Apache Flink改进及其在阿里巴巴搜索中的应用

摘要： 阿里是世界上最大的电子商务零售商，其2015年的年销售额就超过了eBay和Amazon的总和，达3940亿。Alibaba Search，个性化搜索和推荐平台，既是顾客的关键入口，也承担了大部分的在线收益。因此，阿里搜索基础设施团队一直在努力改进产品。

本文整理自阿里搜索基础设施团队研究员蒋晓伟在Flink Forward 2016大会上的演讲，原始演讲视频可以在这里查看。

以下为演讲整理

阿里是世界上最大的电子商务零售商，其2015年的年销售额就超过了eBay和Amazon的总和，达3940亿。Alibaba Search，个性化搜索和推荐平台，既是顾客的关键入口，也承担了大部分的在线收益。因此，阿里搜索基础设施团队一直在努力改进产品。

对于电子商务网站上的搜索引擎，到底什么最重要？必然是实时地为每一位用户提供最相关和准确的搜索结果。以阿里巴巴的规模，这是个棘手的问题，很多技术也根本无法满足需求。Apache Flink就是这样一项技术，而阿里现在通过基于Flink的Blink系统，来支撑它的搜索基础设施的核心，以向终端用户提供相关、精准的信息。本文讲解Flink在阿里搜索中承担的任务，并概括选择Flink的原因。

同时也会提到Blink与Flink的区别，及如何通过data Aritisans和Flink社区将这些改动反馈。倘若开源项目接受了这些改动，Alibaba Search会积极地把系统从Blink转移到原生Apache Flink。

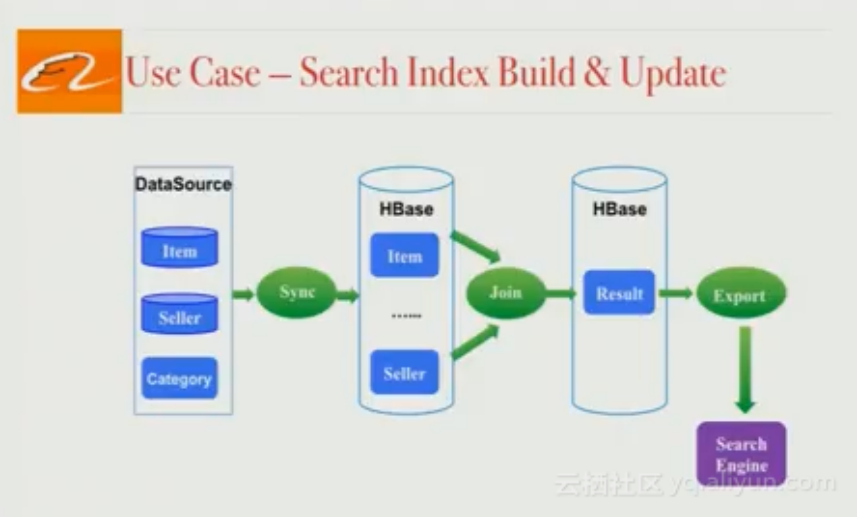
第一部分：阿里搜索中的Flink

一、 文档的建立

向用户提供一流搜索引擎的第一步是建立可供搜索的文档，对于阿里，文档由数百万条商品列表及相关的产品数据构成。搜索文档的建立并不容易，因为产品数据被保存在很多地方，搜索团队要整合所有相关信息才能建立完整的搜索文档。总的来说，这需要三个步骤：

1. 从完全不同的来源（比如MySQL、分布式文件系统）把所有产品数据同步到一个HBase集群。
2. 基于业务逻辑join不同的表格来建立最终可搜索文档，不放称这个HBase表为“结果”表。

