FUNDAMENTOS DE BIG DATA

Roger Robson dos Santos

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Hive na prática: uso e comandos no Hadoop

Roger Robson dos Santos

OBJETIVOS DE APRENDIZAGEM

- Determinar todos os passos necessários para utilização do Hive no Hadoop.
- Descrever de maneira prática o processo de análise de dados no Hadoop por meio do Hive.
- Reconhecer os requisitos para análise de dados por meio do Hive.

Introdução

Entender os processos de uma aplicação que analisa grandes volumes de dados é uma tarefa difícil, mas necessária. Afinal, o mercado atual exige profissionais que saibam utilizar ferramentas como o Hadoop para trabalhar com a enorme quantidade de dados coletados diariamente. Assim, entender o Hadoop e as suas ferramentas é essencial para obter uma vaga privilegiada no mercado.

Neste capítulo, você vai conhecer as técnicas de utilização do Hadoop e suas ferramentas. Além disso, vai ver como utilizar a ferramenta Hive para realizar consultas práticas no Hadoop.

Utilização do Hive no Hadoop

O Apache Hadoop é uma plataforma criada para realizar trabalhos com grandes volumes de dados em *hardware* comum. O Hadoop permite a utilização de *clusters* de computadores que possibilitam ampliar seu *hardware*. Além disso, esse *software* possui ferramentas que detectam e tratam falhas, tornando o sistema altamente disponível para utilização de máquinas em *cluster*. O Hadoop é desenvolvido na linguagem Java (APACHE HADOOP, 2020).

O Hadoop é formado por diversos componentes. A seguir, veja quais são os componentes principais (DEAN; GHEMAWAT, 2004).

- **Hadoop Common:** é um componente que contém utilitários usados em toda a aplicação. Possui diversas bibliotecas para manipulação e serialização de arquivos. Esse componente também é utilizado para disponibilizar integrações de interface para outros sistemas, como CloudSource e AWS S3.
- **Hadoop Distributed File System (HDFS):** é um sistema de arquivos distribuído altamente tolerante a falhas que permite o armazenamento e a manipulação de grandes conjuntos de dados em *clusters* de máquinas de baixo custo.
- **Hadoop MapReduce:** é uma biblioteca especializada no processamento de grandes conjuntos de dados distribuídos, possuindo duas funções principais, a `Map` e a `Reduce`.

Entretanto, há diversos projetos na comunidade Apache que estão adicionando funcionalidade ao Apache Hadoop. A seguir, conheça algumas dessas funcionalidades (APACHE HADOOP, 2020).

- **Ambari:** ferramenta baseada na *web* que permite a administradores supervisionar e monitorar *clusters* Hadoop, HDFS, MapReduce, Hive, Pig, ZooKeeper, HBase, HCatalog, Sqoop e Oozie.
- **Hive:** infraestrutura de *data warehouse* que facilita a leitura e a gravação de grandes volumes de dados com uma sintaxe muito parecida com a da Standard Query Language (SQL).
- **Flume:** sistema que coleta ocorrências no Hadoop por meio de *logs* de monitoramento.
- **HBase:** banco de dados altamente escalável e distribuído com fácil integração ao Hadoop.

- **Pig:** linguagem de consulta de alto nível (Pig Latin) orientada a fluxo de dados. Permite a utilização de uma estrutura de execução para computação paralela.
- **ZooKeeper:** serviço de coordenação de alto desempenho para aplicações distribuídas.
- **HCatalog:** tabela e camada que permitem o gerenciamento do armazenamento do Hadoop para diferentes ferramentas de processamento de dados.
- **Sqoop:** aplicação que permite carregar informações incrementais em uma única tabela, ou realizar consultas SQL que podem ser executadas várias vezes.
- **Oozie:** sistema que permite agendamento de fluxos de trabalhos das tarefas do Hadoop.

Utilizar o MapReduce sempre foi uma tarefa difícil. Para facilitar a utilização do Hadoop, foi criado o Apache Hive, uma infraestrutura baseada no Apache Hadoop utilizada para armazenamento e processamento de dados em código aberto. Sua principal funcionalidade é analisar e consultar grandes volumes de dados estruturados e semiestruturados armazenados em arquivos no Hadoop (GOLDMAN *et al.*, 2018).

