

# Hadoop, Spark and R

Yanchang Zhao  
RDataMining.com

The La Trobe EoY Analytics Symposium 2016  
La Trobe University, Melbourne

17 November 2016



# Outline

Introduction

Hadoop

Spark

R and Big Data

Set Up a Hadoop/Spark Cluster

Online Resources

# Big Data <sup>1</sup>

- ▶ Volume: amount of data; from Terabytes to Petabytes
- ▶ Velocity: speed of data in and out; real time
- ▶ Variety: range of data types and sources; text, images, audio, video

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Big\\_data](https://en.wikipedia.org/wiki/Big_data)

# Big Data <sup>1</sup>

- ▶ Volume: amount of data; from Terabytes to Petabytes
- ▶ Velocity: speed of data in and out; real time
- ▶ Variety: range of data types and sources; text, images, audio, video
- ▶ Variability: inconsistency of data
- ▶ Veracity: quality of data

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Big\\_data](https://en.wikipedia.org/wiki/Big_data)

# Outline

Introduction

**Hadoop**

Spark

R and Big Data

Set Up a Hadoop/Spark Cluster

Online Resources

# Hadoop<sup>2</sup>

- ▶ Apache Hadoop is a framework for running applications on large cluster built of commodity hardware.
- ▶ Hadoop implements a computational paradigm named MapReduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster.
- ▶ Distributed parallel computing
- ▶ Load ballancing
- ▶ Fault tolerant
- ▶ Scales to thousands of nodes



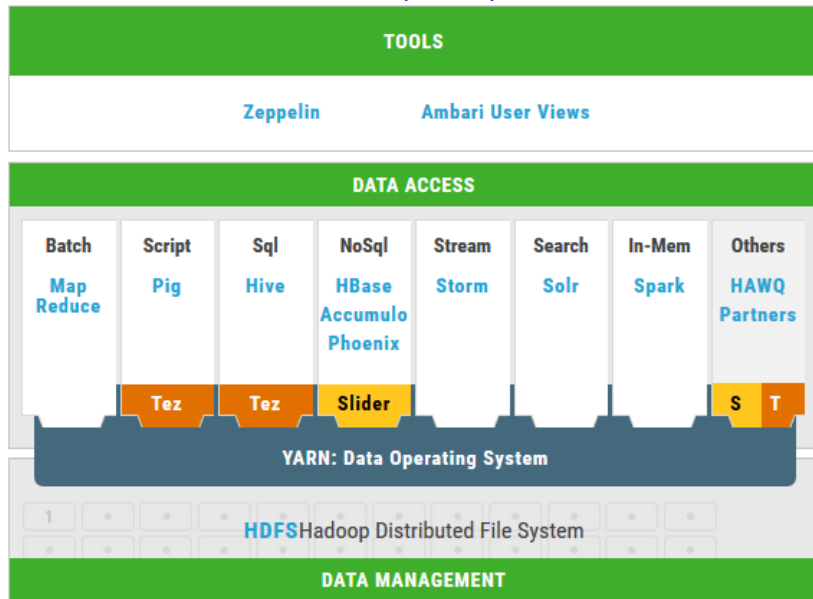
---

<sup>2</sup><http://hadoop.apache.org/>

# Hadoop

- ▶ HDFS: Hadoop Distributed File System
- ▶ YARN: a framework for job scheduling and cluster resource management
- ▶ MapReduce: a framework for parallel processing of large data sets

# Hortonworks Data Platform (HDP) <sup>3</sup>



<sup>3</sup><http://hortonworks.com/products/data-center/hdp/>



# Tools/Applications on Hadoop



- ▶ Pig: a high-level data-flow language and execution framework for parallel computation
- ▶ Hive: a data warehouse infrastructure that provides data summarization and ad hoc querying
- ▶ HBase: the Hadoop database, a distributed, scalable, big data store
- ▶ Cassandra: a scalable multi-master database with no single points of failure
- ▶ Mahout: a scalable machine learning and data mining library

