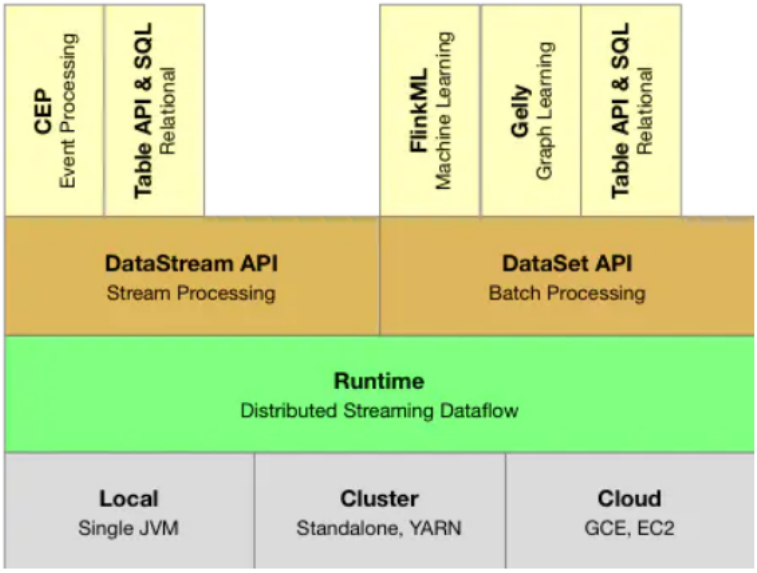
1、Flink简介

1.1 Flink架构

> Apache Flink is a framework and distributed processing engine for stateful computations over *unbounded and bounded* data streams. Flink has been designed to run in *all common cluster environments*, perform computations at *in-memory speed* and at *any scale*

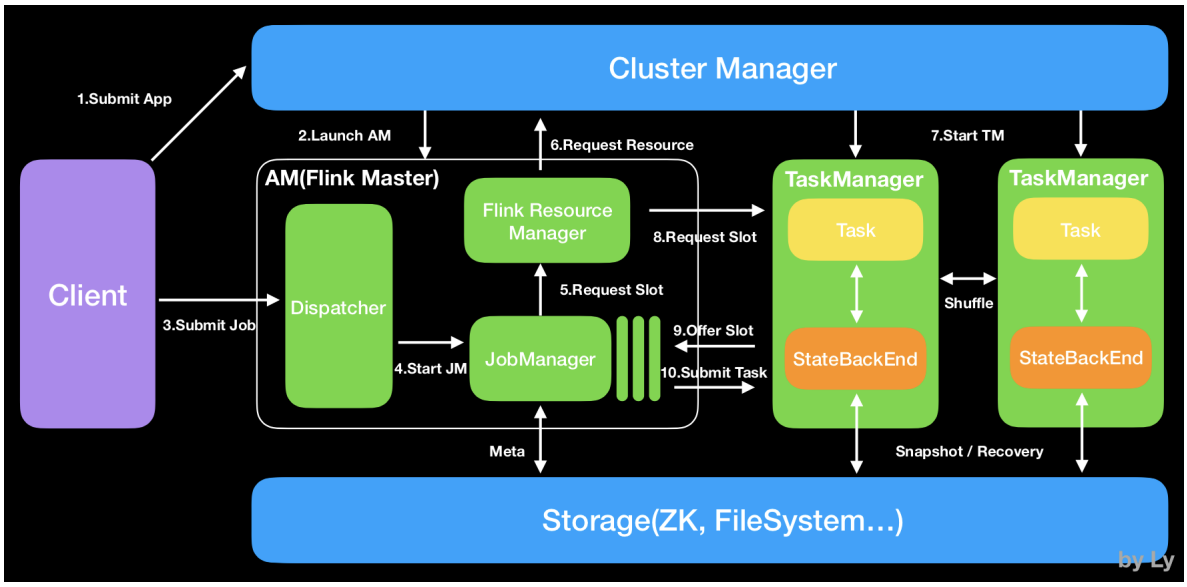
Apache Flink 是一个**分布式处理引擎框架**，用于对无界和有界数据流进行**有状态计算。**Flink 被设计在所有常见的集群环境中运行，以**内存执行速度**和任意规模来执行计算。



