Sqoop

Fernando Anselmo
http://fernandoanselmo.orgfree.com/wordpress/

Versão 1.00 em 14 de abril de 2022

Resumo

Sqoop[1] ou "Apache Sqoop"e em 2021 foi movido para página da Attic para manutenções posteriores (aqui chamarei apenas de Sqoop) é parte do Ecosistema Hadoop criado para a criação e manutenção de software relacionado a *Bulk Data Transfer* para Apache Hadoop e Datastores Estruturados.

1 Parte inicial

Sqoop é uma das ferramenta do ecossistema Hadoop usada para carregar dados de sistemas de gerenciamento de banco de dados relacional (RDBMS) para o Hadoop destinado ao processamento *MapReduce* e exportá-los de volta ao RDBMS. Simplificando, o Sqoop auxilia no trafego para grandes quantidades de dados entrando e saindo do Hadoop com os bancos tradicionais.

Figura 1: *Logo do Apache Sqoop*

Sqoop nasceu da necessidade de muitos dos dados precisavam ser transferidos a partir de banco de dados relacionais para o Hadoop, havia a necessidade de uma ferramenta dedicada para fazer essa tarefa rapidamente. Foi aí que entrou em cena, amplamente usado para transferir esses dados.

Quando se trata de transferir dados, há um certo conjunto de requisitos a serem atendidos. Inclui o seguinte:

- Dados devem ter consistência.
- O aprovisionamento do pipeline de downstream.

- Os usuários devem garantir o consumo dos recursos do sistema de produção

Lembremos que o *MapReduce* não pode acessar diretamente os dados que residem em bancos de dados relacionais externos, deve buscá-los no *Hadoop Distributed File System* (HDFS). Esse método pode expor o sistema ao risco como a geração de carga excessiva dos nós do cluster.

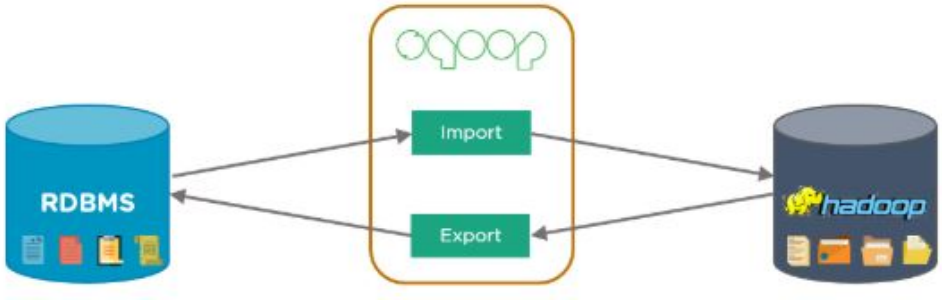

Figura 2: *Funcionalidade do Sqoop*

Para processar dados com o Hadoop é necessário que os dados precisam ser carregados em clusters do Hadoop de várias fontes. No entanto, descobriu-se que o processo para o carregamento dos dados de várias fontes heterogêneas era extremamente desafiador. Os problemas comuns que os administradores encontraram:

- Manter a consistência dos dados.
- Garantir a utilização eficiente dos recursos.
- Não era possível obter os dados em massa para o Hadoop.
- Carregar os dados ao utilizar *script* era lento.

A solução foi Sqoop. O uso do Sqoop no Hadoop ajudou a superar todos os desafios da abordagem tradicional e pode carregar dados em massa do RDBMS para o Hadoop com facilidade. Devemos imaginar o fluxo de dados de uma empresa que executa uma importação noturna do Sqoop para carregar os dados do dia de um RDBMS transacional de produção em um *data warehouse* do **Hive** para análise posterior.

