RFC: Apache Kafka Adapter for PA-RCA

AUGUST 2020

Introduction

The Performance Analyzer RCA (PA-RCA) is a framework that builds on the Performance Analyzer engine to support Root Cause Analysis (RCA) of performance and reliability problems in Elasticsearch clusters. It is modeled as a distributed data flow graph, with different nodes performing different root cause analysis from gathered performance analyzer metrics. Nodes subscribe to other nodes to consume their data (called `flowunits`) and perform their own processing on this data to produce insights.

The PA-RCA framework produces:

1. Deep fine-grained metrics - for different aspects of an Elasticsearch cluster.
2. RCA summaries – the output of analysis done by different nodes in the rca graph.
3. Problem specific decisions – Decider nodes in the graph consume all relevant RCAs pertaining to a problem area, apply a set of rules and create action recommendations.
4. Final action suggestions – Created by the decision maker framework (handles dampening, flip-flopping actions, conflicting actions etc).

In its current stage, fine-grained metrics are rendered via `perfTop`, a simple visualization tool that runs on the terminal.

Motivation

PerfTop and the PA/RCA APIs offer insights into various fine-grained metrics and dashboarding of those metrics in near-real time. These metrics are served from the metricDB files that are generated by the framework and managed locally on the Elasticsearch nodes. Often customers want to retain these metrics or rca outputs for a longer duration to do trend analysis or use it for forensic investigations. Sometimes there is also a need to create our own rule engine which consumes RCA data, PA metrics, decision maker outputs etc. One native way to achieve these goals is to build an agent that periodically polls the RCA APIs and gets the needed responses and pushes them to a persistent external store like S3 (in parquet format) or to another Elasticsearch cluster. This approach comes with additional infra management cost. Alternatively, we can externalize the metrics and rca outputs by pushing them out of the cluster into a message bus (such as Apache Kafka queues) which can then be consumed in various ways for historical analysis, setting up alerts etc.

Requirements

A User should be able to:

- Use the system to conveniently consume rca data and decisions.
- Get the latest updates on the changes happening in the cluster.
- Create additional (derived) metrics from the generated base metrics based on the needs.
- Retain the fine grained data and metrics on a persistent store of their choice.
- Optionally enable this feature to externalize the metrics and rca outputs.
- Set alerts on the rca outputs and decisions.

