

# Databricks连城：Spark SQL结构化数据分析

发表于 2015-06-18 15:12 | 7854次阅读 | 来源 CSDN | 0条评论 | 作者 连城

大数据 开源 Spark Spark SQL

**摘要：**Spark SQL面世已一年有余，它不仅接过了Shark的接力棒，继续为Spark用户提供高性能SQL on Hadoop解决方案，还为Spark带来了通用、高效、多元一体的结构化数据处理能力。本文为连城在2015 Spark技术峰会的演讲总结。

数据科学家们早已熟悉的R和Pandas等传统数据分析框架 虽然提供了直观易用的API，却局限于单机，无法覆盖分布式大数据场景。在Spark 1.3.0以Spark SQL原有的SchemaRDD为蓝本，引入了Spark DataFrame API，不仅为Scala、Python、Java三种语言环境提供了形如R和Pandas的API，而且自然而然地继承了Spark SQL的分布式处理能力。此外，Spark 1.2.0中引入的外部数据源API也得到了进一步的完善，集成了完整的数据写入支持，从而补全了Spark SQL多数据源互操作的最后一块拼图。借小数据分析之力，撼大数据分析之巨石；四两拨千斤，不亦乐乎！

## Part of the core distribution since Spark 1.0 (April 2014)

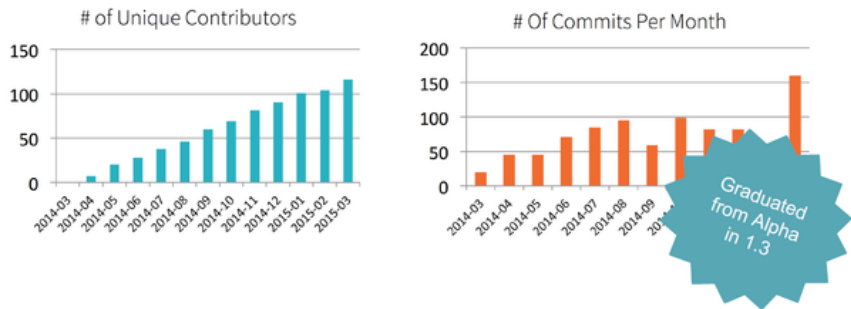


图1：飞速增长中的Spark

Spark SQL是Spark的核心组件之一，于2014年4月随Spark 1.0版一同面世。上图左侧展示了自去年4月份Spark 1.0发布至今开源贡献者数量的增长情况，基本上呈现了一个线性增长的态势。右侧所展示的每月PR数量的增长情况也同样迅猛。值得一提的是，在Spark 1.3当中，Spark SQL终于从alpha阶段毕业，除了部分developer API以外，所有的公共API都已经稳定，可以放心使用了。

作为Shark的继任者，Spark SQL的主要功能之一便是访问现存的Hive数据。在与Hive进行集成的同时，Spark SQL也提供了JDBC/ODBC接口。Tableau、Qlik等第三方工具可以通过该接口接入Spark SQL，借助Spark进行数据处理。

然而，Spark SQL的应用并不局限于SQL。实际上“Spark SQL”这个名字并不恰当。根据Spark官方文档的定义：Spark SQL是一个用于处理结构化数据的Spark组件——该定义强调的是“结构化数据”，而非“SQL”。新近发布的Spark 1.3更加完整的表达了Spark SQL的愿景：让开发者用更精简的代码处理尽量少的数据，同时让Spark SQL自动优化执行过程，以达到降低开发成本，提升数据分析执行效率的目的。为此，我们在Spark 1.3中引入了与R和Python Pandas接口类似的数据Frame API，延续了传统单机数据分析的开发体验，并将之推广到了分布式大数据场景。

### DataFrame

与RDD类似，DataFrame也是一个分布式数据容器。然而DataFrame更像传统数据库的二维表格，除了数据以外，还掌握数据的结构信息，即schema。同时，与Hive类似，DataFrame也支持嵌套数据类型（struct、array和map）。从API易用性的角度上看，DataFrame API提供的是一套高层的关系