2 Hadoop no Docker

O modo mais simples de se conseguir trabalhar com o Hadoop é utilizando o Docker, para baixar a imagem do Hadoop:

```
$ docker pull suhothayan/hadoop-spark-pig-hive:2.9.2
```

Nessa imagem temos outros produtos do Ecossistema Hadoop: Spark, Pig e Hive. Para criar e executar a primeira vez o contêiner (a pasta que este comando for executado será associada a uma pasta interna chamada **/home/tsthadoop**):

```
$ docker run -it -d --name meu-hadoop -v $(pwd):/home/tsthadoop
suhothayan/hadoop-spark-pig-hive:2.9.2
```

Uma vez interrompido o contêiner:

```
$ docker stop meu-hadoop
```

Podemos executá-lo novamente com os seguintes comandos:
```
$ docker start meu-hadoop
$ docker exec -it meu-hadoop /etc/bootstrap.sh bash
```

2.1 Erro de Permissão

Na primeira vez que entramos é dado um erro na execução do script "bootstrap.sh"de permissão negada para executar o script "spark-env.sh", vamos corrigir isso com o comando:
```
# chmod 777 /usr/local/spark/conf/spark-env.sh
```

Vamos sair do bash:
```
# exit
```

Podemos executá-lo novamente:
```
$ docker exec -it meu-hadoop /etc/bootstrap.sh bash
```

E o erro desapareceu.

2.2 Instalar o Sqoop

Porém se leu corretamente essa imagem NÃO POSSUI o Sqoop e agora realmente começa nossa instalação, optei por utilizar essa imagem por já possuir todos os componentes necessários para posterior tratamento dos dados. Desta forma ao invés de realizar um trabalho de obter uma imagem somente com o Sqoop (que existem várias por aí e ter que instalar todos os outros componentes), vamos utilizar uma imagem que estamos acostumados e que já possui todas as outras ferramentas necessárias e nos preocuparmos apenas com o Sqoop.

Outros ganho adicionais desse método é que também pode ser excelente para aprendermos como se adiciona uma ferramenta ao ecossistema do Hadoop além de não ficarmos presos a uma determinada versão (que muitas imagens ainda estão com a versão 1.4.7). E agora podemos "brincar"melhor com as outras ferramentas como o Spark, Hive e Pig que já estão nessa imagem.

No endereço oficial [**?**] existe o link para Download, acessando este vemos todas as versões disponíveis, utilizaremos a última até o momento, clicar no link "1.99.7"e baixar o arquivo **sqoop-1.99.7-bin-hadoop200.tar.gz** e colocá-lo na pasta associativa da imagem.

Retornar a pasta raiz do root:
```
# cd root/
```

Descompactar o arquivo:
```
# tar vfx /home/tsthadoop/sqoop-1.99.7-bin-hadoop200.tar.gz
```

Mover o pasta descompactada para o endereço local (aonde estão os outros produtos do Ecossistema):
```
# mv sqoop-1.99.7-bin-hadoop200 /usr/local/.
```

Criar um link simbólico para facilitar o acesso:
```
# ln -s /usr/local/sqoop-1.99.7-bin-hadoop200 sqoop
```

No arquivo core-site.xml do Hadoop habilitar os acessos, editar o arquivo:
```
# vim $HADOOP_PREFIX/etc/hadoop/core-site.xml.template
```

Pressionar "i"para entrar em modo de edição e adicionar as seguintes propriedades:

```
<property>
  <name>hadoop.proxyuser.sqoop2.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.sqoop2.groups</name>
  <value>*</value>
</property>
```

Pressionar "ESC"para sair do modo de edição e salvar o arquivo pressionando ":wq". Na pasta de configuração do Sqoop precisamos modificar o caminho do Hadoop, editar o arquivo:
`# vim /usr/local/sqoop/conf/sqoop.properties`

Localizar a linha:
`# Hadoop configuration directory`

E modificar o valor da variável (entrar em modo de edição pressionando a tecla "i") para:

```
org.apache.sqoop.submission.engine.mapreduce.configuration.directory =
    /usr/local/hadoop/share/hadoop/mapreduce
```

Pressionar "ESC"para sair do modo de edição e salvar o arquivo pressionando ":wq". O Sqoop precisa acessar bancos de dados, sendo assim precisamos adicionar os arquivos JAR de JDBC dos bancos que desejamos inserir. Criar uma pasta:
`# mkdir -p /var/lib/sqoop2/`

Copiar para esta pasta os arquivos JAR necessários. Vamos modificar o arquivo "etc/bootstrap.sh"o seguinte EXPORT:
`# vim /etc/bootstrap.sh`

E adicionar as seguintes linhas abaixo dos comandos EXPORT:

```
export PATH=$PATH:/usr/local/sqoop/bin
export SQOOP_SERVER_EXTRA_LIB=/var/lib/sqoop2/
```

Aproveitamos e inserimos o caminha do path para a pasta do Sqoop. Pressionar "ESC"para sair do modo de edição e salvar o arquivo pressionando ":wq". Sair da imagem, parar e iniciar o contêiner, entrar no bash novamente.

