Big Data Mining

Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem

Min-Yuh Day, Ph.D.
Assistant Professor
Department of Information Management
Tamkang University

http://mail.tku.edu.tw/myday

2018-10-08
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2018/09/10</td>
<td>Course Orientation for Big Data Mining</td>
</tr>
<tr>
<td>2</td>
<td>2018/09/17</td>
<td>ABC: AI, Big Data, Cloud Computing</td>
</tr>
<tr>
<td>3</td>
<td>2018/09/24</td>
<td>Mid-Autumn Festival (Day off)</td>
</tr>
<tr>
<td>4</td>
<td>2018/10/01</td>
<td>Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data</td>
</tr>
<tr>
<td>5</td>
<td>2018/10/08</td>
<td>Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem</td>
</tr>
<tr>
<td>6</td>
<td>2018/10/15</td>
<td>Foundations of Big Data Mining in Python</td>
</tr>
<tr>
<td>7</td>
<td>2018/10/22</td>
<td>Supervised Learning: Classification and Prediction</td>
</tr>
<tr>
<td>8</td>
<td>2018/10/29</td>
<td>Unsupervised Learning: Cluster Analysis</td>
</tr>
<tr>
<td>9</td>
<td>2018/11/05</td>
<td>Unsupervised Learning: Association Analysis</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>2018/11/12</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>11</td>
<td>2018/11/19</td>
<td>Machine Learning with Scikit-Learn in Python</td>
</tr>
<tr>
<td>12</td>
<td>2018/11/26</td>
<td>Deep Learning for Finance Big Data with TensorFlow</td>
</tr>
<tr>
<td>13</td>
<td>2018/12/03</td>
<td>Convolutional Neural Networks (CNN)</td>
</tr>
<tr>
<td>14</td>
<td>2018/12/10</td>
<td>Recurrent Neural Networks (RNN)</td>
</tr>
<tr>
<td>15</td>
<td>2018/12/17</td>
<td>Reinforcement Learning (RL)</td>
</tr>
<tr>
<td>16</td>
<td>2018/12/24</td>
<td>Social Network Analysis (SNA)</td>
</tr>
<tr>
<td>17</td>
<td>2018/12/31</td>
<td>Bridge Holiday (Extra Day Off)</td>
</tr>
<tr>
<td>18</td>
<td>2019/01/07</td>
<td>Final Project Presentation</td>
</tr>
</tbody>
</table>
Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem
MapReduce Paradigm
MapReduce Paradigm

Big Data

Map

Map0 → Reduce0
Map1 → Reduce1
Map2 → Reduce2
Map3 → Reduce3

MapReduce Data

Reduce

Reduce0 → MapReduce Data
Reduce1 → MapReduce Data
Reduce2 → MapReduce Data
Reduce3 → MapReduce Data

Output Data
MapReduce Word Count

Input

Dog Love Cat
Bird Love Bird
Dog Bird Cat
MapReduce Word Count

Input

Dog Love Cat
Bird Love Bird
Dog Bird Cat

Output

Bird, 3
Cat, 2
Dog, 2
Love, 2

Source: https://www.edureka.co/blog/mapreduce-tutorial/
MapReduce Word Count

Input

Split

Map

Shuffle

Reduce

Output

Dog Love Cat
Bird Love Bird
Dog Bird Cat

Dog Love Cat
Bird Love Bird
Dog Bird Cat

Dog, 1
Love, 1
Cat, 1

Bird, 1
Love, 1
Cat, 1

Bird, (1, 1, 1)

Bird, (1, 1, 1)

Cat, (1, 1)

Cat, (1, 1)

Cat, 2

Cat, 2

Bird, 3

Bird, 3

Bird, 3

Bird, 3

Bird, 3

Love, (1, 1)

Love, (1, 1)

Love, 2

Love, 2

Love, 2

Love, 2

Source: https://www.edureka.co/blog/mapreduce-tutorial/
Hadoop Ecosystem
The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

