FUNDAMENTOS DE BIG DATA

Laerte de Marque

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Hadoop Distributed Filesystem na prática

OBJETIVOS DE APRENDIZAGEM

- Descrever os principais comandos do Hadoop Distributed FileSystem (HDFS) e sua interação com os componentes do HDFS.
- Desenvolver os comandos do HDFS de forma prática, bem como suas finalidades.
- Planejar o uso prático do HDFS incluindo a estrutura dos arquivos em blocos.

Introdução

O HDFS é um sistema de arquivos distribuídos e tem seus principais comandos baseados em um formato Portable Operating System Interface (Posix); esses comandos são compatíveis com o sistema Unix. Quem já usa o Linux com certeza tem facilidade para utilizar os comandos do HDFS.

Neste capítulo, você vai estudar os comandos mais usados no HDFS para operações básicas, como copiar, apagar, fazer verificações de espaço e movimentar arquivos para dentro e fora do sistema de arquivos distribuídos. Você também vai ver como esses comandos interagem com os recursos do sistema HDFS. Além disso, vai acompanhar alguns exemplos práticos que mostram o funcionamento dos comandos e a sua interação com a arquitetura HDFS.

Principais comandos do HDFS e sua interação com os componentes do sistema

A partir de agora, você vai conhecer os principais comandos do HDFS. Os comandos do sistema de arquivos HDFS são similares a muitos comandos do

Linux, especialmente os mais básicos. O *shell* do Hadoop, chamado "FileSystem" ou "Fsshell", oferece diversos comandos para a realização de tarefas relativas aos arquivos presentes nos DataNodes.

A estrutura básica de funcionamento do sistema é a seguinte: o comando interage com o NameNode e, após as devidas permissões, acessa a *Application Programming Interface* (API), capaz de realmente se comunicar com os DataNodes. Em qualquer momento, se você tiver dúvidas em relação a algum comando, pode usar o próprio recurso de ajuda do sistema. Para descobrir todos os comandos do sistema operacional, use o comando `hadoop dfs`.

A seguir, veja quais são os comandos básicos mais usados no HDFS.

- `mkdir`: cria um diretório. Por exemplo, `mkdir teste` cria o diretório `teste` (mesma sintaxe do Linux).
- `du`: demonstra o uso dos discos.
- `ls`: lista arquivos, permissões, tamanhos, etc. (mesma função do Linux).
- `chown`: modifica o dono de um diretório.
- `cp`: faz uma cópia dentro do HDFS. Seus parâmetros são `cp` origem e destino.
- `rm`: remove o diretório vazio. Se estiver cheio, use `rmr` (tenha cuidado com comandos de remoção).
- `copyFromLocal`: copia um arquivo do sistema de arquivos locais para o DFS destino.
- `copyToLocal`: faz o inverso do `copyFromLocal`, isto é, copia do DFS para um sistema de arquivos local.
- `cat`: exibe o conteúdo do arquivo (`.txt`).
- `tail`: mostra o último 1 kB no DFS; no Linux, mostra a última linha (APACHE HADOOP, 2014).

Para executar esses comandos, você pode digitar, por exemplo, `hadoop fs –ls` (se o comando for o `ls`). Nas versões mais atuais, o primeiro comando é o usado (`hdfsd fs –ls`).

Outro ponto importante é relativo à criação de contas no Hadoop. As contas são parte crucial de qualquer sistema operacional. No HDFS, você pode usar os seguintes comandos:

- `hadoop fs -mkdir /user/username`
- `hadoop fs -chown username:username /user/username`

Esses comandos criam o diretório do usuário. Note que o diretório padrão é `user` para os usuários (`/home` do Linux ou `C:\Users` do Windows). O primeiro comando cria um diretório, e o segundo faz com que o proprietário do diretório seja o próprio dono. Ou seja, se o diretório for Carlos, o dono do diretório será Carlos, e não o criador que executa os comandos.