CSDN官方微信  
扫描二维码,向CSDN吐槽  
微信号：CSDNnews

程序员移动端订阅下载

### 每日资讯快速浏览

#### 微博关注

CSDN云计算 北京 朝阳区  
加关注

【IDC敲黑板啦：未来企业IT以混合云为主】数字化是企业转型的必由之路，而数字化技术正在融入企业的血液里。IDC认为，基于第三平台的4+6技术是企业数字化转型过程中的关键要素。http://t.cn/RmzemuQ

4月4日 17:04 转发 | 评论

【混合云异军突起 英特尔的全“芯”体验为企业保驾

#### 相关热门文章

#### 热门标签

Hadoop	AWS	移动游戏
Java	Android	iOS
Swift	智能硬件	Docker
OpenStack	VPN	Spark
ERP	IE10	Eclipse
CRM	JavaScript	数据库
Ubuntu	NFC	WAP

#### 下载专辑



微信小程序开发

资源快速  
Top20 热门  
Python  
源码  
download.csdn.net

【资源优选】第八期：20个最热门  
python源码

devexpress控件常用方法总结九

例

操作，比函数式的RDD API要更加友好，门槛更低。由于与R和Pandas的DataFrame类似，Spark DataFrame很好地继承了传统单机数据分析的开发体验。

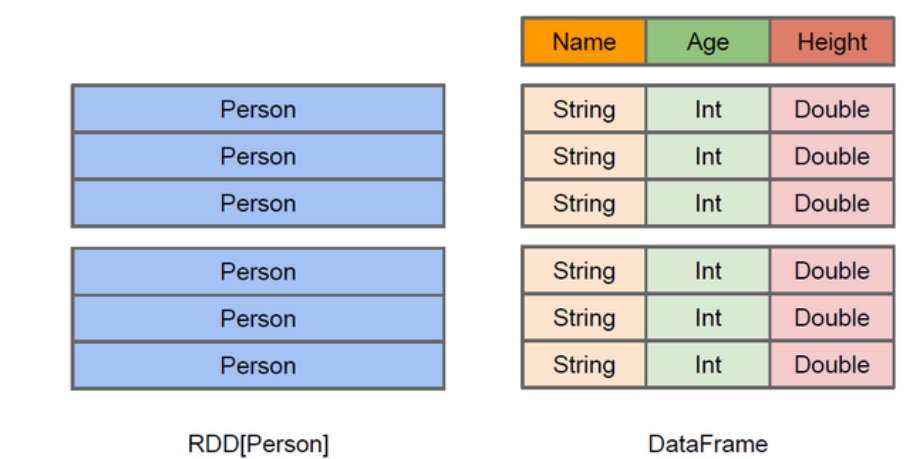


图2：DataFrame和 RDD的区别

上图直观地体现了DataFrame和RDD的区别。左侧的RDD[Person]虽然以Person为类型参数，但Spark框架本身不了解Person`类的内部结构。而右侧的DataFrame却提供了详细的结构信息，使得Spark SQL可以清楚地知道该数据集中包含哪些列，每列的名称和类型各是什么。了解了这些信息之后，Spark SQL的查询优化器就可以进行针对性的优化。举一个不太恰当的例子，其中的差别有些类似于动态类型的Python与静态类型的C++之间的区别。后者由于在编译期有详尽的类型信息，编译期就可以编译出更加有针对性、更加优化的可执行代码。

