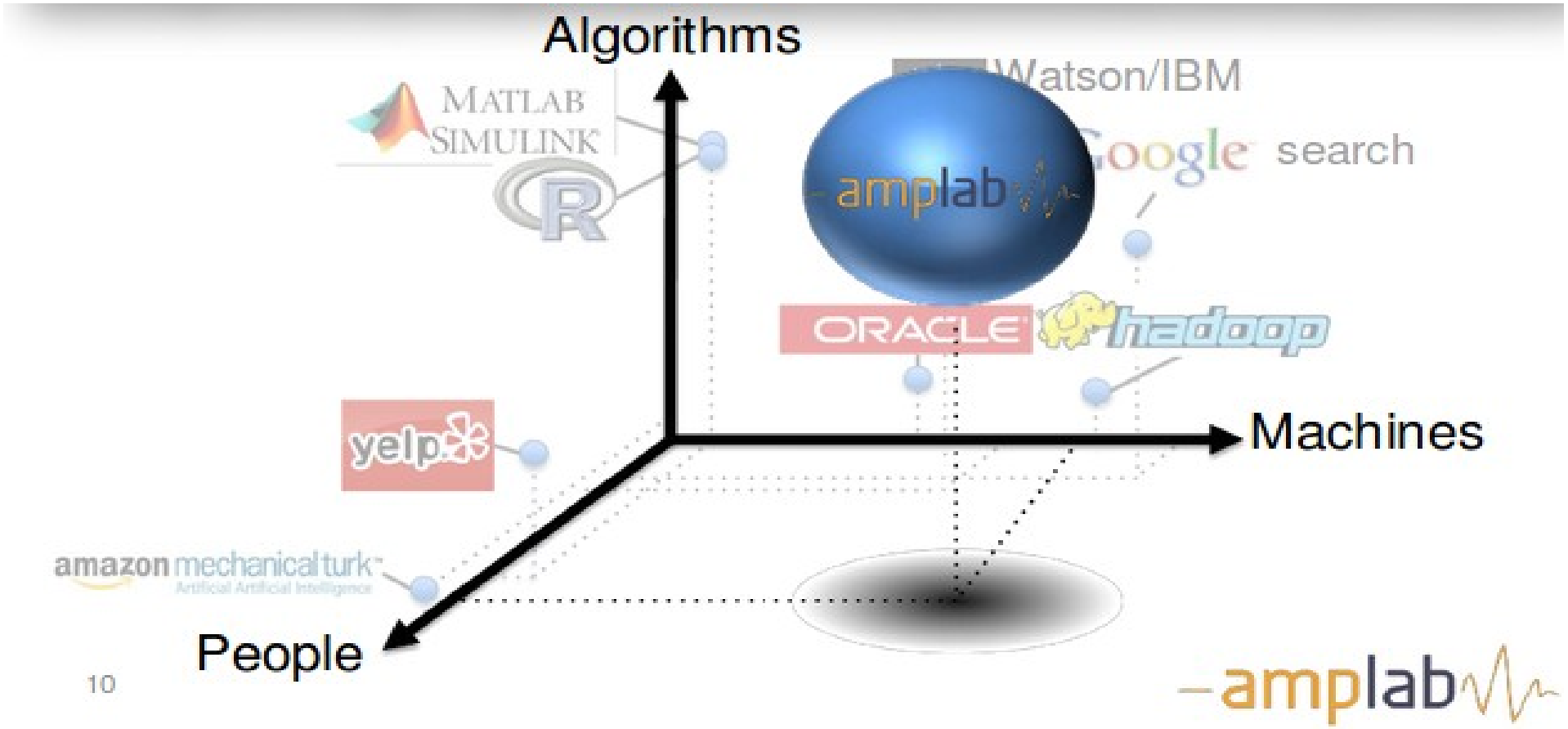


第一课： Spark 生态和安装部署

法律声明

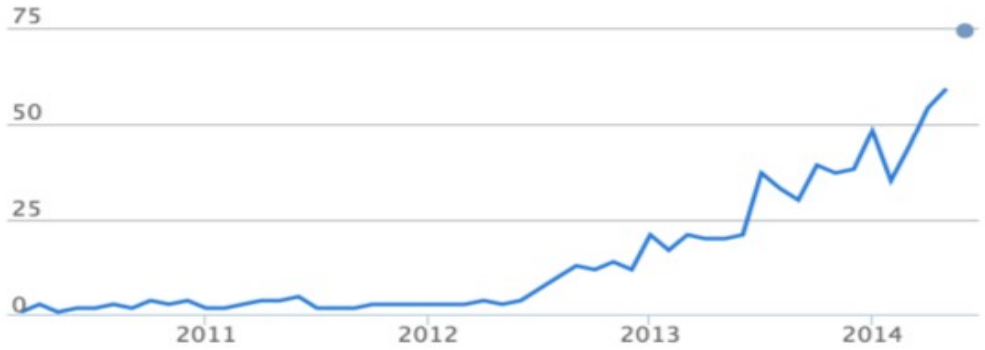
【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

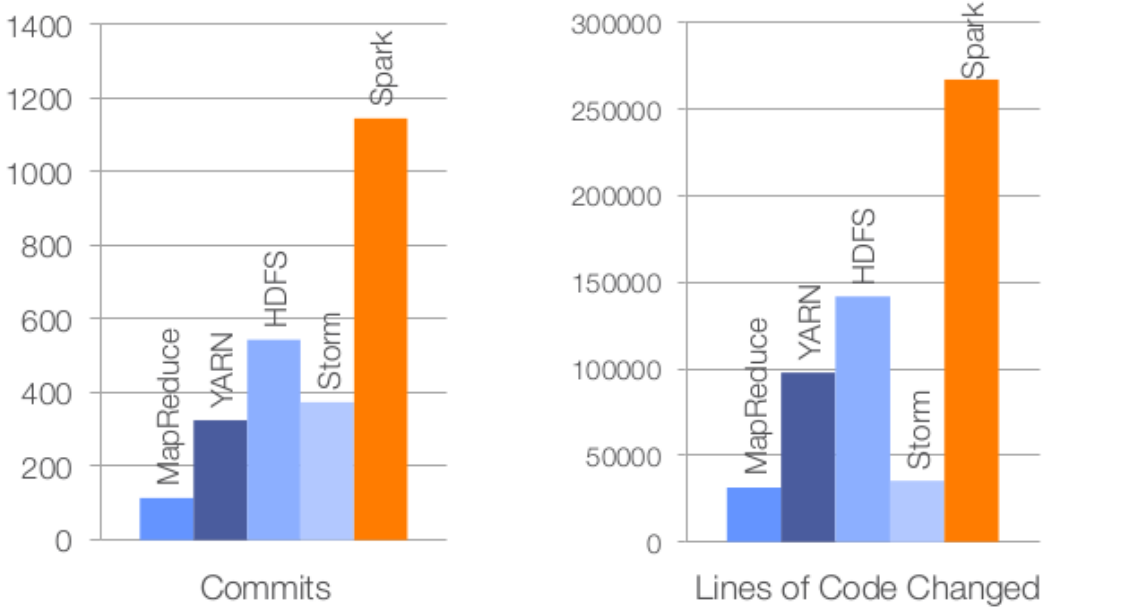
http://edu.dataguru.cn

News

	June 2013	June 2014
total contributors	68	255
companies contributing	17	50
total lines of code	63,000	175,000



Contributors per month to Spark



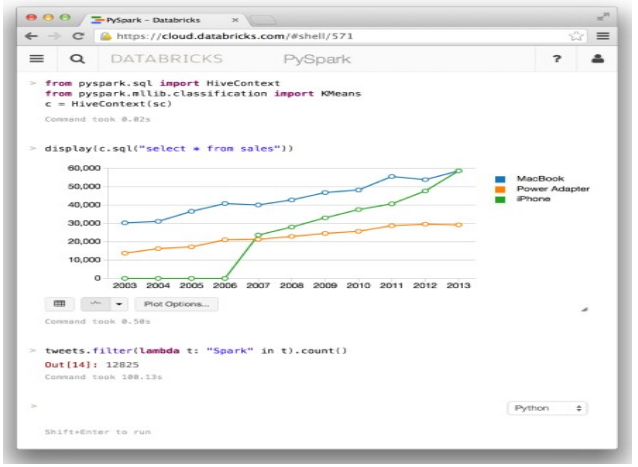
Activity in past 6 months

DATABRICKS

News


AMPLab
A M P
Databricks
Spark
Databricks Cloud
workspace


Databricks Cloud
Databricks Workspace
Spark
Databricks Platform


PySpark - Databricks
DATABRICKS
PySpark
from pyspark.sql import HiveContext
from pyspark.mllib.classification import KMeans
c = HiveContext(sc)
display(c.sql("select * from sales"))
MacBook
Power Adapter
iPhone
tweets.filter(lambda t: "Spark" in t).count()
Out[14]: 12825


Typical Data Pipeline
ETL
Exploration
Dashboards & Reports
Advanced Analytics
Data Products

本课内容

- Spark 是什么?
- Spark 有什么?
- Spark 部署
- Spark 实用工具简介

本课内容

- Spark 是什么?
- Spark 有什么?
- Spark 部署
- Spark 实用工具简介

Spark 是什么?

Spark.apache.org

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

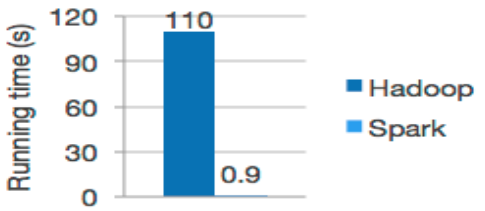
www.databricks.com

Spark 是什么?

Speed

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

Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.



Logistic regression in Hadoop and Spark

Ease of Use

Write applications quickly in Java, Scala or Python.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala and Python shells.

