



APUSIC
固若长城
睿比世界

效能調優

金蝶Apusic分布式消息队列V2.0.6_for_rabbitmq

版權所有 © 深圳市金蝶天燕雲計算股份有限公司2026。保留所有權利。

版權聲明

本文件所涉及的軟件著作權、版權等知識產權已依法進行了註冊，由金蝶天燕雲計算股份有限公司合法擁有。受《中華人民共和國著作權法》《計算機軟件保護條例》《知識產權保護條例》和相關國際版權條約、法律、法規以及其它知識產權法律和條約的保護。未經授權許可，不得非法使用。

免責聲明

本文件包含的版權資訊由金蝶天燕雲計算股份有限公司合法擁有，受法律的保護，金蝶天燕雲計算股份有限公司對本文件可能涉及到的非金蝶天燕雲計算股份有限公司的資訊不承擔任何責任。在法律允許的範圍內，您可以查閱並僅能夠在《中華人民共和國著作權法》規定的合法範圍內複製和打印本文件。任何單位和個人未經金蝶天燕雲計算股份有限公司書面授權許可，不得使用、修改、再發佈本文件的任何部分和內容，否則將被視為侵權，金蝶天燕雲計算股份有限公司有依法追究其責任的權利。

本文件如有更新，不另行通知。對本文件中的問題您可向金蝶天燕雲計算股份有限公司告知或查詢。未經本公司明確授予的任何權利均予保留。

商標聲明

 是深圳市金蝶天燕雲計算股份有限公司向中華人民共和國國家商標局申請註冊的註冊商標，註冊商標專用權由金蝶天燕合法擁有，受法律保護。未經金蝶天燕的書面許可，任何單位及個人不得以任何方式或理由對該商標的任何部分進行使用、複製、修改、傳播、抄錄或與其它產品捆綁使用銷售。凡侵犯金蝶天燕商標權的，金蝶天燕將依法追究其法律責任。本文件提及的其他所有商標或註冊商標，由各自的所有人擁有。

目錄

- 1 前言
 - 1.1 適用對象
 - 1.2 相關文檔
 - 1.3 技術支援
- 2 系統層面優化
 - 2.1 作業系統優化
 - 2.1.1 檔案描述符
 - 2.1.2 核心參數
 - 2.1.3 磁碟優化
 - 2.2 網路優化
- 3 RabbitMQ 配置優化
 - 3.1 記憶體管理
 - 3.2 磁碟管理
 - 3.3 連線與通道優化
 - 3.4 佇列優化
 - 3.4.1 佇列類型選擇
 - 3.4.2 佇列參數優化
 - 3.5 訊息持久化優化
 - 3.5.1 何時使用持久化
 - 3.5.2 批量持久化
- 4 用戶端優化
 - 4.1 連線管理
 - 4.1.1 連線池
 - 4.1.2 自動復原
 - 4.2 生產者優化
 - 4.2.1 批量發送
 - 4.2.2 非同步發送
 - 4.3 消費者優化
 - 4.3.1 QoS 預取
 - 4.3.2 批量確認
 - 4.3.3 多消費者
- 5 叢集優化

- 5.1 節點部署策略
- 5.2 鏡像佇列優化
- 5.3 Quorum Queue 優化
- 5.4 Stream Queue 優化
- 6 監控與調優
 - 6.1 效能指標
 - 6.2 效能測試
 - 6.3 瓶頸分析
- 7 常見問題優化
 - 7.1 訊息堆積優化
 - 7.2 記憶體不足優化
 - 7.3 高延遲優化
- 8 效能優化檢查清單
 - 8.1 系統層面
 - 8.2 RabbitMQ 配置
 - 8.3 用戶端優化
 - 8.4 叢集優化

1 前言

本文檔介紹金蝶Apusic分散式訊息佇列for RabbitMQ（Apusic Distributed Message Queue，簡稱：ADMQ for RabbitMQ）的效能優化最佳實踐，幫助用戶充分利用系統效能。

1.1 適用對象

本文檔適用於IT資訊化業務負責人、研發經理、軟體專案經理、軟體架構師、運維工程師。

1.2 相關文檔

了解更多ADMQ for RabbitMQ產品相關的資訊，請參閱以下ADMQ for RabbitMQ產品手冊文件集：

序號	手冊文檔	說明
1	金蝶Apusic分散式訊息佇列for RabbitMQ 快速使用手冊	簡單介紹了如何快速上手使用ADMQ for RabbitMQ。
2	金蝶Apusic分散式訊息佇列for RabbitMQ 安裝手冊	詳細介紹如何在各作業系統上安裝ADMQ for RabbitMQ，以及ADMQ for RabbitMQ服務啟停等操作。
3	金蝶Apusic分散式訊息佇列for RabbitMQ 訊息引擎使用者手冊	詳細介紹 ADMQ for RabbitMQ 訊息引擎相關功能的使用、配置、管理及配套工具的使用方法。
4	金蝶Apusic分散式訊息佇列for RabbitMQ 管控台使用者手冊	詳細介紹ADMQ for RabbitMQ管控台相關功能的使用和操作說明。
5	金蝶Apusic分散式訊息佇列for RabbitMQ 開發手冊	詳細介紹基於各開發語言進行ADMQ for RabbitMQ用戶端應用開發的說明。
6	金蝶Apusic分散式訊息佇列for RabbitMQ 遷移手冊	詳細介紹從RabbitMQ遷移到ADMQ for RabbitMQ的說明。
7	金蝶Apusic分散式訊息佇列for RabbitMQ 運維手冊	詳細介紹ADMQ for RabbitMQ的監控、運維、安全加固等運維說明。
8	金蝶Apusic分散式訊息佇列for RabbitMQ 效能優化手冊	詳細介紹ADMQ for RabbitMQ效能調優的說明。

