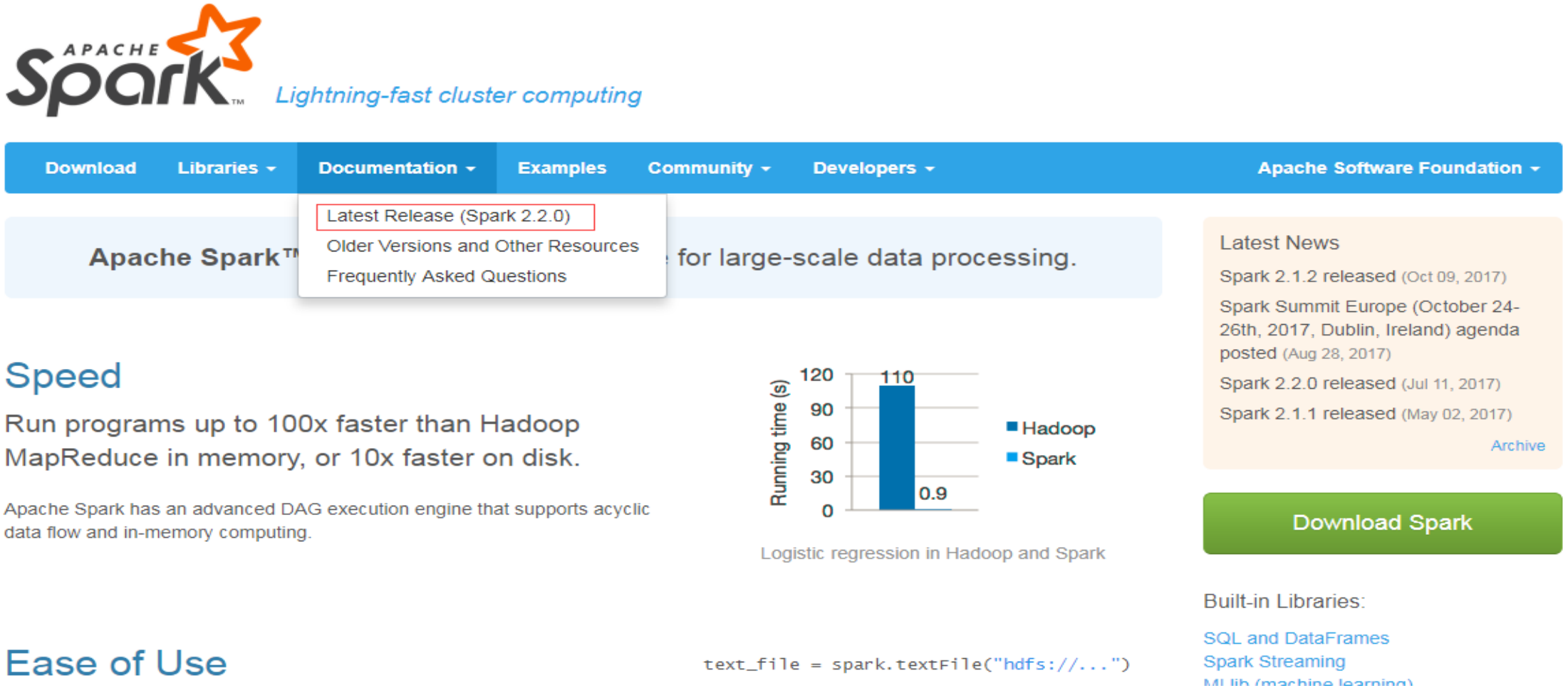
Spark简介

陶时

東北大學 Northeastern University
HORIZON 昊宸科技

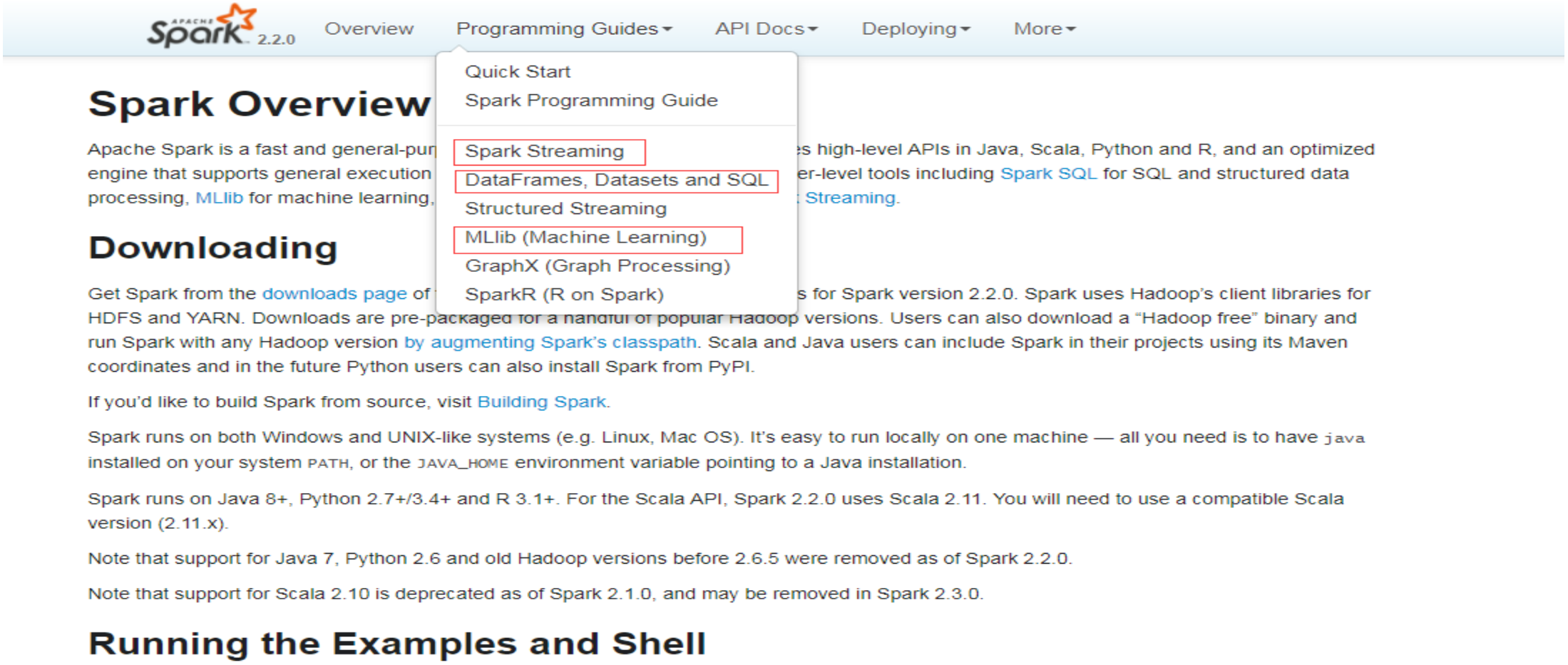
如何学习Spark

- 第一手资料 http://spark.apache.org/。



如何学习Spark

- 重点介绍Spark Core、SQL、MLlib



如何学习Spark

- 官网有丰富的例子，支持多种API、Python Scala Java R。

Starting Point: SparkSession

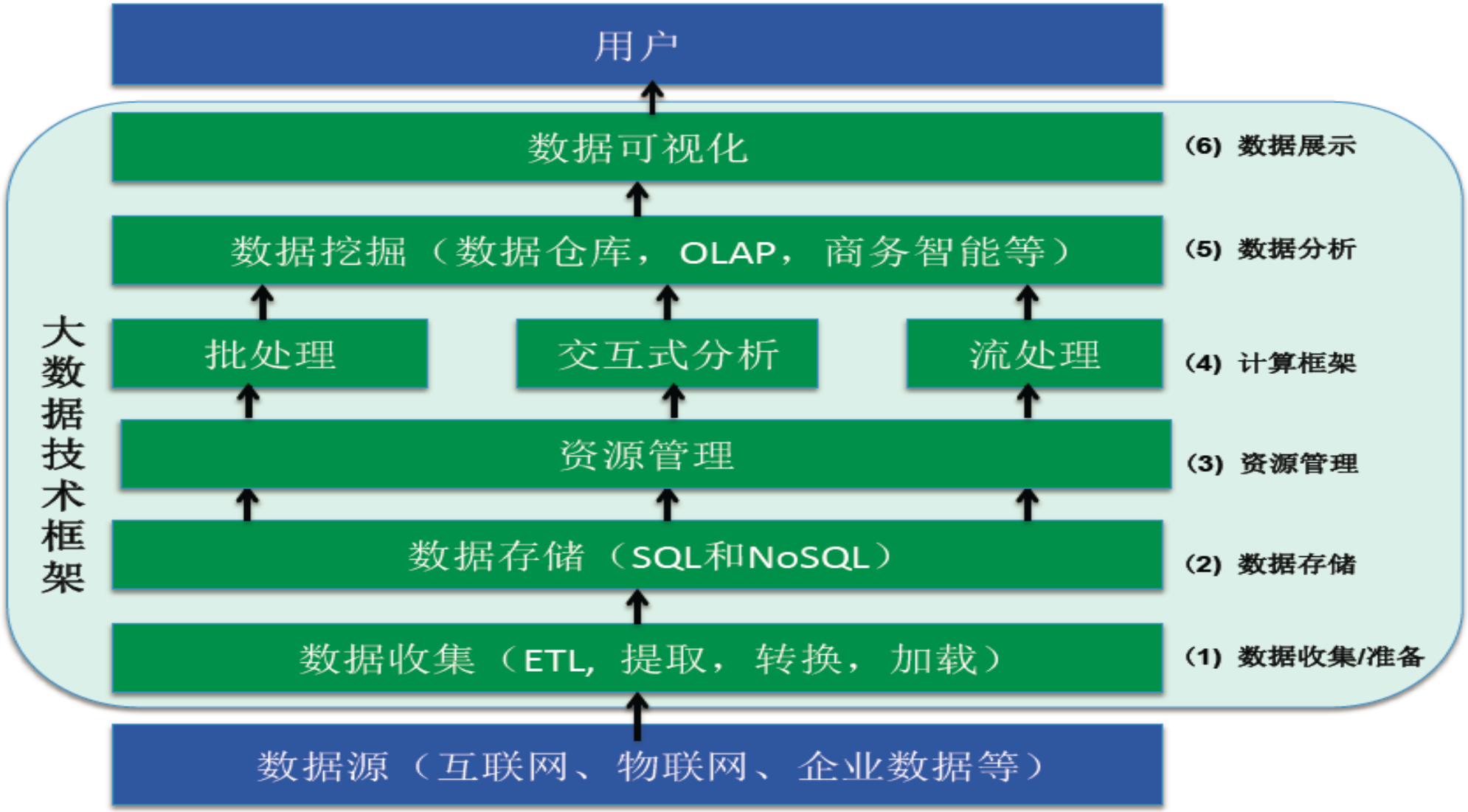
Scala | Java | Python | R

