Hive

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

Versão 1.0 em 1 de março de 2022

Resumo

Hive[1] ou "Apache Hive"(aqui chamarei apenas de Hive) é parte do Ecosistema Hadoop criado para ser um sistema de *Data Warehouse* que é usado para analisar dados estruturados. Foi construído no topo do Hadoop e desenvolvido pelo Facebook. Como resultado, o Hive está intimamente integrado ao Hadoop e foi projetado para funcionar rapidamente em petabytes de dados. O que torna único é a capacidade de consultar grandes conjuntos de dados, aproveitando Apache Tez ou MapReduce, com uma interface semelhante a SQL.

1 Parte inicial

O Hive fornece a funcionalidade de leitura, gravação e gerenciamento de grandes conjuntos de dados que residem em armazenamento distribuído. Executa linguagem semelhantes ao SQL padrão como consultas chamadas HQL (Hive query language) que são convertidas internamente em trabalhos MapReduce. Usar o Hive significa que podemos pular o requisito da abordagem tradicional de escrever programas MapReduce complexos. O Hive suporta linguagem de definição de dados (DDL), linguagem de manipulação de dados (DML) e funções definidas pelo usuário (UDF).

Figura 1: *Logo do Apache Hive*

Além disso tudo, por ser criado utilizando *Open Source* a comunidade Weka e muito ativa e constantemente produz *plugins* (chamados de pacotes) que podem ser incorporados e fornece funcionalidades não previstas originalmente.

Kafka fornece um banco de dados de streaming de vários sistemas de origem. Kafka propõe ter os seguintes recursos mencionados para seus bancos de dados de streaming:

- Rápido e escalável.
- Fornece consultas semelhantes a SQL (ou seja, HQL) que são transformadas implicitamente em trabalhos MapReduce ou Spark.
- É capaz de analisar grandes conjuntos de dados armazenados em HDFS.
- Permite diferentes tipos de armazenamento, como texto simples, RCFile e HBase.
- Utiliza indexação para acelerar as consultas.
- Pode operar em dados compactados armazenados no ecossistema Hadoop.
- Suporta funções definidas pelo usuário (UDFs) onde o usuário pode fornecer sua funcionalidade.

2 Arquitetura do Hive

A seguinte imagem explica como se comporta o fluxo de envio de consulta no Hive:

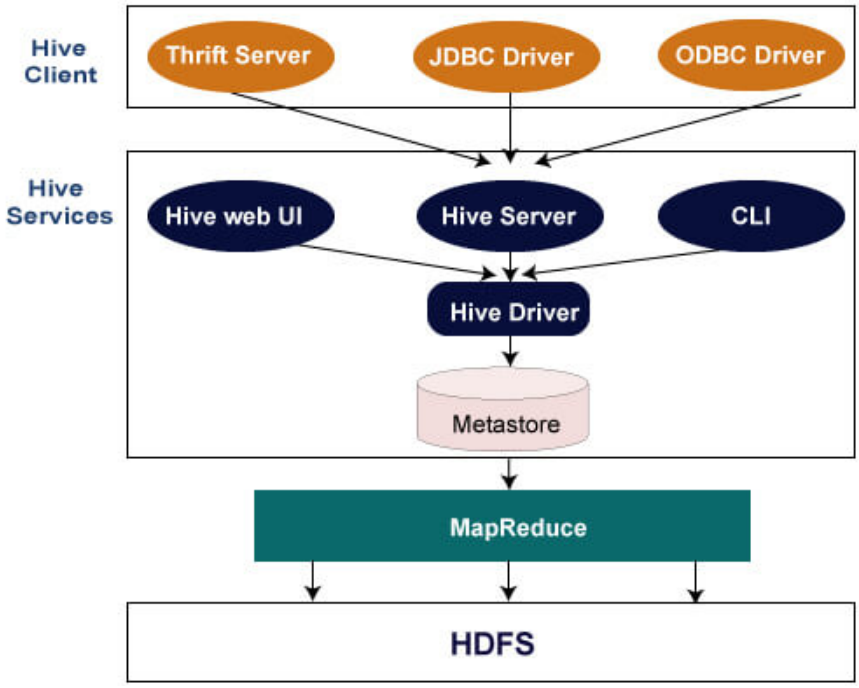

Figura 2: *Arquitetura do Apache Hive*

Na primeira parte do *Hive Client* significa que este permite escrever aplicativos em várias linguagens, incluindo Java, Python e C++. Ele suporta diferentes tipos de clientes, como:

- *Thrift Server* - É uma plataforma de provedor de serviços multilíngue que atende a solicitação de todas as linguagens de programação que suportam o Thrift.
- *Driver JDBC* - É usado para estabelecer uma conexão entre o hive e os aplicativos Java. O Driver JDBC está presente na classe org.apache.hadoop.hive.jdbc.HiveDriver.
- *Driver ODBC* - Permite que os aplicativos que suportam o protocolo ODBC se conectem ao Hive.

Na segunda parte do *Hive Services* temos:

- *Hive Web UI* (Interface de usuário da Web do Hive) - é uma alternativa da CLI do Hive. Fornece uma GUI baseada na Web para executar consultas e comandos do Hive.
- *Hive Server* - É referido como Apache Thrift Server. Aceita a solicitação de diferentes clientes e a fornece ao Hive Driver.
- *Hive CLI* (Command Line Interface) - É um shell onde podemos executar consultas e comandos do Hive.
- *Hive Driver* - Recebe consultas de diferentes fontes, como UI da web, CLI, Thrift e driver JDBC/ODBC. Transfere as consultas para o compilador.
- *Hive MetaStore* - Repositório central que armazena todas as informações de estrutura de várias tabelas e partições do *WareHouse*. Também inclui os metadados das colunas e suas informações, serializadores e desserializadores pode ser usados para ler e gravar dados e nos arquivos HDFS correspondentes onde os dados são armazenados.

Temos ainda duas ferramentas destinadas a suportar essa camada:

- *Hive Compiler* - O objetivo do compilador é analisar a consulta e realizar análise semântica nos diferentes blocos e expressões de consulta. Ele converte instruções HiveQL em trabalhos MapReduce.
- *Hive Execution Engine* - Otimizador responsável por gerar um plano lógico na forma de DAG de tarefas de redução de mapa e tarefas HDFS. No final, o mecanismo de execução executa as tarefas recebidas na ordem de suas dependências.

2.1 Criar o contêiner Docker

O Hive é bem mais intricado e dependente do ambiente do Hadoop que qualquer outro do ecossistema desse, os passos para conseguirmos instalar o Hive já foram vistos na Apostila do Hadoop, porém vamos reproduzir nesta.

Baixar a imagem oficial do Hadoop com o Spark, Pig e Hive:
```
$ docker pull suhothayan/hadoop-spark-pig-hive:2.9.2
```

Nessa imagem teremos ainda outros produtos do Ecossistema Hadoop como o Spark, Pig e Hive que trataremos em outras apostilas.

Para criar e executar a primeira vez o contêiner (a pasta que este comando for executado será associada a uma pasta interna chamada **/home/hadoop**):
```
$ docker run -it --name meu-hadoop -p 50070:50070 -p 8088:8088 -p 8080:8080
-p 10000:10000 -v $(pwd):/home/tsthadoop suhothayan/hadoop-spark-pig-hive:2.9.2
bash
```

