Apusic 金蝶天燕

生成时间：2026-07-13

Migration Guide

金蝶Apusic分布式缓存 V2.0.4

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 Preface

This document is the migration guide for Apusic In-Memory Data Cache (AMDC) V2.0.4, detailing how to migrate Redis to AMDC, as well as migrating older AMDC versions to V2.0.4.

1.1 Target Audience

This document is intended for AMDC product operations engineers, IT system operations engineers, development engineers, software architects, and R&D managers.

1.2 Related Documentation

For more information about AMDC V2.0.4 product, please refer to the following AMDC V2.0.4 product manual documentation set:

No.	Manual Document	Description
1	Apusic In-Memory Data Cache V2.0.4 Quick Start Guide	Brief introduction on how to quickly get started with AMDC.
2	Apusic In-Memory Data Cache V2.0.4 Installation Guide	Detailed introduction on how to install AMDC on various operating systems, AMDC service start/stop operations, and product registration process.
3	Apusic In-Memory Data Cache V2.0.4 Cache Core User Manual	Detailed introduction on the usage, configuration, management of AMDC related functions and supporting tools.
4	Apusic In-Memory Data Cache V2.0.4 Console User Manual	Detailed introduction on the usage and operation instructions of AMDC console related functions.
5	Apusic In-Memory Data Cache V2.0.4 Development Guide	Detailed introduction on AMDC client application development based on various programming languages.
6	Apusic In-Memory Data Cache V2.0.4 Migration Guide	Detailed introduction on migrating AMDC historical versions to V2.0.4, and migrating Redis to AMDC.
7	Apusic In-Memory Data Cache V2.0.4 Operations Guide	Detailed introduction on AMDC monitoring, operations, security hardening and other operational instructions.
8	Apusic In-Memory Data Cache V2.0.4 Performance Optimization Guide	Detailed introduction on AMDC performance tuning.

1.3 Technical Support

AMDC product provides comprehensive technical support services. You can obtain technical support through the following channels:

- Website: www.apusic.com
- Phone: 400-855-5800
- Email: support@apusic.com
- Kingdee Cloud Community: https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

When requesting technical support, please provide the following information:

1. Your name
2. Company information and contact details
3. Operating system and version
4. Product version number
5. Detailed information about exceptions and errors, including logs and screenshots

2 Overview

This manual introduces two migration scenarios to AMDC V2.0.4:

1. Migrating AMDC V2.0.2 and below to AMDC V2.0.4: Use amdc-conf-conv tool to convert AMDC V2.0.2 and below configuration files to AMDC V2.0.4 compatible format, reuse original configuration and data files, then verify functionality after startup.
2. Migrating Redis 6.2 and below to AMDC V2.0.4: Core relies on AMDC's ability to directly load Redis configuration (redis.conf, etc.) and data files (RDB/AOF, etc.).

3 AMDC V2.0.2 and Below Migration to V2.0.4 Instructions

3.1 Preparation

3.1.1 Environment Check

- AMDC V2.0.4 version installed (must match tool version), with proper installation directory permissions (readable and executable)
- AMDC V2.0.2 and below version service is accessible, or RDB cache files and configuration files (yaml format) have been obtained
- Target migration server (deployed AMDC V2.0.4 server) has sufficient disk space to store migrated cache data and configuration files

3.1.2 Tools and Files Preparation

3.1.2.1 Essential Tools

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below RDB cache files (deployed with AMDC V2.0.4 installation package, located in /amdc/ directory)
- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AOF files (deployed in same directory)
- amdc-conf-conv: Used to convert AMDC V2.0.2 and below configuration files to AMDC V2.0.4 compatible format (deployed in same directory)

3.1.2.2 Migration Files Collection

- AMDC V2.0.2 and below configuration files: Server configuration (e.g., conf.yaml) or sentinel configuration (e.g., sentinel_go.yaml)
- AMDC V2.0.2 and below RDB snapshot files: Default name dump.rdb, path usually in installation directory (confirm actual path)
- If AMDC V2.0.2 and below has AOF persistence enabled, prioritize AOF files as migration data source, with RDB files as backup migration data source

3.2 Standalone Mode Migration

Standalone mode migration is suitable for scenarios with only a single AMDC service deployed. Core process is "File Preparation -> Configuration Conversion -> Data Verification & Repair -> Deployment & Startup -> Verification". Simple operation with no dependencies. Specific steps are as follows:

3.2.1 Migration Prerequisites

- Target server has AMDC V2.0.4 installed, with proper installation directory permissions (readable and executable)

- AMDC V2.0.2 and below configuration files (e.g., conf.yaml) and RDB snapshot files (dump.rdb) have been collected. If AOF persistence is enabled, prioritize obtaining AOF files
- Target server has sufficient disk space, at least 2 times the source data file size (reserve space for backup and operation)

3.2.2 Preparation

3.2.2.1 Tools and Configuration Setup

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below RDB cache files (deployed with AMDC V2.0.4 installation package, located in /amdc/ directory)
- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AOF files (deployed in same directory)
- amdc-conf-conv: Used to convert AMDC V2.0.2 and below configuration files to AMDC V2.0.4 compatible format (deployed in same directory)

3.2.2.2 Backup Data and Stop AMDC V2.0.2 and Below Single Node (Full Shutdown)

- Stop write operations to AMDC V2.0.2 and below single node, execute bgsave command again to generate latest RDB file. Backup source AMDC configuration
- Must stop AMDC V2.0.2 and below single node first, prohibit any data writes, ensure source data file final consistency

```
# Execute on node: Stop AMDC V2.0.2 and below AMDC process
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify process stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

3.2.3 Migration Steps

3.2.3.1 Configuration File Migration

Use amdc-conf-conv tool to convert AMDC V2.0.2 and below server configuration to AMDC V2.0.4 compatible format:

1. Backup AMDC V2.0.4 default configuration file on target server (avoid overwrite loss):

```
cp /amdc/amdc.yaml /amdc/amdc.yaml.bak
```

2. Execute configuration conversion (specify --server parameter, adapt to server configuration):

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

3. Verify conversion result: Open /amdc/amdc.yaml, confirm core configuration items (e.g., port, password, data directory, memory limit, etc.) are correctly inherited, no syntax errors

3.2.3.2 Cache Data Migration

1. Verify and repair source RDB or AOF files (prioritize AOF files, more complete data):

```
# If using AOF file
amdc-check-aof /path/to/appendonlydir/appendonly.aof.manifest

# If using RDB file
amdc-check-rdb /path/to/dump.rdb

# If file corrupted, execute repair
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.rdb
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.aof
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.incr.aof

