



APUSIC
固若长城
睿比世界

緩存核心用戶手冊

金蝶Apusic分布式缓存 V2.0.6

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 技術支持
- 2 簡介
- 3 相關概念
- 4 功能清單
- 5 產品安裝
 - 5.1 Redis配置兼容
- 6 使用介紹
 - 6.1 緩存核心使用指南
 - 6.1.1 操作命令
 - 6.1.2 AMDC集群使用指南
 - 6.1.2.1 主從模式
 - 6.1.2.1.1 主從命令
 - 6.1.2.2 哨兵模式
 - 6.1.2.2.1 哨兵命令
 - 6.1.2.3 集群模式
 - 6.1.2.3.1 集群命令
 - 6.1.2.3.2 集群運維命令
 - 6.1.3 SSL使用
 - 6.1.3.1 SSL配置項
 - 6.1.3.2 通過OPENSSL生成證書
 - 6.1.3.3 服務器SSL啟動
 - 6.1.3.4 客戶端SSL連接
 - 6.1.4 國密使用
 - 6.1.4.1 通過Tongsuo生成證書
 - 6.1.4.2 服務器國密配置啟動
 - 6.1.4.3 客戶端SSL連接
 - 6.1.5 MPTCP多路徑傳輸協議
 - 6.1.5.1 MPTCP相關配置項
 - 6.1.5.2 配置MPTCP環境

- 6.1.5.3 服务器MPTCP配置启动
- 6.1.5.4 客户端MPTCP连接
- 6.1.6 数据持久化
 - 6.1.6.1 RDB
 - 6.1.6.1.1 生成RDB文件的方式
 - 6.1.6.1.2 RDB恢复数据的方式
 - 6.1.6.1.3 恢复损坏的RDB文件
 - 6.1.6.2 AOF
 - 6.1.6.2.1 生成AOF文件的方式
 - 6.1.6.2.2 AOF数据恢复方式
 - 6.1.6.2.3 恢复损坏的AOF文件
- 6.1.7 命令审计
- 6.2 amdc-cli客户端使用介绍
 - 6.2.1 客户端参数
 - 6.2.2 客户端使用
 - 6.2.2.1 一般操作
 - 6.2.2.2 统计操作
 - 6.2.2.3 查询操作
 - 6.2.2.4 测试操作
 - 6.2.2.5 LUA操作
 - 6.2.2.6 集群操作
- 6.3 RDB集群数据迁移工具使用介绍
- 6.4 性能测试工具使用介绍
 - 6.4.1 AMDC benchmark参数
- 7 产品所有配置说明
 - 7.1 缓存核心所有配置
 - 7.1.1 缓存配置文件
 - 7.1.2 哨兵配置文件
 - 7.1.3 授权认证中心配置
- 8 产品工具使用说明
 - 8.1 amdc-check-rdb工具使用手册
- 9 常见问题处理
 - 9.0.1 工具位置与运行前提
 - 9.0.1.1 工具位置

- 9.0.1.2 运行前提
- 9.0.2 基本语法
- 9.0.3 核心选项与参数说明
- 9.0.4 常见操作示例
 - 9.0.4.1 校验RDB文件完整性
 - 9.0.4.2 修复损坏的RDB文件
- 9.0.5 注意事项
- 9.1 amdc-check-aof工具使用手册
 - 9.1.1 工具简介
 - 9.1.2 工具位置与运行前提
 - 9.1.2.1 工具位置
 - 9.1.2.2 运行前提
 - 9.1.3 基本语法
 - 9.1.4 核心选项与参数说明
 - 9.1.5 常见操作示例
 - 9.1.5.1 校验AOF文件完整性
 - 9.1.5.2 修复损坏的AOF文件
 - 9.1.6 注意事项
- 9.2 amdc-benchmark 工具使用手册
 - 9.2.1 工具简介
 - 9.2.2 工具位置与运行前提
 - 9.2.2.1 工具位置
 - 9.2.2.2 运行前提
 - 9.2.3 基本语法
 - 9.2.4 核心选项与参数说明
 - 9.2.4.1 连接选项
 - 9.2.4.2 核心测试选项
 - 9.2.4.3 高级扩展参数（多线程 / 集群 / 特殊测试模式）
 - 9.2.5 常见操作示例（核心重点）
 - 9.2.5.1 本地默认测试
 - 9.2.5.2 指定并发数和请求数测试
 - 9.2.5.3 针对性测试核心命令
 - 9.2.5.4 按时间进行压力测试针对性测试核心命令
 - 9.2.5.5 简化输出，快速查看QPS

- 9.2.5.6 測試結果解讀（關鍵指標）
- 9.2.6 注意事項
- 9.3 amdc-conf-conv工具使用手冊
 - 9.3.1 工具簡介
 - 9.3.2 工具位置與運行前提
 - 9.3.2.1 工具位置
 - 9.3.2.2 運行前提
 - 9.3.3 基本語法
 - 9.3.4 核心選項與參數說明
 - 9.3.5 常見操作示例
 - 9.3.5.1 轉換AMDC V2.0.2及以下版本AMDC服務端配置為AMDC V2.0.6版本
 - 9.3.5.2 轉換AMDC V2.0.2及以下版本AMDC哨兵配置為AMDC V2.0.6版本
 - 9.3.6 注意事項
- 9.4 常見問題排查
 - 9.4.1 工具無法運行
 - 9.4.2 校驗 / 修復文件失敗
 - 9.4.3 amdc-cli 無法連接AMDC服務
- 9.5 總結
- 10 常見問題解決
 - 10.1 問題一：端口佔用與解決
 - 10.2 AMDC緩存核心端口修改
 - 10.3 ACL文件加載
 - 10.4 解決AMDC主從故障切換
 - 10.5 請求延遲問題
 - 10.6 如何檢測執行了那些命令
 - 10.7 進程還在但無法連接
 - 10.8 dir 路徑問題
 - 10.9 密碼設置問題
 - 10.10 日誌配置問題
 - 10.11 主從緩衝區配置問題

1 前言

本文档为金蝶Apusic分布式缓存（AMDC）V2.0.6缓存核心产品的用户手册，详细介绍了AMDC缓存核心的功能的使用、配置方法及管理操作等内容。

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、开发工程师等人员。

1.2 相关文档

了解更多AMDC V2.0.6产品相关的信息，请参阅以下AMDC V2.0.6产品手册文档集：

序 號	手册文档	说明
1	金蝶Apusic分布式缓存 V2.0.6 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 V2.0.6 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 V2.0.6 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 V2.0.6 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 V2.0.6 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 V2.0.6 迁移手册	详细介绍AMDC历史版本迁移升级到V2.0.6版本的说明，以及Redis迁移到AMDC的说明。
7	金蝶Apusic分布式缓存 V2.0.6 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 V2.0.6 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：

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

您在取得技術支持時，請提供如下信息：

1. 您的姓名
2. 公司信息與聯繫方式
3. 操作系統及其版本
4. 產品版本號
5. 出現異常及錯誤的日誌、截圖等詳細信息

2 简介

金蝶Apusic分布式缓存软件（Apusic In-Memory Data Cache，簡稱 **AMDC** ），一款適用於全場景、高吞吐量、數據安全的分布式缓存软件，為大規模、高並發、高可用的關鍵應用提供安全可靠的缓存支撐能力。產品兼容Redis協議與持久化數據文件，可簡單快捷平穩替換Redis。

3 相關概念

名詞	含義	使用說明
單機	存儲數據，負責數據讀寫，負責數據同步	AMDC的默認模式為單機模式，該模式下只有一個節點
主從	從主節點複製數據，負責數據讀寫，不參與數據同步	主節點數據的實時副本。它主動連接到指定的主節點，並接收主節點發送的數據流，從而保持與主節點的數據同步。默認情況下，從節點是只讀的，所有寫操作必須發送到主節點。
哨兵	監控主從節點，負責主節點故障轉移，負責主從節點同步	哨兵模式，當主節點掛掉時，自動將從節點切換為主節點；當從節點故障時，哨兵將其標記為下線。
集群	由多台相互獨立的計算機組成的計算機服務系統	在本文中，主從/哨兵/集群三種模式都可以算是集群，cluster集群模式特指多主節點數據分片存儲的模式。

4 功能清單

功能	功能說明	補充說明
多數據類型緩存	提供string、list、hash、set、sortedset、geo、hyperloglog、stream的數據緩存類型。	支持Redis協議兼容的數據結構，可直接替代Redis使用，無需修改應用代碼
發佈訂閱	實現了發佈訂閱功能，增強系統功能性	支持頻道訂閱、模式訂閱，可用於消息隊列、事件通知等場景
ACL權限管控	為服務提供了安全的訪問機制，支持細粒度的訪問權限控制。	基於用戶角色的權限管理，支持命令級別、鍵空間級別的訪問控制
內存數據淘汰策略	提供多種數據淘汰策略，滿足多種數據淘汰要求，提高內存利用率。	包括LRU、LFU、TTL等多種淘汰算法，可根據業務特點靈活選擇
持久化	為緩存核心提高可用性，防止在宕機時丟失數據。	支持RDB快照和AOF日誌兩種持久化方式，保障數據安全
Lua腳本支持	支持使用Lua腳本操作緩存核心。	支持複雜邏輯的原子性操作，提升處理效率
多線程模式	支持多線程並發請求處理，提升系統吞吐量。	採用I/O多路復用技術，充分利用多核CPU性能
主從模式	支持主從備份。	實現數據冗餘，支持讀寫分離，提高系統可用性
哨兵模式	為主從模式提供節點監控、自動故障轉移、故障通知、配置傳播功能。	實現高可用架構，自動檢測故障並切換服務
集群模式	支持彈性伸縮（內存的擴/縮容），並且同時具備了故障轉移功能。	數據分片存儲，支持在線擴容縮容，滿足大數據量場景需求
MPTCP 多路徑傳輸協議	支持多路徑並行數據傳輸，可聚合多條網絡鏈路同時收發數據，突破單鏈路傳輸帶寬限制，實現傳輸鏈路冗餘容錯。	適配大文件、大批量數據同步與傳輸場景，相比傳統單TCP鏈路傳輸性能提升約100%，可有效規避單鏈路網絡抖動、擁堵、中斷問題，大幅提升數據傳輸的吞吐量、穩定性與可靠性。
監控運維	性能監控	提供詳細的性能指標監控，包括QPS、響應時間、內存使用率等
	慢查詢日誌	記錄執行時間超過閾值的命令，便於性能調優
	統計信息	通過INFO命令獲取服務器運行狀態和統計數據

高可用性	故障自动恢复	自动检测节点健康状况，实现故障自动切换
	数据一致性	保证主从节点间的数据一致性
	服务发现	支持自动发现集群节点，简化配置管理
安全特性	访问控制	支持密码认证、SSL/TLS/国密加密传输
	审计日志	记录用户操作日志，满足合规性要求
扩展功能	事务支持	支持MULTI/EXEC等事务命令
	管道操作	支持批量命令执行，减少网络往返时间
	键过期管理	支持TTL、EXPIRE等键生命周期管理功能
开发支持	客户端兼容	完全兼容Redis客户端库（Jedis、Lettuce等）
	协议兼容	支持RESP协议，可无缝替换Redis
	工具链支持	提供CLI工具、基准测试工具等

5 產品安裝

AMDC缓存核心安裝請參考[《金蝶Apusic分佈式缓存 V2.0.6 安裝手冊》](#)

5.1 Redis配置兼容

從AMDC v2.0.2開始，兼容Redis的配置文件，AMDC會自動將Redis.conf中的內容轉換為AMDC的配置內容。

使用方式：`./amdc-server redis.conf`（指定為redis的配置文件）

6 使用介绍

6.1 缓存核心使用指南

分布式缓存是AMDC最核心的能力，是整个产品的中心，其他功能都是建立在数据缓存业务之上。AMDC把数据直接存到内存中，并利用多线程读写分离，实现高效存储，满足不同类型的数据存储，快捷开发，减少类型转换；支持多种数据淘汰策略，合理利用内存空间。

6.1.1 操作命令

命令	说明
ACL LOAD	从配置的ACL文件中重新加载ACL
ACL SAVE	在已配置的ACL文件中保存当前的ACL规则
ACL LIST	列出ACL配置文件格式的当前ACL规则
ACL USERS	列出所有已配置的ACL规则的用户名
ACL GETUSER username	获取特定ACL用户的规则
ACL SETUSER username [rule [rule ...]]	修改或创建特定ACL用户的规则
ACL DELUSER username [username ...]	删除指定的ACL用户和相关规则
ACL CAT [categoryname]	列出类别内的ACL类别或命令
ACL GENPASS [bits]	为ACL用户生成伪随机安全密码
ACL WHOAMI	返回与当前连接关联的用户的名稱
ACL LOG [count or RESET]	列出最新的ACL安全事件
ACL HELP	显示有关不同的子命令的有用文本
APPEND key value	将值附加到键
ASKING	在-ask重定向後由群集客户端发送
AUTH [username] password	对服务器进行身份验证
BGREWRITEAOF	异步重写仅附加文件
BGSAVE [SCHEDULE]	异步将数据集保存到磁盘
BITCOUNT key [start end [BYTE BIT]]	计数字符串中的設置位

BITFIELD key [GET encoding offset] [SET encoding offset value] [INCRBY encoding offset increment] [OVERFLOW WRAP SAT FAIL]	在字符串上執行任意位字段整數操作
BITFIELD_RO key GET encoding offset	在字符串上執行任意位字段整數操作，禁用的只讀變體
BITOP operation destkey key [key ...]	在字符串之間執行按位操作
BITPOS key bit [start [end [BYTE BIT]]]	在字符串中找到第一個位設置或清除
BLPOP key [key ...] timeout	阻塞式刪除並返回第一個元素
BRPOP key [key ...] timeout	阻塞式刪除並返回最後一個元素
BRPOPLPUSH source target timeout	從列表中取出最後一個元素，並插入到另外一個列表的頭部；如果列表沒有元素會阻塞列表直到等待超時或發現可彈出元素為止。
BLMOVE source destination FROM-TOP FROM-BOTTOM TO-TOP TO-BOTTOM timeout	阻塞式返回並刪除存儲在源列表的第一個或最後一個元素（頭尾取決於wherefrom參數），並將該元素壓入到存儲目標列表的第一個或最後一個元素（頭尾取決於wheret to參數）
LMPOP numkeys key [key ...] LEFT RIGHT [COUNT count]	從提供的鍵名列表中彈出第一個非空列表鍵中的一個或多個元素
BLMPOP timeout numkeys key [key ...] LEFT RIGHT [COUNT count]	阻塞式從提供的鍵名列表中彈出第一個非空列表鍵中的一個或多個元素；或阻塞連接直到另一個客戶端推送到它或直到超時
BZPOPMIN key [key ...] timeout	阻塞式刪除並將成員從一個或多個有序集中返回最低分數，或阻塞連接直到另一個客戶端推送到它或直到超時
BZPOPMAX key [key ...] timeout	阻塞式刪除並將成員從一個或多個有序集中返回最高分數，或阻塞連接直到另一個客戶端推送到它或直到超時
CLIENT CACHING YES NO	指示服務器在下一個請求中跟踪或不鍵
CLIENT ID	返回當前連接的客戶端ID
CLIENT INFO	返回有關當前客戶端連接的信息
CLIENT KILL [ip:port] [ID client-id] [TYPE normal master slave pubsub] [USER username] [ADDR ip:port]	殺死客戶端的連接

[LADDR ip:port] [SKIPME yes/no]	
CLIENT LIST [TYPE normal master replica pubsub] [ID client-id [client-id ...]]	獲取客戶端連接列表
CLIENT GETNAME	獲取當前的連接名稱
CLIENT GETREDIR	獲取跟踪通知重定向客戶端ID（如果有）
CLIENT UNPAUSE	恢復暫停的客戶的處理
CLIENT PAUSE timeout [WRITE ALL]	停止客戶端的處理命令一段時間
CLIENT REPLY ON OFF SKIP	指示服務器是否回復命令
CLIENT SETNAME connection-name	設置當前連接名稱
CLIENT TRACKING ON OFF [REDIRECT client-id] [PREFIX prefix [PREFIX prefix ...]] [BCAST] [OPTIN] [OPTOUT] [NOLOOP]	啟用或禁用服務器輔助客戶端緩存支持
CLIENT TRACKINGINFO	返回關於當前連接的服務器輔助客戶端緩存的信息
CLIENT UNBLOCK client-id [TIMEOUT ERROR]	取消阻止從不同連接中阻止阻止命令的客戶端
COMMAND	獲取 Redis 命令詳細信息數組
COMMAND COUNT	獲取Redis命令的總數
COMMAND GETKEYS	給定完整的 Redis 命令提取鍵
COMMAND INFO command-name [command-name ...]	獲取特定Redis命令詳細信息的數組
CONFIG GET parameter [parameter ...]	獲取配置參數的值
CONFIG REWRITE	用內存配置重寫配置文件
CONFIG SET parameter value [parameter value ...]	將配置參數設置為給定的值
CONFIG RESETSTAT	重置 INFO 返回的統計信息
COPY source destination [DB destination-db] [REPLACE]	複製一個鍵
DBSIZE	返回所選數據庫中的鍵數
DEBUG OBJECT key	獲取有關鍵的調試信息
DEBUG SEGFAULT	使服務器崩潰

DECR key	一個鍵的整數值減一
DECRBY key decrement	按給定的數字減少一個鍵的整數值
DEL key [key ...]	刪除一個鍵
DISCARD	放棄在多次執行後發出的所有命令
DUMP key	返回存儲在指定鍵中的值的序列化版本
DELEX key	根據值的比較結果（值相等/不等、哈希摘要匹配/不匹配）來條件性刪除一個 Key
DIGEST key	計算key對應值的摘要（SM3/SHA256等）
ECHO message	回顯給定的字符串
EVAL script numkeys [key [key ...]] [arg [arg ...]]	執行一個Lua腳本服務器端
EvalSHA sha1 numkeys [key [key ...]] [arg [arg ...]]	執行 Lua 腳本服務器端
EXEC	執行 MULTI
EXISTS key [key ...]	判斷key是否存在
EXPIRE key seconds [NX XX GT LT]	以秒為單位設置鍵的生存時間
EXPIREAT key timestamp [NX XX GT LT]	將鍵的到期時間設置為 UNIX 時間戳
EXPIRETIME key	獲取鍵的到期 Unix 時間戳
FAILOVER [TO host port [FORCE]] [ABORT] [TIMEOUT milliseconds]	在此服務器與其副本之一之間啟動協調故障轉移
FLUSHALL [ASYNC SYNC]	從所有數據庫中刪除所有鍵
FLUSHDB [ASYNC SYNC]	從當前數據庫中刪除所有鍵
FUNCTION LOAD	加載服務器端函數（替代Lua腳本）
FUNCTION DELETE	刪除指定函數
FUNCTION FLUSH	清空所有函數
FUNCTION LIST	列出所有函數
FUNCTION STATS	查看函數執行統計
GEOADD key [NX XX] [CH] longitude latitude member [longitude latitude member ...]	在使用有序集表示的地理空間索引中添加一個或多個地理空間項目

GEOHASH key member [member ...]	將地理空間索引的成員作為標準 geohash 字符串返回
GEOPOS key member [member ...]	返回地理空間索引成員的經度和緯度
GEODIST key member1 member2 [m km ft mi]	返回地理空間索引的兩個成員之間的距離
GEORADIUS key longitude latitude radius m km ft mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count [ANY]] [ASC DESC] [STORE key] [STOREDIST key]	查詢表示地理空間索引的有序集，以獲取與點的給定最大距離匹配的成員
GEORADIUSBYMEMBER key member radius m km ft mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count [ANY]] [ASC DESC] [STORE key] [STOREDIST key]	查詢表示地理空間索引的有序集以獲取與成員的給定最大距離匹配的成員
GEOSEARCH key [FROMMEMBER member] [FROMLONLAT longitude latitude] [BYRADIUS radius m km ft mi] [BYBOX width height m km ft mi] [ASC DESC] [COUNT count [ANY]] [WITHCOORD] [WITHDIST] [WITHHASH]	查詢表示地理空間索引的有序集以獲取框或圓區域內的成員
GEOSEARCHSTORE destination source [FROMMEMBER member] [FROMLONLAT longitude latitude] [BYRADIUS radius m km ft mi] [BYBOX width height m km ft mi] [ASC DESC] [COUNT count [ANY]] [STOREDIST]	查詢表示地理空間索引的有序集以獲取框或圓區域內的成員，並將結果存儲在另一個鍵中
GET key	獲取一個鍵的值
GETBIT key offset	返回存儲在 key
GETDEL key	的字符串值中偏移量處的位值，獲取某個 key 的值並刪除 key
GETRANGE key start end	獲取存儲在鍵中的字符串的子字符串
GETSET key value	設置一個鍵的字符串值並返回它的舊值
HDEL key field [field ...]	刪除一個或多個哈希字段
HELLO [protover [AUTH username password] [SETNAME clientname]]	與 Redis 握手
HEXISTS key field	判斷一個 hash 字段是否存在
HGET key field	獲取哈希字段的值
HGETALL key	獲取哈希中的所有字段和值
HINCRBY key field increment	將散列字段的整數值增加給定的數字

