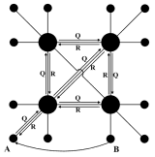


大型分布式系统案例实战 第7周

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

分布式系统之分布式计算框架

- Map-Reduce
- 实时计算框架
- 分布式计算调度平台

Map-Reduce

We want to count all the books in the library. You count up shelf #1, I count up shelf #2. That's map. The more people we get, the faster it goes.

Now we get together and add our individual counts. That's reduce.

用SQL来做类比，map象聚合(aggregate)查询中的group-by子句。Reduce则类似计算group-by起来的行的聚合函数(例如求平均等)。

Map Reduce是包含两个过程：Map过程和Reduce过程。每一个过程都包含键值对作为输入，程序员可以选择键和值的类型。

Map和Reduce的数据流是这样的：

Input ==> Map ==> Mapper Output ==> Sort and shuffle ==> Reduce ==> Final Output

(input) <k1, v1> -> **map** -> <k2, v2> -> **combine** -> <k2, v2> -> **reduce** -> <k3, v3> (output)

How Many Maps?

The number of maps is usually driven by the total size of the inputs, that is, the total number of blocks of the input files.

Thus, if you expect 10TB of input data and have a blocksize of 128MB, you'll end up with 82,000 maps

Map-Reduce

Googele论文中的模型

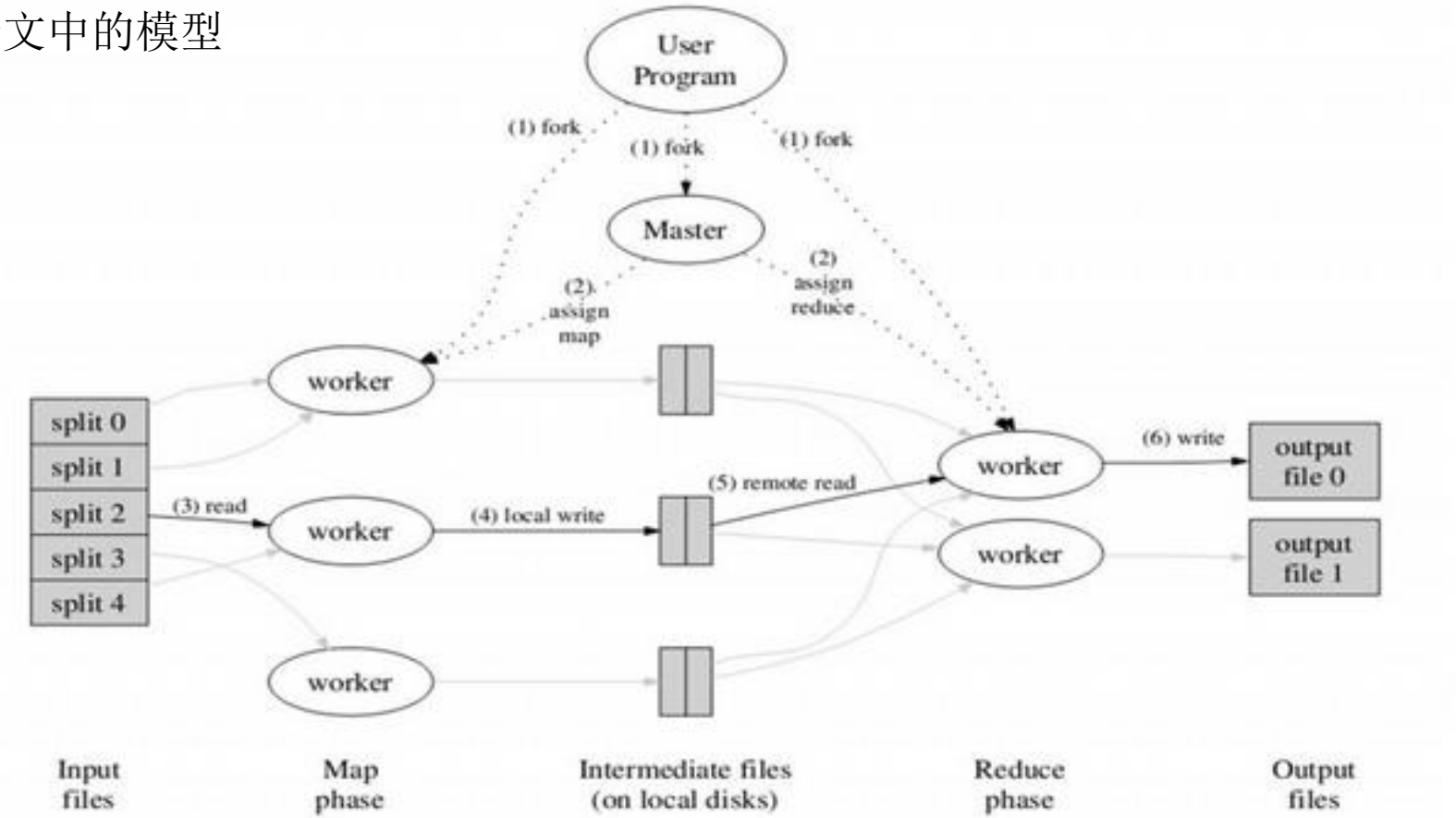
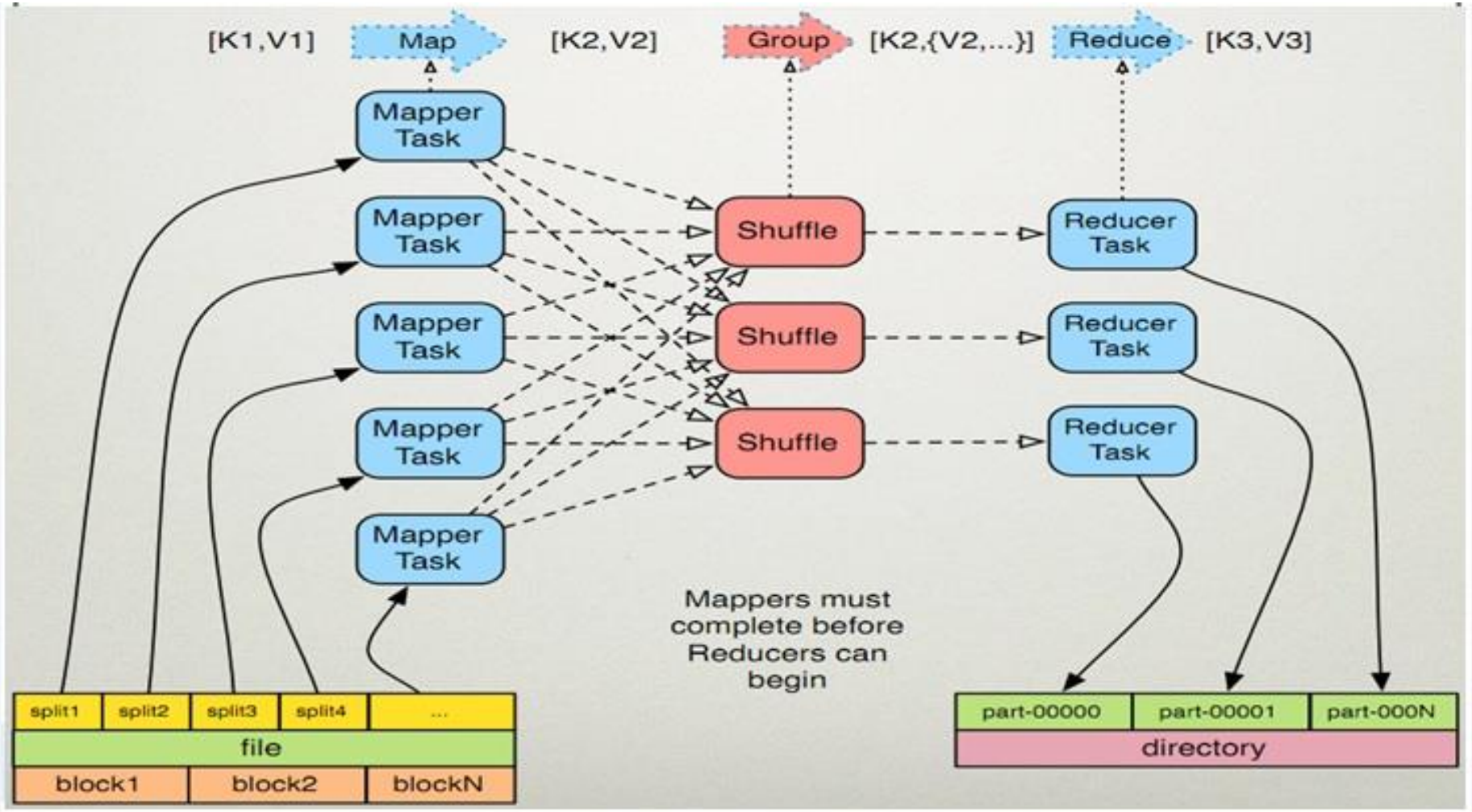


