

Spark技术原理

www.huawei.com





目标

- 学完本课程后，您将能够：
 - 理解**Spark**应用场景，掌握**Spark**特点
 - 掌握**Spark**计算能力及其技术架构
 - 掌握**Spark**组件在**FusionInsight** 平台中的使用





目录

1. Spark概述
2. Spark基本功能技术架构
3. Spark在FusionInsight中的集成情况

Spark概述

是什么？

- **Spark**  一个基于内存的分布式批处理引擎
- 由  贡献到 **Apache** 社区的开源项目，是 **AMP** 大数据栈的基础组件

做什么？

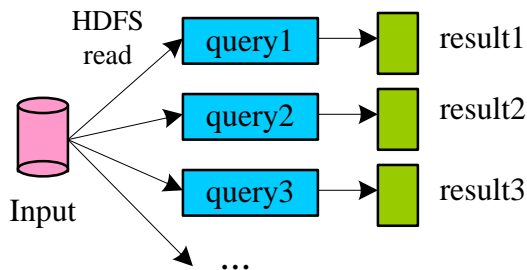
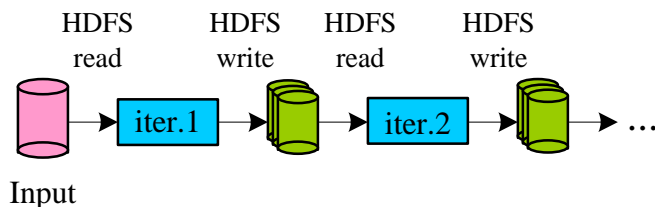
- **数据处理(Data Processing)**: 可以用来快速处理数据，兼具容错性和可扩展性
- **迭代计算(Iterative Computation)**: 支持迭代计算，有效应对多步数据处理逻辑
- **数据挖掘(Data Mining)**: 在海量数据基础上进行复杂的挖掘分析，可支持各种数据挖掘和机器学习算法

Spark适用场景

适用场景

- ✓ 数据处理，**ETL**（抽取、转换、加载）
- ✓ 机器学习
- ✓ 交互式分析
- ✓ 特别适用于迭代计算，数据重复利用场景

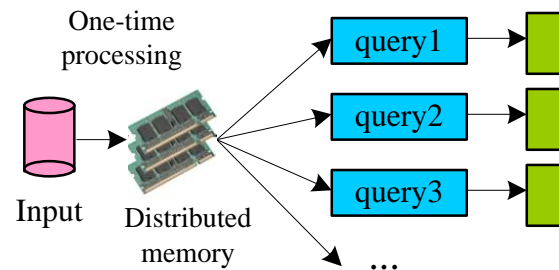
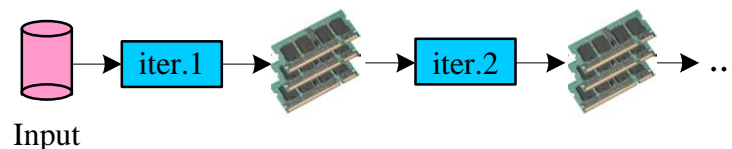
需要反复操作的次数越多，所需读取的数据量越大，受益越大。



Data Sharing in MapReduce

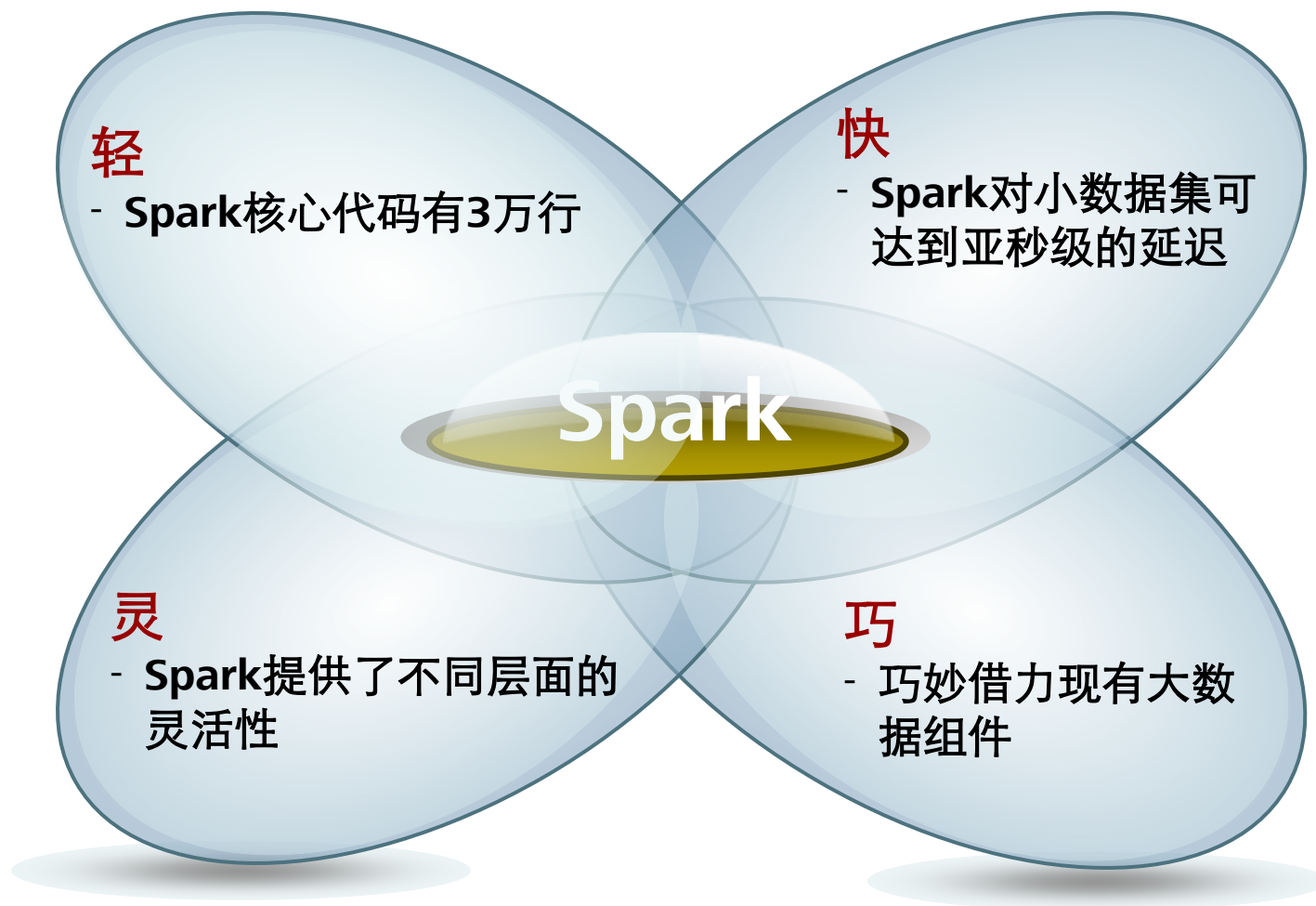
对比Hadoop

- ✓ 性能上提升**高于100倍**
- ✓ **Spark**的中间数据放在内存中，对于迭代运算的效率更高，进行批处理时**更高效**
- ✓ **更低的延迟**
- ✓ **Spark**提供更多的数据集操作类型，编程模型比**Hadoop****更灵活**，开发效率更高
- ✓ **更高的容错能力**（血统机制）