Acessar a pasta associativa:
`# cd /home/tsthadoop`

Verificar se o Sqoop está correto:
`# sqoop2-tool verify`

Se aparecer a mensagem "*Verification was successful.*"está tudo bem, porém caso mostrar "*Verification has failed, please check Server logs for further details.*"devemos verificar o arquivo de Log que se encontra nas pasta @LOGDIR@.

2.3 Executar o Sqoop

Com tudo OK, executamos o servidor do Sqoop:
```
# sqoop2-server start
```

Que deve mostrar a mensagem "*Sqoop2 server started.*". Por padrão utiliza a porta 12000. Podemos modificar a porta na variável "org.apache.sqoop.jetty.port"no arquivo de configuração em "conf/sqoop.properties". Podemos pará-lo com o comando:
```
# sqoop2-server stop
```

Acessar o cliente:
```
# sqoop2-shell
```

Ver os comando disponíveis:
```
sqoop:000> help
```

Agora vamos a parte divertida, a real utilidade do Sqoop que é Importar/Exportar dados de RDBS, para isso utilizamos *Java Database Connectivity* (JDBC).

3 Passos Iniciais

Acessar a pasta associativa:
```
# cd /home/tsthadoop
```

Iniciar o servidor:
```
# sqoop2-server start
```

Com a resposta: "*Sqoop2 server started.*"acessar o cliente:
```
# sqoop2-shell
```

Verificar a versão:
```
sqoop:000> show version -a
```

Conectar o cliente ao servidor:
```
sqoop:000> set server --host 172.17.0.2 --port 12000 --webapp sqoop
```

Após mostrar a mensagem "*Server is set successfully*", verificar se fez efeito:
```
sqoop:000> show server -a
```

Ver os conectores disponíveis:
```
sqoop:000> show connector
```

Ao adicionar a opção -a teremos mais informação:
```
sqoop:000> show connector -a
```

Podemos usar essa opção -a para todos os comandos "show". Porém isso não afetara as respostas quando estivermos importando ou exportando dados, assim vamos deixar o Sqoop nos responder de modo verboso:
```
sqoop:000> set option --name verbose --value true
```

Verificar se fez efeito:
```
sqoop:000> show option
```

Sair do cliente:
```
sqoop:000> :x
```

Encerrar o servidor do Sqoop:
```
# sqoop2-server stop
```

4 Criar uma conexão

Para realizar uma conexão com qualquer RDBMS precisamos de conectores registrados, um deles cria uma conexão JBDC e é esse que utilizaremos, porém precisamos ter o banco de dados a nossa disposição, já que estamos usando o Docker podemos sair do contêiner e criar um novo com o banco que desejamos testar, aqui utilizaremos o MySQL e que este sirva como ponto de partida e entendimento para qualquer outro pois a conexão é basicamente a mesma.

Baixar a imagem oficial:
```
$ docker pull mysql
```

Criar uma instância do banco em um Contêiner:
```
$ docker run --name meu-mysql -e MYSQL_ROOT_PASSWORD=root -p 3306:3306 -d mysql
```

Nas próximas vezes, usamos para iniciar o MySQL:
```
$ docker start meu-mysql
```

Parar o MySQL:
```
$ docker stop meu-mysql
```

Entrar diretamente no gerenciador o comando:
```
$ docker exec -it meu-mysql sh -c 'exec mysql -u root -p'
```

Ou então:
```
$ docker exec -it meu-mysql bash
# mysql -u root -p
```

No MySQL precisamos de um banco, no gerenciado criamos:
```
mysql> CREATE DATABASE tstsqoop;
```

Entrar no banco criado: `mysql> USE tstsqoop;`

Criamos as tabelas que realizaremos as importações e exportações de dados:
```
mysql> CREATE TABLE recebimento (codigo int not null, estado char(2),
valorrec float(5,2), PRIMARY KEY (codigo));
mysql> CREATE TABLE analitico (estado char(2), valortotal float(8,2),
PRIMARY KEY (estado));
```

