

Kafka Monitoring & Tuning

Important Metrics

Performance tuning involves two important metrics:

- Latency measures how long it takes to process one event,
- Throughput measures how many events arrive within a specific amount of time.

Most systems are optimized for either latency or throughput. Kafka is balanced for both. A well-tuned Kafka system has just enough brokers to handle topic throughput, given the latency required to process information as it is received.

Tuning your producers, brokers, and consumers to send, process, and receive the largest possible batches within a manageable amount of time results in the best balance of latency and throughput for your Kafka cluster.

Perfomance Metrics - Producer

```
kafka-producer-perf-test.sh --topic my-perf-test \  
--num-records 100000 \  
--record-size 1024 \  
--throughput -1 \  
--producer-props acks=1 \  
bootstrap.servers=localhost:9092
```

```
[root@kafka0 code]# kafka-producer-perf-test.sh --topic my-perf-test \  
> --num-records 100000 \  
> --record-size 1024 \  
> --throughput -1 \  
> --producer-props acks=1 \  
> bootstrap.servers=localhost:9092  
7666 records sent, 1531.4 records/sec (1.50 MB/sec), 1013.3 ms avg latency, 1876.0 ms max latency.  
12750 records sent, 2545.4 records/sec (2.49 MB/sec), 3116.2 ms avg latency, 5239.0 ms max latency.  
8175 records sent, 1632.4 records/sec (1.59 MB/sec), 6957.0 ms avg latency, 9480.0 ms max latency.  
8775 records sent, 1754.6 records/sec (1.71 MB/sec), 11696.2 ms avg latency, 13566.0 ms max latency.  
14655 records sent, 2931.0 records/sec (2.86 MB/sec), 14802.3 ms avg latency, 15516.0 ms max latency.  
14670 records sent, 2927.6 records/sec (2.86 MB/sec), 12559.2 ms avg latency, 14597.0 ms max latency.  
17295 records sent, 3456.9 records/sec (3.38 MB/sec), 10034.4 ms avg latency, 10787.0 ms max latency.  
12270 records sent, 2412.0 records/sec (2.36 MB/sec), 9613.4 ms avg latency, 10595.0 ms max latency.  
100000 records sent, 2386.179250 records/sec (2.33 MB/sec), 9400.79 ms avg latency, 15516.00 ms max latency, 10062 ms 50th, 15252 ms  
95th, 15469 ms 99th, 15502 ms 99.9th.  
[root@kafka0 code]# █
```

Perfomance Metrics - Consumer

```
#kafka-consumer-perf-test.sh --topic my-perf-test --broker-list kafka0:9092 --messages 100000
```

```
--version          Display Kafka version.  
[root@kafka0 code]# kafka-consumer-perf-test.sh --topic my-perf-test --broker-list kafka0:9092 --messages 100000  
start.time, end.time, data.consumed.in.MB, MB.sec, data.consumed.in.nMsg, nMsg.sec, rebalance.time.ms, fetch.time.ms, fetch.MB.sec,  
fetch.nMsg.sec  
2022-02-28 15:43:59:610, 2022-02-28 15:44:06:745, 98.1396, 13.7547, 100495, 14084.7933, 1933, 5202, 18.8658, 19318.5313  
[root@kafka0 code]#
```

Perfomance Tuning - Producer & Consumer

Increase the no of partitions

Acks – Recommended 2 or 3

Buffer setting – Set as per the batch size

ISR – Set the minimum as per the requirement – 2 or 3 is the optimal

Two parameters are particularly important for latency and throughput:
batch size and linger time

```
[root@kafka0 code]# kafka-producer-perf-test.sh --topic my-perf-test4 --num-records 100000 --record-size 1024 --throughput -1 --producer-props acks=1 bootstrap.servers=kafka0:9092
11946 records sent, 2373.1 records/sec (2.32 MB/sec), 521.9 ms avg latency, 1748.0 ms max latency.
19260 records sent, 3843.5 records/sec (3.75 MB/sec), 2231.3 ms avg latency, 4188.0 ms max latency.
20556 records sent, 4111.2 records/sec (4.01 MB/sec), 5283.8 ms avg latency, 6877.0 ms max latency.
24264 records sent, 4848.9 records/sec (4.74 MB/sec), 6761.5 ms avg latency, 8210.0 ms max latency.
100000 records sent, 4004.324671 records/sec (3.91 MB/sec), 4737.75 ms avg latency, 8210.00 ms max latency, 5609 ms 50th, 7438 ms 95th, 7888 ms 99th, 8152 ms 99.9th.
[root@kafka0 code]#
```

```
[root@kafka0 code]# kafka-producer-perf-test.sh --topic my-perf-test4 --num-records 100000 --record-size 1024 --throughput -1 --producer-props acks=1 batch.size=96384 bootstrap.servers=kafka0:9092
38240 records sent, 7648.0 records/sec (7.47 MB/sec), 69.3 ms avg latency, 1565.0 ms max latency.
100000 records sent, 10260.619741 records/sec (10.02 MB/sec), 74.23 ms avg latency, 1565.00 ms max latency, 68 ms 50th, 140 ms 95th, 175 ms 99th, 204 ms 99.9th.
[root@kafka0 code]#
```