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

2. Copy repaired files to AMDC V2.0.4 installation directory:

```
# Check AMDC V2.0.4 data directory (confirm via dir configuration
item in amdc.yaml, default path below), copy RDB file
cp /path/to/dump.rdb /amdc/

# If using AOF file, must enable AOF configuration first (set
appendonly yes in amdc.yaml), then copy
cp /path/to/appendonlydir /amdc/
```

3.2.3.3 Start AMDC V2.0.4 AMDC Service

1. Start AMDC V2.0.4 AMDC service, load converted configuration file:

```
amdc-server /amdc/amdc.yaml
```

2. Verify service startup status:

```
# Check service process
ps -ef | grep amdc-server

# Check log (default log path /amdc/server.log)
tail -100f /amdc/server.log
```

3.2.3.4 Verify Startup

1. Connect to AMDC V2.0.4 AMDC service, verify data integrity:

```
# Connect using amdc-cli (default port 6359, add -a option if
password configured)
amdc-cli -p 6359 -a test123

# View keyspace statistics
info keyspace
```

2. Verify functionality: Execute write, modify, delete operations, confirm service responds normally without configuration or data related errors

3.2.4 Notes

- Must stop AMDC V2.0.2 and below service during migration to avoid data writes causing source file corruption
- Observe service running for 10-20 minutes after migration, confirm no log errors or other anomalies

3.3 Master-Slave Mode Migration

Master-slave mode migration is suitable for scenarios where AMDC V2.0.2 and below cannot establish master-slave relationship with AMDC V2.0.4 across versions. Core principle is full shutdown -> migrate master node first -> then

migrate slave nodes -> data consistency verification. Migration is completed through RDB or AOF full data import, requiring brief downtime throughout (no cross-version sync). Specific steps are as follows:

3.3.1 Migration Prerequisites

- Both master and slave target servers have AMDC V2.0.4 installed, with proper installation directory permissions (readable, writable, executable), and master-slave nodes are network connected (no firewall blocking port 6359)
- AMDC V2.0.2 and below master and slave node configuration files (conf.yaml), AOF files (priority, more complete data) or RDB snapshot files have been collected, with full backup made
- Notify business team of maintenance window in advance, recommend operating during low business period. Downtime duration is approximately "data file copy + service startup" time (adjust based on data volume, generally 5-30 minutes)
- Target servers have sufficient disk space, at least 2 times source data file size (reserve space for backup and operation)

3.3.2 Preparation

3.3.2.1 Tools and Configuration Setup

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below RDB cache files (deployed with AMDC V2.0.4 installation package, located in /amdc/ directory)
- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AOF files (deployed in same directory)
- amdc-conf-conv: Used to convert AMDC V2.0.2 and below configuration files to AMDC V2.0.4 compatible format (deployed in same directory)

3.3.2.2 Backup Data and Stop AMDC V2.0.2 and Below Master-Slave Cluster (Full Shutdown)

- Stop write operations to AMDC V2.0.2 and below single node, execute bgsave command again to generate latest RDB file. Backup source AMDC configuration
- Must stop all AMDC V2.0.2 and below master and slave nodes first, prohibit any data writes, ensure source data file final consistency

```
# Execute on master node + all slave nodes: Stop AMDC V2.0.2 and
below AMDC process
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify process stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

3.3.3 Migration Steps (Execute in order "Master Node -> All Slave Nodes")

3.3.3.1 Master Node Migration

Master node is the data core. Must complete configuration conversion, data verification/import and service startup first, serving as sync source for subsequent slave nodes.

Step 1: Configuration File Conversion and Optimization

1. Backup AMDC V2.0.4 master node default configuration (avoid overwrite):

```
cp /amdc/amdc.yaml /amdc/amdc-master.yaml.bak
```

2. Convert AMDC V2.0.2 and below master node configuration to AMDC V2.0.4 master node specific configuration (named amdc-master.yaml for distinction):

```
amdc-conf-conv --server --go-yaml /path/to/conf.yaml --c-yaml
/amdc/amdc-master.yaml
```

3. Check AMDC V2.0.4 master node configuration, enable master node identity (no replicaof configuration), confirm core configuration items:

```
vi /amdc/amdc-master.yaml
```

Key configuration confirmation/modification:

- port: 6359 (unified port, if modification needed, sync across all nodes)
- requirepass: your_password (if source master has password, must keep, slave nodes need to configure masterauth later)
- dir: /amdc/ (data directory, consistent with actual plan)
- appendonly: yes (recommend enabling AOF, consistent with source. If source only uses RDB, set to no)
- logfile: amdc-master.log (master node independent log, easy to troubleshoot)

Step 2: Data File Verification, Repair and Import

1. Verify and repair AMDC V2.0.2 and below master node data files (execute on master node, skip repair if files are intact):

```
# If using AOF file (recommended)
amdc-check-aof /path/to/appendonlydir/appendonly.aof.manifest

# If using RDB file
amdc-check-rdb /path/to/dump.rdb

# If file corrupted, execute repair
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.rdb
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.aof
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.incr.aof

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

2. Copy repaired clean data files to AMDC V2.0.4 master node data directory, and modify permissions:

```
# Copy AOF file (recommended, if using RDB execute next line)
cp /path/to/appendonlydir/ /amdc/

# Copy RDB file
cp /path/to/dump.rdb /amdc/

# Set permissions (avoid service unable to read)
chmod 644 /amdc/{appendonlydir/, dump.rdb}
```

Step 3: Start AMDC V2.0.4 Master Node Service

```
# Start master node, specify specific configuration file
amdc-server /amdc/amdc-master.yaml

# Verify master node startup status
```

```
ps -ef | grep amdc-server | grep master # Check process
tail -100f /amdc/amdc-master.log # Check log
```

Log shows "Ready to accept connections" means master node started successfully;

Connect to master node to verify data integrity (core, ensure data import successful):

```
amdc-cli -p 6359 -a password # Connect to master node

info keyspace # View keyspace statistics (confirm data volume)
```

3.3.3.2 Slave Node Migration (All slave nodes execute synchronously)

Slave nodes don't need to import data separately. After startup, AMDC V2.0.4 internal master-slave sync will pull full data from started AMDC V2.0.4 master node, ensuring complete data consistency with master node.

