Cruise Control: Effortless Management of Kafka Clusters

Adem Efe Gencer
Senior Software Engineer
LinkedIn
Kafka: A Distributed Stream Processing Platform

- High throughput & low latency
- Message persistence on partitioned data
- Total ordering within each partition
Key Concepts: Brokers, Topics, Partitions, and Replicas

Kafka Cluster

- : Broker-0
  - 1
  - 2
- : Broker-1
  - 1
  - 1
  - 2
- : Broker-2
  - 1

1: A Replica of Partition-1 of Blue Topic
Key Concepts: Leaders and Followers

- **The Leader Replica**: Broker-0
- **A Follower Replica**: Broker-1

1: The Leader Replica
1: A Follower Replica
Key Concepts: Producers
Key Concepts: Consumers

Broker-0
1 2

Consumer-1

Broker-1
1 1 2

Consumer-2

Broker-2
1
Key Concepts: Failover via Leadership Transfer

: Broker-0
1 2

: Broker-1
1 1 2

X

: Broker-2
1
Key Concepts: Failover via Leadership Transfer
Kafka Incurs Management Overhead

- Large deployments – e.g. @LinkedIn: 2.6K+ Brokers, 44K+ Topics, 5M Partitions, 5T Messages / day

- Frequent hardware failures

- Load skew among brokers

- Kafka cluster expansion and reduction

*Elephant* (CC0): [https://pixabay.com/en/elephant-safari-animal-defence-1421167/]

*"Seesaw at Evelle" by Rachel Coleman (CC BY-SA 2.0)*: [https://www.flickr.com/photos/rmc28/4862153310/]

*"Inflatable Balloons" (Public Domain)*: [https://commons.wikimedia.org/wiki/File:InflatableBalloons.jpg]
Alleviating the Management Overhead

1. Admin Operations for Cluster Maintenance
2. Anomaly Detection with Self-Healing
3. Real-Time Monitoring of Kafka Clusters
Admin Operations for Cluster Maintenance

1. Dynamically balance the cluster load
2. Add / remove brokers
3. Demote brokers – i.e. remove leadership of all replicas
4. Trigger preferred leader election
5. Fix offline replicas
Admin Operations for Cluster Maintenance

- Dynamically balance the cluster load
- Add / remove brokers
- Demote brokers – i.e. remove leadership of all replicas
- Trigger preferred leader election
- Fix offline replicas
Dynamically Balance the Cluster Load

Must satisfy *hard goals*, including:

- Guarantee rack-aware distribution of replicas
- Never exceed the capacity of broker resources – e.g. disk, CPU, network bandwidth
- Enforce operational requirements – e.g. maximum replica count per broker
Dynamically Balance the Cluster Load

Satisfy *soft goals* as much as possible – i.e. best effort

- Balance disk, CPU, inbound/outbound network traffic utilization of brokers
- Balance replica distribution
- Balance potential outbound network load
- Balance distribution of partitions from the same topic
Anomaly Detection with Self-Healing

1. Goal violation – rebalance cluster
2. Broker failure – decommission broker(s)
3. Metric anomaly – demote broker(s)
3 Real-Time Monitoring of Kafka Clusters

: Examine the replica, leader, and load distribution

: Identify under-replicated, under-min-ISR, and offline partitions

: Check the health of brokers, disks, and user tasks
Building Blocks of Management: Moving Replicas

: Broker-0

1  2

: Broker-1

1  2  1

Replica Move
Replica Move

Broader impact, but expensive
• Requires data transfer*

* Replica swap: Bidirectional reassignments of distinct partition replicas among brokers
Building Blocks of Management: Moving Leadership

Broker-0

Broker-1

Leadership Move
Building Blocks of Management: Moving Leadership

Leadership Move
Cheap, but has limited impact
- Affects network bytes out and CPU
A Multi-Objective Optimization Problem

Achieve conflicting cluster management goals while minimizing the impact of required operations on user traffic
ARCHITECTURE

Joy Oil Gas Station Blueprints (Public Domain): https://commons.wikimedia.org/wiki/File:Joy_Oil_gas_station_blueprints.jpg
Cruise Control Architecture

- REST API
  - Sample Store
  - Metric Sampler
  - Monitor
  - Capacity Resolver
  - Analyzer
    - Goal(s)
      - Anomaly Detector
      - Anomaly Notifier
    - Broker Failure
    - Metric Anomaly
    - Goal Violation
  - Executor
  - Executor
  - Backup and Recovery
  - Load History T.
  - Metrics Reporter T.
  - Metrics Reporter
  - Kafka Cluster
    - Anomaly Detector
    - Anomaly Notifier
    - Broker Failure Finder(s)
    - Internal Topic
    - Throttled Proposal Execution
    - Pluggable Component
      - Implements a public interface
      - Accepts custom user code
      - Created and used by Cruise Control and its metrics reporter

- Backup and Recovery
- Load History T.
- Metrics Reporter T.
- Metrics Reporter
- Kafka Cluster
- Anomaly Detector
- Anomaly Notifier
- Broker Failure Finder(s)
- Internal Topic
- Throttled Proposal Execution
- Pluggable Component
  - Implements a public interface
  - Accepts custom user code
  - Created and used by Cruise Control and its metrics reporter
Metrics Reporter

Produces selected Kafka cluster metrics to the configured metrics reporter topic with the configured frequency
Monitor

Generates a model (\ldots) to describe the cluster
Monitor: Cluster Model

- **Topology** – rack, host, and broker distribution
- **Placement** – replica, leadership, and partition distribution
- **Load** – current and historical utilization of brokers and replicas
Monitor: Metric Sampler

