GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

Auto Management for Apache Kafka and Distributed Stateful System in General

Jiangjie (Becket) Qin
Data Infrastructure @LinkedIn

GIAC 2017, 12/23/17@Shanghai

Agenda

- Kafka introduction and terminologies
- Problems to solve
- Our solution
  - Cruise Control Architecture
  - Challenges and Solutions
- Insights and Generalization
  - Problem Generalization
  - Model Generalization
- Q&A

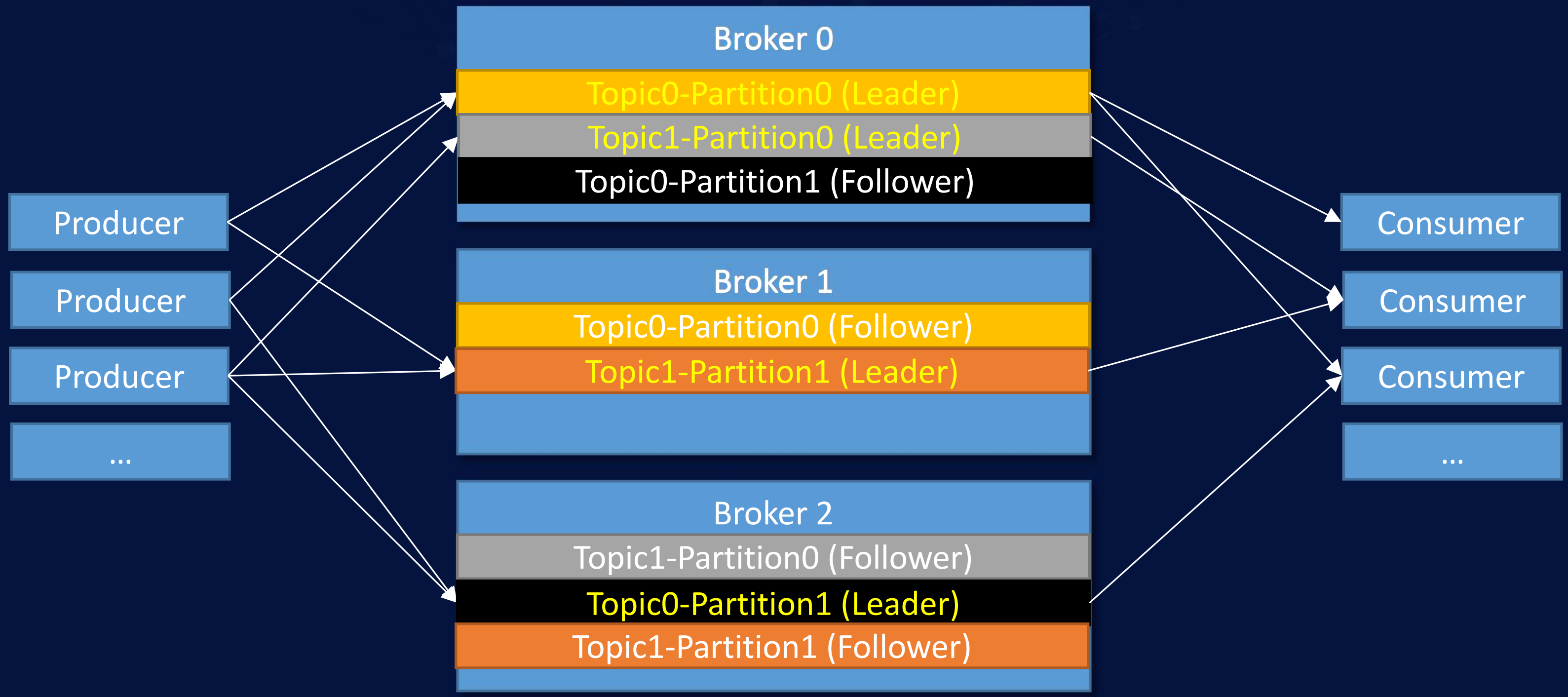
Agenda

- **Kafka introduction and terminologies**
- Problems to solve
- Our solution
  - Cruise Control Architecture
  - Challenges and Solutions
- Insights and Generalization
  - Problem Generalization
  - Model Generalization
- Q&A

What is Kafka

- An open source distributed stream processing platform
  - High throughput
  - Low latency
  - Message persistency
  - Partitioned data
  - Ordering within partitions
  - …

Basic Architecture



Terminologies

- Each **Topic** has multiple **Partitions**
- Each Partition has a few **Replicas**
  - One **Leader** Replica
  - 0+ **Follower** Replicas
- Each **Broker** (Server) hosts many replicas
- The producers and consumers are only served by Leader Replicas
  - Follower replicas are only for data redundancy

Agenda

- Kafka introduction and terminologies
- **Problems to solve**
- Our solution
    - Cruise Control Architecture
    - Challenges and Solutions
- Insights and Generalization
    - Problem Generalization
    - Model Generalization
- Q&A

Operation Challenges

- The scale of Kafka deployment @LinkedIn
  - 1,800+ brokers
  - ~ 40,000 topics
  - > 2.5 trillion messages / day
- Huge operation overhead
  - Hardware failures are norm
  - Workload skews

Requirements for cluster management

- Redundancy
  - Rack awareness
- Hardware Resource Utilization balance
  - CPU
  - Disk usage (size, IO)
  - Network bytes in rate
  - Network bytes out rate
  - Memory
- Heterogeneous cluster support

Requirements for cluster management

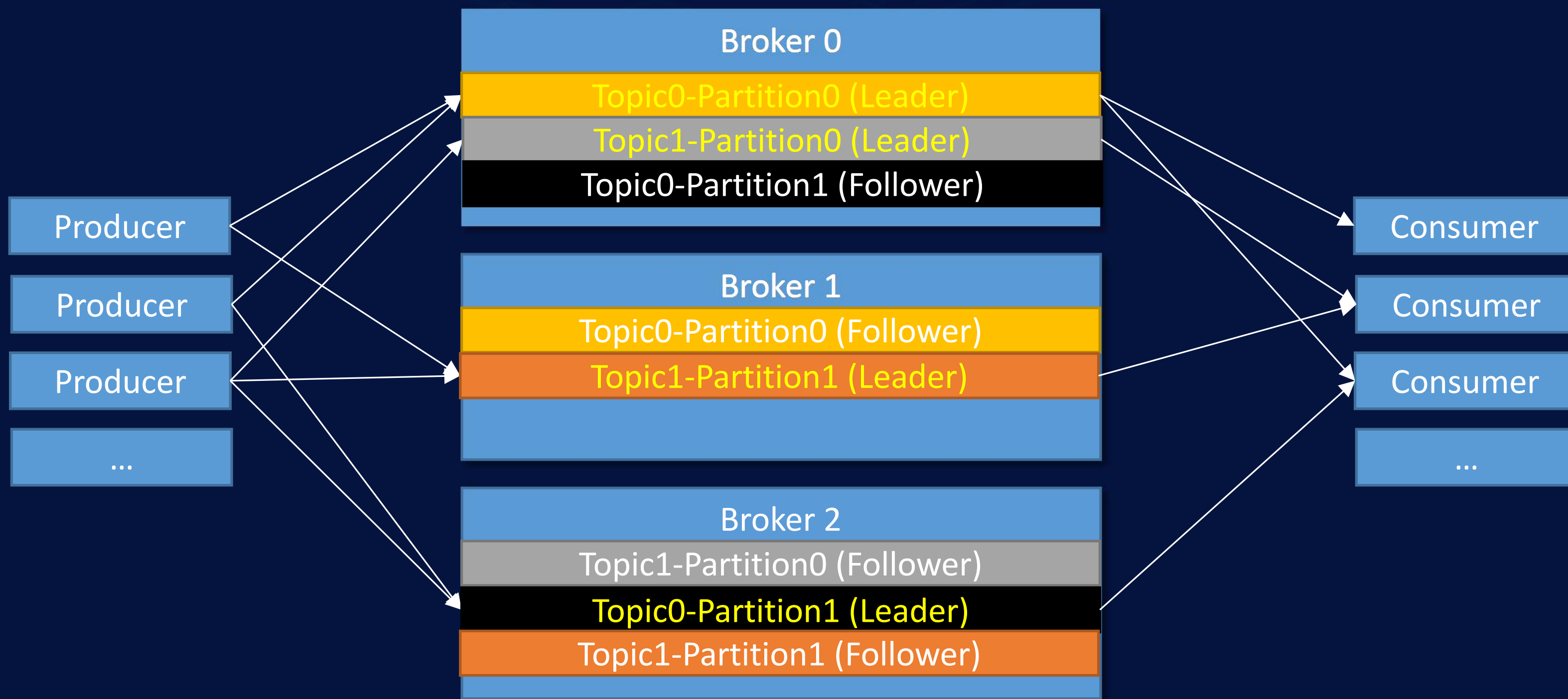
- SLA
  - Latency
  - Throughput
- Self-healing
  - Reduced window of redundancy loss

