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_kafka

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

本文档介绍金蝶Apusic分布式消息队列for Kafka（Apusic Distributed Message Queue，简称：ADMQ for Kafka）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 Kafka产品相关的信息，请参阅以下ADMQ for Kafka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 Kafka 快速使用手册	简单介绍了如何快速上手使用ADMQ for Kafka 。
2	金蝶Apusic分布式消息队列for Kafka 安装手册	详细介绍如何在各操作系统上安装ADMQ for Kafka，以及ADMQ for Kafka服务启停等操作。
3	金蝶Apusic分布式消息队列for Kafka 消息引擎用户手册	详细介绍 ADMQ for Kafka 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 Kafka 管控台用户手册	详细介绍ADMQ for Kafka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 Kafka 开发手册	详细介绍基于各开发语言进行ADMQ for Kafka客户端应用开发的说明。
6	金蝶Apusic分布式消息队列for Kafka 迁移手册	详细介绍从Kafka迁移到ADMQ for Kafka的说明。
7	金蝶Apusic分布式消息队列for Kafka 运维手册	详细介绍ADMQ for Kafka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 Kafka 性能优化手册	详细介绍ADMQ for Kafka性能调优的说明。

1.3 技术支持

ADMQ for Kafka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 概述

性能优化需要从多个层面综合考虑：

- 硬件层面：CPU、内存、磁盘、网络
- 系统层面：操作系统参数、内核调优
- Broker 层面：配置参数优化
- 客户端层面：生产者和消费者配置
- 架构层面：Topic 设计、分区策略

3 硬件与系统优化

3.1 磁盘选型

磁盘类型	推荐场景	说明
NVMe SSD	高吞吐场景	最佳性能，适合核心业务
SSD	生产环境	优于 HDD，推荐使用
HDD	低成本场景	仅适用于日志类场景

优化建议：

- 使用独立的磁盘用于数据目录
- 启用磁盘 RAID 0/10 提高性能
- 避免使用网络存储（NFS）作为主要数据目录

3.2 操作系统参数调优

```
# 文件描述符限制
ulimit -n 1000000

# 网络参数优化
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

# 文件系统优化
mount -o noatime,nodiratime /dev/sda1 /mnt/kafka
```

3.3 网卡优化

- 使用多队列网卡并绑定 CPU
- 启用 jumbo frames (MTU 9000)
- 考虑使用 DPDK 加速（高吞吐场景）

4 Broker 端优化

4.1 网络和线程配置

参数	默认值	推荐值	说明
num.network.threads	8	CPU 核数	处理网络请求的线程数
num.io.threads	16	磁盘数×2	处理磁盘 IO 的线程数
socket.send.buffer.bytes	102400	204800	发送缓冲区大小
socket.receive.buffer.bytes	102400	204800	接收缓冲区大小
queued.max.requests	500	1000	等待 IO 的请求队列长度

4.2 日志存储配置

参数	默认值	推荐值	说明
log.segment.bytes	1GB	512MB	日志段大小，较小的段加快日志清理
log.retention.hours	168	根据业务调整	日志保留时间
log.retention.check.interval.ms	300000	60000	日志清理检查间隔
log.flush.interval.ms	null	不建议修改	建议依赖 PageCache
log.flush.scheduler.interval.ms	null	不建议修改	建议依赖 PageCache

4.3 副本和可靠性配置

参数	推荐值	说明
default.replication.factor	3	生产环境建议 3 副本
min.insync.replicas	2	配合 acks=all 使用
unclean.leader.election.enable	false	禁止非 ISR 选举
replica.lag.time.max.ms	30000	ISR 踢出阈值

4.4 高吞吐场景配置

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

# 日志优化
log.segment.bytes=524288000
log.retention.check.interval.ms=300000

# 压缩优化
compression.type=zstd

# 副本优化
replica.socket.timeout.ms=30000
replica.fetch.max.bytes=1048576
replica.fetch.min.bytes=1
```

5 生产者优化

5.1 核心配置参数

参数	推荐值	说明
acks	all	可靠性优先时使用
compression.type	zstd	ZSTD 压缩比最高
batch.size	16384-32768	批量大小（字节）
linger.ms	5-20	批量等待时间
buffer.memory	64MB	发送缓冲区
max.in.flight.requests.per.connection	5	幂等性启用时≤5
retries	Integer.MAX_VALUE	重试次数
enable.idempotence	true	启用幂等性

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);

// 吞吐优化
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);

// 重试配置
props.put(ProducerConfig.RETRIES_CONFIG, Integer.MAX_VALUE);
props.put(ProducerConfig.RETRY_BACKOFF_MS_CONFIG, 100);
```

