**Storm与Spark：谁才是我们的实时处理利器**

参考：<http://developer.51cto.com/art/201412/460116.htm>

实时商务智能目前已经逐步迈入主流，而Storm与Spark开源项目的支持无疑在其中起到了显著的推动作用。那么问题来了：实时处理到底哪家强？

实时商务智能这一构想早已算不得什么新生事物（早在2006年维基百科中就出现了关于这一概念的页面）。然而尽管人们多年来一直在对此类方案进行探讨，我却发现很多企业实际上尚未就此规划出明确发展思路、甚至没能真正意识到其中蕴含的巨大效益。

为什么会这样？一大原因在于目前市场上的实时商务智能与分析工具仍然非常有限。传统数据仓库环境针对的主要是批量处理流程，这类方案要么延迟极高、要么成本惊人——当然，也可能二者兼具。

然而已经有多款强大而且易于使用的开源平台开始兴起，欲彻底扭转目前的不利局面。其中最值得关注的两大项目分别为Apache Storm与Apache Spark，它们都能为广大潜在用户提供良好的实时处理能力。两套方案都归属于Apache软件基金会，而且除了在功能方面的一部分交集之外、两款工具还各自拥有着独特的特性与市场定位。

**Storm:实时处理领域的Hadoop**

作为一套专门用于事件流处理的分布式计算框架，Storm的诞生可以追溯到当初由BackType公司开发的项目——这家市场营销情报企业于2011年被Twitter所收购。Twitter旋即将该项目转为开源并推向GitHub平台，不过Storm最终还是加入了Apache孵化器计划并于2014年9月正式成为Apache旗下的顶级项目之一。

Storm有时候也被人们称为实时处理领域的Hadoop。Storm项目的说明文档看起来对这种称呼也表示认同：“Storm大大简化了面向庞大规模数据流的处理机制，从而在实时处理领域扮演着Hadoop之于批量处理领域的重要角色。”

为了达成上述目标，Storm在设计思路中充分考虑到大规模可扩展能力、利用一套“故障快速、自动重启”方案为处理提供容错性支持、从而有力地保证了每个元组都能切实得到处理。Storm项目默认为消息采取“至少一次”的处理覆盖保障，但用户也能够根据需要实现“仅为一次”的处理方式。

Storm项目主要利用Clojure编写而成，且既定设计目标在于支持将“流”（例如输入流）与“栓”（即处理与输出模块）结合在一起并构成一套有向无环图（简称DAG）拓扑结构。Storm的拓扑结构运行在集群之上，而Storm调度程序则根据具体拓扑配置将处理任务分发给集群当中的各个工作节点。

大家可以将拓扑结构大致视为MapReduce在Hadoop当中所扮演的角色，只不过Storm的关注重点放在了实时、以流为基础的处理机制身上，因此其拓扑结构默认永远运行或者说直到手动中止。一旦拓扑流程启动，挟带着数据的流就会不断涌入系统并将数据交付给栓（而数据仍将在各栓之间流程继续传递），而这也正是整个计算任务的主要实现方式。随着处理流程的推进，一个或者多个栓会把数据写入至数据库或者文件系统当中，并向另一套外部系统发出消息或者将处理获得的计算结果提供给用户。

Storm生态系统的一大优势在于其拥有丰富的流类型组合，足以从任何类型的来源处获取数据。虽然大家也可以针对某些具备高度特殊性的应用程序编写定制化流，但基本上我们总能从庞大的现有源类型中找到适合需要的方案——从Twitter流API到Apache Kafka再到JMS broker，一切尽皆涵盖于其中。

适配器的存在使其能够轻松与HDFS文件系统进行集成，这意味着Storm可以在必要时与Hadoop间实现互操作。Storm的另一大优势在于它对多语言编程方式的支持能力。尽管Storm本身基于Clojure且运行在JVM之上，其流与栓仍然能够通过几乎所有语言进行编写，其中包括那些能够充分发挥在标准输入/输出基础上使用JSON、并由此实现组件间通信协议优势的非JVM语言。

总体而言，Storm是一套极具可扩展能力、快速惊人且具备容错能力的开源分布计算系统，其高度专注于流处理领域。Storm在事件处理与增量计算方面表现突出，能够以实时方式根据不断变化的参数对数据流进行处理。尽管Storm同时提供原语以实现通用性分布RPC并在理论上能够被用于任何分布式计算任务的组成部分，但其最为根本的优势仍然表现在事件流处理方面。

**Spark:适用于一切的分布式处理方案**

作为另一个专门面向实时分布式计算任务的项目，Spark最初由加州大学伯克利分校的APMLab实验室所打造，而后又加入到Apache孵化器项目并最终于2014年2月成为其中的顶尖项目之一。与Storm类似，Spark也支持面向流的处理机制，不过这是一套更具泛用性的分布式计算平台。