Summary of the requirements

- Dynamic Load Balancing
  - CPU, Disk, Network IO, SLA, Rack Aware…
- Failure detection and self-healing
  - Reassign the replicas on the dead brokers
  - Reduce the window of under-replication
- Other admin operations
  - Add / Decommission brokers, manual leader movement…

Problem to solve

How to manage the Kafka cluster to meet all the requirements?



Two basic operations

- Replica Movement
  - Expensive – require data copy
  - Impact on all hardware resources
- Leader Movement
  - Cheap – no data copy
  - Impact on CPU and network bytes out

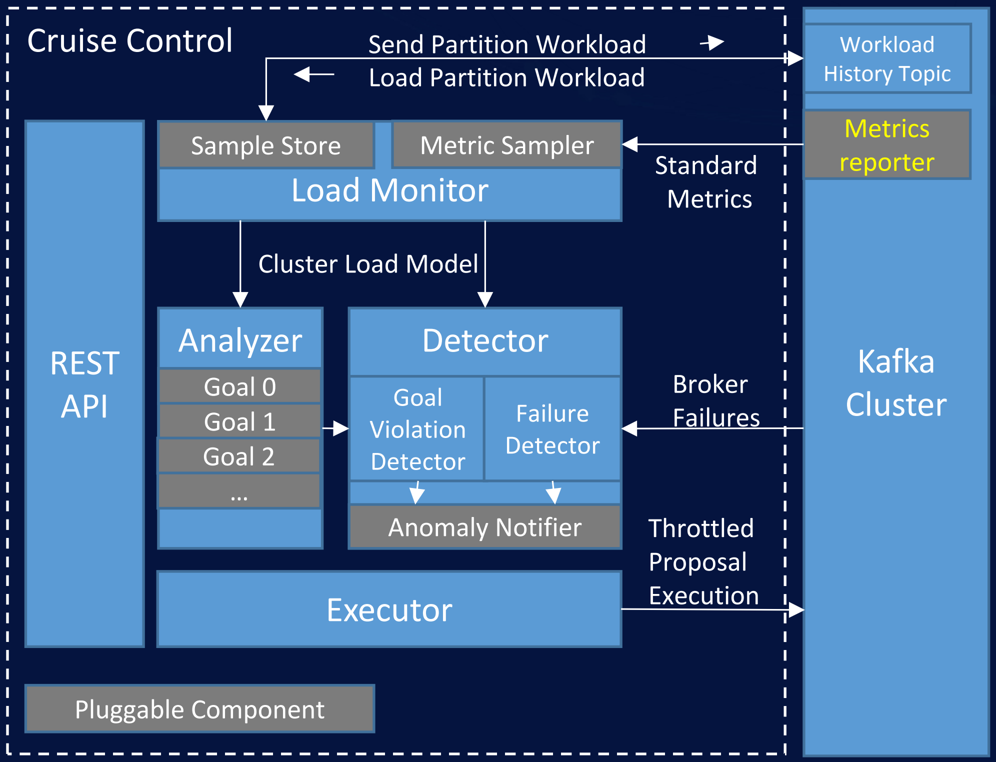
The questions to answer

- Which partition should be moved?
- What should be moved?
  - Leader Movement
  - Replica Movement
- Where to move?
  - Move to which broker
- How much should be moved?
- Moving cost?

Agenda

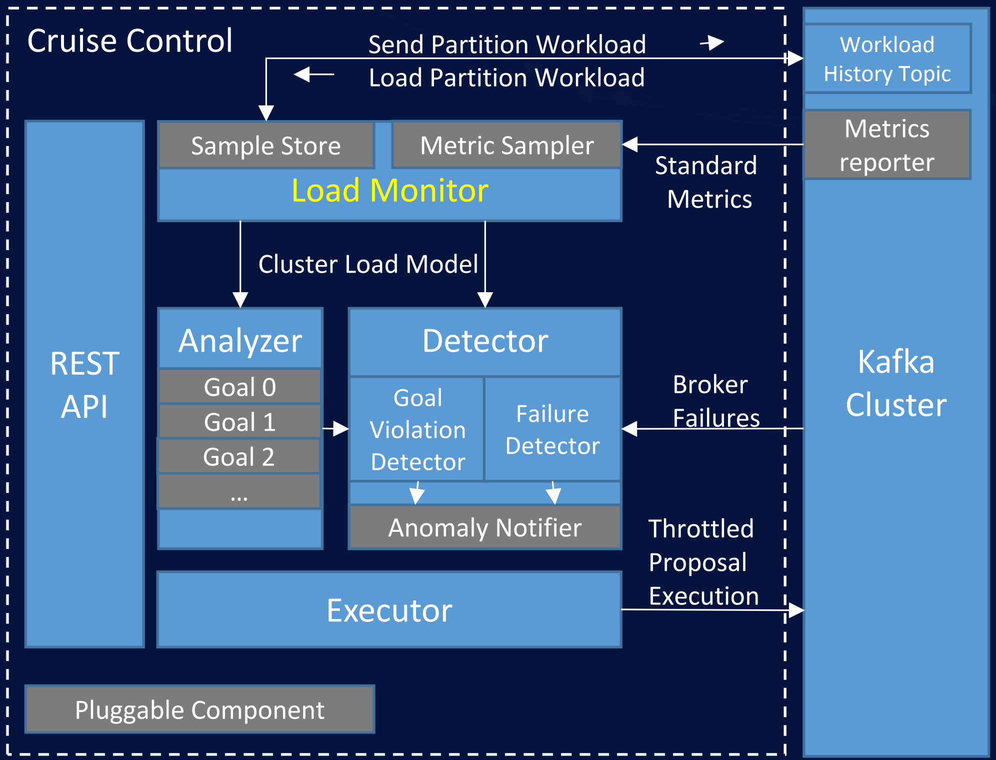
- Kafka introduction and terminologies
- Problems to solve
- **Our solution**
  - **Cruise Control Architecture**
  - Challenges and Solutions
- Insights and Generalization
  - Problem Generalization
  - Model Generalization
- Q&A

