



APUSIC
固若长城
睿比世界

Performance Tuning

Kingdee Apusic Distributed Message Queue
V2.0.6_for_mqtt

Copyright Notice

All intellectual property rights including software copyrights involved in this document have been legally registered and duly owned by Kingdee Apusic Cloud Computing Co., Ltd. Such rights are protected by Copyright Law of the People's Republic of China, Regulations on Protection of Computer Software, Regulations on Intellectual Property Protection, relevant international copyright conventions, laws, regulations and other intellectual property legislations. Unauthorized use is strictly prohibited.

Disclaimer

All copyright information contained herein is legally owned by Kingdee Apusic Cloud Computing Co., Ltd. and protected by applicable laws. The Company shall bear no liability for any information unrelated to Kingdee Apusic contained in this document. You may view, copy and print this document only within the scope permitted by Copyright Law of the People's Republic of China. No entity or individual may use, modify or republish any part or content of this document without written authorization from Kingdee Apusic Cloud Computing Co., Ltd. Any violation shall constitute infringement, and the Company reserves the right to pursue legal liabilities accordingly.

Updates to this document shall not be notified separately. You may consult the Company regarding issues related to this document. All rights not expressly granted by the Company are reserved.

Trademark Notice

 is a registered trademark filed with the National Intellectual Property Administration of PRC by Kingdee Apusic Cloud Computing Co., Ltd. Shenzhen. Exclusive rights to the registered trademark are legally owned by Kingdee Apusic and protected by law. Without written consent of Kingdee Apusic, no entity or individual shall use, copy, modify, disseminate, transcribe, or bundle and sell any part of this trademark with other products in any manner or for any reason. Kingdee Apusic shall pursue legal liabilities against any infringement of its trademark rights. All other trademarks or registered trademarks mentioned in this document belong to their respective owners.

Contents

- 1 Preface
 - 1.1 Target Audience
 - 1.2 Related Documents
 - 1.3 Technical Support
- 2 System-Level Optimization
 - 2.1 Operating System Optimization
 - 2.1.1 File Descriptors
 - 2.1.2 Kernel Parameters
 - 2.1.3 Disk Optimization
- 3 MQTT Configuration Optimization
 - 3.1 Connection Management
 - 3.2 Session Configuration
 - 3.3 Packet Size
 - 3.4 JVM Configuration
 - 3.5 Erlang VM Optimization
- 4 Client Optimization
 - 4.1 QoS Selection
 - 4.2 Connection Management
 - 4.3 Message Size
- 5 Cluster Optimization
 - 5.1 Node Deployment Strategy
 - 5.2 Load Balancing
 - 5.3 Session Persistence
- 6 Monitoring and Tuning
 - 6.1 Performance Metrics
 - 6.2 Performance Testing
 - 6.3 Bottleneck Analysis
- 7 Common Issue Optimization
 - 7.1 Insufficient Connection Count
 - 7.2 Message Latency Optimization
 - 7.3 Insufficient Memory Optimization
 - 7.4 Message Backlog Optimization

• 8 Performance Tuning Checklist

- 8.1 System Level
- 8.2 MQTT Configuration
- 8.3 Client Optimization
- 8.4 Cluster Optimization

1 Preface

This document describes the performance tuning best practices for the Kingdee Apusic Distributed Message Queue for MQTT (Apusic Distributed Message Queue, ADMQ for MQTT), helping users fully utilize system performance.

1.1 Target Audience

This document is intended for IT business leaders, R&D managers, software project managers, software architects, and operations engineers.

1.2 Related Documents

To learn more about ADMQ for MQTT, please refer to the following ADMQ for MQTT product documentation set:

No.	Document	Description
1	ADMQ for MQTT Quick Start Guide	Briefly introduces how to quickly get started with ADMQ for MQTT.
2	ADMQ for MQTT Installation Manual	Details how to install ADMQ for MQTT on various operating systems and how to start/stop ADMQ for MQTT services.
3	ADMQ for MQTT Message Engine User Manual	Details the usage, configuration, management, and supporting tools of ADMQ for MQTT message engine features.
4	ADMQ for MQTT Console User Manual	Details the usage and operation instructions of the ADMQ for MQTT console.
5	ADMQ for MQTT Development Manual	Details ADMQ for MQTT client application development based on various programming languages.
6	ADMQ for MQTT Migration Manual	Details the migration from MQTT Broker to ADMQ for MQTT.
7	ADMQ for MQTT Operations Manual	Details ADMQ for MQTT monitoring, operations, security hardening, and other operational tasks.
8	ADMQ for MQTT Performance Tuning Manual	Details ADMQ for MQTT performance tuning guidelines.