The entry point into all functionality in Spark is the `SparkSession` class. To create a basic `SparkSession`, just use `SparkSession.builder()`:

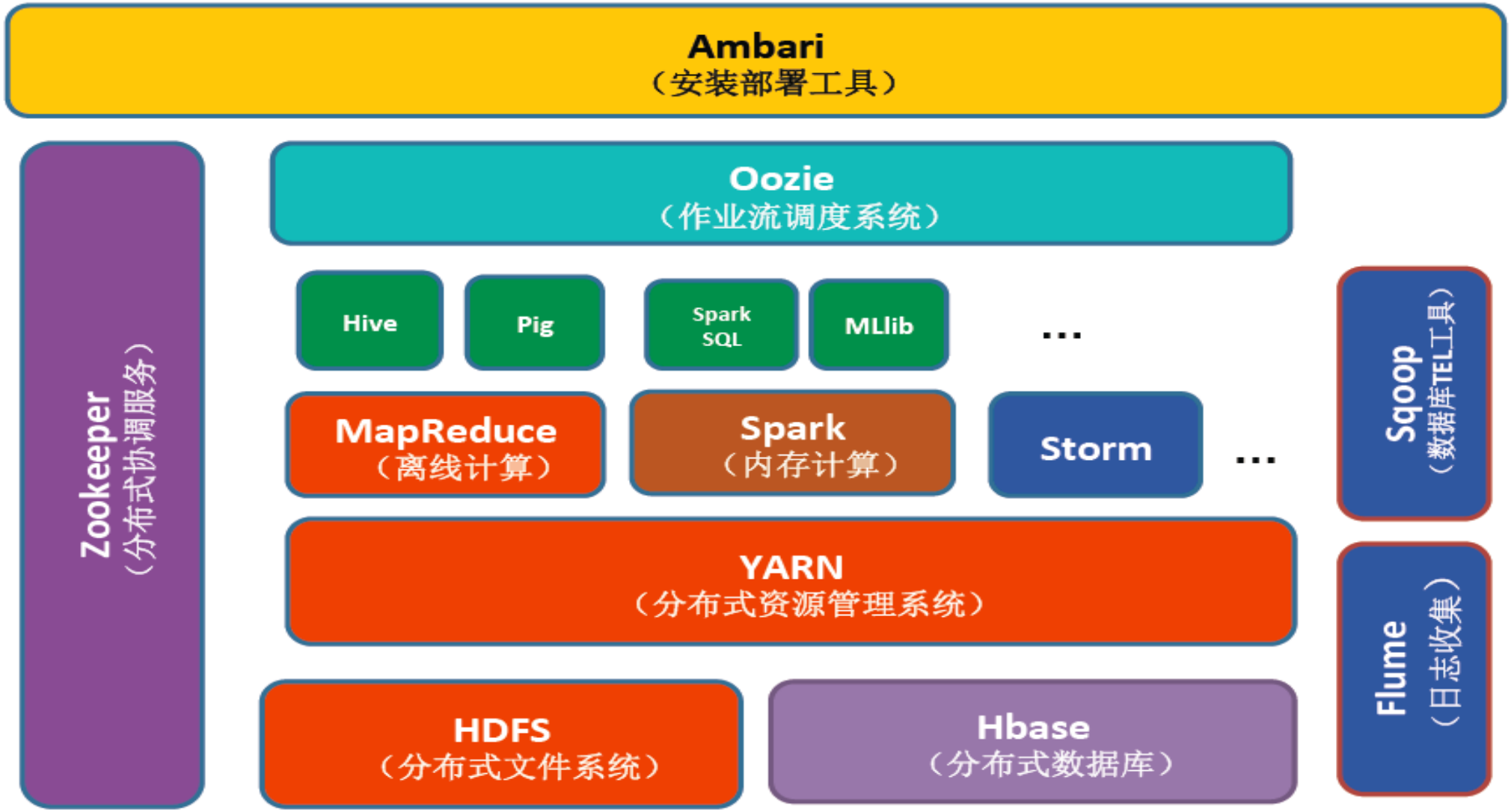
```
import org.apache.spark.sql.SparkSession

val spark = SparkSession
  .builder()
  .appName("Spark SQL basic example")
  .config("spark.some.config.option", "some-value")
  .getOrCreate()

// For implicit conversions like converting RDDs to DataFrames
import spark.implicits._
```

