Mensageria de alta performance com apache kafka

Autor: Bruno Ferreira de Souza

Sumário

- Introdução ao problema de alocação de recursos para aplicação.
- Chamadas Rest/HTTP.
- Comunicação por troca de mensagem.
- Teorema CAP
- O que o kafka tem a ver com isso?
  - Tópicos
  - Particionamento/Hash
  - Redundância
  - Paralelismo
  - Offset
  - Garantias de entrega
  - Garantias de sincronização
  - Leader
  - Produtor / Key de Hash
  - Consumidor / Consumer Group
- Mão na massa.

1. Introdução ao problema de alocação de recursos para aplicações

Monoliticas

- Baixa latência ou nenhuma entre recursos.
- Paralelismo com multi threads.

Chamadas Rest

- Aplicação distribuída em serviços independentes.
- Sofre com a latência da rede.
- Paralelismo com multi Threads.
- Escalonamento horizontal.

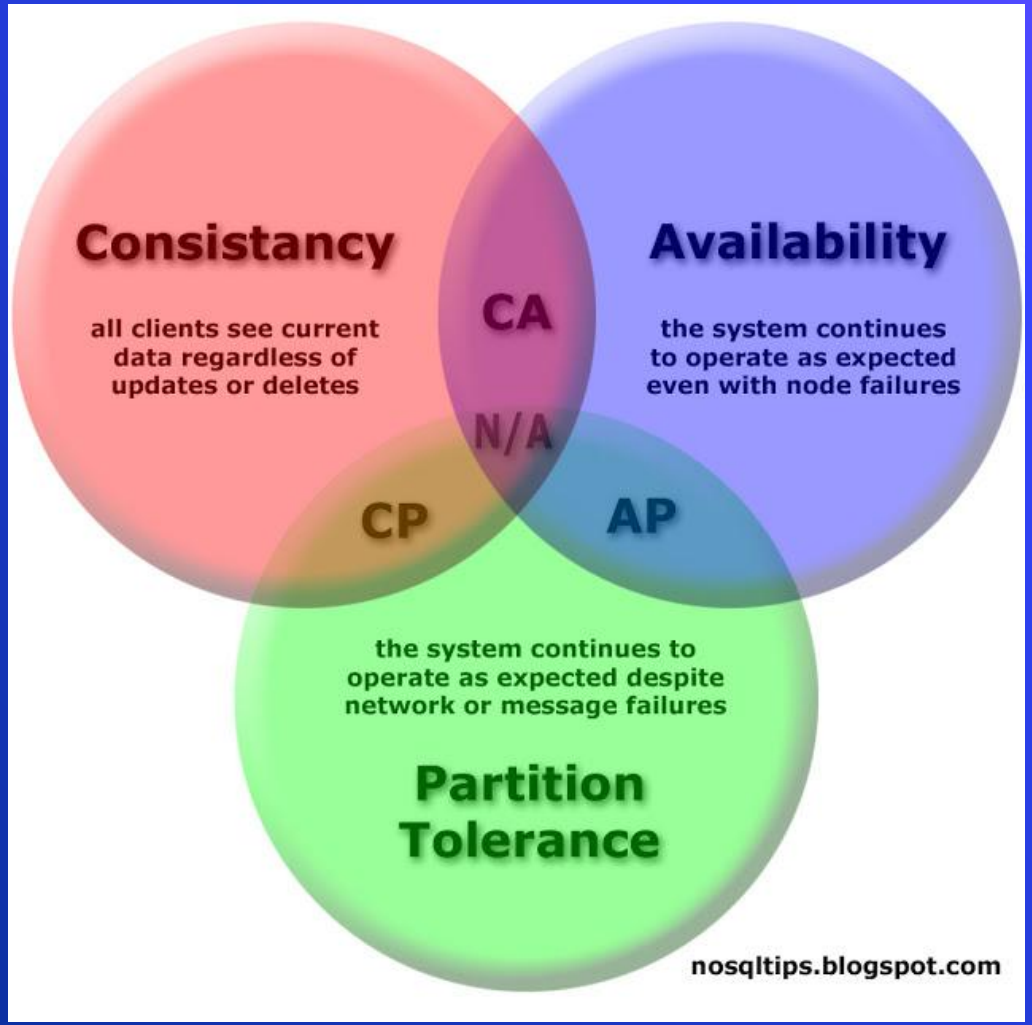
Comunicação por troca de mensagem(Publisher - Subscriber)

- Aplicação distribuída em serviços independentes.
- O produtor não precisa saber quem está recebendo.
- O consumidor não precisa saber quem está publicando a mensagem.
- Um serviço novo só precisa se inscrever nas fila de seu interesse.

