CISC 5597/6935 Distributed Systems

Background:

We learned the different components of a distributed system. In this lab, we are going to practice ourselves with the existing open-source systems. Given a system with comprehensive APIs and configurations, we deploy it on a distributed cluster and explore its applications as well as performances.

Option 1 (Evaluation): Benchmarking Kafka with Zookeeper and Kraft

Apache Kafka was originally shipped with Zookeeper as its distributed coordinator (e.g., consensus algorithms). In the latest release, it replaced Zookeeper with Kraft. In this lab, you are going to design a simplified benchmarking tool to evaluate Kafka with both Zookeeper and Kraft.

Requirements:

1. Deploy Kafka on your 3-node cluster. You can follow our tutorials on GitHub or any other online tutorials. However, it has to be a cluster deployment, not a local setting.
2. Design a producer or any tools to evaluate the throughput performance with different settings.
    a. Number of partitions
    b. Number of replicas
    c. Number of topics
    d. Different batch size
3. Conduct experiments with both Zookeeper and Kraft
4. A comprehensive report with your findings.

References:

1. https://www.clairvoyant.ai/blog/benchmarking-kafka
2. https://openmessaging.cloud/docs/benchmarks/
3. https://www.confluent.io/blog/why-replace-zookeeper-with-kafka-raft-the-log-of-all-logs/
4. https://www.instaclustr.com/blog/apache-kafka-kraft-abandons-the-zookeeper-part-1-partitions-and-data-performance/

Option 2 (Development): Integrating Kafka with external systems (e.g., X, or any other content platforms)

In this lab, you are going to play with the APIs from X (or any other content platforms) and Kafka. You can use any online tools, such as confluent-hub and Aiven.

Requirements:

1. Your system must be built on a cluster, not a local machine. You can use any third-party docker images.
2. Select trending topics from the link to fetch the data. As our cluster is quite small, you don't have to integrate all the data. And you can choose any other content platform.
3. Store the data in Kafka (e.g., producer).
4. Develop consumers to get the data from Kafka.

Reference:

1. https://blog.twitter.com/engineering/en_us/topics/insights/2018/twitters-kafka-adoption-story
2. https://blog.twitter.com/engineering/en_us/topics/infrastructure/2021/processing-billions-of-events-in-real-time-at-twitter-
3. https://aiven.io/blog/connecting-twitter-to-aiven-for-apache-kafka
4. https://www.confluent.io/hub/
5.

Option 3 (Maintenance): Monitoring a Kafka Cluster

Managing a cluster in real-time is an important task. Apache Kafka has several open-source monitoring tools, such as Kpow, Burrow, and Prometheus. In this lab, you are going to integrate a monitoring tool into your Kafka cluster and develop testing programs.

Requirements:

1. Deploy a cluster version of Kafka.
2. Ingrate a monitoring tool – choose any tool you want. Kpow and Burrow are preferred.
3. Develop producers and consumers that submit workloads to the system (you can find the code online).
4. Monitor your cluster with the tool that you installed, and you should see the workloads from the monitoring tool that you developed in Step 2.

References:

1. https://github.com/factorhouse/kpow
2. https://kpow.io/
3. https://docs.kpow.io/config/kafka-connect/
4. https://www.entechlog.com/blog/kafka/monitoring-kafka-consumer-lag/
5. https://github.com/linkedin/Burrow
6. https://medium.com/@prasanta.mohanty/kafka-monitoring-using-burrow-f338fef63792

Option 4: You can also name your own project by submitting a half-page proposal. You will have to talk to me before Monday, Dec. 9th.

Submission:

The maximum group sizes are 2 (5597) and 2 (6935). The submission should include the following:

1. Complete the code (including the code that you used from online articles).
2. A comprehensive report detailing every step performed.
3. A recorded screen video for a demonstration.