HINCRBYFLOAT key field increment	將哈希字段的浮點值增加給定的數量
HKEYS key	獲取散列中的所有字段
HLEN key	獲取哈希中的字段數
HMGET key field [field ...]	獲取所有給定哈希字段的值
HMSET key field value [field value ...]	將多個哈希字段設置為多個值
HSET key field value [field value ...]	設置一個哈希字段的字符串值
HSETNX key field value	設置一個哈希字段的值，僅當該字段不存在時
HRANDFIELD key [count [WITHVALUES]]	從散列中獲取一個或多個隨機字段
HSTRLEN key field	獲取哈希字段值的長度
HVALS key	獲取散列中的所有值
HOTKEYS [[value] [opt] ...]	檢測並列出當前熱鍵
INCR key	鍵的整數值加一
INCRBY key increment	將鍵的整數值增加給定的數量
INCRBYFLOAT key increment	將鍵的浮點值增加給定的數量
INFO [section]	獲取有關服務器的信息和統計信息
LOLWUT [VERSION version]	展示一些計算機藝術和 Redis 版本
KEYS pattern	找到所有匹配給定模式的鍵
LASTSAVE	獲取上次成功保存到磁盤的 UNIX 時間戳
LINDEX key index	通過索引從列表中獲取元素
LINSERT key BEFORE AFTER pivot element	在列表中的另一個元素之前或之後插入一個元素
LLEN key	獲取列表的長度
LPOP key [count]	刪除並獲取列表中的第一個元素
LPOS key element [RANK rank] [COUNT num-matches] [MAXLEN len]	返回列表中匹配元素的索引
LPUSH key element [element ...]	將一個或多個元素添加到列表中

LPUSHX key element [element ...]	將元素添加到列表中，僅當列表存在時
LRange key start stop	從列表中獲取一系列元素
LRem key count element	從列表中刪除元素
LSet key index element	通過索引設置列表中元素的值
LTrim key start stop	將列表修剪到指定範圍
MEMORY DOCTOR	輸出內存問題報告
MEMORY HELP	顯示有關不同子命令的有用文本
MEMORY MALLOC-STATS	顯示分配器內部統計信息
MEMORY PURGE	要求分配器釋放內存
MEMORY STATS	顯示內存使用詳情
MEMORY USAGE key [SAMPLES count]	估計一個key的內存佔用
MGET key [key ...]	獲取所有給定鍵的值
MIGRATE host port key destination-db timeout [COPY] [REPLACE] [AUTH password] [AUTH2 username password] [KEYS key [key ...]]	以原子方式將鍵從 Redis 實例傳輸到另一個實例
MONITOR	實時監聽服務器收到的所有請求
MOVE key db	移動一個鍵到另一個數據庫
MSET key value [key value ...]	將多個鍵設置為多個值
MSETNX key value [key value ...]	僅當不存在任何鍵時才將多個鍵設置為多個值
MULTI	標記一個事務塊的開始
OBJECT ENCODING key	檢查 Redis 對象的內部編碼
OBJECT FREQ key	獲取Redis對象的對數訪問頻率計數器
OBJECT IDLETIME key	獲取自上次訪問 Redis 對象以來的時間
OBJECT REFCOUNT key	獲取鍵值的引用次數
OBJECT HELP	顯示有關不同子命令的有用文本
PERSIST key	從鍵中刪除過期時間

PEXPIRE key milliseconds [NX XX GT LT]	以毫秒為單位設置鍵的生存時間
PEXPIREAT key milliseconds-timestamp [NX XX GT LT]	將鍵的到期時間設置為以毫秒為單位指定的 UNIX 時間戳
PEXPIRETIME key	以毫秒為單位獲取鍵的到期 Unix 時間戳
PFADD key [element [element ...]]	將指定的元素添加到指定的 HyperLogLog.
PFCOUNT key [key ...]	返回 HyperLogLog 在 key(s) 處觀察到的集合的近似基數
PFMERGE destkey sourcekey [sourcekey ...]	將 N 個不同的 HyperLogLog 合併為一個
PING [message]	Ping 服務器
PSETEX key milliseconds value	以毫秒為單位設置鍵值和過期時間
PSUBSCRIBE pattern [pattern ...]	偵聽發佈到與給定模式匹配的頻道的消息
PUBSUB CHANNELS [pattern]	列出活動頻道
PUBSUB NUMPAT	獲取獨特模式訂閱的計數
PUBSUB NUMSUB [channel [channel ...]]	獲取頻道的訂閱者數量
PUBSUB HELP	顯示有用的文字 ab輸出不同的子命令
PTTL key	以毫秒為單位獲取鍵的生存時間
PUBLISH channel message	向頻道發佈消息
PUNSUBSCRIBE [pattern [pattern ...]]	停止收聽發佈到與給定模式匹配的頻道的消息
QUIT	關閉連接
RANDOMKEY	從鍵空間返回一個隨機鍵
READONLY	啟用對集群副本節點連接的讀取查詢
READWRITE	禁用對集群副本節點連接的讀取查詢
RENAME key newkey	重命名一個鍵
RENAMENX key newkey	重命名鍵，僅當新鍵不存在時
RESET	重置連接

RESTORE key ttl serialized-value [REPLACE] [ABSTTL] [IDLETIME seconds] [FREQ frequency]	使用提供的序列化值創建一個鍵，之前使用 DUMP 獲得
ROLE	返回實例在複製上下文中的角色
RPOP key [count]	刪除並獲取列表中的最後一個元素
RPOPLUSH source destination	刪除列表中的最後一個元素，將其添加到另一個列表中並返回它
LMOVE source destination LEFT RIGHT LEFT RIGHT	從列表中彈出一個元素，將其推送到另一個列表並返回它
R PUSH key element [element ...]	將一個或多個元素附加到列表中
R PUSHX key element [element ...]	僅當列表存在時才將元素附加到列表中
SADD key member [member ...]	將一個或多個成員添加到集合中
SAVE	將數據集同步保存到磁盤
SCARD key	獲取集合中的成員數量
SCRIPT DEBUG YES SYNC NO	為執行的腳本設置調試模式
SCRIPT EXISTS sha1 [sha1 ...]	檢查腳本緩存中是否存在腳本
SCRIPT FLUSH [ASYNC SYNC]	從腳本緩存中刪除所有腳本
SCRIPT KILL	終止當前正在執行的腳本
SCRIPT LOAD script	將指定的 Lua 腳本加載到腳本緩存中
SDIFF key [key ...]	減去多組
SDIFFSTORE destination key [key ...]	減去多個集合並將結果集合存儲在一個鍵中
SELECT index	更改當前連接選擇的數據庫
SET key value [EX seconds PX milliseconds EXAT timestamp PXAT milliseconds-timestamp KEEPTTL] [NX XX] [GET]	設置一個鍵的字符串值
SETRBIT key offset value	設置或清除存儲在 key
SETEX key seconds value	設置一個鍵的值和過期時間
SETNX key value	設置鍵的值，僅當鍵不存在時
SETRANGE key offset value	從指定的偏移量開始覆蓋鍵處的部分字符串

SHUTDOWN [NOSAVE SAVE]	將數據集同步保存到磁盤，然後關閉服務器
SINTER key [key ...]	返回由所有給定集合的交集產生的集合的成員
SINTERCARD numkeys key [key ...] [LIMIT limit]	將多個集合相交並返回結果的基數
SINTERSTORE destination key [key ...]	將多個集合相交並將結果集存儲在一個鍵中
SISMEMBER key member [member ...]	返回每個成員是否為存儲在key處的集合的成員
REPLICAOF host port	使服務器成為另一個實例的副本，或將其提升為主服務器
SLOWLOG GET [count]	獲取慢日誌的條目
SLOWLOG LEN	獲取慢日誌的長度
SLOWLOG RESET	清除慢日誌中的所有條目
SLOWLOG HELP	顯示有關不同子命令的有用文本
SMEMBERS key	獲取所有成員
SMOVE source destination member	將成員從一組移動到另一組
SORT key [BY pattern] [LIMIT offset count] [GET pattern [GET pattern ...]] [ASC DESC] [ALPHA] [STORE destination]	對列表、集合或有序集合中的元素進行排序
SORT_RO key [BY pattern] [LIMIT offset count] [GET pattern [GET pattern ...]] [ASC DESC] [ALPHA]	對列表、集合或有序集合中的元素進行排序， SORT 的只讀變體
SPOP key [count]	從集合中刪除並返回一個或多個隨機成員
SRANDMEMBER key [count]	從集合中獲取一個或多個隨機成員
SREM key member [member ...]	從集合中刪除一個或多個成員
STRLEN key	獲取存儲在鍵中的值的長度
SUBSCRIBE channel [channel ...]	收聽發佈到給定頻道的消息
SUNION key [key ...]	返回所有給定集合的並集所產生的集合的成員
SUNIONSTORE destination key [key ...]	返回所有給定集合的並集並存儲到指定集合中

SWAPDB index1 index2	交換兩個Redis數據庫
SYNC	內部命令，從主站發起複製流
PSYNC replicationid offset	內部命令，從主站發起複製流
TIME	返回當前服務器時間
TOUCH key [key ...]	更改鍵的最後訪問時間，返回指定的現有鍵的數量
TTL key	在幾秒鐘內獲得一把鑰匙的生存時間
TYPE key	確定存儲在 key 的類型
UNSUBSCRIBE [channel [channel ...]]	停止收聽發佈到給定頻道的消息
UNLINK key [key ...]	在另一個線程中異步刪除一個鍵，否則它就像 DEL 一樣，但非阻塞
UNWATCH	忘記所有觀看過的鑰匙吧
WAIT numreplicas timeout	等待同步複製當前連接上下文中發送的所有寫命令
WATCH key [key ...]	觀察給定的鍵以確定 MULTI/EXEC 塊的執行
ZADD key [NX XX] [GT LT] [CH] [INCR] score member [score member ...]	將一個或多個成員添加到有序集中，或者如果它已經存在則更新其分數
ZCARD key	獲取有序集中的成員數量
ZCOUNT key min max	用給定值內的分數計算有序集中的成員
ZDIFF numkeys key [key ...] [WITHSCORES]	減去多個有序集
ZDIFFSTORE destination numkeys key [key ...]	減去多個有序集並將結果有序集存儲在一個新鍵中
ZINCRBY key increment member	在有序集中增加成員的分數
ZINTERCARD numkeys key [key ...] [LIMIT limit]	將多個有序集相交並返回結果的基數
ZINTERSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM MIN MAX]	將多個有序集相交並將結果有序集存儲在一個新鍵中
ZLEXCOUNT key min max	計算給定字典範圍內有序集中的成員數量
ZPOPMAX key [count]	刪除並返回有序集中得分最高的成員

ZPOPMIN key [count]	刪除並返回有序集中得分最低的成員
ZMPOP numkeys key [key ...] MIN MAX [COUNT count]	刪除並返回具有有序集中分數的成員
ZRANDMEMBER key [count [WITHSCORES]]	從有序集中獲取一個或多個隨機元素
ZRANGESTORE dst src min max [BYScore BYLEX] [REV] [LIMIT offset count]	將有序集中的一系列成員存儲到另一個鍵中
ZRANGE key min max [BYScore BYLEX] [REV] [LIMIT offset count] [WITHSCORES]	返回有序集中的一系列成員
ZRANGEBYLEX key min max [LIMIT offset count]	返回成員範圍rs 在一個有序的集合中，按字典序排列
ZREVRANGEBYLEX key max min [LIMIT offset count]	返回有序集中的成員範圍，按字典序範圍，從高到低的字符串排序
ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]	按分數返回有序集中的一系列成員
ZRANK key member	確定有序集中成員的索引
ZREM key member [member ...]	從有序集中刪除一個或多個成員
ZREMRANGEBYLEX key min max	刪除給定字典範圍之間有序集中的所有成員
ZREMRANGEBYRANK key start stop	刪除給定索引內有序集中的所有成員
ZREMRANGEBYSCORE key min max	刪除給定分數內有序集中的所有成員
ZREVRANGE key start stop [WITHSCORES]	按索引返回有序集中的一系列成員，分數從高到低排序
ZREVRANGEBYSCORE key max min [WITHSCORES] [LIMIT offset count]	按分數返回有序集中的一系列成員，分數從高到低排序
ZREVRANK key member	確定一個有序集中某個成員的索引，分數從高到低排序
ZSCORE key member	在有序集中獲取與給定成員關聯的分數
ZMSCORE key member [member ...]	獲取與有序集中的給定成員相關聯的分數
ZUNIONSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM MIN MAX]	添加多個有序集並將結果有序集存儲在一個新鍵中
SCAN cursor [MATCH pattern] [COUNT count] [TYPE type]	增量迭代鍵空間

SSCAN key cursor [MATCH pattern] [COUNT count]	增量迭代 Set 元素
HSCAN key cursor [MATCH pattern] [COUNT count]	增量迭代哈希字段和關聯值
ZSCAN key cursor [MATCH pattern] [COUNT count]	增量迭代已排序的集合元素和相關分數
XINFO CONSUMERS key groupname	列出消費者組中的消費者
XINFO GROUPS key	列出流的消費者組
XINFO STREAM key [FULL [COUNT count]]	獲取有關流的信息
XINFO HELP	顯示有關不同子命令的有用文本
XADD key [NOMKSTREAM] [MAXLEN MINID [= ~] threshold [LIMIT count]] * ID field value [field value ...]	將新條目附加到流中
XTRIM key MAXLEN MINID [= ~] threshold [LIMIT count]	將流修剪到（大約如果 '~' 被傳遞）特定大小
XDEL key ID [ID ...]	從流中刪除指定的條目，返回實際刪除的項目數，如果某些 ID 不存在，則可能與傳遞的 ID 數不同
XRANGE key start end [COUNT count]	返回流中的一系列元素，其 ID 與指定的 ID 間隔相匹配
XREVRANGE key end start [COUNT count]	與 XRANGE 相比，以相反的順序（從較大到較小的 ID）返回流中的一系列元素，其中 ID 與指定的 ID 間隔匹配
XLEN key	返回流中的條目數
XREAD [COUNT count] [BLOCK milliseconds] STREAMS key [key ...] ID [ID ...]	返回多個流中從未見過的元素，其 ID 大於調用者為每個流報告的 ID，可以屏蔽
XGROUP CREATE key groupname id \$ [MKSTREAM]	創建一個消費者組
XGROUP CREATECONSUMER key groupname consumername	在消費者組中創建消費者
XGROUP DELCONSUMER key groupname consumername	從消費者組中刪除消費者
XGROUP DESTROY key groupname	銷毀一個消費組
XGROUP SETID key groupname id \$	將消費者組設置為任意最後交付的 ID 值
XGROUP HELP	顯示有關不同子命令的有用文本

XREADGROUP GROUP group consumer [COUNT count] [BLOCK milliseconds] [NOACK] STREAMS key [key ...] ID [ID ...]	使用消費者組從流中返回新條目，或訪問給定消費者的待處理條目的歷史記錄，可以屏蔽
XACK key group ID [ID ...]	將待處理消息標記為正確處理，有效地將其從消費者組的待處理條目列表中刪除，該命令的返回值是成功確認的消息數，即我們在 PEL 中實際能夠解析的 ID
XCLAIM key group consumer min-idle-time ID [ID ...] [IDLE ms] [TIME ms-unix-time] [RETRYCOUNT count] [FORCE] [JUSTID]	在流消費者組的上下文中，此命令更改掛起消息的所有權，以便新的所有者是作為命令參數指定的消費者
XAUTOCLAIM key group consumer min-idle-time start [COUNT count] [JUSTID]	在流消費者組的上下文中，此命令自動更改掛起消息的所有權，以便新的所有者是作為命令參數指定的消費者
XPENDING key group [[IDLE min-idle-time] start end count [consumer]]	從流消費者組待處理條目列表中返回信息和條目，即獲取但從未確認的消息
XIDMPRECORD key pid iid stream-id	持久化流IDMP狀態

6.1.2 AMDC集群使用指南

AMDC集群為高可用可擴展集群，集群方式分別為主從模式、哨兵模式、cluster集群模式。

6.1.2.1 主從模式

AMDC主從模式，即（Master-Slave Replication）主從複製，使用一個AMDC實例作為主機，其餘的作為備份機，主機和備份機的數據完全一致，主機支持數據的寫入和讀取等各項操作，而從機則支持與主機數據的同步和讀取，當其中一台AMDC出現故障時，訪問其他結點的AMDC緩存核心即可。

6.1.2.1.1 主從命令

- 轉變為某個節點的從節點：slaveof [ip] [port]（replicaof no one 將使從節點轉變為主節點）。

6.1.2.2 哨兵模式

哨兵是一個自動監控處理AMDC緩存核心間故障節點轉移工作的AMDC緩存核心端程序，AMDC提供哨兵的命令，哨兵通過發送命令，等待AMDC緩存核心響應，從而監控運行的多個AMDC實例。

6.1.2.2.1 哨兵命令

1. 查看sentinel的狀態命令：
info
2. 獲取sentinel中監控的所有master的節點命令：
sentinel masters

3. 獲取master-name主節點的狀態信息命令：
sentinel master [master-name]
4. 獲取master-name節點下所有的slaves的狀態信息命令：
sentinel slaves [master-name]
5. 通過sentinel中節點名獲取ip地址命令：
sentinel get-master-addr-by-name [master-name]
6. 添加節點命令：
sentinel monitor [name] [ip] [port] [quorum]
7. 重置amdc name匹配指定的狀態命令：
sentinel reset [master-name]
8. 刪除節點命令：
sentinel remove [master-name]
9. 強制節點主觀下線命令：
sentinel failover [master-name]
10. 獲取哨兵集群中其他哨兵節點信息：
sentinel sentinels
11. 為指定的主節點設置訪問密碼：
sentinel set [master-name] auth-pass [password]

6.1.2.3 集群模式

集群模式是AMDC實現主節點的彈性伸縮，主要作用於增大或者減少AMDC可以使用的內存容量，實現通過擴充節點來滿足業務的緩存需求，無需通過增加服務器內存來實現；並且集群模式有與哨兵類似自動故障轉移功能，具有高可用性，是更好的選擇。集群模式所有節點都能互相通信，感知對方的狀態信息，集群可以自動分配主從節點也可以手動指定這些節點

6.1.2.3.1 集群命令

1. 為接收節點分配新的哈希插槽：
CLUSTER ADDSLOTS slot [slot ...]
2. 命令從連接的節點觸發群集配置年齡的增量。如果節點的配置年齡為零，或者小於群集的最大年齡，則該年齡將遞增：
CLUSTER BUMPEPOCH
3. 返回為給定節點處於活動的故障報告的數量：
CLUSTER COUNT-FAILURE-REPORTS node-id
4. 返回指定哈希插槽中的本地鍵數：
CLUSTER COUNTKEYSINSLOT slot
5. 將哈希插槽設置為接收節點中的未綁定：
CLUSTER DELSLOTS slot [slot ...]
6. 強制副本執行其主站的手動故障轉移：
CLUSTER FAILOVER [FORCE|TAKEOVER]