Data Sharing in Spark

Spark的特点





目录

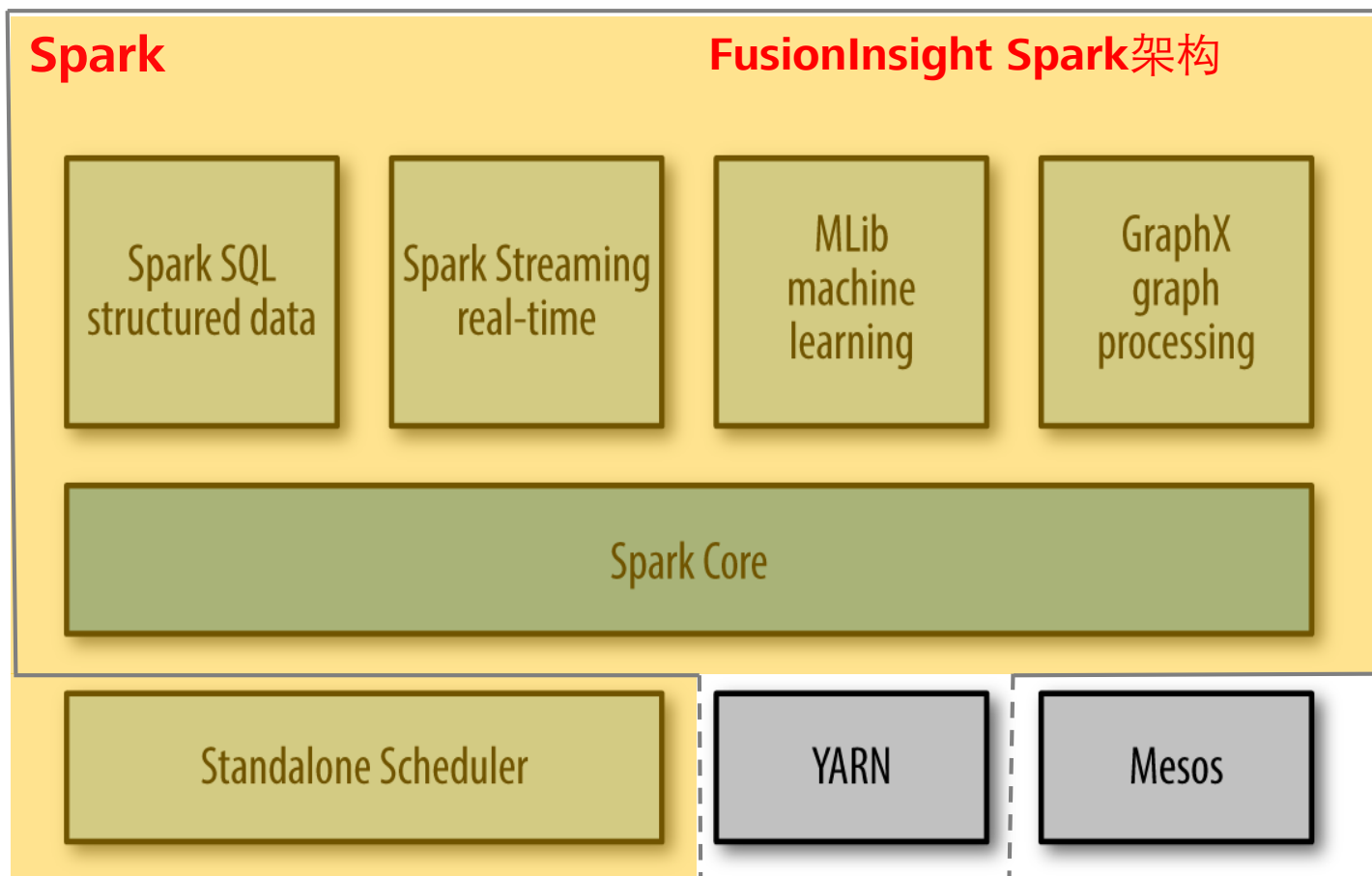
1. Spark概述

2. Spark基本功能技术架构

- SparkCore
- SparkSQL
- SparkStreaming

3. Spark在FusionInsight中的集成情况

Spark技术架构



Spark应用运行流程—关键角色



Client:

需求提出方，负责提交需求（应用）



Driver:

负责应用的业务逻辑和运行规划（**DAG**）



ApplicationMaster:

负责应用的资源管理，根据应用的需要，向资源管理部门（**ResourceManager**）申请资源



ResourceManager:

资源管理部门，负责整个集群的资源统一调度和分配。



Executor:

负责实际计算工作，一个应用会分拆给多个**Executor**来进行计算。

Spark基本概念

Application:

Spark用户程序，提交一次应用为一个**Application**，一个**App**会启动一个**SparkContext**，也就是**Application**的**Driver**，驱动整个**Application**的运行。

Job:

一个**Application**可能包含多个**Job**，每个**action**算子对应一个**Job**；**action**算子有**collect**，**count**等。