Cruise Control Architecture



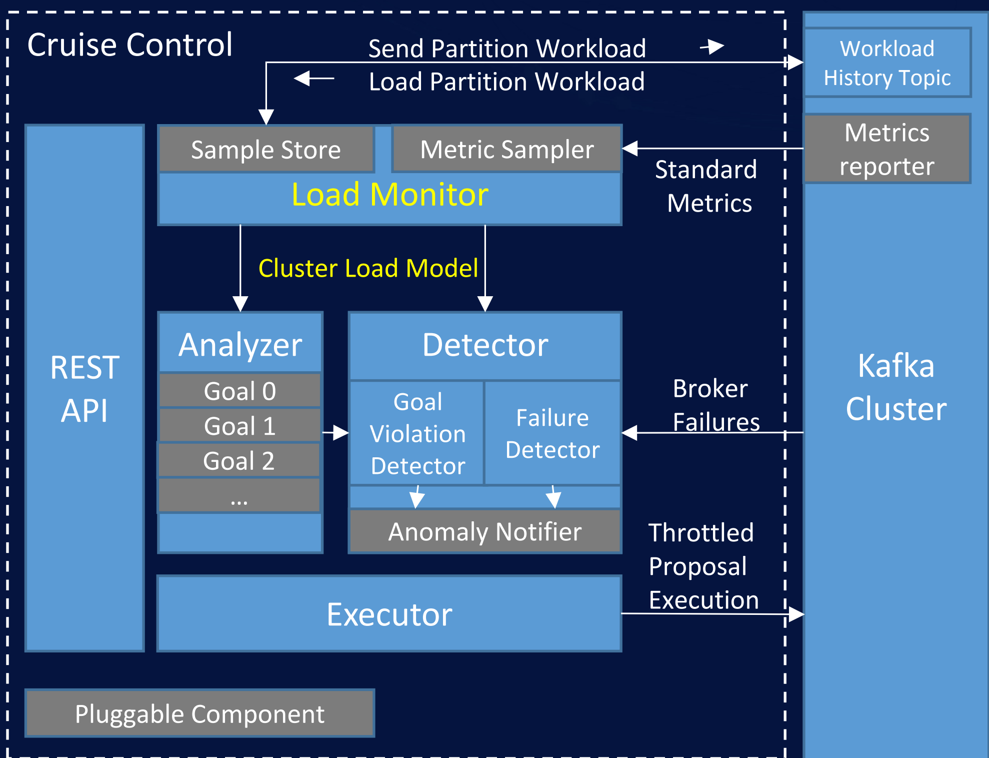
- Metrics reporter collects the standard Kafka metrics and send them to a Kafka topic (__CruiseControlMetrics).

Cruise Control Architecture

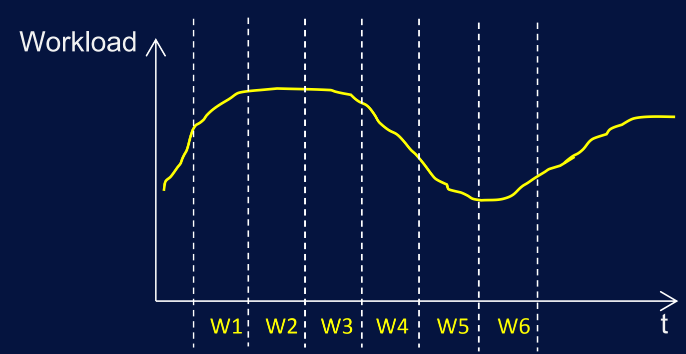


- Load Monitor generates a Cluster Load Model to describe the workload of the cluster

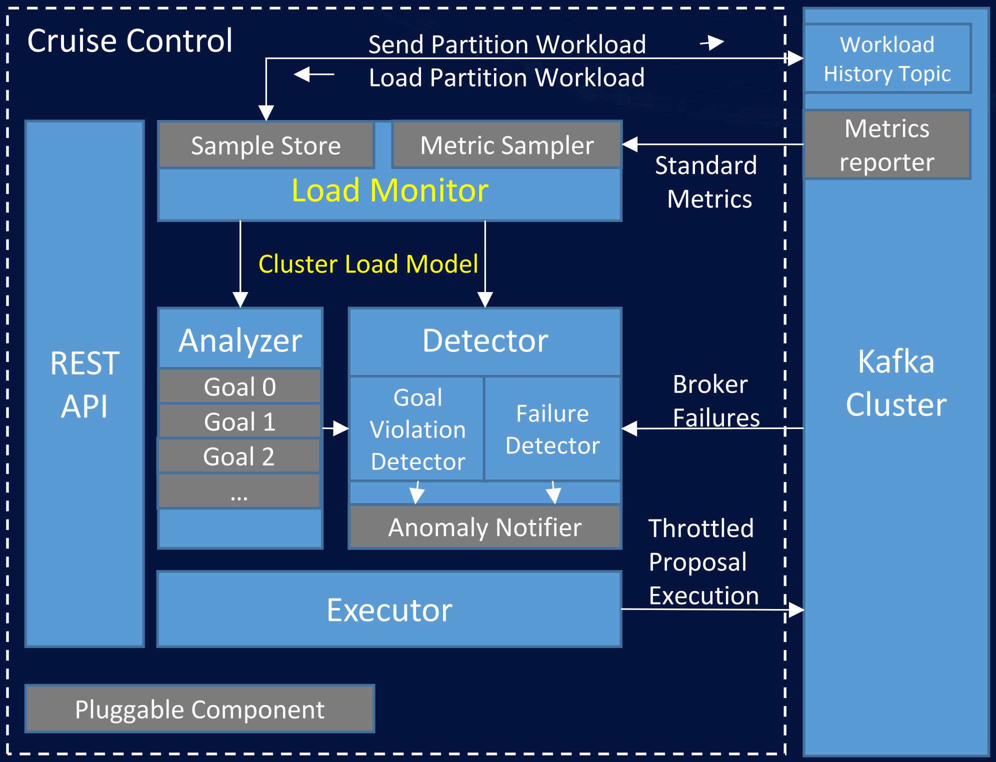
Cruise Control Architecture



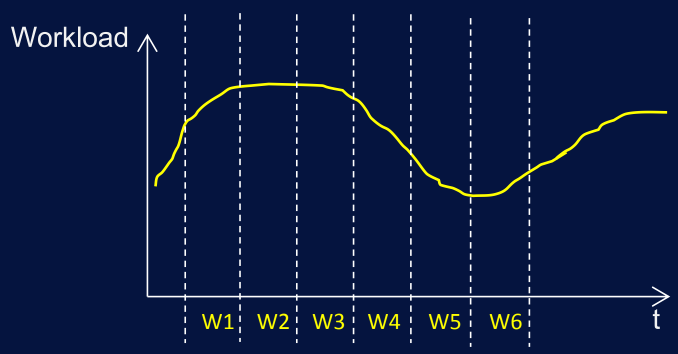
- Cluster Load Model
  - Topology
    - Replica distribution
    - Leader distribution
  - Workload
    - Granular workload (CPU, Disk, Network, etc.) per replica for each monitoring Window (e.g. an hour)



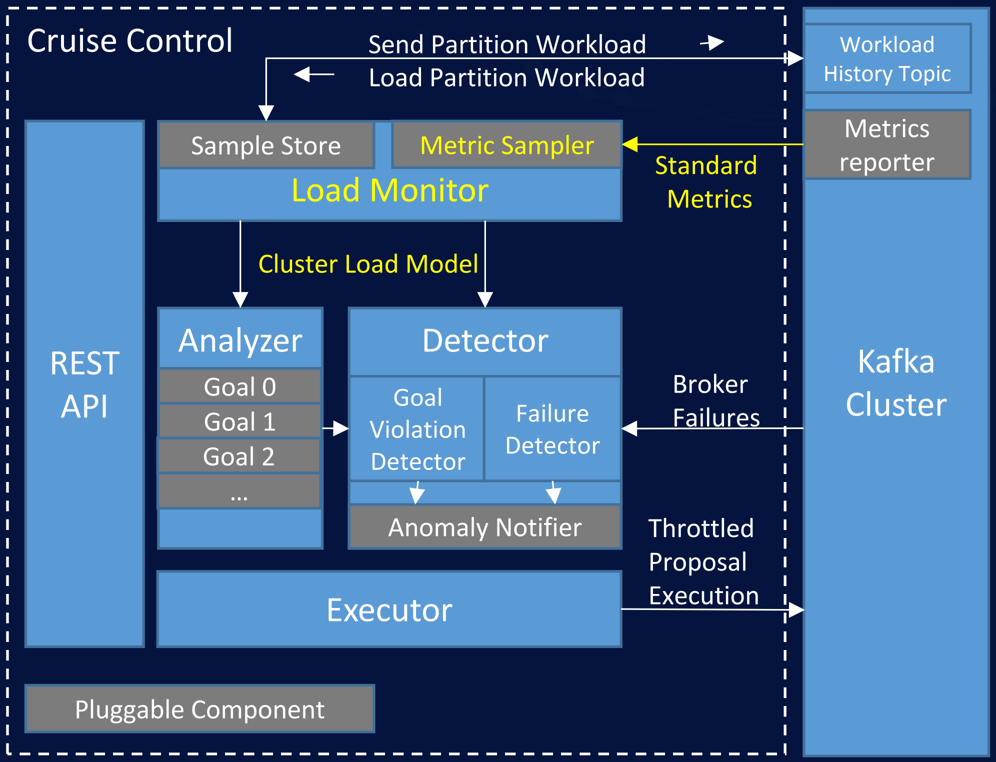
Cruise Control Architecture



- Cluster Load Model
  - Interface to simulate
    - Replica movement
    - Leader movement
- More about this later



