Apache Flink 1.5.0 正式发布，多项重要更新

Apache Flink 1.5.0 于昨天晚上正式发布了。在过去五个月的时间里，Flink 社区共解决了超过 780 个 issues。完整的 changelog 看这里: https://issues.apache.org/jira/secure/ReleaseNote.jspa?version=12341764&projectId=12315522。

Flink 1.5.0 是 1.x.y 版本线上的第六个主要发行版。按照惯例，使用 @Public 注解标记的 API 和之前 1.x.y 版本是兼容的。强烈推荐所有用户下载这个版本去体验。

Flink 1.5 - Streaming Evolved

我们相信，流处理领域和 Apache Flink 一起正在进行另一次重大飞跃。流处理已经不仅仅是代表更快速的分析数据，更是一种构建快速连续数据管道的原则性方法。流处理正在成为构建数据驱动型和数据密集型应用程序的典范 - 它汇集了数据处理逻辑和应用程序/业务逻辑。

为了帮助用户认识到这一变化的潜力，我们在此发布中花费了大量精力修改 Flink 的一些基本组件。我们希望 Flink 对于进行数据工程/数据处理的用户以及构建数据/事件驱动应用程序的用户（当然还有那些在应用程序中将这两方面结合的用户）感到非常容易上手。这是一个持续的旅程，这个版本是第一步，主要有以下更新：

- 我们重新设计并重新实现了 Flink 的大部分流程模型。详细记录参见 FLIP-6：https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=65147077。尽管还没有完成所有的事情，但 Flink 1.5 在 Kubernetes 部署更简单自然，并为所有外部通信切换到HTTP / REST。同时，Flink 1.5 简化了常见集群管理器（YARN，Mesos）上的部署并具有动态资源分配功能。

- 流广播状态(FLINK-4940)将广播流（比如上下文数据，机器学习模型，规则/模式，触发器。。。）与其他可能保持键状态的流相连接，如特征向量，状态机等。而在 Flink 1.5 之前，这样的用例不容易构建。
- 为了改善对严格延迟限制的实时应用程序的支持，我们对 Flink 的网络堆栈进行了重大改进（FLINK-7315）。Flink 1.5 实现了更低的延迟，同时保持了高吞吐量。另外，我们改进了反压（backpressure）下的检查点稳定性。
- 流式 SQL 越来越被认为是一种简单而强大的方式来进行流式分析，构建数据管道，进行特征工程或增量更新应用程序。我们添加了用于流式 SQL 查询的 SQL CLI（FLIP-24），以使该功能更易于使用。

新功能和改进

重写 Flink 的部署和处理模型

重写 Flink 的部署和处理模型（内部称为FLIP-6）已经进行了一年多的时间，并且是 Flink 社区的一项实质性努力。来自多个组织的许多贡献者（例如data Artisans，阿里巴巴和 Dell EMC）合作设计并实现这些特性，这是该项目启动以来 Flink 核心组件的最重大改进。

简而言之，这些改进增加了对 YARN 和 Mesos 调度程序的动态资源分配和动态释放资源的支持，以提高资源利用率，故障恢复以及动态扩展。此外，像 Kubernetes 这样的容器管理基础设施的部署已经简化了，现在所有对 JobManager 的请求都是通过 REST 完成的。 这包括作业提交，取消，请求作业状态，获取保存点等。

这项工作也为 Flink 与 Kubernetes 的未来改进奠定了基础。 在稍后的版本中，可以将作业 docker 化，并作为容器部署的一部分以自然的方式部署它们，比如不需要先启动 Flink 集群。此外，这项工作是支持能够自动调整并行度应用程序的一大步。

请注意，Flink 的编程 API 不受这些改进的影响。

Broadcast State

对广播状态的支持，即在所有并行实例中复制一个函数的状态，一直是一个频繁请求的功能。广播状态的典型用例涉及两个流，一个是服务规则，模式的控制或配置流，另一个是常规的数据流。常规流的处理由控制流的消息配置。通过将规则或模式广播到函数的所有并行实例，可以应用于常规流的所有事件。

当然，广播状态可以进行 checkpoint 和恢复，就像 Flink 中的任何其他状态一样具有 exactly-once 状态一致性保证。

Flink 网络栈的提升

分布式流式应用程序的性能在很大程度上取决于通过网络连接将事件从一个算子转移到另一个算子的组件。在流处理环境中，延迟和吞吐量两个性能指标非常重要。

Flink 1.5 版本中，社区致力于在两个方面改善 Flink 的网络堆栈：基于信用（Credit-based）的流

量控制以及改善传输延迟。基于信用的流量控制将数据量“减少”降到最低，同时保持高吞吐量。这显着减少了在反压情况下完成检查点的时间。 此外，Flink
现在能够在不降低吞吐量的情况下实现更低的延迟。

任务本地状态恢复（Task-Local State Recovery）

Flink 的检查点机制将应用程序状态的副本写入远程持久存储器，并在发生故障时将其加载回去。这种机制确保应用程序失败时状态不会丢失。
但是，如果发生故障，可能需要一段时间才能从远程存储加载状态以恢复应用程序。

Flink 社区正在不断努力提高检查点和恢复效率。 以前版本的突出特点是异步和增量检查点。在此版本中，我们提高了故障恢复的效率。

任务本地状态恢复利用了作业通常由一个算子、TaskManager
或机器崩溃导致失败的事实。在将算子的状态写入远程存储器时，Flink
现在也可以在每台机器的本地磁盘上保留一份副本。 在故障恢复的情况下，调度程序会尝试将任务重新安排到其以前运行的机器上，并从本地磁盘而不是远程存储加载状态，从而加快恢复速度。

