

ALIBABA CLOUD

阿里云

消息队列 Kafka 版 最佳实践

文档版本：20201021

 阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
 危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 危险 重置操作将丢失用户配置数据。
 警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 警告 重启操作将导致业务中断，恢复业务时间约十分钟。
 注意	用于警示信息、补充说明等，是用户必须了解的内容。	 注意 权重设置为0，该服务器不会再接受新请求。
 说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击设置> 网络> 设置网络类型。
粗体	表示按键、菜单、页面名称等UI元素。	在结果确认页面，单击确定。
<code>Courier</code> 字体	命令或代码。	执行 <code>cd /d C:/window</code> 命令，进入Windows系统文件夹。
<i>斜体</i>	表示参数、变量。	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[] 或者 [a b]	表示可选项，至多选择一个。	<code>ipconfig [-all -t]</code>
{ } 或者 {a b}	表示必选项，至多选择一个。	<code>switch {active stand}</code>

目录

1.发布者最佳实践	05
2.订阅者最佳实践	07
3.在线迁移至消息队列Kafka版	11

1.发布者最佳实践

本文介绍消息队列Kafka版发布者的最佳实践，帮助您减少发送消息出错的可能性。本文最佳实践基于Java客户端。对于其它语言的客户端，其基本概念与思想是相通的，但实现细节可能存在差异。

发送消息

发送消息的示例代码如下：

```
Future<RecordMetadata> metadataFuture = producer.send(new ProducerRecord<String, String>(
    topic, //消息主题。
    null, //分区编号。建议为null, 由Producer分配。
    System.currentTimeMillis(), //时间戳。
    String.valueOf(value.hashCode()), //消息键。
    value //消息值。
));
```

完整示例代码请参见[SDK概述](#)。

Key和Value

0.10.2.2版本的Kafka的消息字段只有两个：Key和Value。

- Key：消息的标识。
- Value：消息内容。

为了便于追踪，请为消息设置一个唯一的Key。您可以通过Key追踪某消息，打印发送日志和消费日志，了解该消息的发送和消费情况。

失败重试

分布式环境下，由于网络等原因偶尔发送失败是常见的。导致这种失败的原因可能是消息已经发送成功，但是Ack失败，也有可能是确实没发送成功。

消息队列Kafka版是VIP网络架构，会主动断开空闲连接（30秒没活动），因此，不是一直活跃的客户端会经常收到 "connection rest by peer" 错误，建议重试消息发送。

您可以根据业务需求，设置以下重试参数：

- `retries`，重试次数，建议设置为 `3`。
- `retry.backoff.ms`，重试间隔，建议设置为 `1000`。

异步发送

发送接口是异步的，如果您想得到发送的结果，可以调用`metadataFuture.get(timeout, TimeUnit.MILLISECONDS)`。

线程安全

Producer是线程安全的，且可以往任何Topic发送消息。通常情况下，一个应用对应一个Producer就足够了。

Acks

Acks的说明如下：

- `acks=0`：无需服务端的Response、性能较高、丢数据风险较大。
- `acks=1`：服务端主节点写成功即返回Response、性能中等、丢数据风险中等、主节点宕机可能导致数据丢失。
- `acks=all`：服务端主节点写成功且备节点同步成功才返回Response、性能较差、数据较为安全、主节点和备节点都宕机才会导致数据丢失。

一般建议选择 `acks=1`，重要的服务可以设置 `acks=all`。

Batch

Batch的基本思路是：把消息缓存在内存中，并进行打包发送。Kafka通过Batch来提高吞吐，但同时也会增加延迟，生产时应该对两者予以权衡。在构建Producer时，需要考虑以下两个参数：

- `batch.size`：发往每个分区（Partition）的消息缓存量（消息内容的字节数之和，不是条数）达到这个数值时，就会触发一次网络请求，然后客户端把消息真正发往服务器。
- `linger.ms`：每条消息待在缓存中的最长时间。若超过这个时间，就会忽略 `batch.size` 的限制，然后客户端立即把消息发往服务器。

因此，Kafka客户端什么时候把消息真正发往服务器是由 `batch.size` 和 `linger.ms` 共同决定的。您可以根据具体业务需求进行调整。

OOM

结合Kafka的Batch设计思路，Kafka会缓存消息并打包发送，如果缓存太多，则有可能造成OOM（Out of Memory）。

- `buffer.memory`：所有缓存消息的总体大小超过这个数值后，就会触发把消息发往服务器。此时会忽略 `batch.size` 和 `linger.ms` 的限制。
- `buffer.memory` 的默认数值是32 MB，对于单个Producer来说，可以保证足够的性能。需要注意的是，如果您在同一个JVM中启动多个Producer，那么每个Producer都有可能占用32 MB缓存空间，此时便有可能触发OOM。
- 在生产时，一般没有必要启动多个Producer；如果特殊情况需要，则需要考虑 `buffer.memory` 的大小，避免触发OOM。

分区顺序

单个分区（Partition）内，消息是按照发送顺序储存的，是基本有序的。

默认情况下，消息队列Kafka版为了提升可用性，并不保证单个分区内绝对有序，在升级或者宕机时，会发生少量消息乱序（某个分区挂掉后把消息Failover到其它分区）。

