



效能調優

金蝶Apusic分布式消息队列V2.0.6_for_kafka

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 相關文件
 - 1.3 技術支援
- 2 概述
- 3 硬體與系統最佳化
 - 3.1 磁碟選型
 - 3.2 作業系統參數調優
 - 3.3 網卡最佳化
- 4 Broker 端最佳化
 - 4.1 網路和執行緒配置
 - 4.2 日誌儲存配置
 - 4.3 副本和可靠性配置
 - 4.4 高吞吐場景配置
- 5 生產者最佳化
 - 5.1 核心配置參數
 - 5.2 高吞吐配置
 - 5.3 分割區策略最佳化
- 6 消費者最佳化
 - 6.1 核心配置參數
 - 6.2 高吞吐配置
 - 6.3 消費者多執行緒最佳化
- 7 JVM 調優
 - 7.1 記憶體配置
 - 7.2 GC 調優推薦配置
 - 7.3 常見問題排查
- 8 Topic 設計最佳化
 - 8.1 分割區數設計
 - 8.2 分割區再平衡
 - 8.3 日誌壓縮配置
- 9 監控與診斷
 - 9.1 關鍵效能指標

- 9.2 使用 JMX 監控
- 9.3 使用 Prometheus 監控
- 10 常見效能問題
- 11 效能測試建議
 - 11.1 測試工具
 - 11.2 測試命令

1 前言

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

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、維運工程師。

1.2 相關文件

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

序號	手冊文件	說明
1	金蝶Apusic分散式訊息佇列for Kafka 快速使用手冊	簡單介紹如何快速上手使用ADMQ for Kafka。
2	金蝶Apusic分散式訊息佇列for Kafka 安裝手冊	詳細介紹如何在各作業系統上安裝ADMQ for Kafka，以及ADMQ for Kafka服務啟停等操作。
3	金蝶Apusic分散式訊息佇列for Kafka 訊息引擎使用手冊	詳細介紹 ADMQ for Kafka 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 Kafka 管控台使用手冊	詳細介紹ADMQ for Kafka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 Kafka 開發手冊	詳細介紹基於各開發語言進行ADMQ for Kafka客戶端應用開發的說明。
6	金蝶Apusic分散式訊息佇列for Kafka 遷移手冊	詳細介紹從Kafka遷移到ADMQ for Kafka的說明。
7	金蝶Apusic分散式訊息佇列for Kafka 維運手冊	詳細介紹ADMQ for Kafka的監控、維運、安全強化等維運說明。
8	金蝶Apusic分散式訊息佇列for Kafka 效能優化手冊	詳細介紹ADMQ for Kafka效能調優的說明。

1.3 技術支援

ADMQ for Kafka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 概述

效能優化需要從多個層面綜合考慮：

- 硬體層面：CPU、記憶體、磁碟、網路
- 系統層面：作業系統參數、核心調優
- Broker 層面：配置參數最佳化
- 客戶端層面：生產者和消費者配置
- 架構層面：Topic 設計、分割區策略

3 硬體與系統最佳化

3.1 磁碟選型

磁碟類型	推薦場景	說明
NVMe SSD	高吞吐場景	最佳效能，適合核心業務
SSD	生產環境	優於 HDD，推薦使用
HDD	低成本場景	僅適用於日誌類場景

最佳化建議：

- 使用獨立的磁碟作為資料目錄
- 啟用磁碟 RAID 0/10 提高效能
- 避免使用網路儲存（NFS）作為主要資料目錄

3.2 作業系統參數調優

```
# 檔案描述符限制
ulimit -n 1000000

# 網路參數最佳化
sysctl -w net.core.rmem_max=67108864
sysctl -w net.core.wmem_max=67108864
sysctl -w net.ipv4.tcp_rmem="4096 87380 67108864"
sysctl -w net.ipv4.tcp_wmem="4096 16384 67108864"
sysctl -w net.core.netdev_max_backlog=250000

# 禁用 swap
sysctl -w vm.swappiness=0

# 檔案系統最佳化
mount -o noatime,nodiratime /dev/sda1 /mnt/kafka
```

3.3 網卡最佳化

- 使用多佇列網卡並綁定 CPU
- 啟用 jumbo frames (MTU 9000)
- 考慮使用 DPDK 加速（高吞吐場景）

4 Broker 端最佳化

4.1 網路和執行緒配置

參數	預設值	推薦值	說明
<code>num.network.threads</code>	8	CPU 核心數	處理網路請求的執行緒數
<code>num.io.threads</code>	16	磁碟數×2	處理磁碟 IO 的執行緒數
<code>socket.send.buffer.bytes</code>	102400	204800	傳送緩衝區大小
<code>socket.receive.buffer.bytes</code>	102400	204800	接收緩衝區大小
<code>queued.max.requests</code>	500	1000	等待 IO 的請求佇列長度

