Apusic 金蝶天燕

生成时间：2026-07-13


# 性能优化手册

金蝶Apusic分布式消息队列V2.0.6_for_rabbitmq

## 版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

## 免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

## 商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

# 目录


- 1 前言
  - 1.1 适用对象
  - 1.2 相关文档
  - 1.3 技术支持
- 2 系统层面优化
  - 2.1 操作系统优化
    - 2.1.1 文件描述符
    - 2.1.2 内核参数
    - 2.1.3 磁盘优化
  - 2.2 网络优化
- 3 RabbitMQ 配置优化
  - 3.1 内存管理
  - 3.2 磁盘管理
  - 3.3 连接与通道优化
  - 3.4 队列优化
    - 3.4.1 队列类型选择
    - 3.4.2 队列参数优化
  - 3.5 消息持久化优化
    - 3.5.1 何时使用持久化
    - 3.5.2 批量持久化
- 4 客户端优化
  - 4.1 连接管理
    - 4.1.1 连接池
    - 4.1.2 自动恢复
  - 4.2 生产者优化
    - 4.2.1 批量发送
    - 4.2.2 异步发送
  - 4.3 消费者优化
    - 4.3.1 QoS 预取
    - 4.3.2 批量确认
    - 4.3.3 多消费者
- 5 集群优化

- 5.1 节点部署策略
- 5.2 镜像队列优化
- 5.3 Quorum Queue 优化
- 5.4 Stream Queue 优化

- 6 监控与调优
  - 6.1 性能指标
  - 6.2 性能测试
  - 6.3 瓶颈分析
- 7 常见问题优化
  - 7.1 消息堆积优化
  - 7.2 内存不足优化
  - 7.3 高延迟优化
- 8 性能优化检查清单
  - 8.1 系统层面
  - 8.2 RabbitMQ 配置
  - 8.3 客户端优化
  - 8.4 集群优化

# 1 前言

本文档介绍金蝶Apusic分布式消息队列for RabbitMQ（Apusic Distributed Message Queue，简称：ADMQ for RabbitMQ）的性能优化最佳实践，帮助用户充分利用系统性能。

## 1.1 适用对象

本文档适用于IT信息化业务负责人、研发经理、软件项目经理、软件架构师、运维工程师。

## 1.2 相关文档

了解更多ADMQ for RabbitMQ产品相关的信息，请参阅以下ADMQ for RabbitMQ产品手册文档集：

| 序号 | 手册文档 | 说明 |
| --- | --- | --- |
| 1 | 金蝶Apusic分布式消息队列for RabbitMQ 快速使用手册 | 简单介绍了如何快速上手使用ADMQ for RabbitMQ 。 |
| 2 | 金蝶Apusic分布式消息队列for RabbitMQ 安装手册 | 详细介绍如何在各操作系统上安装ADMQ for RabbitMQ，以及ADMQ for RabbitMQ服务启停等操作。 |
| 3 | 金蝶Apusic分布式消息队列for RabbitMQ 消息引擎用户手册 | 详细介绍 ADMQ for RabbitMQ 消息引擎相关功能的使用、配置、管理及配套工具的使用方法。 |
| 4 | 金蝶Apusic分布式消息队列for RabbitMQ 管控台用户手册 | 详细介绍ADMQ for RabbitMQ管控台相关功能的使用和操作说明。 |
| 5 | 金蝶Apusic分布式消息队列for RabbitMQ 开发手册 | 详细介绍基于各开发语言进行ADMQ for RabbitMQ客户端应用开发的说明。 |
| 6 | 金蝶Apusic分布式消息队列for RabbitMQ 迁移手册 | 详细介绍从RabbitMQ迁移到ADMQ for RabbitMQ的说明。 |
| 7 | 金蝶Apusic分布式消息队列for RabbitMQ 运维手册 | 详细介绍ADMQ for RabbitMQ的监控、运维、安全加固等运维说明。 |
| 8 | 金蝶Apusic分布式消息队列for RabbitMQ 性能优化手册 | 详细介绍ADMQ for RabbitMQ性能调优的说明。 |

## 1.3 技术支持

ADMQ for RabbitMQ产品提供全面的技术支持服务，您可以通过以下方式获得技术支持：

- 网址：www.apusic.com
- 电话：400-855-5800
- 邮箱：support@apusic.com
- 金蝶云社区：https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

