Desempenho da Plataforma Apache Flink

João Brito, M9984
Department of Computer Science
University of Beira Interior
Covilhã, Portugal
joao.pedro.brito{at}ubi.pt

André Martins, M10157
Department of Computer Science
University of Beira Interior
Covilhã, Portugal
andremar1000{at}gmail.com

***Abstract*—Este documento tem a função de apresentar a plataforma Apache Flink e avaliar o respetivo desempenho para dois *benchmarks* diferentes. Os *benchmarks* que serviram para análise desse desempenho foram o Yahoo Streaming Benchmark e o StreamBench. Será também apresentado um ponto de vista geral à arquitetura do Apache Flink, bem como um tutorial de como o instalar e configurar no sistema operativo Linux Ubuntu.**

I. Introdução

O Apache Flink é uma plataforma processamento de fluxo *open-source* desenvolvido pela Apache Software Foundation. É uma plataforma escrita em Java e Scala e executa, de uma maneira arbitrária, programas de forma paralela e canalizada. É uma ferramenta muito utilizada na área de *Big Data* e tem vários pontos que o tornam ideal na área do processamento de grandes quantidades de dados:

- É uma *"true stream processing framework"*, ou seja, não corta o fluxo em micro-*batches*;
- O núcleo do Flink tem a capacidade de processamento distribuído, tolerância a falhas, entre outros;
- Processa eventos a uma velocidade elevada, com baixa latência;
- É uma plataforma de processamento de dados de larga escala com capacidade de processar um grande volume de dados.

É também uma plataforma que responde a vários requisitos de uma maneira bastante eficaz e eficiente, nomeadamente:

- *Batch Processing*;
- *Interactive Processing*;
- *Real-time stream processing*;
- *Graph processing*.

É cada vez mais utilizado como uma alternativa ao *MapReduce*, usado no Hadoop, e consegue processar dados 100 vezes mais rápido que o *MapReduce*. Pode utilizar o HDFS do Hadoop para ler, escrever, guardar e processar dados e não tem um sistema de armazenamento dos mesmos.

Devido a todas estas funcionalidades descritas, vários desenvolvedores desenvolveram os mais diversos *benchmarks* de modo a medir a eficiência que o Apache Flink apresenta durante o processamento de dados. O Yahoo Streaming Benchmark e o StreamBench são dois *benchmarks* de referência que permitem então fazer essa análise ao Apache Flink e foram utilizadas para o desenvolvimento deste estudo.

II. Arquitetura do Apache Flink

Todos os tipos de dados são produzidos através de um fluxo de eventos, como por exemplo, transações de um cartão de crédito ou medidas registadas por sensores. Todos estes dados são gerados como um fluxo. Como tal, podem ser processados de duas maneiras distintas:

- Fluxo ilimitado;
- Fluxo limitado.

O fluxo ilimitado, tal como o nome sugere, tem um início mas não tem um fim definido. É um fluxo que nunca termina e está sempre a providenciar dados enquanto ele é gerado. Os eventos gerados por este fluxo devem ser sempre manuseados depois de terem sido processados pelo sistema. Visto que é ilimitado, é impossível esperar pela chegada de todos os dados pois estes nunca vão estar completos. Como tal, o seu processamento requer que os eventos sejam processados com um determinado critério de ordenação (e.g. instante em que se registou cada evento).

Por outro lado, o fluxo limitado carateriza-se por estar bem delimitado, tendo início e fim. Todos os dados presentes neste tipo de fluxo são processados antes de existir qualquer tipo de computação relacionada com eles. O processamento deste tipo de dados é também conhecido como *batch processing*.

A figura seguinte apresenta um gráfico que dá a entender o formato de ambos os tipos de fluxos, sendo possível ver a grande diferença entre eles e como é que estes se encontram ao longo de um fluxo de dados:

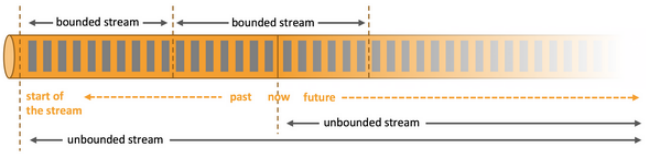

Fig. 1: Fluxos ilimitados e limitados.

Os dois conceitos anteriores são importantes pois o Apache Flink destaca-se no processamento de bases de dados limitadas e ilimitadas, devido ao preciso controlo do tempo e estado. Graças a esse controlo, é possível à plataforma correr qualquer tipo de fluxo ilimitado sem grandes complicações. Já os fluxos limitados, são processados internamente através de algoritmos e estruturas de dados desenhados para bases de dados com um tamanho fixo, aumentando assim a performance da plataforma.

