Apusic 金蝶天燕

生成时间：2026-05-07

性能调优说明文档

金蝶Apusic分布式消息队列V2.0.6

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 版本更新说明

- 1.1 一、集群建议配置
  - 1.1.1 1.1硬件配置
    - 1.1.1.1 1.1.1基准测试平台配置
    - 1.1.1.2 1.1.2生产环境最小配置
    - 1.1.1.3 1.1.3生产环境推荐配置
  - 1.1.2 1.2存储路径配置优化
    - 1.1.2.1 1.2.1 存储节点
    - 1.1.2.2 1.2.2 协调器
  - 1.1.3 1.3 内存分配策略
    - 1.1.3.1 1.3.1整体的推荐分配策略如下：
    - 1.1.3.2 1.3.2 jvm参数设置
- 1.2 二、性能调优
  - 1.2.1 2.1系统层面
    - 1.2.1.1 2.1.1内核参数优化
    - 1.2.1.2 2.1.2资源限制
  - 1.2.2 2.2引擎层面
    - 1.2.2.1 2.2.1 协调器
    - 1.2.2.2 2.2.2 存储节点
    - 1.2.2.3 2.2.3 计算节点
    - 1.2.2.4 2.2.4 rabbitmq插件
    - 1.2.2.5 2.2.5 kafka插件
    - 1.2.2.6 2.2.6 mqtt插件
- 1.3 三、场景化调优
  - 1.3.1 3.1 高吞吐写入场景
    - 1.3.1.1 jvm:
  - 1.3.2 3.2 低延迟读取场景
    - 1.3.2.1 jvm:
  - 1.3.3 3.3 追赶读取场景
    - 1.3.3.1 jvm:
  - 1.3.4 3.4 磁盘空间不足场景
  - 1.3.5 3.5 运行内存不足场景

- 1.4 四、性能问题分析思路
  - 1.4.1 4.1 storage节点提示Not enough non-faulty bookies available报错信息
  - 1.4.2 4.2 broker节点提示Java heap space报错信息
  - 1.4.3 4.3 MQTT性能测试中出现 "too many open files"报错信息
  - 1.4.4 4.4 存储节点写入延迟高，出现 "Write rate limiting" 警告**

1 版本更新说明

本文档最新版本包含历史修改记录如下：

更新日期	手册版本	适用产品	更新说明
2025年12月	V1.0	金蝶Apusic分布式消息队列V2.0.6	首次编写

1.1 一、集群建议配置

1.1.1 1.1硬件配置

1.1.1.1 1.1.1基准测试平台配置

组件	配置	实例数	说明
计算节点（broker）和存储节点(storage)	16C64G500G 10Gbps/1000Mbps	3	配置分别对应CPU、内存、硬盘、网卡，其中CPU为vCPU，硬盘为HDD，网卡分为内外网
协调器（ZooKeeper）	4C16G500G 10Gbps/1000Mbps	3	配置分别对应CPU、内存、硬盘、网卡，其中CPU为vCPU，硬盘为HDD，网卡分为内外网

1.1.1.2 1.1.2生产环境最小配置

组件	配置		
计算节点（broker）和存储节点(storage)	16C64G1TG 10Gbps	3	配置分别对应CPU、内存、硬盘、网卡，其中CPU为vCPU，硬盘推荐NVMe SSD
协调器（ZooKeeper）	4C16G200G 10Gbps	3	配置分别对应CPU、内存、硬盘、网卡，其中CPU为vCPU，硬盘可为SSD或HDD

1.1.1.3 1.1.3生产环境推荐配置

组件	配置	实例数	说明

计算节点（broker）和存储节点(storage)	32C128G2T 10~25Gbps	3~5	配置分别对应CPU、内存、硬盘、网卡，其中CPU为vCPU，硬盘推荐NVMe SSD
协调器（ZooKeeper）	8C32G500G 10~25Gbps	3	配置分别对应CPU、内存、硬盘、网卡，其中CPU为vCPU，硬盘可为SSD或HDD

