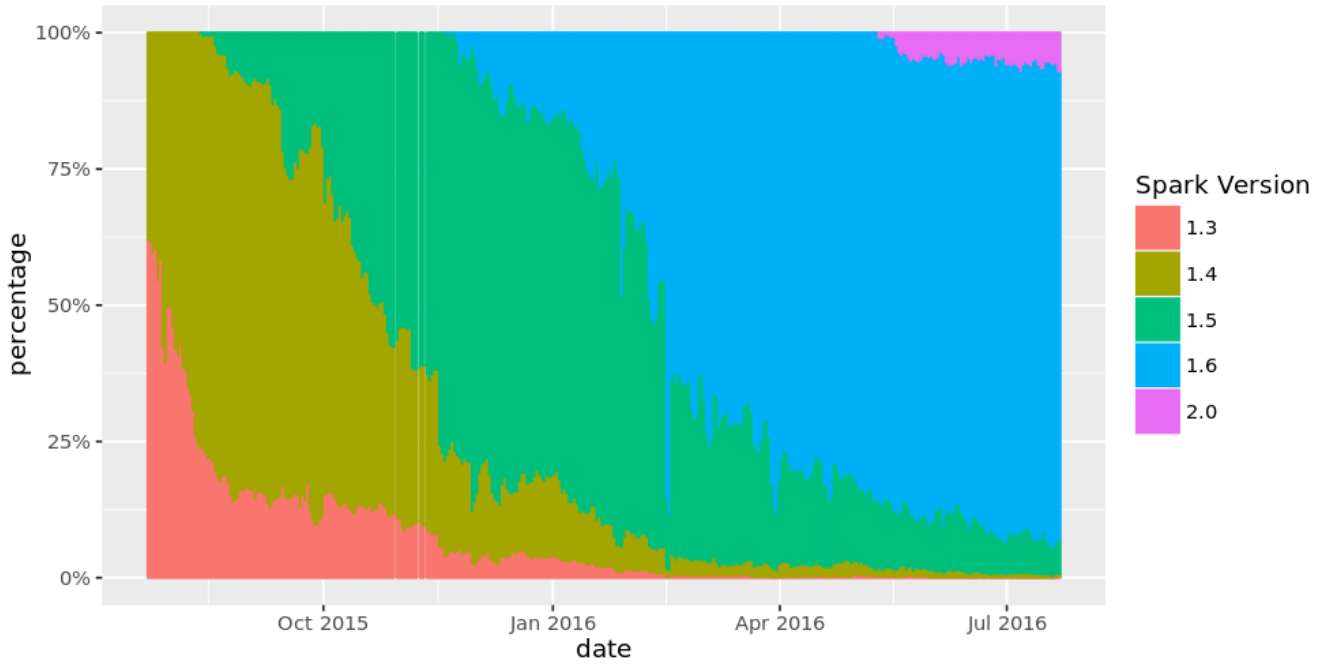
Apache Spark 2.0重大功能介绍

Apache Spark 2.0发布信息可以参见《Apache Spark 2.0.0正式发布及其功能介绍》

我们很荣幸地宣布，自7月26日起Databricks开始提供Apache Spark 2.0的下载，这个版本是基于社区在过去两年的经验总结而成，不但加入了用户喜爱的功能，也修复了之前的痛点。

本文总结了Spark 2.0的三大主题：更简单、更快速、更智能，另有Spark 2.0内容的文章汇总介绍了更多细节。

两个月前，Databricks发布了Apache Spark 2.0的技术预览版，如下表所见，目前我们有10%的集群都在使用这个版本，根据客户使用新版的经验及反馈意见，新版得以发布，Databricks很开心能成为Spark 2.0的首个商业供应商。



现在，我们来深入了解一下Apache Spark 2.0的新特性。

更简单：ANSI SQL与更合理的API

Spark让我们引以为豪的一点就是所创建的API简单、直观、便于使用，Spark 2.0延续了这一传统，并在两个方面凸显了优势：

1、标准的SQL支持；
2、数据框（DataFrame）/Dataset（数据集）API的统一。

在SQL方面，我们已经对Spark的SQL功能做了重大拓展，引入了新的ANSI

SQL解析器，并支持子查询功能。Spark 2.0可以运行所有99个TPC-DS查询（需求SQL：2003中的很多功能支持）。由于SQL是Spark应用所使用的主要接口之一，对SQL功能的拓展大幅削减了将遗留应用移植到Spark时所需的工作。