4.2 日誌儲存配置

參數	預設值	推薦值	說明
<code>log.segment.bytes</code>	1GB	512MB	日誌段大小，較小的段加快日誌清理
<code>log.retention.hours</code>	168	根據業務調整	日誌保留時間
<code>log.retention.check.interval.ms</code>	300000	60000	日誌清理檢查間隔
<code>log.flush.interval.ms</code>	null	不建議修改	建議依賴 PageCache
<code>log.flush.scheduler.interval.ms</code>	null	不建議修改	建議依賴 PageCache

4.3 副本和可靠性配置

參數	推薦值	說明
<code>default.replication.factor</code>	3	生產環境建議 3 副本
<code>min.insync.replicas</code>	2	配合 <code>acks=all</code> 使用
<code>unclean.leader.election.enable</code>	false	禁止非 ISR 選舉
<code>replica.lag.time.max.ms</code>	30000	ISR 踢出閾值

4.4 高吞吐場景配置

網路最佳化

```
num.network.threads=16
num.io.threads=32
socket.send.buffer.bytes=307200
socket.receive.buffer.bytes=307200
queued.max.requests=2000
```

日誌最佳化

```
log.segment.bytes=524288000
log.retention.check.interval.ms=300000
```

壓縮最佳化

```
compression.type=zstd
```

副本最佳化

```
replica.socket.timeout.ms=30000
replica.fetch.max.bytes=1048576
replica.fetch.min.bytes=1
```

5 生產者最佳化

5.1 核心配置參數

參數	推薦值	說明
acks	all	可靠性優先時使用
compression.type	zstd	ZSTD 壓縮比最高
batch.size	16384-32768	批次大小（位元組）
linger.ms	5-20	批次等待時間
buffer.memory	64MB	傳送緩衝區
max.in.flight.requests.per.connection	5	冪等性啟用時≤5
retries	Integer.MAX_VALUE	重試次數
enable.idempotence	true	啟用冪等性

5.2 高吞吐配置

```

Properties props = new Properties();
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"192.168.1.10:9092");
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());

// 可靠性配置
props.put(ProducerConfig.ACKS_CONFIG, "all");
props.put(ProducerConfig.ENABLE_IDEMPOTENCE_CONFIG, true);

// 吞吐最佳化
props.put(ProducerConfig.COMPRESSION_TYPE_CONFIG, "zstd");
props.put(ProducerConfig.BATCH_SIZE_CONFIG, 32768);

```

```

props.put(ProducerConfig.LINGER_MS_CONFIG, 10);
props.put(ProducerConfig.BUFFER_MEMORY_CONFIG, 67108864);
props.put(ProducerConfig.MAX_IN_FLIGHT_REQUESTS_PER_CONNECTION, 5);
props.put(ProducerConfig.REQUEST_TIMEOUT_MS_CONFIG, 30000);
props.put(ProducerConfig.DELIVERY_TIMEOUT_MS_CONFIG, 120000);

// 重試配置
props.put(ProducerConfig.RETRIES_CONFIG, Integer.MAX_VALUE);
props.put(ProducerConfig.RETRY_BACKOFF_MS_CONFIG, 100);

```

5.3 分割區策略最佳化

1. Key 分割區：使用業務 Key 實現順序保證
2. 輪詢分割區：實現負載均衡
3. 自訂分割器：根據業務邏輯分割

```

// 自訂分割器範例
public class CustomPartitioner implements Partitioner {
    @Override
    public int partition(String topic, Object key, byte[] keyBytes,
                        Object value, byte[] valueBytes, Cluster
cluster) {
        // 根據業務邏輯實現分割
        return Math.abs(key.hashCode()) %
cluster.partitionCountFor(topic);
    }
}

```

6 消費者最佳化

6.1 核心配置參數

參數	推薦值	說明
fetch.min.bytes	1	最小獲取位元組數
fetch.max.wait.ms	500	最大等待時間
max.poll.records	500	單次 poll 最大記錄數
session.timeout.ms	30000	會話逾時
heartbeat.interval.ms	10000	心跳間隔
enable.auto.commit	false	手動提交更可靠
auto.offset.reset	earliest	從頭消費
max.partition.fetch.bytes	1048576	每個分割區最大獲取位元組

6.2 高吞吐配置

```
Properties props = new Properties();
props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"192.168.1.10:9092");
props.put(ConsumerConfig.GROUP_ID_CONFIG, "my-consumer-group");
props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());

// 吞吐最佳化
props.put(ConsumerConfig.FETCH_MIN_BYTES_CONFIG, 1);
props.put(ConsumerConfig.FETCH_MAX_WAIT_MS_CONFIG, 500);
props.put(ConsumerConfig.MAX_POLL_RECORDS_CONFIG, 1000);
props.put(ConsumerConfig.MAX_PARTITION_FETCH_BYTES_CONFIG,
10485760);
```

```
// 可靠性配置
props.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, false);
props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
props.put(ConsumerConfig.SESSION_TIMEOUT_MS_CONFIG, 30000);
props.put(ConsumerConfig.HEARTBEAT_INTERVAL_MS_CONFIG, 10000);

// 拉取最佳化
props.put(ConsumerConfig.DEFAULT_API_TIMEOUT_MS_CONFIG, 60000);
```

