Kafka-1

引入消息系统的目的

* 解耦在项目启动之初来预测将来项目会碰到什么需求，是极其困难的。消息系统在处理过程中间插入了一个隐含的、基于数据的接口层，两边的处理过程都要实现这一接口。这允许你独立的扩展或修改两边的处理过程，只要确保它们遵守同样的接口约束。
* 冗余有些情况下，处理数据的过程会失败。除非数据被持久化，否则将造成丢失。消息队列把数据进行持久化直到它们已经被完全处理，通过这一方式规避了数据丢失风险。许多消息队列所采用的"插入 - 获取 - 删除"范式中，在把一个消息从队列中删除之前，需要你的处理系统明确的指出该消息已经被处理完毕，从而确保你的数据被安全的保存直到你使用完毕。
* 扩展性因为消息队列解耦了你的处理过程，所以增大消息入队和处理的频率是很容易的，只要另外增加处理过程即可。不需要改变代码、不需要调节参数。扩展就像调大电力按钮一样简单。
* 灵活性 & 峰值处理能力在访问量剧增的情况下，应用仍然需要继续发挥作用，但是这样的突发流量并不常见；如果为以能处理这类峰值访问为标准来投入资源随时待命无疑是巨大的浪费。使用消息队列能够使关键组件顶住突发的访问压力，而不会因为突发的超负荷的请求而完全崩溃。
* 可恢复性系统的一部分组件失效时，不会影响到整个系统。消息队列降低了进程间的耦合度，所以即使一个处理消息的进程挂掉，加入队列中的消息仍然可以在系统恢复后被处理。
* 顺序保证在大多使用场景下，数据处理的顺序都很重要。大部分消息队列本来就是排序的，并且能保证数据会按照特定的顺序来处理。Kafka 保证一个 Partition 内的消息的有序性。
* 缓冲在任何重要的系统中，都会有需要不同的处理时间的元素。例如，加载一张图片比应用过滤器花费更少的时间。消息队列通过一个缓冲层来帮助任务最高效率的执行———写入队列的处理会尽可能的快速。该缓冲有助于控制和优化数据流经过系统的速度。
* 异步通信很多时候，用户不想也不需要立即处理消息。消息队列提供了异步处理机制，允许用户把一个消息放入队列，但并不立即处理它。想向队列中放入多少消息就放多少，然后在需要的时候再去处理它们。

Kafka 是一种分布式的，基于发布 / 订阅的消息系统。

1，特点

* 以时间复杂度为 O(1) 的方式提供消息持久化能力和访问性能
* 高吞吐率。即使在非常廉价的商用机器上也能做到单机支持每秒 100K 条以上消息的传输。
* 支持Kafka Server 间的消息分区，及分布式消费，同时保证每个 Partition 内的消息顺序传输
* 同时支持离线数据处理和实时数据处理。
* 支持在线水平扩展（Scale out）。

2，术语（Terminology）

* Broker Kafka 集群包含一个或多个服务器，这种服务器被称为 broker
* Topic 每条发布到 Kafka 集群的消息都有一个类别，这个类别被称为 Topic。（物理上不同 Topic 的消息分开存储，逻辑上一个 Topic 的消息虽然保存于一个或多个 broker 上但用户只需指定消息的 Topic 即可生产或消费数据而不必关心数据存于何处）
* Partition Parition 是物理上的概念，每个 Topic 包含一个或多个 Partition.
* Producer 负责发布消息到 Kafka broker
* Consumer 消息消费者，向 Kafka broker 读取消息的客户端。
* Consumer Group 每个 Consumer 属于一个特定的 Consumer Group（可为每个 Consumer 指定 group name，若不指定 group name 则属于默认的 group）。

3，拓扑结构

4，

Topic & Partition

Topic 在逻辑上可以被认为是一个 queue，每条消费都必须指定它的 Topic。

为了使得 Kafka 的吞吐率可以线性提高，物理上把 Topic 分成一个或多个 Partition，每个 Partition 在物理上对应一个文件夹，该文件夹下存储这个 Partition 的所有日志文件和索引文件。

每个日志文件都是一个 log entrie 序列，由多个 segment组成，每个 segment 以该 segment 第一条消息的 offset 命名并以“.kafka”为后缀。另外会有一个索引文件，它标明了每个 segment 下包含的 log entry 的 offset 范围。

Kafka 集群会保留所有的消息（无论其被消费与否），还提供两种策略删除旧数据。一是基于时间，二是基于 Partition 文件大小。

Producer 消息路由

Producer 发送消息到 broker 时，Producer 根据被指定的消息 key 和 Partition 机制来判断应该将这条消息发送到哪个 Parition。

Consumer Group

使用 Consumer high level API 时，同一 Topic 的一条消息只能被同一个 Consumer Group 内的一个 Consumer 消费，但多个 Consumer Group 可同时消费这一消息。

Push vs. Pull

Kafka 选择由 Producer 向 broker push 消息并由 Consumer 从 broker pull 消息的pull 模式。

push 模式的目标是尽可能以最快速度传递消息，但很难适应消费速率不同的消费者

pull 模式可简化 broker 的设计，Consumer 可自主控制消费消息的速率及消费方式（批量/逐条）

Delivery Guarantee

* At most once 消息可能会丢，但绝不会重复传输

->Consumer读完消息先 commit 再处理消息的时候

* At least one 消息绝不会丢，但可能会重复传输

->Producer 向 broker 发送消息时的默认情况

->Consumer读完消息先处理再 commit

* Exactly once 每条消息肯定会被传输一次且仅传输一次（很多时候这是用户所想要的）

->Consumer引入两阶段提交

Zookeeper在Kafka中的作用

Kafka 通过 Zookeeper 管理集群配置，选举 leader，以及在 Consumer Group 发生变化时进行 rebalance。

保证高吞吐率的几个重要的机制

* 每条消息都以append 形式写入Partition 中，属于顺序写磁盘，因此效率非常高（经验证，顺序写磁盘效率比随机写内存还要高）。
* 每一个 Consumer Group的当前消费所到达的 position，为 offset。这个 offset 由 Consumer 控制。正常情况下 Consumer 会在消费完一条消息后递增该 offset。因为 offet 由 Consumer 控制，所以 Kafka broker 是无状态的，它不需要标记哪些消息被哪些消费过，也不需要通过 broker 去保证同一个 Consumer Group 只有一个 Consumer 能消费某一条消息，因此也就不需要锁机制，这也为 Kafka 的高吞吐率提供了有力保障。
* 不同的消息可以并行写入不同 broker 的不同 Partition 里，如果 Partition 机制设置合理，所有到达的消息可以均匀分布到不同的 Partition 里，这样就实现了负载均衡

参考

<https://www.infoq.cn/article/kafka-analysis-part-1>

[infoq.cn/profile/2C3D71FB990091/publish/all](http://infoq.cn/profile/2C3D71FB990091/publish/all)

次回计划 Kafka High Availability

<https://www.infoq.cn/article/kafka-analysis-part-2>

<https://www.infoq.cn/article/kafka-analysis-part-3>