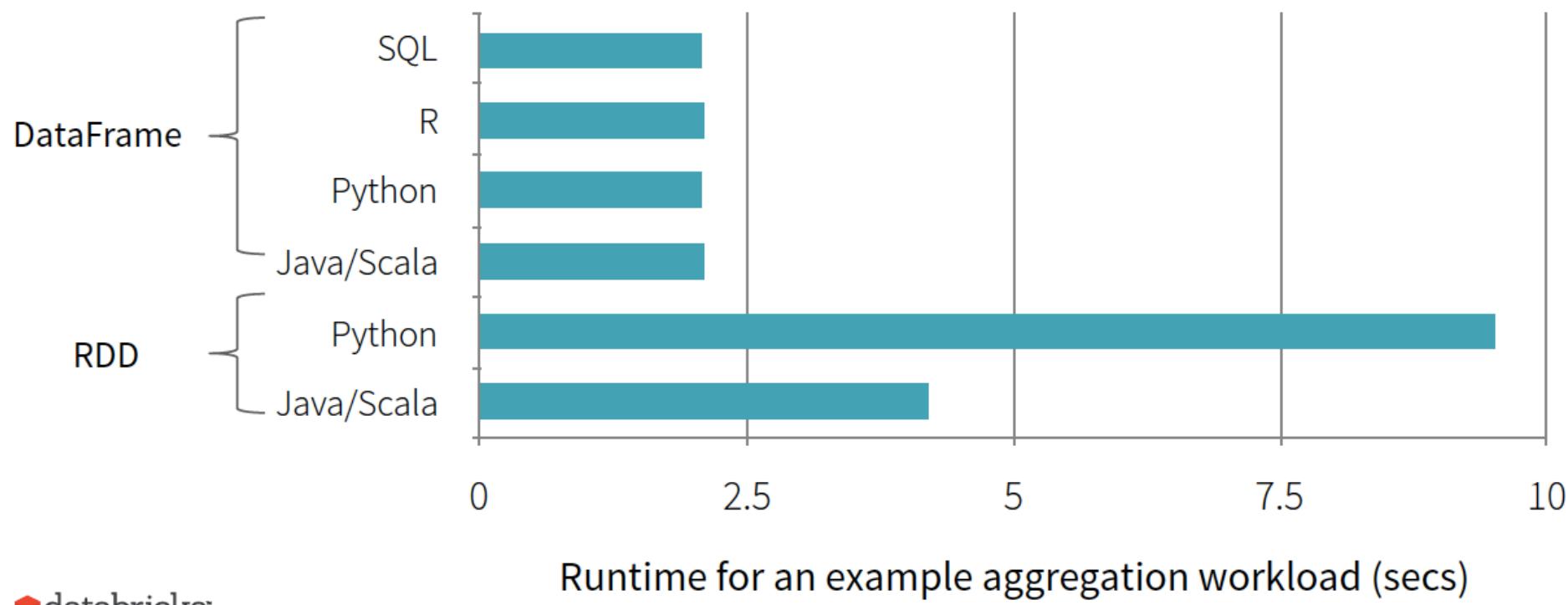
History of Spark APIs



- | | | |
|---|---|---|
| <ul style="list-style-type: none">• Distribute collection of JVM objects• Functional Operators (map, filter, etc.) | <ul style="list-style-type: none">• Distribute collection of Row objects• Expression-based operations and UDFs• Logical plans and optimizer• Fast/efficient internal representations | <ul style="list-style-type: none">• Internally rows, externally JVM objects• “Best of both worlds” type safe + fast |
|---|---|---|

Source: Jump start into Apache Spark and Databricks

Benefit of Logical Plan: Performance Parity Across Languages



Source: Jump start into Apache Spark and Databricks

What is a RDD?

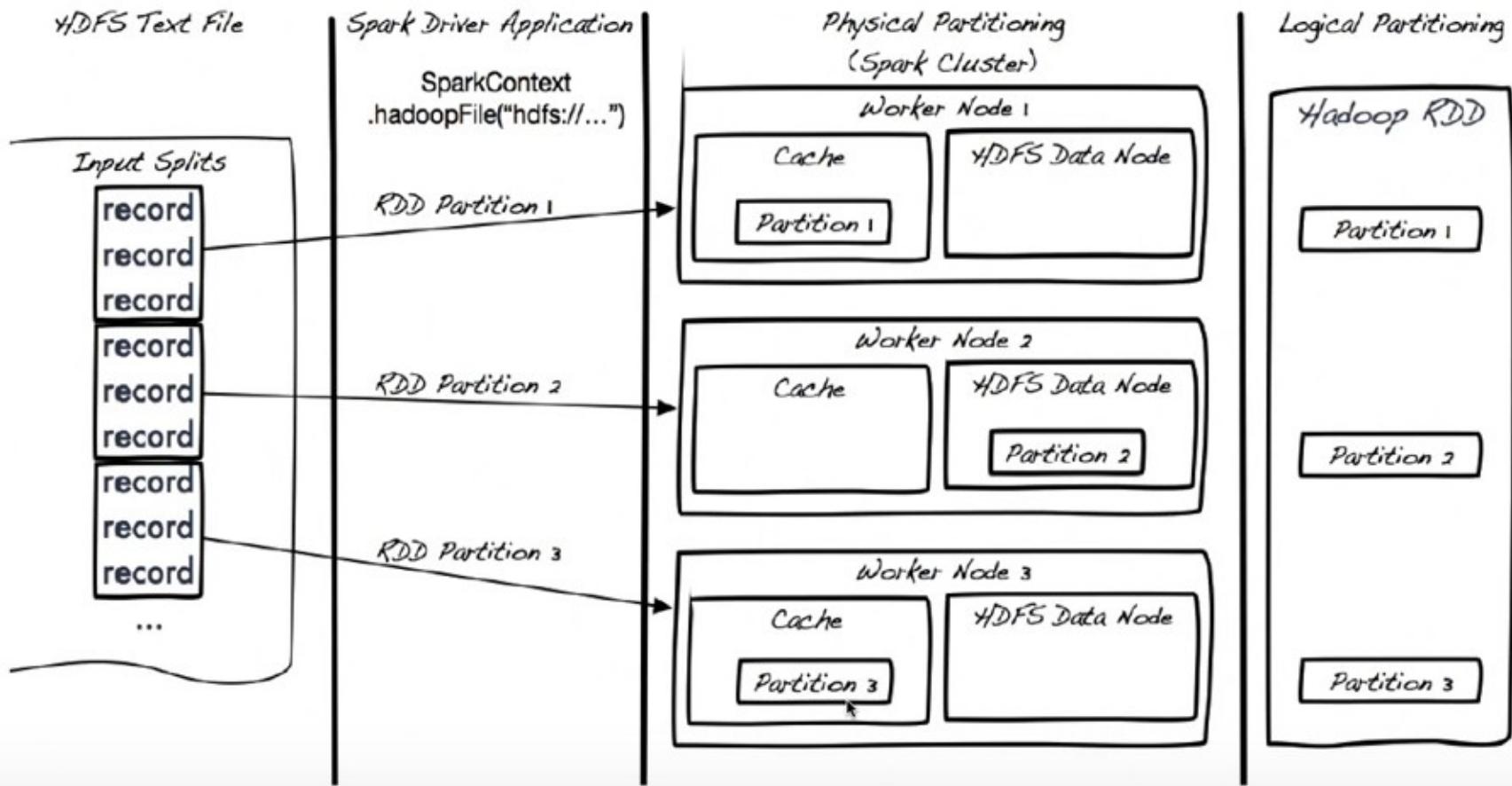
- **Resilient:** if the data in memory (or on a node) is lost, it can be recreated.
- **Distributed:** data is chunked into partitions and stored in memory across the cluster.
- **Dataset:** initial data can come from a table or be created programmatically

RDD:

- Fault tollerant
- Immutable
- Three methods for creating RDD:
 - Parallelizing an existing correction
 - Referencing a dataset
 - Transformation from an existing RDD
- Types of files supported:
 - Text files
 - SequenceFiles
 - Hadoop InputFormat

RDD Creation

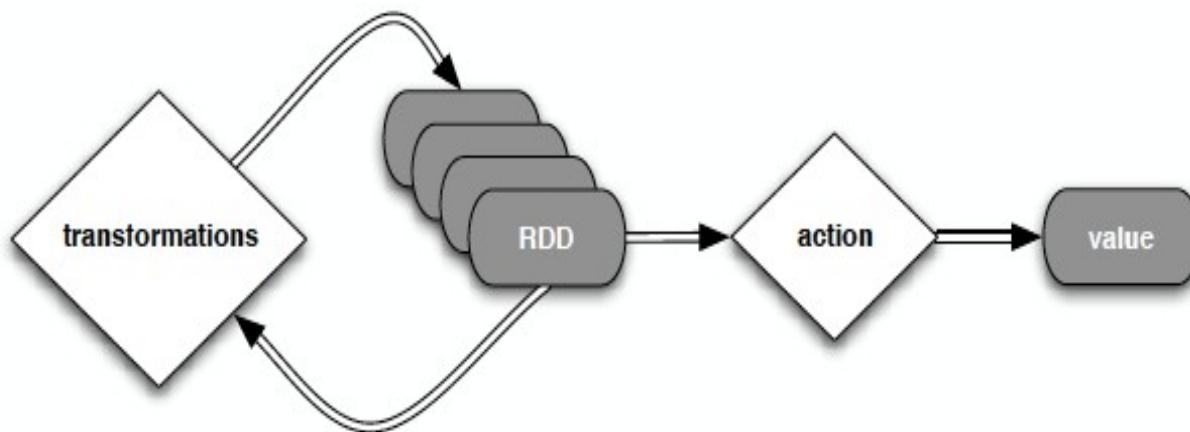
```
hdfsData = sc.textFile("hdfs://data.txt")
```



Source: Pspark: A brain-friendly introduction

RDD: Operations

- **Transformations:** transformations are lazy (not computed immediately)
- **Actions:** the transformed RDD gets recomputed when an action is run on it (default)



Direct Acyclic Graph (DAG)

- View the DAG

linesLength.toDebugString

- Sample DAG

```
res5: String =  
  MappedRDD[4] at map at <console>:16 (3 partitions)  
    MappedRDD[3] at map at <console>:16 (3 partitions)  
      FilteredRDD[2] at filter at <console>:14 (3 partitions)  
        MappedRDD[1] at textFile at <console>:12 (3 partitions)  
          HadoopRDD[0] at textFile at <console>:12 (3 partitions)|
```

Functions Deconstructed

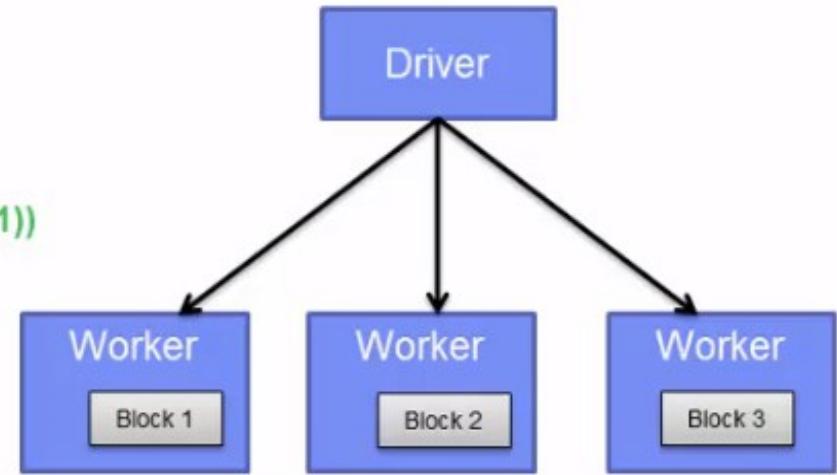
```
import random
flips = 1000000

# lazy eval
coins = xrange(flips) ← Python Generator

# lazy eval, nothing executed
heads = sc.parallelize(coins) \ ← Create RDD
Transformations → .map(lambda i: random.random()) \
    .filter(lambda r: r < 0.5) \
    .count() ← Action (materialize result)
```

What happens when an action is executed

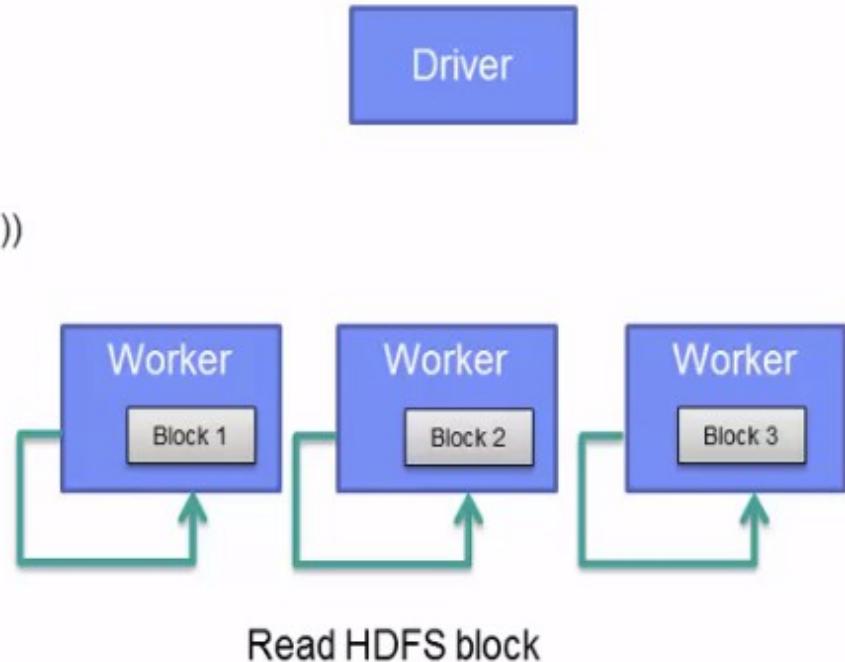
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
// Cache
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



Driver sends the code to be
executed on each block

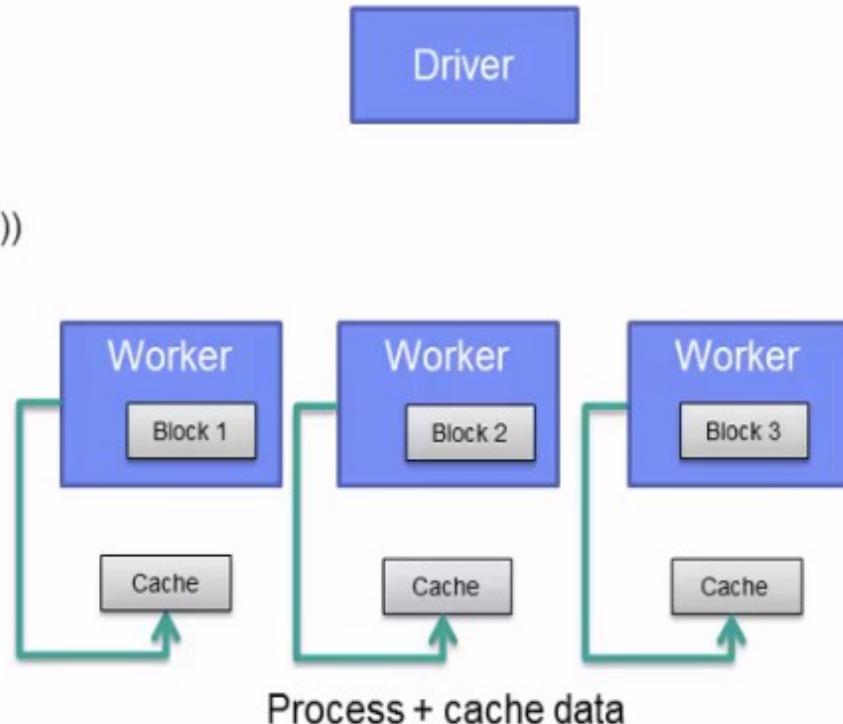
What happens when an action is executed

```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



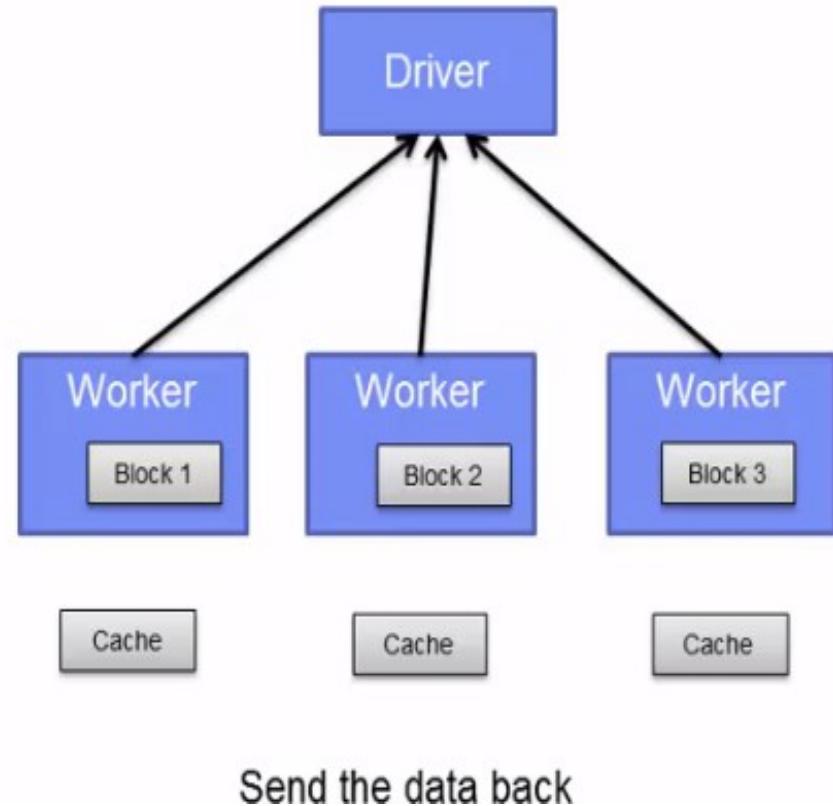
What happens when an action is executed

```
// Creating the RDD  
  
val logFile = sc.textFile("hdfs://...")  
  
// Transformations  
  
val errors = logFile.filter(_.startsWith("ERROR"))  
  
val messages = errors.map(_.split("\t")).map(r => r(1))  
  
//Caching  
  
messages.cache()  
  
// Actions  
  
messages.filter(_.contains("mysql")).count()  
messages.filter(_.contains("php")).count()
```



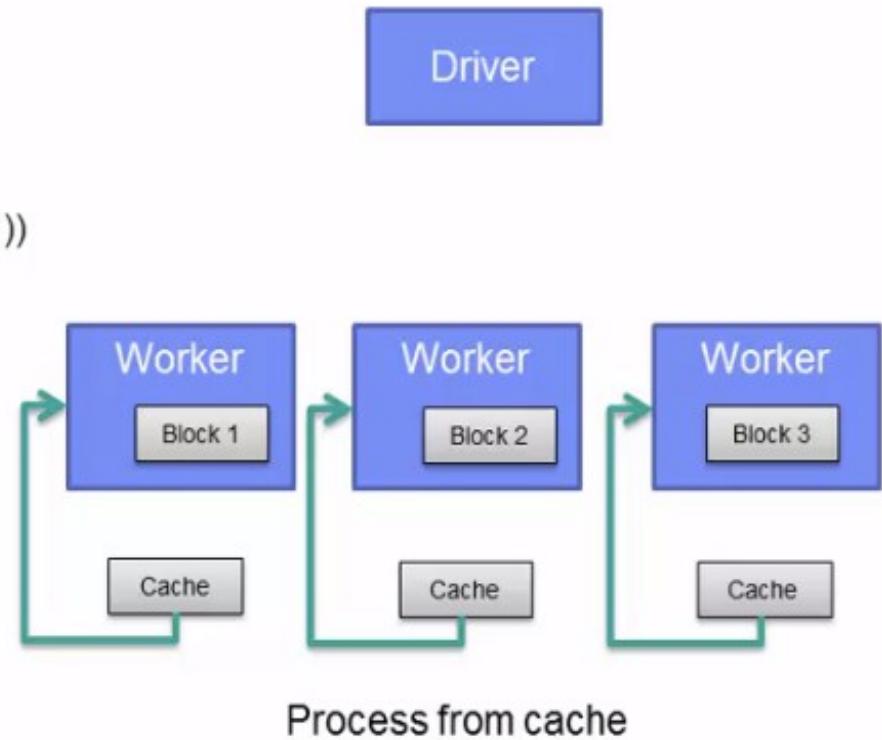
What happens when an action is executed

```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



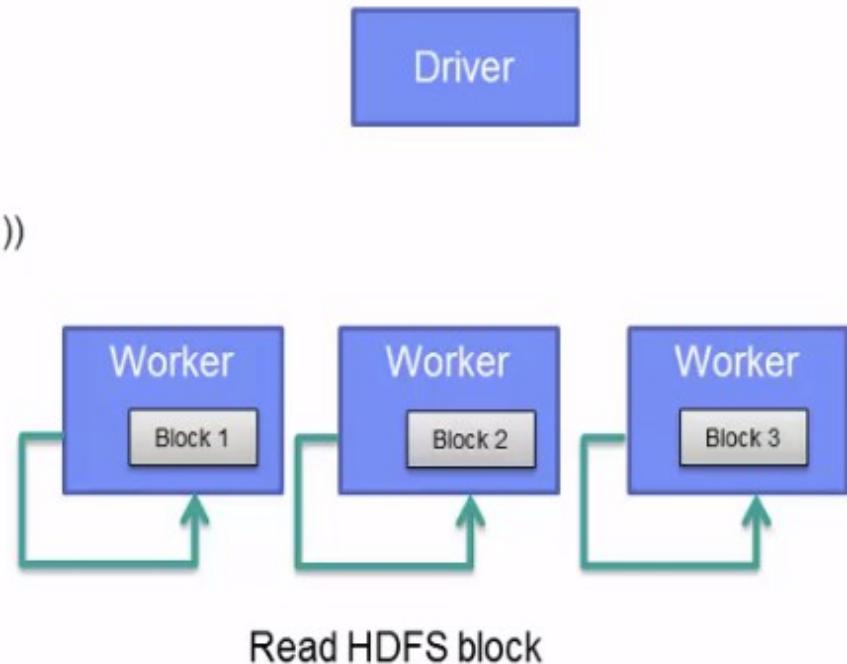
What happens when an action is executed

```
// Creating the RDD
val logFile = sc.textFile("hdfs://... ")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



What happens when an action is executed

```
// Creating the RDD  
  
val logFile = sc.textFile("hdfs://...")  
  
// Transformations  
  
val errors = logFile.filter(_.startsWith("ERROR"))  
  
val messages = errors.map(_.split("\t")).map(r => r(1))  
  
//Caching  
  
messages.cache()  
  
// Actions  
  
messages.filter(_.contains("mysql")).count()  
messages.filter(_.contains("php")).count()
```



Spark: Transformation

<i>transformation</i>	<i>description</i>
<code>map(func)</code>	return a new distributed dataset formed by passing each element of the source through a function <i>func</i>
<code>filter(func)</code>	return a new dataset formed by selecting those elements of the source on which <i>func</i> returns true
<code>flatMap(func)</code>	similar to map, but each input item can be mapped to 0 or more output items (so <i>func</i> should return a Seq rather than a single item)
<code>sample(withReplacement, fraction, seed)</code>	sample a fraction <i>fraction</i> of the data, with or without replacement, using a given random number generator <i>seed</i>
<code>union(otherDataset)</code>	return a new dataset that contains the union of the elements in the source dataset and the argument
<code>distinct([numTasks])</code>	return a new dataset that contains the distinct elements of the source dataset

Spark: Transformation

transformation	description
<code>groupByKey([numTasks])</code>	when called on a dataset of (K, V) pairs, returns a dataset of (K, seq[V]) pairs
<code>reduceByKey(func, [numTasks])</code>	when called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function
<code>sortByKey([ascending], [numTasks])</code>	when called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument
<code>join(otherDataset, [numTasks])</code>	when called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key
<code>cogroup(otherDataset, [numTasks])</code>	when called on datasets of type (K, V) and (K, W), returns a dataset of (K, seq[V], Seq[W]) tuples – also called groupWith
<code>cartesian(otherDataset)</code>	when called on datasets of types T and U, returns a dataset of (T, U) pairs (all pairs of elements)

Single RDD Transformation

filter females to analyze female buying patterns

male1, male2, female1 -> female1

map squared values

2, 5, 6 -> 4, 25, 36

flatMap to break up a sentence into words

my name is ray -> my, name, is, ray

find the **distinct** values in a dataset

apple, apple, banana -> apple, banana

sample two values at random

apple, banana, guava -> banana, apple

Multiple RDD Transformation

union

apple, orange, banana, guava,
banana, pear

intersection

banana

subtract anything shown in Dataset B
from Dataset A

apple, orange

cartesian (every possible pair combo)

(apple, guava), (apple, banana), ...

Dataset A

apple
orange
banana

Dataset B

guava
banana
pear

Pair RDD Transformation

- reduceByKey
- groupByKey
- combineByKey
- mapValues
- flatMapValues
- keys
- values
- subtractByKey
- join
- rightOuterJoin
- leftOuterJoin
- cogroup
- sortByKey

Spark:Actions

action	description
<code>reduce(func)</code>	aggregate the elements of the dataset using a function <code>func</code> (which takes two arguments and returns one), and should also be commutative and associative so that it can be computed correctly in parallel
<code>collect()</code>	return all the elements of the dataset as an array at the driver program – usually useful after a filter or other operation that returns a sufficiently small subset of the data
<code>count()</code>	return the number of elements in the dataset
<code>first()</code>	return the first element of the dataset – similar to <code>take(1)</code>
<code>take(n)</code>	return an array with the first <code>n</code> elements of the dataset – currently not executed in parallel, instead the driver program computes all the elements
<code>takeSample(withReplacement, fraction, seed)</code>	return an array with a random sample of <code>num</code> elements of the dataset, with or without replacement, using the given random number generator seed

Spark:Actions

action	description
<code>saveAsTextFile(path)</code>	write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call <code>toString</code> on each element to convert it to a line of text in the file
<code>saveAsSequenceFile(path)</code>	write the elements of the dataset as a Hadoop SequenceFile in a given path in the local filesystem, HDFS or any other Hadoop-supported file system. Only available on RDDs of key-value pairs that either implement Hadoop's <code>Writable</code> interface or are implicitly convertible to <code>Writable</code> (Spark includes conversions for basic types like <code>Int</code> , <code>Double</code> , <code>String</code> , etc).
<code>countByKey()</code>	only available on RDDs of type <code>(K, V)</code> . Returns a 'Map' of <code>(K, Int)</code> pairs with the count of each key
<code>foreach(func)</code>	run a function <code>func</code> on each element of the dataset – usually done for side effects such as updating an accumulator variable or interacting with external storage systems

Spark: Persistence

<i>transformation</i>	<i>description</i>
MEMORY_ONLY	Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly each time they're needed. This is the default level.
MEMORY_AND_DISK	Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, store the partitions that don't fit on disk, and read them from there when they're needed.
MEMORY_ONLY_SER	Store RDD as serialized Java objects (one byte array per partition). This is generally more space-efficient than deserialized objects, especially when using a fast serializer, but more CPU-intensive to read.
MEMORY_AND_DISK_SER	Similar to MEMORY_ONLY_SER, but spill partitions that don't fit in memory to disk instead of recomputing them on the fly each time they're needed.
DISK_ONLY	Store the RDD partitions only on disk.
MEMORY_ONLY_2, MEMORY_AND_DISK_2, etc	Same as the levels above, but replicate each partition on two cluster nodes.

Accumulators

- Similar to a MapReduce “Counter”
- A global variable to track metrics about your Spark program for debugging.
- Reasoning: Executors do not communicate with each other.
- Sent back to driver

Broadcast Variables

- Similar to a MapReduce “Distributed Cache”
- Sends read-only values to worker nodes.
- Great for lookup tables, dictionaries, etc.

Hands-On: Spark Programming

Functional tools in Python

- map
- filter
- reduce
- lambda
- IterTools
 - Chain, flatmap

Map

```
>>> a= [1,2,3]
```

```
>>> def add1(x) : return x+1
```

```
>>> map(add1, a)
```

Result: [2,3,4]

Filter

```
>>> a= [1,2,3,4]  
  
>>> def isOdd(x) : return x%2==1  
  
>>> filter(isOdd, a)
```

Result: [1,3]

Reduce

```
>>> a= [1,2,3,4]  
  
>>> def add(x,y) : return x+y  
  
>>> reduce(add, a)
```

Result: 10

lambda

```
>>> (lambda x: x + 1)(3)
```

Result: 4

```
>>> map((lambda x: x + 1), [1,2,3])
```

Result: [2,3,4]

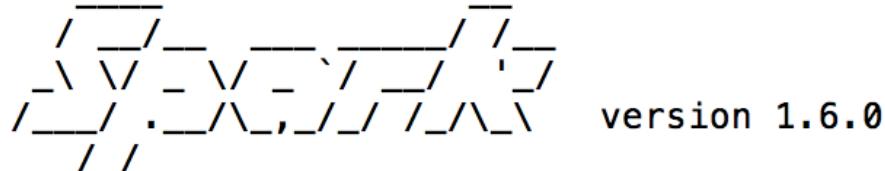
Exercises

- `(lambda x: 2*x)(3) => ?`
- `map(lambda x: 2*x, [1,2,3]) =>`
- `map(lambda t: t[0], [(1,2), (3,4), (5,6)]) =>`
- `reduce(lambda x,y: x+y, [1,2,3]) =>`
- `reduce(lambda x,y: x+y, map(lambda t: t[0], [(1,2), (3,4), (5,6)]))=>`

Start Spark-shell