A tabela de recebimento contém os valores recebidos dos estados, esses devem ser contabilizados na tabela analítico. Adicionamos alguns registros na tabela recebimento:
```
mysql> INSERT INTO recebimento VALUES (1, 'DF', 120.00);
mysql> INSERT INTO recebimento VALUES (2, 'DF', 180.00);
mysql> INSERT INTO recebimento VALUES (3, 'DF', 80.00);
mysql> INSERT INTO recebimento VALUES (4, 'MG', 132.30);
mysql> INSERT INTO recebimento VALUES (5, 'MG', 22.60);
```

Adicionar outros pois quanto mais valores melhor. E nosso trabalho aqui está feito, sair do MySQL:
`mysql> quit`

NÃO parar este contêiner pois precisamos dele ativo. Agora necessitamos de um drive JDBC de conexão, este é um arquivo padrão **.JAR** que pode ser obtido gratuitamente em vários sites. Por exemplo em `https://downloads.mysql.com/archives/c-j/`, selecionar como sistema operacional "*Platform Independent*". Descompactar o arquivo que foi realizado dowload e neste encontramos um arquivo .JAR chamado "mysql-connector-java-8.0.27.jar".

Copiar esse arquivo para o diretório compartilhado na máquina local, vamos retornar para o contêiner do Hadoop:
`$ docker exec -it meu-hadoop2 /etc/bootstrap.sh bash`

Acessar a pasta das bibliotecas JDBC do Sqoop:
`# cd /var/lib/sqoop2/`

Copiar o arquivo JDBC:
`# mv /home/tsthadoop/mysql-connector-java-8.0.27.jar .`

Acessar a pasta associativa:
`# cd /home/tsthadoop`

Copiar os dois arquivos conectores que realizam a conexão nesta pasta:
```
# cp /usr/local/sqoop-1.99.7-bin-hadoop200/server/lib/
sqoop-connector-generic-jdbc-1.99.7.jar .
# cp /usr/local/sqoop-1.99.7-bin-hadoop200/server/lib/
sqoop-connector-hdfs-1.99.7.jar .
```

Iniciar o servidor:
`# sqoop2-server start`

Com a resposta: "*Sqoop2 server started.*"acessar o cliente:
`# sqoop2-shell`

Verificar se o servidor está conectado:
`sqoop:000> show server -a`

Caso não esteja conectar o cliente ao servidor. Criar um link de conexão com o banco:
`sqoop:000> create link -connector generic-jdbc-connector`

Os parâmetros são:

```
Name: Link com MySQL
Driver Class: com.mysql.cj.jdbc.Driver
JDBC Connection String: jdbc:mysql://[IP da Máquina]/tstsqoop
Username: root
Password: root
Fetch Size:
entry# protocol = tcp
entry# useSSL = false
entry#
Identifier enclose: [colocar um espaço aqui]
```

Ao término mostra que foi estabelecido uma Conexão com o banco MySQL. A beleza é que ao

criar um contêiner do MySQL é como se esse tivesse instalado na nossa máquina local pois existe uma associação com a porta 3606, por isso informamos o IP da Máquina local ao invés do IP do Contêiner. Os parâmetros que estão em brancos basta pressionar a tecla ENTER.

O parâmetro *Identifier enclose* deve ser colocado um espaço, pois caso contrário tentará colocar " (aspas duplas) na chamada do schema (que não existe para o conector do MySQL) e na tabela.

Criar um link com o HDFS do Hadoop, com o comando:
`sqoop:000> create link -connector hdfs-connector`

Os parâmetros são:

```
Name: Link com Hadoop
URI: hdfs://localhost:9000
Conf directory: /usr/local/hadoop/etc/hadoop
entry#
```

Verificar os links criados:
`sqoop:000> show link`

Mudar qualquer opção utilizar o comando:
`sqoop:000> update link -name "[nome link]"`

Eliminar um link: `sqoop:000> delete link -name "[nome link]"`

5 Importar Dados

Este é o processo no qual as tabelas individuais são importadas do RDBMS para o HDFS. Para fins de transferência, a linha em uma tabela é considerada um registro no HDFS. Os dados gravados são armazenados na forma de dados de texto em arquivos de texto ou são armazenados em arquivos Sequence e Avro como dados binários.

