Hadoop与Spark入门

覃雄派

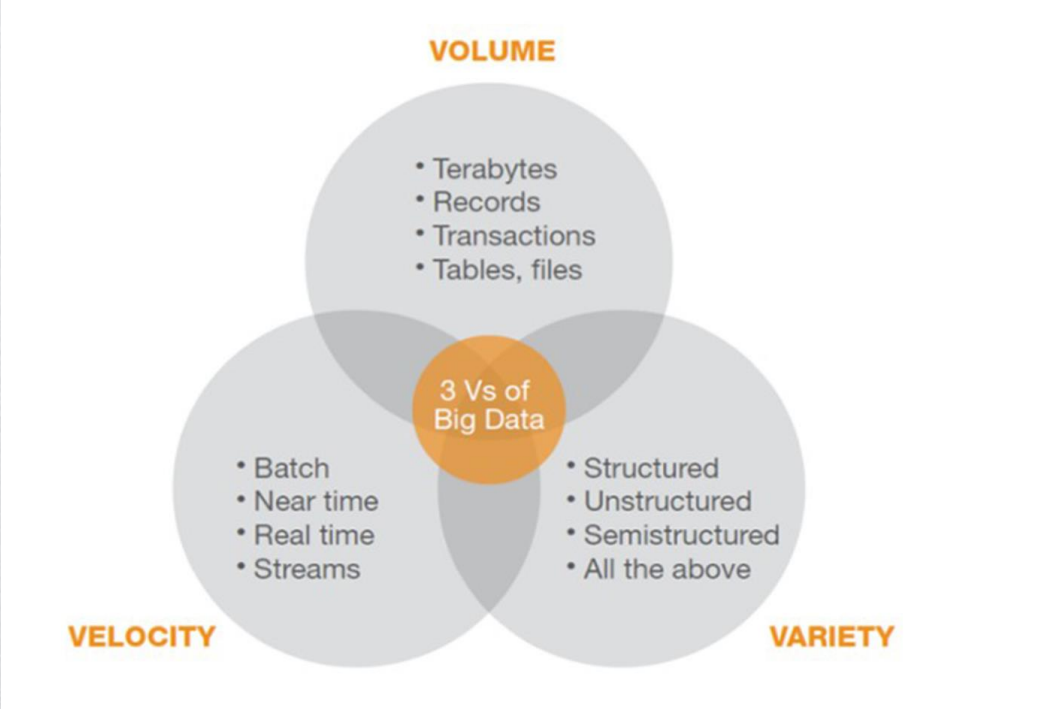
Hadoop与Spark入门

提纲

- 首先了解Hadoop
  - HDFS
  - Map Reduce
- Hadoop的生态系统、成绩与局限性
- Spark的诞生、Spark简介、Spark的优势
  - Spark生态系统
  - RDD、Transformation、Action
  - 宽依赖、窄依赖、及其示例
  - DAG
  - DAG的调度
  - Word Count Scala/Word Count Python/Word Count Java
  - SparkSQL与DataFrame API
- Spark的成绩
- Spark的应用

Hadoop与Spark入门

- Big data has become necessity now

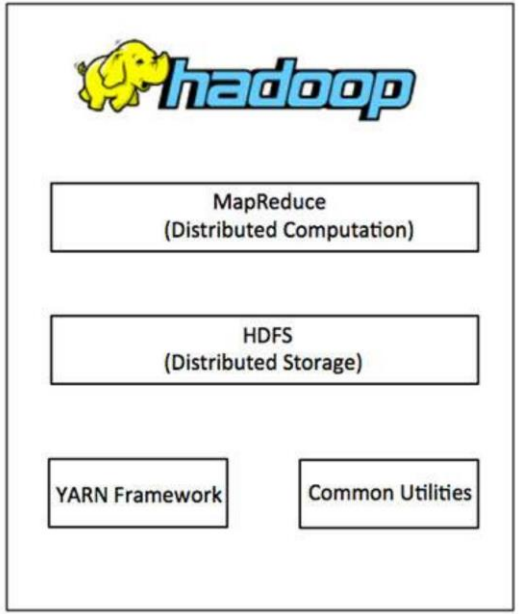


Hadoop与Spark入门

- 先了解一下Hadoop
  - Apache Hadoop是存储和处理大数据的开源软件框架
  - Hadoop项目，最初由Doug Cutting和Mike Cafarella于2005年创建，其最初的目标，是提供Nutch搜索引擎的分布式处理能力
  - 2013年，Hadoop已经从1.0版演化发展到2.0版(YARN)
  - 目前，Hadoop 3.0已经处于General Available状态

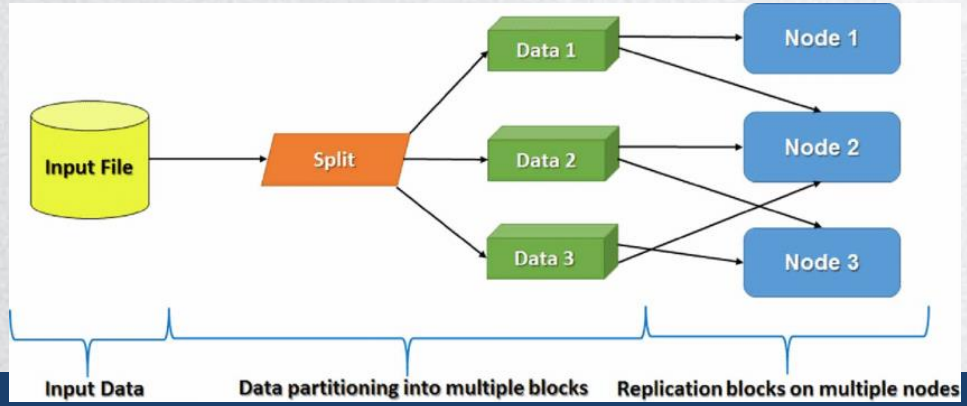
Hadoop与Spark入门

- 先了解一下Hadoop： Hadoop软件框架
  - Hadoop软件框架，包含如下主要模块：
  - (1) Hadoop Common，这个模块包含了其它模块需要的库函数和实用函数
  - (2) Hadoop Distributed File System (HDFS)，这是在由普通服务器组成的集群上运行的分布式文件系统，支持大数据的存储；通过多个节点的并行I/O，提供极高的吞吐能力
  - (3) Hadoop MapReduce，是一种支持大数据处理的编程模型
  - (4) Hadoop YARN，这是Hadoop 2.0的基础模块，它本质上是一个资源管理和任务调度软件框架。它把集群的计算资源管理起来，为调度和执行用户程序提供支持



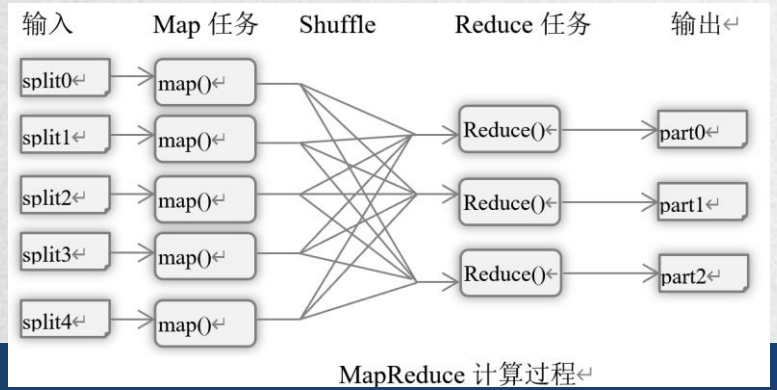
Hadoop与Spark入门

- 先了解一下Hadoop：HDFS
  - Hadoop分布式文件系统(Hadoop Distributed File System, HDFS)，是一个分布式的、高度可扩展的文件系统
  - 一个HDFS集群，一般由一个NameNode和若干DataNode组成，分别负责元信息的管理和数据块的管理
  - HDFS支持TB级甚至PB级大小文件的存储，它把文件划分成数据块(Block)，分布到多台机器上进行存储
  - 为了保证系统的可靠性，HDFS把数据块在多个节点上进行复制(Replicate)



Hadoop与Spark入门

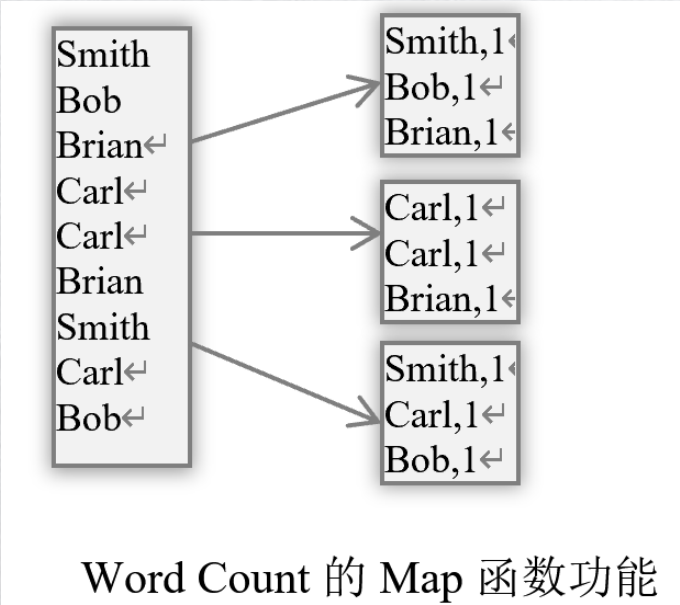
- 先了解一下Hadoop： MapReduce
- MapReduce计算模型
  - MapReduce并行编程模型把计算过程分解为两个主要阶段，即Map 阶段和Reduce 阶段。
  - Map 函数处理<Key, Value>对，产生一系列的中间<Key, Value>对
  - Reduce 函数合并所有具有相同Key值的中间键值对，计算最终结果
  - MapReduce计算模型，可以形式化地表达成Map: <*k1*, *v1*> -> list<*k2*,*v2*>, Reduce: <*k2*, list(*v2*)> ->list<*k3*, *v3*>