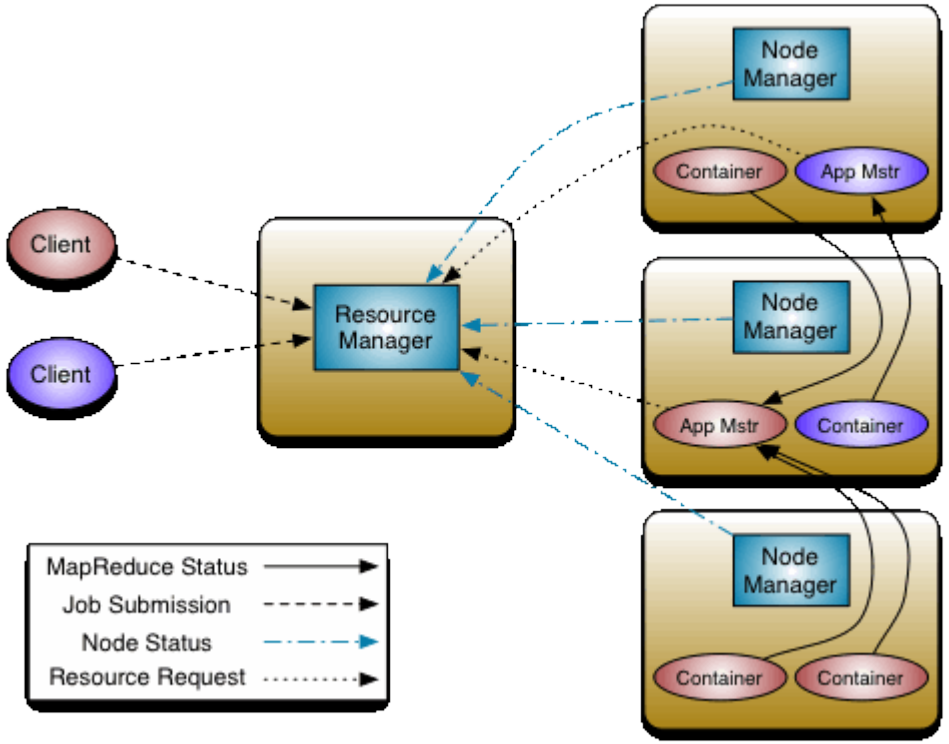
Figure 1: Execution overview

Map-Reduce

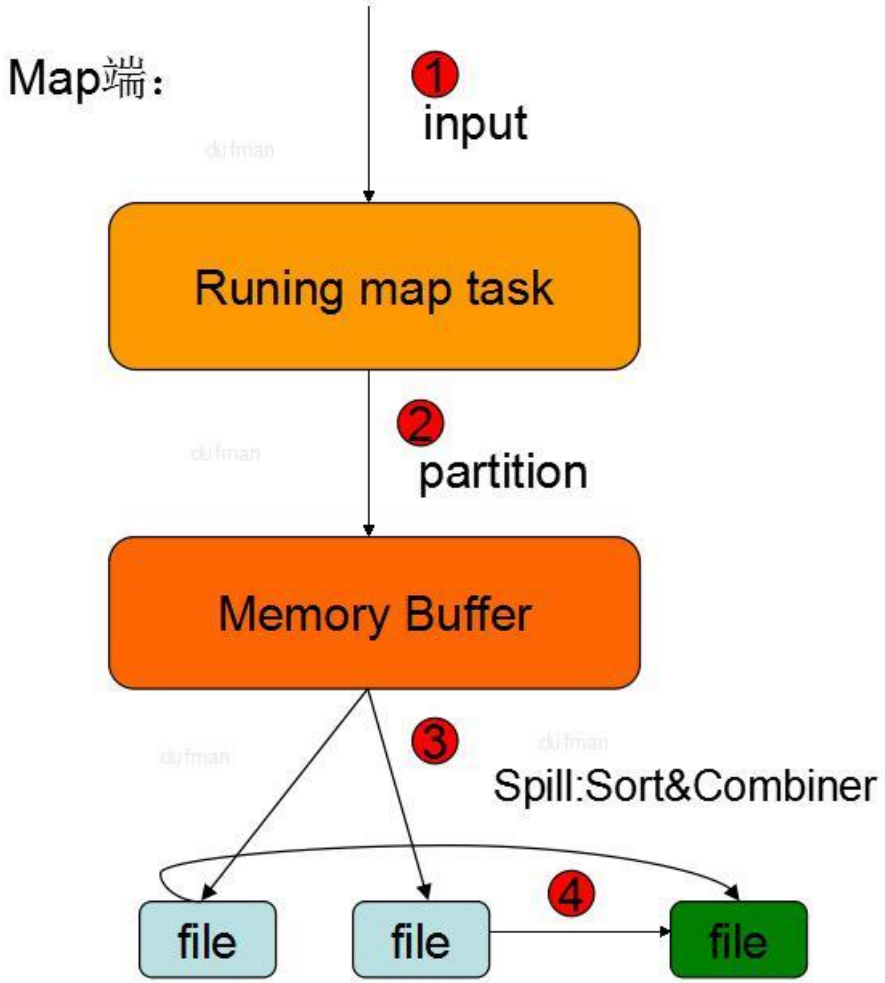

[K1,V1]
Map
[K2,V2]
Group
[K2,{V2,...}]
Reduce
[K3,V3]
Mapper Task
Mapper Task
Mapper Task
Mapper Task
Mapper Task
Shuffle
Shuffle
Shuffle
Reducer Task
Reducer Task
Reducer Task
Mappers must complete before Reducers can begin
split1
split2
split3
split4
...
file
block1
block2
blockN
part-00000
part-00001
part-000N
directory

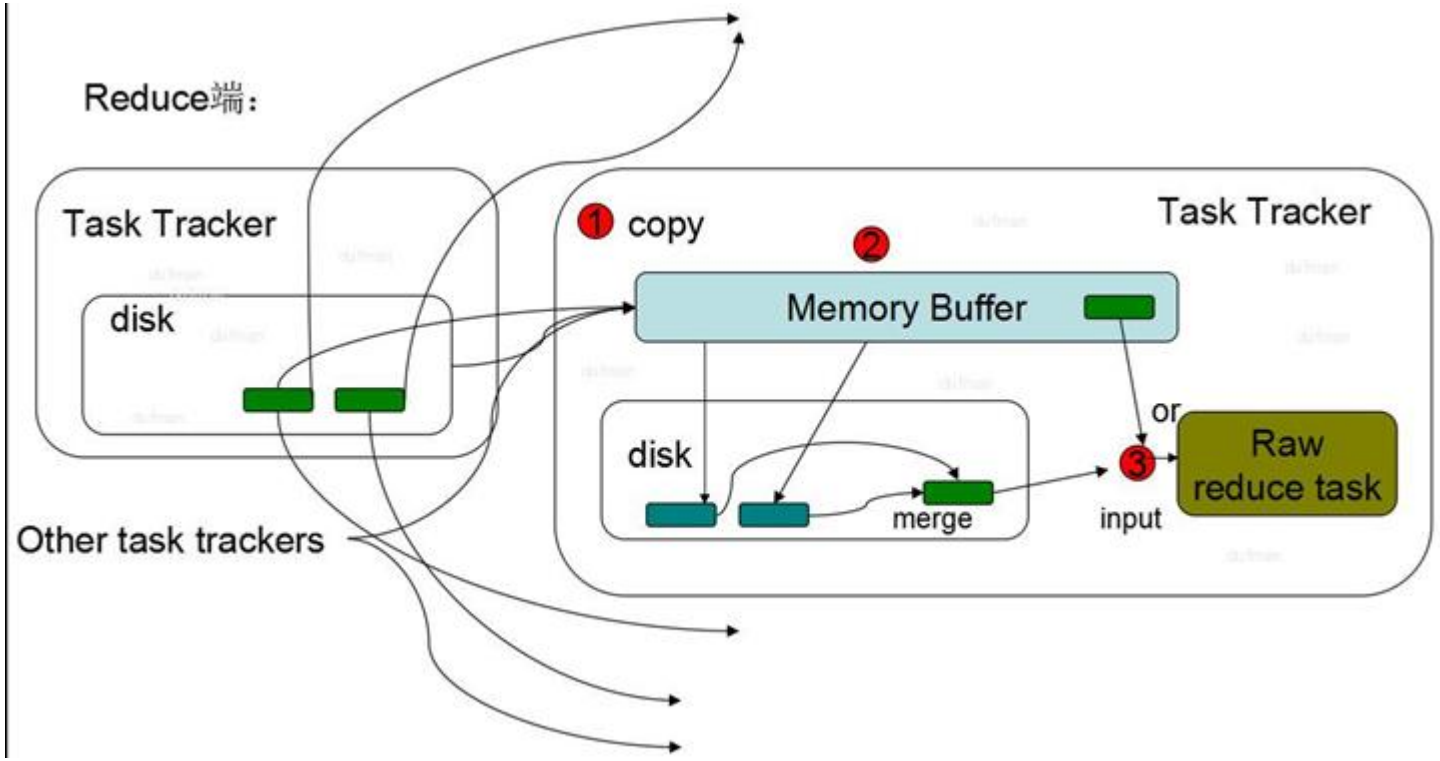
Map-Reduce

Hadoop Yarn框架



Map-Reduce


Map端：
1
input
Runing map task
2
partition
Memory Buffer
3
Spill:Sort&Combiner
4
file
file
file


Reduce端：
Task Tracker
disk
Other task trackers
1
copy
2
Memory Buffer
Task Tracker
disk
merge
or
3
input
Raw
reduce task

Map-Reduce 一个重大的倒退？

- 从大规模数据应用程序模型来说是一个巨大的倒退。
- 不是一个最优实现，因为它使用蛮力来代替索引。
- 一点都不新奇，它只是实现了一个特定的25年前就有的众所周知的技术。
- 失去了大部分目前数据库管理系统的特性。
- 不能兼容所有目前数据库管理系统用户已经依赖的工具。

MapReduce社区看起来感觉他们发现了一个全新的处理大数据集的模型。实际上，MapReduce所使用的技术至少是20年前的。将大数据集划分为小数据集的思想是在Kitsuregawa首次提出的“Application of Hash to Data Base Machine and Its Architecture”的基础上发展出来的一个新的连接算法。