上图是Flink的整体架构。Flink 是可以运行在多种不同的环境中的，例如，它可以直接运行在本地的JVM中，从而提供调试的能力。它也可以运行在 YARN 或者 K8S 这种资源管理系统上面，也可以在各种云环境中执行。

Runtime是Flink 的核心，Runtime层针对不同的执行环境提供了一套统一的分布式作业执行引擎。在Runtime层之上，Flink提供了**DataStream** 和 **DataSet** 两套 API，分别用来编写流作业与批作业；在上还有更高级的API来简化特定作业的编写，如基于流处理的CEP（复杂事件处理库）、SQL&Table库和基于批处理的FlinkML（机器学习库）等、Gelly（图处理库）等。

Runtime层架构：



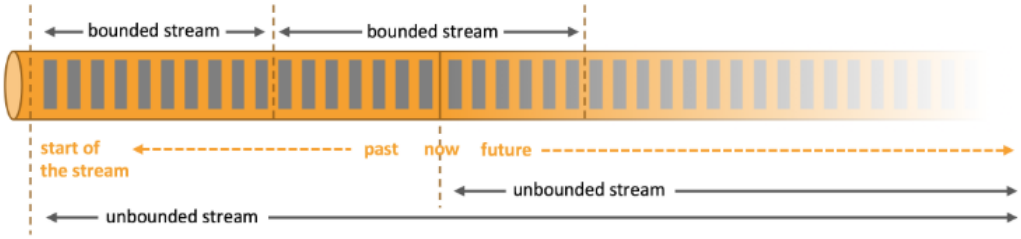
Runtime 层采用了标准的 master-slave 架构，master 部分又包含了三个组件，即 Dispatcher、ResourceManager 和 JobManager。其中，Dispatcher 负责接收用户提供的作业；ResourceManager 负责资源的管理；JobManager 负责管理作业的执行。右侧的两个 TaskExecutor 则是 Slave，负责提供具体的资源并实际执行作业。

1.2 Flink关键特性

流式处理：

批处理：批处理的特点是有界、持久、大量，非常适合需要访问全套记录才能完成的计算工作，一般用于**离线统计。**

流处理：流处理的特点是无界、实时，无需针对整个数据集执行操作，而是对通过系统传输的每个数据项执行操作，一般用于**实时统计。**



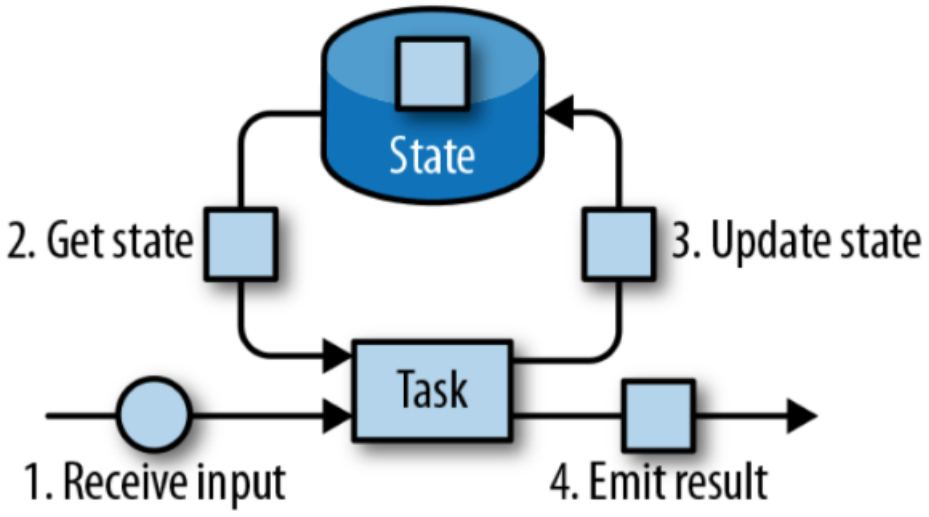
在Flink中，离线数据被看作是有界的数据流，实时数据被看作是没有界限的流，Flink使用灵活的窗口机制来拆分数据流，将一个无界限的数据流拆分为有限大小的“buckets”桶。每个窗口都会绑定一个触发器（Trigger）和一个执行函数。Flink中的几种窗口：

