

Goal of this evaluation:

- 1) Validate quota functionality, i.e. `throughput_per_broker_clientId <= quota` ?
- 2) Evaluate the performance improvement for “slow” and “medium” client in the presence of “fast” clients w.r.t. throughput and latency?

Metrics we use in the evaluation results:

- 1) end-to-end latency at clients
- 2) byte rate at clients

In all the experiments below, we use new producer to produce data, and use old consumer to consume data. And to best understand the performance impact on producer and consumer independently, we always run consumer after producer stops, i.e. producer and consumer won't compete for network or quota resource.

Experiments:**1. Broker throughput validation with artificially generated traffic (single broker)**

Configuration: The test is run on a desktop with one broker, one producer performance configured with `topic=test record-size=10000 --throughput=100000`, and one console consumer which reads from topic “test” at maximum possible throughput. Consumer always runs after producer stops. Bytes-in and bytes-out rates are collected using one minute average after the values stabilize.

Observation:

- 1) Unlimited quota. Broker's bytes-in and bytes-out rates are 85 MBps and 250 MBps.
- 2) 1 MBps quota for both producer and consumer. Broker's bytes-in and bytes-out rates are 0.95 MBps and 0.98 MBps.
- 3) 10 MBps quota for both producer and consumer. Broker's bytes-in and bytes-out rates are 9.8 MBps and 9.9 MBps.
- 4) 50 MBps quota for both producer and consumer. Broker's bytes-in and bytes-out rates are 49 MBps and 49 MBps.

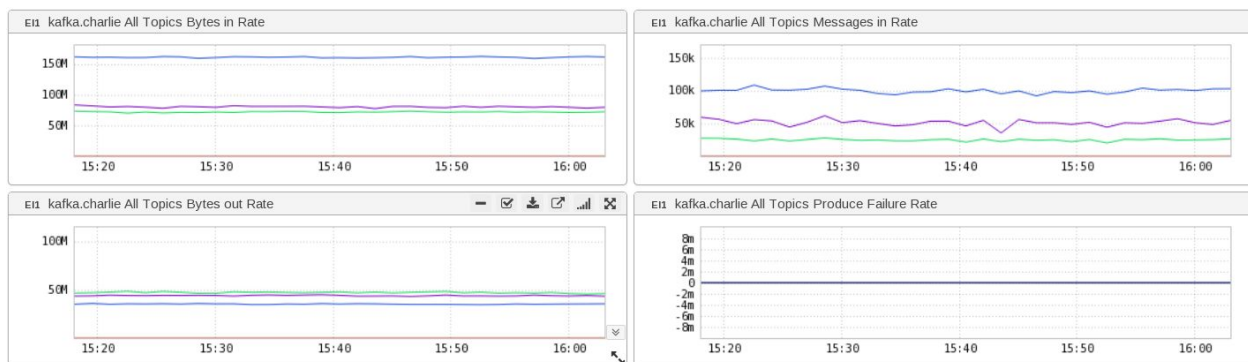
This simple experiment setup suggests that quota is functioning well as expected. With unlimited quota, the produce and consume byte rate is much higher than 50 MBps. When broker is configured with various quotas of at most 50 MBps, the actual produce/consume byte rate is always very close to the configured quota.

2. Broker throughput validation with artificially generated traffic (multiple brokers)

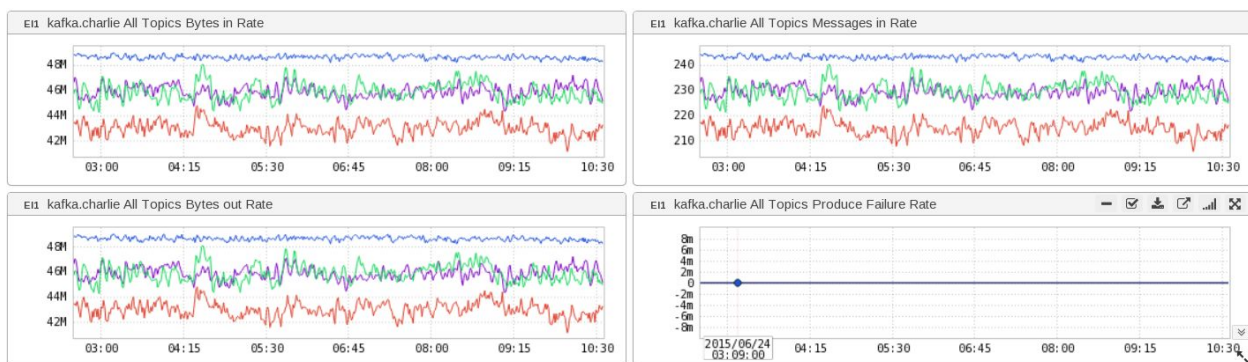
Configuration: 4 quota-enabled brokers are running in a test cluster. 4 producer performance, 1 on each of the 4 brokers, configured with topic=test, compression-codec=0, message-size=200000, threads=3, and new-producer=true. No consumer.