> *提示*
>
> *计算节点和存储节点如果是分开部署，基准测试平台和生产环境最小CPU为8核、内存为32G即可*

1.1.2 1.2存储路径配置优化

1.1.2.1 1.2.1 存储节点

```
#在存储节点的安装路径下./config/storage.conf
#记录 Bookie 所有写操作的事务日志,单独SSD可以提升事务日志性能
journalDirectories=/ssd/journal01
#存储实际的消息数据
ledgerDirectories=/hdd/ledgers01
```

1.1.2.2 1.2.2 协调器

```
#在协调器的安装路径下./config/zookeeper.conf
#存储 ZooKeeper 的快照文件
dataDir=/data/zookeeper/data
#存储 ZooKeeper 的事务日志,单独SSD可以提升事务日志性能
dataLogDir=/ssd/zookeeper/data_log
```

1.1.3 1.3 内存分配策略

1.1.3.1 1.3.1整体的推荐分配策略如下：

- OS预留：1 ~ 2 GB
- JVM内存：1/2 （其中堆内存（Heap） ：1/3，直接内存（Off-Heap）：2/3）
- PageCache：1/2

推荐分配策略示例：

服务器内存为32GB，且Broker和Storage 混合部署方案：

- OS保留：2GB

- Broker JVM：7GB（堆内存2GB + 直接内存5GB）
- Storage JVM：8GB（堆内存2GB + 直接内存6GB）
- PageCache：15GB

服务器内存为64GB，且Broker和Storage 混合部署方案：

- OS保留：2GB
- Broker JVM：15GB（堆内存5GB + 直接内存10GB）
- Storage JVM：16GB（堆内存5GB + 直接内存11GB）
- PageCache：31GB

服务器内存为128GB，且Broker和Storage 混合部署方案：

- OS保留：2GB
- Broker JVM：31GB（堆内存10GB + 直接内存21GB）
- Storage JVM：32GB（堆内存10GB + 直接内存22GB）
- PageCache：63GB

1.1.3.2 1.3.2 jvm参数设置

存储节点：下面以32G内存为例,需要在存储节点安装路径下./config/storage_env.sh中修改参数：

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=5g"}
```

计算节点：下面以32G内存为例,需要在计算节点安装路径下./config/broker_env.sh中修改参数：

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=6g"}
```

1.2 二、性能调优

1.2.1 2.1系统层面

1.2.1.1 2.1.1内核参数优化

在文件 /etc/sysctl.conf 中增加如下内容，执行 sysctl -p 让其生效。

```
vm.dirty_background_ratio=50    # 后台回写阈值
vm.dirty_ratio=50               # 强制回写阈值
vm.dirty_expire_centisecs=12000 # 脏数据最长存在时间
vm.dirty_writeback_centisecs=2000
vm.swappiness=1                 # 最小化交换
```

```
vm.overcommit_memory=1              # 允许内存超分配

vm.drop_caches=1
vm.max_map_count=655360
vm.page-cluster=3
vm.zone_reclaim_mode=0

# 系统全局允许分配的最大文件句柄数
fs.file-max=3245134
```

vm相关参数说明：

选项	默认值	描述
vm.dirty_background_ratio	10	触发pdflunsh后台回写的脏系统内存百分比
vm.dirty_ratio	30	触发一个写入进程开始回写的脏系统内存比例
vm.dirty_expire_centisecs	3000	脏内存符合pdflush条件的最短时间（促进写取消）
vm.dirty_writeback_centisecs	500	pdflush唤醒时间间隔（0为停用）
vm.overcommit_memory	0	0表示利用探索法允许合理的过度分配 1表示一直过度分配；2表示禁止过度分配
vm.swappiness	30	相对于页面缓存回收更倾向用交换（换页）释放内存的程度
vm.drop_caches	0	drop_caches = 1 释放掉PageCache中的clean pages drop_caches = 2 释放掉Slab，包括dentry、inode等 drop_caches = 3 既释放PageCache，又释放Slab
vm.max_map_count	65530	定义了一个进程能拥有的最多的内存区域，默认为 65536
vm.page-cluster	3	每次 swap in 或者 swap out 操作多少内存页，为 2 的指数。 page-cluster=0 表示 1 页；page-cluster=1 表示 2 页 page-cluster=2 表示4 页；page-cluster=3 表示 8 页
vm.zone_reclaim_mode	0	zone_reclaim_mode=0 系统会倾向于从其他节点分配内存 zone_reclaim_mode=1 系统会倾向于从本地节点回收 Cache 内存