在编程API方面，我们合理化了API：

1、在Scala/Java中统一了DataFrames与Dataset：从Spark 2.0开始，DataFrames只是行（row）数据集的typealias了。无论是映射、筛选、groupByKey之类的类型方法，还是select、groupBy之类的无类型方法都可用于Dataset的类。此外，这个新加入的Dataset接口是用作Structured Streaming的抽象，由于Python和R语言中编译时类型安全（compile-time type-safety）不属于语言特性，数据集的概念无法应用于这些语言API中。而DataFrame仍是主要的编程抽象，在这些语言中类似于单节点DataFrames的概念，想要了解这些API的相关信息，请参见相关笔记和文章。

2、SparkSession：这是一个新入口，取代了原本的SQLContext与HiveContext。对于DataFrame API的用户来说，Spark常见的混乱源头来自于使用哪个“context”。现在你可以使用SparkSession了，它作为单个入口可以兼容两者，点击这里来查看演示。注意原本的SQLContext与HiveContext仍然保留，以支持向下兼容。
更简单、性能更佳的Accumulator API：我们设计了一个新的Accumulator API，不但在类型层次上更简洁，同时还专门支持基本类型。原本的Accumulator API已不再使用，但为了向下兼容仍然保留。

3、基于DataFrame的机器学习API将作为主ML API出现：在Spark 2.0中，spark.ml包及其“管道”API会作为机器学习的主要API出现，尽管原本的spark.mllib包仍然保留，但以后的开发重点会集中在基于DataFrame的API上。

4、机器学习管道持久化：现在用户可以保留与载入机器学习的管道与模型了，Spark对所有语言提供支持。查看这篇博文以了解更多细节，这篇笔记中也有相关样例。
R语言的分布式算法：增加对广义线性模型（GLM）、朴素贝叶斯算法（NB算法）、存活回归分析（Survival Regression）与聚类算法（K-Means）的支持。

速度更快：用Spark作为编译器

根据我们2015年对Spark的调查，91%的用户认为对Spark来说，性能是最为重要的。因此，性能优化一直是我们在开发Spark时所考虑的重点。在开始Spark 2.0的规划前，我们思考过这个问题：Spark的速度已经很快了，但能否突破极限，让Spark达到原本速度的10倍呢？

带着这个问题，我们切实考虑了在构建Spark物理执行层面时的方式。如果深入调查现代的数据引擎，比如Spark或者其他MPP数据库，我们会发现：CPU循环大多都做了无用功，比如执行虚拟函数调用，或者向CPU缓存或内存读取/写入中间数据；通过减少CPU循环中的浪费来优化性能，一直是我们在现代编译器上长时间以来的工作重点。

Spark 2.0搭载了第二代Tungsten引擎，该引擎是根据现代编译器与MPP数据库的理念来构建的，它将这些理念用于数据处理中，其主要思想就是在运行时使用优化后的字节码，将整体查询