O Apache Hive possibilita realizar consultas com uma sintaxe muito parecida com a da SQL e realizar trabalhos do MapReduce com facilidade. Assim, torna-se uma ferramenta mais eficiente e confortável para a utilização dos usuários. O Hive tem a sua própria linguagem de programação, a HiveQL (HQL) (CONFLUENCE, 2020). Essa linguagem tem uma sintaxe muito parecida com a da SQL, que permite consultas em trabalhos de MapReduce, ou seja, você não precisa aprender Java para trabalhar com o Hive. As consultas realizadas pelo Hive convertem as consultas SQL em um conjunto de *workers* associados a um *cluster* Hadoop em tarefas MapReduce. O Hive fornece métodos que anexam dados a estruturas de armazenamento do HDFS.

Um exemplo da utilização do Apache Hive se dá no Facebook. A empresa responsável por essa rede social precisa atender a vários requisitos, entre eles executar milhares de tarefas em um *cluster* de computadores que possui um enorme volume de dados de usuários brutos. Esse resultado diariamente gira em torno de 15TB de dados. Hoje, o Apache Hive é utilizado por diversas empresas, como Netflix, IBM, Amazon e Yahoo!.

Como o Hive nasceu?

O Facebook foi uma das primeiras empresas a utilizar o Hadoop. Isso ocorreu devido à demanda de dados que o Facebook gerava diariamente: bancos de dados comuns não eram capazes de administrar toda a pressão desse volume. Para resolver esse problema, o Facebook passou a utilizar o Hadoop com o MapReduce, porém essa atividade exigia muito conhecimento dos técnicos. Por isso, o Facebook criou o Hive para realizar as tarefas MapReduce. O Apache Hive foi a evolução: uma linguagem com a sintaxe muito parecida com a da SQL que permite realizar tarefas complexas do MapReduce com uma facilidade enorme. Além disso, o Hive possui uma interface que possibilita ao usuário enviar diversas consultas SQL com facilidade (GOLDMAN *et al*., 2018).

Arquitetura do Hive

A arquitetura do Hive possui cinco componentes. Veja a seguir.

- ***Driver*:** esse componente atua como um controlador que recebe as instruções da HQL e as executa criando sessões. Além disso, ele monitora o andamento das execuções e o ciclo de vida de cada execução. Por fim, ele salva os metadados importantes gerados durante uma execução da HQL.
- ***Metastore*:** esse componente é responsável por armazenar os metadados de todas as tabelas. Dessa maneira, ele ajuda o *driver* a rastrear conjuntos de dados distribuídos no *cluster* Hadoop.
- ***Compiler*:** esse componente é o compilador das consultas da HQL, ou seja, ele transforma uma consulta em um plano de execução no Hive.
- ***Optimizer*:** esse componente otimiza as execuções do Hive e agrega várias transformações. Além disso, ele pode dividir as tarefas e proporcionar melhor desempenho ao servidor.
- **Executor:** com todos os processos anteriores concluídos, é hora de fazer a execução da tarefa. Esse módulo também é responsável por realizar o *pipeline* das tarefas.

Na Figura 1, veja os componentes e suas relações no Apache Hive. Veja também as três interfaces que descrevem o processo.

- **Thrift:** esse componente permite a interação do Hive com qualquer linguagem de programação que tenha suporte a esse protocolo. Atualmente, existem clientes de terceiro para Python e Ruby.
- **Java DataBase Connectivity (JDBC):** esse componente fornece ao Hive um *driver* de Java puro para conexão com o servidor do Hive.
- **Open DataBase Connectivity (ODBC):** esse componente faz com que os aplicativos suportem o protocolo ODBC e possam se conectar ao Apache Hive. Entretanto, o Hive por padrão não vem com um *driver* ODBC, então é necessário adquirir esse *driver* de um fornecedor.