对于包年包月计费模式下的专业版实例，如果业务要求分区保证严格有序，请在创建Topic时指定保序。

2. 订阅者最佳实践

本文主要介绍消息队列Kafka版订阅者的最佳实践，帮助您减少消费消息出错的可能性。

kafka 最佳实践

消费消息基本流程

消息队列Kafka版订阅者在订阅消息时的基本流程是：

1. Poll数据。
2. 执行消费逻辑。
3. 再次poll数据。

负载均衡

每个Consumer Group可以包含多个消费实例，即可以启动多个消息队列Kafka版Consumer，并把参数 `group.id` 设置成相同的值。属于同一个Consumer Group的消费实例会负载消费订阅的Topic。

例如Consumer Group A订阅了Topic A，并开启三个消费实例C1、C2、C3，则发送到Topic A的每条消息最终只会传给C1、C2、C3的某一个。Kafka默认会均匀地把消息传给各个消息实例，以做到消费负载均衡。

Kafka负载消费的内部原理是，把订阅的Topic的分区，平均分配给各个消费实例。因此，消费实例的个数不要大于分区的数量，否则会有实例分配不到任何分区而处于空跑状态。这个负载均衡发生的时间，除了第一次启动上线之外，后续消费实例发生重启、增加、减少等变更时，都会触发一次负载均衡。

消息队列Kafka版的每个Topic的分区数量默认是16个，已经足够满足大部分场景的需求，且云上服务会根据容量调整分区数。

多个订阅

消息队列Kafka版支持以下多个订阅方式：

- Consumer Group订阅多个Topic。

一个Consumer Group可以订阅多个Topic，多个Topic的消息被Consumer Group中的Consumer均匀消费。例如Consumer Group A订阅了Topic A、Topic B、Topic C，则这三个Topic中的消息，被Consumer Group中的Consumer均匀消费。

Consumer Group订阅多个Topic的示例代码如下：

```
String topicStr = kafkaProperties.getProperty("topic");
String[] topics = topicStr.split(",");
for (String topic: topics) {
    subscribedTopics.add(topic.trim());
}
consumer.subscribe(subscribedTopics);
```

- Topic被多个Consumer Group订阅。

一个Topic可以被多个Consumer Group订阅，且各个Consumer Group独立消费Topic下的所有消息。例如Consumer Group A订阅了Topic A，Consumer Group B也订阅了Topic A，则发送到Topic A的每条消息，不仅会传一份给Consumer Group A的消费实例，也会传一份给Consumer Group B的消费实例，且这两个过程相互独立，相互没有任何影响。

一个Consumer Group对应一个应用

建议一个Consumer Group对应一个应用，即不同的应用对应不同的代码。如果您需要将不同的代码写在同一个应用中，请准备多份不同的kafka.properties。例如kafka1.properties、kafka2.properties。

消费位点

每个Topic会有多个分区，每个分区会统计当前消息的总条数，这个称为最大位点MaxOffset。

消息队列Kafka版Consumer会按顺序依次消费分区内的每条消息，记录已经消费了的消息条数，称为ConsumerOffset。

剩余的未消费的条数（也称为消息堆积量）=MaxOffset-ConsumerOffset。

消费位点提交

消息队列Kafka版消费者有两个相关参数：

- enable.auto.commit：默认值为true。
- auto.commit.interval.ms：默认值为1000，即1s。

这两个参数组合的结果就是，每次poll数据前会先检查上次提交位点的时间，如果距离当前时间已经超过参数auto.commit.interval.ms规定的时长，则客户端会启动位点提交动作。

因此，如果将enable.auto.commit设置为true，则需要在每次poll数据时，确保前一次poll出来的数据已经消费完毕，否则可能导致位点跳跃。

如果想自己控制位点提交，请把enable.auto.commit设为false，并调用commit(offsets)函数自行控制位点提交。

消费位点重置

以下两种情况，会发生消费位点重置：

- 当服务端不存在曾经提交过的位点时（例如客户端第一次上线）。
- 当从非法位点拉取消息时（例如某个分区最大位点是10，但客户端却从11开始拉取消息）。

Java客户端可以通过auto.offset.reset来配置重置策略，主要有三种策略：

- latest：从最大位点开始消费。
- earliest：从最小位点开始消费。
- none：不做任何操作，即不重置。

② 说明

- 建议设置成latest，而不要设置成earliest，避免因位点非法时从头开始消费，从而造成大量重复。
- 如果是您自己管理位点，可以设置成none。

拉取大消息

消费过程是由客户端主动去服务端拉取消息的，在拉取大消息时，需要注意控制拉取速度，注意修改配置：

- max.poll.records：如果单条消息超过1 MB，建议设置为1。
- fetch.max.bytes：设置比单条消息的大小略大一点。
- max.partition.fetch.bytes：设置比单条消息的大小略大一点。

