Apusic 金蝶天燕

生成时间：2026-05-07

性能优化手册

金蝶Apusic分布式缓存 V2.0.4

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 性能瓶颈来源
  - 2.1 内存管理配置
    - 2.1.1 内存管理优化点
    - 2.1.2 优化方式
      - 2.1.2.1 预留空间
      - 2.1.2.2 调整淘汰策略
    - 2.1.3 验证方式
  - 2.2 持久化配置
    - 2.2.1 持久化优化点
      - 2.2.1.1 优化方式
      - 2.2.1.2 场景化策略
  - 2.3 网络配置
    - 2.3.1 优化点
  - 2.4 系统配置

1 前言

本文档为金蝶Apusic分布式缓存（AMDC）V2.0.4的产品性能优化手册，旨在为AMDC V2.0.4用户提供全面、系统的性能优化指导。通过深入分析AMDC自身配置的性能瓶颈，结合典型业务场景的优化案例帮助用户解决在高并发环境下可能遇到的性能问题。

文档内容涵盖了从基础的系统参数调优到复杂的百万级并发连接优化，从常见的超时配置到关键的长连接复用策略，以及在性能测试过程中可能遇到的各种问题排查方法。

通过本文的指导，用户可以：

- 识别和解决AMDC在高并发场景下的性能瓶颈
- 合理配置操作系统和AMDC参数以提升系统整体性能
- 快速定位和解决性能测试中遇到的常见问题
- 根据具体业务场景实施相应的优化方案

1.1 适用对象

本文档适用于AMDC产品运维工程师、IT系统运维工程师、开发工程师、软件架构师及研发经理等人员。

1.2 相关文档

了解更多AMDC V2.0.4产品相关的信息，请参阅以下AMDC V2.0.4产品手册文档集：

序号	手册文档	说明
1	金蝶Apusic分布式缓存 V2.0.4 快速使用手册	简单介绍了如何快速上手使用AMDC 。
2	金蝶Apusic分布式缓存 V2.0.4 安装手册	详细介绍如何在各操作系统上安装AMDC，以及AMDC服务启停操作，产品的注册过程。
3	金蝶Apusic分布式缓存 V2.0.4 缓存核心用户手册	详细介绍 AMDC 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分布式缓存 V2.0.4 管控台用户手册	详细介绍AMDC管控台相关功能的使用和操作说明。
5	金蝶Apusic分布式缓存 V2.0.4 开发手册	详细介绍基于各开发语言进行AMDC客户端应用开发的说明。

6	金蝶Apusic分布式缓存 V2.0.4 迁移手册	详细介绍AMDC历史版本迁移升级到V2.0.4版本的说明，以及Redis迁移到AMDC的说明。
7	金蝶Apusic分布式缓存 V2.0.4 运维手册	详细介绍AMDC的监控、运维、安全加固等运维说明。
8	金蝶Apusic分布式缓存 V2.0.4 性能优化手册	详细介绍AMDC性能调优的说明。

1.3 技术支持

AMDC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 性能瓶颈来源

AMDC作为高性能内存数据库，其性能表现受多维因素制约，需从系统性视角审视瓶颈来源与优化路径，性能瓶颈主要呈现为四大维度：

- 内存维度是核心制约点，表现为物理内存不足触发淘汰机制、大Key操作引发阻塞、内存碎片导致利用率下降以及fork操作时的写时复制开销，这些因素共同构成内存密集型应用的性能天花板。
- 网络维度在高并发场景尤为关键，连接数受限于内核参数与文件描述符，缓冲区设置不当影响吞吐量，网络延迟与带宽瓶颈则直接制约跨节点通信效率。
- CPU维度受单线程架构约束，复杂命令执行、持久化fork操作、主从同步及键过期清理等计算密集型任务均会引发主线程阻塞，导致延迟波动。

接下来我们会从内存管理，持久化配置，网络配置，系统配置几个方面介绍AMDC调优思路。

2.1 内存管理配置

2.1.1 内存管理优化点

- AMDC 所有数据驻留内存，若用光可用 RAM，将触发 OOM（Linux 直接 kill 进程）或引发剧烈 swap，QPS 骤降。
- 热点数据往往只占 20 % 却承担 80 % 请求，把有限内存让给热键，可最大化缓存命中率、降低平均延迟。
- 操作系统自身、后台备份、AOF rewrite、网络缓冲区同样需要 30 %–40 % 空闲内存，否则 fork 时瞬间双倍内存开销会导致失败。