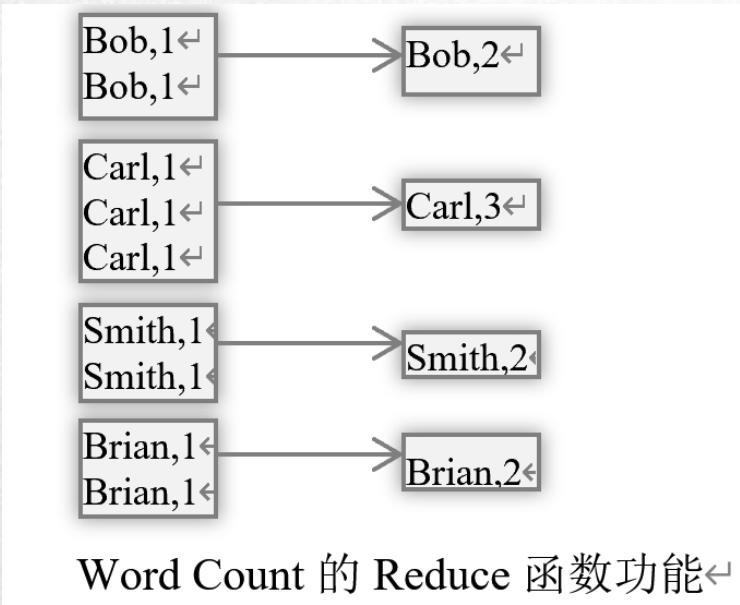
具体请参考后续实例

Hadoop与Spark入门

- 先了解一下Hadoop: Word Count实例
  - MapReduce Word Count实例——Map和Reduce函数



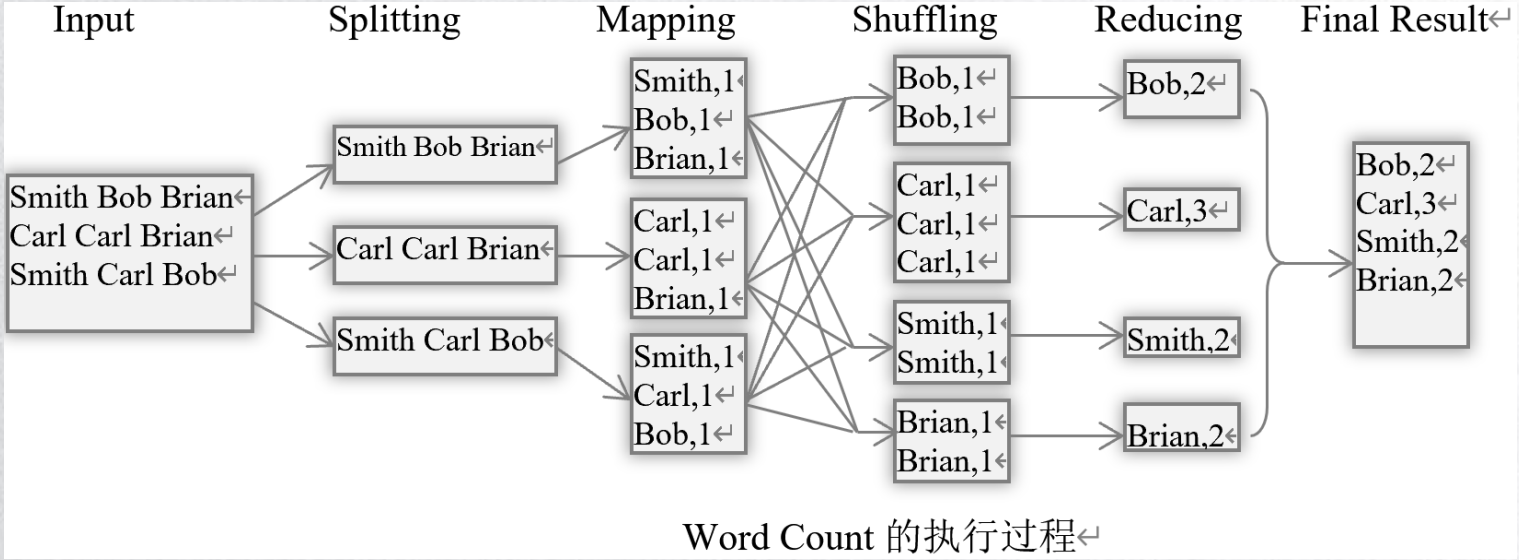
Word Count 的 Map 函数功能



Word Count 的 Reduce 函数功能

Hadoop与Spark入门

- 先了解一下Hadoop： Word Count实例
  - MapReduce Word Count实例 ——总流程



Word Count 的执行过程

Hadoop与Spark入门

Hadoop与Spark入门

- 先了解一下Hadoop：Hadoop的成绩
  - 在2008年，Yahoo公司使用一个拥有910个节点的Hadoop集群，在209秒内完成了1TB数据的排序，打破了Terabyte Sort 评测基准的记录(之前的记录是297秒)
  - 这个事件的重要意义在于，这是用Java编写的开源程序，首次赢得Terabyte Sort评测基准
  - 在2011年3月，Media Guardian媒体集团，把年度创新奖(Innovation Awards of the Year)颁发给了Hadoop项目；评审委员会认为，Hadoop项目是“21世纪的瑞士军刀(Swiss Army Knife of the 21st Century)”
  - Hadoop平台已经成为大数据处理的标准工具；其重要作用被越来越多的人认识到

Hadoop: “The Swiss Army Knife” of the 21st Century

By: David Russell Schilling | December 20th, 2014

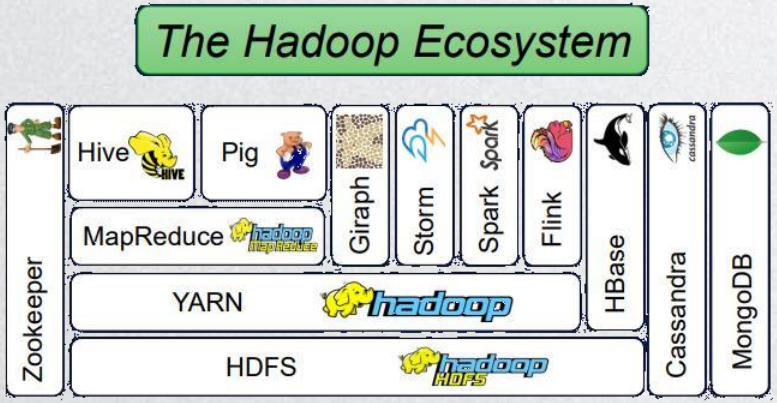
Storing & Processing Up To 25 Petabytes Of Data Reliably

We’ve all heard of “big data”; every day sees the creation of huge amounts of new data, 80% of which is unstructured meaning the information doesn’t come in neat Excel spreadsheets like row and column with names that help a user quickly understand the data.

Unstructured data is like a newspaper article or book in which the only structure available s the grammatical makeup of sentences. Algorithms must be to help identify key concepts, their relationships and there frequency along with other variables in order to assign meaning to the information that is intelligible to humans.

Hadoop与Spark入门

- 先了解一下Hadoop：Hadoop的生态系统
  - 除了简单的SQL汇总之外，研究人员已经把OLAP、数据挖掘、机器学习、信息检索、多媒体数据处理、科学数据处理、图数据处理等复杂的数据处理和分析算法，移植到Hadoop平台上(翻译成MapReduce Job)
  - Hadoop不仅仅是一个处理非结构化数据的工具，当数据按照一定格式进行适当组织以后，Hadoop平台也可以处理结构化数据
  - Hadoop平台以及Hadoop上的各种工具构成了一个生态系统，完成各种大数据集的处理任务



Hadoop与Spark入门

- 先了解一下Hadoop：Hadoop的局限性
- Hadoop最初是为大数据的批处理设计的，它的关注点，在于以尽量高的吞吐量处理这些数据。
  - (1) 它仅仅支持一种计算模型，即MapReduce：MapReduce计算模型的表达能力有限；复杂的数据处理任务，比如机器学习算法和SQL连接查询等，很难表达为一个MapReduce Job，而是需要翻译成一系列的MapReduce Job，这些Job一个接一个地执行
  - (2) 由于MapReduce作业在Map阶段和Reduce阶段执行过程中，需要把中间结果存盘；而且在MapReduce作业间，也需要通过磁盘实现作业之间的数据交换
    - 通过磁盘进行数据交换，效率低下，影响了查询的执行效率
    - 在这个计算模型上，很难再继续减小查询的响应时间

Hadoop与Spark入门

Hadoop与Spark入门

- Spark简介
- Some changes from 2000 to 2010
  - (1)RAM was very costly in 2000, RAM is primary source of data and we use disk for fallback in 2010
  - (2)SQL was the only dominant way for data analysis in 2000, NoSQL is real alternative in 2010
  - (3)Batch processing system ruled the world, Volume was big concern compare to velocity in 2000, Needs of real time are as much important as batch processing, Velocity is as much concern as volume in 2010
  - and more……

Hadoop与Spark入门

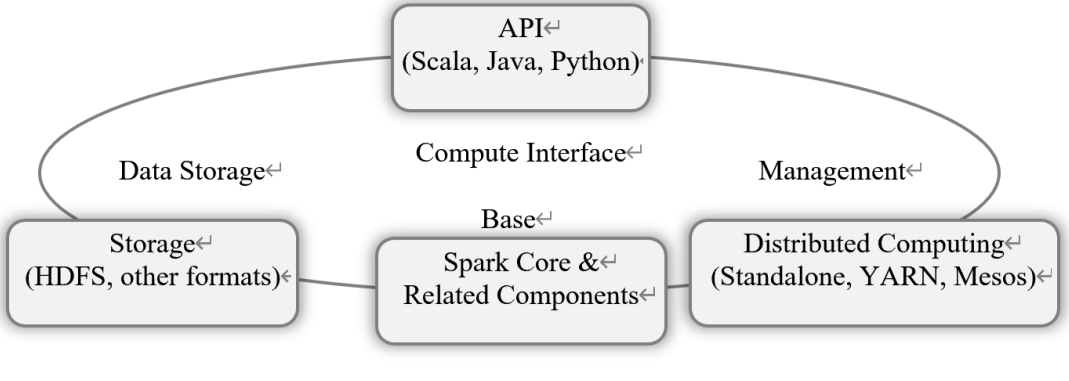
- Spark简介
  - Apache Spark是一个开源的大数据处理框架
  - 它与Hadoop并驾齐驱，是当前主流的大数据处理框架之一
  - Spark于2009年由UC Berkeley的AMP实验室开发，并且在2010年开源，成为一个Apache项目
  - Spark是一个速度快、易用、通用的集群计算系统，能够与Hadoop生态系统和数据源良好地兼容。
    - Exploit RAM as much as disk(基于内存数据处理技术)
    - Handle both batch processing and real time
    - Plays well and compatible with Hadoop

