Spark+R大数据分析入门

李想

安正软件

公司需求

“资管行业在电商、客户画像、金融指标计算等方面存在着大量分布式计算业务需要，为更好的满足客户需求，提升我们的核心竞争力，需要在公司启动对分布式集群技术的研究并形成方案、实施指南。”

“搭建分布式集群需要确定平台最佳架构、搭建方案以及基于平台进行客户画像(Spark)、金融指标（Spark +R）等指标开发，并可以通过指标管理平台发布形成服务接口。”

“近期大数据应用包括：客户盈亏计算、客户标签计算、大数据量汇总、金融风控指标计算，远期应用包括：非结构化文本解析（比如：产品合同要素解析）、话务流水文本解析、网站点击日志解析、客户行为分析等。”

主要内容

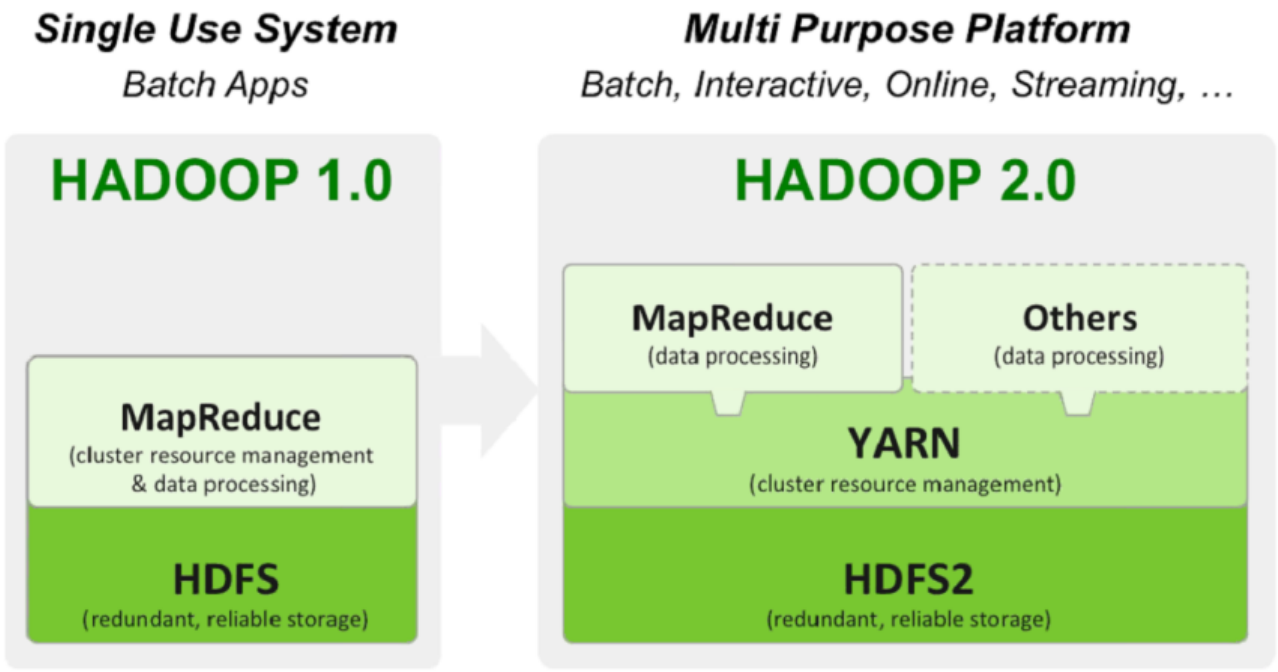
- Hadoop与Spark框架和生态简介
- Hadoop与Spark核心原理介绍
- SparkR与Sparklyr交互式分析
- spark-submit提交作业

开源分布式文件存储及处理框架

Hadoop特点

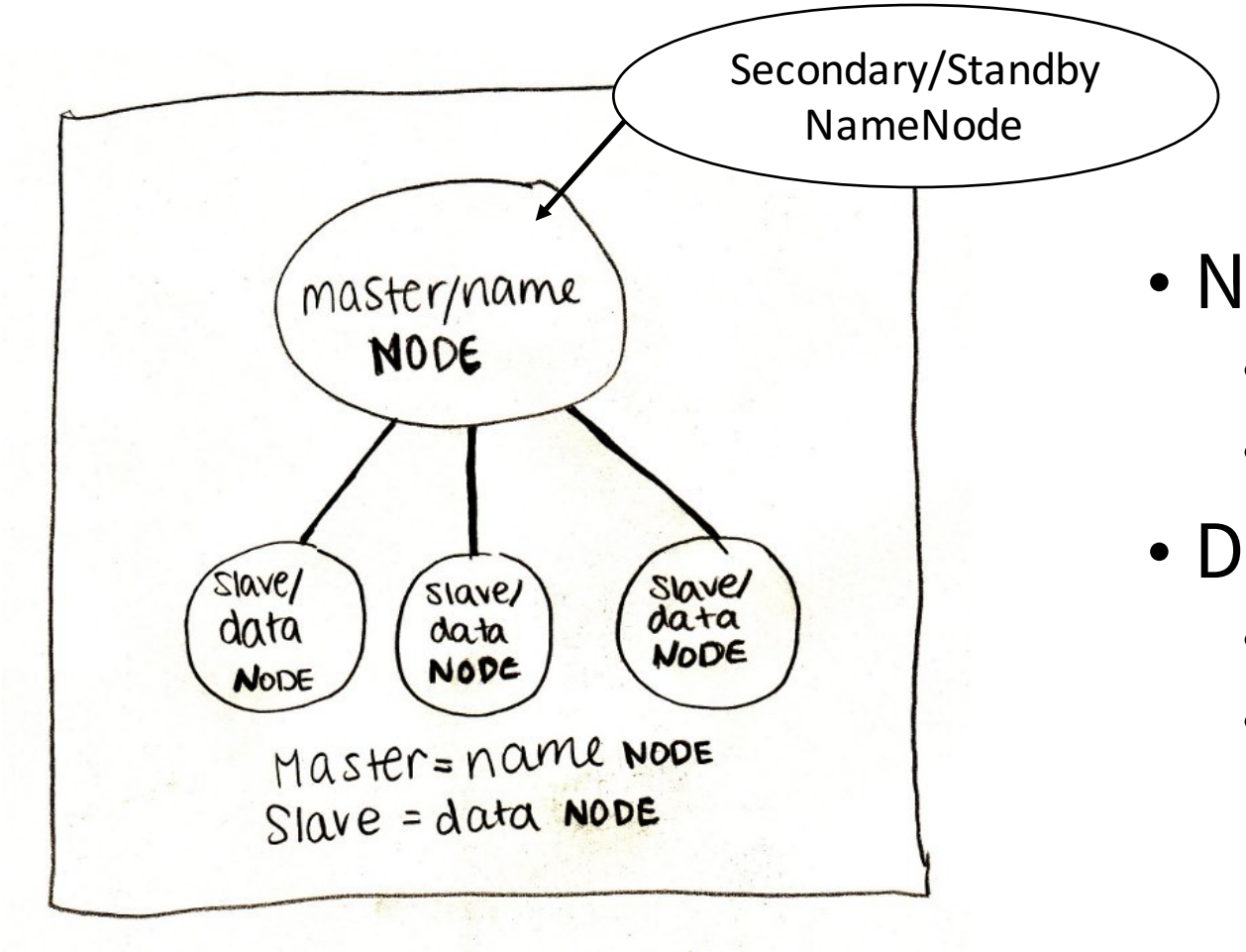
- 高可靠、高容错、高可用：多个数据副本，备用节点；
- 高效：共享集群，并行处理
- 可伸缩：处理海量数据，最多支持10000个节点
- 低成本：商用型服务器，资源利用率高