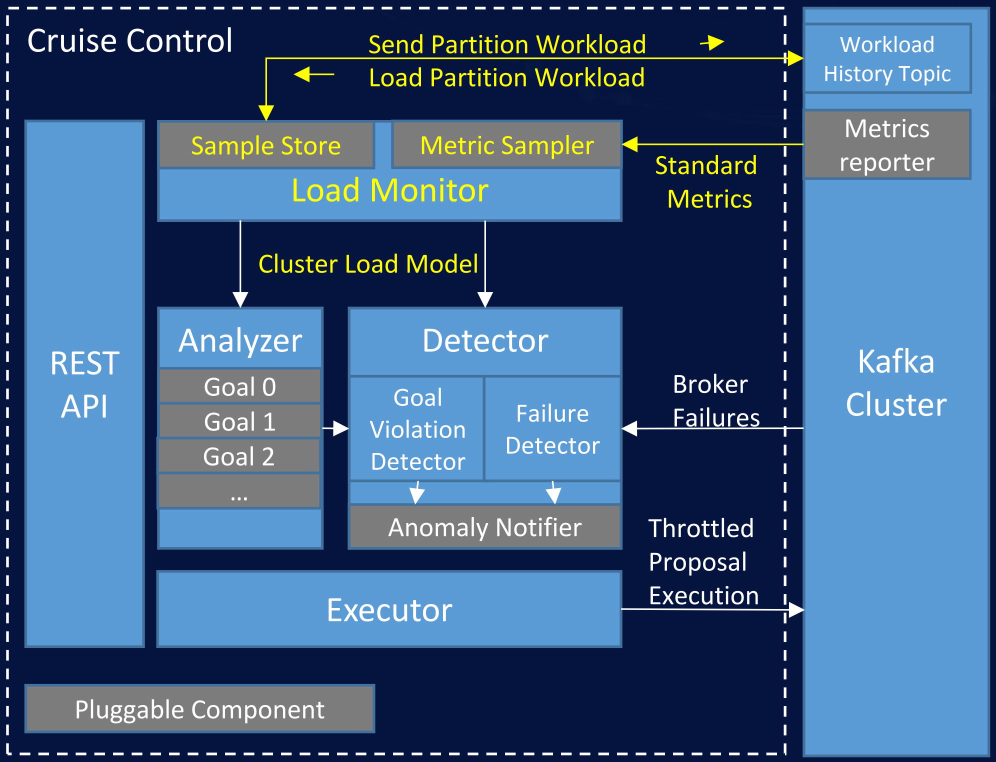
Cruise Control Architecture



- Load Monitor generates a Cluster Load Model to describe the workload of the cluster
  - Metric Sampler – Periodically (e.g. every 5 min) sample the cluster workload. By default read from the cruise control metrics topic.

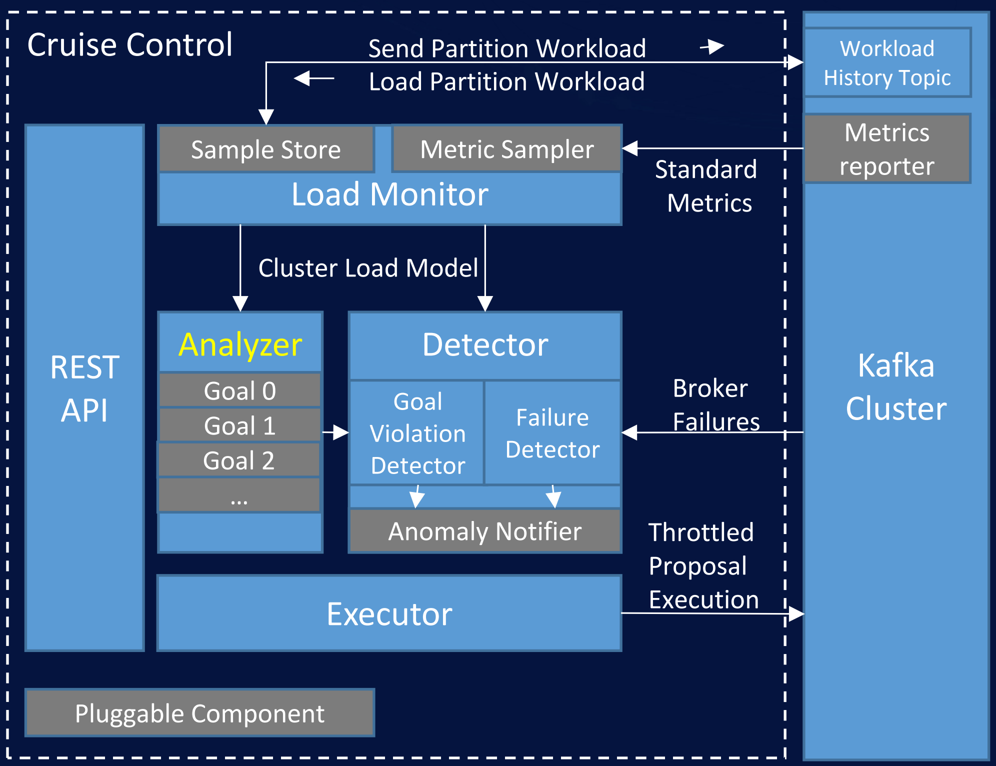
GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

Cruise Control Architecture



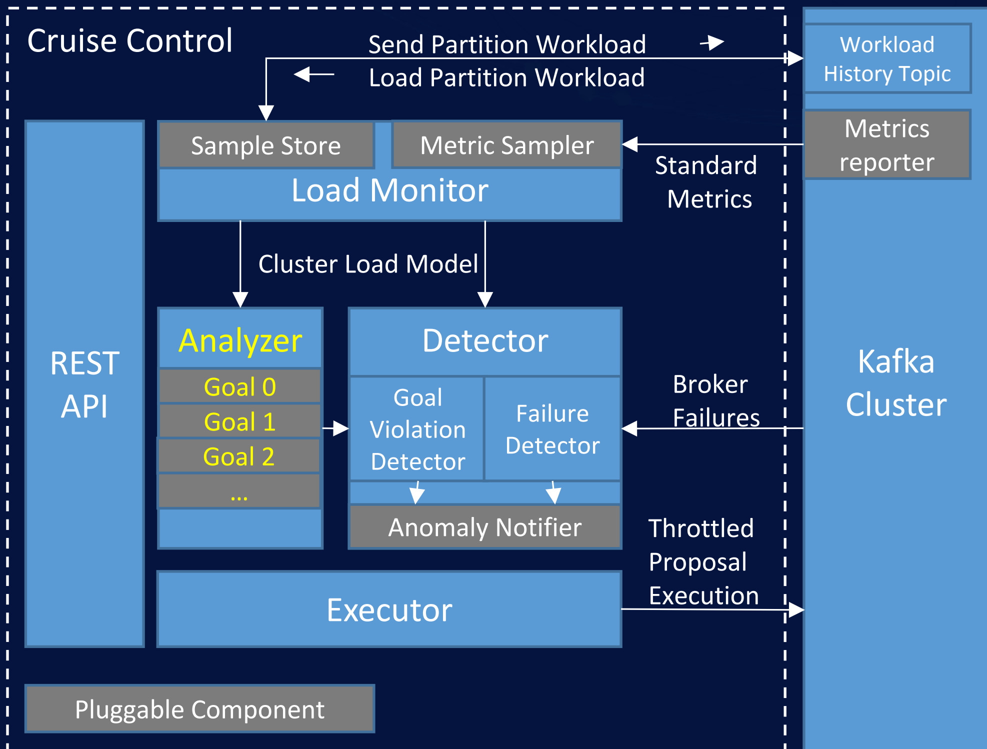
- Load Monitor generates a Cluster Load Model to describe the workload of the cluster
  - Metric Sampler – Periodically (e.g. every 5 min) sample the cluster workload. By default read from the cruise control metrics topic.
  - Sample Store – Save per partition workload to the workload history topic as backup for failure recovery.

