Implementação e Avaliação do Paralelismo de Flink nas Aplicações de Processamento de Log e Análise de Cliques

Gabriel Rustick Fim, Dalvan Griebler

[1] Laboratório de Pesquisas Avançadas para Computação em Nuvem (LARCC), Faculdade Três de Maio (SETREM), Três de Maio, Brasil

gabifimtm@gmail.com, dalvangriebler@setrem.com.br

Resumo. *Este trabalho visou implementar e avaliar o desempenho das aplicações de Processamento de Log e Análise de Cliques no Apache Flink, comparando o desempenho com Apache Storm em um ambiente computacional distribuído. Os resultados mostram que a execução em Flink apresenta um consumo de recursos relativamente menor quando comparada a execução em Storm, mas possui um desvio padrão alto expondo um desbalanceamento de carga em execuções onde algum componente da aplicação é replicado.*

1. Introdução

Com o aumento dos dados produzidos, o processamento e contextualização de dados brutos em tempo real se tornou uma necessidade para extrair informações relevantes para as instituições. Estas são aplicações de software que possuem como características o consumo de dados ilimitados [Deshpande and Kumar 2018], no formato de tuplas ou eventos. Com o aumento na demanda de aplicações que se encaixem nestas características, foram criados *frameworks* que fornecem um modelo de programação que diminui a complexidade do código com relação a detalhes como escalonamento, tolerância a falhas, processamento e otimização de consultas. Obviamente estes *frameworks* possuem algumas similaridades como a modelagem do processamento de dados através de um grafo de fluxo de dados com os vértices sendo os operadores e as arestas os fluxos de dados.

O principal objetivo deste trabalho é fornecer a implementação das aplicações de Processamento de Log e Análise de Cliques no Apache Flink e realizar uma análise comparativa de desempenho. Este trabalho continua e estende os esforços de [Bordin et al. 2020], o qual falaremos um pouco mais na Seção 2. O presente artigo contribui com a implementação de duas aplicações contidas no DSPBench com Apache Flink e comparação do desempenho com as versões em Apache Storm. Este artigo está estruturado da seguintes forma. A Seção 2 contém os trabalhos relacionados. A Seção 3 apresenta os detalhes da implementação. A Seção 4 discute os resultados e a Seção 5 apresenta as conclusões e possíveis trabalhos futuros.

2. Trabalhos Relacionados

No artigo [Bordin et al. 2020], os autores apresentam um conjunto de aplicações em formato de *benchmark* a fim de permitir a avaliação de desempenho de *frameworks* de processamento de dados em fluxo continuo. Os autores descrevem 15 aplicações que são englobadas no conjunto, todas implementadas para execução no *framework* Apache Storm, para fins de comparação os autores também realizaram a implementação de 3 destas aplicações no Apache Spark Streaming. No artigo de [Lu et al. 2014] os autores apresentam sua *benchmark*, denominada StreamBench e contendo 7 aplicações, aplicando esta a dois

frameworks de processamento de dados, Spark Streaming e Apache Storm. No trabalho de [Huang et al. 2010], os autores comparam o desempenho dos *frameworks* Apache Storm, Flink e Spark Streaming através da utilização de seu próprio conjunto de aplicações estilo *benchmark*, denominado HiBench, sendo este composto por 4 aplicações de dados em fluxo contÍnuo.

3. Implementação

Devido à quantidade e complexidade das aplicações presentes no DSPBench, este artigo abrangerá duas aplicações: Análise de Cliques e Processamento de Log. Todas as estratégias de execução empregadas nos códigos do Apache Storm foram portadas para a versão em Apache Flink, uma vez que ambas as implementações deveriam obter os mesmos resultados finais e terem um grafo de componentes similares. Também foram mantidos grande parte dos códigos referentes ao processamento dentro dos componentes das aplicações, sendo somente necessário reescrever algumas linhas de código para a passagem de dados entre os componentes e que continham utilizações de bibliotecas e elementos próprios do Storm para realização do processamento para possíveis substitutos presentes no Flink.

A execução das aplicações nos *frameworks* foi intercalada e repetida 5 vezes por um tempo delimitado de 10 minutos devido a natureza não-infinita da fonte de dados utilizada pelas aplicações. Para cada execução foram calculados a quantidade média de saídas e a utilização de recursos computacionais. Buscando comparar o desempenhos dos *frameworks* para a execução das aplicações, foi realizado a replicação de um componente por aplicação, o que também torna possível realizar uma pequena comparação dos *frameworks* em si. A realização da replicação do componente se deu através de configurações próprias de cada *framework* que permitem a replicação de cada componente. Foram escolhidos os fatores de replicação de 2, 4 e 8 vezes, além da execução *baseline* onde não houve replicação de componente. Para ambas as aplicações o componente replicado foi o denominado GeoFinder, já que este é um componente *stateless*, não realizando nenhuma agregação de dados ou trabalhando com uma necessidade temporal, além de não guardarem nenhum estado para uso posterior no processamento.