Source: http://hadoop.apache.org/
MapReduce + HDFS → Processing + Storage

Source: http://hadoop.apache.org/
Big Data with Hadoop Architecture

LOGICAL ARCHITECTURE

Processing: MapReduce
- Job Tracker
- Task Tracker
- Task Tracker
- Task Tracker
- Mapper
- Mapper
- Mapper
- Shuffle and Sort
- Reducer
- Reducer
- Reducer

Storage: HDFS
- NameNode
- Data Node
- Data Node
- Data Node

PROCESS FLOW

Input Data Set
- Split 0
- Map 0
- Reduce 0
- Reduce 0
- Map 1
- Reduce 0
- Reduce 0
- Map n
- Reduce 0
- Reduce 0

PHYSICAL ARCHITECTURE

Hadoop Cluster
- Master
- Slave
- Slave
- Slave
- Slave
- Slave
- Slave
- Slave
- Slave

Big Data with Hadoop Architecture

Logical Architecture

Processing: MapReduce

Big Data with Hadoop Architecture

Logical Architecture

Storage: HDFS

Big Data with Hadoop Architecture

Process Flow

Input Data Set

Split 0 → Map 0 → Reduce 0

Split 1 → Map 1 → Reduce 0

Split n → Map n → Reduce 0

Big Data with Hadoop Architecture

Hadoop Cluster

Hadoop Ecosystem

Source: Shiva Achari (2015), Hadoop Essentials - Tackling the Challenges of Big Data with Hadoop, Packt Publishing
HDP (Hortonworks Data Platform)
A Complete Enterprise Hadoop Data Platform

Source: http://hortonworks.com/hdp/
Apache Hadoop
Hortonworks Data Platform

Source: http://hortonworks.com/hdp/
Hadoop and Data Analytics Tools

Source: http://hortonworks.com/hdp/
Hadoop 1  →  Hadoop 2

Hadoop 1
- Silos & Largely batch
- Single Processing engine

Hadoop 2 with Tez
- Multiple Engines, Single Data Set
- Batch, Interactive & Real-Time

MapReduce
(Cluster Resource Management & Data Processing)

HDFS
(Hadoop Distributed File System)

YARN: Data Operating System
(Cluster Resource Management)

Script
Pig
Hive
Java
Cascading
Engines
HBase
Accumulo,
Storm,
Solr,
Spark.

SQL
Hive
Tez

HBase
Others
Tez

HDFS
(Hadoop Distributed File System)

Source: http://hortonworks.com/hadoop/tez/
Big Data Solution

Traditional ETL Architecture

Offload ETL with Hadoop (Big Data Architecture)

Spark Ecosystem
Apache Spark is a fast and general engine for large-scale data processing.

Source: http://spark.apache.org/
Logistic regression in Hadoop and Spark

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Source: http://spark.apache.org/
Ease of Use

• Write applications quickly in Java, Scala, Python, R.

Source: http://spark.apache.org/
Word count in Spark's Python API

text_file = spark.textFile("hdfs://...")

text_file.flatMap(lambda line: line.split())
  .map(lambda word: (word, 1))
  .reduceByKey(lambda a, b: a+b)

Source: http://spark.apache.org/
Spark and Hadoop

Source: http://spark.apache.org/
Spark Ecosystem

Spark SQL + DataFrames
Streaming
MLlib Machine Learning
GraphX Graph Computation

Spark Core API
R
SQL
Python
Scala
Java

Source: [https://databricks.com/spark/about](https://databricks.com/spark/about)
Spark Ecosystem

Spark

Spark Streaming
- Kafka
- Flume

MLlib (machine learning)
- H2O

Spark SQL
- Hive

GraphX (graph)
- Titan
  - HBase
  - Cassandra

HDFS

Source: Mike Frampton (2015), Mastering Apache Spark, Packt Publishing
SMACK Stack

- **Spark**
  - fast and general engine for distributed, large-scale data processing

- **Mesos**
  - cluster resource management system that provides efficient resource isolation and sharing across distributed applications

- **Akka**
  - a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM

- **Cassandra**
  - distributed, highly available database designed to handle large amounts of data across multiple datacenters

- **Kafka**
  - a high-throughput, low-latency distributed messaging system designed for handling real-time data feeds