Cruise Control Architecture



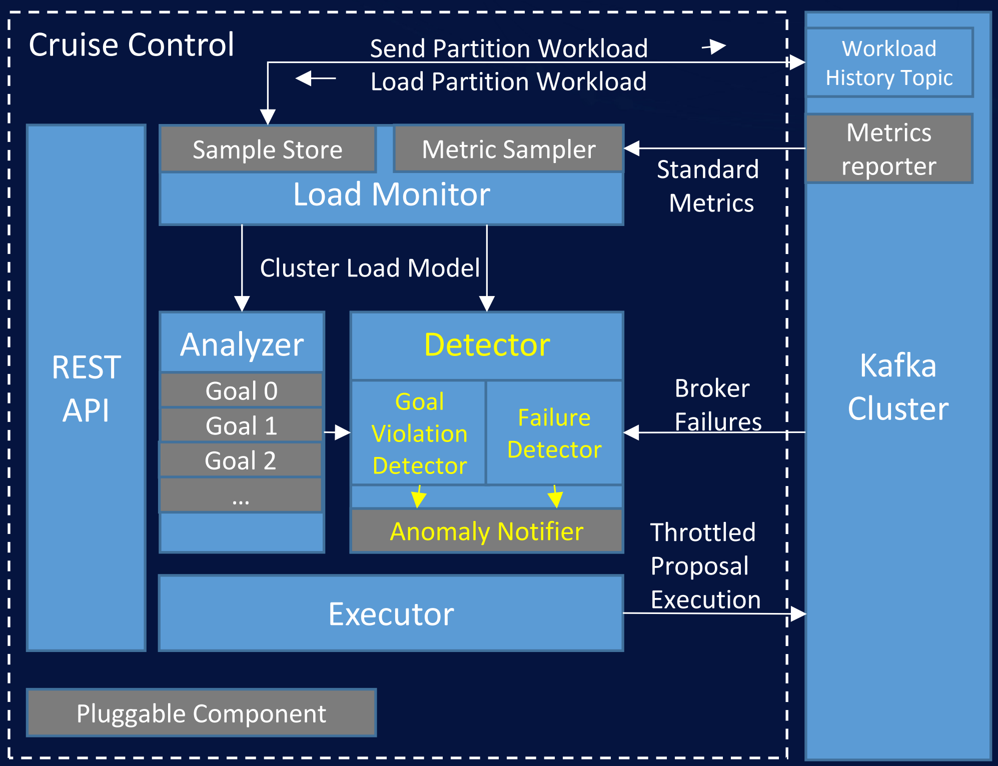
- Analyzer is responsible for generating optimization proposals to achieve pluggable goals.
  - Input – Cluster Load Model
  - Output – A set of optimization proposals (replica and leader movements)
  - Heuristic solution
    - Fast
    - Not globally optimal
    - But is usually good enough

Cruise Control Architecture



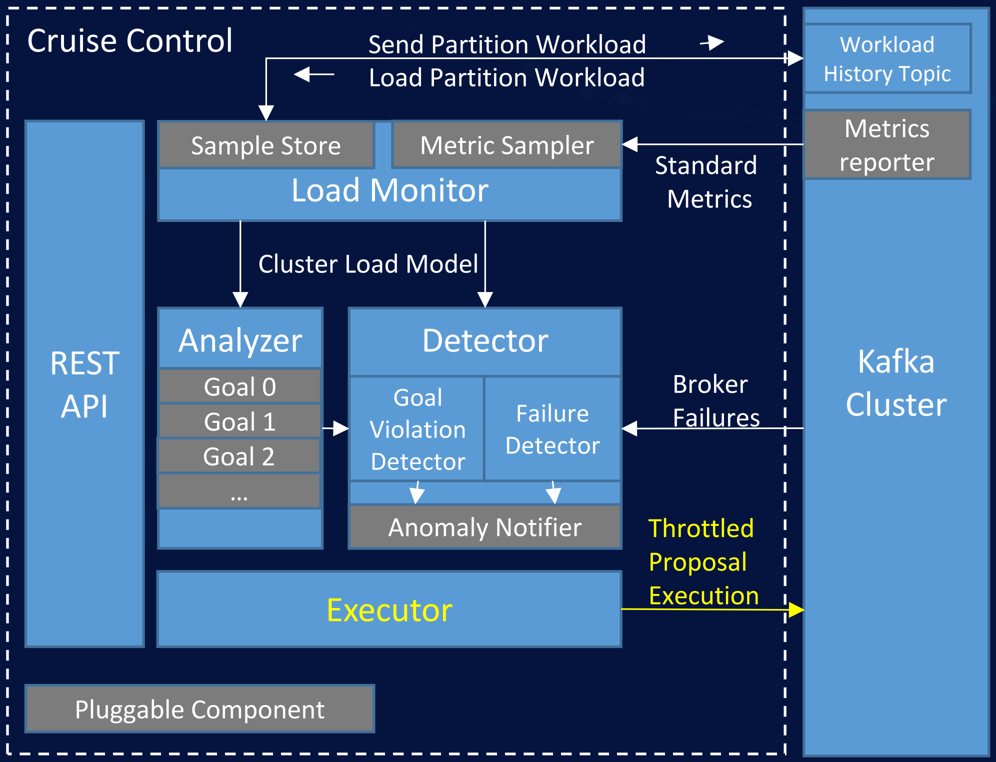
- Goals
  - Are pluggab
    - Impl. o
    - Easy to
  - With differe
    - High pr
  - Hard Goal o
    - Hard G otherw
    - Soft Go
- Some Examp
  - Rack Awar
  - Resource U
    - AVG ±

Cruise Control Architecture



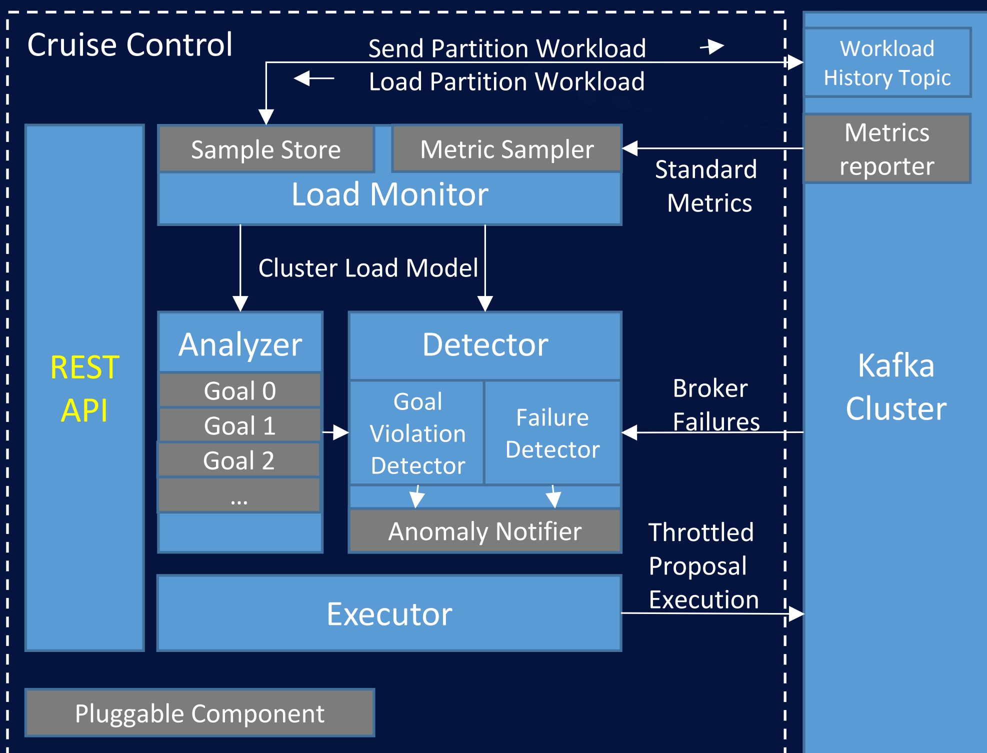
- Detector detects anomalies
  - Two types of anomalies
    - Goal violations
    - Broker failures
  - Anomaly Notifier
    - Notify
      - E.g. email
    - Action Decision
      - Fix
      - Delayed Check
      - Ignore

