5 Vs do Big Data

A proposta de uma solução de *Big Data* é oferecer uma abordagem consistente no tratamento do constante crescimento e da complexidade dos dados. Para tanto, o conceito considera os 5 V's do *Big Data*: o **V**olume, a **V**elocidade, a **V**ariedade, a **V**eracidade e o **V**alor.

•**Volume**: O conceito de volume no *Big Data* é melhor evidenciado pelos fatos do quotidiano: diariamente, o volume de troca de e-mails, transações bancárias, interações em redes sociais, registro de chamadas e tráfego de dados em linhas telefônicas. Todos esses servem de ponto de partida para a compreensão do volume de dados presentes no mundo atualmente.

Estima-se que atualmente o volume total de dados que circulam na internet é de 250 *Exabytes* (10^{18} *bytes*) por ano. (Inmoment, 2014) Todos os dias são criados 2,5 quintilhões de *bytes* em forma de dados, atualmente 90% de todos os dados que estão presentes no mundo foram criados nos últimos 2 anos (IBM). É importante também compreender que o conceito de volume é relativo a variável tempo, ou seja, o que é grande hoje, pode não ser nada amanhã. (Ohlhorst, 2012). Nos anos 90, um *Terabyte* (10^{12} *bytes*) era considerado *Big Data*. Em 2015, teremos no mundo aproximadamente um volume de informação digital de 8 *Zettabytes* (10^{21} *bytes*), um valor infinitamente maior (IBM).;

•**Velocidade**: Você cruzaria uma rua vendado se a última informação que tivesse fosse uma fotografia tirada do tráfego circulante de 5 minutos atrás? Provavelmente não, pois a fotografia de 5 minutos atrás é irrelevante, você precisa saber das condições atuais para poder cruzar a rua em segurança. (Forbes, 2012) A mesma lógica se aplica a empresas, pois necessitam de dados em atuais sobre seu negócio, ou seja, velocidade. Segundo Taurion (2014) a importância da velocidade é tamanha que em algum momento deverá existir uma ferramenta capaz de analisar os dados em tempo real. Atualmente,

os dados são analisados somente após serem armazenados, mas o tempo gasto para o armazenamento em si já desclassifica esse tipo de análise como uma análise 100% em tempo real.

Informação é poder (The Guardian, 2010), e assim sendo a velocidade com a qual você obtém essa informação é uma vantagem competitiva das empresas. Velocidade pode limitar a operação de muitos negócios, quando utilizamos o cartão de crédito por exemplo, se não obtivermos uma aprovação da compra em alguns segundos normalmente pensamos em utilizar outro método de pagamento. É a operadora perdendo uma oportunidade de negócios pela falha na velocidade de transmissão e análise dos dados do comprador.;

•**Variedade**: O volume é apenas o começo dos desafios dessa nova tecnologia, se temos um volume enorme de dados, também obtemos a variedade dos mesmos. Já pensou na quantidade de informações dispersas em redes sociais? *Facebook*, *Twitter* entre outros possuem um vasto e distinto campo de informações sendo ofertadas em público a todo segundo. Podemos observar a variedade de dados em e-mails, redes sociais, fotografias, áudios, telefones e cartões de crédito. (McAffe et al, 2012). Seja qual for a discussão, podemos obter infinitos pontos de vista sobre a mesma. Empresas que conseguem captar a variedade, seja de fontes ou de critérios, agregam mais valor ao negócio (Gartner). O *Big Data* escalona a variedade de informações das seguintes formas (Jewell, Dave et al):

- •Dados estruturados: são armazenados em bancos de dados, sequenciados em tabelas;
- •Dados semi-estruturados: acompanham padrões heterogêneos, são mais difíceis de serem identificados pois podem seguir diversos padrões;
- •Dados não estruturados: são uma mistura de dados com fontes diversificadas como imagens, áudios e documentos online.

Dentre essas 3 categorias, estima-se que até 90% de todos os dados no mundo estão a forma de dados não estruturados. (ICD, 2011).