Observation:

- When default quota per clientId is 1 GBps, the host with most traffic has byte-in-rate around 160 MBps.
This measures the throughput of artificial traffic when quota is effectively unlimited.

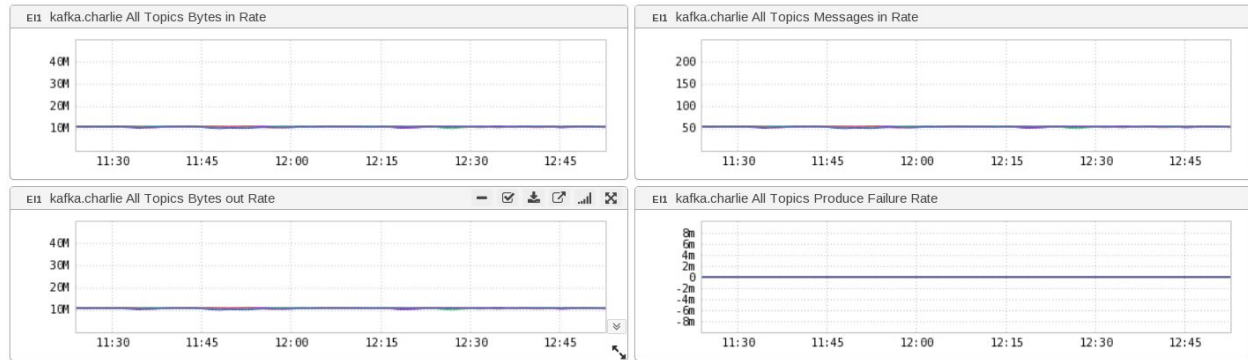


- When default quota per clientId is 50 MBps, the host with most traffic has byte-in-rate around 50 MBps.



- When default quota per clientId is 10 MBps, the host with most traffic has byte-in-rate around 10 MBps.

From experiment 1 and 2 we make the following observation. *Quota functionality is working properly -- the throughput is lower than quota limit imposed by quota. And throughput is generally close to the quota limit, if the throughput without quota enforcement is higher than quota limit.*



3. Producer-only performance (with slow and fast producers)

Configuration: Same broker setting as before. The “slow” producer is configured with `topic=test`, `compression-codec=0`, `message-size=200000`, `threads=1`, `message-send-gap-ms=100`, and `new-producer=true`. The “fast” producers are configured with `topic=test`, `compression-codec=0`, `message-size=200000`, `threads=1`, and `new-producer=true`. Topic `replication-factor=2` and `partition=128`.

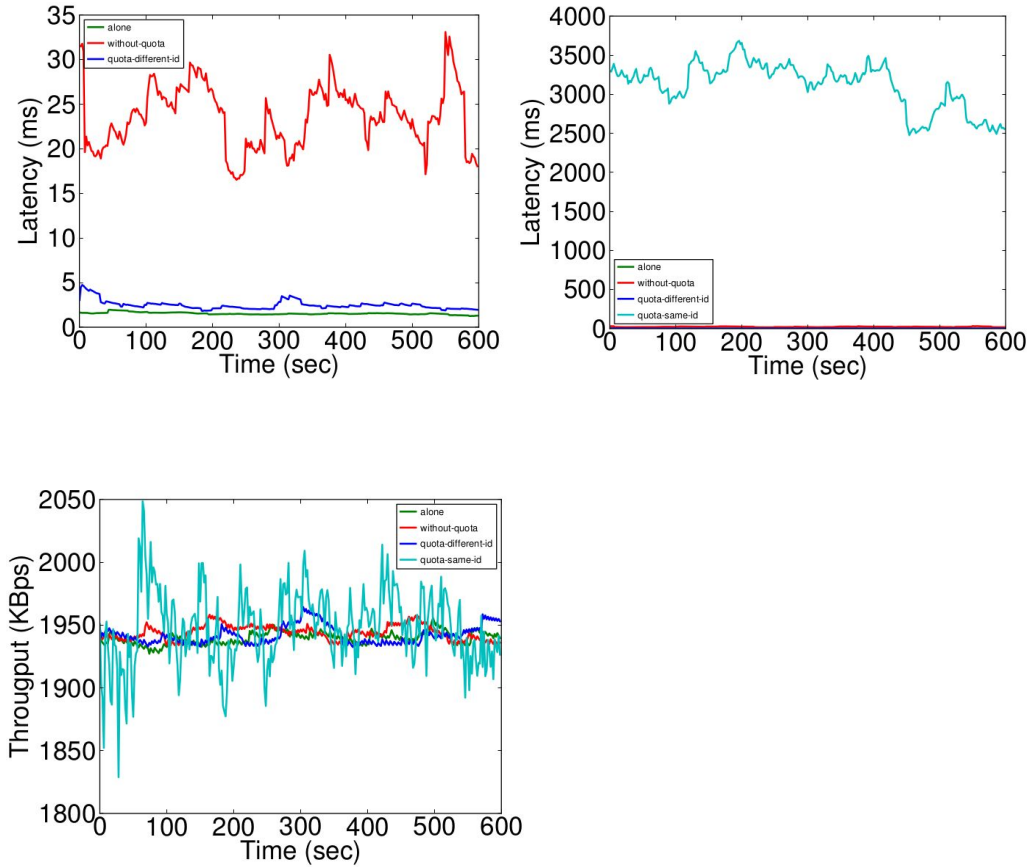
scenario 1 (*alone*): Slow producer runs alone at 2MBps without quota.