```
$spark-shell
```

```
[root@quickstart 201402_babs_open_data]# spark-shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/jars/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
```



```
Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type 'help' for more information.
```

Testing SparkContext

Spark-context

```
scala> sc
```

```
scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@18c07e25
```

Spark Program in Scala: WordCount

```
scala> val file =  
sc.textFile("hdfs:///user/cloudera/input/pg2600.txt")
```

```
scala> val wc = file.flatMap(l => l.split(" ")).map(word =>  
(word, 1)).reduceByKey(_ + _)
```

```
scala>  
wc.saveAsTextFile("hdfs:///user/cloudera/output/wordcountScala  
")
```

HUE [Home](#) [Query Editors](#) [Data Browsers](#) [Workflows](#) [Search](#) [Security](#)

[File Browser](#)

Search for file name Actions

[Home](#) / user / cloudera / output / **wordcountScala**

History Trash

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	↑		cloudera	cloudera	drwxr-xr-x	June 14, 2016 02:05 AM
<input type="checkbox"/>	.		root	cloudera	drwxr-xr-x	June 14, 2016 02:05 AM
<input type="checkbox"/>	_SUCCESS	0 bytes	root	cloudera	-rw-r--r--	June 14, 2016 02:05 AM
<input type="checkbox"/>	part-00000	266.3 KB	root	cloudera	-rw-r--r--	June 14, 2016 02:05 AM
<input type="checkbox"/>	part-00001	272.7 KB	root	cloudera	-rw-r--r--	June 14, 2016 02:05 AM

WordCount output

HUE Home Query Editors Data Browsers Workflows Search Security

File Browser

ACTIONS

Home Page 1 of 67

View as binary

Download

View file location

Refresh

INFO

Last modified June 14, 2016 2:05 a.m.

User root

Group cloudera

Size 266.3 KB

Mode 100644

/ user / cloudera / output / wordcountScala / part-00000

```
(Ermolov.,2)
(mattered,2)
(Ah!,5)
(Koko,1)
(reunion,2)
(denied?",1)
(muslin,,1)
(intimately,3)
(blandly,5)
("Ho!,1)
(wobbers,1)
.lost...,1)
(fought?,1)
(signal.,1)
(Chem,3)
(Friend,1)
(think,",3)
(wasn't,5)
(Fve 1)
```

Spark Program in Python: WordCount

```
$ pyspark
>>> from operator import add
>>> file =
sc.textFile("hdfs://user/cloudera/input/pg2600.txt")
>>> wc = file.flatMap(lambda x: x.split(' ')).map(lambda x:
(x, 1)).reduceByKey(add)
>>> wc.saveAsTextFile("hdfs://user/cloudera/output/
wordcountPython")
```

File Browser

Name	Size	User	Group	Permissions	Date
wordcountPython		guest1	guest1	drwxrwxrwx	January 23, 2016 11:24 PM
.		ubuntu	guest1	drwxr-xr-x	January 23, 2016 11:32 PM
wordcountScala		ubuntu	guest1	drwxr-xr-x	January 23, 2016 11:32 PM
		ubuntu	guest1	drwxr-xr-x	January 23, 2016 11:24 PM

Transformations

```
>>> nums = sc.parallelize([1,2,3])
>>> squared = nums.map(lambda x : x*x)
>>> even = squared.filter(lambda x: x%2 == 0)
>>> evens = nums.flatMap(lambda x: range(x))
```

Actions

```
>>> nums = sc.parallelize([1,2,3])
>>> nums.collect()
>>> nums.take(2)
>>> nums.count()
>>> nums.reduce(lambda:x, y:x+y)
>>> nums.saveAsTextFile("hdfs://user/cloudera/output/test")
```

Key-Value Operations

```
>>> pet = sc.parallelize([('cat',1), ('dog',1), ('cat',2)])  
>>> pet2 = pet.reduceByKey(lambda x, y:x+y)  
>>> pet3 = pet.groupByKey()  
>>> pet4 = pet.sortByKey()
```

Spark Program : Toy_data.txt

Upload a data to HDFS

```
$ wget https://s3.amazonaws.com/imcbucket/data/toy_data.txt
$ hadoop fs -put toy_data.txt /user/cloudera/input
```

The screenshot shows a file browser interface with a red dashed border around the content area. At the top, there are navigation links: Home, / user / cloudera / input, and a pencil icon. To the right are History and Trash buttons. Below the header is a table listing files in the /user/cloudera/input directory. The columns are: Name, Size, User, Group, Permissions, and Date. The table contains the following data:

Name	Size	User	Group	Permissions	Date
↑		cloudera	cloudera	drwxr-xr-x	June 14, 2016 01:19 AM
.		cloudera	cloudera	drwxr-xr-x	June 14, 2016 07:32 AM
pg2600.txt	3.1 MB	root	cloudera	-rw-r--r--	June 13, 2016 09:29 PM
toy_data.txt	136 bytes	root	cloudera	-rw-r--r--	June 14, 2016 07:32 AM

Start pyspark

```
$ pyspark
```

Spark Program : Find Big Spenders

```
>>> file_rdd =  
sc.textFile("hdfs://user/cloudera/input/toy_data.txt")  
>>> import json  
>>> json_rdd = file_rdd.map(lambda x: json.loads(x))  
>>> big_spenders = json_rdd.map(lambda x: tuple((x.keys()[0],int(x.values()[0]))))  
>>> big_spenders.reduceByKey(lambda x,y: x + y).filter(lambda x: x[1] > 5).collect()
```

Hands-on:

Loading data from MySQL

MySQL RDS Server on AWS

A RDS Server is running on AWS with the following configuration

```
> database: imc_db
> username: admin
> password: imcinstiute
>addr: imcdb.cmw65obdqfnx.us-west-2.rds.amazonaws.com
[This address may change]
```

DB Instances > imcinstiutedb

Details Recent Events & Logs

Endpoint: imcinstiutedb.cmw65obdqfnx.us-west-2.rds.amazonaws.com:3306 (authorized)

Configuration Details

Engine	MySQL 5.6.22
License Model	General Public License
Created Time	March 24, 2015 at 9:50:55 PM UTC+7
DB Name	imc_db
Username	admin
Option Group	default:mysql-5-6 (in-sync)
Parameter Group	default.mysql5.6 (in-sync)

Security and Network

Availability Zone	us-west-2c
VPC	vpc-cd510ca5
Subnet Group	default (Complete)
Subnets	subnet-c0510ca8 subnet-ce510ca6 subnet-cf510ca7
Security Groups	rds-launch-wizard (sg-59dee33c) (active)
Publicly Accessible	Yes
Port	3306
Certificate Authority	rds-ca-2015 (Mar 5, 2020)

Table in MySQL RDS

Table 1 > country_tbl : Columns: id, country

Result Set Filter:		Q Search	↻	
	id	country		
▶	1	USA		
	2	Canada		
	61	Japan		
	66	Thailand		
	NULL	NULL		

JdbcRDD

```
JdbcRDD( SparkContext, getConnection: () => Connection,  
sql: String, lowerBound: Long, upperBound: Long,  
numPartitions: Int, mapRow: (ResultSet) => T =  
JdbcRDD.resultSetToObjectArray)
```

Download MySQL driver & Start Spark-shell

```
# wget http://central.maven.org/maven2/mysql/mysql-connector-java/5.1.23/mysql-connector-java-5.1.23.jar
```

Running Spark-shell

```
#spark-shell --jars mysql-connector-java-5.1.23.jar
```

```
...  
16/06/28 15:23:35 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.  
SQL context available as sqlContext.
```

```
scala> █
```

Reading JDBC

```
$ scala> :paste

val url="jdbc:mysql://imcdb.cmw65obdqfnx.us-west-
2.rds.amazonaws.com:3306/imc_db"

val username = "admin"

val password = "imcinstitute"

import org.apache.spark.rdd.JdbcRDD

import java.sql.{Connection, DriverManager, ResultSet}
Class.forName("com.mysql.jdbc.Driver").newInstance

val myRDD = new JdbcRDD(sc, () =>
DriverManager.getConnection(url,username,password) ,
"SELECT * FROM country_tbl LIMIT ?, ?",
0, 5, 2, r =>
r.getString("id") + ", " + r.getString("country"))
myRDD.count
myRDD.foreach(println)
```

Output

```
// Exiting paste mode, now interpreting.  
  
[Stage 5:> (0 + 2) / 2]66,  
Thailand  
1, USA  
2, Canada  
url: String = jdbc:mysql://imcdb.cmw65obdqfnx.us-west-2.rds.amazonaws.com:3306/imc_d  
b  
username: String = admin  
password: String = imcinstiute  
import org.apache.spark.rdd.JdbcRDD  
import java.sql.{Connection, DriverManager, ResultSet}  
myRDD: org.apache.spark.rdd.JdbcRDD[String] = JdbcRDD[3] at JdbcRDD at <console>:43
```

Project: Flight

Flight Details Data

http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236

United States Department of Transportation [About DOT](#) | [Briefing Room](#) | [Our Activities](#)

OFFICE OF THE ASSISTANT SECRETARY FOR RESEARCH AND TECHNOLOGY [About OST-R](#) | [Press Room](#) | [Programs](#) | [OST-R Publications](#) | [Library](#) | [Contact Us](#)

Bureau of Transportation Statistics

[About BTS](#) | [BTS Press Room](#) | [Data and Statistics](#) | [Publications](#) | [Subject Areas](#) | [External Links](#)

[OST-R](#) > [BTS](#)

TranStats

Search this site: Advanced Search

Resources

[Database Directory](#)
[Glossary](#)
[Upcoming Releases](#)
[Data Release History](#)

Data Tools

[Analysis](#)
[Table Profile](#)
[Table Contents](#)

On-Time Performance

[Data Tables](#) [Table Contents](#)

[Download Instructions](#) [Filter Geography](#) [Filter Year](#) [Filter Period](#)

Latest Available Data: November 2015

Prezipped File % Missing Documentation Terms

Field Name	Description	Support Table
Time Period		
<input type="checkbox"/> Year	Year	Get Lookup Table
<input type="checkbox"/> Quarter	Quarter (1-4)	Get Lookup Table
<input type="checkbox"/> Month	Month	Get Lookup Table
<input type="checkbox"/> DayofMonth	Day of Month	Get Lookup Table
<input type="checkbox"/> DayOfWeek	Day of Week	Get Lookup Table
<input type="checkbox"/> FlightDate	Flight Date (yyyymmdd)	
Airline		
<input type="checkbox"/> UniqueCarrier	Unique Carrier Code. When the same code has been used by multiple	Get Lookup Table

Flight Details Data

<http://stat-computing.org/dataexpo/2009/the-data.html>



ASA Sections on:

[Statistical Computing](#)
[Statistical Graphics](#)

[[Computing, Graphics](#)]

[[Awards, Data expo, Video library](#)]

[[Events, News, Newsletter](#)]

[Data expo '09](#)

Get the data

The data comes originally from [RITA](#) where it is [described in detail](#). You can download the data there, or from the bzipped csv files listed below. These files have derivable variables removed, are packaged in yearly chunks and have been more heavily compressed than the originals.

Download individual years:

[1987](#), [1988](#), [1989](#), [1990](#), [1991](#), [1992](#), [1993](#), [1994](#), [1995](#), [1996](#), [1997](#), [1998](#), [1999](#), [2000](#), [2001](#),
[2002](#), [2003](#), [2004](#), [2005](#), [2006](#), [2007](#), [2008](#)

Data expo 09

- [Posters & results](#)
- [Competition description](#)
- [Download the data](#)
- [Supplemental data sources](#)
- [Using a database](#)
- [Intro to command line tools](#)

Data Description

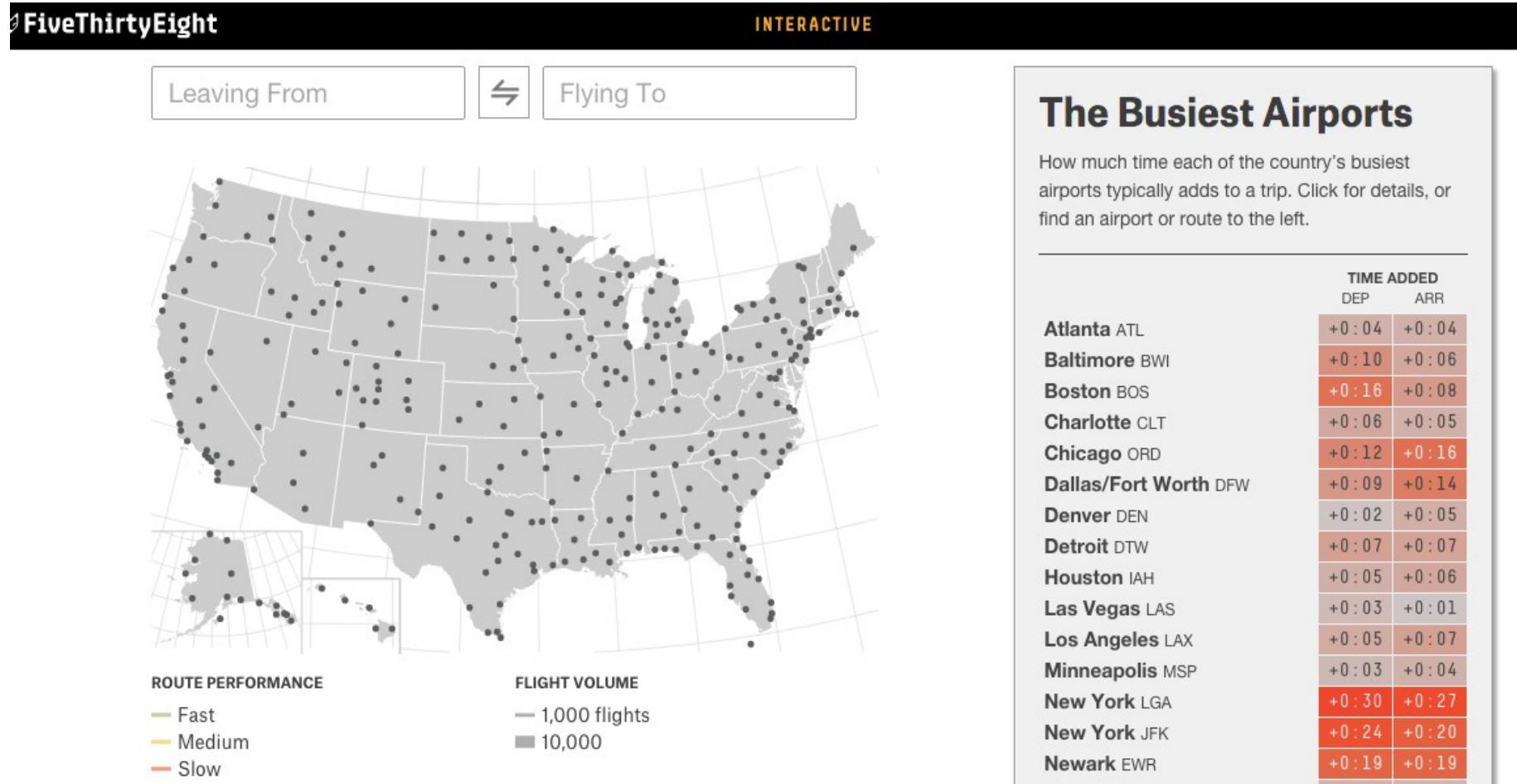
Name	Description
1 Year	1987-2008
2 Month	1-12
3 DayofMonth	1-31
4 DayOfWeek	1 (Monday) - 7 (Sunday)
5 DepTime	actual departure time (local, hhmm)
6 CRSDepTime	scheduled departure time (local, hhmm)
7 ArrTime	actual arrival time (local, hhmm)
8 CRSArrTime	scheduled arrival time (local, hhmm)
9 UniqueCarrier	<u>unique carrier code</u>
10 FlightNum	flight number
11 TailNum	plane tail number
12 ActualElapsedTime	in minutes
13 CRSElapsedTime	in minutes
14 AirTime	in minutes
15 ArrDelay	arrival delay, in minutes
16 DepDelay	departure delay, in minutes
17 Origin	origin <u>IATA airport code</u>
18 Dest	destination <u>IATA airport code</u>
19 Distance	in miles
20 TaxiIn	taxi in time, in minutes
21 TaxiOut	taxi out time in minutes
22 Cancelled	was the flight cancelled?
23 CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24 Diverted	1 = yes, 0 = no
25 CarrierDelay	in minutes
26 WeatherDelay	in minutes
27 NASDelay	in minutes
28 SecurityDelay	in minutes
29 LateAircraftDelay	in minutes

Snapshot of Dataset

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
Year	Month	DayofMo	DayOfWe	DepTime	CRSDepT	ArrTime	CRSArrT	UniqueCa	FlightNum	TailNum	ActualElap	CRSElapse	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance	TaxiIn	TaxiOut
2008	1	5	6	2243	1415	45	1625	WN	1684	N347SW	62	70	41	500	508	SAN	PHX	304	2	
2008	1	5	6	1940	1220	2111	1350	WN	1684	N347SW	91	90	64	441	440	SFO	SAN	447	5	
2008	1	7	1	111	1845	308	2045	WN	405	N644SW	117	120	103	383	386	MDW	JAN	666	4	
2008	1	7	1	2213	1700	2317	1655	WN	1827	N759GS	124	55	75	382	313	IND	MDW	162	10	
2008	1	7	1	2143	1720	26	1820	WN	1430	N644SW	163	60	83	366	263	STL	MDW	251	24	
2008	1	7	1	117	2020	302	2135	WN	490	N651SW	105	75	87	327	297	STL	TUL	351	5	
2008	1	7	1	2358	1855	105	2000	WN	490	N651SW	67	65	50	305	303	MDW	STL	251	4	
2008	1	3	4	2245	1730	2354	1850	WN	186	N792SW	69	80	59	304	315	JAN	HOU	359	3	
2008	1	7	1	2219	1730	35	1935	WN	2474	N710SW	76	65	67	300	289	MDW	CMH	284	2	
2008	1	5	6	2129	1620	2246	1750	WN	1924	N408WN	77	90	56	296	309	SFO	LAS	414	4	
2008	1	3	4	1615	1130	1623	1135	WN	10	N617SW	68	65	56	288	285	MAF	ABQ	332	4	
2008	1	3	4	1736	1305	2031	1555	WN	1837	N761RR	255	290	268	276	271	MDW	SFO	1855	4	
2008	1	5	6	2236	1805	2400	1930	WN	646	N283WN	84	85	71	270	271	LAX	SFO	337	6	
2008	1	3	4	2021	1700	2303	1835	WN	2005	N302SW	162	95	73	268	201	LAS	SFO	414	4	
2008	1	3	4	2059	1620	2216	1750	WN	1924	N761RR	77	90	60	266	279	SFO	LAS	414	6	
2008	1	7	1	2348	2105	307	2250	WN	3137	N358SW	259	165	244	257	163	MCO	MDW	989	1	
2008	1	3	4	2255	1820	509	55	WN	1924	N761RR	194	215	176	254	275	LAS	IND	1591	9	
2008	1	9	3	1458	1040	1725	1315	WN	2556	N501SW	87	95	76	250	258	BNA	BWI	588	4	
2008	1	7	1	2300	1835	113	2105	WN	2804	N420WN	253	270	240	248	265	MDW	PDX	1751	5	
2008	1	5	6	47	2040	151	2145	WN	505	N435WN	64	65	51	246	247	BWI	PVD	328	5	
2008	1	5	6	1558	1225	14	2010	WN	505	N442WN	316	285	250	244	213	SAN	BWI	2295	5	
2008	1	5	6	1931	1540	2104	1705	WN	1179	N718SW	93	85	77	239	231	SAN	OAK	446	7	
2008	1	4	5	1822	1425	2003	1605	WN	753	N726SW	101	100	88	238	237	PDX	OAK	543	6	

FiveThirtyEight

<http://projects.fivethirtyeight.com/flights/>



Spark Program : Upload Flight Delay Data

Upload a data to HDFS

```
$ wget  
https://s3.amazonaws.com/imcbucket/data/flights/2008.csv  
$ hadoop fs -put 2008.csv /user/cloudera/input
```

The screenshot shows the Hue File Browser interface. The top navigation bar includes links for Query Editors, Data Browsers, Workflows, Search, and Security. Below the navigation is a toolbar with icons for file operations like Actions, Move to trash, and a search bar labeled "Search for file name". The main area displays a file tree under the path "/user/cloudera/input". The tree shows three entries: a folder named "2008", a file named "2008.csv", and a file named ".". The "2008.csv" file is highlighted. At the bottom, there are buttons for History and Trash, and a table showing detailed information for each entry.

Name	Size	User	Group	Permissions	Date
2008		cloudera	cloudera	drwxr-xr-x	June 14, 2016 08:54 AM
.		root	cloudera	drwxr-xr-x	June 14, 2016 08:56 AM
2008.csv	657.5 MB	root	cloudera	-rw-r--r--	June 14, 2016 08:56 AM

Spark Program : Navigating Flight Delay Data

```
>>> airline =  
sc.textFile("hdfs://user/cloudera/input/2008.csv")  
>>> airline.take(2)
```

```
[u'Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCARRIER,  
FlightNum,TailNum,ActualElapsedTime,CRSElapsedTime,AirTime,ArrDelay,DepDelay,  
Origin,Dest,Distance,TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverted,CARRIERDelay,  
WeatherDelay,NASDelay,SecurityDelay,LateAircraftDelay', u'2008,1,3,4,2003,1955,2211,2225,WN,335,N712SW,128,150,116,-14,8,IAD,TPA,810,4,8,0,,0,NA,NA,NA,  
NA,NA']
```

Spark Program : Preparing Data

```
>>> header_line = airline.first()
>>> header_list = header_line.split(',')
>>> airline_no_header = airline.filter(lambda row: row != header_line)
>>> airline_no_header.first()
>>> def make_row(row):
...     row_list = row.split(',')
...     d = dict(zip(header_list, row_list))
...     return d
...
>>> airline_rows = airline_no_header.map(make_row)
>>> airline_rows.take(5)
```

Spark Program : Define convert_float function

```
>>> def convert_float(value):
...     try:
...         x = float(value)
...         return x
...     except ValueError:
...         return 0
...
>>>
```

Spark Program : Finding best/worst airline

```
>>> carrier_rdd = airline_rows.map(lambda row:  
(row['UniqueCarrier'],convert_float(row['ArrDelay'])))  
>>> carrier_rdd.take(2)
```

```
16/06/17 17:48:06 INFO scheduler.DAGScheduler: Job 5 finished: runJob at PythonRDD.s  
cala:393, took 0.062345 s  
[(u'WN', -14.0), (u'WN', 2.0)]
```

Spark Program : Finding best/worst airlines

```
>>> mean_delays_dest =  
carrier_rdd.groupByKey().mapValues(lambda delays:  
sum(delays.data)/len(delays.data))  
  
>>> mean_delays_dest.sortBy(lambda t:t[1],  
ascending=True).take(10)  
  
>>> mean_delays_dest.sortBy(lambda t:t[1],  
ascending=False).take(10)
```

```
[..., ...]  
[(u'AQ', -2.8708974358974357), (u'HA', 1.2518519716624075), (u'US', 2.80099826053982  
78), (u'9E', 3.9874908469611912), (u'AS', 4.721360405553864), (u'WN', 5.115703380225  
9031), (u'F9', 6.0841356696810847), (u'OO', 6.4389386397817896), (u'NW', 7.293465879  
6727758), (u'DL', 7.7161646357519178)]
```

```
[(u'AA', 12.202853434950445), (u'OH', 11.404110178283158), (u'YV', 11.32256697917075  
3), (u'UA', 11.001550560048052), (u'B6', 10.859381613638567), (u'CO', 10.80982057596  
6226), (u'XE', 10.320298523403915), (u'EV', 10.00033146217589), (u'MQ', 9.4969706109  
522658), (u'FL', 8.9881574723712561)]
```

Spark SQL

DataFrame

- A distributed collection of rows organized into named columns.
- An abstraction for selecting, filtering, aggregating, and plotting structured data.
- Previously => SchemaRDD

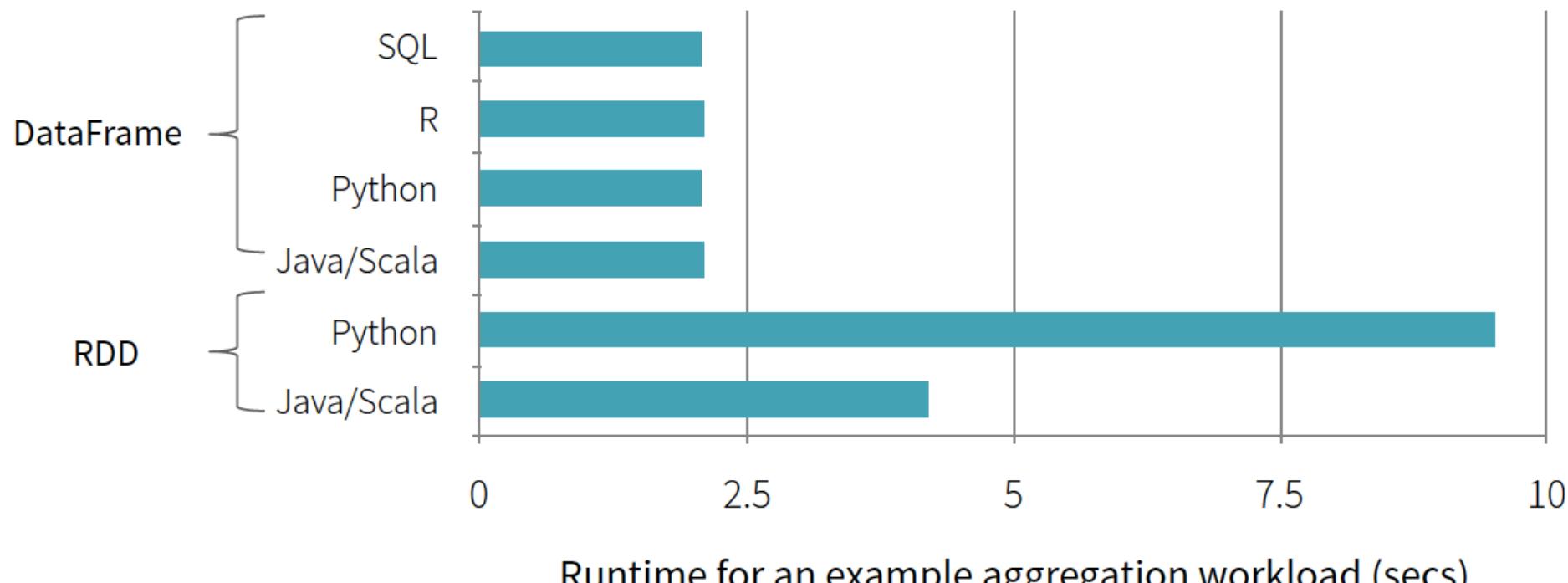
SparkSQL

- Creating and running Spark program faster
 - Write less code
 - Read less data
 - Let the optimizer do the hard work



is about more than SQL.

Benefit of Logical Plan: Performance Parity Across Languages



Source: Jump start into Apache Spark and Databricks

SparkSQL can leverage the Hive metastore

- Hive Metastore can also be leveraged by a wide array of applications
 - Spark
 - Hive
 - Impala
- Available from HiveContext

SparkSQL

```
context = ps.HiveContext(sc)

# query with SQL
results = context.sql(
    "SELECT * FROM people")

# apply Python transformation
names = results.map(lambda p: p.name)
```

Spark SQL

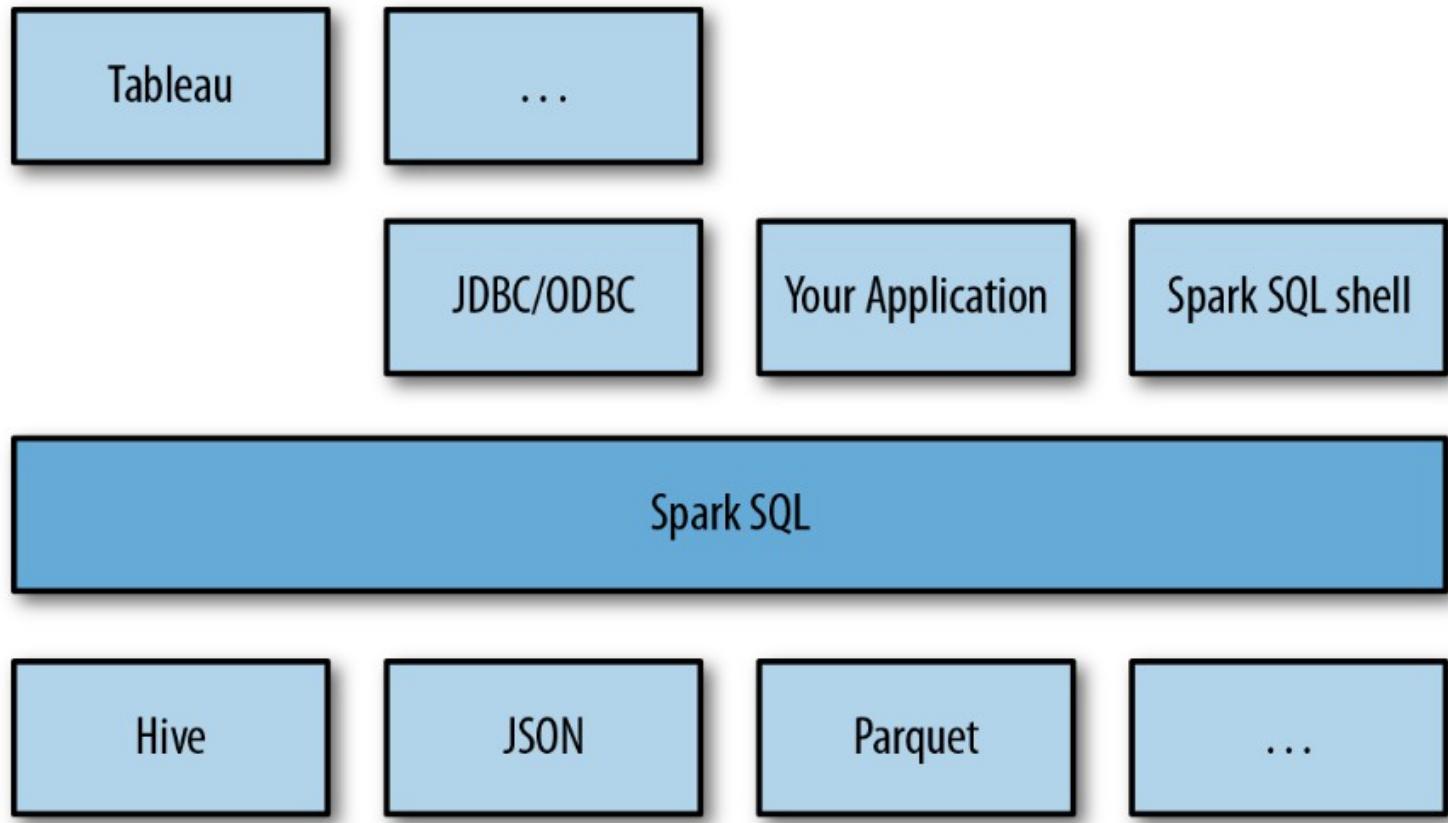
Spark Core

Unified interface for structured data



Image credit: <http://barrymieny.deviantart.com/>

Spark SQL usage



Source: Learning Spark, O'Reilly Media, Inc.

Hands-on: Spark SQL

Link Hive Metastore with Spark-Shell

