Avaliação dos Brokers Kafka e Apache Flume no Contexto de Big Data

Matheus Orlandi de Castro, Cristiano Bertolini, Evandro Preuss[1], Edison Pignaton de Freitas[2]

[1]Departamento de Tecnologia da Informação - UFSM Campus Frederico Westphalen

[2]Instituto de Informática Universidade Federal do Rio Grande do Sul - UFRGS

mo.castro@hotmail.com, cristiano.bertolini@ufsm.br,
evandro.preuss@gmail.com, epfreitas@inf.ufrgs.br

Abstract. *This paper presents a performance analysis of the Kafka e Flume brokers in the big data context. Initially it will be define the services and infrastructure configuration needed, and also the setup of the environment for the experiments. The experiments will compare the performance of the brokers using metrics and with different type of configurations. Also, it will be develop an application, which it will use the big data infrastructure.*

Resumo. *Este artigo apresenta uma proposta para a análise de desempenho dos brokers Kafka e Flume no contexto de big data. Inicialmente serão definidos os serviços necessários além da instalação e configuração dos mesmos, e também, do ambiente esperado para a realização dos experimentos. Após isso a análise será realizada por meio de benchmarks através de métricas. Para a simulação das transferências pelos brokers será, inicialmente utilizado dados sintéticos, ou seja, gerados a partir de scripts e posteriormente, dados reais coletados por meio de uma aplicação a ser desenvolvida.*

1. Introdução

Desde o início da era digital a quantidade de dados vem aumentando, com isso surgiram as primeiras máquinas para armazenar dados, com o passar do tempo a necessidade de uma maior capacidade computacional se tornou necessária, trazendo um novo conceito de armazenamento e processamento. Essa definição trouxe a ideia de paralelismo, este modelo de arquitetura é baseado no uso de *clusters* em que cada máquina tem seu próprio processador, disco, etc [Chen et al. 2014].

Com o grande crescimento do volume de dados, desde a sua geração, passando pela aquisição, análise e armazenamento está surgindo uma nova área de pesquisa chamada *Big Data*. Segundo [Mayer-Schönberger and Cukier 2013], *Big Data* refere-se a coisas que são feitas em grande escala, para extrair novas percepções ou criar novas formas de valor, de maneira que consiga mudar o mercado, organizações e a relação entre os cidadãos e governo.

Este trabalho tem como objetivo principal realizar uma análise de desempenho de dois *Brokers*, *Apache Kafka* e *Apache Flume*, por meio de *benchmarks*. Esses softwares estão enquadrados na fase de aquisição, eles são responsáveis por escalonar o envio de dados por meio de mensagens e entregar ao seu destinatário. Além disso pretende-se desenvolver um *software* que seja capaz de simular um fluxo grande de dados e também realizar trocas de mensagens entre *brokers*, para que se possa fazer

uma avaliação deles e com isso determinar qual *broker* leva vantagem sobre o outro em diferentes configurações de execução.

2. Referencial Teórico

De acordo com Gantz e Reinsel [Gantz and Reinsel 2011] as tecnologias de *Big data* descrevem uma nova geração de tecnologias e arquiteturas, concebida para extrair economicamente valor a partir de volumes muito grandes de uma ampla variedade de dados, permitindo alta velocidade de captura, descoberta e/ou análise. Esta definição engloba a abordagem dos quatro V's, sendo considerada hoje, a mais completa. Além disso *Big data* possui algumas características chave, que juntas são conhecidas como uma cadeia de valor, essa cadeia é composta por quatro pilares principais que são: geração, seguido pela aquisição passando pelo armazenamento e por fim análise dos dados. Neste trabalho, são estudados dois *brokers*: *Kafka* e *Flume*.

Apache Kafka é um *software* de mensagens implementado como um transmissor de *logs* distribuído, adequado para consumo *offline* e *online* de mensagens. Foi projetado para permitir que um único *cluster* sirva como a espinha dorsal de dados para uma grande organização. Ele pode ser expandido de forma elástica e transparente sem tempo de inatividade. Os fluxos de dados são divididos e distribuídos por um conjunto de máquinas para permitir uma maior capacidade em relação a uma única máquina [Apache b].