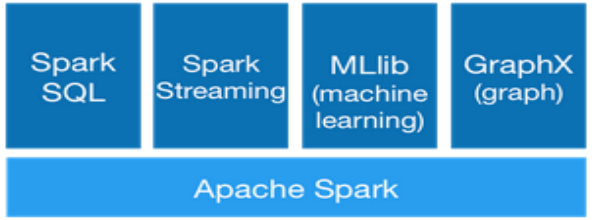
```
file = spark.textFile("hdfs://...")

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

Word count in Spark's Python API

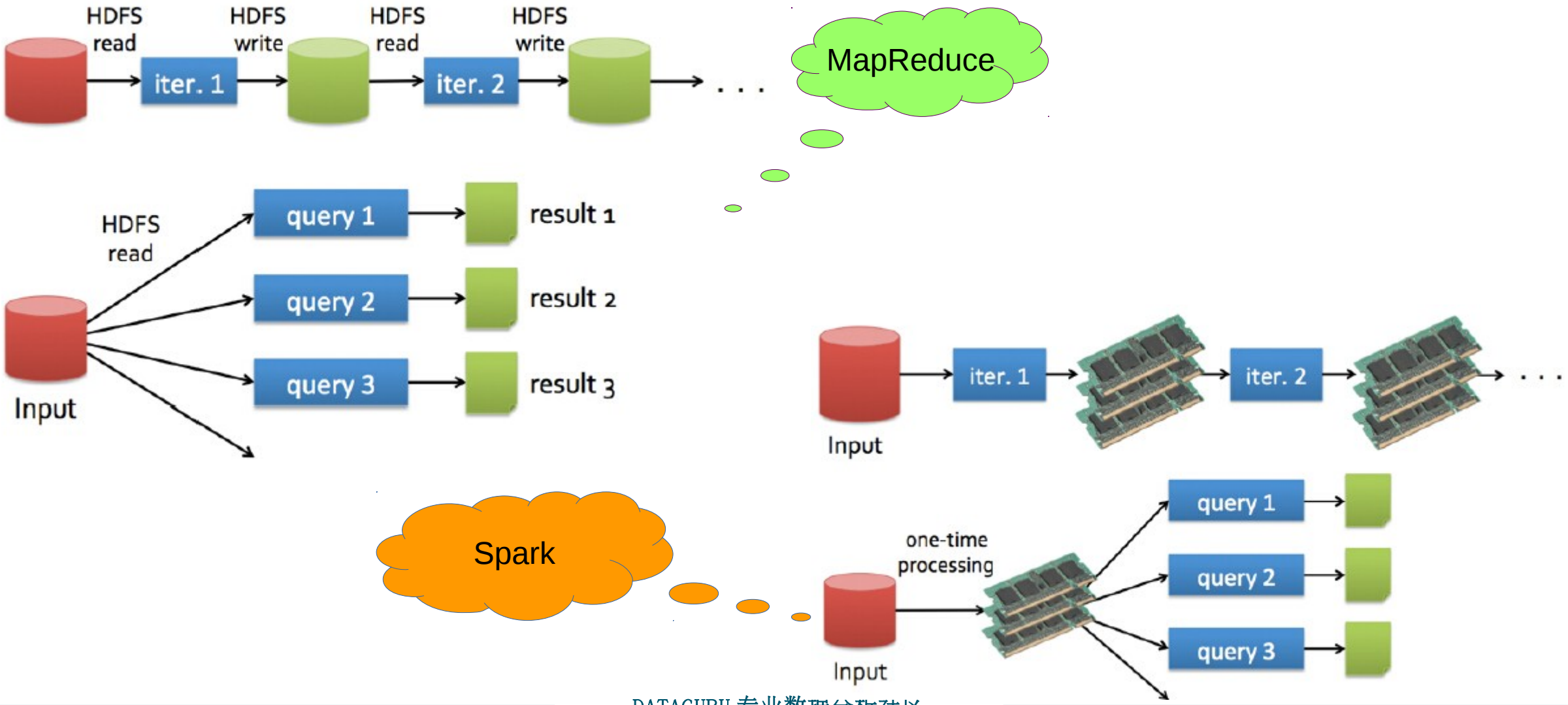
Generality

Combine SQL, streaming, and complex analytics.

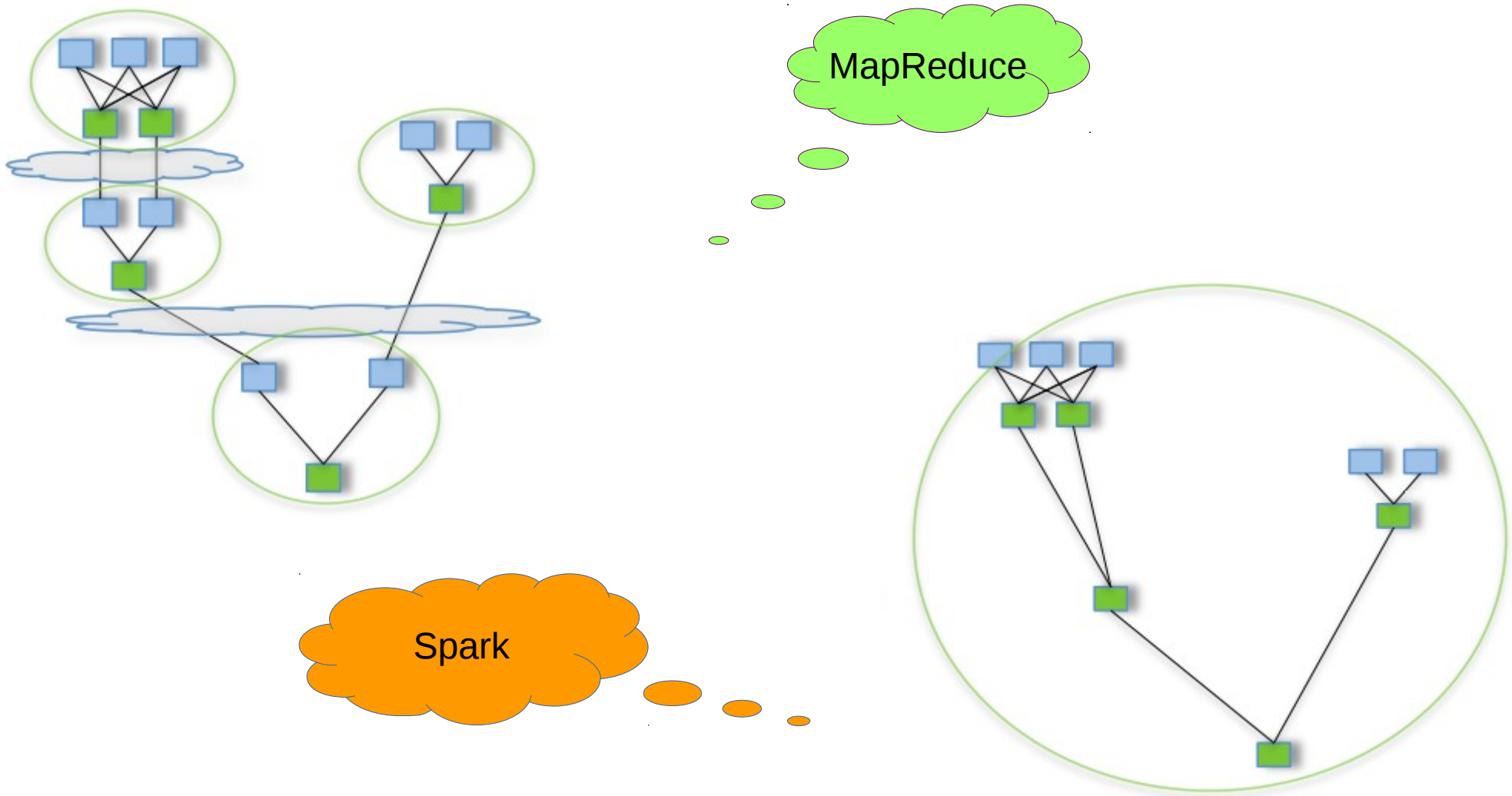
Spark powers a stack of high-level tools including Spark SQL, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these frameworks seamlessly in the same application.



Spark 是什么?


HDFS read
iter. 1
HDFS write
HDFS read
iter. 2
HDFS write
. . .
MapReduce
HDFS read
query 1
result 1
query 2
result 2
query 3
result 3
Input
Input
iter. 1
iter. 2
. . .
Spark
one-time processing
query 1
query 2
query 3
Input

Spark 是什么?



Spark 是什么？

```
67
68  public static void main(String[] args) throws Exception {
69    Configuration conf = new Configuration();
70    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
71    if (otherArgs.length != 2) {
72      System.err.println("Usage: wordcount <in> <out>");
73      System.exit(2);
74    }
75    Job job = new Job(conf, "word count");
76    job.setJarByClass(WordCount.class);
77    job.setMapperClass(TokenizerMapper.class);
78    job.setCombinerClass(IntSumReducer.class);
79    job.setReducerClass(IntSumReducer.class);
80    job.setOutputKeyClass(Text.class);
81    job.setOutputValueClass(IntWritable.class);
82    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
83    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
84    System.exit(job.waitForCompletion(true) ? 0 : 1);
85  }
86 }
87
```

MapReduce

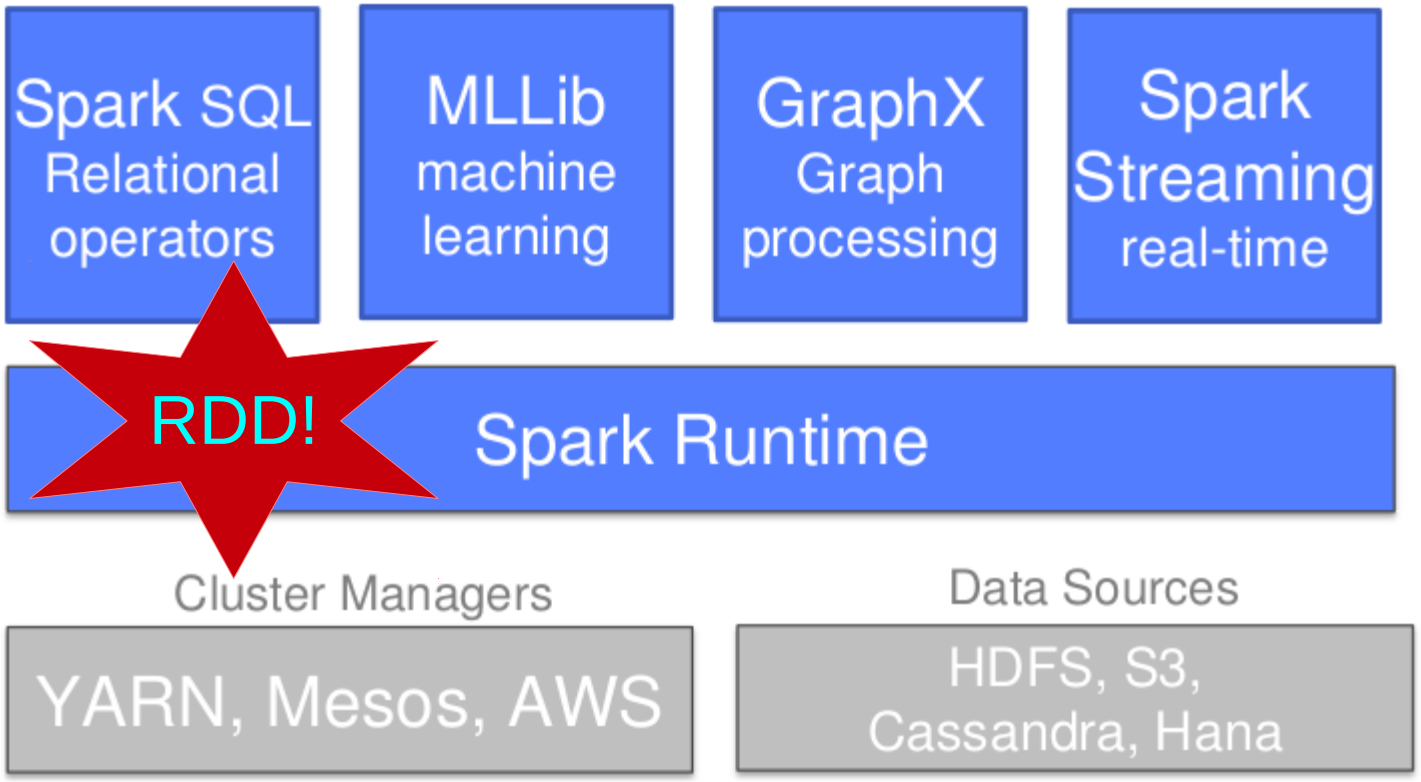
Spark

```
1  package week2
2
3  import org.apache.spark.{SparkContext, SparkConf}
4  import org.apache.spark.SparkContext._
5
6  object WordCount1 {
7    def main(args: Array[String]) {
8      if (args.length == 0) {
9        System.err.println("Usage: WordCount1 <file1>")
10       System.exit(1)
11     }
12
13     val conf = new SparkConf().setAppName("WordCount1")
14     val sc = new SparkContext(conf)
15     sc.textFile(args(0)).flatMap(_.split(" ")).map(x => (x, 1)).reduceByKey(_ + _).take(10).foreach(println)
16     sc.stop()
17   }
18 }
```

Spark 是什么?