## Tools/Applications on Hadoop (continued)

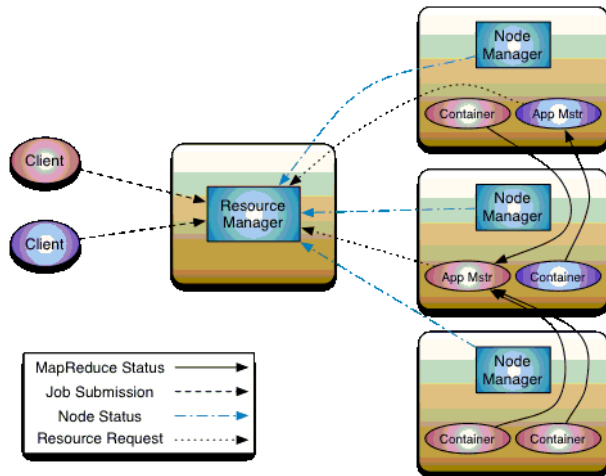
- ▶ Avro: a data serialization system
- ▶ Ambari: a web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters
- ▶ Zeppelin: a web-based notebook that enables interactive data analytics, supporting many interpreters such as Apache Spark, Python, JDBC, Markdown and Shell
- ▶ Tez: a generalized data-flow programming framework, providing a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases
- ▶ Solr: a full-text search and indexing engine that enables large-scale search, navigation, and analytics on textual data
- ▶ Oozie: a tool for Hadoop users to automate commonly performed tasks

# HDFS

- ▶ Hadoop Distributed File System
- ▶ The primary distributed storage used by Hadoop applications
- ▶ Stores very large files across machines in a large cluster
- ▶ NameNode: manages the file system metadata
- ▶ DataNodes: store the actual data
- ▶ A file is chopped into 128MB blocks.
- ▶ Each block is saved in 3 replicas on 3 different DataNodes.

# YARN<sup>4</sup>

- ▶ resource management
- ▶ job scheduling and monitoring

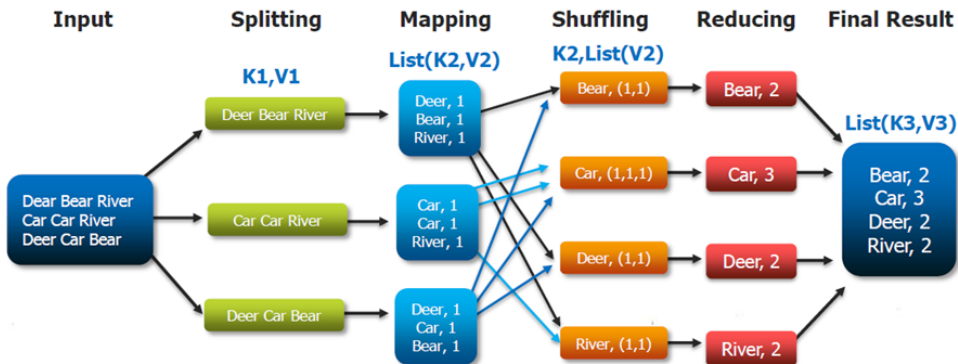


# MapReduce <sup>5</sup>

- ▶ MapReduce expresses a large distributed computation as a sequence of distributed operations on data sets of key-value pairs.
- ▶ A MapReduce computation has two phases, a map phase and a reduce phase.
- ▶ Map: It splits the input data set into a large number of fragments and assigns each fragment to a map task. It also distributes the many map tasks across the cluster. For each input key-value pair  $(K1, V1)$ , the map task invokes a map function that transmutes the input into a different key-value pair  $(K2, V2)$ .
- ▶ Sort/shuffle: sorts the intermediate data set by key and produces a set of  $(K2, \text{list}(V2))$  tuples so that all the values associated with a particular key appear together.
- ▶ Reduce: Each reduce task consumes the fragment of  $(K2, \text{list}(V2))$  tuples assigned to it. For each such tuple it invokes a reduce function that transmutes the tuple into an output key-value pair  $(K3, V3)$ .

# An Example of MapReduce: Word Count <sup>6</sup>

## The Overall MapReduce Word Count Process



# An Example of MapReducing with R<sup>7</sup>

```
library(rmr2)

map <- function(k, lines) {
  words.list <- strsplit(lines, "\\s")
  words <- unlist(words.list)
  return(keyval(words, 1))
}

reduce <- function(word, counts) {
  keyval(word, sum(counts))
}

wordcount <- function(input, output = NULL) {
  mapreduce(input = input, output = output, input.format = "text",
    map = map, reduce = reduce)
}

## Submit job
out <- wordcount(in.file.path, out.file.path)
```

---

<sup>7</sup>From Jeffrey Breen's presentation on *Using R with Hadoop*

# Mahout<sup>8</sup>

Apache Mahout is a suite of machine learning libraries designed to be scalable and robust. It provides 3 major features.