```
scala > val sqlContext = new
org.apache.spark.sql.hive.HiveContext(sc)

scala> sqlContext.sql("CREATE TABLE IF NOT EXISTS movie(userid
STRING, movieid STRING, rating INT, timestamp STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY
'\n'")

scala> sqlContext.sql("LOAD DATA LOCAL INPATH '/root/ml-
100k/u.data' INTO TABLE movie")

scala> val result = sqlContext.sql("SELECT * FROM movie")

scala> result.show()
```

Link Hive Metastore with Spark-Shell

Copy the configuration file

```
$cp /usr/lib/hive/conf/hive-site.xml /usr/lib/spark/conf/
```

Running Spark

```
$spark-shell
```

Exercise 1: HiveContext

```
scala > val sqlContext = new
org.apache.spark.sql.hive.HiveContext(sc)

scala> sqlContext.sql("CREATE TABLE IF NOT EXISTS movie(userid
STRING, movieid STRING, rating INT, timestamp STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY
'\n'")

scala> sqlContext.sql("LOAD DATA LOCAL INPATH '/root/ml-
100k/u.data' INTO TABLE movie")

scala> val result = sqlContext.sql("SELECT * FROM movie")

scala> result.show()
```

HUE Home Query Editors Data Browsers Workflows Search Security

File Browser

Search for file name Actions Move to trash

Home / user / hive / warehouse History Trash

Name	Size	User	Group	Permissions	Date
..		hive	supergroup	drwxrwxrwx	April 05, 2016 07:27 PM
movie		hive	supergroup	drwxrwxrwx	June 29, 2016 08:16 PM
rating		root	supergroup	drwxrwxrwx	June 29, 2016 08:16 PM
		cloudera	supergroup	drwxrwxrwx	June 29, 2016 07:27 PM



HUE [Home](#) [Query Editors](#) [Data Browsers](#) [Workflows](#) [Search](#) [Security](#)

[Databases](#) > default

[STATS](#)

Default Hive database public (ROLE) Location

[TABLES](#)

Search for a table... [View](#) [Browse Data](#) [Drop](#)

<input type="checkbox"/>	Table Name	Comment	Type
<input type="checkbox"/>	movie		grid
<input type="checkbox"/>	rating		grid

Exercise 2: JSON Data

Upload a data to HDFS

```
$ wget https://s3.amazonaws.com/imcbucket/data/hue.json  
$ hadoop fs -put hue.json /user/cloudera
```

Spark SQL : Preparing data

```
>>> from pyspark.sql import HiveContext
>>> hiveCtx = HiveContext(sc)
>>> input =
hiveCtx.read.json("hdfs:///user/cloudera/hue.json")
>>> input.registerTempTable("testTable")
>>> input.printSchema()

root
 |-- fields: struct (nullable = true)
    |-- action: string (nullable = true)
    |-- app: string (nullable = true)
    |-- app_label: string (nullable = true)
    |-- app_name: string (nullable = true)
    |-- applied: string (nullable = true)
    |-- codename: string (nullable = true)
    |-- content_type: long (nullable = true)
    |-- description: string (nullable = true)
    |-- domain: string (nullable = true)
    |-- group: long (nullable = true)
    |-- hue_permission: long (nullable = true)
    |-- migration: string (nullable = true)
    |-- model: string (nullable = true)
    |-- name: string (nullable = true)
    |-- permissions: array (nullable = true)
        |-- element: string (containsNull = true)
    |-- model: string (nullable = true)
    |-- pk: long (nullable = true)
```

Spark SQL : Query Data

```
>>> hiveCtx.sql("SELECT * FROM testTable") .show()
```

fields	model	pk
[null,null,auth,n...	contenttypes.cont...	1
[null,null,auth,n...	contenttypes.cont...	2
[null,null,auth,n...	contenttypes.cont...	3
[null,null,django...	contenttypes.cont...	4
[null,null,django...	contenttypes.cont...	5
[null,null,django...	contenttypes.cont...	6
[null,null,conten...	contenttypes.cont...	7
[null,null,sessio...	contenttypes.cont...	8
[null,null,sites,...	contenttypes.cont...	9
[null,null,admin,...	contenttypes.cont...	10
[null,null,south,...	contenttypes.cont...	11

Spark SQL : Query Data

```
>>> hiveCtx.sql("SELECT * FROM testTable").collect()

[Row(fields=Row(action=None, app=None, app_label=None, app_name=None, applied=None, codename=u'change_dataoutput', content_type=44, description=None, domain=None, group=None, hue_permission=None, migration=None, model=None, name=u'Can change data output', permissions=None), model=u'auth.permission', pk=127), Row(fields=Row(action=None, app=None, app_label=None, app_name=None, applied=None, codename=u'delete_dataoutput', content_type=44, description=None, domain=None, group=None, hue_permission=None, migration=None, model=None, name=u'Can delete data output', permissions=None), model=u'auth.permission', pk=128), Row(fields=Row(action=None, app=None, app_label=None, app_name=None, applied=None, codename=u'add_bundledcoordinator', content_type=45, description=None, domain=None, group=None, hue_permission=None, migration=None, model=None, name=u'Can add bundled coordinator', permissions=None), model=u'auth.permission', pk=129), Row(fields=Row(action=None, app=None, app_label=None, app_name=None, applied=None, codename=u'change_bundledcoordinator', content_type=45, description=None, domain=None, group=None, hue_permission=None, migration=None, model=None, name=u'Can change bundled coordinator', permissions=None), model=u'auth.permission', pk=130)]
```

Exercise 3 SQL Spark MovieLens (csv data)

Upload a data to HDFS then

```
$ pyspark --packages com.databricks:spark-csv_2.10:1.2.0

>>> df =
sqlContext.read.format('com.databricks.spark.csv') .options(header='false') .load('hdfs://user/cloudera/u.user')

>>> df.registerTempTable('user')

>>> sqlContext.sql("SELECT * FROM user") .show()
```

```
+-----+  
|          C0 |  
+-----+  
| 1|24|M|technician...| |
| 2|53|F|other|94043|  
| 3|23|M|writer|32067|  
| 4|24|M|technician...|  
| 5|33|F|other|15213|  
| 6|42|M|executive|...|  
| 7|57|M|administra...|  
| 8|36|M|administra...|  
| 9|29|M|student|01002|  
| 10|53|M|lawyer|90703|  
| 11|39|F|other|30329|  
| 12|28|F|other|06405|  
| 13|47|M|educator|...|  
| 14|45|M|scientist...|  
| 15|49|F|educator|...|  
| 16|21|M|entertain...|  
| 17|30|M|programme...|  
| 18|35|F|other|37212|  
| 19|40|M|librarian...|  
| 20|42|F|homemaker...|  
+-----+  
only showing top 20 rows
```

Exercise 4: Spark SQL Meals Data

Upload a data to HDFS

```
$ wget https://s3.amazonaws.com/imcbucket/data/events.txt
$ wget https://s3.amazonaws.com/imcbucket/data/meals.txt
$ hadoop fs -put events.txt /user/cloudera/input
$ hadoop fs -put meals.txt /user/cloudera/input
```

Spark SQL : Preparing data

```
>>> meals_rdd =  
sc.textFile("hdfs://user/cloudera/input/meals.txt")  
  
>>> events_rdd =  
sc.textFile("hdfs://user/cloudera/input/events.txt")  
  
>>> header_meals = meals_rdd.first()  
  
>>> header_events = events_rdd.first()  
  
>>> meals_no_header = meals_rdd.filter(lambda row:row !=  
header_meals)  
  
>>> events_no_header =events_rdd.filter(lambda row:row !=  
header_events)  
  
>>> meals_json = meals_no_header.map(lambda  
row:row.split(';')).map(lambda row_list:  
dict(zip(header_meals.split(';'), row_list)))  
  
>>> events_json = events_no_header.map(lambda  
row:row.split(';')).map(lambda row_list:  
dict(zip(header_events.split(';'), row_list)))
```

Spark SQL : Preparing data

```
>>> import json
>>> def type_conversion(d, columns) :
...     for c in columns:
...         d[c] = int(d[c])
...     return d
...
...
>>> meal_typed = meals_json.map(lambda
j:json.dumps(type_conversion(j, ['meal_id','price'])))
{'type": "french", "dt": "2013-01-01", "meal_id": 1, "price": 10}

>>> event_typed = events_json.map(lambda
j:json.dumps(type_conversion(j, ['meal_id','userid'])))
{'meal_id": 18, "dt": "2013-01-01", "userid": 3, "event": "bought"}
```

Spark SQL : Create DataFrame

```
>>> meals_dataframe = sqlContext.jsonRDD(meal_typed)
>>> events_dataframe = sqlContext.jsonRDD(event_typed)
>>> meals_dataframe.head()

Row(dt=u'2013-07-10', meal_id=1018, price=13, type=u'italian')

>>> meals_dataframe.printSchema()

root
 |-- dt: string (nullable = true)
 |-- meal_id: long (nullable = true)
 |-- price: long (nullable = true)
 |-- type: string (nullable = true)
```

Spark SQL : Running SQL Query

```
>>> meals_dataframe.registerTempTable('meals')
>>> events_dataframe.registerTempTable('events')
>>> sqlContext.sql("SELECT * FROM meals LIMIT 5").collect()
```

```
[Row(dt=u'2013-07-10', meal_id=1018, price=13, type=u'italian'), Row(dt=u'2013-07-11', meal_id=1019, price=11, type=u'japanese'), Row(dt=u'2013-07-11', meal_id=1020, price=14, type=u'italian'), Row(dt=u'2013-07-11', meal_id=1021, price=14, type=u'italian'), Row(dt=u'2013-07-11', meal_id=1022, price=13, type=u'mexican')]
]
```

```
>>> meals_dataframe.take(5)
```

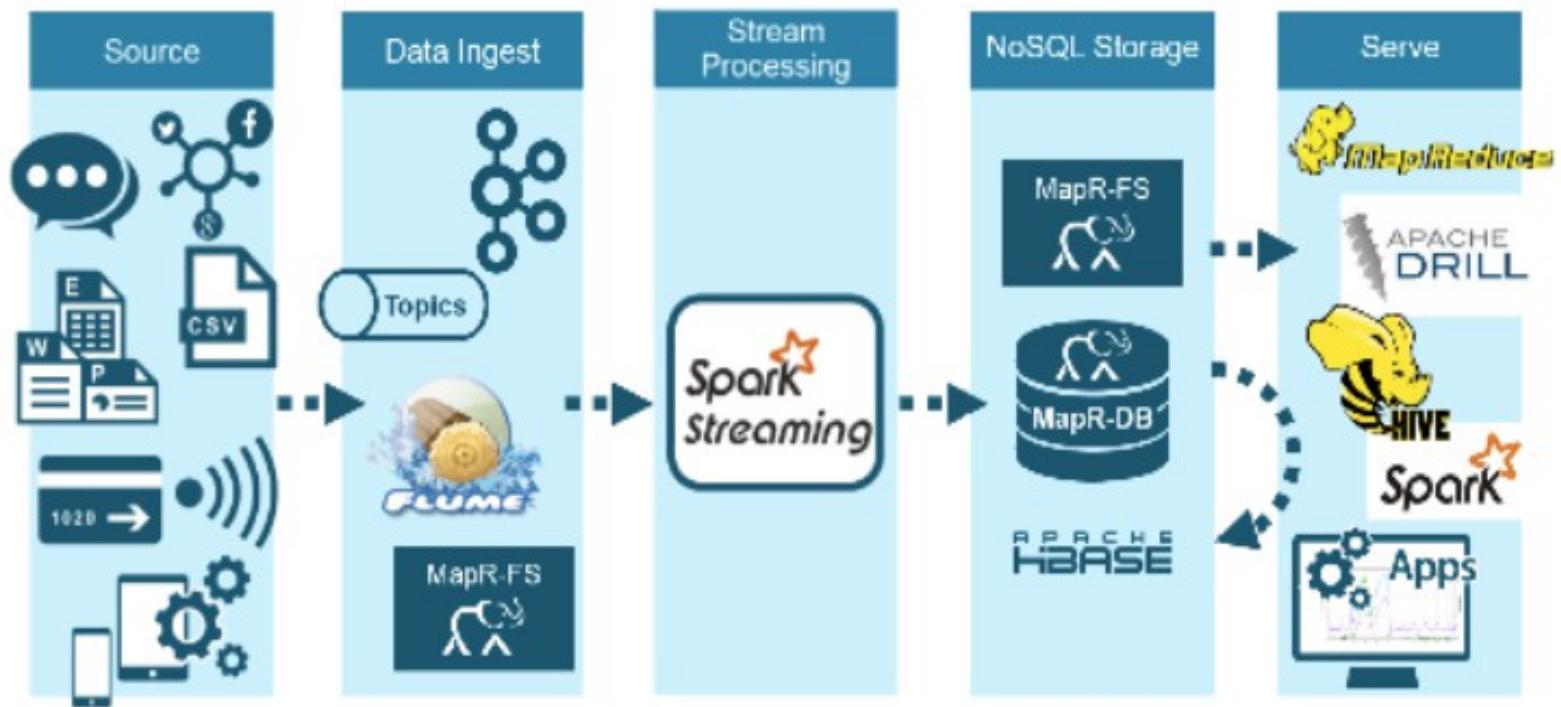
Spark SQL : More complex query