scenario 2 (*without_quota*): Slow producer runs with many fast producers without quota

scenario 3 (*quota_with_different_id*): Slow producer runs with many fast producers with 10MBps quota. Fast producers share the same `clientId` that is different from slow producer’s `clientId`.

scenario 4 (*quota_with_same_id*): Slow producer runs with many fast producers with 10MBps quota. They share the same `clientId`

Observation:



The 3 figures above demonstrate the throughput and latency of the slow producer, for each of the 4 scenarios. The table below shows the average throughput and latency of the slow producer.

	<i>alone</i>	<i>quota_with_different_id</i>	<i>without_quota</i>	<i>quota_with_same_id</i>
Latency (ms)	1.5	2.5	23.6	3115
Throughput (MBps)	~2	~2	~2	~2

We make the following observation. When slow producer is running alone, the latency is smallest. As fast producers join and competes with slow producer for resource at broker, the slower producer gets hit in performance as its latency increases from 1.5 ms to 23.6 ms. By having quota where slow and fast producers use different clientId, we can reduce the performance impact of fast producers on slow producer, as its latency decreases from 23.6 ms to 2.5 ms. *This demonstrates that quota is effective in improving latency of slow producer in the presence fast producers.*

On the other hand, the presence of fast producers doesn't have observable impact on slow producers. Therefore we would expect to see performance improvement of slow producer when quota is enabled.

4. Producer-only performance (with medium and fast producers)

Configuration: Same broker setting as before. The “medium” producer is configured with topic=test, compression-codec=0, message-size=200000, threads=1, message-send-gap-ms=3, and new-producer=true. The “fast” producers are configured with topic=test, compression-codec=0, message-size=200000, threads=1, and new-producer=true. Topic replication-factor=2 and partition=128.

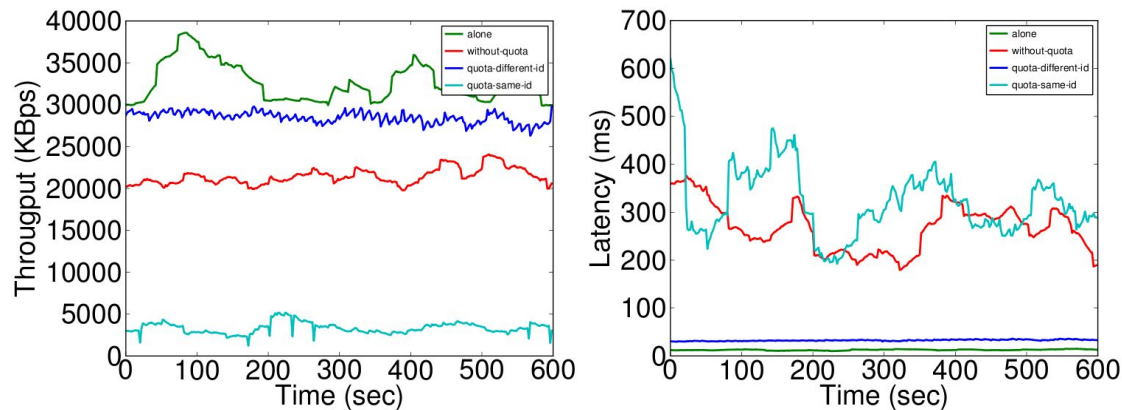
scenario 1 (*alone*): Medium producer runs alone at 32MBps without quota.

scenario 2 (*without_quota*): Medium producer runs with many fast producers without quota.

scenario 3 (*quota_with_different_id*): Medium producer runs with many fast producers with 15MBps quota. Fast producers share the same clientID that is different from medium producer's clientID.

scenario 4 (*quota_with_same_id*): Medium producer runs with many fast producers with 15MBps quota. They share the same clientID

Observation:



The 2 figures above demonstrate the throughput and latency of the slow producer, for each of the 4 scenarios. The table below shows the average throughput and latency of the slow producer.