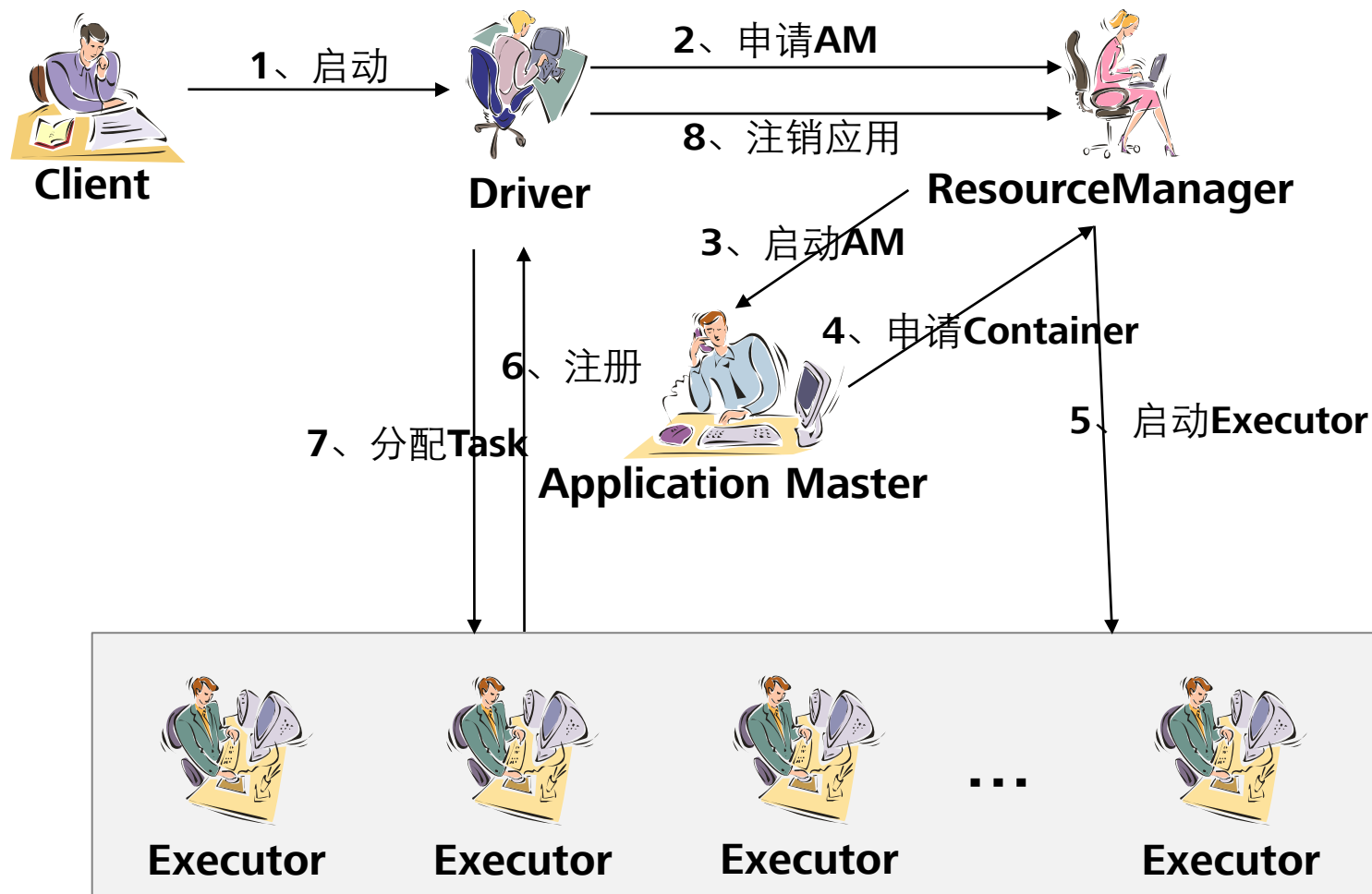
Stage:

每个**Job**可能包含多层**Stage**，划分标记为**shuffle**过程；**Stage**按照依赖关系依次执行。

Task:

具体执行任务的基本单位，被发到**executor**上执行。

Spark应用运行流程



Spark核心概念--RDD

- **RDD (Resilient Distributed Datasets)** 即弹性分布数据集，指的是一个只读的，可分区的分布式数据集。这个数据集的全部或部分可以缓存在内存，在多次计算之间重用。

RDD的生成

- 从Hadoop文件系统（或与Hadoop兼容的其它存储系统）输入创建（如HDFS）
- 从父RDD转换得到新的RDD

RDD的优点

- RDD是只读的，可提供更高的容错能力
- RDD的不可变性，可以实现Hadoop MapReduce的推测式执行
- RDD的数据分区特性，可以通过数据的本地性来提高性能
- RDD都是可序列化的，在内存不足时可自动降级为磁盘存储

RDD

RDD的存储和分区

- 用户可以选择不同的存储级别存储RDD以便重用（11种）
- 当前RDD默认存储于内存，但当内存不足时，RDD会溢出到磁盘中
- RDD在需要进行分区时会根据每条记录Key进行分区，以此保证两个数据集能高效进行Join操作

RDD的特点

- 在集群节点上是不可变的，是已分区的集合对象
- 失败自动重建
- 可以控制存储级别（内存，磁盘等）来进行重用
- 是可序列化的
- 是静态类型的

RDD算子：Transformation和Action

Transformation

返回值还是一个**RDD**，如**map**、**filter**、**join**等。Transformation都是**Lazy**的，代码调用到Transformation的时候，并不会马上执行，需要等到有**Action**操作的时候才会启动真正的计算过程。