- ▶ A simple and extensible programming environment and framework for building scalable algorithms
- ▶ A wide variety of premade algorithms for Scala + Apache Spark, H2O, Apache Flink
- ▶ Samsara, a vector math experimentation environment with R-like syntax which works at scale



---

<sup>8</sup><https://mahout.apache.org/>



# Machine Learning Algorithms in Mahout

- ▶ Collaborative Filtering
- ▶ Classification
- ▶ Clustering
- ▶ Dimensionality Reduction
- ▶ Topic Models

# Outline

Introduction

Hadoop

Spark

R and Big Data

Set Up a Hadoop/Spark Cluster

Online Resources

# Spark<sup>9</sup>

- ▶ a fast and general-purpose cluster computing system
- ▶ provides high-level APIs in Java, Scala, Python and R
- ▶ Spark SQL for SQL and structured data processing
- ▶ MLlib for large scale machine learning
- ▶ GraphX for graph processing
- ▶ Spark Streaming for processing real-time data streams



---

<sup>9</sup><http://spark.apache.org/>

# Spark Cluster

- ▶ Spark can run both by itself, or over existing cluster managers.
- ▶ Options for deployment:
  - ▶ Standalone Deploy Mode
  - ▶ Apache Mesos
  - ▶ Hadoop YARN

# RDD

- ▶ RDD: Resilient Distributed Datasets, a fault-tolerant collection of elements that can be operated on in parallel.
- ▶ Two ways to create RDDs:
  - ▶ parallelizing an existing collection in your driver program
  - ▶ referencing a dataset in an external storage system
- ▶ RDDs support two types of operations:
  - ▶ transformations: create a new dataset from an existing one
  - ▶ actions: return a value to the driver program after running a computation on the dataset.
- ▶ All transformations are lazy, i.e., they do not actually perform any computations until an action is performed.

# DataFrame

- ▶ A Spark DataFrame is a distributed collection of data organized into named columns.
- ▶ It is conceptually equivalent to a table in a relational database or a data frame in R.
- ▶ supports operations like selection, filtering, grouping, aggregation, etc.

# DataFrame Operations: An Example<sup>10</sup>

```
# 1) select flights from JFK  
# 2) group flights by destination  
# 3) count the number of flights to each destination  
dest_flights <- filter(df, df$origin == "JFK") %>%  
  groupBy(df$dest) %>%  
  summarize(count = n(df$dest))
```

Note: % > % is a pipe operation.

---

<sup>10</sup><https://amplab.cs.berkeley.edu/publication/sparkr-scaling-r-programs-with-spark/>

# MLlib

- ▶ Sparks machine learning (ML) library
- ▶ ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- ▶ Featurization: feature extraction, transformation, dimensionality reduction, and selection
- ▶ Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- ▶ Persistence: saving and load algorithms, models, and Pipelines
- ▶ Utilities: linear algebra, statistics, data handling, etc.



# MLlib Algorithms

- ▶ Linear regression, logistic regression, generalized linear regression
- ▶ Decision tree, random forest
- ▶ Gradient-boosted trees
- ▶ K-means, Latent Dirichlet allocation (LDA), Gaussian Mixture Model (GMM)
- ▶ Collaborative filtering
- ▶ Model selection, cross validation

# MLlib: an Example<sup>11</sup>

## Building a generalized linear model

```
df <- createDataFrame(iris)
glm.model <- spark.glm(df, Sepal_Length ~ Sepal_Width + Species,
  family = "gaussian")
summary(glm.model)
pred <- predict(glm.model, df)
showDF(pred)
```

---

<sup>11</sup><https://spark.apache.org/docs/latest/sparkr.html>

# Outline

Introduction

Hadoop

Spark

R and Big Data

Set Up a Hadoop/Spark Cluster

Online Resources

# R and Big Data Platforms

- ▶ Hadoop

- ▶ Hadoop (or YARN) - a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models
- ▶ R Packages: *RHadoop*, *RHIPE*

- ▶ Spark

- ▶ Spark - a fast and general engine for large-scale data processing, which can be 100 times faster than Hadoop
- ▶ *SparkR* - R frontend for Spark

- ▶ H2O

- ▶ H2O - an open source in-memory prediction engine for big data science
- ▶ R Package: *h2o*

- ▶ MongoDB

- ▶ MongoDB - an open-source document database
- ▶ R packages: *rmongodb*, *RMongo*