1.2.1.2 2.1.2资源限制

修改用户的系统资源限制 /etc/security/limits.conf ， 设置打开的最多文件数、最大进程数和锁定内存地址空间：

```
* soft nofile 655350
* hard nofile 655350

* soft nproc 10240
* hard nproc 10240

* hard memlock unlimited
* soft memlock unlimited
```

1.2.2 2.2引擎层面

1.2.2.1 2.2.1 协调器

配置文件路径：在协调器的安装路径下./config/zookeeper.conf

```
# zookeeper.conf
# 每次tick的毫秒数
tickTime=6000

# 启动过程中从leader同部数据的时间限制
# 如果集群规模大，数据量多的话，适当调大此参数
initLimit=20

#用于配置Leader服务器和Follower之间进行心跳检测的最大延时时间
syncLimit=10

# BookKeeper存储其预写日志的目录，多个目录逗号进行分割，防止线程阻塞
dataDir=/data/zookeeper/data

# 指定存储BookKeeper输出ledger的目录。多个ledger目录，需要使用逗号分割
dataLogDir=/data/zookeeper/data_log

# 为提高性能dataDir和dataLogDir可以配置到不同的磁盘

# leader不接受客户端连接,专注于通信和选举等
leaderServes=no
```

```
# 在压测环境建议设为1000，生产上以压测实际结果为准再做调整或者也不调整
maxClientCnxns=1000

# 预先开辟磁盘空间，用于后续写入事务日志。默认是64M，每个事务日志大小就是64M。
# 如果ZK的快照频率较大的话，建议适当减小这个参数。单位KB。
preAllocSize=131072
```

1.2.2.2 2.2.2 存储节点

配置文件路径：在存储节点的安装路径下./config/storage.conf

```
# storage.conf
# BookKeeper存储其预写日志的目录，多个目录逗号进行分割，防止线程阻塞
journalDirectories=/data/bookkeeper/journal01,/data/bookkeeper/journal02

# 指定存储BookKeeper输出ledger的目录。多个ledger目录，需要使用逗号分割
ledgerDirectories=/data/bookkeeper/ledger01,/data/bookkeeper/ledger02

# 为提高性能journalDirectories和ledgerDirectories可以配置到不同的磁盘

# 当journal盘为HDD时，一般会将journalSyncData开关关闭，让数据写入到 PagaCache 中就返回 ACK，从而降低 entry 写入延迟
journalSyncData=false

# 性能相关配置
writeBufferSizeBytes=1048576
# 写Cache的大小。内存是从JVM直接内存分配的。写入缓存用于在刷新到条目日志之前缓冲条目。为了获得良好的性能，它应该足够大，以在刷新间隔中保存大量条目。
dbStorage_writeCacheMaxSizeMb=256
# 读cache的大小。内存是从JVM直接内存分配的。这个读缓存是预先填充的，每当缓存错过时就会预读。默认情况下，它被分配给可用的直接内存的25%。
dbStorage_readAheadCacheMaxSizeMb=64
```