Action

如**count**，**collect**，**save**等，Action操作是返回结果或者将结果写入存储的操作。

Action是Spark应用真正执行的触发动作。

RDD Transformation和Action

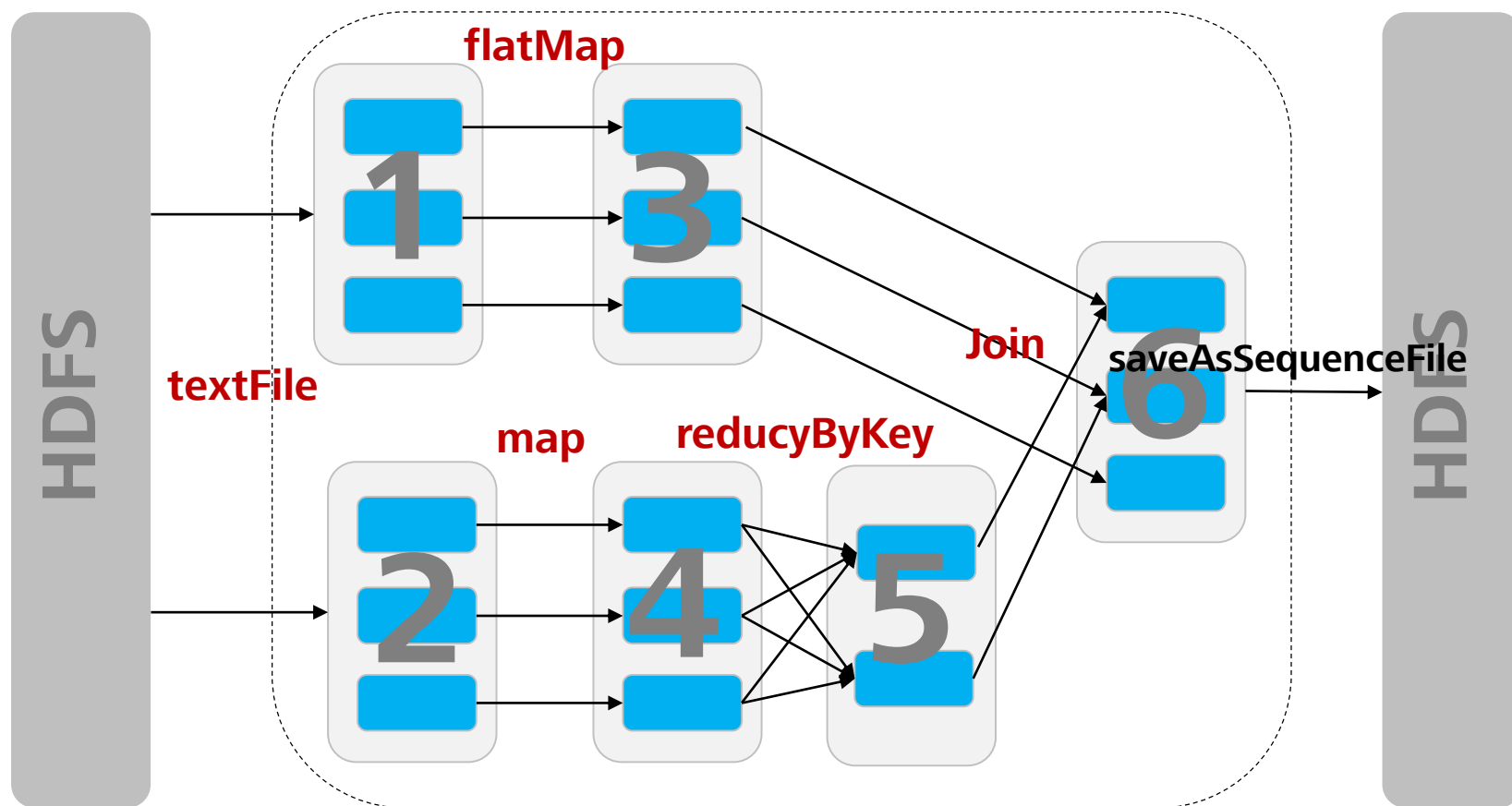
常用的Transformation :

`map(f:T =>U):RDD[T] =>RDD[U]`
`filter(f:T =>Bool):RDD[T]=>RDD[T]`
`flatMap(f:T =>Seq[U]):RDD[T] =>RDD[U]`
`groupByKey(): RDD[(K, V)] => RDD[(K, Seq[V])]`
`reduceByKey(f:(V,V) => V):RDD[(K, V)]=>RDD[(K, V)]`
`union():(RDD[T],RDD[T])=>RDD[T]`
`join():(RDD[(K, V)],RDD[(K, W)])=>RDD[(K, (V, W))]`
`mapValues(f:V=>W): RDD[(K, V)]=>RDD[(K, W)]`
`partitionBy(p:Partitioner[K]):RDD[(K,V)]=>RDD[(K,V)]`

常用的Action :

`count(): RDD[T] => Long`
`collect(): RDD[T] => Seq[T]`
`reduce(f:(T,T) =>T):RDD[T]=>T`
`lookup(k:K):RDD[(K,V)] => Seq[V]`

RDD Transformation和行动Action



样例程序--WordCount

创建**SparkContext**对象，
设置应用名称为
Wordcount。

从**HDFS**加载文本文件，
得到一个**RDD**

调用**RDD**的**Transformation**
进行计算：
将文本文件按空格分割，然
后每个单词计数置为**1**，最后
按相同的**Key**将计数求合。
这一步会分发到各个
Executor上执行。

调用**Action**操作，保存结果。
这一行才触发真正的任务执
行。

object WordCount

{

def main (args: Array[String]): Unit = {

//配置**Spark**应用名称

val conf = new SparkConf().setAppName("WordCount")

val sc: SparkContext = new SparkContext(conf)

val textFile = sc.textFile("hdfs://...")

val counts = textFile.flatMap(line => line.split(" "))

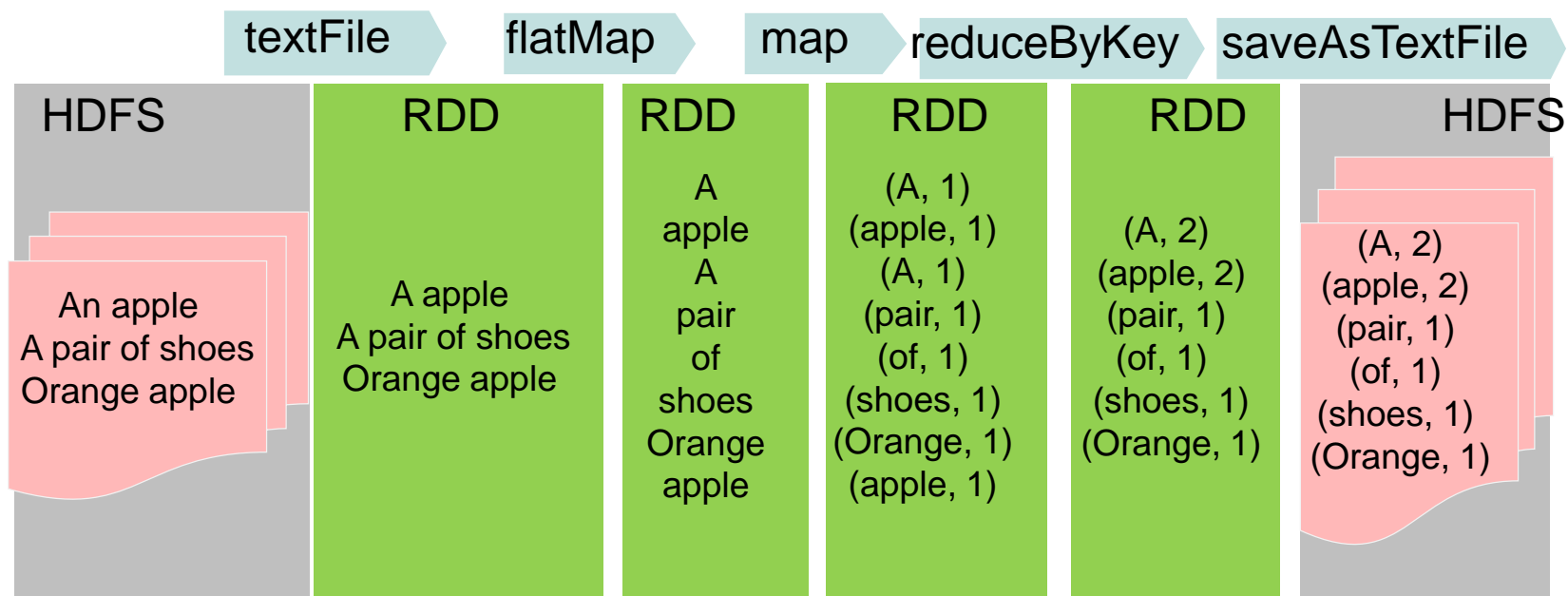
.map(word => (word, 1))

.reduceByKey(_ + _)

counts.saveAsTextFile("hdfs://...")