1.3 Technical Support

ADMQ for MQTT 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 contacting technical support, please provide the following information:

1. Your name
2. Company information and contact details
3. Operating system and its version
4. Product version number
5. Detailed information such as logs and screenshots of the exception or error

2 System-Level Optimization

2.1 Operating System Optimization

2.1.1 File Descriptors

```
# View current limits
ulimit -n

# Modify limits (/etc/security/limits.conf)
* soft nofile 2097152
* hard nofile 2097152
```

2.1.2 Kernel Parameters

```
# /etc/sysctl.conf

# TCP connection optimization
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_fin_timeout = 30
net.core.somaxconn = 65535
net.ipv4.tcp_max_syn_backlog = 65535

# Network buffers
net.core.rmem_max = 134217728
net.core.wmem_max = 134217728
net.ipv4.tcp_rmem = 4096 87380 134217728
net.ipv4.tcp_wmem = 4096 65536 134217728

# Port range
net.ipv4.ip_local_port_range = 1024 65535

# Memory optimization
vm.swappiness = 10
```

```
# Apply configuration
sysctl -p
```

2.1.3 Disk Optimization

Optimization Item	Recommendation
File system	Use XFS file system
Mount options	noatime,nodiratime
Disk scheduler	Use noop or deadline
SSD optimization	Enable TRIM

```
# View disk scheduler
cat /sys/block/sda/queue/scheduler

# Set to deadline
echo deadline > /sys/block/sda/queue/scheduler

# Mount options (/etc/fstab)
/dev/sda1 /var/lib/admq-mqtt xfs noatime,nodiratime 0 0
```

3 MQTT Configuration Optimization

3.1 Connection Management

Parameter	Description	Default	Recommended
max_connections	Maximum connections per listener	1024000	Adjust based on hardware
max_conn_rate	Maximum connection rate per second	1000	Adjust based on hardware
zone.external.idle_timeout	Idle connection timeout	7200s	Adjust based on business
zone.external.keepalive_backoff	Keepalive detection multiplier	0.75	0.75

```
# etc/admq-mqtt.conf
listeners.tcp.default {
    bind = "0.0.0.0:1883"
    max_connections = 1024000
}
```

3.2 Session Configuration

Parameter	Description	Default	Recommended
session_expiry_interval	Session expiration time	2h	Adjust based on business
max_subscriptions	Maximum subscriptions per client	infinity	1000-10000
max_inflight	In-flight window size	32	64-256
max_mqueue_len	Message queue length	1000	Adjust based on memory
mqueue_store_qos0	Whether to store QoS 0 messages	true	false (under high concurrency)

```
# etc/admq-mqtt.conf
zone.external {
    session_expiry_interval = 2h
}
```

```

max_subscriptions = 10000
max_inflight = 128
max_mqueue_len = 10000
mqueue_store_qos0 = false
}

```

3.3 Packet Size

Parameter	Description	Default	Recommended
max_packet_size	Maximum packet size	1MB	Adjust based on business
max_clientid_len	Maximum ClientID length	65535	1024
max_topic_levels	Maximum topic levels	128	7-10
max_topic_alias	Maximum topic aliases	65535	65535

```

# etc/admq-mqtt.conf
zone.external {
    max_packet_size = 1MB
    max_clientid_len = 1024
    max_topic_levels = 10
    max_topic_alias = 65535
}

```

3.4 JVM Configuration

```

# bin/admq-mqtt
JAVA_OPTS="-server -Xms4g -Xmx4g -Xmn2g"
JAVA_OPTS="$JAVA_OPTS -XX:+UseG1GC -XX:G1HeapRegionSize=16m"
JAVA_OPTS="$JAVA_OPTS -XX:MaxDirectMemorySize=2g"

```

3.5 Erlang VM Optimization

```
# vm.args
+P 2097152      # Maximum Erlang processes
+Q 2097152      # Maximum Erlang ports
+A 64           # Asynchronous thread pool size
```

4 Client Optimization

4.1 QoS Selection

Scenario	Recommended QoS
High-frequency Telemetry data, loss-tolerant	QoS 0
General business messages	QoS 1
Critical control commands	QoS 2

4.2 Connection Management

1. **Use connection pooling:** Avoid frequent creation and closing of connections
2. **Set Keepalive appropriately:** Avoid excessively short heartbeat intervals
3. **Enable auto-reconnect:** Prevent network anomalies from causing connection interruption
4. **Control subscription count:** A single client subscribing to too many topics affects performance

4.3 Message Size

1. **Control Payload size:** Avoid sending oversized messages
2. **Batch sending:** Reduce network round trips
3. **Avoid frequent reconnections:** Long connections are better than short connections