Assim como em outros sistemas operacionais, o diretório do usuário pode ser perfilado com uma cota. A cota ideal é 1 TB, como mostra o exemplo a seguir:

```
hadoop fs admin -setSpaceQuota 1t /user/username
```

Agora, você vai ver como funcionam os comandos mais usados. Primeiro, considere o comando `mkdir`. Ele é um comando usado para a criação de pastas. Assim como em outros sistemas operacionais, a estrutura de pastas do HDFS armazena os dados. Veja estes exemplos:

- `hadoop fs -mkdir /empresax`
- `hadoop fs -mkdir /empresax/poa`
- `hadoop fs -mkdir /empresax/sp`
- `hadoop fs -mkdir /empresax/poa/finaceiro`
- `hadoop fs -mkdir /empresax/poa/juridico`
- `hadoop fs -mkdir /empresax/sp/desenvolvedores`
- `hadoop fs -mkdir /empresax/sp/vendedores`

É possível simplificar os comandos usando espaços, desde que os diretórios anteriores não precisem dos diretórios posteriores. Por exemplo: `hadoop fs -mkdir /empresax/poa/finaceiro /empresax/poa/jurídico`. A partir desse comando, será criada uma pasta principal chamada "`empresax`". Dentro dessa pasta, serão criadas pastas das filiais dessa empresa (em São Paulo, será `sp`; em Porto Alegre, será `poa`). Além disso, dentro de cada filial, serão criadas pastas de dados para seus respectivos trabalhadores. Em Porto Alegre, haverá o setor financeiro e o jurídico, enquanto na filial de São Paulo haverá os desenvolvedores e os vendedores.

Fique atento

A separação da empresa em setores facilita a catalogação dos dados no sistema. Essa separação deve ser feita depois de uma análise dos dados que serão coletados, e o modelo demonstrado aqui é simplificado.

Outro comando importante é o `hadoop fs –du –h /hbase`. Ele verifica as bases de dados em todos os DataNodes. Em geral, a saída é um pouco assustadora, pois é possível ver dois tamanhos. Você vai ver algo semelhante a isto em cada linha:

```
4.6 K          13.8 K          /hbase/data
```

Não se assuste: enquanto nos sistemas Linux você terá o retorno de apenas um valor (por exemplo, 4.6 K), no HDFS você terá dois parâmetros de tamanho. Nesse caso, o tamanho real é 4.6 K e o tamanho atual em gravação é 13.8 K. Qual é o motivo dessa diferença? Ela é devida ao número de replicações. Pensando um pouco, você vai perceber que o sistema está usando três replicações (4.6 × 3 = 13.8 K), que consistem no padrão utilizado por ele.

O comando `hadoop fs –ls /nome` é semelhante ao utilizado no Linux. Ele lista o conteúdo de determinado diretório no HDFS. Já o comando `hadoop fs –chown` serve para mudar o dono e o grupo do arquivo ou diretório. Veja um exemplo: o comando `hadoop fs –chown Paulo:RH /data/arquivo.txt` modifica o `arquivo.txt`: o seu dono agora é Paulo, que pertence ao grupo RH.

A seguir, veja outros comandos e alguns exemplos.

- `hadoop fs -cp`
  - Exemplo: `hadoop fs -cp /user/marcos/arquivo.txt /user/carlos`.
  - Serve para copiar o `arquivo.txt` que está na pasta `/user/marcos` para a pasta `/user/Carlos`.
- `hadoop fs -rm`
  - Exemplo: `hadoop fs -rm-r/user/marcos/dados.txt`.
  - Apaga o arquivo `dados.txt` que está na pasta `/user/marcos`.
- `hadoop fs -copyFromLocal`
  - Exemplo: `hadoop fs -copyFromLocal arquivolocal.txt /user/hadoop`.
  - Copia o `arquivolocal.txt` do sistema de arquivos locais para o HDFS em `/user/hadoop`.
- `hadoop fs -copyToLocal`
  - Exemplo: `hadoop fs –copyToLocal /user/hadoop/arqlocal.txt /teste`.
  - Copia o arquivo do HDFS chamado `arqlocal.txt` que está dentro de `/user/hadoop` para a pasta de arquivos do sistema local chamada "`/teste`".

- `hadoop fs -cat`
  - Exemplo: `hadoop fs -cat /user1/texto.txt`.
  - Mostra o conteúdo do arquivo `texto.txt` que está dentro do diretório `user1`.
- `hadoop fs -tail`
  - Exemplo: `hadoop fs -tail /user1/texto.txt`.
  - Mostra o último 1 kB do arquivo `texto.txt` que está dentro de `user1`.

