

# Distributed Message Service for Kafka

## Performance White Paper

**Issue** 01  
**Date** 2023-09-05



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2023. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

# Contents

**1 Performance Test on the Client Production Rate and Server CPU Consumption.... 1**

1.1 Test Scenarios..... 1

1.2 Test Environment..... 1

1.3 Test Procedure..... 3

1.4 Test Results..... 17

**2 TPS Test for Instances of Different Specifications.....18**

2.1 Test Scenarios..... 18

2.2 Test Environment..... 18

2.3 Test Results..... 20

# 1 Performance Test on the Client Production Rate and Server CPU Consumption

---

## 1.1 Test Scenarios

This document describes performance tests on Distributed Message Service (DMS) for Kafka. The performance is measured by the message production rate on the client side and CPU usage on the server side. The tests cover the following scenarios:

- Test scenario 1 (batch size): same Kafka instance, same topics, different **batch.size** settings
- Test scenario 2 (cross-AZ or intra-AZ production): same Kafka instance, same topics, different AZ settings for the client and server
- Test scenario 3 (number of replicas): same Kafka instance, different numbers of replicas
- Test scenario 4 (synchronous or asynchronous replication): same Kafka instance, topics with different replication settings

## 1.2 Test Environment

Perform the following steps to set up the test environment.

### Step 1: Buy a Kafka Instance

Buy a Kafka instance with the following configurations. For details about how to buy a Kafka instance, see [Buying an Instance](#).

- Region: CN-Hong Kong
- Project: CN-Hong Kong
- AZ: AZ1
- Instance name: kafka-test

- Enterprise project: default
- Version: 2.7
- Broker flavor: kafka.2u4g.cluster
- Brokers: 3
- Storage space: ultra-high I/O, 200 GB
- Capacity threshold policy: automatically delete
- VPC: If there is no available VPC, create one by referring to [Preparing Required Resources](#).
- Security group: Select a security group that meets the requirements specified in [Preparing Required Resources](#).
- Other: public network access, Smart Connect, Kafka SASL\_SSL, and automatic topic creation disabled

After the purchase is complete, obtain the address of the Kafka instance on the instance details page.

#### Connection

Username

--

Kafka SASL\_SSL

Disabled Fixed for this instance

Instance Address (Private Network)

IPv4

192.168.0.69:9092 ,192.168.0.42:9092 ,192.168.0.66:9092



## Step 2: Create Topics

Create the following three topics for the [Kafka instance](#). For details about how to create a topic, see [Creating a Topic](#).

- Topic-01: 3 partitions, 1 replica, asynchronous replication
- Topic-02: 3 partition, 3 replicas, asynchronous replication
- Topic-03: 3 partition, 3 replicas, synchronous replication

## Step 3: Obtain the Testing Tool

Download [Kafka CLI v2.7.2](#).

## Step 4: Buy ECSs

Buy two ECSs with the following configurations. For details about how to purchase an ECS, see [Purchasing an ECS](#).

- One ECS is 4 vCPUs | 8 GB, runs Linux, and is configured with the same region, AZ, VPC, subnet, and security group as the Kafka instance purchased in [Step 1: Buy a Kafka Instance](#).
- The other ECS is 4 vCPUs | 8 GB, runs Linux, and is configured with the same region, VPC, subnet, and security group but a different AZ from the Kafka instance purchased in [Step 1: Buy a Kafka Instance](#).

Perform the following operations on the ECSs:

- Install [Java JDK](#) and configure the environment variables **JAVA\_HOME** and **PATH**.

```
export JAVA_HOME=/root/jdk1.8.0_231
export PATH=$JAVA_HOME/bin:$PATH
```

- Download [Kafka CLI v2.7.2](#) and decompress it.

```
tar -zxf kafka_2.12-2.7.2.tgz
```

## 1.3 Test Procedure

Test the message production rate on the client side and the CPU usage on the server side in the following scenarios:

- Test scenario 1 (batch size): same Kafka instance, same topics, different **batch.size** settings
- Test scenario 2 (cross-AZ or intra-AZ production): same Kafka instance, same topics, different AZ settings for the client and server
- Test scenario 3 (number of replicas): same Kafka instance, different numbers of replicas
- Test scenario 4 (synchronous or asynchronous replication): same Kafka instance, topics with different replication settings

**Table 1-1** Test parameters

Partitions	Replicas	Synchronous Replication Enabled	batch.size	Cross-AZ Production
3	1	No	1 KB	No
3	1	No	16 KB	No
3	1	No	1 KB	Yes
3	3	Yes	1 KB	No
3	3	No	1 KB	No

Test script:

```
./kafka-producer-perf-test.sh --producer-props bootstrap.servers=${connection address} acks=1 batch.size=${batch.size} linger.ms=0 --topic ${topic name} --num-records ${num-records} --record-size 1024 --throughput -102400
```

- **bootstrap.servers**: address of the Kafka instance obtained in [Step 1: Buy a Kafka Instance](#).
- **acks**: message synchronization policy. acks=1 indicates asynchronous replication, and acks=-1 indicates synchronous replication.
- **batch.size**: size of messages sent in each batch, in bytes.
- **linger.ms**: interval between two batches.
- **topic**: topic name set in [Step 2: Create Topics](#).

- **num-records**: total number of messages to be sent.
- **record-size**: size of each message.
- **throughput**: number of messages sent per second.

## Test Scenario 1: Varied Batch Sizes

**Step 1** Log in to the client server, go to the **kafka\_2.12-2.7.2/bin** directory, and run the following scripts.

Set **batch.size** to 1 KB, and run the following script:

```
./kafka-producer-perf-test.sh --producer-props  
bootstrap.servers=192.168.0.69:9092,192.168.0.42:9092,192.168.0.66:9092 acks=1 batch.size=1024  
linger.ms=0 --topic Topic-01 --num-records 8000000 --record-size 1024 --throughput 102400
```

The result is as follows:

```
8000000 records sent, 37696.729809 records/sec (36.81 MB/sec), 796.54 ms avg latency, 3838.00 ms max  
latency, 322 ms 50th, 2282 ms 95th, 2745 ms 99th, 3593 ms 99.9th.
```

Message production rate: 37,697 records/second


Set **batch.size** to 16 KB, and run the following script:

```
./kafka-producer-perf-test.sh --producer-props  
bootstrap.servers=192.168.0.69:9092,192.168.0.42:9092,192.168.0.66:9092 acks=1 batch.size=16384  
linger.ms=0 --topic Topic-01 --num-records 100000000 --record-size 1024 --throughput 102400
```

The result is as follows:

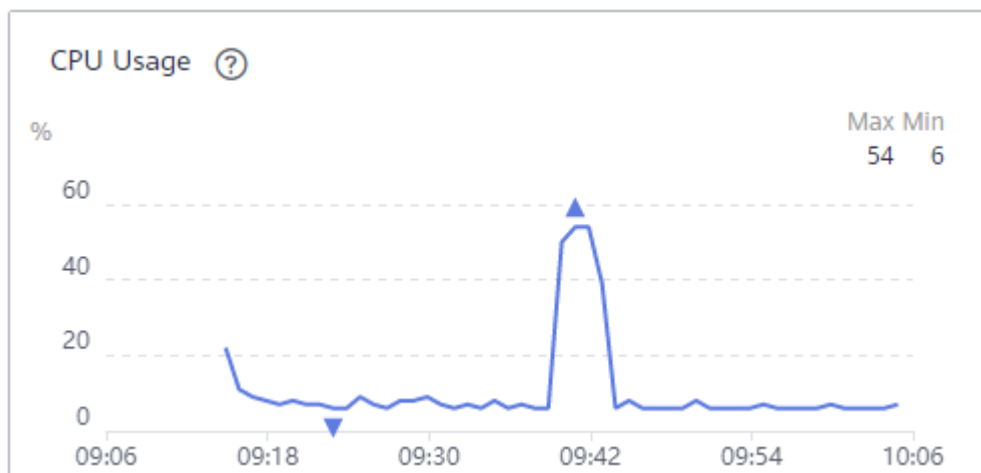
```
100000000 records sent, 102399.318430 records/sec (100.00 MB/sec), 4.62 ms avg latency, 751.00 ms max  
latency, 1 ms 50th, 3 ms 95th, 164 ms 99th, 406 ms 99.9th.
```

Message production rate: 102,399 records/second

**Step 2** Log in to the Kafka console, locate the row that contains the test instance, and click  to go to the Cloud Eye console.

**Step 3** On the **Brokers** tab page, view the CPU usage of the server nodes.

**Figure 1-1** broker-0 CPU usage (batch.size = 1 KB)



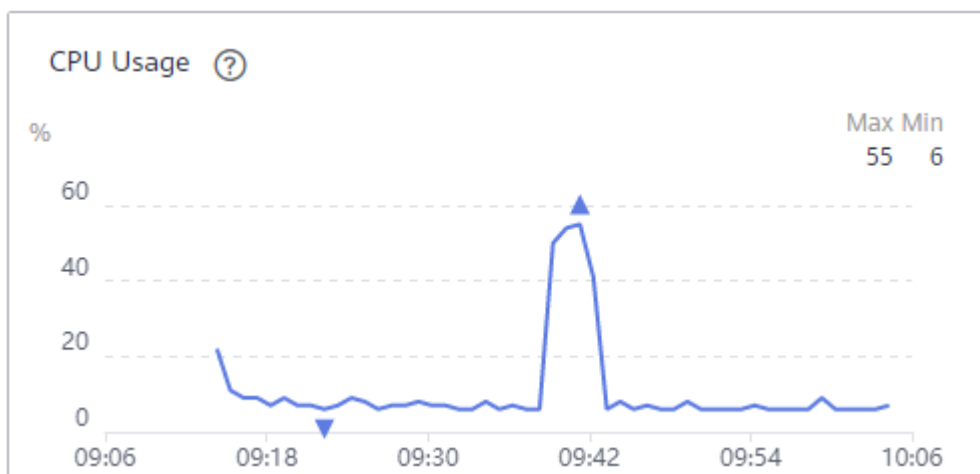
CPU usage: 54%

**Figure 1-2** broker-0 CPU usage (batch.size = 16 KB)



CPU usage: 26%

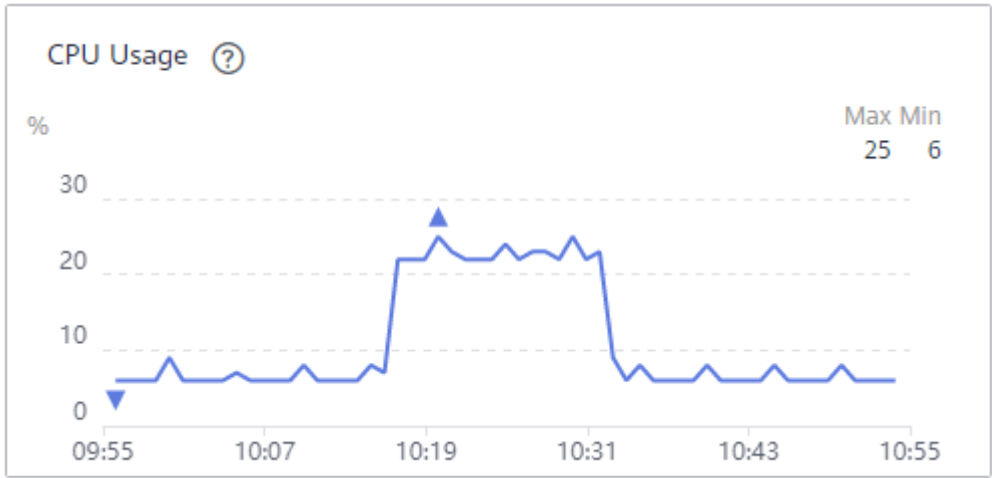
**Figure 1-3** broker-1 CPU usage (batch.size = 1 KB)



CPU usage: 55%

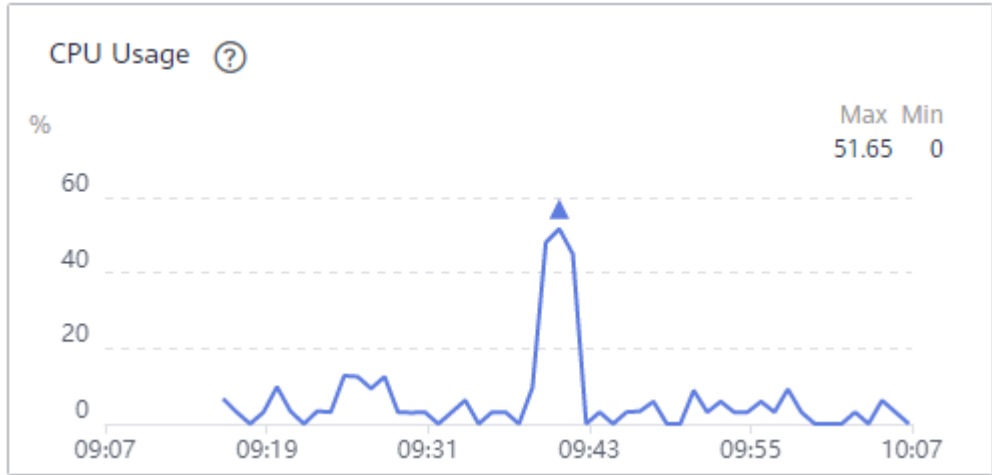


Figure 1-4 broker-1 CPU usage (batch.size = 16 KB)