目录

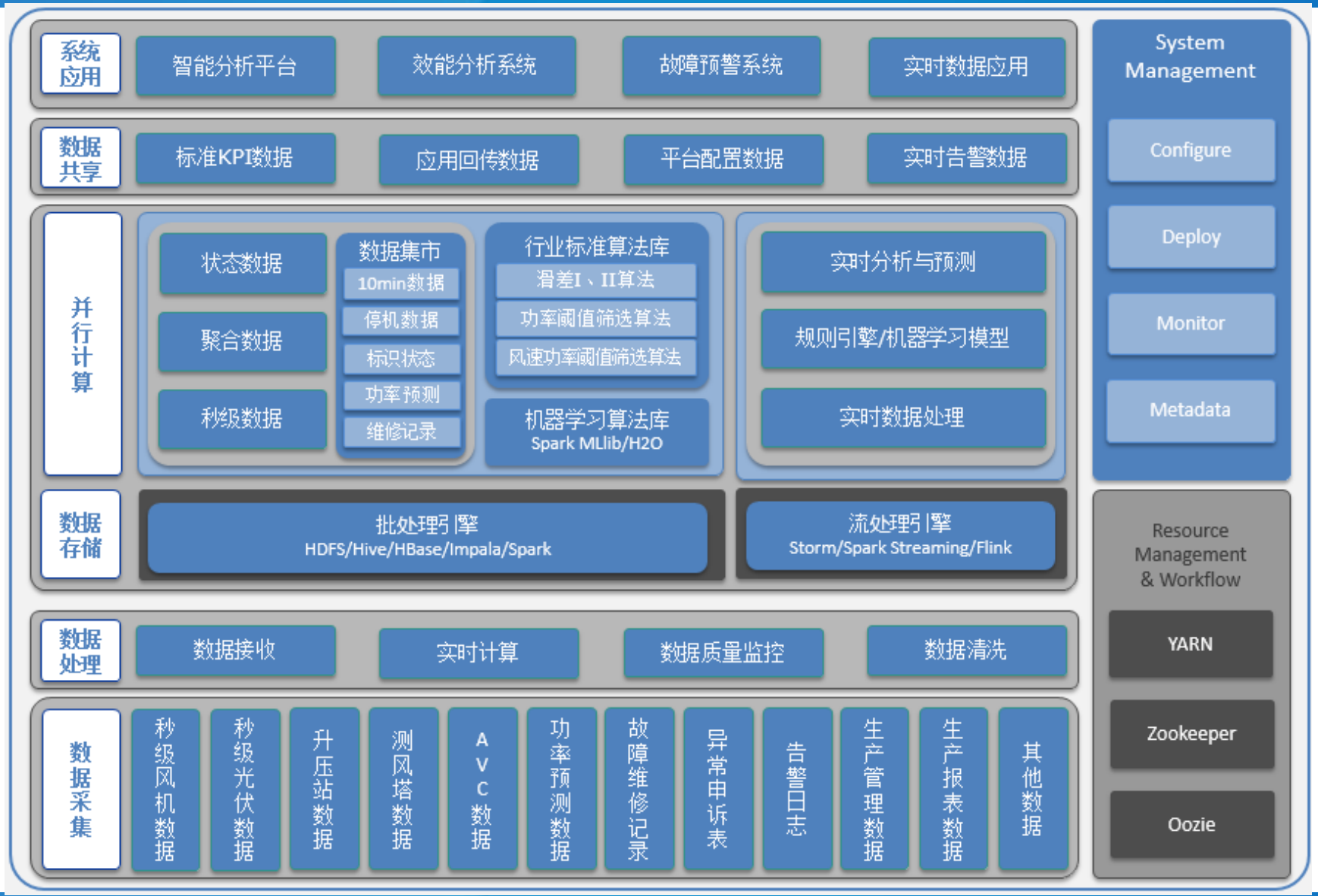
大数据技术框架


用户
数据可视化
(6) 数据展示
数据挖掘（数据仓库，OLAP，商务智能等）
(5) 数据分析
大数据技术框架
批处理
交互式分析
流处理
(4) 计算框架
资源管理
(3) 资源管理
数据存储（SQL和NoSQL）
(2) 数据存储
数据收集（ETL，提取，转换，加载）
(1) 数据收集/准备
数据源（互联网、物联网、企业数据等）

大数据技术框架


Ambari
（安装部署工具）
Zookeeper
（分布式协调服务）
Oozie
（作业流调度系统）
Hive
Pig
Spark
SQL
MLlib
...
MapReduce
（离线计算）
Spark
（内存计算）
Storm
...
Sqoop
（数据库TEL工具）
YARN
（分布式资源管理系统）
Flume
（日志收集）
HDFS
（分布式文件系统）
Hbase
（分布式数据库）

大数据技术框架


系统应用
智能分析平台
效能分析系统
故障预警系统
实时数据应用
数据共享
标准KPI数据
应用回传数据
平台配置数据
实时告警数据
并行计算
状态数据
聚合数据
秒级数据
数据集市
10min数据
停机数据
标识状态
功率预测
维修记录
行业标准算法库
滑差I、II算法
功率阈值筛选算法
风速功率阈值筛选算法
机器学习算法库
Spark MLlib/H2O
实时分析与预测
规则引擎/机器学习模型
实时数据处理
数据存储
批处理引擎
HDFS/Hive/HBase/Impala/Spark
流处理引擎
Storm/Spark Streaming/Flink
数据处理
数据接收
实时计算
数据质量监控
数据清洗
数据采集
秒级风机数据
秒级光伏数据
升压站数据
测风塔数据
AVC数据
功率预测数据
故障维修记录
异常申诉表
告警日志
生产管理数据
生产报表数据
其他数据
System Management
Configure
Deploy
Monitor
Metadata
Resource Management & Workflow
YARN
Zookeeper
Oozie

Spark背景：MapReduce的局限性

- 仅支持Map和Reduce操作(编程非常简单，但是实现功能的代码量大)。
- 处理效率低
  - ✓ Map中间结果写入到磁盘。多个MapReduce之间通过HDFS交换数据。
  - ✓ Map和Reduce端均需要排序。
  - ✓ 无法充分的利用内存。
  - ✓ 任务调度和启用的开销较大。
- 不适合迭代计算(例如：机器学习、图计算等)，交互式处理(数据挖掘)，流处理。
- 编程不够灵活。
- 不支持SQL语句，可以称其为分布式计算的汇编语言。(好在Hive的出现。)

Spark背景：计算框架多样化

➢现有的计算框架

- ✓ 批处理：MapReduce、Hive、Pig…
- ✓ 流式计算：Storm
- ✓ 交互式计算：Impala，Presto

➢是否能有一种灵活的框架可以同时进行批处理、流式计算、交互式计算？

- ✓ 2009年由Berkeley‘s AMPLab开始编写最初的源代码。

目前情况：**星星之火，已经燎原**

- ✓ 目前已经有30+公司100+开发者在提交代码
- ✓ Hadoop最大的厂商Cloudera宣称加大Spark框架的投入来取代Mapreduce
- ✓ Hortonworks的HDP也集成了Spark。
- ✓ Hadoop厂商MapR投入Spark阵营
- ✓ Spark2.2.0与2017年7月发布。

Spark 的版本变迁

- Spark 2.2 是当前版本
- Spark 正式开源是在2010年
- Spark 1.0.0 版本发布于2014-05-30 增加了对Yarn集群的支持
- Spark 1.4.1 DataFrame API
- Spark 1.5.1 增加神经网络算法
- Spark 1.6.3 Spark 1最后一个版本，1.6的稳定版
- Spark 2.0 改变Spark context 定义方式，提倡使用ML 和 DataSet