```
>>> sqlContext.sql("""  
...     SELECT type, COUNT(type) AS cnt FROM  
...             meals  
...     INNER JOIN  
...             events on meals.meal_id = events.meal_id  
...     WHERE  
...             event = 'bought'  
...     GROUP BY  
...             type  
...     ORDER BY cnt DESC  
... """).collect()
```

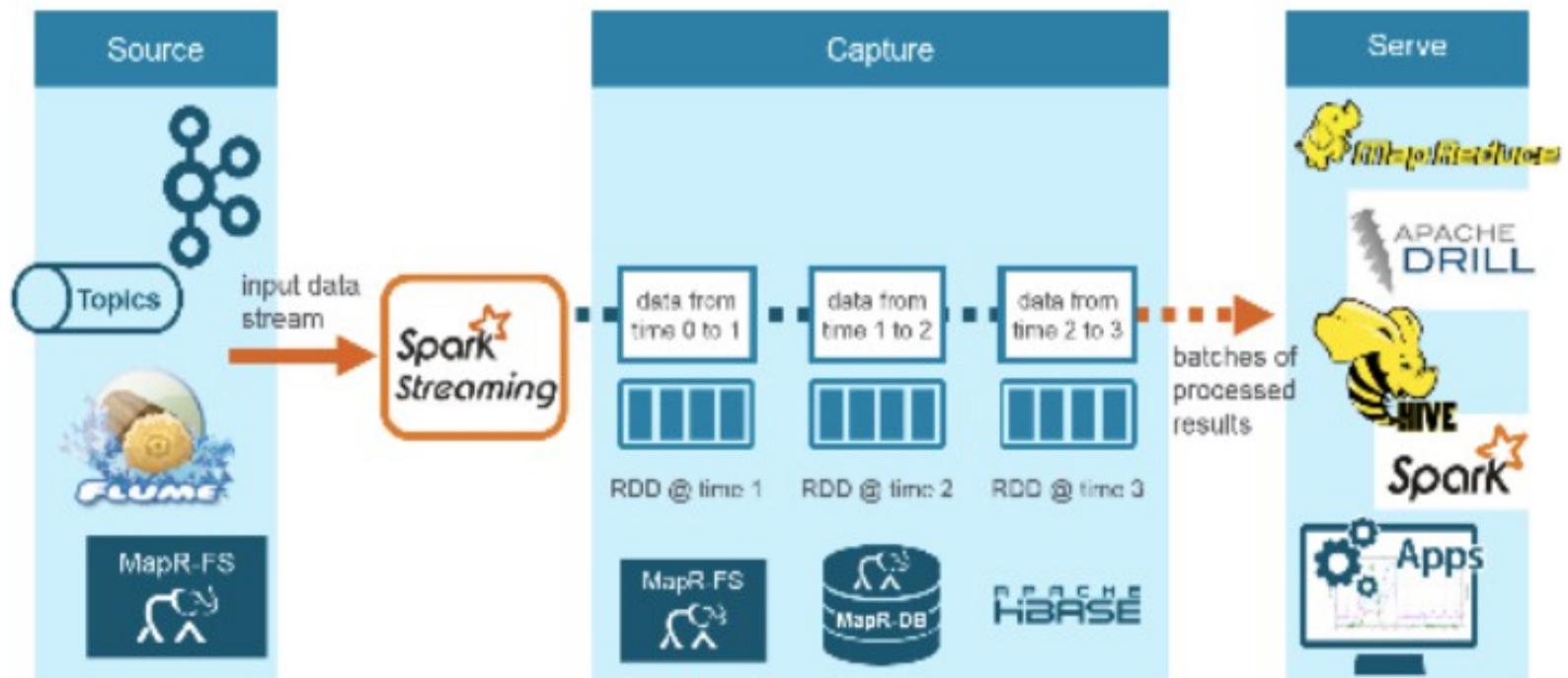
```
[Row(type=u'italian', cnt=22575), Row(type=u'french', cnt=16179), Row(type=u'mexican', cnt=8792), Row(type=u'japanese', cnt=6921), Row(type=u'chinese', cnt=6267), Row(type=u'vetnamese', cnt=3535)]
```

Spark Streaming

Stream Process Architecture

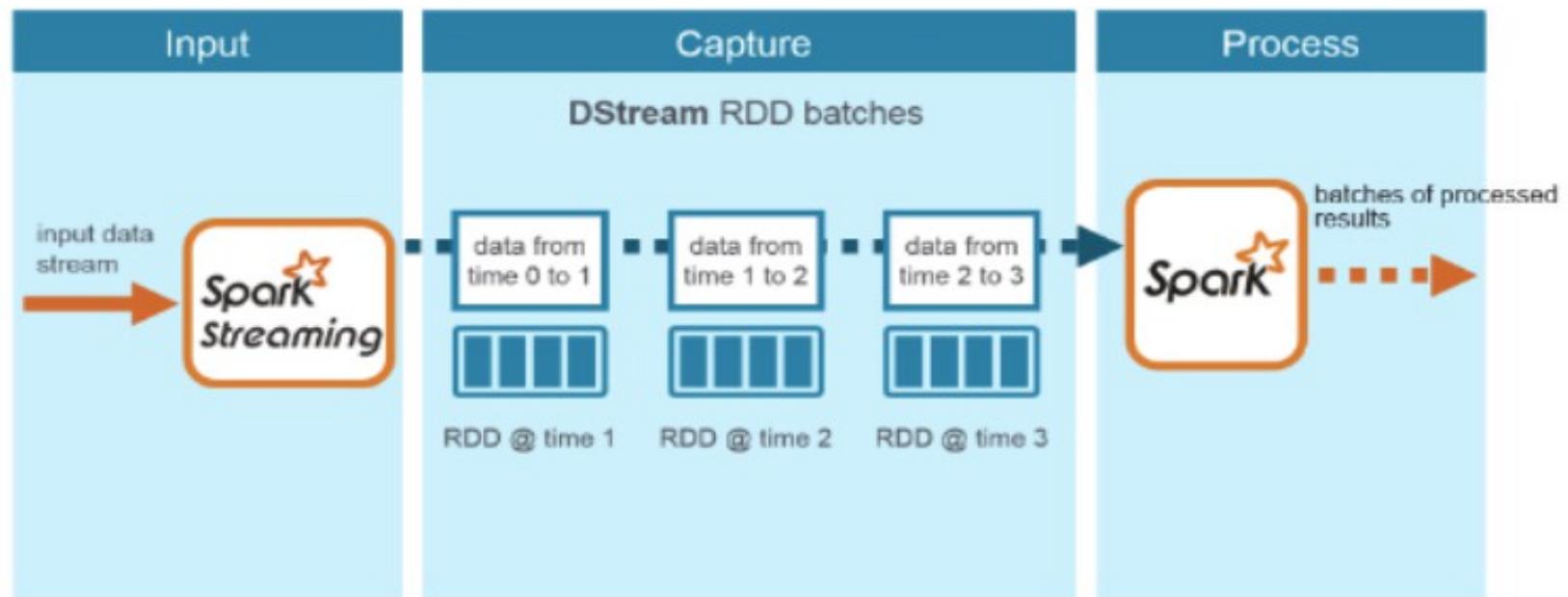


Spark Streaming Architecture

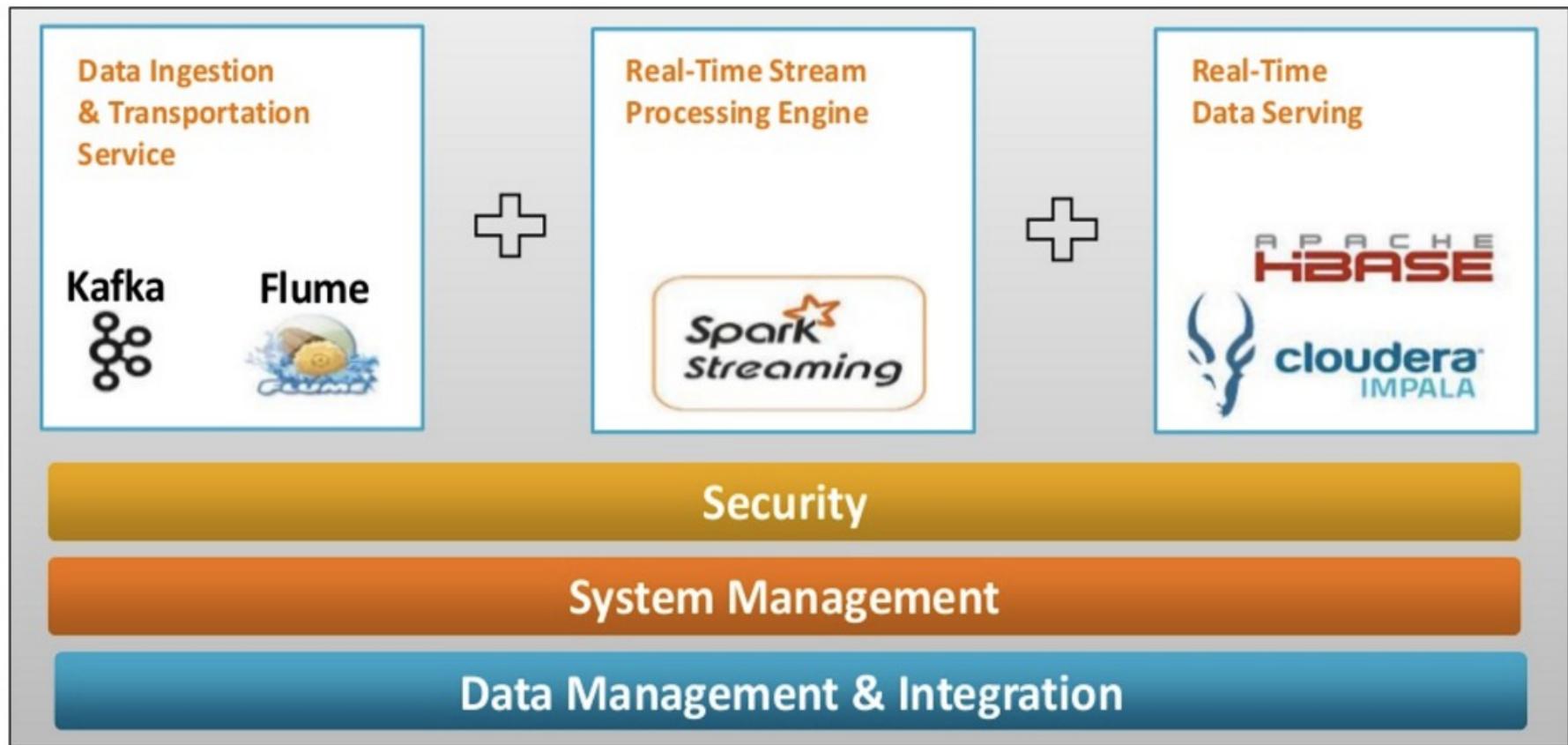


Processing Spark DStreams

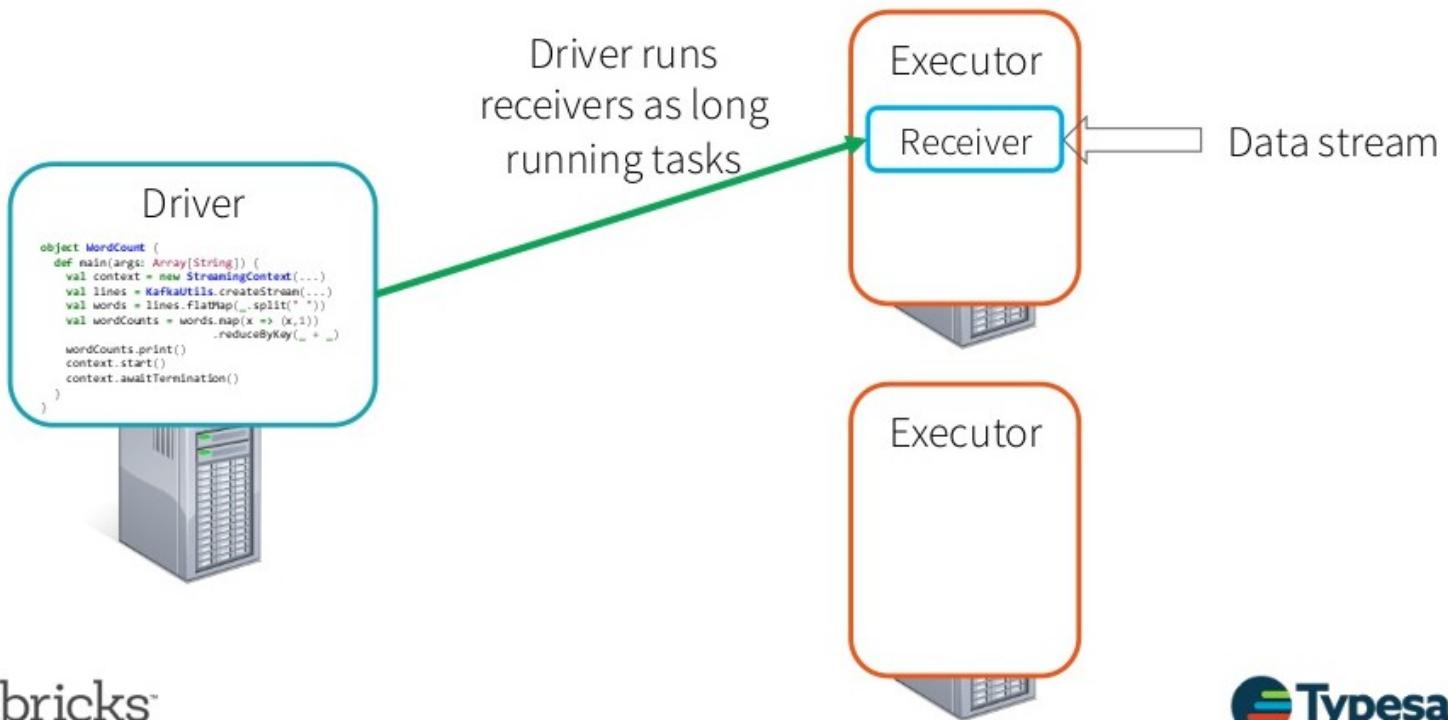
Processed results are pushed out in batches



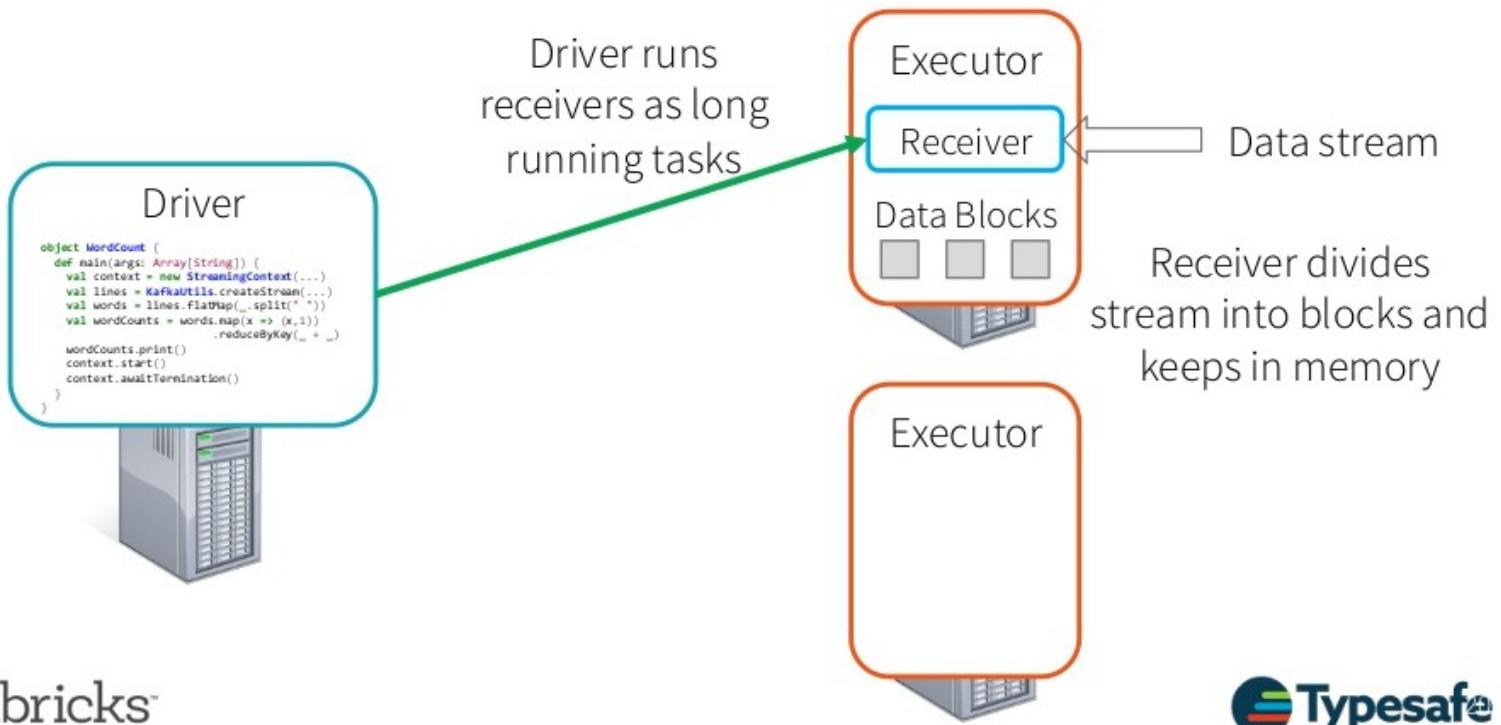
Streaming Architecture



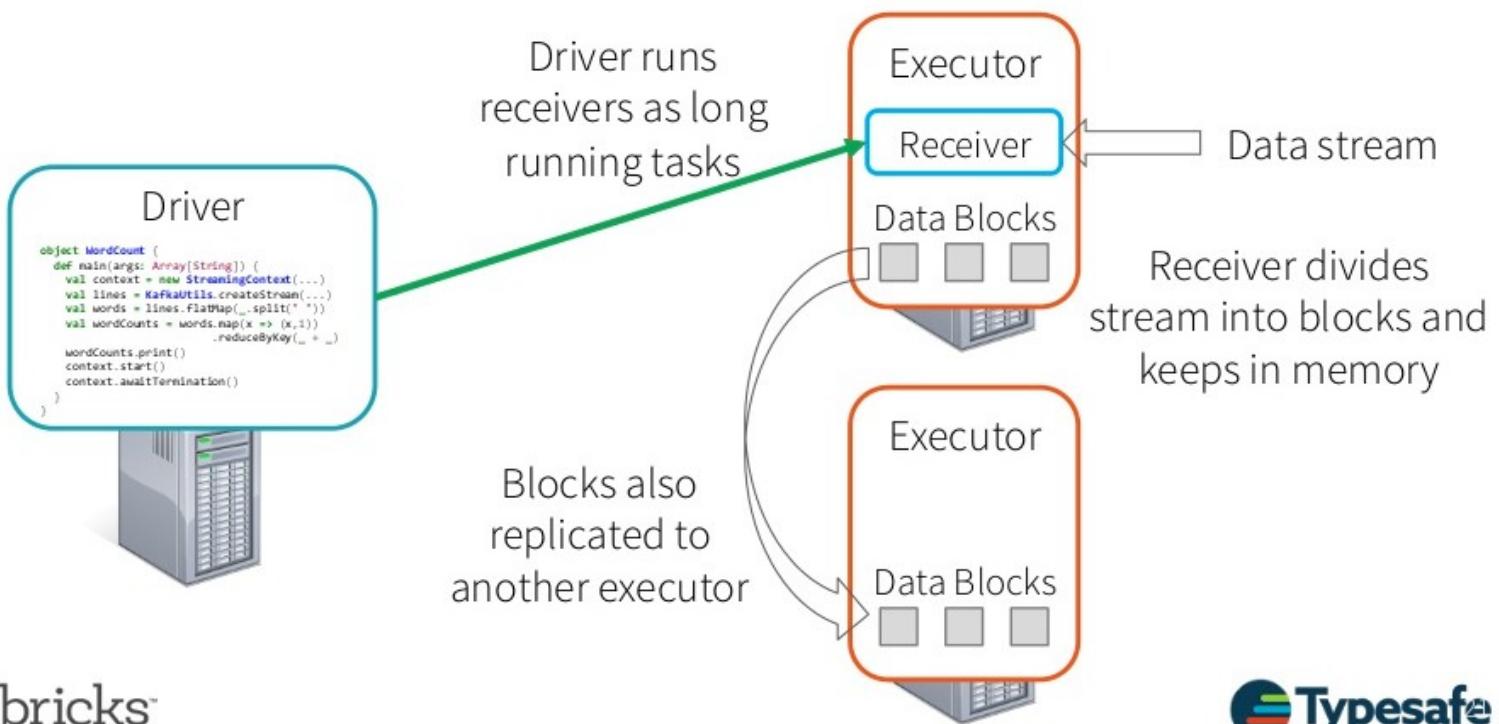
Spark Streaming Application: Receive data



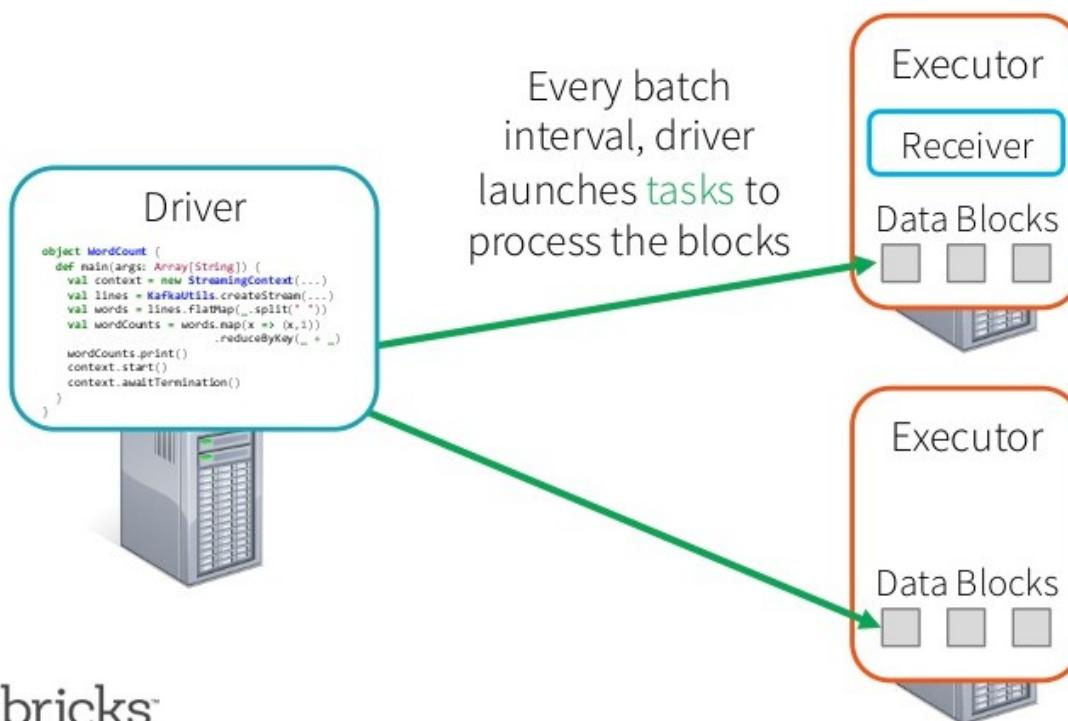
Spark Streaming Application: Receive data



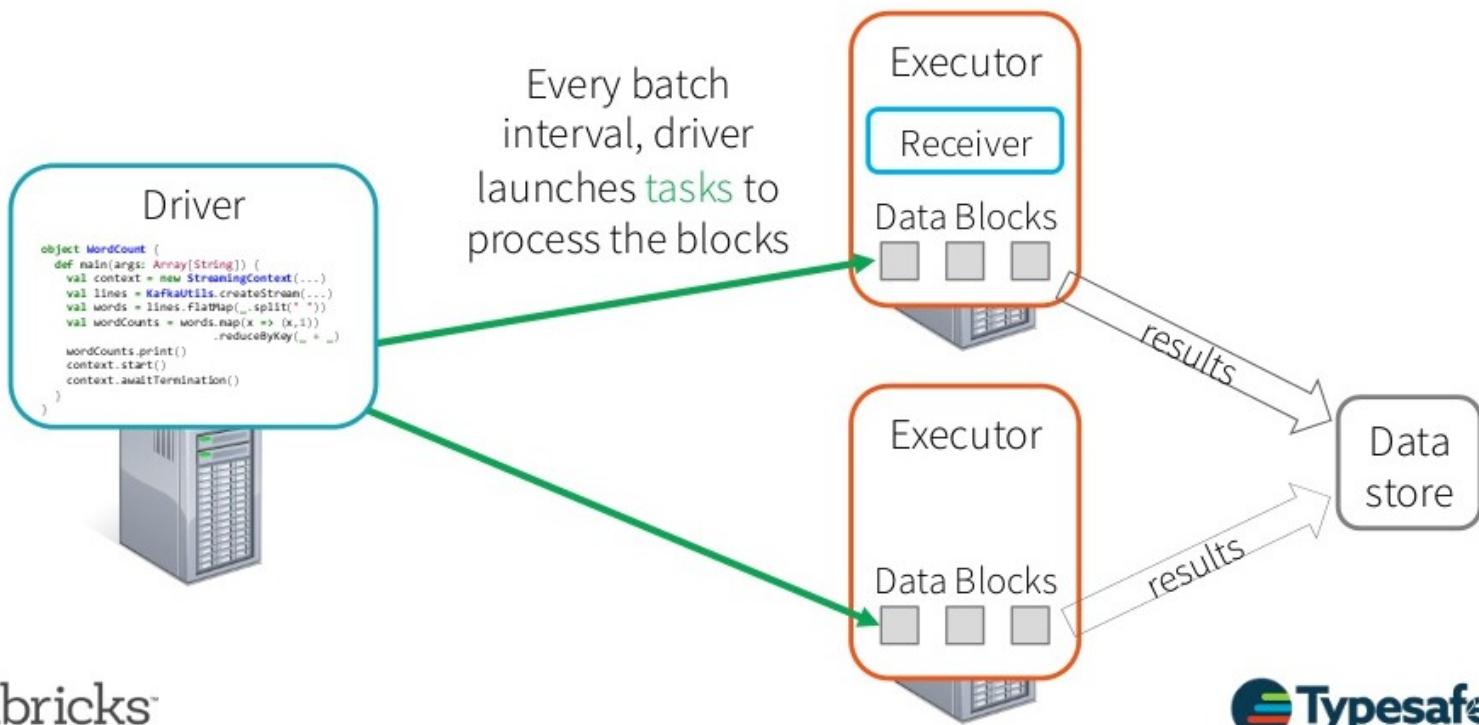
Spark Streaming Application: Receive data



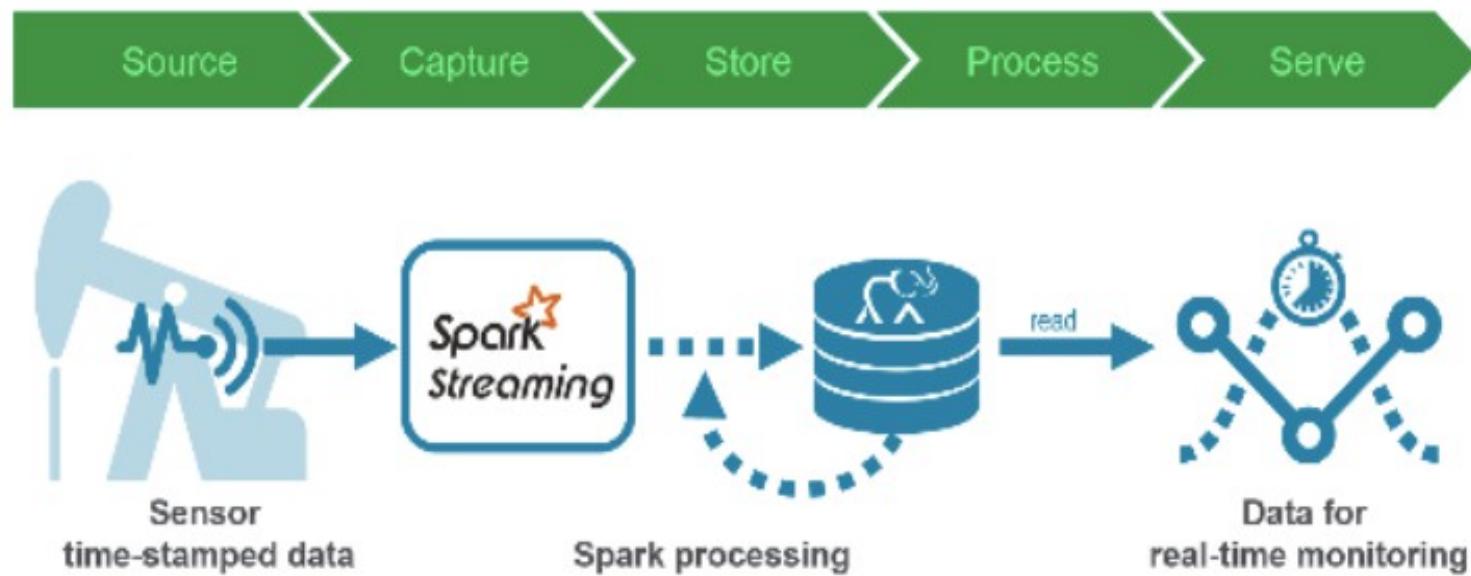
Spark Streaming Application: Process data



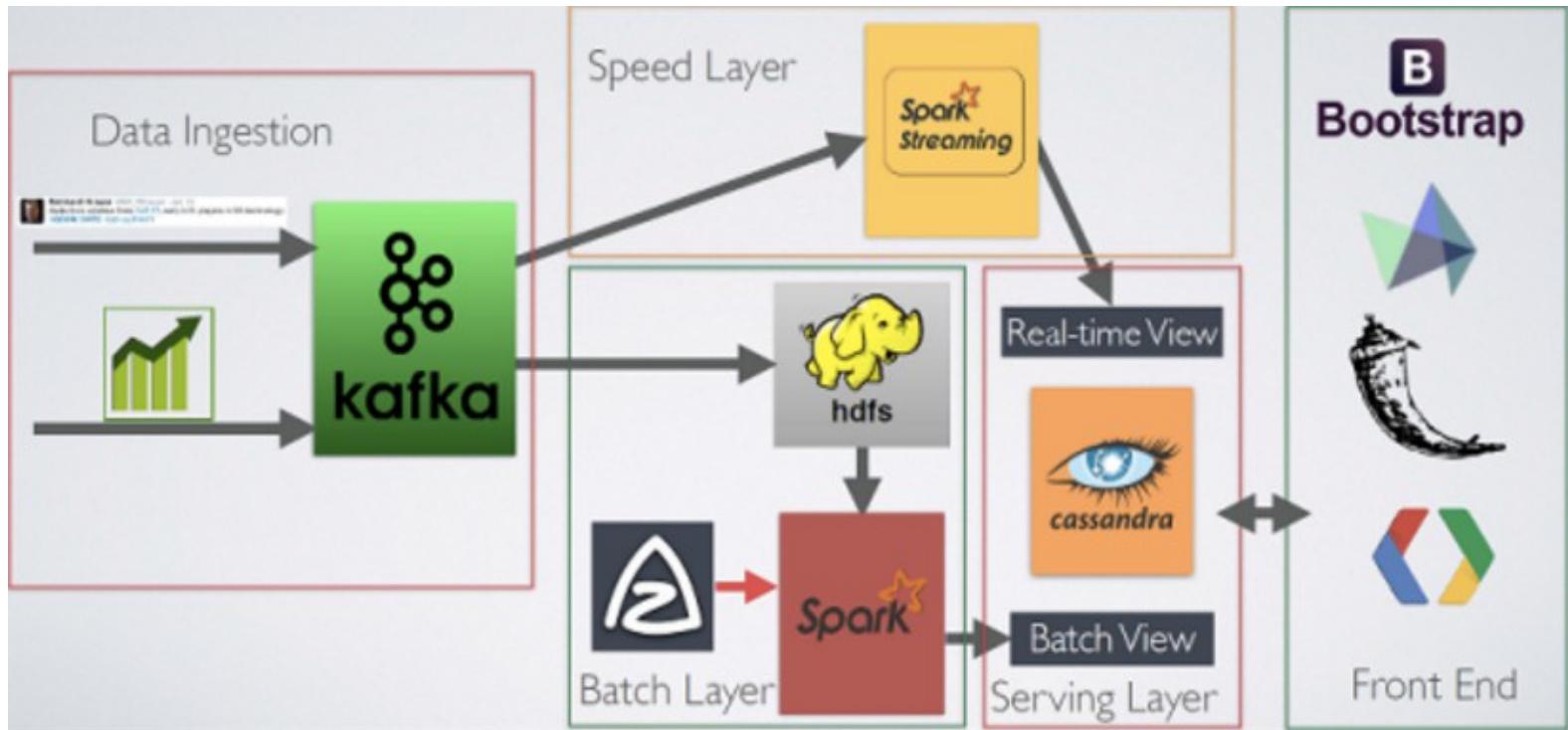
Spark Streaming Application: Process data



Use Case: Time Series Data



Use Case



DStream Functional operations

- flatMap(flatMapFunc)
- filter(filterFunc)
- map(mapFunc)
- mapPartitions(mapPartFunc, preservePartitioning)
- foreachRDD(foreachFunc)

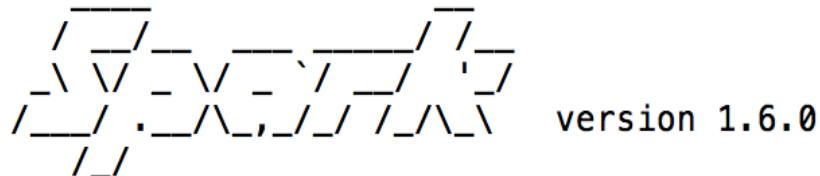
DStream Output operations

- `print()`
- `saveAsHadoopFiles(...)`
- `saveAsTextFiles(...)`
- `saveAsObjectFiles(...)`
- `saveAsNewAPIHadoopFiles(...)`
- `foreachRDD(..)`

Hands-on: Spark Streaming

Start Spark-shell with extra memory

```
[root@quickstart ~]# spark-shell --driver-memory 1G
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/jars/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
```



WordCount using Spark Streaming

```
$ scala> :paste
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.storage.StorageLevel
import StorageLevel._
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
val ssc = new StreamingContext(sc, Seconds(2))
val lines = ssc.socketTextStream("localhost", 8585, MEMORY_ONLY)
val wordsFlatMap = lines.flatMap(_.split(" "))
val wordsMap = wordsFlatMap.map( w => (w,1))
val wordCount = wordsMap.reduceByKey( (a,b) => (a+b))
wordCount.print
ssc.start
```

Running the netcat server on another window

```
ubuntu@ip-172-31-30-238:~$ sudo su
root@ip-172-31-30-238:/home/ubuntu# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              NAMES
581c45e85683        cloudera/quickstart:latest   "/usr/bin/docker-qui   About an hour ago
r ago   Up About an hour   0.0.0.0:8888->8888/tcp   backstabbing_lumiere
root@ip-172-31-30-238:/home/ubuntu# docker exec -i -t 581c45e85683 nc -lk 8585
```

```
test this
It is another test on Spark streaming. It is great
```

Time: 1465924608000 ms

(streaming.,1)
(Spark,1)
(great,1)
(is,2)
(test,1)
(another,1)
(on,1)
(It,2)

Hands-On: Streaming Twitter data

Create a new Twitter App

Login to your Twitter @ twitter.com

The screenshot shows the Twitter mobile application interface. At the top, there is a navigation bar with icons for Home, Notifications, Messages, Discover, and a search bar labeled "Search Twitter". Below the navigation bar is the user's profile section for "imcinstiute" (@imcinstiute), which includes a profile picture, the username, the handle, and statistics for tweets (88), following (9), and followers (23). To the right of the profile section is the main "What's happening?" timeline feed. The first tweet is from "สำนักข่าวเนชั่น" (@nnanews) posted 1 hour ago, mentioning a news story about a large-scale leak of personal data from a Thai government agency. The second tweet is from "HP OpenNFV" (@hpnfv), which is a promoted post about carrier networks and telecommunications. The third tweet is from "Pongsuk Hiranprueck" (@nuishow) posted 2 hours ago, discussing the comparison between WhatsApp and Facebook on Android devices. At the bottom of the screen, there is a "Get more from Twitter" section with three items: "Sign up", "Follow 5 accounts", and "Complete your profile", each accompanied by a green checkmark.

Create a new Twitter App (cont.)

Create a new Twitter App @ apps.twitter.com

The screenshot shows the Twitter Application Management interface. At the top, there's a navigation bar with a Twitter icon and the text "Application Management". On the right side of the bar is a user profile picture and a dropdown arrow. Below the bar, a large blue header bar spans the width of the page. The main content area has a title "Twitter Apps" in large, bold, dark gray font. Underneath the title is a light gray rectangular box containing the text "You don't currently have any Twitter Apps." A red arrow points from the bottom right towards this text. Below the text is a white button with a black border and the text "Create New App" in black.

Create a new Twitter App (cont.)

Enter all the details in the application:

 Application Management



Create an application

Application Details

Name *

IMC_Institute_App 

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

IMC Institute Demo App

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

<http://www.imcinstiute.com>

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.

(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Create a new Twitter App (cont.)

Your application will be created:

The screenshot shows a web browser window with the URL <https://apps.twitter.com/app/8158163> in the address bar. The page title is "Application Management". A green message box at the top states: "Your application has been created. Please take a moment to review and adjust your application's settings." Below this, the application details are listed: "IMC_Institute_App", "Details", "Settings", "Keys and Access Tokens", "Permissions", "Test OAuth". The "Details" tab is selected, showing the app name "IMC Institute Demo App" and the website "http://www.imcinstiute.com".

Your application has been created. Please take a moment to review and adjust your application's settings.

IMC_Institute_App

Test OAuth

Details

Settings

Keys and Access Tokens

Permissions



IMC Institute Demo App

<http://www.imcinstiute.com>

Organization

Information about the organization or company associated with your application. This information is optional.

Organization None

Organization website None

Application Settings

Create a new Twitter App (cont.)

Click on Keys and Access Tokens:

The screenshot shows the Twitter Application Management interface. At the top, there's a navigation bar with a Twitter icon and the text "Application Management". On the right side of the bar is a blue square icon with a white "T" and a dropdown arrow. Below the bar, the application name "IMC_Institute_App" is displayed in large, bold, black font. To the right of the application name is a "Test OAuth" button. Below the application name, there are four tabs: "Details", "Settings", "Keys and Access Tokens" (which is highlighted in blue), and "Permissions". The main content area is titled "Application Settings" and contains the following information:

<i>Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.</i>	
Consumer Key (API Key)	MjpswndxVj27ylnpOoSBrnfLX
Consumer Secret (API Secret)	QYmuBO1smD5Yc3zE0ZF9ByCgeEQxnxUmhRVCisAvPFudYVjC4a
Access Level	Read and write (modify app permissions)
Owner	imcinstitute
Owner ID	921172807

Two specific fields, "Consumer Key (API Key)" and "Consumer Secret (API Secret)", are circled in red at the bottom of the screenshot.

Create a new Twitter App (cont.)

Click on Keys and Access Tokens:

Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

[Create my access token](#)



Create a new Twitter App (cont.)

Your Access token got created:

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token **921172807-EfMXJj6as2dFECDH1vDe5goyTHcxPrF1RIJozqgx**

Access Token Secret **HppZEVip3D5j80GP21a37HxA4y10dH9BHcgEFXUNC9xy**

Access Level Read and write

Owner imcinstiute

Owner ID 921172807

Token Actions

[Regenerate My Access Token and Token Secret](#)

[Revoke Token Access](#)

Download the third-party libraries

```
$ wget http://central.maven.org/maven2/org/apache/spark/spark-
streaming-twitter_2.10/1.2.0/spark-streaming-twitter_2.10-
1.2.0.jar

$ wget
http://central.maven.org/maven2/org/twitter4j/twitter4j-
stream/4.0.2/twitter4j-stream-4.0.2.jar

$ wget
http://central.maven.org/maven2/org/twitter4j/twitter4j-
core/4.0.2/twitter4j-core-4.0.2.jar
```

Run Spark-shell

```
$ spark-shell --jars spark-streaming-twitter_2.10-1.2.0.jar,
twitter4j-stream-4.0.2.jar,twitter4j-core-4.0.2.jar
```

Running Spark commands

```
$ scala> :paste
// Entering paste mode (ctrl-D to finish)

import org.apache.spark.streaming.twitter._
import twitter4j.auth._
import twitter4j.conf._
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
val ssc = new StreamingContext(sc, Seconds(10))
val cb = new ConfigurationBuilder
```

Running Spark commands