Hadoop框架



- Hadoop Common
- Hadoop HDFS
- Hadoop YARN
- Hadoop MapReduce

HDFS架构

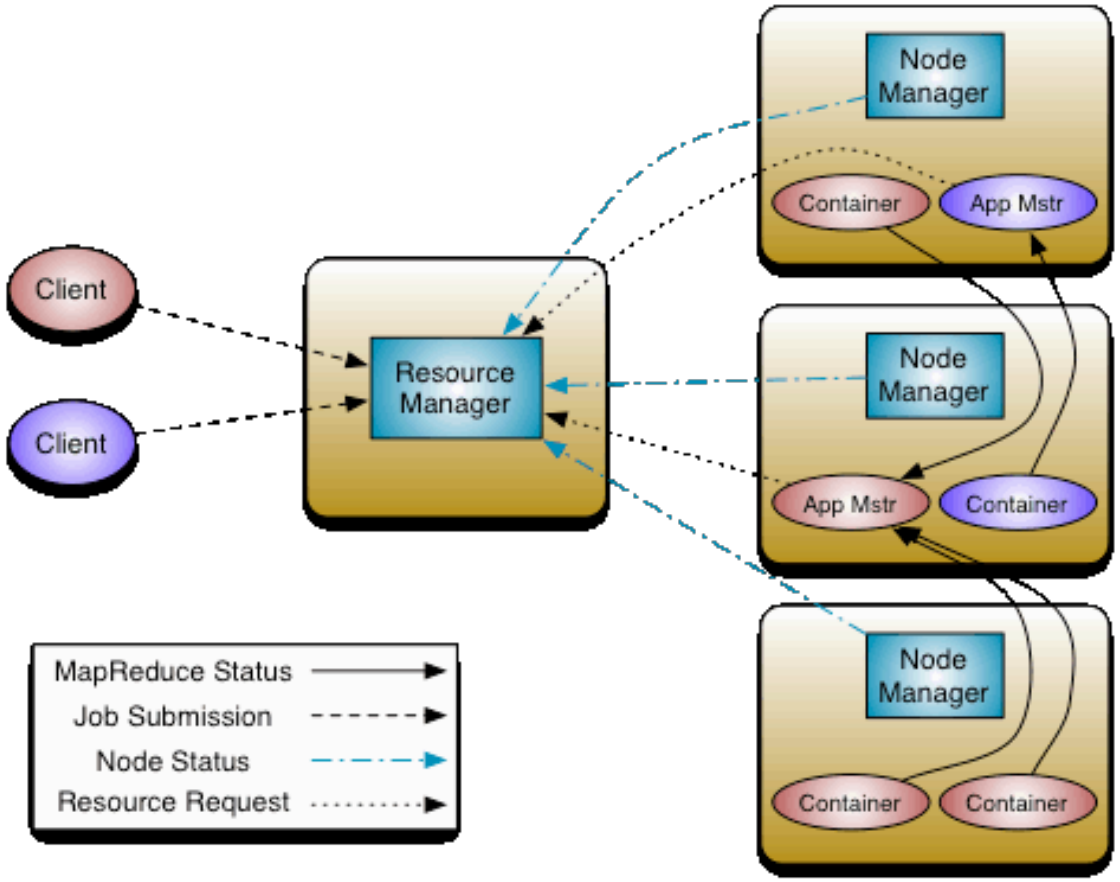


- NameNode
  - Secondary NameNode
  - Standby NameNode
- DataNode
  - Replication 建立数据副本
  - Write once, read often

HDFS架构

- **NameNode**：元数据、文件系统目录、文件位置；
- **Secondary / Standby NameNode**：更新或断点检查，高可用；
- **DataNode**：存储数据区块，数据副本（容错）。