Hadoop与Spark入门

- Spark简介
  - Multi language API
  - Written in Scala but API is not limited to it
  - Offers API in
    - Scala
    - Java
    - Python
  - You can also do SQL using SparkSQL

Hadoop与Spark入门

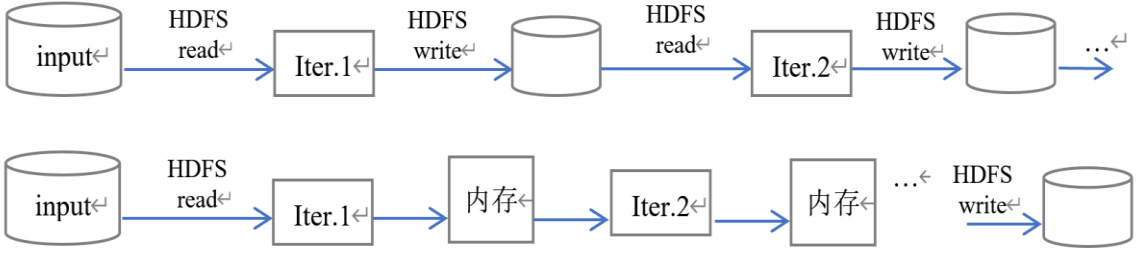
- Spark软件架构
  - 包含4个主要部分，分别是
  - Spark的核心组件和相关组件
    - Spark的核心组件和相关组件，包括Spark Core，以及Spark Streaming、Spark SQL、GraphX、MLLib等组件
  - 数据存储(Data Storage)
  - 应用程序编程接口(API)
  - 和资源管理框架(Management Framework)



Spark 软件架构

Hadoop与Spark入门

- Spark的主要优势
  - (1) Spark的数据处理速度比Hadoop要快很多，这主要得益于其基于内存的数据处理技术
  - (2) 除了Map函数和Reduce函数，Spark提供其它大量的数据原语操作，使用这些操作可以表达复杂的计算任务
  - (3)Spark在一个平台上支持SQL查询、流数据处理、图数据处理、机器学习，它们都基于RDD(下文介绍)数据结构进行操作

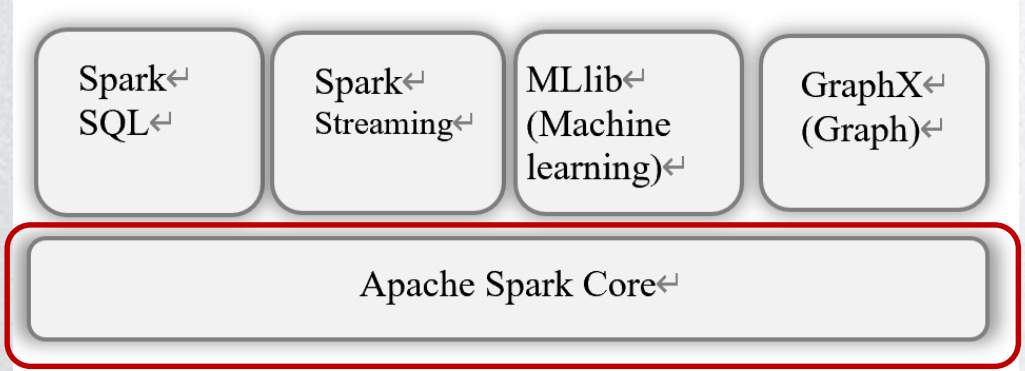


Hadoop MapReduce vs. Spark

Hadoop与Spark入门

Hadoop与Spark入门

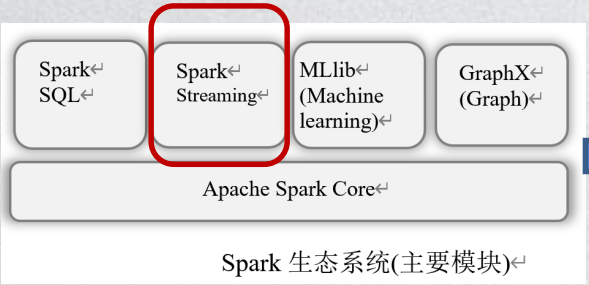
- Spark生态系统：整个Spark软件系统，包含核心模块(Spark Core)，以及若干数据处理分析模块
  - Spark核心模块，是整个系统进行大规模并行和分布式数据处理的基础
  - 它的主要功能是，内存管理和容错保证、集群环境下的作业调度和监控、以及和存储系统的接口和交互等



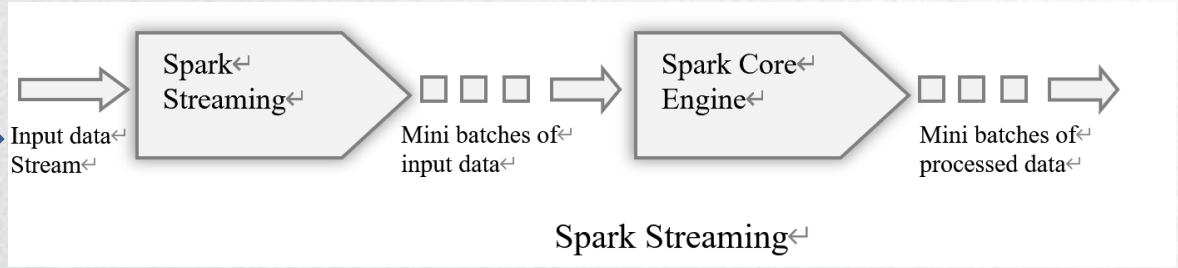
Spark 生态系统(主要模块)

Hadoop与Spark入门

- Spark生态系统：流数据处理(Spark Streaming)
  - 用于处理实时流数据，比如Web服务器的日志文件(Log File)、社交媒体如Twitter、以及各种消息队列如Kafka消息队列等
  - 它采用小批量(Micro Batch)数据处理的方式，即把接收的数据流，分解成一系列的小的RDD(下文介绍)，交给Spark引擎进行处理，从而实现流数据处理，处理结果也是以批量的方式生成的(In Batch)



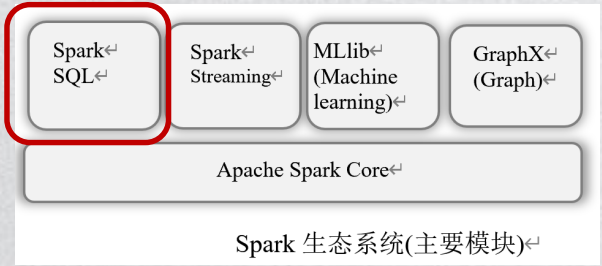
Spark 生态系统(主要模块)



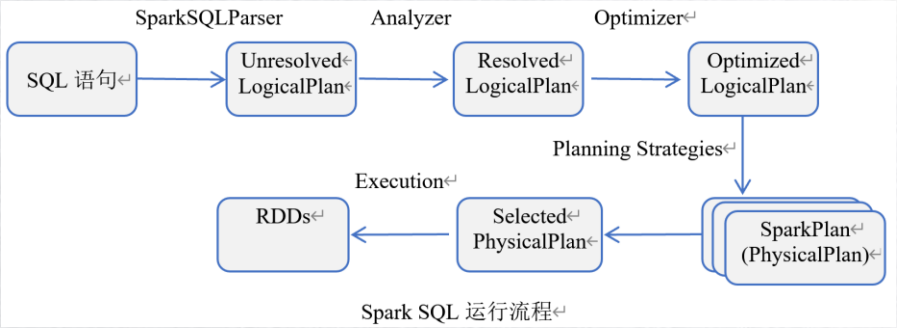
Spark Streaming

Hadoop与Spark入门

- Spark生态系统：SQL查询与结构化数据处理(Spark SQL)
  - 该模块把Spark数据集通过JDBC API暴露出来，让客户程序可以在上面运行SQL查询，也可以把传统的BI(Business Intelligence)和可视化工具连接到该数据集，利用JDBC进行数据查询、汇总和展示等
  - Spark SQL模块支持不同外部数据源(比如JSON、Parquet列存储、以及关系数据库等)的导入、转换和装载，并且支持即席(Ad-Hoc)查询



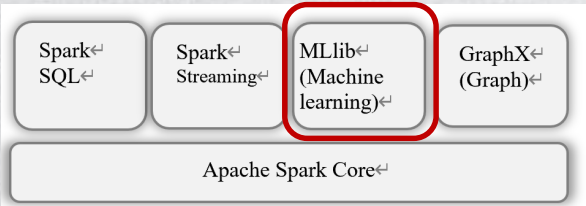
Spark 生态系统(主要模块)



Spark SQL 运行流程

Hadoop与Spark入门

- Spark生态系统：机器学习(Spark MLLib)
  - MLLib是Spark生态系统里可扩展的机器学习模块
  - 它已经实现了众多的算法，包括分类(Classification)、聚类(Clustering)、回归(Regression)、协同过滤(Collaborative Filtering)、降维(Dimensionality Reduction)等



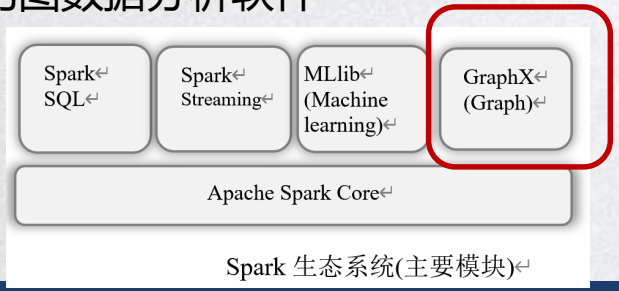
Spark 生态系统(主要模块)