5.3 分区策略优化

1. **Key 分区**：使用业务 Key 实现顺序保证
2. **轮询分区**：实现负载均衡
3. **自定义分区器**：根据业务逻辑分区

```
// 自定义分区器示例
public class CustomPartitioner implements Partitioner {
    @Override
    public int partition(String topic, Object key, byte[] keyBytes,
                         Object value, byte[] valueBytes, Cluster
cluster) {
        // 根据业务逻辑实现分区
        return Math.abs(key.hashCode()) %
cluster.partitionCountFor(topic);
    }
}
```

6 消费者优化

6.1 核心配置参数

参数	推荐值	说明
fetch.min.bytes	1	最小获取字节数
fetch.max.wait.ms	500	最大等待时间
max.poll.records	500	单次 poll 最大记录数
session.timeout.ms	30000	会话超时
heartbeat.interval.ms	10000	心跳间隔
enable.auto.commit	false	手动提交更可靠
auto.offset.reset	earliest	从头消费
max.partition.fetch.bytes	1048576	每个分区最大获取字节

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());

// 吞吐优化
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);

// 拉取优化
props.put(ConsumerConfig.DEFAULT_API_TIMEOUT_MS_CONFIG, 60000);
```

6.3 消费者多线程优化

```
// 多消费者线程示例
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 调优

7.1 内存配置

场景	堆内存	GC 策略
开发/测试	-Xms2g -Xmx2g	G1
生产（小集群）	-Xms8g -Xmx8g	G1
生产（大集群）	-Xms16g -Xmx16g	G1

7.2 GC 调优推荐配置

```
# 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（用于监控）
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 常见问题排查

问题	可能原因	解决方案
频繁 Full GC	堆内存不足	增加堆内存，优化消费逻辑
GC 停顿过长	G1 配置不当	调整 MaxGCPauseMillis
OOM	内存泄漏或配置错误	检查 JVM 配置和客户端配置

8 Topic 设计优化

8.1 分区数设计

目标吞吐量	推荐分区数	说明
10万/秒	10-20	单分区约 5-10万/秒
50万/秒	50-100	需要更多 Broker
100万/秒	100-200	需要大集群

计算公式：

```
分区数 = 目标吞吐量 / 单分区吞吐量
单分区吞吐量 ≈ 5-10万/秒（取决于消息大小和硬件）
```

8.2 分区再均衡

分区增加会导致再均衡，影响性能。建议：

- 提前规划足够的分区数
- 使用黏性分区策略（Kafka 2.4+）

```
# 黏性分区策略（默认）
kafka-topics.sh --alter --topic my-topic --partitions 20
```

8.3 日志压缩配置

```
# 日志压缩配置
cleanup.policy=compact
min.compaction.lag.ms=3600000
delete.retention.ms=86400000
segment.bytes=1073741824
```

9 监控与诊断

9.1 关键性能指标

指标	说明	告警阈值
MessagesInPerSec	每秒消息数	根据业务设定
BytesInPerSec	每秒入站字节	根据业务设定
BytesOutPerSec	每秒出站字节	根据业务设定
UnderReplicatedPartitions	未同步分区数	> 0
OfflinePartitionsCount	离线分区数	> 0
RequestLatencyAvg	平均请求延迟	> 100ms
FetchQueueSize	请求队列大小	持续增长
PurgatorySize	等待中的请求	持续增长

9.2 使用 JMX 监控

```
# 开启 JMX 端口
JMX_PORT=9999 bin/admq-daemon start kafka standalone

# 使用 jconsole 连接
jconsole localhost:9999
```

9.3 使用 Prometheus 监控

配置 Prometheus 采集指标：

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

10 常见性能问题

问题	症状	解决方案
生产者延迟高	发送耗时增加	增加 batch.size，减少 linger.ms
消费者 Lag 大	消费速度慢	增加消费者数量，优化处理逻辑
磁盘 IO 高	写入变慢	使用 SSD，优化日志保留策略
网络瓶颈	吞吐量低	提升网络带宽，使用压缩
内存压力	GC 频繁	增加堆内存，调整 GC 参数

11 性能测试建议

11.1 测试工具

- kafka-producer-perf-test.sh：生产者性能测试
- kafka-consumer-perf-test.sh：消费者性能测试
- kafka-throttle.sh：限流测试

11.2 测试命令

```
# 生产者性能测试
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

# 消费者性能测试
kafka/bin/kafka-consumer-perf-test.sh \
  --topic test-topic \
  --messages 1000000 \
  --group perf-test-group \
  --bootstrap-server 192.168.1.10:9092
```

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