YARN架构

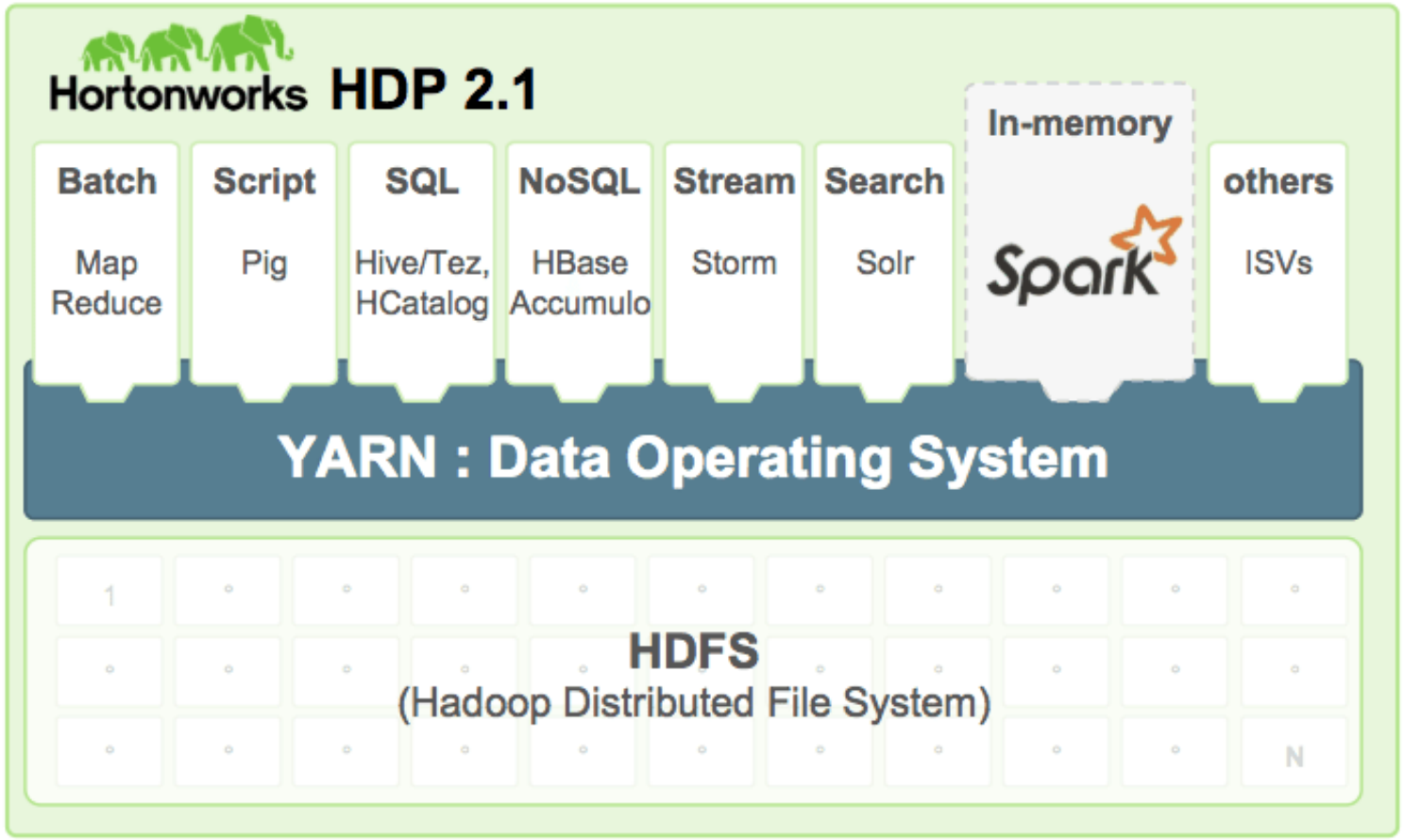


- ResourceManager
  - Application Manager
  - Scheduler
- NodeManager
- Container
- ApplicationMaster

YARN架构

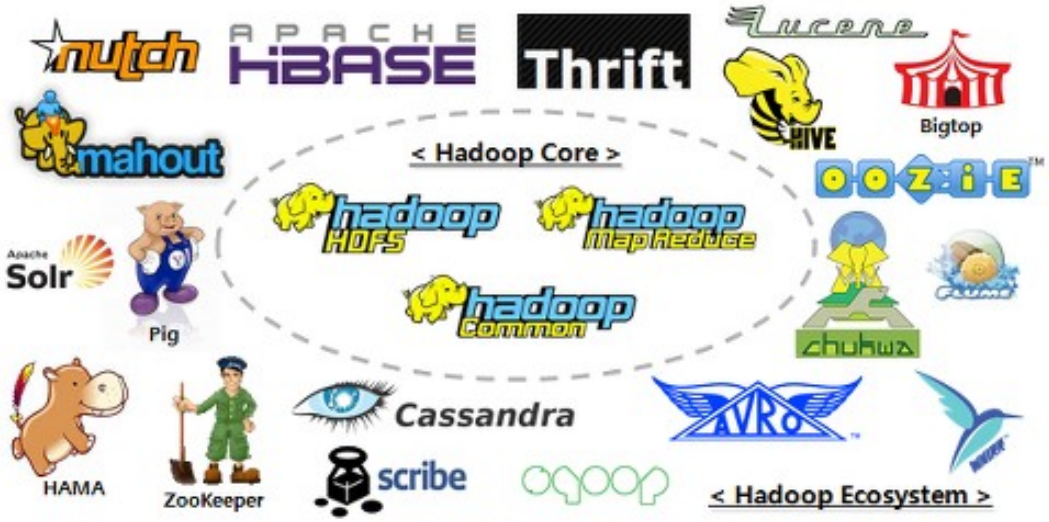
- **ResourceManager**：配置集群资源、作业调度；
- **ApplicationMaster**：向RM请求资源，管理单个作业；
- **NodeManager**：运行和管理工作节点上的任务，与RM保持联系；
- **Container**：资源抽象，封装了某个节点上的多维度资源。