1.3 技術支援

ADMQ for RabbitMQ產品提供全面的技術支援服務，您可以透過以下方式獲得技術支援：

- 網址: www.apusic.com
- 電話: 400-855-5800
- 信箱: support@apusic.com
- 金蝶雲社群: <https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

您在取得技術支援時，請提供如下資訊：

1. 您的姓名
2. 公司資訊與聯絡方式
3. 作業系統及其版本
4. 產品版本號
5. 出現異常及錯誤的日誌、截圖等詳細資訊

2 系統層面優化

2.1 作業系統優化

2.1.1 檔案描述符

```
# 查看目前限制
ulimit -n

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

2.1.2 核心參數

```
# /etc/sysctl.conf

# 增加連接埠範圍
net.ipv4.ip_local_port_range = 1024 65535

# TCP 連線優化
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_fin_timeout = 30
net.core.somaxconn = 65535

# 記憶體優化
vm.swappiness = 10
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10

# 應用配置
sysctl -p
```

2.1.3 磁碟優化

優化項	建議
檔案系統	使用 XFS 檔案系統
掛載選項	noatime,nodiratime
磁碟排程	使用 noop 或 deadline
RAID 配置	使用 RAID 10 或 RAID 0
SSD 優化	啟用 TRIM, 預留 over-provisioning

查看磁碟排程演算法

```
cat /sys/block/sda/queue/scheduler
```

設定為 deadline

```
echo deadline > /sys/block/sda/queue/scheduler
```

2.2 網路優化

網路緩衝區

```
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 優化

```
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 配置優化

3.1 記憶體管理

參數	預設值	建議值	說明
vm_memory_high_watermark.relative	0.4	0.6-0.7	記憶體使用上限比例
vm_memory_high_watermark_paging_ratio	0.5	0.5	開始分頁的記憶體比例
vm_memory_high_watermark.absolute	-	4G	絕對記憶體上限

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

3.2 磁碟管理

參數	預設值	建議值	說明
disk_free_limit.relative	2.0	1.5	磁碟空間下限比例
disk_free_limit.absolute	50MB	1GB	絕對磁碟空間下限

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

3.3 連線與通道優化

參數	預設值	建議值	說明
channel_max	2047	根據需求調整	每個連線的最大通道數
connection_max	infinity	根據需求調整	最大連線數
heartbeat	60	30-60	心跳間隔（秒）
handshake_timeout	10000	10000	握手超時（毫秒）

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

3.4 佇列優化

3.4.1 佇列類型選擇

佇列類型	吞吐量	可靠性	適用場景
Classic Queue	中等	中（需配置鏡像）	一般業務場景
Quorum Queue	中等	高	高可用場景
Stream Queue	極高	高	高吞吐、日誌場景

3.4.2 佇列參數優化

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

# 佇列長度限制，防止無限堆積
args.put("x-max-length", 100000);
args.put("x-max-length-bytes", 1073741824); # 1GB

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

# 佇列 TTL
args.put("x-expires", 7200000); # 2小時無消費者後刪除

# 惰性佇列（減少記憶體使用）
args.put("x-queue-mode", "lazy");

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

3.5 訊息持久化優化

3.5.1 何時使用持久化

場景	建議
關鍵業務訊息	必須持久化 (delivery_mode=2)
日誌訊息	可非持久化
高吞吐場景	權衡可靠性和效能

3.5.2 批量持久化

使用 Confirm 模式 + 批量確認提高持久化效能：

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

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

# 等待全部確認
channel.waitForConfirmsOrDie(5000);
```

4 用戶端優化

4.1 連線管理

4.1.1 連線池

```
# 使用連線池管理連線
ConnectionFactory factory = new ConnectionFactory();
factory.setHost("localhost");
factory.setConnectionTimeout(30000);
factory.setRequestedHeartbeat(30);
factory.setAutomaticRecoveryEnabled(true);

# 單連線多通道 (推薦)
Connection connection = factory.newConnection();
for (int i = 0; i < 10; i++) {
    Channel channel = connection.createChannel();
    # 使用 channel 處理訊息
}
```

