KAFKA PERFORMANCE TUNING

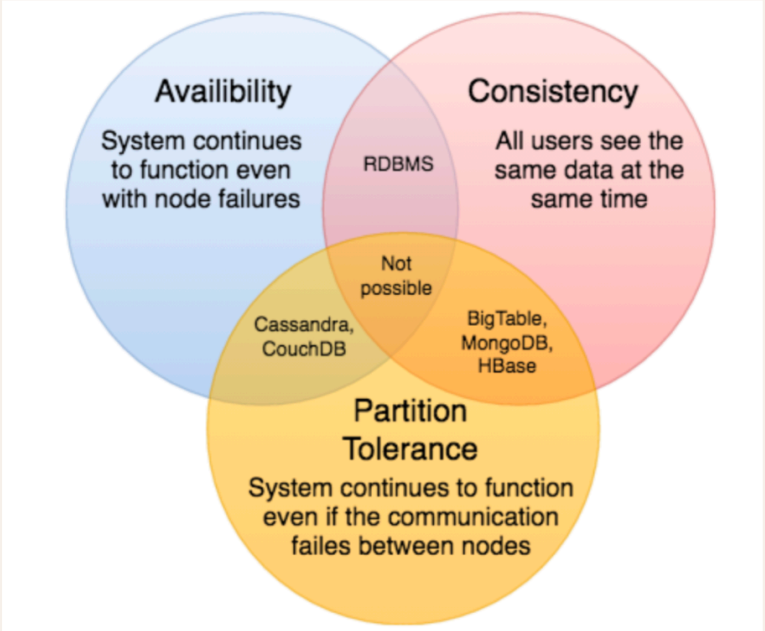
Đơn vị: Công ty CP Giáo dục và Công nghệ QNET

QNET JOINT STOCK COMPANY

Address: 14th Floor, VTC Online Tower
18 Tam Trinh Street. Hoang Mai District
Hanoi, Vietnam

Kafka Performance Tuning

CAP theorem



Định luật CAP

Consistency

Consistency means that all clients see the same data at the same time, no matter which node they connect to. For this to happen, whenever data is written to one node, it must be instantly forwarded or replicated to all the other nodes in the system before the write is deemed 'successful.'

Availability

Availability means that that any client making a request for data gets a response, even if one or more nodes are down. Another way to state this—all working nodes in the distributed system return a valid response for any request, without exception.

Partition tolerance

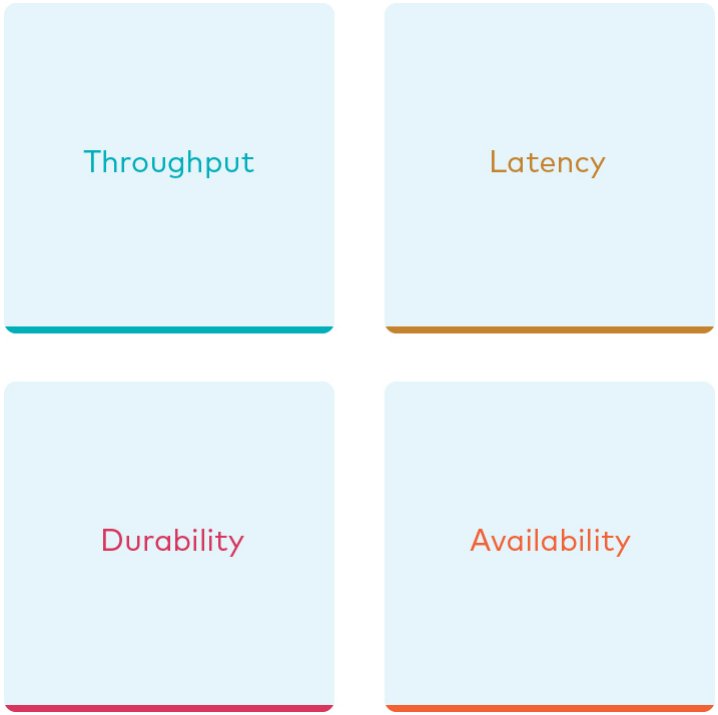
A *partition* is a communications break within a distributed system—a lost or temporarily delayed connection between two nodes. Partition tolerance means that the cluster must continue to work despite any number of communication breakdowns between nodes in the system.