Step 1: Configuration File Conversion and Master-Slave Association

1. Backup AMDC V2.0.4 slave node default configuration:

```
cp /amdc/amdc.yaml /amdc/amdc-slave.yaml.bak
```

2. Convert AMDC V2.0.2 and below slave node configuration to AMDC V2.0.4 slave node specific configuration (named amdc-slave.yaml):

```
amdc-conf-conv --server --go-yaml /path/to/conf.yaml --c-yaml
/amdc/amdc-slave.yaml
```

3. Check AMDC V2.0.4 slave node configuration, specify master node info, enable slave node identity:

```
vi /amdc/amdc-slave.yaml
```

Key configuration confirmation/modification:

- port: 6360 (unified port, if modification needed, sync across all nodes)
- requirepass: your_password (if source master has password, must keep, slave nodes need to configure masterauth)
- dir: /amdc/ (data directory, consistent with actual plan)

- appendonly: yes (recommend enabling AOF, consistent with source. If source only uses RDB, set to no)
- replicaof: master_node_IP master_node_Port # Points to AMDC V2.0.4 master node IP and port
- logfile: amdc-slave.log (slave node independent log, easy to troubleshoot)

Step 2: Start AMDC V2.0.4 Slave Node Service

No need to copy any source data files to slave node data directory. AMDC V2.0.4 slave node will automatically sync from master node:

```
# Start slave node, specify specific configuration file
amdc-server /amdc/amdc-slave.yaml

# Verify slave node startup status
ps -ef | grep amdc-server # Check process
tail -100f /amdc/amdc-slave.log # Check log
```

Log shows "MASTER <-> REPLICA sync: Finished with success" means slave node successfully connected to master node and started syncing.

Step 3: Verify Slave Node Sync Status

Connect to slave node, confirm sync complete and data consistent with master node:

```
amdc-cli -p 6360 -a password # Connect to slave node

info replication # View sync status
```

Key status confirmation:

- role: slave: Slave node identity normal
- master_host: Master node IP, master_port: 6359: Master node pointing correct
- master_link_status: up: Master-slave connection normal

3.3.4 Notes

- Must fully stop AMDC V2.0.2 and below master-slave nodes throughout migration, prohibit data writes, otherwise source data files will be inconsistent and data will be lost after import
- Must verify data integrity after master node data import, then start slave nodes, avoid slave nodes syncing wrong data

- Slave nodes don't need to copy any source data files, completed by AMDC V2.0.4 internal master-slave sync. Manual copying will cause sync conflicts
- Master and slave node core configurations must be consistent (AOF switch, port, password, etc.), otherwise connection will fail or sync will be abnormal
- Observe cluster running for 10-20 minutes after migration, confirm master-slave sync has no delay, logs have no errors, business access is normal

3.4 Sentinel Mode Migration

Sentinel mode migration is suitable for scenarios where AMDC V2.0.2 and below cannot perform master-slave sync with AMDC V2.0.4 across versions, and sentinel cannot monitor cross-version nodes. Core principle is full shutdown -> migrate master-slave nodes first (same as Chapter 6) -> then migrate sentinel nodes -> re-associate monitoring. Migration is completed through "AMDC V2.0.4 master-slave cluster rebuild + sentinel re-monitoring", requiring brief business downtime throughout. Specific steps are as follows:

3.4.1 Migration Prerequisites

- All master-slave nodes and sentinel nodes have AMDC V2.0.4 installed, network connected (open port 6359 for service, port 26369 for sentinel)
- AMDC V2.0.2 and below master-slave node configuration, data files, and sentinel configuration files (sentinel_go.yaml) have been collected, with full backup made
- Plan maintenance window in advance (low period). Downtime duration is "AMDC V2.0.2 and below shutdown + AMDC V2.0.4 master-slave startup + sentinel startup" time (generally 10-40 minutes)
- All target nodes have synchronized time, sufficient disk space (master node data directory reserves 2x source data space)

3.4.2 Preparation

3.4.2.1 Tools and Configuration Setup

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below RDB cache files (deployed with AMDC V2.0.4 installation package, located in /amdc/ directory)
- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AOF files (deployed in same directory)
- amdc-conf-conv: Used to convert AMDC V2.0.2 and below configuration files to AMDC V2.0.4 compatible format (deployed in same directory)

3.4.2.2 Backup Data and Fully Stop AMDC V2.0.2 and Below Sentinel + Master-Slave Cluster

- Stop write operations to AMDC V2.0.2 and below single node, execute bgsave command again to generate latest RDB file. Backup source AMDC configuration
- Must stop all AMDC V2.0.2 and below sentinel nodes and master-slave nodes first, prohibit data writes and sentinel monitoring, ensure source data consistency

```
# Execute on all nodes: Stop AMDC V2.0.2 and below AMDC master-slave
+ sentinel process
ps -ef | grep -E 'amdc-server | amdc-sentinel' | grep -v grep | awk
'{print $2}' | xargs kill -9

# Verify process stopped (no output means success)
ps -ef | grep -E 'amdc-server | amdc-sentinel' | grep -v grep
```

3.4.3 Migration Steps (Execute in order "AMDC V2.0.4 Master-Slave Cluster -> AMDC V2.0.4 Sentinel Cluster")

3.4.3.1 Rebuild AMDC V2.0.4 Master-Slave Cluster (Core, same as previous chapter complete steps)

- Strictly follow all steps in previous chapter master-slave mode migration to complete AMDC V2.0.4 master-slave cluster setup:
- Migrate AMDC V2.0.4 master node: Configuration conversion -> Data verification import -> Startup -> Verify data integrity
- Migrate all AMDC V2.0.4 slave nodes: Configuration conversion (associate master node) -> Startup -> Verify master-slave sync
- Complete master-slave cluster overall verification: Data consistency, master-slave functionality, no log anomalies

This step is the foundation for sentinel migration. Must ensure AMDC V2.0.4 master-slave cluster is fully operational before proceeding with subsequent sentinel migration.

3.4.3.2 Sentinel Node Migration (All sentinel nodes execute synchronously)

Sentinel nodes don't need to import data. Only need to convert AMDC V2.0.2 and below sentinel configuration to AMDC V2.0.4 compatible format, and modify monitoring target to already-built AMDC V2.0.4 master node.