Hadoop庞大的生态



Hadoop庞大的生态

- Ambari™: 基于Web的配置、管理、监控Hadoop集群的工具。对包括Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop等提供支持。Ambari提供了可供观察集群运行状态的仪表盘。
- Avro™: 数据序列化系统。
- Cassandra™: 不存在单点故障的可伸缩的多主节点数据库。
- Chukwa™: 用于管理大型分布式系统的数据集系统。
- HBase™: 可伸缩的分布式数据库，支持大型表的结构化存储。
- Hive™: 数据仓库基础设施，提供数据汇总和临时查询（ad hoc querying）。
- Mahout™: 可伸缩的机器学习和数据挖掘库。
- Pig™: 高级（high-level）数据流语言和分布式计算执行框架。
- **Spark™**: 通用高速计算引擎，提供了简单的和表达式的编程模型，支持广泛的应用，包括ETL，机器学习，流式处理，图计算等功能。
- Tez™: 通用数据流编程框架。
- ZooKeeper™: 对分布式应用程序提供高性能协作服务。
- Storm™: 实时/流式数据处理。
- ……



MRv1缺点

- 扩展性差：资源管理和作业控制均由JobTracker负责
- 可靠性差：单点故障
- 资源利用率低：有的槽位资源紧张，有的槽位资源闲置
- 无法支持多种计算框架：内存计算框架（Spark）、流式计算框架（S4）、迭代式计算框架（Spark）