Perfomance Tuning - Producer & Consumer

The maximum number of consumers in a consumer group for a topic is equal to the number of partitions

```
--version          Display Kafka version.  
[root@kafka0 code]# kafka-consumer-perf-test.sh --topic my-perf-test --broker-list kafka0:9092 --messages 100000  
start.time, end.time, data.consumed.in.MB, MB.sec, data.consumed.in.nMsg, nMsg.sec, rebalance.time.ms, fetch.time.ms, fetch.MB.sec,  
fetch.nMsg.sec  
2022-02-28 15:43:59:610, 2022-02-28 15:44:06:745, 98.1396, 13.7547, 100495, 14084.7933, 1933, 5202, 18.8658, 19318.5313  
[root@kafka0 code]#
```

2 consumer group

```
[root@kafka0 my-kafka-0]# kafka-consumer-perf-test.sh --topic my-perf-test4 --broker-list kafka0:9092 --messages 100000 --group cg  
start.time, end.time, data.consumed.in.MB, MB.sec, data.consumed.in.nMsg, nMsg.sec, rebalance.time.ms, fetch.time.ms, fetch.MB.sec,  
fetch.nMsg.sec  
2022-02-28 16:51:53:378, 2022-02-28 16:52:06:294, 97.6973, 7.5640, 100042, 7745.5869, 3336, 9580, 10.1980, 10442.7975  
[root@kafka0 my-kafka-0]#
```

Perfomance Tuning - Broker

- Leader balanced properly
- It is recommends starting with one partition per physical storage disk and one consumer per partition

```
:~/sync/repticas=2 --bootstrap-server kafka0:9092optc my-perf-test4 --partitions 12 --config retention.ms=86400000 --config min  
[root@kafka0 code]# kafka-topics.sh bootstrap-server kafka0:9092 --describe --topic my-perf-test4  
Topic: my-perf-test4    TopicId: xBrpQBcWQfaDtHf4ZXoxDQ PartitionCount: 12      ReplicationFactor: 1      Configs: segment.bytes=10737  
41824,retention.ms=86400000  
    Topic: my-perf-test4    Partition: 0      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 1      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 2      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 3      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 4      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 5      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 6      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 7      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 8      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 9      Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 10     Leader: 0      Replicas: 0      Isr: 0  
    Topic: my-perf-test4    Partition: 11     Leader: 0      Replicas: 0      Isr: 0  
[root@kafka0 code]#
```

Perfomance Tuning - MirrorMaker

num.streams parameter controls the number of consumer threads in Mirror Maker.
kafka-producer-perf-test can be used to generate load on the source cluster.

You can test and measure performance of Mirror Maker with
different **num.streams** values (start from 1 and increase it gradually).

Kafka Broker Metrics via JConsole 1 of 2

Connection Window Help

pid: 79602 kafka.Kafka /Users/rick/kafka-training/lab10/solution/config/server-2.properties

Overview Memory Threads Classes VM Summary MBeans

Attribute value

Name	Value
Count	308725

Refresh

MBeanAttributeInfo

Name	Value
Attribute:	
Name	Count
Description	Attribute exposed for management
Readable	true
Writable	false
Is	false
Type	long

Descriptor

Name	Value
Count	

JMImplementation
com.sun.management
java.lang
java.nio
java.util.logging
kafka
kafka.cluster
kafka.controller
kafka.coordinator
kafka.log
kafka.network
kafka.server
BrokerTopicMetrics
DelayedFetchMetrics
DelayedOperationPurgatory
Fetch
FetcherLagMetrics
FetcherStats
BytesPerSec
ReplicaFetcherThread-1
10.0.0.115
9092
Attributes
Count

Kafka Broker JConsole Metrics 2 of 2

pid: 79602 kafka.Kafka /Users/rick/kafka-training/lab10/solution/config/server-2.prc

Overview Memory Threads Classes VM Summary MBeans

Attribute value

Name	Value
Value	1374

Refresh

MBeanAttributeInfo

Name	Value
Attribute:	
Name	
Description	Attribute exposed for management
Readable	true
Writable	false
Is	false
Type	java.lang.Object

Descriptor

Name	Value
Value	

Value

Kafka Consumer JConsole Metrics

pid: 8131 com.intellij.rt.execution.application.AppMain com.cloudurable.kafka.consumer.ConsumerBlueMain