Step 1: Configuration File Conversion and Monitoring Target Modification

1. Backup AMDC V2.0.4 sentinel default configuration:

```
cp /amdc/sentinel.yaml /amdc/sentinel-c.yaml.bak
```

2. Convert AMDC V2.0.2 and below sentinel configuration to AMDC V2.0.4 sentinel specific configuration (named sentinel-c.yaml):

```
amdc-conf-conv --sentinel --go-yaml /path/to/sentinel_go.yaml --c-
yaml /amdc/sentinel.yaml
```

3. Check AMDC V2.0.4 sentinel configuration, confirm sentinel key configuration:

```
vi /amdc/sentinel.yaml
```

Key configuration modification or confirmation (core is monitor configuration item, pointing to AMDC V2.0.4 master node):

```
# Sentinel core monitoring configuration (replace with AMDC V2.0.4
master node info, other parameters consistent with source)

SENTINEL:

sentinel monitor: mymaster 192.168.1.100 6359 2 # mymaster:
monitoring name (consistent with source), followed by C master node
IP, port, quorum threshold

sentinel down-after-milliseconds: mymaster 30000 # Subjective down
timeout (consistent with source)

sentinel failover-timeout: mymaster 180000 # Failover timeout
(consistent with source)

sentinel auth-pass: mymaster password # Consistent with C master
node requirepass, omit if no password
```

Other configuration confirmation:

- port: 26359 (sentinel default port, consistent with source)
- logfile: sentinel.log (sentinel independent log)
- dir: /amdc/ (sentinel data directory, stores monitoring state)

```
# All sentinel nodes start synchronously, specify specific
configuration file
amdc-sentinel /amdc/sentinel.yaml

# Verify sentinel startup status
ps -ef | grep amdc-sentinel # Check process
tail -100f /amdc/sentinel.log # Check log
```

Step 2: Start AMDC V2.0.4 Sentinel Service

Log shows "Sentinel ID is xxxxxxxxxxxxxxxxxxxxxxx" means sentinel started successfully.

Key verification: Sentinel has successfully discovered all nodes in AMDC V2.0.4 master-slave cluster

```
# Connect to sentinel port (26369), query monitoring status

amdc-cli -p 26359

info sentinel # View sentinel monitoring info

sentinel slaves mymaster # View monitored slave node list
```

Verification requirements:

- Slave node list shows IP and port are all AMDC V2.0.4 slave node info, status is online

3.4.4 Notes

- Must fully stop AMDC V2.0.2 and below sentinel + master-slave nodes before migration, avoid sentinel continuously monitoring stopped nodes, or data writes causing source file inconsistency
- Must complete AMDC V2.0.4 master-slave cluster build and verification before starting sentinel nodes, otherwise sentinel won't be able to monitor valid nodes, causing migration failure
- Monitoring name (e.g., mymaster), quorum threshold, timeout in sentinel configuration should be consistent with source, reduce client configuration changes
- All sentinel nodes' monitoring configurations must be completely identical (monitoring target, password, threshold, etc.), otherwise sentinel cluster will split, failover will be abnormal
- Observe for 10-20 minutes after migration, focus on monitoring: No sentinel log errors, no master-slave switch anomalies, sync delay within acceptable range, business access uninterrupted

3.5 Cluster Mode Migration

Cluster mode migration is suitable for scenarios where AMDC V2.0.2 and below cannot establish cluster with AMDC V2.0.4 across versions, cannot sync shard data across versions. Core principle is full shutdown -> full shard data export -> AMDC V2.0.4 cluster initialization -> batch shard data import -> cluster verification. Since cluster is multi-master multi-slave shard architecture, no cross-version sync possible, migration is completed through "full data export-import", requiring business downtime throughout. Specific steps are as follows:

3.5.1 Migration Prerequisites

- All target cluster nodes (same count as source cluster: M masters N slaves) have AMDC V2.0.4 installed, network connected (open port 6359 for service, port 16359 for cluster gossip communication)
- AMDC V2.0.2 and below cluster all nodes' configuration files, AOF, RDB data files have been collected, and source cluster topology info recorded (execute cluster nodes to get slot assignment, master-slave relationship)
- Plan long maintenance window in advance (adjust based on data volume, recommend low period). Downtime duration is "data export + cluster initialization + data import + verification" time (approximately 30-60 minutes for data under 10G)
- All target nodes have sufficient disk space (each node reserves 2x source node data space), synchronized time, firewall allows ports 6359, 16359

3.5.2 Preparation

3.5.2.1 Tools and Configuration Setup

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below RDB cache files (deployed with AMDC V2.0.4 installation package, located in /amdc/ directory)
- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AOF files (deployed in same directory)
- amdc-conf-conv: Used to convert AMDC V2.0.2 and below configuration files to AMDC V2.0.4 compatible format (deployed in same directory)

3.5.2.2 Unified Environment and Record Source Topology

1. Create unified directory on all target nodes and set permissions:

2. Observe AMDC V2.0.2 and below cluster core topology:

Connect to any AMDC V2.0.2 and below cluster node, execute following command, save output (AMDC V2.0.4 cluster initialization will fully reuse):

```
amdc-cli cluster nodes # Record: master-slave node IP and port, node ID, slot assignment (e.g., 0-5460)
```

Or view node.conf file, which records cluster node IP, port, node ID, slot assignment info:

```
29755a26cc60c056b3af39eb66dc5ccbfbdd20c1 127.0.0.1:6383@16383 slave
882b57434ef8c361fcea44c327379880181afla6 0 1769505692837 2 connected

882b57434ef8c361fcea44c327379880181afla6 127.0.0.1:6380@16380 master
- 0 1769505692432 2 connected 5461-10922

980c1df981902c2df617ca81d4df7219b1e86a8b 127.0.0.1:6384@16384
myself,slave 2793d6ad527eee582aa30890d7c8407e9c112b4f003 connected

43d25d48fac16f9b8f6b519f88a889179b8a6e87 127.0.0.1:6382@16382 slave
ac9f0c10e742b4570f6436db8af281c1d51ea570 0 1769505692331 1 connected

2793d6ad527eee582aa30890d7c8407e9c112b4f 127.0.0.1:6381@16381 master
- 0 1769505692332 3 connected 10923-16383

ac9f0c10e742b4570f6436db8af281c1d51ea570 127.0.0.1:6379@16379 master
- 0 1769505692228 1 connected 0-5460
```

Record each AMDC V2.0.2 and below master node IP and port number, backup corresponding machine's AOF or RDB files for subsequent data migration to AMDC V2.0.4 cluster

3.5.2.3 Backup Data and Fully Stop AMDC V2.0.2 and Below Cluster (Prohibit Any Writes)

- Stop write operations to AMDC V2.0.2 and below single node, execute bgsave command again to generate latest RDB file. Backup source AMDC configuration

```
# All AMDC V2.0.2 and below cluster nodes execute: Stop process
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify process stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

3.5.2.4 Data File Pre-processing (All source nodes execute)

Verify + repair each AMDC V2.0.2 and below master node's AOF or RDB file, ensure clean data files, avoid data corruption after importing to AMDC V2.0.4:

```
# Prioritize AOF file (process each node separately, cluster is
shard storage, each node data is independent)
amdc-check-aof /path/to/nodeX/appendonlydir/appendonly.aof.manifest

# If no AOF, use RDB file
amdc-check-rdb /path/to/nodeX/dump.rdb

# If file corrupted, execute repair
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.rdb
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.aof
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.incr.aof

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

3.5.3 Migration Steps (Execute in order "Configuration Conversion -> AMDC V2.0.4 Cluster Initialization -> Shard Data Import -> Cluster State Verification")