2.1.2 优化方式

2.1.2.1 预留空间

根据服务器内存总量合理设置，建议预留 30%-40% 内存给操作系统及其他进程，避免内存溢出。例如，若服务器内存为 `16GB`，可将 `maxmemory` 设为 `10GB`，配置语句为 `maxmemory 10gb`。

2.1.2.2 调整淘汰策略

根据业务数据特性选择合适的内存淘汰策略。当业务中热点数据集中，且需优先保留热点数据时，推荐使用 `volatile-lru`（淘汰设置了过期时间的键中最近最少使用的）或 `allkeys-lru`（淘汰所有键中最近最少使用的）。

若业务对数据过期时间敏感，可选择volatile-ttl（淘汰设置了过期时间的键中剩余时间最短的）。配置示例：`maxmemory-policy volatile-lru`。

2.1.3 验证方式

```
#内存用量
amdc-cli INFO memory | grep used_memory_human
#观察 ≤ 10 GB；同时 used_memory_rss_human 与 used_memory_human 差值 < 200 MB（碎片低）。

#淘汰计数
amdc-cli INFO stats | grep evicted_keys
#若持续增长且命中下降，说明 10 GB 偏小，可调大或再评估热键比例。

#命中率
amdc-cli INFO stats | grep keyspace_hits
#keyspace_hits / (hits + misses) ≥ 95 % 为健康。

#延迟毛刺
amdc-cli --latency -p 6379
#p99 ≤ 1 ms（本机万兆网卡、SSD 下）。
```

2.2 持久化配置

2.2.1 持久化优化点

RDB持久化主要问题：

- 数据丢失窗口：两次快照间宕机会丢失期间所有写入
- 磁盘I/O峰值：保存期间可能产生大量磁盘写入

AOF持久化主要问题：

- AOF重写膨胀：BGREWRITEAOF期间双写压力（旧AOF追加 + 新AOF重写）
- fsync性能损耗：appendfsync always 性能下降约90%

2.2.1.1 优化方式

关于RDB 智能保存触发（避免频繁bgsave），需要我们选择合适的触发保存的条件

```
save 900 1      # 15分钟至少1个key变化
save 300 10     # 5分钟至少10个key变化
```

```
save 60 10000  # 1分钟至少10000个key变化
```

反之，AOF的保存，我们则更需要关注重写时候的压力和刷盘的频率。

```
# 1. 合理的fsync策略（95%场景选everysec）
appendfsync everysec  # 平衡性能与安全
# appendfsync no      # 仅当可承受分钟级数据丢失时使用

# 2. 自动重写控制
auto-aof-rewrite-percentage 100    # 增长100%时触发重写
auto-aof-rewrite-min-size 64mb     # AOF文件最小重写阈值

# 3. 优化重写过程
no-appendfsync-on-rewrite yes      # 重写期间不fsync，避免双重阻塞
aof-rewrite-incremental-fsync yes # 增量式同步，减少单次fsync数据量
```

或采取更中和的方式， 混合持久化（RDB + AOF），此模式能够在尽量接近aof级别的数据安全的情况下，有着较于aof模式更小的磁盘占用，属于一种平衡aof和rdb两者优缺的方式。

```
aof-use-rdb-preamble yes   # 开启混合模式
# 配合以下参数:
aof-rewrite-incremental-fsync yes
rdbcompression yes         # RDB部分仍压缩
```

2.2.1.2 场景化策略

缓存型应用（数据可重建） 推荐策略：仅RDB 配置示例：

```
  save 3600 1       # 每小时备份一次
  stop-writes-on-bgsave-error no   # 允许写入继续
```

但是这是建立在恢复速度快（毫秒级，磁盘占用最小化，数据丢失1小时可接受的情况下。 与此同时我们还需要配合多级缓存（本地缓存+amdc），并设置过期时间自动淘汰。

金融交易类（零数据丢失要求） AOF(always) + RDB冷备 配置示例：

```
appendfsync always
auto-aof-rewrite-percentage 50   # 更频繁重写减少恢复时间
aof-rewrite-min-size 32mb
```