Cruise Control Architecture



- Executor carries out the proposals generated by the analyzer
  - The execution
    - Should not impact existing user traffic
    - Should be interruptible

Cruise Control Architecture



- REST API – User interaction
  - GUI is under development

Agenda

- Kafka introduction and terminologies
- Problems to solve
- **Our solution**
  - Cruise Control Architecture
  - **Challenges and Solutions**
- Insights and Generalization
  - Problem Generalization
  - Model Generalization

Challenges

- Trustworthy Workload Modeling (Workload Monitor)
- Complexity of Dynamic Workload Balancing (Analyzer)
- Fast Optimization Resolution (Analyzer)
- False Alarm in Failure (Failure Detector)
- Controlled Balancing Execution (Executor)
- And so on…

- See detailed discussion: https://www.slideshare.net/JiangjieQin/introduction-to-kafka-cruise-control-68180931

Challenges

- Trustworthy Workload Modeling (Workload Monitor)
  → Good Data
- Complexity of Dynamic Workload Balancing (Analyzer)
- Fast Optimization Resolution (Analyzer)
- False Alarm in Failure (Failure Detector)
- Controlled Balancing Execution (Executor)
- And so on…

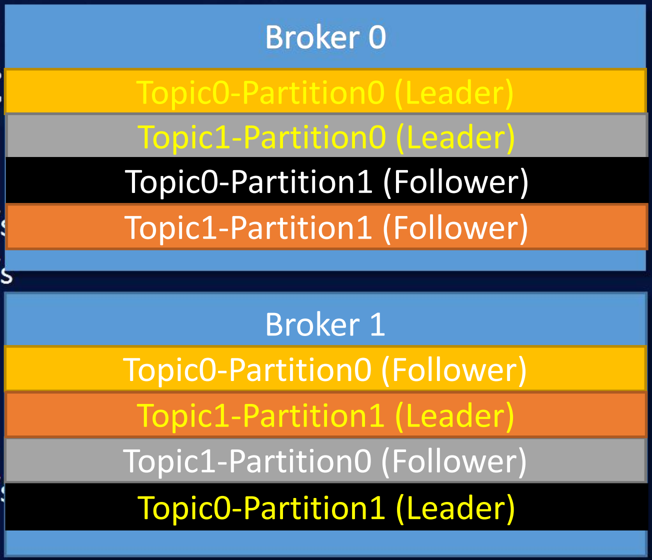
- See detailed discussion: https://www.slideshare.net/JiangjieQin/introduction-to-kafka-cruise-control-68180931

Trustworthy Workload Modeling

- Are metric samples accurate?
- Are there any missing metric samples?
- Are all the metric samples consistent with each other?

Trustworthy Workload Modeling

- Assuming everything is perfect
- Optimize CPU for the following c
  - Broker 0: CPU=80%
    - T0P0: DISK=1 GB, NW_IN=10 MB/s, NW_OUT=30 MB/s
    - T1P0: DISK=1 GB, NW_IN=20 MB/s, NW_OUT=40 MB/s
    - T0P1: …, T1P1: …
  - Broker 1: CPU=30%
    - T0P0: DISK=1 GB, NW_IN=10 MB/s, NW_OUT=0 MB/s
    - T1P1: DISK=2 GB, NW_IN=30 MB/s, NW_OUT=60 MB/s
    - T1P0: …, T0P1: …



- Move something from Broker 0 to Broker 1!
  - Let’s move the leader of Topic0-Partition0 to broker 1.
- What is the CPU utilization of Broker 1 after the move? Should we move more?

Trustworthy Workload Modeling

- Some metrics can be easily aggregated
  - E.g. Bytes In Rate, Bytes Out Rate, Messages In Rate, etc.
- Some metrics are difficult to “aggregate”
  - E.g. CPU, Memory, Latency
  - We need to estimate or predict

Agenda

- Kafka introduction and terminologies
- Problems to solve
- How do we solve it
  - Cruise Control Architecture
  - Challenges and solutions
- **Insights and Generalization**
  - **Problem generalization**
  - Model Generalization

Problem Generalization

- Rethink of the problem to solve
  - Given a topology
    - E.g. replica distribution, leader distribution
  - and associated metrics,
    - E.g. Partition Bytes In Rate, Partition Bytes Out Rate, Messages In Rate, Request Rate, etc.
  - optimize for some specific metrics
    - E.g. Broker CPU Usage, Broker DISK Usage, Broker Network IO Usage, Broker Memory Usage, Request Latency, etc.

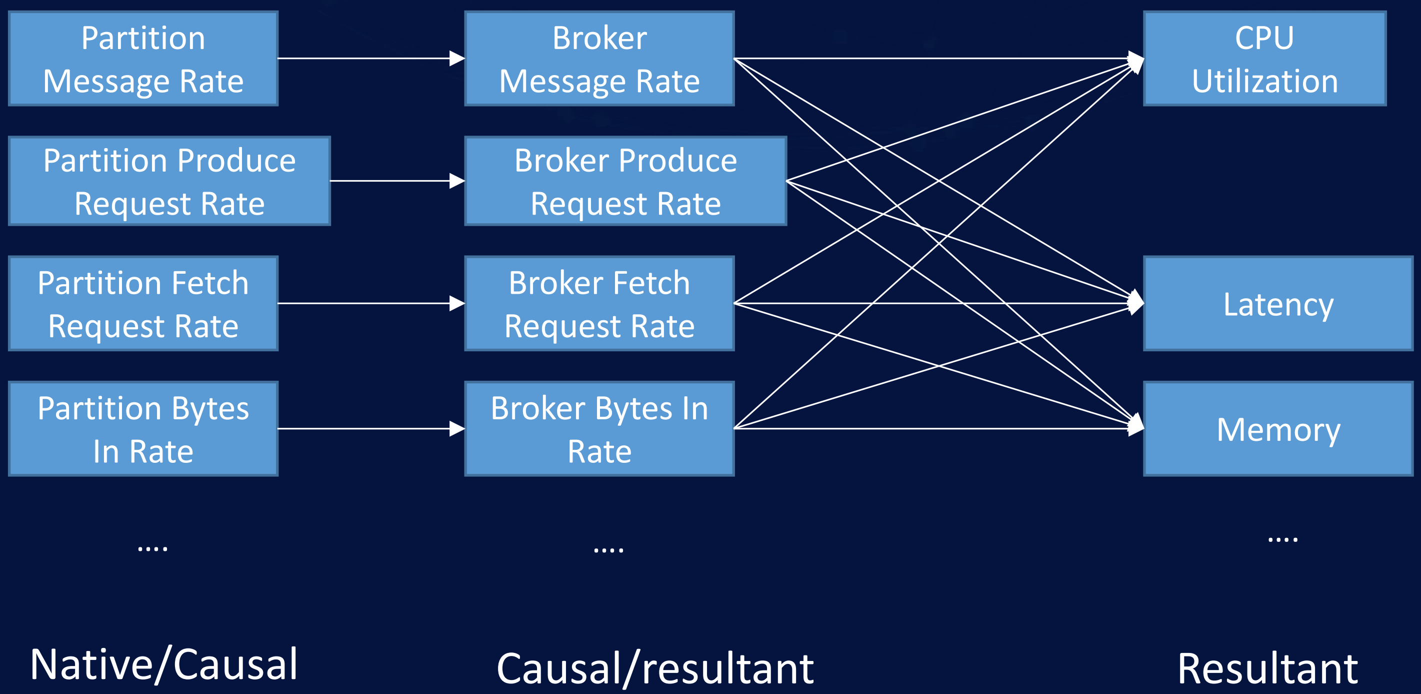
Understand the metrics – Native Metrics

- Some of the metrics are natural attributes of a given system
  - E.g. the partition bytes in is only drive by the applications (assuming not using quota)
- Those metrics are Native Metrics
  - The values cannot be changed
  - The distribution CAN be changed