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 bash
```

E podemos verificar a versão do Hadoop e Hive que estão instalados:
```
$ hadoop version
$ hive --version
```

O endereço da pasta do Hive:
```
$ cd /usr/local/hive
```

3 Uma pasta para seus Dados

Observe que no comando de criação do contêiner foi associada uma pasta (que é a pasta atual) com uma pasta interna deste, então por exemplo, se estamos na pasta "/Aplicativos/hadoop-model"e iniciamos o contêiner essa pasta estará associada.

No caso do Hive este cria uma pasta chamada "metastore_db"e isso pode ser um problema, pois se mudamos de pasta o Hive pode até mesmo não iniciar, então na primeira vez que executar o Hive tenha a certeza da pasta que se encontra.

E inclusive não se esqueça de sempre estar na pasta associada quando for iniciar o Hive:
```
# cd /home/tsthadoop/
# hive
```

3.1 Se caso der problema?

Pode acontecer dessa pasta se corromper, aí não temos outra saída a não ser recriar a pasta, remova esta do diretório:
```
# rm -rf metastore_db
```

Recriá-la com o comando:
```
# schematool -dbType derby -initSchema
```

Isso mesmo que percebemos isso é uma pasta do banco "Apache Derby".

4 Comando SQL

O Hive trabalha essencialmente com comandos "similares"aos do SQL, vamos começar olhando quais são as bases existentes:
```
hive> show databases;
```

Criar uma nova base de dados:
```
hive> create database amostra;
```

Entrar nessa base de dados:
```
hive> use amostra;
```

Verificar as tabelas existentes:
```
hive> show tables;
```

4.1 Buscar dados do Hadoop

Agora vamos brincar um pouco da integração entre o Hive e o Hadoop, para isso existe um arquivo chamado "201808_BF_Amostra.csv"que deve ser adicionado ao HDFS, para isso, saimos do Hive:
`hive> exit;`

Adicionar os arquivos no HDFS:
`# hadoop fs -put 201808_BF_Amostra.csv`

Retornamos ao hive:
`# hive`

E entre novamente no database que criamos:
`hive> use amostra;`

Não é por escolha minha, mas infelizmente todos os campos dessa base são do tipo String, foram todos colocados em aspas duplas, então como desejo apenas testar a funcionalidade vamos criar todos os campos da tabela assim:

```
hive> create table amostra2018 (anoPag string, ano2 string, estado string,
cod string, cidade string, cpf string, nome string, valor string) ROW FORMAT
DELIMITED FIELDS TERMINATED BY ";";
```

O comando SQL é basicamente a mesma coisa o final é que existem algumas palavras chaves indicando para o Hive como deve se comportar, para importarmos os dados:
`hive> LOAD DATA INPATH '201808_BF_Amostra.csv' into table amostra2018;`

Podemos verificar os dados com:
`hive> select * from amostra2018 limit 10;`

Ou sem usar a cláusula limit para vermos todos. Agora o mais interessante é, saia do Hive e digitar o comando:
`# hadoop fs -ls`

E descobrimos que o arquivo "sumiu"do HDFS, pois o Hive disse para o Hadoop: "Ele agora é meu!". E ficou com a guarda do arquivo.

5 Acesso via JDBC

Nada disso teria graça senão usássemos Java para demonstrar as ligações do Hive, para isso vamos utilizar o Spring Tool Suite no qual criamos um projeto *Maven Project*. Só que antes de fazermos isso precisamos configurar algumas questões de segurança.

Primeiro precisamos saber qual o nosso hostname:
`# hostname`

Anote esse número (que é o CONTEINER ID), de posse deste, vamos editar o arquivo **core-site.xml** do Hadoop com o seguinte comando:
`# vim /usr/local/hadoop-2.9.2/etc/hadoop/core-site.xml`

Deixe-o com a seguinte codificação:

```
1 <configuration>
```

```
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://[hostname]:9000</value>
  </property>
  <property>
    <name>hadoop.proxyuser.root.hosts</name>
    <value>*</value>
  </property>
  <property>
    <name>hadoop.proxyuser.root.groups</name>
    <value>*</value>
  </property>
  <property>
    <name>hive.execution.engine</name>
    <value>spark</value>
  </property>
</configuration>
```

Para salvar pressione ESC e digite **:wq**. Essas configurações permitirão que o usuário root consiga atravessar o proxy do Hadoop e acessar o Hive. Agora precisamos ativar o serviço do Hive para permitir a conexão:

```
# hive --service hiveserver2 &
```

Quando terminar de processar basta dar Enter para retornar ao terminal e podemos testar se está funcionando com o comando **beeline**:

```
# beeline -u jdbc:hive2://[hostname]:10000/ -n root
```

Que deve entrar no bash do Hive, e damos o comando: `dbc:hive2://[hostname]:10000/> show databases;`

Para verificarmos as bases de dados ativas e receberemos a **default**. E agora estamos prontos.

A grande vantagem de criarmos um projeto tipo Maven que podemos inserir todas as bibliotecas que necessitamos sem termos o menor trabalho, basta apenas buscamos estas do repositório central do Maven através de sua declaração no arquivo **pom.xml**:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>hiveproj</groupId>
  <artifactId>hiveproj</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Projeto em Hive</name>
  <description>Exemplo do projeto em Hive</description>
  <dependencies>
    <dependency>
      <groupId>org.apache.hive</groupId>
      <artifactId>hive-jdbc</artifactId>
      <version>2.3.5</version>
    </dependency>
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-common</artifactId>
      <version>2.9.2</version>
```

```
    </dependency>
  </dependencies>
</project>
```

Assim de forma simples temos as bibliotecas do Hive. Vamos começar com uma classe que além de testar nossa conexão criará uma tabela:

```
package hiveproj;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class HiveTeste {
  private static String driverName = "org.apache.hive.jdbc.HiveDriver";

  public static void main(String[] args) throws SQLException {
    try{
      Class.forName(driverName);
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
      System.exit(1);
    }

    Connection con =
    DriverManager.getConnection("jdbc:hive2://172.17.0.2:10000/default", "root", "");
    Statement stmt = con.createStatement();
    String tableName = "HiveByJava";
    stmt.execute("drop table if exists " + tableName);
    stmt.execute("create table " + tableName +
      " (key int, value string) ROW FORMAT DELIMITED FIELDS TERMINATED BY \",\"");
    System.out.println("Tabela criada com sucesso!");

    String sql = "show tables";
    System.out.println("Executando: " + sql);
    ResultSet res = stmt.executeQuery(sql);
    if (res.next()) {
       System.out.println(res.getString(1));
    }

    sql = "describe " + tableName;
    System.out.println("Executando: " + sql);
    res = stmt.executeQuery(sql);
    while (res.next()) {
      System.out.println(res.getString(1) + "\t" + res.getString(2));
    }

    System.out.println("Tabela criada sem problemas");
  }
}
```

Não pretendo ficar "enchendo linguiça"nessa apostila mostrando o que é possível realizar com uma conexão JDBC (para isso inclusive tenho meu primeiro livro de Java escrito), mas uma dica, não realizar comandos de inserção pois o Hive não gosta muito deles e tem que acertar várias bases.

Então se precisar inserir dados na tabela use a técnica mostrada no início.

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

6 Conclusão

Hive é um sistema de armazenamento de dados distribuído e tolerante a falhas que permite análises de dados em grande escala. Um data warehouse fornece um armazenamento central de informações que podem ser facilmente analisadas para tomar decisões informadas e orientadas por dados. Permite que os usuários leiam, gravem e gerenciem petabytes de dados usando SQL.

Hive foi criado para permitir que não programadores familiarizados com SQL trabalhem com petabytes de dados, usando uma interface semelhante a SQL chamada HiveQL. Os bancos de dados relacionais tradicionais são projetados para consultas interativas em conjuntos de dados pequenos e médios e não processam bem conjuntos de dados enormes. Em vez disso, usa processamento em lote para funcionar rapidamente em um banco de dados distribuído muito grande.

Hive transforma as consultas HiveQL em trabalhos MapReduce ou Apache Tez que são executados na estrutura de agendamento de trabalho distribuído do Apache Hadoop ou do YARN (*Yet Another Resource Negotiator*). Consulta dados armazenados em uma solução de armazenamento distribuído, como o HDFS (*Hadoop Distributed File System*) ou o Amazon S3. Armazena seus metadados de banco de dados e tabela em um metastore, que é um banco de dados ou armazenamento com suporte de arquivo que permite fácil abstração e descoberta de dados.

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 Apache Hive
`https://hive.apache.org/`

[2] Tutorials Point sobre Hive
`https://www.tutorialspoint.com/hive/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`