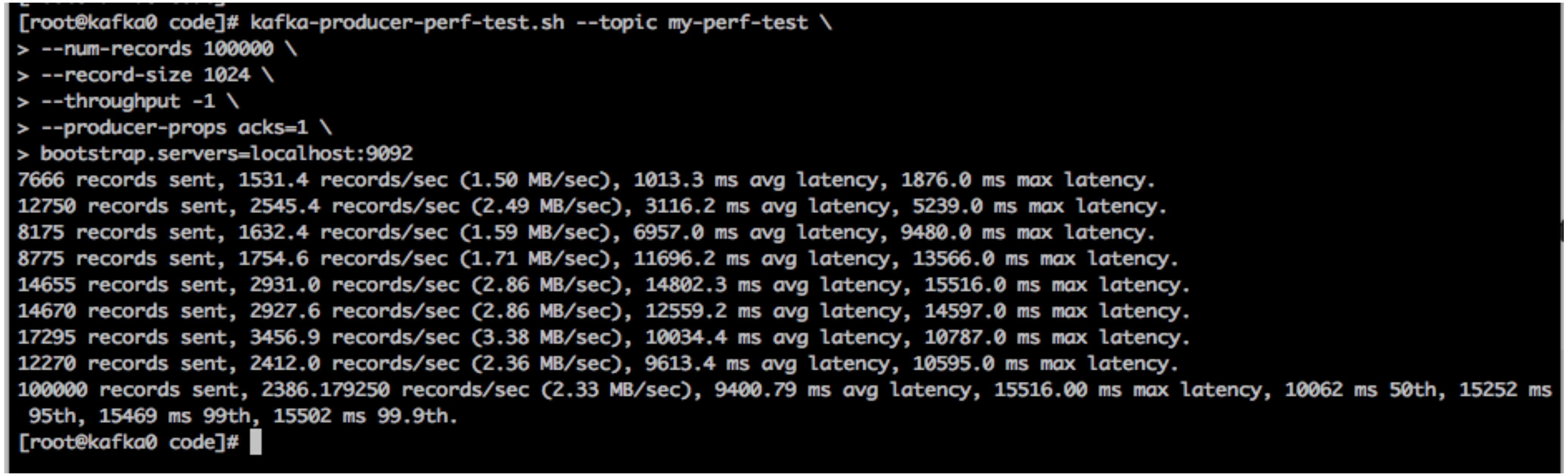
[illegible]

Kafka Monitoring & Tuning

Perfomance Metrics - Producer

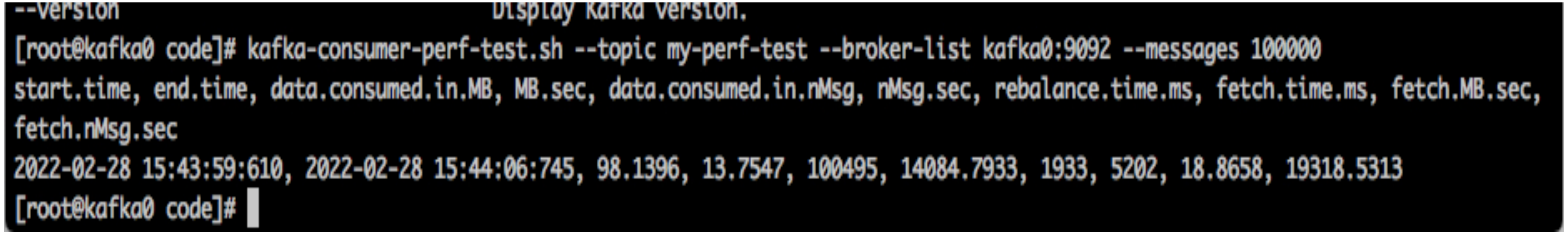
```
kafka-producer-perf-test.sh --topic my-perf-test \
--num-records 100000 \
--record-size 1024 \
--throughput -1 \
--producer-props acks=1 \
bootstrap.servers=localhost:9092
```



```
[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

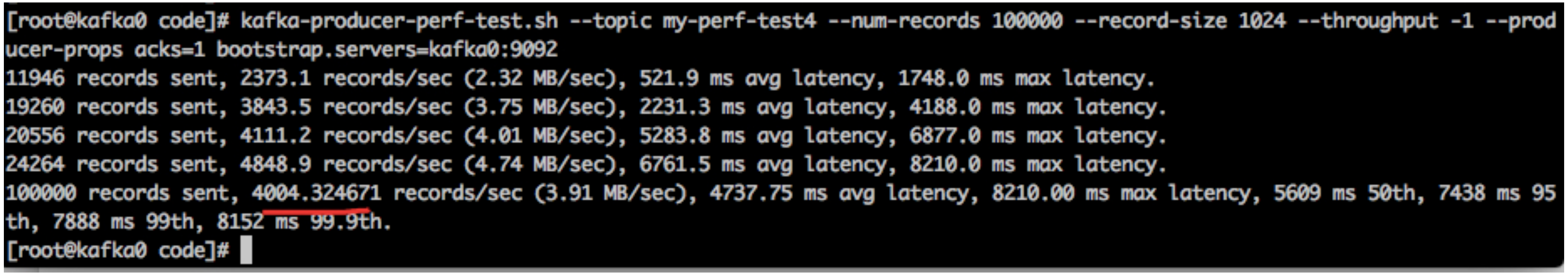
#kafka-consumer-perf-test.sh --topic my-perf-test --broker-list kafka0:9092 --messages 100000



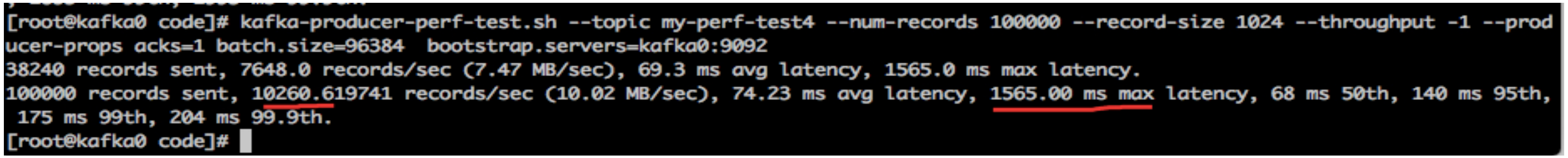
```
--version                               Display Kafka version.