}

样例程序--WordCount

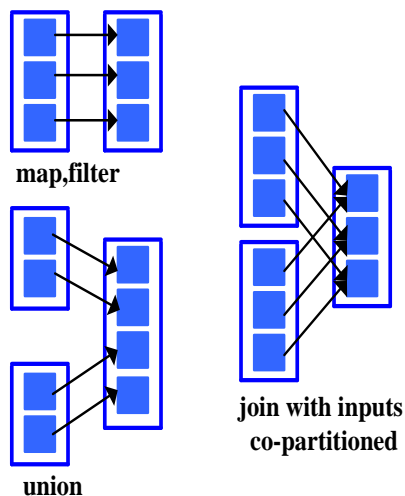


Spark核心概念 – 宽依赖和窄依赖

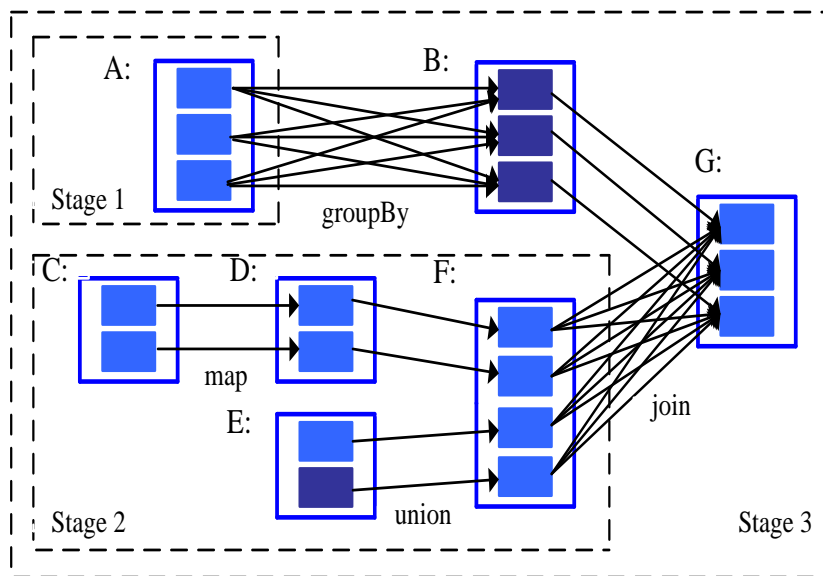
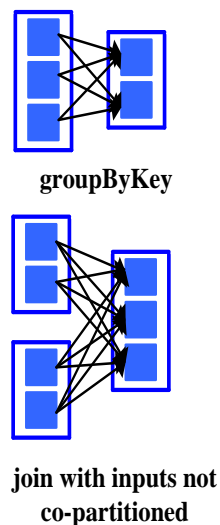
RDD父子依赖关系:

- 窄依赖 (**Narrow**) 指父RDD的每一个分区最多被一个子RDD的分区所用。
- 宽依赖 (**Wide**) 指子RDD的分区依赖于父RDD的所有分区，是**Stage**划分的依据。

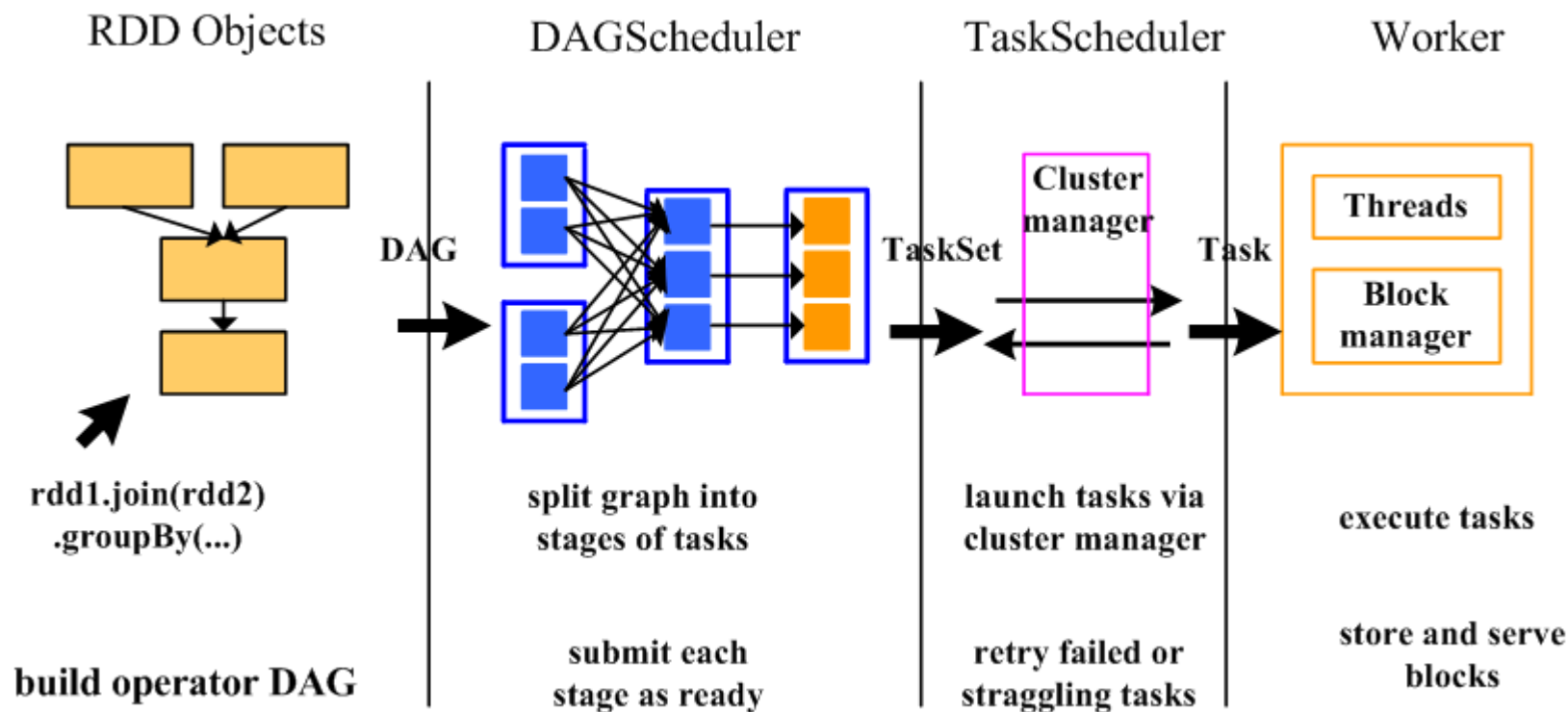
Narrow Dependencies:



Wide Dependencies:



Spark应用调度





目录

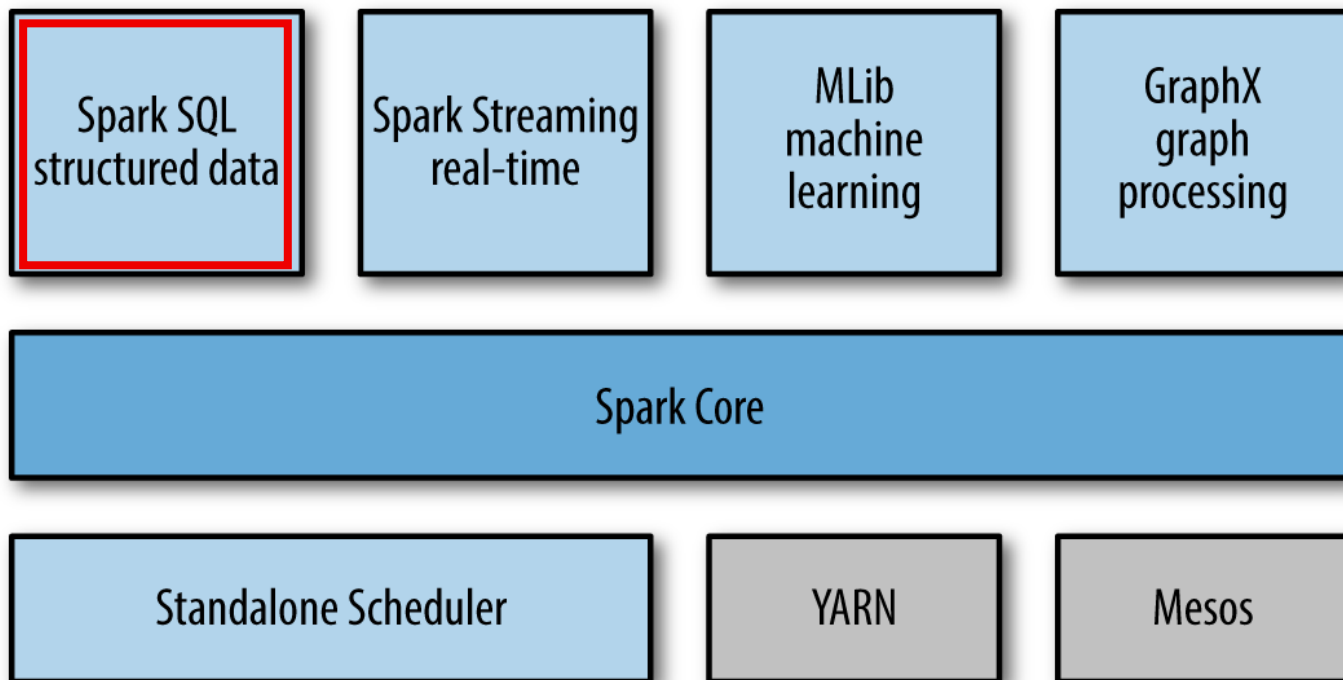
1. Spark概述

2. Spark基本功能技术功架

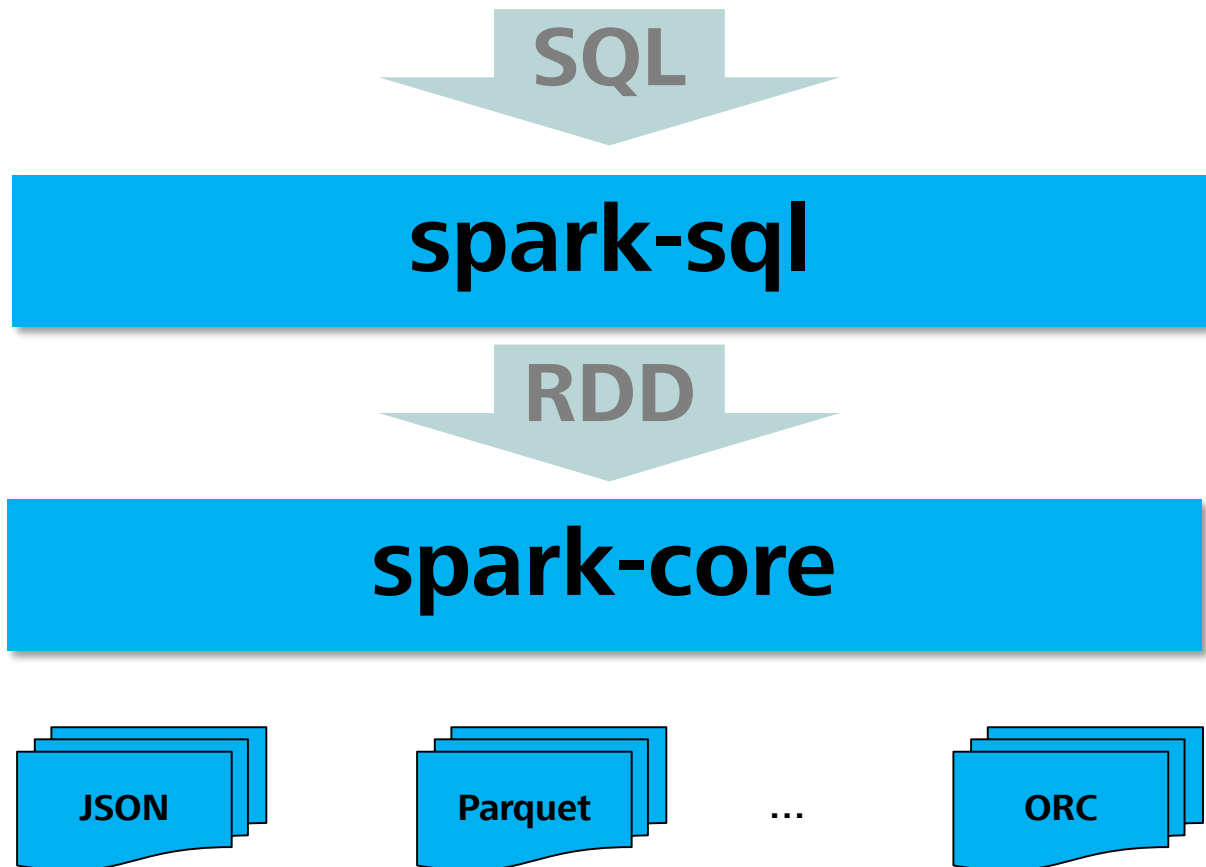
- SparkCore
- SparkSQL
- SparkStreaming

3. Spark在FusionInsight中的集成情况

SparkSQL所处位置



SparkSQL原理





目录

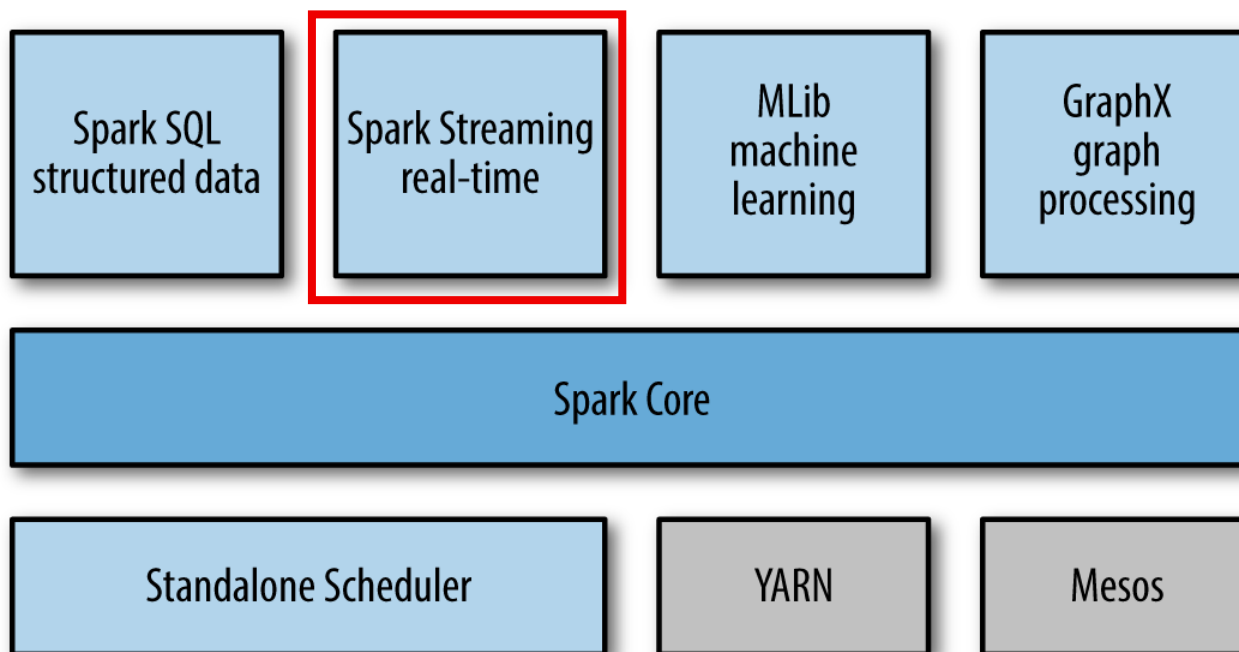
1. Spark概述

2. Spark基本功能技术功架

- SparkCore
- SparkSQL
- SparkStreaming

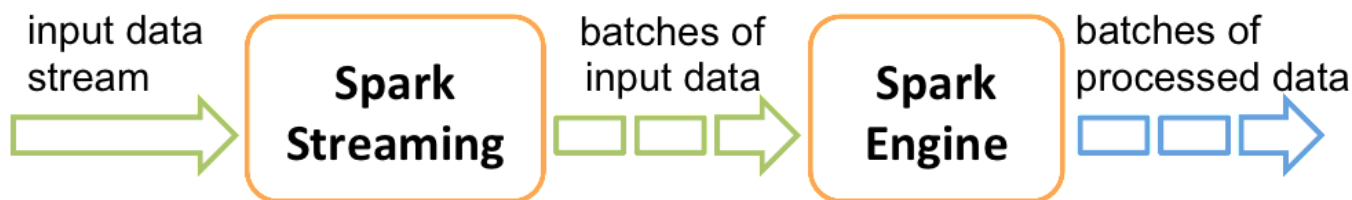
3. Spark在FusionInsight中的集成情况

SparkStreaming所处位置

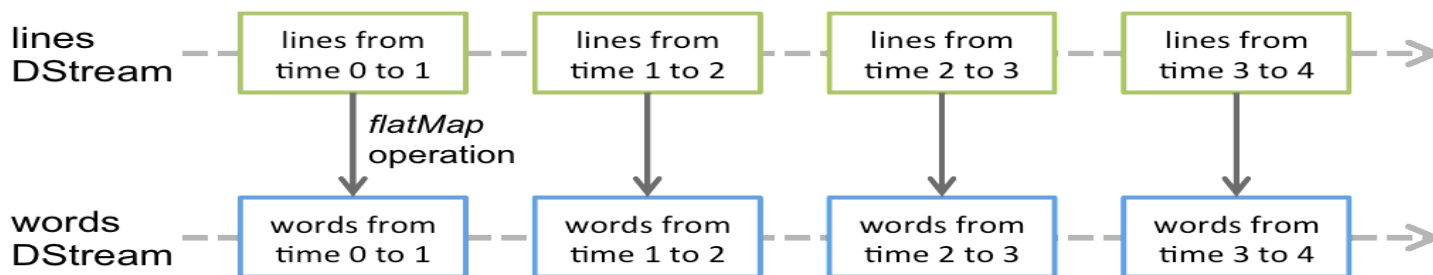


SparkStreaming原理

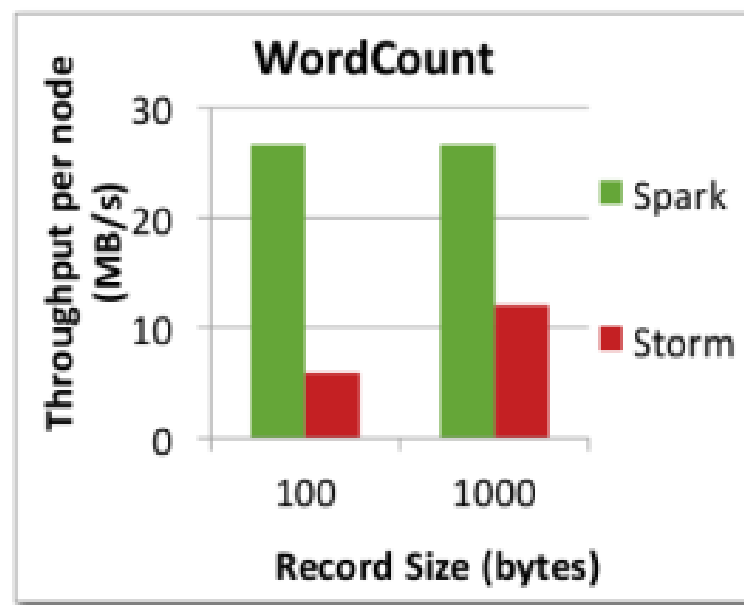
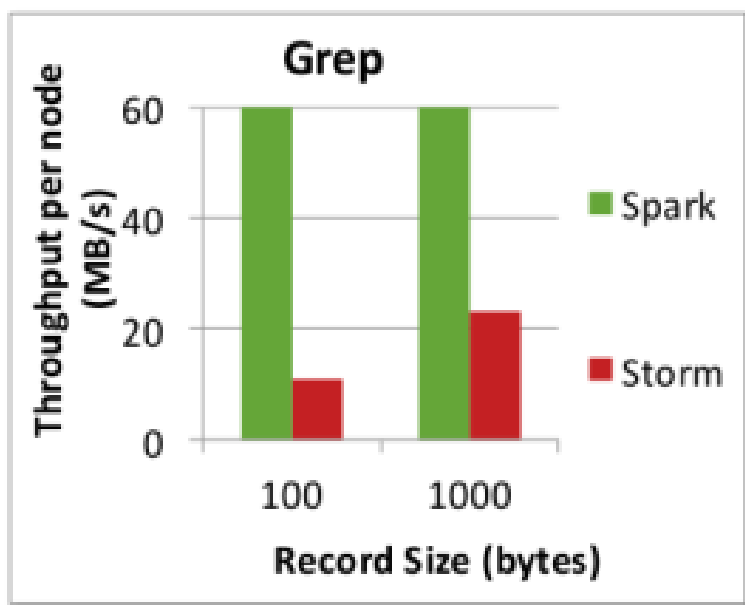
- **Spark Streaming**接收实时的输入数据流，然后将这些数据切分为批数据供**Spark**引擎处理，**Spark**引擎将数据生成最终的结果数据。



- 使用**DStream**从**Kafka**和**HDFS**等源获取连续的数据流，**Dstreams**由一系列连续的**RDD**组成，每个**RDD**包含确定时间间隔的数据，任何对**Dstreams**的操作都转换成对**RDD**的操作



SparkStreaming对比Storm





目录

1. Spark概述
2. Spark基本功能技术功架