Overview Memory Threads Classes VM Summary MBeans

JMImplementation
com.sun.management
java.lang
java.nio
java.util.logging
kafka.consumer
app-info
consumer-coordinator-metrics
blue
blue-0
Attributes
join-time-max
commit-latency-avg
sync-time-avg
join-rate
assigned-partitions
sync-rate
commit-rate

Attribute value

Name	Value
commit-rate	0.7556252098958917

Refresh

MBeanAttributeInfo

Name	Value
Attribute:	
Name	commit-rate
Description	The number of commit calls per second
Readable	true
Writable	false
Is	false
Type	double

W - MB/sec of data that will be written

R - Replication factor

C - Number of consumer groups, that is the number of readers for each write

L - all consumers are lagging all the time

Based on this, we can calculate our cluster-wide I/O requirements:

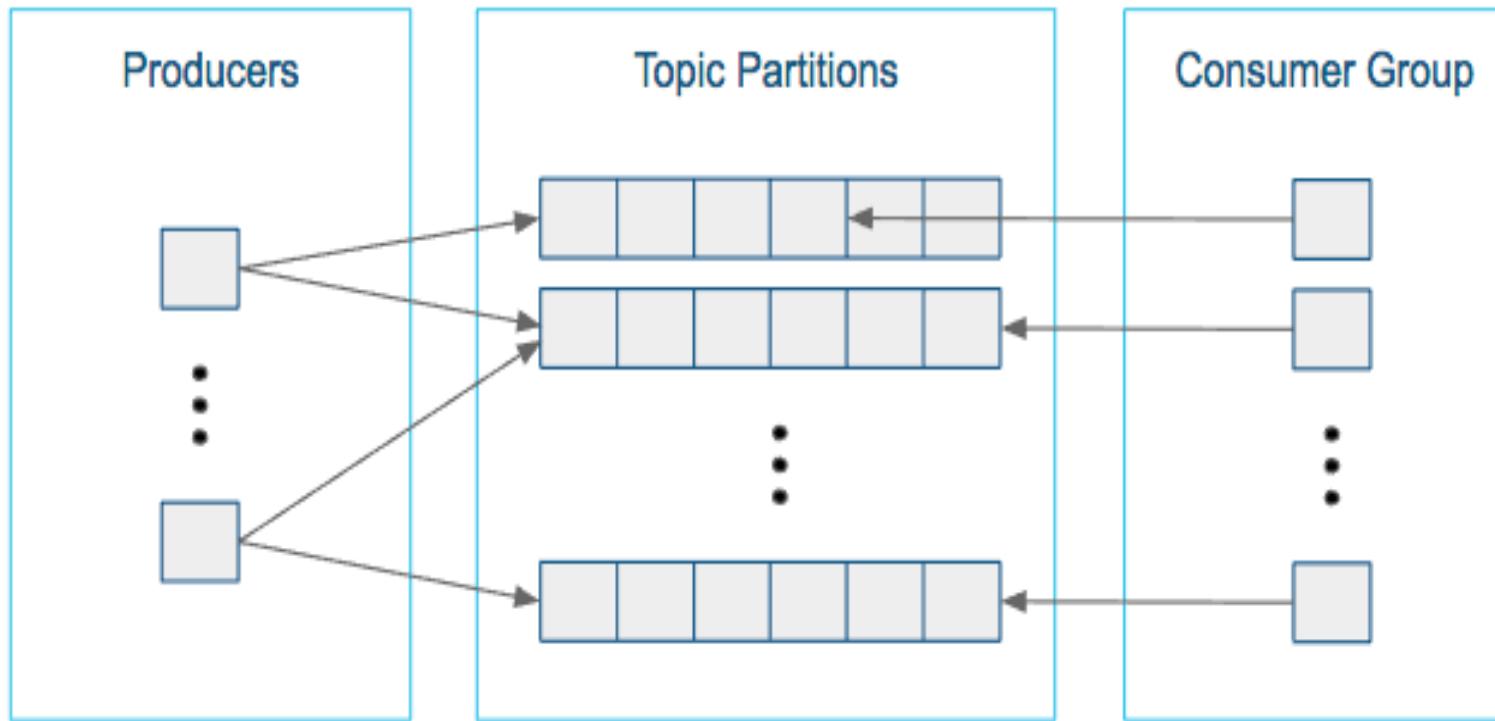
- Disk Throughput (Write + Read): $W * R + L * W$
- Network Read Throughput: $(R + C - 1) * W$
- Network Write Throughput: $W * R$

For example, if you have a 1 Gigabit Ethernet card with full duplex, then that would give 125 MB/sec read and 125 MB/sec write; likewise 6 7200 SATA drives might give roughly 300 MB/sec read + write throughput. Once we know the total requirements, as well as what is provided by one machine, you can divide to get the total number of machines needed

Choosing the Number of Partitions for a Topic

Tos

is the key to achieving a high degree of parallelism with respect to writes to and reads and to distribute load



For example, if you want to be able to read 1 GB/sec, but your consumer is only able process 50 MB/sec, then you need at least 20 partitions and 20 consumers in the consumer group. Similarly, if you want to achieve the same for producers, and 1 producer can only write at 100 MB/sec, you need 10 partitions.