Visto que o Apache Flink é um sistema distribuído e requer recursos computacionais de modo a que possa executar aplicações, está integrado com vários *softwares* de gestão de recursos em *cluster*, como é o caso do Apache Mesos e dos Kubernetes. No entanto, também pode ser executado de uma maneira independente.

Quando é lançada uma aplicação, a plataforma identifica automaticamente os recursos necessários baseado nas configurações da aplicação, sendo que esse pedido é feito então ao gestor de recursos. Toda esta comunicação é feita através de chamadas REST, um estilo de arquitectura de software que define um conjunto de restrições a serem usadas para a criação de serviços Web, o que facilita a integração nos mais diversos ambientes.

Permite também correr milhares de tarefas em paralelo e de maneira distribuída por vários *clusters*, o que permite às aplicações usarem uma quantidade ilimitada de recursos, como tempo de CPU, memória principal ou até mesmo rede.

Por último, as aplicações são otimizadas para acesso local. O estado da tarefa é mantido na memória principal da máquina e, sempre que excede o limite, acede às estruturas presentes no disco. Devido a todos os acessos serem locais, a latência do processamento é extremamente baixa. O Apache Flink garante então a consistência dos estados em caso de falha, através de mecanismos de *backup* que guardam o estado local da aplicação na memória da máquina. Todo este processo está representado na seguinte figura.

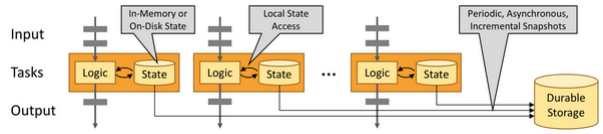

Fig. 2: Metodologia de acesso local das aplicações.

III. Instalação e Configuração do Flink

A instalação da plataforma Apache Flink foi feita para o sistema operativo Linux Ubuntu, versão 20.04. Esta plataforma necessita de várias funcionalidades/dependências extra para ser executada devidamente. Destas, destaca-se, em primeiro lugar, o interpretador Python, que pode ser instalado usando o seguinte comando:

```
$ sudo apt-get install
    python-software-properties
```

A segunda funcionalidade é o Java, um dos componentes principais para que o Apache Flink seja executado sem qualquer erro. Primeiramente, é preciso instalar o JRE (*Java Runtime Environment*) usando o comando:

```
$ sudo apt install default-jre
```

De seguida, é preciso também instalar o JDK (*Java Development Kit*), através do comando:

```
$ sudo apt install default-jdk
```

Após os passos anteriormente descritos, passamos então à instalação do Apache Flink.

O primeiro passo é fazer o *download* do ficheiro ".tar" do site oficial da plataforma.

De seguida, é necessário fazer a descompressão desse ficheiro, que pode ser feita através do comando:

```
$ tar xzf
    flink-<version>-bin-hadoop26-scala_2.11.tgz
```

onde $< version >$ é o número da versão da plataforma que foi descarregada.

Após a descompressão, é originada uma pasta na mesma directoria com o nome "flink-X", onde X é o número da versão do Apache Flink.

Por último, através do terminal, navega-se até à pasta "bin" dentro da pasta de instalação da plataforma e executa-se o ficheiro "start-clusters.sh", através do comando:

```
$ ./start-local.sh
```

Após todos estes passos, o Apache Flink está pronto a ser utilizado e considera-se como configurado. Para verificar que está operacional, pode-se aceder à Web UI oferecido pelo Apache Flink, que terá um aspeto semelhante ao da seguinte figura:

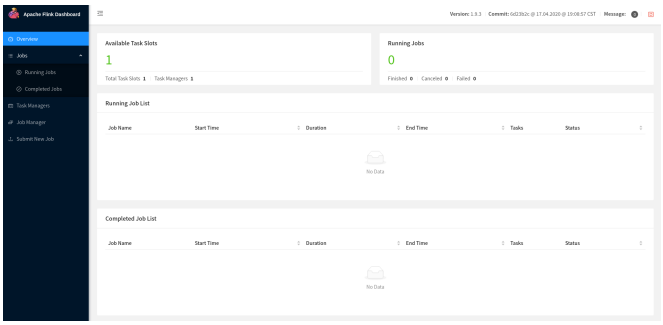

Fig. 3: Web UI do Apache Flink.

Também é possivel fazer essa verificação através do comando "jps", cuja saída da sua execução será semelhante à seguinte:

```
3814 StandaloneSessionClusterEntrypoint
4343 Jps
4249 TaskManagerRunner
```

IV. Benchmarks Usados para Avaliação do Desempenho

A presente secção terá como foco a descrição discussão de detalhes de implementação dos dois *benchmarks* utilizados para avaliação de desempenho do Apache Flink.