3. 将这个HBase表通过一个文件或一系列更新导出。

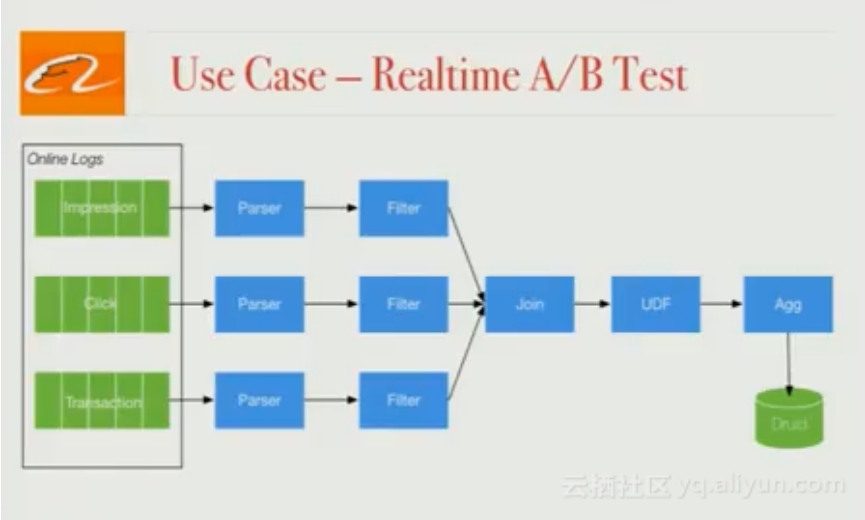


这三个阶段实际上运行在两条不同的管道中，遵循着“lambda架构”，即一条full-build管道和一条incremental build管道。

- 在full-build管道会处理所有的数据源，通常是批处理任务。
- 在incremental build管道主要处理批处理完成后的更新。比如商品价格、详情、库存的变动，它们必须尽快反映在搜索结果中，通常做流计算。

二、 对搜索算法的实时A/B测试

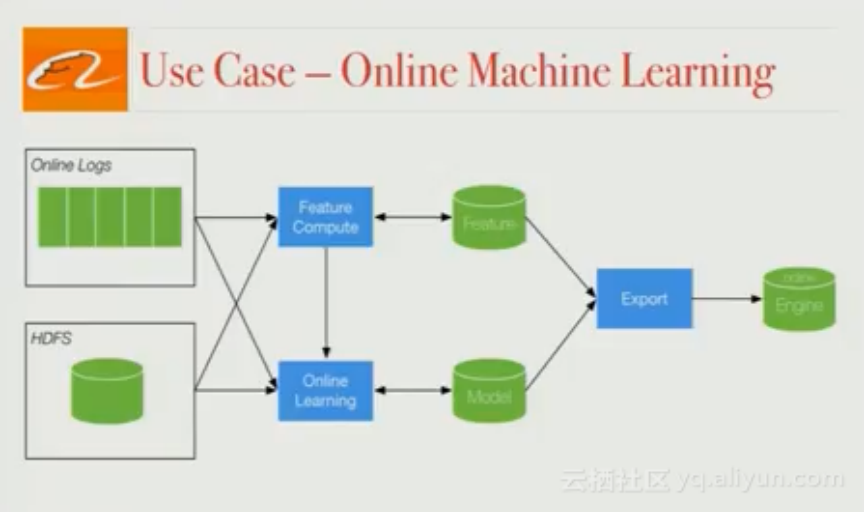
工程师会定期测试不同的搜索算法，他们需要尽快对性能进行评估。现在，这种评估每天进行一次，但将来希望让分析实时地进行。因此这里使用Blink来构建实时的A/B测试框架。在线日志（展示次数，点击数，交易数）由分析程序收集和处理，然后根据业务逻辑进行过滤和join。接下来，数据被聚合进Druid，在Druid中，可以通过请求来实现复杂的OLAP分析，并看到不同的算法运行的效果。



三、 在线机器学习

这个方面有不少应用，首先谈谈实时特征更新。阿里搜索排名中，产品CTR，库存、点击数等数据被作为特征值。这些数据会随时间变化，如果总能拿到最新的数据，就能为用户提供相关性更好的搜索排名。Flink管道提供了在线特征更新的功能，大大提高了转化率。

另外，一年中会有几次大型促销（如11.11），这时用户的行为也会发生巨变。交易量会剧增，往往比往日高出好几倍。之前训练好的模型在这种场景下就失效了，所以需要依靠日志和Flink流处理任务来进行在线机器学习，根据实时数据建立模型，大幅提升了在这些不寻常，又十分重要的日子里的转化率。



第二部分：选择能够解决问题的框架

选择Flink来驱动搜索引擎架构有四个方面原因：

- 敏捷性：希望用一个代码库来维护整个搜索架构，同时期望一个高级的API来表达业务逻辑。
- 一致性：卖家或产品数据库的变更必须反映在最终的搜索结果上，因此需要“at-least-once”语义（Flink在公司其他应用场景中也可能需要“exactly-once”语义）。
- 低延迟：库存的变更必须要立刻反映在搜索结果中；毕竟不希望排名靠前的是售罄的产品。
- 开销：阿里有大量数据要处理。就规模而言，效率的提升会大幅节省开销，因此需要一个足够高效的框架来解决高流量问题。

一般来说，对于统一的批处理和流处理有两种做法。第一种方是以批处理为出发点并在此基础上建立流处理。可惜这很难满足苛刻的延迟要求，因为通过微型批处理来模拟流处理的方法会产生固定的系统开销，且延迟越低，开销越大。对阿里的规模而言，每次微型批处理需要解决1000秒的任务量。过程中链接被重新建立，状态被重新加载。因此微型批处理的方法的代价会过高。