计算类型及应用场景

- 批处理计算

✓对时间没有严格的要求，吞吐率高。

- 迭代式于DAG计算

✓机器学习算法

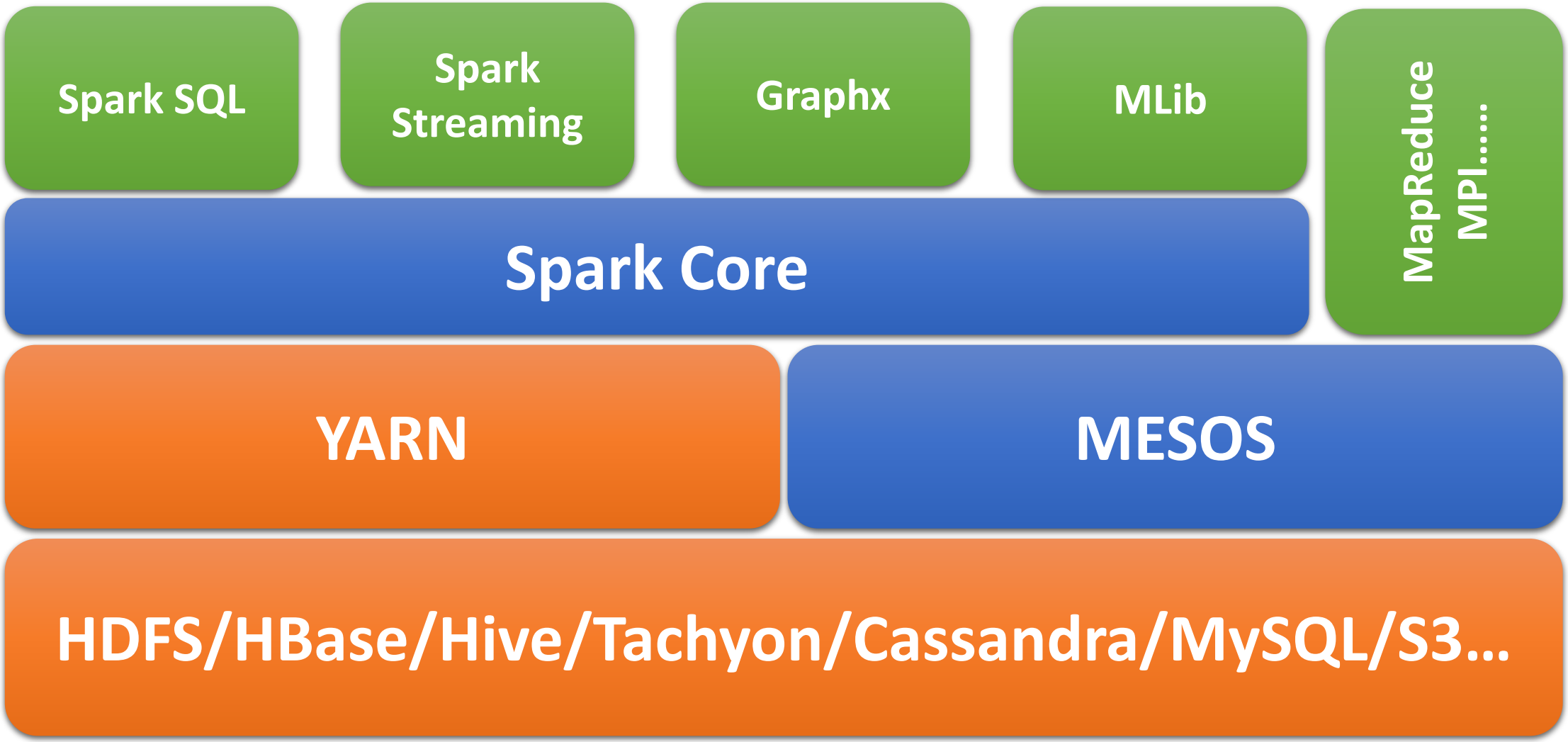
- 交互式计算

✓支持类SQL语言，快速进行数据分析。

- 流式计算

✓数据像流水一样流入进来，需实时对其处理和分析。

Spark生态系统


Spark SQL
Spark Streaming
Graphx
MLib
MapReduce
MPI.....
Spark Core
YARN
MESOS
HDFS/HBase/Hive/Tachyon/Cassandra/MySQL/S3…

Spark特性

➢速度(比MapReduce快10~100倍)

- ✓ 内存计算引擎，提供了Cache机制来支持需要反复迭代计算或者多次数据共享，减少数据读取的IO开销。
- ✓ DAG引擎，减少多次计算之间中间结果写入到HDFS的开销。
- ✓ 使用多线程池来减少task启动开销，shuffle过程中避免了不必要的sort操作和减少磁盘IO操作。

➢易用

- ✓ 丰富的API。支持多种语言(JAVA、Scala、Python、R)。

➢通用

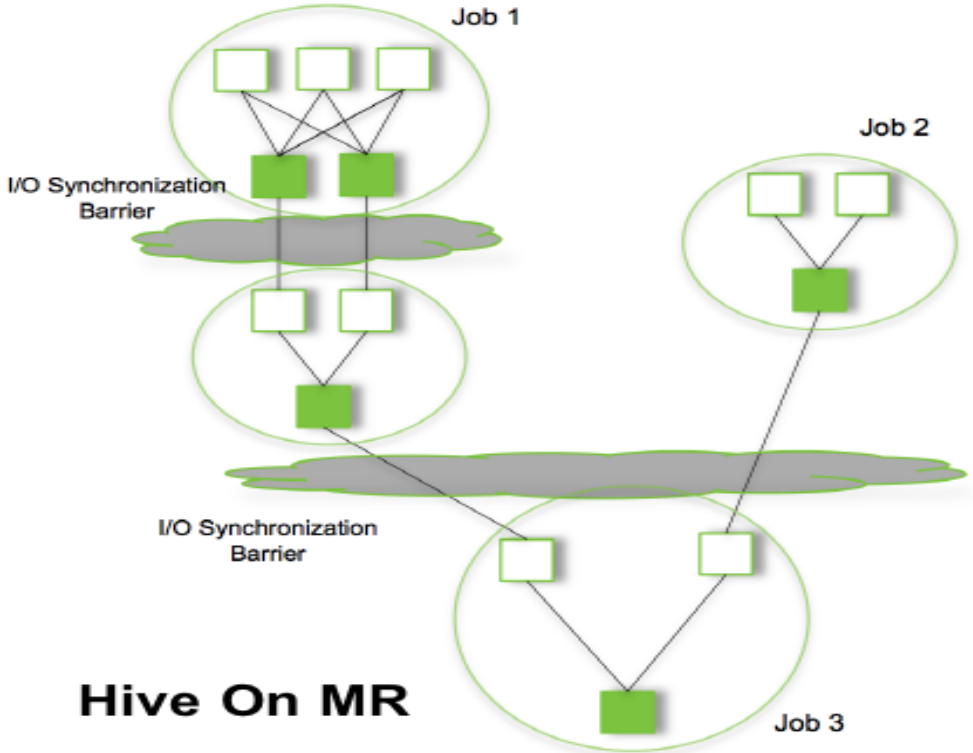
- ✓ Spark涵盖了批处理、流式计算、交互式计算、图计算、机器学习。这些计算都可以用一个框架(Spark)解决。

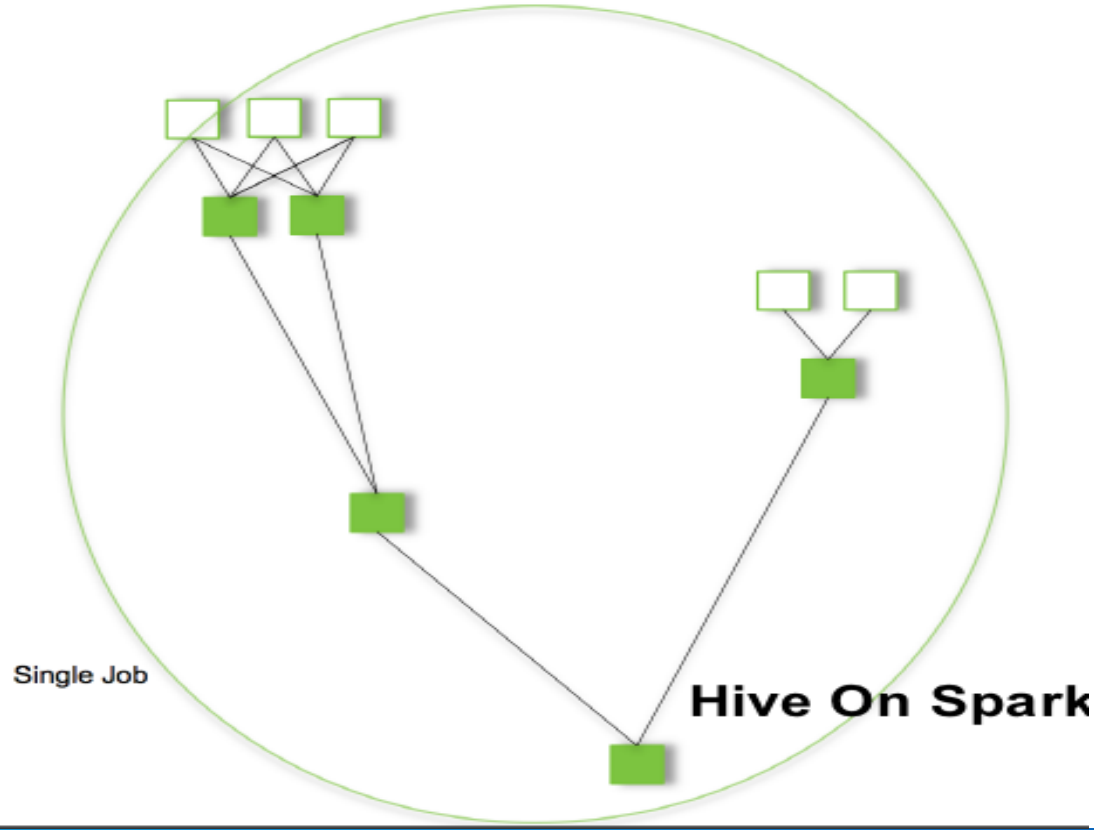
➢ 运行环境多样化

- ✓ Spark于Hadoop有了很好的集成，支持多种数据源，可以读写HDFS/HBase/Hive等的数据。
- ✓ Spark和YARN集成，可以运行在YARN上。
- ✓ Spark可以运行在Mesos上。
- ✓ Spark也可以脱离Hadoop等独立运行。

Spark特性-DAG

```
SELECT a.state, COUNT(*), AVERAGE(c.price)
FROM a
JOIN b ON (a.id = b.id)
JOIN c ON (a.itemId = c.itemId)
GROUP BY a.state
```



