



Performance Tuning

金蝶Apusic分布式消息队列V2.0.6_for_kafka

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 Overview
- 3 Hardware and OS Optimization
 - 3.1 Disk Selection
 - 3.2 OS Parameter Tuning
 - 3.3 NIC Optimization
- 4 Broker-Side Optimization
 - 4.1 Network and Thread Configuration
 - 4.2 Log Storage Configuration
 - 4.3 Replication and Reliability Configuration
 - 4.4 High-Throughput Scenario Configuration
- 5 Producer Optimization
 - 5.1 Core Configuration Parameters
 - 5.2 High-Throughput Configuration
 - 5.3 Partition Strategy Optimization
- 6 Consumer Optimization
 - 6.1 Core Configuration Parameters
 - 6.2 High-Throughput Configuration
 - 6.3 Consumer Multi-Threading Optimization
- 7 JVM Tuning
 - 7.1 Memory Configuration
 - 7.2 Recommended GC Tuning Configuration
 - 7.3 Common Problem Diagnosis
- 8 Topic Design Optimization
 - 8.1 Partition Count Design
 - 8.2 Partition Rebalancing
 - 8.3 Log Compaction Configuration
- 9 Monitoring and Diagnostics
 - 9.1 Key Performance Metrics

- 9.2 Monitoring with JMX
- 9.3 Monitoring with Prometheus
- 10 Common Performance Issues
- 11 Performance Testing Recommendations
 - 11.1 Test Tools
 - 11.2 Test Commands

1 Preface

This document introduces the best practices for performance optimization of ADMQ for Kafka (Apusic Distributed Message Queue for Kafka), helping users fully leverage 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

For more information about ADMQ for Kafka, refer to the following ADMQ for Kafka product documentation set:

No.	Document	Description
1	ADMQ for Kafka Quick Start Guide	Briefly describes how to quickly get started with ADMQ for Kafka.
2	ADMQ for Kafka Installation Guide	Details how to install ADMQ for Kafka on various operating systems, as well as starting and stopping ADMQ for Kafka services.
3	ADMQ for Kafka Message Engine User Guide	Details the usage, configuration, and management of ADMQ for Kafka message engine features and supporting tools.
4	ADMQ for Kafka Console User Guide	Details the usage and operation of the ADMQ for Kafka console features.
5	ADMQ for Kafka Developer Guide	Details how to develop ADMQ for Kafka client applications using various programming languages.
6	ADMQ for Kafka Migration Guide	Details the migration from Kafka to ADMQ for Kafka.
7	ADMQ for Kafka Operations Guide	Details monitoring, operations, and security hardening for ADMQ for Kafka.
8	ADMQ for Kafka Performance Tuning Guide	Details performance tuning for ADMQ for Kafka.

1.3 Technical Support

ADMQ for Kafka 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 version
4. Product version number
5. Detailed information such as logs and screenshots of anomalies and errors

2 Overview

Performance optimization requires a comprehensive approach across multiple layers:

- Hardware: CPU, memory, disk, network
- OS: Operating system parameters, kernel tuning
- Broker: Configuration parameter optimization
- Client: Producer and consumer configurations
- Architecture: Topic design, partitioning strategy

3 Hardware and OS Optimization

3.1 Disk Selection

Disk Type	Recommended Scenario	Description
NVMe SSD	High-throughput scenarios	Best performance, suitable for core business
SSD	Production environments	Better than HDD, recommended
HDD	Low-cost scenarios	Only suitable for log-type scenarios

Optimization Recommendations:

- Use dedicated disks for data directories
- Enable disk RAID 0/10 for better performance
- Avoid using network storage (NFS) as the primary data directory

3.2 OS Parameter Tuning

```
# File descriptor limits
ulimit -n 1000000

# Network parameter optimization
sysctl -w net.core.rmem_max=67108864
sysctl -w net.core.wmem_max=67108864
sysctl -w net.ipv4.tcp_rmem="4096 87380 67108864"
sysctl -w net.ipv4.tcp_wmem="4096 16384 67108864"
sysctl -w net.core.netdev_max_backlog=250000

# Disable swap
sysctl -w vm.swappiness=0

# File system optimization
mount -o noatime,nodiratime /dev/sda1 /mnt/kafka
```

3.3 NIC Optimization

- Use multi-queue NICs with CPU affinity
- Enable jumbo frames (MTU 9000)
- Consider DPDK acceleration (high-throughput scenarios)

4 Broker-Side Optimization

4.1 Network and Thread Configuration

Parameter	Default	Recommended	Description
<code>num.network.threads</code>	8	CPU cores	Number of threads handling network requests
<code>num.io.threads</code>	16	Disks × 2	Number of threads handling disk I/O
<code>socket.send.buffer.bytes</code>	102400	204800	Send buffer size
<code>socket.receive.buffer.bytes</code>	102400	204800	Receive buffer size
<code>queued.max.requests</code>	500	1000	Request queue length waiting for I/O