# R and Hadoop

- ▶ Packages: *RHadoop*, *RHive*
- ▶ RHadoop<sup>12</sup> is a collection of R packages:
  - ▶ *rhdfs* - connect to Hadoop Distributed File System (HDFS)
  - ▶ *rhbase* - connect to the NoSQL HBase database
  - ▶ *plyrmr* - perform common data manipulation operations on very large data sets stored on Hadoop
  - ▶ *rmr2* - perform data analysis with R via MapReduce on a Hadoop cluster
  - ▶ *ravro* - read and write avro files
- ▶ You can play with it on a single PC (in standalone or pseudo-distributed mode), and your code developed on that will be able to work on a cluster of PCs (in full-distributed mode)!
- ▶ Step-by-Step Guide to Setting Up an R-Hadoop System  
<http://www.rdatamining.com/big-data/r-hadoop-setup-guide>

---

<sup>12</sup><https://github.com/RevolutionAnalytics/RHadoop/wiki>

# R and Spark

- ▶ SparkR
- ▶ sparklyr

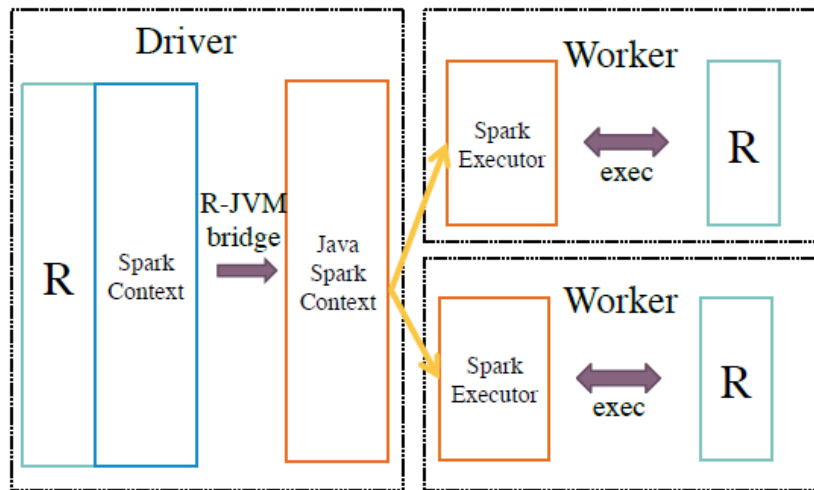
- ▶ SparkR: R on Spark
- ▶ an R package that provides a light-weight frontend to use Apache Spark from R
- ▶ initially developed at the AMPLab, UC Berkeley
- ▶ has been a part of the Apache Spark since v1.4 released in June 2015
- ▶ provides a distributed data frame implementation that supports operations like selection, filtering, aggregation etc. (similar to R data frames, dplyr) but on large datasets.
- ▶ supports distributed machine learning using MLlib.
- ▶ SparkR: Scaling R Programs with Spark. Shivaram Venkataraman et al., In Proc. of SIGMOD'16.<sup>13</sup>

---

<sup>13</sup><https://amplab.cs.berkeley.edu/publication/sparkr-scaling-r-programs-with-spark/>

<sup>14</sup><https://spark.apache.org/docs/latest/sparkr.html>

# SparkR Architecture<sup>15</sup>



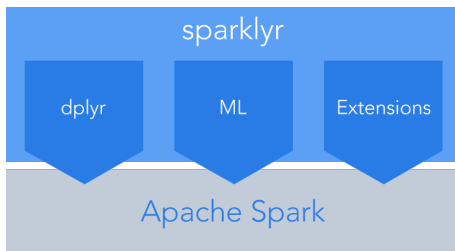


# Machine Learning Algorithms Supported by SparkR

- ▶ Generalized Linear Model
- ▶ Accelerated Failure Time (AFT)
- ▶ Survival Regression Model
- ▶ Naive Bayes Model
- ▶ K-means

## sparklyr<sup>16</sup>

- ▶ sparklyr - an R interface for Apache Spark
- ▶ Provide a complete dplyr backend for data manipulation
- ▶ Filter and aggregate Spark datasets then bring them into R for analysis and visualization
- ▶ Distributed machine learning from R: using Spark MLlib or H2O Sparkling Water
- ▶ Create extensions that call the full Spark API and provide interfaces to Spark packages.