- 时间窗口：时间窗口按时间对流元素进行分组。例如，一分钟的滚动时间窗口收集一分钟的元素，并在一分钟过后对窗口中的所有元素应用一个函数。（怎么定义“时间”？）
- 计数窗口：按照元素的个数来对数据流进行拆分，列如：含有100个元素的滚动计数窗口收集100个元素，并在收到第100个元素时对数据流进行计算。
- 会话窗口：会话窗口本身是没有起止时间，它是针对于进入的数据创建的窗口。例如统计用户的在线时长信息，用户会定时上报相关数据，从用户首次上报开始创建窗口，用户定期产生打点数据会进入该窗口，如果5分钟没有收到用户的数据则判断该用户退出，即触发该用户的在线时长计算。
- 自定义窗口：通过Flink提供的API，可以自定义一些窗口来实现业务逻辑。

状态管理：

状态：

可以认为state是一个本地变量或是一个实例变量。

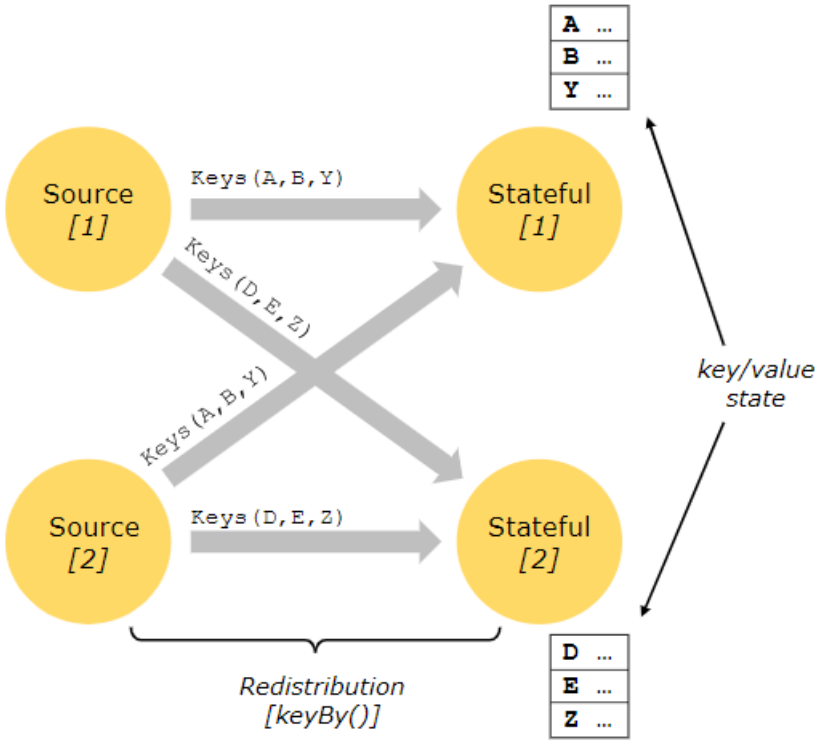


一个task接收一些输入数据。当处理数据时，task会访问state，并根据输入数据和state的信息更新它的state。一个简单的例子如：一个task持续计算迄今它接收到了多少条记录。当task接收到一个新的记录时，它会访问state获取当前的count数，增加count，更新state，并释放新的count作为结果输出。