3. Spark在FusionInsight中的集成情况

Spark的WebUI呈现

FusionInsight平台为**Spark**服务提供了管理监控的可视化界面，通过**Web UI**界面，可完成以下动作：

- 1.服务状态信息、角色信息以及开放的配置项
- 2.管理操作：启停**spark**、下载**spark**客户端、同步配置
- 3.服务总体概况
- 4.角色的显示和健康状况，点击相应角色可查看角色下的实例

FusionInsight Manager

系统概览 服务管理 主机管理 告警管理 审计管理 租户管理 系统设置

服务 > Spark 服务状态

服务状态 实例 服务配置 资源贡献排名 1

启动服务 停止服务 下载客户端 更多操作 2

Spark 概述 3

健康状态	良好
配置状态	已同步
版本	1.5.1
Spark WebUI	JobHistory(167-52-0-27) JobHistory(167-52-0-4)

操作状态和健康状态 4

角色	操作状态	健康状态
JDBCServer	2 已启动	2 良好
JobHistory	2 已启动	2 良好
SparkResource	5 已启动	5 良好

Spark常驻进程

- **JDBCServer**

- 实际上是一个长驻的**spark**应用，对外提供**JDBC**的服务。
- 用户可以通过执行**beeline**或者**JDBC**脚本，连接**JDBCServer**，执行**sql**语句。
- 主备部署，无单点故障。

- **JobHistory**

- 该进程用于提供**HistoryServer**页面，展示历史**Spark**应用的执行信息。
- 双节点负荷分担，无单点故障。

Spark与其他组件交互

在**FusionInsight**集群中，**Spark**主要与以下组件进行交互：

- **HDFS**：**Spark**在**HDFS**文件系统中读写数据(必选)