Teorema CAP

Teorema CAP

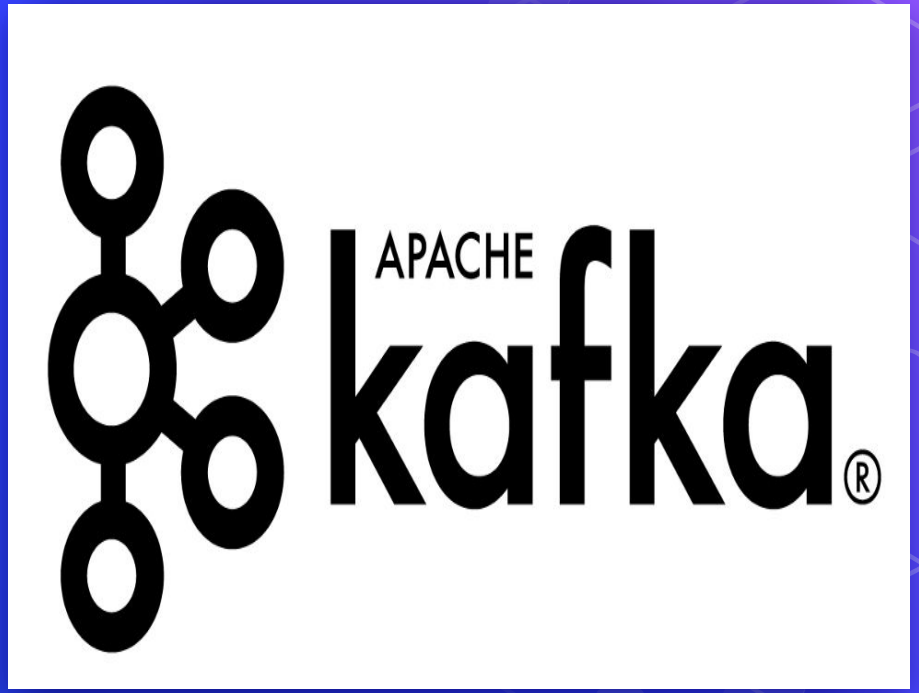
- O Teorema cap joga uma luz sobre os problemas enfrentados na hora de decidir qual frente queremos atacar no nosso sistema.
- Ele consiste em 3 pilares, entretanto só é possível escolher 2.


Consistancy
all clients see current data regardless of updates or deletes
CA
Availability
the system continues to operate as expected even with node failures
N/A
CP
AP
the system continues to operate as expected despite network or message failures
Partition Tolerance
nosqltips.blogspot.com

O que o kafka tem a ver com isso?

O que o kafka tem a ver com isso?

O que o kafka tem a ver com isso?



Apache KAFKA

- O Kafka vem ganhando cada vez mais adoção em sistemas de alta disponibilidade devido a sua alta performance em processamento de mensageria.

E o que ele tem de bom ?

Tópicos

- É a forma mais simples de separar um bloco de mensagem por categoria.

Redundância

- O mecanismo de redundância do kafka permite criar um cluster de mensageria muito eficiente.
- Tornando o sistema tolerante a falhas de um nó, uma vez que suas mensagens podem estar replicadas em outro nó.

Particionamento / Hash

- Mecanismo muito explorado pelo kafka para aumento de performance com paralelismo, uma vez que dividindo as mensagens recebidas em partições se torna mais simples o papel de delegar aquele pedaço para outro Nó no cluster.

Paralelismo

- Com tudo que foi dito anteriormente tópicos,particionamento, redundância...
- O Paralelismo no kafka é possível uma vez que cada nó no cluster pode se tornar responsável por um ou N partições.

OFFSET

- O offset é uma forma simples para identificar que uma mensagem já foi consumida por um serviço que escuta um determinado tópico.

Garantia de entrega

- O kafka também fornece alguns recursos muito importantes para a aplicação produtora.

  Ex:

  Enviar uma mensagem e esperar um “ok” do kafka indicando que a mensagem foi recebida com sucesso.

Garantia de sincronização

- Como estamos falando de um serviço em cluster... e se o nó que recebeu minha mensagem cair?
- O kafka oferece uma configuração para o produtor receber um “ok” só quando todos as réplicas já estão sincronizadas.

Leader

- Um leader é responsavel por x partições.
- Caso esse leader caia o kafka elege um novo leader.

Produtor / key de hash

- O produtor pode enviar mensagens para tópicos.
  - Garantia de entrega.
  - Garantia de sincronização.
  - Sem garantia de entrega.
- O produtor deve passar uma key para ser aplicar na funçao hash e determinar para qual partição dentro do tópico a mensagem será guardada.

Consumidor / Consumer group

- O consumidor pode se cadastrar nos tópicos de seu interesse
  - É possivel usar pattern para nome de tópicos.
- O consumer group informa para o kafka quem está consumindo aquela mensagem, para o caso de ter mais de um serviço do mesmo consumer group ouvindo aquele tópico.
  - A mensagem não será duplicada para serviços do mesmo group

Vamos ver como funciona?

Obrigado!

Alguma dúvida?

Você pode me encontrar em:

GitHub: @Brunofs

brunocssjm@gmail.com