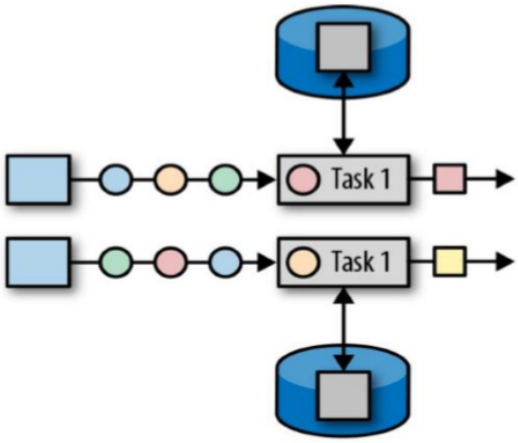
状态类型：

键控状态（Keyed State）：

根据输入数据流中的key来维护和访问。



算子状态（Operator state）：

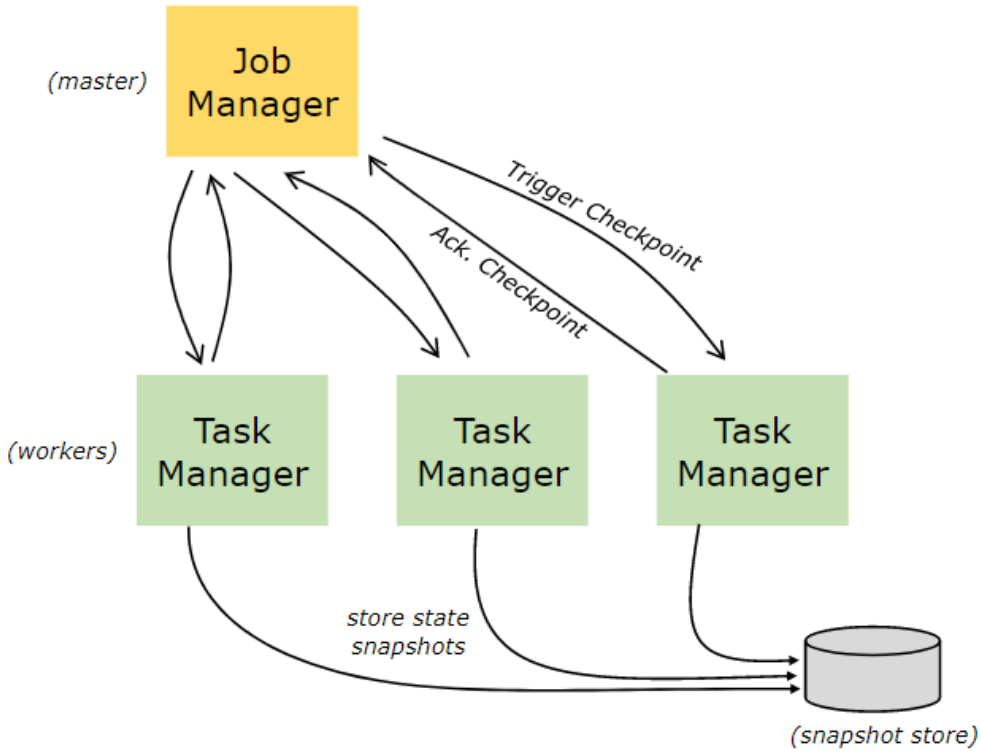


算子状态对于同一任务是共享的

算子状态不能由相同或不同算子的另一个任务访问

状态后端（State Backends）：

状态后端负责状态的存储，访问及维护。下图为状态后端存储检查点（checkpoint）的流程。



---

exactly-once 语义保证

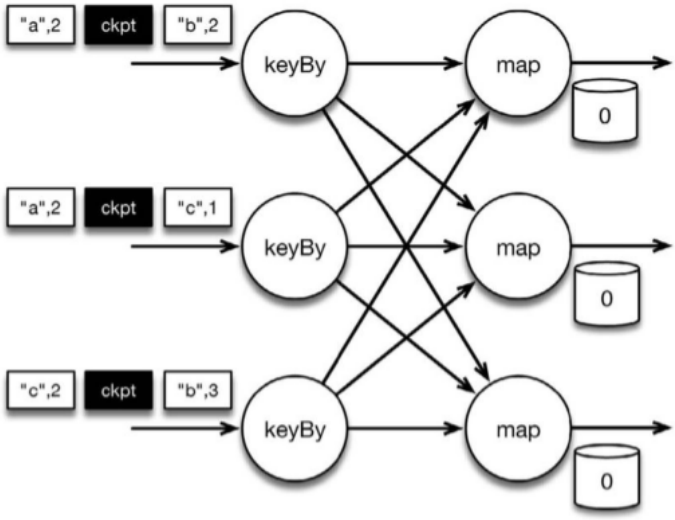
什么是exactly-once：

> 指在存在各种故障的情况下，流应用程序中的所有算子都会保证每个事件都会被计算一次。