•**Veracidade**: Um em cada 3 líderes não confiam nos dados que recebem (IBM). Para colher bons frutos do processo do *Big Data* é necessário obter dados verídicos, de acordo com a realidade. O conceito de velocidade, já descrito, é bem alinhado ao conceito de veracidade pela necessidade constante de análise em tempo real, isso significa, de dados que condizem com a realidade daquele momento, pois dados passados não podem ser considerados dados verídicos para o momento em que é analisado. A relevância dos dados coletados é tão importante quanto o primeiro conceito. A verificação dos dados coletados para adequação e relevância ao propósito da análise é um ponto chave para se obter dados que agreguem valor ao processo. (Hurwitz, Nugent, Halper & Marcia Kaufman).;

•**Valor**: Quanto maior a riqueza de dados, mais importante é saber realizar as perguntas certas no início de todo processo de análise (Brown, Eric, 2014). É necessário estar focado para a orientação do negócio, o valor que a coleta e análise dos dados trará para o negócio. Não é viável realizar todo o processo de *Big Data* se não se tem questionamentos que ajudem o negócio de modo realístico. Da mesma forma é importante estar atento aos custos envolvidos nessa operação, o valor agregado de todo esse trabalho desenvolvido, coleta, armazenamento e análise de todos esses da/dos tem que compensar os custos financeiros envolvidos (Taurion, 2013).

HADOOP E MAP REDUCE

Pouca gente na indústria de computação não deve ter se deparado com o termo “Big Data” e “Hadoop”. Essas são algumas das palavras da moda que surgem com frequência nos dias de hoje. Apesar de às vezes superestimado, trata-se de algo muito importante para todas as empresas de análises e os responsáveis por políticas. Vejamos então sobre o que é esse buzz todo.

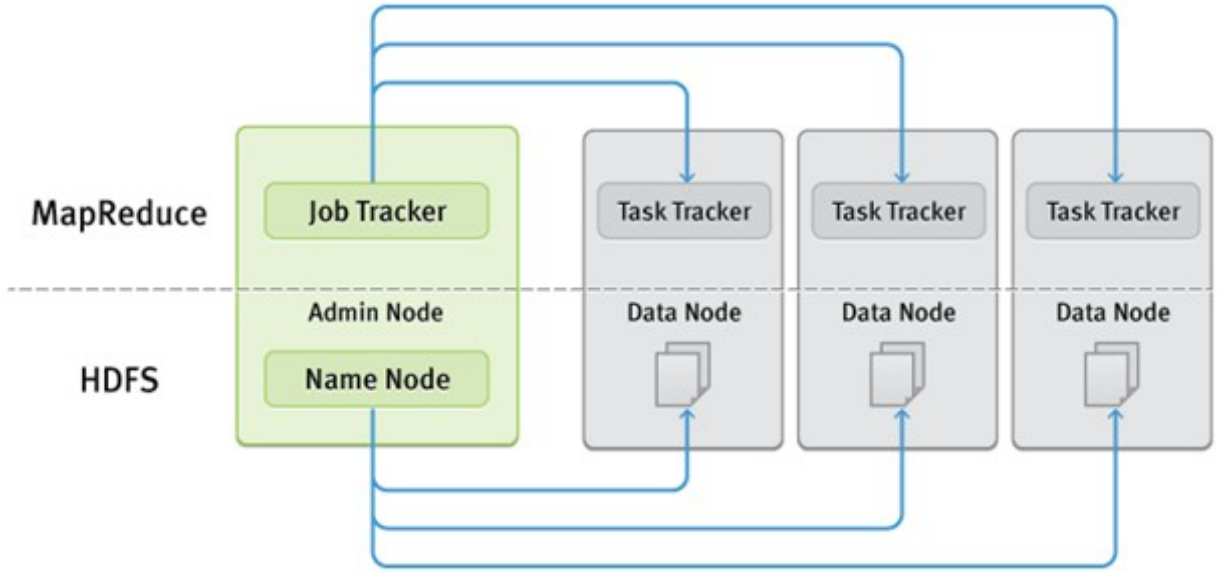
Desde o princípio da Internet, quantidades massivas de dados de usuários têm sido geradas. Particularmente nos últimos anos, mídias sociais como Facebook, Twitter e blogs criaram quantidades obscenas de dados de usuários. De acordo com o Gartner, Big Data são grandes quantidades de dados, em alta velocidade, gerados por uma multiplicidade de fontes. Por serem criados de forma quase aleatória, esses dados não possuem estrutura. Essas informações podem ser analisadas para ajudar em tomadas de decisões mais eficientes e inteligentes. Por causa dessas características, a manipulação e o processamento de Big Data necessita de ferramentas e técnicas especiais. É aqui que entra o Hadoop.