7. 删除节点自己的插槽信息：
CLUSTER FLUSHSLOTS
8. 从节点表中删除节点：
CLUSTER FORGET node-id
9. 在指定的哈希插槽中返回本地键名称：
CLUSTER GETKEYSINSLOT slot count
10. 提供有关AMDC Cluster节点状态的信息：
CLUSTER INFO
11. 返回指定键的哈希槽：
CLUSTER KEYSLOT key
12. 强制一个节点集群与另一个节点握手：
CLUSTER MEET ip port
13. 返回节点 id：
CLUSTER MYID
14. 获取节点的集群配置：
CLUSTER NODES
15. 将当前节点设置为指定主节点的从节点（slave）：
CLUSTER REPLICATE [master-node-id]
16. 将 key 原子性迁移到另一个实例：
MIGRATE [host] [port] [key] [destination-db] [timeout] [COPY] [REPLACE] [KEYS key1 key2...]
17. 设置槽的状态（导入、导出、归属）：
CLUSTER SETSLOT [STATE] []
18. 强制当前节点将当前集群配置立即保存到 nodes.conf 文件：
CLUSTER SAVECONFIG
19. 重置节点的集群配置：
CLUSTER RESET [HARD|SOFT]
20. 设置节点的配置纪元：
CLUSTER SET-CONFIG-EPOCH [epoch]
21. 返回指定主节点的所有副本节点信息：
CLUSTER REPLICAS [node-id]
22. 返回集群中所有槽的映射信息，包括每个槽的主节点和副本节点：
CLUSTER SLOTS

6.1.2.3.2 集群运维命令

以下命令是 AMDC 提供的用于集群日常运维的辅助工具，通过 amdc-cli 执行（非 Cluster 协议命令）

命令	说明
<code>amdc-cli --cluster create ip:port1 ... --cluster-replicas 1</code>	创建集群（自动 meet + 分配槽 +

	主從)
<code>amdc-cli --cluster check ip:port</code>	檢查集群健康狀態
<code>amdc-cli --cluster info ip:port</code>	查看集群統計信息
<code>amdc-cli --cluster fix ip:port</code>	修復槽未覆蓋等問題（實驗性）
<code>amdc-cli --cluster reshard ip:port</code>	在線重新分片（遷移槽）
<code>amdc-cli --cluster add-node new_ip:new_port existing_host:existing_port</code>	添加新節點
<code>amdc-cli --cluster del-node host:port node_id</code>	刪除節點

說明：這些命令主要用於簡化集群的創建、檢查、維護和擴展操作，適用於管理員進行集群管理。其中 `--cluster fix` 和 `--cluster reshard` 是高級功能，建議在充分理解其行為後再使用。

6.1.3 SSL使用

SSL是AMDC內置的加密通訊協議，啟動SSL相關配置即可實現與客戶端的SSL雙向認證加密通訊，可以使用OPENSSL生成相應的證書和密鑰文件。

SSL配置在amdc.yaml和sentinel.yaml中都存在，使用方式一致，具體配置解析可參考緩存配置文件章節或如下：

6.1.3.1 SSL配置項

根據TLS配置的常見參數，以下是補充完整的表格：

參數名稱	解析	使用
tls-port	SSL監聽端口，如果僅開啟SSL監聽，需要把Network下的port端口設置為0	port: 6369
tls-cert-file	服務端SSL證書	tls-cert-file: "./certs/ssl_tls_cert/server.crt"
tls-key-file	服務端SSL證書的密鑰	tls-key-file: "./certs/ssl_tls_cert/server.key"
tls-key-file-pass	服務端SSL證書密鑰的密碼	tls-key-file-pass: "your_password"
tls-ca-cert-file	證書簽發機構的證書文件，即可信的根證書	tls-ca-cert-file: "./certs/ssl_tls_cert/ca.crt"

tls-ca-cert-dir	證書簽發機構的證書文件路徑，如有多個可信根證書，可使用該參數配置	tls-ca-cert-dir: ""
tls-client-cert-file	客戶端SSL證書，為集群/主從模式使用	tls-client-cert-file: "./certs/ssl_tls_cert/client.crt"
tls-client-key-file	客戶端SSL證書的密鑰，為集群/主從模式使用	tls-client-key-file: "./certs/ssl_tls_cert/client.key"
tls-client-key-file-pass	客戶端SSL證書密鑰的密碼	tls-client-key-file-pass: "your_password"
tls-dh-params-file	Diffie-Hellman參數文件，用於前向保密	tls-dh-params-file: "./certs/ssl_tls_cert/dhparam.pem"
tls-protocols	TLS協議版本，指定允許使用的TLS協議版本	tls-protocols: "TLSv1.2 TLSv1.3"
tls-ciphers	SSL/TLS加密算法套件（用於TLSv1.2及以下版本）	tls-ciphers: "ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256"
tls-ciphersuites	TLSv1.3密碼套件（用於TLSv1.3版本）	tls-ciphersuites: "TLS_CHACHA20_POLY1305_SHA256:TLS_AES_256_GCM_SHA384"
tls-prefer-server-ciphers	是否優先使用服務器端定義的加密套件而非客戶端定義的	tls-prefer-server-ciphers: "yes"
tls-session-caching	是否啟用TLS會話緩存	tls-session-caching: "yes"
tls-session-cache-size	TLS會話緩存的最大大小	tls-session-cache-size: 1000
tls-session-cache-timeout	TLS會話緩存的超時時間（秒）	tls-session-cache-timeout: 300
tls-auth-clients	服務端是否驗證客戶端證書，默認需要客戶端提供證書並且服務端做驗證，如果不需要客戶端可以配置	tls-auth-clients: ""

	為"no", 設置為 optional, 則客戶端可以提供證書進行驗證, 也可以不提供	
tls-replication	主從模式是否使用TLS通訊, 當master當前參數為true, 那麼從節點也必須為true	tls-replication: false
tls-cluster	集群模式是否使用TLS通訊	tls-cluster: false

6.1.3.2 通過OPENSSL生成證書

可以使用其他方法獲得所需的證書, 只要合法可用即可。

對於SSL雙向認證:

- 服務器需要CA證書、server證書、server私鑰;
- 客戶端需要CA證書, client證書、client私鑰。

步驟:

1. 下載安裝openssl, 官網/source/index.html (openssl.org)
2. openssl.cnf文件:

```
[ server_cert ]
keyUsage = digitalSignature, keyEncipherment
nsCertType = server

[ client_cert ]
keyUsage = digitalSignature, keyEncipherment
nsCertType = client
```

3. 創建CA證書:

- `openssl genrsa -out ca.key 4096`

- `openssl req -x509 -new -nodes -sha256 -key ca.key -days 3650 -subj "/O=APUSIC /CN=AMDC" -out ca.crt`

4. 創建服務器私鑰與證書

- `openssl genrsa -out amdc.key 2048`
- `openssl req -new -sha256 -subj "/O=APUSIC /CN=AMDC" -key amdc.key | openssl x509 -req -CA ca.crt -CAkey ca.key -CAcreateserial -days 365 -extfile openssl.cnf -extensions server_cert -out amdc.crt`

5. 創建客戶端私鑰與證書

- `openssl genrsa -out client.key 2048`
- `openssl req -new -sha256 -subj "/O=APUSIC /CN=AMDC" -key client.key | openssl x509 -req -CA ca.crt -CAkey ca.key -CAcreateserial -days 365 -extfile openssl.cnf -extensions client_cert -out client.crt`

6.1.3.3 服務器SSL啟動

1. 修改amdc.yaml（關鍵段落）

```
TLS/SSL:
# TLS 監聽端口, 需配合 port 0 使用以僅啟用 TLS
tls-port: 0
# 服務器 TLS 證書文件路徑 (PEM 格式)
tls-cert-file: "./cert/amdc.crt"
# 服務器 TLS 私鑰文件路徑 (PEM 格式)
tls-key-file: "./cert/amdc.key"
# TLS 私鑰文件的密碼 (若加密)
tls-key-file-pass: ""
# 客戶端 TLS 證書文件路徑 (用於客戶端身份驗證)
tls-client-cert-file: ""
# 客戶端 TLS 私鑰文件路徑
tls-client-key-file: ""
# 客戶端 TLS 私鑰文件的密碼 (若加密)
tls-client-key-file-pass: ""
# DH 參數文件路徑
tls-dh-params-file: ""
```

```
# 用於驗證客戶端和節點的 CA 證書文件路徑
tls-ca-cert-file: "./cert/cat.crt"
```

2. 啟動amdc `./amdc-server amdc.yaml`

6.1.3.4 客戶端SSL連接

- 使用amdc-cli連接: `./amdc-cli -p 6369 --tls --cert ./client.crt --key ./client.key --cacert ./ca.crt`
- Go語言連接:

```
func TestTls(t *testing.T) {
    caCert, err := ioutil.ReadFile("./certs/ca.crt")
    if err != nil {
        panic(err)
    }
    caCertPool := x509.NewCertPool()
    caCertPool.AppendCertsFromPEM(caCert)
    cliCert, err := tls.LoadX509KeyPair("./certs/client.crt", "./certs/client.key")
    if err != nil {
        panic(err)
    }
    cli := redis.NewClient(&redis.Options{
        Addr: "127.0.0.1:6369",
        TLSConfig: &tls.Config{
            Certificates: []tls.Certificate{cliCert}, //客戶端證書，双向認證必須攜帶
            RootCAs:     caCertPool,               //校驗服務器證書【CA證書】
            InsecureSkipVerify: true,               //不用校驗服務器證書
        },
        DialTimeout: time.Minute,
    })

    fmt.Println(cli.Info("server").String())
}
```

6.1.4 國密使用

國密是AMDC內置的加密通訊協議，啟動國密相關配置即可實現與客戶端的國密雙向認證加密通訊，可以使用Tongsuo國密庫生成相應的證書和密鑰文件。

國密加密配置統一適配amdc.yaml 與 sentinel.yaml 配置文件，兩份文件的國密參數使用規則完全一致。相較於常規SSL加密配置，國密TLS配置需區分協議版本指定對應加密套件：使用TLSv1.2協議需配置參數 `tls-ciphers: ECC-SM2-SM4-GCM-SM3:ECC-SM2-SM4-CBC-SM3`；使用TLSv1.3協議需配置參數 `tls-ciphersuites: TLS_SM4_GCM_SM3`，以此實現標準國密全鏈路加密通信。

6.1.4.1 通過Tongsuo生成證書

Tongsuo（銅鎖）是OpenAtom開源社區推出的現代化開源密碼學基礎庫，前身是BabaSSL，兼容OpenSSL接口體系，專為國產商用密碼合規場景打造，具備完整的國密算法與安全通信協議能力，是當前國產化中間件TLS國密改造的主流底層依賴庫。

對於國密雙向認證：

- 服務器需要CA證書、server證書、server私鑰；
- 客戶端需要CA證書、client證書、client私鑰。

步驟：

1. 下載安裝Tongsuo並編譯，國密版本openssl在Tongsuo/app目錄下
2. openssl.cnf文件：

```
[ server_cert ]
keyUsage = digitalSignature, keyEncipherment
nsCertType = server

[ client_cert ]
keyUsage = digitalSignature, keyEncipherment
nsCertType = client
```

3. 創建CA證書：

- `Tongsuo/app/openssl ecparam -genkey -name SM2 -out ca.key`
- `Tongsuo/app/openssl req -x509 -new -nodes -sm3 -key ca.key -days 3650 -subj "/O=APUSIC /CN=AMDC" -out ca.crt`

4. 創建服務器私鑰與證書

- `Tongsuo/app/openssl ecparam -genkey -name SM2 -out amdc.key`
- `Tongsuo/app/openssl -new -sm3 -subj "/O=APUSIC /CN=AMDC" -key amdc.key | openssl x509 -req -sm3 -CA ca.crt -CAkey ca.key -CAcreateserial -days 365 -out amdc.crt`

5. 創建客戶端私鑰與證書

- `Tongsuo/app/openssl ecparam -genkey -name SM2 -out client.key`

- `Tongsuo/app/openssl -new -sm3 -subj "/O=APUISC /CN=AMDC" -key client.key | openssl x509 -req -sm3 -CA ca.crt -CAkey ca.key -CAcreateserial -days 365 -extfile openssl.cnf -extensions client_cert -out client.crt`

6.1.4.2 服務器國密配置啟動

1. 修改amdc.yaml (關鍵段落)

TLS/SSL:

```
# TLS 監聽端口, 需配合 port 0 使用以僅啟用 TLS
tls-port: 6369

# 服務器 TLS 證書文件路徑 (PEM 格式)
tls-cert-file: "./cert/amdc.crt"

# 服務器 TLS 私鑰文件路徑 (PEM 格式)
tls-key-file: "./cert/amdc.key"

# TLS 私鑰文件的密碼 (若加密)
tls-key-file-pass: ""

# 客戶端 TLS 證書文件路徑 (用於客戶端身份驗證)
tls-client-cert-file: ""

# 客戶端 TLS 私鑰文件路徑
tls-client-key-file: ""

# 客戶端 TLS 私鑰文件的密碼 (若加密)
tls-client-key-file-pass: ""

# DH 參數文件路徑
tls-dh-params-file: ""

# 用於驗證客戶端和節點的 CA 證書文件路徑
tls-ca-cert-file: "./cert/cat.crt"

# TLSv1.3 允許的密碼套件, 國密使用 TLS_SM4_GCM_SM3
tls-ciphersuites: TLS_SM4_GCM_SM3
```

2. 啟動amdc `./amdc-server amdc.yaml`

6.1.4.3 客戶端SSL連接

- 使用amdc-cli連接: `./amdc-cli --tls -p 6369 --cert ./amdc.crt --key ./amdc.key --cacert ./ca.crt --tls-ciphersuites TLS_SM4_GCM_SM3`

6.1.5 MPTCP多路徑傳輸協議

AMDC V2.0.6版本支持多路徑傳輸協議MPTCP，MPTCP是Linux內核5.6版本開始引入的，該協議允許多個網絡路徑同時傳輸數據，從而提高傳輸效率。

6.1.5.1 MPTCP相關配置項

```
# MPTCP 多路徑傳輸控制協議，支持多鏈路並發傳輸，提升鏈路聚合帶寬與傳輸吞吐，需依賴 Linux 5.6 及以上內核版本
mptcp-enabled: yes
# 主從複製是否開啟MPTCP
repl-mptcp-enabled: yes
```

6.1.5.2 配置MPTCP環境

前置要求：操作系統內核版本需 $\geq$ Linux 5.6。Linux 5.6 及以上內核原生內置 MPTCP 核心功能，無需手動編譯內核模塊；5.6 以下內核無原生 MPTCP 能力，不支持該多路徑傳輸配置方案。

MPTCP 多路徑傳輸依賴多物理網卡鏈路聚合，需服務器配備多張物理網卡並接入光模塊，實現多鏈路並行傳輸。以下以兩台雙網卡服務器為例，完成服務端與客戶端 MPTCP 鏈路聚合配置。

服務端設備：聚合網卡 IP 192.168.10.25、192.168.11.25

```
root@user:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp94s0f0np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq
state UP group default qlen 1000
    link/ether 6c:fe:54:26:7a:70 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.25/24 brd 192.168.10.255 scope global enp94s0f0np0
        valid_lft forever preferred_lft forever
    inet6 fe80::6efe:54ff:fe26:7a70/64 scope link
        valid_lft forever preferred_lft forever
3: enp94s0f1np1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq
```

```
state UP group default qlen 1000
    link/ether 6c:fe:54:26:7a:71 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.25/24 brd 192.168.11.255 scope global enp94s0f1np1
        valid_lft forever preferred_lft forever
    inet6 fe80::6efe:54ff:fe26:7a71/64 scope link
        valid_lft forever preferred_lft forever
```

客戶端設備：聚合網卡 IP 192.168.10.26、192.168.11.26

```
root@user:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp94s0f0np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq
state UP group default qlen 1000
    link/ether 6c:fe:54:22:a9:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.26/24 brd 192.168.10.255 scope global enp94s0f0np0
        valid_lft forever preferred_lft forever
    inet6 fe80::6efe:54ff:fe22:a930/64 scope link
        valid_lft forever preferred_lft forever
3: enp94s0f1np1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq
state UP group default qlen 1000
    link/ether 6c:fe:54:22:a9:31 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.26/24 brd 192.168.11.255 scope global enp94s0f1np1
        valid_lft forever preferred_lft forever
    inet6 fe80::6efe:54ff:fe22:a931/64 scope link
        valid_lft forever preferred_lft forever
```

1. 前置校驗命令

核查內核版本

```
uname -r
```

確認MPTCP內核編譯啟用（5.6+默認開啟）

```
sysctl net.mptcp.enabled
```

若返回net.mptcp.enabled = 1代表內核 MPTCP 功能就緒；為 0 時執行sysctl -w net.mptcp.enabled=1臨時開啟，寫入/etc/sysctl.conf實現永久啟用。

2. MPTCP 信息查看命令**# 查看已配置的MPTCP多鏈路端點信息**

```
ip mptcp endpoint show
```

查看MPTCP子流、地址接入數量限制參數

```
ip mptcp limits show
```

3. 服務端配置

服務端網卡配置signal標識：向外通告本機多網卡 IP 地址，供客戶端主動建立多條子流鏈路

綁定兩張網卡為MPTCP通告地址

```
ip mptcp endpoint add 192.168.10.25 id 1 dev enp94s0f0np0 signal
ip mptcp endpoint add 192.168.11.25 id 2 dev enp94s0f1np1 signal
```

全局MPTCP資源限制優化：最大並發子流8、最多接收8個遠端新增地址

```
ip mptcp limits set subflows 8
ip mptcp limits set add_addr_accepted 8
```

4. 客戶端配置

客戶端subflow：通過本網卡主動創建 MPTCP 子流；fullmesh：全互聯模式，自動和服務端所有通告 IP 建立子流聚合帶寬

雙網卡fullmesh全鏈路聚合，剩餘網卡常規子流模式

```
ip mptcp endpoint add 192.168.10.26 id 1 dev enp94s0f0np0 subflow
fullmesh
ip mptcp endpoint add 192.168.11.26 id 2 dev enp94s0f1np1 subflow
fullmesh
```

同服務端統一限制參數

```
ip mptcp limits set subflows 8
ip mptcp limits set add_addr_accepted 8
```

5. MPTCP 连通性 & 带宽测试 (iperf3版本 >= 3.20)

服务端

```
iperf3 -s --mptcp
```

客户端 (对接服务端任意一个 IP 即可自动聚合全部链路)

```
iperf3 -c 192.168.11.25 --mptcp
```

带宽测试对比

- 未开启 MPTCP:

```
root@user:~# iperf3 -c 192.168.11.25
Connecting to host 192.168.11.25, port 5201
[ 5] local 192.168.11.26 port 51502 connected to 192.168.11.25 port 5201
[ ID] Interval           Transfer    Bitrate        Retr  Cwnd
[ 5]  0.00-1.00    sec   1.10 GBytes  9.45 Gbits/sec    0   1.33 MBytes
[ 5]  1.00-2.00    sec   1.09 GBytes  9.39 Gbits/sec    0   1.78 MBytes
[ 5]  2.00-3.00    sec   1.10 GBytes  9.43 Gbits/sec    0   1.95 MBytes
[ 5]  3.00-4.00    sec   1.10 GBytes  9.40 Gbits/sec    0   2.09 MBytes
[ 5]  4.00-5.00    sec   1.10 GBytes  9.41 Gbits/sec    0   2.21 MBytes
[ 5]  5.00-6.00    sec   1.09 GBytes  9.40 Gbits/sec    0   3.04 MBytes
[ 5]  6.00-7.00    sec   1.10 GBytes  9.43 Gbits/sec    0   3.20 MBytes
[ 5]  7.00-8.00    sec   1.09 GBytes  9.39 Gbits/sec    0   3.20 MBytes
[ 5]  8.00-9.00    sec   1.09 GBytes  9.38 Gbits/sec    0   3.20 MBytes
[ 5]  9.00-10.00   sec   1.09 GBytes  9.38 Gbits/sec    0   3.20 MBytes
-----
[ ID] Interval           Transfer    Bitrate        Retr
[ 5]  0.00-10.00   sec  11.0 GBytes  9.42 Gbits/sec    0
[ 5]  0.00-10.00   sec  11.0 GBytes  9.41 Gbits/sec
sender
receiver

iperf Done.
```

- 开启 MPTCP:

```

root@user:~# iperf3 -c 192.168.11.25 --mptcp
Connecting to host 192.168.11.25, port 5201
[ 5] local 192.168.11.26 port 24404 connected to 192.168.11.25 port 5201
[ ID] Interval           Transfer     Bitrate      Retr  Cwnd
[ 5]  0.00-1.00    sec   2.16 GBytes  18.6 Gbits/sec    2   3.70 MBytes
[ 5]  1.00-2.00    sec   2.12 GBytes  18.2 Gbits/sec    2   5.86 MBytes
[ 5]  2.00-3.00    sec   2.01 GBytes  17.3 Gbits/sec  142   4.55 MBytes
[ 5]  3.00-4.00    sec   2.07 GBytes  17.8 Gbits/sec    1   4.81 MBytes
[ 5]  4.00-5.00    sec   2.33 GBytes  20.0 Gbits/sec    2   4.82 MBytes
[ 5]  5.00-6.00    sec   2.42 GBytes  20.8 Gbits/sec    0   5.21 MBytes
[ 5]  6.00-7.00    sec   2.39 GBytes  20.6 Gbits/sec    0   5.21 MBytes
[ 5]  7.00-8.00    sec   2.34 GBytes  20.1 Gbits/sec    0   5.21 MBytes
[ 5]  8.00-9.00    sec   2.34 GBytes  20.1 Gbits/sec   68   4.04 MBytes
[ 5]  9.00-10.00   sec   1.96 GBytes  16.8 Gbits/sec    2   2.94 MBytes
-----
[ ID] Interval           Transfer     Bitrate      Retr
[ 5]  0.00-10.00    sec   22.6 GBytes  19.4 Gbits/sec  219
[ 5]  0.00-10.02    sec   22.6 GBytes  19.3 Gbits/sec
sender
receiver

iperf Done.

