**Interim Report – Effectiveness of Rebalancing a Kafka Cluster**

**Introduction**

In the modern era, almost every organization generates a massive amount of data every day. These data can vary from simple user end actions like logins, likes, comments, clicks and search, to more operational and system metrics data such as latency, errors, CPU usage, memory, network usage, to name a few. The production of these data and the requirement for real-time analysis on this data as per the organizational needs creates a significant challenge for the systems to cope with these sheer volumes of data and provide optimal performance daily. (Romero and Oliveira, 1989)( Kafka: a Distributed Messaging System for Log Processing)

## One of the critical methods to manage these massive data in a distributed approach is to use the publish/subscribe messaging interaction paradigm. (Dobbelaere and Esmaili, 2017). In this publish/subscribe messaging system. the publisher categorizes the message somehow, and the recipient (subscriber) signs up to receive specific types of this communications. A broker, or a central location that broadcasts the messages, is standard in pub/sub systems. (Walkenbach, 2010). Apache Kafka and RabbitMQ are the two most commercial open-source pub/sub systems used widely for managing the vast amount of data in the industries for the last couple of years. (Dobbelaere and Esmaili, 2017). Between these two, Apache Kafka showed superior performance when experimentally verified (Kafka: a Distributed Messaging System for Log Processing) and can even process millions of data per second. (Goodhope *et al.*, 2012)

## Apache Kafka is a distributed messaging platform capable of very high performance and horizontally scalable. (Thein, 2014) In Kafka, Topics stores the messages that get replicated and distributed across the Brokers, which form the cluster. (Walkenbach, 2010). In an ideal scenario, for a balanced cluster, the Topics are evenly distributed amongst the Brokers. However, in practicality, due to faulty design of the system architecture, unpredictable workloads or addition/removal of brokers, the cluster could become unbalanced. Tools exist to rebalance a Kafka cluster by switching which Topic replica is the master or moving replicas to under-utilised Brokers. (Kumar, 2018) The latter operation of moving replicas can be expensive and take hours to migrate data between brokers. The cluster operates at reduced capacity because the rebalance utilises the I/O and network bandwidth. (Qin, 2016)

**Aim and Objectives**

## The high-level aim of the project is to investigate two things on Kafka which will be running on Kubernetes. Firstly, what are the optimal values for tuning a rebalance of a Kafka cluster to trade off speed versus impact? Secondly, a stretch goal is to look at the effects of rebalancing on a running cluster. For example, how long will it take? Will this have an impact on the current workload? What are the financial ramifications of charging for some data transfers but not others?

Furthermore, these are the objectives of the project to achieve the aims mentioned above.

**Lab Setup**

## A laboratory will host the test rig comprising the infrastructure, libraries, installable, and monitoring tools to carry out the experimental study.

## 

## Basic Metrics Gathering

## 

## Following the lab setup, a balanced Kafka cluster will be created, and basic system and application metrics will be extracted from the monitoring tools. These metrics shall be the baseline for the entire experiment.

**Unbalanced Cluster Creation**

The existing cluster will be purposedly unbalanced, and the resultant metrics will be collected for comparing it with the metrics gathered from the balanced cluster.

## 

**Implementation of Cruise Control**

Various Cruise Control configuration execution on top of an unbalanced cluster will be performed to rebalance it and obtain the corresponding metrics.

**Tuning**

Fine-tune the rebalance by adjusting the parameters, and the best settings will be obtained to help estimate and calculate trade-offs.

**Data Analysis**

## Data Analysis will be carried out by gathering all the system and application metrics data such as CPU, Memory, Network I/O, Consumer Lag and Consumer Group Lag, to name a few, in the baseline, unbalanced, cruise control and tuning stages. Moreover, using the results of this data analysis, estimations and conclusions will be made about the project's second aim, as mentioned above.

**Progress Overview**

Since this project work is being carried out as part of the Industry MSC project for Red Hat, the supervisors at Red Hat suggested few reading materials for getting acquainted with the world of Apache Kafka.

Extensive research was carried out to understand the concepts of Kafka and its working. Firstly, why was Kafka needed and what business challenges made the creation of Kafka a necessity was explored. Following which the critical concepts of Kafka such as installation requirements, Kafka Producers, Kafka Consumers, Kafka Brokers, language in which Kafka is written and general administration of a Kafka cluster was explored. Furthermore, the critical concepts of reliable data delivery and how to achieve it in Kafka were also researched. One of the critical resources helping in this research was the official Kafka guide - (Walkenbach, 2010), written by the creators of Kafka themselves, which provided the complete understanding of Kafka, from the basics to the advanced level. Moreover, the utility of Zookeeper (Hunt et al., 2019) in running a Kafka cluster was also investigated.