Source: Anton Kirillov (2015), Data processing platforms architectures with Spark, Mesos, Akka, Cassandra and Kafka, Big Data AW Meetup
Hadoop vs. Spark

Source: Shiva Achari (2015), Hadoop Essentials - Tackling the Challenges of Big Data with Hadoop, Packt Publishing
Hadoop Distribution

- Apache Hadoop
- Amazon Elastic Map Reduce (EMR)
  - https://aws.amazon.com/emr/
- Cloudera CDH
- Hortonworks Sandbox
  - https://hortonworks.com/products/sandbox/
Steps to Install Hadoop on a Personal Computer (Windows/OS X)

Source: https://www.youtube.com/watch?v=rO-V1mxhzcM&list=PLyZEf-TOnZen8E5m5TIpIsdok2fyKDNRa&index=5
Hadoop: Linux Based Software

Source: https://www.youtube.com/watch?v=r0-V1mhzcm&list=PLyZEF-TOnZen8E5m5Tlpsdok2fyKDNRa&index=5
Appliance

Personal Computer (Windows / OS X)

Virtual Machine (VirtualBox / VMWare)

Linux

Hadoop

Source: https://www.youtube.com/watch?v=rO-V1mzhzcM&list=PLyZEf-TOnZen8E5m5TIplstdk2fyKDNRa&index=5
Connection to Hadoop

Personal Computer (Windows / OS X)

Virtual Machine (VirtualBox / VMWare)

Linux

Hadoop

Browser

Access from host

Source: https://www.youtube.com/watch?v=rO-V1mhzC/M&list=PLyZEF-TOonZen8E5mSTIpl/sdok2fyKDNRa&index=5
Steps to Install Hadoop on a Personal Computer (Windows/OS X)

Step 1. Download and Install VirtualBox

Step 2. Download Appliance

Step 3. Import Appliance

Step 4. Configure Virtual Machine (VM)

Step 5. Start Virtual Machine (VM)

Step 6. Test Connection From Host

Source: https://www.youtube.com/watch?v=rO-V1mhzcm&list=PLyZEF-TOnZen8E5m5TlpIsdok2fyKDNRA&index=5
Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

Download VirtualBox 5.1

Hot picks:
- Pre-built virtual machines for developers at Oracle Tech Network
- Hyperbox Open-source Virtual Infrastructure Manager project site
- phpVirtualBox AJAX web interface project site
- IQEmu automated Windows VM creation, application integration http://mirage335-site.member.hacdc.org:6380/wiki/Category:IQEmu

https://www.virtualbox.org/
Steps to Install Hadoop on a Personal Computer (Windows/OS X)

1. Download and Install VirtualBox
2. Download Appliance
3. Import Appliance
4. Configure Virtual Machine (VM)
5. Start Virtual Machine (VM)
6. Test Connection From Host

Source: https://www.youtube.com/watch?v=rO-V1mhzcm&list=PLyZEf-TOnZen8E5m5TIplsdok2fyKDNRa&index=5
Get started on Hadoop with these tutorials based on the Hortonworks Sandbox

TUTORIALS

Get started on Hadoop with these tutorials based on the Hortonworks Sandbox

DEVELOP WITH HADOOP

Start developing with Hadoop. These tutorials are designed to ease your way into developing with Hadoop:

Apache Spark on HDP

Hands-on Tour of Apache Spark in 5 Minutes
Apache Spark is a fast, in-memory data processing engine with elegant and expressive development APIs in Scala, Java, Python, and R that allow data workers to efficiently execute machine learning algorithms that require fast iterative access to datasets (see Spark API Documentation for more info). Spark on Apache Hadoop YARN enables deep integration with Hadoop […]

http://hortonworks.com/tutorials/
Welcome to Apache™ Hadoop®!

What Is Apache Hadoop?
The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The project includes these modules:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Other Hadoop-related projects at Apache include:

- **Ambavi™**: A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop. Ambavi also provides a dashboard for viewing cluster health such as heatmaps and ability to view MapReduce, Pig and Hive applications visually along with features to diagnose their performance characteristics in a user-friendly manner.
- **Avro™**: A data serialization system.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa™**: A data collection system for managing large distributed systems.
- **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
- **Hive™**: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- **Mahout™**: A Scalable machine learning and data mining library.
- **Pig™**: A high-level data-flow language and execution framework for parallel computation.