```
15      sc.textFile(args(0)).flatMap(_.split(" ")).map(x => (x, 1)).reduceByKey(_ + _).take(10).foreach(println)
```

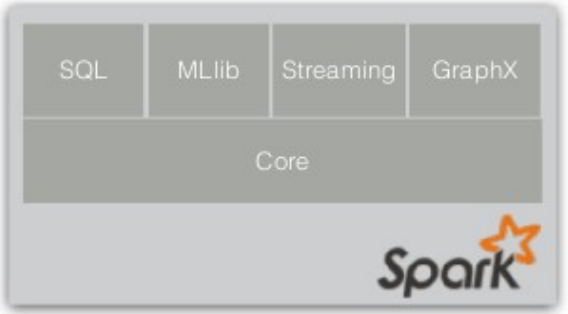
Spark 是什么?


Spark SQL
Relational
operators
MLLib
machine
learning
GraphX
Graph
processing
Spark
Streaming
real-time
RDD!
Spark Runtime
Cluster Managers
YARN, Mesos, AWS
Data Sources
HDFS, S3,
Cassandra, Hana

Spark 是什么?

Spark

An SDK for Big Data Applications



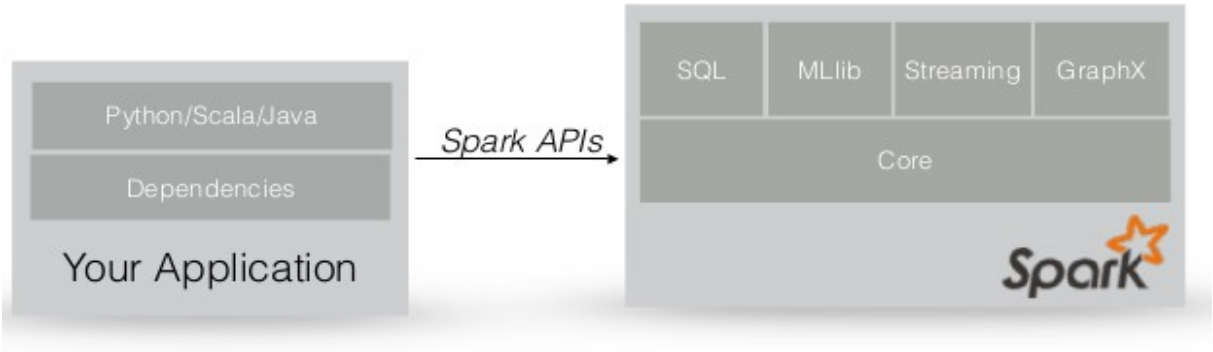
Unified System With Libraries to Build a Complete Solution

Full-featured Programming Environment

Single, Consistent Interface for Developers to Write Against

Runtimes available on several platforms

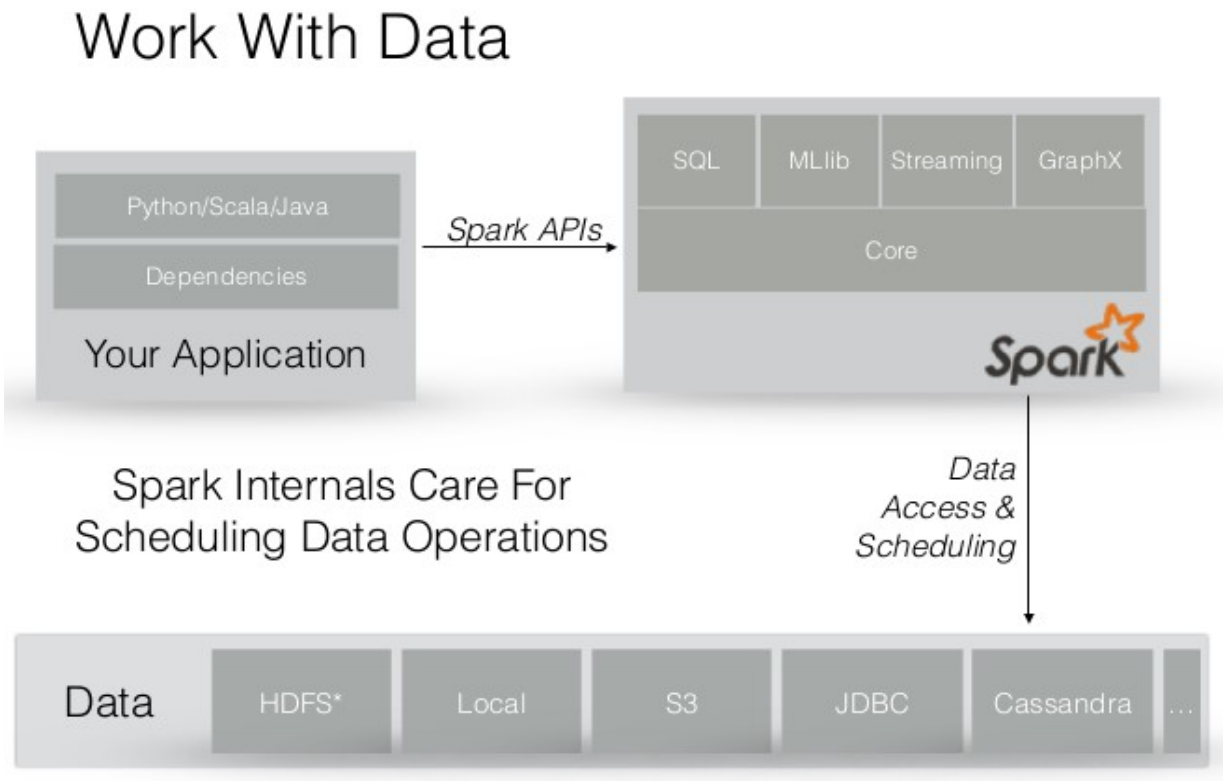
Develop Big Data Applications

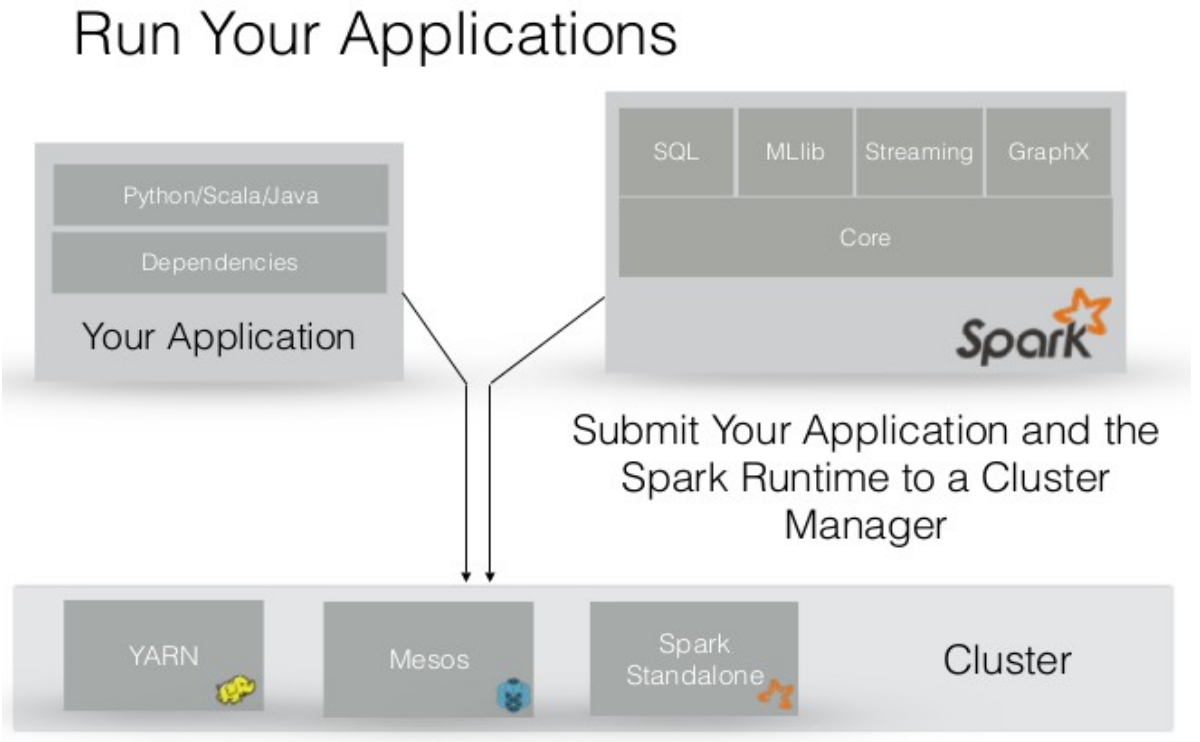


Develop Applications…

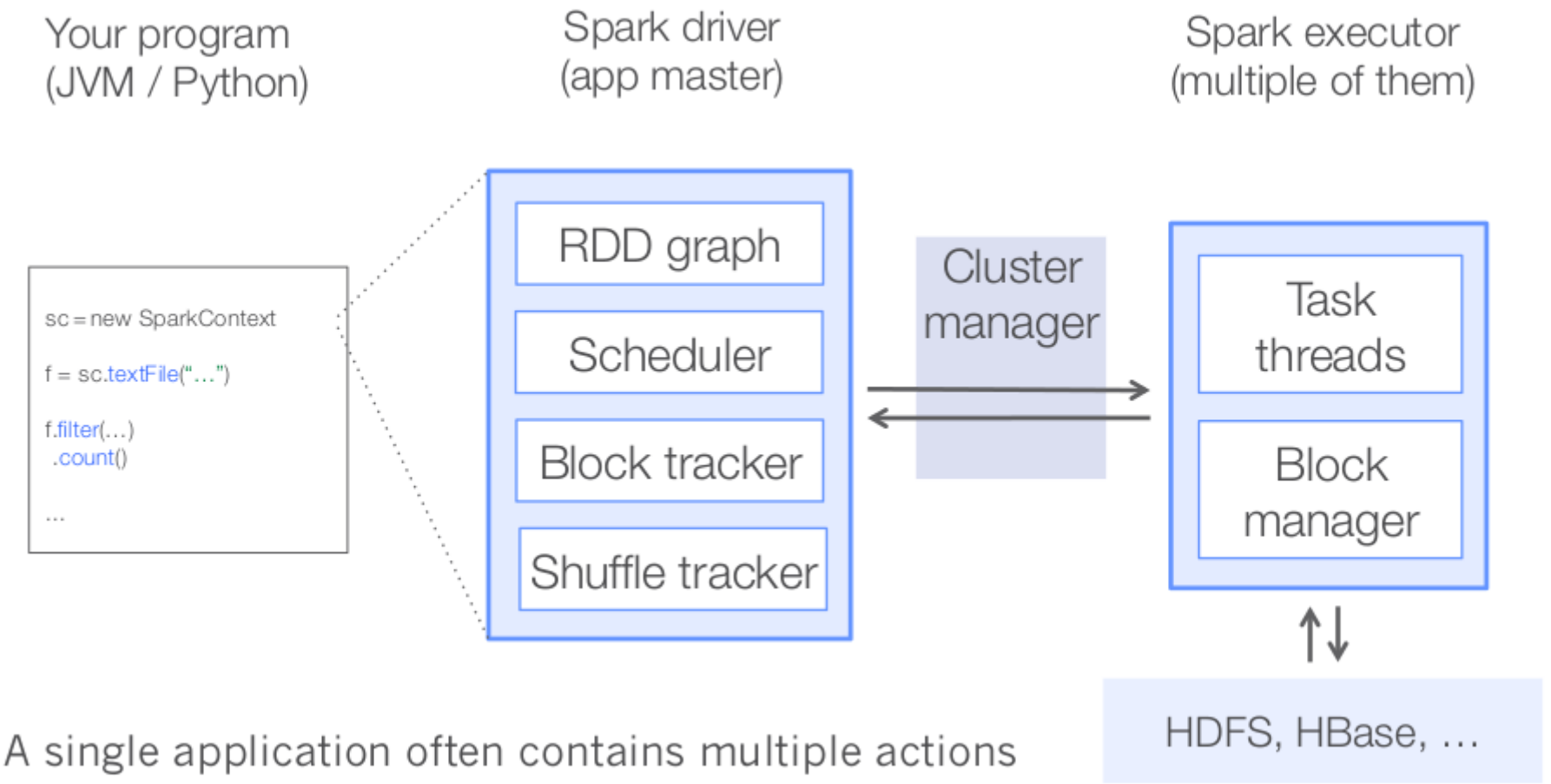
… using your preferred language,
… using existing libraries,
… using Spark's Public APIs
(SparkContext, RDDs)

Spark 是什么?


Work With Data
Python/Scala/Java
Dependencies
Your Application
Spark APIs
SQL
MLlib
Streaming
GraphX
Core
Spark
Spark Internals Care For
Scheduling Data Operations
Data
Access &
Scheduling
Data
HDFS*
Local
S3
JDBC
Cassandra
...


Run Your Applications
Python/Scala/Java
Dependencies
Your Application
SQL
MLlib
Streaming
GraphX
Core
Spark
Submit Your Application and the
Spark Runtime to a Cluster
Manager
YARN
Mesos
Spark
Standalone
Cluster

Spark 是什么?


Your program
(JVM / Python)
Spark driver
(app master)
Spark executor
(multiple of them)
sc = new SparkContext
f = sc.textFile("…")
f.filter(…)
.count()
…
RDD graph
Scheduler
Block tracker
Shuffle tracker
Cluster
manager
Task
threads
Block
manager
HDFS, HBase, …
A single application often contains multiple actions

Spark 是什么?

Patrick Wendell

The future of Spark is libraries

Critical component of any successful runtime

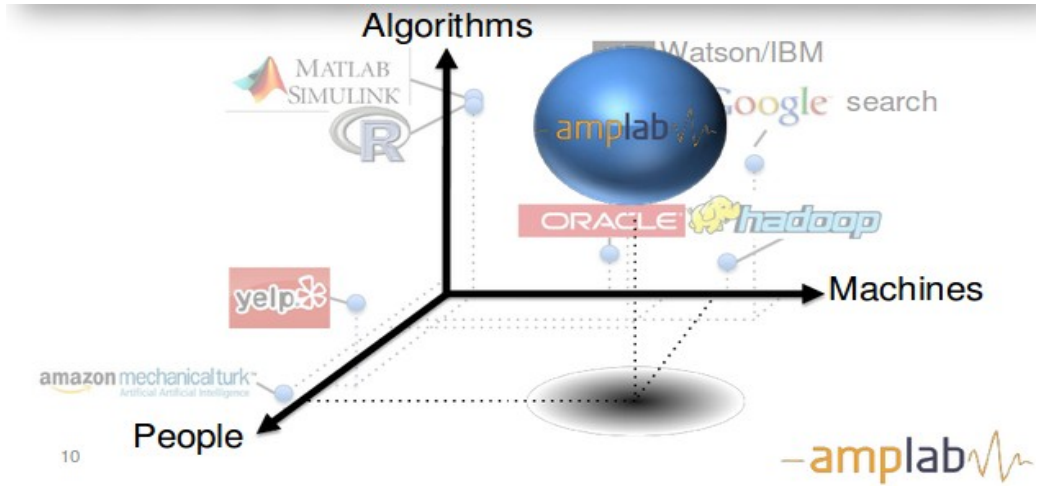
Packaged and distributed with Spark to provide full inter-operability

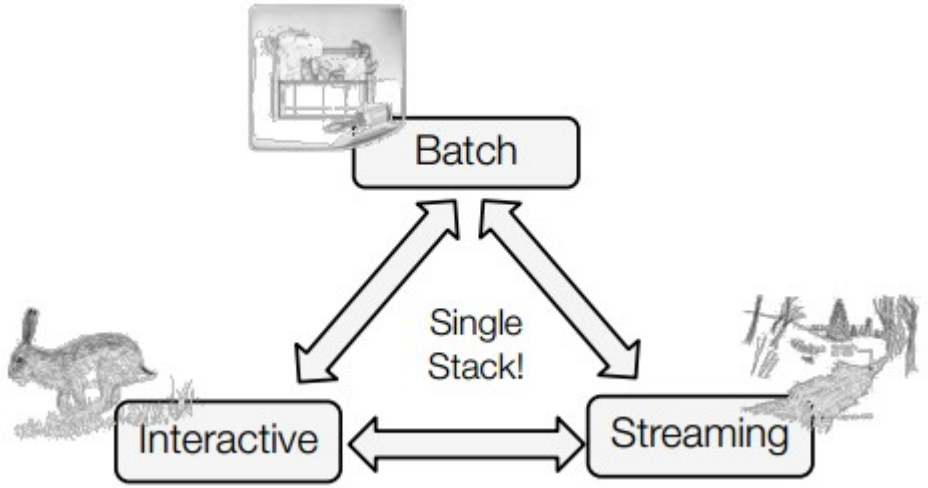
Lead by experts in respective fields, highly curated and integrated with Spark core API

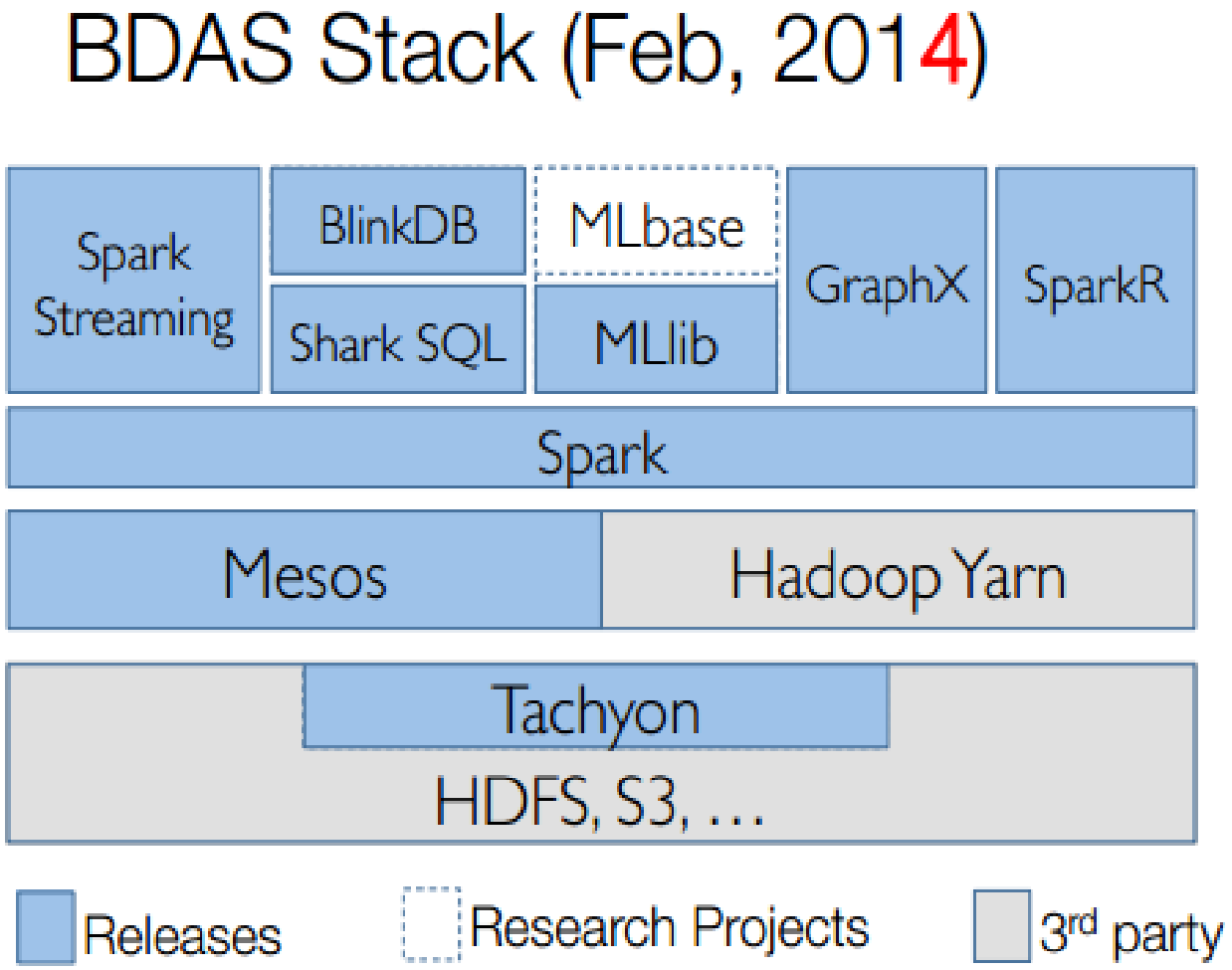
本课内容

- Spark 是什么?
- Spark 有什么?
- Spark 部署
- Spark 实用工具简介

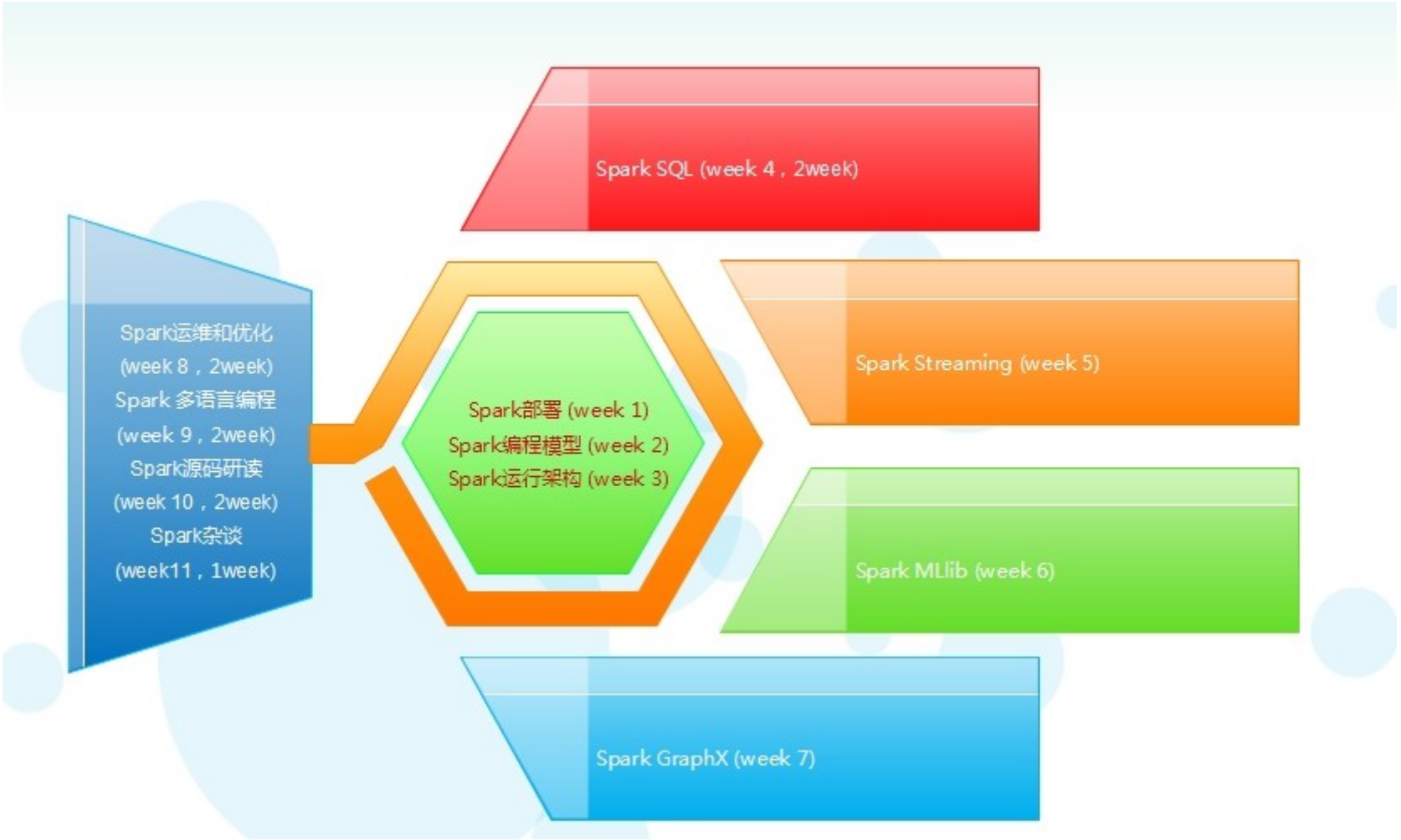
Spark 生态