CPU usage: 25%

Figure 1-5 broker-2 CPU usage (batch.size = 1 KB)



CPU usage: 51.65%

**Figure 1-6** broker-2 CPU usage (batch.size = 16 KB)



CPU usage: 36.45%

----End

## Test Scenario 2: Cross-AZ or Intra-AZ Production

**Step 1** Log in to the client server, go to the **kafka\_2.12-2.7.2/bin** directory, and run the following scripts.

**Configure the same AZ for the client and the instance**, and run the following script:

```
./kafka-producer-perf-test.sh --producer-props  
bootstrap.servers=192.168.0.69:9092,192.168.0.42:9092,192.168.0.66:9092 acks=1 batch.size=1024  
linger.ms=0 --topic Topic-01 --num-records 8000000 --record-size 1024 --throughput 102400
```

The result is as follows:

```
8000000 records sent, 37696.729809 records/sec (36.81 MB/sec), 796.54 ms avg latency, 3838.00 ms max  
latency, 322 ms 50th, 2282 ms 95th, 2745 ms 99th, 3593 ms 99.9th.
```

Message production rate: 37,697 records/second


**Configure different AZs for the client and the instance**, and run the following script:

```
./kafka-producer-perf-test.sh --producer-props  
bootstrap.servers=192.168.0.69:9092,192.168.0.42:9092,192.168.0.66:9092 acks=1 batch.size=1024  
linger.ms=0 --topic Topic-01 --num-records 4000000 --record-size 1024 --throughput 102400
```

The result is as follows:

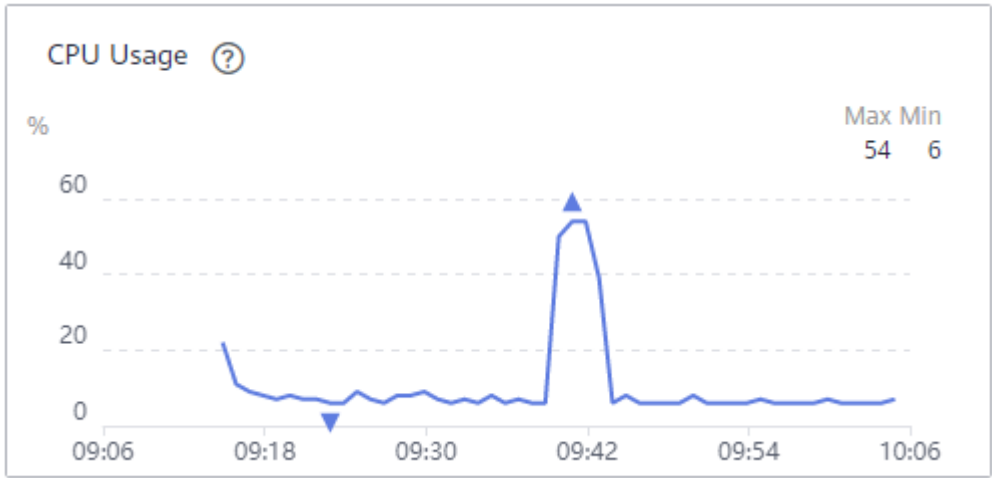
```
4000000 records sent, 15358.152107 records/sec (15.00 MB/sec), 1944.09 ms avg latency, 8179.00 ms max  
latency, 13 ms 50th, 6049 ms 95th, 6549 ms 99th, 8086 ms 99.9th.
```

Message production rate: 15,358 records/second

**Step 2** Log in to the Kafka console, locate the row that contains the test instance, and click  to go to the Cloud Eye console.

**Step 3** On the **Brokers** tab page, view the CPU usage of the server nodes.

Figure 1-7 broker-0 CPU usage (same AZ)



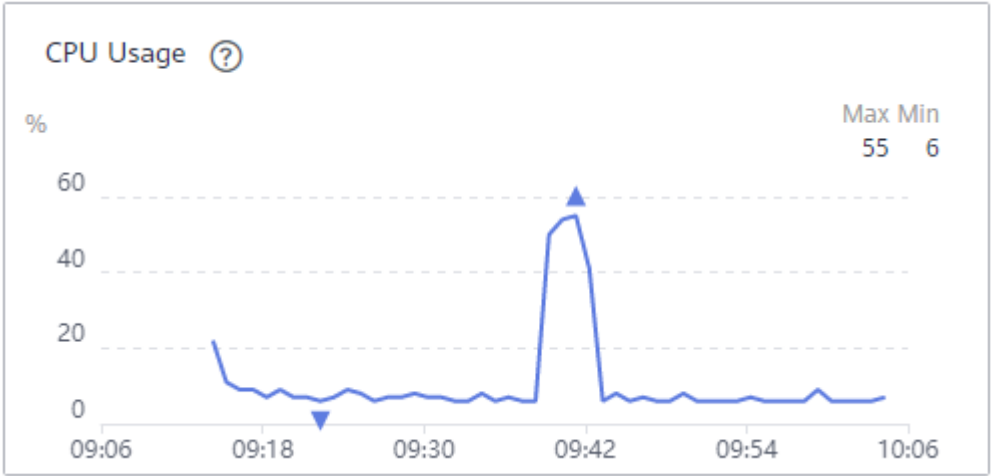
CPU usage: 54%

Figure 1-8 broker-0 CPU usage (different AZs)



CPU usage: 28%

Figure 1-9 broker-1 CPU usage (same AZ)



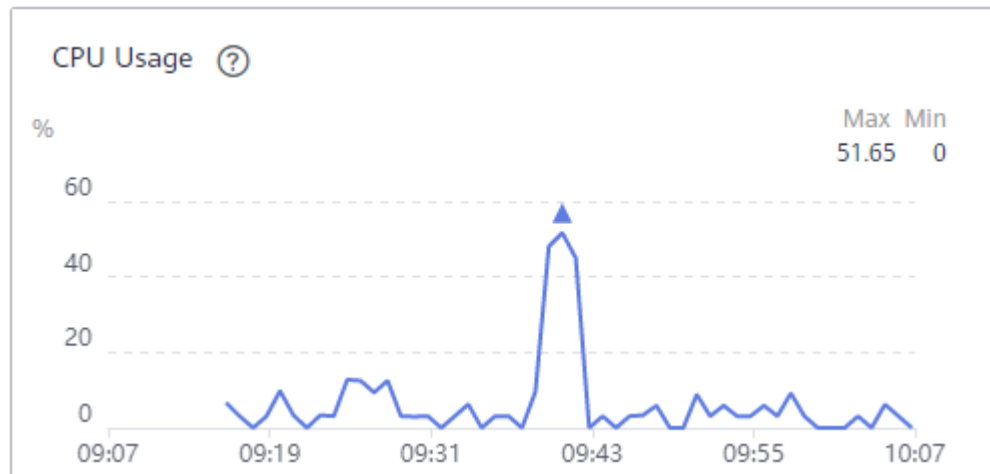
CPU usage: 55%

Figure 1-10 broker-1 CPU usage (different AZs)