<sup>16</sup><http://spark.rstudio.com/>

# MLlib Algorithms

- ▶ `ml_kmeans`: K-means Clustering
- ▶ `ml_linear_regression`: Linear Regression
- ▶ `ml_logistic_regression`: Logistic Regression
- ▶ `ml_survival_regression`: Survival Regression
- ▶ `ml_generalized_linear_regression`: Generalized Linear Regression
- ▶ `ml_decision_tree`: Decision Trees
- ▶ `ml_random_forest`: Random Forests
- ▶ `ml_gradient_boosted_trees`: Gradient-Boosted Trees
- ▶ `ml_pca`: Principal Components Analysis
- ▶ `ml_naive_bayes`: Naive-Bayes
- ▶ `ml_multilayer_perceptron`: Multilayer Perceptron
- ▶ `ml_lda`: Latent Dirichlet Allocation
- ▶ `ml_one_vs_rest`: One vs Rest

# H2O Machine Learning Algorithms

- ▶ `h2o.glm`: Generalized Linear Model
- ▶ `h2o.deeplearning`: Multilayer Perceptron
- ▶ `h2o.randomForest`: Random Forest
- ▶ `h2o.gbm`: Gradient Boosting Machine
- ▶ `h2o.naiveBayes`: Naive Bayes
- ▶ `h2o.prcomp`: Principal Components Analysis
- ▶ `h2o.svd`: Singular Value Decomposition
- ▶ `h2o.glrn`: Generalized Low Rank Model
- ▶ `h2o.kmeans`: K-Means Clustering
- ▶ `h2o.anomaly`: Anomaly Detection
- ▶ `h2o.ensemble`, `h2ostack`: Ensemble/stacking

# Outline

Introduction

Hadoop

Spark

R and Big Data

Set Up a Hadoop/Spark Cluster

Online Resources

# Set up a Hadoop/Spark Cluster

- ▶ On-premise cluster
  - ▶ Apache
  - ▶ Hortonworks
  - ▶ MapR
  - ▶ Cloudera:  
<http://www.cloudera.com/>
- ▶ Cloud solutions
  - ▶ Amazon Web Services (AWS):  
<https://aws.amazon.com/>
  - ▶ Microsoft Azure:  
<http://azure.microsoft.com>
  - ▶ Google Cloud Platform:  
<https://cloud.google.com/hadoop/>

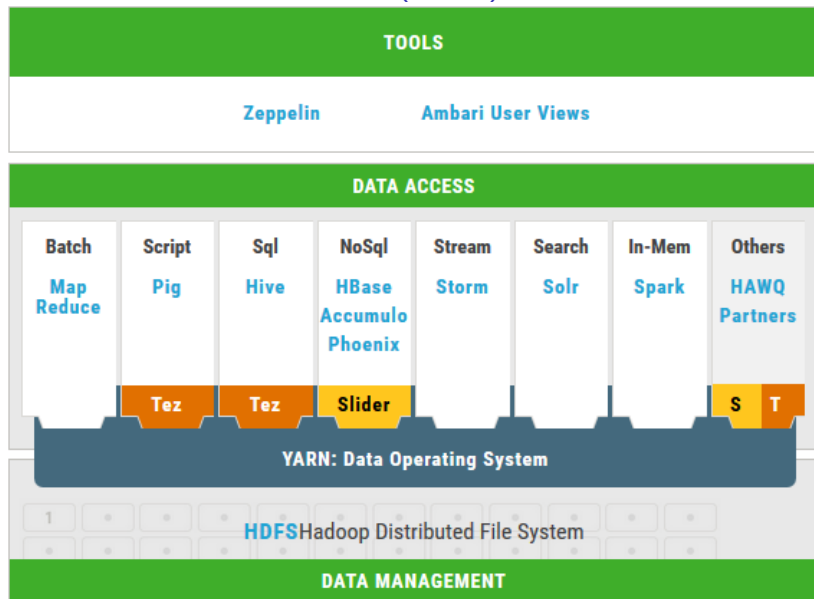
Download Hadoop and Spark from Apache.org and install them