```

6. 參數模式說明

參數標識	適用端	功能說明
signal	服務端	對外發佈本機多 IP 信息，被動等待客戶端發起子流
subflow	客戶端	允許從當前網卡出口建立 MPTCP 子流
fullmesh	客戶端	全拓撲組網，自動和對端全部地址建立子流，最大化鏈路帶寬疊加

6.1.5.3 服務器MPTCP配置啟動

1. 修改amdc.yaml（關鍵段落）

```

# MPTCP 多路徑傳輸控制協議，支持多鏈路並發傳輸，提升鏈路聚合帶寬與傳輸吞吐，需依賴 Linux 5.6 及以上內核版本
mptcp-enabled: yes

```

2. 啟動amdc `./amdc-server amdc.yaml`

6.1.5.4 客戶端MPTCP連接

- 使用amdc-cli連接：`./amdc-cli -p 6359 --mptcp`

6.1.6 數據持久化

amdc用戶提供了RDB、AOF兩種持久化方式，在[操作命令](#)中也有相關的操作方式，在此將展開兩種持久方式的使用以及相關的工具。

6.1.6.1 RDB

RDB持久化方式，會將[amdc](#)中的數據以RDB格式存儲到硬盤中。

6.1.6.1.1 生成RDB文件的方式

生成RDB文件的方式有三種：

1. 配置文件中進行[save](#)配置RDB的生成條件，觸發條件時系統自動使用[bgsave](#)生成RDB文件；
2. 使用[save](#)命令，使用該命令後將會阻塞所有請求，在RDB文件生成後，恢復請求處理；
3. 使用[bgsave](#)命令，該命令不會阻塞請求，但是生成RDB文件的速度會比[save](#)慢。

6.1.6.1.2 RDB恢復數據的方式

1. 將一個或多個RDB文件放到緩存的配置文件[dir](#)中指定的目錄下，啟動[amdc](#)將會自動加載RDB數據。

6.1.6.1.3 恢復損壞的RDB文件

amdc提供RDB文件修復工具

使用方式：`./amdc-check-rdb [rdb-filename.rdb]`

6.1.6.2 AOF

AOF持久化方式，會將[amdc](#)中的數據以aof格式寫入到文本文件中，存儲到[appendonlydir](#)文件夾下，並且[amdc](#)將會以追加的方式將新的請求追加到文本中。

6.1.6.2.1 生成AOF文件的方式

1. 在配置文件中將[AppendOnly](#)改為"yes"即可。

6.1.6.2.2 AOF數據恢復方式

1. 將一個或多個AOF文件或整個[appendonlydir](#)文件夾放到緩存的配置文件[dir](#)中指定的目錄下，啟動[amdc](#)將會自動加載AOF數據。

6.1.6.2.3 恢復損壞的AOF文件

amdc提供AOF文件修復工具，但是此工具使用的前提是，必須是最後一個追加文件損壞才能修復。這麼設計的原因是，若是base文件就已經損壞，就意味著大部分的數據都已經丟失，沒有修復的必要。

使用方式：`./aof-check-aof --fix [filename.aof|filename.mainfest]`

6.1.7 命令審計

AMDC提供[monitor](#)命令，可以監控所有的請求信息。配合shell客戶端實現監控內容輸出到特定的文件中，以此來實現。

請求信息的記錄格式：`時間戳 [數據庫號 客戶端地址：客戶端端口號] 命令`

使用方式：`amdc-cli -h [ip] -p [port] [-a password] monitor >amdc_commands.log`

6.2 amdc-cli客户端使用介绍

为了方便AMDC使用，shell客户端给用户带来方便快捷的命令行使用方式，并提供帮助建立集群、管理集群slot、LRU测试等实用功能。

6.2.1 客户端参数

使用方式：`amdc-cli [参数] [命令 [命令参数 ...]]`

参数	说明
-h [hostname]	缓存核心的ip（默认值：127.0.0.1）
-p [port]	缓存核心的端口（默认值：6359）
-a [password]	缓存核心的requirepass密码，可以使用AMDCCLI_AUTH环境变量来设置和输入密码，这样会更安全
--user [username]	使用ACL用户登录时的用户名
--pass [password]	使用ACL用户登录时对应的密码
--askpass	无视AMDCCLI_AUTH环境变量，强制使用直接输入的密码
-u [uri]	缓存核心的uri地址（兼容Redis协议）
-r [repeat]	重复执行特定的命令[repeat]次
-i [interval]	使用-r的时候，设置每隔多少秒执行重复一次
-n [db]	数据库编号（默认有16个数据库，编号0-15）
-3	切换为RESP3协议
-x	从stdin读取的输入作为amdc-cli的最后一个参数[br/]例： <code>amdc-cli -x set key [/opt/file</code>
-d [delimiter]	原始格式的响应块之间的分隔符(如：\n)
-D [delimiter]	多个原始格式的响应之间分隔符(如：\n)
-c	连接集群模式
-e	当命令执行错误的時候返回错误代码
--raw	使用原始格式输出（当tty不是默认输出设备时）

--no-raw	強制格式化輸出，即使標準輸出不是tty
--quoted-input	強制將輸入作為帶引號的字符串處理
--csv	輸出為CSV格式
--show-pushes [yn]	是否打印 resp3 推送消息，默認開啟
--stat	動態打印緩存核心的狀態信息：內存/客戶端連接數等
--lru-test	模擬具有 80-20 分佈（key的使用度）的緩存工作負載
--replica	模擬一個備份節點展示從主節點接收到的命令
--slave [host:ip]	指定一個節點為當前連接節點的主節點
--rdb [filename]	從緩存核心獲取一個RDB文件作為本地文件
--pipe	從stdin傳輸原始的amdc協議（兼容Redis的）到緩存核心
--pipe-timeout [n]	pipe模式下的超時時間
--bigkeys	查找value內存佔用大的key
--memkeys	查找key-value內存佔用大的key
--memkeys-samples	查找key-value內存佔用大的key，輸出的信息更簡潔
--hotkeys	查找使用次數多的key，但是需要緩存策略為lfu
--scan	用scan命令列出所有的key
--pattern [pat]	使用--scan/--bigkeys/--hotkeys的時候進行key的正則表達式匹配，默認為*
--quoted-pattern [pat]	和--pattern差不多，但是指定字符串類型需要用引號括起來，否則會被認為是非二進制安全字符串
--intrinsic-latency [sec]	系統固有延遲測試，測試會持續多少秒
--eval [file]	發送和執行LUA腳本文件
--ldb	和--eval一起使用，啟用amdc lua debugger（調試器）
--ldb-sync-mode	和--ldb一樣，但是會與debugger同步，緩存核心將會被阻塞
--cluster [command] [args...] [opts...]	集群管理命令和參數，詳細命令和參數可以使用
--cluster help	查看幫助信息

--verbose	詳細模式
--no-auth-warning	使用-a直接輸入密碼時不顯示warning信息
--help	幫助信息
--version	amdc-cli 版本信息

使用方式：amdc-cli --cluster [命令 [命令參數 ...]]

一級參數	二級參數	說明
create host1:port1 ... hostN:portN		創建集群
	--cluster-replicas [arg]	從節點個數
check host:port		檢查集群
	--cluster-search-multiple-owners	檢查是否有槽同時被分配給了多個節點
info host:port		查看集群狀態
fix host:port		修復集群
	--cluster-search-multiple-owners	修復槽的重複分配問題
reshard host:port		指定集群的任意一節點進行遷移slot，重新分 slots
	--cluster-from [arg]	需要從哪些源節點上遷移slot，可從多個源節點完成遷移，以逗號隔開，傳遞的是節點的 node id，還可以直接傳遞--from all，這樣源節點就是集群的所有節點，不傳遞該參數的話，則會在遷移過程中提示用戶輸入
	--cluster-to [arg]	slot需要遷移的目的節點的node id，目的節點只能填寫一個，不傳遞該參數的話，則會在遷移過程中提示用戶輸入
	--cluster-slots [arg]	需要遷移的slot數量，不傳遞該參數的話，則會在遷移過程中提示用戶輸入。
	--cluster-yes	指定遷移時的確認輸入
	--cluster-timeout [arg]	設置migrate命令的超時時間

	--cluster-pipeline [arg]	定義cluster getkeysinslot命令一次取出的key數量，不傳的話使用默認值為10
	--cluster-replace	是否直接replace到目標節點
rebalance host:port		指定集群的任意一節點進行平衡集群節點slot數量
	--cluster-weight [node1=w1...nodeN=wN]	指定集群節點的權重
	--cluster-use-empty-masters	設置可以讓沒有分配slot的主節點參與，默認不允許
	--cluster-timeout [arg]	設置migrate命令的超時時間
	--cluster-simulate	模擬rebalance操作，不會真正執行遷移操作
	--cluster-pipeline [arg]	定義cluster getkeysinslot命令一次取出的key數量，默認值為10
	--cluster-threshold [arg]	遷移的slot閾值超過threshold，執行rebalance操作
	--cluster-replace	是否直接replace到目標節點
add-node new_host:new_port existing_host:existing_port		添加節點，把新節點加入到指定的集群，默認添加主節點
	--cluster-slave	新節點作為從節點，默認隨機一個主節點
	--cluster-master-id [arg]	給新節點指定主節點
del-node host:port node_id		刪除給定的一個節點，成功後關閉該節點服務
call host:port command arg arg .. arg		在集群的所有節點執行相關命令
set-timeout host:port milliseconds		設置cluster-node-timeout
import host:port		將外部amdc數據導入集群(非cluster模式的節點)
	--cluster-from [arg]	將指定實例的數據導入到集群
	--cluster-copy	migrate時指定copy

--cluster-replace

migrate時指定replace

6.2.2 客戶端使用

AMDC客戶端語言涵蓋了主流的編程語言，例如Java、PHP、Python、C、C++、Node.js等，可以通過amdc-cli進行連接，同時也支持Redis客戶端連接amdc進行操作，兩者效果一致。

6.2.2.1 一般操作

- 通過url方式連接amdc-server

```
./amdc-cli -u amdc://user:password@127.0.0.1:6359/db
```

- 顯示amdc-cli 命令幫助信息 --help

略。

- 使用amdc-cli客戶端連接amdc

```
./amdc-cli -h host -p port -a password
```

-h 用於指定 ip

-p 用於指定端口

-a 用於指定認證密碼

- 將返回數據輸出到當前命令行 --raw（隱藏數據類型）和--no-raw
- 連續運行n次相同命令 -r 設置運行間隔時間-i（秒）

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 -r 5 -i 2 incr k3
(integer) 9
(integer) 10
(integer) 11
(integer) 12
(integer) 13
[root@linux-4-190 loopier]#
```

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 -r 5 -i 2 incr k3
(integer) 9
(integer) 10
(integer) 11
(integer) 12
(integer) 13
[root@linux-4-190 loopier]#
```

- 連接指定db： -n

```
[root@linux-4-190 ~]# ./amdc-cli -h 127.0.0.1 -p 6359 -n 2
127.0.0.1:6359[2]>
127.0.0.1:6359[2]>
127.0.0.1:6359[2]> set k5 v
OK
127.0.0.1:6359[2]> get k5
"v"
127.0.0.1:6359[2]>
```

- 執行命令輸出以逗號分隔的格式(csv格式) --csv

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --csv lrange k4 0 -1
"6","5","4","3","2","1","0"
[root@linux-4-190 loopier]#
```

- 標準輸入 (stdin) 讀取數據作為amdc-cli的最後一個參數 -x

```
127.0.0.1:6359>
[root@linux-4-190 loopier]# echo "amdc-cli v2.0" | ./amdc-cli -h 127.0.0.1 -p 6359 -x set k7
OK
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359
127.0.0.1:6359> get k7
"amdc-cli v2.0\n"
127.0.0.1:6359>
127.0.0.1:6359>
```

- rdb備份 --rdb

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --rdb 1.rdb  
SYNC sent to master, writing 249 bytes to '1.rdb'  
Transfer finished with success.  
[root@linux-4-190 loopier]# cat 1.rdb  
REDIS0009♦ctime♦2cused-mem♦repl-stream-db♦repl-id(dd399ab99571c81dce95f06f6f9167d9f87d944c♦  
repl-offset♦7amdcli v2.0
```

- 從機模式，將客戶端當作服務端的從節點 --slave

```

127.0.0.1:6359>
[127.0.0.1:6359]# ./amdc-cli -h 127.0.0.1 -p 6359
127.0.0.1:6359>
127.0.0.1:6359>
127.0.0.1:6359>
127.0.0.1:6359> set k10 v10
OK
127.0.0.1:6359>

```

```

[127.0.0.1:6359]# ./amdc-cli -h 127.0.0.1 -p 6359 --slave
SYNC with master, discarding 278 bytes of bulk transfer...
SYNC done. Logging commands from master.
"PING",
"PING",
"PING",
"PING",
"PING",
"PING",
"PING",
"PING",
"PING",
"SELECT","0"
"set","k10","v10",
"PING",

```

- 管道模式，一次執行多個命令 --pipe

```

[127.0.0.1:6359]# cat a.txt
set k50 v50
rpush k60 1 2 3 4 5 6 7 8 9
hset k70 key70 val70
[127.0.0.1:6359]# cat a.txt | ./amdc-cli -h 127.0.0.1 -p 6359 --pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 3
[127.0.0.1:6359]# ./amdc-cli -h 127.0.0.1 -p 6359
127.0.0.1:6359> get k50
"v50"
127.0.0.1:6359> lrange k60 0 -1
1) "1"
2) "2"
3) "3"
4) "4"
5) "5"
6) "6"
7) "7"
8) "8"
9) "9"
127.0.0.1:6359> HGET k70 key70
"val70"
127.0.0.1:6359>

```

6.2.2.2 統計操作

- 連續統計模式，實時查看amdc key個數，佔用內存等數據 --stat

```
[root@linux-4-190 ~]#
[root@linux-4-190 ~]# ./amdc-cli -h 127.0.0.1 -p 6359 --stat
----- data ----- load ----- child -
keys      mem      clients blocked requests      connections
6         1.04M    2         0         26778 (+0)      37
6         1.13M    2         0         26779 (+1)      37
6         1.18M    2         0         26780 (+1)      37
6         1.22M    2         0         26781 (+1)      37
6         1.26M    2         0         26782 (+1)      37
6         1.30M    2         0         26783 (+1)      37
6         1.34M    2         0         26784 (+1)      37
```

- 查找大key --bigkeys

```
[root@linux-4-190 ~]#
[root@linux-4-190 ~]# ./amdc-cli -h 127.0.0.1 -p 6359 --bigkeys

# Scanning the entire keyspace to find biggest keys as well as
# average sizes per key type. You can use -i 0.1 to sleep 0.1 sec
# per 100 SCAN commands (not usually needed).

[00.00%] Biggest string found so far 'k6' with 1024 bytes
[00.00%] Biggest list found so far 'k4' with 4 items

----- summary -----

Sampled 6 keys in the keyspace!
Total key length in bytes is 12 (avg len 2.00)

Biggest string found 'k6' has 1024 bytes
Biggest list found 'k4' has 4 items

0 hashes with 0 fields (00.00% of keys, avg size 0.00)
0 sets with 0 members (00.00% of keys, avg size 0.00)
5 strings with 1047 bytes (83.33% of keys, avg size 209.40)
0 zsets with 0 members (00.00% of keys, avg size 0.00)
1 lists with 4 items (16.67% of keys, avg size 4.00)
0 streams with 0 entries (00.00% of keys, avg size 0.00)
[root@linux-4-190 ~]#
```

- 查找热key --hotkeys

```
# Scanning the entire keyspace to find biggest keys as well as
# average sizes per key type.  You can use -i 0.1 to sleep 0.1 sec
# per 100 SCAN commands (not usually needed).

[00.00%] Biggest string found so far '"k7"' with 2 bytes
[00.00%] Biggest zset found so far '"k30"' with 3 members
[00.00%] Biggest string found so far '"k18"' with 3 bytes
[00.00%] Biggest list found so far '"k4"' with 8 items
[50.00%] Biggest set found so far '"k40"' with 3 members
[50.00%] Biggest hash found so far '"k5"' with 2 fields

----- summary -----

Sampled 20 keys in the keyspace!
Total key length in bytes is 51 (avg len 2.55)

Biggest list found '"k4"' has 8 items
Biggest hash found '"k5"' has 2 fields
Biggest string found '"k18"' has 3 bytes
Biggest set found '"k40"' has 3 members
Biggest zset found '"k30"' has 3 members

1 lists with 8 items (05.00% of keys, avg size 8.00)
1 hashes with 2 fields (05.00% of keys, avg size 2.00)
16 strings with 41 bytes (80.00% of keys, avg size 2.56)
0 streams with 0 entries (00.00% of keys, avg size 0.00)
1 sets with 3 members (05.00% of keys, avg size 3.00)
1 zsets with 3 members (05.00% of keys, avg size 3.00)
[root@linux-4-190 loopier]#
```

6.2.2.3 查詢操作

- 獲取所有鍵列表 --scan

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --scan
k6
k7
k1
k4
k2
k3
[root@linux-4-190 loopier]#
```

- 掃描指定的key --pattern (正則表達式)

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --scan --pattern k*
k6
k7
k1
k4
k2
k3
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --scan --pattern *6
k6
[root@linux-4-190 loopier]#
```

- 查看一段時間內 amdc的最小、最大、平均訪問延遲 --latency或--latency-history

```
[root@linux-4-190 loopier]#
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --latency
min: 0, max: 2, avg: 0.19 (2561 samples)
```

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --latency-history
min: 0, max: 4, avg: 0.24 (1146 samples)
```

6.2.2.4 測試操作

- 測量n秒內amdc的延遲時間 --intrinsic-latency

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --intrinsic-latency 5
Max latency so far: 6 microseconds.
Max latency so far: 10 microseconds.
Max latency so far: 12 microseconds.
Max latency so far: 13 microseconds.
Max latency so far: 18 microseconds.
Max latency so far: 20 microseconds.
Max latency so far: 28 microseconds.
Max latency so far: 49 microseconds.
Max latency so far: 382 microseconds.
Max latency so far: 400 microseconds.
Max latency so far: 1170 microseconds.

1236720 total runs (avg latency: 4.0430 microseconds / 4042.95 nanoseconds per run).
Worst run took 289x longer than the average latency.
[root@linux-4-190 loopier]#
```

- LRU 模擬，模擬緩存淘汰策略 --lru-test

```
[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --lru-test 100000
28000 Gets/sec | Hits: 27884 (99.59%) | Misses: 116 (0.41%)
28250 Gets/sec | Hits: 28250 (100.00%) | Misses: 0 (0.00%)
28000 Gets/sec | Hits: 28000 (100.00%) | Misses: 0 (0.00%)
27000 Gets/sec | Hits: 26969 (99.89%) | Misses: 31 (0.11%)
29000 Gets/sec | Hits: 28929 (99.76%) | Misses: 71 (0.24%)
28750 Gets/sec | Hits: 28738 (99.96%) | Misses: 12 (0.04%)
29500 Gets/sec | Hits: 29500 (100.00%) | Misses: 0 (0.00%)
28500 Gets/sec | Hits: 28500 (100.00%) | Misses: 0 (0.00%)
28500 Gets/sec | Hits: 28382 (99.59%) | Misses: 118 (0.41%)
28250 Gets/sec | Hits: 28250 (100.00%) | Misses: 0 (0.00%)
29500 Gets/sec | Hits: 29500 (100.00%) | Misses: 0 (0.00%)
29250 Gets/sec | Hits: 29250 (100.00%) | Misses: 0 (0.00%)
28750 Gets/sec | Hits: 28750 (100.00%) | Misses: 0 (0.00%)
28500 Gets/sec | Hits: 28251 (99.13%) | Misses: 249 (0.87%)
28750 Gets/sec | Hits: 28750 (100.00%) | Misses: 0 (0.00%)
28750 Gets/sec | Hits: 28500 (99.13%) | Misses: 250 (0.87%)
29500 Gets/sec | Hits: 29500 (100.00%) | Misses: 0 (0.00%)
26250 Gets/sec | Hits: 26022 (99.13%) | Misses: 228 (0.87%)
```