### 外部数据源API

然而对于用户来说，只有一个结构化的数据抽象还是不够的。数据往往会以各种各样的格式存储在各种各样的系统之上，而用户会希望方便地从不同的数据源获取数据，进行混合处理，再将结果以特定的格式写回数据源或直接予以某种形式的展现。Spark 1.2引入的外部数据源API正是为了解决这一问题而产生的。Spark SQL外部数据源API的一大优势在于，可以将查询中的各种信息下推至数据源处，从而充分利用数据源自身的优化能力来完成列剪枝、过滤条件下推等优化，实现减少IO、提高执行效率的目的。自1.2发布以来，社区内涌现出了多种多样的外部数据源。下图是Spark 1.3支持的各种数据源的一个概览（左侧是Spark SQL内置支持的数据源，右侧为社区开发者贡献的数据源）。在外部数据源API的帮助下，DataFrame实际上成为了各种数据格式和存储系统进行数据交换的中间媒介：在Spark SQL内，来自各处的数据都被加载为DataFrame混合、统一成单一形态，再以之基础进行数据分析和价值提取。

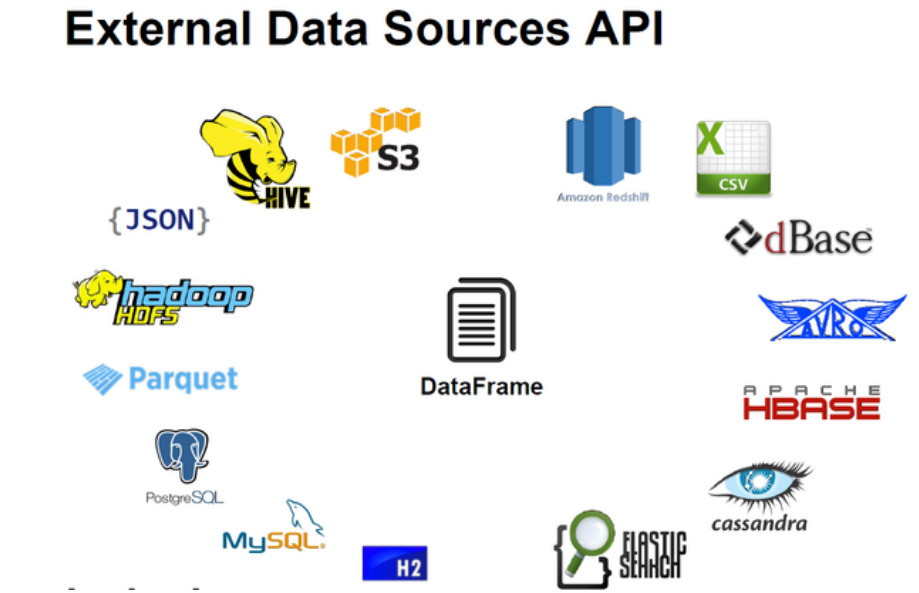


图3：DataFrame支持 的各种外部数据源

### Spark SQL助力大数据分析

精简代码

**BDC**  
2014 中国大数据技术大会  
BIG DATA CONFERENCE CHINA

**[资源优选]第二十二期：Redis优质源码合集**

download.csdn.net

DataFrame带来的最明显的优点之一就是帮助用户进一步精简代码。下图展示了分别用Hadoop MR、Python RDD API和Python DataFrame API实现同一业务逻辑的三段代码片段。显然Hadoop MR的代码量最大，而且并不容易看清楚业务逻辑到底是什么。Python RDD API的版本精简了许多，但仍然不容易看出到底是在干什么。Python DataFrame API的版本相较Python RDD API的版本又更精进了一步；更重要的是，凡是略懂SQL的人，都可以一眼看出它在做什么——可见，DataFrame API不仅可以令代码更加精简，而且显著提升了可读性。Spark 1.3提供了Python、Scala、Java三种语言的DataFrame API binding，供用户按需选用。