4.2 Log Storage Configuration

Parameter	Default	Recommended	Description
<code>log.segment.bytes</code>	1GB	512MB	Log segment size, smaller segments speed up log cleanup
<code>log.retention.hours</code>	168	Adjust per business	Log retention time
<code>log.retention.check.interval.ms</code>	300000	60000	Log cleanup check interval
<code>log.flush.interval.ms</code>	null	Not recommended to modify	Recommended to rely on PageCache
<code>log.flush.scheduler.interval.ms</code>	null	Not recommended to modify	Recommended to rely on PageCache

4.3 Replication and Reliability Configuration

Parameter	Recommended	Description
<code>default.replication.factor</code>	3	3 replicas recommended for production
<code>min.insync.replicas</code>	2	Use with acks=all
<code>unclean.leader.election.enable</code>	false	Disable non-ISR leader election

replica.lag.time.max.ms	30000	ISR eviction threshold
-------------------------	-------	------------------------

4.4 High-Throughput Scenario Configuration

```
# Network optimization
num.network.threads=16
num.io.threads=32
socket.send.buffer.bytes=307200
socket.receive.buffer.bytes=307200
queued.max.requests=2000

# Log optimization
log.segment.bytes=524288000
log.retention.check.interval.ms=300000

# Compression optimization
compression.type=zstd

# Replica optimization
replica.socket.timeout.ms=30000
replica.fetch.max.bytes=1048576
replica.fetch.min.bytes=1
```

5 Producer Optimization

5.1 Core Configuration Parameters

Parameter	Recommended	Description
acks	all	Use when reliability is prioritized
compression.type	zstd	ZSTD offers the highest compression ratio
batch.size	16384-32768	Batch size (bytes)
linger.ms	5-20	Batch wait time
buffer.memory	64MB	Send buffer
max.in.flight.requests.per.connection	5	≤5 when idempotence is enabled
retries	Integer.MAX_VALUE	Number of retries
enable.idempotence	true	Enable idempotence

5.2 High-Throughput Configuration

```
Properties props = new Properties();
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"192.168.1.10:9092");
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());

// Reliability configuration
props.put(ProducerConfig.ACKS_CONFIG, "all");
props.put(ProducerConfig.ENABLE_IDEMPOTENCE_CONFIG, true);

// Throughput optimization
props.put(ProducerConfig.COMPRESSION_TYPE_CONFIG, "zstd");
props.put(ProducerConfig.BATCH_SIZE_CONFIG, 32768);
```

```

props.put(ProducerConfig.LINGER_MS_CONFIG, 10);
props.put(ProducerConfig.BUFFER_MEMORY_CONFIG, 67108864);
props.put(ProducerConfig.MAX_IN_FLIGHT_REQUESTS_PER_CONNECTION, 5);
props.put(ProducerConfig.REQUEST_TIMEOUT_MS_CONFIG, 30000);
props.put(ProducerConfig.DELIVERY_TIMEOUT_MS_CONFIG, 120000);

// Retry configuration
props.put(ProducerConfig.RETRIES_CONFIG, Integer.MAX_VALUE);
props.put(ProducerConfig.RETRY_BACKOFF_MS_CONFIG, 100);

```

5.3 Partition Strategy Optimization

1. Key Partitioning: Use business keys for ordering guarantees
2. Round-Robin Partitioning: Achieve load balancing
3. Custom Partitioner: Partition based on business logic

```

// Custom partitioner example
public class CustomPartitioner implements Partitioner {
    @Override
    public int partition(String topic, Object key, byte[] keyBytes,
                        Object value, byte[] valueBytes, Cluster
cluster) {
        // Implement partitioning based on business logic
        return Math.abs(key.hashCode()) %
cluster.partitionCountFor(topic);
    }
}

```

6 Consumer Optimization

6.1 Core Configuration Parameters

Parameter	Recommended	Description
<code>fetch.min.bytes</code>	1	Minimum fetch bytes
<code>fetch.max.wait.ms</code>	500	Maximum wait time
<code>max.poll.records</code>	500	Maximum records per poll
<code>session.timeout.ms</code>	30000	Session timeout
<code>heartbeat.interval.ms</code>	10000	Heartbeat interval
<code>enable.auto.commit</code>	false	Manual commit is more reliable
<code>auto.offset.reset</code>	earliest	Consume from beginning
<code>max.partition.fetch.bytes</code>	1048576	Maximum fetch bytes per partition

6.2 High-Throughput Configuration

```
Properties props = new Properties();
props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"192.168.1.10:9092");
props.put(ConsumerConfig.GROUP_ID_CONFIG, "my-consumer-group");
props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());

// Throughput optimization
props.put(ConsumerConfig.FETCH_MIN_BYTES_CONFIG, 1);
props.put(ConsumerConfig.FETCH_MAX_WAIT_MS_CONFIG, 500);
props.put(ConsumerConfig.MAX_POLL_RECORDS_CONFIG, 1000);
props.put(ConsumerConfig.MAX_PARTITION_FETCH_BYTES_CONFIG,
10485760);
```

```
// Reliability configuration
props.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, false);
props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
props.put(ConsumerConfig.SESSION_TIMEOUT_MS_CONFIG, 30000);
props.put(ConsumerConfig.HEARTBEAT_INTERVAL_MS_CONFIG, 10000);

// Fetch optimization
props.put(ConsumerConfig.DEFAULT_API_TIMEOUT_MS_CONFIG, 60000);
```