- MapReduce需要将任务划分成map和reduce两个阶段，map阶段产生的中间结果要写回磁盘，而在这两个阶段之间需要进行shuffle操作。Shuffle操作需要从网络中的各个节点进行数据拷贝，使其往往成为最为耗时的步骤，这也是Hadoop MapReduce慢的根本原因之一，大量的时间耗费在网络磁盘IO中而不是用于计算。在一些特定的计算场景中，例如像逻辑回归这样的迭代式的计算，MapReduce的弊端会显得更加明显
- 在map阶段，当拥有相同键的数据拥有大幅度差异的时候。这个差异，反过来导致某些reduce实例花费比其它实例更长甚至常很多的时间来运行。结果就是计算的运行时间由速度最慢的那个reduce实例决定。

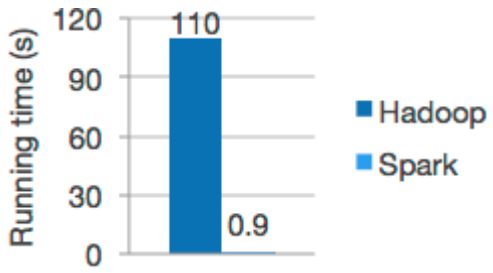
公司已经在几年前停止使用MapReduce 。 **谷歌认为在MapReduce上无法完成：很难迅速获取数据，不能进行批处理和流处理，而且经常需要部署和运行MapReduce集群**

Google停用MapReduce，高调发布Cloud Dataflow，**是这近十年分析经验的成果**。——2014年

Google已经停用自己研发的，部署在服务器上，用以分析数据的MapReduce，转而支持一个新的超大规模云分析系统Cloud Dataflow。

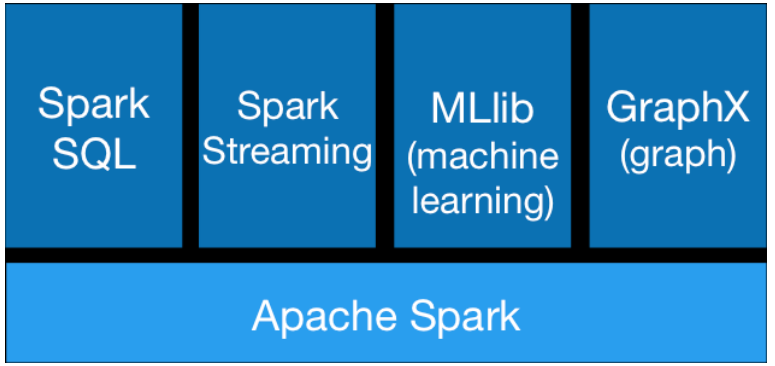
Spark

Spark具有先进的DAG执行引擎，支持cyclic data flow和内存计算. 因此，它的运行速度，在内存中是Hadoop MapReduce的100倍，在磁盘中是10倍. 这样的性能指标，真的让人心动啊. Spark的API更为简单，提供了80个High Level的操作，可以很好地支持并行应用. 它的API支持Scala、Java和Python，并且可以支持交互式的运行Scala与Python. 来看看Spark统计Word字数的程序：. 看看Hadoop的Word Count例子，简直弱爆了，爆表的节奏啊



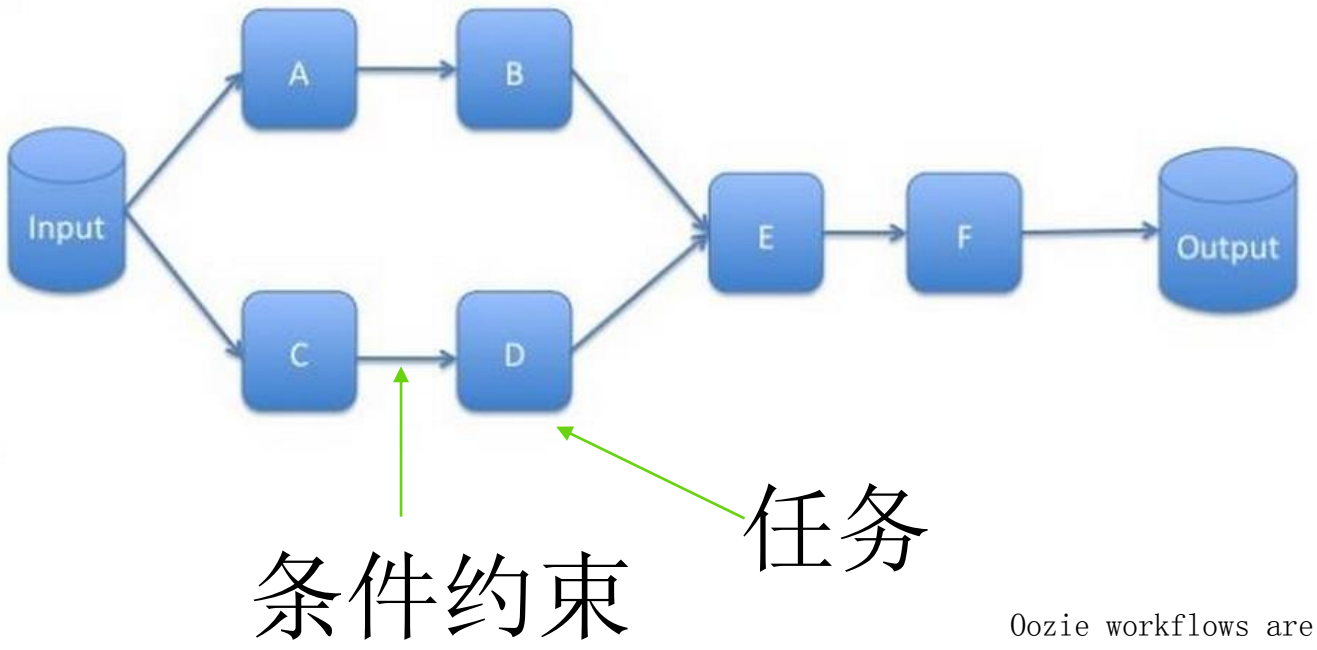

```
text_file = spark.textFile("hdfs://...")

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

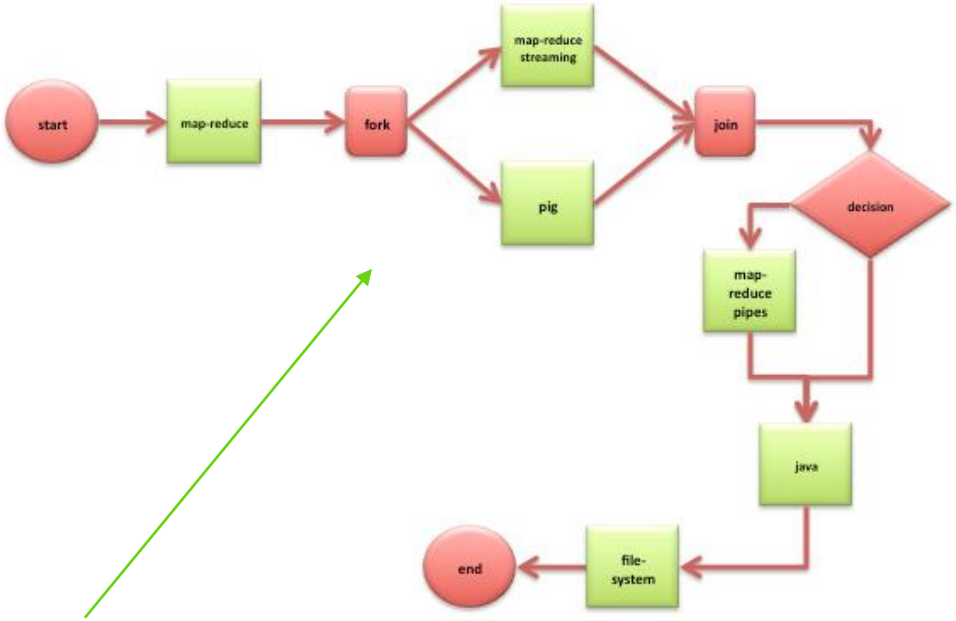
Word count in Spark's Python API



Spark DAG

在图论中，如果一个有向图无法从任意顶点出发经过若干条边回到该点，则这个图是一个有向无环图（DAG图）

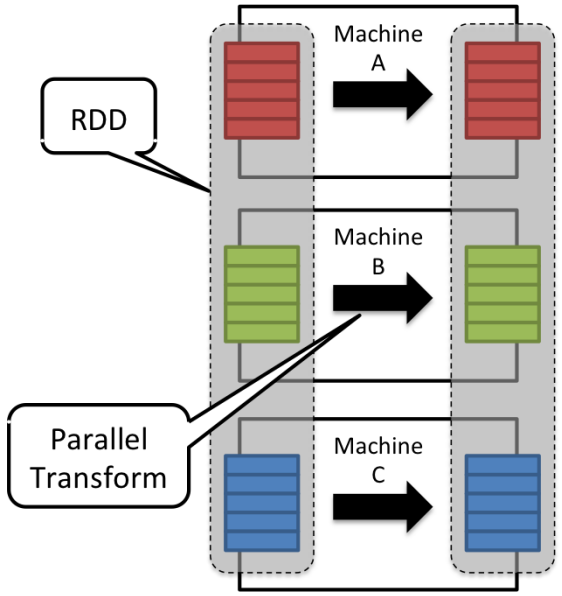


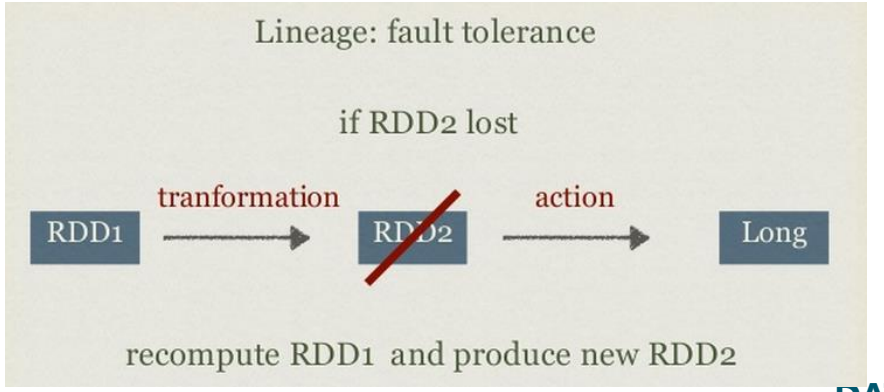


Oozie workflows are actions arranged in a control dependency DAG (Direct Acyclic Graph). An Oozie workflow may contain the following types of actions nodes: map-reduce, map-reduce streaming, map-reduce pipes, pig, file-system, sub-workflows, java, http (no yet implemented), email (not yet implemented) and ssh (deprecated).

Spark RDD

RRD（弹性分布式数据集)是一些对象的只读集合, 被划分到多台机器上, 并且在某个划分块丢失之后可以重建. 用户可以显式的把RDD缓存在内存中以加速迭代计算，RRD模型提供了一个抽象的数据架构，我们不必担心底层数据的分布式特性，而应用逻辑可以表达为一系列转换处理。





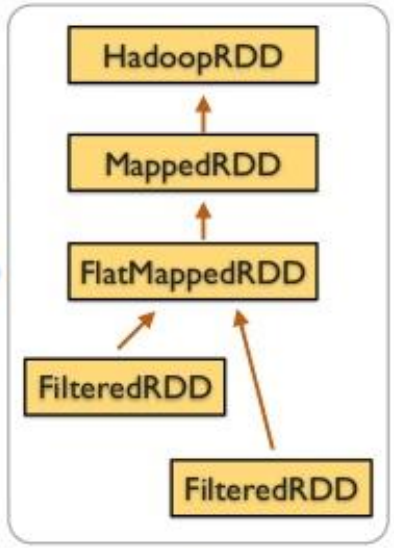
RDD Lineage

RDD Transformations