MLLib 支持的机器学习算法

	离散数据	连续数据
有监督学习	Classification、LogisticRegression、SVM、DecisionTree、RandomForest、NaiveBayes、MultilayerPerceptron、GBT、OneVsRest…	LinearRegression、Regression DecisionTree、RandomForest、GBT、AFTSurvivalRegression、IsotonicRegression…
无监督学习	Kmeans Clustering、BisectingKMeans、GaussianMixture、PowerIterationClustering、Latent Dirichlet Allocation (LDA)…	Dimensionality Reduction、Matrix Factorization、PCA、SVD、Alternating Least Squares (ALS)、Weighted Least Squares (WLS)…

Hadoop与Spark入门

- Spark生态系统：图数据处理(Spark GraphX)
  - GraphX支持图数据的并行处理；用户可以利用该模块，对图数据进行探索式分析(Exploratory Analysis)以及迭代式计算(Iterative Graph Computation)
  - GraphX对RDD进行了扩展，称为RDPG(Resilient Distributed Property Graph)；RDPG是一个把不同属性赋予各个节点和各条边的有向图(Multi Graph)
  - 为了支持图数据的处理，GraphX提供了一系列操作供用户使用，包括子图(Sub Graph)、顶点连接(Join Vertices)、消息聚集(Aggregate Message)等，还提供了Pregel (Google公司的图数据处理软件)API的变种
    - 在此基础上，GraphX包含了经典的图处理算法，比如PageRank等
    - 方便在此之上开发更加复杂的图数据分析软件



Spark 生态系统(主要模块)

Hadoop与Spark入门

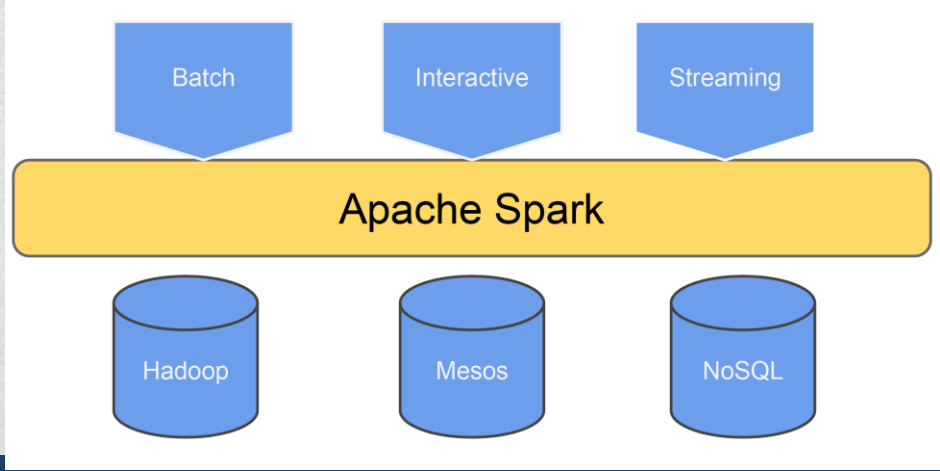
- Spark & Hadoop Ecosystem的配合与竞争

Hadoop	Spark
HDFS ✓✓	None：Spark依赖于HDFS
Hive ✓	SparkSQL ✓✓
Apache Mahout：过期不再维护	MLLib ✓✓
Apache Impala ✓	SparkSQL ✓✓
Apache Giraph ✓	GraphX ✓✓
Apache Storm ✓✓	Spark Streaming ✓

Hadoop与Spark入门

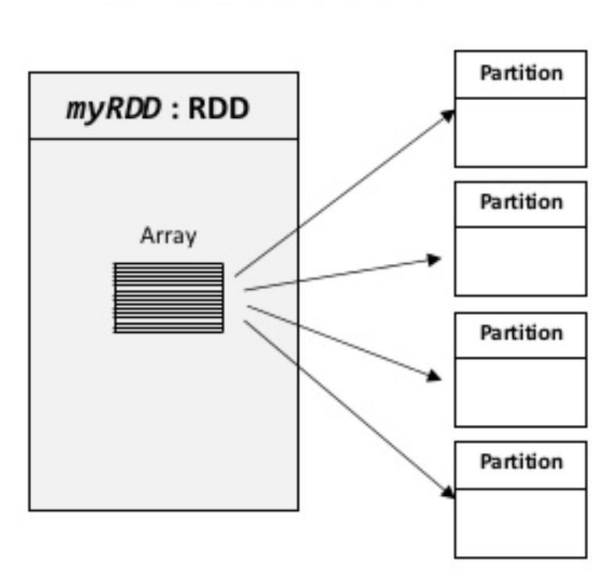
Hadoop与Spark入门

- Spark is a Unified Platform for Big Data Apps
  - (1)All different processing systems in spark share same abstraction called RDD
    - 下文介绍
  - (2)As they share same abstraction you can mix and match different kind of processing in same application



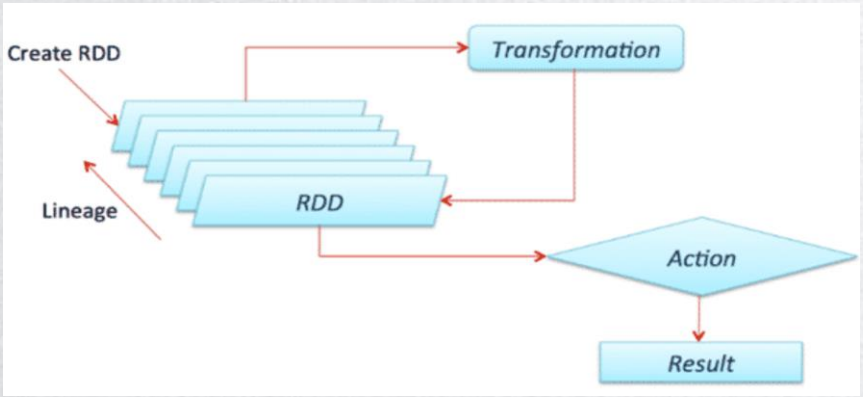
Hadoop与Spark入门

- RDD及其处理
  - 弹性分布数据集(Resilient Distributed Dataset, RDD)是Spark软件系统的核心概念
  - 它是一个容错的、不可更新的(Immutable)分布式数据集，支持并行处理
  - 简单来讲，RDD可以看作数据库里的一张表；Spark把隶属于一个RDD的数据，划分成不同的分区(Partition)
  - 分区是RDD的下级概念
  - 对RDD进行分区，并且把分区分布到集群环境(不同节点)，有利于对数据进行并行处理
  - 我们可以从外部数据源比如HDFS、HBase等创建RDD数据集，也可以在Spark程序中，直接创建RDD，比如把一个List转换成一个RDD



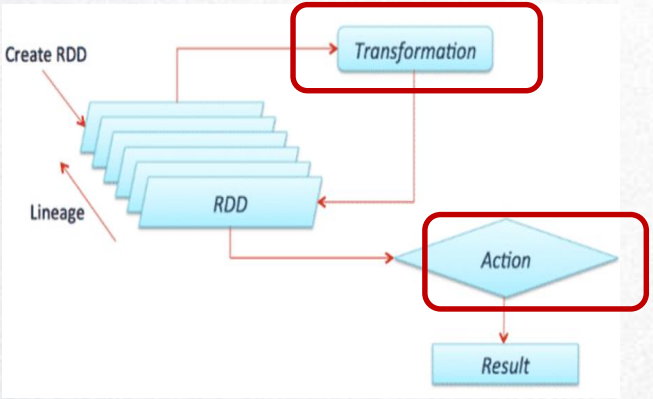
Hadoop与Spark入门

- RDD及其处理
  - RDD采用基于血统(Lineage)的容错机制，也就是它记住每个RDD是如何从其它RDD转换(Transformation)而来的
  - 当某个RDD损坏的时候，Spark系统从上游RDD重新计算和创建本RDD的数据
  - RDD数据集是不可更新的(Immutable)
  - 我们可以对一个RDD进行转换(Transformation)，但是转换之后会返回一个新的RDD，而原来的RDD保持不变



Hadoop与Spark入门

- RDD及其处理
  - RDD数据集支持两种操作，分别是转换(Transformation)和动作(Action)
  - 对一个RDD施加转换操作，将返回一个新的RDD(New RDD)
    - 典型的转换操作包括map、filter、flatMap、groupByKey、reduceByKey、aggregateByKey、pipe、以及coalesce等操作
  - 动作操作施加于RDD数据集，经过对RDD数据集的计算，返回一个新的结果(New Result)，这个结果将返回客户端
    - 典型的动作包括reduce、collect、count、first、take、countByKey、以及foreach等操作



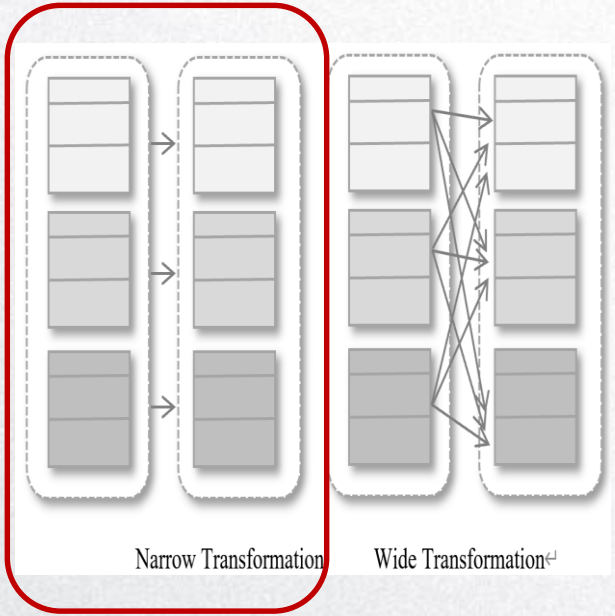
Hadoop与Spark入门

- RDD的transformations & actions

Transformations	*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]
	sample(*fraction* : Float)	:	RDD[T] ⇒ RDD[T] (Deterministic sampling)
	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))]
	cogroup()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (Seq[V], Seq[W]))]
	crossProduct()	:	(RDD[T], RDD[U]) ⇒ RDD[(T, U)]
	mapValues(*f* : V ⇒ W)	:	RDD[(K, V)] ⇒ RDD[(K, W)] (Preserves partitioning)
	sort(*c* : Comparator[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	partitionBy(*p* : Partitioner[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
Actions	*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] (On hash/range partitioned RDDs)
	save(*path* : String)	:	Outputs RDD to a storage system, *e.g.*, HDFS