[root@kafka0 code]# kafka-consumer-perf-test.sh --topic my-perf-test --broker-list kafka0:9092 --messages 100000
start.time, end.time, data.consumed.in.MB, MB.sec, data.consumed.in.nMsg, nMsg.sec, rebalance.time.ms, fetch.time.ms, fetch.MB.sec,
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



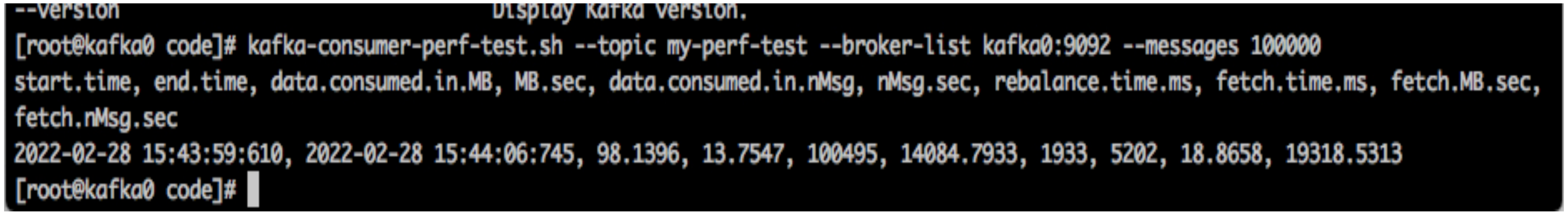
```
[root@kafka0 code]# kafka-producer-perf-test.sh --topic my-perf-test4 --num-records 100000 --record-size 1024 --throughput -1 --prod
ucer-props acks=1 bootstrap.servers=kafka0:9092
11946 records sent, 2373.1 records/sec (2.32 MB/sec), 521.9 ms avg latency, 1748.0 ms max latency.