- 模擬顯示從主服務器接收到的命令的副本 --replica

```
[root@linux-4-190 loopier]# redis-cli -h 127.0.0.1 -p 6359 --replica
sending REPLCONF capa eof
SYNC with master, discarding 478 bytes of bulk transfer...
SYNC done. Logging commands from master.
"PING"
"PING"
"PING"
"PING"
"PING"
"PING"
"PING"
"PING"
"PING"
"PING"
"PING"
```

6.2.2.5 LUA操作

- 執行lua腳本

```
[root@linux-4-190 loopier]# cat script.lua
return redis.call('GET','k1')

[root@linux-4-190 loopier]# ./amdc-cli -h 127.0.0.1 -p 6359 --eval ./script.lua
"v1"
[root@linux-4-190 loopier]# █
```

- 啟動lua調試模式：--ldb 或 --ldb-sync-mode

```
lua debugger>
[root@linux-4-190 loopier]# redis-cli -h 127.0.0.1 -p 6359 --eval ./script.lua --ldb
Lua debugging session started, please use:
quit      -- End the session.
restart   -- Restart the script in debug mode again.
help      -- Show Lua script debugging commands.

v1
lua debugger>
lua debugger>
[root@linux-4-190 loopier]# redis-cli -h 127.0.0.1 -p 6359 --eval ./script.lua --ldb-sync-mode
Lua debugging session started, please use:
quit      -- End the session.
restart   -- Restart the script in debug mode again.
help      -- Show Lua script debugging commands.

v1
lua debugger>
lua debugger> █
```

6.2.2.6 集群操作

目前多租戶版本不支持集群操作

6.3 RDB集群數據遷移工具使用介紹

目前多租戶版本不支持集群操作

6.4 性能測試工具使用介紹

amdc-benchmark，是提供給用戶測試AMDC的專用性能測試工具，以便快速了解AMDC的性能情況，並為調優提供可靠依據。

6.4.1 AMDC benchmark參數

amdc-benchmark

舉例: `amdc-benchmark [-h <host>] [-p <port>] [-c <clients>] [-n <requests>]`

命令參數

```

-h <hostname>      實例host名稱（默認127.0.0.1）
-p <port>           實例端口（默認6359）
-a <password>       實例密碼
-c <clients>        並發連接數（默認50）
-n <requests>       總請求數（默認100000）
-d <size>           SET/GET 數據的大小，以bytes為單位（默認2）
-dbnum <db>         指定db庫（默認AMDC有0-15號庫，當前參數默認0）
-k <boolean>        1=保持連接 0=重新連接（默認1）
-r <keyspacelen>    SET/GET/INCR 使用隨機 key， SADD 使用隨機值，
keyspacelen指隨機數據的最大長度（最大長度範圍是0-12）
-P <numreq>         使用管道合併指定數量的請求，默認1（不使用管道）
-e                 若服務器返回信息錯誤，在標準輸出中顯示（每秒顯示錯誤不超過1
個）
-q                 僅顯示 query/sec 值
--csv              輸出為CSV格式
-l                循環，一直進行測試
-t <tests>          指定測試命令
-I                空閒模式，開N個空閒連接並等待
--threads <num>    開啟指定數量的線程，默認1

```

memtier-benchmark參數

舉例: `mementier-benchmark [-h <host>] [-p <port>] [-c <clients>] [-n <requests>] [-t <threads>]`

命令參數

<code>-h <hostname></code>	實例host名稱 (默認127.0.0.1)
<code>-p <port></code>	實例端口 (默認6359)
<code>-s <socket></code>	使用Unix域套接字
<code>-a <password></code>	認證密碼
<code>--tls</code>	開啟TLS加密
<code>-c <clients></code>	客戶端數量
<code>-t <threads></code>	線程數
<code>--ratio</code>	讀寫比例
<code>--pipeline <num></code>	管道命令數
<code>--randomdata</code>	使用隨機數據
<code>--hide-histogram</code>	隱藏詳細的直方圖信息
<code>--csv <file></code>	輸出csv格式文件
<code>--test-time <sec></code>	持續測試時間

7 產品所有配置說明

7.1 緩存核心所有配置

AMDC緩存核心配置三個可主動配置的配置文件，分別為緩存配置（amdc.yaml）、哨兵配置（sentinel.yaml）和授權中心配置（acls.properties）。

7.1.1 緩存配置文件

AMDC緩存核心的配置文件存放在：/安裝根目錄/amdc/amdc.yaml，部分配置可以通過控制台進行修改，以下為AMDC緩存核心詳細配置：

參數名稱	默認值	備註
bind	127.0.0.1,:::1	監聽的網絡地址，默認僅監聽本地迴環地址；-前綴表示地址不可用時不影響啟動
bind-source-addr	""	強制所有出站連接都從指定的本地IP發送，默認不設置，由操作系統路由表自動選擇
port	6359	監聽的 TCP 端口，設為 0 則不監聽 TCP 套接字
protected-mode	yes	保護模式，當未指定 bind 且無密碼時，僅允許本地迴環地址和 Unix 域套接字連接
mptcp-enabled	no	MPTCP 多路徑傳輸控制協議，支持多鏈路並發傳輸，提升鏈路聚合帶寬與傳輸吞吐，需依賴 Linux 5.6 及以上內核版本
tcp-backlog	511	TCP 監聽的 backlog 大小，實際值可能受系統 somaxconn 和 tcp_max_syn_backlog 限制
unixsocket	""	Unix 域套接字路徑，未指定則不監聽 Unix 套接字
unixsocketperm	0	Unix 域套接字的權限
timeout	0	客戶端空閒超時時間（秒），0 表示禁用
tcp-keepalive	300	TCP 保活時間（秒），用於檢測死連接和維持網絡連接
socket-mark-id	0	監聽套接字指定標記 ID，用於實現高級路由與流量過濾能力，默認 0 不開啟
daemonize	no	是否以守護進程模式運行

supervised	no	與監控系統（upstart/systemd）的交互模式，默認不交互
pidfile	"/var/run/amdc_6359.pid"	pid 文件路徑
loglevel	notice	日誌級別：debug、verbose、notice、warning
logfile	""	日誌文件路徑，空字符串表示輸出到標準輸出
syslog-enabled	no	是否啟用系統日誌
syslog-ident	"AMDC"	系統日誌標識
syslog-facility	local0	系統日誌設備，需為 USER 或 LOCAL0-LOCAL7
crash-log-enabled	yes	是否啟用崩潰日誌
crash-memcheck-enabled	yes	是否在崩潰日誌中啟用快速內存檢查
worker-threads	1	設置工作線程數，默認為 1
databases	16	數據庫數量，默認使用 DB 0
always-show-logo	no	是否在啟動日誌中顯示 ASCII 藝術 logo
set-proc-title	yes	是否修改進程標題以顯示運行時信息
proc-title-template	"{title} {listen-addr} {server-mode}"	進程標題模板，支持多個變量
locale-collate	""	字符串比較 / 排序用的語言區域規則
loadmodule	""	啟動時加載指定的模塊，若加載失敗服務器會終止啟動，可多次使用以加載多個模塊
tls-port	0	TLS 監聽端口，需配合 port 0 使用以僅啟用 TLS
tls-cert-file	""	服務器 TLS 證書文件路徑（PEM 格式）
tls-key-file	""	服務器 TLS 私鑰文件路徑（PEM 格式）
tls-key-file-pass	""	TLS 私鑰文件的密碼（若加密）
tls-client-cert-file	""	客戶端 TLS 證書文件路徑（用於客戶端身份驗證）
tls-client-key-file	""	客戶端 TLS 私鑰文件路徑
tls-client-key-file-pass	""	客戶端 TLS 私鑰文件的密碼（若加密）

tls-dh-params-file	""	DH 參數文件路徑
tls-ca-cert-file	""	用於驗證客戶端和節點的 CA 證書文件路徑
tls-ca-cert-dir	""	用於驗證客戶端和節點的 CA 證書目錄路徑
tls-auth-clients	yes	客戶端證書驗證策略，默認 yes，TLS 端口上的所有客戶端都必須使用合法的客戶端證書進行身份認證
tls-auth-clients-user	off	開啟 TLS 客戶端證書自動登錄，CN:服務端從客戶端 TLS 證書裡讀取 CN 字段，off:關閉證書自動鑒權
tls-replication	no	是否在主從複製中啟用 TLS
tls-cluster	no	是否在集群中啟用 TLS
tls-protocols	""	允許的 TLS 版本，大小寫不敏感
tls-ciphers	""	TLSv1.2 及以下允許的密碼套件，國密使用 ECC-SM2-SM4-GCM-SM3:ECC-SM2-SM4-CBC-SM3
tls-ciphersuites	""	TLSv1.3 允許的密碼套件，國密使用 TLS_SM4_GCM_SM3
tls-prefer-server-ciphers	no	是否優先使用服務器的密碼套件偏好
tls-session-caching	yes	是否啟用 TLS 會話緩存以加速重連
tls-session-cache-size	20480	TLS 會話緩存的最大條目數，0 表示無限制
tls-session-cache-timeout	300	TLS 會話緩存的超時時間（秒）
save	3600 1,300 100,60 10000	RDB 持久化觸發條件，格式為 <秒數> <修改次數>; save "" 禁用自動保存 RDB
stop-writes-on-bgsave-error	no	RDB 持久化失敗時是否停止接受寫入
rdbcompression	yes	是否使用 LZF 壓縮 RDB 文件中的字符串對象
rdbchecksum	yes	是否在 RDB 文件末尾添加 CRC64 校驗和，禁用可提升性能
sanitize-dump-payload	no	加載 RDB 或 RESTORE 數據時是否進行全面校驗，默認因集群遷移問題暫設為 no

dbfilename	"dump.rdb"	RDB 文件名稱
rdb-del-sync-files	no	無持久化配置時是否刪除複製用的 RDB 文件，僅在 AOF 和 RDB 均禁用時有效
dir	"/"	數據文件（RDB、AOF）的存放目錄
replicaof	""	主從複製配置，eg: replicaof: " "
masterauth	""	主服務器的認證密碼，eg: masterauth: ""
masteruser	""	用於複製的主服務器用戶名，eg: masteruser: ""
repl-mptcp-enabled	no	主從複製是否開啟MPTCP
replica-serve-stale-data	yes	複製同步期間是否響應客戶端請求（可能返回過期數據）
replica-read-only	yes	從服務器是否為只讀模式
repl-diskless-sync	no	是否啟用無盤複製（直接通過套接字傳輸 RDB）
repl-diskless-sync-delay	5	無盤複製時等待新從節點的延遲時間（秒）
repl-diskless-sync-max-replicas	0	無盤複製時，提前觸發同步的副本數量閾值
repl-diskless-load	disabled	從節點加載 RDB 的方式，disabled（先存盤）、on-empty-db（僅空庫時無盤）、swapdb（內存保留舊數據）
repl-ping-replica-period	10	從節點向主節點發送 PING 的間隔（秒）
repl-timeout	60	複製超時時間（秒），涵蓋 SYNC 傳輸、主從心跳等
repl-disable-tcp-nodelay	no	複製鏈接是否禁用 TCP_NODELAY，yes 可減少帶寬但增加延遲
repl-backlog-size	1mb	複製積壓緩衝區大小，影響部分重同步能力
repl-backlog-ttl	3600	最後一個從節點斷開後，積壓緩衝區釋放的延遲時間（秒），0 表示不釋放
replica-priority	100	從節點優先級，用於 Sentinel 選主，0 表示不可選為主

propagation-error-behavior	ignore	當主從複製 / AOF 恢復時遇到執行失敗時服務端行為，ignore（默認）遇到錯誤命令跳過，繼續運行；panic 遇到錯誤，直接崩潰退出；panic-on-replicas 只有副本節點遇到複製流錯誤時才崩潰
replica-ignore-disk-write-errors	no	副本節點寫入磁盤失敗時，服務端行為，no（默認）直接崩潰，yes 忽略錯誤繼續運行
replica-announced	yes	從節點是否被 Sentinel 識別、上報，管控端、上層客戶端能感知此從節點
min-replicas-to-write	0	主節點接受寫入所需的最少從節點數，0 禁用此功能
min-replicas-max-lag	10	從節點的最大滯後時間（秒），配合 min-replicas-to-write 使用
replica-announce-ip	""	從節點向主節點公佈的 IP 地址（用於 NAT/Docker 等場景）
replica-announce-port	0	從節點向主節點公佈的端口（用於 NAT/Docker 等場景）
tracking-table-max-keys	1000000	客戶端緩存跟蹤表的最大鍵數，0 表示無限制
user	""	user: " "
acllog-max-len	128	ACL 日誌的最大條目數
aclfile	""	外部 ACL 用戶文件路徑，與配置文件內用戶配置不可混用
requirepass	""	默認用戶的密碼（等價於 ACL 的 default 用戶密碼）
acl-pubsub-default	allchannels	新用戶默認的 Pub/Sub 頻道權限，allchannels（允許所有）、resetchannels（禁止所有）
rename-command	""	命令重命名，eg: rename-command: "flushall """, rename-command: "flushall no-flushall"
maxclients	10000	最大並發客戶端連接數，受系統文件描述符限制
maxmemory	0	內存使用上限，達到後按 maxmemory-policy 處理
maxmemory-policy	volatile-lru	內存達到上限時的淘汰策略，如 volatile-lru、allkeys-lfu 等

maxmemory-samples	5	LRU/LFU/TTL 算法的採樣數，越大越精確但 CPU 消耗越高
maxmemory-eviction-tenacity	10	淘汰處理的激進程度，0（最低延遲）到 100（優先處理淘汰）
replica-ignore-maxmemory	yes	從節點是否忽略自身的 maxmemory 配置（默認由主節點處理淘汰）
active-expire-effort	1	主動過期鍵的掃描力度，1-10，越大越頻繁（可能增加 CPU 和延遲）
lazyfree-lazy-eviction	no	內存淘汰時是否使用惰性刪除（後台線程釋放內存）
lazyfree-lazy-expire	no	鍵過期時是否使用惰性刪除
lazyfree-lazy-server-del	no	服務器內部刪除鍵（如 RENAME 覆蓋）時是否使用惰性刪除
replica-lazy-flush	no	從節點全量同步時是否惰性清空數據庫
lazyfree-lazy-user-del	no	用戶執行 DEL 命令時是否使用惰性刪除
lazyfree-lazy-user-flush	no	用戶執行 FLUSH 命令未指定同步 / 異步時的默認行為
oom-score-adj	no	是否控制內核 OOM killer 的進程評分，yes/relative 為相對調整，absolute 為絕對設置
oom-score-adj-values	0 200 800	OOM 評分調整值，分別對應主節點、從節點、後台子進程
disable-thp	yes	是否禁用內核透明大頁（THP），避免 fork 和 CoW 的性能問題
appendonly	no	是否啟用 AOF 持久化
appendfilename	"appendonly.aof"	AOF 文件名稱
appenddirname	"appendonlydir"	所有 appendonly 文件的專用存儲目錄
appendfsync	everysec	AOF 刷盤策略：no（依賴系統）、always（每次寫入）、everysec（每秒一次，默認）
no-appendfsync-on-rewrite	no	AOF 重寫期間是否禁用主進程的 fsync，可能增加數據丟失風險但提升性能

auto-aof-rewrite-percentage	100	AOF 自動重寫的觸發百分比（當前大小較上次重寫增長比例）
auto-aof-rewrite-min-size	64mb	AOF 自動重寫的最小文件大小
aof-load-truncated	yes	啟動時是否加載被截斷的 AOF 文件，no 則會報錯退出
aof-use-rdb-preamble	yes	AOF 重寫時是否使用 RDB 前綴（提升重寫和恢復速度）
aof-timestamp-enabled	no	是否支持在 AOF 中記錄時間戳，以實現從特定時間點恢復數據的功能
shutdown-timeout	10	主節點關機 / 停止服務時，最大等待副本同步的超時時間（單位：秒），設為 0 關閉該等待機制，立刻停機
shutdown-on-sigint	save	服務端收到 SIGINT(Ctrl+C) 時的關機行為，default 若配置了自動持久化 save 規則，則阻塞生成 RDB 快照; save 無視自動 save 配置，強制落地 RDB 持久化; nosave 即使配置了 save 規則，停機也不生成 RDB; now 直接跳過同步等待，快速關機; force 忽略各類異常阻礙，強制退出進程
shutdown-on-sigterm	save	服務端收到 SIGTERM 時的關機行為，同上
busy-reply-threshold	5000	Lua 腳本的最大執行時間（毫秒），超時後僅允許 SCRIPT KILL 和 SHUTDOWN NOSAVE
cluster-enabled	no	是否啟用集群模式
cluster-config-file	"node-6359.conf"	集群配置文件路徑，由節點自動維護
cluster-node-timeout	15000	集群節點超時時間（毫秒），用於判斷節點故障
cluster-port	0	集群內部通信專用端口，默認 0 時自動用業務端口 + 10000
cluster-replica-validity-factor	10	從節點故障轉移的有效性因子，影響故障轉移條件
cluster-migration-barrier	1	從節點遷移的屏障（原主節點需保留的從節點數）
cluster-allow-	yes	是否允許從節點自動遷移到無從節點的主節點

replica-migration		
cluster-require-full-coverage	yes	集群是否需要所有哈希槽都被覆盖才接受请求
cluster-replica-no-failover	no	从节点是否禁止自动故障转移（仍可手动触发）
cluster-allow-reads-when-down	no	集群不可用时，节点是否仍响应其负责槽的读请求
cluster-allow-pubsubshard-when-down	yes	集群全局异常下线时，是否允许节点基于本地槽位所有权，持续提供分片發佈訂閱服務，保障消息鏈路高可用
cluster-link-sendbuf-limit	0	集群节点间通信链路的发送缓冲区内存上限，防止慢节点导致缓冲区无限膨胀、佔满内存，默认 0 不限制
cluster-announce-hostname	""	集群节点对外宣布的主机名 / 域名，用于 TLS SNI、DNS 路由、容器化 / 云环境，默认空表示不启用
cluster-announce-human-nodename	""	为集群节点设置自定义友好名称，作为节点 ID 的补充标识
cluster-preferred-endpoint-type	ip	指定集群向客户端暴露的连接地址类型：ip（默认）、hostname 或 unknown-endpoint，决定 MOVED 重定向与 CLUSTER SLOTS 返回的地址格式
cluster-compatibility-sample-ratio	0	定义集群模式下命令兼容性检查的抽样比例（0-100），用于检测跨槽位等不兼容命令，默认 0 不开启，开启比例越高安全性越强但性能开销越大
cluster-slot-stats-enabled	no	是否启用集群槽位资源统计，支持监控 CPU / 网络 / 内存 使用情况，用于热点识别、负载均衡与槽位迁移优化
cluster-slot-migration-write-pause-timeout	10000	集群槽位迁移时，源节点暂停写入的最大超时时间（毫秒）
cluster-slot-migration-handoff-max-lag-bytes	1mb	设置槽位迁移触发移交阶段的最大允许複製滞后字节数，当源目节点间待同步数据量低于该值时，进入槽位移交流程、源节点暂停写入
cluster-announce-ip	""	集群节点对外公布的 IP 地址（用于 NAT/Docker 场景）
cluster-announce-tls-port	0	集群节点对外公布的 TLS 客户端端口（用于 NAT/Docker 场景）