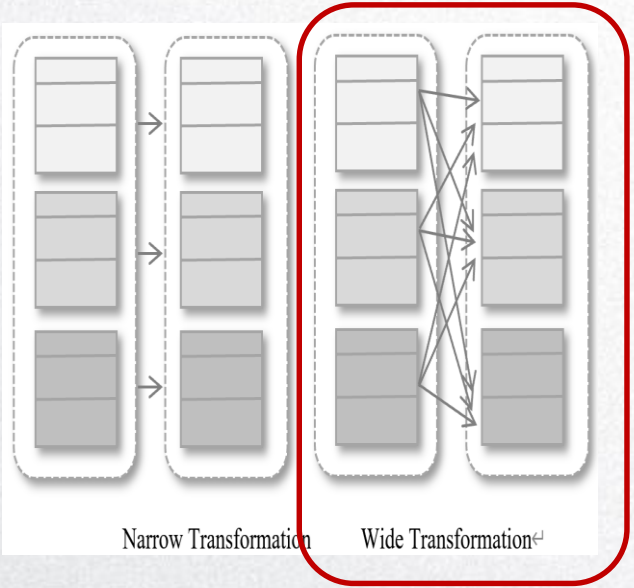
Hadoop与Spark入门

- 宽依赖和窄依赖
  - 在 DAG 里， 父 子 RDD 的 各 个 分 区 (Partition)之间，有两种依赖关系，分别是宽依赖和窄依赖
  - 窄依赖，指的是每个父RDD的分区，最多被一个子RDD的分区使用到(父RDD的某个分区的数据经过转换操作，产生子RDD的分区)
  - （请参见后面的示例）



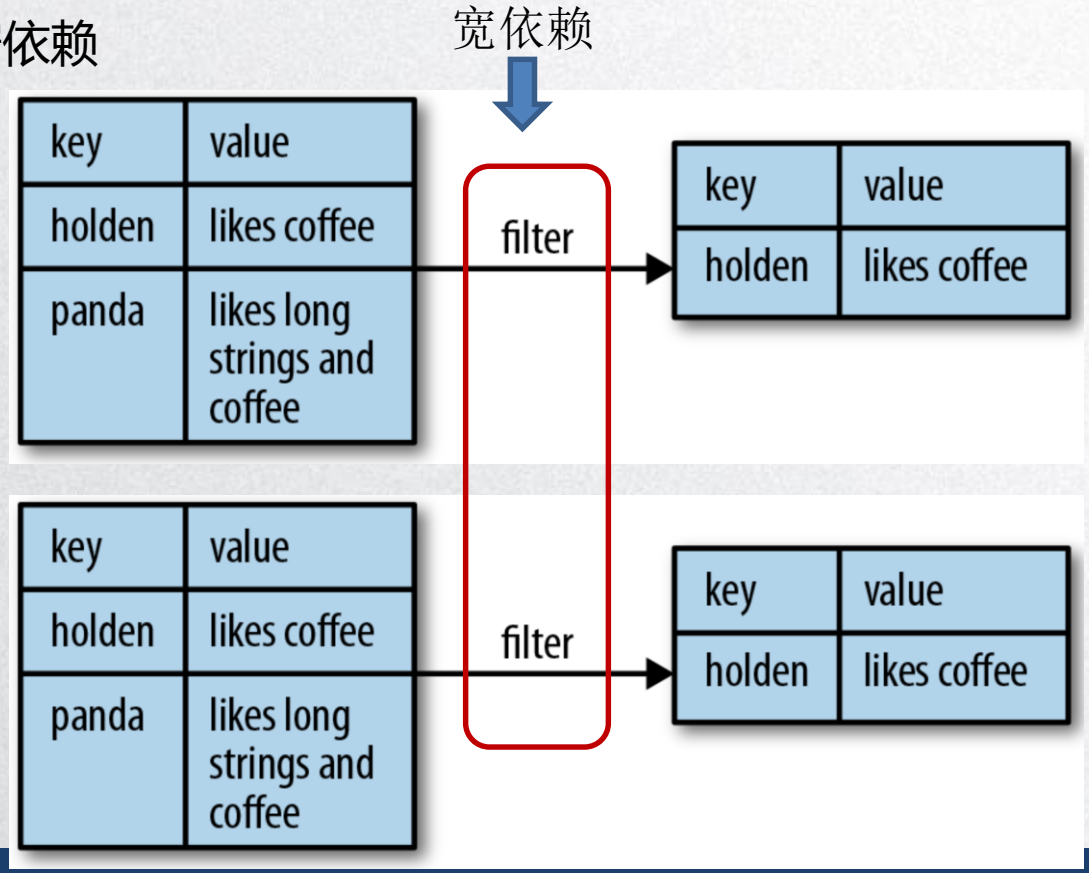
Hadoop与Spark入门

- 宽依赖和窄依赖
  - 在 DAG 里， 父 子 RDD 的 各 个 分 区 (Partition)之间，有两种依赖关系，分别是宽依赖和窄依赖
  - 宽依赖，指的是多个子RDD的分区，依赖于同一个父RDD的分区的情形
  - groupByKey 、 reduceByKey 、 sortByKey等操作，需要用到宽依赖，以便获得正确的结果
  - （请参见后面的示例）