Kafka Performance Tuning

Tuning process

Kafka Performance Tuning

Decide Which services goals to optimize

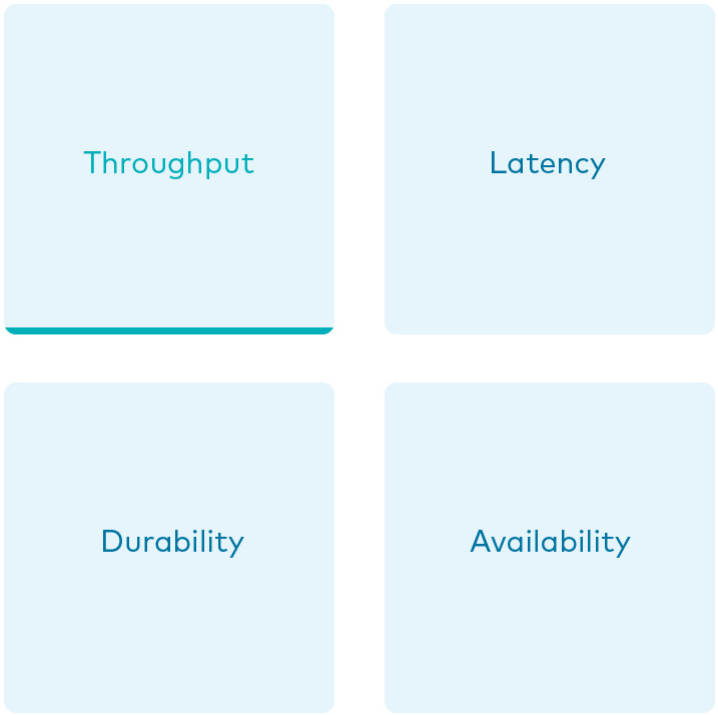


We consider four goals which often involve tradeoffs with one another:

- throughput
- latency
- durability
- availability

Kafka Performance Tuning

Optimize for Throughput



To optimize for throughput, the producers, brokers, and consumers need to move as much data as they can within a given amount of time. For high throughput, you are trying to maximize the rate at which this data moves. This data rate should be as fast as possible. A topic partition is the unit of parallelism in Kafka. Messages to different partitions can be sent in parallel by producers, written in parallel by different brokers, and read in parallel by different consumers. In general, a higher number of topic partitions results in higher throughput, and to maximize throughput, you want enough partitions to utilize all brokers in the cluster.

Kafka Performance Tuning

Optimize for Throughput

Producer:

• batch.size: increase to 100000 – 200000 (default 16384) • linger.ms: increase to 10 – 100 (default 0)
• compression.type=lz4 (default none, i.e., no compression) • acks=1 (default 1)

• buffer.memory: increase if there are a lot of partitions (default 33554432) Consumer:

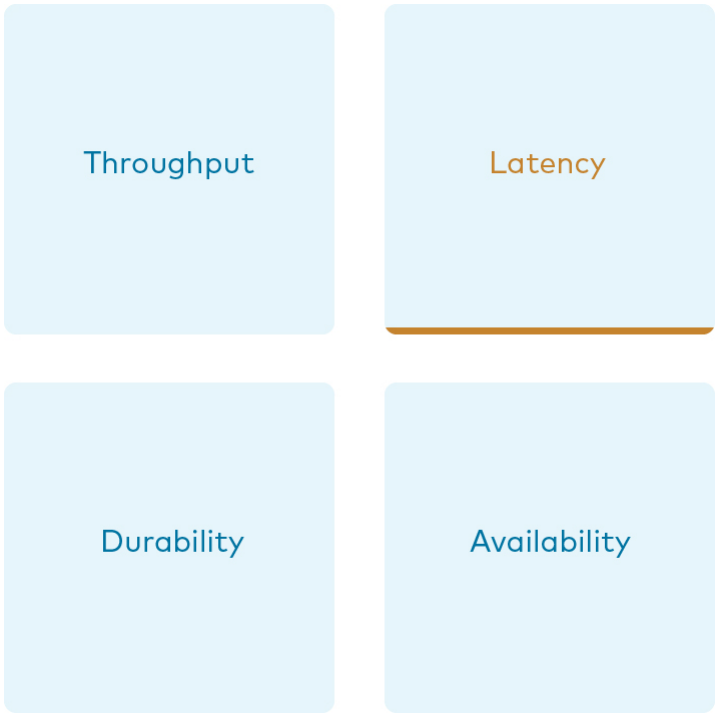
• fetch.min.bytes: increase to ~100000 (default 1)

Consumer:

• fetch.min.bytes: increase to ~100000 (default 1)

Kafka Performance Tuning

Optimize for Latency



- Number of partitions on the brokers
- Number of fetcher threads used to replicate messages to other brokers
- Time waiting for a batch of producer (default linger.ms=0)
- Enable compression
- Number of acknowledgments the producer requires the leader broker to have received before considering a request complete

