Kafka Monitoring: What Matters!

Amrit Sarkar

THIS IS NOT A CONTRIBUTION
Agenda

- Kafka Basics
- Performance Areas
- Need for Observability
  - Monitoring Options
- Performance classification around Components
- Kafka Consumer Lag evaluation
  - absolute to relative
- Trend Analysis
Kafka Basics

Kafka moves data between producers (writers) and consumers (readers), with data protection, high availability, low latency at high scale!

Use cases: Metrics, Log Aggregation Solution & Stream Processing

Brokers use ZK to manage and share state***

* ZK has been deprecated/removed in newer versions
Kafka Basics

Topics are divided into at least one partition.

New record is appended to one of the partitions.

Unique Offset for each record. Consumers track offsets for reading locations.

All partitions must be consumed for the topic.
Kafka Basics

Topics are divided into at least one partition.

New record is appended to one of the partitions.

All partitions must be consumed for the topic.

Consumer Group Leader rebalances partitions as consumers add / leave group.

Unique Offset for each record. Consumers track offsets for reading locations.
Kafka Basics

Kafka Cluster

<table>
<thead>
<tr>
<th>Topic</th>
<th>Partition</th>
<th>Replica</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>P0</td>
<td>R1</td>
</tr>
<tr>
<td>T1</td>
<td>P1</td>
<td>R1</td>
</tr>
<tr>
<td>T2</td>
<td>P0</td>
<td>R1</td>
</tr>
<tr>
<td>T3</td>
<td>P0</td>
<td>R1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic</th>
<th>Partition</th>
<th>Replica</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>P0</td>
<td>R2</td>
</tr>
<tr>
<td>T1</td>
<td>P1</td>
<td>R2</td>
</tr>
<tr>
<td>T2</td>
<td>P0</td>
<td>R2</td>
</tr>
<tr>
<td>T3</td>
<td>P0</td>
<td>R2</td>
</tr>
</tbody>
</table>

*ZK has been deprecated / removed in newer versions*
### Kafka Basics

**Controller Broker**

- Create and Delete Topics
- Partitions States & Leaders reassignment

---

**Kafka Cluster**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Partition</th>
<th>Replica</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>P0</td>
<td>R1</td>
</tr>
<tr>
<td>T1</td>
<td>P1</td>
<td>R1</td>
</tr>
<tr>
<td>T2</td>
<td>P0</td>
<td>R1</td>
</tr>
<tr>
<td>T3</td>
<td>P0</td>
<td>R1</td>
</tr>
<tr>
<td>T1</td>
<td>P0</td>
<td>R2</td>
</tr>
<tr>
<td>T1</td>
<td>P1</td>
<td>R2</td>
</tr>
<tr>
<td>T2</td>
<td>P0</td>
<td>R2</td>
</tr>
<tr>
<td>T3</td>
<td>P0</td>
<td>R2</td>
</tr>
</tbody>
</table>

*ZK has been deprecated / removed in newer versions*
Performance Areas

- Throughput & Latency
  - Production Rates
  - Consumption Rates
  - Consumers’ Lag

- Data Integrity
  - Reads Confirmation
  - Writes Confirmation

- Fault Tolerance
  - No business impact on failure

- Resource Usage
Why do we need Observability?

Pre-built dashboards monitor and alert for anticipated future performance issues.
Explore and quickly identify unanticipated issue root causes in an observability scenario.

Kafka doesn't self report problems, it reports metrics
Active Controller Count (ACC)
Monitoring tells you metrics represent a problem
Alerts on multiple ACC
Observability guides to fix the problem
Servers restart, hostname failures, ZK health...
Performance classification around Components

Each component acts as a potential factor in the performance of Kafka messaging system.

- Producers
  - Rate
- Brokers
  - Topics Health
  - Load Distribution
- Topics
  - Partition Health
  - Load Distribution
- Consumers
  - Rate
  - Lag
- Generic
  - Transmission (Network) health
  - Capacity

* ZK has been deprecated / removed in newer versions
Monitoring Options - Getting Metrics In

- Confluent Control Center
- KafDrop
- Yahoo Kafka Manager
- Cruise Control
- Kafka Monitor
- Kafka Tool
... and more
Producer: Rate

- `send()` function call (or similar) to push data → `RecordMetadata` object
  - `offset()` function returns a `LONG` → offset of the record in the topic-partition.