您在取得技术支持时，请提供如下信息：

1. 您的姓名
2. 公司信息与联系方式
3. 操作系统及其版本
4. 产品版本号
5. 出现异常及错误的日志、截图等详细信息

# 2 系统层面优化

## 2.1 操作系统优化

### 2.1.1 文件描述符

```
# 查看当前限制
ulimit -n

# 修改限制 (/etc/security/limits.conf)
* soft nofile 65535
* hard nofile 65535
```

### 2.1.2 内核参数

```
# /etc/sysctl.conf

# 增加端口范围
net.ipv4.ip_local_port_range = 1024 65535

# TCP 连接优化
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_fin_timeout = 30
net.core.somaxconn = 65535

# 内存优化
vm.swappiness = 10
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10

# 应用配置
sysctl -p
```

### 2.1.3 磁盘优化

| 优化项 | 建议 |
| --- | --- |
| 文件系统 | 使用 XFS 文件系统 |
| 挂载选项 | noatime,nodiratime |
| 磁盘调度 | 使用 noop 或 deadline |
| RAID 配置 | 使用 RAID 10 或 RAID 0 |
| SSD 优化 | 启用 TRIM，预留 over-provisioning |

```
# 查看磁盘调度算法
cat /sys/block/sda/queue/scheduler

# 设置为 deadline
echo deadline > /sys/block/sda/queue/scheduler
```

## 2.2 网络优化

```
# 网络缓冲区
net.core.rmem_max = 134217728
net.core.wmem_max = 134217728
net.ipv4.tcp_rmem = 4096 87380 134217728
net.ipv4.tcp_wmem = 4096 65536 134217728

# TCP 优化
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_sack = 1
net.ipv4.tcp_no_metrics_save = 1
```

# 3 RabbitMQ 配置优化

## 3.1 内存管理

| 参数 | 默认值 | 推荐值 | 说明 |
| --- | --- | --- | --- |
| vm_memory_high_watermark.relative | 0.4 | 0.6-0.7 | 内存使用上限比例 |
| vm_memory_high_watermark_paging_ratio | 0.5 | 0.5 | 开始分页的内存比例 |
| vm_memory_high_watermark.absolute | - | 4G | 绝对内存上限 |

```
# rabbitmq.conf
vm_memory_high_watermark.relative = 0.6
vm_memory_high_watermark_paging_ratio = 0.5
```

## 3.2 磁盘管理

| 参数 | 默认值 | 推荐值 | 说明 |
| --- | --- | --- | --- |
| disk_free_limit.relative | 2.0 | 1.5 | 磁盘空间下限比例 |
| disk_free_limit.absolute | 50MB | 1GB | 绝对磁盘空间下限 |

```
# rabbitmq.conf
disk_free_limit.relative = 1.5
disk_free_limit.absolute = 1GB
```

## 3.3 连接与通道优化

| 参数 | 默认值 | 推荐值 | 说明 |
| --- | --- | --- | --- |
| channel_max | 2047 | 根据需求调整 | 每个连接的最大通道数 |
| connection_max | infinity | 根据需求调整 | 最大连接数 |
| heartbeat | 60 | 30-60 | 心跳间隔（秒） |
| handshake_timeout | 10000 | 10000 | 握手超时（毫秒） |

```
# rabbitmq.conf
channel_max = 2047
heartbeat = 30
handshake_timeout = 10000
```

## 3.4 队列优化

### 3.4.1 队列类型选择

| 队列类型 | 吞吐量 | 可靠性 | 适用场景 |
| --- | --- | --- | --- |
| Classic Queue | 中等 | 中（需配置镜像） | 一般业务场景 |
| Quorum Queue | 中等 | 高 | 高可用场景 |
| Stream Queue | 极高 | 高 | 高吞吐、日志场景 |

### 3.4.2 队列参数优化

```
Map<String, Object> args = new HashMap<>();

# 队列长度限制，防止无限堆积
args.put("x-max-length", 100000);
args.put("x-max-length-bytes", 1073741824);  # 1GB

# 消息 TTL
args.put("x-message-ttl", 3600000);  # 1小时

# 队列 TTL
args.put("x-expires", 7200000);  # 2小时无消费者后删除

# 惰性队列（减少内存使用）
args.put("x-queue-mode", "lazy");

channel.queueDeclare("optimized.queue", true, false, false, args);
```