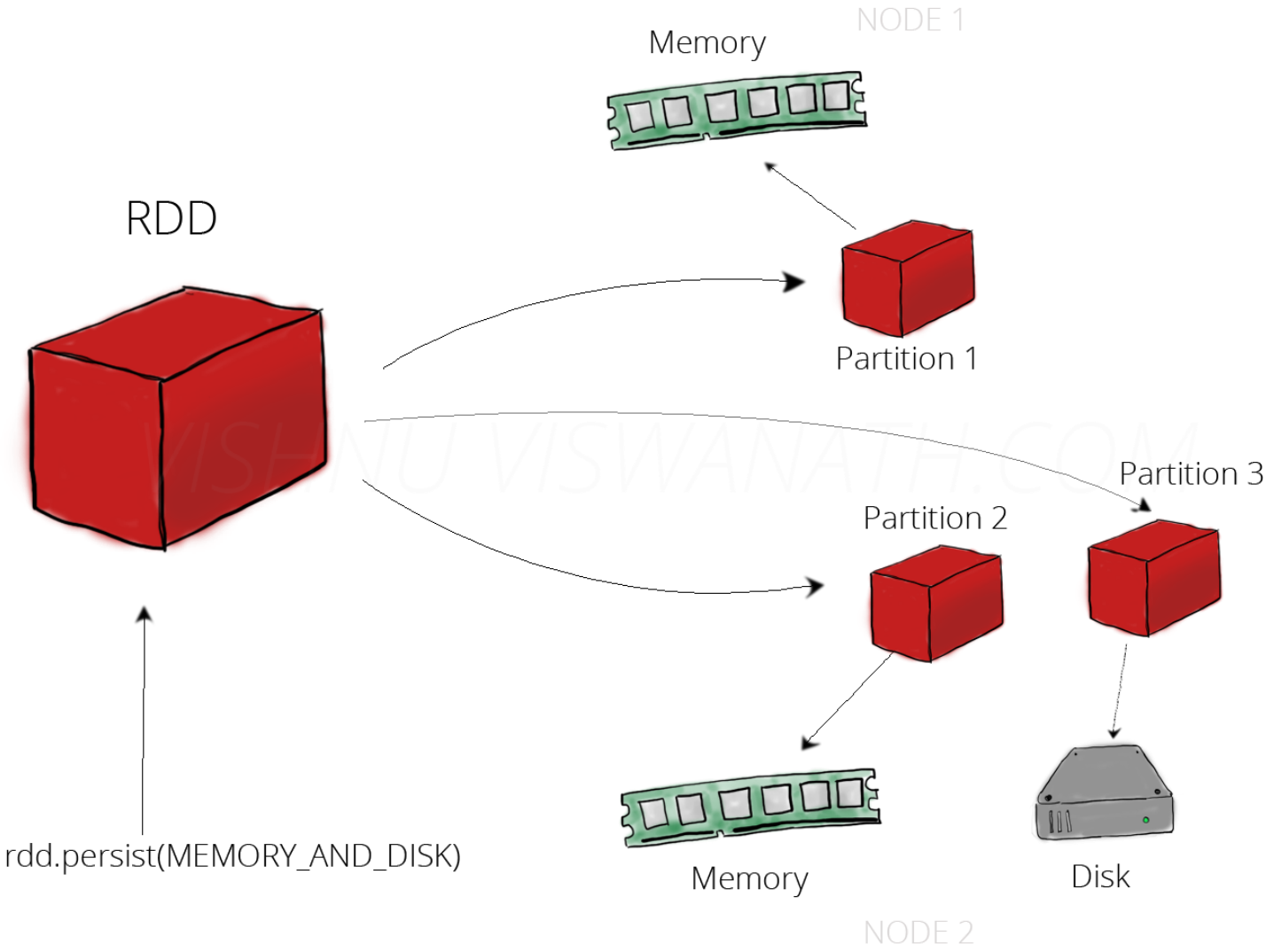


Spark核心概念-RDD

- RDD:弹性分布式数据集(Resilient Distributed Datasets)
  - ✓ RDD是Spark中计算和数据的抽象，它标识已经分片(partition),不可变的并能够被并行计算的数据集合。
  - ✓ RDD可以被存储在磁盘中也可以被存储到内存中。
  - ✓ RDD提供给我们很多方便的数据变换方法，这些变换操作分为两种类型(Transformation和Action)
  - ✓ RDD的生成方式有两种：数据源读入，其他RDD通过Transformation操作转换。
  - ✓ RDD失败后自动重构。

Spark核心概念-RDD

- 一个RDD由多个Partition构成(计算时一个partition对应一个task)。
- 每个Partion数据可以存储在内存中也可以存储在磁盘中(用户可控制)。
- RDD的操作分为Trasformation和Action两种操作。