Algorithms
MATLAB SIMULINK
Watson/IBM
Google search
amplab
ORACLE
hadoop
yelp
Machines
amazon mechanical turk
People
10
amplab

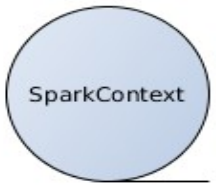

Batch
Single Stack!
Interactive
Streaming

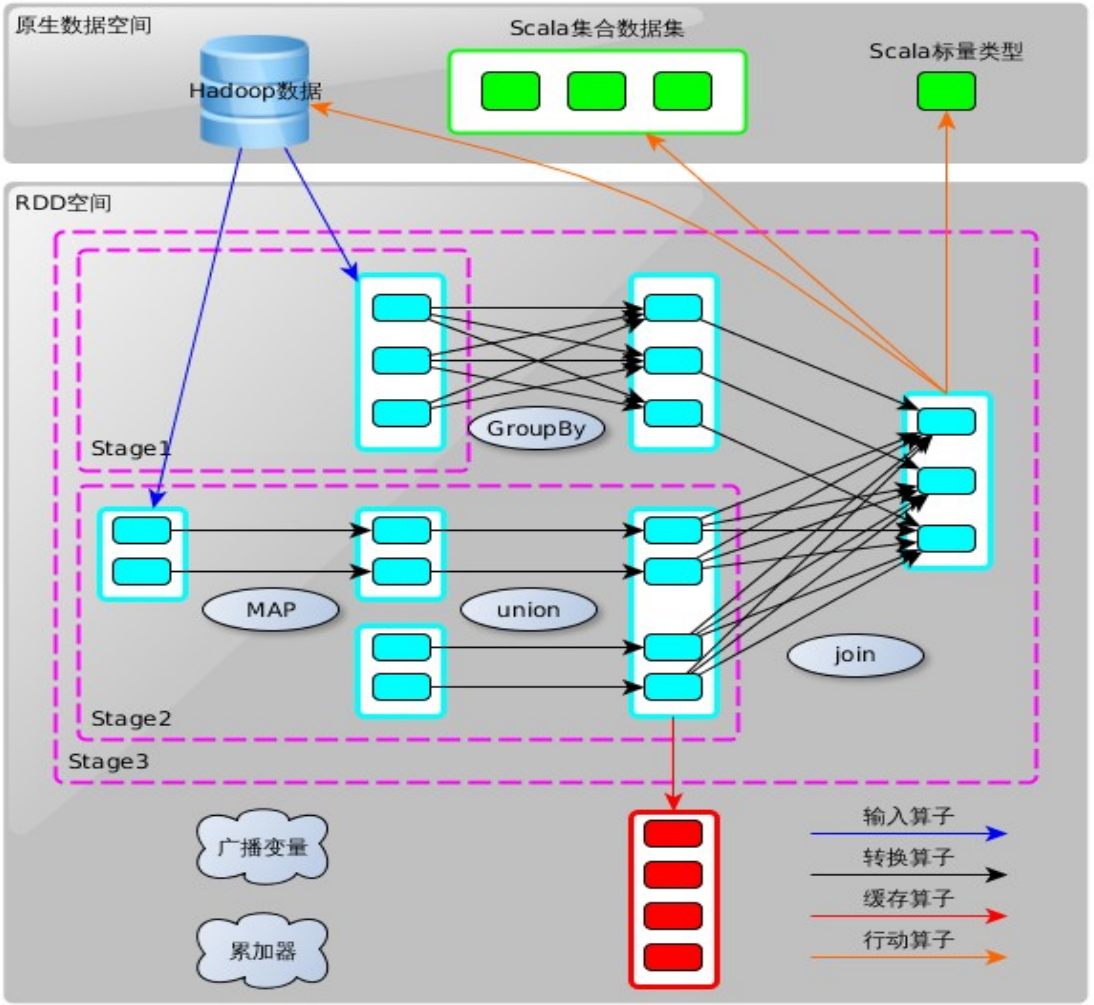

BDAS Stack (Feb, 2014)
Spark Streaming
BlinkDB
Shark SQL
MLbase
MLlib
GraphX
SparkR
Spark
Mesos
Hadoop Yarn
Tachyon
HDFS, S3, …
Releases
Research Projects
3rd party

Spark 生态


Spark SQL (week 4，2week)
Spark运维和优化
(week 8，2week)
Spark 多语言编程
(week 9，2week)
Spark源码研读
(week 10，2week)
Spark杂谈
(week11，1week)
Spark部署 (week 1)
Spark编程模型 (week 2)
Spark运行架构 (week 3)
Spark Streaming (week 5)
Spark MLlib (week 6)
Spark GraphX (week 7)

Spark 生态之 Spark Core





Transformations

The following table lists some of the common transformations supported by Spark. Refer to the RDD API doc (Scala, Java, Python) and pair RDD functions doc (Scala, Java) for details.

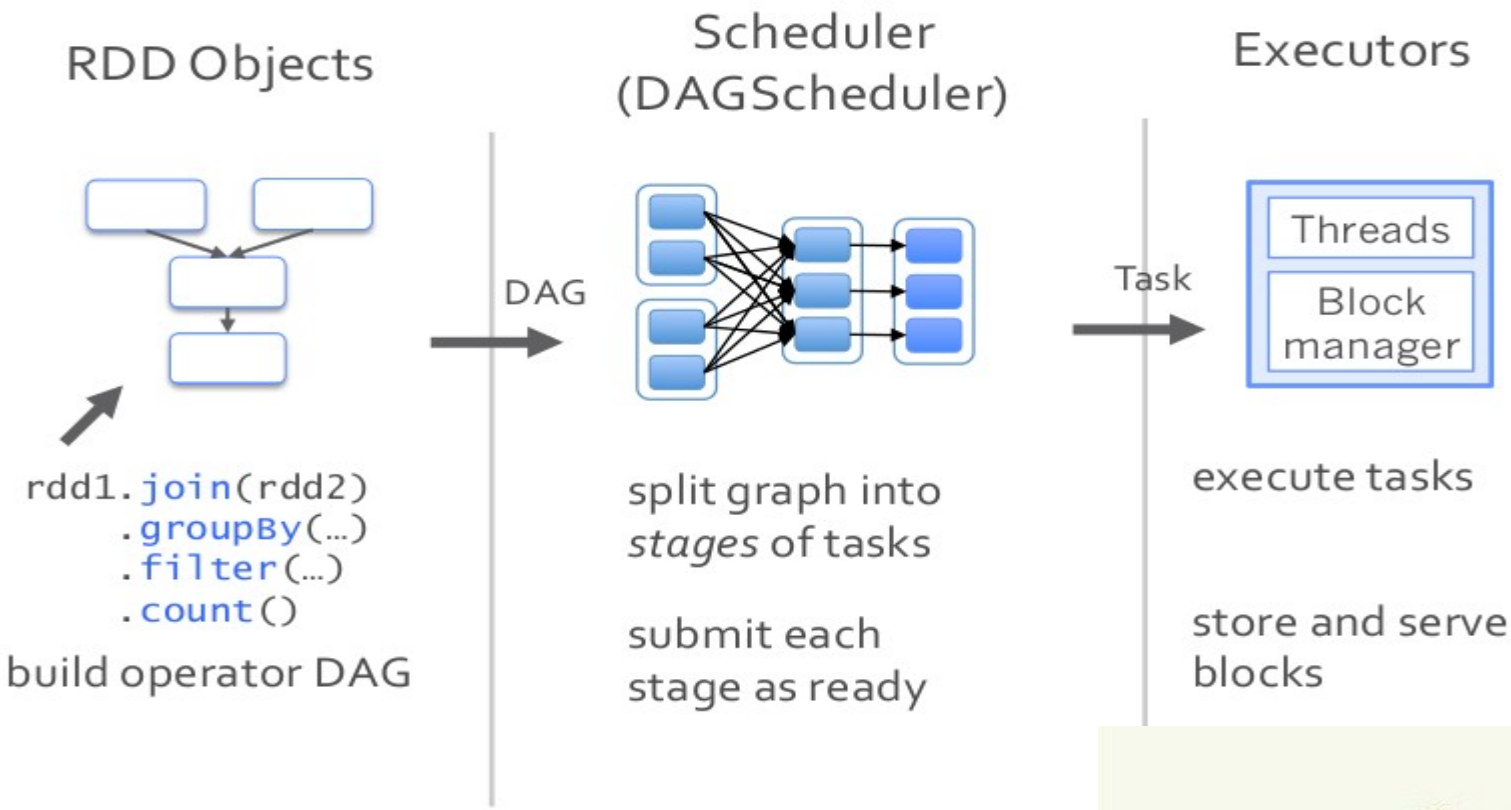
Transformation	Meaning
map(*func*)	Return a new distributed dataset formed by passing each element of the source through a function *func*.
filter(*func*)	Return a new dataset formed by selecting those elements of the source on which *func* returns true.
flatMap(*func*)	Similar to map, but each input item can be mapped to 0 or more output items (so *func* should return a Seq rather than a single item).
mapPartitions(*func*)	Similar to map, but runs separately on each partition (block) of the RDD, so *func* must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.
mapPartitionsWithIndex(*func*)	Similar to mapPartitions, but also provides *func* with an integer value representing the index of the partition, so *func* must be of type (Int, Iterator<T>) => Iterator<U> when running on an RDD of type T.
sample(*withReplacement*, *fraction*, *seed*)	Sample a fraction *fraction* of the data, with or without replacement, using a given random number generator seed.
union(*otherDataset*)	Return a new dataset that contains the union of the elements in the source dataset and the argument.

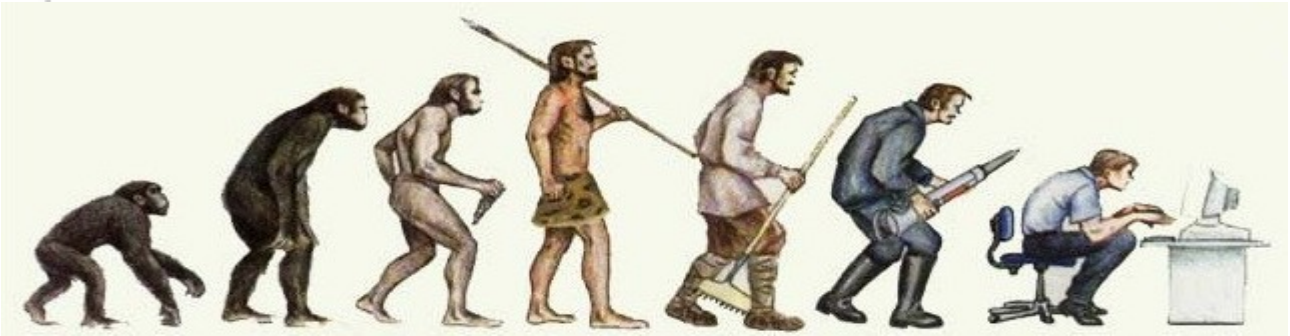
Actions

The following table lists some of the common actions supported by Spark. Refer to the RDD API doc (Scala, Java, Python) and pair RDD functions doc (Scala, Java) for details.