拉取大消息的核心是逐条拉取的。

拉取公网

通过公网消费消息时，通常会因为公网带宽的限制导致连接被断开，此时需要注意控制拉取速度，修改配置：

1. `fetch.max.bytes`：建议设置成公网带宽的一半（注意该参数的单位是bytes，公网带宽的单位是bits）
2. `max.partition.fetch.bytes`：建议设置成`fetch.max.bytes`的三分之一或者四分之一。

消息重复和消费幂等

消息队列Kafka版消费的语义是`at least once`，也就是至少投递一次，保证消息不丢失，但是无法保证消息不重复。在出现网络问题、客户端重启时均有可能造成少量重复消息，此时应用消费端如果对消息重复比较敏感（例如订单交易类），则应该做消息幂等。

以数据库类应用为例，常用做法是：

- 发送消息时，传入key作为唯一流水号ID。
- 消费消息时，判断key是否已经消费过，如果已经消费过了，则忽略，如果没消费过，则消费一次。

当然，如果应用本身对少量消息重复不敏感，则不需要做此类幂等检查。

消费失败

消息队列Kafka版是按分区逐条消息顺序向前推进消费的，如果消费端拿到某条消息后执行消费逻辑失败，例如应用服务器出现了脏数据，导致某条消息处理失败，等待人工干预，那么有以下两种处理方式：

- 失败后一直尝试再次执行消费逻辑。这种方式有可能造成消费线程阻塞在当前消息，无法向前推进，造成消息堆积。
- 由于消息队列Kafka版没有处理失败消息的设计，实践中通常会打印失败的消息或者存储到某个服务（例如创建一个Topic专门用来放失败的消息），然后定时检查失败消息的情况，分析失败原因，根据情况处理。

消费延迟

消息队列Kafka版的消费机制是由客户端主动去服务端拉取消息进行消费的。因此，一般来说，如果客户端能够及时消费，则不会产生较大延迟。如果产生了较大延迟，请先关注是否有堆积，并注意提高消费速度。

消费阻塞以及堆积

消费端最常见的问题就是消费堆积，最常造成堆积的原因是：

- 消费速度跟不上生产速度，此时应该提高消费速度，详情请参见[提高消费速度](#)。
- 消费端产生了阻塞。

消费端拿到消息后，执行消费逻辑，通常会执行一些远程调用，如果这个时候同步等待结果，则有可能造成一直等待，消费进程无法向前推进。

消费端应该竭力避免堵塞消费线程，如果存在等待调用结果的情况，建议设置等待的超时时间，超时后作为消费失败进行处理。

提高消费速度

提高消费速度有以下两个办法：

- 增加Consumer实例个数。

可以在进程内直接增加（需要保证每个实例对应一个线程，否则没有太大意义），也可以部署多个消费实例进程；需要注意的是，实例个数超过分区数量后就不再能提高速度，将会有消费实例不工作。

- 增加消费线程。

增加Consumer实例本质上也是增加线程的方式来提升速度，因此更加重要的性能提升方式是增加消费线程，最基本的步骤如下：

- i. 定义一个线程池。
- ii. Poll数据。
- iii. 把数据提交到线程池进行并发处理。
- iv. 等并发结果返回成功后，再次poll数据执行。

消息过滤

消息队列Kafka版自身没有消息过滤的语义。实践中可以采取以下两个办法：

- 如果过滤的种类不多，可以采取多个Topic的方式达到过滤的目的。
- 如果过滤的种类多，则最好在客户端业务层面自行过滤。

实践中请根据业务具体情况进行选择，也可以综合运用上面两种办法。

消息广播

消息队列Kafka版没有消息广播的语义，可以通过创建不同的Consumer Group来模拟实现。

订阅关系

同一个Consumer Group内，各个消费实例订阅的Topic最好保持一致，避免给排查问题带来干扰。

3.在线迁移至消息队列Kafka版

本文介绍如何选择并使用通用迁移和数据迁移两种方案将自建Kafka集群迁移至消息队列Kafka版。

场景描述

本文介绍了如何选择并使用通用迁移和数据迁移两种方案将阿里云自建Kafka集群迁移到消息队列Kafka版。同时本文可以作为线下IDC自建Kafka集群等场景迁移到消息队列Kafka版的参考手册。

解决的问题

- 如何通用迁移消息队列Kafka版。
- 如何数据迁移消息队列Kafka版。

部署架构图



选用的产品

- 消息队列Kafka版

消息队列Kafka版是阿里云基于Apache Kafka构建的高吞吐量、高可扩展性的分布式消息队列服务，广泛用于日志收集、监控数据聚合、流式数据处理、在线和离线分析等，是大数据生态中不可或缺的产品之一，阿里云提供全托管服务，免运维、免部署，更专业、更可靠、更安全。

更多关于消息队列Kafka版的介绍，参见[消息队列Kafka版产品详情页](#)。

- 云监控

云监控服务可用于收集获取阿里云资源的监控指标或用户自定义的监控指标，探测服务可用性，以及针对指标设置警报。使您全面了解阿里云上的资源使用情况、业务的运行状况和健康度，并及时收到异常报警做出反应，保证应用程序顺畅运行。

更多关于云监控的介绍，参见[云监控产品详情页](#)。

详细信息

[查看最佳实践详情](#)

更多最佳实践

[查看更多阿里云最佳实践](#)