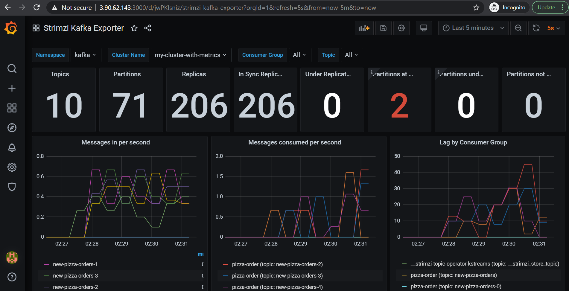
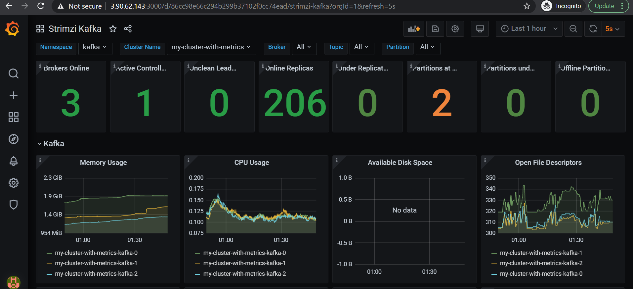
Besides the understanding of Kafka, the impact of real-time workload and the resultant rebalance mechanism was also explored. Various methods used for rebalancing were studied, out of which the most appropriate method- the Cruise Control created by LinkedIn (Qin, 2016) was selected for our experimentation purpose in the project.

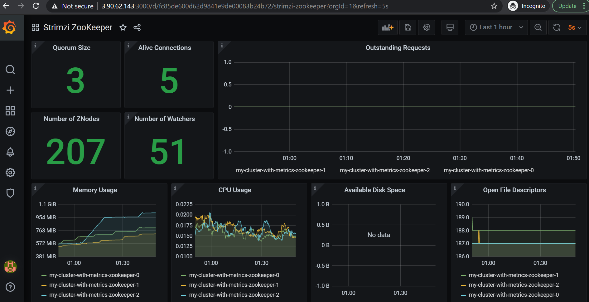
Additionally, the Linux foundation entity Strimzi using which the entire project activity was to be carried out as suggested by the Red Hat team, was also investigated.

Once the initial research was carried out for Kafka, the next stage of the project work was undertaken: the setup of Lab for experimenting. At first, an AWS cloud EC2 instance was selected to setup the infrastructure for the test rig. The required installation files, python libraries were installed in that instance. Moreover, as it was suggested by the Red Hat team to make use of Strimzi – Kafka on Kubernetes concept to experiment; therefore, a Minikube (Moilanen, 2018) was installed in the instance. Inside this Minikube, a Kafka cluster was set up comprising a broker, bootstrap server, zookeeper, entity operator, and cluster operator by following the official Strimzi documentation of Kafka.

Due to Pythons easy readability and coding capability, it was decided as the programming language for the application development for this project. Initially, a primary hello world type of Python application was created to understand the flow of the producer-consumer model of Kafka on the freshly installed Kafka cluster inside Minikube. Following the successful understanding of this basic flow, a more complex pizza ordering python application was created, creating unlimited orders for pizza across various outlets and sending the respective order information from the producer to the consumer via a broker for consumption. Furthermore, this application was converted using the docker containerisation technique to include the multiple producers and multiple consumers, sending and receiving message simultaneously. Therefore, creating a real-world data flow mechanism as desired in a balanced Kafka cluster.

Moreover, for gathering the metrics mentioned in the objectives above for different project activities, the Prometheus-Grafana monitoring solution was also installed inside the Minikube and configured with the Kafka cluster. This activity concluded the Lab setup work for our project. Below mentioned figures show the successful setup of the Lab and the primary data observed for the test rig from that.



Currently, work is in progress to introduce scalability to the containerised application to proceed to the next stage of the project that gathers metrics from a balanced cluster involving multiple producers and consumers to create a baseline for the experiment. The following is depicted in the project plan provided below.

**Project Plan**

The following PDF files shows the complete Project Plan in a Gantt-Chart format. Alternately, the Gantt Chart for the plan can also be viewed via the public link provided in the Appendix section.



**References**

Kumar, I. (no date) ‘Autonomous workload rebalancing in Kafka’.

Romero, C. and Oliveira, H. P. (1989) ‘Exact solutions in brans-dicke theory: A dynamical system approach’, *Astrophysics and Space Science*, 159(1), pp. 1–9. doi: 10.1007/BF00640482.

**Appendix**

<https://sharing.clickup.com/g/h/69qng-120/06dbf6b36741b4a>