Action	Meaning
reduce(*func*)	Aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	Return the number of elements in the dataset.
first()	Return the first element of the dataset (similar to take(1)).
take(*n*)	Return an array with the first *n* elements of the dataset. Note that this is currently not executed in parallel. Instead, the driver program computes all the elements.
takeSample(*withReplacement*, *num*, *seed*)	Return an array with a random sample of *num* elements of the dataset, with or without replacement, using the given random number generator seed.
takeOrdered(*n*, *[ordering]*)	Return the first *n* elements of the RDD using either their natural order or a custom comparator.

Spark 生态之 Spark Core

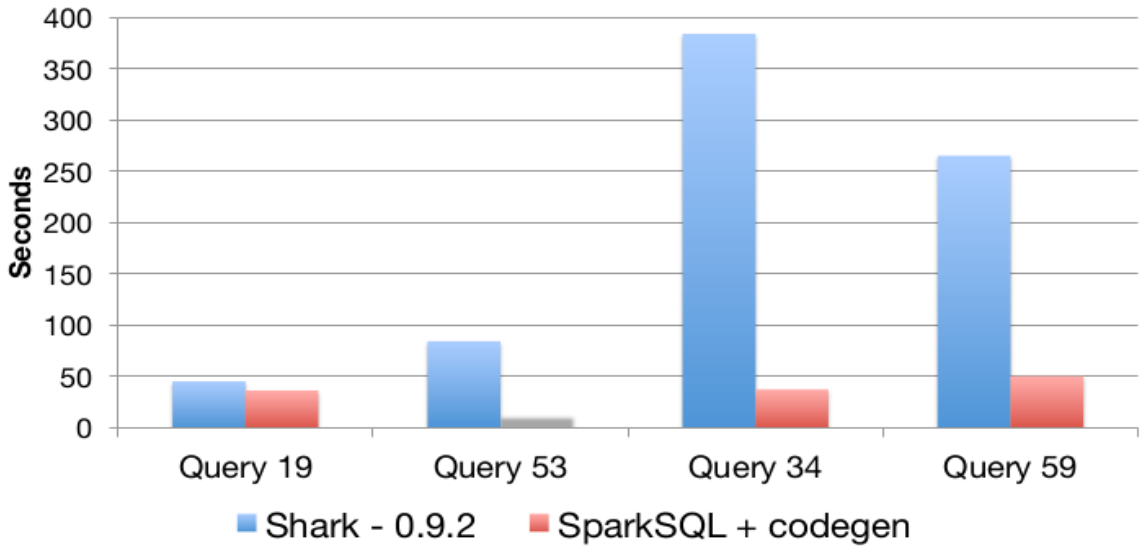




Spark 生态之 Spark Core

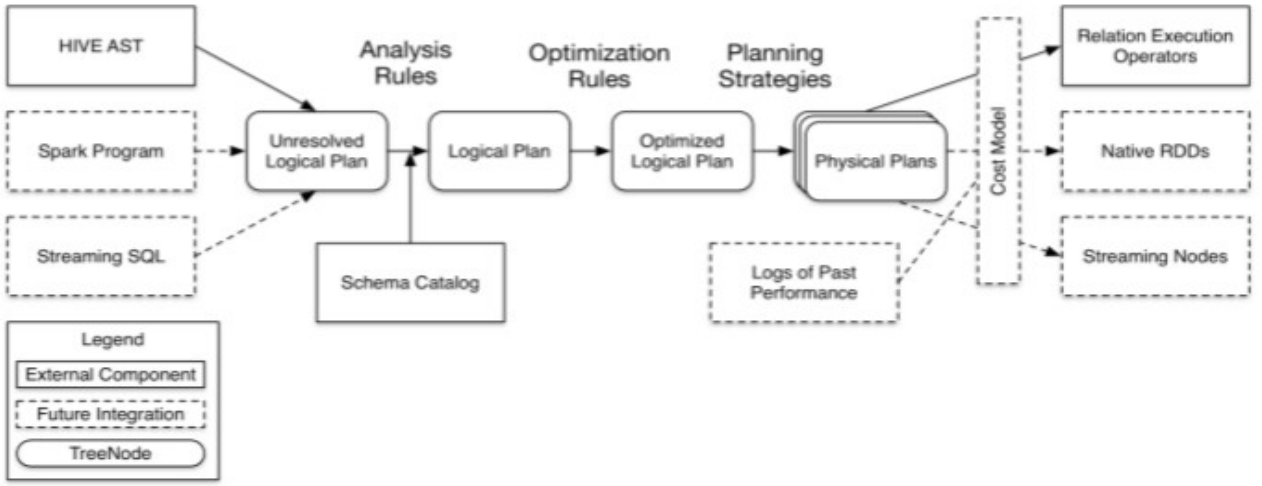
- 内存计算
- 提供了支持 DAG 图的分布式并行计算框架，减少多次计算之间中间结果 IO 开销
- 提供 Cache 机制来支持多次迭代计算或者数据共享，减少 IO 开销
- RDD 之间维护了血统关系，一旦 RDD fail 掉了，能通过父 RDD 自动重建，保证了容错性
- 移动计算而非移动数据， RDD Partition 可以就近读取分布式文件系统中的数据块到各个节点内存中进行计算
- 使用多线程池模型来减少 task 启动开稍
- shuffle 过程中避免不必要的 sort 操作
- 采用容错的、高可伸缩性的 akka 作为通讯框架
- 。。。

1.1 Preview: TPC-DS Results



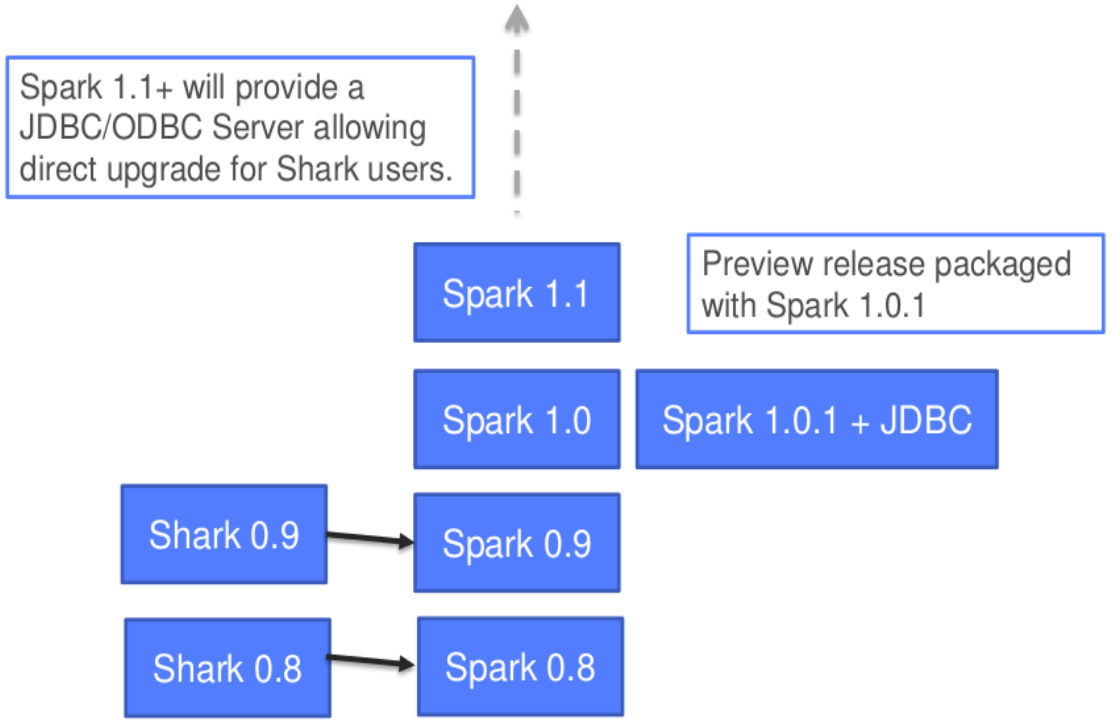
Unified Data Abstraction

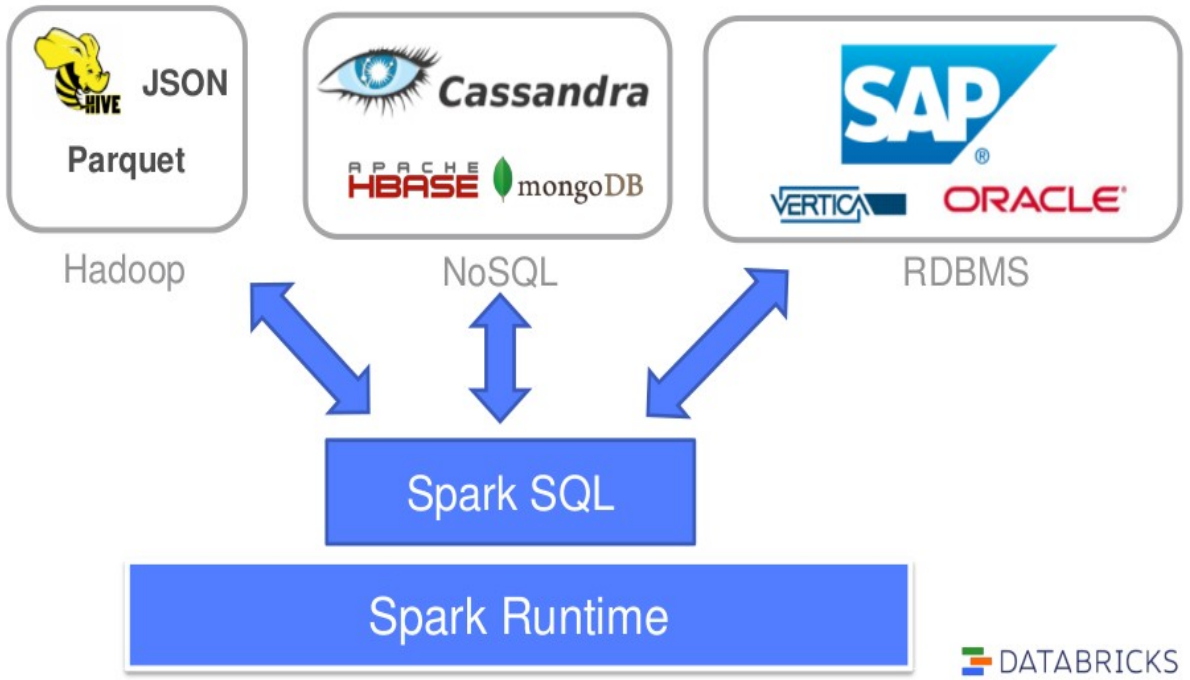
Spark 生态之 Spark SQL



- 引入了新的 RDD 类型 SchemaRDD，可以象传统数据库定义表一样来定义 SchemaRDD， SchemaRDD 由定义了列数据类型的行对象构成。
- SchemaRDD 可以从 RDD 转换过来，也可以从 Parquet 文件读入，也可以使用 HiveQL 从 Hive 中获取。
- 在应用程序中可以混合使用不同来源的数据，如可以将来自 HiveQL 的数据和来自 SQL 的数据进行 join 操作。
- 内嵌 catalyst 优化器对用户查询语句进行自动优化

Spark 生态之 Spark SQL



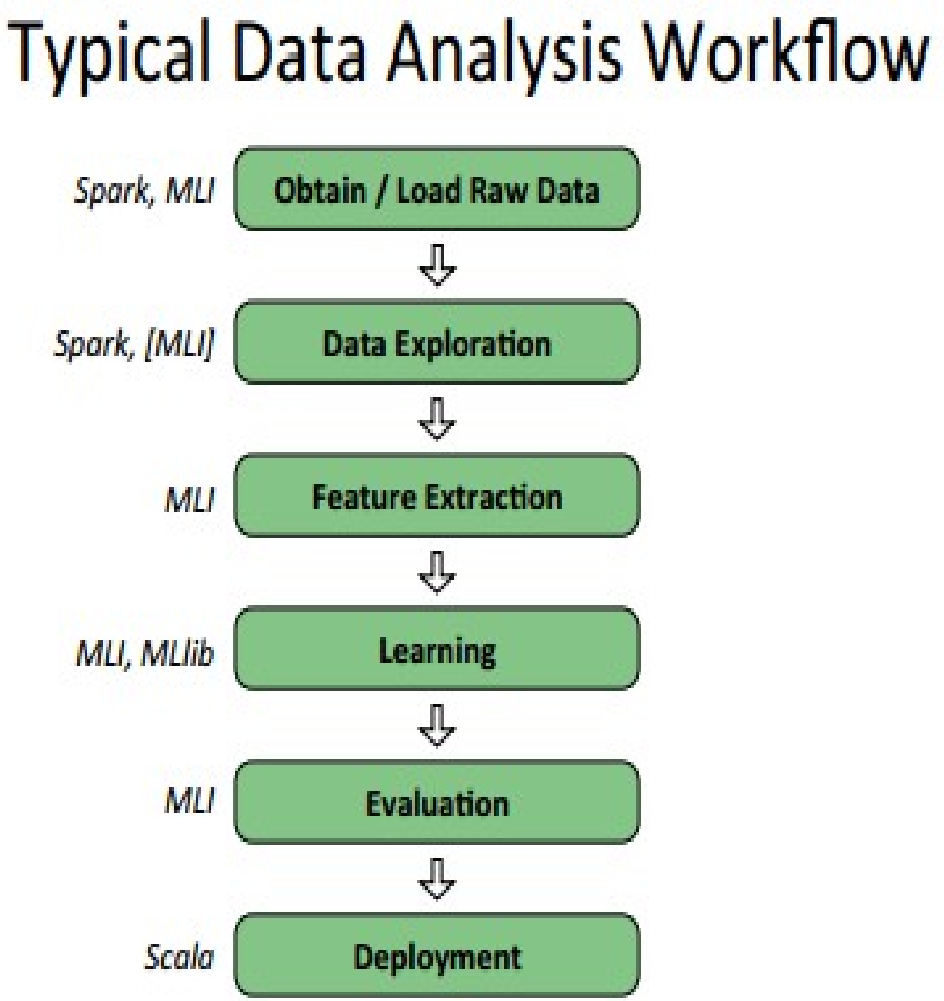


Spark 生态之 Spark Streaming

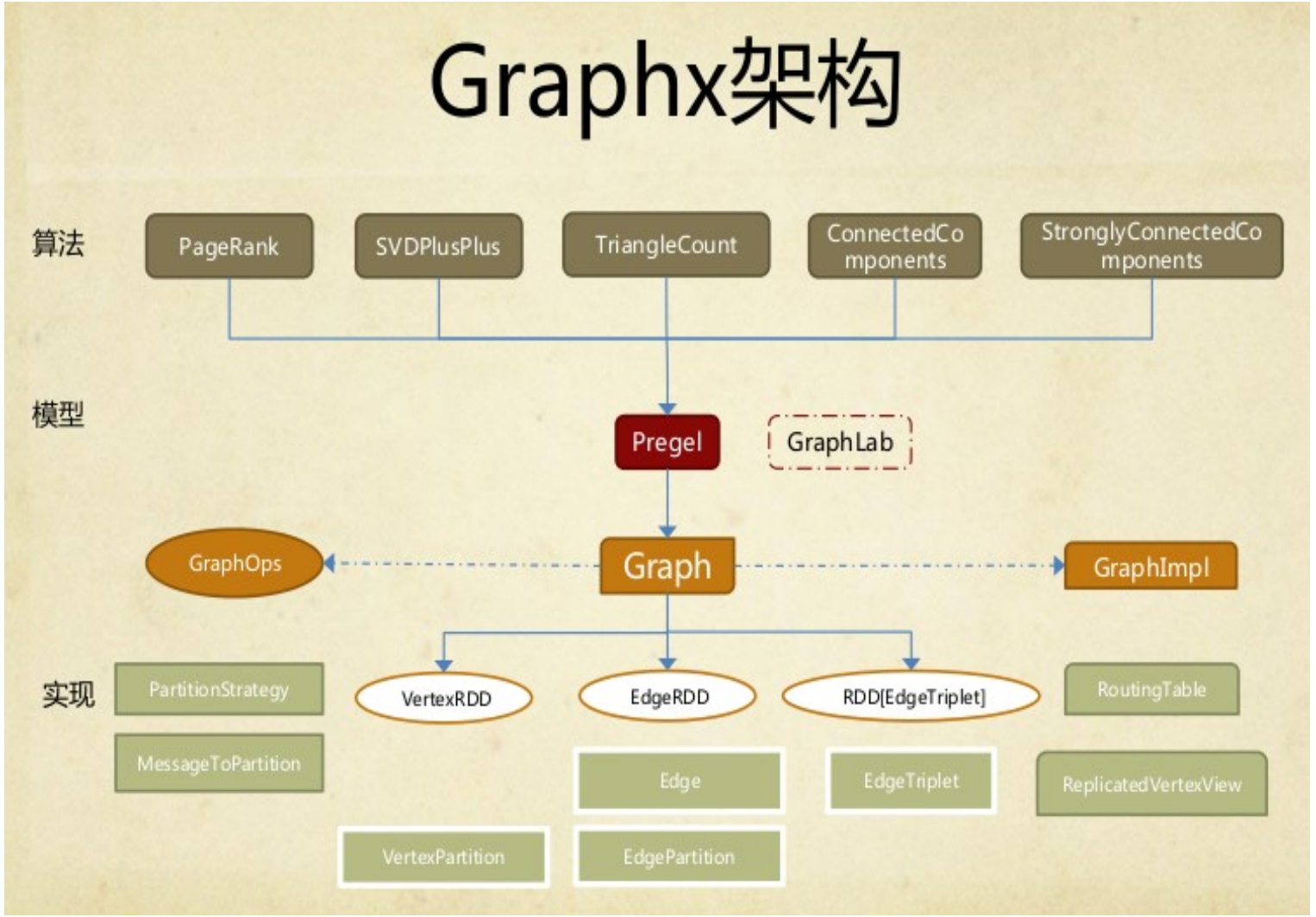
- 将流式计算分解成一系列短小的批处理作业
- 将失败或者执行较慢的任务在其它节点上并行执行
- 较强的容错能力（基于 RDD 继承关系 Lineage）
- 使用和 RDD 一样的语义

Spark 生态之 MLbase/MLlib

- ML Optimizer 优化器会选择最适合的、已经实现好了的机器学习算法和相关参数
- MLI 是一个进行特征抽取和高级 ML 编程抽象的算法实现的 API 或平台
- MLlib 基于 Spark 的底层分布式机器学习库，可以不断的扩充算法
- MLRuntime 基于 Spark 计算框架，将 Spark 的分布式计算应用到机器学习领域。

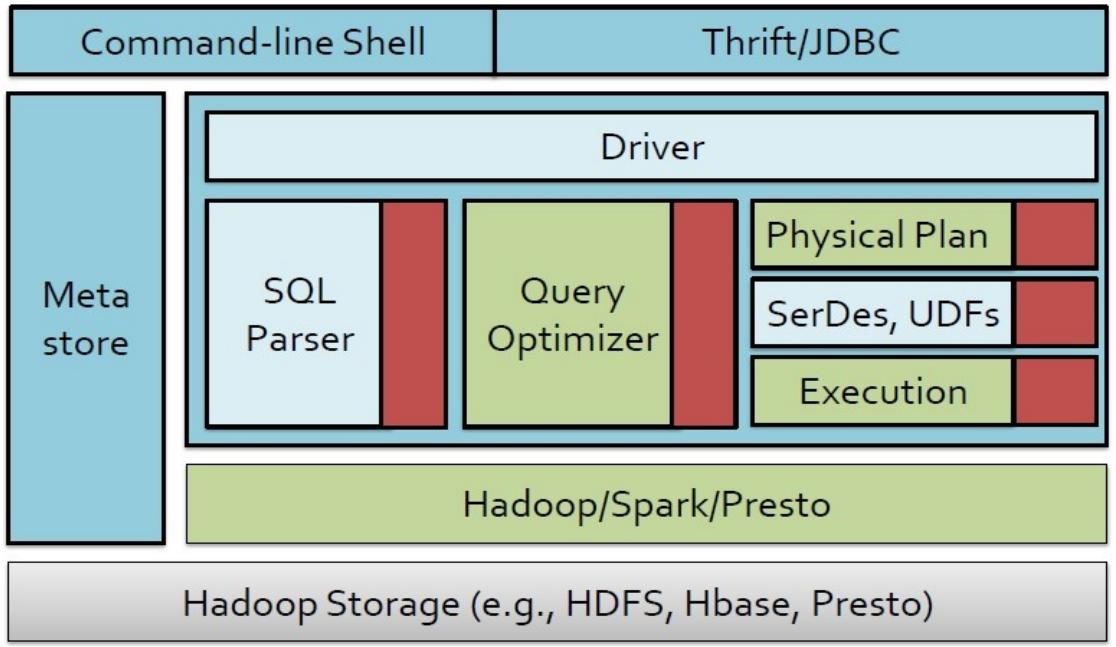


Spark 生态之 GraphX



- GraphX 定义了一个新的概念：弹性分布式属性图，一个每个顶点和边都带有属性的定向多重图；
- 引入了三种核心 RDD： Vertices、Edges、Triplets；
- 开放了一组基本操作（如 subgraph，joinVertices，and mapReduceTriplets）；
- 不断地扩展图形算法和图形构建工具来简化图分析工作。

BlinkDB Architecture