Importação (ou mesmo a Exportação) de dados é o movimento de um link para outro, isso é realizado através de um Job, para criarmos o Job de Importação usamos:
`sqoop:000> create job -f "Link com MySQL" -t "Link com Hadoop"`

Os parâmetros são:

```
Name: Recebimento
Schema name:
Table name: recebimento
SQL statement:
Column names:
element# estado
element# valorrec
element#
Partition column: codigo
Partition column nullable:
Boundary query:
Check column:
Last value:
Override null value:
Null value:
```

```
File format: 0
Compression codec: 0
Custom codec:
Output directory: hdfs://localhost:9000/user/root/tstsqoop
Append mode:
Extractors: 1
Loaders: 1
Extra mapper jars:
There are currently 0 values in the list:
element#
```

Como resposta temos: *New job was successfully created with validation status OK and name Recebimento*. Podemos ver o Job com:
`sqoop:000> show job`

Para executar o Job:
`sqoop:000> start job -name Recebimento`

Os dados foram colocados no HDFS e aplicado um Map/Reduce, saímos do shell do Sqoop:
`sqoop:000> :x`

Verificamos a pasta destino do HDFS:
`# hdfs dfs -ls /user/root/tstsqoop`

E encontraremos o arquivo (ATENÇÃO os números podem variar pois são gerados automaticamente com base em um HASH) importado, no MEU caso o arquivo é:

```
Found 1 items
-rw-r--r--   1 root supergroup         20 2022-04-03 17:42
    /user/root/tstsqoop/60ce7fbc-1d59-4da1-ae2a-86591beae4d8.txt
```

Verificamos seu conteúdo com:
`# hdfs dfs -cat /user/root/tstsqoop/60ce7fbc-1d59-4da1-ae2a-86591beae4d8.txt`

E veremos os dados do banco MySQL conforme solicitado. Que podemos utilizar para realizar um Map/Reduce ou qualquer outra ação.

Mais fontes de informação podem ser obtidas em diversos sites que apresenta tutoriais completos sobre o Sqoop como a Tutorials Point[2].

6 Conclusão

É inegável que para quem utiliza Hadoop, o trabalho interessante começa depois que os dados são carregados no HDFS. Os desenvolvedores podem "brincar"com os dados para encontrar os *insights* mágicos ocultos nesse Big Data. Para isso, os dados que residem nos sistemas de RDBS precisam ser transferidos ao HDFS, trabalhar com os dados e serem enviados de volta aos RDBS. A realidade do mundo **Big Data**, os desenvolvedores sentem que a transferência de dados entre sistemas RDBS e HDFS não é tão interessante, tediosa, mas necessária. Os desenvolvedores sempre podem escrever scripts personalizados para transferir dados para dentro e para fora do Hadoop (que como já visto é um processo mais lento), assim o Sqoop oferece uma alternativa excelente para resolver esse problema.

Sqoop automatiza a maior parte do processo, depende do banco de dados para descrever o esquema dos dados a serem importados. Utiliza a estrutura MapReduce para importar e exportar os dados, que fornece mecanismo paralelo e tolerância a falhas. Facilita a vida dos administradores ao fornecer uma interface de linha de comando. Necessitamos apenas inserir informações básicas como detalhes de autenticação de origem, destino e banco de dados no comando Sqoop que cuida da parte restante.

Sou um entusiasta do mundo **Open Source** e novas tecnologias. Qual a diferença entre Livre e Open Source? Livre significa que esta apostila é gratuita e pode ser compartilhada a vontade. Open Source além de livre todos os arquivos que permitem a geração desta (chamados de arquivos fontes) devem ser disponibilizados para que qualquer pessoa possa modificar ao seu prazer, gerar novas, complementar ou fazer o que quiser. Os fontes da apostila (que foi produzida com o LaTex) está disponibilizado no GitHub [5]. Veja ainda outros artigos que publico sobre tecnologia através do meu Blog Oficial [3].

Referências

[1] Página do Sqoop
`https://attic.apache.org/projects/sqoop.html`

[2] Tutorials Point sobre Sqoop
`https://www.tutorialspoint.com/sqoop/index.htm`

[3] Fernando Anselmo - Blog Oficial de Tecnologia
`http://www.fernandoanselmo.blogspot.com.br/`

[4] Encontre essa e outras publicações em
`https://cetrex.academia.edu/FernandoAnselmo`

[5] Repositório para os fontes da apostila
`https://github.com/fernandoans/publicacoes`