```
words = sc.textFile("hdfs://large/file/")
            .map(_.toLowerCase)
            .flatMap(_.split(" "))
alpha = words.filter(_.matches("[a-z]+"))
nums  = words.filter(_.matches("[0-9]+"))
```

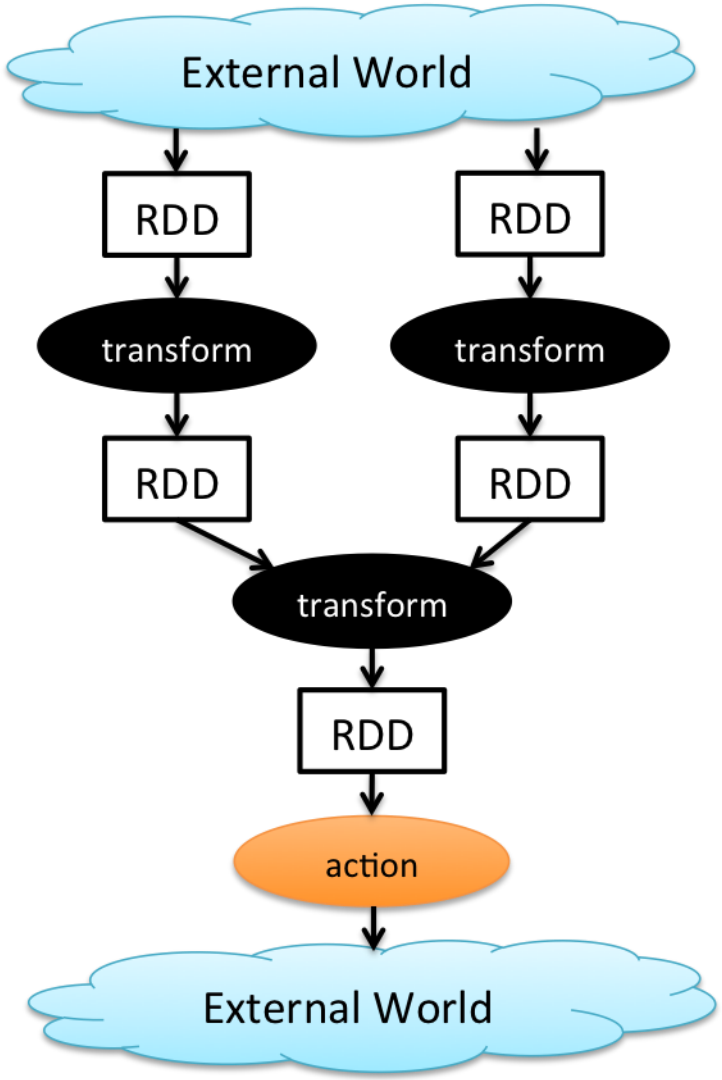
```
alpha.count()
```

Action (run job on the cluster)

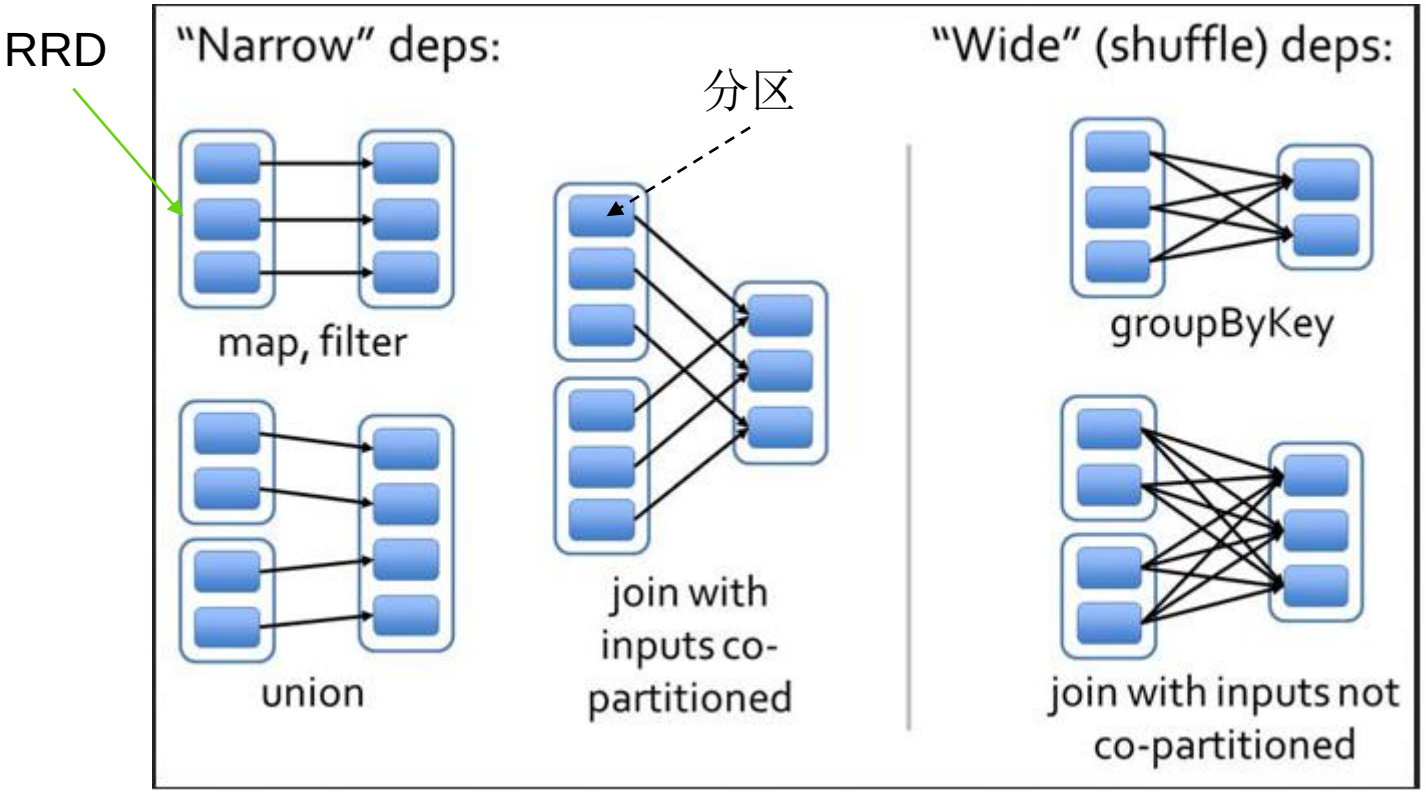


Lineage
(built on the driver by the transformations)

Spark RDD

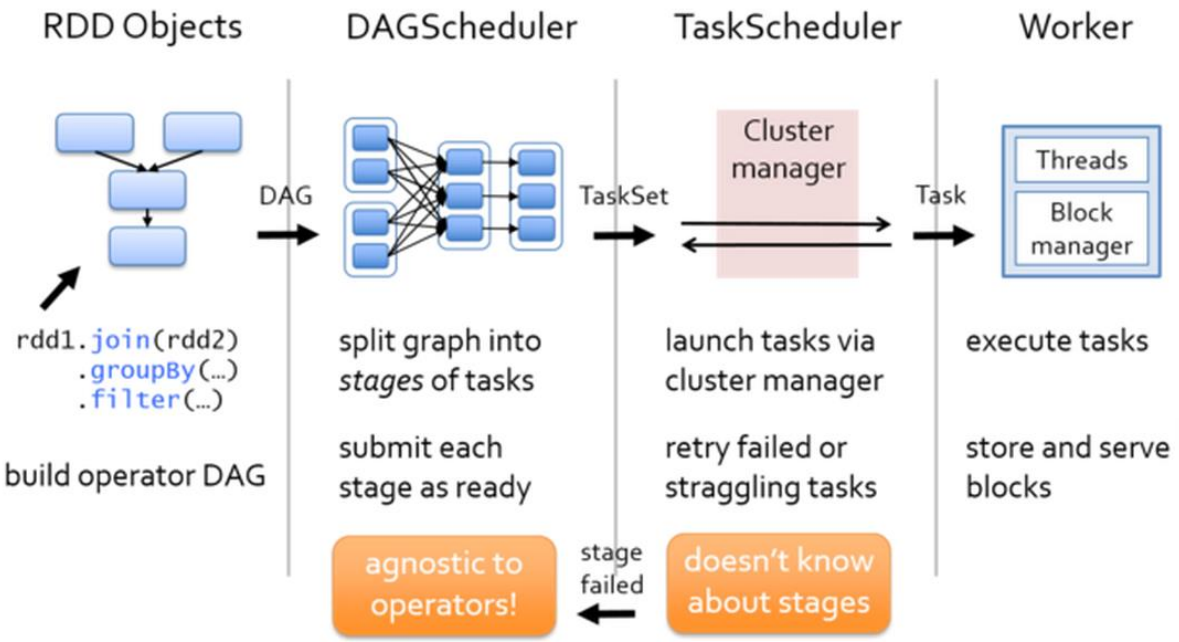


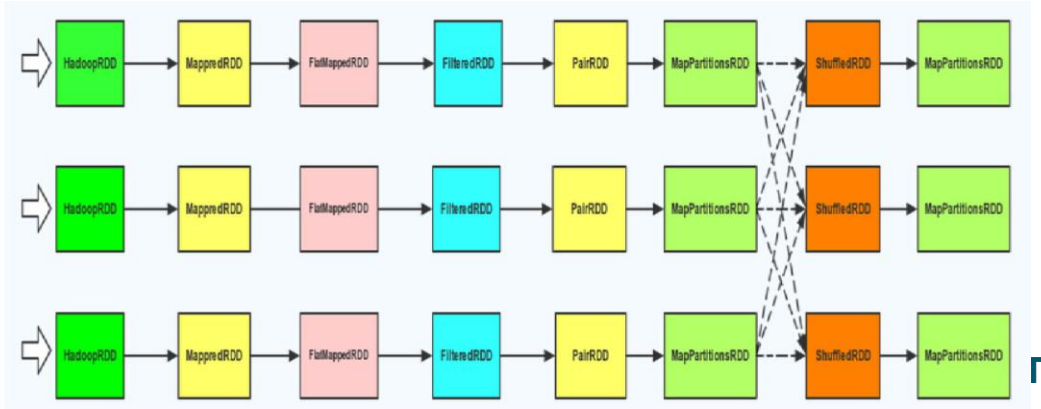
RDD的依赖关系：

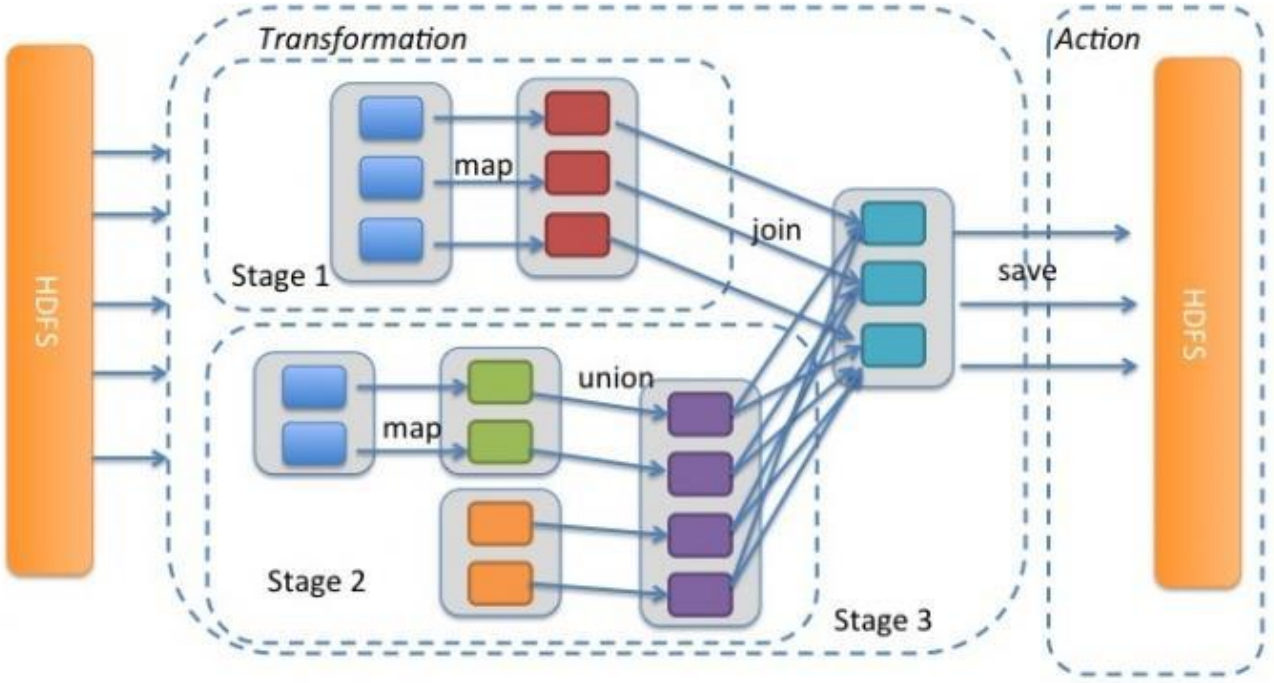


实时计算框架

Spark 运行框架



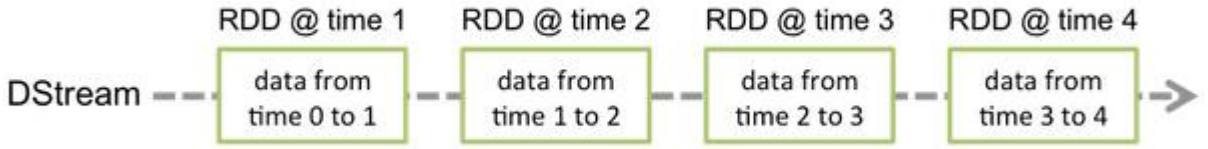




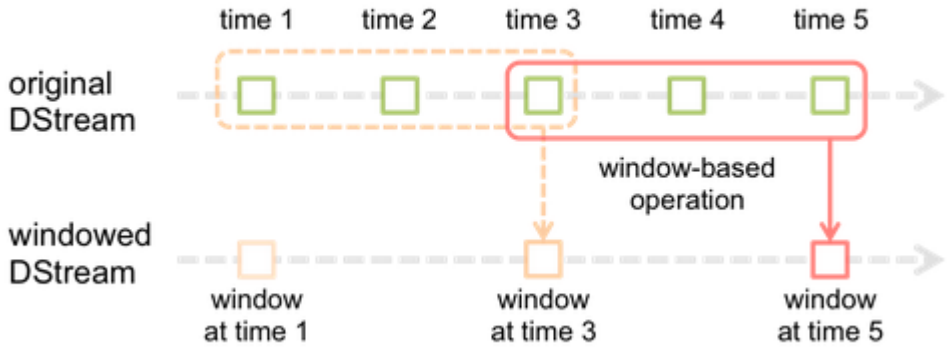
Spark Streaming

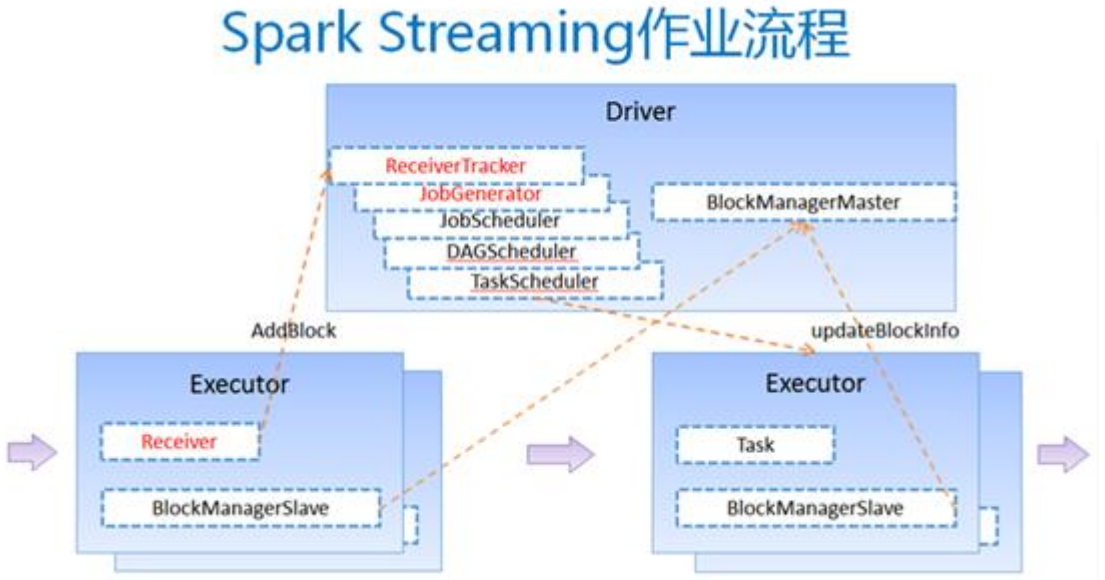
```
TwitterUtils.createStream(...)
    .filter(_.getText.contains("Spark"))
    .countByWindow(Seconds(5))
```

Counting tweets on a sliding window

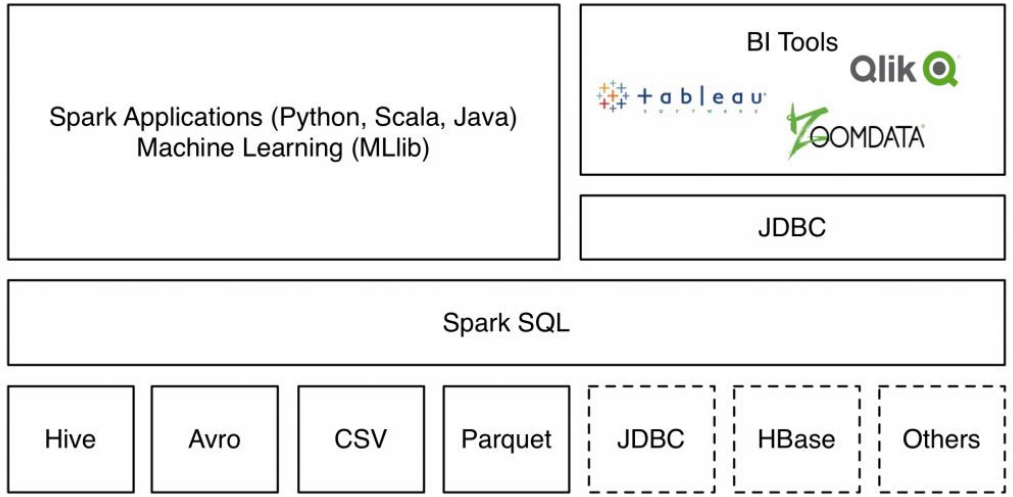


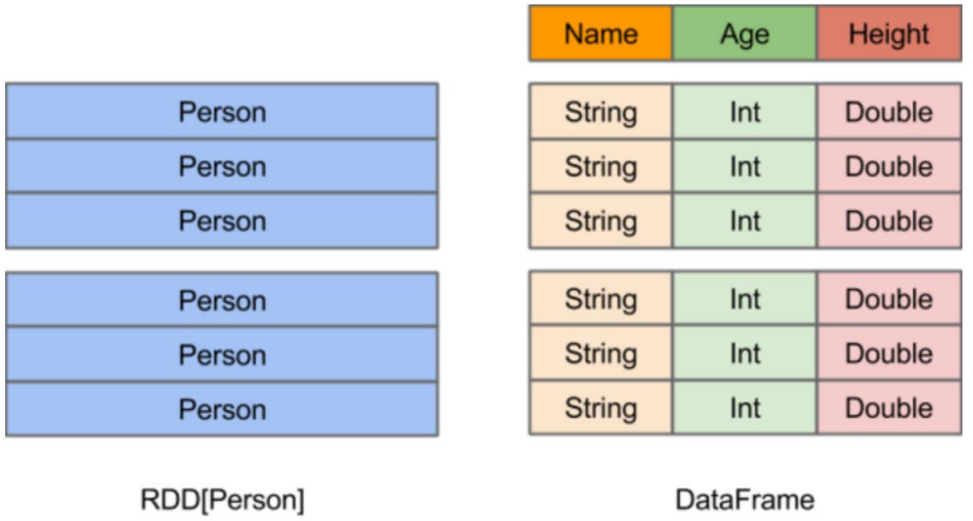

```
// Reduce last 30 seconds of data, every 10 seconds
val windowedWordCounts = pairs.reduceByKeyAndWindow(_ + _, Seconds(30), Seconds(10))
```

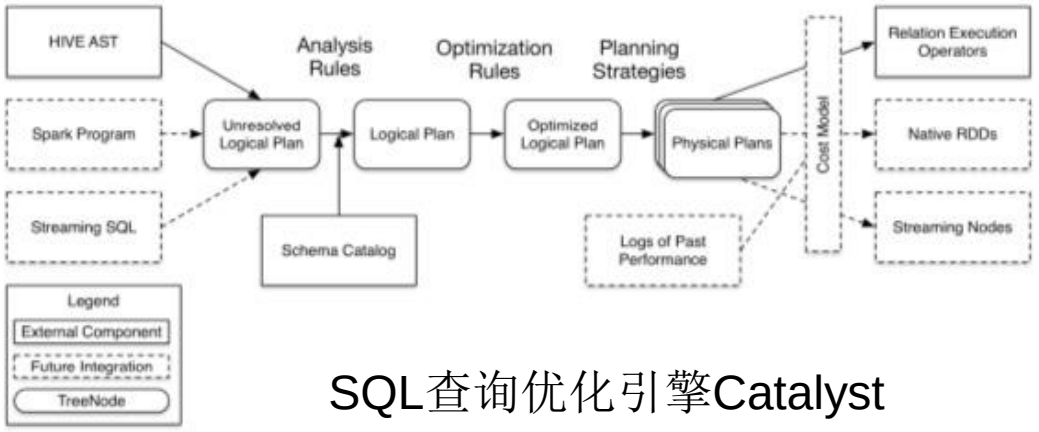




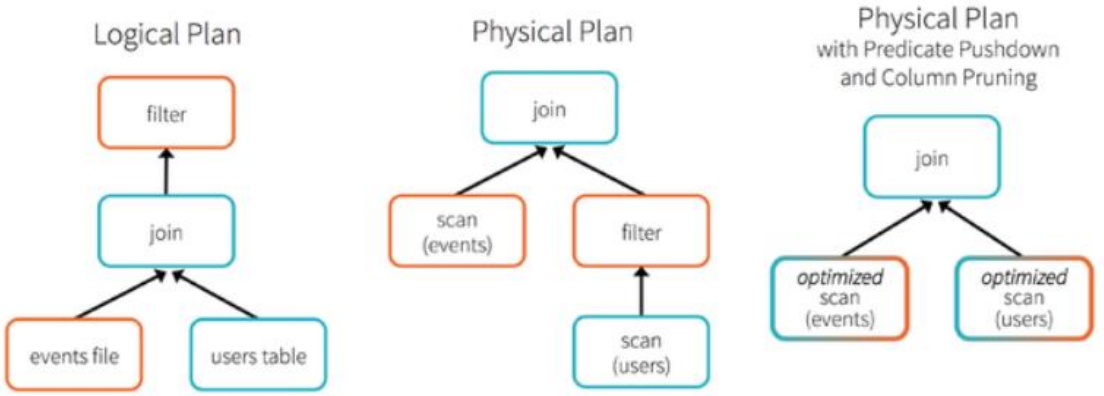
Spark SQL





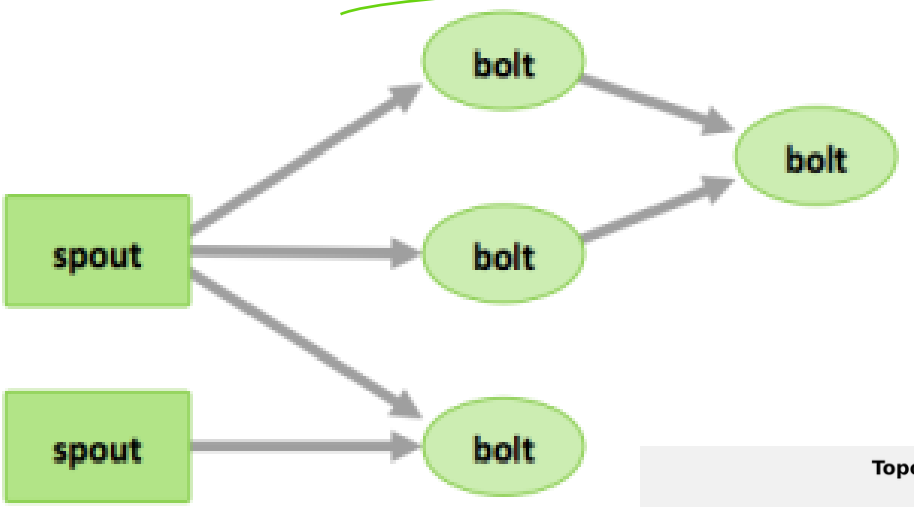


SQL查询优化引擎Catalyst

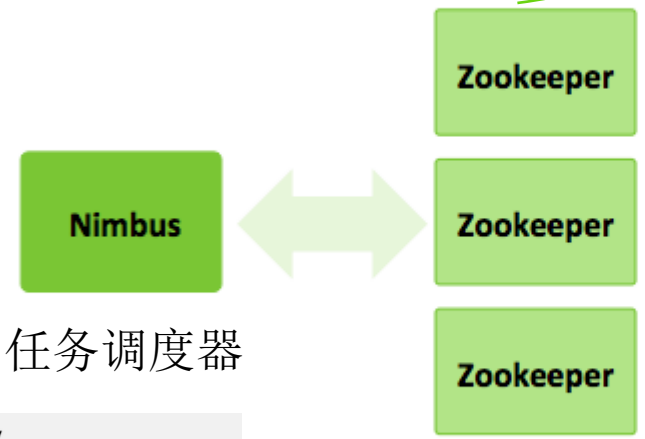


SQL执行计划

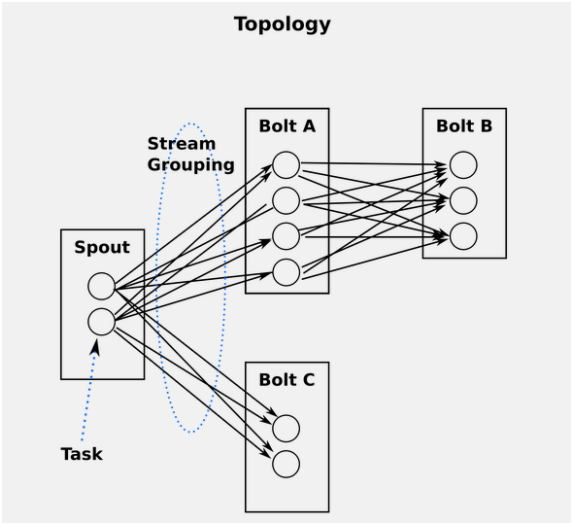
Apache Storm



Storm **Topologies**



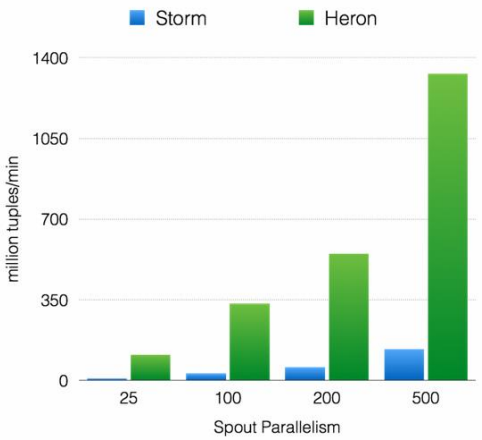
任务调度器

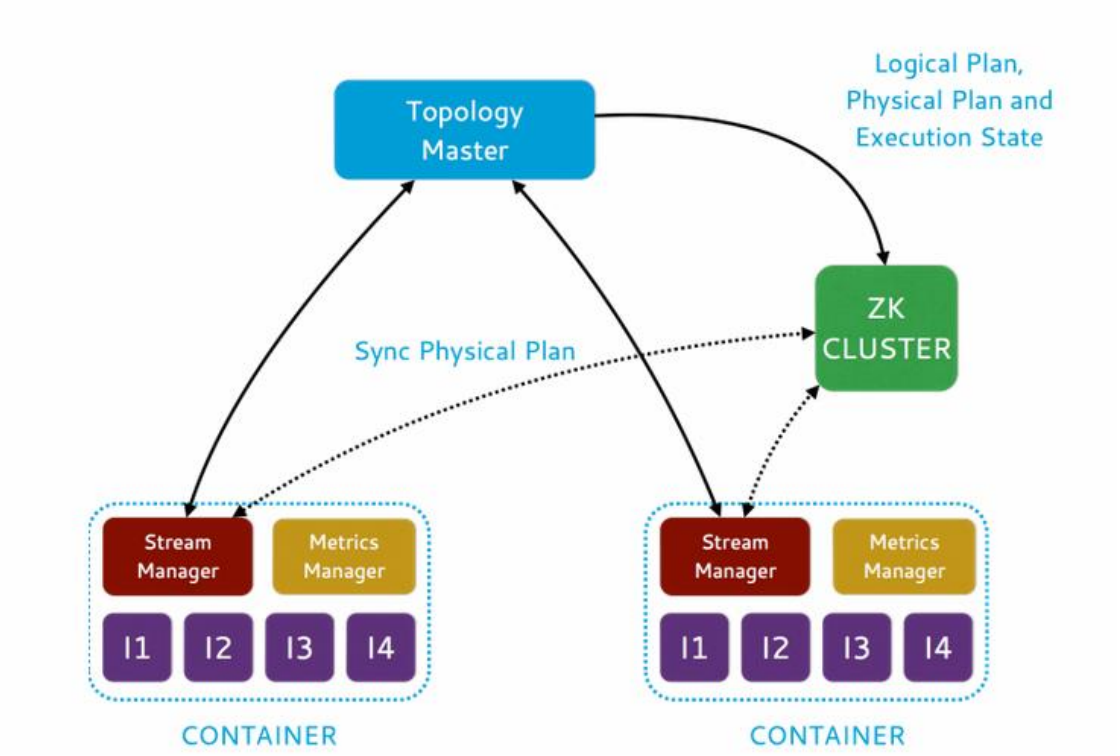


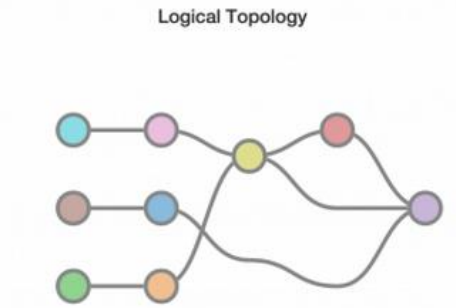
Worker集群

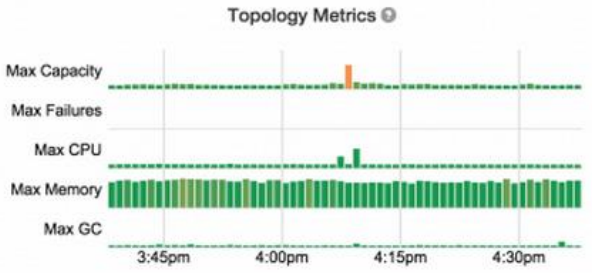
实时计算框架 Twiter Heron

Twitter Has Replaced Storm with Heron





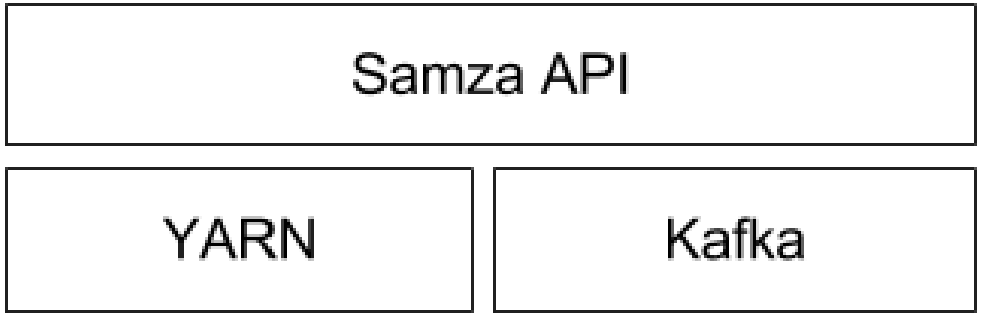


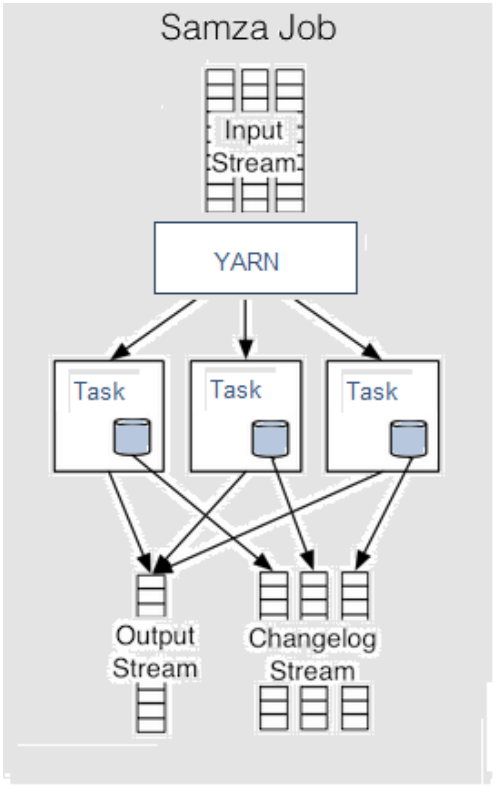


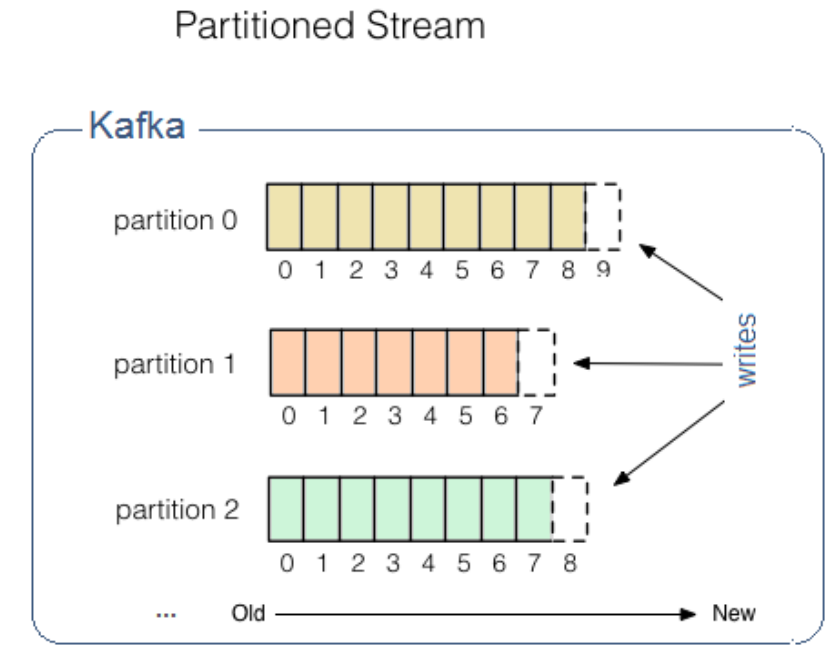
We would also like to thank the Storm community for teaching us numerous lessons and for moving the state of distributed real-time processing systems forward.

实时计算框架 LinkedIn Samza Kafka

对于实时流，Samza采用了不同的做法。首先，它希望流能够分区。它希望这些流是有顺序的。如果先读了消息3，又读了消息4，那么就无法在一个单独的分区里颠倒它们的顺序。它还希望流能够回放，就是说以后可以回头重读一条消息。它希望流具备容错能力。如果分区1里面的一台主机不复存在，那么流在其它主机上应该仍然可读。

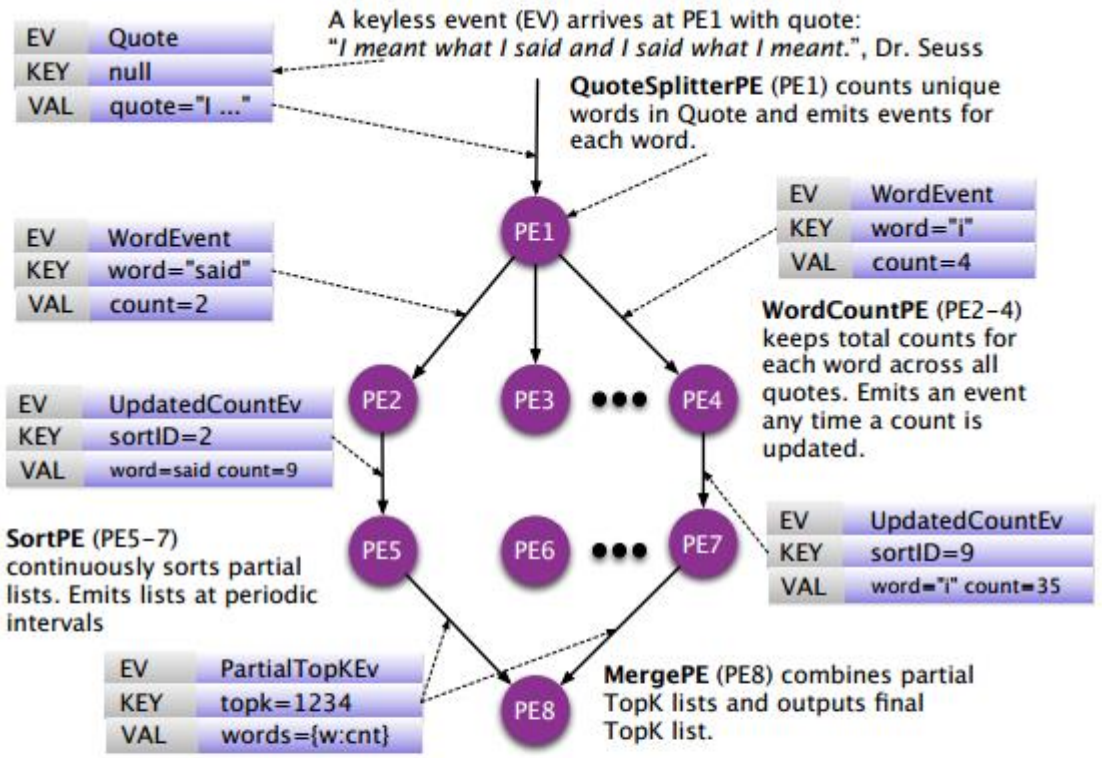






samza

实时计算框架 Apache s4



PE ID	PE Name	Key Tuple
PE1	QuoteSplitterPE	null
PE2	WordCountPE	word="said"
PE4	WordCountPE	word="i"
PE5	SortPE	sortID=2
PE7	SortPE	sortID=9