*Offset value can be pushed to metrics store for visualisation.*
Producer: Compression & Latency

- Bigger batches →
  - higher throughput
  - less compression
  - Small enough to keep GC ↓ (<< 10mb)

- Batch Size in Bytes → Optimal High
  - `kafka.producer:type=producer-metrics,client-id="[client-id]" batch-size-avg`

- Compression Rate → LOW
  - `kafka.producer:type=producer-metrics,client-id="[client-id]" compression-rate-avg`

- Request Latency → LOW
  - `kafka.producer:type=producer-metrics,client-id="[client-id]" request-latency-avg`
The Big Four - Key Metrics (JMX)

- Number of active controllers, **must be = 1**
  
  ```
  kafka.controller:name=ActiveControllerCount, type=KafkaController
  ```

- Number of under min ISR partitions, **must be = 0**
  
  ```
  kafka.server:name=UnderMinIsrPartitionCount, type=ReplicaManager
  
  Checkout: 'UnderReplicatedPartitions' metric too
  ```

- Number of offline partitions, **must be = 0**
  
  ```
  kafka.controller:name=OfflinePartitionsCount, type=KafkaController
  ```

- Consumer Lag (per partition)
  
  ```
  kafka.consumer:name=MaxLag, type=ConsumerFetcherManager,clientId=[-.w]+)
  ```
The Big Four - Key Metrics (JMX)

Kafka Active Controller Count

Kafka Offline Partitions Count

broker-0: 0
broker-1: 0

Kafka Under Min In-Sync Replicas Partitions

<table>
<thead>
<tr>
<th>Topic</th>
<th>broker-0</th>
<th>broker-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>_consumer_offsets</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dummy</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>filtered</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>greeting</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>longMessage</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>multi_topic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>partitioned</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>quickstart-events</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>topic1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Brokers' Health

● Load Skewness (number of partitions on a broker)
  \[ \text{kafka.server:name=PartitionCount,} \]
  \[ \text{type=ReplicaManager} \]

● Log Flush Latency
  \[ \text{kafka.log:name=LogFlushRateAndTimeMs,} \]
  \[ \text{type=LogFlushStats} \]

● Network Request & Error Rate
  \[ \text{kafka.network:name=RequestsPerSec/ErrorsPerSec,} \]
  \[ \text{type=RequestMetrics} \]

● Fetcher Lag (per topic per partition)
  \[ \text{kafka.server:name=ConsumerLag,} \]
  \[ \text{type=FetcherLagMetrics,clientId=([^\w]+),topic=([^\w]+),} \]
  \[ \text{partition=([^0-9]+)} \]
Brokers' Health
Consumers: Committing Offsets

Topic

Topics are divided into at least one partition

Partition 0

Partition 1

0 | 1 | 2 | 3 | 4 | ...

Producer

Kafka Cluster

Broker Z

__consumer_offsets

Write offset 4 for CG A
for P1

Write offset 2 for CG B
for P1

Consumer Group A

Consumer Group B

Write

Read
Consumers: Commit Rate

Too Frequent Commit Rate

- Increased Network Overhead
- Increased Load on Broker
- Potential Higher Latency

Ideally

- Commit after Batch of Messages $\rightarrow$ Less Frequent Rate
- commitAsync $\rightarrow$ Improve Throughput
- Tune `auto.commit.interval.ms`
**Consumption Rate**

`__consumer_offsets` topic can be consumed; offset long value emitted as metric to visualise **every consumer-partition’s committed offset** in near real-time.

*Reference Burrow (discussed later) which already does it.*
Lag, Evaluations & Alerts: Burrow

Monitoring tool provides **consumer lag check as a service**

Exposes **offset lag** for all consumer-partition combination as **Prometheus metrics**.

Monitors committed offsets and **calculates the status of those consumers** on demand.

- Able to send alerts
Lag - Offsets Trend Evaluation

Lag = Head Offset of the Broker - Consumer's Offset

Store metrics locally in TSDB format. Evaluation runs on a sliding window periodically.

Status of consumer: OK
Lag Series with no Uptrend & Consumer Offset Series not Stalled
Lag - Consumer is Slow!

Status of consumer: **WARNING**
Lag Series with Uptrend & Consumer Offset Series not Stalled
Lag - Consumer Stalled

Status of consumer: **STALLLED**
Lag Series with Uptrend & Consumer Offset Series Stalled
Lag - Observability

Do we need to worry about this uptrend?
Lag - Time Based

Diff (Last Consumed Offset, Last Produced Offset)

Producer Rate
Lag - Time Based

Normal Trend!
Trend Analysis

Keep track of high-level metrics for:

- Rate of Topic Growth → Do we need more partitions?
- Weekly / Monthly / Periodic Producer / Consumer Rate → Keeping tabs on abnormal spikes!
- TTL / Retention data long enough to hold data for consumption → If time lag for consumer-partition goes beyond control!
- Infrastructure supporting Kafka cluster requirements → CPU, Memory, Network, IO capacity, GC Activity
- Zookeeper supporting Kafka cluster requirements → How many topics / partitions state can be kept?

* ZK has been deprecated / removed in newer versions
References
Thank You!
@sarkaramrit2