Flume é um serviço distribuído para coletar, agregar e mover grandes quantidades de dados de *logs*, de forma eficiente e confiável. Ele usa um modelo de dados simples que permite a aplicação analítica *online* [Apache a]. A arquitetura do *flume* é muito simples de ser entendida, de acordo com Hoffman [Hoffman 2013], os três principais componentes dele são: as fontes, que são responsáveis por fazer a coleta dos dados que serão transmitidos, após isso, estes dados são transportados por meio de um canal de comunicação e por fim chegam ao *sink*, que é o último componente do *flume* que um determinado dado percorre, chegando neste ponto os arquivos podem ser distribuídos em uma base de dados não relacional, em um sistema de arquivos distribuídos, etc..

3. Solução Proposta e Resultados Preliminares

Inicialmente será utilizado *scripts* desenvolvidos em *shell script* que sejam capazes de gerar arquivos de diversos tamanhos e em grandes quantidades. Posteriormente um protótipo de uma aplicação que seja capaz de coletar dados reais e realize troca de mensagens entre os *brokers* deverá ser desenvolvido, para que se consiga uma análise mais precisa. Com isso será necessário realizar a configuração dos *brokers* além da instalação do *software Hadoop*, que é uma estrutura voltada para ambientes distribuídos que permite realizar análises e armazenamento de grandes conjuntos de dados estruturados e não estruturados.

O protótipo será responsável por produzir os dados necessários, após isso, estes dados gerados serão coletados por um ou mais agentes do *flume*, sendo que cada agente pode possuir uma ou mais *sources*, *channels* e *sinks*. Quando terminado o transporte ao longo do *flume* estes dados serão armazenados no sistema de arquivos do *Hadoop*. Da mesma forma, o protótipo interage com o *kafka* com a diferença apenas no funcionamento do *broker*, já que o *kafka* não é capaz de realizar a coleta destes dados, neste cenário então, o protótipo deverá ser capaz de além de produzir os dados, alimentar o *broker*, enviando mensagens para uma ou mais partições definidas

previamente. Observando a Figura 1, pode ser verificado o esquema de funcionamento da aplicação.

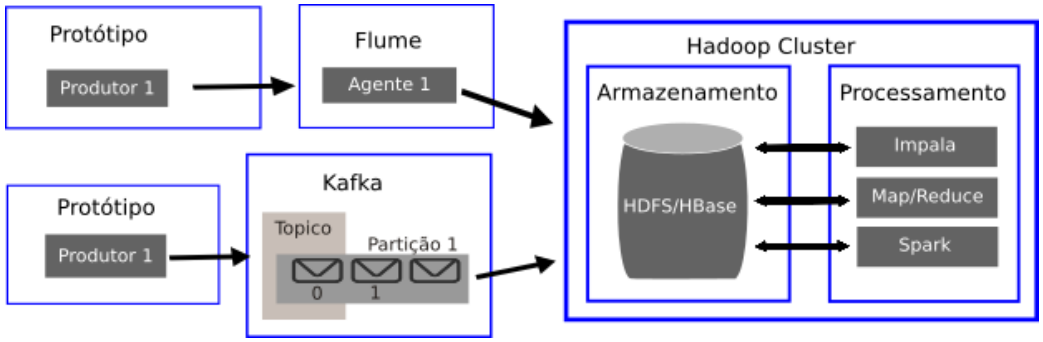

Figura 1. Interação entre o protótipo, Apache Kafka, Apache Flume e Hadoop

A aplicação em questão tem como finalidade recuperar *Tweets* públicos além da possibilidade de conseguir filtrar os *tweets* de acordo com parâmetros informados e ainda, agregar essas informações e formatá-las em um padrão aceito pelos *brokers*. Além disso, para possibilitar o funcionamento deve ser gerada duas chaves de acesso, sendo uma delas secreta e dois *Tokens* de acesso, sendo um deles secreto também. Estes parâmetros devem ser informados no código fonte para permitir o funcionamento da aplicação.