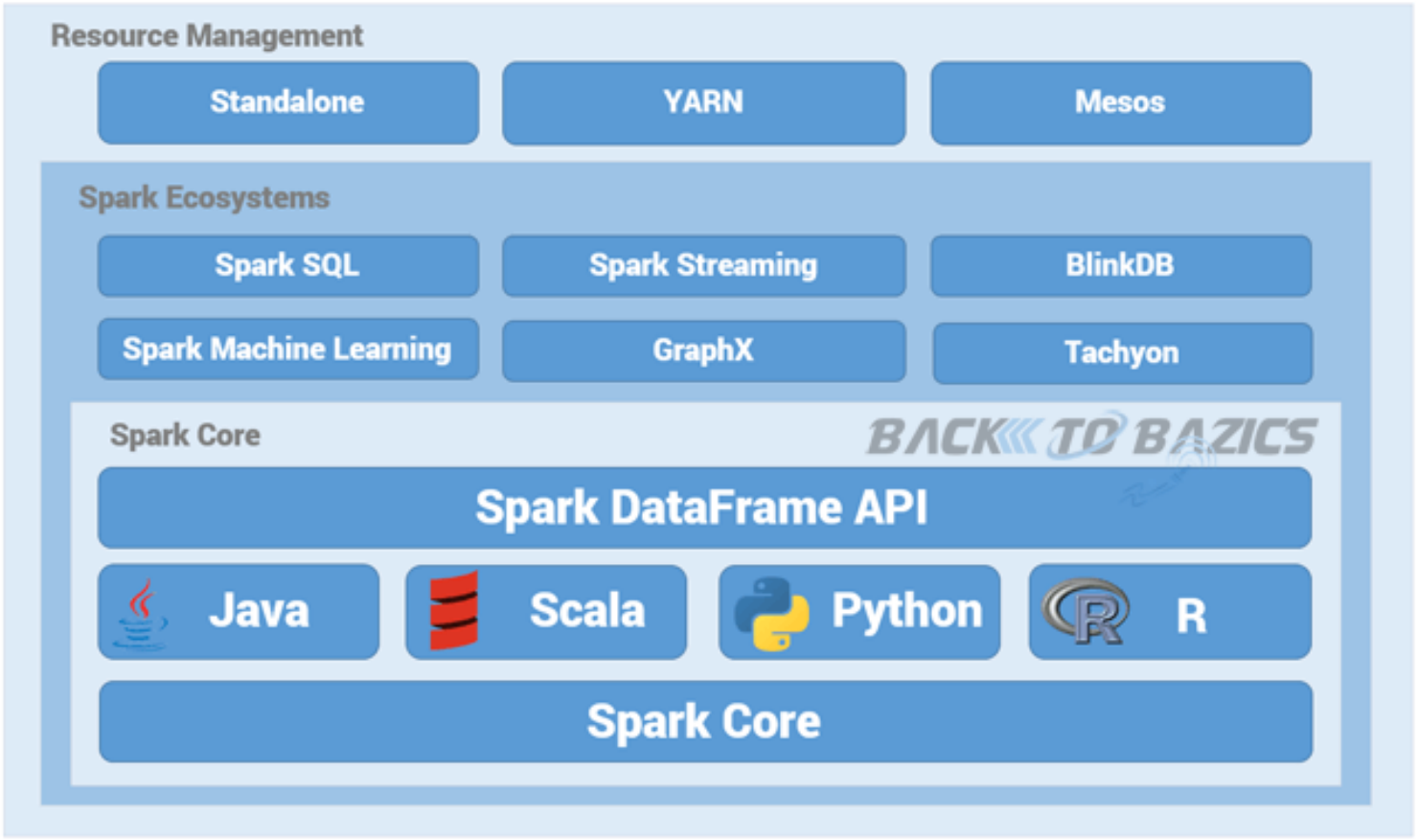
类Hadoop MapReduce的开源通用并行计算系统

Spark特点

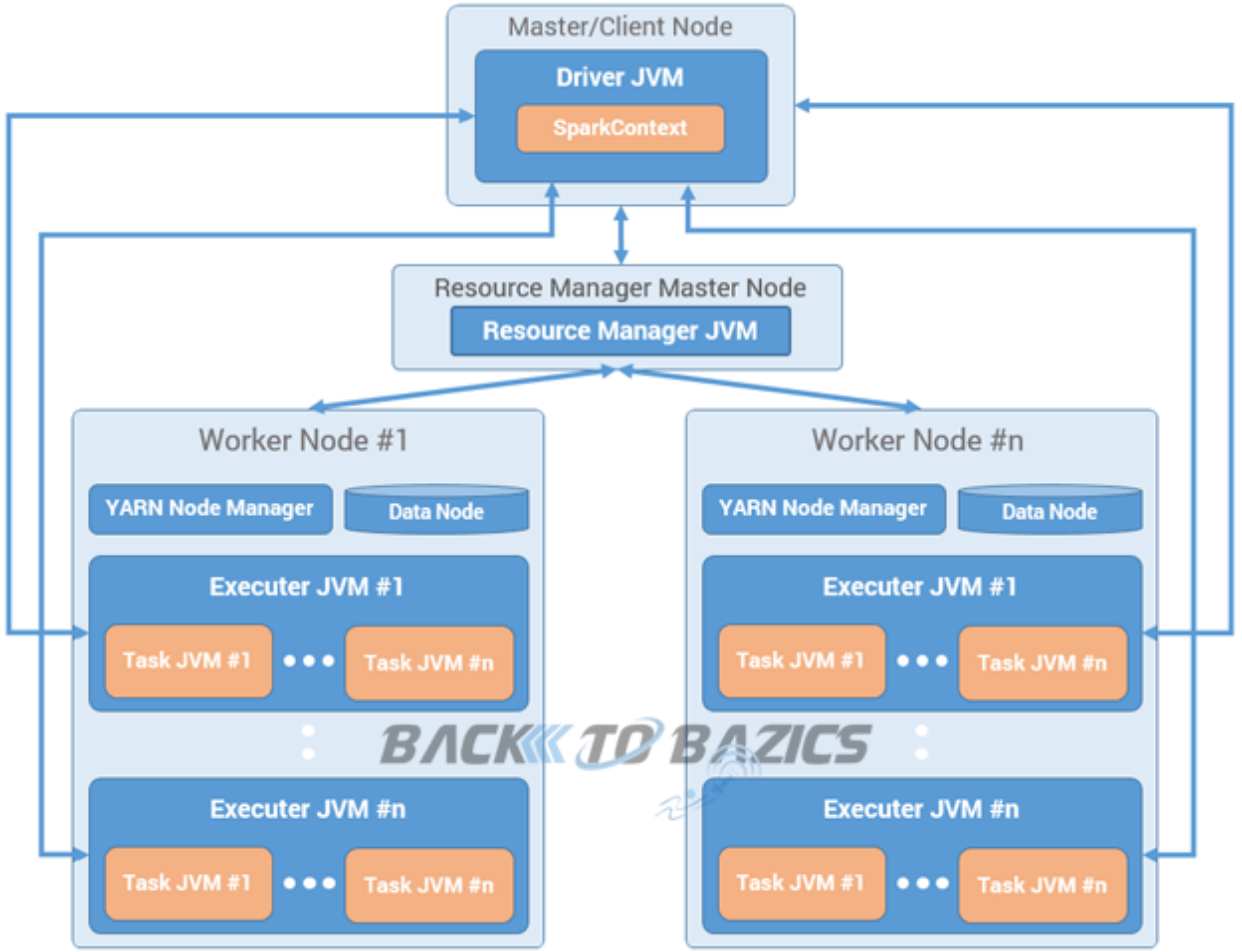
- 一站式解决方案
- 内存计算：高速
- 惰性求值
- 高度可扩展
- 易用：多种编程语言API
- 多种数据源

- 中间数据存放：spark存放于内存中；
- 通用性：Spark具有更多的数据集操作；
- 容错手段：Spark通过checkpoint；MapReduce通过数据副本。
- 拓展性和易用性：Spark具有丰富的API和提供交互式shell。

Spark框架


Resource Management
Standalone
YARN
Mesos
Spark Ecosystems
Spark SQL
Spark Streaming
BlinkDB
Spark Machine Learning
GraphX
Tachyon
Spark Core
BACK TO BAZICS
Spark DataFrame API
Java
Scala
Python
R
Spark Core

Spark on YARN



- Driver Program
- Cluster Manager
- Worker Node
- Executors
- Tasks

Spark架构组成

- **Driver Program**
  - 把用户程序转化为任务，并分发至执行器节点
- **Cluster Manager**
  - 管理集群资源
- **Workers**
  - 运行于工作节点的工作进程，为Spark应用程序提供计算资源
- **Executors**
  - 执行任务，在内存中缓存数据
- **Tasks**
  - 最小的工作单位，执行计算，返回结果

Spark关键术语

- **Application** 应用程序
  - 包含受驱动器管理的一个或多个作业
- **Job** 作业
  - 一系列包含有**行动操作（actions）**的任务
- **Stage** 步骤
  - 可以被spark并行执行的单个作业中的一些列任务
  - 每个步骤的计算能够产生可以被留存（persist）下来的中间结果
- **Task** 任务
  - 分发至单个执行器的工作单元

理解这几个术语：https://stackoverflow.com/questions/28973112/what-is-spark-job

Spark应用程序运行流程

1. **提交应用：`spark-submit`**
2. **启动驱动器**：由spark-submit启动，调用main()方法
3. 驱动器程序与集群管理器通信，**申请资源**用于启动执行器节点
4. 集群管理器为驱动器程序**启动执行器节点，分配slave节点资源**
5. 驱动器进程**执行**用于应用中的操作：把用户程序转换为任务并发送到执行器进程
6. 任务在执行器程序中进行**计算**并保存
7. **终止**执行器进程，释放资源