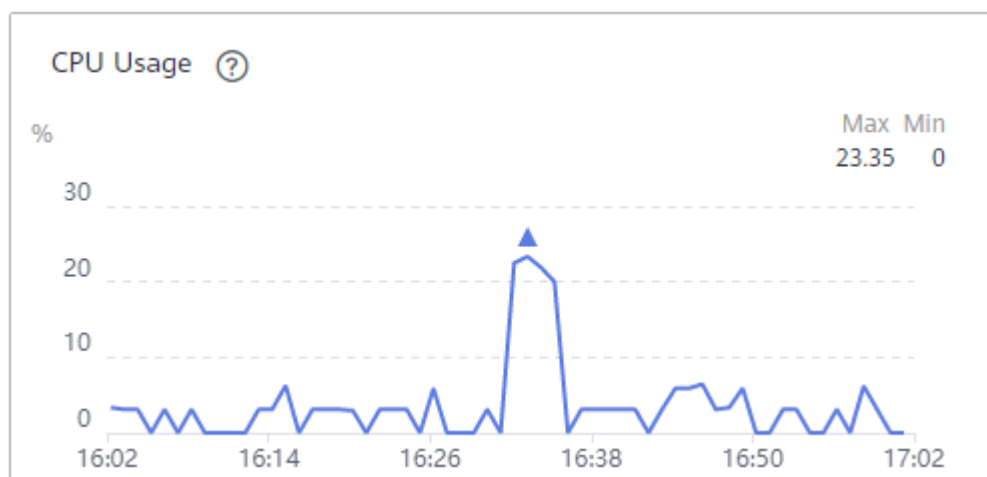
CPU usage: 28%

**Figure 1-11** broker-2 CPU usage (same AZ)



CPU usage: 51.65%

**Figure 1-12** broker-2 CPU usage (different AZs)



CPU usage: 23.35%

----End

### Test Scenario 3: Varied Numbers of Replicas

**Step 1** Log in to the client server, go to the **kafka\_2.12-2.7.2/bin** directory, and run the following scripts.

For the **one-replica** topic, run the following script:

```
./kafka-producer-perf-test.sh --producer-props  
bootstrap.servers=192.168.0.69:9092,192.168.0.42:9092,192.168.0.66:9092 acks=1 batch.size=1024  
linger.ms=0 --topic Topic-01 --num-records 8000000 --record-size 1024 --throughput 102400
```

The result is as follows:

```
8000000 records sent, 37696.729809 records/sec (36.81 MB/sec), 796.54 ms avg latency, 3838.00 ms max  
latency, 322 ms 50th, 2282 ms 95th, 2745 ms 99th, 3593 ms 99.9th.
```

Message production rate: 37,697 records/second


For the **three-replica** topic, run the following script:

```
./kafka-producer-perf-test.sh --producer-props  
bootstrap.servers=192.168.0.69:9092,192.168.0.42:9092,192.168.0.66:9092 acks=1 batch.size=1024  
linger.ms=0 --topic Topic-02 --num-records 4000000 --record-size 1024 --throughput 102400
```

The result is as follows:

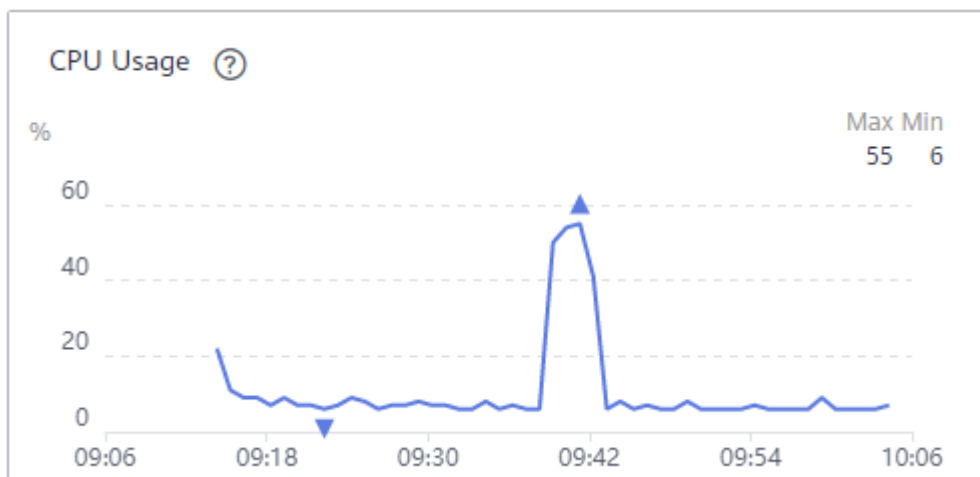
```
4000000 records sent, 15245.877896 records/sec (14.89 MB/sec), 1963.88 ms avg latency, 7471.00 ms max  
latency, 306 ms 50th, 5854 ms 95th, 6682 ms 99th, 7439 ms 99.9th.
```

Message production rate: 15,246 records/second

**Step 2** Log in to the Kafka console, locate the row that contains the test instance, and click  to go to the Cloud Eye console.

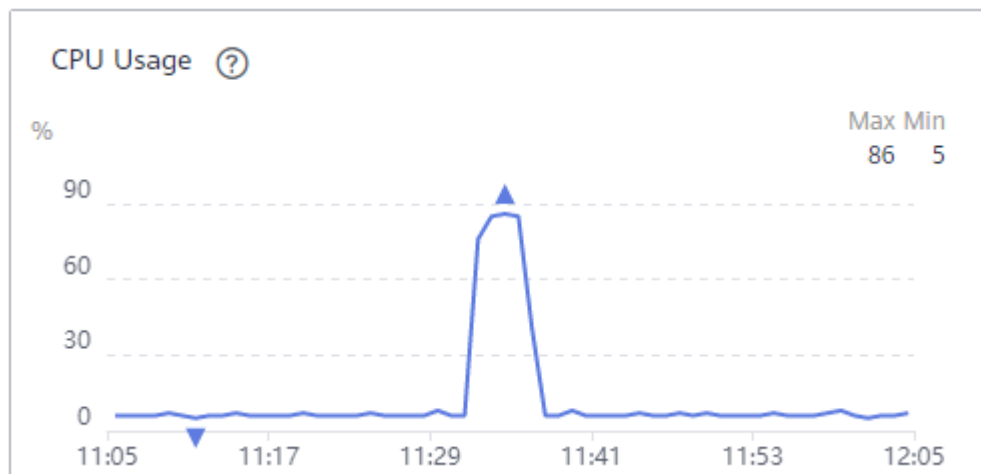
**Step 3** On the **Brokers** tab page, view the CPU usage of the server nodes.