4. Resultados

Os experimentos foram executados na infraestrutura do LARCC[1]. Criou-se sete máquinas virtuais dentro da ferramenta OpenNebula 6.0.0.2 com o virtualizador KVM, sendo um Mestre com 2 vCPUS, 2GB de memória RAM e 20GB de disco e três Trabalhadores com 4 vCPUS, 4GB de memória RAM e 20GB de disco, um nodo rodando o Apache Kafka com 2 vCPUS, 12GB de memória RAM e 30GB de disco e um nodo rodando o Apache ZooKeeper com 3 vCPUS, 4GB de memória RAM e 10GB de disco. Além disso a um nodo onde é executado a ferramenta de monitoramento de infraestrutura Zabbix para monitoramento e retirada de métricas dos outros nodos presentes no ambiente. O ambiente físico é composto por três máquinas HP Proliant DL385 G6, com um processador AMD Opteron 2425 2100 MHz 6-Core e 32 GB de memória RAM DDR3, cada uma possuindo 5 interfaces de rede Gigabit, onde estão sendo executados as VMs contendo o Zabbix e ZooKeeper, e uma máquina possuindo processador Intel Xeon Silver 4108 8-core/16 threads, 64 GB de memória DDR4 e HD 2 TB .

[1]Laboratório de Pesquisas Avançadas para Computação em Nuvem: http://larcc.setrem.com.br/

As máquinas virtuais onde os testes foram executados possuem as versões de software: Sistema operacional Ubuntu *Server* 20.04.3 LTS (Focal Fossa) (5.4.0-1071-kvm), JDK e JRE 1.8.0_352, Apache Flink 1.16.0, Apache Storm 2.4.0, Apache ZooKeeper 3.8.0, Apache Kafka 3.3.1, Zabbix 6.0.12. A Figura 1 ilustra o consumo de recursos para a execução da aplicação Análise de Cliques no Flink e Storm. É notável que apesar da execução em Flink possuir uma utilização média de recursos menor que a execução em Storm, ela apresenta um desvio padrão alto na utilização de CPU e raramente consome recursos do Nodo3, apontando para um desbalanceamento na divisão de carga durante o processamento.