Choosing the Number of Partitions for a Topic

Tos

So a simple formula could be:

$$\text{#Partitions} = \max(N_p, N_c)$$

where:

- N_p is the number of required producers determined by calculating: T_T/T_p
- N_c is the number of required consumers determined by calculating: T_T/T_c
- T_T is the total expected throughput for our system
- T_p is the max throughput of a single producer to a single partition
- T_c is the max throughput of a single consumer from a single partition

As guideline for optimal performance, you should not have more than 3000 partitions per broker and not more than 30,000 partitions in a cluster

Kafka Performance Broker Configuration

Tos

JVM and Garbage Collection

```
-server -XX:+UseG1GC -XX:MaxGCPauseMillis=20  
-XX:InitiatingHeapOccupancyPercent=35 -XX:+DisableExplicitGC  
-Djava.awt.headless=true -Djava.net.preferIPv4Stack=true
```

Network and I/O Threads

num.network.threads : Set this value mainly based on number of producers, consumers and replica fetchers.

queued.max.requests: controls how many requests are allowed in the request queue before blocking network threads.

num.io.threads: specifies the number of threads that a broker uses for processing requests from the request queue (might include disk I/O).

Java GC config

Xms 6g Xmx 6g

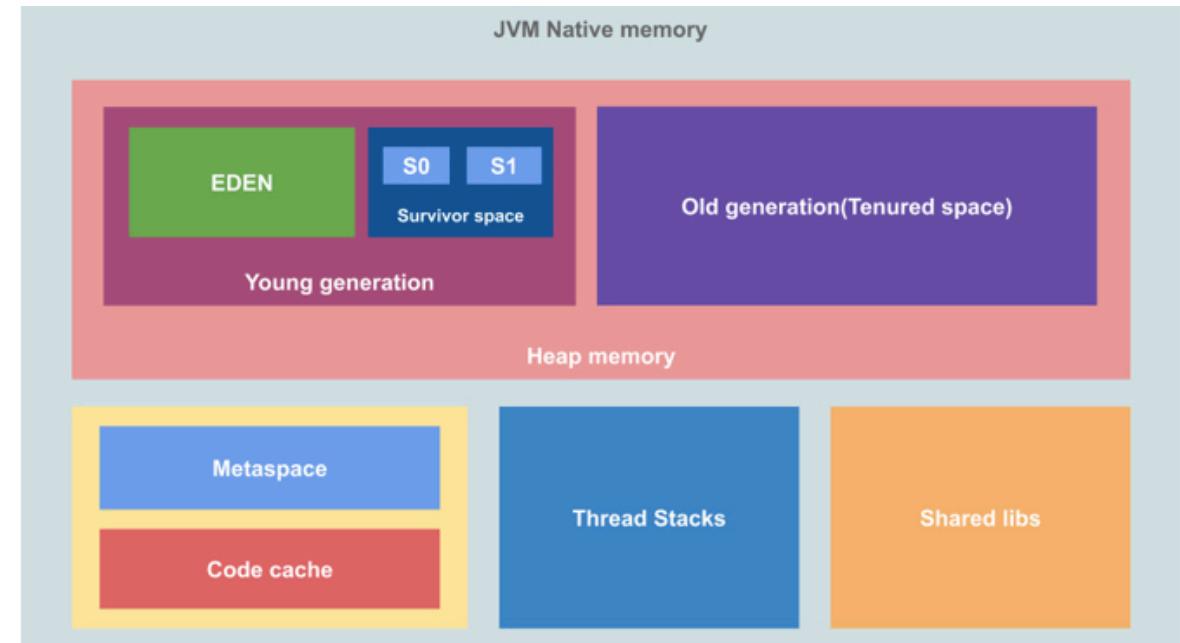
-XX:MetaspaceSize=96m

-XX:+UseG1GC

-XX:MaxGCPauseMillis=20

-XX:InitiatingHeapOccupancyPercent=35

-XX:G1HeapRegionSize=16M



- ❖ Heap Space should be 25% to 35% of available space for server
- ❖ Leave 50% for OS, Remember Kafka uses OS page cache
- ❖ Other tweaks for GC to limit overhead

Lab : Monitoring, Performance & Tuning