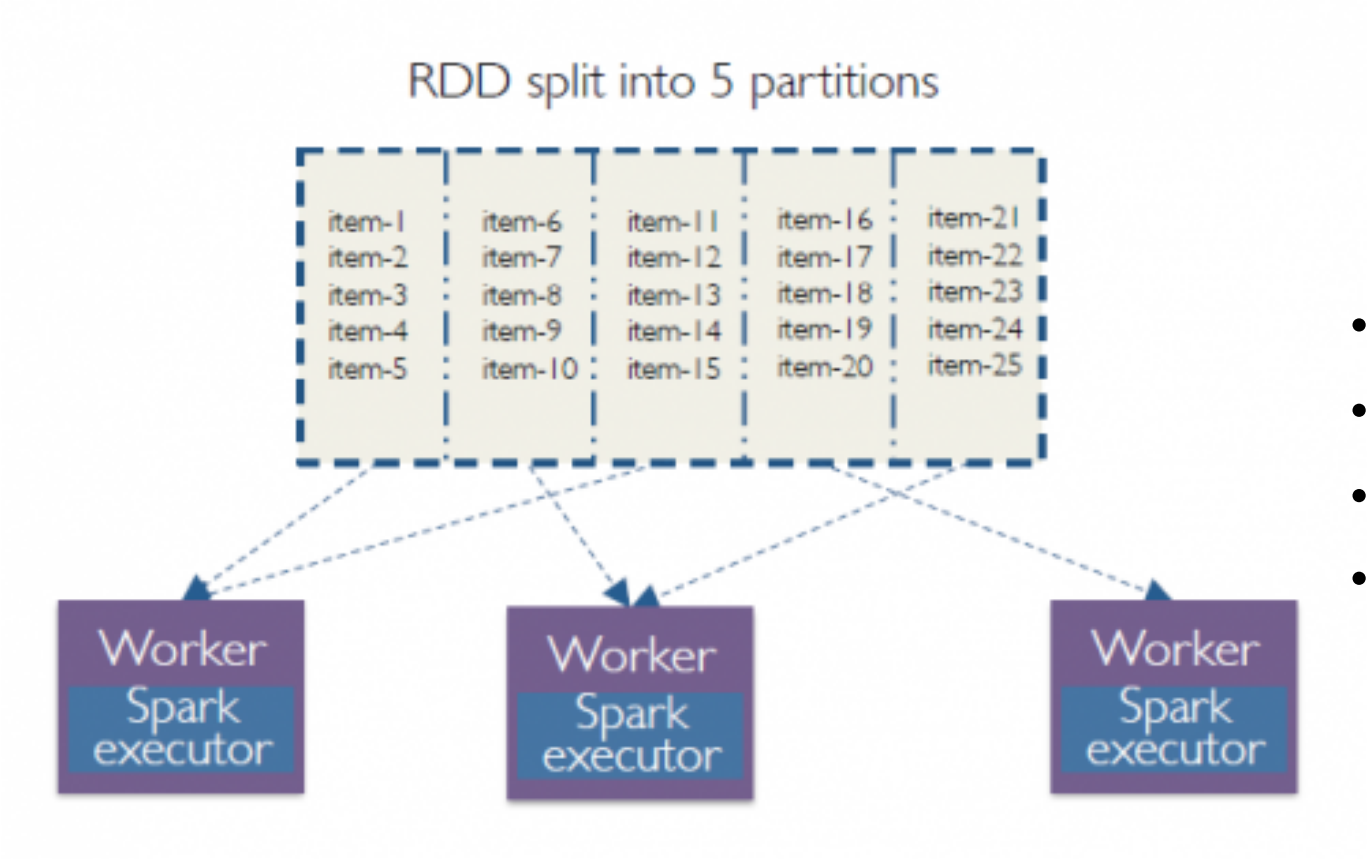
Spark核心：**RDD**

Resilient Distributed Dataset：弹性分布式数据集

RDD?

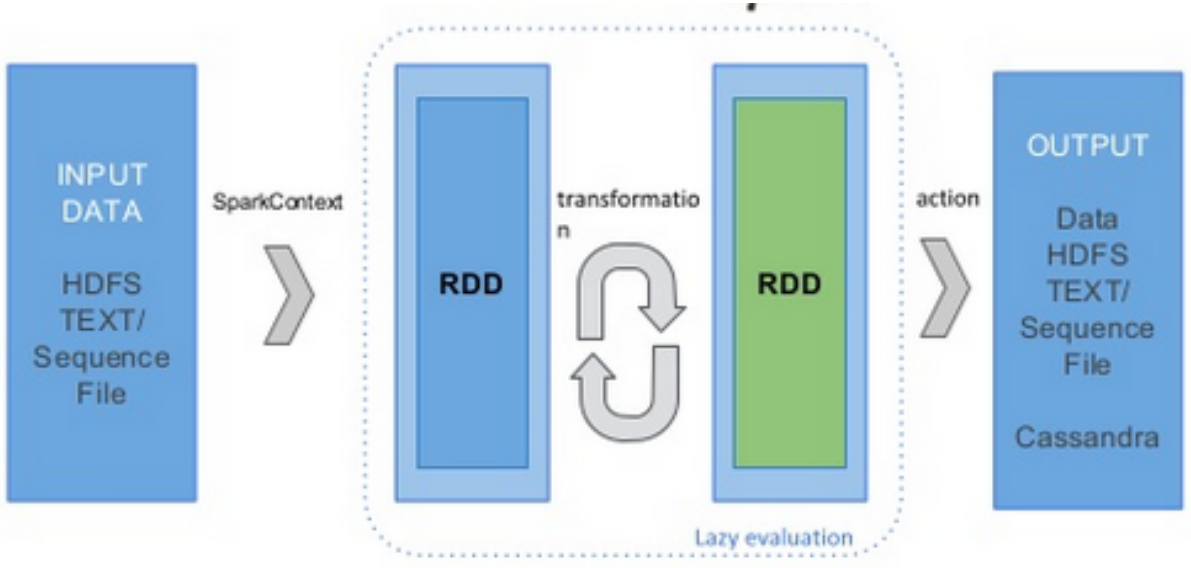
- 分布于集群上的**只读**对象集；
- 通过并行的转换操作（transformations）构建；
- 通过“血缘关系（lineage）”自动重建实现**容错**；
- 可控的数据留存（persistence）：RAM，HDFS，etc.
- 具有一组包含分片所在的数据块的位置列表。

RDD?