此时我们可以考虑使用一些性能补偿的手段来弥补数据安全带来的性能下降，比如使用性能更好的磁盘，使用主从架构，在从节点完成持久化操作。

2.3 网络配置

2.3.1 优化点

为了达到长期稳定运行的目标，我们的网络配置优化不仅会从性能瓶颈处理，已经高可用、故障恢复等方面的配置优化上加以说明介绍。

网络稳定离不开内核参数优化，以及节点之间的通信正常。

下列是网络配置的几个基础且关键的配置，需要结合内核参数调整，并结合业务需要调整主从或者集群节点间的通信配置。

```
maxclients 10000                    # 最大客户端连接数（需结合系统限制）
tcp-backlog 511                     # 半连接队列长度，必须 ≤ somaxconn
timeout 300                         # 空闲连接超时（秒），0表示禁用
repl-timeout 60                     # 主从复制超时
cluster-node-timeout 15000          # 集群节点超时（毫秒）
```

在高可用场景下，我们需要调整主从节点之间交互的一些参数配置来保障整个服务系统的稳定。其中我们大多数时候需要根据不同的业务和内存需要调整 repl-backlog-size，过大会占用内存，过小可能会使得增量同步不完全从而导致全量同步，可能导致cpu占用和内存过高。

```
# 主节点配置
repl-backlog-size 512mb             # 复制积压缓冲区（建议：内存的10%）
repl-backlog-ttl 3600               # 缓冲区保留时间（1h）
repl-disable-tcp-nodelay no         # 启用Nagle算法，减少包数

# 从节点配置
repl-ping-replica-period 10         # 从节点ping主节点频率
repl-timeout 60                     # 复制超时时间（需>ping周期×3）
```

同时需要注意哨兵的配置上 哨兵数量公式（保证仲裁）： 哨兵数量 = 2N + 1 (N为容忍的故障哨兵数) 最小推荐：3个哨兵，容忍1个故障

同时哨兵的部署不同于服务，我们需要独立部署它，不能和amdc 实例同机，导致单点故障。 以下是哨兵配置的参数建议：

```
# 1. 监控配置（每个哨兵都需要）
sentinel monitor mymaster 10.0.1.100 6379 2
# 格式：sentinel monitor <主节点名称> <IP> <端口> <法定票数>

# 2. 主观下线判断
sentinel down-after-milliseconds mymaster 30000
# 值说明：
# - <10000ms（内网低延迟环境）
# - 30000ms（标准生产环境）
# - >60000ms（跨机房或高延迟网络）

# 3. 故障转移超时控制
sentinel failover-timeout mymaster 180000
# 作用：控制故障转移各阶段的超时时间

# 4. 并行同步限制
sentinel parallel-syncs mymaster 1
# 值说明：
# - 1：最安全，新主节点压力小
# - 2-3：平衡同步速度与主节点负载
# - >3：可能导致新主节点过载

# 5. 节点通信优化
sentinel resolve-hostnames no       # 禁用DNS解析，使用IP
sentinel announce-ip 10.0.1.200     # 哨兵通告IP
sentinel announce-port 26379        # 哨兵通告端口
```

当我们在使用集群的时候，我们需要注意节点之间的超时时间设置，一般采取默认值即可，不宜过小，导致频繁进行集群检测，也不宜过大，导致节点宕机却未能监测。 cluster-announce-ip 是在集群部署中明确指定节点对外服务的IP

```
# 集群节点配置
cluster-enabled yes
cluster-config-file nodes.conf
cluster-require-full-coverage no     # 避免少数槽不可用导致全集群不可用

# 节点超时与心跳
cluster-node-timeout 15000           # 节点超时时间（影响故障转移）
cluster-replica-validity-factor 10   # 从节点数据有效性因子
cluster-announce-port 6379
cluster-announce-bus-port 16379      # 集群总线端口（端口+10000）

# 跨机房部署优化
cluster-announce-ip 192.168.1.100    # 明确指定节点IP
```