- ▶ Hadoop

`http://hadoop.apache.org/releases.html`

- ▶ Spark

`http://spark.apache.org/downloads.html`

Hortonworks Data Platform (HDP) <sup>17</sup>

<sup>17</sup><http://hortonworks.com/products/data-center/hdp/>



# Hortonworks Sandbox

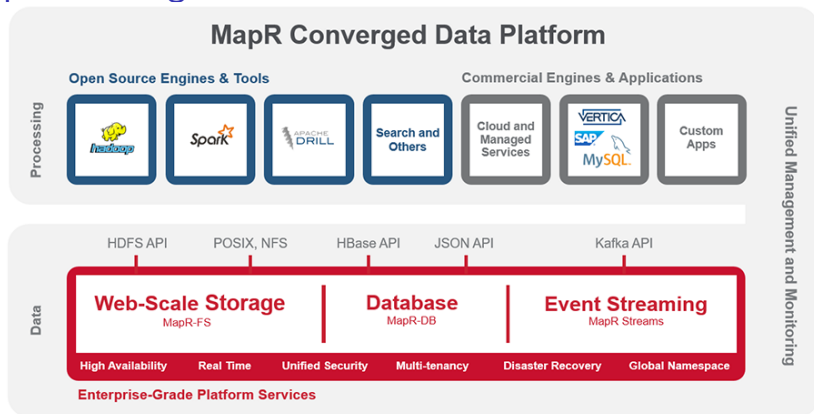
- ▶ A personal, portable Apache Hadoop and its ecosystem environment
- ▶ On a virtual machine: Virtual Box, VMware, Docker
- ▶ On cloud: Microsoft Azure
- ▶ Good for learning Hadoop, Spark, Pig, Hive, etc.
- ▶ Download for free: <http://hortonworks.com/downloads/>

# Hortonworks Sandbox

- ▶ A personal, portable Apache Hadoop and its ecosystem environment
- ▶ On a virtual machine: Virtual Box, VMware, Docker
- ▶ On cloud: Microsoft Azure
- ▶ Good for learning Hadoop, Spark, Pig, Hive, etc.
- ▶ Download for free: <http://hortonworks.com/downloads/>

To set up a cluster, use Hortonworks Data Platform, not Sandbox.

# MapR Converged Data Platform <sup>18</sup>



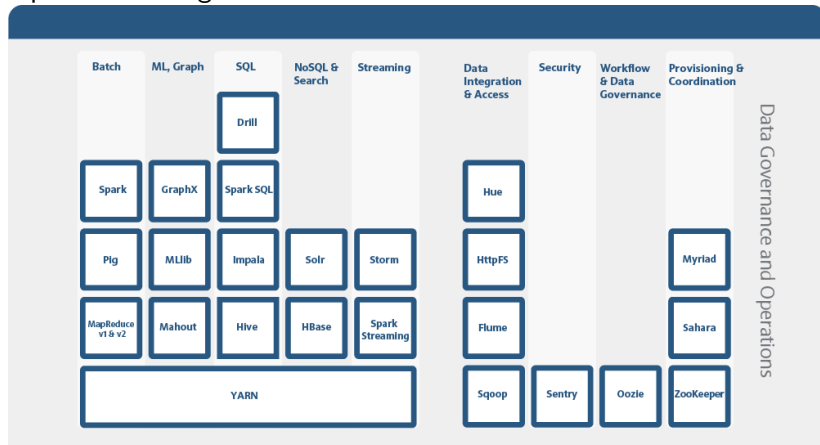
- ▶ It integrates Hadoop, Spark, and Apache Drill with real-time database capabilities, global event streaming, and scalable enterprise storage.
- ▶ Free community edition:

<https://www.mapr.com/products/hadoop-download>

<sup>18</sup><https://www.mapr.com/products/mapr-converged-data-platform>

# MapR Converged Data Platform <sup>19</sup>

## Open source engines and tools



<sup>19</sup> <https://www.mapr.com/products/mapr-converged-data-platform>

# Outline

Introduction

Hadoop

Spark

R and Big Data

Set Up a Hadoop/Spark Cluster

Online Resources

# Online Resources

- ▶ RDataMining Reference Card

<http://www.rdatamining.com/docs/R-refcard-data-mining.pdf>

- ▶ Free online courses and documents

<http://www.rdatamining.com/resources/>

<http://www.rdatamining.com/big-data/resources/>

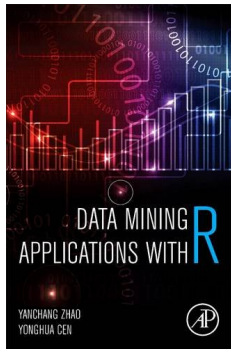
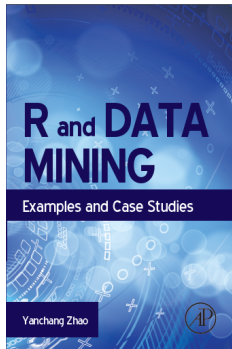
- ▶ RDataMining Group on LinkedIn (22,000+ members)

<http://group.rdatamining.com>

- ▶ Twitter (2,700+ followers)

@RDataMining

# The End



Thanks!

Email: [yanchang\(at\)RDataMining.com](mailto:yanchang(at)RDataMining.com)

Twitter: @RDataMining