cluster-announce-port	0	集群節點對外公佈的客戶端端口（用於 NAT/Docker 場景）
cluster-announce-bus-port	0	集群節點對外公佈的總線端口（用於 NAT/Docker 場景）
slowlog-log-slower-than	10000	慢日誌記錄的閾值（微秒），負數禁用，0 記錄所有命令
slowlog-max-len	128	慢日誌的最大條目數，超出後淘汰舊條目
latency-monitor-threshold	0	延遲監控的閾值（毫秒），0 禁用監控
latency-tracking	yes	是否開啟命令延遲追蹤
latency-tracking-info-percentiles	"50 99 99.9"	定義 INFO 展示的延遲百分位，默認 50/99/99.9
notify-keyspace-events	""	鍵空間事件通知配置，詳見配置註釋說明
hash-max-listpack-entries	512	哈希表使用listpack編碼的最大條目數
hash-max-listpack-value	64	哈希表使用listpack編碼的最大值大小（字節）
list-max-listpack-size	-2	列表使用listpack編碼的節點大小限制，-1 至 -5 對應 4Kb 至 64Kb，正數為條目數
list-compress-depth	0	列表的壓縮深度，0 禁用壓縮，正數表示首尾不壓縮的節點數
set-max-intset-entries	512	集合使用整數集編碼的最大條目數（元素為 64 位有符號整數）
set-max-listpack-entries	128	集合使用listpack編碼的最大條目數
set-max-listpack-value	64	集合使用listpack編碼的最大值大小（字節）
zset-max-listpack-entries	128	有序集合使用listpack編碼的最大條目數
zset-max-listpack-value	64	有序集合使用listpack編碼的最大值大小（字節）

hll-sparse-max-bytes	3000	HyperLogLog 使用稀疏表示的最大字节数（含 16 字节头）
stream-node-max-bytes	4kb	流数据结构中单个节点的最大字节数
stream-node-max-entries	100	流数据结构中单个节点的最大条目数
stream-idmp-duration	100	IDMP 消息唯一标识保留时长（秒），默认 100 秒，超时自动清理
stream-idmp-maxsize	100	每个生产者每个流最多记录的 ID 数量，默认 100，满了自动淘汰最舊
activeresharding	yes	是否启用主动重哈希，定期释放哈希表内存
client-output-buffer-limit normal	0 0 0	普通客户端的输出缓冲区限制（硬限制、软限制、软限制持续时间）
client-output-buffer-limit replica	0 0 0	从节点客户端的输出缓冲区限制
client-output-buffer-limit pubsub	32mb 8mb 60	Pub/Sub 客户端的输出缓冲区限制
client-query-buffer-limit	1gb	客户端查询缓冲区的最大大小
lookahead	16	设置客户端管道命令提前解码与预取的數量
maxmemory-clients	5%	客户端内存驱逐机制：限制所有客户端连接总内存上限，超阈值则优先断开佔用最大的连接，防止 OOM
proto-max-bulk-len	512mb	协议中批量请求的最大大小（字节），需≥1mb
hz	10	后台任务执行频率（赫兹），1-500，越高響應性越好但 CPU 消耗增加
dynamic-hz	yes	是否启用动态 Hz，客户端多时自动提高频率
aof-rewrite-incremental-fsync	yes	AOF 重寫時是否每生成 32MB 數據執行一次 fsync，減少延遲峰值
rdb-save-incremental-fsync	yes	RDB 保存時是否每生成 32MB 數據執行一次 fsync，減少延遲峰值
lfu-log-factor	10	LFU 淘汰算法的計數器對數因子，影響計數器增長速度

lfu-decay-time	1	LFU 計數器的衰減時間（分鐘），超過此時長計數器減半
max-new-connections-per-cycle	10	限制每個事件循環週期能接收多少個客戶端連接，防止瞬間大量新連接把服務器卡死
max-new-tls-connections-per-cycle	1	限制每個事件循環週期能接收多少個TLS加密客戶端連接，防止瞬間大量新連接把服務器卡死
activedefrag	no	是否啟用主動內存碎片整理（需 Jemalloc 支持）
active-defrag-ignore-bytes	100mb	觸發主動碎片整理的最小碎片浪費字節數
active-defrag-threshold-lower	10	觸發主動碎片整理的最小碎片率（百分比）
active-defrag-threshold-upper	100	觸發最大力度碎片整理的碎片率（百分比）
active-defrag-cycle-min	1	碎片整理的最小 CPU 佔比（百分比），達到下限時使用
active-defrag-cycle-max	25	碎片整理的最大 CPU 佔比（百分比），達到上限時使用
active-defrag-max-scan-fields	1000	每次主字典掃描處理的最大字段數（適用於集合、哈希等結構）
jemalloc-bg-thread	yes	是否啟用 Jemalloc 的後台線程進行內存釋放
server-cpulist	""	服務器線程的 CPU 親和性列表（如 "0-7:2" 表示 0、2、4、6 核）
bio-cpulist	""	後台 I/O 線程的 CPU 親和性列表
aof-rewrite-cpulist	""	AOF 重寫子進程的 CPU 親和性列表
bgsave-cpulist	""	BGSAVE 子進程的 CPU 親和性列表
ignore-warnings	""	需要忽略的警告列表（空格分隔），如 "ARM64-COW-BUG"
key-memory-histograms	no	是否開啟按數據類型的鍵內存分配直方圖統計

7.1.2 哨兵配置文件

哨兵配置文件為：`sentinel.yaml`

參數名稱	默認值	備註
bind	127.0.0.1、-::1	監聽的網絡地址，默認僅監聽本地迴環地址
port	26359	監聽的 TCP 端口，設為 0 則不監聽 TCP 套接字
protected-mode	no	保護模式，未指定 bind 且無密碼時僅允許本地迴環連接
unixsocket	空	Unix 域套接字路徑，未指定則不監聽
unixsocketperm	0	Unix 域套接字權限
daemonize	no	是否以守護進程模式運行
pidfile	/var/run/amdc-sentinel.pid	PID 文件路徑
loglevel	notice	日誌級別：debug、verbose、notice、warning
logfile	空	日誌文件路徑，空表示輸出到標準輸出
syslog-enabled	no	是否啟用系統日誌
syslog-ident	AMDC	系統日誌標識
syslog-facility	local0	系統日誌設備，支持 USER / LOCAL0~LOCAL7
dir	./	數據文件存放目錄
maxclients	10000	最大並發客戶端連接數
user	空	ACL 用戶配置，格式：
acllog-max-len	128	ACL 日誌最大條目數
aclfile	空	外部 ACL 用戶文件路徑
requirepass	空	默認用戶密碼
tls-port	0	TLS 監聽端口，需配合 port 0 使用

tls-cert-file	空	服務器 TLS 證書文件 (PEM)
tls-key-file	空	服務器 TLS 私鑰文件 (PEM)
tls-key-file-pass	空	TLS 私鑰密碼
tls-client-cert-file	空	客戶端 TLS 證書文件
tls-client-key-file	空	客戶端 TLS 私鑰文件
tls-client-key-file-pass	空	客戶端 TLS 私鑰密碼
tls-dh-params-file	空	DH 參數文件
tls-ca-cert-file	空	客戶端/節點驗證 CA 證書
tls-ca-cert-dir	空	CA 證書目錄
tls-auth-clients	yes	客戶端證書驗證策略：no=不接受、optional=可選但合法
tls-auth-clients-user	off	TLS 證書自動登錄：CN=讀取證書用戶，off=關閉
tls-replication	no	主從複製是否啟用 TLS
tls-cluster	no	集群是否啟用 TLS
tls-protocols	空	允許的 TLS 版本
tls-ciphers	空	TLSv1.2 及以下密碼套件
tls-ciphersuites	空	TLSv1.3 密碼套件
tls-prefer-server-ciphers	no	是否優先使用服務端密碼套件
tls-session-caching	yes	啟用 TLS 會話緩存
tls-session-cache-size	20480	TLS 會話緩存最大條目數
tls-session-cache-timeout	300	TLS 會話緩存超時時間 (秒)
sentinel monitor	mymaster 127.0.0.1 6359 2	監控主節點：名稱、IP、端口、法定票數
sentinel auth-user	mymaster 空	主從節點認證用戶名
sentinel auth-pass	mymaster 空	主從節點認證密碼
sentinel down-after-milliseconds	mymaster 30000	主觀下線超時 (毫秒)

sentinel sentinel-user	空	Sentinel 集群通信用户名
sentinel sentinel-pass	空	Sentinel 集群通信密码
sentinel parallel-syncs	mymaster 1	故障转移时并发同步从节点数
sentinel failover-timeout	mymaster 180000	故障转移超时（毫秒）
sentinel notification-script	mymaster 空	警告事件通知脚本
sentinel client-reconfig-script	mymaster 空	故障转移后客户端重配置脚本
sentinel deny-scripts-reconfig	yes	禁止运行时修改脚本配置
sentinel rename-command	mymaster 空 空	命令重命名
sentinel resolve-hostnames	no	是否支持主机名解析（依赖 DNS）
sentinel announce-hostnames	no	是否对外公布主机名
sentinel announce-ip	空	NAT/Docker 环境对外公布 IP
sentinel announce-port	0	NAT/Docker 环境对外公布端口
sentinel master-reboot-down-after-period	mymaster 0	主节点重启加载数据期间不触发故障转移

7.1.3 授权认证中心配置

使用认证中心时，填写[acls.properties](#)；需要提供apusic license认证中心地址，认证中心地址必须是ip:port格式，多个地址用，隔开。

注：该配置在旧版的amdc.yaml中存在，新版本中已移除。但仍兼容旧版本。

配置项	默认值	说明
apusic_acls_enable	false	是否开启apusic license认证中心，如果开启则使用true，否则是false
apusic_acls_authUrls	""	认证中心地址，必须是ip:port格式，多个地址用，隔开
apusic_acls_ns	"public"	命名空间
apusic_acls_tenant	""	租户名称

8 產品工具使用說明

8.1 amdc-check-rdb工具使用手冊

9 常見問題處理

amdc-check-rdb是AMDC的RDB文件校驗與修復工具，用於檢查RDB快照文件的完整性、合法性，排查文件損壞問題，並可對部分損壞的RDB文件進行修復，避免因RDB文件損壞導致AMDC啟動失敗或數據丟失。

核心作用：

- 校驗RDB文件是否完整、無損壞，檢測文件格式是否符合AMDC規範；
- 修復因意外中斷（如服務器宕機、斷電）導致損壞的RDB文件；
- 查看RDB文件的基礎信息（如版本、數據量、過期時間等）。

9.0.1 工具位置與運行前提

9.0.1.1 工具位置

AMDC安裝完成後，amdc-check-rdb工具位於AMDC安裝目錄文件夾下：

```
Linux： /amdc/amdc-check-rdb（默認路徑，若自定義安裝需對應調整）
```

9.0.1.2 運行前提

- 無需啟動AMDC服務，可直接在命令行運行；
- 需擁有RDB文件的讀取權限（若文件權限不足，需使用chmod修改權限，Linux）；
- RDB文件路徑需正確（絕對路徑或相對路徑均可，相對路徑以當前終端所在目錄為準）。

9.0.2 基本語法

amdc-check-rdb的語法簡潔，核心僅需指定待處理的RDB文件路徑，可選參數用於修復或查看詳細信息：

```
#基礎語法（校驗RDB文件）
```

```
amdc-check-rdb [options] /path/to/dump.rdb
```

```
#常用簡寫（無選項時，默認僅校驗文件完整性）
```

```
amdc-check-rdb dump.rdb
```

9.0.3 核心選項與參數說明

amdc-check-rdb 選項較少，重點關注修復和調試相關選項，具體如下：

選項	說明	使用場景
--fix	修復損壞的RDB 文件，將可恢復的數據保留，丟棄無法修復的部分	RDB 文件損壞，AMDC啟動失敗時
--help (簡寫-h)	顯示工具的幫助信息，包括所有選項說明和基本語法	忘記命令語法時快速查詢

9.0.4 常見操作示例

9.0.4.1 校驗RDB文件完整性

無需選項，直接指定RDB文件路徑，工具會自動校驗文件是否完整、格式是否正確：

#校驗RDB文件

```
amdc-check-rdb/path/to/dump.rdb
```

正常輸出（文件完整）：

```
[offset 0] Checking RDB file dump.rdb
[offset 29] AUX FIELD amdc-ver = '2.0.6'
[offset 46] AUX FIELD amdc-bits = '64'
[offset 58] AUX FIELD ctime = '1769518212'
[offset 73] AUX FIELD used-mem = '2442688'
[offset 91] AUX FIELD repl-stream-db = '0'
[offset 141] AUX FIELD repl-id =
'069cb616c524d6ec885da6218b669f1d9c970125'
[offset 156] AUX FIELD repl-offset = '0'
[offset 172] AUX FIELD aof-preamble = '0'
[offset 181] Checksum OK
[offset 181] \o/ RDB looks OK! \o/
[info] 0 keys read
[info] 0 expires
[info] 0 already expired
```

異常輸出（文件損壞）：

```

[offset 0] Checking RDB file dump.rdb
[offset 21] AUX FIELD ctime = '1768265496'
[offset 42] AUX FIELD used-mem = '4275241848'
[offset 60] AUX FIELD repl-stream-db = '0'
[offset 11 ] AUX FIELD repl-id =
'f0a6b7b58edlece135f1ba84bcc8add612e8b347'
[offset 135] AUX FIELD repl-offset = '67232446953'
[offset 137] Selecting DB ID 0
--- RDB ERROR DETECTED ---
[offset 73027] Internal error in RDB reading offset 0, function at
rdb.c:1936 -> Intset integrity check failed.
[additional info] While doing: read-object-value
[additional info] Reading key
'jszt.894679007767823360.wf.isallownextparticipant.crrccardispatch.audi
[additional info] Reading type 11 (set-intset)
[info] 159 keys read
[info] 158 expires
[info] 157 already expired
692:C 02 Feb 2026 10:49:23.182 # Terminating server after rdb file
reading failure.

```

9.0.4.2 修復損壞的RDB文件

當RDB文件損壞時，使用--fix選項進行修復，修復後dump文件名為 dump-<進程號>.rdb：

```

#修復損壞的RDB文件
amdc-check-rdb --fix /path/to/dump.rdb

```

修復成功輸出：

```

[offset 0] Checking RDB file --fix
[offset 21] AUX FIELD ctime = '1768265496'
[offset 42] AUX FIELD used-mem = '4275241848'
[offset 60] AUX FIELD repl-stream-db = '0'
[offset 10] AUX FIELD repl-id =
'f0a6b7b58edlece135f1ba84bcc8add612e8b347'

```

```

[offset 135] AUX FIELD repl-offset = '67232446953'
[offset 137] Selecting DB ID 0
--- RDB ERROR DETECTED---
[offset 73027] Internal error in RDB reading offset 0, function at
rdb.c:1936 -> Intset integrity check failed.
[additional info] While doing: read-object-value
[additional info] Reading key
'jszt.894679007767823360.wf.isallownextparticipant.crrccardispatch.audi
[additional info] Reading type 11 (set-intset)
[info] 159 keys read
[info] 158 expires
[info] 157 already expired
--- RDB ERROR DETECTED---
[offset 178543] Internal error in RDB reading offset 0, function at
rdb.c:1936 -> Intset integrity check failed.
[additional info] While doing: read-object-value
[additional info] Reading key
'jszt.894679007767823360.wf.isallownextparticipant.crrclinefeedback.suk
[additional info] Reading type 11 (set-intset)
[info] 392 keys read
[info] 390 expires
[info] 388 already expired
--- RDB ERROR DETECTED---
[offset 260382] Internal error in RDB reading offset 0, function at
rdb.c:1936 -> Intset integrity check failed.
[additional info] While doing: read-object-value
[additional info] Reading key
'jszt.894679007767823360.wf.isallownextparticipant.ebruleexeccase.save'
[additional info] Reading type 11(set-intset)
[info] 571 keys read
[info] 568 expires
[info] 564 already expired
--- RDB ERROR DETECTED---
[offset 136547728] Invalid object type: 111
[additional info] While doing: read-type
[info] 114982 keys read

```

```
[info] 114589 expires
[info] 113699 already expired
[offset 136547728] \o/ RDB partially fixed: dump-790.rdb! \o/
[info] 114982 / 406455 keys fixed
```

9.0.5 注意事項

- 修復文件前是否需要備份原RDB文件：--fix選項不會直接覆蓋原文件，修復後dump文件名為 dump-<進程號>.rdb，安全起見操作前可以備份RDB文件；
- 僅能修復部分損壞的文件：若RDB文件頭部信息損壞、校驗和嚴重錯誤，工具可能無法修復，此時需依賴備份文件；
- 修復後可能丟失部分數據：對於RDB文件中格式異常的鍵值對，工具會嘗試跳過並丟棄該數據；若檢測到無法修復的數據片段，則會停止掃描文件剩餘內容，因此工具可能會僅完成了文件局部數據的修復；
- 工具版本需與AMDC版本匹配：不同版本的RDB文件格式可能有差異，使用與AMDC服務同版本的amdc-check-rdb工具，避免校驗或修復失敗。

9.1 amdc-check-aof工具使用手冊

9.1.1 工具簡介

amdc-check-aof是AOF(Append Only File) 文件校驗與修復工具，用於檢查AOF文件的完整性、語法正確性，修復因意外中斷導致的AOF文件損壞，確保AMDC啟動時能正常加載AOF文件恢復數據。

核心作用：

- 校驗AOF文件的語法正確性、完整性，檢測文件是否存在損壞；
- 修復因服務器宕機、斷電等意外導致的AOF文件截斷、語法錯誤；
- 移除AOF文件中無效的命令，確保AMDC能正常加載。

9.1.2 工具位置與運行前提

9.1.2.1 工具位置

與amdc-check-rdb位於同一目錄，即AMDC的安裝目錄文件夾下：

```
Linux: /amdc/amdc-check-aof
```

9.1.2.2 運行前提

無需啟動AMDC服務，可直接在命令行運行；

- 擁有AOF文件的讀取、寫入權限（修復時需寫入文件）；
- AOF文件路徑正確，若使用相對路徑，需確保終端當前目錄與文件路徑匹配。

9.1.3 基本語法

amdc-check-aof的語法與amdc-check-rdb類似，核心指定AOF文件路徑，可選參數用於修復、調試：

#基礎語法（校驗AOF文件）

```
amdc-check-aof [options]
/path/to/appendonlydir/appendonly.aof.manifest
```

#常用簡寫（無選項時，默認校驗文件完整性）

```
amdc-check-aof appendonlydir/appendonlydir/appendonly.aof.manifest
```

9.1.4 核心選項與參數說明

amdc-check-aof的選項重點關注修復、強制修復和詳細輸出，具體如下：

選項	說明	使用場景
--fix	修復損壞的AOF 文件，移除無效命令、截斷錯誤部分，修復後覆蓋原文件（建議先備份）	AOF 文件損壞，AMDC啟動失敗
--truncate-to-timestamp	精準裁剪到指定時間戳，保留該時間點前所有數據	已知服務異常時間點、需精準保留指定時間前的數據
--help（簡寫 -h）	顯示工具幫助信息，包括所有選項和語法	忘記命令時快速查詢

9.1.5 常見操作示例

9.1.5.1 校驗AOF文件完整性

直接指定AOF文件路徑，工具自動校驗文件語法和完整性：

#校驗AOF文件

```
amdc-check-aof /path/to/appendonlydir/appendonly.aof.manifest
```

正常輸出aof-use-rdb-preamble no（文件完整）：

```

Start checking Multi Part AOF
Start to check BASE AOF (RESP format).
AOF analyzed:
filename=appendonly.aof.1.base.aof,size=127,ok_up_to=127,ok_up_to_line=
BASE AOF appendonly.aof.1.base.aof is valid
Start to check INCR files.
INCR AOF appendonly.aof.1.incr.aof is empty
All AOF files and manifest are valid

```

正常輸出aof-use-rdb-preamble yes (文件完整)：

```

Start checking Multi Part AOF
Start to check BASE AOF (RDB format).
[offset 0] Checking RDB file appendonlydir/appendonly.aof.1.base.rdb
[offset 25] AUX FIELD amdc-ver = '2.0.6'
[offset 38] AUX FIELD amdc-bits = '64'
[offset 50] AUX FIELD ctime = '1770690624'
[offset 65] AUX FIELD used-mem = '836208'
[offset 81] AUX FIELD aof-preamble = '1'
[offset 83] Selecting DB ID 0
[offset 151] Checksum OK
[offset 151] \o/ RDB looks OK! \o/
[info] 2 keys read
[info] 0 expires
[info] 0 already expired
RDB preamble is OK, proceeding with AOF tail...
AOF analyzed: filename=appendonly.aof.1.base.rdb, size=151,
ok_up_to=151, ok_up_to_line=1, diff=0
BASE AOF appendonly.aof.1.base.rdb is valid
Start to check INCR files.
INCR AOF appendonly.aof.1.incr.aof is empty
All AOF files and manifest are valid

```

異常輸出 aof-use-rdb-preamble no (文件損壞)：

```

Start checking Multi Part AOF
Start to check BASE AOF (RESP format).
AOF appendonly.aof.1.base.aof format error
AOF analyzed:
filename=appendonly.aof.1.base.aof,size=194,ok_up_to=127,ok_up_to_line=
AOF appendonly.aof.1.base.aof is not valid. Use the --fix option to
try fixing it.