**Figure 1-13** broker-0 CPU usage (one replica)



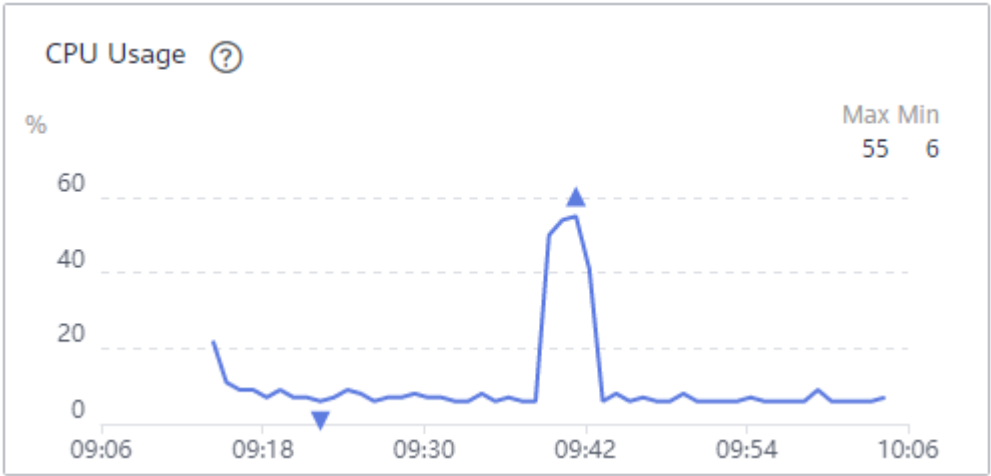
CPU usage: 54%

**Figure 1-14** broker-0 CPU usage (three replicas)



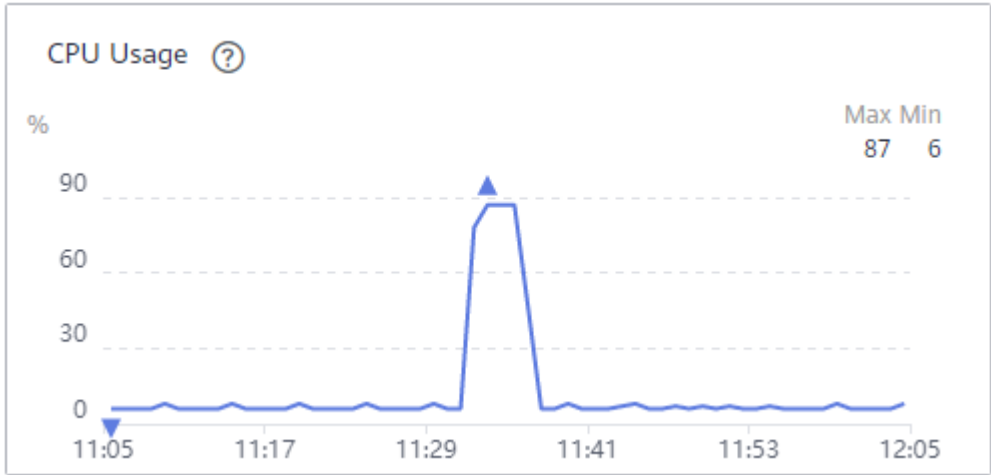
CPU usage: 86%

Figure 1-15 broker-1 CPU usage (one replica)



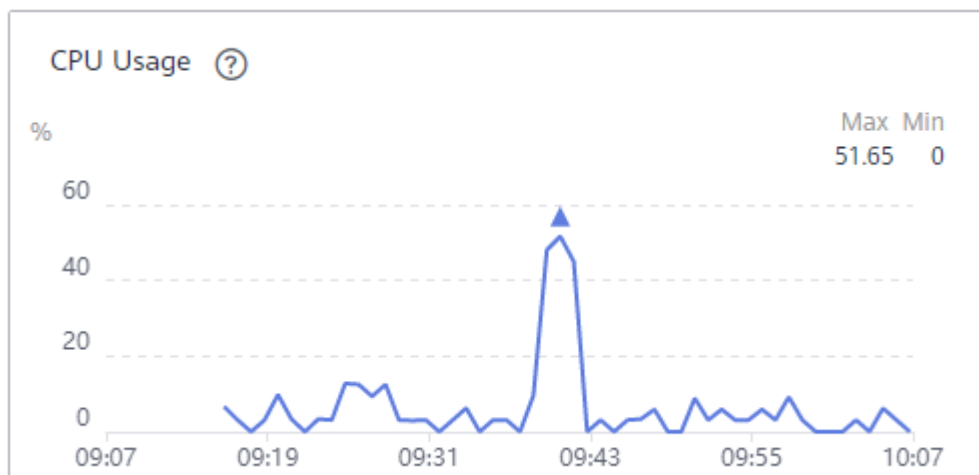
CPU usage: 55%

Figure 1-16 broker-1 CPU usage (three replicas)



