Spark布道者陈超：Spark Ecosystem & Internals

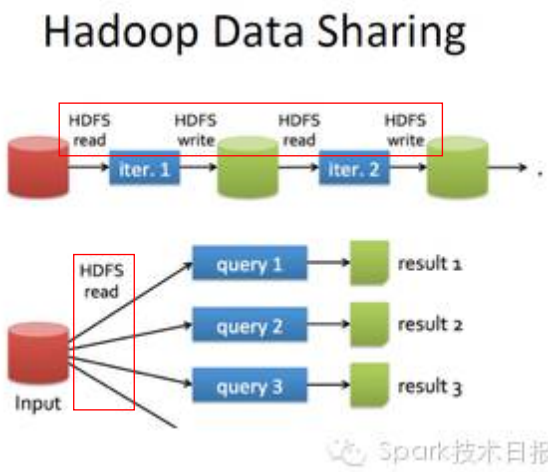
2015-02-03 Spark技术日报 Spark生态系统和内部组件

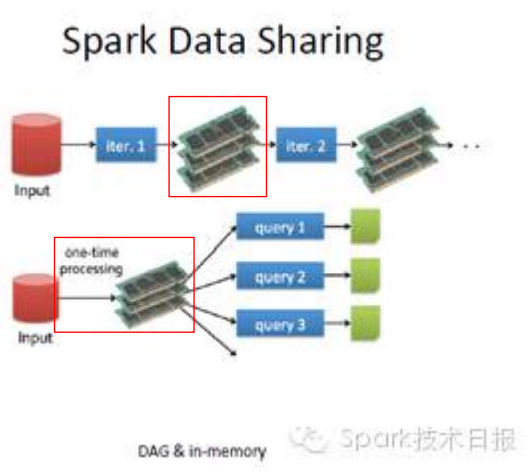
在分享中，陈超(@CrazyJvm，ChinaScala)首先简短的介绍了Spark社区在2014年的发展：目前Spark的发布版本是1.2，整个2014年Spark共发布了3个主要版本——1.0、1.1、1.2。随后，陈超对Spark生态圈进行了详细的分析：

Spark：What & Why？

Spark是一个非常快，并且普适性非常强的一个大数据处理引擎。谈到Spark，首先就是一些常见特性：速度快、易用、通用和兼容Hadoop。首先通用，Spark可以支撑批处理、流计算、图计算、机器学习等众多应用场景；其次，与Hadoop良好的兼容。鉴于大多数的企业仍选用HDFS来存数据，Spark的设计与HDFS有着非常好的兼容性——假如数据存储在HDFS，那么不做任何数据迁移工作就可以直接使用Spark。

Spark vs. Hadoop 数据共享方式





为什么

对于为什么要选择Spark，如上图所示，陈超从迭代计算和HDFS同批数据的多维度查询两个方面将之与Hadoop进行了对比：

问题：

迭代计算。在这个场景下，Hadoop需要多次读写HDFS（磁盘），造成了大量的IO和序列化、反序列化等额外开销。此外，每次写HDFS都需要写入3份，因此造成了备份方面的开销。

HDFS同批数据的多维度查询。对HDFS同一批数据做成百或上千维度查询

时，Hadoop每次做一个独立的query，也就是每次都要从磁盘读取这个数据。因为每次都从磁盘中读取同一组数据，效率显然可以继续提高。

理由：

而在这两种场景中，Spark可以使用内存缓存中间/常用数据，从而在避免磁盘IO开销的同时，还将大幅度提高性能。

Why Spark is so Fast?

Spark一直以快速著称，那么除下之前所说的内存，又是什么特性让Spark可以如此之快？在这里，陈超提到了DAG（有向无环图，下文详细介绍）、Thread Model（线程模型）和Optimization（比如延迟调度）3个方面。

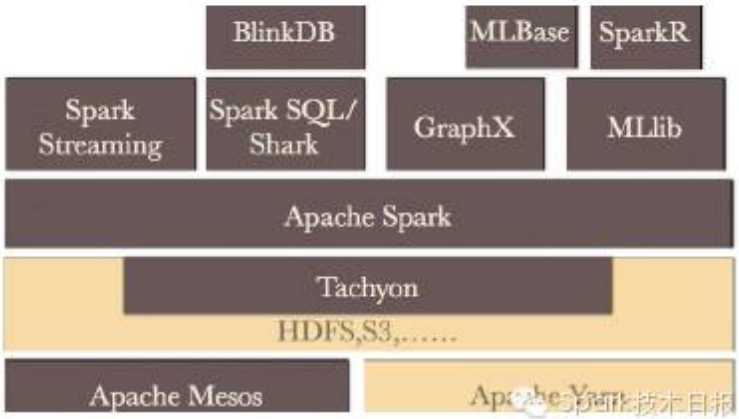
Thread Model。Hadoop基于进程模型，每次启动一个task都需要新启动一个子JVM进行计算，可能也会存在JVM Reuse，这里即使避开JVM Reuse中存在的问题不谈，每次JVM启动时已经造成了不菲的开销。而Spark在应用程序启动时就启动了线程池，所以任务的启动开销非常小。

Optimization——延迟调度。当任务下达到某台主机时，恰好该主机的计算资源（CPU、内存等）已被耗尽，这个时候，Spark会采用延迟调度的机制，让其等待一小会，而不是将该台主机上需要计算的数据通过网络传输到另外的主机上。使用这个机制，在计算数据体积非常大时，有着很大的优势。 也就是所谓的“让计算跟着数据走，而不是数据跟着计算走”。

Spark解析

伯克利数据分析协议栈

BDAS



其中包括：资源管理框架，Apache YARN、Apache Mesos；基于内存的分布式文件系统，Tachyon；随后是Spark，更上面则是实现各种功能的系统，比如机器学习MLlib库，图计算GraphX，流计算Spark Streaming。再上面比如：SparkR，分析师的最爱；BlinkDB，我们可以强迫它几秒钟内给我们查询结果。

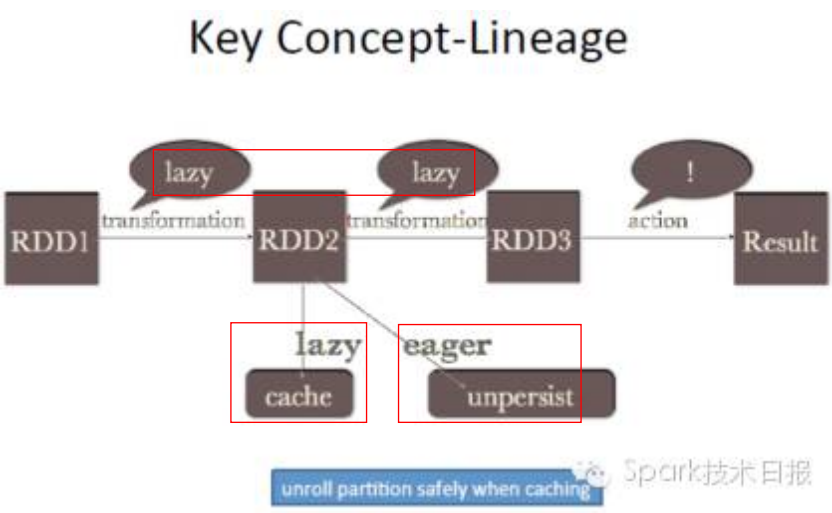
正是这个生态圈，让Spark可以实现“one stack to rule them all”，它既可以完成批处理也可以从事流计算，从而避免了去实现两份逻辑代码。而整个Spark的理论基础就是RDD：

RDD的核心理念

RDD可以想象为一个个的partitions，退一步也可理解为一个非常大的List(1,2,....9)，使用3个partition分别保存这个List的3个元素，而每个partition（或者split）都会有一个函数去计算。同时，RDD之间是可以相互依赖的。然后，可以为Key-value RDD指定partitioner, RDD中的每个split也都有各自的preferred location。

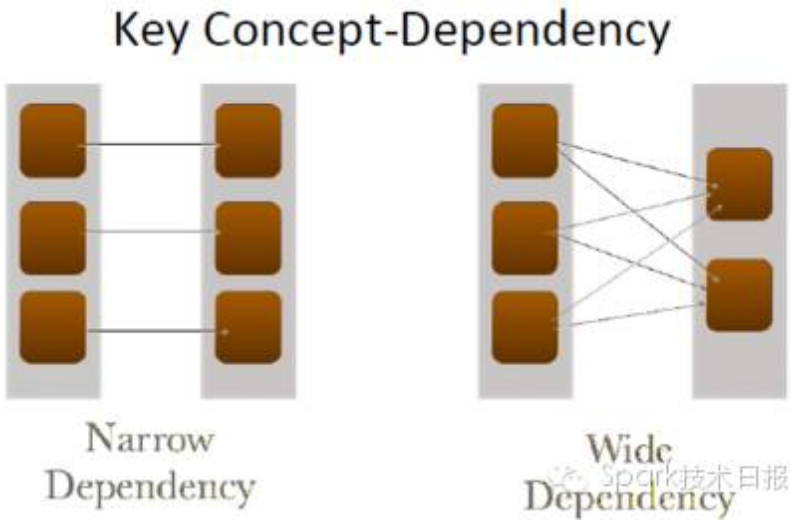
最后一个preferred locations，这个理念存在于当下的众多分布式系统中，也就是计算跟着数据走。通常情况下，转移计算的时间远远小于转移数据的时间。对于Hadoop来说，因为数据在磁盘中，磁盘本地性通常达到了顶峰，而对于Spark来讲，因为数据（可以）保存在内存中，所以内存本地性才具备最高优先级。

运行原理



上图表述了Spark运行原理：rdd1、rdd2、rdd3等等一直转换到另外一个RDD。需要注意的是，这里面存在的是一个延迟的执行，也就是转换不会立刻执行。Spark只会在元数据中记录这个过程，但是不会真正的执行，这个要注意一点，只有在碰到action的时候才会真正的去执行。这个时候需要注意的是，比如上图RDD2所做的cache，这个操作同样是lazy的，同样在碰到action的时候才会执行。就在这里，坑出现了，即使persist与cache使用的是相同的接口，但是unpersist却是eager的。从1.1版本开始，cache其实已经有了更安全的做法，但是涉及过多内核细节，这里就不做多的解释。

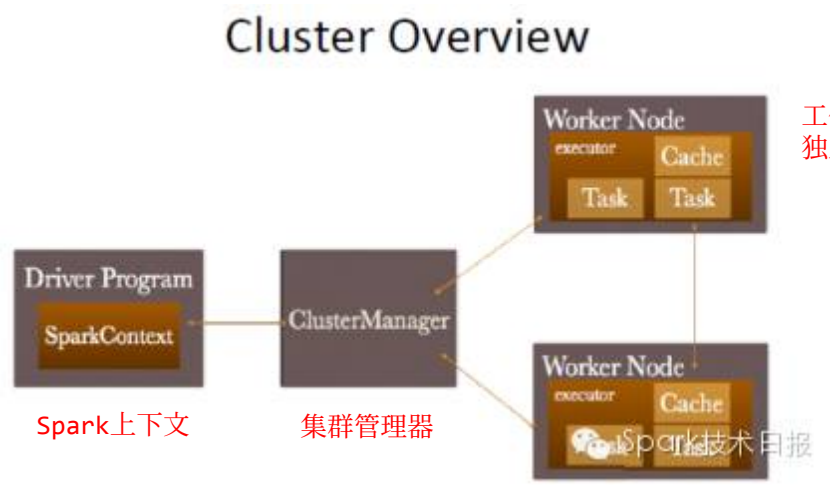
RDD的依赖性



narrow dependency和wide dependency是Spark中另外两个重要的概念。对比后者，narrow dependency无论是在从容错上，还是在执行效率上都占有优势。

ClusterManager：目前来讲，在国内采用率更大的显然是YARN。

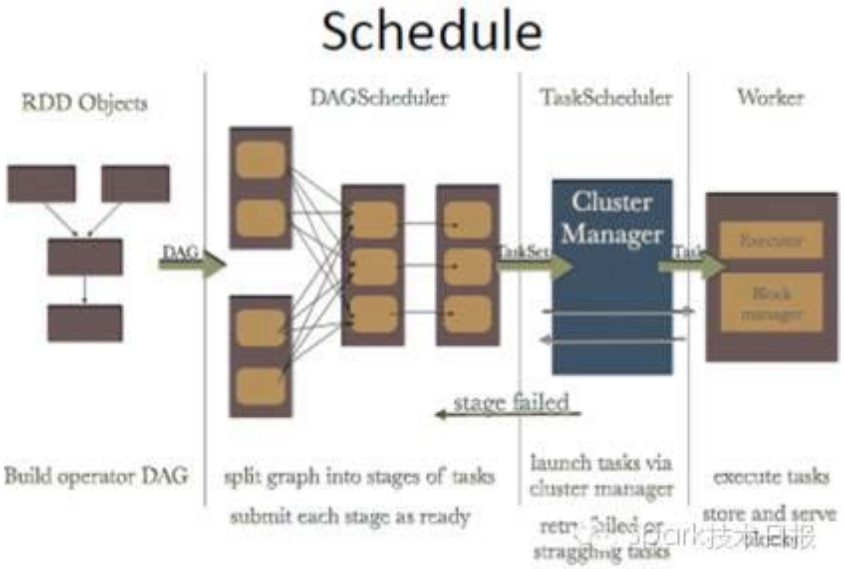
Cluster overview



SparkContext，写代码时生成，并向ClusterManager请求资源。ClusterManager会负责连接到Worker Node取得资源，其中executor才是task的真正执行者。这里有三个需要注意的点：第一，ClusterManager是可插拔的，可以任意选择；第二点，因为driver program需要发送任务给Worker Node，因此提交任务的地方不要离Worker Node特别远。第三点比较重要的一点，每个应用程序在每个Worker Node上都会有独立的executor，并且不同应用程序的executor(间)是不可以共享数据的。

PS：YARN通过Container来封装资源，因此在YARN中Worker对应的是Container。

调度



最初，Spark程序会隐式地建立一个逻辑上有向无环图（DAG），随后DAGScheduler

会将DAG切分成一个个stage，随后这些stage会被传送给TaskScheduler，之后再传送给Worker上的executor执行。其中executor会以多线程的模式执行。

Shuffle

从理论上讲，Spark Shuffle从未超过MapReduce，直到改完以后才OK。当下，Shuffle使用的是基于PULL的模式，中间文件会写到磁盘，同时，在每个partition都会建立hash map。需要注意的是，在可以跨keys spill的同时，主机内存必须可以装进单key-value。

在监控上，之前的版本中，只有当一个任务结束时，才可以收集这个任务的运行数据，这点在当下的版本已被改进。

生态系统简析

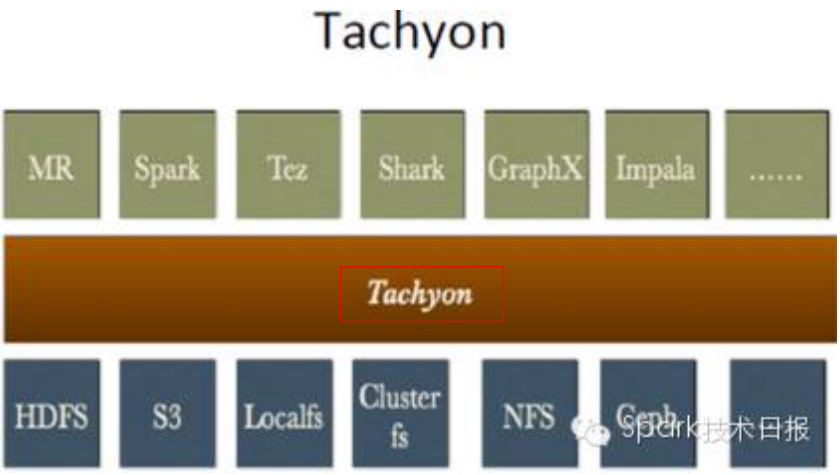
流处理

Spark Streaming：Spark Streaming实质上仍然是批处理，但是把之前大的批处理拆为小的batch。同时，当下Spark Streaming已支持限流，当流量很大时，Spark可以挡住。此外，它还可以支持实时机器学习。在Spark Streaming中，数据丢失一般因为两种情况——worker failure和driver failure。在之前版本中，可能会存在小部分的数据丢失，而在1.2版本发布后，reliable receiver模式保证了所有数据不会丢失，这点在Kafka的连接上非常适用。

MLlib：当下的算法已经非常丰富，包括分类、聚类、回归、协同过滤、降维等等。ML Pipeline可以大幅度的减少开发时间，它可以帮开发者打通数据收集、数据清理、特征提取、模型训练、测试、评估、上线整个流程。

Graphx：在这里，Spark的优势是既能处理表视图，也能处理图视图。

Spark SQL：Spark生态圈中最火的组件，目的很简单，用来支持SQL标准。对比Spark SQL，因为基于MapReduce的进程模型，Hive中存在许多一直未修复的多线程bug。值得一提的是，Spark SQL的贡献者中，一半以上是华人。



Tachyon可以支撑几乎所有框架

Tachyon：内存分布式系统，让不同的Job或者框架分享数据，从而绕过HDFS，以更快地速度执行。同时，它还可以避免任务失败时的数据重算。最后，Tachyon可以让系统避免多次GC。

SparkR：让R语言调用Spark。原理是Spark Context通过JNI调用Java Spark Context，随后通过Worker上的Executor调用R的shell来执行。现在存在的问题是，每次task执行时都需要启动R shell，所以还亟待优化。

BlinkDB，一个任性的数据库

BlinkDB：很任性的一个数据库，允许操作者带着time bounds或者error bounds去查。原理是在原始数据上维护一组多维样本，当然其中还需要一个动态的样本选择策略。

JobServer：提供了一个RESTful接口来提交和管理Apache Spark job、jars及job contexts，即Spark as a Service。