Hadoop与Spark入门

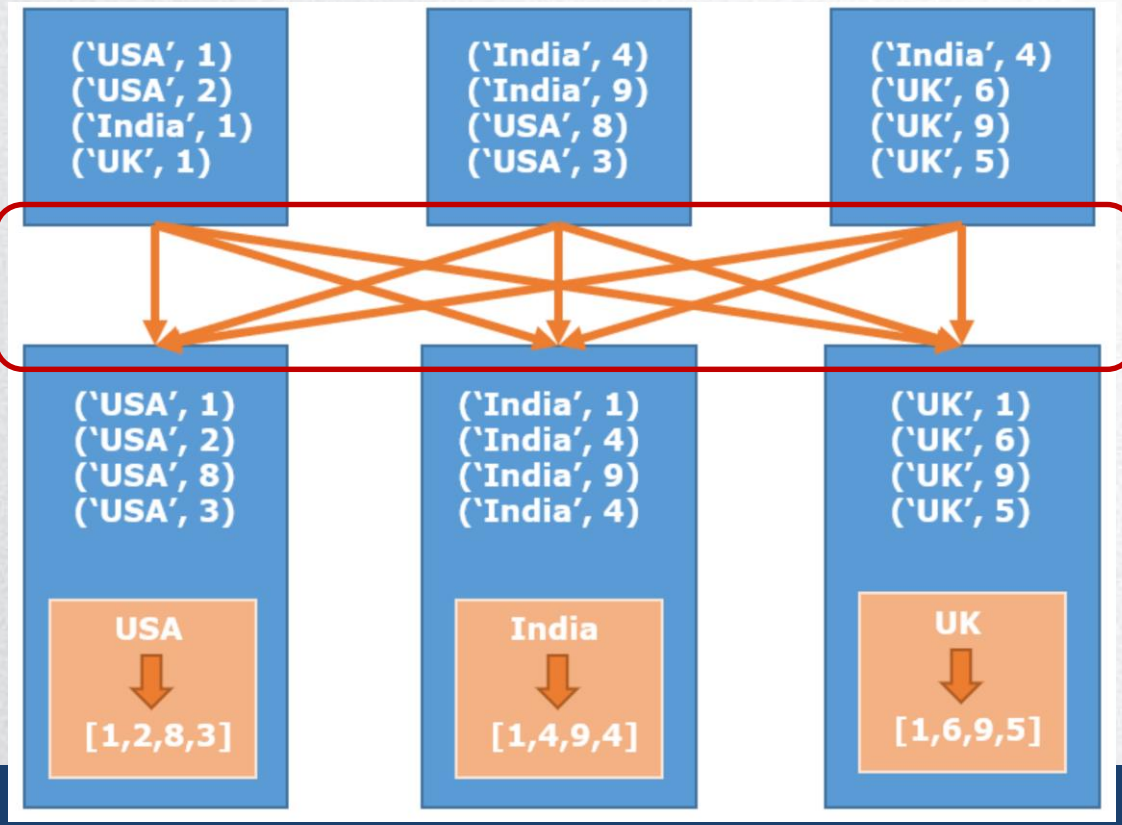
- Filter过滤：窄依赖



– Partition0

– Partition1

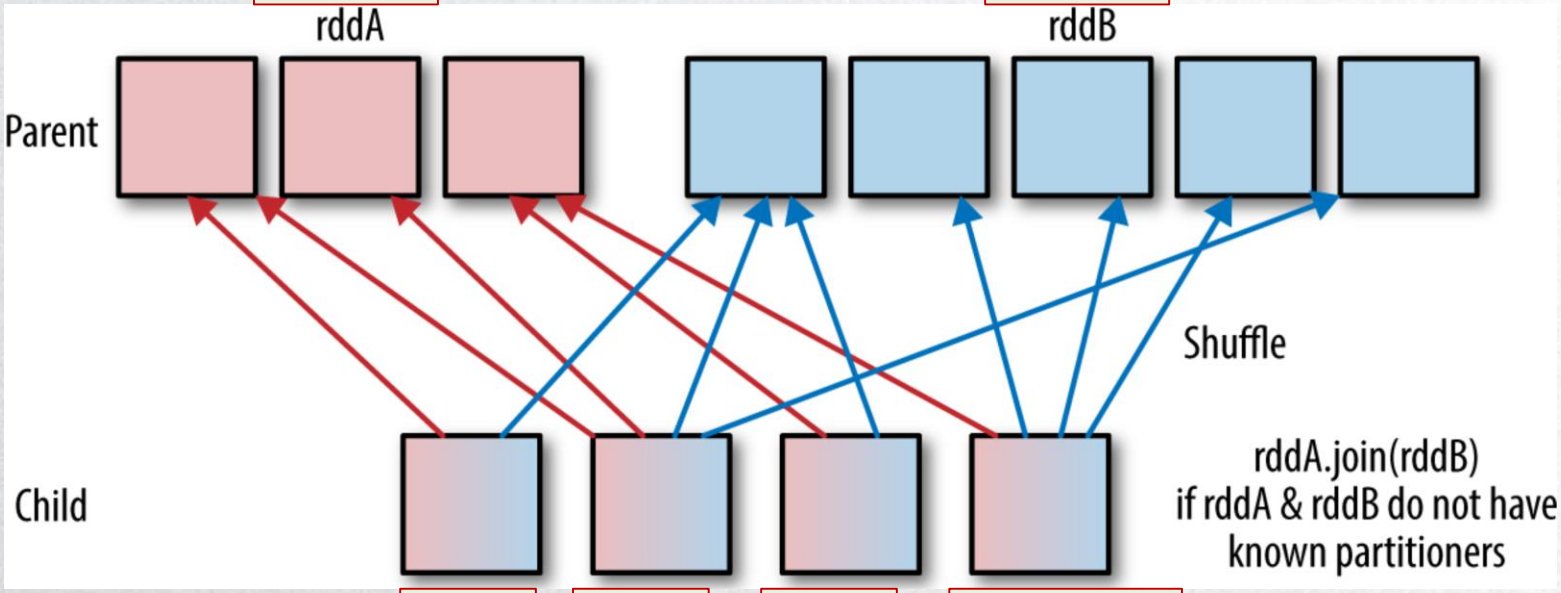
- Group by 分组
  - 宽依赖



Hadoop与Spark入门

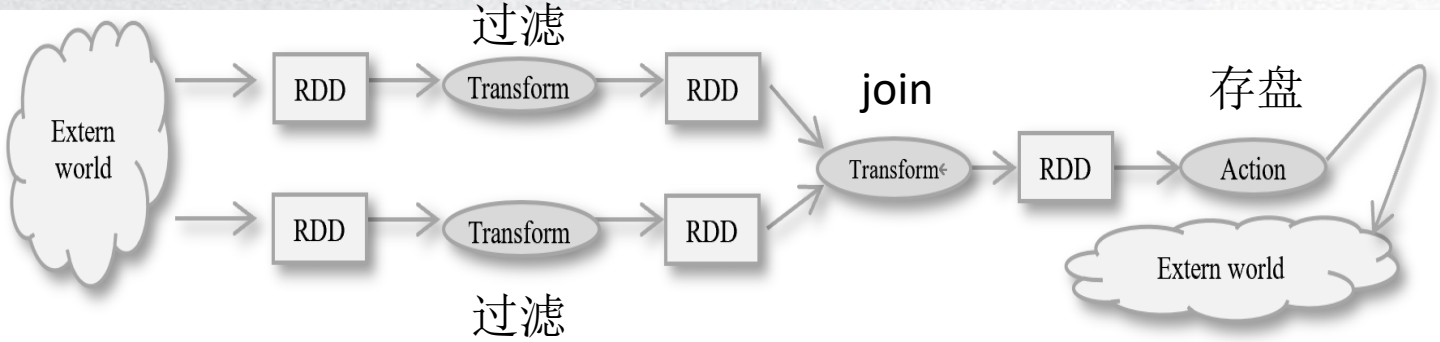
统计每个院系的学生人数

- 两个表格(RDD)Join：宽依赖



Hadoop与Spark入门

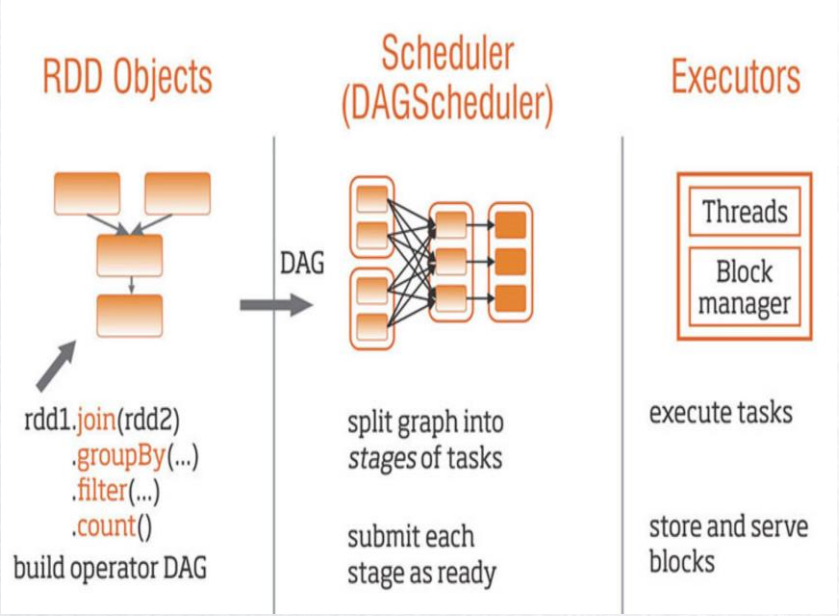
- DAG——表达复杂的数据处理流程
  - 一系列RDD、转换操作、动作操作等，一起构成一个 DAG，用以表达复杂的计算
  - 如下实例，装载两个表格，分别过滤，然后进行Join操作，最后保存到HDFS



一个典型的 DAG

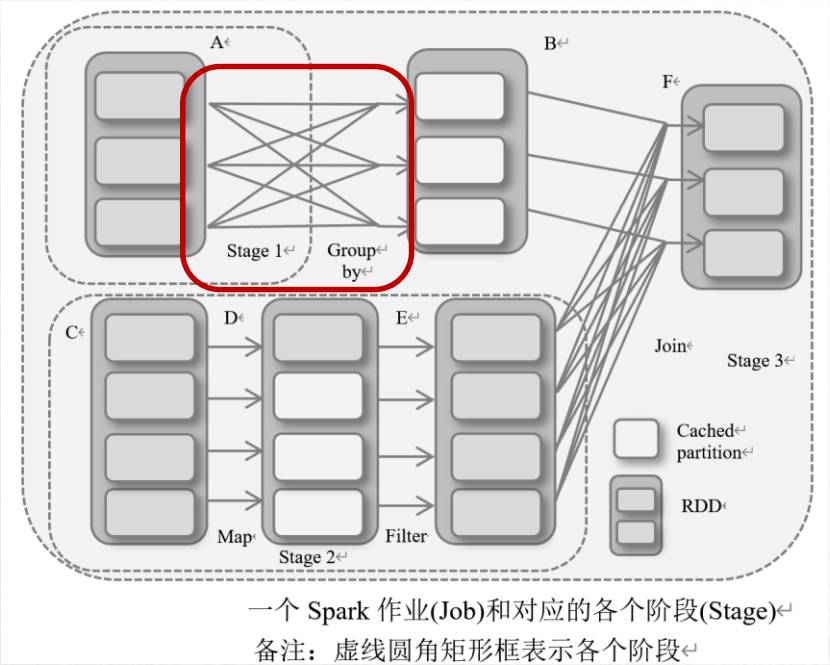
Hadoop与Spark入门

- DAG的调度——即Job调度
  - DAGScheduler 是面向阶段(Stage-Oriented)的DAG执行调度器
  - DAGScheduler使用作业(Job)和阶段(Stage)等基本概念，进行作业调度
  - DAGScheduler检查依赖的类型，它把一系列窄依赖RDD组织成一个阶段
  - 而对于宽依赖，则需要跨越连续的阶段
  - 参考后续示例



Hadoop与Spark入门

- DAG的调度
  - 在Stage 1里，对RDD A(是一张表，整个是一个RDD，划分成一系列分区，分布在各个节点上)进行Groupby处理，生成RDD B
  - RDD A和RDD B之间的依赖关系是宽依赖（因为是Group BY操作）

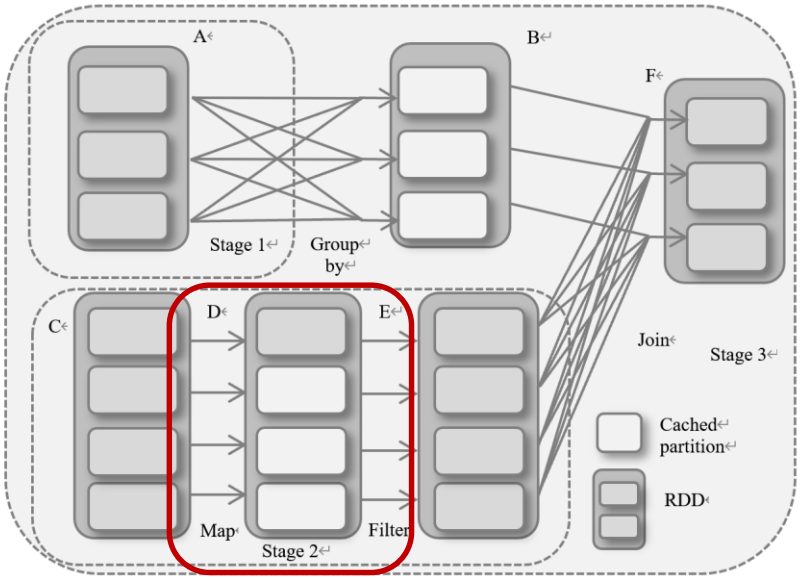


一个 Spark 作业(Job)和对应的各个阶段(Stage)

备注：虚线圆角矩形框表示各个阶段

Hadoop与Spark入门

- DAG的调度
  - 在Stage 2里，对RDD C(是另外一张表，整个是一个RDD，划分成一系列分区，分布在各个节点上)进行map、filter等操作，依次生成RDD D、RDD E等中间结果
  - RDD C与RDD D之间、以及RDD D与RDD E之间的依赖关系是窄依赖