扩展对 SQL 和表API Join 的支持

在1.5.0版本中，Flink 添加了对窗口化 outer equi-joins 的支持。
如下所示的查询允许在有限的时间范围内将事件时间和处理时间进行连接：

```
SELECT d.rideId, d.departureTime, a.arrivalTime
FROM Departures d LEFT OUTER JOIN Arrivals a
  ON d.rideId = a.rideId
  AND a.arrivalTime BETWEEN
     d.deptureTime AND d.departureTime + '2' HOURS
```

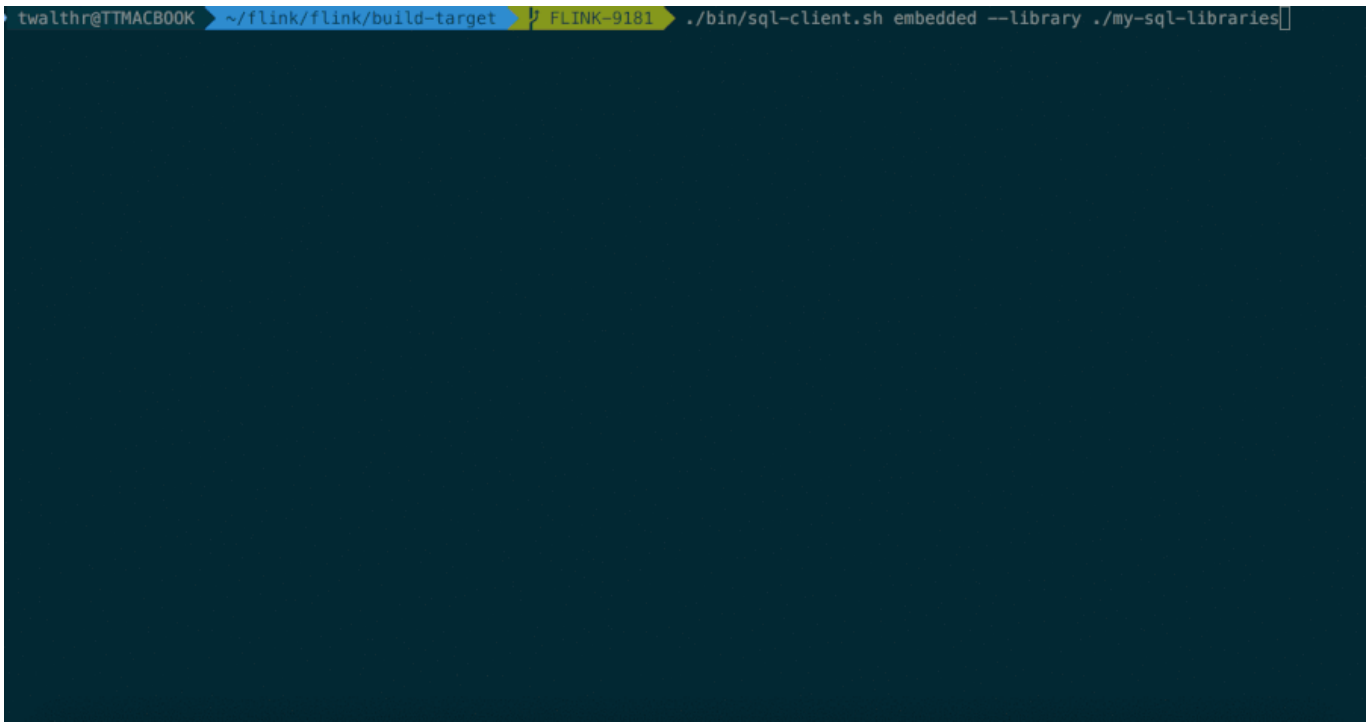
对于不应在有限时间间隔内连接两个流式表的情况，Flink SQL
现在还支持非窗口式内连接。这可以实现全历史匹配，这在许多标准 SQL 语句中很常见:

```
SELECT u.name, u.address, o.productId, o.amount
FROM Users u JOIN Orders o
  ON u.userId = o.userId
```

SQL CLI 客户端

几个月前，社区开始努力添加一项服务来执行流和批处理 SQL 的查询（FLIP-24）。新的 SQL CLI

客户端是这项工作的第一步，并提供了一个 SQL shell 来对数据流的进行探索性查询。
下面的动画显示了此功能的预览：



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

其他特性和改进

- OpenStack 提供了用于在资源池上创建公共和私有云的软件。Flink 现在支持 OpenStack 的类 S3 文件系统 Swift，用于保存检查点和保存点。Swift 可以在没有 Hadoop 依赖的情况下使用。
- 改进从连接器读取或向连接器写入 JSON 消息。现在可以通过解析一个标准的 JSON 模式来配置序列化器和反序列化器。SQL CLI 客户端能够读取来自 Kafka 的 JSON 记录。
- 应用程序可以在无需手动触发保存点的情况下进行伸缩。实际上，Flink 仍然会保存一个保存点，然后停止应用程序并重新调整并行度。
- 改进了 watermark 和延迟的度量标准，Flink 现在捕获所有操作器（包括数据源在内）的最小化 watermark。此外，为了更好地与常用指标系统集成，延迟度量指标进行了重新设计。
- FileInputFormat（和其他多种输入格式）现在支持从多个路径读取文件。
- BucketingSink 支持自定义扩展规范。
- CassandraOutputFormat 可用于发送 Row 对象。
- Kinesis 消费者客户端允许更大程度的定制化。

当然还有其他很多重要的更新，这里就不一一列举了，详情请参见官方网站。

本文链接: 【】（ ）