19260 records sent, 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 95
th, 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 --prod
ucer-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]#
```

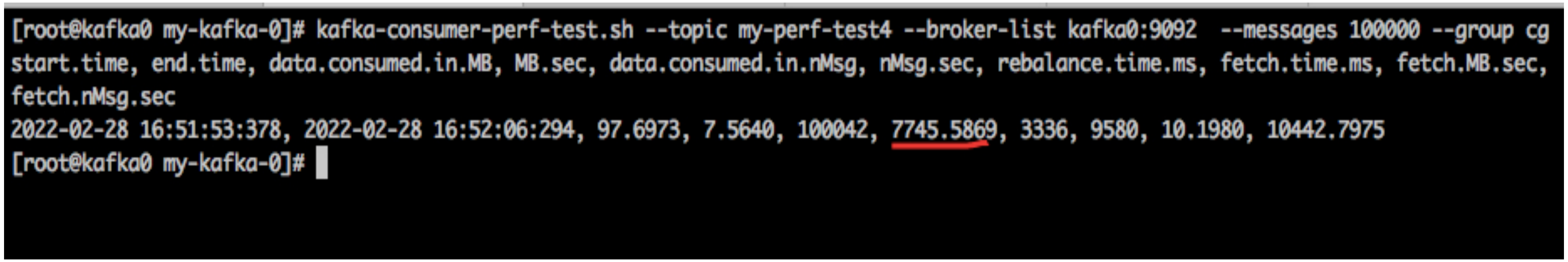
Perfomance Tuning - Producer & Consumer

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]#
```

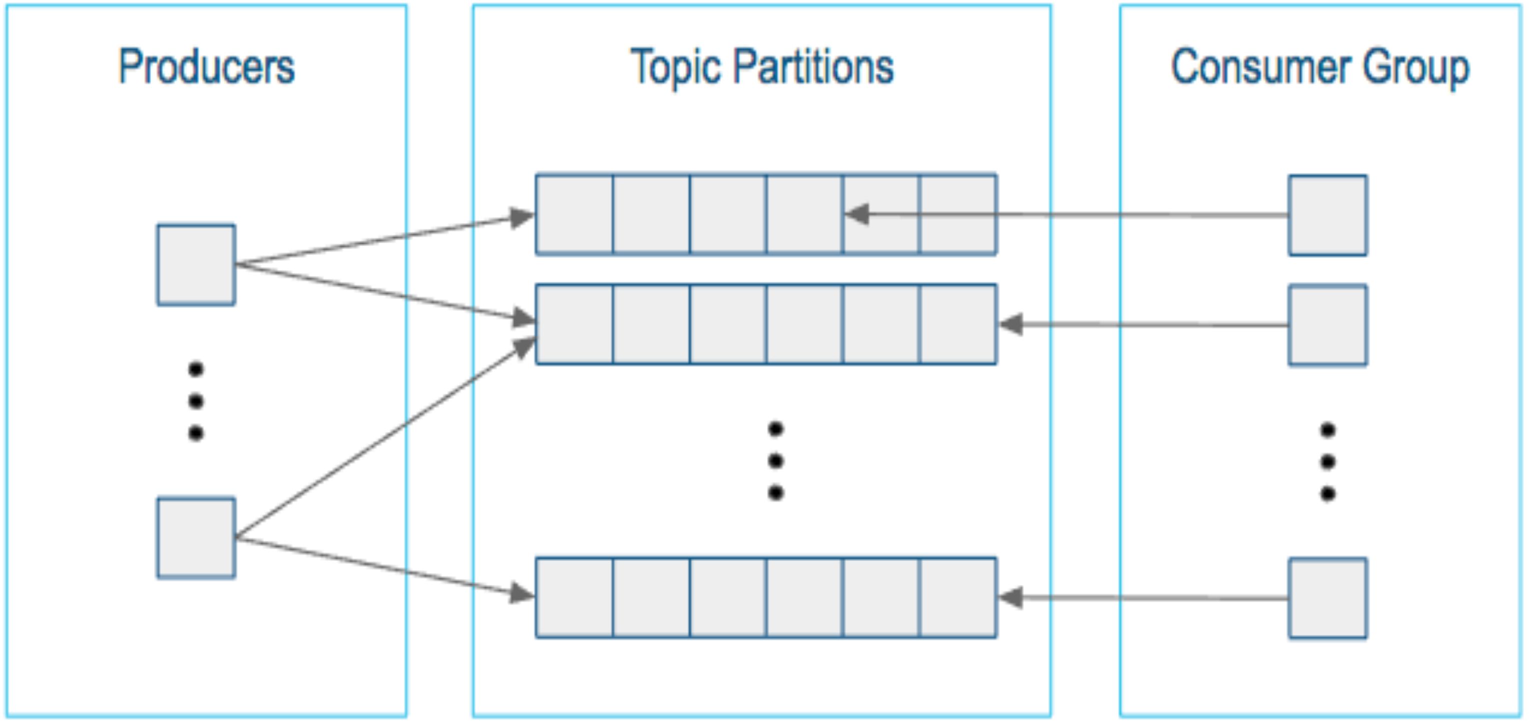
2 consumer group



```
[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]#
```

Choosing the Number of Partitions for a Topic

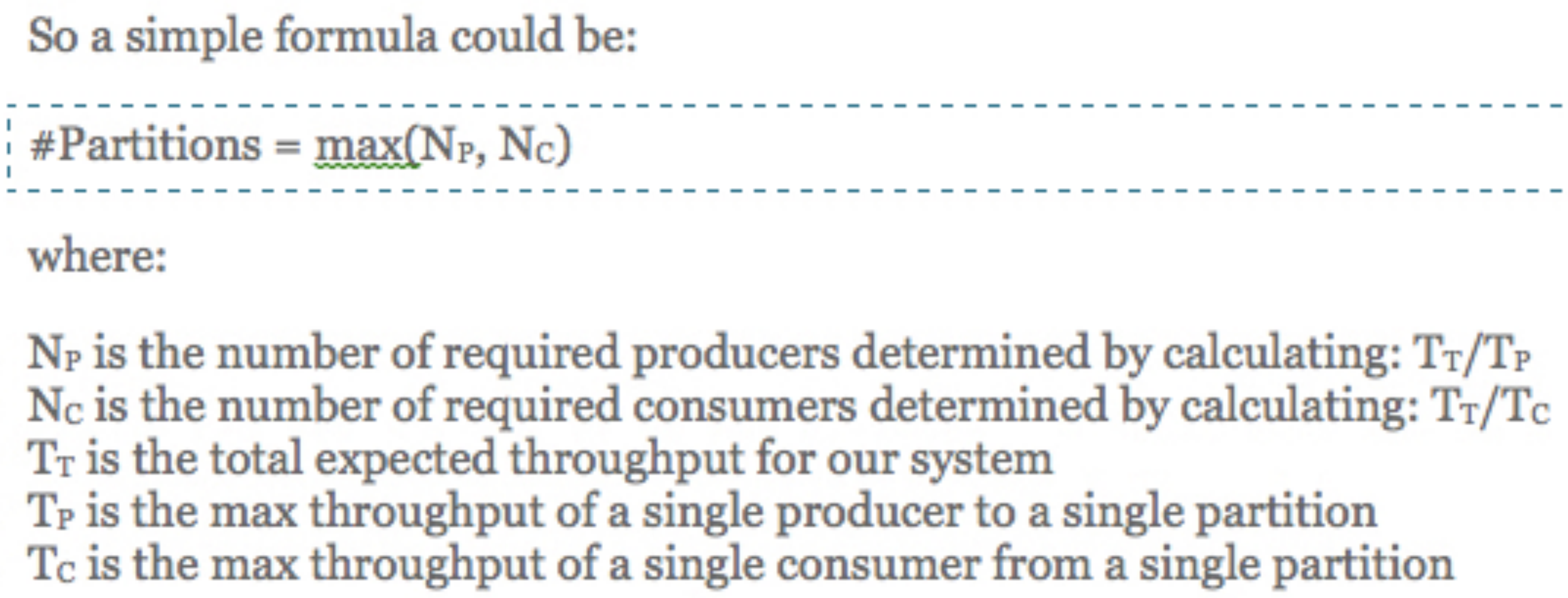
is the key to achieving a high degree of parallelism with respect to writes to and reads and to distribute load