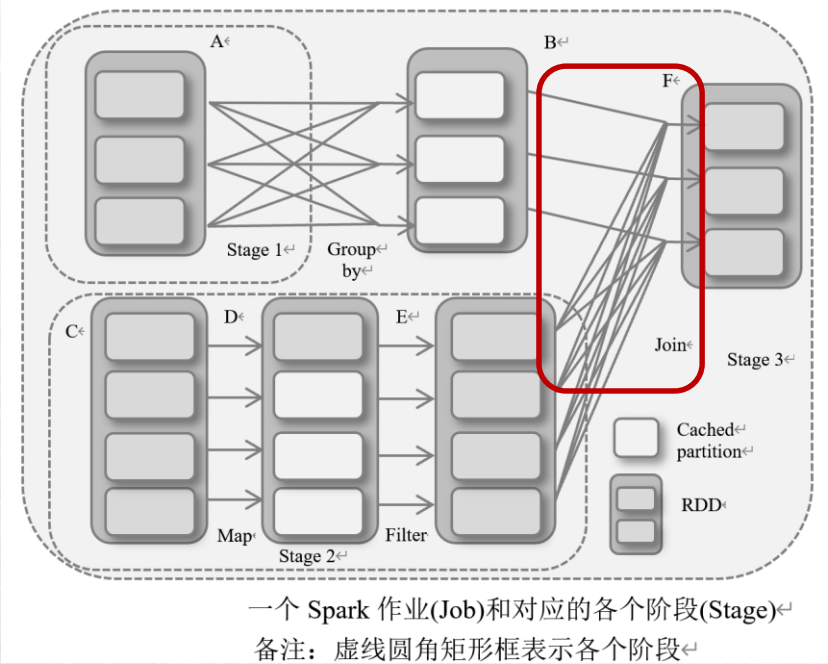


一个 Spark 作业(Job)和对应的各个阶段(Stage)

备注：虚线圆角矩形框表示各个阶段

Hadoop与Spark入门

- DAG的调度
  - 在Stage 3里，对RDD B和RDD E做join操作
  - RDD B和RDD F之间的依赖关系是窄依赖，而RDD E和RDD F之间的依赖关系是宽依赖
  - 通过这三个阶段的处理，我们实现表A和表B之间的连接操作



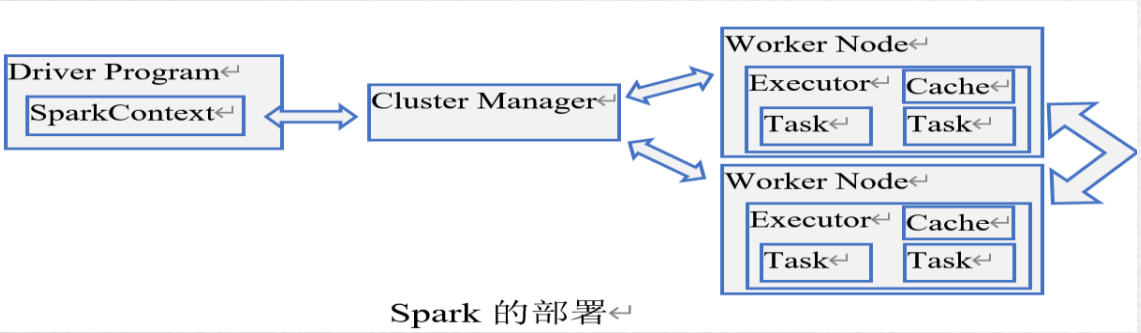
一个 Spark 作业(Job)和对应的各个阶段(Stage)

备注：虚线圆角矩形框表示各个阶段

Hadoop与Spark入门

Hadoop与Spark入门

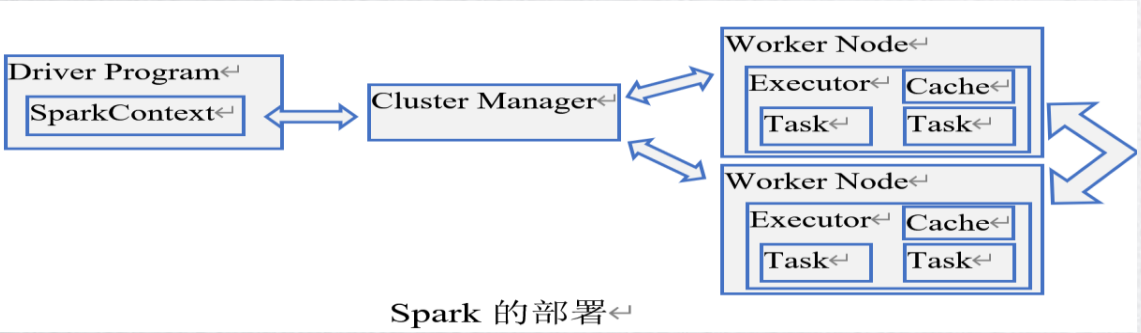
- Spark的部署
  - Driver负责运行应用程序的main函数，创建SparkContext
  - Spark Context持有到Cluster Manager的连接；每个Spark应用程序都是一组独立的进程，由Spark Context协调运行
    - 当Spark Context连接到Cluster Manager以后，它申请获得从节点上(Slaves)的一系列Executor
    - 这些Executor独立工作(运行Task)又相互沟通，负责完成整个作业(Job)，即应用程序的处理要求



Spark 的部署

Hadoop与Spark入门

- Spark的部署
  - Worker是在Cluster的从节点上运行的进程，用于执行应用程序
    - 一个应用程序启动以后，它在各个Worker上申请Executors，来真正执行应用程序的处理逻辑(Task)
  - Cluster Manager负责为应用程序分配资源
    - Spark支持3种Cluster Manager，即Standalone、Mesos和YARN
    - 这些Cluster Manager在调度算法、安全性、监控能力等方面有差别



Spark 的部署

Hadoop与Spark入门

Hadoop与Spark入门

- Spark程序示例——WordCount

Scala

```
val textFile =
sc.textFile("/path/to/file.txt")

val counts = textFile
     .flatMap(line => line.split(" "))
     .map(word => (word, 1))
     .reduceByKey(_ + _)

counts.saveAsTextFile("/path/to/output")
```

- 每行按照空格切割
- 每个word变成<word,1>
- 累加每个word的count

Hadoop与Spark入门

- Spark程序示例——WordCount

Python

```
text_file =
sc.textFile("/path/to/file.txt")

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

counts.saveAsTextFile("/path/to/output")
```

- 每行按照空格切割
- 每个word变成<word,1>
- 累加每个word的count

Hadoop与Spark入门

- Spark程序示例——WordCount

Java 8

```
JavaRDD<String> textFile = sc.textFile("/path/to/file.txt");

JavaPairRDD<String, Integer> counts = lines
    .flatMap(line -> Arrays.asList(line.split(" ")));
    .mapToPair(w -> new Tuple2<String, Integer>(w, 1))
    .reduceByKey((a, b) -> a + b);

counts.saveAsTextFile("/path/to/output");
```

Hadoop与Spark入门

- SQL查询与结构化数据处理(Spark SQL)
  - DataFrame API

- A distributed collection of rows organized into named columns
  - You know the names of the columns and data types
- Like Pandas and R
- Unlike RDDs, DataFrame's keep track of their schema and support various relational operations that lead to more optimized execution
  - Catalyst Optimizer

Hadoop与Spark入门

- SQL查询与结构化数据处理(Spark SQL)
  - DataFrame API

SQL Statement:

```
SELECT name, avg(age)
FROM people
GROUP BY name
```

Python

```
sqlContext.table("people")
     .groupBy("name")
     .agg("name", avg("age"))
     .collect()
```

Can be written as:

Scala

```
sqlContext.table("people")
     .groupBy("name")
     .agg("name", avg("age"))
     .collect()
```

Java

```
Row[] output = sqlContext.table("<SQL>")
     .groupBy("name")
     .agg("name", avg("age"))
     .collect();
```

Hadoop与Spark入门

- SQL查询与结构化数据处理(Spark SQL)
  - DataFrame API——使用SQLContext执行SQL

Scala

```
val df = sqlContext.sql("<SQL>")
```

Python

```
df = sqlContext.sql("<SQL>")
```

Java

```
Dataset<Row> df = sqlContext.sql("<SQL>");
```

Hadoop与Spark入门

Hadoop与Spark入门

- Spark的成绩
  - 2014年，Spark击败Hadoop，在十分之一的节点数量上，以快三倍的速度，完成100TB数据的排序(Daytona Gray Sort Contest)
  - 同时，它也是目前为止在PB级数据排序方面最快的开源引擎

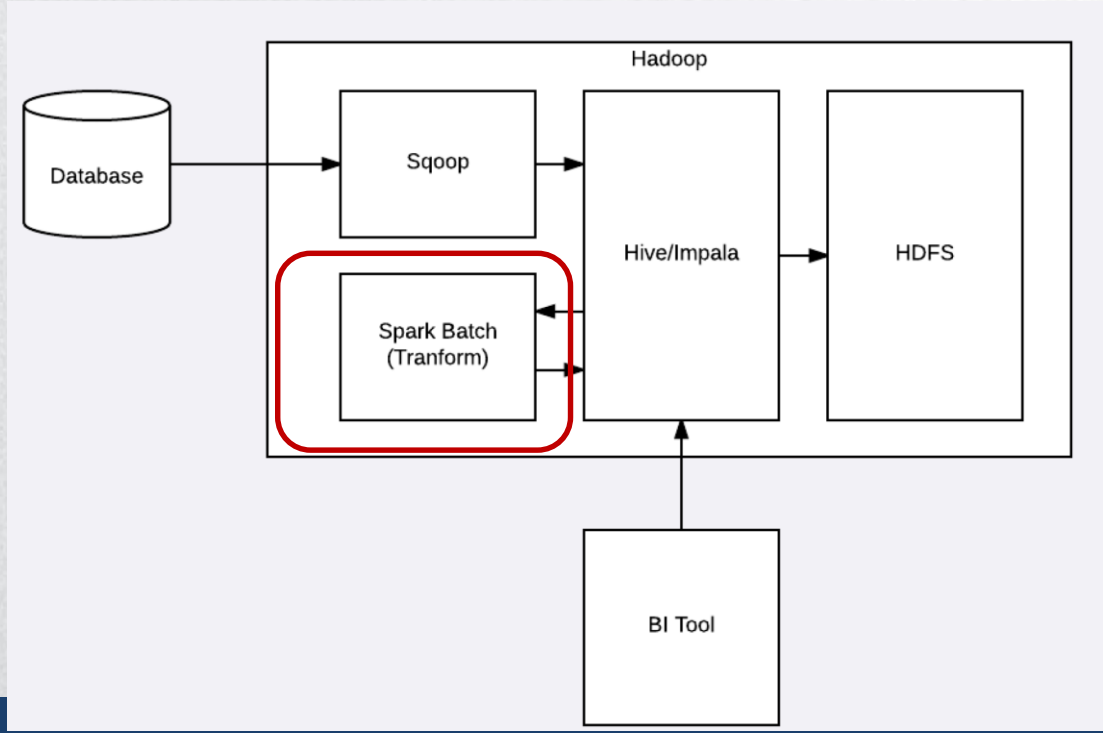
Hadoop 和 Spark100TB 数据排序结果的比较(数据来源 Databricks)

