Advanced Institute for Artificial Intelligence

Streams

Advanced Institute for Artificial Intelligence – AI2

https://advancedinstitute.ai

Agenda

Agenda

- Introdução a Streams
- Apache Kafka
- Exemplos

Streams

Batch Processing

- Conjunto de entrada é de um tamanho finito e conhecido, é sabido quando os dados de entrada terminam
- Em uma aplicação real muitos conjuntos de dados não tem um tamanho definido pois eles são recebidos gradativamente.
  - Facebook, Twitter, etc
- Como segmentar a base em partes iguais para processar? Como tratar os assuntos de interesse em um momento particular?

Streams

Batch Processing

- O dataset nunca está “completo”, e por isso o Batch Processing deve dividir os dados em blocos de tamanho específico
- Processar dados referentes a um dia ou hora, por exemplo
- O problema disso é que as mudanças da entrada podem demorar para serem refletidas na saída.

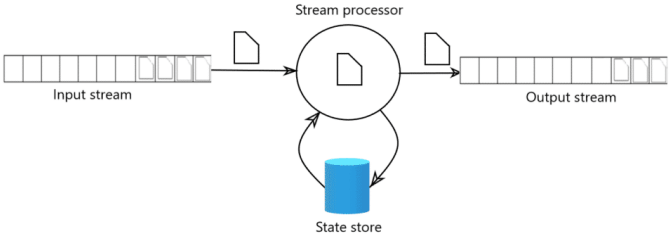
Streams

Processamento de fluxo

- Para evitar atrasos as aplicações poderiam ser preparadas para processar os dados a medida que são disponibilizados
- O StreamProcessing ou processamento de fluxo é uma técnica para trabalhar os dados, a medida que são produzidos
- Se assemelha com Batch Processing, porém se abandona o conceito de bloco e se processa os eventos na medida em que eles ocorrem.

Streams

O processador de stream posicionado entre quem produz dados e quem consome dados, inibe disputas por um recursos compartilhado de armazenamento



Streams

Transmissão de fluxo de dados

- Em processamento de fluxo as entradas são tratadas como arquivo binário, que é analisado em uma sequência de registros chamados de eventos
  - Um evento contém um timestamp
  - Um evento pode ser uma ação do usuário ou originado de uma máquina
  - Um evento pode ser codificado em uma string, em JSON ou em binário
- A princípio um arquivo ou banco de dados é suficiente para conectar produtores e consumidores.
  - Um produtor escreve todo evento gerado em um datastore e cada consumidor checa o datastore periodicamente para checar novos eventos
- Centralizar o processo dessa forma pode provocar erros e não é escalável

Streams

- O processamento de fluxos é mais eficiente quando os consumidores são notificados, quando um novo evento ocorre.
- Arquivos e/ou banco de dados não são ferramentas adequadas para enviar notificações de eventos
- Ferramentas como o Apacha Kafka [1] tem como objetivo controlar a notificação para sistemas de processamento de fluxo

[1] https://kafka.apache.org/

Streams

Sistemas de mensagens

- Diferentes sistemas adotam uma gama de abordagens no modelo de publicação/inscrição e não existe uma abordagem correta para todas as situações.
- Para escolher uma abordagem é útil fazer duas perguntas:
  - O que acontece se o produtor envia mensagens mais rápido do que o consumidor pode processar?
    - Nessa situação o sistema pode ignorar mensagens, guardar as mensagens em uma fila ou aplicar backpressure (impedir que o produtor envie novas mensagens)
  - O que acontece se um nodo falha ou fica offline?
    - Se não há problema de às vezes perder mensagens pode-se optar por uma taxa de transferência maior e uma latência menor.

Sistemas de mensagens

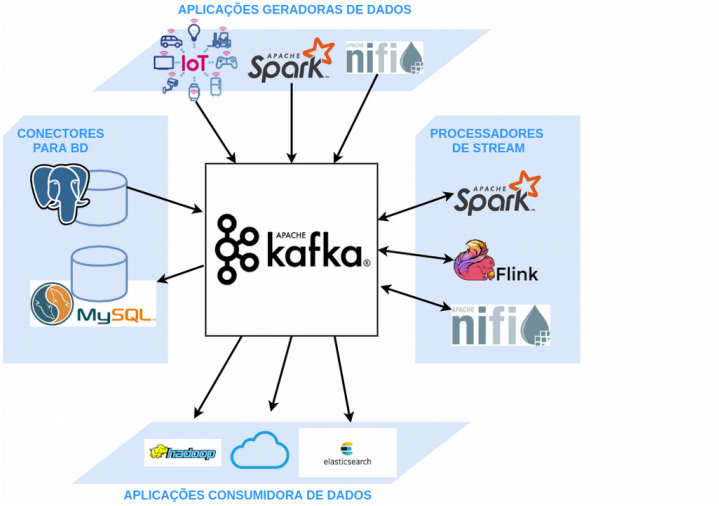
- Mensagens diretas:
  - Geralmente é necessário que o código da aplicação esteja ciente da possibilidade de perda de mensagens.
  - Assumem que o produtor e consumidor estão sempre online.
  - Alguns protocolos permitem que o produtor tente reenviar mensagens.