Desenvolvimento dos comandos do HDFS

Nesta seção, você vai ver alguns dos comandos do HDFS e suas interações com a arquitetura do sistema. Todos os comandos interagem com o NameNode e com os DataNodes. Para acionar qualquer comando do sistema, é necessário ser administrador; caso contrário, os comandos não funcionarão. Em um *cluster* Hadoop usando o HDFS, existe um NameNode, e é ele que determina o que poderá ser lido e escrito nos DataNodes, que são realmente os locais de armazenamento dos dados, representados por seus blocos (frações). Ainda existe no *cluster* Hadoop o Secondary NameNode, mas ele é responsável por tarefas mais simples, como os *logs* de gerenciamento, e não pode substituir o NameNode em si, sendo apenas um *backup* dos metadados.

A seguir, você vai ver exemplos desses três comandos e toda a sua interação com a arquitetura do HDFS. Comece imaginando que o *cluster* Hadoop está dividido em 12 máquinas, em dois *racks* diferentes. Nesse *cluster*, há um servidor NameNode, um servidor Secondary NameNode e 10 máquinas, que serão os DataNodes. Ainda em relação ao sistema local, suponha que seja usado um Linux com apenas um servidor.

Um comando simples é um comando que serve para copiar arquivos. A ideia aqui é copiar um arquivo chamado `relatorio.mdb` do sistema local para o sistema HDFS. Ou seja, esse arquivo está em uma pasta local do sistema de arquivos local e será colocado no HDFS, que é o sistema de arquivos distribuídos. O comando usado para isso é o `copyFromLocal`. Esse comando é capaz de copiar do sistema de arquivos local para o *cluster* HDFS. Nesse exemplo, o comando poderia ser:

```
hadoop fs -copyFromLocal /Users/Administrator/arquivos/
relatorio.mdb /hbase/documentos
```

Agora você vai ver o passo a passo do que acontece no sistema após o comando ser digitado. Observe a Figura 1, a seguir.

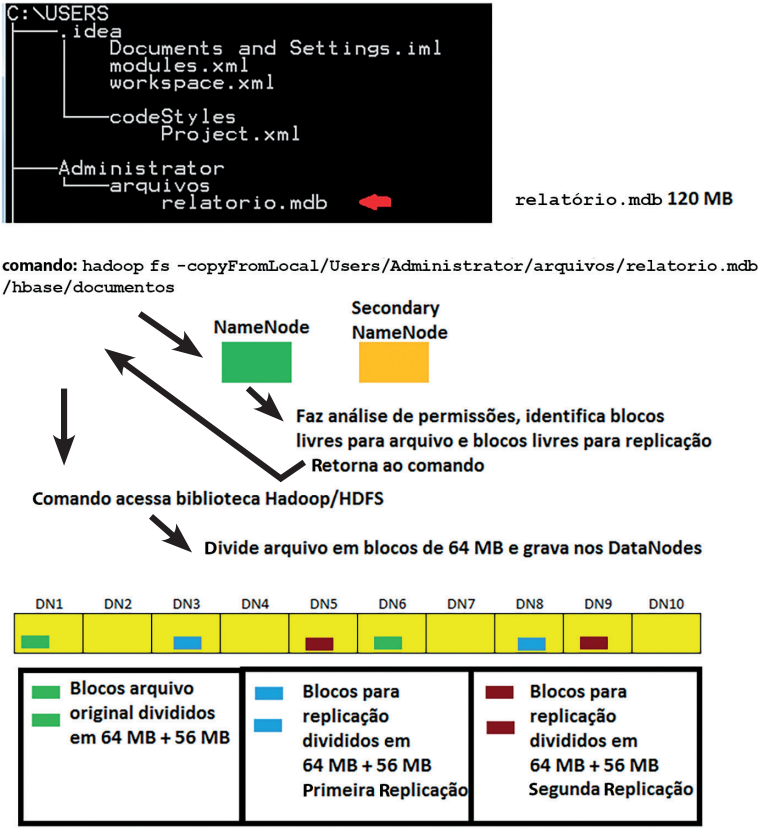

Figura 1. Etapas de um comando na arquitetura HDFS: entrada de dados no armazenamento HDFS.