- **YARN**：**Spark**任务的运行依赖**Yarn**来进行资源的调度管理(必选)
- **Hive**：**Spark-sql**的元数据库和数据文件与**Hive**完全共用(必选)
- **Zookeeper**：**JDBCServer**的**HA**的实现依赖于**Zookeeper**的协调(必选)
- **Kafka**：**Spark**可以接收**Kafka**发送的数据流(可选)
- **HBase**：**Spark**可以操作**HBase**的表(可选)



本章小结

- 对**Spark**的产生背景和应用场景给予简单介绍，同时介绍了**spark**的特点。
- 介绍了**Spark**的基本概念，技术架构，着重介绍了**Spark**任务的进程运行，**Spark On Yarn**模式，以及**Spark** 的应用调度。
- 介绍了**Spark**在**FusionInsight HD**中的集成情况。

思考

- 1、**Spark**的特点有哪些？
- 2、**Spark**和相对于**MR**的优势是什么？
- 3、**Spark**的应用场景有哪些？
- 4、**Spark** 宽依赖窄依赖的区别是什么？



习题

- 填空

1. **RDD**的算子分为：_____和_____两类
2. _____模块是**Spark**最核心的模块
3. **RDD**的依赖类型分为_____和_____两种类型
4. **FusionInsight**集成的**Spark**有_____、_____两个常驻进程



学习推荐

- 华为**Learning**网站
 - <http://support.huawei.com/learning/Index!toTrainIndex>
- 华为**Support**案例库
 - <http://support.huawei.com/enterprise/servicecenter?lang=zh>

Thank you

www.huawei.com