6.3 Consumer Multi-Threading Optimization

```
// Multi-consumer thread example
ExecutorService executor = Executors.newFixedThreadPool(4);
for (int i = 0; i < 4; i++) {
    final int threadId = i;
    executor.submit(() -> {
        KafkaConsumer<String, String> consumer = createConsumer();
        consumer.subscribe(Arrays.asList("my-topic"));
        while (true) {
            ConsumerRecords<String, String> records =
consumer.poll(100);
            for (ConsumerRecord<String, String> record : records) {
                processRecord(record);
            }
            consumer.commitSync();
        }
    });
}
```

7 JVM Tuning

7.1 Memory Configuration

Scenario	Heap Memory	GC Strategy
Dev/Test	-Xms2g -Xmx2g	G1
Production (small cluster)	-Xms8g -Xmx8g	G1
Production (large cluster)	-Xms16g -Xmx16g	G1

7.2 Recommended GC Tuning Configuration

```
# G1 GC configuration
KAFKA_HEAP_OPTS="-Xms16g -Xmx16g -XX:+UseG1GC \
  -XX:MaxGCPauseMillis=20 \
  -XX:InitiatingHeapOccupancyPercent=35 \
  -XX:G1HeapRegionSize=16m \
  -XX:+ParallelRefProcEnabled"

# JMX configuration (for monitoring)
KAFKA_JMX_OPTS="-Dcom.sun.management.jmxremote \
  -Dcom.sun.management.jmxremote.port=9999 \
  -Dcom.sun.management.jmxremote.ssl=false \
  -Dcom.sun.management.jmxremote.authenticate=false"
```

7.3 Common Problem Diagnosis

Issue	Possible Cause	Solution
Frequent Full GC	Insufficient heap memory	Increase heap memory, optimize consumption logic
Long GC pauses	Improper G1 configuration	Adjust MaxGCPauseMillis
OOM	Memory leak or misconfiguration	Check JVM and client configurations

8 Topic Design Optimization

8.1 Partition Count Design

Target Throughput	Recommended Partitions	Description
100K/sec	10-20	~50-100K/sec per partition
500K/sec	50-100	Requires more Brokers
1M/sec	100-200	Requires large cluster

Formula:

```
Partitions = Target Throughput / Throughput per Partition
Throughput per Partition ≈ 50-100K/sec (depends on message size and hardware)
```

8.2 Partition Rebalancing

Adding partitions triggers rebalancing, affecting performance. Recommendations:

- Plan sufficient partitions in advance
- Use sticky partition strategy (Kafka 2.4+)

```
# Sticky partition strategy (default)
kafka-topics.sh --alter --topic my-topic --partitions 20
```

8.3 Log Compaction Configuration

```
# Log compaction configuration
cleanup.policy=compact
min.compaction.lag.ms=3600000
delete.retention.ms=86400000
segment.bytes=1073741824
```

9 Monitoring and Diagnostics

9.1 Key Performance Metrics

Metric	Description	Alert Threshold
MessagesInPerSec	Messages per second	Set per business
BytesInPerSec	Inbound bytes per second	Set per business
BytesOutPerSec	Outbound bytes per second	Set per business
UnderReplicatedPartitions	Under-replicated partitions	> 0
OfflinePartitionsCount	Offline partitions	> 0
RequestLatencyAvg	Average request latency	> 100ms
FetchQueueSize	Request queue size	Continuously growing
PurgatorySize	Pending requests	Continuously growing

9.2 Monitoring with JMX

```
# Enable JMX port
JMX_PORT=9999 bin/admq-daemon start kafka standalone

# Connect using jconsole
jconsole localhost:9999
```

9.3 Monitoring with Prometheus

Configure Prometheus to collect metrics:

```
scrape_configs:
  - job_name: 'kafka'
    static_configs:
      - targets: ['localhost:12305']
```

10 Common Performance Issues

Issue	Symptom	Solution
High producer latency	Increased send time	Increase batch.size, reduce linger.ms
Large consumer lag	Slow consumption	Increase consumer count, optimize processing logic
High disk I/O	Slower writes	Use SSD, optimize log retention policy
Network bottleneck	Low throughput	Increase network bandwidth, use compression
Memory pressure	Frequent GC	Increase heap memory, adjust GC parameters

11 Performance Testing Recommendations

11.1 Test Tools

- kafka-producer-perf-test.sh: Producer performance testing
- kafka-consumer-perf-test.sh: Consumer performance testing
- kafka-throttle.sh: Throttle testing

11.2 Test Commands

```
# Producer performance test
kafka/bin/kafka-producer-perf-test.sh \
  --topic test-topic \
  --num-records 1000000 \
  --record-size 1024 \
  --throughput -1 \
  --producer-props bootstrap.servers=192.168.1.10:9092

# Consumer performance test
kafka/bin/kafka-consumer-perf-test.sh \
  --topic test-topic \
  --messages 1000000 \
  --group perf-test-group \
  --bootstrap-server 192.168.1.10:9092
```

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