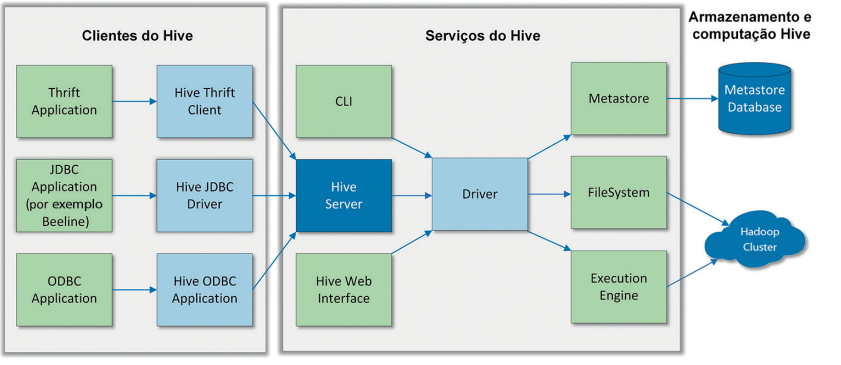

Figura 1. Arquitetura do Apache Hive.
Fonte: Adaptada de White (2015).

Um serviço que roda padrão dentro da Java Virtual Machine (JVM) do serviço do Apache Hive é o *metastore*, responsável por ser o repositório central do Hive. Esse serviço contém uma instância de banco de dados do tipo Derby integrada ao disco local. Essa instância é conhecida como "*metastore* incorporado". Veja na Figura 2.

O *metastore* incorporado é uma maneira simples de utilizar o Hive, entretanto só permite uma conexão por vez ao banco Derby. Caso o usuário tente iniciar mais sessões, será produzido um erro. A solução para suportar diversas conexões é utilizar um banco de dados independente, como mostra a Figura 2: opção *metastore* local ou *metastore* remoto, que se conecta ao banco de dados, porém agora em um processo separado do JVM do Hive. Lembre-se de que conexão funcionará em qualquer banco de dados compatível com JDBC.

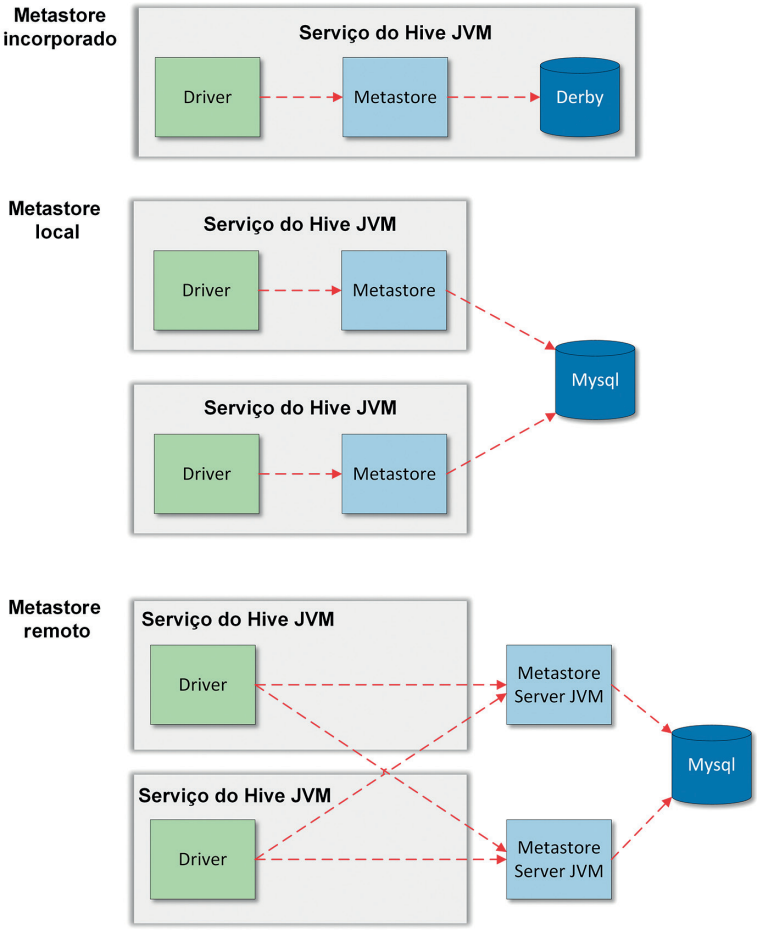

Figura 2. Configurações do *metastore*.
Fonte: Adaptada de White (2015).