## Write Less Code

```
private IntWritable one = new IntWritable(1);
private IntWritable output = new IntWritable();
protected void map(LongWritable key, Text value, Context context) {
    String[] fields = value.split("\t");
    output.set(Integer.parseInt(fields[1]));
    context.write(one, output);
}

private IntWritable one = new IntWritable(1);
private DoubleWritable average = new DoubleWritable();
protected void reduce(IntWritable key, Iterable<IntWritable> values, Context context) {
    int sum = 0;
    int count = 0;

    for(IntWritable value : values) {
        sum += value.get();
        count++;
    }

    average.set(sum / (double) count);
    context.write(key, average);
}
```

```
sc.textFile("hdfs://...")\
  .map(lambda x: (x[0], [x[1], 1]))\
  .reduceByKey(
    lambda x, y: [x[0] + y[0], x[1] + y[1]])\
  .map(lambda x: [x[0], x[1][0] / x[1][1]])\
  .collect()

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

图4：Hadoop MR、Python RDD API、Python DataFrame API代码示例

除此以外，Spark SQL还针对大数据处理中的一些常见场景和模式提供了一些便利的工具，使得用户在处理不同项目中重复出现的模式时可以避免编写重复或高度类似的代码：

• JSON schema自动推导

JSON 是一种可读性良好的重要结构化数据格式，许多原始数据往往以JSON的形式存在。然而JSON数据的体积却过于庞大，不利于批量数据分析。因此一个常见的数据处理步骤就是将JSON转换为ORC、Parquet等高效的列式存储格式。然而，不同版本的JSON数据往往具有不同的schema（例如新版本的Twitter API返回的数据可能比老版本的API返回的数据多出若干列）。人工合并整个JSON数据集所有记录的schema是一件十分枯燥繁琐的任务。Spark SQL在处理JSON数据时可以自动扫描整个数据集，得到所有记录中出现的数据列的全集，推导出完整的schema。（对于同名但不同类型的列，Spark SQL会尝试规约出一个公共类型。）