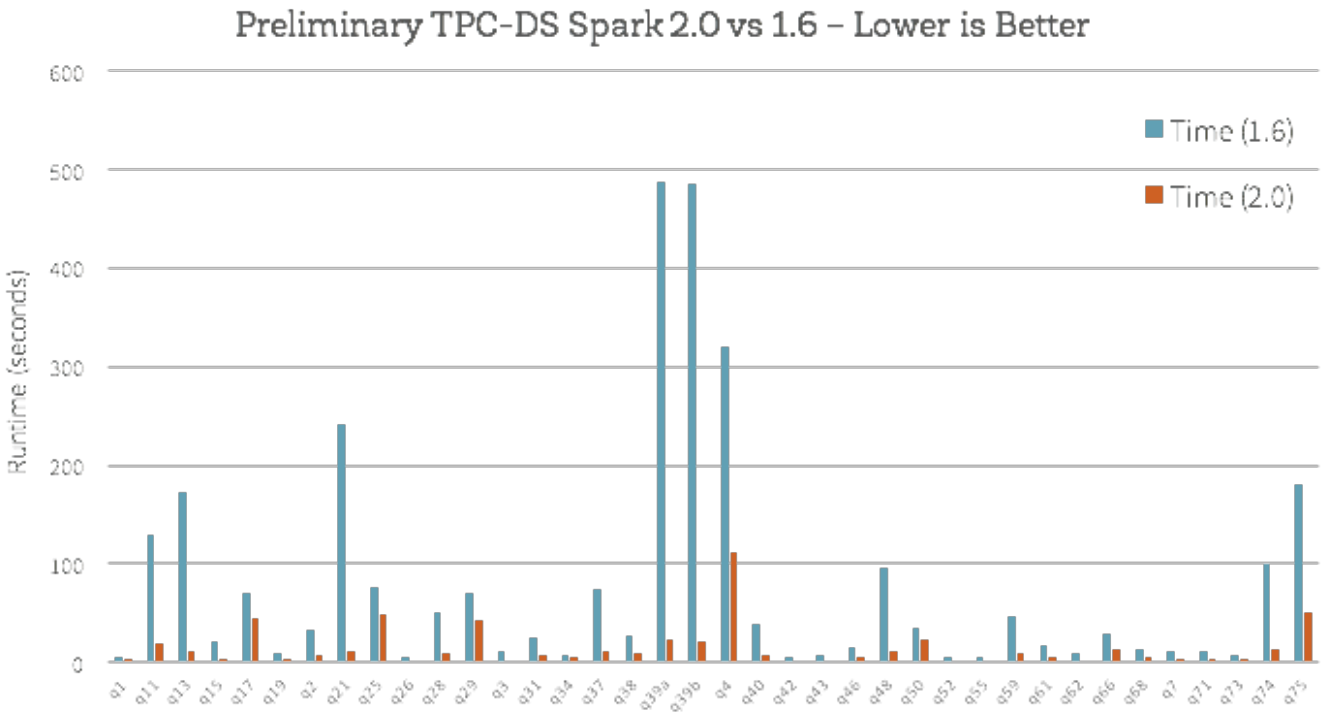
合成为单个函数，不再使用虚拟函数调用，而是利用CPU来注册中间数据。我们将这一技术称为“whole-stage code generation”。

在测试、对比Spark 1.6与Spark 2.0时，我们列出了在单核中处理单行数据所花费的时间（以十亿分之一秒为单位），下面的表格列出了Spark 2.0的优化内容。Spark 1.6包含代码生成技术（code generation）的使用，这一技术如今在一些顶尖的商业数据库中也有运用，正如我们看到的那样，使用了新whole-stage code generation技术后，速度比之前快了一个数量级。

在这篇笔记中可以查看其运用：我们在单台机器上对10亿记录执行了aggregations和joins操作。

primitive	Spark 1.6	Spark 2.0
filter	15ns	1.1ns
sum w/o group	14ns	0.9ns
sum w/ group	79ns	10.7ns
hash join	115ns	4.0ns
sort (8-bit entropy)	620ns	5.3ns
sort (64-bit entropy)	620ns	40ns
sort-merge join	750ns	700ns

这个新的引擎在执行端对端查询时是如何运作的？我们使用TPC-DS查询做了些初步分析，以对比Spark 1.6与Spark 2.0：



除此之外，为了改进Catalyst optimizer优化器对诸如nullability propagation之类常见查询的效果，我们还做了许多工作；另外还改进了矢量化Parquet解码器，新解码器的吞吐量增加了三倍。

点击Apache Spark作为编译器：深入介绍新的Tungsten执行引擎查看Spark 2.0优化的更多细节。

更智能：Structured Streaming

作为首个尝试统一批处理与流处理计算的工具，Spark Streaming一直是大数据处理的领导者。首个流处理API叫做DStream，在Spark 0.7中初次引入，它为开发者提供了一些强大的特性，包括：只有一次语义，大规模容错，以及高吞吐。

然而，在处理了数百个真实世界的Spark Streaming部署之后，我们发现需要在真实世界做决策的应用经常需要不止一个流处理引擎。他们需要深度整合批处理堆栈与流处理堆栈，整合内部存储系统，并且要有处理业务逻辑变更的能力。因此，各大公司需要不止一个流处理引擎，并且需要能让他们开发端对端"持续化应用"的全栈系统。

Spark 2.0使用一个新的API：Structured Streaming模块来处理这些用例，与现有流系统相比，Structured Streaming有三个主要的改进：

