



Performance Tuning

Kingdee Apusic Distributed Message Queue
V2.0.6_for_rocketmq

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
 - 2.2 Network Optimization
- 3 RocketMQ Configuration Optimization
 - 3.1 Broker Configuration
 - 3.2 Storage Configuration
 - 3.3 JVM Configuration
 - 3.3.1 NameServer JVM
 - 3.3.2 Broker JVM
- 4 Client Optimization
 - 4.1 Producer Optimization
 - 4.1.1 Send Mode Selection
 - 4.1.2 Producer Parameters
 - 4.1.3 Batch Send
 - 4.2 Consumer Optimization
 - 4.2.1 Consumer Thread Count
 - 4.2.2 Consumption Batch Size
 - 4.2.3 Consumption Offset Strategy
- 5 Cluster Optimization
 - 5.1 Node Deployment Strategy
 - 5.2 Master-Slave Replication Optimization
 - 5.2.1 Synchronous Replication vs Asynchronous Replication
 - 5.3 Flush Strategy Optimization
 - 5.3.1 Synchronous Flush vs Asynchronous Flush
 - 5.4 Topic Queue Count Optimization

- 6 Monitoring and Tuning
 - 6.1 Performance Metrics
 - 6.2 Performance Testing
 - 6.3 Bottleneck Analysis
- 7 Common Issue Optimization
 - 7.1 Message Accumulation Optimization
 - 7.2 Send Performance Optimization
 - 7.3 Consumption Lag Optimization
- 8 Performance Optimization Checklist
 - 8.1 System Level
 - 8.2 RocketMQ Configuration
 - 8.3 Client Optimization
 - 8.4 Cluster Optimization

1 Preface

This document introduces the best practices for performance optimization of Kingdee Apusic Distributed Message Queue for RocketMQ (Apusic Distributed Message Queue for RocketMQ, referred to as: ADMQ for RocketMQ), helping users fully leverage system performance.

1.1 Target Audience

This document is applicable to IT information business leaders, R&D managers, software project managers, software architects, and operations engineers.

1.2 Related Documents

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

No.	Document	Description
1	Kingdee Apusic Distributed Message Queue for RocketMQ Quick Start Guide	Briefly introduces how to quickly get started with ADMQ for RocketMQ.
2	Kingdee Apusic Distributed Message Queue for RocketMQ Installation Manual	Details how to install ADMQ for RocketMQ on various operating systems, as well as starting and stopping ADMQ for RocketMQ services.
3	Kingdee Apusic Distributed Message Queue for RocketMQ Message Engine User Manual	Details the use, configuration, and management of ADMQ for RocketMQ message engine related functions and supporting tools.
4	Kingdee Apusic Distributed Message Queue for RocketMQ Console User Manual	Details the use and operation of ADMQ for RocketMQ console related functions.
5	Kingdee Apusic Distributed Message Queue for RocketMQ Development Manual	Details the development of ADMQ for RocketMQ client applications based on various development languages.
6	Kingdee Apusic Distributed Message Queue for RocketMQ Migration Manual	Details the migration from RocketMQ to ADMQ for RocketMQ.
7	Kingdee Apusic Distributed Message Queue for RocketMQ Operations Manual	Details monitoring, operation, security hardening, and other operational instructions for ADMQ for RocketMQ.

8	Kingdee Apusic Distributed Message Queue for RocketMQ Performance Optimization Manual	Details performance tuning instructions for ADMQ for RocketMQ.
---	---	--

1.3 Technical Support

ADMQ for RocketMQ products provide comprehensive technical support services. You can obtain technical support through the following methods:

- 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 you obtain 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 logs, screenshots, etc. of exceptions and errors

2 System-Level Optimization

2.1 Operating System Optimization

2.1.1 File Descriptors

```
# View current limit
ulimit -n

# Modify limit (/etc/security/limits.conf)
* soft nofile 65535
* hard nofile 65535
```

2.1.2 Kernel Parameters

```
# /etc/sysctl.conf

# Increase port range
net.ipv4.ip_local_port_range = 1024 65535

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

# Memory optimization
vm.swappiness = 10
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10

# Apply configuration
sysctl -p
```

2.1.3 Disk Optimization

Optimization Item	Suggestion
File System	Use XFS file system
Mount Options	noatime,nodiratime
Disk Scheduler	Use noop or deadline
RAID Configuration	Use RAID 10 or RAID 0
SSD Optimization	Enable TRIM, reserve over-provisioning

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

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

# Mount options (/etc/fstab)
/dev/sda1 /data/rocketmq xfs noatime,nodiratime 0 0
```

2.2 Network Optimization

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

# TCP optimization
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_sack = 1
net.ipv4.tcp_no_metrics_save = 1
```