3.5.3.1 Initialize AMDC V2.0.4 Empty Cluster

Step 1: All AMDC V2.0.4 Node Configuration Conversion and Cluster Enablement

1. Configuration conversion:

```
amdc-conf-conv --server --go-yaml /path/to/nodeX/conf.yaml --c-yaml
/amdc/amdc.yaml
```

2. Check configuration file:

```
vi /amdc/amdc.yaml
```

Key configuration:

- port: 6359 (sentinel default port, consistent with source)
- logfile: amdc-cluster-XXX.log (node independent log)
- dir: /amdc/ (unified data directory)
- requirepass: password (consistent with source, all nodes same)
- cluster-enabled: yes (enable cluster mode)
- cluster-config-file: nodes-XXX.conf (each node's separate cluster configuration file)
- cluster-node-timeout: 15000 (cluster node timeout, consistent with source)
- cluster-require-full-coverage: yes (enable full slot coverage check)

3. After all node configurations complete, start all AMDC V2.0.4 nodes (only start service, cluster not formed yet):

```
amdc-server /amdc/amdc-cluster-XXX.yaml
```

3.5.3.1.1 Verify all nodes started successfully (log shows "Ready to accept connections")

Step 2: Execute Cluster Initialization (Slot + Master-Slave Assignment)

Select any AMDC V2.0.4 master node, execute amdc-cli --cluster create command. Strictly follow source topology to specify node IP, port and slot assignment. Example with 3 master 3 slave (source topology slots 0-5460, 5461-10922, 10923-16383):

```
# Command format: amdc-cli --cluster create master1 master2 master3
slave1 slave2 slave3 --cluster-replicas 1
# --cluster-replicas 1: means 1 master 1 slave, consistent with
source

amdc-cli --cluster create 192.168.1.10:6359 192.168.1.11:6359
192.168.1.12:6359 192.168.1.13:6359 192.168.1.14:6359
192.168.1.15:6359 --cluster-replicas 1
```

After successful execution, verify cluster initialization status:

```
amdc-cli -p 6359 -a password cluster nodes
```

Requirements: All slots properly assigned to cluster nodes, all node status is myself, master or slave, no fail status.

3.5.3.2 Shard Data Import

First start AMDC V2.0.4 standalone node to load RDB or AOF file, then use AMDC V2.0.4 amdc-cli --cluster import command to automatically re-map and import data from standalone node to cluster. This is cluster standardized import method, no need to manually operate data files, avoiding file permission, format incompatibility issues.

```
# Core command format (execute in one line, newline \ is for
formatting)

amdc-cli --cluster import cluster_master_IP:cluster_master_Port \
--cluster-from standalone_IP:standalone_Port \
--cluster-copy \
--cluster-replace \
-a cluster_password
```

Parameter details:

- --cluster-copy: Required parameter. When copying data, preserve source standalone node original data, avoid source data deletion
- --cluster-replace: Optional parameter. If cluster has same-named key as source node, directly overwrite (recommend adding during migration to ensure data consistent with source, choose based on business needs)

Since cluster is shard storage, each AMDC V2.0.2 and below source shard only stores corresponding slot data. Need to execute import command for each shard in order "source shard master node -> AMDC V2.0.4 corresponding shard master node". Execute on any AMDC V2.0.4 cluster operation node throughout. No need to operate on target master node locally.

Example: 3 master 3 slave cluster (source master1 -> C master1, source master2 -> C master2, source master3 -> C master3)

```
# Import shard 1: source master1(192.168.1.1:6359) -> AMDC V2.0.4
master1(192.168.1.10:6359)

amdc-cli --cluster import 192.168.1.10:6359 \
```

```
--cluster-from 192.168.1.1:6359 \
--cluster-copy \
--cluster-replace \
-a cluster_password

# Import shard 2: source master2(192.168.1.2:6359) -> AMDC V2.0.4
master2(192.168.1.11:6359)

amdc-cli --cluster import 192.168.1.11:6359 \
--cluster-from 192.168.1.2:6359 \
--cluster-copy \
--cluster-replace \
-a cluster_password

# Import shard 3: source master3(192.168.1.3:6359) -> AMDC V2.0.4
master3(192.168.1.12:6359)

amdc-cli --cluster import 192.168.1.13:6359 \
--cluster-from 192.168.1.3:6359 \
--cluster-copy \
--cluster-replace \
-a cluster_password
```

3.5.3.3 Cluster State Repair and Data Verification

Due to manual data import, need to execute cluster state repair to ensure slots match data, no slot loss or data disorder:

```
# Verify slot coverage (must show all 16384 slots assigned)
amdc-cli -p 6359 -a password cluster info
cluster info output shows cluster_slots_assigned:16384 means all
slots assigned, no omissions.
```

3.5.4 Notes

- Cluster migration must be full shutdown, and all source cluster nodes prohibit writes. Since cluster is shard storage, single node data write will cause shard data inconsistency

- Cluster gossip communication port 16359 (6359+10000) must be open, otherwise nodes cannot discover cluster topology, causing cluster split
- When executing amdc-cli --cluster import command, --cluster-copy is required parameter. Omitting this parameter will directly delete AMDC V2.0.2 and below source node original data, causing irreversible source data loss. --cluster-replace recommended during migration to ensure overwrite of same-named test keys in AMDC V2.0.4 cluster, completely consistent with source data
- If source cluster has data skew (certain shard has very large data volume), must evaluate AMDC V2.0.4 cluster corresponding node's disk and memory resources before import. After migration, promptly expand skewed shards to avoid cluster performance bottleneck affecting business access speed
- After migration completion, recommend observing cluster running for 10-20 minutes, focus on monitoring: No data loss in each shard, master node failover auto-triggers, cross-shard operations execute normally, master-slave sync has no persistent delay, business access latency meets expectations

4 Redis 6.2 and Below Migration to AMDC V2.0.4 Instructions

4.1 Preparation

4.1.1 Environment Check

- Target server has AMDC V2.0.4 installed, with proper installation directory permissions (readable and executable)
- Redis RDB snapshot files (dump.rdb) have been collected. If AOF persistence is enabled, prioritize obtaining AOF files
- Target server has sufficient disk space, at least 2 times source data file size (reserve space for backup and operation)

4.1.2 Tools and Files Preparation

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below AMDC's RDB cache files (deployed with AMDC V2.0.4 AMDC installation package, located in /amdc/ directory)
- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AMDC's AOF files (deployed in same directory)

4.2 Standalone Mode Migration

Standalone mode migration is suitable for scenarios with only a single AMDC service deployed. Core process is "File Preparation -> Data Verification & Repair -> Deployment & Startup -> Verification". Specific steps are as follows:

4.2.1 Migration Prerequisites

- Target server has AMDC V2.0.4 installed, with proper installation directory permissions (readable and executable)
- Redis RDB snapshot files (dump.rdb) have been collected. If AOF persistence is enabled, prioritize obtaining AOF files
- Target server has sufficient disk space, at least 2 times source data file size (reserve space for backup and operation)

4.2.2 Preparation

4.2.2.1 Data and Configuration Backup

- Check AMDC V2.0.4 installation directory and data storage directory permissions (ensure AMDC user can read/write)

- Copy RDB file and Redis configuration file to target AMDC specified backup directory (e.g., /amdc 644), ensure AMDC can read

4.2.2.2 Data Backup and Stop Redis Single Node

- Stop business write operations to Redis, execute bgsave command again to generate latest RDB file. Backup source Redis configuration (redis.conf)
- Must stop Redis single node first, prohibit any data writes, ensure source data file final consistency

```
# Execute on node: Stop Redis process
ps -ef | grep redis-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify process stopped (no output means success)
ps -ef | grep redis-server | grep -v grep
```

4.2.3 Migration Steps

4.2.3.1 Cache Data Migration

1. Verify and repair source RDB or AOF files (prioritize AOF files, more complete data):

```
# If using AOF file
amdc-check-aof /path/to/appendonly.aof

# If using RDB file
amdc-check-rdb /path/to/dump.rdb

# If file corrupted, execute repair
amdc-check-aof --fix /path/to/appendonly.aof

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

2. Copy repaired files to AMDC V2.0.4 installation directory:

```
# Check AMDC V2.0.4 data directory (confirm via dir configuration
item in amdc.yaml, default path below), copy RDB file
cp /path/to/dump.rdb /amdc/
```

```
# If using AOF file, must enable AOF configuration first (set
appendonly yes in amdc.yaml), then copy
cp /path/to/appendonly.aof /amdc/
```

4.2.3.2 Modify redis.conf Configuration

1. Modify Redis service configuration file (redis.conf), append following configuration items to end of configuration file:

```
license /path/to/license.xml
apusic-acls /path/to/acls.properties
worker-threads 1
```

4.2.3.3 Start AMDC Service

1. Start AMDC V2.0.4 AMDC service, directly load Redis configuration file:

```
amdc-server /amdc/redis.conf
```

2. Verify service startup status:

```
# Check service process
ps -ef | grep redis-server

# Check log (default log path /amdc/server.log)
tail -100f /amdc/server.log
```

4.2.3.4 Verify Startup

1. Connect to AMDC V2.0.4 AMDC service, verify data integrity:

```
# Connect using amdc-cli (Redis default port 6379, add -a option if
password configured)
amdc-cli -p 6379 -a test123
```

```
# View keyspace statistics
info keyspace
```

2. Verify functionality: Execute write, modify, delete operations, confirm service responds normally without configuration or data related errors

4.2.4 Notes

- Must stop Redis service during migration to avoid data writes causing source file corruption
- Observe service running for 10-20 minutes after migration, confirm no log errors or other anomalies

4.3 Master-Slave Mode Migration

Master-slave mode migration is suitable for scenarios deploying one master multiple slaves AMDC service. Core process is "File Preparation -> Data Verification & Repair -> Deployment & Startup -> Verification". Specific steps are as follows:

4.3.1 Migration Prerequisites

- Both master and slave target servers have AMDC V2.0.4 installed, with proper installation directory permissions (readable, writable, executable), and master-slave nodes are network connected (no firewall blocking port 6359)
- Redis master and slave node configuration files (redis.conf), AOF files (priority, more complete data) or RDB snapshot files have been collected, with full backup made
- Notify business team of maintenance window in advance, recommend operating during low business period. Downtime duration is approximately "data file copy + service startup" time (adjust based on data volume, generally 5-30 minutes)
- Target servers have sufficient disk space, at least 2 times source data file size (reserve space for backup and operation)

4.3.2 Preparation

4.3.2.1 Tools and Files Preparation

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below AMDC's RDB cache files (deployed with AMDC V2.0.4 AMDC installation package, located in /amdc/ directory)
- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AMDC's AOF files (deployed in same directory)

4.3.2.2 Data Backup and Stop Redis Master-Slave Cluster (Full Shutdown)

- Stop business write operations to Redis, execute bgsave command again to generate latest RDB file. Backup source Redis configuration (redis.conf)

- Must stop all Redis master and slave nodes first, prohibit any data writes, ensure source data file final consistency

```
# Execute on master node + all slave nodes: Stop Redis process
ps -ef | grep redis-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify process stopped (no output means success)
ps -ef | grep redis-server | grep -v grep
```

4.3.3 Migration Steps (Execute in order "Master Node -> All Slave Nodes")

4.3.3.1 Cache Data Migration

1. Verify and repair source Redis master node data files (execute on master node, skip repair if files are intact):

```
# If using AOF file (recommended)
amdc-check-aof /path/to/appendonly.aof

# If using RDB file
amdc-check-rdb /path/to/dump.rdb

# If file corrupted, execute repair
amdc-check-aof --fix /path/to/appendonly.aof

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

2. Copy repaired clean data files to AMDC V2.0.4 master node data directory, and modify permissions:

```
# Copy AOF file (recommended, if using RDB execute next line)
cp /path/to/appendonly.aof /amdc/

# Copy RDB file
cp /path/to/dump.rdb /amdc/
```

```
# Set permissions (avoid service unable to read)
chmod 644 /amdc/{appendonly.aof, dump.rdb}
```

4.3.3.2 Configuration File Migration (redis.conf)

1. Copy Redis master/slave node's redis.conf to AMDC master/slave node root directory.
2. Modify Redis service configuration file (redis.conf), append following configuration items to end of configuration file:

```
license /path/to/license.xml
apusic-acls /path/to/acls.properties
worker-threads 1
```

4.3.3.3 Start AMDC Master-Slave Node Service

1. Start AMDC master node first: Switch to master node AMDC installation directory, execute startup command, confirm master node service started successfully

```
# Start slave node, specify specific configuration file
amdc-server /amdc/redis.conf

# Verify slave node startup status
ps -ef | grep amdc-server # Check process
tail -100f /amdc/server.log # Check log
```

2. Then start all AMDC slave nodes: Start slave node AMDC service one by one, check each slave node's startup log, confirm no errors

```
# Start slave node, specify specific configuration file
amdc-server /amdc/redis.conf

# Verify slave node startup status
ps -ef | grep amdc-server # Check process
tail -100f /amdc/server.log # Check log
```