Spark核心概念-RDD的基本操作

- Transformation

把一个RDD转换为另外一个RDD。

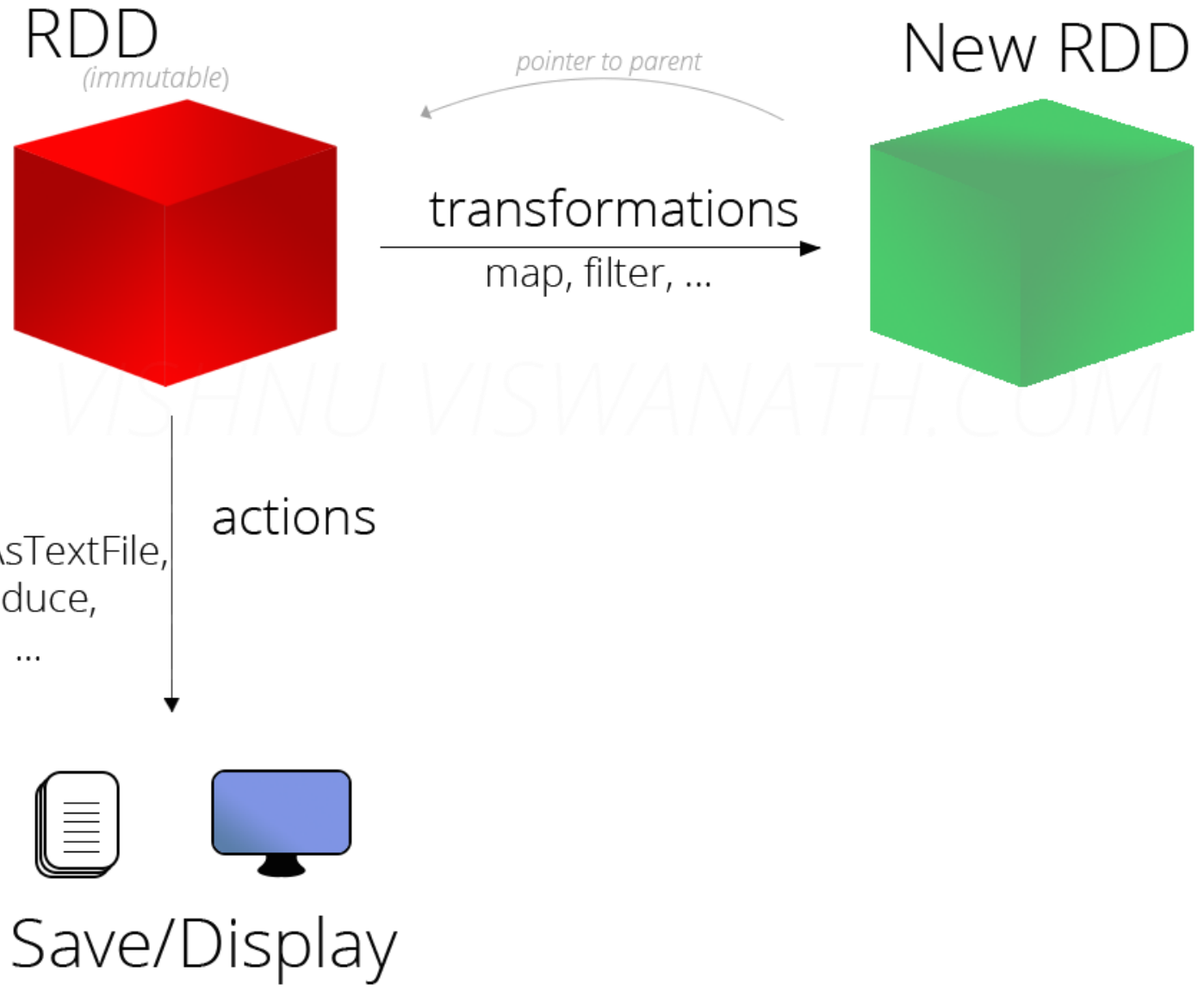
例如：map,filter,groupBy,reduceBy

等。

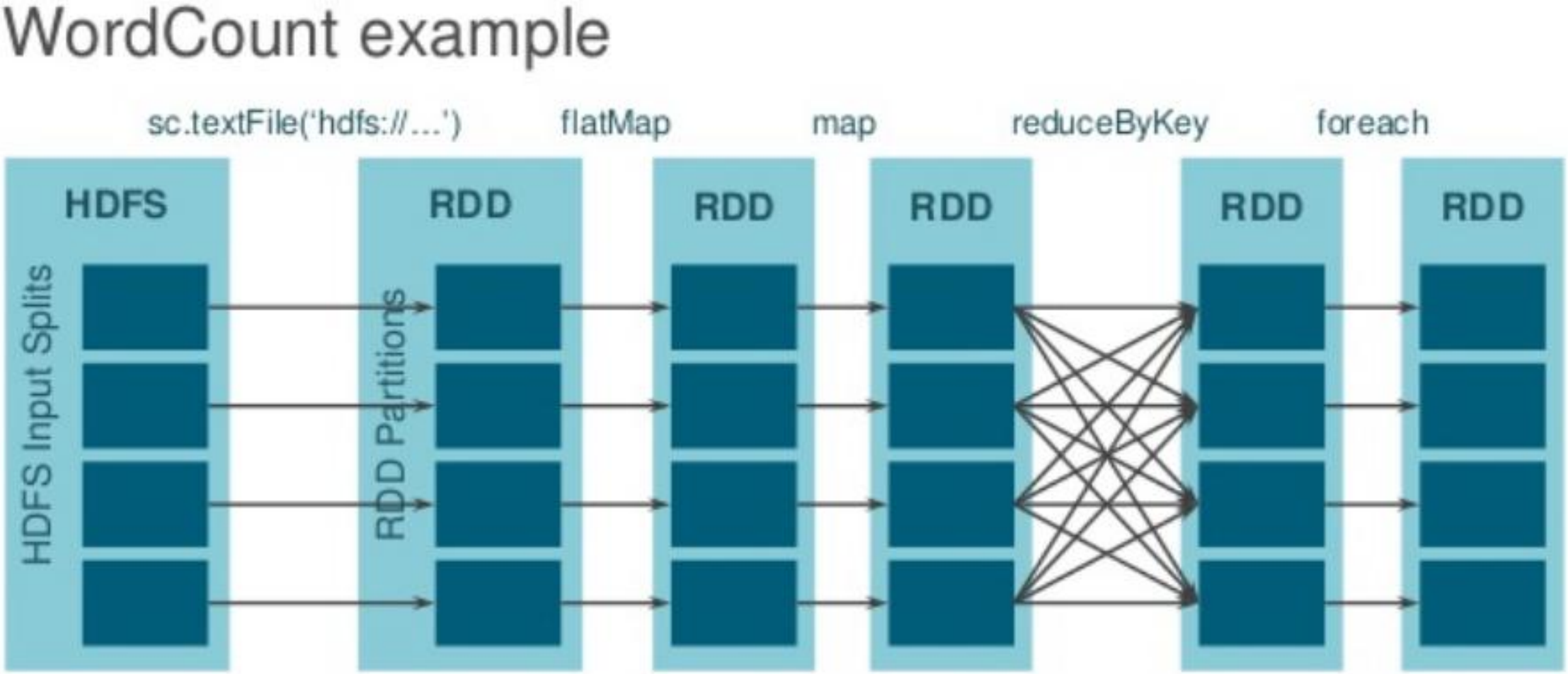
- Action

通过RDD计算得到一个或者一组值。

例如：count



Spark核心概念-RDD的基本操作



Spark核心概念-RDD的缓存位置

- RDD既可以存储为内存中也可以存储在磁盘里面。Spark为我们提供了一下几种方案。

存储级别	描述
MEMORY_ONLY	将RDD 作为反序列化的的对象存储JVM 中。如果RDD不能被内存装下，一些分区将不会被缓存，并且在需要的时候被重新计算。 这是是默认的级别
MEMORY_AND_DISK	将RDD 作为反序列化的的对象存储在JVM 中。如果RDD不能被与内存装下，超出的分区将被保存在硬盘上，并且在需要时被读取
MEMORY_ONLY_SER	将RDD 作为序列化的的对象进行存储（每一分区占用一个字节数组）。 通常来说，这比将对象反序列化的空间利用率更高，尤其当使用fast serializer,但在读取时会比较占用CPU
MEMORY_AND_DISK_SER	与MEMORY_ONLY_SER 相似，但是把超出内存的分区将存储在硬盘上而不是在每次需要的时候重新计算
DISK_ONLY	只将RDD 分区存储在硬盘上
MEMORY_ONLY_2	与上述的存储级别一样，但是将每一个分区都复制到两个集群结点上

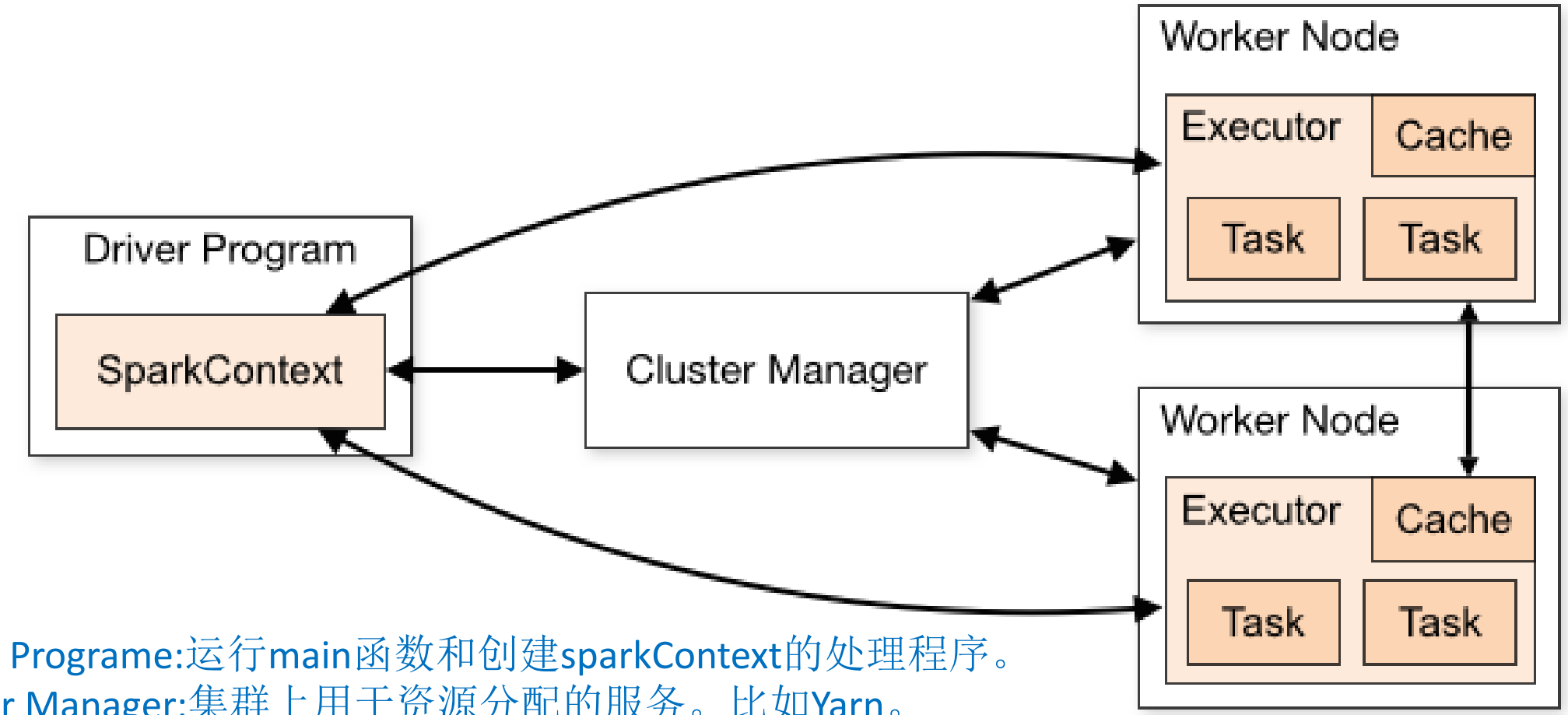
Spark核心概念-RDD的缓存位置

- 用户可以根据RDD提供的方法选择缓存方案。

```
JavaRDD<String> data = jsc.textFile("/data/external/dataCompletion/2017-05-11/cp_86.1000.11_20170605.csv");
data.cache();
data.persist(StorageLevel.MEMORY_AND_DISK());
```

```
val NONE = new StorageLevel(false, false, false, false)
val DISK_ONLY = new StorageLevel(true, false, false, false)
val DISK_ONLY_2 = new StorageLevel(true, false, false, false, 2)
val MEMORY_ONLY = new StorageLevel(false, true, false, true)
val MEMORY_ONLY_2 = new StorageLevel(false, true, false, true, 2)
val MEMORY_ONLY_SER = new StorageLevel(false, true, false, false)
val MEMORY_ONLY_SER_2 = new StorageLevel(false, true, false, false, 2)
val MEMORY_AND_DISK = new StorageLevel(true, true, false, true)
val MEMORY_AND_DISK_2 = new StorageLevel(true, true, false, true, 2)
val MEMORY_AND_DISK_SER = new StorageLevel(true, true, false, false)
val MEMORY_AND_DISK_SER_2 = new StorageLevel(true, true, false, false, 2)
val OFF_HEAP = new StorageLevel(false, false, true, false)
```