	<i>alone</i>	<i>quota_with_different_id</i>	<i>without_quota</i>	<i>quota_with_same_id</i>
Latency (ms)	12.6	32.6	268.7	323

Throughput (MBps)	32	28	21	3.3
-------------------	----	----	----	-----

We make the following observation. When medium producer is running alone, the latency is smallest. As fast producers join and competes with medium producer for resource at broker, the medium producer gets hit in performance as its latency increases from 12.6 ms to 268.7 ms. By having quota where medium and fast producers use different clientId, we can reduce the performance impact of fast producers on medium producer, as its latency decreases from 268.7 ms to 32.6 ms. *This demonstrates that quota is effective in improving latency of medium producer in the presence fast producers.*

On the other hand, we find that quota also effective improves the throughput of medium producer in the presence of fast producers. When medium producer is running alone, the throughput is highest. As fast producers join and competes with medium producer for resource at broker, the medium producer gets hit in performance as its throughput decreases from 32 MBps to 21 MBps. By having quota where medium and fast producers use different clientId, we can reduce the performance impact of fast producers on medium producer, as its throughput increases from 21 MBps to 28 MBps. *This demonstrates that quota is effective in improving throughput of medium producer in the presence of fast producers.*

5. Consumer-only performance during bootstrap

Configuration: We use only one broker here. The topic is configured with compression-codec=0, message-size=200000, and replication-factor=1. All clients and broker run on different machines.

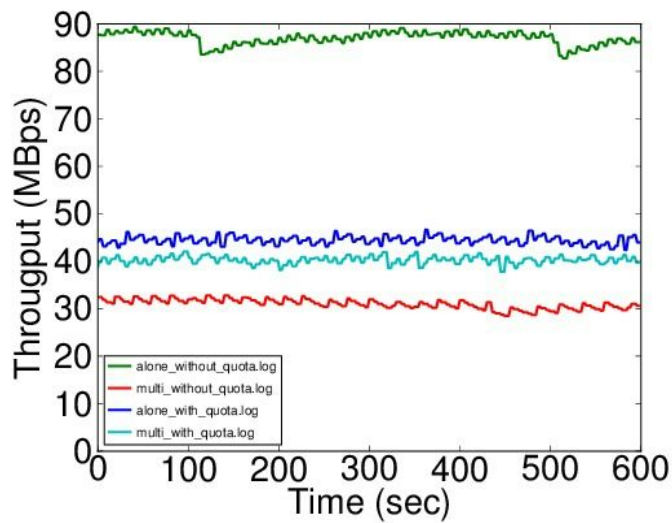
scenario 1 (*alone_without_quota*): start 1 console-consumer without quota

scenario 2 (*multi_without_quota*): start 3 console-consumers without quota on 3 different machines. Two of them share the same clientId. We record the read throughput of the other consumer.

scenario 3 (*alone_with_quota*): start 1 console-consumer with quota=50MB per clientId

scenario 4 (*multi_with_quota*): start 3 console-consumers with quota=50MB per clientId. Two of them share the same clientId. We record the read throughput of the other consumer.

Observation:



The figure above demonstrates the throughput of the monitored consumer, for each of the 4 scenarios. The table below shows the average throughput of the monitored consumer.

	<i>alone_without_quota</i>	<i>alone_with_quota</i>	<i>multi_with_quota</i>	<i>multi_without_quota</i>
Throughput (MBps)	87	44.3	40.3	31

We make the following observations. When the consumer runs alone, it achieves the highest throughput at 87 MBps. However, as multiple other consumers join, the monitored consumer's throughput drops from 87 MBps to 31 MBps. Alternatively, if we enforce 50 MBps quota per clientId, then the throughput of the monitored consumer will increase from 31 MBps to 44.3 MBps, roughly 43% increase in throughput. *This demonstrates that quota is effective in improving throughput of small consumer group in the presence of large consumer group.*

Summary and our answer to the questions presented at the beginning:

- 1) Quota functionality is working properly -- the throughput is lower than quota limit imposed by quota. And throughput is generally close to the quota limit, if the throughput without quota enforcement is higher than quota limit.
- 2) Quota is effective in improving latency of small producer in the presence fast producers.
- 3) Quota is effective in improving both latency and throughput of medium producer in the presence of fast producers.
- 4) Quota is effective in improving throughput of small consumer group in the presence of large consumer group.