4.3.3.4 Master-Slave Relationship Verification

- Execute command on AMDC master node: info replication command, view slave node list, confirm all slave nodes are normally connected to master node (master_link_status is up)
- Execute command on each AMDC slave node: info replication command, confirm sync with master node is normal (offset consistent)
- Execute data read/write test, master node set key, slave node get key, confirm data sync is normal

4.3.4 Notes

- Must fully stop Redis master-slave nodes throughout migration, prohibit data writes, otherwise source data files will be inconsistent and data will be lost after import
- Must verify data integrity after master node data import, then start slave nodes, avoid slave nodes syncing wrong data
- Slave nodes don't need to copy any source data files, completed by AMDC V2.0.4 internal master-slave sync. Manual copying will cause sync conflicts
- Master and slave node core configurations must be consistent (AOF switch, port, password, etc.), otherwise connection will fail or sync will be abnormal
- Observe cluster running for 10-20 minutes after migration, confirm master-slave sync has no delay, logs have no errors, business access is normal

4.4 Sentinel Mode Migration

AMDC V2.0.4 supports loading Redis configuration redis.conf and sentinel.conf configuration files. Core process is full shutdown -> migrate master-slave nodes first -> then migrate sentinel nodes -> re-associate monitoring. Migration is completed through "master-slave cluster rebuild + sentinel re-monitoring", requiring brief business downtime throughout. Specific steps are as follows:

4.4.1 Migration Prerequisites

- All master-slave nodes and sentinel nodes have AMDC V2.0.4 installed, network connected (open port 6379 for service, port 26379 for sentinel)
- Redis master-slave node configuration (redis.conf), data files, sentinel configuration files (sentinel.conf) have been collected, with full backup made
- Plan maintenance window in advance (low period). Downtime duration is "Redis shutdown + AMDC V2.0.4 master-slave startup + sentinel startup" time (generally 10-40 minutes)
- All target nodes have synchronized time, sufficient disk space (master node data directory reserves 2x source data space)

4.4.2 Preparation

4.4.2.1 Tools and Files Preparation

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below AMDC's RDB cache files (deployed with AMDC V2.0.4 AMDC installation package, located in /amdc/ directory)

- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AMDC's AOF files (deployed in same directory)

4.4.2.2 Data Backup and Fully Stop Redis Sentinel + Master-Slave Cluster

- Stop business write operations to Redis, execute bgsave command again to generate latest RDB file. Backup source Redis configuration (redis.conf)
- Must stop all Redis sentinel nodes and master-slave nodes first, prohibit data writes and sentinel monitoring, ensure source data consistency

```
# All nodes execute: Stop Redis master-slave + sentinel process
ps -ef | grep -E 'redis-server | redis-sentinel' | grep -v grep |
awk '{print $2}' | xargs kill -9

# Verify process stopped (no output means success)
ps -ef | grep -E 'redis-server | redis-sentinel' | grep -v grep
```

4.4.3 Migration Steps

4.4.3.1 Cache Data Migration

1. Verify and repair source Redis master node data files (execute on master node, skip repair if files are intact):

```
# If using AOF file (recommended)
amdc-check-aof /path/to/appendonly.aof

# If using RDB file
amdc-check-rdb /path/to/dump.rdb

# If file corrupted, execute repair
amdc-check-aof --fix /path/to/appendonly.aof

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

2. Copy repaired clean data files to AMDC V2.0.4 master node data directory, and modify permissions:

```
# Copy AOF file (recommended, if using RDB execute next line)
cp /path/to/appendonly.aof /amdc/
```

```
# Copy RDB file
cp /path/to/dump.rdb /amdc/

# Set permissions (avoid service unable to read)
chmod 644 /amdc/{appendonly.aof, dump.rdb}
```

4.4.3.2 Configuration File Migration (redis.conf)

1. Copy Redis master/slave node's redis.conf to AMDC master/slave node root directory.
2. Modify Redis service configuration file (redis.conf), append following configuration items to end of configuration file:

```
license /path/to/license.xml
apusic-acls /path/to/acls.properties
worker-threads 1
```

3. Sentinel configuration file (sentinel.conf) doesn't need modification, can be copied to AMDC sentinel node root directory.

4.4.3.3 Start AMDC Master-Slave Node Service

1. Start AMDC master node first: Switch to master node AMDC installation directory, execute startup command, confirm master node service started successfully

```
# Start slave node, specify specific configuration file
amdc-server /amdc/redis.conf

# Verify slave node startup status
ps -ef | grep amdc-server # Check process
tail -100f /amdc/server.log # Check log
```

2. Then start all AMDC slave nodes: Start slave node AMDC service one by one, check each slave node's startup log, confirm no errors

```
# Start slave node, specify specific configuration file
amdc-server /amdc/redis.conf
```

```
# Verify slave node startup status
ps -ef | grep amdc-server # Check process
tail -100f /amdc/server.log # Check log
```

4.4.3.4 Start AMDC Sentinel Node Service

1. Then start all AMDC sentinel nodes, check sentinel service process and log status, ensure successful startup

```
# Start slave node, specify specific configuration file
amdc-sentinel /amdc/sentinel.conf

# Verify slave node startup status
ps -ef | grep amdc-sentinel # Check process
tail -100f /amdc/sentinel.log # Check log
```

4.4.3.5 Sentinel Functionality Verification

- Execute command on any AMDC sentinel node: info sentinel command, view sentinel monitored master-slave node count and status, confirm sentinel can normally identify AMDC master-slave nodes
- Test if business can automatically obtain new master node address through sentinel, read/write business is normal; after testing, restore original AMDC master node service

4.4.4 Notes

- Must fully stop Redis sentinel + master-slave nodes before migration, avoid sentinel continuously monitoring stopped nodes, or data writes causing source file inconsistency
- After copying sentinel.conf, if source and target AMDC master node IP are different, must modify master node IP in sentinel monitor parameter, otherwise sentinel cannot monitor target master node
- Must start master-slave nodes first, then start sentinel nodes, ensure sentinel can normally discover master-slave nodes at startup, avoid monitoring failure
- Observe for 10-20 minutes after migration, focus on monitoring: No sentinel log errors, no master-slave switch anomalies, sync delay within acceptable range, business access uninterrupted

4.5 Cluster Mode Migration

AMDC V2.0.4 supports loading Redis configuration redis.conf and node.conf configuration files. Core process is "File Preparation -> Data Verification & Repair -> Deployment & Startup -> Verification". Specific steps are as follows:

4.5.1 Migration Prerequisites

- All target cluster nodes (same count as source cluster: M masters N slaves) have AMDC V2.0.4 installed, network connected
- Redis cluster all nodes' configuration files, AOF, RDB data files have been collected, and source cluster topology info recorded (execute cluster nodes to get slot assignment, master-slave relationship)
- Plan long maintenance window in advance (adjust based on data volume, recommend low period). Downtime duration is "data export + cluster initialization + data import + verification" time (approximately 30-60 minutes for data under 10G)
- All target nodes have sufficient disk space (each node reserves 2x source node data space)

4.5.2 Preparation

4.5.2.1 Tools and Files Preparation

- amdc-check-rdb: Used to verify and repair AMDC V2.0.2 and below AMDC's RDB cache files (deployed with AMDC V2.0.4 AMDC installation package, located in /amdc/ directory)
- amdc-check-aof: Used to verify and repair AMDC V2.0.2 and below AMDC's AOF files (deployed in same directory)

4.5.2.2 Data Backup and Fully Stop Redis Cluster (Prohibit Any Writes)

- Stop business write operations to Redis, execute bgsave command again to generate latest RDB file. Backup source Redis configuration (redis.conf)