典型症状月解决方法：

```
# 症状
[ERR] Node ... is not empty. Either the node already knows other
nodes

# 解决步骤：
1. 检查每个节点的 cluster-announce-ip 是否正确
2. 确保所有端口在防火墙中开放（6379和16379） #此问题在大多数默认部署时会有遗漏，需要留意。
3. 清除旧集群配置：rm nodes.conf，然后重新创建集群
```

2.4 系统配置

为了缓存系统的稳定高效，有些时候我们需要将调优的范围延申到系统级别。 以下的参数调整可供参考。

1. vm.overcommit_memory - 内存超配策略

```
# 当前值查看
cat /proc/sys/vm/overcommit_memory
```

```
# 永久修改（/etc/sysctl.conf）
vm.overcommit_memory = 1
```

一般来说，该参数的默认值为0 ，它的含义是内核检查可用内存，如果不足可能拒绝分配（可能导致amdc 服务fork失败） 而其他值 1 --总是允许分配（即使超出物理内存） 2 --严格限制，分配不能超过物理内存 + swap 对此，我们的建议是设置为1，否则amdc执行bgsave时，fork子进程可能因"内存不足"而失败.(实际不需要双倍内存，但内核会检查)

2. 透明大页（Transparent HugePages）

```
# 检查状态
cat /sys/kernel/mm/transparent_hugepage/enabled

# 禁用命令
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag

# 永久禁用（CentOS/RHEL 7+）
grubby --update-kernel=ALL --args="transparent_hugepage=never"
```

我们在此建议禁用，否则可能导致bgsave时fork延迟增加数倍，内存使用量可能增加30-40%，运行中可能产生不可预测的延迟峰值

3. net.core.somaxconn - 连接队列大小

```
# 查看当前值
cat /proc/sys/net/core/somaxconn

# 永久修改
net.core.somaxconn = 65535
```

对于这个参数，我们的建议是增大，因为amdc的tcp-backlog配置（默认511）受此参数限制，高并发时，如果队列太小，客户端会收到"Connection refused"。典型场景：秒杀活动时，大量连接瞬间到达。

4. ulimit文件描述符限制

```
# 查看amdc进程限制
cat /proc/$(pidof amdc-server)/limits | grep "open files"
```

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

该参数的默认值通常为1024，其作用为限制单个进程能打开的文件数（每个连接占用一个文件描述符），我们建议增大，因为：每个客户端连接消耗1个文件描述符，且持久化也会打开文件（RDB/AOF）默认1024只能支持约1000个并发连接。

生产建议：至少65535，大型集群可能需要更高。也存在现在大多数客户端都自带连接池，实际的消耗需要具体情况具体分析。

5. 内存分配器优化

```
# 安装jemalloc（性能更好的内存分配器）
yum install jemalloc  # CentOS/RHEL
apt-get install libjemalloc-dev  # Ubuntu/Debian

# 启动amdc时使用
LD_PRELOAD=/usr/lib/libjemalloc.so amdc-server /etc/amdc.conf
```

一般默认使用glibc 的malloc jemalloc优势：内存碎片更少（对长时间运行的amdc很重要）多线程分配效率更高，专门为长时间运行的服务优化。 因此优势，我们建议使用jemalloc ， 可以减少30-50%的内存碎片，长时间运行后内存使用更稳定

6. 网络缓冲区调整

```
# 永久修改（/etc/sysctl.conf）
net.core.rmem_max = 134217728
net.core.wmem_max = 134217728
net.ipv4.tcp_rmem = 4096 87380 134217728
net.ipv4.tcp_wmem = 4096 65536 134217728
```

上述的参数作用主要是用于控制TCP发送和接受缓冲区大小 因为默认值一般较小，所以我们建议是增大，因为我们的服务可能需要返回大量的数据，其中要么是大value，要么是批量操作，缓冲区太小的话会导致TCP窗口的缩小，降低吞吐量，尤其是在高带宽、高延迟的网络中更加重要。

生产建议：

- rmem_max/wmem_max：`128MB`
- tcp_rmem/tcp_wmem：最小`4KB`，默认`64KB/85KB`，最大`128MB`

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