3 RocketMQ Configuration Optimization

3.1 Broker Configuration

Parameter	Default	Recommended	Description
sendMessageThreadPoolNums	1	CPU cores	Number of send message threads
pullMessageThreadPoolNums	16+	Adjust based on load	Number of pull message threads
queryMessageThreadPoolNums	8	Adjust based on load	Number of query message threads
adminBrokerThreadPoolNums	16	Adjust based on load	Number of admin operation threads
brokerRole	ASYNC_MASTER	SYNC_MASTER	Broker role
flushDiskType	ASYNC_FLUSH	SYNC_FLUSH	Flush disk mode
flushIntervalCommitLog	500ms	Based on reliability requirements	CommitLog flush interval
flushIntervalConsumeQueue	1000ms	Based on reliability requirements	ConsumeQueue flush interval

```
# broker.conf
sendMessageThreadPoolNums=16
pullMessageThreadPoolNums=16
brokerRole=SYNC_MASTER
flushDiskType=SYNC_FLUSH
flushIntervalCommitLog=500
flushIntervalConsumeQueue=1000
```

3.2 Storage Configuration

Parameter	Default	Recommended	Description
mappedFileSizeCommitLog	1GB	1GB	CommitLog file size

mappedFileSizeConsumeQueue	300MB	300MB	ConsumeQueue file size
flushCommitLogTimed	false	false	Whether to flush disk periodically
deleteWhen	04	04	Time to delete expired files
fileReservedTime	72	Based on disk capacity	File retention time (hours)
diskMaxUsedSpaceRatio	75	75	Maximum disk usage ratio

```
# broker.conf
mappedFileSizeCommitLog=1073741824
mappedFileSizeConsumeQueue=300000000
deleteWhen=04
fileReservedTime=72
diskMaxUsedSpaceRatio=75
```

3.3 JVM Configuration

3.3.1 NameServer JVM

```
# bin/runserver.sh
JAVA_OPT="${JAVA_OPT} -server -Xms4g -Xmx4g -Xmn2g"
JAVA_OPT="${JAVA_OPT} -XX:+UseG1GC -XX:G1HeapRegionSize=16m"
JAVA_OPT="${JAVA_OPT} -XX:G1ReservePercent=25"
JAVA_OPT="${JAVA_OPT} -XX:InitiatingHeapOccupancyPercent=30"
```

3.3.2 Broker JVM

```
# bin/runbroker.sh
JAVA_OPT="${JAVA_OPT} -server -Xms8g -Xmx8g -Xmn4g"
JAVA_OPT="${JAVA_OPT} -XX:+UseG1GC -XX:G1HeapRegionSize=16m"
JAVA_OPT="${JAVA_OPT} -XX:G1ReservePercent=25"
JAVA_OPT="${JAVA_OPT} -XX:InitiatingHeapOccupancyPercent=30"
JAVA_OPT="${JAVA_OPT} -XX:MaxDirectMemorySize=4g"
```

4 Client Optimization

4.1 Producer Optimization

4.1.1 Send Mode Selection

Send Mode	Throughput	Reliability	Use Case
Synchronous Send	Medium	High	Scenarios requiring confirmation
Asynchronous Send	High	Medium	High throughput scenarios
One-way Send	Very High	Low	Log collection, etc.
Batch Send	Very High	Medium	High throughput scenarios

4.1.2 Producer Parameters

```
DefaultMQProducer producer = new DefaultMQProducer("test-group");
producer.setNamesrvAddr("localhost:9876");

// Send timeout
producer.setSendMsgTimeout(3000);

// Message body compression threshold (compress if exceeded)
producer.setCompressMsgBodyOverHowmuch(1024 * 4);

// Maximum message size
producer.setMaxMessageSize(1024 * 1024 * 4);

// Retry count
producer.setRetryTimesWhenSendFailed(2);
producer.setRetryTimesWhenSendAsyncFailed(2);

// Whether to retry another Broker on internal send failure
producer.setRetryAnotherBrokerWhenNotStoreOK(false);
```

4.1.3 Batch Send

```
List<Message> messages = new ArrayList<>();
for (int i = 0; i < 100; i++) {
    messages.add(new Message("batch-topic", "tag-a", ("msg-" +
i).getBytes()));
}
// Batch send, single batch should not exceed 1MB
producer.send(messages);
```

4.2 Consumer Optimization

4.2.1 Consumer Thread Count

```
DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("test-
group");
consumer.setNamesrvAddr("localhost:9876");

// Minimum consumer thread count
consumer.setConsumeThreadMin(20);

// Maximum consumer thread count
consumer.setConsumeThreadMax(64);

// Consumption timeout (minutes)
consumer.setConsumeTimeout(15);
```