O Hadoop é uma implementação de código aberto do paradigma de programação Map-Reduce. Map-Reduce é um paradigma de programação introduzido pelo Google para processar e analisar grandes conjuntos de dados. Todos esses programas que são desenvolvidos nesse paradigma realizam o processamento paralelo de conjuntos de dados e podem, portanto, ser executados em servidores sem muito esforço. A razão para a escalabilidade desse paradigma é a natureza intrinsecamente distribuída do funcionamento da solução. Uma grande tarefa é dividida em várias tarefas pequenas que são então executadas em paralelo em máquinas diferentes e então combinadas para chegar à solução da tarefa maior que deu início a tudo. Os exemplos de uso do Hadoop são analisar padrões de usuários em sites de e-commerce e sugerir novos produtos que eles possam comprar.

Isso é tradicionalmente chamado de sistema de recomendações e pode ser encontrado em todos os principais sites de e-commerce. Ele pode ser utilizado também para processar grandes grafos como o Facebook etc. A razão pela qual o Hadoop simplificou o processamento paralelo se dá pelo fato de o desenvolvedor não precisar se preocupar com problemas relativos ao processamento em paralelo. Ele pode escrever apenas as funções de como quer que os dados sejam processados.

Componentes do Apache Hadoop

O framework do Hadoop é formado por dois componentes principais: armazenamento e processamento. O primeiro é o HDFS (Hadoop Distributed File System), que manipula o armazenamento de dados entre todas as máquinas na qual o cluster do Hadoop está sendo executado. O segundo, o Map-Reduce, manipula a parte do processamento do framework. Vamos olhar as duas individualmente.



HDFS (Hadoop Distributed File System)

O HDFS é um sistema de arquivos escalonável e distribuído, cujo desenho é baseado fortemente no GFS (Google File System), que também é um sistema de arquivo distribuído. Sistemas de arquivo distribuídos são necessários, uma vez que os dados se tornem grandes demais para serem armazenados em apenas uma máquina. Por conta disso, toda a complexidade e as incertezas provenientes do ambiente de rede entram em cena, o que faz com que sistemas de arquivos de rede sejam mais complexos do que sistemas de arquivos comuns. O HDFS armazena todos os arquivos em blocos. O tamanho do bloco padrão é 64Mb. Todos os arquivos no HDFS possuem múltiplas réplicas, o que auxilia o processamento em paralelo. Os clusters HDFS possuem dois tipos de nós – primeiro um namenode, que é um master, e múltiplos datanodes, que são nós slave. Fora esses dois, também é possível ter namenodes secundários.

Namenode: administra o namespace do sistema de arquivos. Ele gerencia todos os arquivos e diretórios. Namenodes possuem o mapeamento entre arquivos e os blocos nos quais estes estão armazenados. Todos os arquivos são acessados usando esses namenodes e datanodes.
Datanode: armazena os dados em forma de blocos. Datanodes se reportam a namenodes sobre os arquivos que possuem armazenados para que o namenode esteja ciente e os dados possam ser processados. Namenode é talvez o principal ponto crucial de falha do sistema, sem o qual os dados não podem ser acessados.
Namenodes secundários: esse node é responsável por checar a informação do namenode. No caso de falha, podemos usar esse nó para reiniciar o sistema.

Map-Reduce

Map-Reduce é um paradigma de programação em que cada tarefa é especificada em termos de funções de mapeamento e redução. Ambas as tarefas rodam paralelamente no cluster. O

armazenamento necessário para essa funcionalidade é fornecido pelo HDFS. A seguir estão os principais componentes do Map-Reduce.

Job Tracker: tarefas de Map-Reduce são submetidas ao Job Tracker. Ele precisa falar com o Namenode para conseguir os dados. O Job Tracker submete a tarefa para os nós task trackers. Esses task tracker precisam se reportar ao Job Tracker em intervalos regulares, especificando que estão “vivos” e efetuando suas tarefas. Se o task tracker não se reportar a eles, então o nó é considerado “morto” e seu trabalho é redesignado para outro task tracker. O Job tracker é novamente um ponto crucial de falha. Se o Job Tracker falhar, não poderemos rastrear as tarefas.
Task Tracker: o Task Tracker aceita as tarefas to Job Tracker. Essas tarefas são tanto de map, reduce ou ambas (shuffle). O Task Tracker cria um processo JVM separado para cada tarefa a fim de se certificar de que uma falha no processo não resulte em uma falha de Task Tracker. Task trackers também se reportam ao Job Tracker continuamente para que este possa manter o registro de tarefas bem ou mal sucedidas.

HDFS

O HDFS é um projeto da Apache Software Foundation e um subprojeto do projeto Apache Hadoop (veja Recursos). O Hadoop é ideal para armazenar grandes quantidades de dados, do porte de terabytes e pentabytes, e usa o HDFS como sistema de armazenamento. O HDFS permite a conexão de *nós* (computadores pessoais padrão) contidos nos clusters por meio dos quais os arquivos de dados são distribuídos. É possível acessar e armazenar os arquivos de dados como um sistema de arquivos contínuo. O acesso aos arquivos de dados é gerenciado de um modo em *fluxo*, o que significa que aplicativos ou comandos são executados diretamente por meio do modelo de processamento MapReduce (veja Recursos, novamente).

O HDFS é tolerante a falhas e disponibiliza acesso de alto rendimento a grandes conjuntos de dados. Este artigo explora os principais recursos desse sistema e apresenta uma visualização de alto nível da arquitetura do HDFS.

Visão geral do HDFS

O HDFS tem muitas similaridades com outros sistemas de arquivos distribuídos, mas é diferente em vários aspectos. Uma diferença notável é o modelo WORM (write-once-read-

many) do HDFS que afrouxa as exigências do controle de simultaneidade, simplifica a persistência de dados e habilita acesso de alto rendimento.

Outro atributo exclusivo do HDFS é o argumento que, normalmente, é melhor localizar a lógica de processamento próxima dos dados, ao invés de mover os dados para o espaço do aplicativo.

O HDFS restringe a gravação dos dados rigorosamente a um gravador por vez. Os bytes são sempre anexados ao final do fluxo e há a garantia de que os fluxos de bytes serão armazenados na ordem gravada.

O HDFS tem muitos objetivos. Estes são alguns dos mais notáveis:

•Tolerância a falhas pela detecção de falhas e aplicação de recuperação rápida, automática

•Acesso a dados por meio do fluxo MapReduce

•Modelo de simultaneidade simples e robusto

•Lógica de processamento próxima aos dados, ao invés dos dados estarem próximos à lógica de processamento

•Portabilidade entre sistemas operacionais e hardware padrão heterogêneos

•Escalabilidade para armazenar e processar de modo confiável grandes quantidades de dados

•Economia pela distribuição de dados e pelo processamento entre clusters de computadores pessoais padrão

•Eficiência pela distribuição de dados e pela lógica para processá-los em paralelo nos nós em que os dados estão localizados

•Confiabilidade pela manutenção automática de várias cópias dos dados e pela reimplementação automática da lógica de processamento no caso de falhas

O HDFS fornece interfaces para os aplicativos a fim de movê-los para perto de onde se localizam os dados, como descrito na próxima seção.

Interfaces de aplicativo no HDFS

É possível acessar o HDFS de muitos modos diferentes. O HDFS disponibiliza uma interface de programação de aplicativos (API) Java™ e um wrapper nativo em linguagem C para a API Java. Além disso, é possível usar um navegador da web para buscar arquivos no HDFS.

O aplicativo descrito na Tabela 1 também está disponível para interface com o HDFS.

Tabela 1. Aplicativos que podem servir de interface com o HDFS

Aplicativo	Descrição
FileSystem (FS) shell	Uma interface da linha de comandos para shells comuns do Linux® e UNIX® (bash, csh etc.) que permite a interação com os dados do HDFS.
DFSAdmin	Um conjunto de comandos que pode ser usado para administrar um cluster HDFS.
`fsck`	Um subcomando do comando/aplicativo do Hadoop. É possível usar o comando `fsck` para procurar por inconsistências nos arquivos, por exemplo, blocos perdidos, mas não para corrigir tais inconsistências.
Nós de nome e nós de dados	Estão integrados nos servidores da web que permitem que os administradores verifiquem o status atual de um cluster.

O HDFS tem um conjunto de recursos extraordinário com expectativa elevada, graças à sua arquitetura simples, porém eficiente.

Arquitetura do HDFS

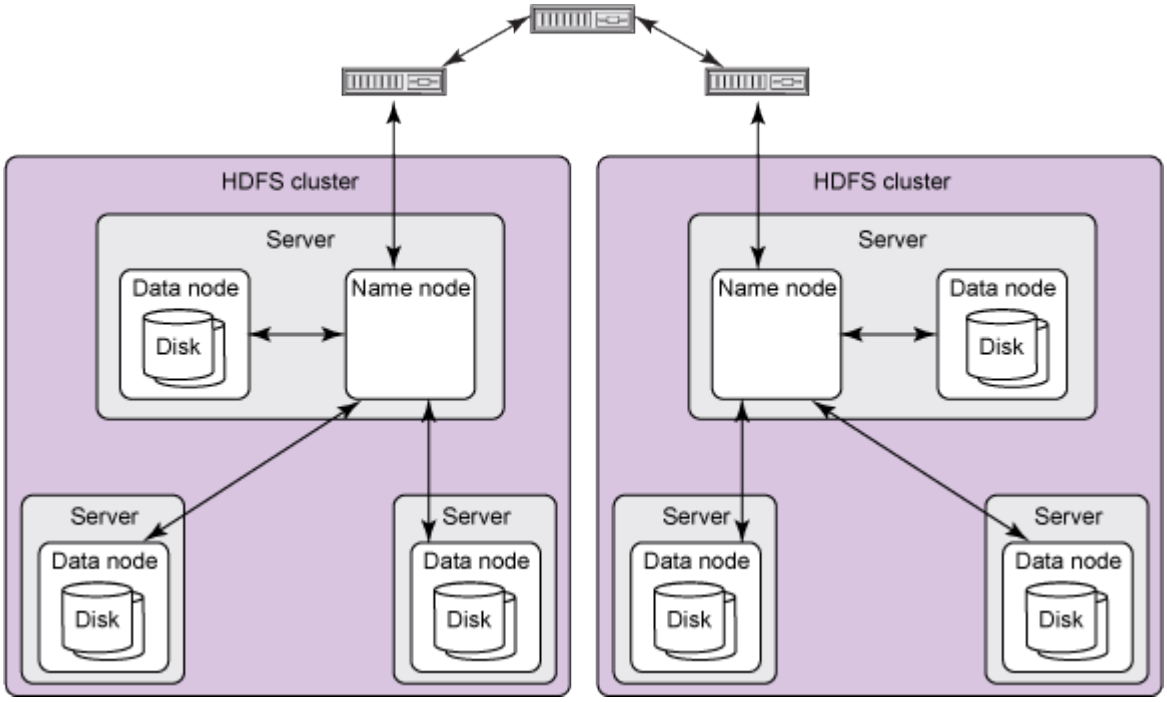
O HDFS é composto por clusters de nós interconectados no local onde os arquivos e diretórios residem. Um cluster HDFS consiste em um único nó, conhecido como um `NameNode`, que gerencia o namespace do sistema de arquivos e regula o acesso do cliente aos arquivos. Além disso, os nós de dados (`DataNodes`) armazenam dados como blocos dentro dos arquivos.

Nós de nome e nós de dados

Dentro do HDFS, um nó de nome gerencia operações de namespace do sistema de arquivos do tipo abrir, fechar e renomear arquivos e diretórios. Um nó de nome também mapeia blocos de dados a nós de dados, os quais gerenciam as solicitações de leitura e gravação dos clientes HDFS. Os nós de dados também criam, excluem e replicam blocos de dados de acordo com as instruções do nó de nome dominante.

A Figura 1 ilustra a arquitetura de alto nível do HDFS.

Figura 1. A arquitetura do HDFS



Como ilustrado na Figura 1, cada cluster contém um nó de nome. Esse design facilita um modelo simplificado para gerenciamento de cada namespace e mediação da distribuição de dados.

Relacionamentos entre nós de nome e nós de dados

Nós de nome e nós de dados são componentes de software desenvolvidos para executar independentemente em máquinas padrão entre sistemas operacionais heterogêneos. O HDFS foi desenvolvido com a linguagem de programação Java; assim, qualquer máquina com suporte para essa linguagem pode executar o HDFS. Um cluster de instalação típico tem uma máquina dedicada com um nó de nome e, possivelmente, um nó de dados em operação. Cada uma das outras máquinas no cluster executa um nó de dados.

Os nós de dados ficam em loop continuamente, solicitando instruções ao nó de nome. Um nó de nome não pode se conectar diretamente a um nó de dados; ele simplesmente retorna os valores das funções chamadas por um nó de

Protocolos de comunicação

Todos os protocolos de comunicação do HDFS são desenvolvidos no protocolo TCP/IP. Os clientes HDFS se conectam a uma porta do Protocolo de Controle de Transmissões (TCP) aberta no nó de nome e se comunicam com esse nó por meio de um protocolo proprietário baseado em Chamada de Procedimento Remoto (RPC). Os nós de dados conversam com o nó de nome através de um protocolo proprietário baseado em bloco.

dados. Cada nó de dados mantém um soquete do servidor aberto de modo que o código do cliente ou outros nós de dados possam ler ou gravar dados. O nó de nome conhece o host ou porta para esse soquete do servidor e fornece as informações aos clientes ou outros nós de dados interessados. Veja a barra lateral Communications protocols para obter mais informações sobre a comunicação entre nós, nós de dados e clientes.
O nó de nome mantém e administra mudanças no namespace do sistema de arquivos.

Namespace do sistema de arquivos

O HDFS suporta uma organização hierárquica tradicional de arquivos em que um usuário ou um aplicativo pode criar diretórios e armazenar arquivos neles. A hierarquia do namespace do sistema de arquivos é similar à maioria dos outros sistemas de arquivos existentes; é possível criar, renomear, reposicionar e remover arquivos.
O HDFS também suporta sistemas de arquivos de terceiros como o CloudStore e o Simple Storage Service (S3) (veja Recursos).

Replicação de dados

O HDFS replica blocos de arquivos para tolerância a falhas. Um aplicativo pode especificar o número de réplicas de um arquivo no instante em que ele é criado e é possível alterar esse número a qualquer momento depois disso. O nó de nome toma todas as decisões referentes à replicação de bloco.
O HDFS usa um modelo inteligente de colocação de réplica para fins de confiabilidade e desempenho. A otimização da colocação de réplica torna o HDFS exclusivo na maioria dos outros sistemas de arquivos distribuídos e é facilitado por uma política de colocação de réplica com reconhecimento de rack que usa a largura de banda da rede de modo eficiente.
Ambientes de grande porte do HDFS normalmente operam em várias instalações de computadores. A comunicação entre dois nós de dados em instalações diferentes é, normalmente,

Reconhecimento de rack
Normalmente, clusters HDFS de grande porte estão acondicionados em várias instalações (racks). O tráfego de rede entre nós diferentes dentro da mesma instalação é mais eficiente do que o tráfego de rede entre instalações. Um nó de nome tenta colocar réplicas de um bloco em várias instalações para aprimorar a tolerância a falhas. No entanto, o HDFS deixa aos administradores a decisão sobre a que instalação um nó pertence. Assim, cada nó conhece o ID do seu rack; é isso que significa *reconhecimento de rack*.

mais lenta do que entre nós de dados na mesma instalação. Assim, o nó de nome tenta otimizar a comunicação entre nós de dados. O nó de nome identifica o local dos nós de dados pelos seus IDs de rack.

Organização de dados

Um dos principais objetivos do HDFS é suportar arquivos grandes. O tamanho de um bloco típico do HDFS é 64 MB. Assim, cada arquivo HDFS é composto por um ou mais blocos de 64 MB. O HDFS tenta colocar cada bloco em nós de dados separados.

Processo de criação de arquivo

Manipular arquivos no HDFS é similar aos processos usados com outros sistemas de arquivos. No entanto, como o HDFS é um sistema de várias máquinas que parece ser um único disco, todo o código de manipulação de arquivos no HDFS usa uma subclasse do objeto `org.apache.hadoop.fs.FileSystem` (veja Recursos).

O código mostrado na Listagem 1 ilustra um processo típico de criação de arquivo no HDFS.

Listagem 1. Processo típico de criação de arquivo no HDFS

```
byte[] fileData = retrieveFileDataFromSomewhere();
String filePath = retrieveFilePathStringFromSomewhere();
Configuration config = new Configuration();  // assumes to
automatically load
                                             // hadoop-default.xml
and hadoop-site.xml
org.apache.hadoop.fs.FileSystem hdfs =
org.apache.hadoop.fs.FileSystem.get(config);
org.apache.hadoop.fs.Path path = new
org.apache.hadoop.fs.Path(filePath);
org.apache.hadoop.fs.FSDataOutputStream outputStream =
hdfs.create(path);
outputStream.write(fileData, 0, fileData.length);
```

Preparando-se para confirmar

Quando um cliente cria um arquivo no HDFS, ele primeiro armazena os dados em um arquivo local temporário. Em seguida, ele redireciona as gravações subsequentes para o arquivo temporário. Quando o arquivo temporário acumula dados suficientes para preencher

um bloco do HDFS, o cliente informa isso para o nó de nome e este, por sua vez, converte o arquivo para um nó de dados permanente. A seguir, o cliente fecha o arquivo temporário e esvazia quaisquer dados remanescentes para o nó de dados recém-criado. A essa altura, o nó de nome confirma o nó de dados para o disco.

Enfileiramento de replicações

Quando um cliente acumula um bloco de dados do usuário repleto, ele recupera uma lista de nós de dados que contém uma réplica desse bloco no nó de nome. O cliente, por sua vez, esvazia o bloco de dados repleto para o primeiro nó de dados especificado na lista de réplica. À medida que o nó recebe chunks de dados, ele os grava no disco e transfere cópias para o próximo nó de dados na lista. O próximo nó de dados faz o mesmo. Esse processo de *enfileiramento* é repetido até que o fator de replicação seja satisfeito.

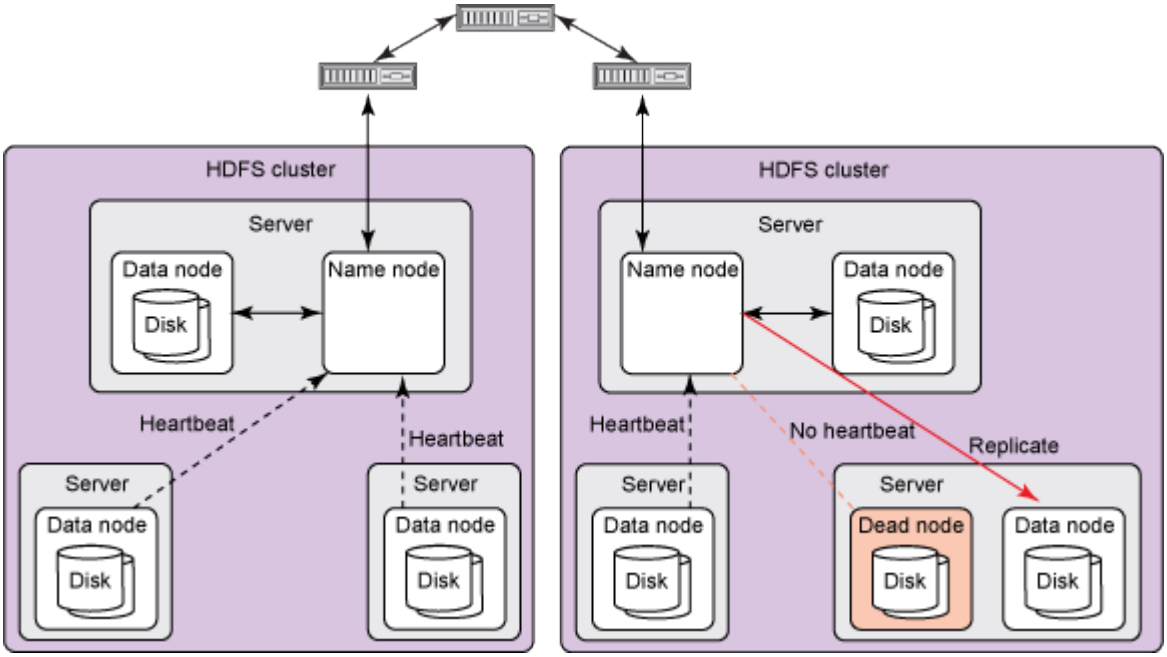
Confiabilidade do armazenamento de dados

Um objetivo importante do HDFS é armazenar dados de modo confiável, mesmo quando ocorrem falhas dentro dos nós de nome, dos nós de dados ou das partições de rede.
A detecção é a primeira etapa empregada pelo HDFS para superar as falhas. O HDFS usa mensagens de pulsação para detectar a conectividade entre os nós de nome e de dados.