```
cb.setDebugEnabled(true).setOAuthConsumerKey("MjpswndxVj27ylnp
OoSBrnfLX").setOAuthConsumerSecret("QYmuBO1smD5Yc3zE0ZF9ByCgeE
QxnxUmhRVCisAvPFudYVjC4a").setOAuthAccessToken("921172807-
EfMXJj6as2dFECDH1vDe5goyTHcxPrF1RIJozqgx").setOAuthAccessToken
Secret("HbpZEVip3D5j80GP21a37HxA4y10dH9BHcgEFXUNcA9xy")

val auth = new OAuthAuthorization(cb.build)

val tweets = TwitterUtils.createStream(ssc, Some(auth))

val status = tweets.map(status => status.getText)

status.print

ssc.checkpoint("hdfs://user/cloudera/data/tweets")

ssc.start

ssc.awaitTermination
```

HUE [Query Editors](#) [Data Browsers](#) [Workflows](#) [Search](#) [Security](#)

[File Browser](#)

Actions Move to trash

[Home](#) / [user](#) / [cloudera](#) / [data](#) / [tweets](#)

History Trash

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	↑		root	cloudera	drwxr-xr-x	June 14, 2016 09:53 AM
<input type="checkbox"/>	.		root	cloudera	drwxr-xr-x	June 14, 2016 09:54 AM
<input type="checkbox"/>	a2fb06d7-1b37-4e55-a38d-ec3ea3bc当地		root	cloudera	drwxr-xr-x	June 14, 2016 09:53 AM
<input type="checkbox"/>	checkpoint-1465923240000	4.9 KB	root	cloudera	-rw-r--r--	June 14, 2016 09:54 AM
<input type="checkbox"/>	checkpoint-1465923240000.bk	4.9 KB	root	cloudera	-rw-r--r--	June 14, 2016 09:54 AM
<input type="checkbox"/>	checkpoint-1465923250000	4.9 KB	root	cloudera	-rw-r--r--	June 14, 2016 09:54 AM
<input type="checkbox"/>	checkpoint-1465923250000.bk	4.9 KB	root	cloudera	-rw-r--r--	June 14, 2016 09:54 AM
<input type="checkbox"/>	checkpoint-1465923260000	4.9 KB	root	cloudera	-rw-r--r--	June 14, 2016 09:54 AM
<input type="checkbox"/>	checkpoint-1465923260000.bk	4.9 KB	root	cloudera	-rw-r--r--	June 14, 2016 09:54 AM
<input type="checkbox"/>	checkpoint-1465923270000	4.9 KB	root	cloudera	-rw-r--r--	June 14, 2016 09:54 AM



Lecture

Understanding Kafka

Introduction

Open-source message broker project



An open-source message broker project developed by the Apache Software Foundation written in Scala. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. It is, in its essence, a "massively scalable pub/sub message queue architected as a distributed transaction log", making it highly valuable for enterprise infrastructures.

What is Kafka?

- An apache project initially developed at LinkedIn
- Distributed publish-subscribe messaging system
- Designed for processing of real time activity stream data
e.g. logs, metrics collections
- Written in Scala
- Does not follow JMS Standards, neither uses JMS APIs

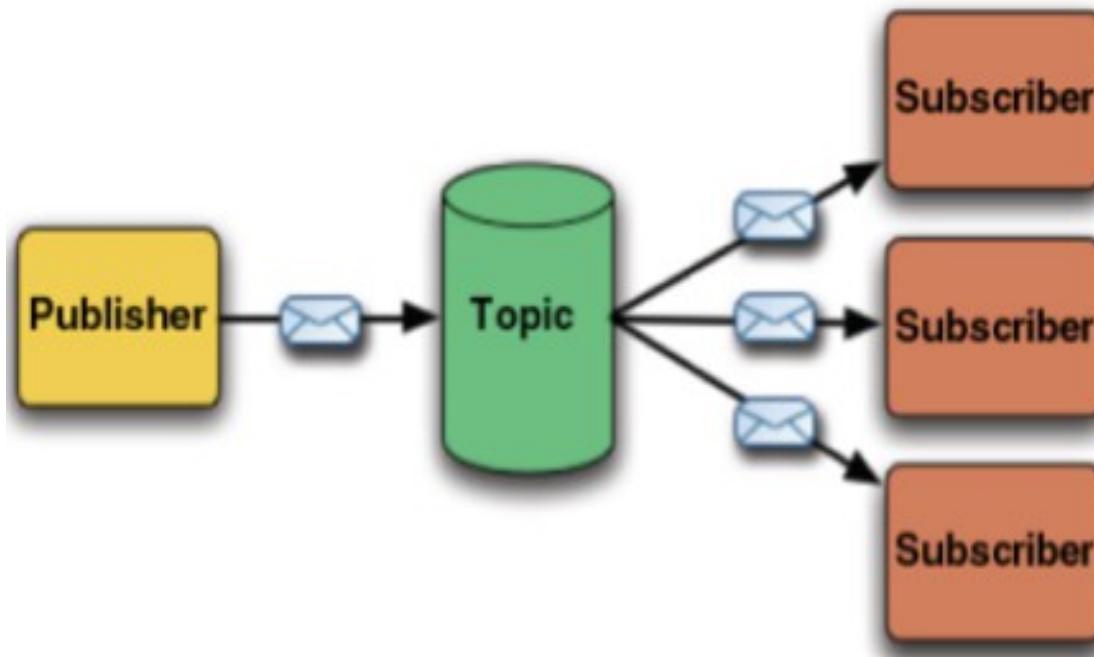
Kafka: Features

- Persistent messaging
- High-throughput
- Supports both queue and topic semantics
- Uses Zookeeper for forming a cluster of nodes (producer/consumer/broker)
- and many more...

Why Kafka?

- Built with speed and scalability in mind.
- Enabled near real-time access to any data source
- Empowered hadoop jobs
- Allowed us to build real-time analytics
- Vastly improved our site monitoring and alerting capability
- Enabled us to visualize and track our call graphs.

Messaging System Concept: Topic



Source: Real time Analytics with Apache Kafka and Spark, Rahul Jain

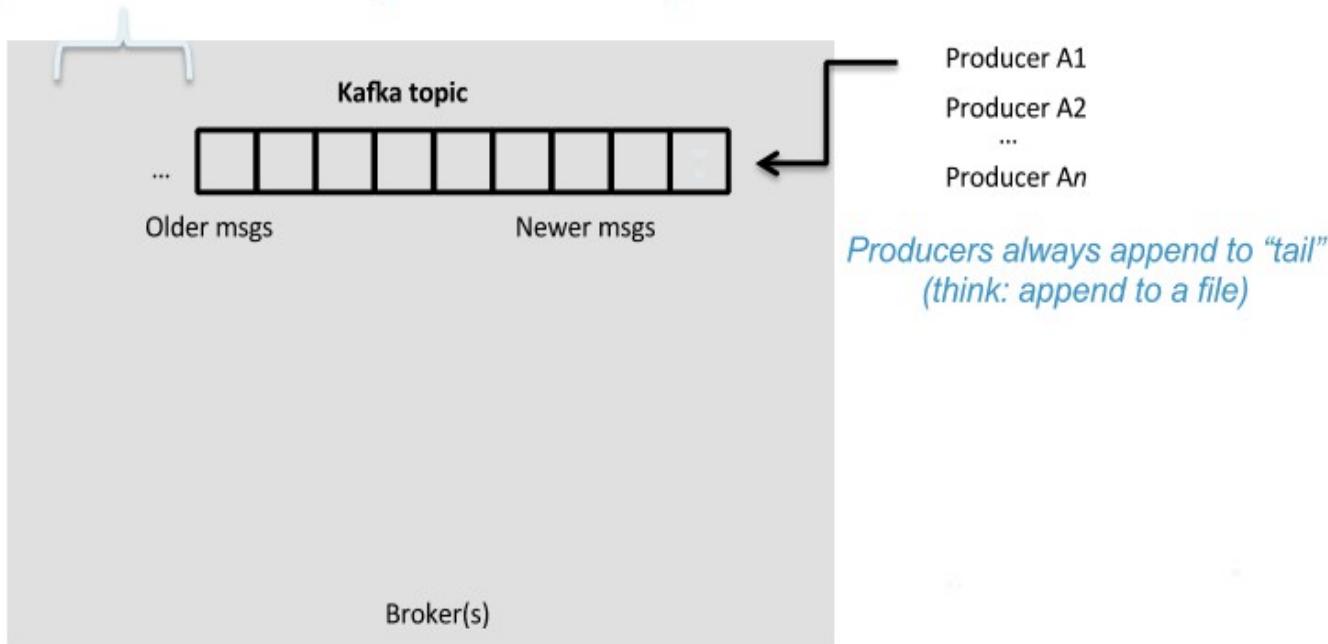
Terminology

- Kafka maintains feeds of messages in categories called topics.
- Processes that publish messages to a Kafka topic are called producers.
- Processes that subscribe to topics and process the feed of published messages are called consumers.
- Kafka is run as a cluster comprised of one or more servers each of which is called a broker.

Topics

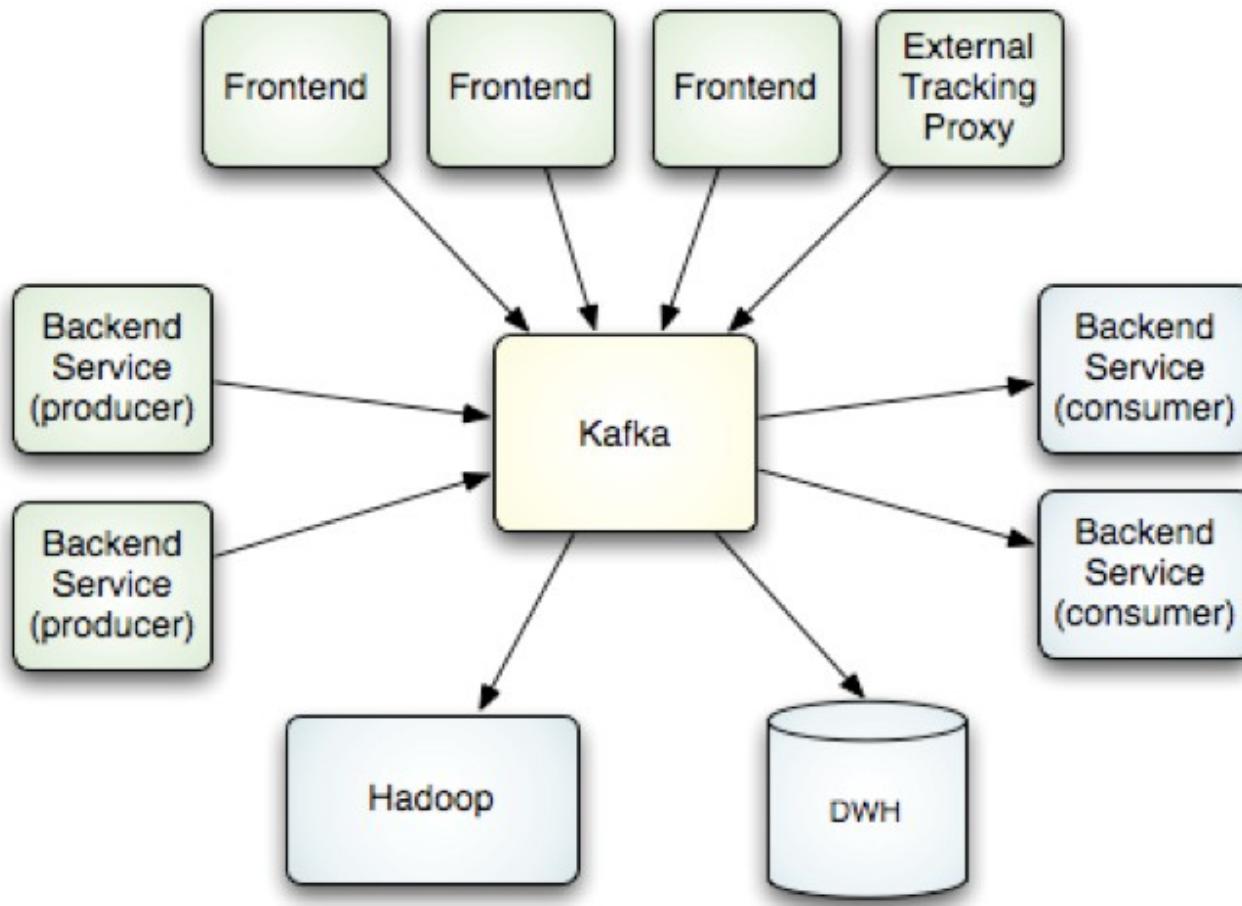
- Topic: feed name to which messages are published

Kafka prunes “head” based on age or max size or “key”



Source: Apache Kafka with Spark Streaming - Real Time Analytics Redefined

Kafka

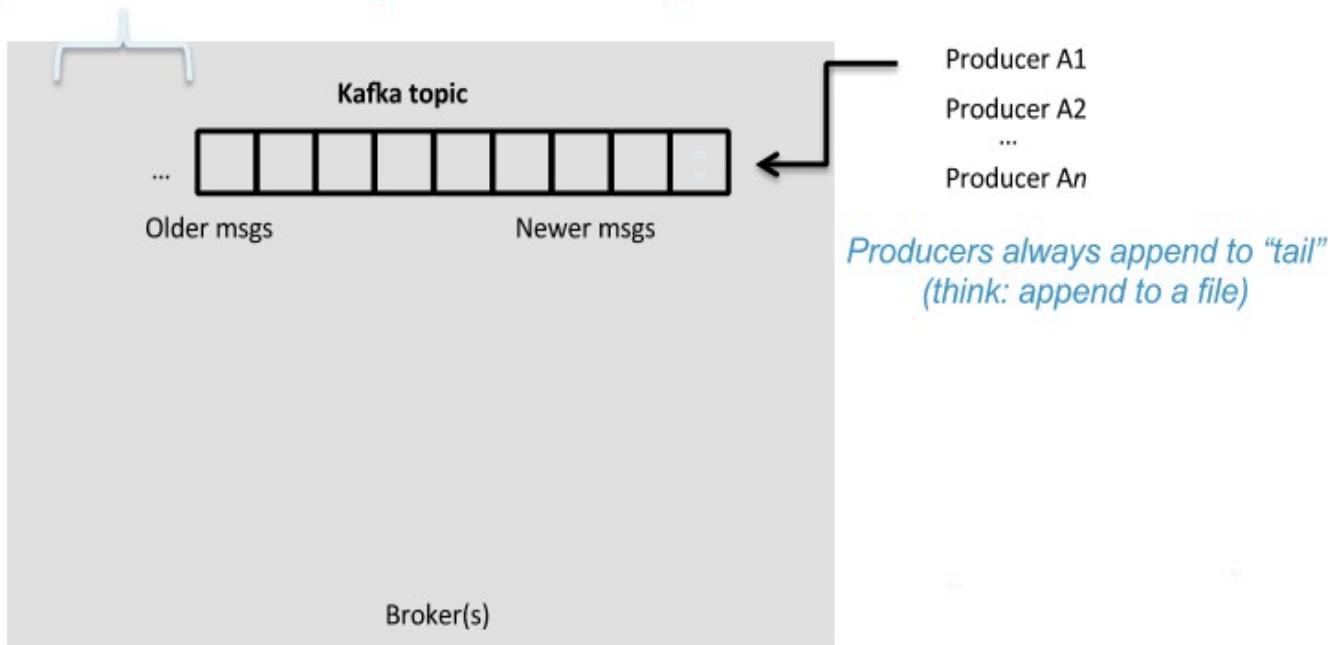


Source: Real time Analytics with Apache Kafka and Spark, Rahul Jain

Topics

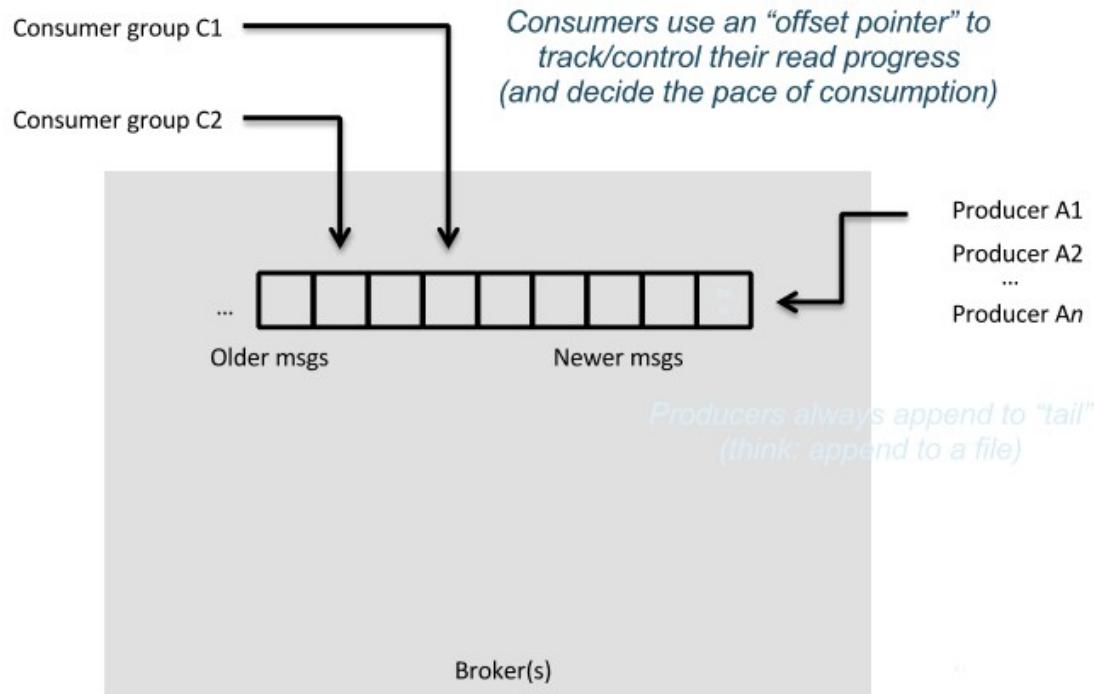
- Topic: feed name to which messages are published

Kafka prunes “head” based on age or max size or “key”



Source: Apache Kafka with Spark Streaming - Real Time Analytics Redefined

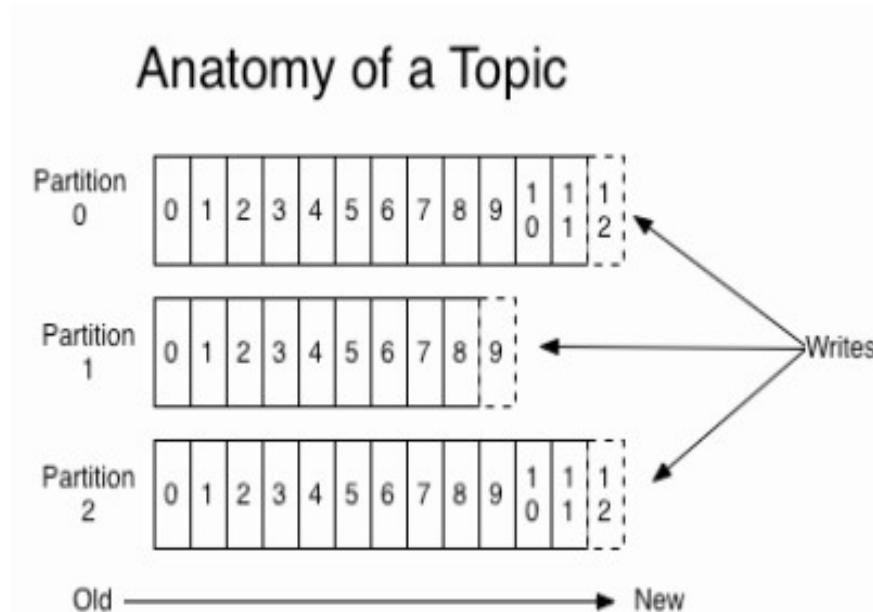
Topics



Source: Apache Kafka with Spark Streaming - Real Time Analytics Redefined

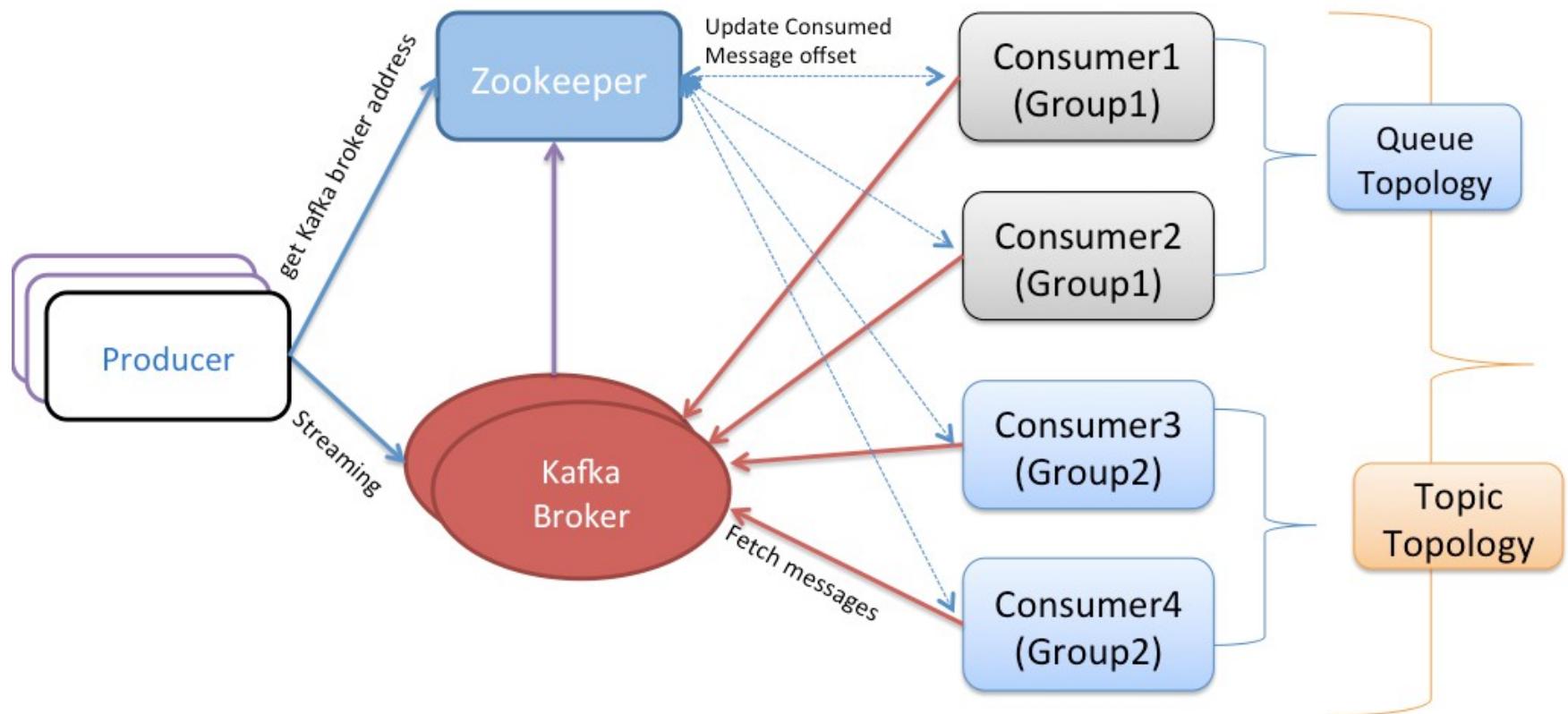
Topics

- A topic consists of partitions.
- Partition: ordered + immutable sequence of messages that is continually appended



Source: Apache Kafka with Spark Streaming - Real Time Analytics Redefined

Kafka Architecture



Source: Real time Analytics with Apache Kafka and Spark, Rahul Jain

Hands-on

SparkStreaming from Kafka

Install & Start Kafka Server

```
# wget http://www-us.apache.org/dist/kafka/0.9.0.1/kafka_2.10-  
0.9.0.1.tgz  
# tar xzf kafka_2.10-0.9.0.1.tgz  
# cd kafka_2.10-0.9.0.1  
# bin/kafka-server-start.sh config/server.properties&
```

```
[2016-06-23 04:37:21,426] INFO Kafka commitId : 23c69d62a0cabf06 (o  
rg.apache.kafka.common.utils.AppInfoParser)  
[2016-06-23 04:37:21,430] INFO [Kafka Server 0], started (kafka.ser  
ver.KafkaServer)  
[2016-06-23 04:37:21,446] INFO New leader is 0 (kafka.server.Zookeee  
perLeaderElector$LeaderChangeListener)
```

Running Kafka Producer

```
# bin/kafka-console-producer.sh --topic test --broker-list  
localhost:9092
```

type some random messages followed by Ctrl-D to finish

```
[root@quickstart kafka_2.10-0.9.0.1]# bin/kafka-console-producer.sh  
--topic test --broker-list localhost:9092  
This is a test message from IMC Institute
```

Big Data School

Test

```
[root@quickstart kafka_2.10-0.9.0.1]# █
```

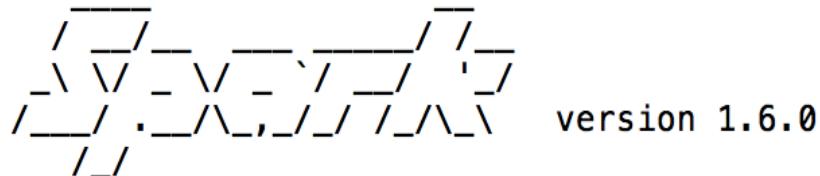
Running Kafka Consumer

```
# bin/kafka-console-consumer.sh --topic test --zookeeper localhost:2181 --from-beginning
```

```
[root@quickstart kafka_2.10-0.9.0.1]# bin/kafka-console-consumer.sh --topic test --zookeeper localhost:2181 --from-beginning
This is a test message from IMC Institute
Big Data School
Test
```

Start Spark-shell with extra memory

```
[root@quickstart ~]# spark-shell --driver-memory 1G
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/jars/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
```



Spark Streaming with Kafka

```
$ scala> :paste
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.storage.StorageLevel
import StorageLevel._
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.kafka.KafkaUtils
val ssc = new StreamingContext(sc, Seconds(2))
val kafkaStream = KafkaUtils.createStream(ssc,
"localhost:2181","spark-streaming-consumer-group", Map("spark-
topic" -> 5))
kafkaStream.print()
ssc.start
```

Running Kafka Producer on another terminal

```
# docker ps
```

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	PORTS
NAMES		
c77e4dc1ed9b	cloudera/quickstart:latest	"/usr/bin/docker-q
ui 22 minutes ago	Up 22 minutes	0.0.0.0:8888->8888/tcp
trusting_newton		

```
# docker exec -i -t c77e4dc1ed9b /bin/bash
```

```
[root@quickstart ~]# cd /root/kafka_2.10-0.9.0.1
[root@quickstart kafka_2.10-0.9.0.1]# bin/kafka-console-
producer.sh --broker-list localhost:9092 --topic spark-topic
```

Test & View the result

```
[root@quickstart kafka_2.10-0.9.0.1]# bin/kafka-console-producer.sh  
--broker-list localhost:9092 --topic spark-topic  
Hello from IMC Institute
```

Result from another terminal

```
85200 replicated to only 0 peer(s) instead of 1 peers  
-----  
Time: 1466658086000 ms  
-----  
(null,Hello from IMC Institute)
```

Hands-on

Realtime analytics with Kafka

Java Code: Kafka Producer

```
import java.util.Properties;  
  
import kafka.producer.KeyedMessage;  
import kafka.producer.ProducerConfig;  
import java.io.*;  
  
public class HelloKafkaProducer {  
    final static String TOPIC = "java-topic";  
    public static void main(String[] argv) {  
        Properties properties = new Properties();  
  
        properties.put("metadata.broker.list","localhost:9092");  
  
        properties.put("serializer.class","kafka.serializer.StringEncoder");  
    }  
}
```

Java Code: Kafka Producer (cont.)

```
try(BufferedReader br = new BufferedReader(new
FileReader(argv[0]))) {
    StringBuilder sb = new StringBuilder();
    ProducerConfig producerConfig = new
ProducerConfig(properties);
    kafka.javaapi.producer.Producer<String, String>
producer = new kafka.javaapi.producer.Producer<String,
String>(producerConfig);
    String line = br.readLine();

    while (line != null) {
        KeyedMessage<String, String> message
=new KeyedMessage<String, String>(TOPIC, line);
        producer.send(message);
        line = br.readLine();
    }
}
```

Java Code: Kafka Producer (cont.)

```
        producer.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    }

}
```

Compile & Run the program

```
// Using a vi Editor to edit the sourcecode
# vi HelloKafkaProducer.java
// Alternatively
# wget
https://s3.amazonaws.com/imcbucket/apps/HelloKafkaProducer.java

// Compile progeram
# export CLASSPATH=".:/root/kafka_2.10-0.9.0.1/libs/*"
# javac HelloKafkaProducer.java

//prepare the data
# cd
# wget https://s3.amazonaws.com/imcbucket/input/pg2600.txt
# cd kafka_2.10-0.9.0.1
// Run the program
# java HelloKafkaProducer /root/pg2600.txt
```

Spark Streaming with Kafka : Wordcount

```
$ scala> :paste
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.storage.StorageLevel
import StorageLevel._
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.kafka.KafkaUtils
val ssc = new StreamingContext(sc, Seconds(2))
val kafkaStream = KafkaUtils.createStream(ssc,
"localhost:2181","spark-streaming-consumer-group", Map("java-
topic" -> 5))
```

Spark Streaming with Kafka : WordCount

```
val lines = kafkaStream.map(_.value)  
val words = lines.flatMap(_.split(" "))  
val wordCounts = words.map(x => (x, 1L)).reduceByKey(_ + _)  
wordCounts.print()  
  
ssc.start()  
ssc.awaitTermination()
```

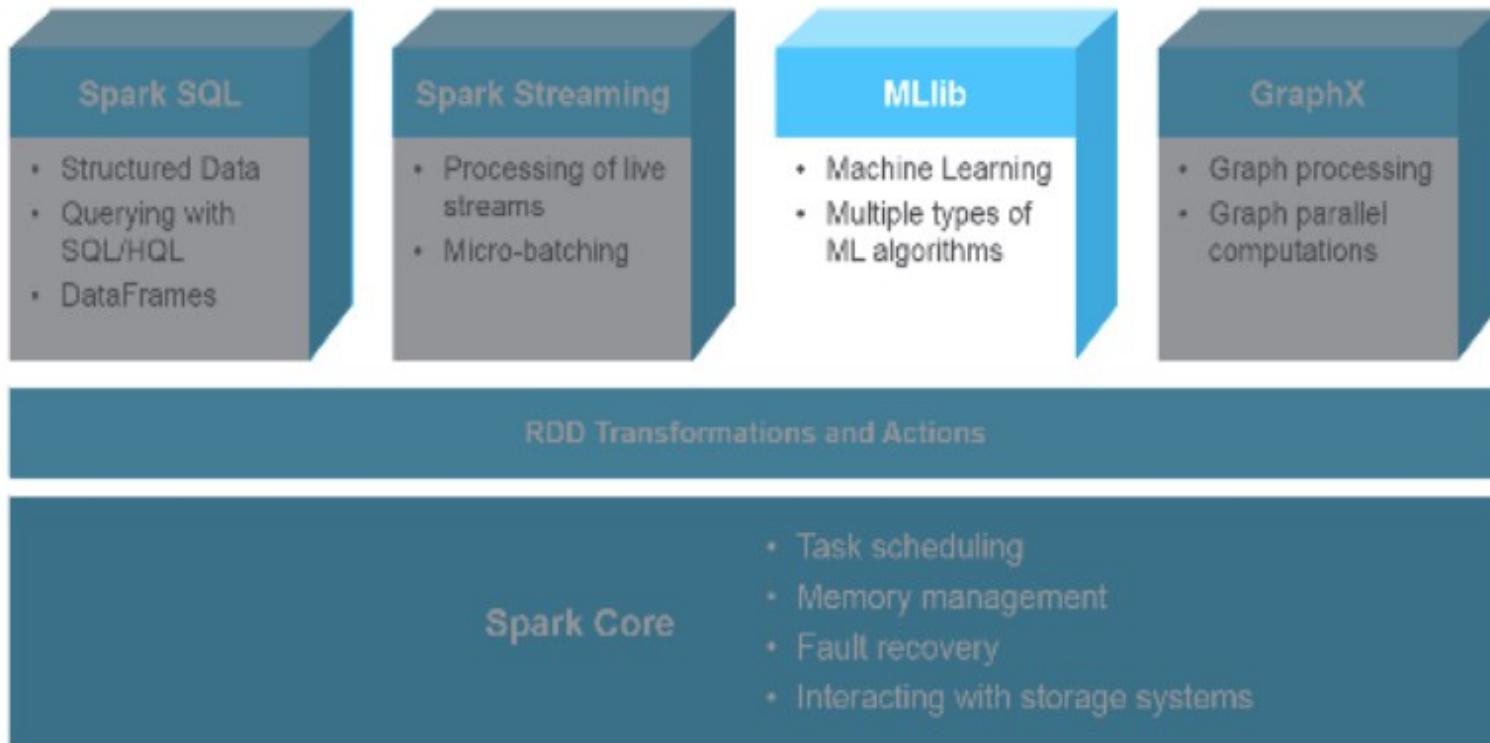
Output