Análise de dados no Hadoop por meio do Hive

Agora você vai conhecer os comandos e *scripts* para análise de dados no Hadoop por meio do Apache Hive. Com o Apache Hive, é possível criar tabelas e realizar consultas muito rápido, com uma sintaxe muito parecida com a da SQL. Para entender a utilização do Apache Hive, é importante que você esteja imerso em um ambiente realista. Por isso, suponha que você foi contratado por uma grande empresa e precisa realizar diversas atividades atribuídas pelo seu supervisor. Veja quais são essas atividades:

- copiar arquivos locais no HDFS;
- criar uma tabela específica;
- adicionar os dados do HDFS na tabela;
- realizar consultas simples e com mais de uma tabela;
- apagar a tabela e os dados enviados para o HDFS.

Cópia de arquivos locais no HDFS

A primeira etapa das atividades é armazenar os dados no HDFS. O supervisor informa que os dados que serão copiados estão no diretório `/home/usuario/dados.txt` e devem ser copiados para o caminho do HDFS `/usuario/dados/funcionário _ novo/`. A seguir, veja como seria realizada a cópia dos dados.

Exemplo

```
hadoop fs -copyFromLocal /home/usuario/dados.txt /
usuario/dados/funcionário _ novo/
```

Em seguida, é necessário confirmar que os dados foram copiados com sucesso. Para listar e verificar se os dados estão corretos no HDFS, utilize este comando: `hadoop fs -ls /usuario/dados/funcionário _ novo/`.

Criação de tabela específica

Agora que todos os dados estão armazenados no HDFS, você precisa criar uma tabela no banco de dados com o nome `cliente`. Dentro dessa tabela, devem constar os seguintes campos: `id`, `nome`, `idade`, `endereco` e `data _ cadastro`.

O exemplo a seguir mostra como deve ser feita a tabela. Primeiramente, utilize o comando `CREATE TABLE`, para criar uma tabela. Em seguida, insira o nome da tabela `cliente` e, dentro dos parênteses, os campos com seus formatos. Na próxima linha, utilize o comando `PARTITIONED BY`, que permite a partição da tabela e a criação do campo `data _ cadastro`. Por fim, sinalize o armazenamento em formato binário da tabela com o comando `STORED AS SEQUENCEFILE`.

Exemplo

```
CREATE TABLE cliente (id INT, nome STRING, idade
INT, endereco STRING)
PARTITIONED BY (data _ cadastro STRING)
STORED AS SEQUENCEFILE;
```

Assim que a tabela estiver criada, você pode utilizar o comando `SHOW TABLES;`, que mostra todas as tabelas já criadas no sistema. Você também pode utilizar um prefixo da tabela, como `SHOW TABLES cliente.*;`. Para ter certeza de que todas as colunas da tabela foram criadas corretamente, você pode utilizar o comando `DESCRIBE cliente;`, que retorna as colunas e os seus tipos.

Agora suponha que, após a criação da tabela, seu supervisor peça para que você altere o nome da tabela de `cliente` para `cliente _ novo` e logo em seguida adicione uma coluna chamada `sexo`. Para alterar o nome da tabela, utilize o comando `ALTER TABLE cliente RENAME TO cliente _ novo;`. Em seguida, para adicionar uma nova coluna, utilize o comando `ALTER TABLE cliente _ novo ADD COLUMS (sexo STRING);`, que vai adicionar a coluna `sexo`.

Adição de dados do HDFS em tabela

Depois de criar a tabela, você pode inserir os dados para começar a trabalhar. Seu supervisor lhe informa que os dados a serem utilizados nessa tabela são aqueles inseridos no HDFS anteriormente, que devem estar no caminho `/usuario/dados/funcionário _ novo/dados.txt`. Sua tarefa é importar esses dados para a tabela que você acabou de criar. Além disso, você deve adicionar uma partição com a data 15 de setembro de 2020.

O Apache Hive permite inserir esses dados na tabela de uma forma muito simples: basta utilizar o comando mostrado no exemplo a seguir.