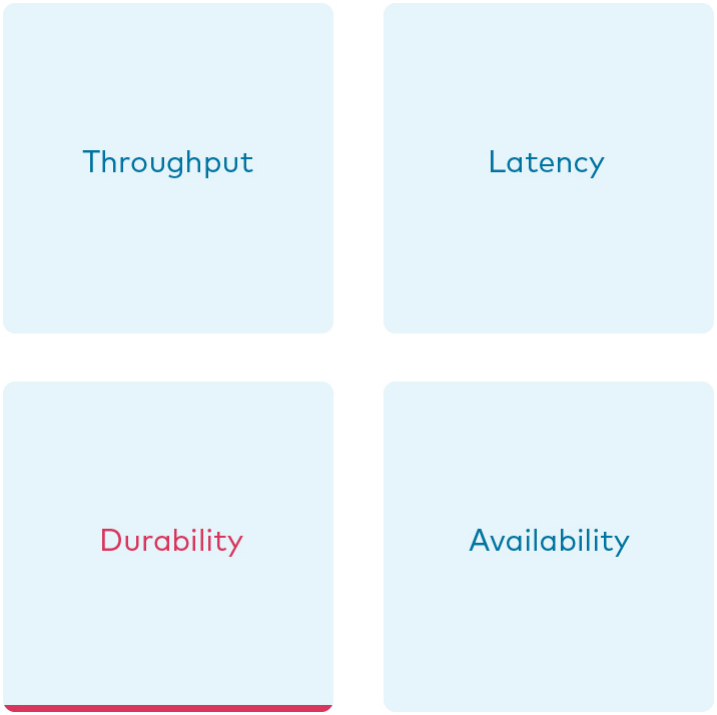
Kafka Performance Tuning

Optimize for Latency

- Producer:
  - linger.ms=0 (default 0)
  - compression.type=none (default none, i.e., no compression)
  - ·acks=1 (default 1)
- Consumer
  - fetch.min.bytes=1 (default 1)
- Broker:
  -
    · num.replica.fetchers: increase if followers can't keep up with the leader (default 1)

Kafka Performance Tuning

Optimize for Durability



Durability is all about reducing the chance for a message to get lost. The most important feature that enables durability is replication, which ensures that messages are copied to multiple brokers. If a broker has a failure, the data is available from at least one other broker.

Kafka Performance Tuning

Optimize for Durability

Producer:

- `replication.factor=3` (topic override available)
- `acks=all` (default 1)
- `enable.idempotence=true` (default false), to handle message duplication and ordering
- `max.in.flight.requests.per.connection=1` (default 5), to prevent out of order messages when not using an idempotent producer

Kafka Performance Tuning

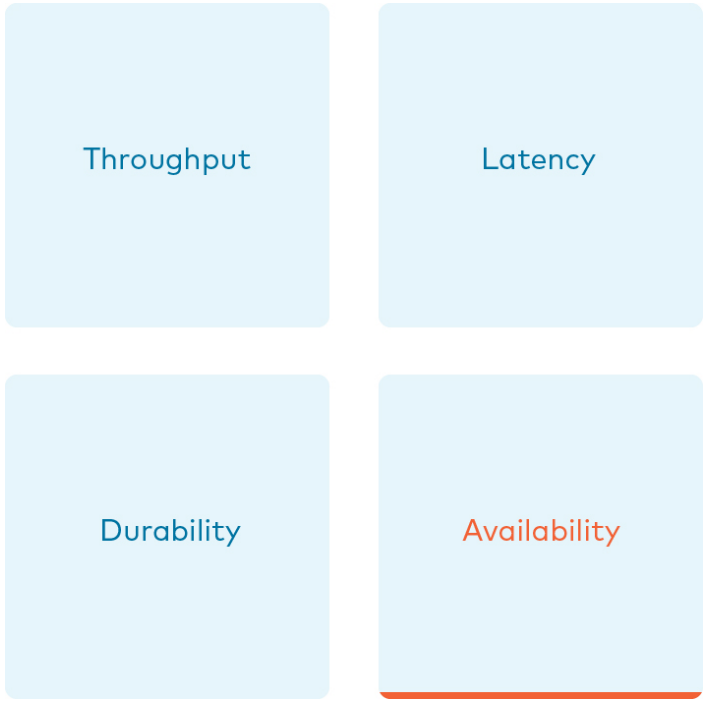
Optimize for Durability

Broker:

- `default.replication.factor=3` (default 1)
- `auto.create.topics.enable=false` (default true)
- `min.insync.replicas=2` (default 1); topic override available
- `unclean.leader.election.enable=false` (default false); topic override available
- `broker.rack`: rack of the broker (default null)
- `log.flush.interval.messages`, `log.flush.interval.ms`: for topics with very low throughput, set message interval or time interval low as needed (default allows the OS to control flushing); topic override available

Kafka Performance Tuning

Optimize for Availability



Kafka Performance Tuning

Optimize for Availability

Consumer:

- `session.timeout.ms`: as low as feasible (default 10000)

Streams:

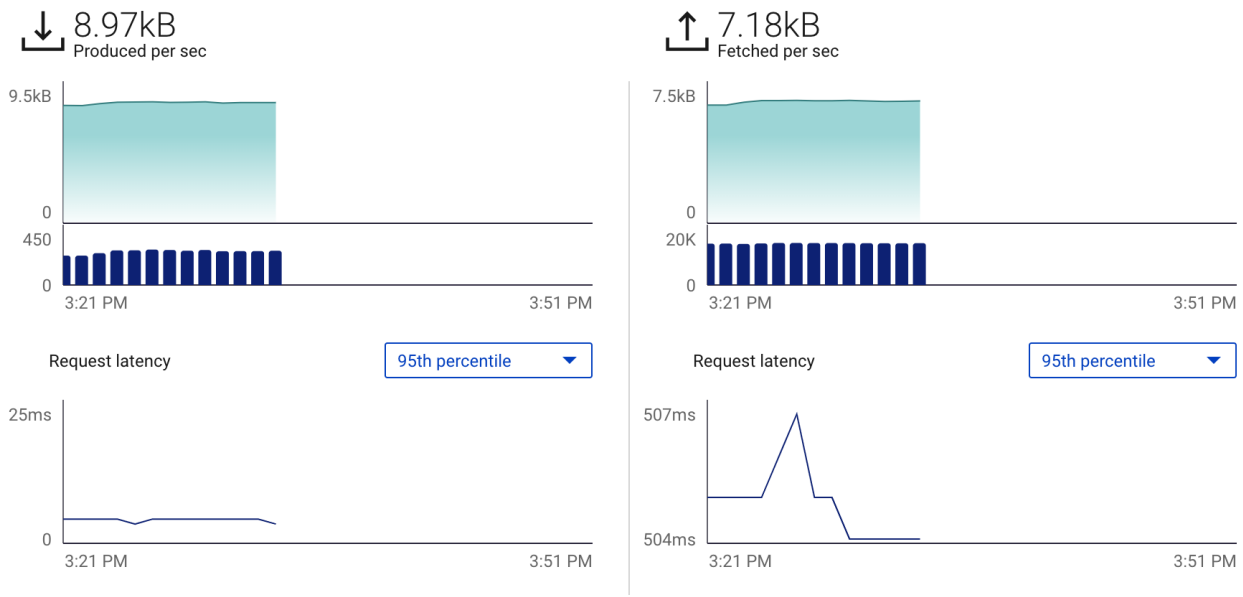
- `StreamsConfig.NUM_STANDBY_REPLICAS_CONFIG`: 1 or more (default 0)
- Streams applications have embedded producers and consumers, so also check those configuration recommendations

Broker:

- `unclean.leader.election.enable=true` (default false); topic override available
- `min.insync.replicas=1` (default 1); topic override available
- `num.recovery.threads.per.data.dir`: number of directories in `log.dirs` (default 1)

Kafka Performance Tuning

Bendmarking



Kafka Performance Tuning

Bendmarking

Metric	Description
`kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec`	The bytes in per second the broker is receiving
`kafka.server:type=BrokerTopicMetrics,name=BytesOutPerSec`	The bytes out per second the broker is sending
`kafka.server:type=BrokerTopicMetrics,name=MessagesInPerSec`	Number of incoming messages per second
`kafka.network:type=RequestMetrics,name=RequestsPerSec,request={Produce|FetchConsumer|FetchFollower}`	Number of requests per second, for produce, consumer fetch, and replica follower fetch
`kafka.network:type=RequestMetrics,name=TotalTimeMs,request={Produce|FetchConsumer|FetchFollower}`	Total time a request takes to be completed, for produce, consumer fetch, and replica follower fetch