6.3 消費者多執行緒最佳化

```
// 多消費者執行緒範例
ExecutorService executor = Executors.newFixedThreadPool(4);
for (int i = 0; i < 4; i++) {
    final int threadId = i;
    executor.submit(() -> {
        KafkaConsumer<String, String> consumer = createConsumer();
        consumer.subscribe(Arrays.asList("my-topic"));
        while (true) {
            ConsumerRecords<String, String> records =
consumer.poll(100);
            for (ConsumerRecord<String, String> record : records) {
                processRecord(record);
            }
            consumer.commitSync();
        }
    });
}
```

7 JVM 調優

7.1 記憶體配置

場景	堆積記憶體	GC 策略
開發/測試	-Xms2g -Xmx2g	G1
生產（小型叢集）	-Xms8g -Xmx8g	G1
生產（大型叢集）	-Xms16g -Xmx16g	G1

7.2 GC 調優推薦配置

```
# G1 GC 配置
KAFKA_HEAP_OPTS="-Xms16g -Xmx16g -XX:+UseG1GC \
  -XX:MaxGCPauseMillis=20 \
  -XX:InitiatingHeapOccupancyPercent=35 \
  -XX:G1HeapRegionSize=16m \
  -XX:+ParallelRefProcEnabled"

# JMX 配置（用於監控）
KAFKA_JMX_OPTS="-Dcom.sun.management.jmxremote \
  -Dcom.sun.management.jmxremote.port=9999 \
  -Dcom.sun.management.jmxremote.ssl=false \
  -Dcom.sun.management.jmxremote.authenticate=false"
```

7.3 常見問題排查

問題	可能原因	解決方案
頻繁 Full GC	堆積記憶體不足	增加堆積記憶體，最佳化消費邏輯
GC 停頓過長	G1 配置不當	調整 MaxGCPauseMillis
OOM	記憶體洩漏或配置錯誤	檢查 JVM 配置和客戶端配置

8 Topic 設計最佳化

8.1 分割區數設計

目標吞吐量	推薦分割區數	說明
10萬/秒	10-20	單分割區約 5-10萬/秒
50萬/秒	50-100	需要更多 Broker
100萬/秒	100-200	需要大型叢集

計算公式：

分割區數 = 目標吞吐量 / 單分割區吞吐量
單分割區吞吐量 $\approx$ 5-10萬/秒（取決於訊息大小和硬體）

8.2 分割區再平衡

分割區增加會導致再平衡，影響效能。建議：

- 提前規劃足夠的分割區數
- 使用黏性分割區策略（Kafka 2.4+）

```
# 黏性分割區策略（預設）  
kafka-topics.sh --alter --topic my-topic --partitions 20
```

8.3 日誌壓縮配置

```
# 日誌壓縮配置  
cleanup.policy=compact  
min.compaction.lag.ms=3600000  
delete.retention.ms=86400000  
segment.bytes=1073741824
```

9 監控與診斷

9.1 關鍵效能指標

指標	說明	告警閾值
MessagesInPerSec	每秒訊息數	根據業務設定
BytesInPerSec	每秒入站位元組	根據業務設定
BytesOutPerSec	每秒出站位元組	根據業務設定
UnderReplicatedPartitions	未同步分割區數	> 0
OfflinePartitionsCount	離線分割區數	> 0
RequestLatencyAvg	平均請求延遲	> 100ms
FetchQueueSize	請求佇列大小	持續增長
PurgatorySize	等待中的請求	持續增長

9.2 使用 JMX 監控

```
# 開啟 JMX 埠
JMX_PORT=9999 bin/admq-daemon start kafka standalone

# 使用 jconsole 連線
jconsole localhost:9999
```

9.3 使用 Prometheus 監控

配置 Prometheus 採集指標：

```
scrape_configs:
  - job_name: 'kafka'
    static_configs:
      - targets: ['localhost:12305']
```

10 常見效能問題

問題	症狀	解決方案
生產者延遲高	傳送耗時增加	增加 batch.size，減少 linger.ms
消費者 Lag 大	消費速度慢	增加消費者數量，最佳化處理邏輯
磁碟 IO 高	寫入變慢	使用 SSD，最佳化日誌保留策略
網路瓶頸	吞吐量低	提升網路頻寬，使用壓縮
記憶體壓力	GC 頻繁	增加堆積記憶體，調整 GC 參數

11 效能測試建議

11.1 測試工具

- kafka-producer-perf-test.sh: 生產者效能測試
- kafka-consumer-perf-test.sh: 消費者效能測試
- kafka-throttle.sh: 限流測試

11.2 測試命令

生產者效能測試

```
kafka/bin/kafka-producer-perf-test.sh \  
  --topic test-topic \  
  --num-records 1000000 \  
  --record-size 1024 \  
  --throughput -1 \  
  --producer-props bootstrap.servers=192.168.1.10:9092
```

消費者效能測試

```
kafka/bin/kafka-consumer-perf-test.sh \  
  --topic test-topic \  
  --messages 1000000 \  
  --group perf-test-group \  
  --bootstrap-server 192.168.1.10:9092
```

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。

Apusic
金蝶天燕

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