Exemplo

```
LOAD DATA INPATH /usuario/dados/funcionário _ novo/dados.txt INTO TABLE cliente _ novo
PARTITION(date='2020-09-15')
```

Caso fosse necessário adicionar dados do arquivo local e não do HDFS, você poderia utilizar esta sintaxe: `LOAD DATA LOCAL INPATH /home/cliente/dados.txt INTO TABLE cliente _ novo PARTITION(date='2020-09-15')`.

Realização de consultas simples e com mais de uma tabela

Depois de inserir os dados na tabela, você deve realizar uma consulta para verificar se todos eles foram inseridos com sucesso. Para isso, utilize o comando `INSERT OVERWRITE TABLE cliente_temporario`, que é uma especificação do Hive, em que toda consulta obrigatoriamente cria uma tabela temporária que armazena a consulta realizada.

Em seguida, utilize o comando `SELECT cliente.*`, que vai selecionar todos os dados da tabela `cliente`. Depois, insira o comando `FROM cliente`, que permite selecionar a tabela da qual você vai coletar os dados. Por fim, utilize o comando `WHERE cliente.data_cadastro = '2020-09-15'`, que vai trazer os dados da data especificada. A seguir, confira o exemplo completo.

Exemplo

```
INSERT OVERWRITE TABLE cliente_temporario
SELECT cliente.*
FROM cliente
WHERE cliente.data_cadastro = 1;
```

Realizar uma consulta em uma tabela com o Hive é uma tarefa muito fácil. Mas como seria realizar uma consulta unindo duas tabelas? Suponha que seu supervisor peça para que você adicione a cidade do cliente buscando na tabela `usuario` e unindo as duas tabelas por meio da coluna `id`.

Essa atividade é um pouco mais complicada, mas nada que possa lhe assustar. Primeiramente, é necessário criar uma tabela temporária igual à do exemplo anterior, com o comando `INSERT OVERWRITE TABLE cliente_temporario_cidade`. Já o comando `SELECT` será um pouco diferente; devido à união das tabelas, esse comando ficará assim: `SELECT c.*, u.cidade`. E o comando `FROM` fará a união das tabelas da seguinte maneira: `FROM usuario u JOIN cliente c ON (c.id = u.id)`. Por fim, será inserido o comando `WHERE c.data_cadastro = '2020-09-15';`. Veja o exemplo a seguir.

Exemplo

```
INSERT OVERWRITE TABLE cliente_temporario_cidade
SELECT SELECT c.*, u.cidade
FROM usuario u JOIN cliente c ON (c.id = u.id)
WHERE c.data_cadastro = '2020-09-15';
```

Apagamento de tabela e dados enviados para o HDFS

Para finalizar suas atividades, seu supervisor pede para você apagar a tabela e os dados que foram criados no HDFS durante os testes. Para apagar a tabela e todos os dados contidos nela, utilize o comando `DROP TABLE cliente;`. Para apagar os arquivos do HDFS, use este comando: `hadoop fs -rm -r /usuario/dados/funcionário _ novo/dados.txt`.

Requisitos para análise de dados por meio do Hive

Como toda aplicação, o Hadoop possui requisitos para sua instalação. A plataforma exige como pré-requisitos mínimos: Core I5, 8GB RAM, 200GB de armazenamento de disco e sistema operacional Linux. Além disso, é necessário instalar o *software* JDK 8 e o MYSQL (ou outro SGBDR). Vale ainda lembrar que é muito simples adicionar outras máquinas como *cluster* no Hadoop.

O Apache Hadoop pode ser executado de diversas maneiras. A seguir, veja algumas delas.

- **Local:** utiliza uma única máquina, entretanto a *performance* da aplicação é péssima. Se você deseja obter uma boa *performance*, utilize algumas máquinas como *cluster*.
- **MapReduce:** é possível utilizar o MapReduce para transformar consultas em programas de MapReduce. Entretanto, essa opção tem uma péssima *performance*, devido à dificuldade de utilização e criação de tarefas MapReduce.
- **Hive on Spark em *clusters* Yarn:** permite uma boa *performance*. Essa funcionalidade utiliza o mecanismo de execução do Spark, entretanto aplica os recursos computacionais gerenciados pelo Yarn. Assim, logo que uma consulta é enviada ao Spark, ela automaticamente é carregada pelo Yarn.
- **Hive on Spark em *clusters* Spark:** muito parecida com a anterior, possui uma boa *performance*. Essa funcionalidade utiliza o próprio mecanismo do Spark para gerenciar as atividades carregadas.