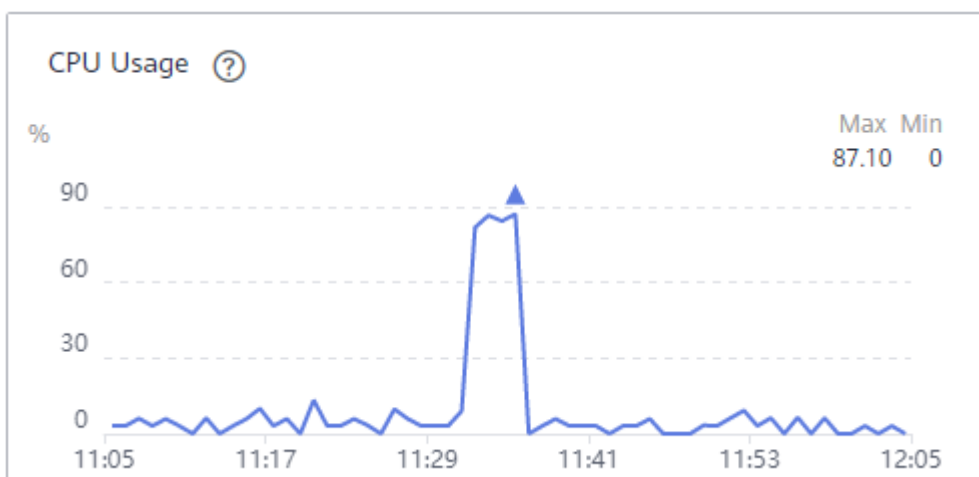
CPU usage: 87%

**Figure 1-17** broker-2 CPU usage (one replica)



CPU usage: 51.65%

**Figure 1-18** broker-2 CPU usage (three replicas)



CPU usage: 87.10%

----End

## Test Scenario 4: Synchronous/Asynchronous Replication

**Step 1** Log in to the client server, go to the **kafka\_2.12-2.7.2/bin** directory, and run the following scripts.

For **asynchronous replication**, run the following script:

```
./kafka-producer-perf-test.sh --producer-props
bootstrap.servers=192.168.0.69:9092,192.168.0.42:9092,192.168.0.66:9092 acks=1 batch.size=1024
linger.ms=0 --topic Topic-02 --num-records 4000000 --record-size 1024 --throughput 102400
```

The result is as follows:

```
4000000 records sent, 15245.877896 records/sec (14.89 MB/sec), 1963.88 ms avg latency, 7471.00 ms max
latency, 306 ms 50th, 5854 ms 95th, 6682 ms 99th, 7439 ms 99.9th.
```



Message production rate: 15,246 records/second


For **synchronous replication**, run the following script:

```
./kafka-producer-perf-test.sh --producer-props  
bootstrap.servers=192.168.0.69:9092,192.168.0.42:9092,192.168.0.66:9092 acks=-1 batch.size=1024  
linger.ms=0 --topic Topic-03 --num-records 1000000 --record-size 1024 --throughput 102400
```

The result is as follows:

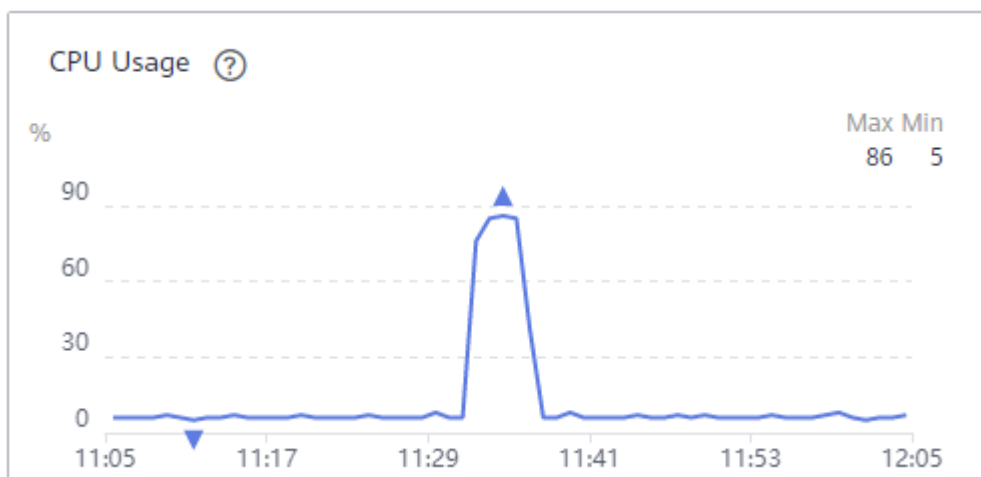
```
1000000 records sent, 5180.783438 records/sec (5.06 MB/sec), 5692.27 ms avg latency, 10312.00 ms max  
latency, 5579 ms 50th, 7538 ms 95th, 9481 ms 99th, 10219 ms 99.9th.
```

Message production rate: 5181 records/second

**Step 2** Log in to the Kafka console, locate the row that contains the test instance, and click  to go to the Cloud Eye console.

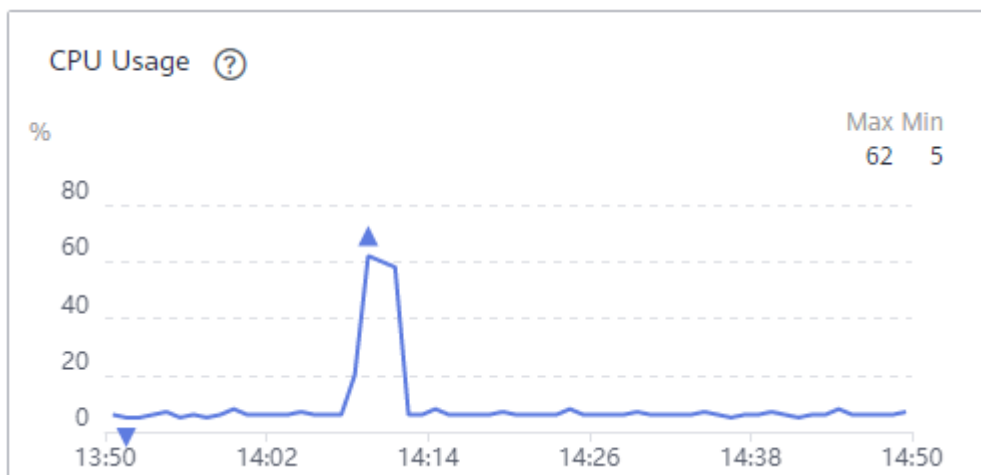
**Step 3** On the **Brokers** tab page, view the CPU usage of the server nodes.