```

異常輸出 aof-use-rdb-preamble yes (文件損壞)：

```

Start checking Multi Part AOF
Start to check BASE AOF (RDB format).
[offset 0] Checking RDB file appendonlydir/appendonly.aof.1.base.rdb
[offset 25] AUX FIELD amdc-ver = '2.0.6'
[offset 38] AUX FIELD amdc-bits = '64'
[offset 50] AUX FIELD ctime = '1770690624'
[offset 65] AUX FIELD used-mem = '836208'
[offset 81] AUX FIELD aof-preamble = '1'
[offset 83] Selecting DB ID 0
[offset 151] Checksum OK
[offset 151] \o/ RDB looks OK! \o/
[info] 2 keys read
[info] 0 expires
[info] 0 already expired
RDB preamble is OK, proceeding with AOF tail...
AOF appendonly.aof.1.base.rdb format error
AOF analyzed:
filename=appendonly.aof.1.base.rdb,size=167,ok_up_to=151,ok_up_to_line=
AOF appendonly.aof.1.base.rdb is not valid. Use the --fix option to
try fixing it.

```

9.1.5.2 修復損壞的AOF文件

AOF文件損壞時，優先使用--fix選項修復，修復前務必備份原文件：

#1. 備份原AOF文件 (Linux)

```
cp /path/to/appendonlydir /path/to/appendonlydir.bak
```

#2. 常規修復

```
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.aof
```

修復成功輸出 aof-use-rdb-preamble no (文件損壞)：

```
Start checking Old-Style AOF
AOF appendonlydir/appendonly.aof.1.base.aof format error
AOF analyzed:
filename=appendonlydir/appendonly.aof.1.base.aof,size=146,ok_up_to=127,
This will shrink the AOF appendonlydir/appendonly.aof.1.base.aof
from 146 bytes, with 19 bytes, to 127 bytes
Continue? [y/N]: y
Successfully truncated AOF appendonlydir/appendonly.aof.1.base.aof
```

修復成功輸出 aof-use-rdb-preamble yes (文件損壞)：

```
Start checking Old-Style AOF
[offset 0] Checking RDB file appendonlydir/appendonly.aof.1.base.rdb
[offset 25] AUX FIELD amdc-ver = '2.0.6'
[offset 38] AUX FIELD amdc-bits = '64'
[offset 50] AUX FIELD ctime = '1770690624'
[offset 65] AUX FIELD used-mem = '836208'
[offset 81] AUX FIELD aof-preamble = '1'
[offset 83] Selecting DB ID 0
[offset 151] Checksum OK
[offset 151] \o/ RDB looks OK! \o/
[info] 2 keys read
[info] 0 expires
[info] 0 already expired
RDB preamble is OK, proceeding with AOF tail...
AOF appendonlydir/appendonly.aof.1.base.rdb format error
```

```

AOF analyzed:
filename=appendonlydir/appendonly.aof.1.base.rdb,size=167,ok_up_to=151,
This will shrink the AOF appendonlydir/appendonly.aof.1.base.rdb
from 167 bytes, with 16 bytes, to 151 bytes
Continue? [y/N]: y
Successfully truncated AOF appendonlydir/appendonly.aof.1.base.rdb

```

9.1.6 注意事項

- 備份優先：修復AOF文件前，必須備份原文件，避免修復失敗導致數據徹底丟失；
- 修復原理：工具通過截斷無效命令、修正語法錯誤來修復文件，會丟棄無法識別的命令，修復後需檢查數據完整性；
- 與AOF重寫的區別：amdc-check-aof用於修復損壞文件，而AMDC的AOF重寫（BGREWRITEAOF）用於壓縮AOF文件，二者功能不同；
- appendonlydir 目錄下可通過 --fix 參數修復指定文件，支持的文件包括：appendonly.aof.x.base.rdb、appendonly.aof.x.base.aof、appendonly.aof.x.incr.aof 為可選指定。

9.2 amdc-benchmark 工具使用手冊

9.2.1 工具簡介

amdc-benchmark是AMDC自帶的性能測試工具，用於模擬多客戶端並發訪問AMDC服務，測試AMDC在不同場景下的性能指標，包括每秒請求數（QPS）、響應時間（延遲）、並發連接數等，幫助運維人員和開發人員評估AMDC服務的承載能力、優化配置參數，確保AMDC服務在生產環境中穩定運行。

核心作用：

- 測試AMDC服務在不同並發連接數、不同命令類型下的吞吐量（QPS）；
- 評估AMDC服務的響應延遲（最小延遲、最大延遲、平均延遲），排查性能瓶頸；
- 模擬不同數據量、不同命令組合的訪問場景，驗證AMDC配置（如內存、線程數）的合理性；
- 對比AMDC在不同環境（如單機、集群）、不同版本下的性能差異，為部署方案提供依據。

提示：amdc-benchmark屬於壓力測試工具，測試時會佔用AMDC服務的CPU、內存及網絡資源，生產環境需謹慎使用，建議在測試環境或業務低峰期執行，避免影響線上業務。

9.2.2 工具位置與運行前提

9.2.2.1 工具位置

與前三個工具位於同一目錄，即AMDC安裝文件夾下，無需額外安裝，隨AMDC一起部署：

```
Linux: /amdc/amdc-benchmark
```

提示：若將AMDC安裝目錄添加到系統環境變量，可直接在終端輸入amdc-benchmark 運行，無需輸入完整路徑。

9.2.2.2 運行前提

- 必須啟動AMDC服務：amdc-benchmark 需連接AMDC服務才能執行測試，確保amdc-server 已正常運行；
- 網絡通暢：本地測試需確保 127.0.0.1:6359（默認端口）可訪問；遠程測試需確保AMDC服務端口（默認 6359）開放，且配置允許遠程連接；
- 權限充足：運行amdc-benchmark的用戶需擁有終端執行權限，若AMDC服務設置密碼，測試時需指定密碼；
- 資源預留：測試前預留足夠的CPU、內存資源，避免測試環境自身資源不足，影響測試結果準確性。

9.2.3 基本語法

amdc-benchmark的語法核心是指定連接參數和測試參數，通過選項控制測試場景，基礎語法如下：

```
#基礎語法（連接AMDC服務 + 執行測試）
```

```
amdc-benchmark [連接選項] [測試選項]
```

```
#常用簡寫（本地默認連接，無額外選項，執行默認測試）
```

```
amdc-benchmark
```

說明：

- 連接選項：用於指定AMDC服務的主機、端口、密碼等，與amdc-cli的連接選項基本一致；
- 測試選項：用於控制並發數、測試命令、測試時長、數據量等，是amdc-benchmark 的核心選項；
- 默認測試：若不指定任何選項，工具會以10個並發連接、測試16種常用AMDC命令，每種命令執行10000次，最終輸出性能報告。

9.2.4 核心選項與參數說明

amdc-benchmark的選項分為「連接選項」和「測試選項」兩類，其中測試選項是重點，具體如下：

9.2.4.1 連接選項

選項	取值類型	說明	默認值
-h <host>	字符串（IP / 域名）	指定AMDC 服務的主機地址(IP)，本地測試可省略，遠程測試必須指定	127.0.0.1
-p <port>	整數（端口號）	指定AMDC 服務的端口號	6359
-s <socket>	字符串（套接字路徑）	指定AMDC 服務的密碼（若服務設置了 requirepass 配置）	無（默認無密碼）
-a <password>	字符串	指定連接超時時間（秒），避免測試時連接阻塞	無（默認不超時）

9.2.4.2 核心測試選項

選項	取值類型	說明	默認值
-c <clients>	整數	指定並發連接的客戶端數量，模擬多用戶同時訪問	50
-n <requests>	整數	指定每種測試命令的總請求數，所有客戶端共同完成	100000
-d <size>	整數（字節）	指定SET/GET命令的value 數據大小，模擬真實業務的數值長度	3
--dbnum <db>	整數（數據庫編號）	指定測試使用的AMDC 數據庫（SELECT 命令），對應AMDC 的db0~db15	0
-k <boolean>	布爾值（1/0）	連接保活策略：1 = 保持連接（長連接）、0 = 每次請求重連（短連接），模擬真實業務連接模式	1
-r <keyspacelen>	整數	為SET/GET/INCR等命令使用隨機鍵名、SADD用隨機值、ZADD 用隨機成員和分數；自動將參數中的randint 替換為 0 ~ keyspacelen-1 的12位隨機數，避免熱點鍵，貼合真實場景	無
-P <numreq>	整數	開啟管道請求，每次管道打包 < numreq > 個請求發送，模擬業務批量操作（提升吞吐）	無
--help	無	顯示工具幫助信息，包括所有選項和基本語法	忘記命令語法時快速查詢
--version	無	顯示amdc-benchmark 工具版本，與AMDC 服務版本一致	確認版本兼容性

9.2.4.3 高級擴展參數（多線程 / 集群 / 特殊測試模式）

選項	取值類型	說明	默認值
--threads <num>	整數	開啟多線程模式運行壓測工具，利用多核CPU 提升壓測端性能（避免壓測端成為瓶頸）	4
--cluster	無（開關參數）	開啟集群模式，適配AMDC Cluster 集群壓測，自動識別槽位、分發請求	關閉
-l	無（開關參數）	循環測試模式：無限運行測試，直到手動按Ctrl+C停止，用於長時間穩定性壓測	關閉
-I	無（開關參數）	空閒模式：僅打開-c 指定的N 個空閒連接，不發送任何請求，用於測試AMDC 的空閒連接承載能力	關閉

9.2.5 常見操作示例（核心重點）

以下示例覆蓋日常最常用的測試場景，結合實際業務需求，適配AMDC V2.0.3、V2.0.4、V2.0.5和V2.0.6版本，可直接參考使用：

9.2.5.1 本地默認測試

不指定任何選項，執行默認測試：50個並發客戶端、測試16種默認命令，每種命令執行100000次，輸出完整性能報告：

```
amdc-benchmark
```

默認測試命令包括：SET、GET、INCR、LPUSH、LPOP、SADD、SPOP等常用命令，適合快速了解本地AMDC的基礎性能。

9.2.5.2 指定並發數和請求數測試

模擬真實並發場景，指定並發客戶端數量和每種命令的請求數，是最常用的測試方式：

#示例：50個並發客戶端，每種命令執行100000次，測試本地AMDC性能

```
amdc-benchmark -c 50 -n 100000
```

#示例：100個並發客戶端，每種命令執行50000次，測試遠程AMDC（帶密碼）

```
amdc-benchmark -h 192.168.1.100 -p 6359 -a your_password -c 100 -n 50000
```

9.2.5.3 針對性測試核心命令

實際業務中通常只使用少數幾種AMDC命令，通過-t選項指定命令，避免無用測試，提高效率：

#示例：測試SET和GET命令，50個並發，每種命令執行100000次

```
amdc-benchmark -t SET,GET -c 50 -n 100000
```

#示例：測試哈希命令（HSET、HGET），30個並發，值大小為1KB

```
amdc-benchmark -t HSET,HGET -c 30 -n 50000 -d 1024
```

9.2.5.4 按時間進行壓力測試對性測試核心命令

模擬長時間穩定訪問，指定測試持續時間，適合測試Redis長時間運行的穩定性：

#示例：循環測試（手動停止），測試SET命令，60個並發，值大小為512字節

```
amdc-benchmark -t SET -c 60 -l -d 512
```

9.2.5.5 簡化輸出，快速查看QPS

無需詳細延遲信息，僅查看每種命令的QPS，適合快速對比性能差異：

#示例：默認測試，簡化輸出QPS

```
amdc-benchmark -q
```

#示例：測試遠程AMDC的SET、GET命令，簡化輸出

```
amdc-benchmark -h 192.168.1.100 -a your_password -t SET,GET -c 50 -n 100000 -q
```

9.2.5.6 測試結果解讀（關鍵指標）

執行測試後，工具會輸出詳細報告，核心關注以下4個指標，以SET命令為例：

```
SET: 100000 requests completed in 0.89 seconds
50 parallel clients
3 bytes payload
keep alive:1
host configuration "save": 3600 1 300 100 60 10000
host configuration "appendonly": no
multi-thread:no
Latencyby percentile distribution:
```

```

0.000% <= 0.103 ms (cumulative count 123)
50.000% <= 0.215 ms (cumulative count 50123)
90.000% <= 0.327 ms (cumulative count 90123)
99.000% <= 0.439 ms (cumulative count 99012)
99.900% <= 0.551 ms (cumulative count 99901)
99.990% <= 0.663 ms (cumulative count 99990)
99.999% <= 0.775 ms (cumulative count 99999)
100.000% <= 0.887 ms (cumulative count 100000)

Cumulative distribution of latencies:
0.103 ms: 123 requests (0.12%)
0.215 ms: 50123 requests (50.12%)
0.327 ms: 90123 requests (90.12%)
0.439 ms: 99012 requests (99.01%)
0.551 ms: 99901 requests (99.90%)
0.663 ms: 99990 requests (99.99%)
0.775 ms: 99999 requests (99.999%)
0.887 ms: 100000 requests (100.00%)

Summary:
throughput summary: 112359.55 requests per second
latency summary (msec):
| avg    | min    | p50    | p95    | p99    | max    |
|-----|-----|-----|-----|-----|-----|
| 0.220  | 0.080  | 0.215  | 0.383  | 0.439  | 0.887  |

```

核心指標解讀：

- throughput summary: QPS（每秒請求數），核心性能指標，數值越高越好（示例中為112359.55 QPS）；
- avg / min / max：平均延遲、最小延遲、最大延遲（毫秒），數值越低越好；
- p50 / p95 / p99：延遲分位數，代表50%/95%/99%的請求延遲不超過該數值（示例中99%的請求延遲 ≤ 0.439ms），分位數更能反映真實性能；
- requests completed：完成的總請求數，與-n指定的請求數一致，說明測試正常。

9.2.6 注意事項

- 生產環境謹慎使用：amdc-benchmark會佔用AMDC大量CPU、內存和網絡資源，可能導致線上業務卡頓、超時，建議僅在測試環境或業務低峰期執行，且測試後及時停止；
- 測試環境與生產環境一致：測試時盡量保證AMDC配置（內存、線程數、持久化方式）、服務器硬件（CPU、內存）與生產環境一致，否則測試結果無參考價值；
- 避免單一測試場景：真實業務場景複雜，建議結合並發數、數據量、命令類型多場景測試，全面評估性能；
- 排除干擾因素：測試時關閉AMDC持久化(appendonly no、save ""), 避免持久化操作影響測試結果；同時關閉服務器上其他佔用資源的進程；
- 密碼安全：使用-a選項輸入密碼時，密碼會明文顯示在終端歷史記錄中，建議測試完成後清理終端歷史，或通過其他方式隱藏密碼；
- 合理設置測試參數：並發數不宜過高（超出服務器承載能力），請求數/測試時長不宜過短（避免測試結果波動過大），建議多次測試取平均值。

9.3 amdc-conf-conv工具使用手冊

9.3.1 工具簡介

amdc-conf-conv是AMDC專屬的配置文件格式轉換工具，核心用於將AMDC V2.0.2及以下版本AMDC的配置文件轉換為AMDC V2.0.6版本AMDC的配置文件，解決不同版本AMDC配置文件格式不兼容的問題，實現AMDC V2.0.2及以下版本AMDC向AMDC V2.0.6版本AMDC的平滑遷移，無需手動修改配置項，提升版本遷移的效率和準確性。

核心作用：

- 一鍵轉換AMDC V2.0.2及以下版本AMDC的YAML配置文件為AMDC V2.0.6版本AMDC兼容的YAML配置文件，自動適配配置項格式、命名規範；
- 支持服務端（server）、哨兵（sentinel）兩種配置類型的精準轉換，分別匹配對應版本的功能配置；
- 保留原配置文件中的所有有效配置項，確保轉換後AMDC V2.0.6版本AMDC可直接加載使用；
- 規避手動轉換配置的語法錯誤、項缺失問題，降低版本遷移的配置風險。

配置轉換工具執行邏輯為「先讀取--c-yaml模版文件的原有結構與基礎配置->提取--go-yaml指向的AMDC V2.0.2及以下版本配置文件的有效配置項->將模版文件中對應配置項替換為AMDC V2.0.2及以下版本的配置值->最終覆蓋寫入--c-yaml指向的文件」，並非直接將AMDC V2.0.2及以下版本配置生成新的AMDC V2.0.6版本配置文件，--c-yaml指向的配置文件一定不能為空文件。

9.3.2 工具位置與運行前提

9.3.2.1 工具位置

與AMDC其他配套工具（如amdc-check-aof、amdc-check-rdb）位於同一目錄，隨AMDC 安裝包一同部署，無需額外安裝：

```
Linux: /amdc/amdc-conf-conv
```

9.3.2.2 運行前提

- 無需啟動AMDC服務，可直接在命令行運行；
- 擁有AMDC V2.0.2及以下版本配置文件的讀取權限，以及目標AMDC V2.0.6版本配置文件的讀寫權限；
- 確保AMDC V2.0.2及以下/AMDC V2.0.6版本AMDC的配置文件（YAML格式）路徑正確、文件完整無損壞，可正常讀取。

9.3.3 基本語法

amdc-conf-conv 語法規則固定，需指定配置類型和源或目標配置文件路徑，無默認執行邏輯，必須傳入完整有效參數才能運行，基礎語法如下：

```
#基礎語法（指定配置類型 + 源AMDC V2.0.2及以下版配置 + 目標C版配置）
amdc-conf-conv [--server|--sentinel] --go-yaml <amdc_go.yaml> --c-yaml <amdc.yaml>
```

9.3.4 核心選項與參數說明

amdc-check-aof的選項重點關注修復、強制修復和詳細輸出，具體如下：

選項	說明	使用場景
--server	指定轉換的是AMDC 服務端（server）類型的配置文件； [--server	--sentinel]` 二選一必傳參數，不可省略、不可同時使用
--sentinel	指定轉換的是AMDC 哨兵（sentinel）類型的配置文件； [--server	--sentinel]` 二選一必傳參數，不可省略、不可同時使用
--go-yaml	AMDC V2.0.2及以下版本AMDC 配置文件的完整路徑（絕對或相對），指定轉換源文件	所有轉換場景，必傳源文件路徑
--c-yaml	轉換後C 版本AMDC 配置文件的完整路徑（絕對或相對）	所有轉換場景，必傳目標文件路徑
--help	顯示工具幫助信息，包括所有選項和語法	忘記命令時快速查詢

9.3.5 常見操作示例

以下示例覆盖server和sentinel兩種核心配置類型的轉換場景，適配Linux環境，可直接參考使用，建議執行前備份相關配置文件。

9.3.5.1 轉換AMDC V2.0.2及以下版本AMDC服務端配置為AMDC V2.0.6版本

指定--server參數，轉換AMDC V2.0.2及以下版本服務端配置文件conf.yaml為AMDC V2.0.6版本服務端配置文件amdc.yaml:

```
#使用絕對路徑（推薦，無路徑兼容問題）
```

```
amdc-conf-conv --server --go-yaml /amdc/conf.yaml --c-yaml
/amdc/amdc.yaml
```

```
#使用相對路徑（需確保終端當前目錄為/amdc）
```

```
cd /amdc
```

```
amdc-conf-conv --server --go-yaml conf.yaml --c-yaml amdc.yaml
```

轉換成功:

```
Skip unsupported configuration item: ReplSlaveIgnoreMaxMemory
Skip unsupported configuration item: PoolSliceSize
Skip unsupported configuration item: PoolSliceNum
Skip unsupported configuration item: PoolNewWorkerNum
Skip unsupported configuration item: AofNoFsyncOnRewrite
Skip unsupported configuration item: GoroutineMaxCpuNum
Skip unsupported configuration item: SlowLogSlowerThan
Skip unsupported configuration item: LuaMaxLocalVarNum
Skip unsupported configuration item: LuaMaxStackSize
Skip unsupported configuration item: ReplServeStaleData
Skip unsupported configuration item: ReplPingSlavePeriod
Skip unsupported configuration item: Ip
Skip unsupported configuration item: Segment
\o/ Config file converted successfully! (conf.yaml -> amdc.yaml)\o/
```

conf.yaml轉換前:

onf.yaml

Network:**MaxClients: 10000**

连接在空闲超过设置的时间后, 服务器会主动关闭连接, 当设置为0时, 不启用超时机制, timeout单位

Timeout: 0

TCP keepalive, 单位为秒, 当为0时不设置tcp keepalive

TcpKeepAlive: 300**General:**

db数量

Databases: 16

日志等级, 用于过滤输出日志, 包括debug, info, warn, error, fatal五个等级

LogLevel: "info"