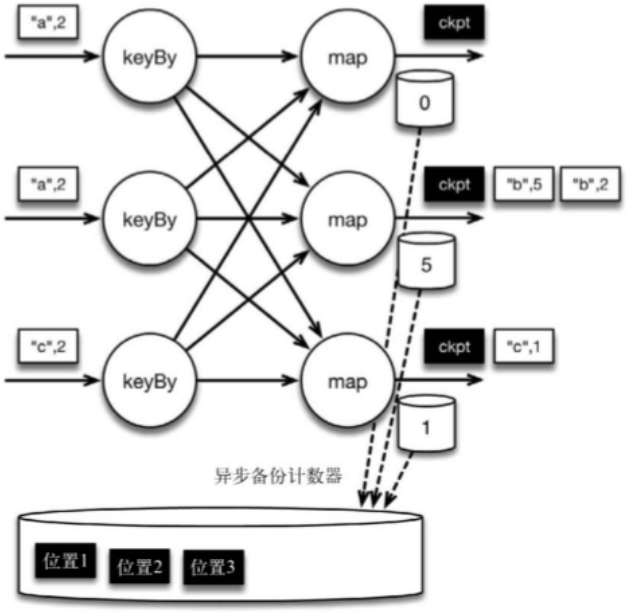
Flink如何确保exactly-once:

Flink采用**基于分布式快照的算法**，利用其定义的检查点（checkpoint）来保证exactly-once。

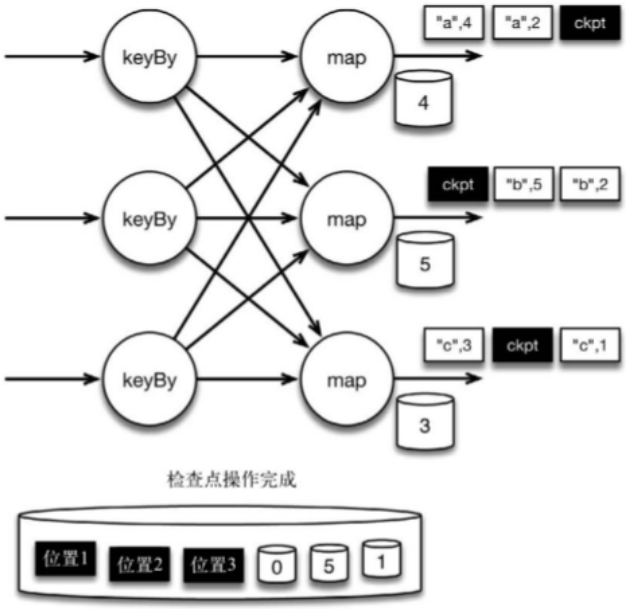
检查点（checkpoint）是在某个时间状态下所有任务的一份快照，在这个时间状态下，所有的任务都恰好完成处理一个相同的输入数据。流应用程序的所有算子状态都会定期的做checkpoint，当出现故障时，每个算子的状态都会进行回滚。



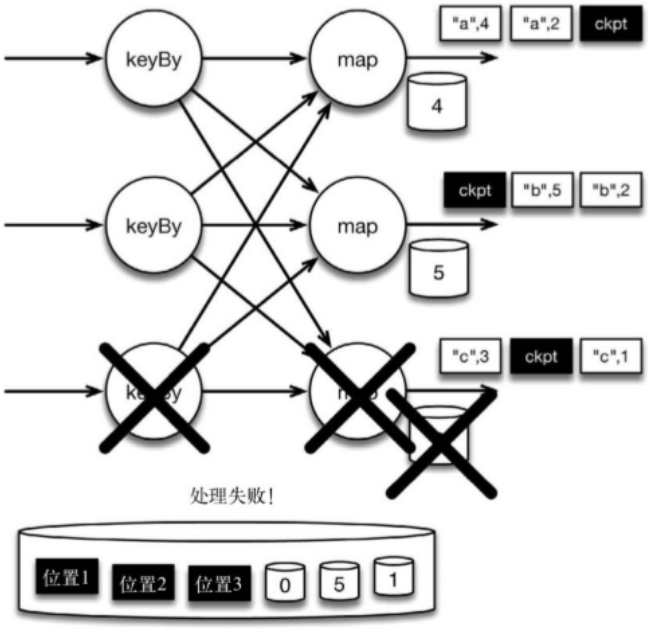
初始状态：a、b、c 三组的初始计数状态都是 0，ckpt 表示检查点分割线，位于ckpt之前的数据在先进行处理。