**Figure 1-19** broker-0 CPU usage (asynchronous replication)



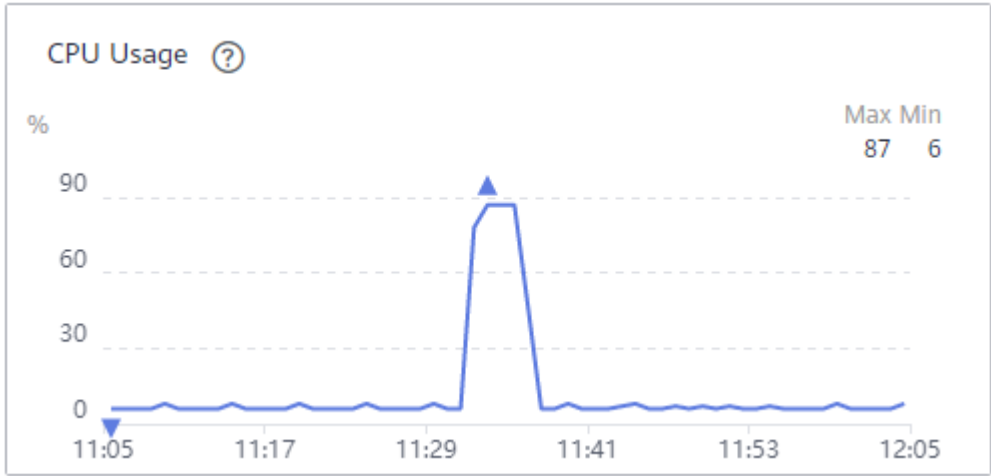
CPU usage: 86%

**Figure 1-20** broker-0 CPU usage (synchronous replication)



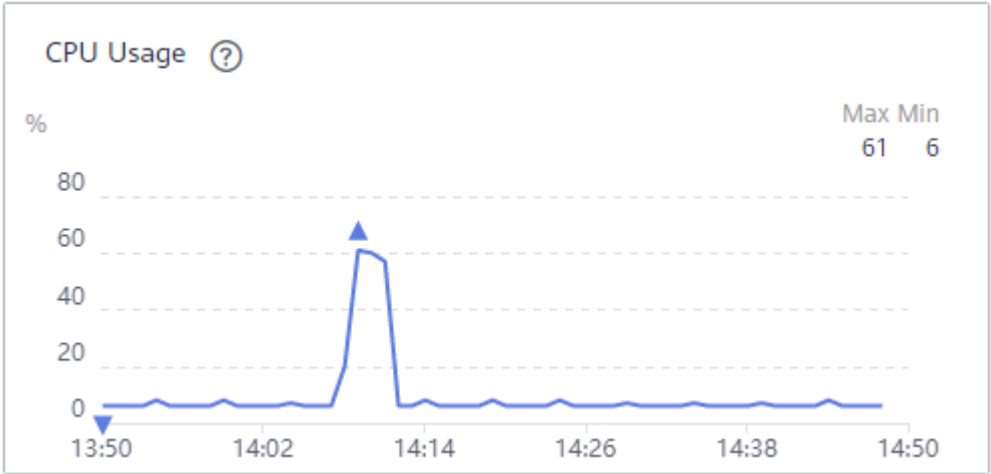
CPU usage: 62%

Figure 1-21 broker-1 CPU usage (asynchronous replication)



CPU usage: 87%

Figure 1-22 broker-1 CPU usage (synchronous replication)



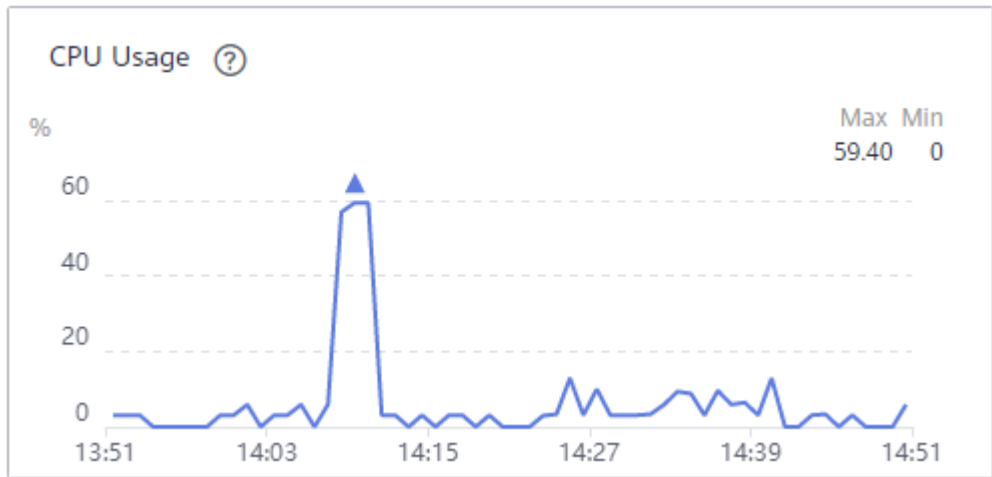
CPU usage: 61%

Figure 1-23 broker-2 CPU usage (asynchronous replication)



CPU usage: 87.10%

Figure 1-24 broker-2 CPU usage (synchronous replication)



CPU usage: 59.40%

-----End

## 1.4 Test Results

Table 1-2 Test results

Partitions	Replicas	Synchronous Replication Enabled	batch.size	Cross-AZ Production	Message Production Rate on the Client Side (Records/Second)	CPU Usage on the Server Side (broker-0)	CPU Usage on the Server Side (broker-1)	CPU Usage on the Server Side (broker-2)
3	1	No	1 KB	No	37,697	54%	55%	51.65%
3	1	No	16 KB	No	102,399	26%	25%	36.45%
3	1	No	1 KB	Yes	15,358	28%	28%	23.35%
3	3	Yes	1 KB	No	5181	62%	61%	59.40%
3	3	No	1 KB	No	15,246	86%	87%	87.10%

Based on the test results, the following conclusions are drawn (for reference only):

- When the **batch.size** of production requests is 16 times larger, the message production rate increases, and the CPU usage decreases.
- Compared with cross-AZ production, intra-AZ production significantly increases message production rate and CPU usage.
- When the number of replicas changes from 1 to 3, the message production rate decreases significantly, and the CPU usage increases.
- Compared with synchronous replication, asynchronous replication increases the message production rate and the CPU usage.

# 2 TPS Test for Instances of Different Specifications

## 2.1 Test Scenarios

TPS tests can be performed in the following scenarios:

- Test scenario 1 (whether SASL is enabled): same topic, different SASL settings
- Test scenario 2 (synchronous or asynchronous replication): same instance, topics with different replication settings
- Test scenario 3 (synchronous or asynchronous flushing): same instance, topics with different flushing settings
- Test scenario 4 (disk type): same topic, instances with different disk types
- Test scenario 5 (number of partitions): same instance, topics with different number of partitions

## 2.2 Test Environment

Perform the following steps to set up the test environment.

### Step 1: Buy Instances

Buy instances as shown in [Table 2-1](#). For details about how to buy an instance, see [Buying an Instance](#).