日志文件输出目录, 当设置为空字符串时, 日志文件不会写入磁盘, eg: LogFile: "/tmp/server

LogFile: "./server-6359.log"

license文件位置

LicensePath: "./license.lic"**MemoryManagement:**

最大内存限制, 如果maxmemory值为0, 表示不做限制, 如果数字后没有单位, 则默认单位为字节, 单位

eg: "1gb", "1GB", "1000mb", "1000m", "1000000KB", "1000000kb", "1000000000B"

MaxMemory: "6GB"

缓存淘汰策略, noeviction, volatile-lru, allkeys-lru, volatile-random, allkeys-r

MaxMemoryPolicy: "volatile-lru"

每次缓存淘汰时的采样数量

MaxMemorySamples: 5

lfu-log-factor可以调整计数器counter的增长概率, lfu-log-factor越大, counter增长概率越

计算公式为: $1 / (\text{old_value} * \text{lfu_log_factor} + 1)$ **LFULogFactor: 10.000000**

lfu-decay-time是一个以分钟为单位的数值, 可以调整counter的减少速度

LFUDecayTime: 1.000000

从节点是否忽略maxmemory检查

ReplSlaveIgnoreMaxMemory: "yes"

PoolSliceSize

PoolSliceSize: 40

PoolSliceNum

PoolSliceNum: 40

PoolSliceNum

PoolNewWorkerNum: 4**SnapShooting:**

save <seconds> <changes>, 指定在多长时间, 有多少次更新操作后, server会将rdb文件写入

Save:

- "3600 1"

- "300 10"

- "60 10000"

当 Save: "" 为空时, 表示RDB未设置

amdc.yaml轉換後:

mdc.yaml

REPLICATION:

```
# 从节点向主节点公布的端口 (用于 NAT/Docker 等场景)
replica-announce-port: 0
```

KEYS TRACKING:

```
# 客户端缓存跟踪表的最大键数, 0 表示无限制
tracking-table-max-keys: 1000000
```

SECURITY:

```
# user: "<user> <acl rules...>"
user: default on nopass ~* &* +@all
# ACL 日志的最大条目数
acllog-max-len: 128
# 外部 ACL 用户文件路径, 与配置文件内用户配置不可混用
aclfile: ""
# 默认用户的密码 (等价于 ACL 的 default 用户密码)
requirepass: "test123"
# 新用户默认的 Pub/Sub 频道权限, allchannels (允许所有)、resetchannels (禁止所有)
acl-pubsub-default: allchannels
# 命令重命名, eg: rename-command: "flushall ''", rename-command: "flushall no-
rename-command: ""
```

CLIENTS:

```
# 最大并发客户端连接数, 受系统文件描述符限制
maxclients: 10000
```

MEMORY MANAGEMENT:

```
# 内存使用上限, 达到后按 maxmemory-policy 处理
maxmemory: 6gb
# 内存达到上限时的淘汰策略, 如 volatile-lru、allkeys-lfu 等
maxmemory-policy: volatile-lru
# LRU/LFU/TTL 算法的采样数, 越大越精确但 CPU 消耗越高
maxmemory-samples: 5
# 淘汰处理的激进程度, 0 (最低延迟) 到 100 (优先处理淘汰)
maxmemory-eviction-tenacity: 10
# 从节点是否忽略自身的 maxmemory 配置 (默认由主节点处理淘汰)
replica-ignore-maxmemory: yes
# 主动过期键的扫描力度, 1-10, 越大越频繁 (可能增加 CPU 和延迟)
active-expire-effort: 1
```

LAZY FREEING:

```
# 内存淘汰时是否使用惰性删除 (后台线程释放内存)
lazyfree-lazy-eviction: no
# 键过期时是否使用惰性删除
lazyfree-lazy-expire: no
# 服务器内部删除键 (如 RENAME 覆盖) 时是否使用惰性删除
```

9.3.5.2 轉換AMDC V2.0.2及以下版本AMDC哨兵配置為AMDC V2.0.6版本

指定--sentinel參數，轉換AMDC V2.0.2及以下版本哨兵配置文件sentinelgo.yaml為AMDC V2.0.6版本哨兵配置文件sentinel.yaml:

#絕對路徑示例

```
amdc-conf-conv --sentinel --go-yaml /amdc/sentinelgo.yaml --c-yaml  
/amdc/sentinel.yaml
```

#相對路徑示例（終端當前目錄為 / amdc）

```
amdc-conf-conv --sentinel --go-yaml sentinelgo.yaml --c-yaml  
sentinel.yaml
```

修復成功輸出:

```
Config file converted successfully! (sentinelgo.yaml->sentinel.yaml)
```

sentinel_go.yaml轉換前:

Sentinel:

```

# 绑定的ip地址, 可以绑定多个ip地址, 支持ipv4/ipv6地址, eg:
# Bind:
# - "127.0.0.1"
# - "::1"
Bind:
- "0.0.0.0"
# 端口号
port: 26369
# 日志等级, 用于过滤输出日志, 包括debug, info, warn, error, fatal五个等级
loglevel: notice
# 日志文件输出目录, 当设置为空字符串时, 日志文件不会写入磁盘
logfile: ""
# sentinel相关配置项
SentinelItems:
- "sentinel monitor mymaster 127.0.0.1 6369 1" # 设置监听的主节点信息, IP地址/ 端口
- "sentinel down-after-milliseconds mymaster 30000" # 设置主节点主观下线的超时时间
- "sentinel failover-timeout mymaster 180000"
- "sentinel parallel-syncs mymaster 1"
# - "sentinel auth-pass mymaster 123456" # 设置哨兵访问主节点的密码,
# - "sentinel config-epoch mymaster 0" # 设置节点的选举纪元
# - "sentinel leader-epoch mymaster 0"
# - "sentinel known-replica mymaster 127.0.0.1 6380" # 设置其他已知的从节点信息
# - "sentinel current-epoch 0" # 设置当前哨兵选举纪元

```

SSL:

```

# 是否开启ssl, 如果开启则使用true, 否则是false
Enable: false
# SSL监听端口, 如果仅开启SSL监听, 需要把NetWord: Port端口设置为0

# 服务端SSL证书文件
TlsCertFile: "./certs/ssl_tls_cert/server.crt"
# 服务端SSL证书的密钥
TlsKeyFile: "./certs/ssl_tls_cert/server.key"
# 证书签发机构的证书文件, 即可信从根证书
TlsCaCertFile: "./certs/ssl_tls_cert/ca.crt"
# 证书签发机构的证书文件夹, 即多个可信的根证书, 可放在一个文件夹中
TlsCaCertDir: ""
# 客户端SSL证书文件, 为集群/ 主从模式使用
TlsClientCertFile: "./certs/ssl_tls_cert/client.crt"
# 客户端SSL证书的密钥, 为集群/ 主从模式使用
TlsClientKeyFile: "./certs/ssl_tls_cert/client.key"
# 服务器是否验证客户端证书, 设置为yes需要客户端提供证书, 并且服务端作验证。 如果不需要验证则
# 设置为optional, 即客户端可以提供证书也可以不提供证书, 但是如果客户端提供证书, 那么amdc服
TlsAuthClients: "no"
# 主从模式是否使用TLS通信, 当master开启了tls模式, 那么从节点必须开启, 否则无法连接master

```

sentinel.yaml轉換後:

TLS/SSL:

```
# TLSV1.3 允许的密码套件
tls-ciphersuites: ""
# 是否优先使用服务器的密码套件偏好
tls-prefer-server-ciphers: no
# 是否启用 TLS 会话缓存以加速重连
tls-session-caching: yes
# TLS 会话缓存的最大条目数, 0 表示无限制
tls-session-cache-size: 20480
# TLS 会话缓存的超时时间 (秒)
tls-session-cache-timeout: 300
```

SENTINEL:

```
# 定义要监控的主节点, 格式为 <master-name> <ip> <amdc-port> <quorum> (如 mymaster
sentinel monitor: mymaster 127.0.0.1 6369 1
# 主从节点的认证用户名, 格式为 <master-name> <username>
sentinel auth-user: mymaster ""
# 主从节点的认证密码, 格式为 <master-name> <password>
sentinel auth-pass: mymaster ""
# 判定节点主观下线的超时时间 (毫秒), 默认 30000 (30 秒)
sentinel down-after-milliseconds: mymaster 30000
# Sentinel 之间通信的认证用户名
sentinel sentinel-user: ""
# Sentinel 之间通信的认证密码其他设置
sentinel sentinel-pass: ""
# 故障转移时从节点同步新主节点的并发数量, 默认 1
sentinel parallel-syncs: mymaster 1
# 故障转移超时时间 (毫秒), 默认 180000 (3 分钟)
sentinel failover-timeout: mymaster 180000
# 发生警告级事件时执行的通知脚本 (如邮件、短信通知), 格式为 <master-name> <script-path>
sentinel notification-script: mymaster ""
# 故障转移后通知客户端重新配置的脚本, 格式为 <master-name> <script-path>
sentinel client-reconfig-script: mymaster ""
# 是否禁止运行时修改脚本配置, 默认 yes
sentinel deny-scripts-reconfig: yes
# 重命名命令, 格式为 <master-name> <old-command-name> <new-command-name>
sentinel rename-command: mymaster "" ""
# 是否支持主机名 (默认 no, 需依赖 DNS)
sentinel resolve-hostnames: no
# 是否对外公布主机名 (默认 no)
sentinel announce-hostnames: no
# NAT/Docker 环境下对外公布的 IP
sentinel announce-ip: ""
# NAT/Docker 环境下对外公布的端口
sentinel announce-port: 0
```

9.3.6 注意事项

- 模版文件直接覆蓋：工具完成配置項替換後，會直接覆蓋--c-yaml指向的文件，無任何彈窗或命令行提示，轉換前務必對該模版文件進行完整備份，避免原有AMDC V2.0.6版本配置結構或基礎項丟失；
- 參數規則嚴格遵循：需二選一指定--server或--sentinel配置類型，不可省略、不可同時使用；--go-yaml(AMDC V2.0.2及以下版本源配置) 和--c-yaml(AMDC V2.0.6版本替換模版) 為必傳參數，後必須緊跟實際文件路徑，參數缺失、格式錯誤或路徑無效會直接觸發工具報錯並退出；
- 轉換方向與類型唯一：本工具僅支持AMDC V2.0.2及以下版本AMDC配置->AMDC V2.0.6版本AMDC配置的單向替換，不支持AMDC V2.0.6版本向AMDC V2.0.2及以下版本的反向轉換，也不兼容其他版本AMDC的配置替換；且-server或--sentinel 需與源文件、模版文件的配置類型一致（服務端配置僅能替換服務端模版，哨兵配置僅能替換哨兵模版）；
- 替換後必須驗證有效性：替換完成（無報錯輸出）後，需先檢查--c-yaml文件是否正常覆蓋、配置項是否已完成替換，再啟動對應類型的AMDC V2.0.6版本AMDC（服務端或哨兵）加載該文件，確認服務能正常啟動無配置相關報錯，避免因配置項匹配異常導致服務啟動失敗。

9.4 常見問題排查

9.4.1 工具無法運行

問題1：終端輸入amdc-check-rdb提示“command not found” (Linux)；

- 原因：工具路徑未添加到系統環境變量；
- 解決：輸入完整路徑運行（如/usr/local/amdc/amdc-check-rdb），或添加環境變量（echo "export PATH=/usr/local/amdc/:\$PATH" >> /etc/profile，source /etc/profile）。

9.4.2 校驗 / 修復文件失敗

問題1:amdc-check-rdb或amdc-check-aof 提示“Permission denied”；

- 原因：無文件讀取/寫入權限；
- 解決：使用chmod修改文件權限(chmod 644 dump.rdb)，或使用sudo運行工具(sudo amdc-check-rdb --fix dump.rdb)。

問題2：修復後AMDC仍無法啟動；

- 原因：文件損壞過於嚴重，或工具版本與AMDC版本不匹配；
- 解決：使用備份文件恢復，或升級或降級工具版本，與AMDC服務版本一致。

9.4.3 amdc-cli 無法連接AMDC服務

問題1：連接提示“Could not connect to AMDC at 127.0.0.1:6359: Connection refused”;

- 原因：AMDC服務未啟動；
- 解決：啟動AMDC服務（`amdc-server /path/to/amdc.yaml`, Linux）。

問題2：遠程連接提示 “Could not connect to AMDC at 192.168.1.100:6359:Connection timed out”;

- 原因：防火牆未開放6359端口，或AMDC配置不允許遠程連接；
- 解決：開放防火牆端口(`firewall-cmd --add-port=6359/tcp --permanent`, Linux)，修改AMDC配置(`bind 0.0.0.0`, `protected-mode no`)，重啟AMDC服務。

問題3：連接後執行命令提示“NOAUTH Authentication required”;

- 原因：AMDC服務設置了密碼，連接時未輸入；
- 解決：連接時用-a選項輸入密碼，或交互模式輸入AUTH密碼。

9.5 總結

本文檔詳細介紹了AMDC四個核心工具的使用方法，重點總結如下：

- `amdc-check-rdb`: 專注於RDB文件的校驗與修復，核心選項`--fix`，修復前務必備份；
- `amdc-check-aof`: 專注於AOF文件的校驗與修復，支持常規修復(`--fix`)，風險需謹慎；
- `amdc-benchmark`: AMDC自帶的性能測試工具，用於測試AMDC服務的吞吐量、響應時間等指標，核心是通過選項控制測試場景，適配不同性能評估需求。
- `amdc-conf-conv`: `amdc-conf-conv`是AMDC專屬的配置文件格式轉換工具，用於將AMDC V2.0.2及以下版本AMDC的配置文件轉換為AMDC V2.0.6版本AMDC的配置文件。

以上四個工具均為AMDC自帶，無需額外安裝，日常運維中需熟練掌握，尤其是`amdc-cli`和`amdc-benchmark`，可高效完成AMDC的各類操作、故障排查及性能評估。使用過程中，務必注意數據安全和測試規範，避免誤操作導致數據丟失或服務異常。

10 常見問題解決

10.1 問題一：端口佔用與解決

安裝AMDC會佔用一些端口，同時客戶端鏈接需要依賴內部的DNS服務，如果不能保證端口未被使用，或者是安裝機器環境缺乏有效DNS，則會導致啟動失敗。產品默認佔用的端口：

- AMDC緩存核心默認端口：6359
- AMDC哨兵服務默認端口：26359

如果啟動之前上述端口被佔用，將導致啟動失敗。可以切換端口或停止正在使用該端口的程序。

10.2 AMDC緩存核心端口修改

修改配置文件:vim amdc.yaml, port字段

更多配置請參考：[緩存配置 \(amdc.yaml\)](#)

10.3 ACL文件加載

將ACL文件存放於AMDC緩存配置項“ACLFile”指定的目錄之下，並在AMDC緩存命令行中執行 ACL load 即可重新加載ACL文件中的用戶配置。

注：

1. 重新加載ACL文件將會替換緩存中原有的ACL規則。
2. 若ACL文件中有一項或多項是不合法的數據，整個文件都將無法被加載，緩存系統中現有的規則將繼續使用。

10.4 解決AMDC主從故障切換

1. 緩存核心集群中若slave故障，系統將自動重連 若Slave出現故障導致宕機，恢復正常後會自動重新連接，Master收到Slave的連接後，將其完整的數據文件發送給Slave，如果Master同時收到多個Slave發來的同步請求，Master只會在後台啟動一個進程保存數據文件，然後將其發送給所有的Slave，確保Slave正常。
2. 緩存核心集群中若master出現故障，手動重啟主服務即可實現客戶端對系統的讀寫操作 當緩存核心master出現故障時，仍支持客戶端對從服務的讀操作，需要手動重啟主服務，待主服務重啟系統恢復正常後，再支持客戶端對主服務的讀寫操作。

10.5 請求延遲問題

若AMDC請求出現明顯延遲，可以通過以下辦法查明具體問題：

1. slowlog命令：該命令可以查詢出執行時間較長的命令（時間長度可以通過amdc.yaml文件中的slowlog參數進行配置），從而更好地跟蹤代碼執行過程。
2. -bigkeys：使用amdc-cli客戶端，連接並加上 -bigkeys參數，來查詢value最大的值，從而判斷是否因為值過大導致請求延時。
3. -memkeys：同上，該參數查詢key-value整體的內存佔用大小，從而判斷是否因為key-value過大導致請求延時。

10.6 如何檢測執行了那些命令

使用amdc-cli進行連接，然後使用monitor命令即可監控AMDC持續接受到了哪些命令，也可使用以下命令後台記錄到monitor.log中：

```
amdc-cli -h [ip] -p [port] [-a password] monitor >monitor.log 2>&1
&
```

想要停止記錄monitor.log時使用以下命令：

```
ps -ef | grep amdc-cli | grep -v grep | awk '{print $2}' | xargs
kill -9
```

10.7 進程還在但無法連接

在客戶端超過Linux系統限制或者使用內存超過license限制內存時，有可能會出現進程在，但是連接超時/無法連接的情況。

1. 客戶端超過Linux系統限制：某些Linux系統會默認客戶端連接數上限為1024，當業務應用過多，產生超過1024的客戶端連接時，超過部分會被Linux系統限制無法繼續連接。

解決辦法：永久設置用戶級別的文件描述符限制，在/etc/security/limits.conf中添加：

```
* soft nofile 65535
* hard nofile 65535
```

注：不同的Linux系統可能修改的方式有差異，如果以上無法生效，請自行搜索對應系統的TCP連接數。

2. 使用內存超過license限制內存：license的默認內存限制一般為6G（大部分都是），當業務系統寫入超過6G的數據並持續寫入時，會造成AMDC頻繁觸發內存淘汰算法，造成AMDC的性能下降，在持續大量的並發情況下會出現某些請求超時，從而導致業務段報錯。

解決辦法：擴容即可解決對應問題，擴容有兩種方法，一為申請更大內存的license文件，直接擴大節點可使用的內存（不建議超過16G，過大會導致查詢效率變低）；二為做成分佈式集群模式（對應Redis的Cluster模式）。

10.8 dir 路徑問題

配置文件中的配置項 "dir" 配置的路徑，會在程序讀取完配置文件後，將程序的當前執行文件路徑轉為dir配置的地址，並非amdc-server該可執行文件的文件路徑。在此請注意：處理方式一：證書路徑、日誌文件路徑等，統一用絕對路徑填寫，穩固可靠。處理方式二：如若依舊需要使用相對路徑，請對配置中的路徑檢查，請基於配置項 "dir" 配置的路徑來修改對應證書、日誌等等的路徑。

10.9 密碼設置問題

在使用amdc主從模式的時候，我們知道需要配置requirepass為節點增設密碼。

這裡注意，還要對從節點的 masterauth 配置項填入主節點的 requirepass，以便從節點和主節點正常交互。否則如果主節點設置了 requirepass 而從節點未設置 masterauth，或密碼錯誤，複製連接會立即被主節點拒絕，在日誌中會出現 (error) NOAUTH Authentication required. 或 (error) ERR invalid password 錯誤。

--建議無論集群服務還是主從服務，無論主節點、從節點，都將requirepass、masterauth 配置完整，防止主從切換後，舊主的認證失效問題。

同理，哨兵監控的同時需要配置節點的密碼。

哨兵需要持續監控所有主從節點，並在主節點故障時執行自動故障轉移。因此，哨兵也必須能夠“進入”這些受密碼保護的節點。這是通過哨兵自身的配置文件中的 sentinel auth-pass 指令實現的：sentinel auth-pass

10.10 日誌配置問題

系統可以在機器上進行日誌配置管理amdc的server.log，結合自身業務對默認寫入server.log的日誌做簡單的處理。能讓日誌更加個性化易用。

10.11 主從緩衝區配置問題

repl-backlog-size 是主從緩衝區的大小，默認配置為1mb，無法對大量數據交互的主從服務提供緩衝功能，建議往大調至128mb。

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