Streams

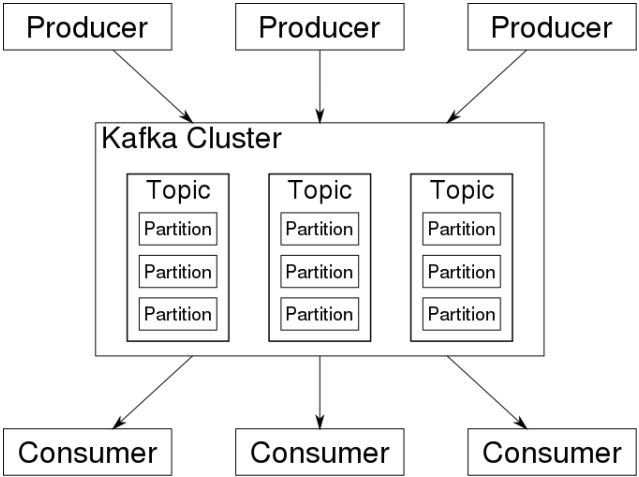
Fila de mensagens:

- Essencialmente um banco de dados otimizado para aguentar fluxos de mensagens.
- Designa mensagens individuais para os consumidores e os consumidores confirmam o recebimento de cada mensagem quando eles terminam de processá-la com sucesso.
- Mensagens são deletadas da fila quando o recebimento é confirmado.
- Se a conexão com um cliente é encerrada ou ocorre um time out antes do recebimento do ack, se assume que a mensagem não foi processada e ela é reenviada para outro consumidor.
- Consumidores devem estar preparados para processar mensagens fora de ordem

Streams


APLICAÇÕES GERADORAS DE DADOS
IoT
APACHE Spark
APACHE nifi
CONECTORES PARA BD
MySQL
APACHE kafka
PROCESSADORES DE STREAM
APACHE Spark
Flink
APACHE nifi
hadoop
elasticsearch
APLICAÇÕES CONSUMIDORA DE DADOS

Streams


Producer
Producer
Producer
Kafka Cluster
Topic
Partition
Partition
Partition
Topic
Partition
Partition
Partition
Topic
Partition
Partition
Partition
Consumer
Consumer
Consumer

Streams

Elementos do Apache Kafka

- Produtor: gera mensagens
- Consumidor: consome mensagens
- Servidor: controla o envio e recebimento de mensagens
- Broker de mensagens: gerencia o envio das mensagens entre os diferentes tópicos
- Zoekeeper [2]: módulo apache para coordenação distribuída

[2] https://zookeeper.apache.org/

Streams

Cenário de exemplo

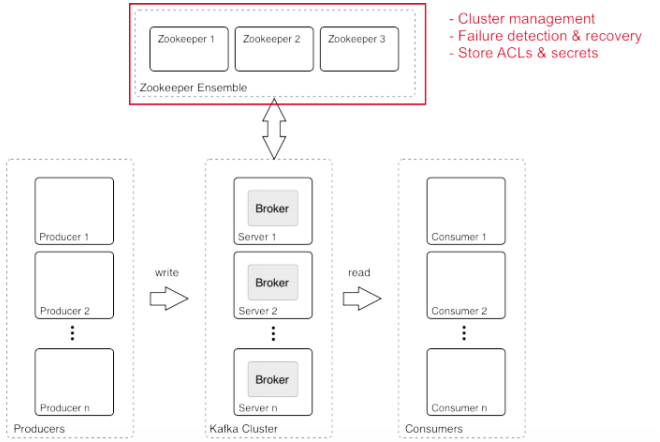
- Produtor: gerar mensagens a partir de busca no twitter
  - Exemplo: ler apenas mensagem de um determinado assunto
- Consumidor: consumir as mensagens e processar (Listar as hashtags que aparecem)
- A aplicação não vai apresentar um bug se perder alguma das mensagens
- A aplicação sempre vai receber os dados gerados em tempo real
  - Adequado para aplicações que demandam processamento de operações em tempo real (passado não é considerado)

Streams

Dissociando produtores e consumidores

- Os consumidores lentos não afetam os produtores
- Adicionar consumidores sem afetar os produtores
- A falha do consumidor não afeta o sistema

Zoekeeper


Zookeeper 1
Zookeeper 2
Zookeeper 3
Zookeeper Ensemble
- Cluster management
- Failure detection & recovery
- Store ACLs & secrets
Producer 1
Producer 2
Producer n
Producers
write
Broker
Server 1
Broker
Server 2
Broker
Server n
Kafka Cluster
read
Consumer 1
Consumer 2
Consumer n
Consumers

Streams

Tópicos

- Fluxos de mensagens “relacionadas” no Kafka
  - Representação lógica
  - Categoriza mensagens em grupos
- Os desenvolvedores definem os tópicos
- Produtores podem mandar mensagens para diversos tópicos
- Número ilimitado de tópicos