Na Figura 1, note que o NameNode autorizou a gravação dos blocos no HDFS e informou ao comando onde devem ser feitas as gravações e onde devem ser gravadas as réplicas. Observe que a estrutura de gravação utiliza blocos.

O NameNode registra onde os blocos estão gravados, e não é necessário que a gravação seja sequencial (por exemplo, pedaço 1 gravado no DataNode1, pedaço 2 gravado no DataNode2, réplica 1 gravada no DataNode3 e assim por diante). Isso não é necessário mesmo que haja espaço disponível nessa sequência. Como mostra a Figura 1, o arquivo original de tamanho 120 MB foi dividido em dois blocos, um de 64 MB e outro de 56 MB (em cor verde). Depois, ele foi replicado duas vezes.

Agora acompanhe um segundo exemplo, baseado nos mesmos arquivos e diretórios. Imagine que o arquivo `relatorio.mdb` só exista no sistema HDFS

(arquivo em blocos e distribuído) e que você queira copiá-lo para o sistema de arquivos local. Esse arquivo agora está em `/hbase/documentos` e você quer enviá-lo para a pasta `/Users/Administrator/arquivos`. Nesse caso, o comando adequado seria:

```
hadoop fs -copyToLocal /hbase/documentos/relatorio.mdb /
            Users/Administrator/arquivos
```

Após o comando, a operação interna se desenrola de modo um pouco diferente se comparada àquela mostrada na Figura 1. Veja na Figura 2, a seguir.

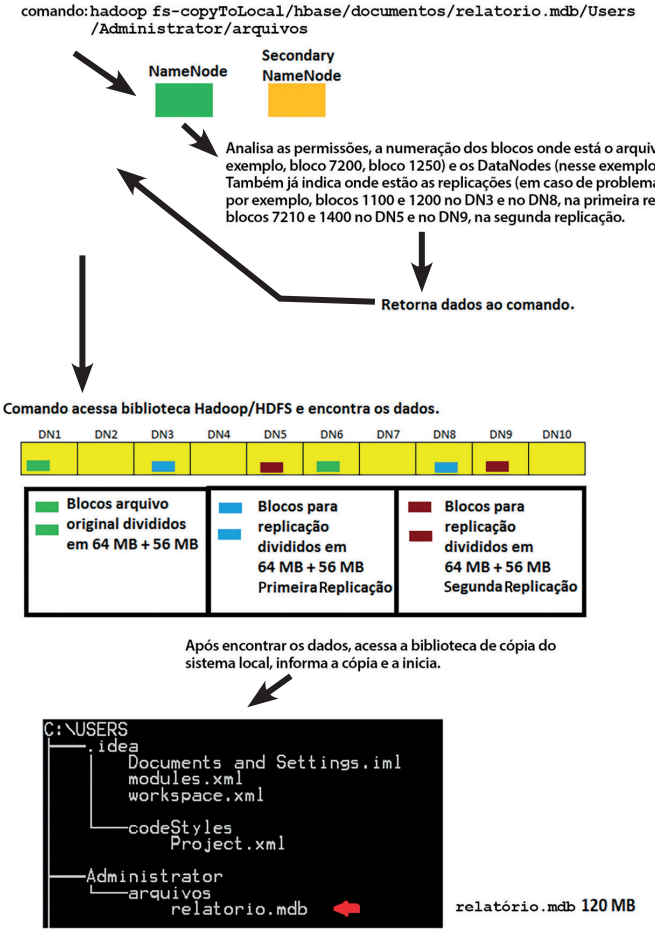

Figura 2. Etapas de um comando de cópia de dados do HDFS para o sistema de arquivos locais.

Note que agora, apesar de a operação ser inversa, a arquitetura trabalha de modo muito diferente. O NameNode é consultado e verifica as permissões, como no caso anterior, porém agora ele apenas lê em sua tabela quais são os blocos em que estão o arquivo e as suas replicações, além dos respectivos DataNodes. Após essa etapa, e munido dessa informação, o NameNode a repassa ao comando que acessa a biblioteca HDFS, que começa a cópia do arquivo. Após o acesso aos dados, o comando repassa ao sistema operacional local uma operação de gravação desses dados.