## 3.5 消息持久化优化

### 3.5.1 何时使用持久化

| 场景 | 建议 |
| --- | --- |
| 关键业务消息 | 必须持久化（delivery_mode=2） |
| 日志消息 | 可非持久化 |
| 高吞吐场景 | 权衡可靠性和性能 |

### 3.5.2 批量持久化

使用 Confirm 模式 + 批量确认提高持久化性能：

```
channel.confirmSelect();
channel.setConfirmCallback(...);

# 批量发送
for (int i = 0; i < 1000; i++) {
    channel.basicPublish(EXCHANGE, ROUTING_KEY,
        MessageProperties.PERSISTENT_TEXT_PLAIN,
        message.getBytes());
}

# 等待全部确认
channel.waitForConfirmsOrDie(5000);
```

# 4 客户端优化

## 4.1 连接管理

### 4.1.1 连接池

```
# 使用连接池管理连接
ConnectionFactory factory = new ConnectionFactory();
factory.setHost("localhost");
factory.setConnectionTimeout(30000);
factory.setRequestedHeartbeat(30);
factory.setAutomaticRecoveryEnabled(true);

# 单连接多通道（推荐）
Connection connection = factory.newConnection();
for (int i = 0; i < 10; i++) {
    Channel channel = connection.createChannel();
    # 使用 channel 处理消息
}
```

### 4.1.2 自动恢复

```
factory.setAutomaticRecoveryEnabled(true);
factory.setNetworkRecoveryInterval(5000);
factory.setTopologyRecoveryEnabled(true);
factory.setChannelRpcTimeout(10000);
```

## 4.2 生产者优化

### 4.2.1 批量发送

```
# 批量发送消息
List<Message> messages = new ArrayList<>();
for (int i = 0; i < 1000; i++) {
```

```
    messages.add(new Message(...));
}

# 使用事务或 Confirm 模式保证可靠性
channel.confirmSelect();
for (Message msg : messages) {
    channel.basicPublish(EXCHANGE, ROUTING_KEY, null,
msg.toBytes());
}
channel.waitForConfirmsOrDie();
```

#### 4.2.2 异步发送

```
# 使用线程池异步发送
ExecutorService executor = Executors.newFixedThreadPool(10);
for (Message msg : messages) {
    executor.submit(() -> {
        channel.basicPublish(EXCHANGE, ROUTING_KEY, null,
msg.toBytes());
    });
}
```

## 4.3 消费者优化

#### 4.3.1 QoS 预取

```
# 设置 QoS，控制未确认消息数量
channel.basicQos(100);  # 每个消费者最多 100 条未确认消息

# 全局 QoS
channel.basicQos(0, 100, true);  # 整个通道最多 100 条
```

#### 4.3.2 批量确认

```
# 消费消息并批量确认
List<Long> deliveryTags = new ArrayList<>();
for (int i = 0; i < 100; i++) {
    GetResponse response = channel.basicGet(QUEUE, false);
    if (response != null) {
        deliveryTags.add(response.getEnvelope().getDeliveryTag());
        processMessage(response);
    }
}

# 批量确认
if (!deliveryTags.isEmpty()) {
    channel.basicAck(deliveryTags.get(deliveryTags.size() - 1),
true);
}
```

#### 4.3.3 多消费者

```
# 创建多个消费者并行处理
for (int i = 0; i < 10; i++) {
    Channel channel = connection.createChannel();
    channel.basicQos(50);

    DeliverCallback callback = (tag, delivery) -> {
        processMessage(delivery);
        channel.basicAck(delivery.getEnvelope().getDeliveryTag(),
false);
    };

    channel.basicConsume(QUEUE, false, callback, tag -> {});
}
```

# 5 集群优化

## 5.1 节点部署策略

| 部署策略 | 说明 | 适用场景 |
| --- | --- | --- |
| 同机房部署 | 所有节点在同一机房 | 低延迟、高吞吐 |
| 跨机房部署 | 节点分布在不同机房 | 高可用、容灾 |
| 异构部署 | 不同配置节点混合 | 成本优化 |