Apache Hadoop

http://hadoop.apache.org/releases.html#Download

Apache Hadoop Releases

Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA-256.

<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Tarball</th>
<th>GPG</th>
<th>SHA-256</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0.0-alpha2</td>
<td>25 January, 2017</td>
<td>source</td>
<td>signature</td>
<td>checksum file</td>
</tr>
<tr>
<td>3.0.0-alpha1</td>
<td>03 September, 2016</td>
<td>binary</td>
<td>signature</td>
<td>checksum file</td>
</tr>
<tr>
<td>2.7.3</td>
<td>25 August, 2016</td>
<td>source</td>
<td>signature</td>
<td>checksum file</td>
</tr>
<tr>
<td>2.6.5</td>
<td>08 October, 2016</td>
<td>binary</td>
<td>signature</td>
<td>227785DC 6E3F6EF8..</td>
</tr>
<tr>
<td>2.5.2</td>
<td>19 Nov, 2014</td>
<td>source</td>
<td>signature</td>
<td>3A843F18 73D9951A..</td>
</tr>
<tr>
<td></td>
<td></td>
<td>binary</td>
<td>signature</td>
<td>001AD18D 4B6D0FE5..</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>139EF872 09C5637E..</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0BD84850 A3825208..</td>
</tr>
</tbody>
</table>

To verify Hadoop releases using GPG:

1. Download the release hadoop-X.Y.Z-src.tar.gz from a mirror site.
2. Download the signature file hadoop-X.Y.Z-src.tar.gz.asc from Apache.
3. Download the Hadoop KEYS file.
4. gpg --import KEYS
5. gpg --verify hadoop-X.Y.Z-src.tar.gz.asc

To perform a quick check using SHA-256:

1. Download the release hadoop-X.Y.Z-src.tar.gz from a mirror site.
2. Download the checksum hadoop-X.Y.Z-src.tar.gz.mds from Apache.
Apache Hadoop YARN

Source: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
Apache Spark™ is a unified analytics engine for large-scale data processing.

**Speed**
Run workloads 100x faster.

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.

**Ease of Use**
Write applications quickly in Java, Scala, Python, R, and SQL.

Spark offers over 80 high-level operators that make it easy to build parallel

```
df = spark.read.json("logs.json")
df.where("age > 21")
  .select("name.first").show()
```

Spark’s Python DataFrame API
Read JSON files with automatic schema inference

**Built-in Libraries:**
- SQL and DataFrames
- Spark Streaming
- MLlib (machine learning)
- GraphX (graph)

Apache Spark™ is a powerful open source processing engine built around speed, ease of use, and sophisticated analytics. It was originally developed at UC Berkeley in 2009.

The largest open source project in data processing.

Since its release, Spark has seen rapid adoption by enterprises across a wide range of industries. Internet powerhouses such as Yahoo, Baidu, and Tencent, have eagerly deployed Spark at massive scale, collectively processing multiple petabytes of data on clusters of over 8,000 nodes. It has quickly become the largest open source community in big data, with over 750 contributors from 200+ organizations.

The creators of Spark founded Databricks in 2013.

Databricks continues to grow the Spark project by contributing broadly, with both roadmap development and community evangelism.

“At Databricks, we’re working hard to make Spark easier to use and run than ever, through our efforts on both the Spark codebase and support materials around it. All of our work on Spark is open source and goes directly to Apache.”

— Matei Zaharia, VP, Apache Spark, Founder & CTO, Databricks

https://databricks.com/spark/about
The Spark Platform

Spark SQL
Spark Streaming
MLlib
GraphX

Spark Core

Spark Standalone
Hadoop YARN
Mesos

Source: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-overview.html
Spark Cluster Overview

Source: http://spark.apache.org/docs/latest/cluster-overview.html
3 Types of Spark Cluster Manager

• **Standalone**
  – a simple cluster manager included with Spark that makes it easy to set up a cluster.