- RDD保存对分片对象的引用；
- 每个分片对象引用一份数据子集；
- 分片被分发至集群中；
- 每个分片或分割默认存储在内存中。

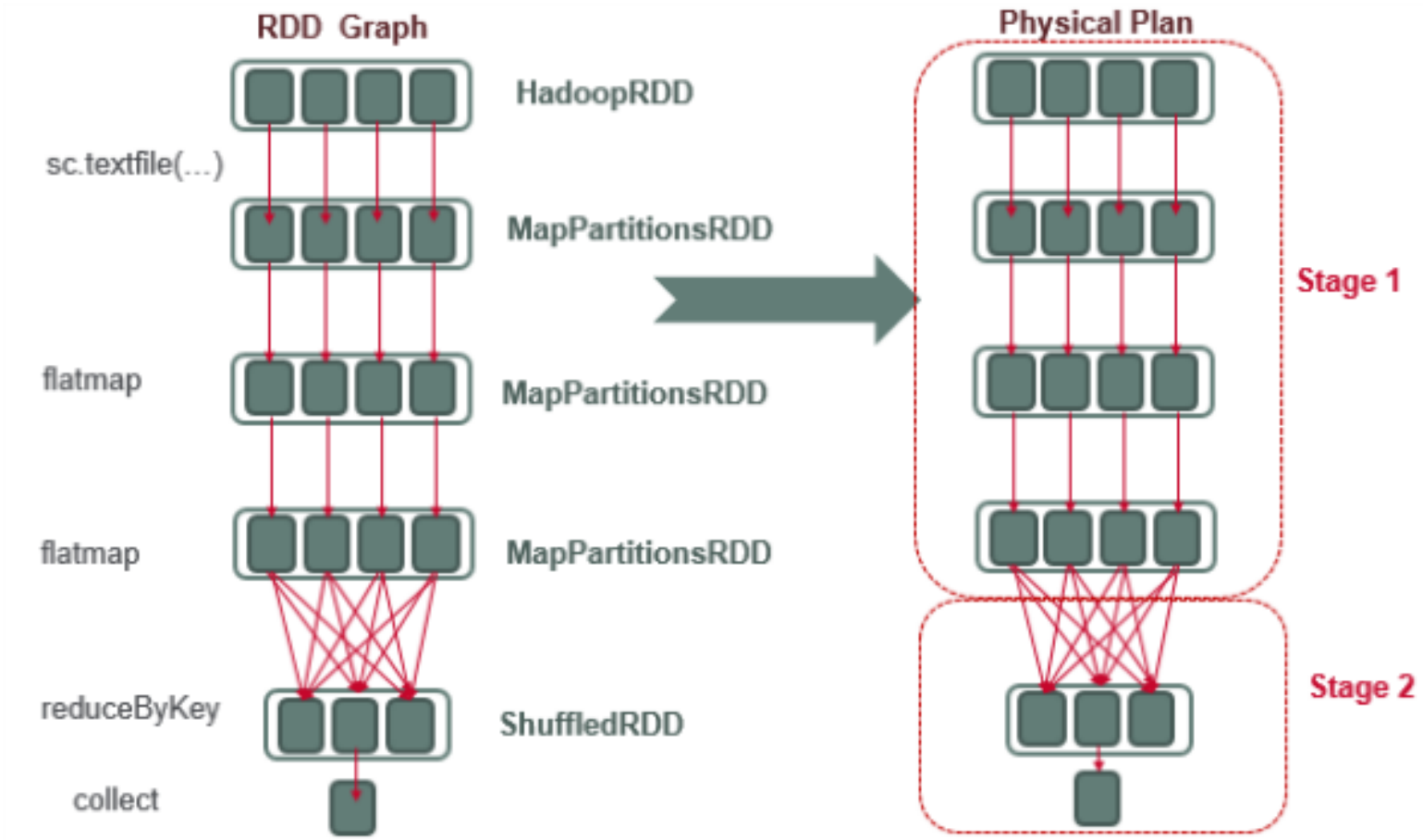
RDD操作



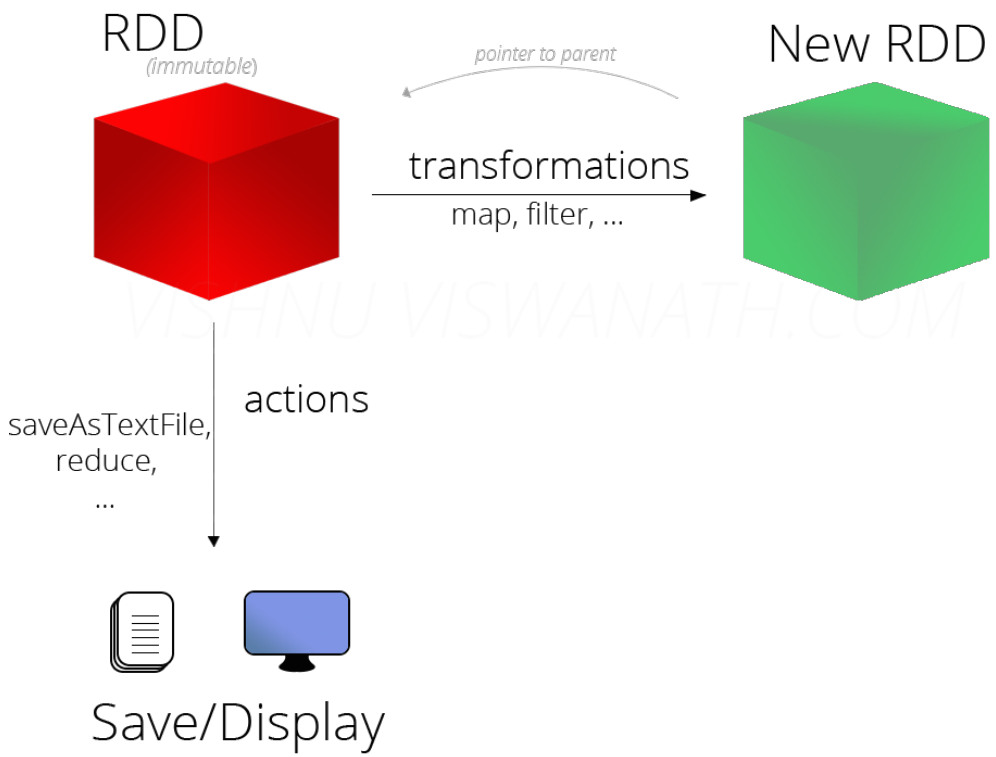
- **创建RDD**：读取数据至Spark集群；
- **转化操作**（Transformations）
  - 从其他RDD生成新RDD (map, filter, groupBy)；
  - 通过有向图（DAG：Directed Acyclic Graph ）惰性运算；
- **行动操作**（Actions）：返回结果或将结果写入存储。

http://www.bogotobogo.com/Hadoop/BigData_hadoop_Apache_Spark_Programming_Model_RDD.php

RDD操作


RDD Graph
Physical Plan
HadoopRDD
sc.textfile(...)
MapPartitionsRDD
flatmap
MapPartitionsRDD
flatmap
MapPartitionsRDD
reduceByKey
ShuffledRDD
collect
Stage 1
Stage 2

RDD操作


RDD
(immutable)
pointer to parent
New RDD
transformations
map, filter, ...
actions
saveAsTextFile,
reduce,
...
Save/Display

应用程序提交

配置

- sparkConfig
- spark-submit或spark-shell参数
- $SPARK_HOME/conf/spark-defaults.conf

常用参数

- spark.app.name：应用程序的名称，将在UI和日志数据中出现
- spark.master：集群管理器连接的地方
- spark.driver.cores：driver程序运行需要的cpu内核数
- spark.driver.memory：driver进程使用的内存数
- spark.executor.memory：每个executor进程使用的内存数
- spark.driver.extraClassPath：附加到driver的classpath的额外的classpath实体（例如，从数据库获取数据，需指派数据库驱动类库路径）

详细参考官网文档或者参考：http://blog.javachen.com/2015/06/07/spark-configuration.html

sparkConfig

- 一般用来设置通用属性，如master URL、应用程序名称等等。
- 通过**set()**方法设置的任意键值对

```
val conf = new SparkConf()
     .setMaster("local[2]")
     .setAppName(“Spark app”)
     .set("spark.executor.memory", "1g")
val sc = new SparkContext(conf)
```

conf/spark-defaults.conf

```
spark.master = spark://5.6.7.8:7077
spark.executor.memory = 512m