Fique atento

Essas configurações são definidas em arquivos de configuração contidos no Hadoop. Elas também podem ser configuradas diretamente no console do Hive.

Quando se fala de *big data*, toda otimização é necessária, devido aos imensos volumes de dados que podem trafegar em um sistema. Assim, é necessário conhecer métodos que agilizem esse processo. O Hive é uma aplicação que permite utilizar diversos padrões e formatos de arquivos, e é necessário conhecer o melhor formato para utilização. Veja a seguir.

- **Texto:** formato simples de texto, como `CSV`, `TSV` e `TXT`.
- **RCFile:** formato antigo binário, que era baseado em chave e valor.
- **Sequence File:** formato de arquivos binários do Hadoop.
- **Apache Parquet:** ótimo formato para consultas em larga escala, sendo normalmente o formato preferido pelos usuários.
- **Apache Optimized Row Columnar (ORC):** formato que possui alto grau de compactação e *performance*. Também possui suporte para índices, Acid e conteúdos complexos.
- **Apache Avro:** formato bastante popular. Usado para serialização e troca de dados.

Muitas ferramentas que vêm se popularizando para ajudar na implantação de soluções de *big data* podem ser utilizadas no Hadoop, sendo chamadas de "SQL-on-Hadoop". A seguir, veja quais são as principais ferramentas grátis disponíveis.

- **Apache Drill:** *software* de código aberto utilizado para análise interativa de conjuntos de dados de larga escala.
- **Apache Impala:** *software* de consultas SQL com processamento paralelo em *clusters* Hadoop.
- **Presto:** *software* de consultas SQL distribuído com alto desempenho em grandes volumes de dados. Permite que usuários consultem várias fontes de dados, como Hadoop, MySQL, Cassandra, Kafka, MongoDB, AWS S3 e Alluxio.
- **Spark SQL:** *software* que permite consultar estruturas de dados dentro de execuções do Spark, utilizando SQL ou mesmo uma Application Programming Interface (API) de DataFrame.
- **Apache Phoenix:** *software* de código aberto para consultas em bancos de dados relacionais que suporta OLTP no Hadoop usando o Apache HBase.

Referências

APACHE HADOOP. *Hadoop MapReduce Client*. [Wakefield]: Apache Software Foundation, 2020. Disponível em: https://hadoop.apache.org/docs/current/hadoop-mapreduce-client. Acesso em: 31 ago. 2020.

CONFLUENCE. *Apache Hive*: home. [Wakefield]: Apache Software Foundation, 2020. Disponível em: https://cwiki.apache.org/confluence/display/Hive/Home. Acesso em: 31 ago. 2020.

DEAN, J.; GHEMAWAT, S. MapReduce: simplified data processing on large clusters. *In*: SYMPOSIUM ON OPERATING SYSTEM DESIGN AND IMPLEMENTATION, 6., 2004, San Francisco. *Proceedings* [...]. San Francisco: OSDI'04, 2004. p. 137-150. Disponível em: https://research.google/pubs/pub62. Acesso em: 15 out. 2020.

GOLDMAN, A. *et al*. *Apache Hadoop*: conceitos teóricos e práticos, evolução e novas possibilidades. Florianópolis: UFSC, 2018. Disponível em: http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5645/JAI_2012_Cap%203_Apache%20Hadoop.pdf. Acesso em: 15 out. 2020.

WHITE, T. *Hadoop*: the definitive guide. 4th ed. Boston: O'Reilly Media, 2015.

Leituras recomendadas

DU, D. *Apache Hive essentials*. Birmingham: Packt, 2015.

HOLMES, A. *Hadoop in practice*: includes 104 techniques. 2nd ed. Shelter Island: Manning, 2014.

SINGH, C.; KUMAR, M. *Mastering Hadoop 3*: big data processing at scale to unlock unique business insights. Birmingham: Packt, 2019.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS