Spark 及其生态圈简介

目 录

Spark 及其生态圈简介

1 简介

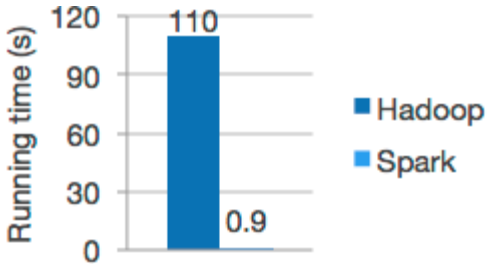
1.1 Spark 简介

Spark 是加州大学伯克利分校 AMP 实验室（Algorithms, Machines, and People Lab）开发通用内存并行计算框架。Spark 在 2013 年 6 月进入 Apache 成为孵化项目，8 个月后成为 Apache 顶级项目，速度之快足见过人之处，Spark 以其先进的设计理念，迅速成为社区的热门项目，围绕着 Spark 推出了 Spark SQL、Spark Streaming、MLLib 和 GraphX 等组件，也就是 BDAS（伯克利数据分析栈），这些组件逐渐形成大数据处理一站式解决平台。从各方面报道来看 Spark 抱负并非池鱼，而是希望替代 Hadoop 在大数据中的地位，成为大数据处理的主流标准，不过 Spark 还没有太多大项目的检验，离这个目标还有很大路要走。

Spark 使用 Scala 语言进行实现，它是一种面向对象、函数式编程语言，能够像操作本地集合对象一样轻松地操作分布式数据集（Scala 提供一个称为 Actor 的并行模型，其中 Actor 通过它的收件箱来发送和接收非同步信息而不是共享数据，该方式被称为：Shared Nothing 模型）。在 Spark 官网上介绍，它具有运行速度快、易用性好、通用性强和随处运行等特点。

- 运行速度快

  Spark 拥有 DAG 执行引擎，支持在内存中对数据进行迭代计算。官方提供的数据表明，如果数据由磁盘读取，速度是 Hadoop MapReduce 的 10 倍以上，如果数据从内存中读取，速度可以高达 100 多倍。



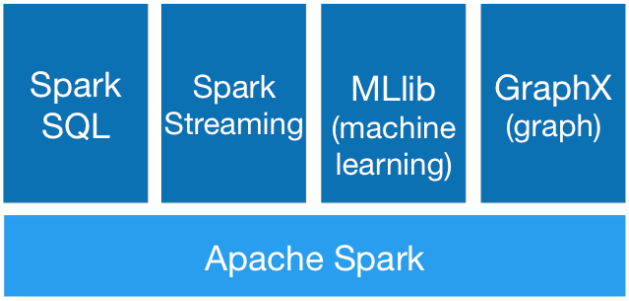
- 易用性好

  Spark 不仅支持 Scala 编写应用程序，而且支持 Java 和 Python 等语言进行编写，特别是 Scala 是一种高效、可拓展的语言，能够用简洁的代码处理较为复杂的处理工作。

- 通用性强

  Spark 生态圈即 BDAS（伯克利数据分析栈）包含了 Spark Core、Spark SQL、Spark

Streaming、MLLib 和 GraphX 等组件，这些组件分别处理 Spark Core 提供内存计算框架、SparkStreaming 的实时处理应用、Spark SQL 的即席查询、MLlib 或 MLbase 的机器学习和 GraphX 的图处理，它们都是由 AMP 实验室提供，能够无缝的集成并提供一站式解决平台。



- 随处运行

Spark 具有很强的适应性，能够读取 HDFS、Cassandra、HBase、S3 和 Techyon 为持久层读写原生数据，能够以 Mesos、YARN 和自身携带的 Standalone 作为资源管理器调度 job，来完成 Spark 应用程序的计算。

1.2 Spark 与 Hadoop 差异

Spark 是在借鉴了 MapReduce 之上发展而来的，继承了其分布式并行计算的优点并改进了 MapReduce 明显的缺陷，具体如下：

首先，Spark 把中间数据放到内存中，迭代运算效率高。MapReduce 中计算结果需要落地，保存到磁盘上，这样势必会影响整体速度，而 Spark 支持 DAG 图的分布式并行计算的编程框架，减少了迭代过程中数据的落地，提高了处理效率。

其次，Spark 容错性高。Spark 引进了弹性分布式数据集 RDD (Resilient Distributed Dataset) 的抽象，它是分布在一组节点中的只读对象集合，这些集合是弹性的，如果数据集一部分丢失，则可以根据“血统”（即充许基于数据衍生过程）对它们进行重建。另外在 RDD 计算时可以通过 CheckPoint 来实现容错，而 CheckPoint 有两种方式：CheckPoint Data，和 Logging The Updates，用户可以控制采用哪种方式来实现容错。

最后，Spark 更加通用。不像 Hadoop 只提供了 Map 和 Reduce 两种操作，Spark 提供的

数据集操作类型有很多种，大致分为：Transformations 和 Actions 两大类。Transformations 包括 Map、Filter、FlatMap、Sample、GroupByKey、ReduceByKey、Union、Join、Cogroup、MapValues、Sort 和 PartionBy 等多种操作类型，同时还提供 Count, Actions 包括 Collect、Reduce、Lookup 和 Save 等操作。另外各个处理节点之间的通信模型不再像 Hadoop 只有 Shuffle 一种模式，用户可以命名、物化，控制中间结果的存储、分区等。

1.3 Spark 的适用场景

目前大数据处理场景有以下几个类型：

1. 复杂的批量处理（Batch Data Processing），偏重点在于处理海量数据的能力，至于处理速度可忍受，通常的时间可能是在数十分钟到数小时；
2. 基于历史数据的交互式查询（Interactive Query），通常的时间在数十秒到数十分钟之间
3. 基于实时数据流的数据处理（Streaming Data Processing），通常在数百毫秒到数秒之间

目前对以上三种场景需求都有比较成熟的处理框架，第一种情况可以用 Hadoop 的 MapReduce 来进行批量海量数据处理，第二种情况可以 Impala 进行交互式查询，对于第三中情况可以用 Storm 分布式处理框架处理实时流式数据。以上三者都是比较独立，各自一套维护成本比较高，而 Spark 的出现能够一站式平台满意以上需求。

通过以上分析，总结 Spark 场景有以下几个：

- Spark 是基于内存的迭代计算框架，适用于需要多次操作特定数据集的应用场合。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，数据量小但是计算密集度较大的场合，受益就相对较小
- 由于 RDD 的特性，Spark 不适用那种异步细粒度更新状态的应用，例如 web 服务的存储或者是增量的 web 爬虫和索引。就是对于那种增量修改的应用模型不适合
- 数据量不是特别大，但是要求实时统计分析需求

1.4 Spark 演进时间表

演进时间表：

- 2009 年由 Berkeley's AMPLab 开始编写最初的源代码
- 2010 年开放源代码
- 2013 年 6 月进入 Apache 孵化器项目
- 2014 年 2 月成为 Apache 的顶级项目（8 个月时间）

- 2014 年 5 月底 Spark1.0.0 发布
- 2014 年 9 月 Spark1.1.0 发布
- 2014 年 12 月 Spark1.2.0 发布

目前情况：

- 目前已经有 30+公司 100+开发者在提交代码
- Hadoop 最大的厂商 Cloudera 宣称加大 Spark 框架的投入来取代 Mapreduce
- Hortonworks
- Hadoop 厂商 MapR 投入 Spark 阵营
- Apache Mahout 放弃 MapReduce，将使用 Spark 作为后续算子的计算平台

1.5 Spark 成功案例

目前大数据在互联网公司主要应用在广告、报表、推荐系统等业务上。在广告业务方面需要大数据做应用分析、效果分析、定向优化等，在推荐系统方面则需要大数据优化相关排名、个性化推荐以及热点点击分析等。这些应用场景的普遍特点是计算量大、效率要求高。Spark 恰恰满足了这些要求，该项目一经推出便受到开源社区的广泛关注和好评。并在近两年内发展成为大数据处理领域最炙手可热的开源项目。

本章将列举国内外应用 Spark 的成功案例。

1. 腾讯

广点通是最早使用 Spark 的应用之一。腾讯大数据精准推荐借助 Spark 快速迭代的优势，围绕“数据+算法+系统”这套技术方案，实现了在“数据实时采集、算法实时训练、系统实时预测”的全流程实时并行高维算法，最终成功应用于广点通 pCTR 投放系统上，支持每天上百亿的请求量。

基于日志数据的快速查询系统业务构建于 Spark 之上的 Shark ,利用其快速查询以及内存表等优势，承担了日志数据的即席查询工作。在性能方面，普遍比 Hive 高 2-10 倍，如果使用内存表的功能，性能将会比 Hive 快百倍。

2. Yahoo

Yahoo 将 Spark 用在 Audience Expansion 中的应用。Audience Expansion 是广告中寻找目标用户的一种方法：首先广告者提供一些观看了广告并且购买产品的样本客户，据此进行

学习，寻找更多可能转化的用户，对他们定向广告。Yahoo 采用的算法是 logistic regression。同时由于有些 SQL 负载需要更高的服务质量，又加入了专门跑 Shark 的大内存集群，用于取代商业 BI/OLAP 工具，承担报表/仪表盘和交互式/即席查询，同时与桌面 BI 工具对接。目前在 Yahoo 部署的 Spark 集群有 112 台节点，9.2TB 内存。

3. 淘宝

阿里搜索和广告业务，最初使用 Mahout 或者自己写的 MR 来解决复杂的机器学习，导致效率低而且代码不易维护。淘宝技术团队使用了 Spark 来解决多次迭代的机器学习算法、高计算复杂度的算法等。将 Spark 运用于淘宝的推荐相关算法上,同时还利用 Graphx 解决了许多生产问题，包括以下计算场景：基于度分布的中枢节点发现、基于最大连通图的社区发现、基于三角形计数的关系衡量、基于随机游走的用户属性传播等。

4. 优酷土豆

优酷土豆在使用 Hadoop 集群的突出问题主要包括：第一是商业智能 BI 方面，分析师提交任务之后需要等待很久才得到结果；第二就是大数据量计算，比如进行一些模拟广告投放之时，计算量非常大的同时对效率要求也比较高，最后就是机器学习和图计算的迭代运算也是需要耗费大量资源且速度很慢。

最终发现这些应用场景并不适合在 MapReduce 里面去处理。通过对比，发现 Spark 性能比 MapReduce 提升很多。首先，交互查询响应快，性能比 Hadoop 提高若干倍；模拟广告投放计算效率高、延迟小（同 hadoop 比延迟至少降低一个数量级）；机器学习、图计算等迭代计算，大大减少了网络传输、数据落地等，极大的提高的计算性能。目前 Spark 已经广泛使用在优酷土豆的视频推荐（图计算）、广告业务等。

1.6 Spark 术语

1.6.1 Spark 运行模式

运行环境	模式	描述
Local	本地模式	常用于本地开发测试，本地还分为local单线程和local-cluster多线程;
Standalone	集群模式	典型的Mater/slave模式，不过也能看出Master是有单点故障的；Spark支持 ZooKeeper来实现HA
On yarn	集群模式	运行在yarn资源管理器框架之上，由yarn负责资源管理，Spark负责任务调度和计算
On mesos	集群模式	运行在mesos资源管理器框架之上，由mesos负责资源管理，Spark负责任务调度和计算

On cloud	集群模式	比如AWS的EC2，使用这个模式能很方便的访问Amazon的S3; Spark支持多种分布式存储系统：HDFS和S3

1.6.2 Spark 常用术语

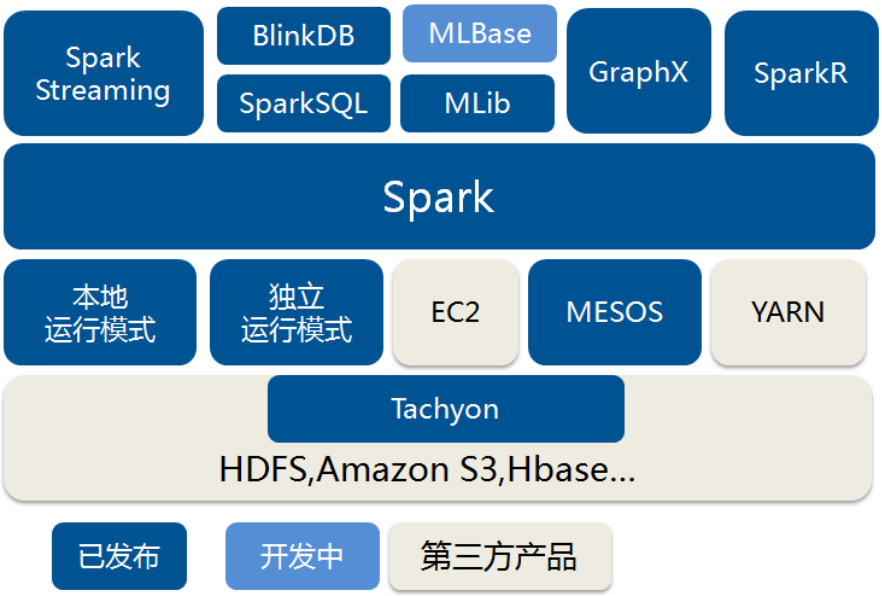
术语	描述
Application	Spark的应用程序，包含一个Driver program和若干Executor
SparkContext	Spark应用程序的入口，负责调度各个运算资源，协调各个Worker Node上的Executor
Driver Program	运行Application的main()函数并且创建SparkContext
Executor	是为Application运行在Worker node上的一个进程，该进程负责运行Task，并且负责将数据存在内存或者磁盘上。 每个Application都会申请各自的Executor来处理任务
Cluster Manager	在集群上获取资源的外部服务 (例如：Standalone、Mesos、Yarn)
Worker Node	集群中任何可以运行Application代码的节点，运行一个或多个Executor进程
Task	运行在Executor上的工作单元
Job	SparkContext提交的具体Action操作，常和Action对应
Stage	每个Job会被拆分很多组task ,每组任务被称为Stage ,也称 TaskSet
RDD	是Resilient distributed datasets的简称，中文为弹性分布式数据集;是Spark最核心的模块和类
DAGScheduler	根据Job构建基于Stage的DAG，并提交Stage给TaskScheduler
TaskScheduler	将Taskset提交给Worker node集群运行并返回结果
Transformations	是Spark API的一种类型，Transformation返回值还是一个RDD，所有的Transformation采用的都是懒策略，如果只是将Transformation提交是不会执行计算的
Action	是Spark API的一种类型，Action返回值不是一个RDD，而是一个scala集合；计算只有在Action被提交的时候计算才被触发。

2 生态系统

Spark 生态圈也称为 BDAS（伯克利数据分析栈），是伯克利 APMLab 实验室打造的，力图在算法（Algorithms）、机器（Machines）、人（People）之间通过大规模集成来展现大数据

应用的一个平台。伯克利 AMPLab 运用大数据、云计算、通信等各种资源以及各种灵活的技术方案，对海量不透明的数据进行甄别并转化为有用的信息，以供人们更好的理解世界。该生态圈已经涉及到机器学习、数据挖掘、数据库、信息检索、自然语言处理和语音识别等多个领域。

Spark 生态圈以 Spark Core 为核心，从 HDFS、Amazon S3 和 HBase 等持久层读取数据，以 MESS、YARN 和自身携带的 Standalone 为资源管理器调度 Job 完成 Spark 应用程序的计算。 这些应用程序可以来自于不同的组件，如 Spark Shell/Spark Submit 的批处理、Spark Streaming 的实时处理应用、Spark SQL 的即席查询、BlinkDB 的权衡查询、MLlib/MLbase 的机器学习、GraphX 的图处理和 SparkR 的数学计算等等。



2.1 Spark Core

前面介绍了 Spark Core 的基本情况，以下总结一下 Spark 内核架构：

- 提供了有向无环图（DAG）的分布式并行计算框架，并提供 Cache 机制来支持多次迭代计算或者数据共享，大大减少迭代计算之间读取数据局的开销，这对于需要进行多次迭代的数据挖掘和分析性能有很大提升
- 在 Spark 中引入了 RDD (Resilient Distributed Dataset) 的抽象，它是分布在一组节点中的只读对象集合，这些集合是弹性的，如果数据集一部分丢失，则可以根据“血统”对它们进行重建，保证了数据的高容错性；
- 移动计算而非移动数据，RDD Partition 可以就近读取分布式文件系统中的数据块到各个节点内存中进行计算
- 使用多线程池模型来减少 task 启动开稍
- 采用容错的、高可伸缩性的 akka 作为通讯框架

2.2 SparkStreaming

SparkStreaming 是一个对实时数据流进行高通量、容错处理的流式处理系统，可以对多种数据源（如 Kdfka、Flume、Twitter、Zero 和 TCP 套接字）进行类似 Map、Reduce 和 Join 等复杂操作，并将结果保存到外部文件系统、数据库或应用到实时仪表盘。

Spark Streaming 构架

- 计算流程：Spark Streaming 是将流式计算分解成一系列短小的批处理作业。这里的批处理引擎是 Spark Core，也就是把 Spark Streaming 的输入数据按照 batch size（如 1 秒）分成一段一段的数据（Discretized Stream），每一段数据都转换成 Spark 中的 RDD（Resilient Distributed Dataset），然后将 Spark Streaming 中对 DStream 的 Transformation 操作变为针对 Spark 中对 RDD 的 Transformation 操作，将 RDD 经过操作变成中间结果保存在内存中。整个流式计算根据业务的需求可以对中间的结果进行叠加或者存储到外部设备。下图显示了 Spark Streaming 的整个流程。

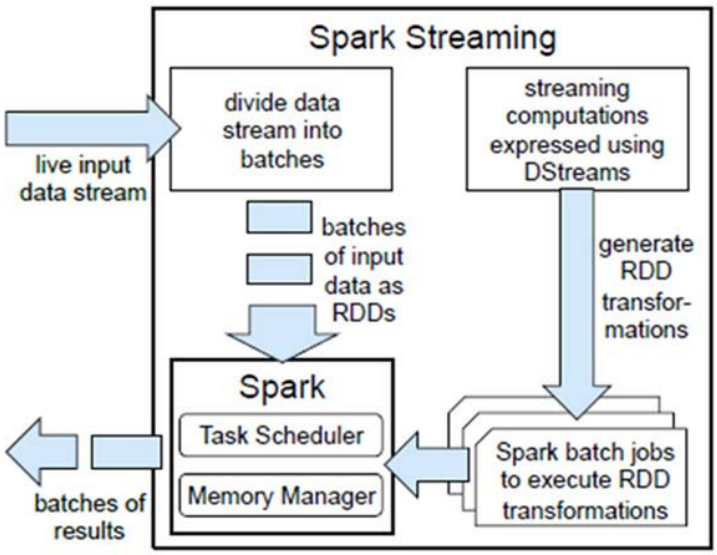
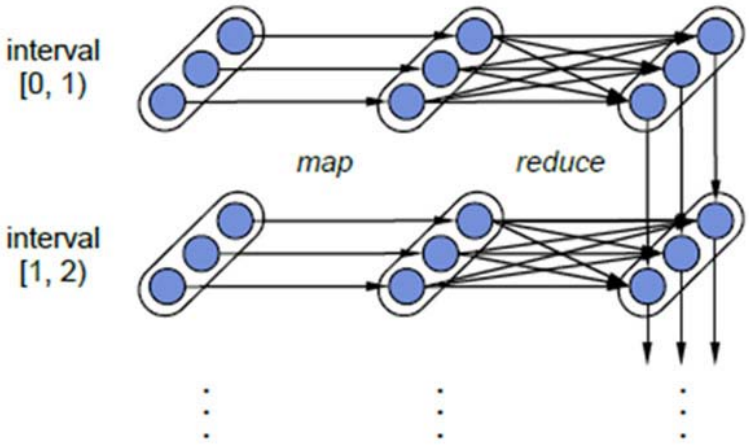


图 Spark Streaming 构架

- 容错性：对于流式计算来说，容错性至关重要。首先我们要明确一下 Spark 中 RDD 的容错机制。每一个 RDD 都是一个不可变的分布式可重算的数据集，其记录着确定性的操作继承关系（lineage），所以只要输入数据是可容错的，那么任意一个 RDD 的分区（Partition）出错或不可用，都是可以利用原始输入数据通过转换操作而重新算出的。

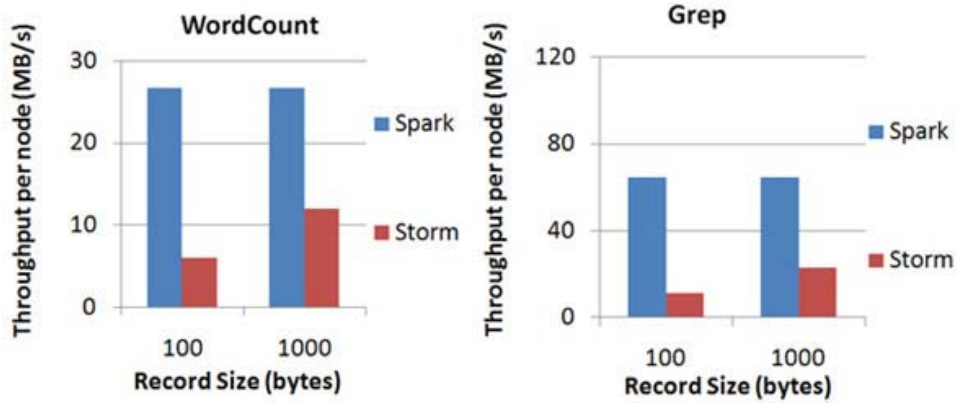
对于 Spark Streaming 来说，其 RDD 的传承关系如下图所示，图中的每一个椭圆形表示一个 RDD，椭圆形中的每个圆形代表一个 RDD 中的一个 Partition，图中的每一列的多个 RDD 表示一个 DStream（图中有三个 DStream），而每一行最后一个 RDD 则表示每一个 Batch Size 所产生的中间结果 RDD。我们可以看到图中的每一个 RDD 都是通过 lineage 相

连接的，由于 Spark Streaming 输入数据可以来自于磁盘，例如 HDFS（多份拷贝）或是来自于网络的数据流（Spark Streaming 会将网络输入数据的每一个数据流拷贝两份到其他的机器）都能保证容错性，所以 RDD 中任意的 Partition 出错，都可以并行地在其他机器上将缺失的 Partition 计算出来。这个容错恢复方式比连续计算模型（如 Storm）的效率更高。



Spark Streaming 中 RDD 的 lineage 关系图

- 实时性：对于实时性的讨论，会牵涉到流式处理框架的应用场景。Spark Streaming 将流式计算分解成多个 Spark Job，对于每一段数据的处理都会经过 Spark DAG 图分解以及 Spark 的任务集的调度过程。对于目前版本的 Spark Streaming 而言，其最小的 Batch Size 的选取在 0.5~2 秒钟之间（Storm 目前最小的延迟是 100ms 左右），所以 Spark Streaming 能够满足除对实时性要求非常高（如高频实时交易）之外的所有流式准实时计算场景。
- 扩展性与吞吐量：Spark 目前在 EC2 上已能够线性扩展到 100 个节点（每个节点 4Core），可以以数秒的延迟处理 6GB/s 的数据量（60M records/s），其吞吐量也比流行的 Storm 高 2～5 倍，图 4 是 Berkeley 利用 WordCount 和 Grep 两个用例所做的测试，在 Grep 这个测试中，Spark Streaming 中的每个节点的吞吐量是 670k records/s，而 Storm 是 115k records/s。



Spark Streaming 与 Storm 吞吐量比较图

2.3 Spark SQL

Shark 是 SparkSQL 的前身，它发布于 3 年前，那个时候 Hive 可以说是 SQL on Hadoop 的唯一选择，负责将 SQL 编译成可扩展的 MapReduce 作业，鉴于 Hive 的性能以及与 Spark 的兼容，Shark 项目由此而生。

Shark 即 Hive on Spark，本质上是通过 Hive 的 HQL 解析，把 HQL 翻译成 Spark 上的 RDD 操作，然后通过 Hive 的 metadata 获取数据库里的表信息，实际 HDFS 上的数据和文件，会由 Shark 获取并放到 Spark 上运算。Shark 的最大特性就是快和与 Hive 的完全兼容，且可以在 shell 模式下使用 rdd2sql()这样的 API，把 HQL 得到的结果集，继续在 scala 环境下运算，支持自己编写简单的机器学习或简单分析处理函数，对 HQL 结果进一步分析计算。

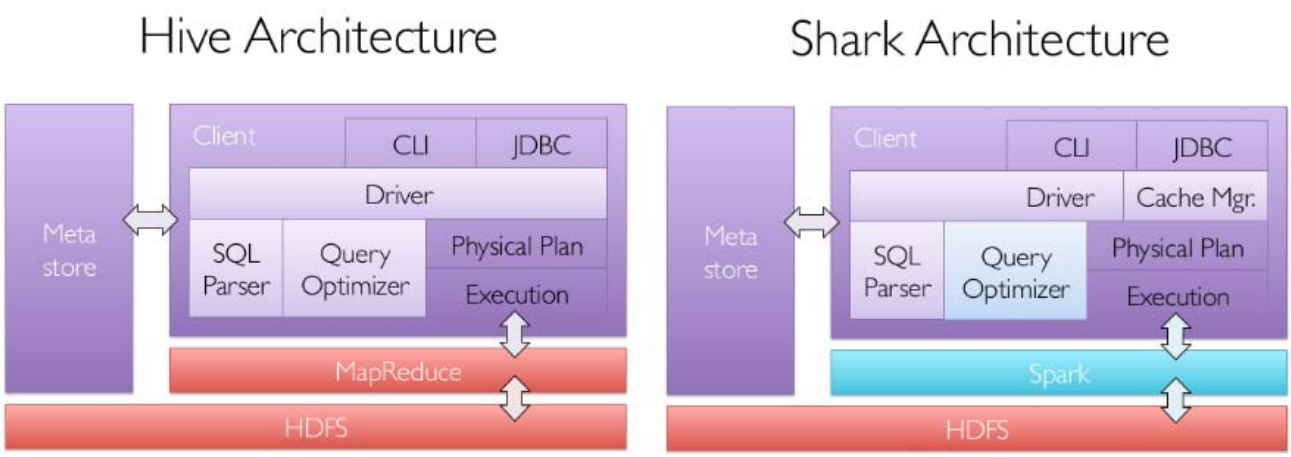
在 2014 年 7 月 1 日的 Spark Summit 上，Databricks 宣布终止对 Shark 的开发，将重点放到 Spark SQL 上。Databricks 表示，Spark SQL 将涵盖 Shark 的所有特性，用户可以从 Shark 0.9 进行无缝的升级。在会议上，Databricks 表示，Shark 更多是对 Hive 的改造，替换了 Hive 的物理执行引擎，因此会有一个很快的速度。然而，不容忽视的是，Shark 继承了大量的 Hive 代码，因此给优化和维护带来了大量的麻烦。随着性能优化和先进分析整合的进一步加深，基于 MapReduce 设计的部分无疑成为了整个项目的瓶颈。因此，为了更好的发展，给用户提供一个更好的体验，Databricks 宣布终止 Shark 项目，从而将更多的精力放到 Spark SQL 上。

Spark SQL 允许开发人员直接处理 RDD，同时也可查询例如在 Apache Hive 上存在的外部数据。Spark SQL 的一个重要特点是其能够统一处理关系表和 RDD，使得开发人员可以轻松地使用 SQL 命令进行外部查询，同时进行更复杂的数据分析。除了 Spark SQL 外，Michael 还谈到 Catalyst 优化框架，它允许 Spark SQL 自动修改查询方案，使 SQL 更有效地执行。

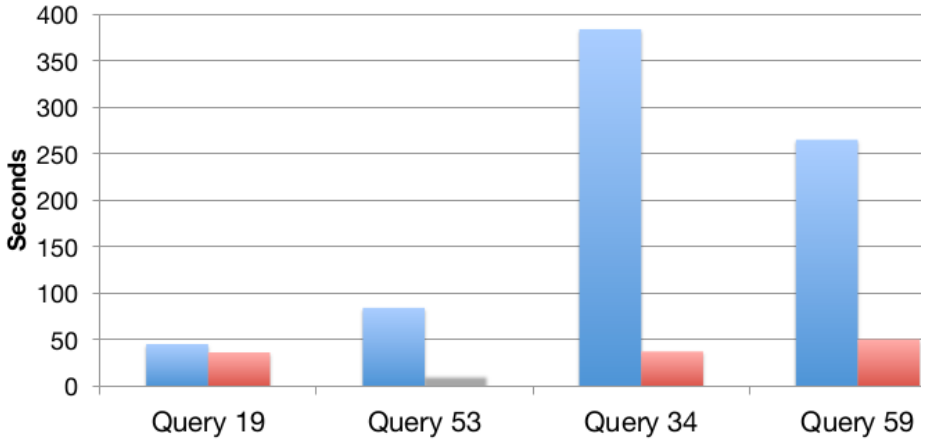
还有 Shark 的作者是来自中国的博士生辛湜（Reynold Xin），也是 Spark 的核心成员，具体信息可以看他的专访 http://www.csdn.net/article/2013-04-26/2815057-Spark-Reynold

Spark SQL 的特点:

- 引入了新的 RDD 类型 SchemaRDD，可以象传统数据库定义表一样来定义 SchemaRDD，SchemaRDD 由定义了列数据类型的行对象构成。SchemaRDD 可以从 RDD 转换过来，也可以从 Parquet 文件读入，也可以使用 HiveQL 从 Hive 中获取。
- 内嵌了 Catalyst 查询优化框架，在把 SQL 解析成逻辑执行计划之后，利用 Catalyst 包里的一些类和接口，执行了一些简单的执行计划优化，最后变成 RDD 的计算
- 在应用程序中可以混合使用不同来源的数据，如可以将来自 HiveQL 的数据和来自 SQL 的数据进行 Join 操作。



Shark 的出现使得 SQL-on-Hadoop 的性能比 Hive 有了 10-100 倍的提高， 那么，摆脱了 Hive 的限制，SparkSQL 的性能又有怎么样的表现呢？虽然没有 Shark 相对于 Hive 那样瞩目地性能提升，但也表现得非常优异，如下图所示：



为什么 sparkSQL 的性能会得到怎么大的提升呢？主要 sparkSQL 在下面几点做了优化：

1. **内存列存储（In-Memory Columnar Storage）** sparkSQL 的表数据在内存中存储不是采用原生态的 JVM 对象存储方式，而是采用内存列存储；
2. **字节码生成技术（Bytecode Generation）** Spark1.1.0 在 Catalyst 模块的 expressions 增加了 codegen 模块，使用动态字节码生成技术，对匹配的表达式采用特定的代码动态编译。另外对 SQL 表达式都作了 CG 优化， CG 优化的实现主要还是依靠 Scala2.10 的运行时放射机制（runtime reflection）；
3. **Scala 代码优化** SparkSQL 在使用 Scala 编写代码的时候，尽量避免低效的、容易 GC 的代码；尽管增加了编写代码的难度，但对于用户来说接口统一。

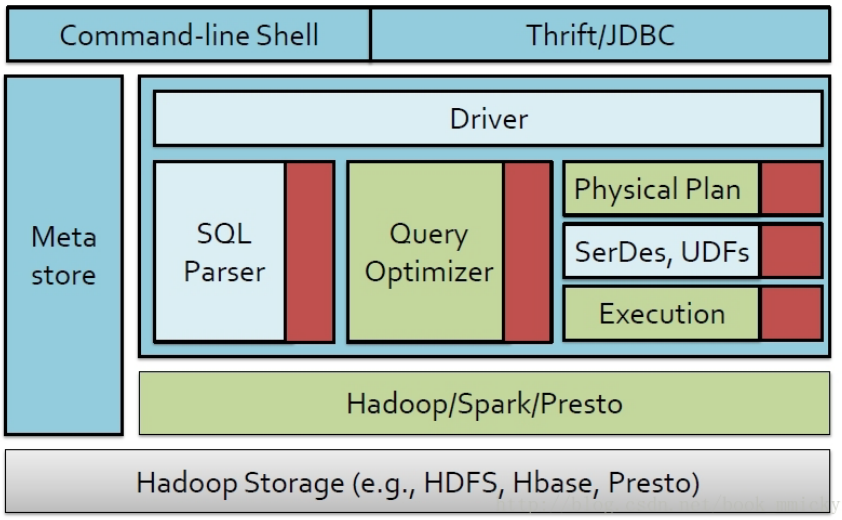
2.4 BlinkDB

BlinkDB 是一个用于在海量数据上运行交互式 SQL 查询的大规模并行查询引擎，它允许用户通过权衡数据精度来提升查询响应时间，其数据的精度被控制在允许的误差范围内。为了达

到这个目标，BlinkDB 使用两个核心思想:

- 一个自适应优化框架，从原始数据随着时间的推移建立并维护一组多维样本；
- 一个动态样本选择策略，选择一个适当大小的示例基于查询的准确性和（或）响应时间需求。

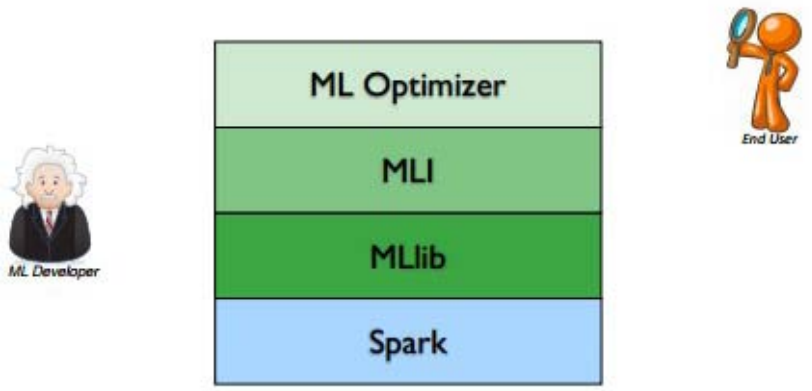
和传统关系型数据库不同，BlinkDB 是一个很有意思的交互式查询系统，就像一个跷跷板，用户需要在查询精度和查询时间上做一权衡；如果用户想更快地获取查询结果，那么将牺牲查询结果的精度；同样的，用户如果想获取更高精度的查询结果，就需要牺牲查询响应时间。用户可以在查询的时候定义一个失误边界。



2.5 MLBase/MLlib

MLBase 是 Spark 生态圈的一部分专注于机器学习，让机器学习的门槛更低，让一些可能并不了解机器学习的用户也能方便地使用 MLbase。MLBase 分为四部分：MLlib、MLI、ML Optimizer 和 MLRuntime。

- ML Optimizer 会选择它认为最适合的已经在内部实现好了的机器学习算法和相关参数，来处理用户输入的数据，并返回模型或别的帮助分析的结果；
- MLI 是一个进行特征抽取和高级 ML 编程抽象的算法实现的 API 或平台；
- MLlib 是 Spark 实现一些常见的机器学习算法和实用程序，包括分类、回归、聚类、协同过滤、降维以及底层优化，该算法可以进行可扩充； MLRuntime 基于 Spark 计算框架，将 Spark 的分布式计算应用到机器学习领域。



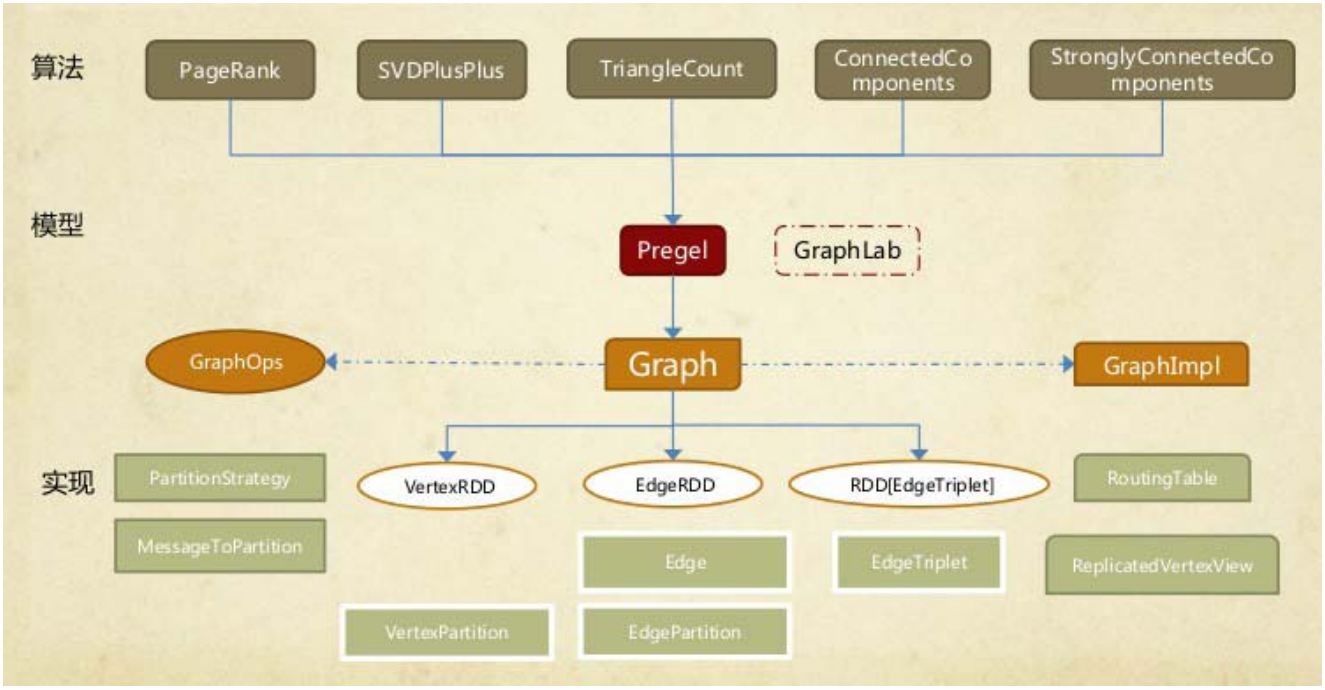
总的来说，MLBase 的核心是他的优化器，把声明式的 Task 转化成复杂的学习计划，产出最优的模型和计算结果。与其他机器学习 Weka 和 Mahout 不同的是：

- MLBase 是分布式的，Weka 是一个单机的系统；
- MLBase 是自动化的，Weka 和 Mahout 都需要使用者具备机器学习技能，来选择自己想要的算法和参数来做处理；
- MLBase 提供了不同抽象程度的接口，让算法可以扩充
- MLBase 基于 Spark 这个平台

2.6 GraphX

GraphX 是 Spark 中用于图(e.g., Web-Graphs and Social Networks)和图并行计算(e.g., PageRank and Collaborative Filtering)的 API,可以认为是 GraphLab(C++)和 Pregel(C++)在 Spark(Scala)上的重写及优化，跟其他分布式图计算框架相比，GraphX 最大的贡献是，在 Spark 之上提供一栈式数据解决方案，可以方便且高效地完成图计算的一整套流水作业。GraphX 最先是伯克利 AMPLAB 的一个分布式图计算框架项目，后来整合到 Spark 中成为一个核心组件。

GraphX 的核心抽象是 Resilient Distributed Property Graph，一种点和边都带属性的有向多重图。它扩展了 Spark RDD 的抽象，有 Table 和 Graph 两种视图，而只需要一份物理存储。两种视图都有自己独有的操作符，从而获得了灵活操作和执行效率。如同 Spark，GraphX 的代码非常简洁。GraphX 的核心代码只有 3 千多行，而在此之上实现的 Pregel 模型，只要短短的 20 多行。GraphX 的代码结构整体下图所示，其中大部分的实现，都是围绕 Partition 的优化进行的。这在某种程度上说明了点分割的存储和相应的计算优化的确是图计算框架的重点和难点。



GraphX 的底层设计有以下几个关键点。

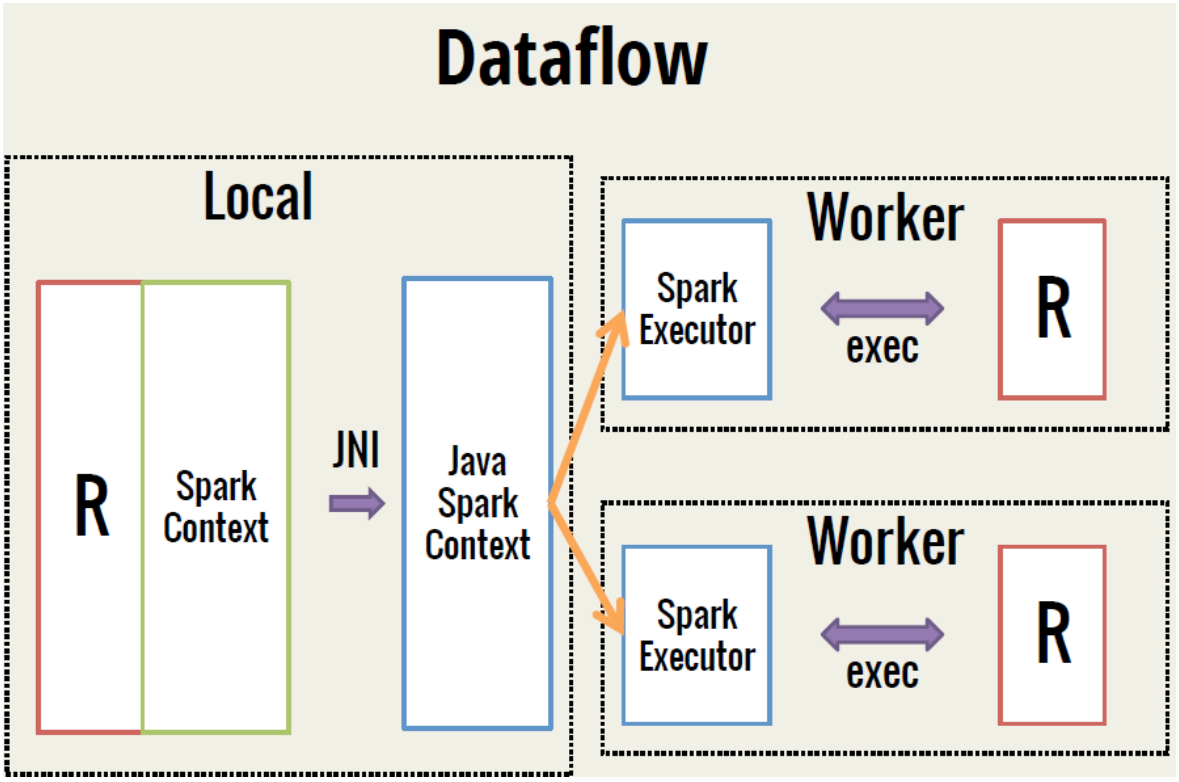
1. 对 Graph 视图的所有操作，最终都会转换成其关联的 Table 视图的 RDD 操作来完成。这样对一个图的计算，最终在逻辑上，等价于一系列 RDD 的转换过程。因此，Graph 最终具备了 RDD 的 3 个关键特性：Immutable、Distributed 和 Fault-Tolerant。其中最关键的是 Immutable（不变性）。逻辑上，所有图的转换和操作都产生了一个新图；物理上，GraphX 会有一定程度的不变顶点和边的复用优化，对用户透明。
2. 两种视图底层共用的物理数据，由 RDD[Vertex-Partition]和 RDD[EdgePartition]这两个 RDD 组成。点和边实际都不是以表 Collection[tuple]的形式存储的，而是由 VertexPartition/EdgePartition 在内部存储一个带索引结构的分片数据块，以加速不同视图下的遍历速度。不变的索引结构在 RDD 转换过程中是共用的，降低了计算和存储开销。
3. 图的分布式存储采用点分割模式，而且使用 partitionBy 方法，由用户指定不同的划分策略（PartitionStrategy）。划分策略会将边分配到各个 EdgePartition，顶点 Master 分配到各个 VertexPartition，EdgePartition 也会缓存本地边关联点的 Ghost 副本。划分策略的不同会影响到所需要缓存的 Ghost 副本数量，以及每个 EdgePartition 分配的边的均衡程度，需要根据图的结构特征选取最佳策略。目前有 EdgePartition2d、EdgePartition1d、RandomVertexCut 和 CanonicalRandomVertexCut 这四种策略。在淘宝大部分场景下，EdgePartition2d 效果最好。

2.7 SparkR

SparkR 是 AMPLab 发布的一个 R 开发包，使得 R 摆脱单机运行的命运，可以作为 Spark 的 job 运行在集群上，极大得扩展了 R 的数据处理能力。

SparkR 的几个特性：

- 提供了 Spark 中弹性分布式数据集（RDD）的 API，用户可以在集群上通过 R shell 交互性的运行 Spark job。
- 支持序化闭包功能，可以将用户定义函数中所引用到的变量自动序化发送到集群中其他的机器上。
- SparkR 还可以很容易地调用 R 开发包 ,只需要在集群上执行操作前用 includePackage 读取 R 开发包就可以了，当然集群上要安装 R 开发包。



2.8 Tachyon

Tachyon 是一个高容错的分布式文件系统 ,允许文件以内存的速度在集群框架中进行可靠的共享，就像 Spark 和 MapReduce 那样。通过利用信息继承，内存侵入，Tachyon 获得了高性能。Tachyon 工作集文件缓存在内存中，并且让不同的 Jobs/Queries 以及框架都能内存的速度来访问缓存文件”。因此，Tachyon 可以减少那些需要经常使用的数据集通过访问磁盘来获得的次数。Tachyon 兼容 Hadoop，现有的 Spark 和 MR 程序不需要任何修改而运行。

在 2013 年 4 月，AMPLab 共享了其 Tachyon 0.2.0 Alpha 版本的 Tachyon，其宣称性能为 HDFS 的 300 倍，继而受到了极大的关注。Tachyon 的几个特性如下：

- **JAVA-Like File API**

Tachyon 提供类似 JAVA File 类的 API,

- **兼容性**

Tachyon 实现了 HDFS 接口，所以 Spark 和 MR 程序不需要任何修改即可运行。

- **可插拔的底层文件系统**

Tachyon 是一个可插拔的底层文件系统，提供容错功能。tachyon 将内存数据记录在底层文件系统。它有一个通用的接口，使得可以很容易的插入到不同的底层文件系统。目前支持 HDFS，S3，GlusterFS 和单节点的本地文件系统，以后将支持更多的文件系统。

参考资料：

（1）Spark 官网 http://spark.apache.org

（2）《Spark1.0.0 生态圈一览》 http://blog.csdn.net/book_mmicky/article/details/29362405

（3）《大数据计算新贵 Spark 在腾讯雅虎优酷成功应用解析》 http://www.csdn.net/article/2014-06-05/2820089

（4）《Spark Streaming：大规模流式数据处理的新贵》 http://www.csdn.net/article/2014-01-28/2818282-Spark-Streaming-big-data

（5）Spark SQL 介绍《sparkSQL1.1 入门》http://blog.csdn.net/bluejoe2000/article/details/41247857

（6）《快刀初试：Spark GraphX 在淘宝的实践》 http://www.csdn.net/article/2014-08-07/2821097

（7）《基于 Spark 的图计算框架 GraphX 入门介绍》 http://suanfazu.com/t/ji-yu-sparkde-tu-ji-suan-kuang-jia-graphx-ru-men-jie-shao/244

（8）【Spark 专刊】Spark 最佳学习路径（作者：黄忠）