有鉴于此，我们不妨将Spark视为Hadoop当中一套足以取代MapReduce的潜在备选方案——二者的区别在于，Spark能够运行在现有Hadoop集群之上，但需要依赖于YARN对于资源的调度能力。除了Hadoop YARN之外，Spark还能够以Mesos为基础实现同样的资源调度或者利用自身内置调度程度作为独立集群运行。值得注意的是，如果不将Spark与Hadoop配合使用，那么运行在集群之上时某些网络/分布式文件系统（包括NFS、AFS等）仍然必要，这样每个节点才能够切实访问底层数据。

Spark项目由Scala编写而成，而且与Storm一样都支持多语言编程——不过Spark所提供的特殊API只支持Scala、Java以及Python。Spark并不具备“流”这样的特殊抽象机制，但却拥有能够与存储在多种不同数据源内的数据实现协作的适配器——具体包括HDFS文件、Cassandra、HBase以及S3。

Spark项目的最大亮点在于其支持多处理模式以及支持库。没错，Spark当然支持流模式，但这种支持能力仅源自多个Spark模块之一，其预设模块除了流处理之外还支持SQL访问、图形操作以及机器学习等。

Spark还提供一套极为便利的交互shell，允许用户利用Scala或者Python API以实时方式快速建立起原型及探索性数据分析机制。在使用这套交互shell时，大家会很快发现Spark与Storm之间的另一大差异所在：Spark明显表现出一种偏“功能”的取向，在这里大部分API使用都是由面向原始操作的连续性方法调用来实现的——这与Storm遵循的模式完全不同，后者更倾向于通过创建类与实现接口来完成此类任务。先不论两种方案孰优孰劣，单单是风格的巨大差异已经足以帮助大家决定哪款系统更适合自己的需求了。

与Storm类似，Spark在设计当中同样高度重视大规模可扩展能力，而且Spark团队目前已经拥有一份大型用户文档、其中列出的系统方案都运行着包含成千上万个节点的生产性集群。除此之外，Spark还在最近的2014年Daytona GraySort竞赛当中获得了优胜，成为目前承载100TB级别数据工作负载的最佳选择。Spark团队还保留了多份文档，其中记录着Spark ETL如何负责数PB级别生产工作负载的运营。

Spark是一套快速出色、可扩展能力惊人且极具灵活性的开源分布式计算平台，与Hadoop以及Mesos相兼容并且支持多川计算模式，其中包括流、以图形为核心的操作、SQL访问外加分布式机器学习等。Spark的实际扩展记录令人满意，而且与Storm一样堪称构建实时分析与商务智能系统的卓越平台。

**您会如何选择**

那么大家又该如何在Storm与Spark之间做出选择呢？

如果大家的需求主要集中在流处理与CEP（即复杂事件处理）式处理层面，而且需要从零开始为项目构建一套目标明确的集群设施，那么我个人更倾向于选择Storm——特别是在现有Storm流机制能够确切满足大家集成需求的情况下。这一结论并不属于硬性要求或者强制规则，但上述因素的存在确实更适合由Storm出面打理。

在另一方面，如果大家打算使用现有Hadoop或者Mesos集群，而且/或者既定流程需要涉及与图形处理、SQL访问或者批量处理相关的其它实质性要求，那么Spark则值得加以优先考虑。

另一个需要考量的因素是两套系统对于多语言的支持能力，举例来说，如果大家需要使用由R语言或者其它Spark无法原生支持的语言所编写的代码，那么Storm无疑在语言支持宽泛性方面占据优势。同理可知，如果大家必须利用交互式shell通过API调用实现数据探索，那么Spark也能带来Storm所不具备的优秀能力。

最后，大家可能希望在做出决定前再对两套平台进行一番详尽分析。我建议大家先利用这两套平台各自建立一个小规模概念验证项目——而后运行自己的基准工作负载，借此在最终选择前亲身体验二者的工作负载处理能力是否与预期相一致。

当然，大家也不一定非要从二者之中选择其一。根据各位工作负载、基础设施以及具体要求的不同，我们可能会找出一种将Storm与Spark加以结合的理想方案——其它同样可能发挥作用的工具还包括Kafka、Hadoop以及Flume等等。而这正是开源机制的最大亮点所在。

无论大家选择哪一套方案，这些工具的存在都切实表明实时商务智能市场的游戏规则已经发生了变化。曾经只能为少数精英所掌握的强大选项如今已经进入寻常百姓家——或者说，至少适用于多数中等规模或者大型企业。不要浪费资源，充分享受由此带来的便利吧。

英文：<http://www.infoworld.com/article/2854894/application-development/spark-and-storm-for-real-time-computation.html>