```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
WITHIN 2 SECONDS
```

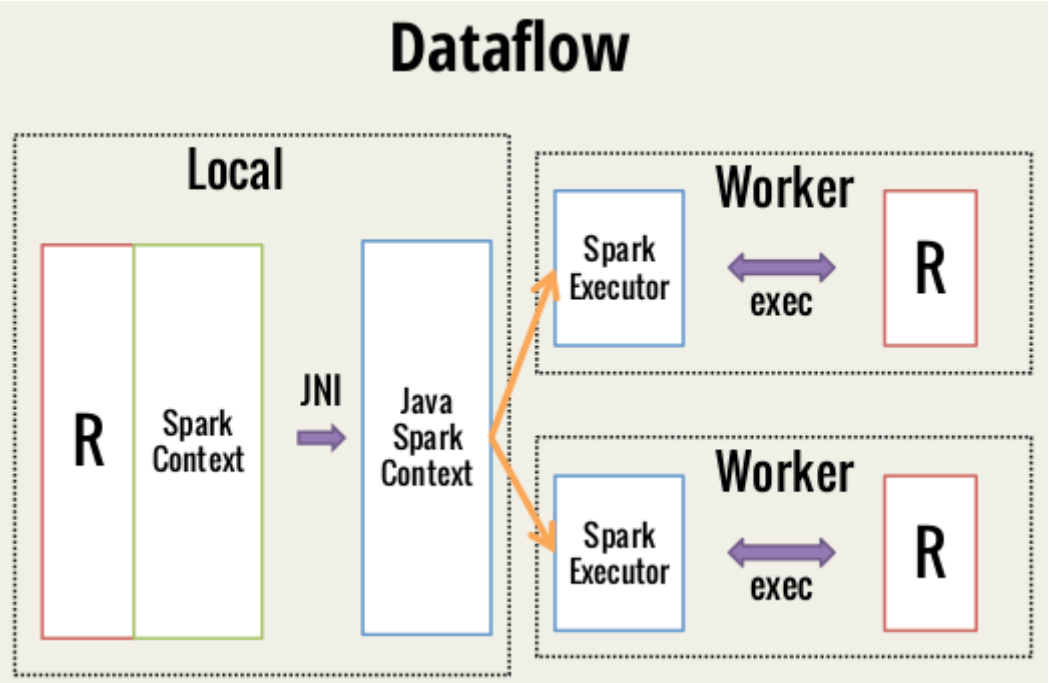
Queries with Time Bounds

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
ERROR 0.1 CONFIDENCE 95.0%
```