Spark核心概念-spark组件

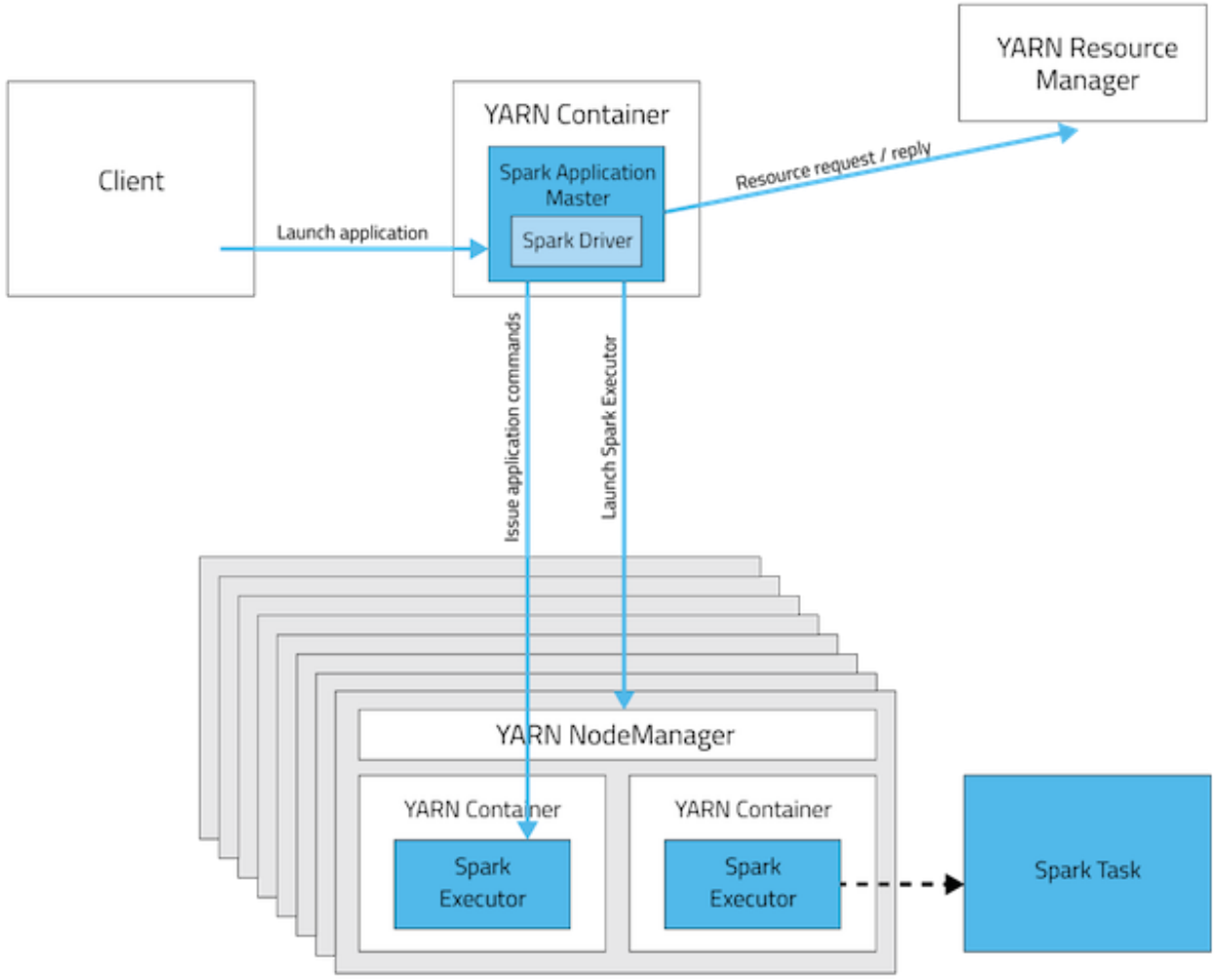


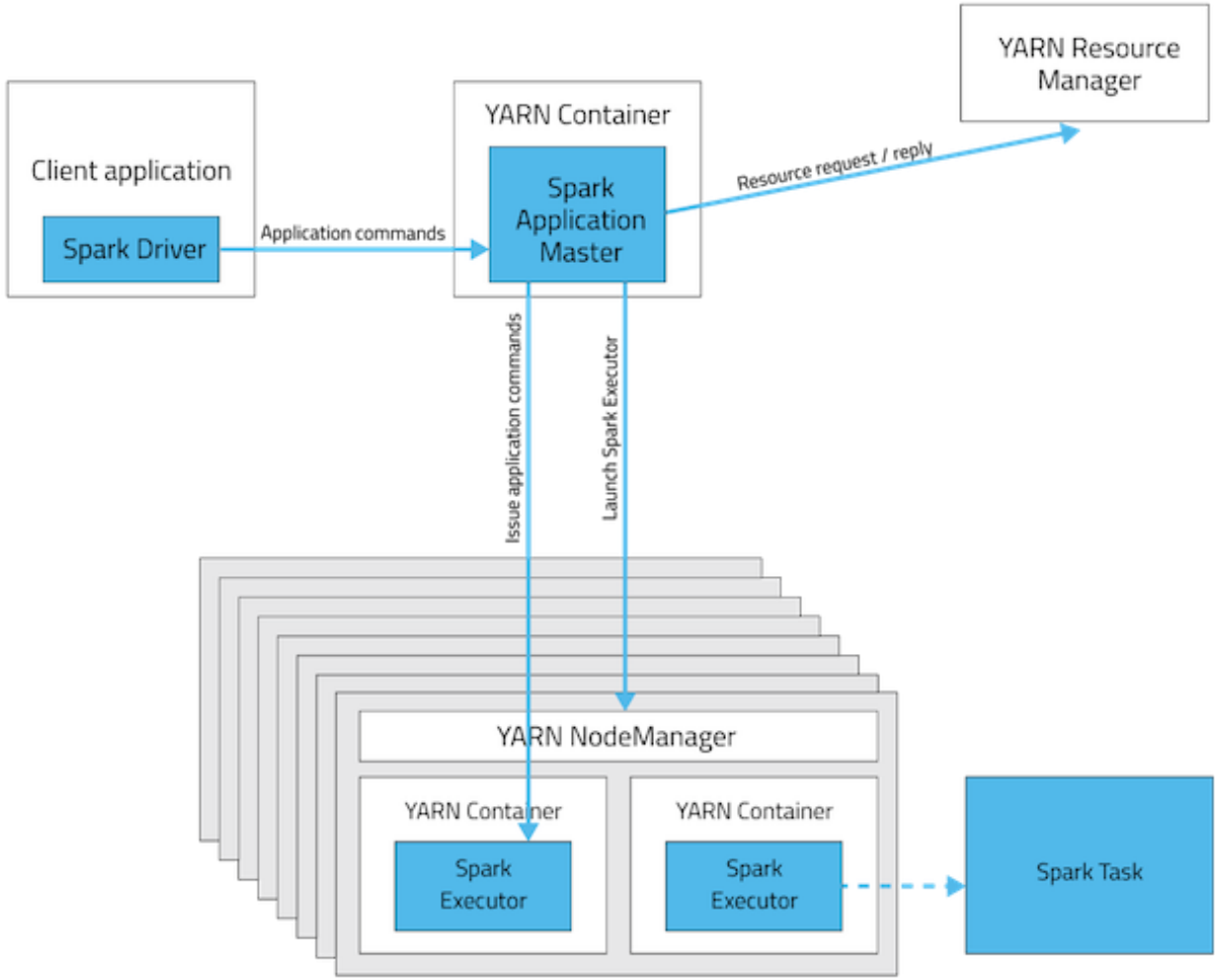
- Driver Programe:运行main函数和创建sparkContext的处理程序。
- Cluster Manager:集群上用于资源分配的服务。比如Yarn。
- Worker Node:运行spark程序的节点。
- Executor:每个应用程序都有自己的executor。每个excutor包含多个task.