• **Apache Mesos**
  – a general cluster manager that can also run Hadoop MapReduce and service applications.

• **Hadoop YARN**
  – the resource manager in Hadoop 2.

Spark’s EC2 launch scripts make it easy to launch a standalone cluster on Amazon EC2.

Source: http://spark.apache.org/docs/latest/cluster-overview.html
Running Spark on cluster
Cluster Managers (Schedulers)

• Spark’s own Standalone cluster manager
• Hadoop YARN
• Apache Mesos

Source: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-cluster.html
Spark Developer Resources

Databricks provides a number of free resources online for Spark training, including course materials, video archives, sample apps, knowledge base, etc.

Note that course materials for these workshops are provided online under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) license.

Content Archives
Highlights of recent blogs, videos, and other community content:
- Apache Spark channel on YouTube (meetup livestream archives)
- Spark events worldwide
- Spark Packages
- O'Reilly Radar – Spark articles

Spark ODBC Driver Download
Spark's ODBC driver lets you connect Business Intelligence (BI) and other third party applications to the Spark SQL server.
GET IT HERE

https://databricks.com/spark/developer-resources
SparkContext and RDDs
Resilient Distributed Dataset (RDD)
Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica
University of California, Berkeley

Abstract

We present Resilient Distributed Datasets (RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner. RDDs are motivated by two types of applications that current computing frameworks handle inefficiently: iterative algorithms and interactive data mining tools. In both cases, keeping data in memory can improve performance by an order of magnitude. To achieve fault tolerance efficiently, RDDs provide a restricted form of shared memory, based on coarse-grained transformations rather than fine-grained updates to shared state. However, we show that RDDs are expressive enough to capture a wide class of computations, including recent specialized programming models for iterative jobs, such as Pregel, and new applications that these models do not capture. We have implemented RDDs in a system called Spark, which we evaluate through a variety of user applications and benchmarks.

1 Introduction

Cluster computing frameworks like MapReduce [10] and Dryad [19] have been widely adopted for large-scale data analytics. These systems let users write parallel computation, which can dominate application execution times.

Recognizing this problem, researchers have developed specialized frameworks for some applications that require data reuse. For example, Pregel [22] is a system for iterative graph computations that keeps intermediate data in memory, while HaLoop [7] offers an iterative MapReduce interface. However, these frameworks only support specific computation patterns (e.g., looping a series of MapReduce steps), and perform data sharing implicitly for these patterns. They do not provide abstractions for more general reuse, e.g., to let a user load several datasets into memory and run ad-hoc queries across them.

In this paper, we propose a new abstraction called resilient distributed datasets (RDDs) that enables efficient data reuse in a broad range of applications. RDDs are fault-tolerant, parallel data structures that let users explicitly persist intermediate results in memory, control their partitioning to optimize data placement, and manipulate them using a rich set of operators.

The main challenge in designing RDDs is defining a programming interface that can provide fault tolerance efficiently. Existing abstractions for in-memory storage on clusters, such as distributed shared memory [24], key-value stores [25], databases, and Piccolo [27], offer an
RDD

• **Resilient**
  
  — fault-tolerant and so able to recompute missing or damaged partitions on node failures with the help of **RDD lineage graph**.

• **Distributed** across clusters.

• **Dataset**
  
  — a collection of **partitioned data**.

Source: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd.html
Resilient Distributed Datasets (RDD) are the primary abstraction in Spark – a fault-tolerant collection of elements that can be operated on in parallel.

Source: Databricks (2016), Intro to Apache Spark
Resilient Distributed Dataset (RDD)

Spark’s “Interface” to data
Spark RDD

distributed and partitioned RDD

Source: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd.html
RDD Lineage Graph
(RDD operator graph)

Source: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd.html
Resilient Distributed Dataset (RDD) Operations

Transformations

Actions

Source: Liondatasystems (2015), Intro to Apache Spark (Brain-Friendly Tutorial)
Spark RDD Operations

```scala
val file = sc.textFile("README.md")
val allWords = file.flatMap(_.split("\s+"))
val words = allWords.filter(_非Empty)
val pairs = words.map(_._1)
val reducedByKey = pairs.reduceByKey(_ + _)
val top10words = reducedByKey.takeOrdered(10)(Ordering[Int].reverse.on(_._2))
```
Get Started using Apache Spark on a Personal Computer (Windows/Mac OS X)

1. Install Java JDK
2. Download Spark
3. Run Spark Shell
4. Create RDD

Source: Databricks (2016), Intro to Apache Spark
Apache Spark™ is a unified analytics engine for large-scale data processing.