Queries with Error Bounds

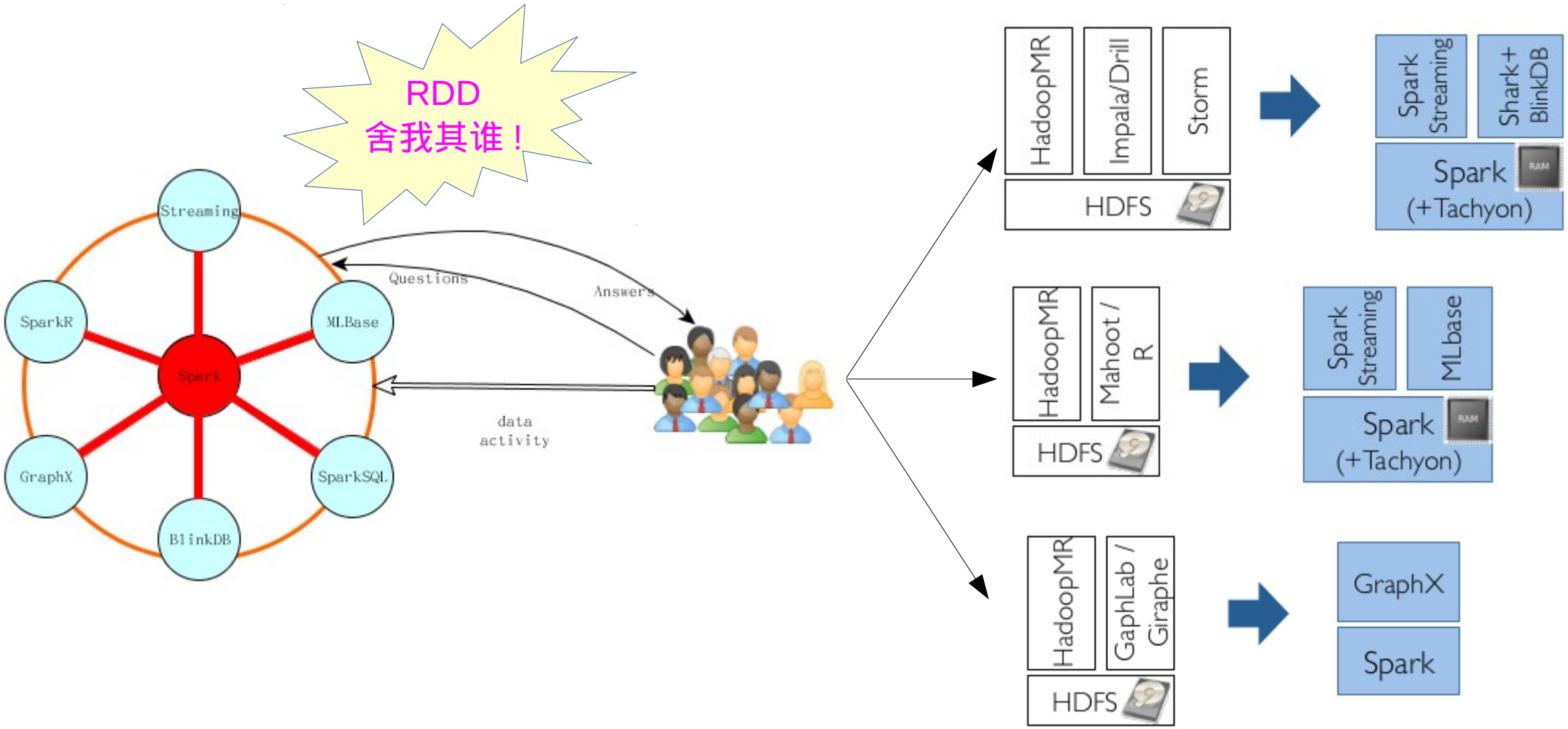
- BlinkDB 的设计核心思想：
  - 通过采样，建立并维护一组多维度样本
  - 查询进来时，选择合适的样本来运行查询

Spark 生态之 SparkR



- 提供了 Spark 中弹性分布式数据集（RDD）的 API，用户可以在集群上通过 R shell 交互性的运行 Spark job。
- 支持序化闭包功能，可以将用户定义函数中所用到的变量自动序化发送到集群中其他机器上。
- SparkR 还可以很容易地调用 R 开发包，只需要在集群上执行操作前用 includePackage 读取 R 开发包就可以了，当然集群上要安装 R 开发包。

Spark 生态


RDD
舍我其谁！
Streaming
SparkR
MLBase
Spark
GraphX
SparkSQL
BlinkDB
Questions
Answers
data
activity
HadoopMR
Impala/Drill
Storm
HDFS
Spark Streaming
Shark+ BlinkDB
Spark (+Tachyon)
RAM
HadoopMR
Mahoot / R
HDFS
Spark Streaming
MLbase
Spark (+Tachyon)
RAM
HadoopMR
GaphLab / Giraphe
HDFS
GraphX
Spark

本课内容

- Spark 是什么?
- Spark 有什么?
- Spark 部署
- Spark 实用工具简介

Spark 部署

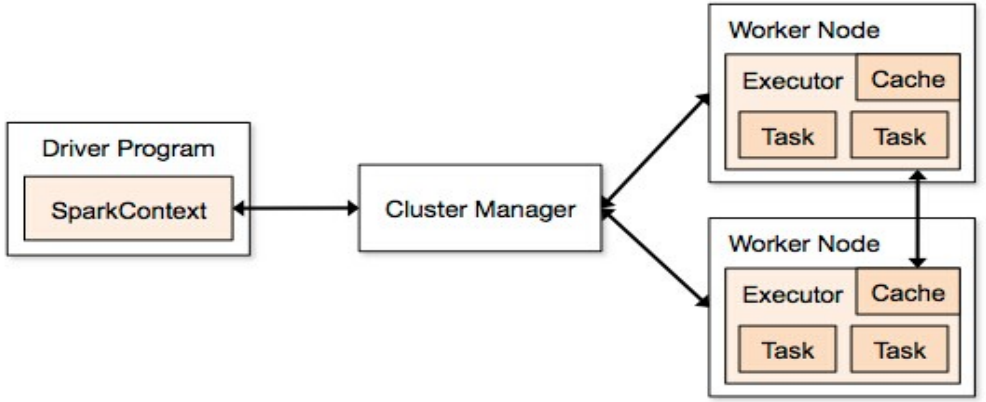
- Spark 部署简介
- Spark 源码编译
- Spark 部署包生成
- Spark Standalone 部署
- Spark Standalone HA 部署
- Spark Standalone 伪分布式部署