	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1 000 TB
Elapsed Time	72 mins	23 mins	234 mins
#Nodes	2 100	206	190
#Cores	50 400 physical nodes	6 592 virtualized nodes	6 080 virtualized nodes
Cluster disk throughput	3 150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	1.42 TB/min	4.27 TB/min	4.27 TB/min
Sort rate/node	0.67 GB/min	20.7 GB/min	22.5 GB/min

更快
更好地资源

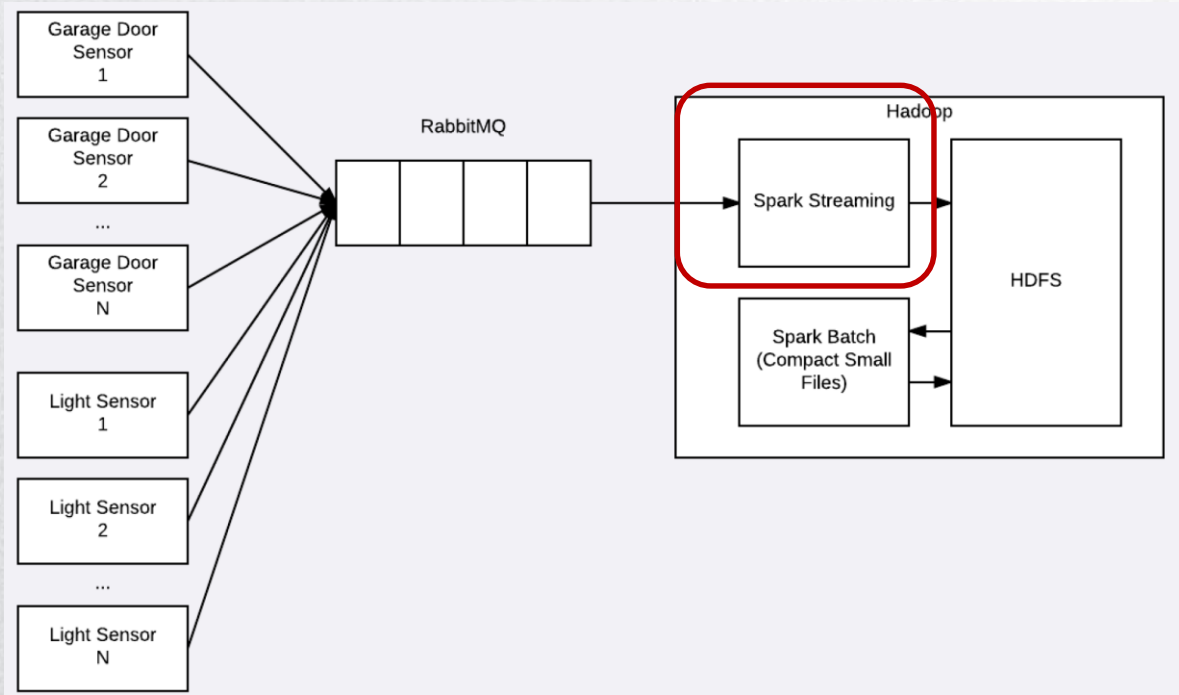
Hadoop与Spark入门

- Spark的use case——ETL



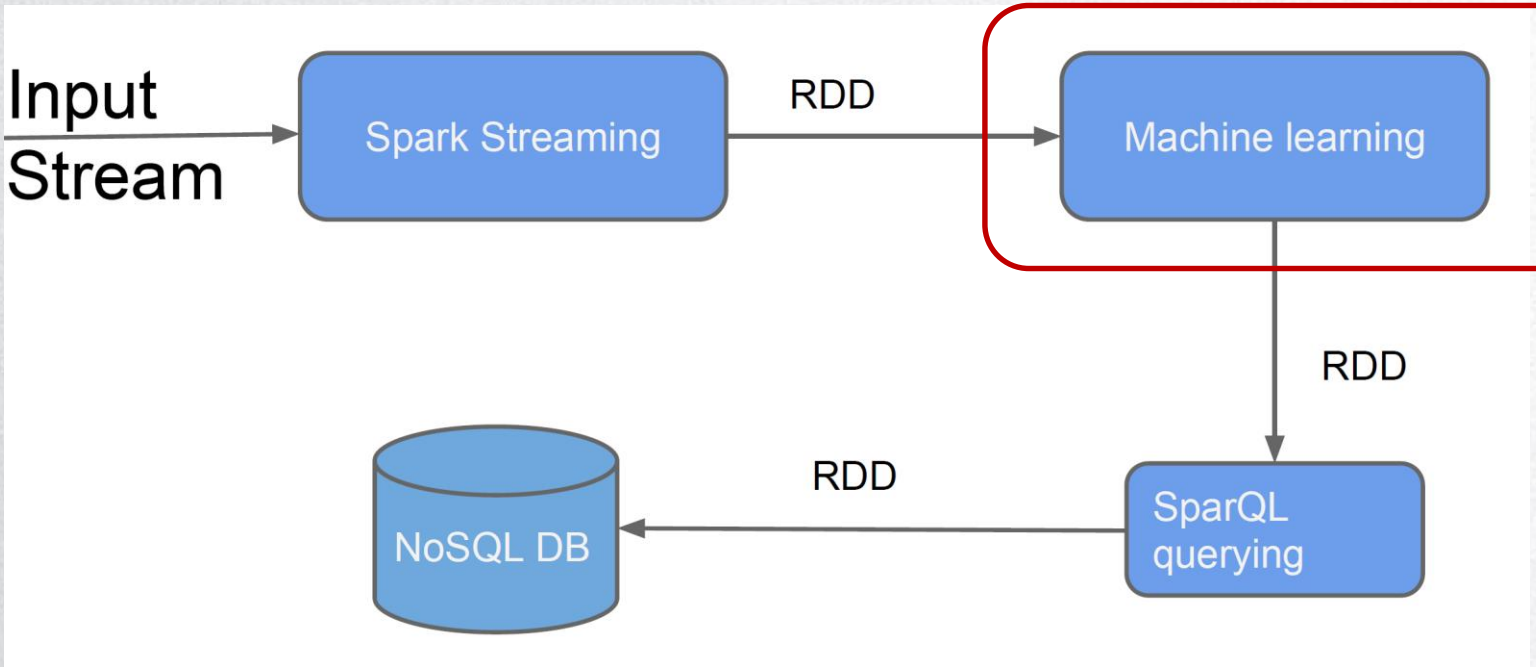
Hadoop与Spark入门

- Spark的use case——Streaming processing



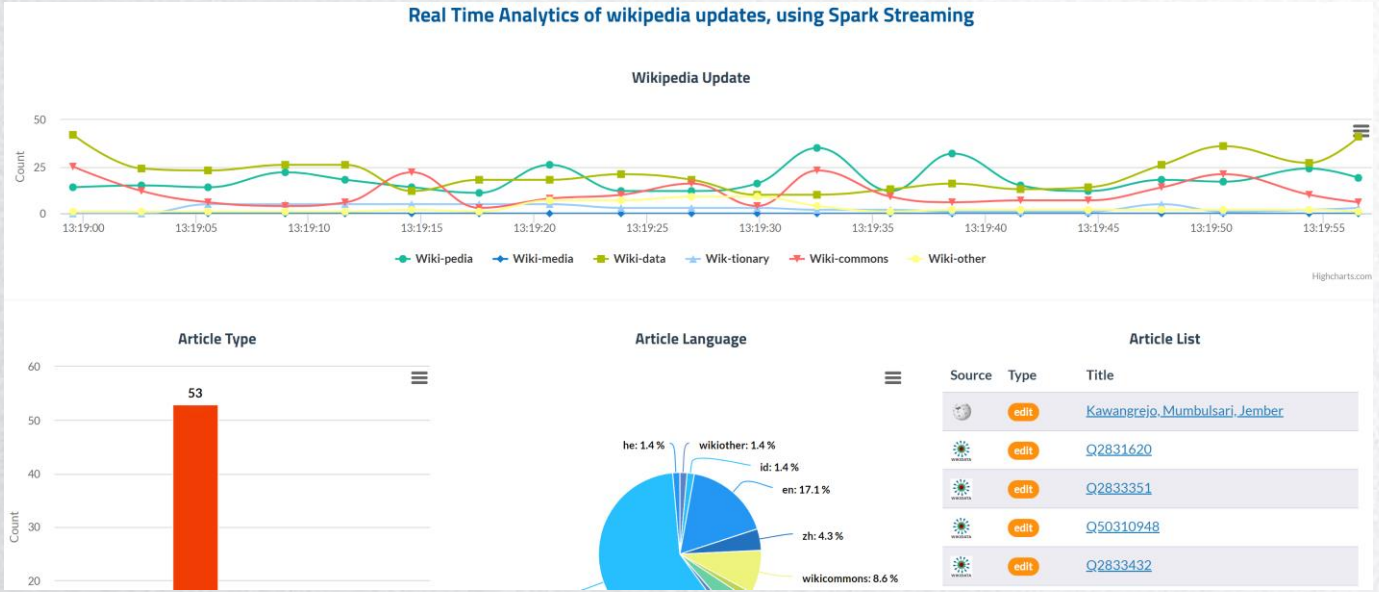
Hadoop与Spark入门

- Spark的use case——Spam detection



Hadoop与Spark入门

- Spark的use case——Real Time Analytics of Wikipedia updates, using Spark Streaming



https://bigdata.stratebi.com/spark-streaming/index.htm

Hadoop与Spark入门

- Spark的use case——and many more…
  - Spark可以在一个处理流程中
    - 基于同样的底层数据结构即RDD
    - 完成流数据处理、批量查询、交互式查询、图数据处理和机器学习模型的训练和应用