A. Yahoo Streaming Benchmark

O Yahoo Streaming Benchmark foi um programa previamente desenvolvido para correr juntamente com o Apache Storm. No entanto, com o decorrer dos anos, foi necessário começar a desenvolver um benchmark que corre-se juntamente com outras tecnologias, que tivessem maior rendimento e performance que o Apache Storm. Desta maneira, podia começar-se a dar recomendações aos clientes de qual a melhor ferramenta a ser usada em diversas situações.

Assim, foi desenvolvido este *benchmark*, que tem algumas funcionalidades úteis e que devolvem alguns resultados que podem ser analisados. No entanto, e como os próprios desenvolvedores afirmam, o programa desenvolvido tem problemas no que toca a abordar diversos aspetos de áreas fundamentais, pois não tem uma aplicação real, sendo apenas um desenvolvimento básico para alguns testes. Como tal, esta ferramenta apenas aborda um caso específico.

Começando então com a instalação e configuração do *benchmark*, é necessária a instalação de dois pré-requisitos: o software "Maven" e o software "Leiningen". Os seguintes comandos produzem esse efeito:

```
$ sudo apt-get install maven

$ sudo apt-get install leiningen
```

Após esta instalação, procede-se à execução do ficheiro "stream-bench.sh". A sua execução apresenta um menu de argumentos que podem ser utilizados aquando da sua execução. Como tal, o primeiro passo é fazer o *setup*, através do comando:

```
$ ./stream-bench.sh SETUP
```

Durante o setup poderão encontrar-se vários problemas. O primeiro problema ocorrer se o programa estiver a ser executado num JRE. Tal inviabiliza o *setup*, visto que este precisa de ser executado no JDK. Como tal, é preciso fazer essa mudança, sendo que uma possibilidade é desinstalar o Java e reinstalar apenas o JDK. É aconselhado o uso da versão 8.

O segundo problema encontrado foi a lentidão das transferências dos repositórios relativos ao software "APEX", uma vez que estes já não se encontravam disponíveis para *download* no *website* recomendado. Como tal, e visto que o "APEX" não era necessário, eliminou-se do código fonte tudo o que era relacionado com esse *software*, de maneira a aumentar a velocidade da compilação e execução do *setup*.

De resto, todas as transferências necessárias foram realizadas, bem como os demais passos, culminando no correto *setup* deste *benchmark* (finalizado com uma mensagem de sucesso).

Um facto a ter em conta é que este *benchmark* traz a sua própria instalação do Apache Flink, usando assim a versão 1.6.0 do *software*.

O próximo passo corresponde à inicialização do Apache Flink, com recurso ao comando:

```
$ ./stream-bench.sh START_FLINK
```

Deste modo, o servidor é inicializado, possibilitando a execução do programa com o Apache Flink.

Por último, é então executado o teste:

```
$ ./stream-bench.sh FLINK_TEST
```

Durante esta execução, são inicializados os servidores "Zookeeper", "Redis" e "Kafka", de modo a que a execução do teste seja bem sucedida. Um possível problema é a Virtual Machine do Java não ser bem inicializada, o que pode indicar que a versão do Java que está a ser utilizada não é a correta. Se este problema não ocorrer, vão ser então apresentados os resultados respetivos da execução do teste.

De notar que durante a execução, se acedermos ao *dashboard* do Apache Flink é possível acompanhar toda a execução do projeto em tempo real.

B. StreamBench

O StreamBench é um projeto que mede a performance de plataformas de *streaming* populares.

Durante a execução do projeto, é comparada a performance da plataforma "SABER", uma plataforma eficiente de processamento de fluxo de dados, usando dois sistemas de processamentos de dados: o Apache Spark e o Apache Flink. No entanto, para a análise efetuada apenas se vão ter em conta os resultados dados pelo Apache Flink. Também vai ser avaliada outra plataforma, a "StreamBox", que é uma plataforma que faz processamento de dados sem ordem num único servidor.

Para ser possível correr este benchmark, foi necessária a instalação do software "Maven", através do comando:

```
$ sudo apt-get install maven
```

De seguida, sendo que o Apache Flink já está a ser executado, foi preciso entrar na pasta "Flink", presente na pasta principal do *benchmark* e executar o ficheiro "run_flink.sh", através dos comandos:

```
$ cd /StreamBench/flink

$ ./run_flink.sh
```

Durante a execução deparou-se com um erro no qual o programa não encontrava classes necessárias à sua execução. Constatou-se que existia incompatibilidade do programa com a versão 11 do Java e, como tal, fez-se um *downgrade* para a versão 8, de modo a que fosse possível fazer a execução do programa.

Por último foram efetuadas as transferências de diversos módulos e repositórios necessários, sendo que após todo este processo começaram a ser dadas informações sobre a capacidade de processamento de dados e de latência durante o processamento efectuado pelo programa. Os resultados obtidos serão apresentados na seguinte secção.