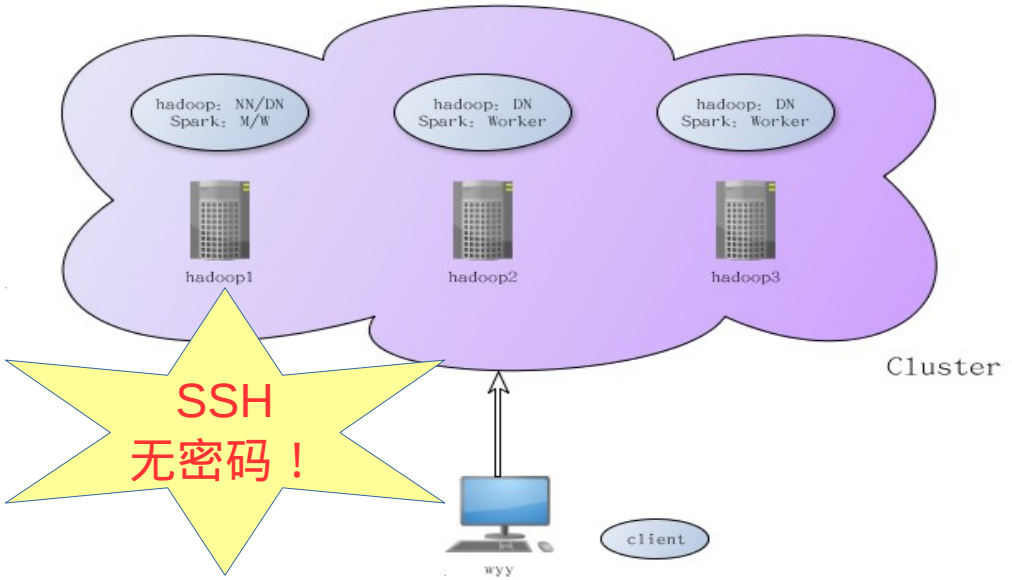
- TIPS：
  Spark1.0.0 for hadoop2 下载地址：
  http://d3kbcqa49mib13.cloudfront.net/spark-1.0.0-bin-hadoop2.tgz
  Spark1.0.0 源代码下载地址：
  http://d3kbcqa49mib13.cloudfront.net/spark-1.0.0.tgz
  github 下载最新源代码：
  git clone https://github.com/apache/spark.git

Spark 部署

- Spark 应用程序部署
  - local
  - Spark Standalone
  - Hadoop YARN
  - Apache Mesos
  - Amazon EC2

- Spark Standalone 集群部署
  - Standalone
  - Standalone HA





Spark 部署

- Spark 源码编译
  - SBT 编译

```
SPARK_HADOOP_VERSION=2.2.0 SPARK_YARN=true sbt/sbt assembly
```

  - Maven 编译

```
export MAVEN_OPTS="-Xmx2g -XX:MaxPermSize=512M -XX:ReservedCodeCacheSize=512m"
mvn -Pyarn -Dhadoop.version=2.2.0 -Dyarn.version=2.2.0 -DskipTests clean package
```

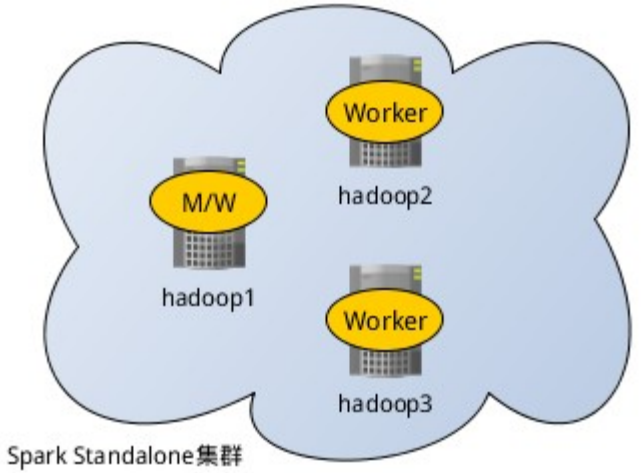
Spark 部署

- Spark 部署包生成命令 make-distribution.sh
  --hadoop VERSION： Hadoop 版本号，不加此参数时 hadoop 版本为 1.0.4 。
  --with-yarn：是否支持 Hadoop YARN，不加参数时为不支持 yarn 。
  --with-hive：是否在 Spark SQL 中支持 hive，不加此参数时为不支持 hive 。
  --skip-java-test：是否在编译的过程中略过 java 测试，不加此参数时为略过。
  --with-tachyon：是否支持内存文件系统 Tachyon，不加此参数时不支持 tachyon 。
  --tgz：在根目录下生成 spark-$VERSION-bin.tgz，不加此参数时不生成 tgz 文件，只生成 /dist 目录。
  --name NAME：和— tgz 结合可以生成 spark-$VERSION-bin-$NAME.tgz 的部署包，不加此参数时 NAME 为 hadoop 的版本号。

- 部署包生成
  - 生成支持 yarn 、 hadoop2.2.0 的部署包：
    ./make-distribution.sh --hadoop 2.2.0 --with-yarn --tgz
  - 生成支持 yarn 、 hive 的部署包：
    ./make-distribution.sh --hadoop 2.2.0 --with-yarn --with-hive --tgz

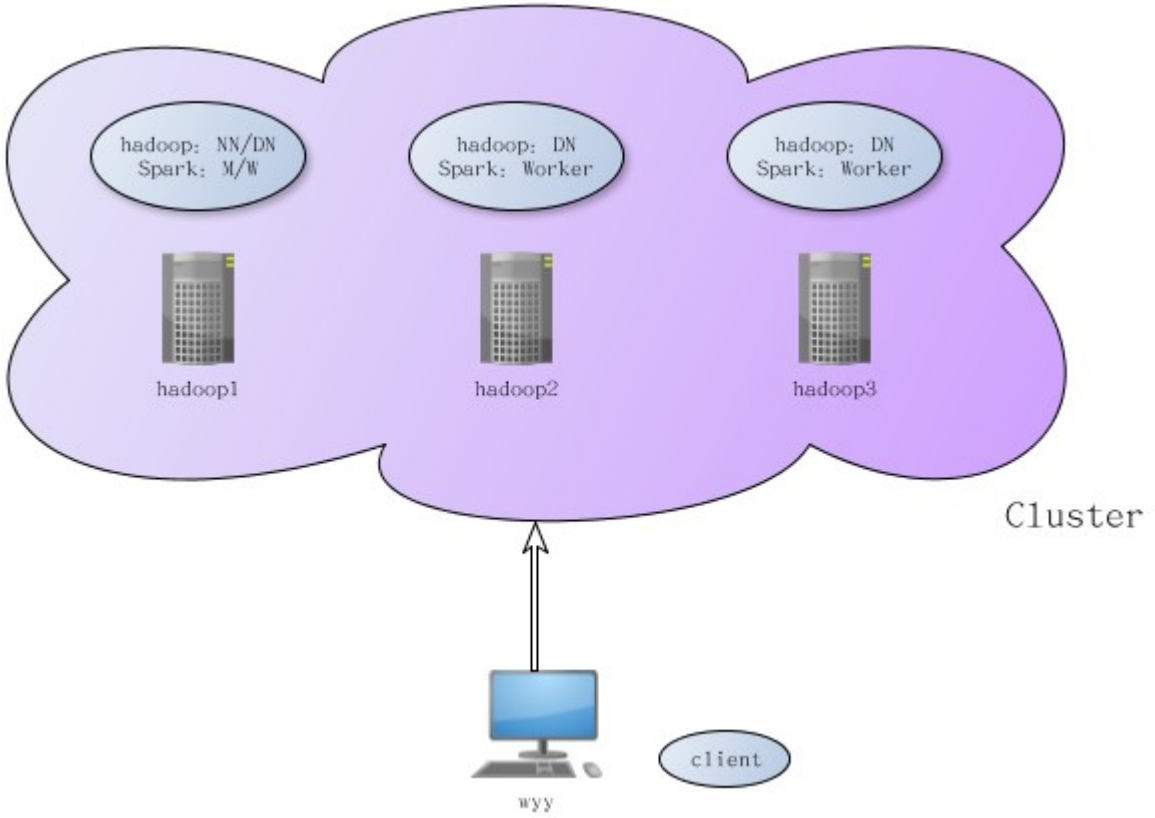
Spark 部署

- Spark Standalone 集群部署
  - Java 的安装
  - ssh 无密码登录
  - Spark 安装包解压
  - Spark 配置文件配置
    - 文件 conf/slave
      - hadoop1
      - hadoop2
      - hadoop3
    - 文件 conf/spark-env.sh
      - export SPARK_MASTER_IP=hadoop1
      - export SPARK_MASTER_PORT=7077
      - export SPARK_WORKER_CORES=1
      - export SPARK_WORKER_INSTANCES=1
      - export SPARK_WORKER_MEMORY=3g



Spark 部署

- Spark Client 部署
  - Java 的安装
  - ssh 无密码登录
  - Spark 安装包部署
  - 测试



Spark 部署

- Spark Standalone HA 部署
  - 基于文件系统的 HA
    - spark.deploy.recoveryMode　　设成 FILESYSTEM
    - spark.deploy.recoveryDirectory　　Spark 保存恢复状态的目录
    - Spark-env.sh 里对 SPARK_DAEMON_JAVA_OPTS 设置
    - export SPARK_DAEMON_JAVA_OPTS="-Dspark.deploy.recoveryMode=FILESYSTEM -Dspark.deploy.recoveryDirectory=/app/hadoop/spark100/recovery"
  - 基于 zookeeper 的 HA
    - spark.deploy.recoveryMode　　设置成 ZOOKEEPER
    - park.deploy.zookeeper.url　　ZooKeeper URL
    - spark.deploy.zookeeper.dir　　ZooKeeper 保存恢复状态的目录，缺省为 /spark
    - spark-env 里对 SPARK_DAEMON_JAVA_OPTS 设置
    - export SPARK_DAEMON_JAVA_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER -Dspark.deploy.zookeeper.url=hadoop1:2181,hadoop2:2181,hadoop3:2181 -Dspark.deploy.zookeeper.dir=/spark"

Spark 部署

- Spark Standalone 伪分布式部署
  - Java 的安装
  - ssh 无密码登录
  - Spark 安装包部署
  - 配置
  - 测试

本课内容

- Spark 是什么?
- Spark 有什么?
- Spark 部署
- Spark 实用工具简介

Spark 工具简介

- Spark 工具简介
  - Spark 交互工具 spark-shell
  - Spark 应用程序部署工具 spark-submit
  - 参数说明参见 http://blog.csdn.net/book_mmicky/article/details/25714545

Options:

--master MASTER_URL　　spark://host:port, mesos://host:port, yarn, or local.

--deploy-mode DEPLOY_MODE　driver运行之处，client运行在本机，cluster运行在集群

--class CLASS_NAME　　应用程序包的要运行的class

--name NAME　　应用程序名称

--jars JARS　　用逗号隔开的driver本地jar包列表以及executor类路径

--py-files PY_FILES　　用逗号隔开的放置在Python应用程序PYTHONPATH上的.zip, .egg, .py文件列表

--files FILES　　用逗号隔开的要放置在每个executor工作目录的文件列表

--properties-file FILE　设置应用程序属性的文件放置位置，默认是conf/spark-defaults.conf

--driver-memory MEM　　driver内存大小，默认512M

--driver-java-options　driver的java选项

--driver-library-path　driver的库路径Extra library path entries to pass to the driver

--driver-class-path　　driver的类路径，用--jars 添加的jar包会自动包含在类路径里

--executor-memory MEM　executor内存大小，默认1G

Spark 工具简介

Spark standalone with cluster deploy mode only:

--driver-cores NUM　　driver使用内核数，默认为1

--supervise　　如果设置了该参数，driver失败是会重启

Spark standalone and Mesos only:

--total-executor-cores NUM　executor使用的总核数

YARN-only:

--executor-cores NUM　　每个executor使用的内核数，默认为1

--queue QUEUE_NAME　　提交应用程序给哪个YARN的队列，默认是default队列

--num-executors NUM　　启动的executor数量，默认是2个

--archives ARCHIVES　　被每个executor提取到工作目录的档案列表，用逗号隔开

Spark 工具简介

- spark-shell
- 例子
  - bin/spark-shell --executor-memory 2g --driver-memory 1g --master spark://hadoop1:7077
  - scala> val rdd=sc.textFile("hdfs://hadoop1:8000/dataguru/data/NOTICE")
  - scala> rdd.cache()
  - scala> val wordcount=rdd.flatMap(_.split(" ")).map(x=>(x,1)).reduceByKey(_+_)
  - scala> wordcount.take(10)
  - scala> val wordsort=wordcount.map(x=>(x._2,x._1)).sortByKey(false).map(x=>(x._2,x._1))
  - scala> wordsort.take(10)

Spark 工具简介

- spark-submit
- 例子：
  - bin/spark-submit --master spark://hadoop1:7077 --class org.apache.spark.examples.SparkPi --executor-memory 2g lib/spark-examples-1.0.0-hadoop2.2.0.jar 1000
  - bin/spark-submit --master spark://hadoop1:7077 --class org.apache.spark.examples.SparkPi --executor-memory 2g --total-executor-cores 2 lib/spark-examples-1.0.0-hadoop2.2.0.jar 1000

课程回顾

- Spark 是一个大数据处理引擎（或者说是开发包），其核心是 Spark Core，基础是 RDD；
- Spark 应用程序有两部分组成： driver 和 executor
- Spark 应用程序可以在多种集群里运行： Mesos、YARN、Spark Standalone、AWS。。。
- Spark 应用程序的部署工具是 spark-submit
- Spark 的源码编译方法
  - Maven
  - Sbt
  - make-distribution
- Spark 应用程序之所以快不仅仅是由于基于内存计算，还和其工作原理相关
  - DAG
  - Schedule
  - Cache()
  - ...

Thanks

FAQ 时间