### Speed
Run workloads 100x faster.

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.

![Logistic regression in Hadoop and Spark](image)

### Ease of Use
Write applications quickly in Java, Scala, Python, R, and SQL.

Spark offers over 80 high-level operators that make it easy to build parallel

```python
df = spark.read.json("logs.json")
df.where("age > 21")
  .select("name.first").show()
```

Spark's Python DataFrame API
Read JSON files with automatic schema inference

**Built-in Libraries:**
- SQL and DataFrames
- Spark Streaming
- MLlib (machine learning)
- GraphX (graph)
spark download

download apache spark™

1. choose a spark release: 2.3.2 (sep 24 2018)
2. choose a package type: pre-built for apache hadoop 2.7 and later
3. download spark: spark-2.3.2-bin-hadoop2.7.tgz
4. verify this release using the 2.3.2 signatures and checksums and project release keys.

note: starting version 2.0, spark is built with scala 2.11 by default. scala 2.10 users should download the spark source package and build with scala 2.10 support.

link with spark

spark artifacts are hosted in maven central. you can add a maven dependency with the following coordinates:

groupId: org.apache.spark
artifactId: spark-core_2.11
version: 2.3.2

installing with pypi

pyspark is now available in pypi. to install just run pip install pyspark.
Get Started using Apache Spark on a Personal Computer (Windows/Mac OS X)

Step 1. Install Java JDK

Step 2. Download Spark

Step 3. Run Spark Shell

Spark's interactive shell
./bin/spark-shell
val data = 1 to 10000

Step 4. Create RDD

Source: Databricks (2016), Intro to Apache Spark
Get Started using Apache Spark on a Personal Computer (Windows/Mac OS X)

Step 1. Install Java JDK

Step 2. Download Spark

Step 3. Run Spark Shell

Step 4. Create RDD

val distData = sc.parallelize(data)
distData.filter(_ < 10).collect()
Word Count

Hello World

in Big Data
Spark Examples

Spark Examples

These examples give a quick overview of the Spark API. Spark is built on the concept of distributed datasets, which contain arbitrary Java or Python objects. You create a dataset from external data, then apply parallel operations to it. The building block of the Spark API is its RDD API. In the RDD API, there are two types of operations: transformations, which define a new dataset based on previous ones, and actions, which kick off a job to execute on a cluster. On top of Spark's RDD API, high level APIs are provided, e.g. DataFrame API and Machine Learning API. These high level APIs provide a concise way to conduct certain data operations. In this page, we will show examples using RDD API as well as examples using high level APIs.

RDD API Examples

Word Count

In this example, we use a few transformations to build a dataset of (String, Int) pairs called counts and then save it to a file.

Python

```python
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" "))
  .map(lambda word: (word, 1))
  .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

Scala

```scala
val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
  .map(word => (word, 1))
  .reduceByKey((a, b) => a + b)
counts.saveAsTextFile("hdfs://...")
```

Java

```java
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Encoder;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.dataframe.
import org.apache.spark.sql.functions.
import org.apache.spark.sql.types.
import org.apache.spark.sql.window.