1.2.2.3 2.2.3 计算节点

配置文件路径：在计算节点的安装路径下./config/broker.conf

```
# broker.conf
# 默认Bundle数量为4，该值应为Broker节点的整数倍
defaultNumberOfNamespaceBundles=12

# 消费确认过的消息超过该大小后会触发删除策略
defaultRetentionSizeInMB=409600

# 消费确认过的消息超过指定时间后触发删除策略
defaultRetentionTimeInMinutes=2880

# 未被消费确认的消息大存储大小（-1表示没有限制，保持默认）
# 如果保持默认需要通过set-message-ttl设置过期时间，防止磁盘爆满
backlogQuotaDefaultLimitGB=-1

# 未被消费确认的消息超过存储大小的策略（保持默认）
backlogQuotaDefaultRetentionPolicy=producer_request_hold

# 创建Ledger时指定Ensemble的大小
managedLedgerDefaultEnsembleSize=2

# 创建Ledger时指定Quorum的大小
managedLedgerDefaultWriteQuorum=2

# 创建Ledger时指定ack Quorum的大小
managedLedgerDefaultAckQuorum=2

# 追赶读，从BookKeeper中读取的最大条目数。缺省值是100条。
dispatcherMaxReadBatchSize=500
```

1.2.2.4 2.2.4 rabbitmq插件

配置文件路径：在计算节点的安装路径下./config/broker.conf

```
# broker.conf
# 一个连接上可以同时存在的最大通道数
amqpMaxNoOfChannels=2048
```

参数解析:

选项	默认值	描述
amqpMaxNoOfChannels	64	一个连接上可以同时存在的最大通道数。允许与客户端协商的最大通道数，不包括协议中使用的特殊信道号0。设置为0意味着“无限”，这是一个危险的值，因为应用程序有时会发生通道泄漏。使用更多的通道会增加代理的内存占用。
amqpMaxFrameSize	4194304 (4MB)	Default: 0 连接上的最大帧长度。与客户端协商的允许最大frame大小，设置为 0 表示无限制，但在某些QPid客户端会引发bug.。设置较大的值可以提高吞吐量;设置一个较小的值可能会提高延迟。
amqpHeartBeat	60 (s)	AoP连接的默认心跳超时。表示服务器在协商连接参数时建议的心跳超时时间的值。如果两端都设置为0，心跳停止(不推荐这样做)。

1.2.2.5 2.2.5 kafka插件

配置文件路径：在计算节点的安装路径下./config/broker.conf

```
# broker.conf
# 消息条目的格式
entryFormat=kafka

# 每次从游标读取的最大条目数
maxReadEntriesNum=20

# 限制请求队列的大小
maxQueuedRequests=1000
```

参数说明:

选项	默认值	描述
entryFormat	pulsar	如果它被设置为' kafka '，就没有不必要的编码和解码工作，这有助于提高性能。但是，在这种情况下，一个主题不能被混合的Pulsar客户机和Kafka客户机使用。如果设置为' mixed_kafka '，则支持一些非官方Kafka客户端实现。 - **注**:与' mixed_kafka '相比，当参数设置为' kafka '时性能提高2 ~ 3倍。
maxReadEntriesNum	5	增加该值可以使FETCH请求每次读取更多字节。 **说明**:Kafka协议处理器目前不检查最大字节限制。因此，如果该值太大，则可能超出内存或网络限制。

maxQueuedRequests	500	限制请求队列的大小，如Kafka服务器中的queuedmax .requests。10G网络设为1000左右。增大queued.max.requests能够缓存更多的请求，以撑过业务峰值。如果过大，会造成内存的浪费。

注：maxReadEntriesNum在基准测试平台设置为20，可根据集群内存或网络配置情况适当调大。

1.2.2.6 2.2.6 mqtt插件

```
# 一个连接上可以同时存在的最大通道数
maxNoOfChannels=64
```

选项	默认值	描述
maxNoOfChannels	1	一个连接上可以同时存在的最大通道数。允许与客户端协商的最大通道数，不包括协议中使用的特殊信道号0。设置为0意味着“无限”，这是一个危险的值，因为应用程序有时会发生通道泄漏。使用更多的通道会增加代理的内存占用。
maxFrameSize	4194304 (4MB)	Default: 0 连接上的最大帧长度。与客户端协商的允许最大frame大小，设置为 0 表示无限制，但在某些QPid客户端会引发bug.。设置较大的值可以提高吞吐量;设置一个较小的值可能会提高延迟。