Por fim, um terceiro exemplo, que diz respeito a um comando que parece mais simples, mas que demanda uma operação interna bastante complicada. Esse comando é o `rm -r`, que serve para apagar e substitui o obsoleto `rmr`. Ele exclui diretórios e qualquer arquivo constante neles; se usado com a opção `-r`, haverá uma exclusão recursiva.

A exclusão recursiva limpa tudo o que estiver dentro do diretório. Por *default*, o arquivo apagado é movido para a lixeira dentro da classe `FileSystem`, com o método `getTrashRoot(Path path)`. É muito importante você se lembrar de que nas últimas versões a lixeira do HDFS vem desabilitada por padrão; o interessante seria ativá-la no arquivo `core-site.xml`, apenas definindo um valor no parâmetro `fs.trash.interval`.

Fique atento

Mesmo com a lixeira ativada, se realmente for necessário apagar o arquivo, você pode usar o comando `expunge`.

Agora, você vai ver o que acontece no sistema de arquivos HDFS quando se usa o `rm`. Suponha que você digitou:

```
hadoop fs -rm -r /hbase/documentos/relatorio.mdb
```

Nesse caso, você vai apagar o arquivo `relatorio.mdb` (suponha que a configuração da lixeira esteja desativada). As permissões para apagar o arquivo também seguem o padrão Posix. Veja o que afirma White (2015, p. 52, tradução nossa):

> O HDFS possui um modelo de permissões para arquivos e diretórios muito parecido com o modelo Posix. Existem três tipos de permissão: a permissão de leitura (`r`), a permissão de gravação (`w`) e a permissão de execução (`x`).

Apenas uma ressalva sobre a permissão de execução: arquivos não podem ser executados no HDFS, mas, para um diretório filho, a permissão do pai é necessária para o acesso. Em resumo: apagar o arquivo `relatorio.mdb` só será possível se você tiver permissões efetivas para isso, e tais permissões serão conferidas pelo NameNode.

Veja o funcionamento de uma deleção com permissões apropriadas e sem lixeira ativa na Figura 3, a seguir, que ilustra a situação posterior ao uso deste comando:

```
hadoop fs -rm -r /hbase/documentos/relatorio.mdb
```

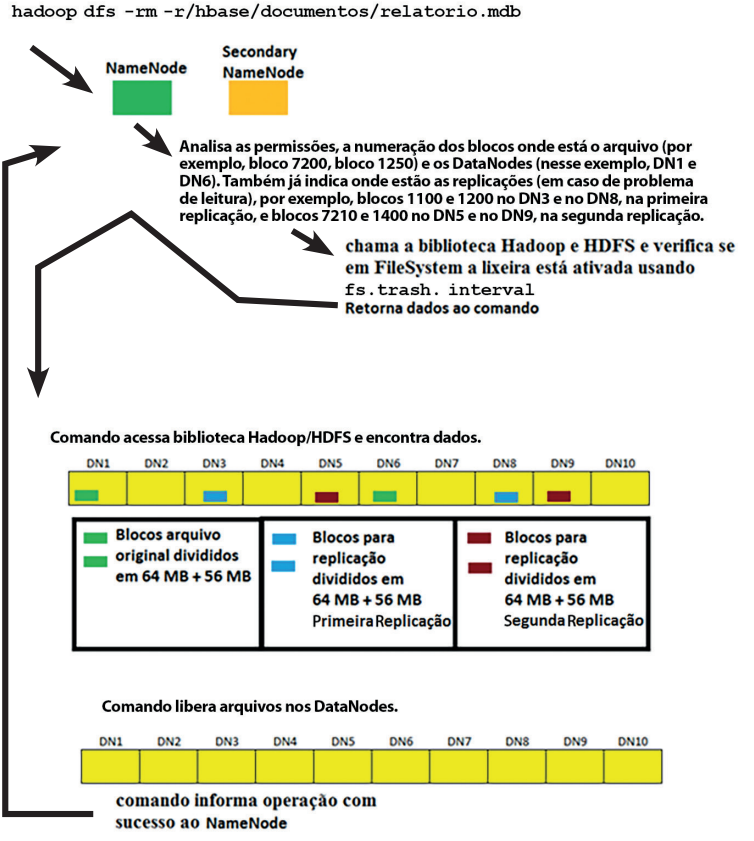

Figura 3. Etapas de um comando na arquitetura HDFS para eliminação de um diretório/arquivo.

Esses exemplos mostram o funcionamento do processo interno realizado pelos comandos. Os comandos passam primeiro pelo controle do NameNode e, após receberem a permissão adequada, conseguem acessar os DataNodes. A partir daí, fazem somente o que o NameNode liberou, isto é, agem de acordo com suas permissões.

Uso prático do HDFS

Para o uso prático do HDFS, é necessário conhecer alguns conceitos e executar algumas configurações básicas. O armazenamento do HDFS sempre é feito em blocos. Todos os blocos sempre têm o mesmo tamanho e são integralmente aproveitados, diferentemente de blocos do NTFS ou do Ext4, sistemas de arquivos comuns que se baseiam em blocos mínimos. No NTFS, por exemplo, cada bloco tem 8 kB (8.192 bytes) e, mesmo gravando apenas um caractere, um bloco ocupa 8 kB, o que gera uma perda de espaço conhecida como "*slack space*". No HDFS, não acontece isso: todos os 64 kB do bloco são preenchidos com informações úteis, não gerando *slack space*. Veja:

> Existem arquivos que, devido ao seu grande tamanho, não podem ser armazenados em um único disco rígido. Esta situação fica mais evidente quando nos referimos aos arquivos de aplicações "Big Data". Uma alternativa nesse caso é criar uma forma de dividir esses grandes arquivos e distribuí-los em um aglomerado de máquinas (GOLDMAN *et al.*, 2012, documento *on-line*).

Na Figura 4, veja a diferença entre os blocos em sistemas comuns e aqueles do HDFS.

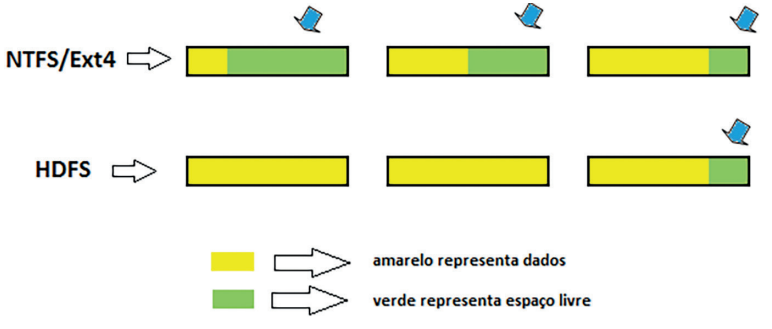

Figura 4. Comparativo dos blocos NTFS/Ext4 e HDFS.

Observe que o HDFS mantém blocos inteiros de dados independentemente do tamanho do arquivo, e somente o último bloco pode ter dados menores. A vantagem do HDFS é que o armazenamento é muito mais eficiente. Além disso, quanto mais blocos houver, mais eficiente será o armazenamento, pois a perda com espaço pode ocorrer apenas no último bloco.

O HDFS também faz a réplica dos blocos para a tolerância a falhas. No caso de leitura, para aumentar a velocidade de processamento e retornar em menor tempo, o HDFS retorna a réplica que está mais perto de quem solicita o dado. Ou seja, se você quer ler o arquivo X e ele está no *rack* de servidores 1 e no *rack* de servidores 3, o HDFS vai retornar a cópia que está mais perto de você (considerando não a distância, mas o tempo de processamento).

Por último, os blocos de dados podem partir de 64 MB, um número muito alto quando comparado com aqueles dos `clusters` de sistemas de arquivos não distribuídos, como NTFS e Ext4, que têm no máximo 64 kB. O bloco HDFS inicia já 1.024 vezes maior do que os blocos de sistemas não distribuídos.

Agora, você vai conhecer melhor a arquitetura a fim de planejar o uso prático do HDFS com base em seus blocos e replicações. A Figura 5, a seguir, mostra o funcionamento da arquitetura.

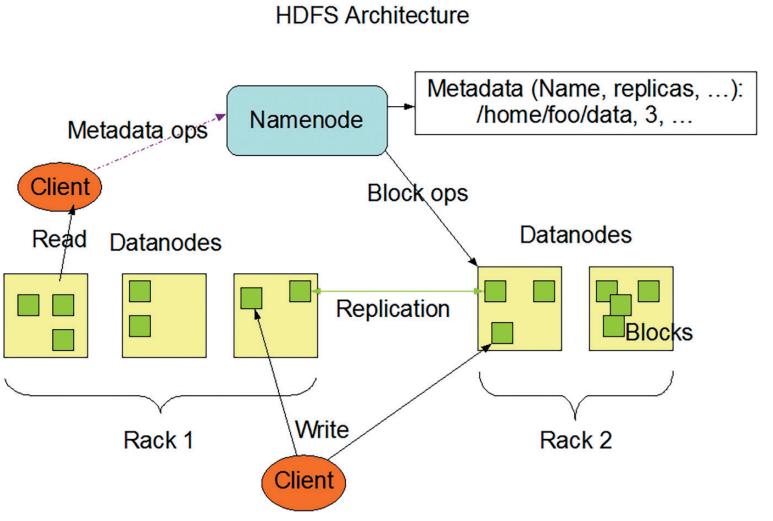

Figura 5. NameNode e DataNodes.
Fonte: Borthakur (2019, documento *on-line*).

Na Figura 5, observe que o cliente acessa o `NameNode` solicitando a permissão para o que quer fazer. O `NameNode`, por meio de suas informações (`metadados` — locais, réplicas, permissões, espaços disponíveis, etc.), retorna ao cliente o que deve ser feito dentro dos `DataNodes`, que têm a estrutura baseada em blocos.

Os blocos, como você já sabe, são importantes para a replicação e a tolerância a falhas. Então, os blocos e replicações não são opcionais e precisam ser definidos pelo administrador do sistema ou deixados na configuração padrão durante a instalação. Conhecendo melhor o esquema de replicação dos blocos, você pode entender como a arquitetura trabalha com os dados nos DataNodes (Figura 6).

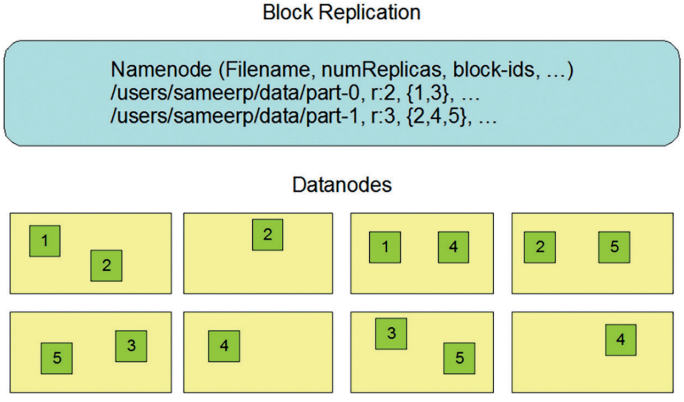

Figura 6. Esquema de replicação do bloco.
Fonte: Borthakur (2019, documento *on-line*).

Em síntese, o HDFS trabalha com blocos e com replicações. Para utilizar esses recursos, você precisa conhecer os arquivos principais de configuração e o modo como as replicações influem no espaço de uso do sistema.

Arquivos básicos de configuração do HDFS

É importante destacar os principais arquivos de configuração do HDFS. Veja a seguir (APACHE HADOOP, 2019).

- **Configuração padrão somente leitura:** `core-default.xml`, `hdfs-default.xml`, `yarn-default.xml` e `mapred-default.xml`.
- **Configuração específica do *site*:** `etc / hadoop / core-site.xml`, `etc / hadoop / hdfs-site.xml`, `etc / hadoop / yarn-site.xml` e `etc / hadoop / mapred-site.xml`.

A seguir, veja as principais configurações práticas dos arquivos mencionados.

- No caso do arquivo `hdfs-site.xml`, é possível definir o tamanho dos blocos. Lembre-se de que um bloco padrão é de 64 MB, e você pode fazer alguns testes para verificar a capacidade de processamento. O parâmetro a ser mudado após um comando de edição (por exemplo, `vi hdfs-site.xml`) é: `dfs.block.size`.
- Com o `dfs.datanode.data.dir`, você pode modificar as replicações que o sistema deve fazer, que por padrão são três. Para isso, basta editar o arquivo usando `vi dfs.datanode.data.dir`, colocando o valor desejado. Quanto mais replicações existirem, mais segurança você terá (porém com menos espaço livre em disco). Assim, um valor alto geralmente é usado se há bastante espaço disponível nos DataNodes (APACHE HADOOP, 2019).

