|  |
| --- |
|  |
| **RocketMQ最佳实践** |
| v3.0.0 |
|  |
| ©Alibaba 淘宝消息中间件项目组 |
| 2013/9/23 |

**文档变更历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **主要更改内容** | **更改人** | **更改时间** |
| 1 | 建立初始版本 | 誓嘉vintage.wang@gmail.com | 2013/9/23 |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |

**目录**

[1 前言 1](#_Toc367714854)

[2 Producer最佳实践 1](#_Toc367714855)

[2.1 发送消息注意事项 1](#_Toc367714856)

[2.2 消息发送失败如何处理 2](#_Toc367714857)

[2.3 选择oneway形式发送 2](#_Toc367714858)

[2.4 发送顺序消息注意事项 2](#_Toc367714859)

[3 Consumer最佳实践 2](#_Toc367714860)

[3.1 消费失败处理方式 2](#_Toc367714861)

[3.2 消费速度慢处理方式 2](#_Toc367714862)

[3.2.1 提高消费速度 2](#_Toc367714863)

[3.2.2 跳过非重要消息 3](#_Toc367714864)

[3.3 消费打印日志 3](#_Toc367714865)

[3.4 利用服务器消息过滤，避免多余的消息传输 4](#_Toc367714866)

[4 新上线一个应用需要注意什么 4](#_Toc367714867)

# 前言

本文档旨在描述RocketMQ使用过程中的一些最佳实践，建议用户这样做，但是非必须。

# Producer最佳实践

## 发送消息注意事项

1. 一个应用尽可能用一个Topic，消息子类型用tags来标识，tags可以由应用自由设置。只有发送消息设置了tags，消费方在订阅消息时，才可以利用tags在broker做消息过滤。

message.setTags("TagA");

1. 每个消息在业务层面的唯一标识码，要设置到keys字段，方便将来定位消息丢失问题。服务器会为每个消息创建索引（哈希索引），应用可以通过topic，key来查询这条消息内容，以及消息被谁消费。由于是哈希索引，请务必保证key尽可能唯一，这样可以避免潜在的哈希冲突。

// 订单Id

String orderId = "20034568923546";

message.setKeys(orderId);

1. 消息发送成功或者失败，要打印消息日志，务必要打印sendresult和key字段。
2. send消息方法，只要不抛异常，就代表发送成功。但是发送成功会有多个状态，在sendResult里定义。

* SEND\_OK  
  消息发送成功
* FLUSH\_DISK\_TIMEOUT  
  消息发送成功，但是服务器刷盘超时，消息已经进入服务器队列，只有此时服务器宕机，消息才会丢失
* FLUSH\_SLAVE\_TIMEOUT  
  消息发送成功，但是服务器同步到Slave时超时，消息已经进入服务器队列，只有此时服务器宕机，消息才会丢失
* SLAVE\_NOT\_AVAILABLE  
  消息发送成功，但是此时slave不可用，消息已经进入服务器队列，只有此时服务器宕机，消息才会丢失

1. 对于消息不可丢失应用，务必要有消息重发机制

例如如果消息发送失败，存储到数据库，能有定时程序尝试重发，或者人工触发重发。

## 消息发送失败如何处理

## 选择oneway形式发送

## 发送顺序消息注意事项

# Consumer最佳实践

## 消费失败处理方式

## 消费速度慢处理方式

### 提高消费速度



‑1消费并行度与消费吞吐量关系



‑2消费并行度与消费RT关系

绝大部分消息消费行为属于IO密集型，及可能是操作数据库，或者调用RPC，这类消费行为的消费速度在于后端数据库或者外系统的吞吐量，通过增加消费并行度，可以提供总的消费吞吐量，但是并行度增加到一定程度，反而会下降，如图所示，呈现抛物线形式。

所以应用必须要设置合理的并行度。

CPU密集型除外。

### 跳过非重要消息

发生消息堆积时，如果消费速度一直追不上发送速度，可以选择丢弃不重要的消息

## 消费打印日志

如果消息量较少，建议在消费入口方法打印消息，方便后面排查问题。

## 利用服务器消息过滤，避免多余的消息传输

# 新上线一个应用需要注意什么