Otimização do Mecanismo de Confirmação de Entrega de Mensagens no Apache Kafka

Iago C. Corrêa[1], **Patrícia Pitthan Barcelos**[1]

[1]Pós-Graduação em Ciência da Computação (PGCC)
Universidade Federal de Santa Maria - UFSM
Santa Maria – RS – Brasil

{icorrea,pitthan}@inf.ufsm.br

Resumo. *O Apache Kafka é uma plataforma de mensageria e streaming de dados, baseada no modelo produtor-consumidor, que implementa confiabilidade na entrega de mensagens através de Reconhecimento Positivo. Entretanto, em caso de falha, não é possível recuperação de mensagens. Este trabalho propõe a implementação de um mecanismo de confirmação de entrega de mensagens baseado em Reconhecimento Negativo. O mecanismo utiliza-se da persistência em cache local de forma a aumentar a confiabilidade.*

1. Introdução

O Apache Kafka [Apache 2012] é um ecossistema de mensageria e *streaming* de dados de alto desempenho que vem sendo amplamente utilizado para pré-processamento e transporte de quantidades massivas de dados entre aplicações. O Kafka segue um modelo "produtor-consumidor" em que mensagens são geradas por produtores e enviadas a um *cluster* para serem armazenadas em "tópicos", tornando-se aptas para consumo.

O Kafka implementa confirmação de entrega de mensagens através de reconhecimento positivo (*positive acknowledgement*, ou *ack*). Um *ack* notifica o produtor quando uma mensagem é entregue com sucesso ao seu destino. Há três tipos de *acks*, os quais diferem no que se refere à confiabilidade e ao desempenho. Entretanto, nenhum deles notifica o produtor quando uma mensagem enviada não alcança seu destino. Este trabalho propõe a implementação de um mecanismo de confirmação de entrega de mensagens baseada em reconhecimento negativo (*negative acknowledgement*), ou *nack*. O mecanismo proposto inclui um módulo interceptor, responsável por armazenar, em cache local, as mensagens enviadas, possibilitando seu reenvio em caso de perda.

2. A arquitetura

O mecanismo de confirmação de entrega de mensagens do Kafka está relacionado com a garantia de persistência no *cluster*. Sendo assim, os valores permitidos para a configuração de *acks* são 0, 1 e -1. Em produções de mensagens com *ack* igual a 0, denominadas produções *fire-and-forget*, não há confirmação efetiva de entrega da mensagem, i.e., mensagens são consideradas entregues logo que são enviadas. Com *ack* igual a 1, produtores recebem confirmação de entrega assim que ocorre a primeira persistência da mensagem. Já com *ack* igual a -1, uma confirmação de entrega de mensagem é enviada ao produtor quando a mensagem foi persistida e replicada em todos os nodos, podendo ser obtida a partir de outros locais, em caso de falha.

Há um *trade-off* relevante que deve ser avaliado durante a definição de qual configuração de *ack* utilizar, baseado em uma relação entre confiabilidade e desempenho. Métricas comumente utilizadas para avaliar desempenho no Kafka são a vazão e a latência [Le Noac'h et al. 2017]. A vazão refere-se à quantidade de mensagens enviadas em um intervalo de tempo, enquanto a latência está relacionada ao tempo que uma mensagem leva para chegar ao seu destino.

Neste contexto, produções de mensagens com *ack* igual a 0 apresentam maior *throughput* e menor latência [Dobbelaere and Esmaili 2017], uma vez que não aguardam a confirmação de entrega. Neste caso, o usuário aceita o risco de perder mensagens. Além disso, o *ack* igual a 0 impossibilita reenvio de mensagens, já que não é possível identificar quais mensagens foram perdidas. O envio de mensagens que requerem *ack* igual a 1 apresentam uma relação de equilíbrio entre confiabilidade e desempenho. Com este *ack*, a confirmação de entrega ocorre quando a mensagem foi persistida em apenas um local. O desempenho é pior se comparado com *ack* igual a 0 e melhor se comparado com *ack* igual a -1. Entretanto, o nodo detentor da mensagem consiste em um ponto único de falha (*SPOF - single point of failure*). Produções de mensagens com *ack* igual a -1 apresentam o pior desempenho. Além de garantir a persistência da mensagem, existe a necessidade de replicá-la em outros nodos. Porém, estas produções demonstram-se mais confiáveis.

Este trabalho enfoca o aumento da confiabilidade na entrega de mensagens em produções de mensagens *fire-and-forget*. Para isso, o trabalho propõe a implementação de um mecanismo de confirmação de entrega de mensagens baseado em *nack*. A produção de mensagens com *nack* poderá notificar o produtor caso uma mensagem não seja entregue com sucesso. Para possibilitar o reenvio em caso de perda, o mecanismo implementará um agente responsável por interceptar o envio de mensagens ao *cluster*, armazenando-as em cache local. Caso um *nack* seja recebido pelo produtor, a mensagem "perdida" pode ser recuperada na cache local e reenviada ao destino.

3. Considerações Finais

Um dos problemas de produções de mensagens *fire-and-forget* no Apache Kafka é o desconhecimento por parte do produtor quando há perdas de mensagens. O presente trabalho apresenta uma proposta de implementação de uma arquitetura que permitirá a recuperação de mensagens em casos de perdas. Com a implementação de reconhecimentos negativos e o auxílio de um agente monitor, objetiva-se manter o desempenho de produções com *ack* igual a 0, aumentando a confiabilidade na entrega de mensagens.

Referências

Apache (2012). Apache kafka is a distributed streaming platform. what exactly does that mean? https://kafka.apache.org/intro.

Dobbelaere, P. and Esmaili, K. S. (2017). Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: Industry paper. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, DEBS '17, pages 227–238, New York, NY, USA. ACM.

Le Noac'h, P., Costan, A., and Bougé, L. (2017). A performance evaluation of apache kafka in support of big data streaming applications. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4803–4806.