SparkConf conf = new SparkConf().setAppName("WordCount").
  .setMaster("local[4]");
JavaSparkContext jsc = new JavaSparkContext(conf);

val textFile = jsc.textFile("hdfs://...");
val counts = textFile.flatMap(line => line.split(" "))
  .map(word => (word, 1))
  .reduceByKey((a, b) => a + b)
counts.saveAsTextFile("hdfs://...");
```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" "))
  .map(lambda word: (word, 1))
  .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")

Source: http://spark.apache.org/examples.html
val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
  .map(word => (word, 1))
  .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
JavaRDD<String> textFile = sc.textFile("hdfs://...");
JavaRDD<String> words = textFile.flatMap(new FlatMapFunction<String, String>() {
    public Iterable<String> call(String s) { return Arrays.asList(s.split(" ")); }
}).
JavaPairRDD<String, Integer> pairs = words.mapToPair(new PairFunction<String, String, Integer>() {
    public Tuple2<String, Integer> call(String s) { return new Tuple2<String, Integer>(s, 1); }
}).
JavaPairRDD<String, Integer> counts = pairs.reduceByKey(new Function2<Integer, Integer, Integer>() {
    public Integer call(Integer a, Integer b) { return a + b; }
}).
counts.saveAsTextFile("hdfs://...");

Source: http://spark.apache.org/examples.html
Spark Word Count

Python

text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) 
    .map(lambda word: (word, 1)) 
    .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")

Scala

val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
    .map(word => (word, 1))
    .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")

Java

JavaRDD<String> textFile = sc.textFile("hdfs://...");
JavaRDD<String> words = textFile.flatMap(new FlatMapFunction<String, String>() {
    public Iterable<String> call(String s) { return Arrays.asList(s.split(" ")); }
});
JavaPairRDD<String, Integer> pairs = words.mapToPair(new PairFunction<String, String, String, Integer>() {
    public Tuple2<String, Integer> call(String s) { return new Tuple2<String, Integer>(s, 1); }
});
JavaPairRDD<String, Integer> counts = pairs.reduceByKey(new Function2<Integer, Integer, Integer>() {
    public Integer call(Integer a, Integer b) { return a + b; }
});
counts.saveAsTextFile("hdfs://...");

Source: http://spark.apache.org/examples.html
text_file = sc.textFile("input_readme.txt")
counts = text_file.flatMap(lambda line: line.split(" "))
   .map(lambda word: (word, 1))
   .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("output_wordcount.txt")

Source: http://spark.apache.org/examples.html
WordCount Example

- Input
  
  Hello World Bye World
  Hello Hadoop Goodbye Hadoop

- For the given sample input the map emits
  
  < Hello, 1>
  < World, 1>
  < Bye, 1>
  < World, 1>
  < Hello, 1>
  < Hadoop, 1>
  < Goodbye, 1>
  < Hadoop, 1>

- < Bye, 1>
  < Goodbye, 1>
  < Hadoop, 2>
  < Hello, 2>
  < World, 2>
Object-Oriented Meets Functional

Have the best of both worlds. Construct elegant class hierarchies for maximum code reuse and extensibility, implement their behavior using higher-order functions. Or anything in-between.

LEARN MORE

http://www.scala-lang.org/
Python

Compound Data Types

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. More about lists in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. Learn More

https://www.python.org/
PySpark: Spark Python API
Welcome to Spark Python API Docs!

Contents:

- pyspark package
  - Subpackages
  - Contents
- pyspark.sql module
  - Module Context
  - pyspark.sql.types module
  - pyspark.sql.functions module
  - pyspark.sql.streaming module
- pyspark.streaming module
  - Module contents
  - pyspark.streaming.kafka module
  - pyspark.streaming.kinesis module
  - pyspark.streaming.flume.module
- pyspark.ml package
  - ML Pipeline APIs
  - pyspark.ml.param module
  - pyspark.ml.feature module
  - pyspark.ml.classification module
  - pyspark.ml.clustering module
  - pyspark.ml.linalg module
  - pyspark.ml.recommendation module
  - pyspark.ml.regression module
  - pyspark.ml.stat module
  - pyspark.ml.tuning module

http://spark.apache.org/docs/latest/api/python/
Big Data & AI Landscape 2018

Source: http://mattturck.com/bigdata2018/
References

• EMC Education Services (2015), Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data, Wiley

• Shiva Achari (2015), Hadoop Essentials - Tackling the Challenges of Big Data with Hadoop, Packt Publishing

• Mike Frampton (2015), Mastering Apache Spark, Packt Publishing