4. Trabalhos Relacionados

Córdova [Córdova 2014] realizou um estudo com o objetivo de analisar duas das plataformas de código aberto mais notáveis na área de processamento de *Big Data*, *Storm* e *Spark*. Segundo o autor do estudo, ambos os sistemas oferecem capacidades de processamento em tempo real através de micro-batches, mas o funcionamento dos mecanismos, respostas de falhas, velocidade de processamento e muitas outras características são diferentes. Após a realização dos experimentos, Córdova [Córdova 2014] constatou que com registros pequenos o *Storm* foi em torno de 40% mais rápido em relação ao *Spark*. No entanto, com o aumento do tamanho dos registros o *Spark* conseguiu melhorar seu desempenho, chegando a superar o *Storm*.

A análise realizada por Córdova [Córdova 2014] foi executada utilizando serviços que são aplicados na fase de processamento de dados, em contrapartida o trabalho proposto irá realizar também uma análise, comparando dois *brokers* voltados para a etapa de aquisição e transferência de dados.

Ionescu [Ionescu 2015] apresenta uma pesquisa que busca comparar a velocidade de processamento para envio e recebimento de mensagens, carga de memoria e as plataformas que os *brokers RabbitMQ* e *ActiveMQ* podem gerenciar. Duas aplicações desenvolvidas em *JAVA* foram implementadas para que ocorra a simulação de envio e recebimento de mensagens. No primeiro teste, que busca verificar a degradação da performance do *broker* com o aumento do tamanho da mensagem, o *ActiveMQ* obteve um desempenho superior em relação ao *RabbitMQ* levando em conta o envio de mensagens, já no recebimento, o *RabbitMQ* conseguiu superar seu rival. Em uma avaliação diferente, que envolve o aumento no numero de clientes enviando

mensagens para o *broker*, a performance de ambos teve um decréscimo, entretanto o *RabbitMQ* conseguiu um desempenho superior.

Em comparação com o trabalho proposto, ambos possuem muitas semelhanças, entre elas a de análise de desempenho entre dois mensageiros, a diferença básica entre os dois trabalhos está exatamente nos dois *brokers*, onde Ionescu [Ionescu 2015] utiliza o *RabbitMQ* e *ActiveMQ* e o trabalho proposto utilizará o *Apache Flume* e o *Apache Kafka*.

5. Conclusões

No decorrer do desenvolvimento deste trabalho pode-se constatar alguns pontos que deverão ser aprofundados na execução da próxima etapa, como o funcionamento dos dois *brokers* que serão utilizados, além de seus diversos parâmetros de configurações e como eles irão impactar na execução dos experimentos a serem realizados.

Além disso, pode-se observar também, o grande consumo de processamento, armazenamento e de memória por todos os serviços utilizados, sendo necessária uma estrutura de hardware extremamente poderosa, capaz de minimizar alguns gargalos como taxa de transferência de disco e taxa de transferência entre adaptadores de rede. Estes gargalos devem ser observados com muito cuidado, devido que a única limitação para a realização dos experimentos devem ser apenas os *brokers*, ou seja, não pode haver nenhum tipo de obstáculo externo durante estas execuções. Ao final deste trabalho, espera-se obter resultados que constatem em quais ambientes cada *broker* irá conseguir se sobressair em relação ao outro.

Referências

Apache. Apache Flume: documentação. https://flume.apache.org//. Acessado: 28/05/2016.

Apache. Apache Kafka: documentação. http://kafka.apache.org/. Acessado: 28/05/2016.

Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. Mobile Networks and Applications, 19(2):171–209.

Córdova, P. (2014). Analysis of real time stream processing systems considering latency.

Gantz, J. and Reinsel, D. (2011). Extracting value from chaos. IDC iview, 1142:1–12. Hoffman, S. (2013). Apache Flume: Distributed Log Collection for Hadoop. Packt Publishing Ltd.

Ionescu, V. M. (2015). The analysis of the performance of rabbitmq and activemq. In RoEduNet International Conference-Networking in Education and Research (RoEduNet NER), 2015 14th, pages 132–137. IEEE.

Mayer-Schönberger, V. and Cukier, K. (2013). Big data: A revolution that will transform how we live, work, and think. Houghton Mifflin Harcourt.