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:

#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

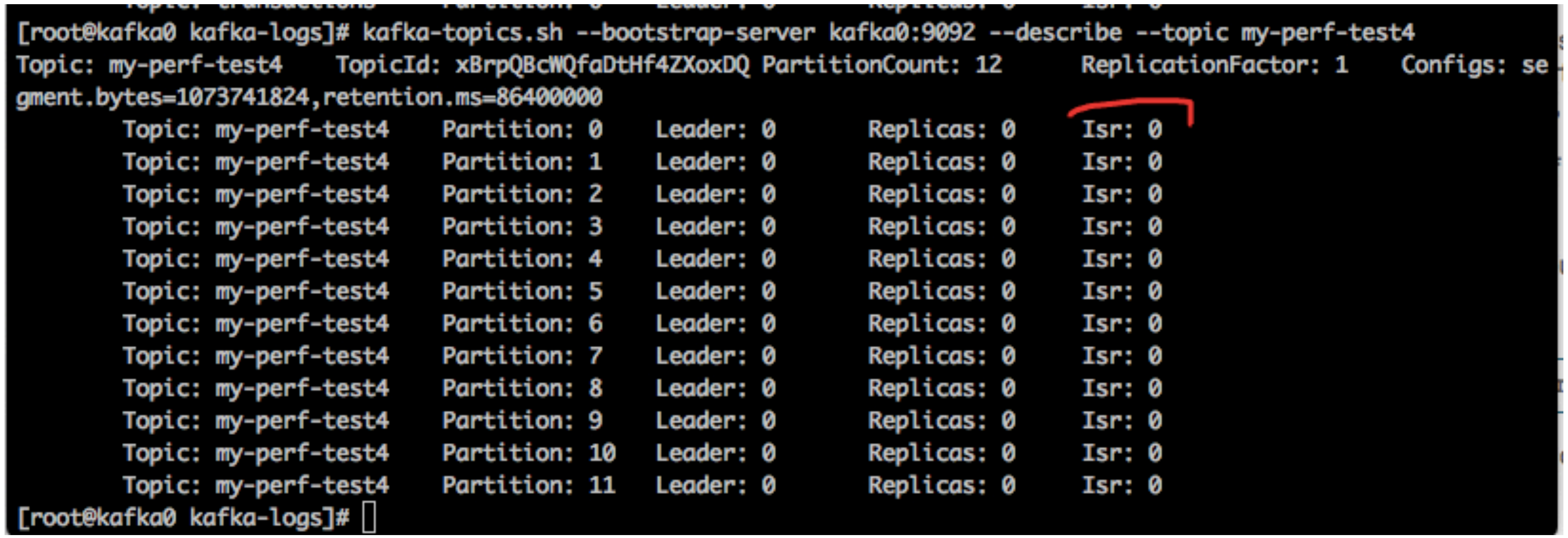
As guideline for optimal performance, you should not have more than 3000 partitions per broker and not more than 30,000 partitions in a cluster

Kafka Performance Broker Configuration

Tos

ISR Management

•**num.replica.fetchers** These threads are responsible for replicating messages from a source broker (that is, where partition leader resides). Increasing this value results in higher I/O parallelism and fetcher throughput. Of course, there is a trade-off: brokers use more CPU and network.
•**replica.fetch.min.bytes** controls the minimum number of bytes to fetch from a follower replica. If there is not enough bytes, wait up to replica.fetch.wait.max.ms.
•**replica.fetch.wait.max.m**s controls how long to sleep before checking for new messages from a fetcher replica. This value should be less than **replica.lag.time.max.ms**, otherwise the replica is kicked out of the ISR set.



```
[root@kafka0 kafka-logs]# kafka-topics.sh --bootstrap-server kafka0:9092 --describe --topic my-perf-test4
Topic: my-perf-test4    TopicId: xBrpQBcWQfaDtHf4ZXoxDQ PartitionCount: 12      ReplicationFactor: 1    Configs: se
gment.bytes=1073741824,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 kafka-logs]# 
```