1.3 三、场景化调优

1.3.1 3.1 高吞吐写入场景

场景说明:

消息中间件能持续高速接收并持久化海量数据，避免因写入压力导致性能下降或数据丢失。典型如物联网设备数据上报、日志收集等场景。

引擎:

```
# zookeeper.conf
#通过将快照文件和事务分开存储，事务日志需要高频写和低延迟，通过切换SSD 磁盘，最大化元数据变更的处理效率。
#存储 ZooKeeper 的快照文件
dataDir=/data/zookeeper/data
#存储 ZooKeeper 的事务日志,单独SSD可以提升事务日志性能
dataLogDir=/ssd/zookeeper/data_log
```

```
# storage.conf
#记录 Bookie 所有写操作的事务日志,单独SSD可以提升事务日志性能
journalDirectories=/ssd/journal01
#存储实际的消息数据
ledgerDirectories=/hdd/ledgers01

# broker.conf
# 在没有明确要求多副本的情况下，减少副本数，可以提高系统的消息吞吐量
# 创建Ledger时指定Ensemble的大小
managedLedgerDefaultEnsembleSize=2

# 创建Ledger时指定Quorum的大小
managedLedgerDefaultWriteQuorum=2

# 创建Ledger时指定ack Quorum的大小
managedLedgerDefaultAckQuorum=2
```

1.3.1.1 jvm:

下面以32G内存为例,需要在存储节点安装路径下./config/storage_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=5g"}
```

下面以32G内存为例,需要在计算节点安装路径下./config/broker_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=6g"}
```

测试工具:

```
#OpenMessaging Benchmark（标准基准测试）
#pulsar.yaml
producer:
  batchingEnabled: true          # 开启消息批量发送
  sendMessageEnable: false       #关闭单条消息直接发送
  batchingMaxPublishDelayMs: 20     # 批量最大等待延迟：批次未达大小上限时，最多等待20ms后强制发送
  # 单个批次最大字节数
```

```
  batchingMaxBytes: 1048576       #可适当增大至2M（2097152）或者4M
(4194304)
  # 待发送队列满时阻塞生产线程（而非抛异常），OMB高并发下避免测试中断，实现流量削峰
  blockIfQueueFull: true
  # 生产者本地待发送消息队列长度
  pendingQueueSize: 3000
  compressionType: ZSTD      # 使用消息压缩算法ZSTD
  maxPendingMessages: 6000  # 生产者总待处理消息上限

consumer:
  # Broker 向消费者推送消息的预取队列长度
  receiverQueueSize: 10000
  # 共享订阅（多消费者负载均衡消费，OMB测试最大化消费并行度，提升整体吞吐）
  subscriptionType: Shared
  # 批量确认最大等待延迟：50ms内的确认请求合并发送，减少确认请求的网络开销
  acknowledgeGroupTimeMillis: 20
  # 消费者消息优先级（0为最高优先级，OMB测试无需区分优先级，使用默认最高级保证消费
速率)
  priorityLevel: 0
```

1.3.2 3.2 低延迟读取场景

场景说明:

消费者需要实时获取最新消息，要求尾部读取延迟极低，对响应时间高度敏感。典型如股票行情推送、实时风控等场景，要求消息一旦写入就能被立即读取。

引擎:

```
# zookeeper.conf
#通过将快照文件和事务分开存储，事务日志需要高频写和低延迟，通过切换SSD 磁盘，最大化
元数据变更的处理效率。
#存储 ZooKeeper 的快照文件
dataDir=/data/zookeeper/data
#存储 ZooKeeper 的事务日志,单独SSD可以提升事务日志性能
dataLogDir=/ssd/zookeeper/data_log

# storage.conf
```

```
#记录 Bookie 所有写操作的事务日志,单独SSD可以提升事务日志性能
journalDirectories=/ssd/journal01
#存储实际的消息数据
ledgerDirectories=/hdd/ledgers01

# broker.conf
# 在没有明确要求多副本的情况下，减少副本数，可以提高系统的消息吞吐量
# 创建Ledger时指定Ensemble的大小
managedLedgerDefaultEnsembleSize=2

# 创建Ledger时指定Quorum的大小
managedLedgerDefaultWriteQuorum=2

# 创建Ledger时指定ack Quorum的大小
managedLedgerDefaultAckQuorum=2
```