V. Experiências e Resultados

A. Yahoo Streaming Benchmark

Para o *benchmark* da Yahoo, só foi possível obter valores de latência, uma vez que, na sua versão atual e testada, não está implementada qualquer métrica de *throughput* (capacidade de processamento).

A figura seguinte apresenta um excerto do *output* produzido por este *benchmark*:

```
Falling behind by: 132 ms
Falling behind by: 132 ms
Falling behind by: 131 ms
Falling behind by: 130 ms
Falling behind by: 129 ms
Falling behind by: 128 ms
Falling behind by: 127 ms
Falling behind by: 126 ms
Falling behind by: 125 ms
```

Fig. 4: Valores de latência para o Yahoo Streaming Benchmark.

Como é constatado na figura 4, este *benchmark* apresenta valores de latência consideráveis, o que pode condicionar a eficiência do mesmo em aplicações reais. De notar, que a imagem referida apresenta valores para um dos testes efetuados. Em outras instâncias de teste, os valores de latência chegaram a um mínimo de 101ms, sendo ainda elevado para o pretendido.

B. StreamBench

Por sua vez, este *benchmark* apresenta valores de latência e *throughput*, como se pode observar na seguinte figura:

```
Throughput:8000000.0, Average:8075250.0
ThroughputLogging:1591800131977,50000000
Throughput:8103000.0, Average:8080800.0
ThroughputLogging:1591800133211,60000000
Throughput:8103000.0, Average:8084500.0
ThroughputLogging:1591800134445,70000000
Throughput:8090000.0, Average:8085285.714285715
ThroughputLogging:1591800135681,80000000
Throughput:8130000.0, Average:8090875.0
ThroughputLogging:1591800136911,90000000
Throughput:8051000.0, Average:8086444.444444444
ThroughputLogging:1591800138153,100000000
Throughput:8103000.0, Average:8088100.0
ThroughputLogging:1591800139388,110000000
Max Latency is 4 msec. Latency is 4 msec
Throughput:8110000.0, Average:8090090.909090909
ThroughputLogging:1591800140620,120000000
Throughput:8210000.0, Average:8100083.333333333
ThroughputLogging:1591800141838,130000000
```

Fig. 5: Valores de latência e *throughput* para o StreamBench.

De acordo com os resultados obtidos, podemos constatar que, para diversas aplicações, um servidor de vários núcleos aplicado a um *cluster* tem uma melhor capacidade de processamento de fluxo de dados do que um servidor que usa vários *clusters*. Tal situação abre uma janela de oportunidade para que se reduzam custos operacionais ao substituir sistemas com vários nós de processamento.

C. Avaliação Comparativa

Com base nos resultados apresentados nas secções anteriores, podem-se derivar as seguintes conclusões:

- O *benchmark* StreamBench parece superiorizar-se em relação ao Yahoo Streaming Benchmark, reportando valores de latência bastante inferiores (o que é, naturalmente, uma propriedade desejável).
- O *benchmark* da Yahoo apresenta algumas lacunas em termos de implementação, concretizando-se na falta de mais métricas de desempenho (*throughput*) e/ou de outras funcionalidades fundamentais.
- Seguindo o ponto anterior, o StreamBench conta com mais métricas de avaliação, o que é benéfico e útil (particularmente para um estudo comparativo como o que se descreve no presente documento).

VI. Conclusão

O documento descrito procurou descrever, num primeiro momento, a arquitetura do Apache Flink e avaliar a sua performance com dois *benchmarks* distintos. No decurso da execução destes, foram encontrados alguns problemas, normalmente associados ao Java e às suas versões. Tal detalhe é claramente um fator a considerar em projetos de natureza semelhante.

A nível mais prático e com base nos testes realizados, pode-se afirmar que o StreamBench é superior ao equivalente da Yahoo, não só pelas funcionalidades a mais como pelos valores de latência inferiores. Deve, portanto, ser priorizado em aplicações reais.

Por fim, e tendo em conta que o projeto aqui descrito tem um caráter introdutório, as conclusões reportadas poderão ser complementadas e reforçadas num trabalho futuro, com a realização de mais testes e avaliação de outros *benchmarks*.

References

[1] Download Apache Flink [Online] https://flink.apache.org/downloads.html

[2] Download StreamBench [Online] https://github.com/lsds/StreamBench

[3] Download Yahoo Streaming Benchmark [Online] https://github.com/yahoo/streaming-benchmarks

[4] Instalar Flink no Ubuntu [Online] https://data-flair.training/blogs/apache-flink-installation-on-ubuntu/

[5] Arquitetura Apache Flink [Online] https://flink.apache.org/flink-architecture.html