```
{“Name”: “Alice”, “Gender”: “F”, “Height”: 160}
{“Name”: “Bob”, “Gender”: “M”, “Height”: 175, “Age”: 20}
{“Name”: “Cavin”, “Gender”: “M”, “Height”: 180.3}
```

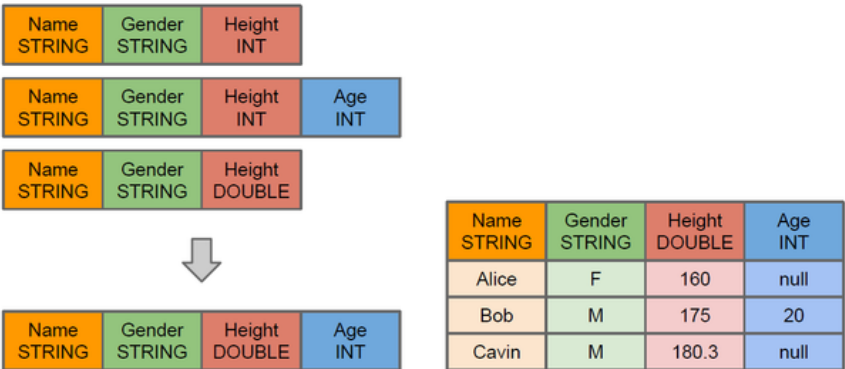


图5：Spark对不规整JSON数据的处理

上图展示了Spark SQL对三条不规整的个人信息JSON记录进行整理和schema推导的过程。第2条记录跟第1条记录类似，但多出了一个age字段，第3条与前两条也很类似，但是身高字段的类型是double而不是int。对此，Spark SQL的JSON数据源作出的处理是，将出现的所有列都纳入

最终的schema中，对于名称相同但类型不同的列，取所有类型的公共父类型（例如int和 double 的公共父类型为double）。通过这样的处理，我们最终就得到了右下方的DataFrame。

• Hive风格的分区表

Hive 的分区表可以认为是一种简易索引。分区表的每一个分区的每一个分区列都对应于一级目录，目录以<列名>=<列值>的格式命名。Spark 1.3中的Parquet数据源实现了自动分区发现的功能：当数据以Hive分区表的目录结构存在时，无须Hive metastore中的元数据，Spark SQL也可以自动将之识别为分区表。于是，在处理这张表时，分区剪枝等分区特有的优化也可以得以实施。

提升执行效率

利用DataFrame API，不仅代码可以更加精简，更重要的是，执行效率也可以得到提升。下图对比了用Scala、Python的RDD API和DataFrame API实现的累加一千万整数对的四段程序的性能对比。可以看到，Python DataFrame API相对于Python RDD API的执行效率有了近五倍的提升。这是因为在DataFrame API实际上仅仅组装了一段体积小巧的逻辑查询计划，Python端只需将查询计划发送到JVM端即可，计算任务的大头都由JVM端负责。在使用 Python RDD API时，Python VM和JVM之间需要进行大量的跨进程数据交换，从而拖慢了Python RDD API的速度。

值得注意的是，不仅Python API有了显著的性能提升，即便是使用Scala，DataFrame API的版本也要比RDD API快一倍。上述示例的逻辑极为简单，查询优化器的作用不明显，那么为什么会有加速效果呢？RDD API是函数式的，强调不变性，在大部分场景下倾向于创建新对象而不是修改老对象。这一特点虽然带来了干净整洁的API，却也使得Spark应用程序在运行期倾向于创建大量临时对象，对GC造成压力。在现有RDD API的基础之上，我们固然可以利用mapPartitions方法来重载RDD单个分片内的数据创建方式，用复用可变对象的方式来减小对象分配和GC的开销，但这牺牲了代码的可读性，而且要求开发者对Spark运行时机制有一定的了解，门槛较高。另一方面，Spark SQL在框架内部已经在各种可能的情况下尽量重用对象，这样做虽然在内部会打破了不变性，但在将数据返回给用户时，还会重新转为不可变数据。利用 DataFrame API进行开发，可以免费地享受到这些优化效果。

减少数据读取

分析大数据，最快的方法就是——忽略它。这里的“忽略”并不是熟视无睹，而是根据查询条件进行恰当的剪枝。

上文讨论分区表时提到的分区剪 枝便是其中一种——当查询的过滤条件中涉及到分区列时，我们可以根据查询条件剪掉肯定不包含目标数据的分区目录，从而减少IO。

对于一些“智能”数据格式，Spark SQL还可以根据数据文件中附带的统计信息来进行剪枝。简单来说，在这类数据格式中，数据是分段保存的，每段数据都带有最大值、最小值、null值数量等一些基本的统计信息。当统计信息表某一数据段肯定不包括符合查询条件的目标数据时，该数据段就可以直接跳过（例如某整数列a某段的最大值为100，而查询条件要求a > 200）。