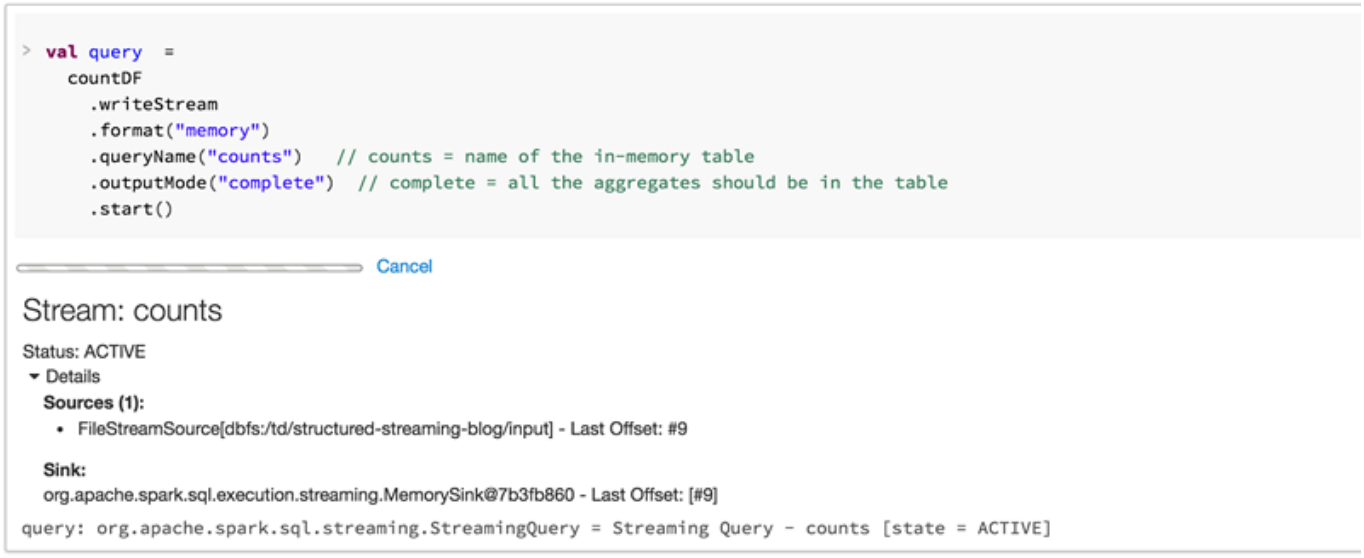
1、与批处理作业集成的API：想要运行流数据计算，开发者可针对DataFrame/Dataset API编写批处理计算，过程非常简单，而Spark会自动在流数据模式中执行计算，也就是说在数据输入时实时更新结果。强大的设计令开发者无需费心管理状态与故障，也无需确保应用与批处理作业的同步，这些都由系统自动解决。此外，针对相同的数据，批处理任务总能给出相同的结果。

2、与存储系统的事务交互： Structured Streaming会在整个引擎及存储系统中处理容错与持久化的问题，使得程序员得以很容易地编写应用，令实时更新的数据库可靠地提供、加入静态数据或者移动数据。

3、与Spark的其它组件的深入集成： Structured Streaming支持通过Spark SQL进行流数据的互动查询，可以添加静态数据以及很多已经使用DataFrames的库，还能让开发者得以构建完整的应用，而不只是数据流管道。未来，我们希望能有更多与MLlib及其它libraries的集成出现。

Spark 2.0搭载了初始alpha版的Strutured Streaming API，这是一个附在DataFrame/Dataset API上的（超小）扩展包。统一之后，对现有的Spark用户来说使用起来非常简单，他们能够利用在Spark 批处理API方面的知识来回答实时的新问题。这里关键的功能包括：支持基于事件时间的处理，无序/延迟数据，sessionization以及非流式数据源与Sink的紧密集成。

我们还更新了Databricks workspace以支持Structured Streaming。例如，在启动streaming查询时，notebook UI会自动显示其状态。



Streaming很明显是一个非常广泛的话题，因此想要了解Spark 2.0中Structured Streaming的更多信息，请关注本博客。

结论

Spark的用户最初使用Spark是因为它的易用性与高性能。Spark 2.0在这些方面达到了之前的两倍，并增加了对多种工作负载的支持，请尝试一下新版本吧。

英文原文：Introducing Apache Spark 2.0：https://databricks.com/blog/2016/07/26/introducing-apache-spark-2-0.html