当 map 算子处理完前 3 条数据并收到检查点分界线时，它们会将状态以异步的方式写入持久化存储，当 map 算子的状态备份和检查点分界线的位置备份被确认之后，该检查点操作就可以被标记为完成。



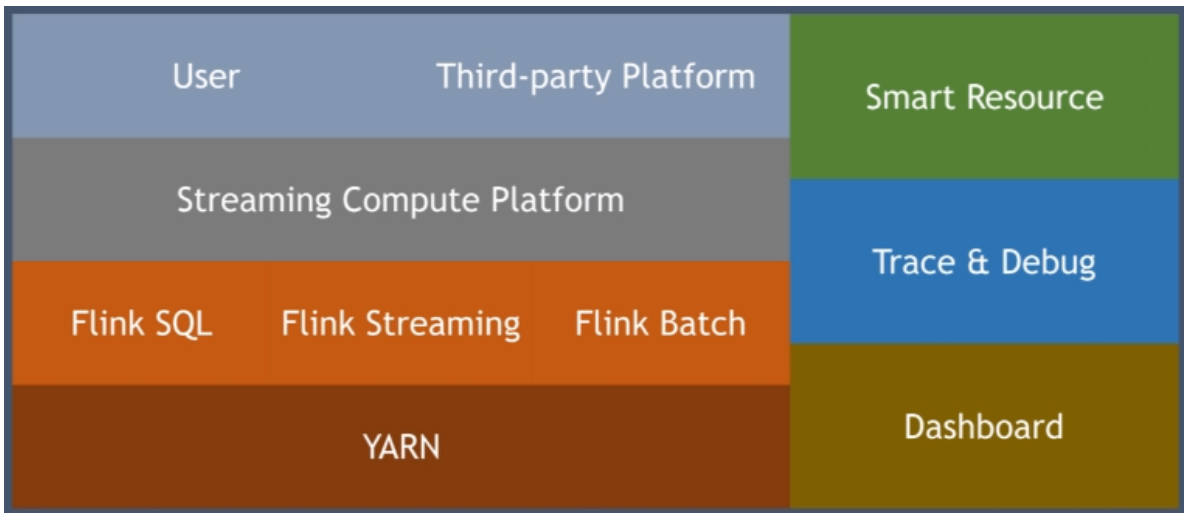
出错之后会进行回滚：



2、应用场景

字节：

平台架构：构建在YARN上，分为了三种不同的作业类型：SQL任务，流式计算和批处理，上层是一个流式计算平台，用户可以在上方提交并管理自己的任务。右侧是为用户提供的额外功能，帮助用户能够更好地控制作业。



应用场景：

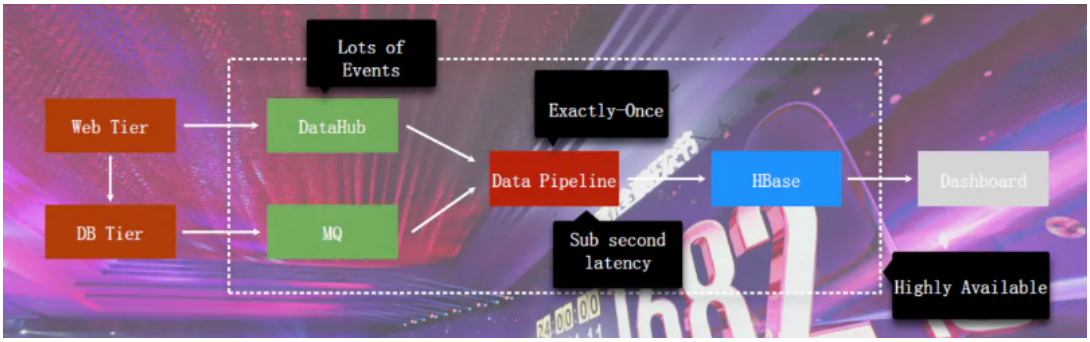
数仓：用户行为的解析和清洗、使用Flink SQL构建实时的数仓、不同数据源的导入导出。