Understand the metrics – Causal Relationship

- The causal relationship between metrics
  - Some metrics are *caused by* other metrics
    - E.g. Broker CPU utilization is *caused by* Broker Bytes In Rate, Broker Messages In Rate, Broker Bytes Out Rate, Broker Request Rate, etc.
    - E.g. Broker Bytes In Rate is caused by Partition Bytes In Rate.
  - The metrics that causes other metrics are Causal Metrics
  - The metrics that are caused by other metrics are Resultant Metrics

Metrics Dependency DAG



Understand the metrics – Causal Relationship

- The causal relation has different representations
  - Simple aggregation
    - *BrokerBytesInRate = AllPartitionBytesInOnBroker*
  - More complicated function
    - *BrokerCpuUsage = f(BrokerBytesInRate, BrokerBytesOutRate, ….)*

Understand the metrics – Causal Relationship

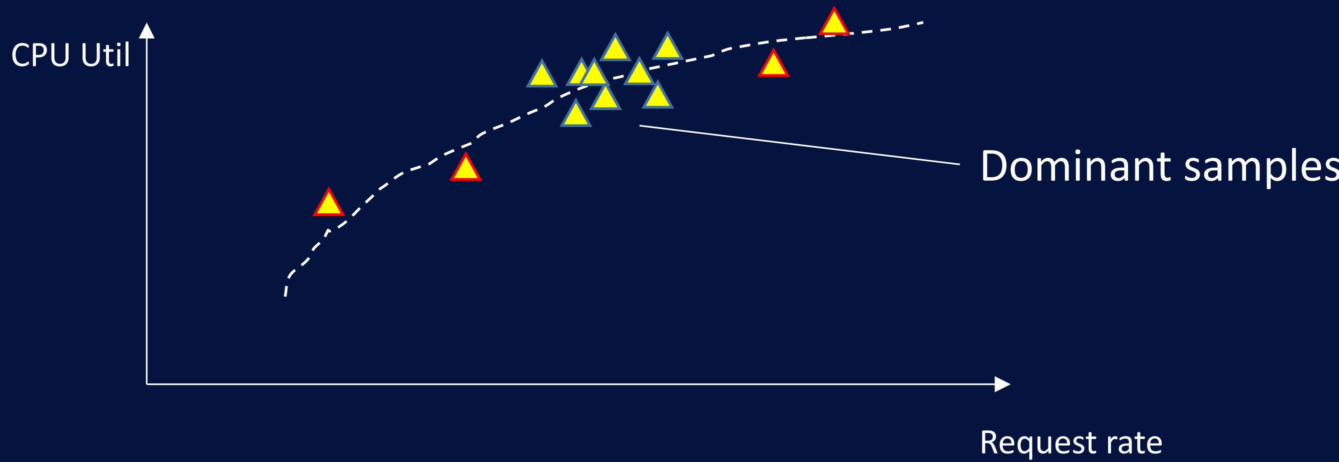
- Define the causal relationship with linear function:
  - Given Causal Metrics $CM_1, CM_2, \ldots$ and Resultant Metric RM

$$RM = a_0 + a_1 * CM_1 + a_2 * CM_2 + \ldots$$

- A polynomial function can also be used
  - Can still be achieved through linear regression
- Some more complicated model is also possible

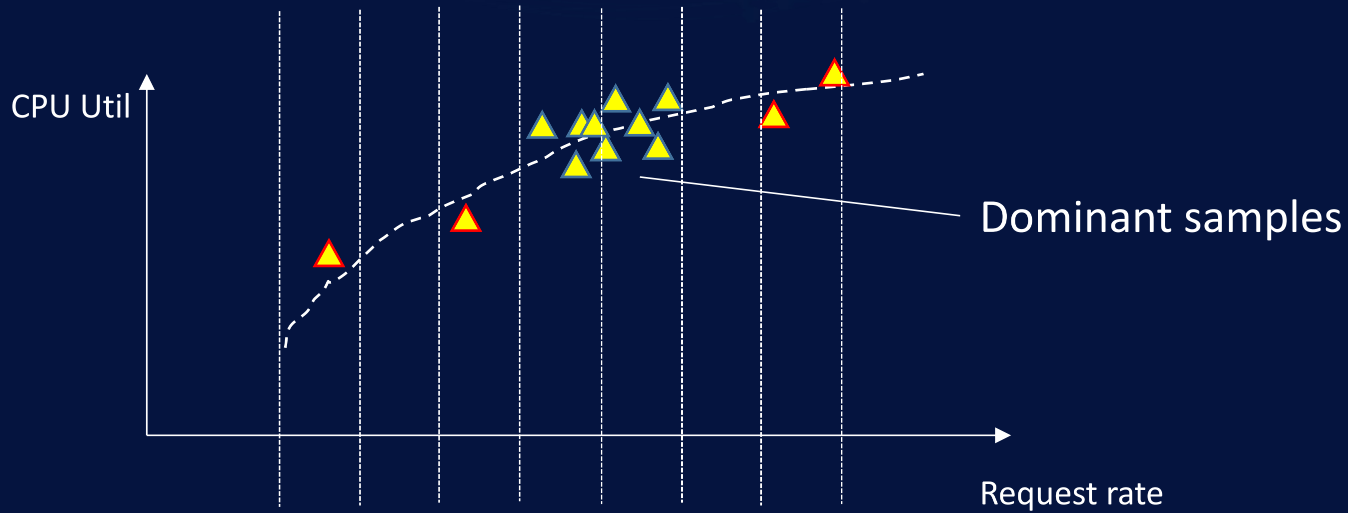
Metric Sample Selection For Regression

- A typical metric sample distribution



Metric Sample Selection For Regression

- A typical metric sample distribution



Metric Sample Selection For Regression

- Dependent metrics
  - E.g. Leader Bytes In and Replication Bytes Out are dependent

Understand the metrics

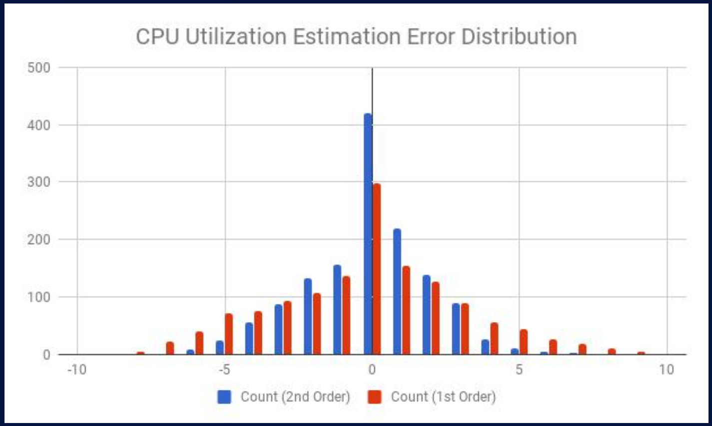
- Optimize CPU for the following case
  - Broker 0: CPU=80%
    - T0P0: DISK=1 GB, NW_IN=10 MB/s, NW_OUT=30 MB/s
    - T1P0: DISK=1 GB, NW_IN=20 MB/s, NW_OUT=40 MB/s
    - T0P1: …, T1P1: …
  - Broker 1: CPU=30%
    - T0P0: DISK=1 GB, NW_IN=10 MB/s, NW_OUT=0 MB/s
    - T1P1: DISK=2 GB, NW_IN=30 MB/s, NW_OUT=60 MB/s
    - T1P0: …, T0P1: …
- Move something from Broker 0 to Broker 1!
  - Let's move the leader of Topic0-Partition0 to broker 1.
- What is the CPU utilization of Broker 1 after the move? Should we move more?
  - Derive from the causal relationship function