```
-----  
Time: 1467251428000 ms  
-----
```

```
(Ermolov.,4)  
(mattered,4)  
(Ah!,10)  
(intimately,6)  
(Koko,2)  
(denied?",2)  
(muslin,,2)  
(reunion,4)  
(blandly,10)  
("Ho!,2)  
...  
-----
```

Spark MLlib

What is MLlib?



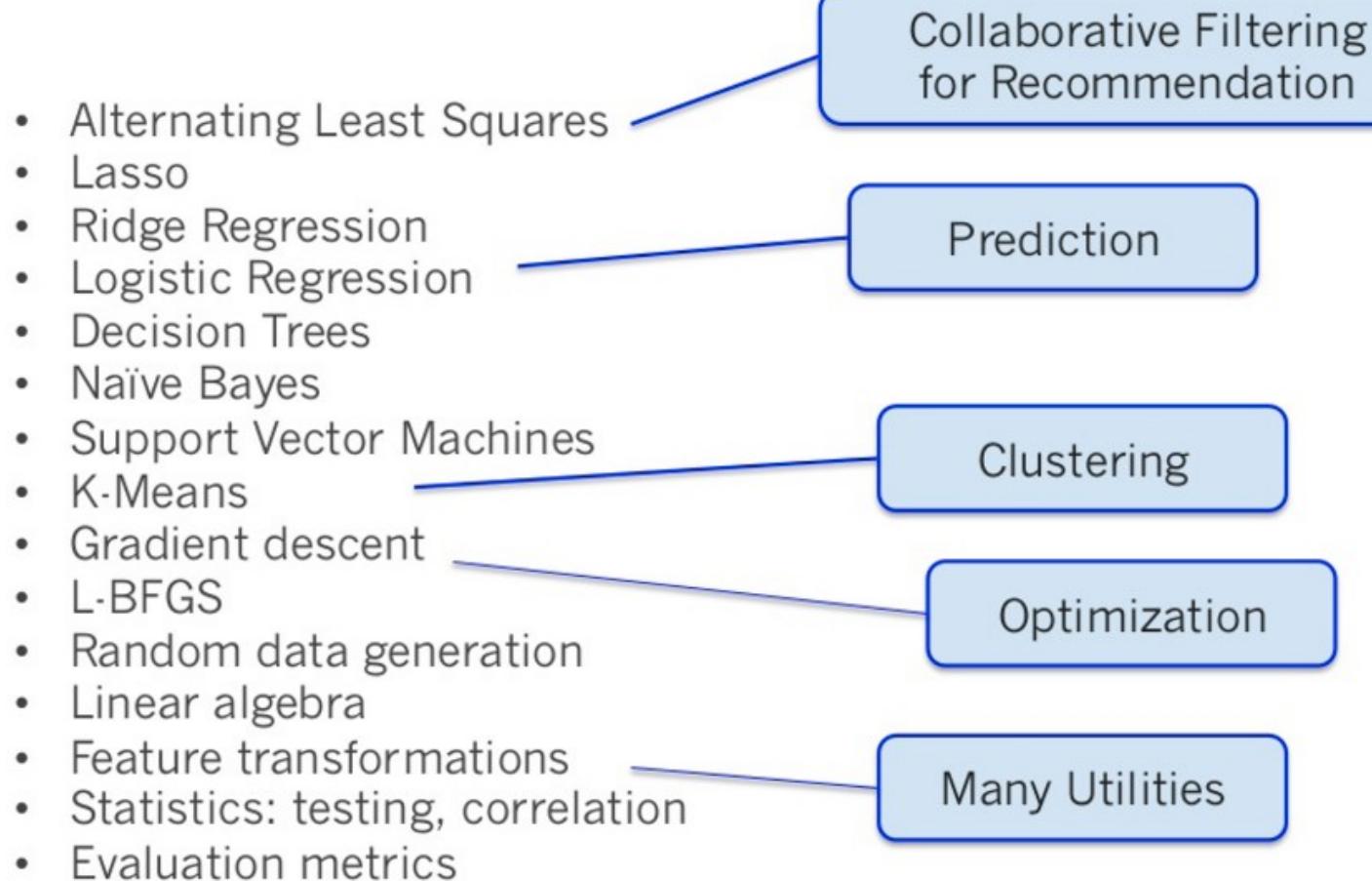
What is MLlib?

- MLlib is a Spark subproject providing machine learning primitives:
 - initial contribution from AMPLab, UC Berkeley
 - shipped with Spark since version 0.8
 - 33 contributors

Mllib Algorithms

- **Classification:** logistic regression, linear support vector machine(SVM), naive Bayes
- **Regression:** generalized linear regression (GLM)
- **Collaborative filtering:** alternating least squares (ALS)
- **Clustering:** k-means
- **Decomposition:** singular value decomposition (SVD), principal component analysis (PCA)

What is in MLlib?



MLlib: Benefits

- Part of Spark
- Scalable
- Support: Python, Scala, Java
- Broad coverage of applications & algorithms
- Rapid developments in speed & robustness

Machine Learning

Machine learning is a scientific discipline that explores the construction and study of algorithms that can learn from data.

[Wikipedia]

Vectors

- A point is just a set of numbers. This set of numbers or coordinates defines the point's position in space.
- Points and vectors are same thing.
- Dimensions in vectors are called features
- Hyperspace is a space with more than three dimensions.
- Example: A person has the following dimensions:
 - Weight
 - Height
 - Age
- Thus, the interpretation of point (160,69,24) would be 160 lb weight, 69 inches height, and 24 years age.

Vectors in MLlib

- Spark has local vectors and matrices and also distributed matrices.
 - Distributed matrix is backed by one or more RDDs.
 - A local vector has numeric indices and double values, and is stored on a single machine.
- Two types of local vectors in MLlib:
 - **Dense vector** is backed by an array of its values.
 - **Sparse vector** is backed by two parallel arrays, one for indices and another for values.
- Example
 - Dense vector: [160.0,69.0,24.0]
 - Sparse vector: (3,[0,1,2],[160.0,69.0,24.0])

Vectors in Mllib (cont.)

- Library
 - import org.apache.spark.mllib.linalg.{Vectors,Vector}
- Signature of **Vectors.dense**:
 - def dense(values: Array[Double]): Vector
- Signature of **Vectors.sparse**:
 - def sparse(size: Int, indices: Array[Int], values: Array[Double]): Vector

Example

```
scala> import org.apache.spark.mllib.linalg.{Vectors,Vector}
import org.apache.spark.mllib.linalg.{Vectors, Vector}

scala> val dvPerson = Vectors.dense(160.0,69.0,24.0)
dvPerson: org.apache.spark.mllib.linalg.Vector = [160.0,69.0,24.0]

scala> val svPerson = Vectors.sparse(3,Array(0,1,2),Array(160.0,69.0,24.0))
svPerson: org.apache.spark.mllib.linalg.Vector = (3,[0,1,2],[160.0,69.0,24.0])
```

Labeled point

- Labeled point is a local vector (sparse/dense),), which has an associated label with it.
- Labeled data is used in supervised learning to help train algorithms.
- Label is stored as a double value in **LabeledPoint**.

Type	Label values
Binary classification	0 or 1
Multiclass classification	0, 1, 2...
Regression	Decimal values

Example

```
scala> import org.apache.spark.mllib.linalg.{Vectors,Vector}
scala> import org.apache.spark.mllib.regression.LabeledPoint
scala> val willBuySUV =
LabeledPoint(1.0,Vectors.dense(300.0,80,40))

scala> val willNotBuySUV =
LabeledPoint(0.0,Vectors.dense(150.0,60,25))

scala> val willBuySUV =
LabeledPoint(1.0,Vectors.sparse(3,Array(0,1,2),Array(300.0,80,
40)))

scala> val willNotBuySUV =
LabeledPoint(0.0,Vectors.sparse(3,Array(0,1,2),Array(150.0,60,
25)))
```

Example (cont)

```
# vi person_libsvm.txt
```

```
0 1:150 2:60 3:25
1 1:300 2:80 3:40
```

```
scala> import org.apache.spark.mllib.util.MLUtils
scala> import org.apache.spark.rdd.RDD
scala> val persons =
MLUtils.loadLibSVMFile(sc,"hdfs://user/cloudera/person_libsvm
.txt")
scala> persons.first()

res0: org.apache.spark.mllib.regression.LabeledPoint = (0.0,(3,[0,1
,2],[150.0,60.0,25.0]))
```

Matrices in MLlib

- Spark has local matrices and also distributed matrices.
 - Distributed matrix is backed by one or more RDDs.
 - A local matrix stored on a single machine.
- There are three types of distributed matrices in MLlib:
 - **RowMatrix**: This has each row as a feature vector.
 - **IndexedRowMatrix**: This also has row indices.
 - **CoordinateMatrix**: This is simply a matrix of MatrixEntry. A MatrixEntry represents an entry in the matrix represented by its row and column index

Example

```
scala> import org.apache.spark.mllib.linalg.{Vectors,Matrix,  
Matrices}  
  
scala> val people = Matrices.dense(3,2,Array(150d,60d,25d,  
300d,80d,40d))
```

```
people: org.apache.spark.mllib.linalg.Matrix =  
150.0 300.0  
60.0 80.0  
25.0 40.0
```

```
scala> val personRDD =  
sc.parallelize(List(Vectors.dense(150,60,25),  
Vectors.dense(300,80,40)))  
  
scala> import org.apache.spark.mllib.linalg.distributed.  
{IndexedRow, IndexedRowMatrix, RowMatrix, CoordinateMatrix,  
MatrixEntry}  
  
scala> val personMat = new RowMatrix(personRDD)
```

Example

```
scala> print(personMat numRows)
scala> val personRDD = sc.parallelize(List(IndexedRow(0L,
Vectors.dense(150,60,25)), IndexedRow(1L,
Vectors.dense(300,80,40))))
scala> val pirmat = new IndexedRowMatrix(personRDD)
scala> val personMat = pirmat.toRowMatrix
scala> val meRDD = sc.parallelize(List(
  MatrixEntry(0,0,150), MatrixEntry(1,0,60),
  MatrixEntry(2,0,25), MatrixEntry(0,1,300),
  MatrixEntry(1,1,80),MatrixEntry(2,1,40) ))
scala> val pcmat = new CoordinateMatrix(meRDD)

scala> print(pcmat numRows)
3
scala> print(pcmat numCols)
2
```

Statistic functions

- Central tendency of data—mean, mode, median
- Spread of data—variance, standard deviation
- Boundary conditions—min, max

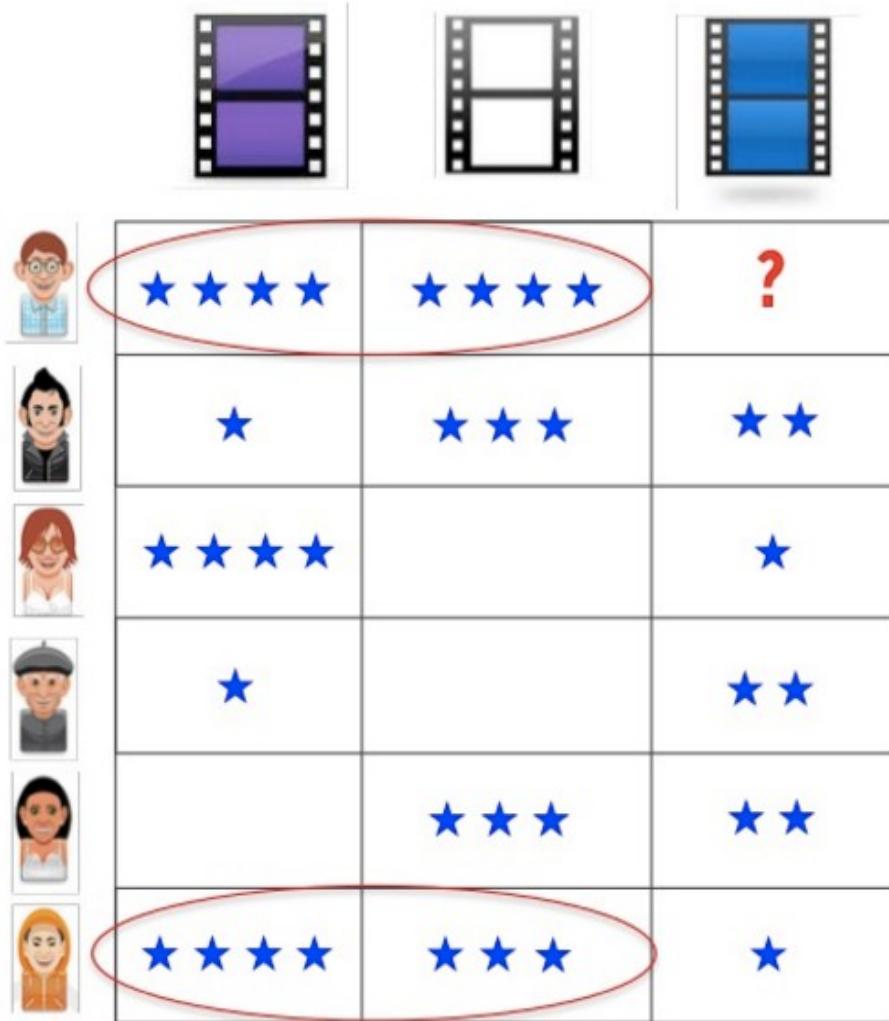
Example

```
scala> import org.apache.spark.mllib.linalg.{Vectors,Vector}
scala> import org.apache.spark.mllib.stat.Statistics
scala> val personRDD =
sc.parallelize(List(Vectors.dense(150,60,25),
Vectors.dense(300,80,40)))
scala> val summary = Statistics.colStats(personRDD)
```

```
scala> print(summary.mean)
[225.0,70.0,32.5]
scala> print(summary.variance)
[11250.0,200.0,112.5]
scala> print(summary.numNonzeros)
[2.0,2.0,2.0]
scala> print(summary.count)
2
scala> print(summary.max)
[300.0,80.0,40.0]
```

Hands-on Movie Recommendation

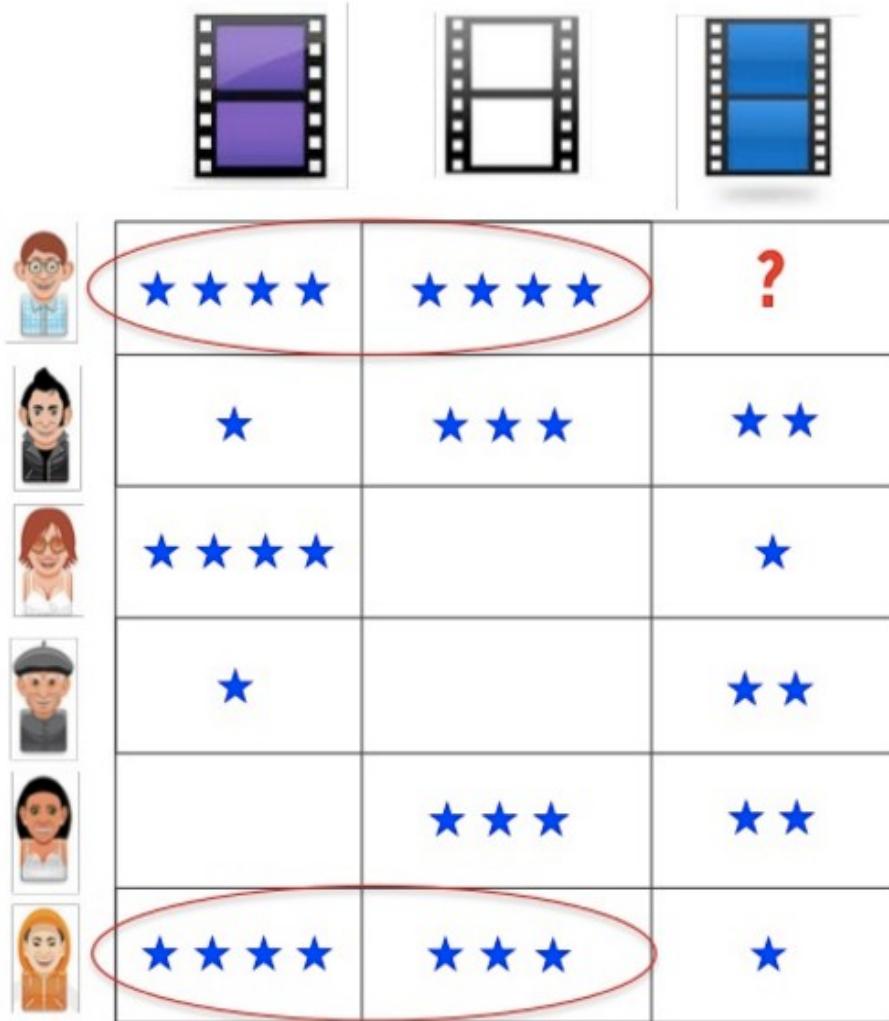
Recommendation



Goal: Recommend movies to users



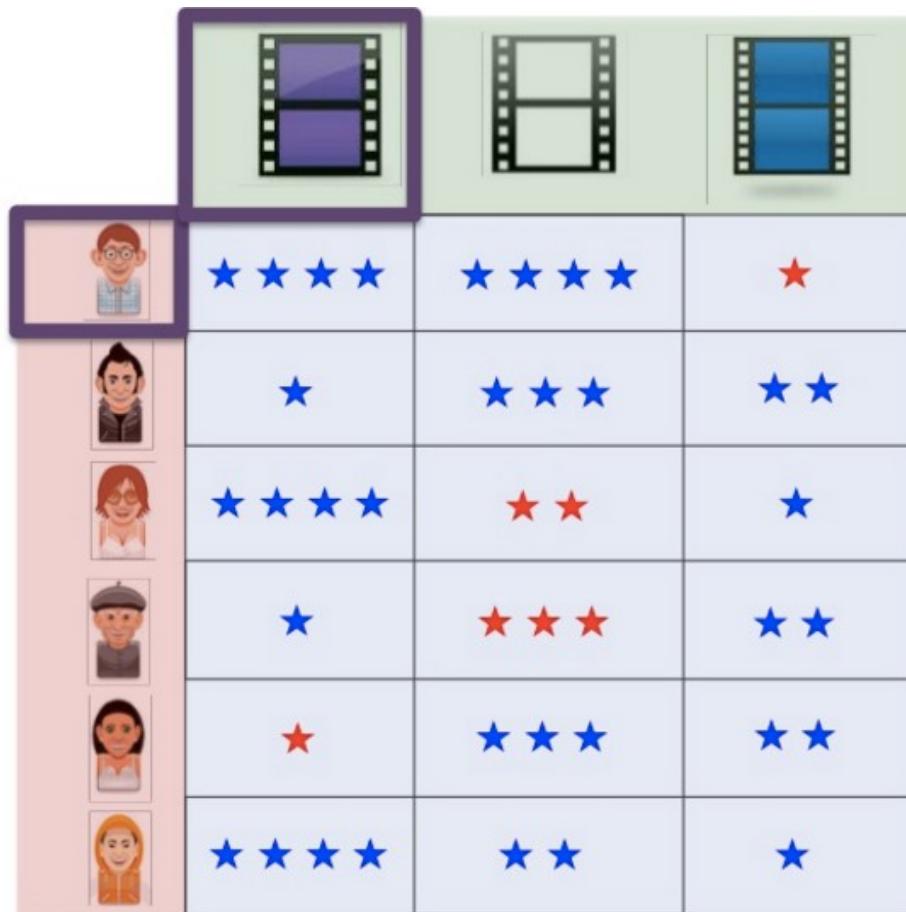
Recommendation: Collaborative Filtering



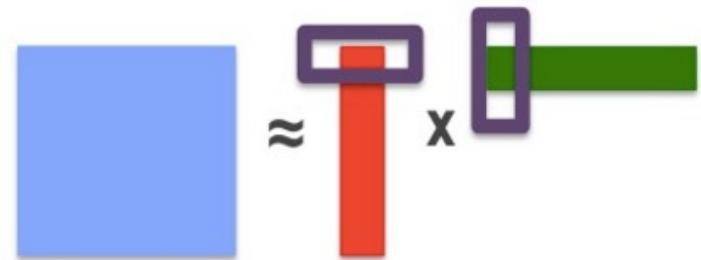
Goal: Recommend movies to users



Recommendation



Solution: Assume ratings are determined by a small number of factors.



25M Users, 100K Movies
→ 2.5 trillion ratings
With 10 factors/user
→ 250M parameters

Recommendation: ALS

Algorithm

Alternating update of user/movie factors

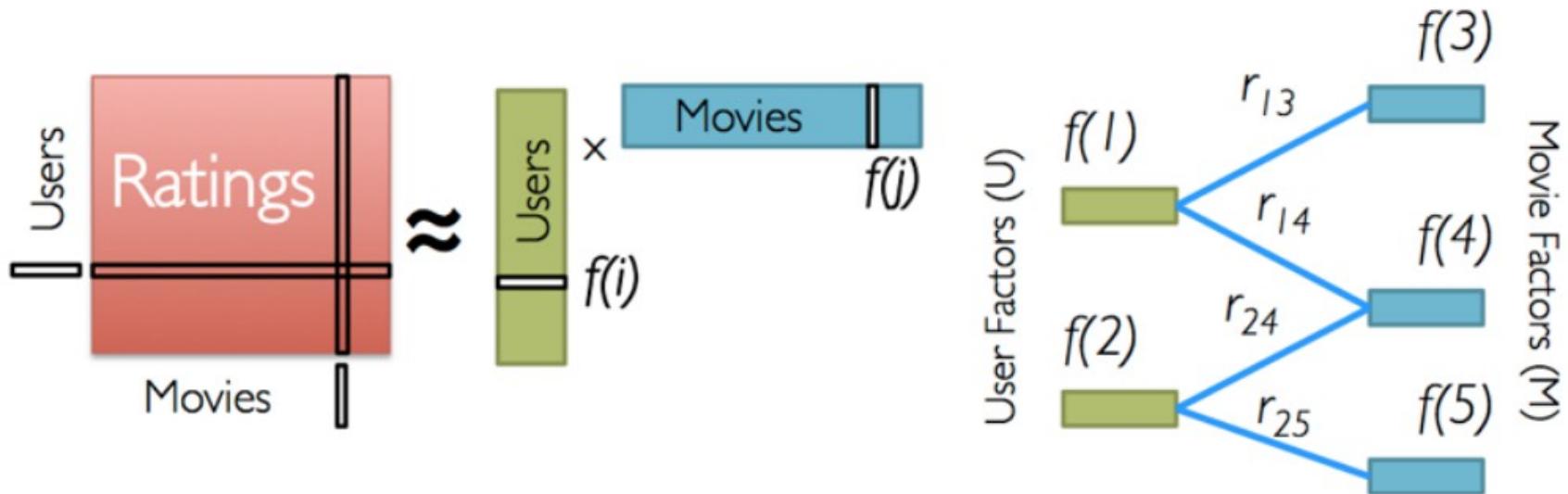
$$\text{Blue Box} = \text{Red Box} \times \text{Green Box}$$

Can update factors in parallel

Must be careful about communication



Alternating least squares (ALS)



Iterate:

$$f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda ||w||_2^2$$

MLlib: ALS Algorithm

- **numBlocks** is the number of blocks used to parallelize computation (set to -1 to autoconfigure)
- **rank** is the number of latent factors in the model
- **iterations** is the number of iterations to run
- **lambda** specifies the regularization parameter in ALS
- **implicitPrefs** specifies whether to use the explicit feedback ALS variant or one adapted for an implicit feedback data
- **alpha** is a parameter applicable to the implicit feedback variant of ALS that governs the baseline confidence in preference observations

MovieLen Dataset

1) Type command > `wget`

`http://files.grouplens.org/datasets/movielens/ml-100k.zip`

2) Type command > `yum install unzip`

3) Type command > `unzip ml-100k.zip`

4) Type command > `more ml-100k/u.user`

```
[root@quickstart guest1]# more ml-100k/u.user
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
11|39|F|other|30329
```

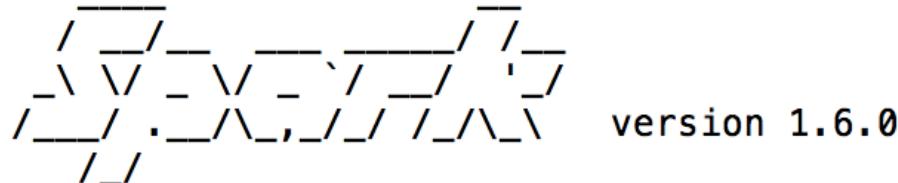
Moving dataset to HDFS

- 1) Type command > `cd ml-100k`
- 2) Type command > `hadoop fs -mkdir /user/cloudera/movielens`
- 3) Type command > `hadoop fs -put u.user /user/cloudera/movielens`
- 4) Type command > `hadoop fs -put u.data /user/cloudera/movielens`
- 4) Type command > `hadoop fs -put u.genre /user/cloudera/movielens`
- 5) Type command > `hadoop fs -put u.item /user/cloudera/movielens`
- 6) Type command > `hadoop fs -ls /user/cloudera/movielens`

```
[root@quickstart ml-100k]# hadoop fs -ls /user/cloudera/movielens
Found 3 items
-rw-r--r-- 1 root cloudera          202 2016-07-01 06:34 /user/clou
dera/movielens/u.genre
-rw-r--r-- 1 root cloudera      236344 2016-07-01 06:35 /user/clou
dera/movielens/u.item
-rw-r--r-- 1 root cloudera      22628 2016-07-01 06:34 /user/clou
dera/movielens/u.user
[root@quickstart ml-100k]# □
```

Start Spark-shell with extra memory

```
[root@quickstart ml-100k]# spark-shell --driver-memory 4g
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/jars/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
```



Extracting features from the MovieLens dataset

```
scala> val rawData =  
sc.textFile("hdfs://user/cloudera/movielens/u.data")  
scala> rawData.first()
```

```
res0: String = 196      242      3      881250949
```

```
scala> val rawRatings = rawData.map(_.split("\t")).take(3)  
scala> rawRatings.first()
```

```
res2: Array[String] = Array(196, 242, 3)
```

```
scala> import org.apache.spark.mllib.recommendation.Rating  
scala> val ratings = rawRatings.map { case Array(user, movie,  
rating) => Rating(user.toInt, movie.toInt, rating.toDouble) }  
scala> ratings.first()
```

Training the recommendation model

```
scala> import org.apache.spark.mllib.recommendation.ALS  
scala> val model = ALS.train(ratings, 50, 10, 0.01)
```

Note: We'll use rank of 50, 10 iterations, and a lambda parameter of 0.01

```
scala> model.userFeatures.count  
res5: Long = 943
```

```
scala> model.productFeatures.count  
res6: Long = 1682
```

```
scala> val predictedRating = model.predict(789, 123)  
predictedRating: Double = 3.2037183608258197
```

Inspecting the recommendations

```
scala> val movies =  
sc.textFile("hdfs://user/cloudera/movielens/u.item")  
  
scala> val titles = movies.map(line =>  
line.split("\\|").take(2)).map(array  
=>(array(0).toInt,array(1))).collectAsMap()
```

titles: scala.collection.Map[Int, String] = Map(137 -> Big Night (1996), 891 -> Bent (1997), 550 -> Die Hard: With a Vengeance (1995), 1205 -> Secret Agent, The (1996), 146 -> Unhook the Stars (1996), 864 -> My Fellow Americans (1996), 559 -> Interview with the Vampire (1994), 218 -> Cape Fear (1991), 568 -> Speed (1994), 227 -> Star Trek VI: The Undiscovered Country (1991), 765 -> Boomerang (1992), 1115 -> Twelfth Night (1996), 774 -> Prophecy, The (1995), 433 -> Heathers (1989), 92 -> True Romance (1993), 1528 -> Nowhere (1997), 846 -> To Gillian on Her 37th Birthday (1996), 1187 -> Switchblade Sisters (1975), 1501 -> Prisoner of the Mountains (Kavkazsky Plennik) (1996), 442 -> Amityville Curse, The (1990), 1160 -> Love! Valour! Compassion! (1997), 101 -> Heavy Metal (1981), 1196 -> Sa...)

Inspecting the recommendations (cont.)