Flink则采用了另一种方式。它以流处理为出发点，并在此基础上建立了批处理的解决方案，也就是把批处理当成了流的一种特殊情况。而且通过这种方式，用户也可以对批处理进行优化。

第三部分：Blink是什么

Blink是Flink的一个分支，通过维护它来满足阿里的一些特殊需求。当前，Blink在几个不同的集群上运行，而每个集群大约有1000台机器。因此，大规模运行的性能至关重要。

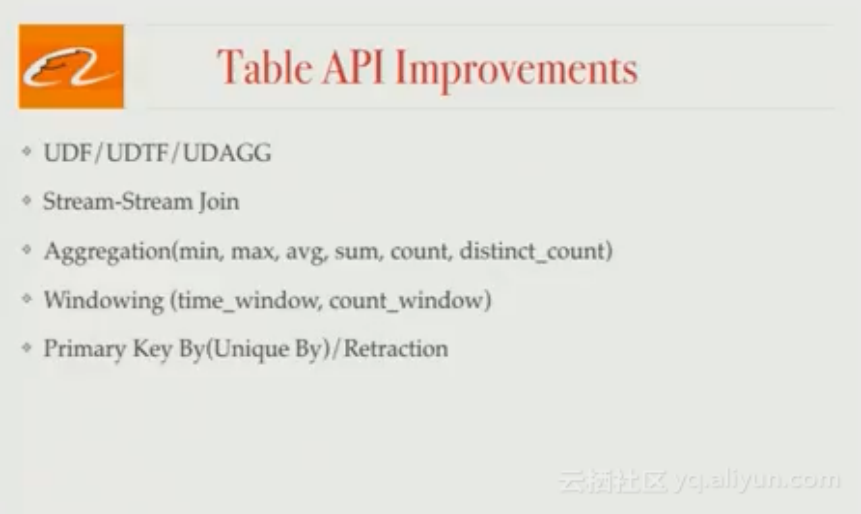
Blink的改进主要体现在两方面：

- 把Table API做的更完整，这样就能用相同的SQL操作批处理和流处理
- 提高了YARN模式的健壮性，并使得它完美兼容Flink的API及更广阔的生态系统

Table API

首先添加了对可定义函数的支持，这样用户就能更方便地把业务逻辑写入Flink。这里还添加了流对流连接，这并不容易，但Flink对状态的支持简化了相关工作。另外还添加了一些不同的aggregations，其中最有趣的当属distinct_count和windowing支持。

（编者按：FLIP-11包含了一系列上文提及的Table API和对SQL的改进，供有兴趣的读者参考）

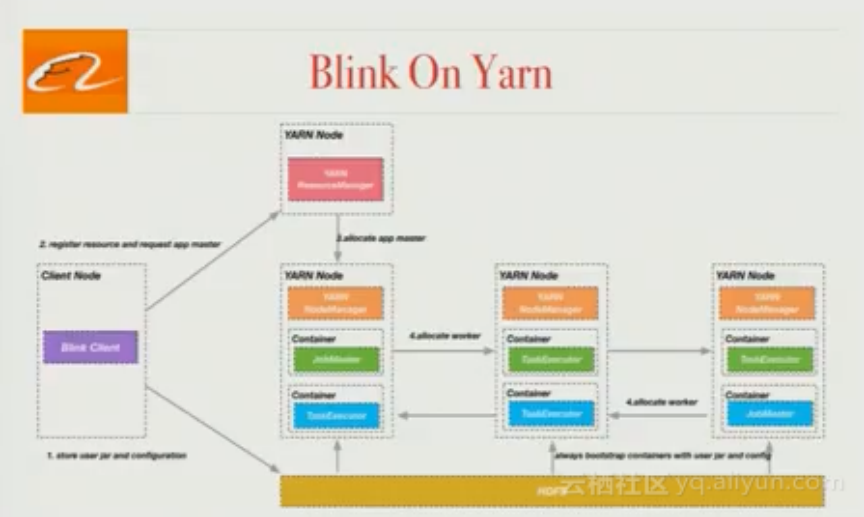


接下来，将会通过四个方面来谈runtime中的改进。

Blink on Yarn

当项目开始时，Flink支持两种集群模式：standalone及Yarn。在Yarn模式中，一个job不能动态请求和释放资源，而是要提前获得所有所需资源。不同的工作共享JVM进程，从而资源利用率优先于隔离性。