风控和安全：异常检测、基于规则的报警

机器学习：数据预处理、用于模型训练时的资源管理和调度。

阿里巴巴：

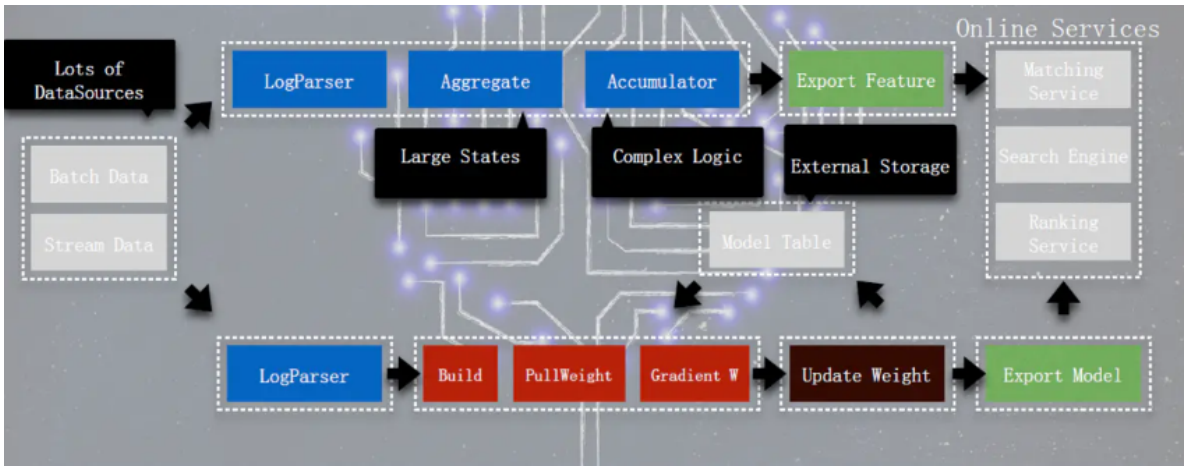
双11大屏：



阿里的在线服务系统和数据库会实时产生大量日志数据并进入消息队列，FlinkJob会从消息队列中实时读取处理这些数据，然后将各种统计分析结果实时更新到KV/Table存储系统中，例如：HBase，终端用户可以通过Dashboard实时看到各种维度的数据统计分析结果。