Generates a model to describe the cluster
- Periodically (e.g. every 5 min) consumes the reported metrics to model the load on brokers and partitions
Generates a model (¶¶) to describe the cluster

- Periodically (e.g., every 5 min) consumes the reported metrics to model the load on brokers and partitions
- Produces broker and partition models to load history topic, and uses the stored data to recover upon failure
Monitor: Capacity Resolver

Generates a model to describe the cluster

- Periodically (e.g., every 5 min) consumes the reported metrics to model the load on brokers and partitions
- Produces broker and partition models to load history topic, and uses the stored data to recover upon failure
- Gathers the broker capacities from a pluggable resolver
Analyzer

Generates proposals to achieve goals via a fast and near-optimal heuristic solution
Analyzer: Goals

Generates proposals to achieve goals via a fast and near-optimal heuristic solution:

- **Priorities** – custom order of optimization
- **Strictness** – hard (e.g. rack awareness) or soft (e.g. resource utilization balance) optimization demands
- **Modes** – e.g. kafka-assigner (https://github.com/linkedin/kafka-tools)
Analyzer: Proposals

Generates proposals to achieve goals via a fast and near-optimal heuristic solution

Proposals – in order of priority:
• Leadership move > Replica move > Replica swap
Executor

Proposal execution:
• Dynamically controls the maximum number of concurrent leadership / replica reassignments
• Ensures only one execution at a time
• Enables graceful cancellation of ongoing executions
Integration with replication quotas (KIP-73)
Anomaly Detector

Identifies, notifies, and fixes (self-healing):
• Violation of anomaly detection goals
• Broker failures
• Metric anomalies
• Disk failures (JBOD)

: Faulty vs Healthy Cluster

: Reactive vs. Proactive Mitigation
Anomaly Detector: Goal Violations and Self-Healing

Checks for the violation of the anomaly detection goals

- Identifies *fixable* and *unfixable* goal violations
- Self-healing triggers a cluster rebalance operation
- Avoids false positives due to broker failure, upgrade, restart, or release certification
Anomaly Detector: Broker Failures

Concerned with whether brokers are responsive:
- Ignores the internal state deterioration of brokers
- Identifies fail-stop failures
Anomaly Detector: Broker Failures and Self-Healing

Checks for broker failures:

- Enables a grace period to lower false positives – e.g. due to upgrade, restart, or release certification
- Self-healing triggers a remove operation for failed brokers
Anomaly Detector: Reactive Mitigation

Cluster maintenance becomes costly

Requires immediate attention of affected services

Poor user experience due to frequent service interruptions

Server & network failures

- Size of clusters
- Volume of user traffic
- Hardware degradation

---

Anomaly Detector

<table>
<thead>
<tr>
<th>Goal Violation</th>
<th>Metric Anomaly</th>
<th>Broker Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finder(s)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Anomaly Notifier
Anomaly Detector: Metric Anomaly

Checks for abnormal changes in broker metrics – e.g. a recent spike in log flush time:

• Self-healing triggers a demote operation for slow brokers
Anomaly Detector: Metric Anomaly

Compares current and historical metrics to detect slow brokers:

- The comparison in the default finder is based on the percentile rank of the latest metric value
- Metrics of interest are configurable – e.g. local time of produce / consume / follower fetch, log flush time
- Supports multiple active finders
Anomaly Detector: Proactive Mitigation

In-place fix of slow / faulty brokers is non-trivial

- The root cause could be a hardware issue (e.g. a misbehaving disk), a software glitch, or a traffic shift
- Hence, the mitigation strategies are agnostic of the particular issue with the broker

REST API

Supports sync and async endpoints including:

- Cluster Load
- Partition Load
- Proposals
- Kafka Cluster State
- Cruise Control State
- User Tasks

GET

- Add / Remove / Demote Broker
- Rebalance Cluster
- Fix Offline Replicas (JBOD)
- Stop Ongoing Execution
- Pause / Resume Sampling
- Admin – ongoing behavior changes

POST

GUI & multi-cluster management
Managing the Manager – Monitoring Cruise Control

Reported JMX metrics include:

- **Executor**: Started, stopped, and ongoing executions in different modes, and the status of balancing tasks
- **Anomaly Detector**: Broker failure, goal violation, and metric anomaly rate
- **Monitor**: Cluster model and sampling performance
- **Analyzer**: Stats on proposal generation
Evaluation: Remove Brokers and Rebalance

1. All Topics Bytes-In Rate [per broker]

2. Incoming data (bytes/s)

Time (hours)
Evaluation: Remove Brokers and Rebalance

1. All Topics Bytes-Out Rate [per broker]
Evaluation: Remove Brokers and Rebalance

1. Number of Partitions [per broker]

2. Time (hours)

Partition count

- 1.4k
- 1.2k
- 1.0k
- 0.8k
- 0.6k
- 0.4k
- 0.2k
- 0k

04 (12:00) 04 (15:00) 04 (20:00) 05 (00:00) 05 (04:00) 05 (08:00)
Summary

A system that provides effortless management of Kafka clusters

✓ Admin Operations for Cluster Maintenance
✓ Anomaly Detection with Self-healing
✓ Real-Time Monitoring of Kafka Clusters
✓ Integration with Other Systems – e.g. Apache Helix
More…

: Open source repository (https://github.com/linkedin/cruise-control)

: Gitter room (https://gitter.im/kafka-cruise-control)

: UI (https://github.com/linkedin/cruise-control-ui)
Rate today’s session

Session page on conference website

O'Reilly Events App