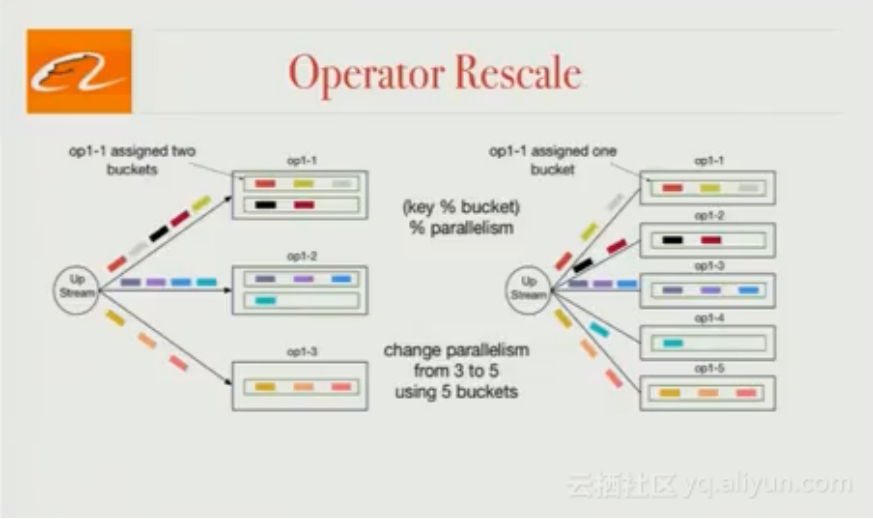
Blink引进了一种新的架构，每个job都有一个JobMaster来请求和释放它所需的资源，不同的job不能在同一个Java进程中运行。这就在job和task之间实现了隔离。阿里的团队正与Flink社区合作将这部分工作合并到开源项目中。这些改进可参考FLIP-6（除了Yarn以外还涉及其他的集群管理器）。



Operator Rescale

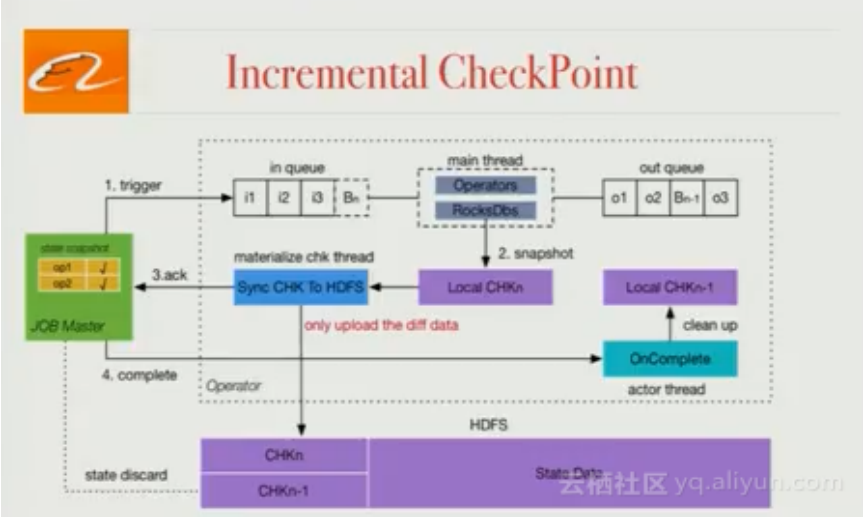
在生产中，客户端需要在保持状态的同时改变operators的并行模式。刚开始做Blink时，Flink还不支持这一点。而Blink引进了“bucket”的概念作为管理状态的单元。bucket的数量比task多得多，且每个task与多个桶绑定。当parallelism发生变化时，会重新给任务分配“bucket”。通过这种方式就能在维持状态的同时改变operators的parallelism了。

（编者按：Flink社区已经在Flink 1.2中解决了这一问题，该特性存在于主分支的最新版本中。Flink的“key groups”概念和这里所说的“bucket”基本等价的，但采用了不同的数据结构来实现。详情参考FLINK-3755 in Jira.）



增量Checkpointing

Flink中，Checkpointing的建立有两个阶段：在本地获取状态快照，并把它放到HDFS（或者其他的存储系统）上，这样整个状态快照就能和每个快照一起存储在HDFS中了。可惜这里的状态太大了，这种方法行不通，因此Blink只把被修改的状态存储在HDFS中。这种方式大幅度提高了Checkpointing建立的效率，使得在生产中使用大的状态成为可能。



异步I/O

在生产环境中，性能瓶颈往往在于访问外部存储，如HBase。为了解决这一问题，这里引入了异步I/O技术。这部分工作也被回馈给了开源社区，细节见FLIP-12。

（编者按：因为FLIP-12内容较多，Data Artisans将在近期发表单独介绍它的文章。在此一笔带过，如果想了解更多请参考FLIP writeup。另外，本文发表时这部分功能已加入Flink。）

原文链接：Blink: How Alibaba Uses Apache Flink （作者/蒋晓伟）