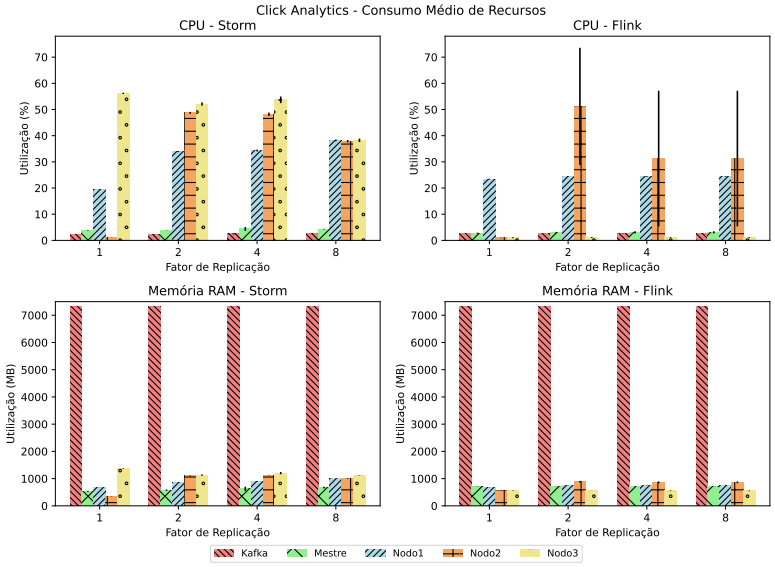

Figura 1. Consumo Médio de Recursos da aplicação Análise de Cliques

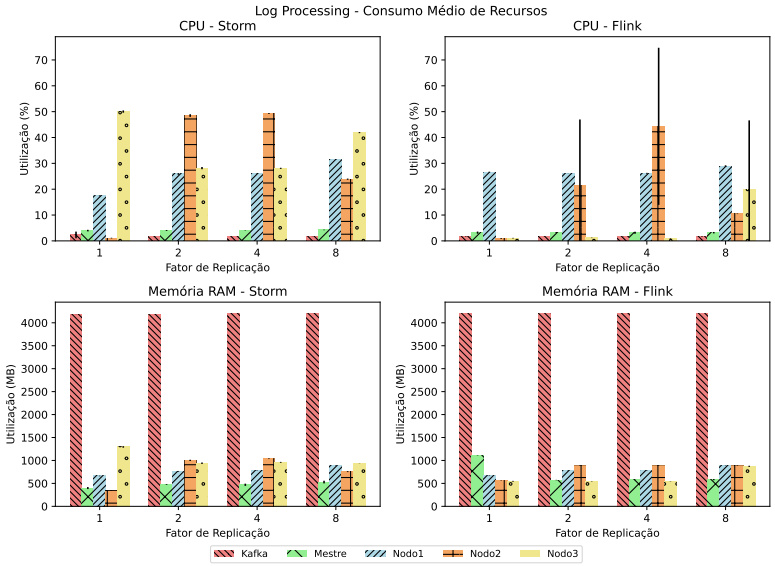

Figura 2. Consumo Médio de Recursos da aplicação Processamento de Log

A Figura 2 demonstra o consumo de recursos para a execução da aplicação Processamento de Log no Flink e Storm. É notável que assim como a execução do Análise

de Cliques, a execução em Flink apresenta uma utilização média de recursos menor que a execução em Storm e um alto desvio padrão, novamente apontando para um possível desbalanceamento de carga na execução em Flink. A Figura 3 mostra a média de operações de saída obtidas pela execução da aplicação, obtido através do calculo médio da quantia total de dados emitidos pelo último componente das aplicações nas 5 execuções em cada fator de replicação. Como pode ser visto, ambas as execuções possuem pouca variação de desempenho acima do fator de replicação 2, apontando a existência de gargalos nos componentes que compõem as aplicações.

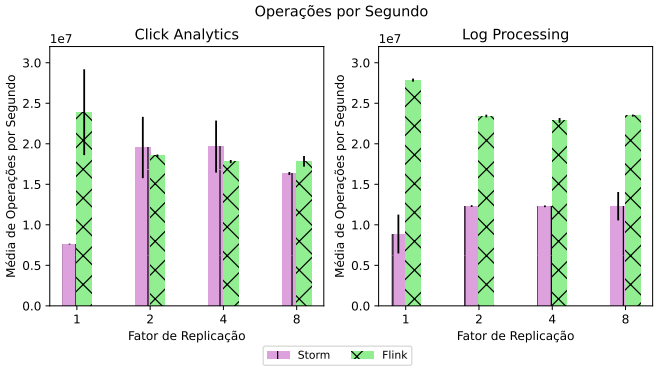

Figura 3. Média Total de Saídas das Aplicações

5. Conclusões

Esse trabalho apresentou uma breve expansão do conjunto de aplicações do DSPBench para o *framework* Apache Flink e posteriormente comparou os resultados de execução de duas aplicações com o Apache Storm. Considerando as execuções em Apache Storm e Apache Flink no ambiente computacional distribuído dos experimentos, os resultados com diferentes fatores de replicação revelaram que apesar do Apache Flink obter uma execução superior ao Apache Storm com o fator de replicação de 1, ele apresenta uma diminuição do número médio de operações de saída quando o fator de replicação é aumentado. Ao contrário do Apache Storm, que não possui um média de operações de saída alta na execução com fator de replicação 1 e consegue uma melhora neste quesito com o aumento do fator de replicação. Como trabalhos futuros é interessante avaliar o desempenho das execuções com a replicação de múltiplos componentes e com variantes graus de réplicas e partições na geração dos tópicos dentro do Apache Kafka.

Referências

Bordin, M. V., Griebler, D., Mencagli, G., Geyer, C. F. R., and Fernandes, L. G. (2020). DSPBench: a Suite of Benchmark Applications for Distributed Data Stream Processing Systems. *IEEE Access*, 8(na):222900–222917.

Deshpande, A. and Kumar, M. (2018). *Artificial Intelligence for Big Data: Complete guide to automating Big Data solutions using Artificial Intelligence techniques*. Packt Publishing.

Huang, S., Huang, J., Dai, J., Xie, T., and Huang, B. (2010). The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, pages 41–51.

Lu, R., Wu, G., Xie, B., and Hu, J. (2014). Stream Bench: Towards benchmarking modern distributed stream computing frameworks. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 69–78.