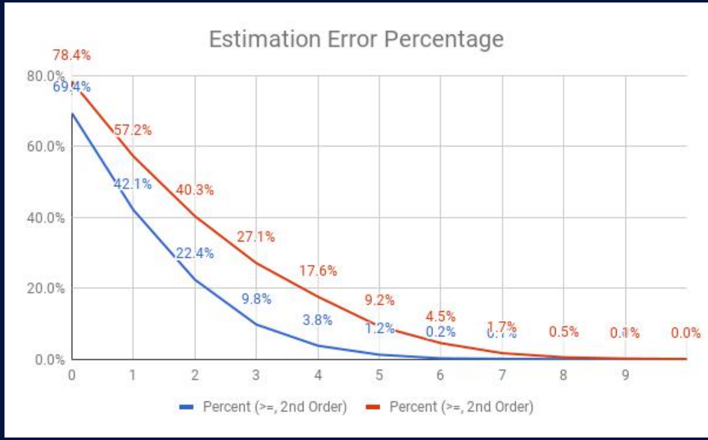
Problem Generalization

- Rethink of the problem to solve
  - Given a topology
    - E.g. replica distribution, leader distribution
  - and associated metrics,
    - E.g. Partition Bytes In Rate, Partition Bytes Out Rate, Messages In Rate, Request Rate, etc.
  - Optimize for some specific metrics
    - E.g. Broker CPU Usage, Broker DISK Usage, Broker Network IO Usage, Broker Memory Usage, Request Latency, etc.
  - By changing topology (causal metric distribution)
    - E.g. leader movement, replica movement
  - Or changing configurations (causal relationship function)
    - E.g. Compression type, caching policy, etc.

Resource Estimation Experiment

- CPU estimation
  - 0% - 35%
  - Synthetic traffic
  - Causal Metrics
    - LEADER_BYTES_IN
    - LEADER_BYTES_OUT
    - REPLICATION_BYTES_IN
    - REPLICATION_BYTES_OUT
    - MESSAGES_IN_RATE
    - PRODUCE_RATE
    - FETCH_RATE
- Online training and verification

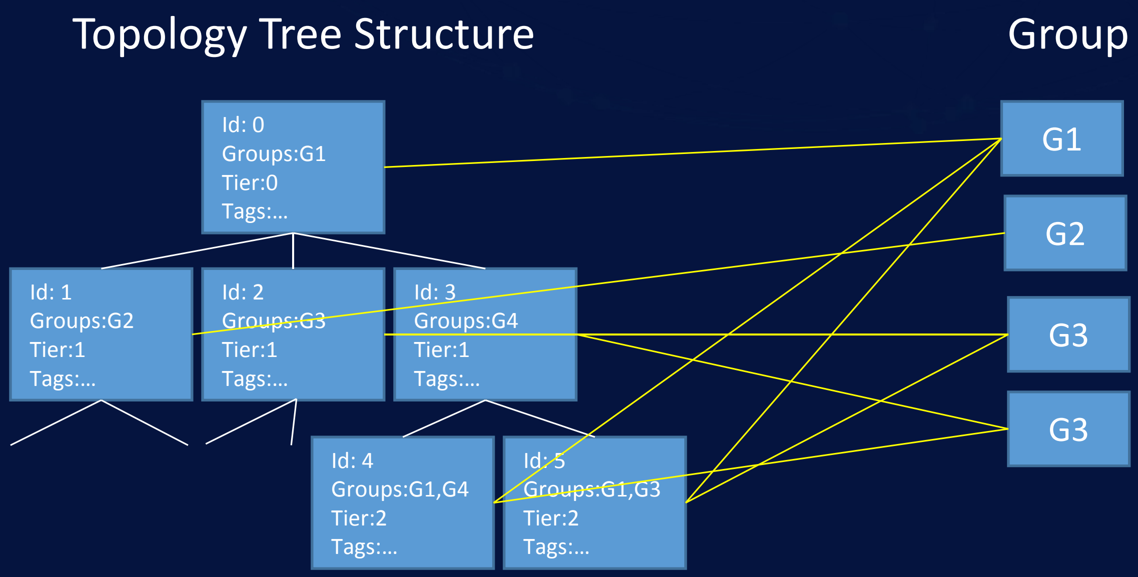




Agenda

- Kafka introduction and terminologies
- Problems to solve
- How do we solve it
  - Cruise Control Architecture
  - Challenges and solutions
- **Insights and Generalization**
  - Problem generalization
  - **Model Generalization**

Model Generalization

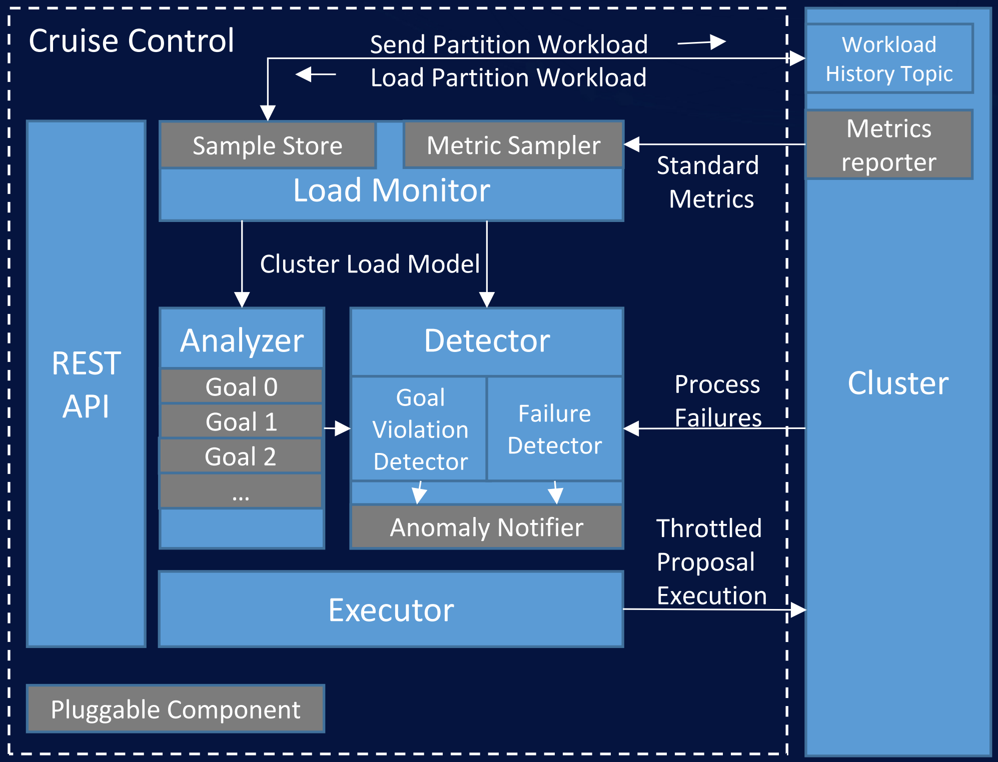


Arbitrarily group the nodes in the topology together.

Model Generalization

- Topology Tree
  - Physical hierarchy of the system
  - Ownership among nodes
- Group Registry
  - Logical grouping
  - Flat structure, no ownership
  - Quick access to a set of nodes
- Key-value based model, easy to scale.

Cruise Control Architecture



- The architecture is general enough
- The cluster load model will carry the causal relationship functions

Comparison to existing solutions

- Automatic cluster load balancing for stateful system
  - Cloud management system (Kubernetes, Docker, etc.)
    - Balancing by moving the entire process
    - Application unaware
  - Highly customized system (e.g. Microsoft Azure Storage)
    - Partial state movement
    - Tightly coupled with a specific system
  - Cruise Control
    - Application aware
    - Generalized distributed system model
    - Partial state movement
    - Estimation and prediction

Future Work

- Scalability
  - Currently everything is in memory
  - Would like to abstract the cluster load model to use a K-V based interface
- Integration with more projects (Apache Samza, Apache Helix, etc)
- Parallel computation on the optimization proposals
- GUI and multi-cluster management

Links

- Cruise Control
  - https://github.com/linkedin/cruise-control (github repo)
  - https://gitter.im/kafka-cruise-control/Lobby (gitter room for questions)
  - https://engineering.linkedin.com/blog/2017/08/open-sourcing-kafka-cruise-control (blog post)
- Other LinkedIn open source projects
  - https://github.com/linkedin/

Q&A