PE8	MergePE	topK=1234

Figure 1. Word Count Example

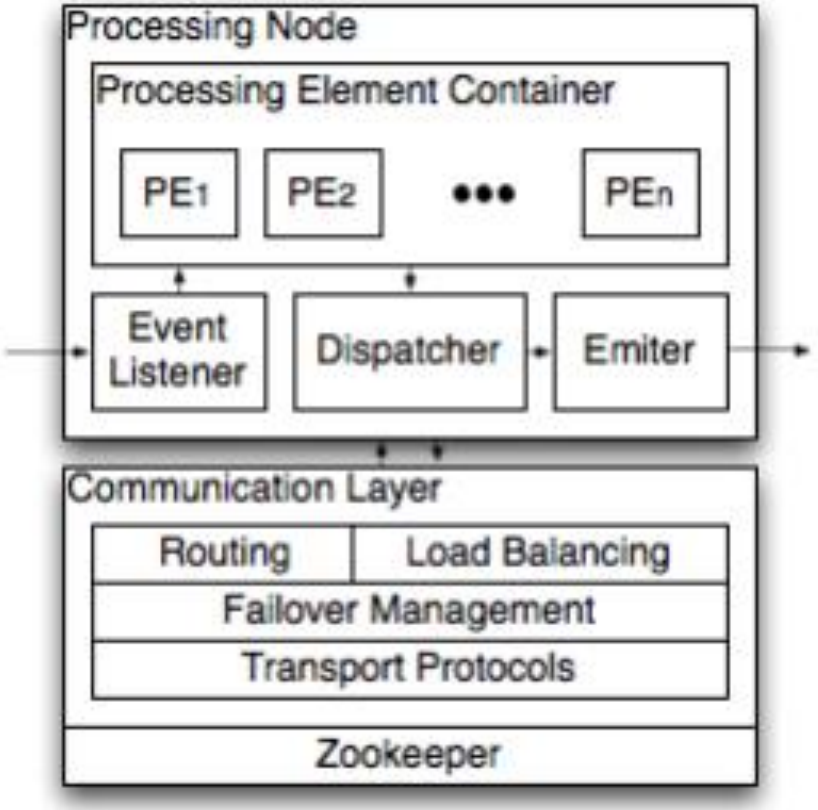
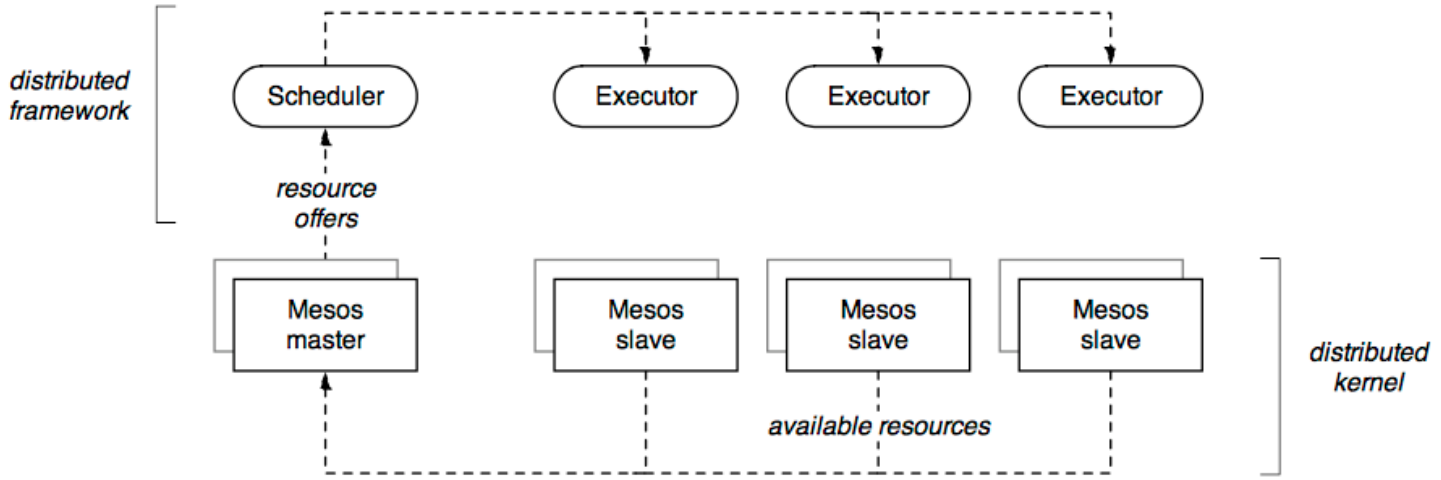


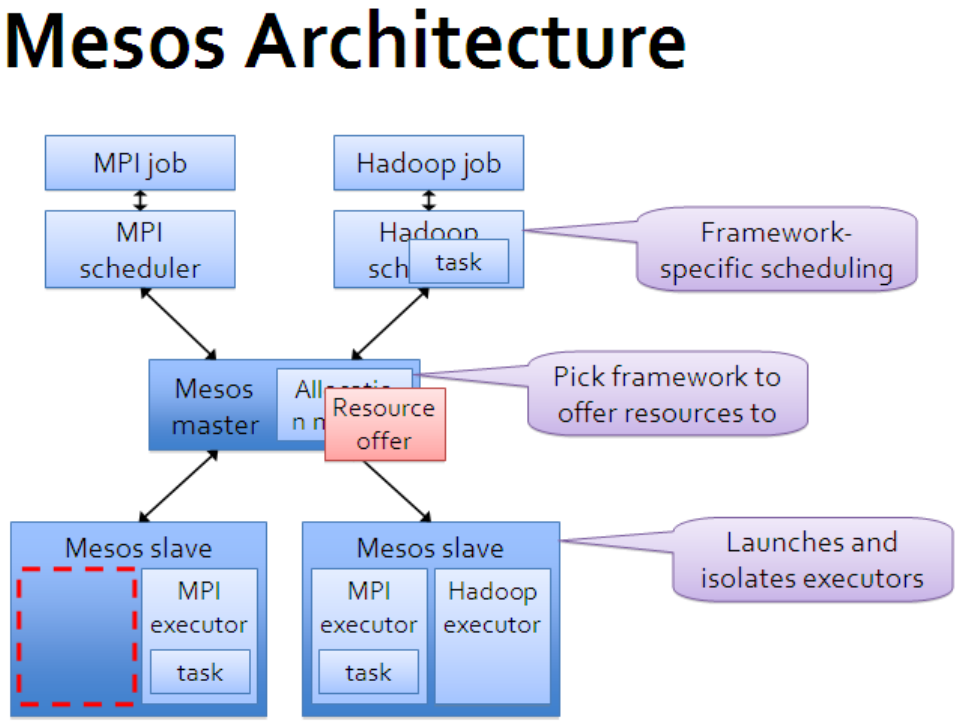
Figure 2. Processing Node

分布式计算调度平台 Apache Mesos

What is Mesos?
A distributed systems kernel
Mesos is built using the same principles as the Linux kernel, only at a different level of abstraction. The Mesos kernel runs on every machine and provides applications (e.g., Hadoop, Spark, Kafka, Elastic Search) with API's for resource management and scheduling across entire datacenter and cloud environments.

Apache Mesos 将CPU、内存、存储介质以及其它计算机资源从物理机或者虚拟机中抽象出来，构建支持容错和弹性的分布式系统，并提供高效的运行能力。





Thanks

FAQ时间