Com essas informações, você já pode planejar o uso do sistema considerando a velocidade, a segurança e mesmo a capacidade de espaço em seus discos. Se você optar por um bloco de dados grande (maior do que 64 MB), o sistema vai ser mais ágil. De qualquer modo, sempre é necessário elaborar alguns testes para entender a velocidade real em sua estrutura e seus equipamentos. Em geral, um bloco maior deixa tudo mais rápido, mas existe um limite que poderá ser percebido em testes.

Lembre-se de que a quantidade de replicações influencia o nível de tolerância a falhas. Contudo, muitas replicações podem ser um problema, pois consomem muito espaço. Por exemplo, ao mudar de um sistema com três replicações para um com cinco replicações, de imediato o sistema começa a ocupar mais espaço: se você tem 10 GB de espaço de disco total e 3 GB ocupados, com cinco replicações, passará a ter 5 GB ocupados. No Quadro 1, veja uma síntese desse exemplo.

Quadro 1. Espaço usado por replicações

Tamanho total	Tamanho usado atualmente	Replicações	Espaço livre total	Espaço livre efetivo
12 GB	3 GB	3 replicações	9 GB	3 GB
12 GB	5 GB	5 replicações	7 GB	1,4 GB

Como esses resultados foram obtidos? Muitos administradores se baseiam apenas no aumento e no espaço livre, mas você deve considerar também o espaço livre efetivo. A lógica é fácil de entender: se o seu espaço livre com três replicações equivale a 9 GB, você pode usar apenas 3 GB efetivamente, pois precisa de três replicações (9 GB ÷ 3 replicações = 3 GB). Já se o seu espaço livre passa a ser 7 GB e você precisa de cinco replicações, então seu espaço livre equivale a apenas 1,4 GB (7 GB ÷ 5 replicações). Por isso, um bom planejamento, especialmente da quantidade de réplicas, é importante para que as replicações não se tornem um problema de armazenamento em seu sistema HDFS.

Fique atento

Ao longo deste capítulo, você conheceu melhor o sistema do HDFS e estudou a interação entre os seus comandos. Como você verificou, independentemente da operação, na arquitetura HDFS, o NameNode é sempre consultado internamente. Após a consulta ao NameNode, ele verifica primeiro as permissões de segurança, ou seja, avalia se quem está pedindo algum acesso de leitura ou gravação pode realmente fazer isso. Se o acesso for concedido, por exemplo, a determinado usuário (João) que quer gravar/ler um arquivo no HDFS, o NameNode somente o autoriza após fazer todas as verificações de segurança. Só depois é que serão determinadas as divisões, suas replicações e os DataNodes em que devem ser gravados/lidos os blocos. Todas essas operações de segurança fazem com que o HDFS seja um sistema muito confiável e seguro, o que o leva a ser adotado por diversas empresas.

Referências

APACHE HADOOP. *Apache Hadoop 2.4.1.* California: Apache Software Foundation, 2014. Disponível em: https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html. Acesso em: 13 set. 2020.

APACHE HADOOP. *Hadoop Cluster Setup.* California: Apache Software Foundation, 2019. Disponível em: https://hadoop.apache.org/docs/r2.10.0/hadoop-project-dist/hadoop-common/ClusterSetup.html. Acesso em: 13 set. 2020.

BORTHAKUR, D. *HDFS architecture guide*. California: Apache Software Foundation, 2019. Disponível em: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Acesso em: 13 set. 2020.

BRAMER, M. *Principles of data mining:* undergraduate topics in computer science. 3rd. ed. Heidelberg: Springer, 2016.

GOLDMAN, A. *et al.* Apache hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. *In:* CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 32., 2012. *Anais* [...]. Curitiba: 2012. Disponível em: //www2.sbc.org.br/csbc2012/anais_csbc/eventos/jai/index.html. Acesso em: 13 set. 2020.

WHITE, T. *Hadoop the definitive guide:* storage and analysis at internet scale. 4th. Ed. California: O´Reilly, 2015.

Leitura recomendada

SAMMER, E. *Hadoop operations:* a guide for developers and administrators. California: O´Reilly, 2012.

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