Spark的运行模式

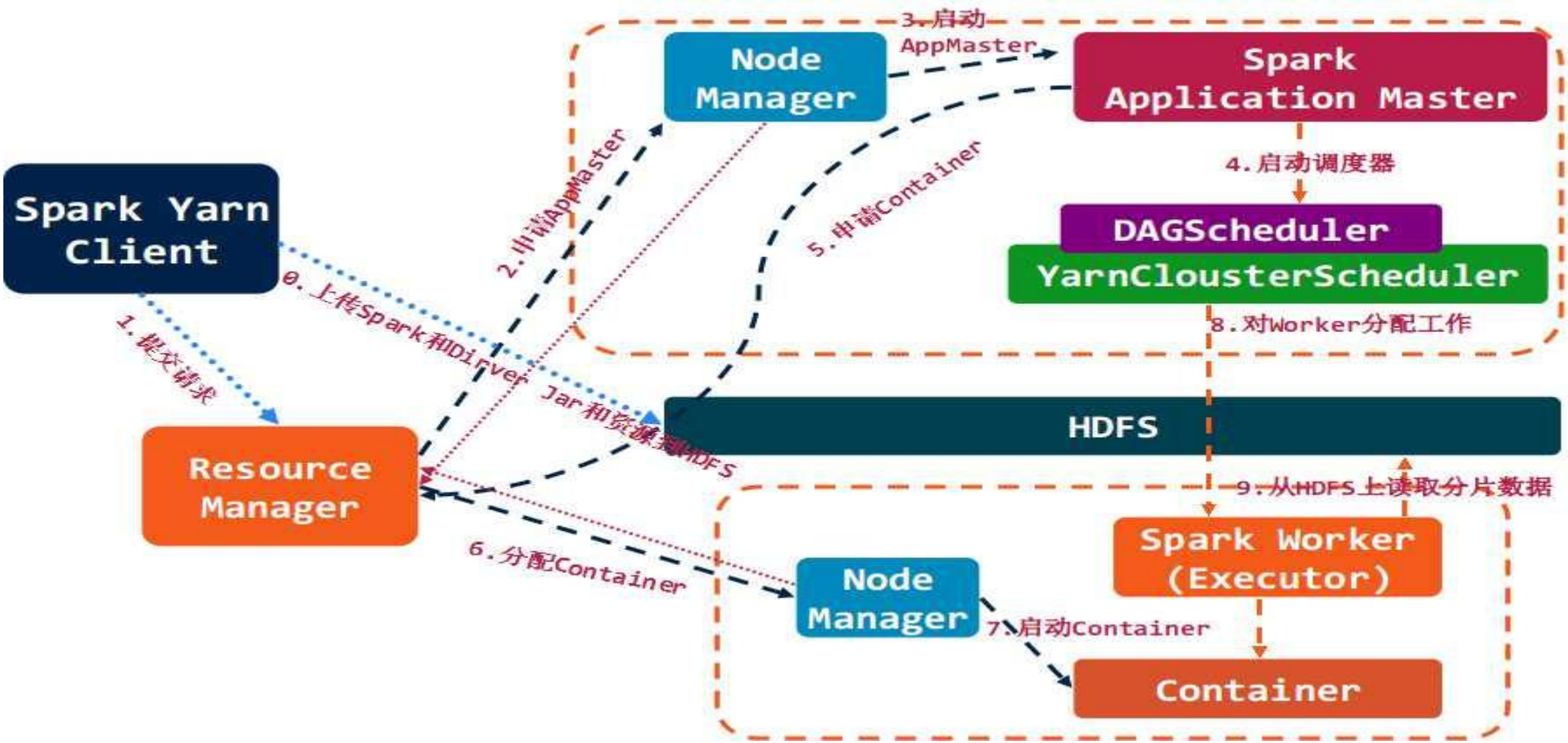
- **Local模式**
  - ✓ 单机运行，通常用于测试。
- **Standalone模式**
  - ✓ 独立运行在一个spark的集群中。
- **Spark on Yarn/Mesos模式**
  - ✓ Spark程序运行在资源管理器上，例如YARN/Mesos
  - ✓ Spark on Yarn存在两种模式
    - yarn-client
    - yarn-cluster

Spark的运行模式


Client
Launch application
YARN Container
Spark Application Master
Spark Driver
Resource request / reply
YARN Resource Manager
Issue application commands
Launch Spark Executor
YARN NodeManager
YARN Container
Spark Executor
YARN Container
Spark Executor
Spark Task


Client application
Spark Driver
Application commands
YARN Container
Spark Application Master
Resource request / reply
YARN Resource Manager
Issue application commands
Launch Spark Executor
YARN NodeManager
YARN Container
Spark Executor
YARN Container
Spark Executor
Spark Task

Spark on YARN



Spark安装

1.官网下载spark包。

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

2.将spark安装包上传至集群，并解压。以下操作二选一。

3.部署standalone模式的spark集群。

修改conf/slaves文件，添加spark的各个节点ip地址。

4.Spark on yarn模式。

添加环境变量：HADOOP_CONF_DIR=hadoop配置文件所在路径。

Spark 编程

1.构建工具 Maven 和 SBT。

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

2.JDK 1.8　Scala 2.11.11　Spark 2.2.0

　JDK 1.7　Scala 2.10.x　　Spark 1.6.3

Spark WorkCount实例

```
import org.apache.spark._
import SparkContext._
var sparkConf = new SparkConf().setAppName(“wordCount“)
var sc = new SparkContext(sparkConf)
val lines = sc.textFile("/data/external/dataCompletion/2017-05-11/cp_86.1000.11_20170605.csv")
lines.flatMap(_.split(" ")).map((_, 1)).reduceByKey(_+_).collect.foreach(println)
```

- 惰性执行

Transformation只会记录RDD的转换关系，并不会触发计算。

Action会触发程序的执行(分布式)的算子。

Spark 提交任务

```
./bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

参照:http://spark.apache.org/docs/latest/submitting-applications.html

Spark-shell

- 启动spark-shell

```
./bin/spark-shell --master local[4]
./bin/spark-shell --master spark://idh104:7077
./bin/spark-shell --master yarn-client
```

- 在spark-shell中写测试代码

```
val lines = sc.textFile("/test/wordCount.txt")
lines.count()
lines.first()
lines.take(10)
val linesWithfilter = lines.filter(line => line.contains(“the"))
linesWithfilter.count()
```

Spark-UI

Spark 1.6.2 | Jobs | Stages | Storage | Environment | Executors | SQL | **Spark shell** application UI

Spark Jobs (?)

Total Uptime: 9.5 min
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

Completed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <console>:30	2017/06/09 18:09:16	0.6 s	2/2	4/4

Spark 1.6.2 | Jobs | Stages | Storage | Environment | Executors | SQL | **Spark shell** application U

Details for Job 0

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▸ DAG Visualization

Completed Stages (2)

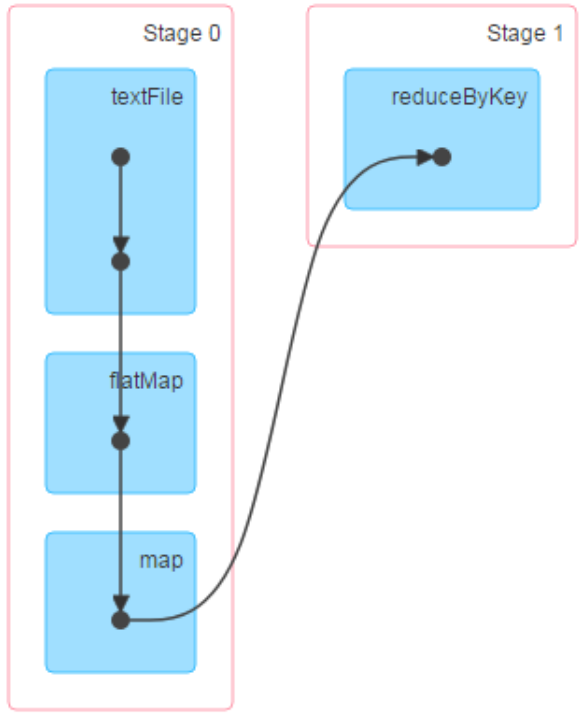
Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	collect at <console>:30 +details	2017/06/09 18:09:16	55 ms	2/2			16.8 KB	
0	map at <console>:30 +details	2017/06/09 18:09:16	0.4 s	2/2	126.5 KB			16.8 KB

Spark 1.6.2 | Jobs | Stages | St

Details for Job 0

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▾ DAG Visualization



作业

1.安装spark，并启动spark-shell；分别用local/standalone/yarn模式运行workcount。

1)截取spark-UI执行进度。

2)截取执行成功后输出的结果。

3)Spark on yarn模式，截取8088端口页面的截图。

2.请对Spark的RDD做简要的概述。

3.请对Saprk的Transformation和Action做简要描述，以及spark的懒执行是什么？

THANKS