```
scala> val moviesForUser = ratings.keyBy(_.user).lookup(789)
```

```
moviesForUser: Seq[org.apache.spark.mllib.recommendation.Rating] = WrappedArray(Rating(789,1012,4.0), Rating(789,127,5.0), Rating(789,475,5.0), Rating(789,93,4.0), Rating(789,1161,3.0), Rating(789,286,1.0), Rating(789,293,4.0), Rating(789,9,5.0), Rating(789,50,5.0), Rating(789,294,3.0), Rating(789,181,4.0), Rating(789,1,3.0), Rating(789,1008,4.0), Rating(789,508,4.0), Rating(789,284,3.0), Rating(789,1017,3.0), Rating(789,137,2.0), Rating(789,111,3.0), Rating(789,742,3.0), Rating(789,248,3.0), Rating(789,249,3.0), Rating(789,1007,4.0), Rating(789,591,3.0), Rating(789,150,5.0), Rating(789,276,5.0), Rating(789,151,2.0), Rating(789,129,5.0), Rating(789,100,5.0), Rating(789,741,5.0), Rating(789,288,3.0), Rating(789,762,3.0), Rating(789,628,3.0), Rating(789,124,4.0))
```

```
scala> moviesForUser.sortBy(-_.rating).take(10).map(rating =>  
(titles(rating.product), rating.rating)).foreach(println)
```

```
(Godfather, The (1972),5.0)  
(Trainspotting (1996),5.0)  
(Dead Man Walking (1995),5.0)  
(Star Wars (1977),5.0)  
(Swingers (1996),5.0)  
(Leaving Las Vegas (1995),5.0)  
(Bound (1996),5.0)  
(Fargo (1996),5.0)  
(Last Supper, The (1995),5.0)  
(Private Parts (1997),4.0)
```

Top 10 Recommendation for userid 789

```
scala> val topKRecs = model.recommendProducts(789,10)  
scala> topKRecs.map(rating => (titles(rating.product),  
rating.rating)) .foreach(println)
```

```
(GoodFellas (1990),5.561893309975536)  
(Apocalypse Now (1979),5.359509740087787)  
(Being There (1979),5.253109995320087)  
(Carrie (1976),5.214960672591296)  
(Aliens (1986),5.18467232737804)  
(Psycho (1960),5.184123552034558)  
(One Flew Over the Cuckoo's Nest (1975),5.174956083257432)  
(Full Monty, The (1997),5.145369582639113)  
(Flirting With Disaster (1996),5.128468420256269)  
(Heavy Metal (1981),5.112027118820185)
```

Evaluating Performance: Mean Squared Error

```
scala> val actualRating = moviesForUser.take(1)(0)
scala> val predictedRating = model.predict(789,
actualRating.product)
scala> val squaredError = math.pow(predictedRating -
actualRating.rating, 2.0)
```

```
scala> val actualRating = moviesForUser.take(1)(0)
actualRating: org.apache.spark.mllib.recommendation.Rating = Rating(789,1012,4.0)

scala> val predictedRating = model.predict(789, actualRating.product)
predictedRating: Double = 3.9903742702273326

scala> val squaredError = math.pow(predictedRating - actualRating.rating, 2.0)
squaredError: Double = 9.265467365641563E-5
```

Overall Mean Squared Error

```
scala> val usersProducts = ratings.map{ case Rating(user, product, rating) => (user, product)}
```

```
scala> val predictions = model.predict(usersProducts).map{
```

```
  case Rating(user, product, rating) => ((user, product), rating)}
```

```
scala> val ratingsAndPredictions = ratings.map{
```

```
  case Rating(user, product, rating) => ((user, product), rating)}
```

```
} .join(predictions)
```

```
scala> val MSE = ratingsAndPredictions.map{
```

```
  case ((user, product), (actual, predicted)) =>
```

```
    math.pow((actual - predicted), 2)}
```

```
} .reduce(_ + _) / ratingsAndPredictions.count
```

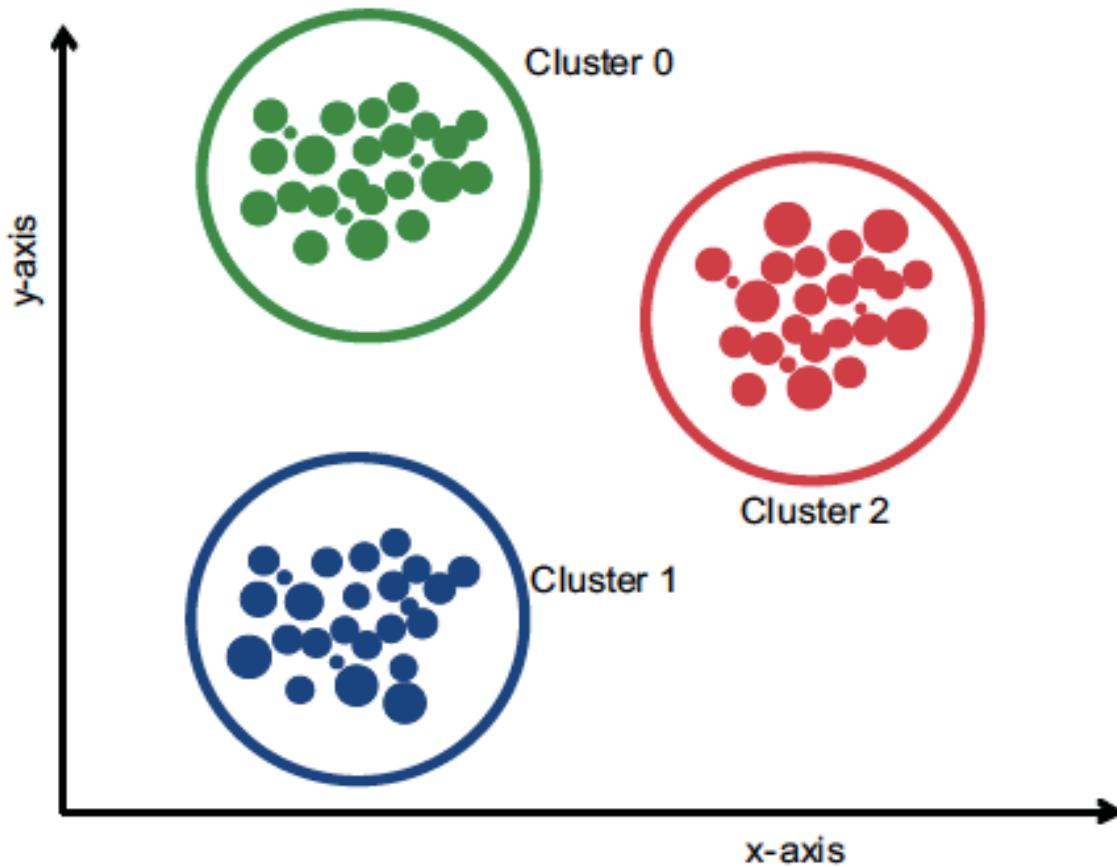
```
scala> println("Mean Squared Error = " + MSE)
```

```
Mean Squared Error = 0.097528985120825
```

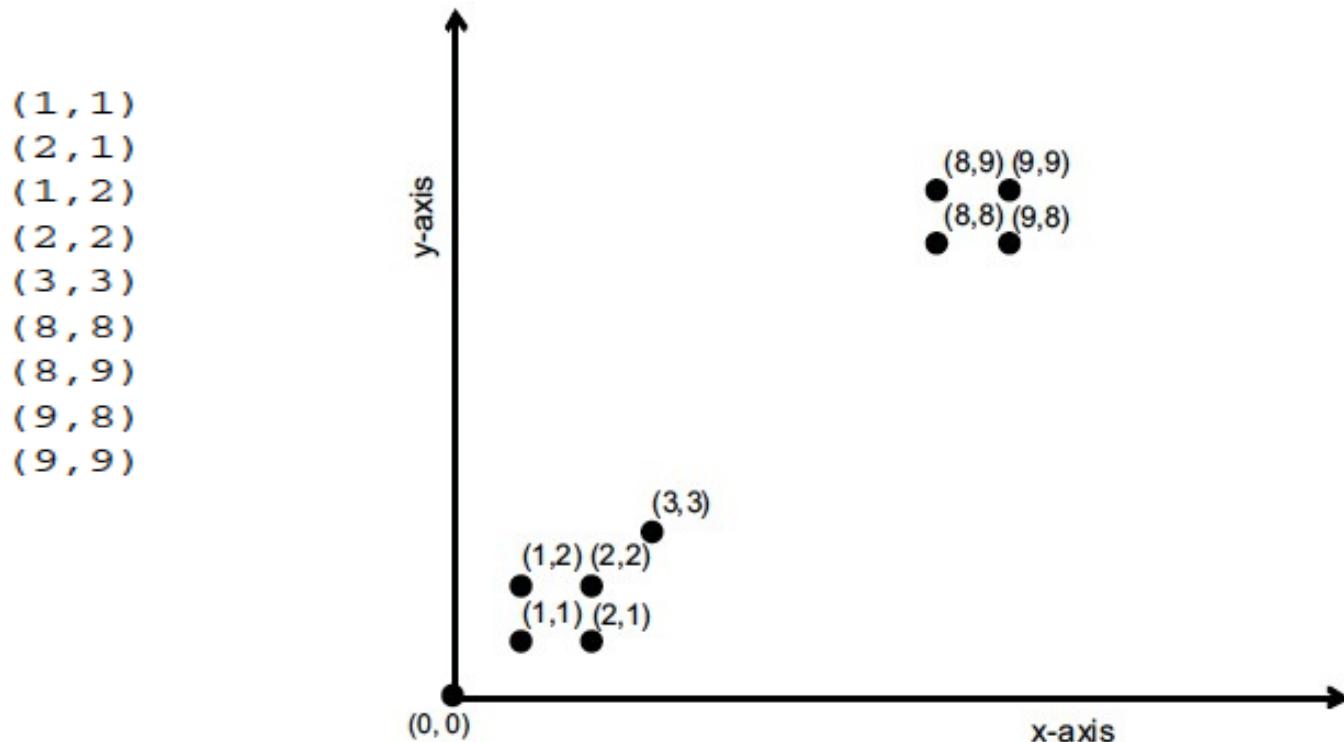
Clustering using K-Means

Clustering use cases

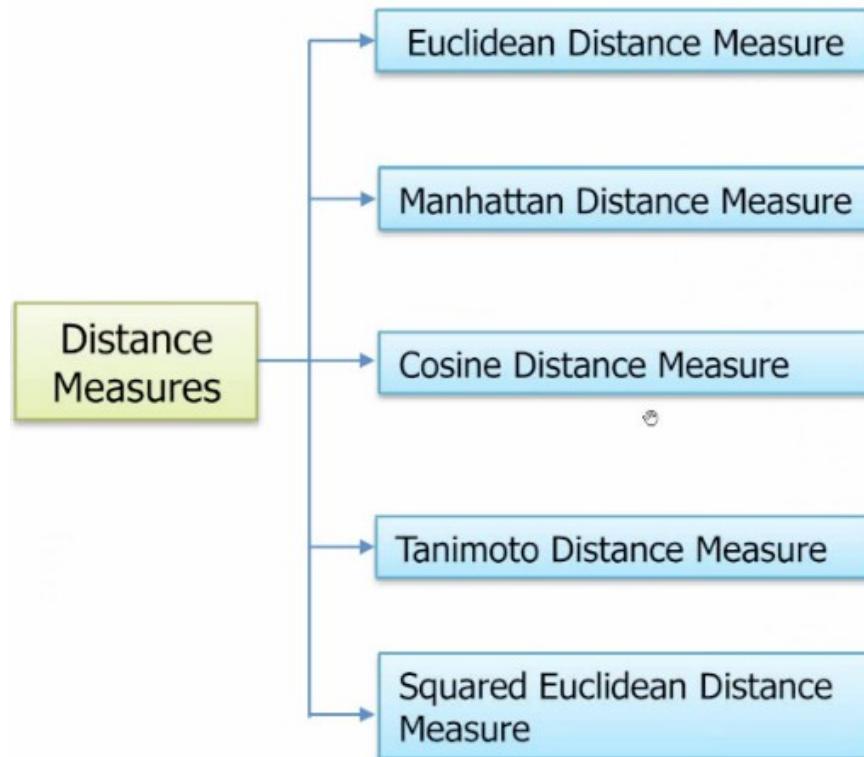
- Market segmentation
- Social network analysis: Finding a coherent group of people in the social network for ad targeting
- Data center computing clusters
- Real estate: Identifying neighborhoods based on similar features
- Text analysis: Dividing text documents, such as novels or essays, into genres



Sample Data



Distance Measures



Distance Measures

- Euclidean distance measure

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

- Squared Euclidean distance measure

$$d = (a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2$$

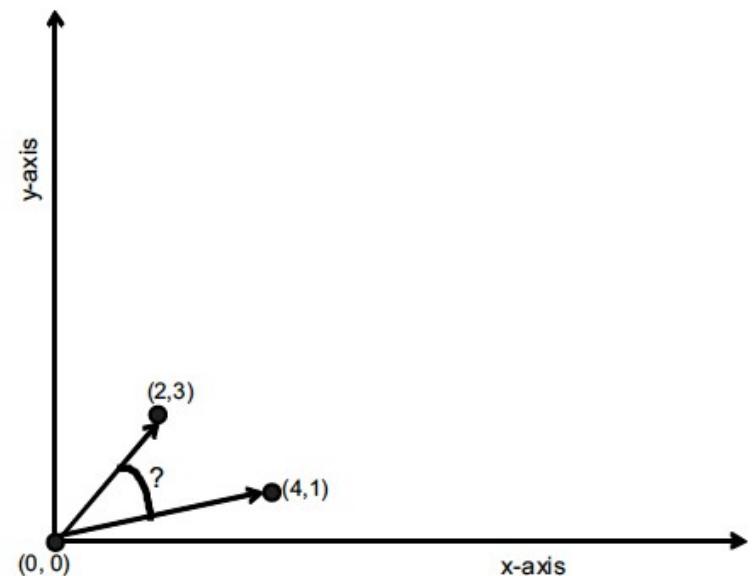
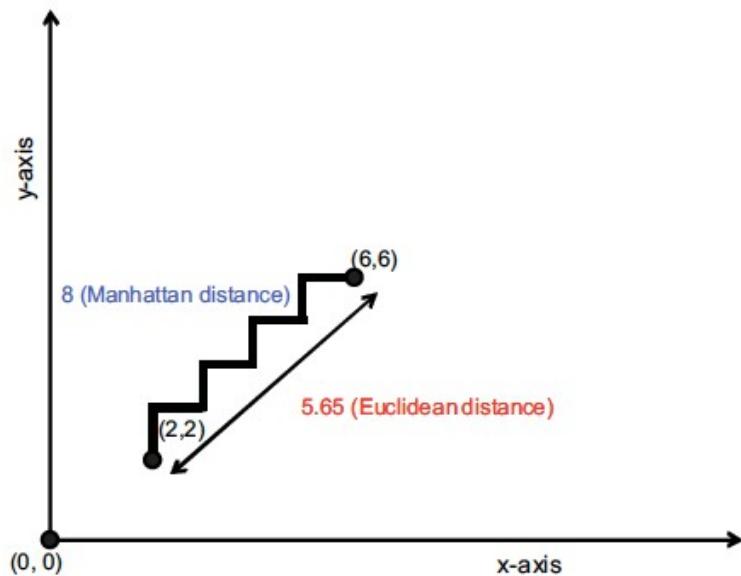
- Manhattan distance measure

$$d = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$$

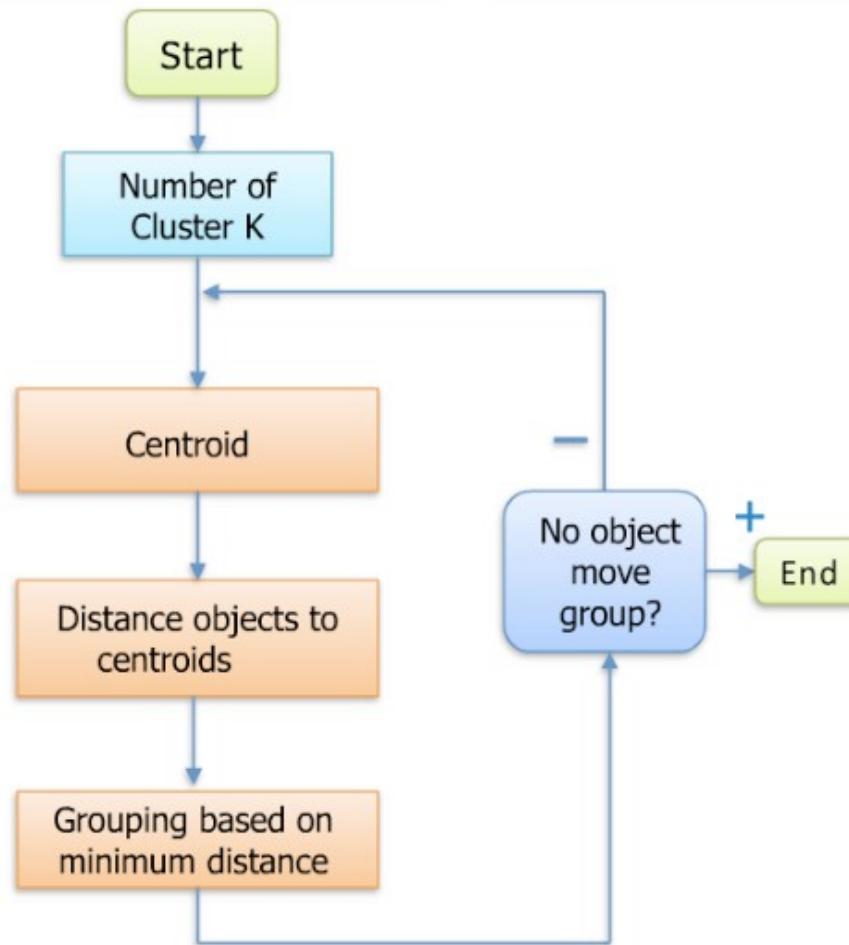
- Cosine distance measure

$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \dots + a_n b_n)}{(\sqrt{a_1^2 + a_2^2 + \dots + a_n^2}) \sqrt{(b_1^2 + b_2^2 + \dots + b_n^2)})}$$

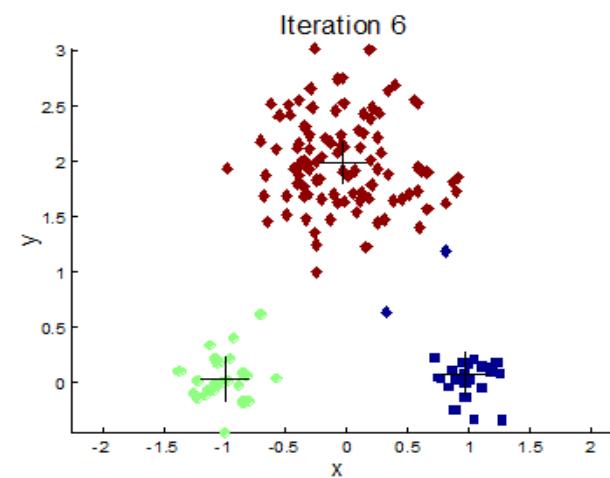
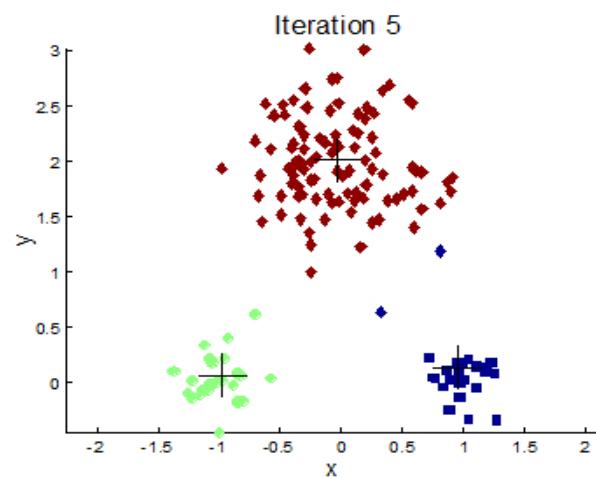
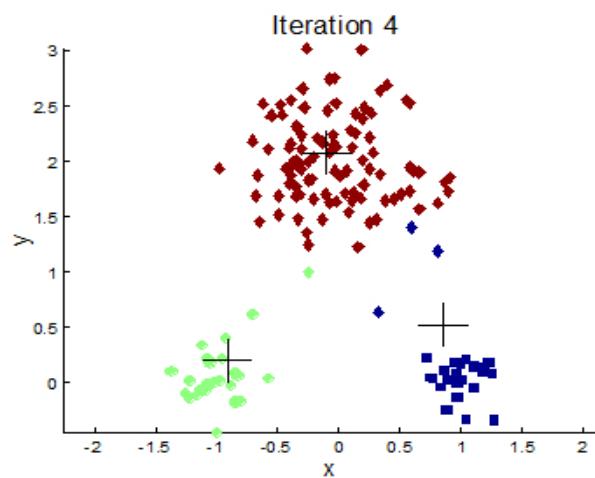
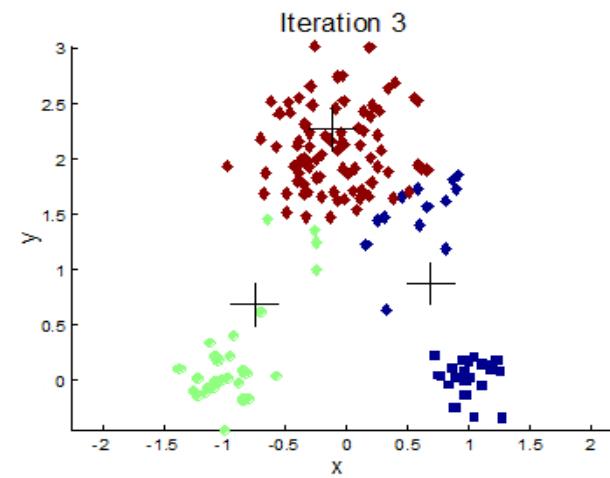
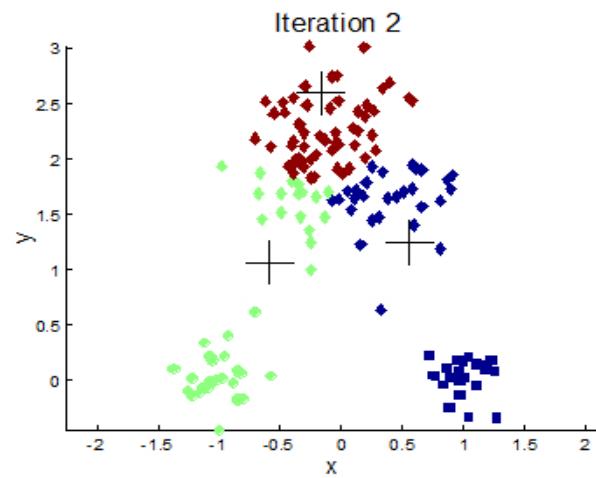
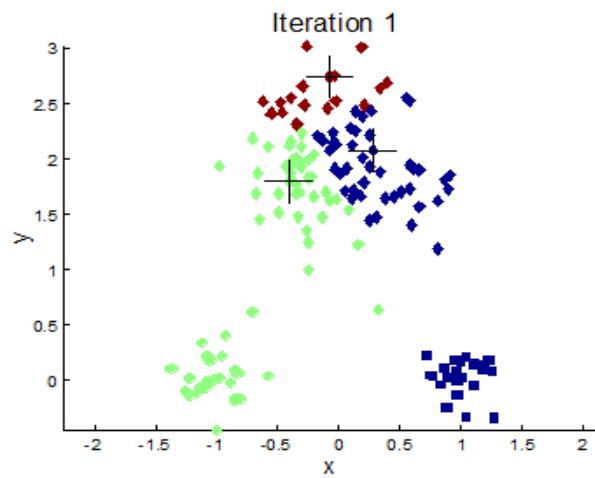
Distance Measures



K-Means Clustering

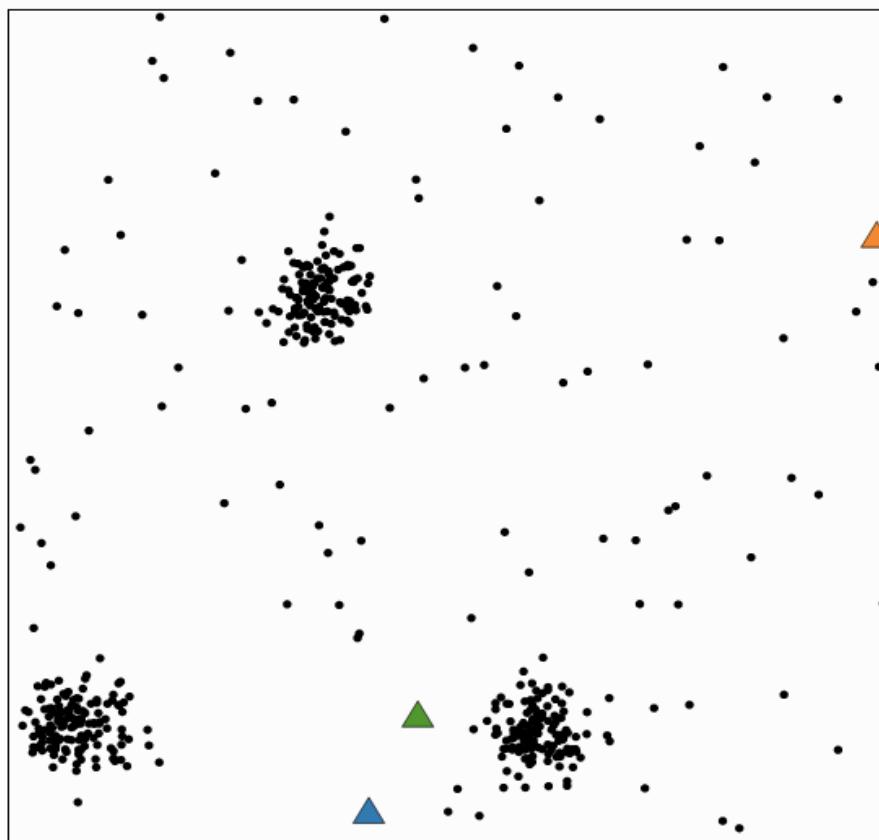


Example of K-Means Clustering



<http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html>

Visualizing K-Means Clustering



Mean square point-centroid distance: not yet calculated

Made by Karanveer Mohan for EE103. Source code on [Github](#).

The k -means algorithm is an iterative method for clustering a set of N points (vectors) into k groups or clusters of points.

Algorithm

Repeat until convergence:

Find closest centroid

Find the closest centroid to each point, and group points that share the same closest centroid.

Update centroid

Update each centroid to be the mean of the points in its group.

[Find closest centroid](#)

Data

Clustered points Random
Number of clusters : 3
Number of centroids: 3

[New points](#)

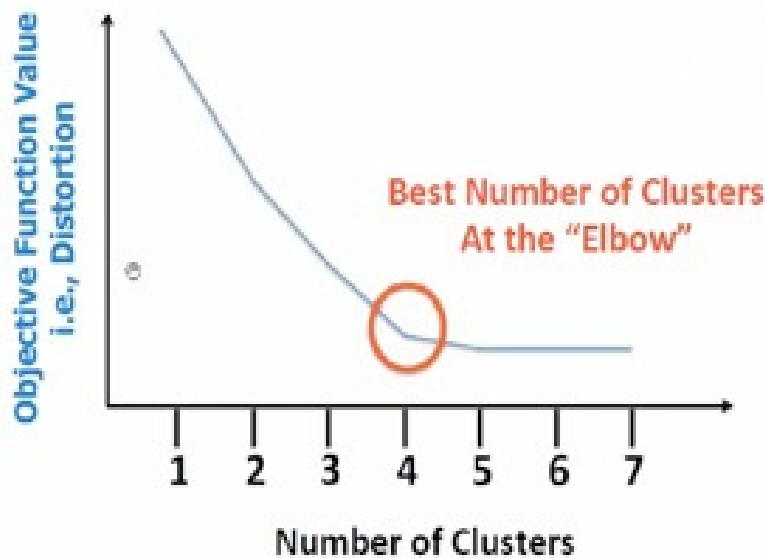
[New centroids](#)

K-Means with different distance measures

Distance measure	Number of Iterations	Vectors ^a In cluster 0	Vectors In cluster 1
EuclideanDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8
SquaredEuclideanDistanceMeasure	5	0, 1, 2, 3, 4	5, 6, 7, 8
ManhattanDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8
CosineDistanceMeasure	1	1	0, 2, 3, 4, 5, 6, 7, 8
TanimotoDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8

Choosing number of clusters

Elbow method



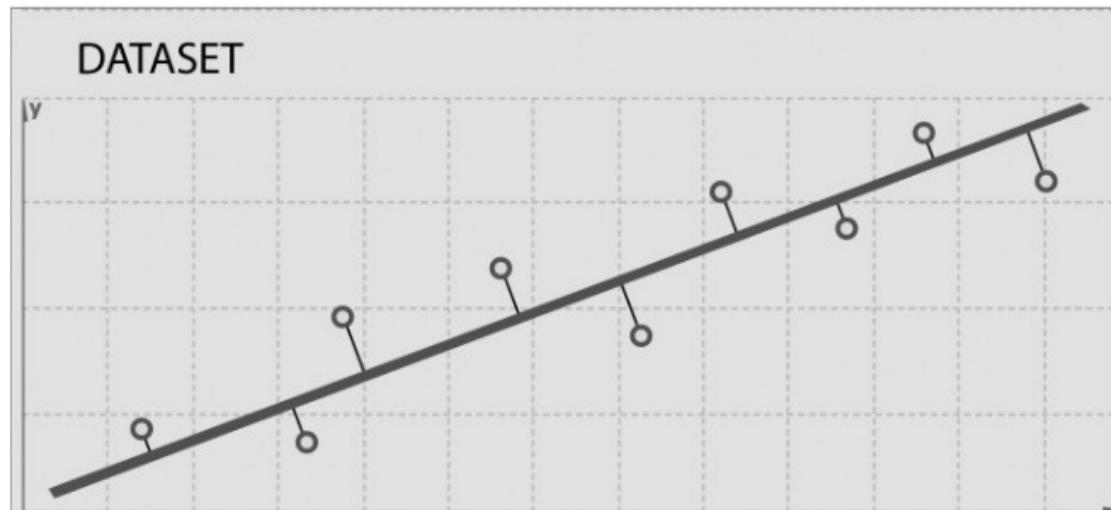
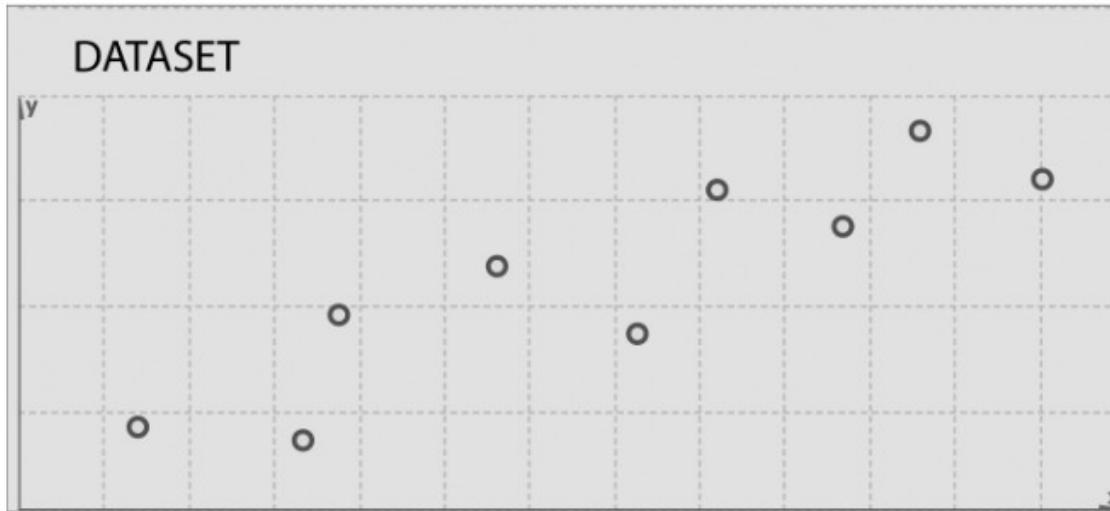
$$\text{Distortion} = \sum_{i=1}^m (x_i - c_i)^2 = \sum_{j=1}^k \sum_{i \in \text{OwnedBy}(\mu_j)} (x_i - \mu_j)^2$$

(within cluster sum of squares)

Dimensionality reduction

- Process of reducing the number of dimensions or features.
- Dimensionality reduction serves several purposes
 - Data compression
 - Visualization
- The most popular algorithm: Principal component analysis (PCA).

Dimensionality reduction



Dimensionality reduction with SVD

- Singular Value Decomposition (SVD): is based on a theorem from linear algebra that a rectangular matrix A can be broken down into a product of three matrices

$$A = USV^T$$

$$U^T U = I$$

$$V^T V = I$$

Dimensionality reduction with SVD

- The basic idea behind SVD
 - Take a high dimension, a highly variable set of data points
 - Reduce it to a lower dimensional space that exposes the structure of the original data more clearly and orders it from the most variation to the least.
- So we can simply ignore variation below a certain threshold to massively reduce the original data, making sure that the original relationship interests are retained.

Hands-on

Clustering on MovieLens Dataset

Extracting features from the MovieLens dataset

```
scala> val rawData =  
sc.textFile("hdfs://user/cloudera/movielens/u.item")  
scala> println(movies.first)
```

1|Toy Story (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0

```
scala> val genres =  
sc.textFile("hdfs://user/cloudera/movielens/u.genre")  
scala> genres.take(5).foreach(println)
```

unknown|0
Action|1
Adventure|2
Animation|3
Children's|4

Extracting features from the MovieLens dataset (cont.)

