Insper - Megadados

Explicando e Explorando o Kafka
Ferramenta de streaming de dados.

Alunos: Antonio Andraues, Gabriel Francato e Raphael Azevedo

São Paulo – SP

Sumário

Introdução

O projeto visa como objetivo principal explicar as principais aplicações do Kafka, bem como instalar e utilizar essa ferramenta muito importante para a organização de grandes fluxos de dados em aplicações com diferentes propósitos.

Um dos grandes problemas que surgiram junto com a big data foi o problema de armazenar esses dados, os bancos geralmente têm dificuldade em lidar com grandes fluxos de dados e muitas vezes é necessário criar uma pipeline que irá organizar e definir qual a sequência e qual a ordem que cada operação será executada, assim a perda de dados pode chegar a zero e além disso fica muito mais fácil gerenciar o fluxo de dados da sua aplicação.

O Kakfa é o software desenvolvido incialmente pela Linkedin e tem como principal funcionalidade justamente o gerenciamento do fluxo de dados antes deste encontrar o banco de dados.

O que é o Kafka?

Apache Kafka é uma plataforma open-source de processamento de streams desenvolvida pela Apache Software Foundation, escrita em Scala e Java. Inicialmente Kafka foi desenvolvido como uma ferramenta interna do LinkedIn para cuidar dos seus Logs, posteriormente se tornando opensource.

Nas palavras da própria Apache Kafka em seu site: “Apache Kafka é uma plataforma distribuída de mensagens e streaming”. Sendo uma plataforma distribuída, Kafka é projetado para rodar em mais de uma máquina, e aproveita disso para utilizar o poder de processamento e armazenamento extra das máquinas para mover e transformar grande volumes de dados.

Conceitos

Para entender o que é Kafka, vamos analisar os conceitos por trás de seu funcionamento, procurado entender suas partes independemente.

Mensagens

Mensagens são o principal recurso do Kafka, todo dado que trafega o fluxo do Kafka é uma mensagem. Mensagens são enviadas por producers a um Kafka Cluster, e são categorizadas em diferentes tópicos. Mensagens no Cluster podem ser resgatadas por consumers interessados nos tópicos correspondentes. Uma mensagem pode ser qualquer coisa, desde uma string simples até um arquivo JSON.

Tópicos

Um Tópico é como mensagens são categorizadas dentro do Kafka. Toda mensagem enviada ao Kafka permanecerá em um tópico, e sempre ordenada. Para garantir essa ordenação, tópicos são particionados em n partições. Quando um tópico recebe uma mensagem, o Kafka aloca essa mensagem pra partição adequada dependendo de sua key (chave). Isso garante que mensagens com a mesma key fiquem na mesma partição, dando ordem as mensagens produzidas.

Producer

Um Producer é aquele responsável por enviar mensagens a um tópico. De forma simples, um producer *produz* mensagens e as publica em um tópico relevante, com uma key qualquer associada a essa mensagem. A mensagem produzida é ordenada na partição do tópico de acordo com sua key, como mencionado no conceito anterior.

Consumer

Um consumer no Kafka é quem lê as mensagens dentro de um tópico. Apesar do que o nome indica, o consumer não realmente *consome* a mensagem, pois ela ainda permanecerá no tópico após ser lida.

É possível ter vários consumers no mesmo tópico, e cada um indica a posição em que parou de ler. Assim, podemos ter diversos consumers em diferentes pontos de um tópico, realizando ações diferentes. Ou, se desejar, pode até ter um grupo de consumidores em um mesmo tópico na mesma partição, onde eles leem mensagens com um offset entre suas posições, para poder consumir o tópico com muita mais agilidade.

Apache Zookeeper

Zookeeper é um serviço centralizado também feito pelo Apache. Ele fornece diversos serviços que são de alguma forma utilizados por aplicações distribuídas. O Kafka utiliza do Zookeeper delegando funcões a ele. Se o Kafka não for capaz de realizar uma funcionalidade específica, ele a redireciona para o Zookeeper, que faz uso de seus serviços para cumprir o que precisa.

Kafka Brokers | Kafka Clusters

Um Kafka Broker é como uma instância (uma máquina). Um Kafka Cluster é conjunto de um ou mais instancias de Brokers. Se rodarmos o Kafka localmente, criamos um Broker na nossa máquina, que formaria um Kafka Cluster de um único Broker.