spark.eventLog.enabled = true
spark.serializer =
org.apache.spark.serializer.KryoSerializer
```

spark-submit参数

```
# Run on a YARN cluster
export HADOOP_CONF_DIR=XXX
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master yarn \
  --deploy-mode cluster \  # can be client for client mode
  --executor-memory 20G \
  --num-executors 50 \
  /path/to/examples.jar \
  1000
```

SparkR & sparklyr

Spark与R的取长补短

- Spark具有强大的分布式高性能计算能力
- R具有大量机器学习和统计计算工具包

Spark + (?) + R

- **SparkR** @2015, Spark v1.4 → v2.2.0
- **sparklyr** @2016.09, Rstudio → v0.6.0

SparkR vs **sparklyr**

- SparkR对R代码的并行计算的执行上比sparklyr占优，但sparklyr的开发者在逐步完善中（sparklyr v0.6.0，已经支持分布式计算）；
- SparkR依赖于对应版本的Spark，在兼容上存在一定的限制；而sparklyr希望其运行不依赖特定Spark版本；
- SparkR与R生态中的其他包的兼容性低；sparklyr致力于与其他包兼容，例如：dplyr、MLlib、h2o等；
- sparklyr更强调可扩展性，为用户提供了更多的自由度；
- sparklyr发布在CRAN，更易安装；
- sparklyr与RStudio整合，交互式分析更友好；
- 另外还有一些小差异，如SparkR中的`as.Dataframe()`会保留Date格式数据，而sparklyr中的对应函数`copy_to()`不保留Date格式。

参考：
https://stackoverflow.com/questions/39494484/sparkr-vs-sparklyr
https://github.com/rstudio/sparklyr/issues/502

交互式分析

- SparkR入门：**`sparkr_intro.html`**
- Sparklyr入门：**`sparklyr_intro.html`**

spark-submit提交Rscript

- Bash脚本实例：**`submit-me-r-script.sh`**

后记

搭建和维护集群的**挑战**

- 复杂的生态，学习曲线较陡
- 分布式系统管理
- 系统与数据安全
- 高可用、高性能、伸缩性
- 适应业务需求和未来发展

如何学习大数据分析?

- 掌握一门编程语言：R/Python, Scala/Java
- 了解Hadoop与Spark等大数据框架的架构
- 理解大数据框架参数调优和性能优化

- 其他
  - DRY：不要重复机械的事情
  - 版本控制：Git、Github
  - 文档化：R Notebook/R Markdown、Jupyter Notebook
  - Just Googling

谢谢！

PS. 文档中的图片均来自于网络，没有列出来源。