```
scala> val genreMap = genres.filter(!_isEmpty).map(line =>  
line.split("\\|")).map(array=> (array(1),  
array(0))).collectAsMap
```

```
genreMap: scala.collection.Map[String, String] = Map(2 -> Adventure, 5 -> Comedy, 12  
-> Musical, 15 -> Sci-Fi, 8 -> Drama, 18 -> Western, 7 -> Documentary, 17 -> War, 1  
-> Action, 4 -> Children's, 11 -> Horror, 14 -> Romance, 6 -> Crime, 0 -> unknown, 9  
-> Fantasy, 16 -> Thriller, 3 -> Animation, 10 -> Film-Noir, 13 -> Mystery)
```

Extracting features from the MovieLens dataset (cont.)

```
scala> val titlesAndGenres = movies.map(_.split("\\\\|")).map
{ array =>
    val genres = array.toSeq.slice(5, array.size)
    val genresAssigned = genres.zipWithIndex.filter { case (g, idx) =>
        g == "1"
    }.map { case (g, idx) =>
        genreMap(idx.toString)
    }
    (array(0).toInt, (array(1), genresAssigned))
}

scala> println(titlesAndGenres.first)
(1,(Toy Story (1995), ArrayBuffer(Animation, Children's, Comedy)))
```

Training the recommendation model

```
scala> :paste
import org.apache.spark.mllib.recommendation.ALS
import org.apache.spark.mllib.recommendation.Rating
val rawData =
sc.textFile("hdfs:///user/cloudera/movielens/u.data")
val rawRatings = rawData.map(_.split("\t")).take(3))
val ratings = rawRatings.map{ case Array(user, movie,
rating) => Rating(user.toInt, movie.toInt,
rating.toDouble) }
ratings.cache
val alsModel = ALS.train(ratings, 50, 10, 0.1)
import org.apache.spark.mllib.linalg.Vectors
val movieFactors = alsModel.productFeatures.map { case (id,
factor) => (id, Vectors.dense(factor)) }
val movieVectors = movieFactors.map(_._2)
val userFactors = alsModel.userFeatures.map { case (id,
factor) => (id, Vectors.dense(factor)) }
val userVectors = userFactors.map(_._2)
```

Normalization

```
scala> :paste
import org.apache.spark.mllib.linalg.distributed.RowMatrix
val movieMatrix = new RowMatrix(movieVectors)
val movieMatrixSummary =
movieMatrix.computeColumnSummaryStatistics()
val userMatrix = new RowMatrix(userVectors)
val userMatrixSummary =
userMatrix.computeColumnSummaryStatistics()
println("Movie factors mean: " + movieMatrixSummary.mean)
println("Movie factors variance: " +
movieMatrixSummary.variance)
println("User factors mean: " + userMatrixSummary.mean)
println("User factors variance: " +
userMatrixSummary.variance)
```

Output from Normalization

```
Movie factors mean:  
[0.28047737659519767, 0.26886479057520024, 0.2935579964446398, 0.27821738264113755,  
 ...  
Movie factors variance:  
[0.038242041794064895, 0.03742229118854288, 0.044116961097355877, 0.057116244055791986  
 / ...  
User factors mean:  
[0.2043520841572601, 0.22135773814655782, 0.2149706318418221, 0.23647602029329481, ...  
User factors variance:  
[0.037749421148850396, 0.02831191551960241, 0.032831876953314174, 0.036775110657850954  
 / ...
```

Training a clustering model

```
val userClusterModel = KMeans.train(userVectors, numClusters, numIterations,  
numRuns)
```

```
scala> import org.apache.spark.mllib.clustering.KMeans  
scala> val numClusters = 5  
scala> val numIterations = 10  
scala> val numRuns = 3  
scala> val movieClusterModel = KMeans.train(movieVectors,  
numClusters, numIterations, numRuns)
```

Making predictions using a clustering model

```
scala> val movie1 = movieVectors.first
scala> val movieCluster = movieClusterModel.predict(movie1)
scala> val predictions =
movieClusterModel.predict(movieVectors)

scala> println(predictions.take(10).mkString(","))
3,0,0,0,0,0,0,2,0,0
```

Interpreting cluster predictions

```
scala> :paste
import breeze.linalg._
import breeze.numerics.pow
def computeDistance(v1: DenseVector[Double], v2: DenseVector[Double]) = pow(v1 - v2, 2).sum
val titlesWithFactors = titlesAndGenres.join(movieFactors)
val moviesAssigned = titlesWithFactors.map { case (id, ((title, genres), vector)) =>
    val pred = movieClusterModel.predict(vector)
    val clusterCentre = movieClusterModel.clusterCenters(pred)
    val dist =
        computeDistance(DenseVector(clusterCentre.toArray),
        DenseVector(vector.toArray))
    (id, title, genres.mkString(" ")), pred, dist)
}
}
```

Interpreting cluster predictions (cont.)

```
val clusterAssignments = moviesAssigned.groupBy { case (id, title, genres, cluster, dist) => cluster }.collectAsMap
for ( (k, v) <- clusterAssignments.toSeq.sortBy(_.value)) {
    println(s"Cluster $k:")
    val m = v.toSeq.sortBy(_.value)
    println(m.take(20).map { case (_, title, genres, _, d) => (title, genres, d) }.mkString("\n"))
    println("=====\\n")
}
```

Cluster 0:

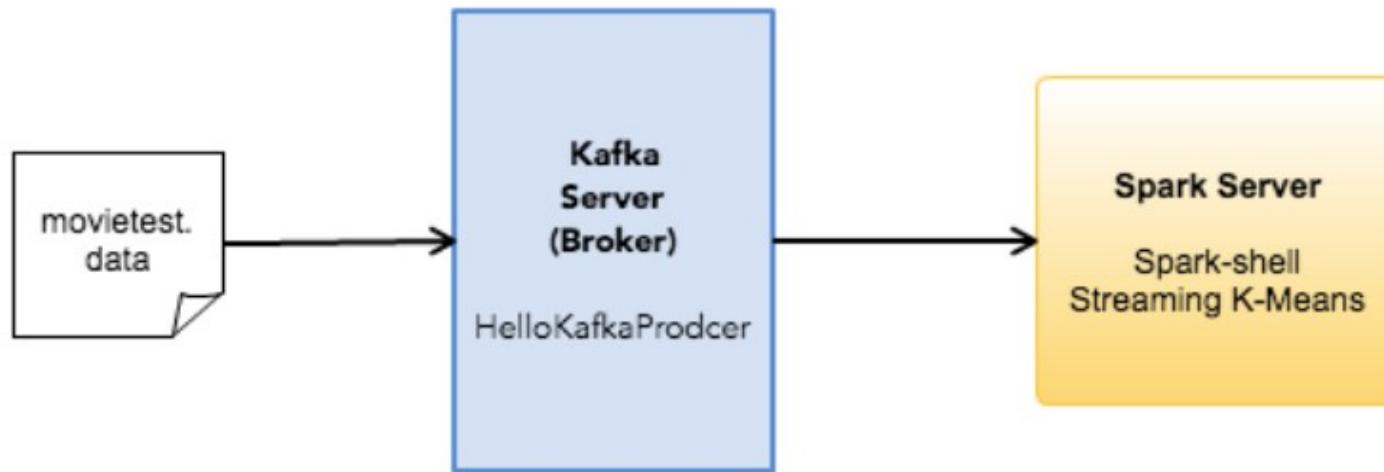
(Last Time I Saw Paris, The (1954), Drama, 0.15088816303186323)
(Witness (1985), Drama Romance Thriller, 0.2018937474956098)
(Substance of Fire, The (1996), Drama, 0.26580331444304967)
(King of the Hill (1993), Drama, 0.27090751738692787)
(Mamma Roma (1962), Drama, 0.30508676553769926)
(Beans of Egypt, Maine, The (1994), Drama, 0.31880331503649484)
(Scream of Stone (Schrei aus Stein) (1991), Drama, 0.33904627647373703)
(All Things Fair (1996), Drama, 0.3449680047501059)
(Angel and the Badman (1947), Western, 0.3519092167012976)
(Nelly & Monsieur Arnaud (1995), Drama, 0.3630059139776454)
(Così (1996), Comedy, 0.3781303586431162)
(Object of My Affection, The (1998), Comedy Romance, 0.39398318062694987)
(Wife, The (1995), Comedy Drama, 0.399375163806288)
(They Made Me a Criminal (1939), Crime Drama, 0.42158316491602227)
(Spellbound (1945), Mystery Romance Thriller, 0.42881078192699107)
(Spirits of the Dead (Tre passi nel delirio) (1968), Horror, 0.43991392186284806)
(Farewell to Arms, A (1932), Romance War, 0.44324604591789385)
(Sleepover (1995), Comedy Drama, 0.4473239416648149)
(Love Is All There Is (1996), Comedy Drama, 0.4473239416648149)
(Century (1993), Drama, 0.4473239416648149)

Real-time Machine Learning using Streaming K-Means

Online learning with Spark Streaming

- **Streaming regression**
 - trainOn: This takes DStream[LabeledPoint] as its argument.
 - predictOn: This also takes DStream[LabeledPoint].
- **Streaming KMeans**
 - An extension of the mini-batch K-means algorithm

Streaming K-Means Program



MovieLen Training Dataset

- The rows of the training text files must be vector data in the form

[x₁, x₂, x₃, . . . , x_n]

1) Type command > wget

<https://s3.amazonaws.com/imcbucket/data/movietest.data>

2) Type command > more movietest.data

```
[196,242,3]
[186,302,3]
[22,377,1]
[244,51,2]
[166,346,1]
[298,474,4]
[115,265,2]
[253,465,5]
[305,451,3]
[6,86,3]
```

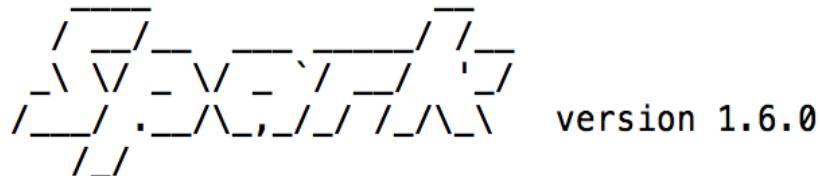
Install & Start Kafka Server

```
# wget http://www-us.apache.org/dist/kafka/0.9.0.1/kafka_2.10-  
0.9.0.1.tgz  
# tar xzf kafka_2.10-0.9.0.1.tgz  
# cd kafka_2.10-0.9.0.1  
# bin/kafka-server-start.sh config/server.properties&
```

```
[2016-06-23 04:37:21,426] INFO Kafka commitId : 23c69d62a0cabf06 (o  
rg.apache.kafka.common.utils.AppInfoParser)  
[2016-06-23 04:37:21,430] INFO [Kafka Server 0], started (kafka.ser  
ver.KafkaServer)  
[2016-06-23 04:37:21,446] INFO New leader is 0 (kafka.server.Zookeee  
perLeaderElector$LeaderChangeListener)
```

Start Spark-shell with extra memory

```
[root@quickstart ~]# spark-shell --driver-memory 1G
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/jars/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
```



Streaming K-Means

```
$ scala> :paste
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.clustering.StreamingKMeans
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.storage.StorageLevel
import StorageLevel._
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.kafka.KafkaUtils
val ssc = new StreamingContext(sc, Seconds(2))
```

```
val kafkaStream = KafkaUtils.createStream(ssc,  
"localhost:2181","spark-streaming-consumer-group", Map("java-  
topic" -> 5))  
  
val lines = kafkaStream.map(_._2)  
  
val ratings = lines.map(Vectors.parse)  
  
val numDimensions = 3  
  
val numClusters = 5  
  
val model = new StreamingKMeans()  
  
.setK(numClusters)  
.setDecayFactor(1.0)  
.setRandomCenters(numDimensions, 0.0)  
  
model.trainOn(ratings)  
  
model.predictOn(ratings).print()  
  
ssc.start()  
  
ssc.awaitTermination()
```

Running HelloKafkaProducer on another windows

- Open a new ssh windows

```
root@imcdocker:/home/imcinstitute# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              NAMES
6ea0909137a3        clouderakafka:latest   "/usr/bin/docker-qui"   11 minutes ago   Up 11 minutes      0.0.0.0:8888->8888/tcp   cranky_franklin
root@imcdocker:/home/imcinstitute# docker exec -it 6ea0909137a3 /bin/bash
```

Java Code: Kafka Producer

```
import java.util.Properties;  
  
import kafka.producer.KeyedMessage;  
import kafka.producer.ProducerConfig;  
import java.io.*;  
  
public class HelloKafkaProducer {  
    final static String TOPIC = "java-topic";  
    public static void main(String[] argv) {  
        Properties properties = new Properties();  
  
        properties.put("metadata.broker.list","localhost:9092");  
  
        properties.put("serializer.class","kafka.serializer.StringEncoder");  
    }  
}
```

Java Code: Kafka Producer (cont.)

```
try(BufferedReader br = new BufferedReader(new
FileReader(argv[0]))) {
    StringBuilder sb = new StringBuilder();
    ProducerConfig producerConfig = new
ProducerConfig(properties);
    kafka.javaapi.producer.Producer<String, String>
producer = new kafka.javaapi.producer.Producer<String,
String>(producerConfig);
    String line = br.readLine();

    while (line != null) {
        KeyedMessage<String, String> message
=new KeyedMessage<String, String>(TOPIC, line);
        producer.send(message);
        line = br.readLine();
    }
}
```

Java Code: Kafka Producer (cont.)

```
        producer.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    }

}
```

Compile & Run the program

```
// Using a vi Editor to edit the sourcecode  
  
# vi HelloKafkaProducer.java  
  
// Alternatively  
  
# wget  
https://s3.amazonaws.com/imcbucket/apps/HelloKafkaProducer.java  
  
// Compile progeram  
  
# export CLASSPATH=".:/root/kafka_2.10-0.9.0.1/libs/*"  
# javac HelloKafkaProducer.java  
  
//prepare the data  
  
# cd  
  
# wget https://s3.amazonaws.com/imcbucket/input/pg2600.txt  
# cd kafka_2.10-0.9.0.1  
  
// Run the program  
  
# java HelloKafkaProducer /root/movietest.data
```

Example Result

```
16/07/10 09:12:18 WARN storage.BlockManager: Block input-0-1468141938000 replicated  
to only 0 peer(s) instead of 1 peers
```

```
-----  
Time: 1468141938000 ms
```

```
-----  
4  
0  
3  
2  
1  
0  
2  
4  
1  
3  
...  
...
```

Project:

**Machine Learning using
NYC Taxi Trip Data**

NYC's Taxi Trip Data

<http://www.andresmh.com/nyctaxitrips/>

NYC Taxi Trips

Data obtained through a FOIA request

[View the Project on GitHub](#)
andresmh/nyctaxitrips

[View On
GitHub](#)

[Read the
story](#)

[Check the
viz](#)

NYC Taxi Data

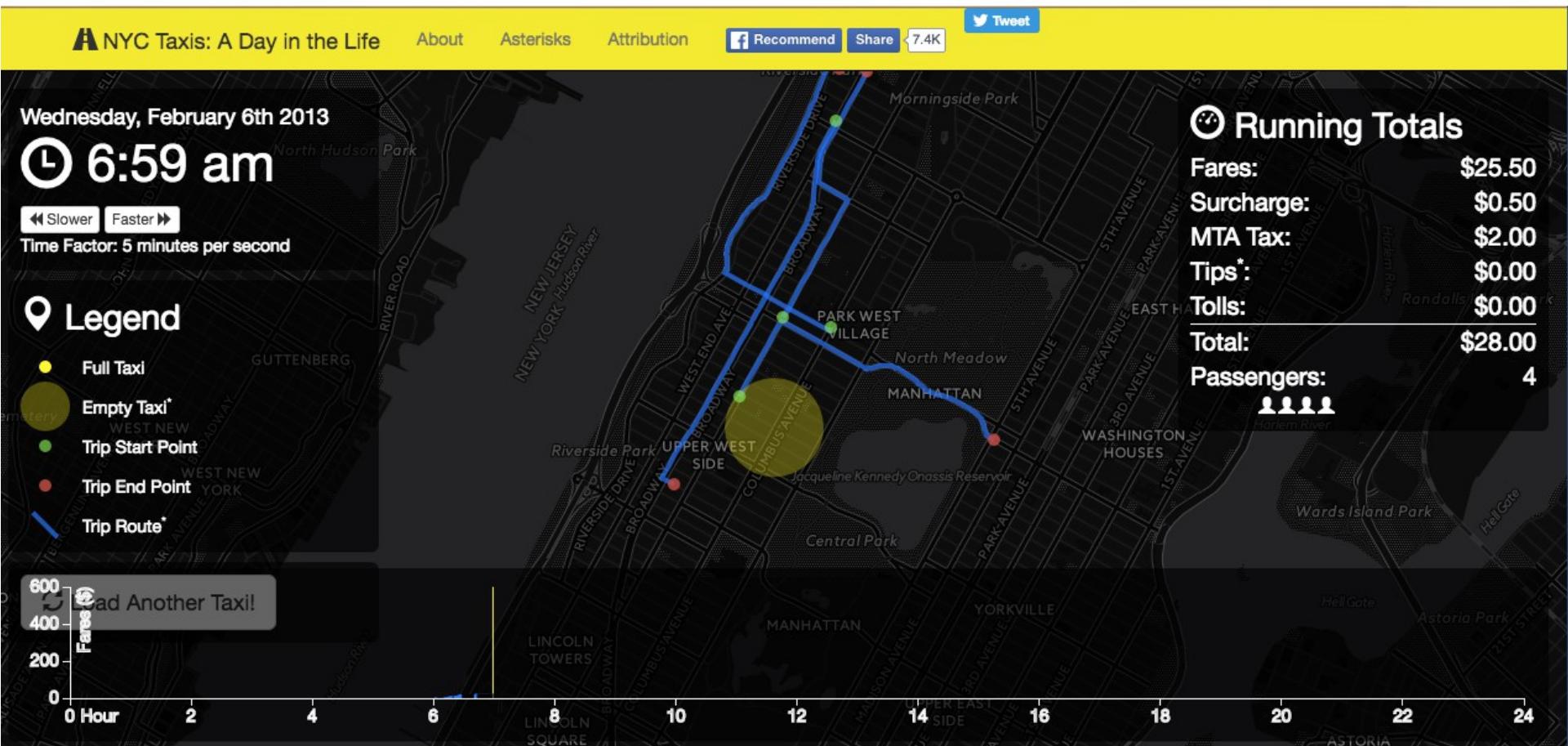
Trips

1. [trip_data_1.csv.zip](#)
2. [trip_data_2.csv.zip](#)
3. [trip_data_3.csv.zip](#)
4. [trip_data_4.csv.zip](#)
5. [trip_data_5.csv.zip](#)
6. [trip_data_6.csv.zip](#)
7. [trip_data_7.csv.zip](#)
8. [trip_data_8.csv.zip](#)
9. [trip_data_9.csv.zip](#)
10. [trip_data_10.csv.zip](#)
11. [trip_data_11.csv.zip](#)
12. [trip_data_12.csv.zip](#)



NYC Taxi : A Day in Life

<http://nyctaxi.herokuapp.com/>



Temporal Data with JodaTime and NScalaTime

```
import com.github.nscala_time.time.Imports._

val dt1 = new DateTime(2014, 9, 4, 9, 0)
val dt2 = new DateTime(2014, 10, 31, 15, 0)
dt1 < dt2
val dt3 = dt1 + 60.days
dt3 > dt2
```

```
import java.text.SimpleDateFormat

val format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
val date = format.parse("2014-10-12 10:30:44")
val datetime = new DateTime(date)
```

Temporal Data with JodaTime and NScalaTime

```
val d = new Duration(dt1, dt2)
d.getMillis
d.getStandardHours
d.getStandardDays
```

Exploring the Esri Geometry API

```
import com.esri.core.geometry.Geometry
import com.esri.core.geometry.GeometryEngine
import com.esri.core.geometry.SpatialReference
class RichGeometry(val geometry: Geometry,
  val spatialReference: SpatialReference =
    SpatialReference.create(4326)) {
  def area2D() = geometry.calculateArea2D()
  def contains(other: Geometry): Boolean = {
    GeometryEngine.contains(geometry, other,
      spatialReference)
  }
  def distance(other: Geometry): Double = {
    GeometryEngine.distance(geometry, other,
      spatialReference)
  }
}
```

Exploring the Esri Geometry API (cont.)

```
object RichGeometry {  
    implicit def wrapRichGeo(g: Geometry) = {  
        new RichGeometry(g)  
    }  
}  
  
import RichGeometry._
```

Intro to GeoJSON

```
$ wget https://nycdatastable.s3.amazonaws.com/  
/2013-08-19T18:15:35.172Z/nyc-borough-boundaries-  
polygon.geojson
```

```
$ mv nyc-borough-boundaries-polygon.geojson nyc-  
boroughs.geojson
```

```
import spray.json.JsValue  
  
case class Feature(  
    val id: Option[JsValue],  
    val properties: Map[String, JsValue],  
    val geometry: RichGeometry) {  
    def apply(property: String) = properties(property)  
    def get(property: String) = properties.get(property)  
}
```

Intro to GeoJSON (cont.)

```
case class FeatureCollection(features: Array[Feature])  
  extends IndexedSeq[Feature] {  
  def apply(index: Int) = features(index)  
  def length = features.length  
}  
  
implicit object FeatureJsonFormat extends  
  RootJsonFormat[Feature] {  
  def write(f: Feature) = {  
    val buf = scala.collection.mutable.ArrayBuffer(  
      "type" -> JsString("Feature") ,  
      "properties" -> JsObject(f.properties) ,  
      "geometry" -> f.geometry.toJson)  
    f.id.foreach(v => { buf += "id" -> v })  
    JsObject(buf.toMap)  
  }  
}
```

Intro to GeoJSON (cont.)

```
def read(value: JsValue) = {  
    val js0 = value.asJsObject  
    val id = js0.fields.get("id")  
    val properties =  
        js0.fields("properties").asJsObject.fields  
    val geometry =  
        js0.fields("geometry").convertTo[RichGeometry]  
    Feature(id, properties, geometry)  
}  
}
```

Preparing the New York City Taxi Trip Data

```
$ hadoop fs -mkdir taxidata
$ hadoop fs -put trip_data_1.csv taxidata/
$ mvn package
$ spark-shell --jars target/ch08-geotime-1.0.0.jar

val taxiRaw = sc.textFile("taxidata")
val taxiHead = taxiRaw.take(10)
taxiHead.foreach(println)
```

Preparing the New York City Taxi Trip Data (cont.)

```
import com.esri.core.geometry.Point
import com.github.nscala_time.time.Imports._

case class Trip(
  pickupTime: DateTime,
  dropoffTime: DateTime,
  pickupLoc: Point,
  dropoffLoc: Point)

val formatter = new SimpleDateFormat(
  "yyyy-MM-dd HH:mm:ss")

def point(longitude: String, latitude: String): Point = {
  new Point(longitude.toDouble, latitude.toDouble)
}
```

Preparing the New York City Taxi Trip Data (cont.)

```
def parse(line: String): (String, Trip) = {  
    val fields = line.split(',')  
    val license = fields(1)  
    val pickupTime = new DateTime(formatter.parse(fields(5)))  
    val dropoffTime = new DateTime(formatter.parse(fields(6)))  
    val pickupLoc = point(fields(10), fields(11))  
    val dropoffLoc = point(fields(12), fields(13))  
  
    val trip = Trip(pickupTime, dropoffTime, pickupLoc,  
dropoffLoc)  
    (license, trip)  
}
```

Handling Invalid Records at Scale

```
def safe[S, T](f: S => T): S => Either[T, (S, Exception)] =  
{  
    new Function[S, Either[T, (S, Exception)]] with  
    Serializable {  
        def apply(s: S): Either[T, (S, Exception)] = {  
            try {  
                Left(f(s))  
            } catch {  
                case e: Exception => Right((s, e))  
            }  
        }  
    }  
}
```

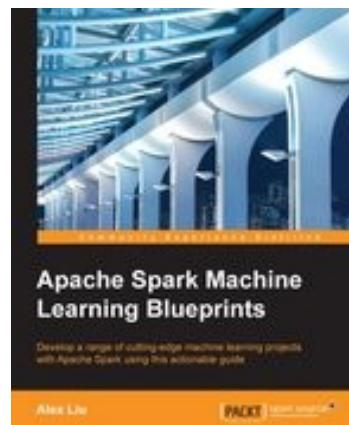
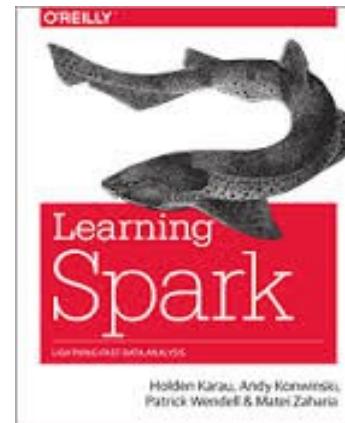
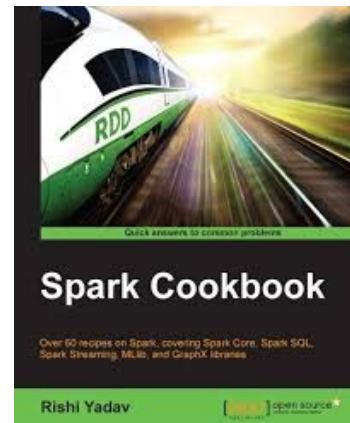
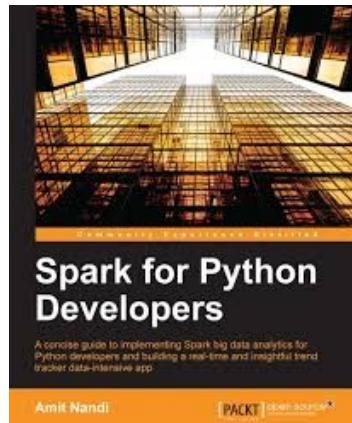
Handling Invalid Records at Scale (cont.)

```
val safeParse = safe(parse)
val taxiParsed = taxiRaw.map(safeParse)
taxiParsed.cache()

taxiParsed.map(_.isLeft) .
countByValue() .
foreach(println)

...
val taxiBad = taxiParsed.
filter(_.isRight) .
map(_.right.get)
```

Recommended Books





Big Data using Hadoop Workshop

27-28 July 2016

<http://www.imcinstitute.com/hadoop> Tel: 088-192-7975



Instructor:
Dr.Thanachart
Numnonda

Hadoop



HIVE



Hadoop



YARN

Fee: 5,500 Baht
(Early Bird)
Ex VAT

Fee: include
Lunch Break
Course Material

Venue
Connection
@MRT Ladprao

Thank you

www.imcinstitute.com
www.facebook.com/imcinstitute