Como mencionado anteriormente, o Kafka é uma aplicação distribuída, ou seja, feita para rodar em várias máquinas. De tal forma, para escalar nossa aplicação, basta instanciar mais **n** Brokers no mesmo Cluster. Isso seria equivalente a adicionar a capacidade de processamento e armazenamento de **n** máquinas a esse Cluster.

Contexto de Aplicações

Kafka já se tornou uma opção popular para empresas que precisam processar streaming de dados ou transportar dados de um sistema para o outro. Empresas como o Netflix, Uber, Spotify, já usam Kafka em partes de sua stack.

É uma ferramenta muito útil para uma série de atividades, são elas:

Serviço de Mensagem: Milhões de mensagens podem ser enviadas e recebidas em tempo real pelo Kafka.

Processamento de dados em Tempo Real: Kafka pode ser utilizado para processamento contínuo de streaming de dados em tempo real.

Gerenciamento de Logs: o Kafka pode ser utilizado para coletar diversos logs de diferentes locais e armazena-los em uma única localização.

Gerador de Comentários para Log: Pode ser utilizado para como uma ferramenta externa para comentar sistemas distribuídos.

Gerenciamento de Eventos: Uma sequência de eventos ordenados no tempo podem ser organizados, gerenciados e processados pelo Kafka.

Manual de Instalação e Utilização

Basicamente para instalação seguiremos o passo a passo disponibilizado em inglês no site: https://kafka.apache.org/quickstart

Windows:

Onde denotar `bin/` no tutorial de Linux basta trocar por `bin\windows\` e mudar a extensão dos arquivos de `.ssh` para `.bat`

Linux/Mac-os - Unix-based:

O primeiro passo é o de baixar e extrair os binários. Para isso basta executar os seguintes comandos:

wget https://www-us.apache.org/dist/kafka/2.3.0/kafka_2.12-2.3.0.tgz
tar -xzf kafka_2.12-2.3.0.tgz
cd kafka_2.12-2.3.0

Como dito anteriormente o Kafka utiliza o Zookeeper como gerenciador... Então antes de inicializar um server Kafka é necessário inicializar o server do Zookeeper.

bin/zookeeper-server-start.sh config/zookeeper.properties

```
INFO Reading configuration from: config/zookeeper.properties
(org.apache.zookeeper.server.quorum.QuorumPeerConfig)
…
INFO binding to port 0.0.0.0/0.0.0.0:2181
(org.apache.zookeeper.server.NIOServerCnxnFactory)
```

(resposta esperada)

A porta mostrada acima pode vir a ser diferente da sua. Devido a escolha interna pelo Zookeeper.

Este comando basicamente começa um server com as configurações fornecidas no arquivo config/zookeeper.properties assim como o comando a seguir.

bin/kafka-server-start.sh config/server.properties

Porém este inicializa o server do Kafka.

```
[2013-04-22 15:01:47,028] INFO Verifying properties
(kafka.utils.VerifiableProperties)
[2013-04-22 15:01:47,051] INFO Property socket.send.buffer.bytes is
overridden to 1048576 (kafka.utils.VerifiableProperties)
```

(resposta esperada)

Agora com os dois terminais abertos podemos ir a uma demonstração de uso:

Criação de um tópico teste:

bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic teste

Listando para ver se o tópico foi realmente criado:

bin/kafka-topics.sh --list --bootstrap-server localhost:9092

```
test
```

Mandar mensagens para o server (producer):

bin/kafka-console-producer.sh --broker-list localhost:9092 --topic teste

```
Teste123
123
```

Consumindo as mensagens (consumer):

bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

```
Teste123
123
```

Caso queira só mensagens ainda não visualizadas basta tirar o argumento - -from-beginning. Assim caso tire este argumento e rode novamente o mesmo comando acima a resposta deve ser vazia.

Mais detalhes:

Caso queira ver com mais detalhes e/ou queria visualizar um exemplo pelo qual forcamos a visualização de dados de uma replica visualize este link:
https://kafka.apache.org/quickstart#quickstart_multibroker

Código Demo:

https://github.com/gabrielvf1/Megadados2019-Projeto2/

Nele será possível ver uma aplicação real implementada com a biblioteca Kafka-python.

Referências

https://medium.com/@gabrielqueiroz/o-que-%C3%A9-esse-tal-de-apache-kafka-a8f447cac028
https://dzone.com/articles/what-is-kafka
http://blog.adnansiddiqi.me/getting-started-with-apache-kafka-in-python/
https://kafka.apache.org/quickstart