此外，Spark SQL也可以充分利用RCFile、ORC、Parquet等列式存储格式的优势，仅扫描查询真正涉及的列，忽略其余列的数据。

查询优化

Spark SQL的第三个目标，就是让查询优化器帮助我们优化执行效率，解放开发者的生产力，让新手也可以写出高效的程序。

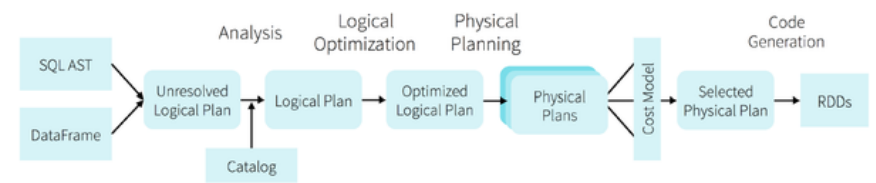


图6：Spark SQL查询优化引擎

DataFrame的背后是 Spark SQL的全套查询优化引擎，其整体架构如上图所示。通过SQL/HiveQL parser或是DataFrame API构造的逻辑执行计划经过analyzer的分析之后再经优化得到优化执行计划，接着再转为物理执行计划，并最终转换为RDD DAG在Spark引擎上执行。

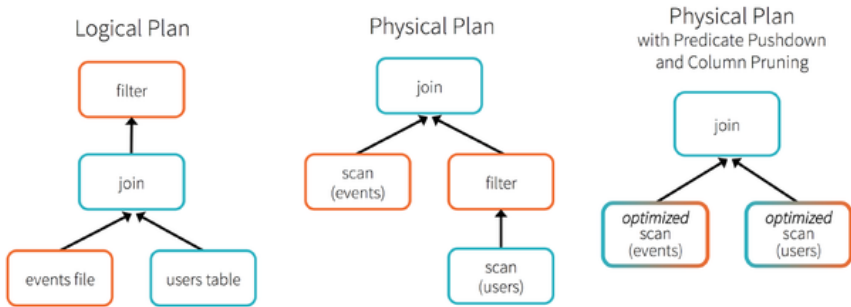


图7：人口数据分析示例

为了说明查询优化，我们来看上图展示的人口数据分析的示例。图中构造了两个DataFrame，将它们join之后又做了一次filter操作。如果原封不动地执行这个执行计划，最终的执行效率是不高的。因为join是一个代价较大的操作，也可能会产生一个较大的数据集。如果我们能将filter下推到join下方，先对DataFrame进行过滤，再join过滤后的较小的结果集，便可以有效缩短执行时间。而Spark SQL的查询优化器正是这样做的。简而言之，逻辑查询计划优化就是一个利用基于关系代数的等价变换，将高成本的操作替换为低成本操作的过程。

得到的优化执行计划在转换成物理执行计划的过程中，还可以根据具体的数据源的特性将过滤条件下推只数据源内。最右侧的物理执行计划中Filter之所以消失不见，就是因为溶入了用于执行最终的读取操作的表扫描节点内。

对于普通开发者而言，查询优化器的意义在于，即便是经验并不丰富的程序员写出的次优的查询，也可以被尽量转换为高效的形式予以执行。

### DataFrame As The New RDD

在Spark 1.3中，DataFrame已经开始替代RDD成为新的数据共享抽象。以下的Spark ML示例搭建了一整套由切词、词频计算、逻辑回归等多个环节组成的机器学习流水线。该流水线的输入、各环节间的数据交换，以及流水线的输出结果，都是以 DataFrame来表示的。

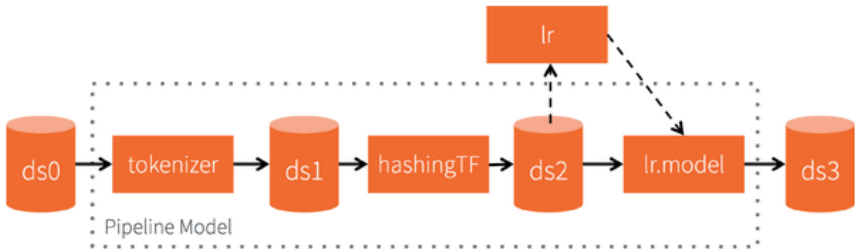


图8：机器学习流水线

相对于RDD，DataFrame有几个特点：

- 1. 包含schema信息，能够进行针对性的优化。
- 2. 对用户有更加友好、更直观的API。
- 3. 与外部数据源API紧密集成，可以用作多种存储格式和存储系统间的数据交换媒介。

作为一个比RDD更加高效的数据共享抽象，DataFrame使得我们可以更加便捷地搭建一体化的大数据流水线。

顶0

踩0

推荐阅读相关主题： 数据分析    连城    结构    sql    数据科学家    机器学习

相关文章    最新报道

Apache Ignite——新一代数据库缓存系统

IBM陈冠诚：如何使用OpenStack、Docker和Spark...

轻松搞定TB级数据，开源GraphLab突破人类图计算"...



搭建高可用的MongoDB集群（上）：MongoDB的配...  
5种方式将机器学习带到Java、Python以及Go等编程...  
SQL on Hadoop的最新进展及7项相关技术分享

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2018, CSDN.NET, All Rights Reserved 