4.1.2 自動復原

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

4.2 生產者優化

4.2.1 批量發送

```
# 批量發送訊息
List<Message> messages = new ArrayList<>();
for (int i = 0; i < 1000; i++) {
```

```

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

    # 使用交易或 Confirm 模式保證可靠性
    channel.confirmSelect();
    for (Message msg : messages) {
        channel.basicPublish(EXCHANGE, ROUTING_KEY, null,
            msg.toBytes());
    }
    channel.waitForConfirmsOrDie();

```

4.2.2 非同步發送

```

    # 使用執行緒池非同步發送
    ExecutorService executor = Executors.newFixedThreadPool(10);
    for (Message msg : messages) {
        executor.submit(() -> {
            channel.basicPublish(EXCHANGE, ROUTING_KEY, null,
                msg.toBytes());
        });
    }

```

4.3 消費者優化

4.3.1 QoS 預取

```

    # 設定 QoS, 控制未確認訊息數量
    channel.basicQos(100); # 每個消費者最多 100 條未確認訊息

    # 全域 QoS
    channel.basicQos(0, 100, true); # 整個通道最多 100 條

```

4.3.2 批量確認

消費訊息並批量確認

```
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);
    }
}
```

批量確認

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

4.3.3 多消費者

建立多個消費者並行處理

```
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 叢集優化

5.1 節點部署策略

部署策略	說明	適用場景
同機房部署	所有節點在同一機房	低延遲、高吞吐
跨機房部署	節點分佈在不同機房	高可用、容災
異構部署	不同配置節點混合	成本優化

5.2 鏡像佇列優化

```
# 優化鏡像同步策略
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 優化

```
# 調整 Raft 參數
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 優化

```
Map<String, Object> args = new HashMap<>();  
# 段檔案大小  
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 監控與調優

6.1 效能指標

指標	健康範圍	優化建議
CPU 使用率	< 70%	擴容或優化業務邏輯
記憶體使用率	< vm_memory_high_watermark	增加記憶體或優化佇列配置
磁碟使用率	< 80%	擴容或清理資料
訊息入速率	根據業務需求	優化生產者或擴容
訊息出速率	接近入速率	優化消費者或增加消費者
未確認訊息	< QoS 預取數	優化消費者處理速度
連線數	< connection_max	優化連線使用或使用連線池

6.2 效能測試

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

# 參數說明
# -x: 生產者數量
# -y: 消費者數量
# -u: 佇列名稱
# -a: 自動確認
# --auto-delete: 是否自動刪除佇列
```

6.3 瓶頸分析

1. CPU 瓶頸：訊息處理邏輯複雜、加密解密、大量連線
2. 記憶體瓶頸：佇列堆積、大量連線、未啟用惰性佇列
3. 磁碟瓶頸：大量持久化訊息、磁碟 I/O 不足
4. 網路瓶頸：跨機房部署、大量資料傳輸

7 常見問題優化

7.1 訊息堆積優化

1. 增加消費者數量
2. 優化消費者處理邏輯
3. 設定佇列長度限制
4. 使用死信佇列處理過期訊息

7.2 記憶體不足優化

1. 增加實體記憶體
2. 調整 vm_memory_high_watermark
3. 啟用惰性佇列
4. 清理不必要的佇列和訊息
5. 減少連線數

7.3 高延遲優化

1. 使用自動確認（非關鍵訊息）
2. 減少持久化頻率
3. 優化網路配置
4. 使用更近的節點
5. 減少訊息大小

8 效能優化檢查清單

8.1 系統層面

- ☐ 檔案描述符限制已調整
- ☐ 核心參數已優化
- ☐ 磁碟使用 XFS 檔案系統
- ☐ 網路參數已優化
- ☐ 系統資源充足（CPU、記憶體、磁碟）

8.2 RabbitMQ 配置

- ☐ 記憶體水位線合理設定
- ☐ 磁碟空間限制合理設定
- ☐ 心跳間隔適當
- ☐ 連線和通道數限制合理
- ☐ 佇列 Leader 分配策略適當

8.3 用戶端優化

- ☐ 使用連線池
- ☐ 使用單連線多通道
- ☐ 啟用自動復原
- ☐ 合理設定 QoS
- ☐ 使用批量確認
- ☐ 選擇合適的確認模式

8.4 叢集優化

- ☐ 節點選擇合適的佇列類型
- ☐ 鏡像佇列策略合理
- ☐ Quorum Queue 參數優化
- ☐ Stream Queue 保留策略合理
- ☐ 叢集規模滿足業務需求

全國統一服務熱線
4008-555-800

金蝶天燕雲計算股份有限公司（簡稱“金蝶天燕”）成立於2000年，前身為“金蝶中間件公司”，是金蝶集團旗下基礎軟件平台提供商，雲計算國家標準制定企業，國家信創產業核心軟件企業。金蝶天燕是國家863重點研發計劃與核高基重大專項承接企業，也是“兩網一站四庫十二金”國家重點工程的基礎平台提供商，產品廣泛應用於政府、軍工、金融、能源等關鍵行業，累計服務客戶總數超過10萬家。



雲計算國家標準制定企業
金蝶集團旗下基礎軟件企業
資訊技術應用創新核心企業
官網：www.apusic.com