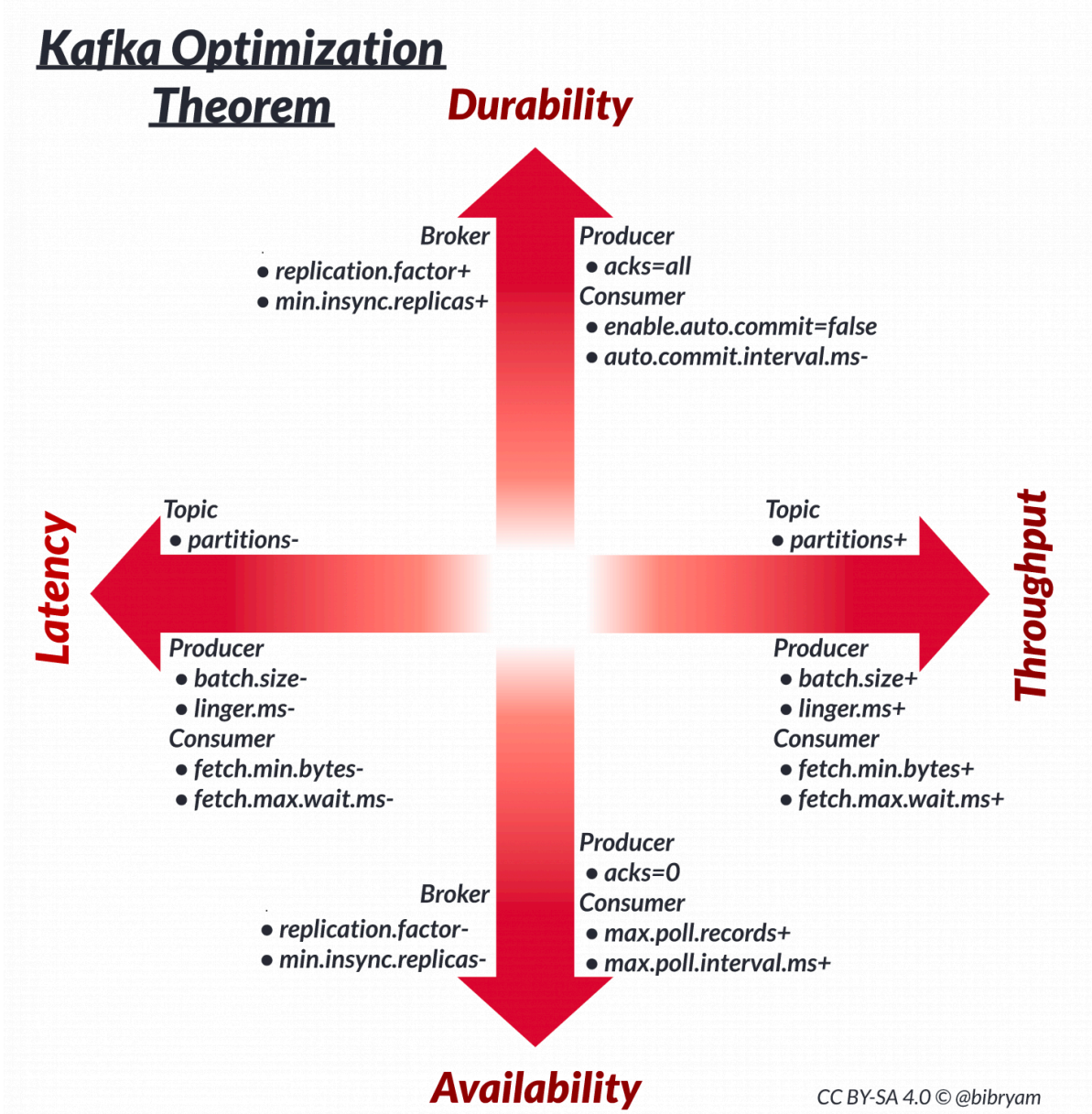
Kafka Performance Tuning

Monitor consumer lag

Consumer group Name		Messages behind	Consumers	Topics
EN_WIKIPEDIA_GT_1_COUNTS-consumer	•••	0	1	1
_confluent-ksql-default_query_CSAS_WIKIPEDIANOBOT_0	•••	7	2	1
_confluent-ksql-default_query_CSAS_EN_WIKIPEDIA_GT_1_COUNTS_3	•••	1	2	1
_confluent-controlcenter-5-2-1-1	•••	2541	1	15
_confluent-controlcenter-5-2-1-1-command	•••	0	1	1
connect-elasticsearch-ksql	•••	22	1	1
WIKIPEDIANOBOT-consumer	•••	11	1	1
connect-replicator	•••	89	0	1
_confluent-ksql-default_query_CTAS_EN_WIKIPEDIA_GT_1_2	•••	4	4	2
_confluent-ksql-default_query_CSAS_WIKIPEDIABOT_1	•••	12	2	1

Kafka Performance Tuning

Parameter



DISCUSSION

Quality Network for Education and Technology

XIN CHÂN THÀNH CẢM ƠN!