4.2.2 Consumption Batch Size

```
// Maximum number of messages per consumption
consumer.setConsumeMessageBatchMaxSize(10);

// Maximum number of messages per pull
consumer.setPullBatchSize(32);
```

```
// Pull interval (milliseconds)
consumer.setPullInterval(0);
```

4.2.3 Consumption Offset Strategy

```
// Start consumption from the last offset
consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_LAST_OFFSET)

// Start consumption from the first offset
consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_FIRST_OFFSET)

// Start consumption from a specified time
consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_TIMESTAMP);
consumer.setConsumeTimestamp("20260101000000");
```

5 Cluster Optimization

5.1 Node Deployment Strategy

Deployment Strategy	Description	Use Case
Same Data Center Deployment	All nodes in the same data center	Low latency, high throughput
Cross Data Center Deployment	Nodes distributed across different data centers	High availability, disaster recovery
Heterogeneous Deployment	Mix of different configuration nodes	Cost optimization

5.2 Master-Slave Replication Optimization

5.2.1 Synchronous Replication vs Asynchronous Replication

Replication Mode	Reliability	Performance	Use Case
Synchronous Replication	High	Low	Strong consistency scenarios like financial transactions
Asynchronous Replication	Medium	High	General business scenarios

```
# broker.conf
# Synchronous replication
brokerRole=SYNC_MASTER

# Asynchronous replication
brokerRole=ASYNC_MASTER
```

5.3 Flush Strategy Optimization

5.3.1 Synchronous Flush vs Asynchronous Flush

Flush Mode	Reliability	Performance	Use Case
Synchronous Flush	High	Low	Strong consistency scenarios

Asynchronous Flush	Medium	High	General business scenarios
--------------------	--------	------	----------------------------

```
# broker.conf
# Synchronous flush
flushDiskType=SYNC_FLUSH

# Asynchronous flush
flushDiskType=ASYNC_FLUSH
```

5.4 Topic Queue Count Optimization

Scenario	Queue Count Suggestion
Low Throughput	4-8
Medium Throughput	8-16
High Throughput	16-32
Very High Throughput	32-64

```
# Specify queue count when creating Topic
bin/mqadmin updateTopic -n localhost:9876 -c DefaultCluster -t test-
topic -r 16 -w 16
```

6 Monitoring and Tuning

6.1 Performance Metrics

Metric	Healthy Range	Optimization Suggestion
Broker TPS	Based on hardware configuration	Scale up or optimize producers
Consumption TPS	Close to production TPS	Add consumers or optimize consumption logic
Message Accumulation	< 1 million	Add consumers or optimize consumption logic
Disk Usage	< 75%	Scale up or clean expired messages
Memory Usage	< 80%	Increase memory or optimize JVM parameters
Consumption Lag	< 1 second	Optimize consumption logic or add consumers

6.2 Performance Testing

```
# Test using benchmark tool
bin/benchmark.sh producer -n localhost:9876 -t test-topic -s 128 -w
16

# Parameter description
# -n: NameServer address
# -t: Topic name
# -s: Message size (bytes)
# -w: Concurrency count
```

6.3 Bottleneck Analysis

1. **CPU Bottleneck:** Insufficient send/consume threads, compression/decompression, serialization
2. **Memory Bottleneck:** Frequent JVM GC, insufficient DirectBuffer

3. **Disk Bottleneck:** Flush mode, insufficient disk I/O, file system performance
4. **Network Bottleneck:** Cross-data center deployment, insufficient bandwidth, network latency

7 Common Issue Optimization

7.1 Message Accumulation Optimization

1. Increase consumer count
2. Optimize consumer processing logic
3. Increase consumption thread count
4. Use batch consumption
5. Scale up Broker nodes

7.2 Send Performance Optimization

1. Use asynchronous sending
2. Use batch sending
3. Increase producer count
4. Optimize message size
5. Disable message tracing (if not needed)

7.3 Consumption Lag Optimization

1. Increase consumption thread count
2. Optimize consumption logic
3. Use concurrent consumption
4. Scale up consumers
5. Check for slow consumers

8 Performance Optimization 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 RocketMQ Configuration

- ☐ Broker thread count set appropriately
- ☐ Flush strategy selected based on business
- ☐ Replication strategy selected based on business
- ☐ File retention time set appropriately
- ☐ JVM parameters have been optimized

8.3 Client Optimization

- ☐ Send mode selected appropriately
- ☐ Consumption thread count set appropriately
- ☐ Batch size set appropriately
- ☐ Retry count set appropriately
- ☐ Consumption offset strategy selected appropriately

8.4 Cluster Optimization

- ☐ Topic queue count set appropriately
- ☐ Master-slave replication strategy selected appropriately
- ☐ Flush strategy selected appropriately
- ☐ Cluster scale meets business requirements
- ☐ Node deployment strategy selected appropriately

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