## 5.2 镜像队列优化

```
# 优化镜像同步策略
bin/admq rabbitmq admin ctl set_policy ha-optimized "^important-.*"
'
{
    "ha-mode": "exactly",
    "ha-params": 2,
    "ha-sync-mode": "automatic",
    "ha-promote-on-shutdown": "when-synced"
}'
```

## 5.3 Quorum Queue 优化

```
# 调整 Raft 参数
bin/admq rabbitmq admin ctl eval 'application:set_env(rabbit,
quorum_commands_soft_limit, 256).'
bin/admq rabbitmq admin ctl eval 'application:set_env(rabbit,
quorum_commands_hard_limit, 512).'
```

## 5.4 Stream Queue 优化

```
Map<String, Object> args = new HashMap<>();
# 段文件大小
args.put("x-stream-max-segment-size-bytes", 104857600);  # 100MB

# 保留策略
args.put("x-stream-max-retention-size-bytes", 10737418240L);  # 10GB
args.put("x-stream-max-retention-time", 604800000);  # 7天

channel.queueDeclare("stream.queue", true, false, false, args);
```

# 6 监控与调优

## 6.1 性能指标

| 指标 | 健康范围 | 优化建议 |
| --- | --- | --- |
| CPU 使用率 | < 70% | 扩容或优化业务逻辑 |
| 内存使用率 | < vm_memory_high_watermark | 增加内存或优化队列配置 |
| 磁盘使用率 | < 80% | 扩容或清理数据 |
| 消息入速率 | 根据业务需求 | 优化生产者或扩容 |
| 消息出速率 | 接近入速率 | 优化消费者或增加消费者 |
| 未确认消息 | < QoS 预取数 | 优化消费者处理速度 |
| 连接数 | < connection_max | 优化连接使用或使用连接池 |

## 6.2 性能测试

```
# 使用 perf-test 工具测试
rabbitmq-perf-test -x 10 -y 10 -u "test-queue" -a --auto-delete
false

# 参数说明
# -x: 生产者数量
# -y: 消费者数量
# -u: 队列名称
# -a: 自动确认
# --auto-delete: 是否自动删除队列
```

## 6.3 瓶颈分析

1. **CPU 瓶颈**：消息处理逻辑复杂、加密解密、大量连接
2. **内存瓶颈**：队列堆积、大量连接、未启用惰性队列
3. **磁盘瓶颈**：大量持久化消息、磁盘 I/O 不足
4. **网络瓶颈**：跨机房部署、大量数据传输

# 7 常见问题优化

## 7.1 消息堆积优化

1. 增加消费者数量
2. 优化消费者处理逻辑
3. 设置队列长度限制
4. 使用死信队列处理过期消息

## 7.2 内存不足优化

1. 增加物理内存
2. 调整 vm_memory_high_watermark
3. 启用惰性队列
4. 清理不必要的队列和消息
5. 减少连接数

## 7.3 高延迟优化

1. 使用自动确认（非关键消息）
2. 减少持久化频率
3. 优化网络配置
4. 使用更近的节点
5. 减少消息大小

# 8 性能优化检查清单

## 8.1 系统层面

- [ ] 文件描述符限制已调整
- [ ] 内核参数已优化
- [ ] 磁盘使用 XFS 文件系统
- [ ] 网络参数已优化
- [ ] 系统资源充足（CPU、内存、磁盘）

## 8.2 RabbitMQ 配置

- [ ] 内存水位线合理设置
- [ ] 磁盘空间限制合理设置
- [ ] 心跳间隔适当
- [ ] 连接和通道数限制合理
- [ ] 队列 Leader 分配策略适当

## 8.3 客户端优化

- [ ] 使用连接池
- [ ] 使用单连接多通道
- [ ] 启用自动恢复
- [ ] 合理设置 QoS
- [ ] 使用批量确认
- [ ] 选择合适的确认模式

## 8.4 集群优化

- [ ] 节点选择合适的队列类型
- [ ] 镜像队列策略合理
- [ ] Quorum Queue 参数优化
- [ ] Stream Queue 保留策略合理
- [ ] 集群规模满足业务需求

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com