```
# All Redis cluster nodes execute: Stop process
ps -ef | grep redis-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify process stopped (no output means success)
ps -ef | grep redis-server | grep -v grep
```

4.5.3 Migration Steps

4.5.3.1 Data File Pre-processing (All source nodes execute)

Verify + repair each Redis master node's AOF or RDB file, ensure clean data files, avoid data corruption after importing to AMDC:

```
# Prioritize AOF file (process each node separately, cluster is
shard storage, each node data is independent)
amdc-check-aof /path/to/nodeX/appendonly.aof
```

```
# If no AOF, use RDB file
amdc-check-rdb /path/to/nodeX/dump.rdb

# If file corrupted, execute repair
amdc-check-aof --fix /path/to/appendonly.aof

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

4.5.3.2 Configuration File Migration (redis.conf, node.conf)

1. Copy each Redis cluster node's redis.conf to AMDC corresponding cluster node root directory.
2. Modify Redis service configuration file (redis.conf), append following configuration items to end of configuration file:

```
license /path/to/license.xml
apusic-acls /path/to/acls.properties
worker-threads 1
```

3. Copy each Redis cluster node's node.conf to AMDC corresponding cluster node root directory.

4.5.3.3 Start AMDC Cluster Nodes

- Start all AMDC cluster nodes one by one, execute startup command, check each node's startup log, confirm no errors (focus on cluster connection and slot loading status)

```
# Start slave node, specify specific configuration file
amdc-server /amdc/redis.conf

# Verify slave node startup status
ps -ef | grep amdc-server # Check process
tail -100f /amdc/server.log # Check log
```

4.5.3.4 Cluster State Verification

- Execute command on any AMDC node: cluster info, confirm cluster status is ok, all slots assigned completely, no unassigned slots

- Execute command: cluster slots, view slot assignment consistent with source, each node's responsible slots correct
- Execute data read/write test, cross-shard query test, confirm data complete, business normal

4.5.4 Notes

- Cluster migration must be full shutdown, and all source cluster nodes prohibit writes. Since cluster is shard storage, single node data write will cause shard data inconsistency
- If source cluster has data skew (certain shard has very large data volume), must evaluate AMDC V2.0.4 cluster corresponding node's disk and memory resources before import. After migration, promptly expand skewed shards to avoid cluster performance bottleneck affecting business access speed
- After migration completion, recommend observing cluster running for 10-20 minutes, focus on monitoring: No data loss in each shard, master node failover auto-triggers, cross-shard operations execute normally, master-slave sync has no persistent delay, business access latency meets expectations

5 Common Issues Troubleshooting

5.0.1 Tool Cannot Run

Issue 1: Terminal shows "command not found" when entering amdc-check-rdb (Linux)

- Cause: Tool path not added to system environment variable
- Solution: Enter full path to run (e.g., /usr/local/amdc/amdc-check-rdb), or add environment variable (echo "export PATH=/usr/local/amdc/:$PATH" >> /etc/profile, source /etc/profile)

5.0.2 Verification or Repair File Failed

Issue 1: amdc-check-rdb or amdc-check-aof shows "Permission denied"

- Cause: No file read or write permission
- Solution: Use chmod to modify file permissions (chmod 644 dump.rdb), or use sudo to run tool (sudo amdc-check-rdb --fix dump.rdb)

Issue 2: AMDC still cannot start after repair

- Cause: File corruption too severe, or tool version doesn't match AMDC version
- Solution: Use backup file to restore, or upgrade or downgrade tool version to match AMDC service version

5.0.3 amdc-cli Cannot Connect to AMDC Service

Issue 1: Connection shows "Could not connect to AMDC at 127.0.0.1:6359: Connection refused"

- Cause: AMDC service not started
- Solution: Start AMDC service (amdc-server /path/to/amdc.yaml, Linux)

Issue 2: Remote connection shows "Could not connect to AMDC at 192.168.1.100:6359: Connection timed out"

- Cause: Firewall hasn't opened port 6359, or AMDC configuration doesn't allow remote connection
- Solution: Open firewall port (firewall-cmd --add-port=6359/tcp --permanent, Linux), modify AMDC configuration (bind 0.0.0.0, protected-mode no), restart AMDC service

Issue 3: Command execution after connection shows "NOAUTH Authentication required"

- Cause: AMDC service has password set, not entered during connection
- Solution: Enter password using -a option during connection, or enter AUTH password in interactive mode

6 Summary

This document detailed the migration process from AMDC V2.0.2 and below and Redis 6.2 and below to AMDC V2.0.4. Key summary:

6.1 AMDC V2.0.2 and Below to AMDC V2.0.4 Migration Process

- Complete environment check, tools and files preparation before migration, ensure sufficient permissions and complete files
- Configuration file migration is achieved through amdc-conf-conv tool. Must specify configuration type, source or target file path. Must backup target configuration file before conversion
- Cache data migration is based on RDB files. Must first verify file integrity through amdc-check-rdb, repair corrupted files before deploying to AMDC V2.0.4 installation directory
- After migration, verify service startup status, data integrity and functionality availability, ensure migration successful
- If encountering tool operation, file processing, service startup issues during migration, can refer to common issues troubleshooting chapter for quick resolution

6.2 Redis 6.2 and Below to AMDC V2.0.4 Migration Process

- Install AMDC, extract installation package to target directory, complete basic installation
- Stop source Redis service, collect corresponding mode core files (configuration files + data files) and backup
- Use Redis configuration file to start AMDC nodes in corresponding mode order, ensure no errors
- Verify data and functionality normal, switch business traffic, keep source for rollback

Following this document's operation process, smooth migration from AMDC V2.0.2 and below and Redis 6.2 and below to AMDC V2.0.4 can be achieved, ensuring cache service continuity and data security.

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com