5 Cluster Optimization

5.1 Node Deployment Strategy

Deployment Strategy	Description	Suitable Scenarios
Same data center	All nodes in the same data center	Low latency, high throughput
Cross data center	Nodes distributed across different data centers	High availability, disaster recovery
Heterogeneous deployment	Mixed nodes with different configurations	Cost optimization

5.2 Load Balancing

1. **DNS round-robin:** Distribute client connections via DNS round-robin
2. **Hardware load balancer:** Use LVS, F5 and other hardware load balancers
3. **Software load balancer:** Use Nginx, HAProxy and other software load balancers

5.3 Session Persistence

Mode	Description	Suitable Scenarios
In-memory session	Session data stored in memory	High performance, low reliability
Persistent session	Session data persisted to disk	High reliability

6 Monitoring and Tuning

6.1 Performance Metrics

Metric	Healthy Range	Optimization Suggestion
CPU usage	< 70%	Scale up or optimize business logic
Memory usage	< 80%	Increase memory or optimize configuration
Disk usage	< 80%	Scale up or clean logs
Connection count	< max_connections	Scale up or optimize connection usage
Message throughput	Based on hardware configuration	Optimize producers or scale up
Message latency	< 10ms	Optimize network or consumers
Slow subscription count	< 100	Optimize consumption logic

6.2 Performance Testing

```
# Test using emqtt_bench tool
./emqtt_bench conn -h localhost -p 1883 -c 10000 -i 10

# Parameter description
# -h: Broker address
# -p: Port
# -c: Concurrent connections
# -i: Connection interval (milliseconds)

# Publish test
./emqtt_bench pub -h localhost -p 1883 -c 1000 -I 10 -t
"sensor/%c/temperature" -s 100

# Subscribe test
./emqtt_bench sub -h localhost -p 1883 -c 1000 -t
"sensor/+ /temperature"
```

6.3 Bottleneck Analysis

1. **CPU bottleneck:** Encryption/decryption, message routing, rule engine processing
2. **Memory bottleneck:** Large number of connections, session data, message queues
3. **Disk bottleneck:** Persistent messages, log writes
4. **Network bottleneck:** Large number of small messages, large payloads, network latency

7 Common Issue Optimization

7.1 Insufficient Connection Count

1. Increase file descriptor limits
2. Adjust the max_connections parameter
3. Scale up cluster nodes
4. Use load balancers to distribute connections

7.2 Message Latency Optimization

1. Optimize network configuration
2. Reduce QoS 2 usage
3. Optimize consumer processing logic
4. Use shared subscriptions for load balancing

7.3 Insufficient Memory Optimization

1. Increase physical memory
2. Reduce session expiration time
3. Limit message queue length
4. Disable QoS 0 message storage

7.4 Message Backlog Optimization

1. Increase consumer count
2. Optimize consumer processing logic
3. Use shared subscriptions
4. Check if ACL rules are blocking consumption

8 Performance Tuning Checklist

8.1 System Level

- ☐ File descriptor limits have been adjusted
- ☐ Kernel parameters have been optimized
- ☐ Disk uses XFS file system
- ☐ Network parameters have been optimized
- ☐ System resources are sufficient (CPU, memory, disk)

8.2 MQTT Configuration

- ☐ max_connections set appropriately
- ☐ Session expiry interval set appropriately
- ☐ In-flight window size set appropriately
- ☐ Message queue length set appropriately
- ☐ Packet size limits set appropriately
- ☐ Erlang VM parameters have been optimized

8.3 Client Optimization

- ☐ QoS selection is appropriate
- ☐ Keepalive interval is reasonable
- ☐ Subscription count is reasonably controlled
- ☐ Message size is reasonably controlled
- ☐ Use long connections and auto-reconnect

8.4 Cluster Optimization

- ☐ Node deployment strategy is appropriate
- ☐ Load balancer has been configured
- ☐ Cluster size meets business requirements
- ☐ Session persistence strategy is appropriate

UNIFIED SERVICE HOTLINE

4008-555-800

Kingdee Apusic Cloud Computing Co., Ltd. (referred to as “Kingdee Apusic”) was founded in 2000. Its predecessor was “Kingdee Middleware Company.” It is a basic software platform provider under Kingdee Group, a national standards-setting enterprise for cloud computing, and a core software enterprise in the national IT application innovation industry. Kingdee Apusic is an enterprise undertaking the National 863 Key R&D Program and the National Major Science and Technology Project for Core Electronic Components, High-end General-purpose Chips and Fundamental Software, and is also a basic platform provider for the national key projects of “Two Networks, One Website, Four Databases, and Twelve Golden Projects.” Its products are widely used in key industries such as government, military, finance, and energy, and it has cumulatively served more than 100,000 customers.



National standard setter for cloud computing
Basic software enterprise under Kingdee Group
Core enterprise for IT application innovation
Website: www.apusic.com