1.3.2.1 jvm:

下面以32G内存为例,需要在存储节点安装路径下./config/storage_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=5g"}
```

下面以32G内存为例,需要在存储节点安装路径下./config/broker_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=6g"}
```

测试工具:

```
#OpenMessaging Benchmark（标准基准测试）

#pulsar.yaml
producer:
  batchingEnabled: false           # 不开启消息批量发送
  batchingMaxPublishDelayMs: 1     # 批量最大等待延迟
  batchingMaxBytes: 131072         #单个批次最大字节数
  # 待发送队列满时阻塞生产线程（而非抛异常），OMB高并发下避免测试中断，实现流量削峰
  blockIfQueueFull: true
  # 生产者本地待发送消息队列长度
```

```
  pendingQueueSize: 0

consumer:
  # Broker 向消费者推送消息的预取队列长度
  receiverQueueSize: 10000
  # 共享订阅（多消费者负载均衡消费，OMB测试最大化消费并行度，提升整体吞吐）
  subscriptionType: Shared
```

1.3.3 3.3 追赶读取场景

场景说明:

消费者因故障或处理慢落后生产端大量消息，需要快速读取并处理积压的历史数据。

引擎:

```
# zookeeper.conf
#通过将快照文件和事务分开存储，事务日志需要高频写和低延迟，通过切换SSD 磁盘，最大化元数据变更的处理效率。
#存储 ZooKeeper 的快照文件
dataDir=/data/zookeeper/data
#存储 ZooKeeper 的事务日志,单独SSD可以提升事务日志性能
dataLogDir=/ssd/zookeeper/data_log

# storage.conf
#记录 Bookie 所有写操作的事务日志,单独SSD可以提升事务日志性能
journalDirectories=/ssd/journal01
#存储实际的消息数据
ledgerDirectories=/hdd/ledgers01

# broker.conf
# 在没有明确要求多副本的情况下，减少副本数，可以提高系统的消息吞吐量
# 创建Ledger时指定Ensemble的大小
managedLedgerDefaultEnsembleSize=2

# 创建Ledger时指定Quorum的大小
managedLedgerDefaultWriteQuorum=2
```

```
# 创建Ledger时指定ack Quorum的大小
managedLedgerDefaultAckQuorum=2
```

1.3.3.1 jvm:

下面以32G内存为例,需要在存储节点安装路径下./config/storage_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=5g"}
```

下面以32G内存为例,需要在存储节点安装路径下./config/broker_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=6g"}
```

测试工具:

```
#OpenMessaging Benchmark（标准基准测试）

#pulsar.yaml
producer:
  batchingEnabled: false           # 不开启消息批量发送
  batchingMaxPublishDelayMs: 1     # 批量最大等待延迟
  batchingMaxBytes: 131072         #单个批次最大字节数
  # 待发送队列满时阻塞生产线程（而非抛异常），OMB高并发下避免测试中断，实现流量削峰
  blockIfQueueFull: true
  # 生产者本地待发送消息队列长度
  pendingQueueSize: 0

consumer:
  # Broker 向消费者推送消息的预取队列长度
  receiverQueueSize: 10000
  # 共享订阅（多消费者负载均衡消费，OMB测试最大化消费并行度，提升整体吞吐）
  subscriptionType: Shared
  # 批量确认最大等待延迟：50ms内的确认请求合并发送，减少确认请求的网络开销
  acknowledgeGroupTimeMillis: 20
  # 消费者消息优先级（0为最高优先级，OMB测试无需区分优先级，使用默认最高级保证消费速率）
  priorityLevel: 0
```