**Table 2-1** Instance parameters

Instance	Brokers	Flavor	SASL	Disk Type
kafka-01	3	kafka.2u4g.clust er	Enabled	Ultra-high I/O
kafka-02	3	kafka.4u8g.clust er	Enabled	Ultra-high I/O

Instance	Brokers	Flavor	SASL	Disk Type
kafka-03	3	kafka.8u16g.clus ter	Enabled	Ultra-high I/O
kafka-04	3	kafka.12u24g.clu ster	Enabled	Ultra-high I/O
kafka-05	3	kafka.16u32g.clu ster	Enabled	Ultra-high I/O
kafka-06	3	kafka.2u4g.cluste r	Disabled	Ultra-high I/O
kafka-07	3	kafka.4u8g.cluste r	Disabled	Ultra-high I/O
kafka-08	3	kafka.8u16g.clus ter	Disabled	Ultra-high I/O
kafka-09	3	kafka.12u24g.clu ster	Disabled	Ultra-high I/O
kafka-10	3	kafka.16u32g.clu ster	Disabled	Ultra-high I/O
kafka-11	3	kafka.2u4g.cluste r	Disabled	High I/O
kafka-12	3	kafka.4u8g.cluste r	Disabled	High I/O
kafka-13	3	kafka.8u16g.clus ter	Disabled	High I/O
kafka-14	3	kafka.12u24g.clu ster	Disabled	High I/O
kafka-15	3	kafka.16u32g.clu ster	Disabled	High I/O

## Step 2: Create Topics

Create topics as shown in [Table 2-2](#). For details about how to create a topic, see [Creating a Topic](#).

**Table 2-2** Topic parameters

Topic	Synchronous Replication	Synchronous Flushing	Replicas	Partitions
topic-01	Disabled	Disabled	3	30
topic-02	Enabled	Disabled	3	30

Topic	Synchronous Replication	Synchronous Flushing	Replicas	Partitions
topic-03	Disabled	Enabled	3	30
topic-04	Disabled	Disabled	3	3
topic-05	Disabled	Disabled	3	12
topic-06	Disabled	Disabled	3	100

### Step 3: Obtain the Testing Tool

Download [Kafka CLI v2.7.2](#).

### Step 4: Buy an ECS

Buy a Linux ECS (with the same region, AZ, VPC, subnet, and security group as the [Kafka instance](#)). For details about how to purchase an ECS, see [Purchasing an ECS](#).

Perform the following operations on the ECSs:

- Install [Java JDK](#) and configure the environment variables `JAVA_HOME` and `PATH`.  

```
export JAVA_HOME=/root/jdk1.8.0_231
export PATH=$JAVA_HOME/bin:$PATH
```
- Download [Kafka CLI v2.7.2](#) and decompress it.  

```
tar -zxf kafka_2.12-2.7.2.tgz
```

## 2.3 Test Results

### Test Command

```
./kafka-producer-perf-test.sh --producer-props bootstrap.servers=${connection address} acks=1
batch.size=16384 linger.ms=10 --topic ${topic name} --num-records 10000000 --record-size 1024 --
throughput -1 --producer.config ../config/producer.properties
```

- bootstrap.servers**: address of the Kafka instance obtained in [Step 1: Buy Instances](#).
- acks**: message synchronization policy. `acks=1` indicates asynchronous replication, and `acks=-1` indicates synchronous replication.
- batch.size**: size of messages sent in each batch, in bytes.
- linger.ms**: interval between two batches.
- topic**: topic name set in [Step 2: Create Topics](#).
- num-records**: total number of messages to be sent.
- record-size**: size of each message.
- throughput**: number of messages sent per second.

## Test Results

**Test scenario 1 (whether SASL is enabled): same topic (30 partitions, 3 replicas, asynchronous replication, and asynchronous flushing), instances with SASL enabled or disabled. The test result is as follows.**

**Table 2-3** Test result

Flavor	Disk Type	Brokers	TPS (SASL Enabled)	TPS (SASL Disabled)
kafka.2u4g.cluster	Ultra-high I/O	3	100,000	280,000
kafka.4u8g.cluster	Ultra-high I/O	3	170,000	500,000
kafka.8u16g.cluster	Ultra-high I/O	3	200,000	730,000
kafka.12u24g.cluster	Ultra-high I/O	3	320,000	790,000
kafka.16u32g.cluster	Ultra-high I/O	3	360,000	1,000,000

**Test scenario 2 (synchronous/asynchronous replication): same instance (ultra-high I/O, three brokers, SASL disabled), topics with different replication settings, and number of producer processes is three. The test result is as follows.**

**Table 2-4** Test result

Flavor	Synchronous Flushing	Replicas	Partitions	TPS (Synchronous Replication)	TPS (Asynchronous Replication)
kafka.2u4g.cluster	Disabled	3	30	100,000	292,000
kafka.4u8g.cluster	Disabled	3	30	230,000	496,000
kafka.8u16g.cluster	Disabled	3	30	342,000	766,000
kafka.12u24g.cluster	Disabled	3	30	383,000	802,000
kafka.16u32g.cluster	Disabled	3	30	485,000	1,080,000



**Test scenario 3 (synchronous/asynchronous replication flushing): same instance (ultra-high I/O, three brokers, SASL disabled), topics with different flushing settings. The test result is as follows.**

**Table 2-5** Test result

Flavor	Synchron ous Replicati on	Replicas	Partitions	TPS (Synchron ous Flushing)	TPS (Asynchro nous Flushing)
kafka.2u4g.clus ter	Disabled	3	30	30,000	280,000
kafka.4u8g.clus ter	Disabled	3	30	32,500	500,000
kafka.8u16g.clu ster	Disabled	3	30	36,100	730,000
kafka.12u24g.cl uster	Disabled	3	30	37,400	790,000
kafka.16u32g.cl uster	Disabled	3	30	40,400	1,000,000

**Test scenario 4 (different disk types): same topic (30 partitions, 3 replicas, asynchronous replication, and asynchronous flushing) with different disk types. The test result is as follows.**

**Table 2-6** Test result

Flavor	Brokers	SASL	TPS (High I/O)	TPS (Ultra-High I/O)
kafka.2u4g.clust er	3	Disabled	110,000	250,000
kafka.4u8g.clust er	3	Disabled	135,000	380,000
kafka.8u16g.clus ter	3	Disabled	213,000	480,000
kafka.12u24g.clu ster	3	Disabled	240,000	577,000
kafka.16u32g.clu ster	3	Disabled	280,000	840,000

**Test scenario 5 (different numbers of partitions): same instance (ultra-high I/O, three brokers, SASL disabled), topics with different number of partitions. The test result is as follows.**

**Table 2-7** Test result

Flavor	Synchr onous Flushi ng	Synchr onous Replic ation	Replic as	TPS (3 Partitions )	TPS (12 Partitions )	TPS (100 Partitions )
kafka.2u4g.cl uster	Disabl ed	Disabl ed	3	250,000	260,000	250,000
kafka.4u8g.cl uster	Disabl ed	Disabl ed	3	330,000	280,000	260,000
kafka.8u16g.cl uster	Disabl ed	Disabl ed	3	480,000	410,000	340,000
kafka.12u24g. cluster	Disabl ed	Disabl ed	3	570,000	750,000	520,000
kafka.16u32g. cluster	Disabl ed	Disabl ed	3	840,000	1,000,000	630,000