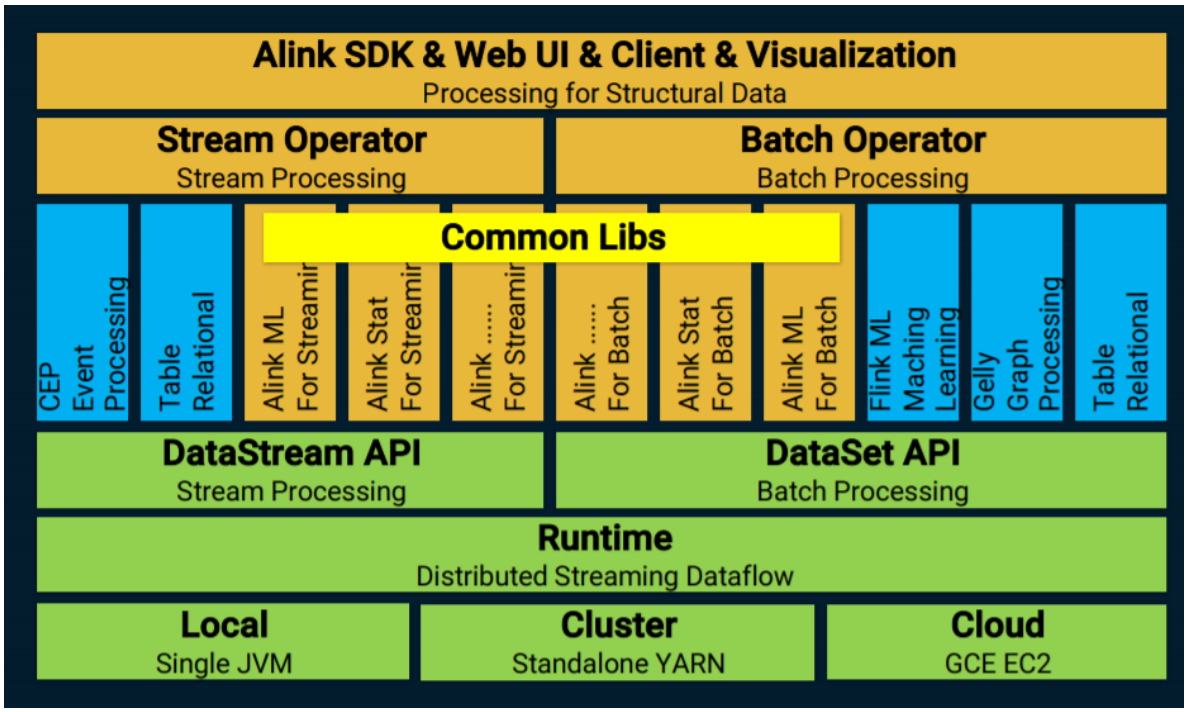
实时的机器学习:

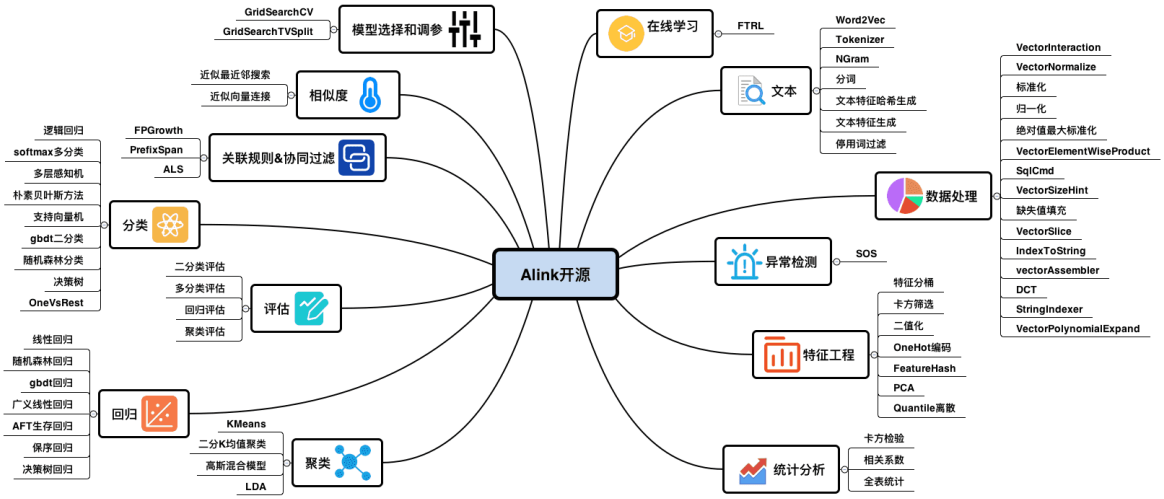
传统的机器学习对Feature的收集和Model的训练频率较低，无法适应不断变化的应用需求。例如在双11时，商品的价格、活动的规则与平时完全不同，依据之前的数据进行训练得不到最优的效果。因此，只有实时收集Feature，训练Model才能拟合出较为满意的结果。



在线学习系统的优势在于可以**实时收集并处理用户的行为数据，从而进行实时流式的特征计算和在线训练，并将模型的增量更新实时同步回在线系统，形成数据闭环。**

Alink（Alibaba、Algorithm、 AI、 Flink、 Blink）已开源





3、思考

1、关于离线和实时计算场景的划分。

离线：数据预处理、模型的训练

实时：查询、在线推理（数据的获取、模型的推送）