1.3.4 3.4 磁盘空间不足场景

场景说明:

存储空间接近上限，需要紧急释放磁盘空间或优化存储效率。

通过命令行调整交换机消息保留策略，控制磁盘占用量，具体操作命令如下:

```
bin/admqctl admin namespaces set-retention mqtt-data/vhost01 --size
10737418240 --time 300
```

配置说明:

消息保存时间（--time）：300 秒（即 5 分钟），超时后自动清理过期消息;

消息总存储容量（--size）：10737418240 字节（即 10GB），达到容量上限时触发旧消息清理。

1.3.5 3.5 运行内存不足场景

场景说明:

内存使用率持续高位，频繁触发Full GC，影响服务性能。典型如Topic数量过多（>1000个）、消费者组庞大、缓存配置过大等场景，需要优化内存分配策略。

调整JVM参数:

下面以32G内存为例,需要在存储节点安装路径下./config/storage_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=5g"}
```

下面以32G内存为例,需要在存储节点安装路径下./config/broker_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=6g"}
```

1.4 四、性能问题分析思路

1.4.1 4.1 storage节点提示Not enough non-faulty bookies available报错信息

问题原因:

- 无法达到数据写入所需的副本数要求

- 健康可用的Bookie节点数量不足

优化建议:

1.降低副本配置:

```
# broker.conf
# 创建Ledger时指定Ensemble的大小
managedLedgerDefaultEnsembleSize=2

# 创建Ledger时指定Quorum的大小
managedLedgerDefaultWriteQuorum=2

# 创建Ledger时指定ack Quorum的大小
managedLedgerDefaultAckQuorum=2
```

2.重启所有的broker节点和异常的storage节点。

1.4.2 4.2 broker节点提示Java heap space报错信息

问题原因:

- broker节点堆内存不足

优化建议:

1.调整broker节点 JVM参数:

下面以32G内存为例,需要在存储节点安装路径下./config/broker_env.sh中修改参数:

```
ADMQ_MEM=${ADMQ_MEM:-"-Xms2g -Xmx2g -XX:MaxDirectMemorySize=5g"}
```

2.重启异常broker节点

1.4.3 4.3 MQTT性能测试中出现 "too many open files"报错信息

问题原因:

- 进程打开了超过系统限制的文件描述符
- 操作系统资源耗尽，无法分配新的文件句柄

优化建议:

1.临时提高限制

```
# 当前会话生效
ulimit -n 100000
# 对于运行中的进程，需要重启才能应用新限制
```

2.永久修改系统限制

```
# 编辑系统限制配置文件
vi /etc/security/limits.conf

* soft nofile 655350
* hard nofile 655350

* soft nproc 10240
* hard nproc 10240

* hard memlock unlimited
* soft memlock unlimited
```

1.4.4 4.4 存储节点写入延迟高，出现 "Write rate limiting" 警告**

问题原因:

- Journal磁盘性能瓶颈：使用机械硬盘或未分离Journal/Ledger目录
- 操Bookie内存配置不足：读/写缓存太小
- Entry日志过大：大消息导致频繁flush

优化建议

1.调整存储节点的配置

```
# storage.conf

# 1. 操作系统级优化
# 使用SSD分离journal和ledger目录
journalDirectory=/data/bookkeeper/journal
ledgerDirectories=/data/bookkeeper/ledger01,/data/bookkeeper/ledger02
```

```
# 2. 写入参数优化
journalSyncData=false  # 异步刷盘，根据数据安全要求权衡
flushInterval=1000

# 3. 性能相关配置
writeBufferSizeBytes=1048576=
dbStorage_writeCacheMaxSizeMb=256
dbStorage_readAheadCacheMaxSizeMb=64
```

2.重启所有的存储节点。

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