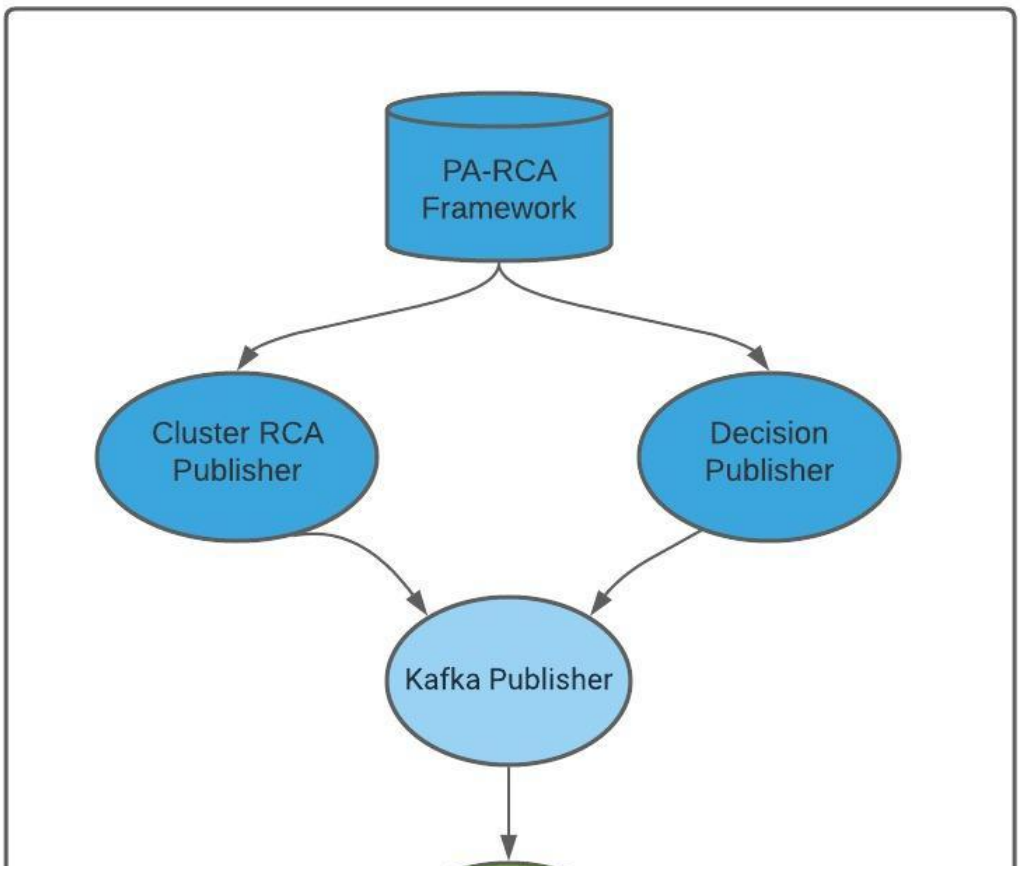
Proposed Design

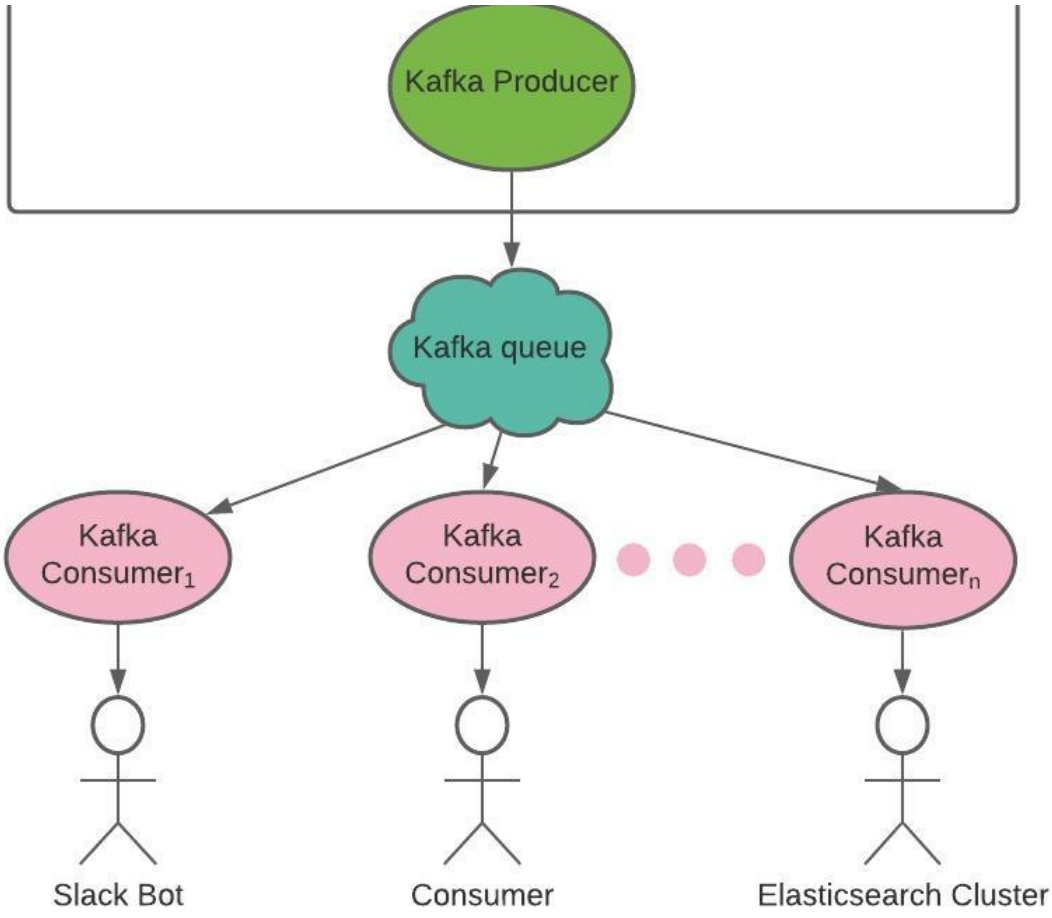
We propose to implement a Kafka adapter which periodically streams the RCA outputs and pushes them to a Kafka queue that can be configured by the user. Users can then build their own consumers on top of this queue to get the latest updates on the changes happening in the cluster. The adapter should be a push model that can persistently push the latest RCA cluster summaries and decisions into configurable message buses.

The introduction of Kafka queue opens up opportunities to consume the data in various ways

1. Data produced by pa-rca should be written on to a Kafka queue, which can then be used for consumption by different agents. Subscribers to the queue may be `Slack-bot` that notifies the dev ops team with recommendation.
2. RCA data and decisions can also be published via Kafka queue to different consumers. This will enable users to get notified on changes like high heap usage, along with the top consumers for heap.
3. Cluster state updates in ES are an important source of information for operational RCAs. These can be listened to and emitted by the rca framework.
4. Kafka queues will provide a path to ingest this data into other data sources or ES clusters for analysis.

Detailed Design





Key Design Components: Cluster RCA summary Listener, Decision Listener, Kafka producer, Kafka consumer.
We also provide a couple of reference implementations for the consumers - Slack bot, and Kafka-elasticsearch sink connector.

Cluster RCA Publisher: The cluster RCA publisher can listen to and store the latest cluster RCA updates. Plugins can be added to listen to the cluster RCA summaries and filter out the required information for consumption.

Decision publisher: The decision publisher can emit the final output stream of action recommendations to mitigate the root cause problem. Plugins can be added to decision publisher to listen to the latest action.

Kafka Producer: The Kafka producer can push formatted cluster RCA summaries and decisions into specific Kafka queue at the given interval.

Kafka Consumer: The Kafka consumer will read data from the Kafka queue and redirect data into two directions.

Slack Bot: The decisions data in the Kafka queue is first consumed by the Kafka consumer and then sent to a Slack bot via Slack's Webhooks URL. The Slack bot will send the message containing the decisions to the users that subscribe to the Slack channel.

Kafka-elasticsearch Sink Connector: The sink connecter can connect the Kafka queue and push the RCA data to other system such as local elasticsearch clusters. Once the RCA data is written into local elasticsearch cluster, users can then query the data and conduct analysis via the elasticsearch API.

Use Cases

- Customers can query RCA data and conduct analysis or extend the PA-RCA framework on their own elastic search clusters.
- Edit configurations in a configuration file including Kafka bootstrap server, topic, and interval (larger than the minimal value) to receive RCA data. These configurations will be used for Kafka producer setup.
- Customers can use the Webhooks URL of their own Slack application in the Kafka adapter to periodically receive decision on clusters from the Slack bot.

Appendix

Open source components used to build the tool

- Apache Kafka
- Slack API
- Confluent Kafka-Elasticsearch Sink Connector