比较Kafka Stream和Flink

流式计算

一般流式计算会与批量计算相比较。在流式计算模型中，输入是持续的，可以认为在时间上是无界的，也就意味着，永远拿不到全量数据去做计算。同时，计算结果是持续输出的，也即计算结果在时间上也是无界的。流式计算一般对实时性要求较高，同时一般是先定义目标计算，然后数据到来之后将计算逻辑应用于数据。同时为了提高计算效率，往往尽可能采用增量计算代替全量计算。

**Apache Flink**

—高性能集群数据处理框架

* 中等集群上的每秒千万级事件的吞吐量
* 毫秒级输出延迟
* 保证应用状态正好消费一次，及端至端正好一次的发送（例：Kafka -> Flink -> HDFS ）
* 支持指定事件时间来源以精确地处理乱序事件
* 高可用
* 流批一体的SQL查询API

**Kafka Streams API**

—基于Kafka的流式处理类库

* 内置于你的应用
* 充分利用了[Kafka的分区机制](http://www.jasongj.com/2015/03/10/KafkaColumn1/#Topic-amp-Partition)和[Consumer的Rebalance机制](http://www.jasongj.com/2015/08/09/KafkaColumn4/#High-Level-Consumer-Rebalance)

|  |  |  |
| --- | --- | --- |
|  | **Apache Flink** | **Kafka Streams API** |
| 计算任务的部署 | 框架自行处理应用的部署。可能是 standalone Flink clusters,或使用YARN, Mesos, 容器 (Docker, Kubernetes) | 用户自行管理应用部署 |
| 生命周期 | 在 Flink cluster中以job的形式管理用户的处理逻辑 | 在用户的应用里面执行处理逻辑 |
| 典型的使用者 | 数据处理基础设施team或 BI team | 业务开发team |
| 协调 | Flink Master (JobManager) | 在 Kafka cluster 中协调负载均衡和容错. |
| 流式数据源 | Kafka, 文件系统, 或其他MQ | 只使用Kafka接收事件 |
| 接收处理结果 | Kafka, 其他MQs, 文件系统, 分析数据库（analytical database）, KV数据库, 成为另一个Flink Job的应用状态, 及其他外部系统 | Kafka, 应用状态, 操作数据库（operational database） 及其他外部系统 |
| 数据源是否有尽 | 有尽+无尽 | 无尽 |
| 语意保证 | Flink state内部Exactly once; 使用的数据源和接收方端到端正好一次 (e.g., Kafka to Flink to HDFS); 当使用Kafka 为接收方时，至少一次（未来会支持正好一次） | 端到端正好一次 |

总而言之，由于两者体系结构的差异，我们将它们视为互补的系统。 **Streams API**使流处理可以作为应用程序编程模型进行访问，由于以下原因，作为微服务构建的应用程序可以从**Kafka**的核心竞争力（性能，可伸缩性，安全性，可靠性以及端到端的一次准确使用）中受益。它与**Kafka**中的核心抽象紧密集成。另一方面，**Flink**非常适合部署在现有集群中的应用程序，并受益于吞吐量，延迟，事件时间语义，保存点和操作功能，一次保证应用程序状态，端到端完全**-**一次担保（今天与**Kafka**作为接收器一起使用时除外）和批处理。

参考

**<https://www.confluent.io/blog/apache-flink-apache-kafka-streams-comparison-guideline-users/>**

**<http://www.jasongj.com/kafka/kafka_stream/>**