Pulsações do HDFS

Vários motivos podem causar perda de conectividade entre os nós de nome e de dados. Consequentemente, cada nó de dados envia mensagens de pulsação periódicas para o nó de nome, de modo que este possa detectar a perda de conectividade se ele parar de recebê-las. O nó de nome marca os nós de dados que não respondem às pulsações como inativos e deixa de enviar solicitações adicionais a eles. Os dados armazenados em um nó inativo não estão mais disponíveis para um cliente HDFS nesse nó, que seja eficazmente removido do sistema. Se a inatividade de um nó fizer com que o fator de replicação de blocos de dados caia abaixo do seu valor mínimo, o nó de nome iniciará a replicação adicional para recolocar a replicação em um estado normalizado.
A Figura 2 ilustra o processo de envio de mensagens de pulsação do HDFS.

Figura 2. O processo de pulsação do HDFS



Rebalanceamento do bloco de dados

Nem sempre é possível colocar blocos de dados uniformemente entre nós de dados, o que significa que o espaço usado para um ou mais nós de dados pode estar subutilizado. Assim, o HDFS suporta o rebalanceamento de blocos de dados usando diversos modelos. Um modelo pode mover blocos de dados de um nó de dados para outro automaticamente quando o espaço livre em um nó fica muito reduzido. Outro modelo pode criar dinamicamente réplicas adicionais e rebalancear outros blocos de dados em um cluster, caso ocorra um aumento súbito na demanda de um determinado arquivo. O HDFS também disponibiliza o comando `hadoop balance` para tarefas manuais de rebalanceamento.

Um motivo comum para o rebalanceamento é a inclusão de novos nós de dados em um cluster. Ao posicionar novos blocos, os nós de nome consideram vários parâmetros antes de escolher que nós de dados os receberão. Algumas das considerações são:

•Políticas de gravação de réplica de bloco

•Prevenção de perda de dados devido à falha da instalação ou do rack

•Redução da E/S da rede entre instalação

•Difusão uniforme de dados entre os nós de dados em um cluster

O recurso de rebalanceamento de cluster do HDFS é apenas um mecanismo que ele usa para manter a integridade dos dados. Outros mecanismos são abordados a seguir.

Integridade de dados

O HDFS percorre grandes extensões para assegurar a integridade dos dados entre clusters. Ele usa validação de soma de verificação nos conteúdos dos arquivos do HDFS armazenando somas de verificação calculadas em arquivos ocultos, separados, no mesmo namespace que os dados reais. Quando um cliente recupera os dados do arquivo, é possível verificar se os dados recebidos correspondem à soma de verificação armazenada no arquivo associado.

O namespace do HDFS é armazenado usando um log de transação mantido por cada nó de nome. O namespace do sistema de arquivos, junto com os mapeamentos de blocos do arquivo e as propriedades do sistema do arquivo, são armazenados em um arquivo chamado FsImage. Quando um nó de nome é inicializado, ele lê o arquivo FsImage junto com outros arquivos e aplica as transações e informações de estado localizadas nesses arquivos.

Atualização síncrona de metadados

Um nó de nome usa um arquivo de log conhecido como EditLog para registrar persistentemente cada transação ocorrida nos metadados do sistema de arquivos do HDFS. Se os arquivos EditLog ou FsImage vierem a estar corrompidos, a instância do HDFS à qual eles pertencem deixará de funcionar. Assim, um nó de nome suporta várias cópias dos arquivos FsImage e EditLog. Com a presença de várias cópias desses arquivos, qualquer mudança em um dos arquivos será disseminada simultaneamente para todas as cópias. Quando um nó de nome reinicia, ele usa a última versão consistente do FsImage e do EditLog na sua autoinicialização.

Permissões do HDFS para usuários, arquivos e diretórios

O HDFS implementa um modelo de permissão para arquivos e diretórios que tem muito em comum com o modelo Portable Operating System Interface (POSIX); por exemplo, cada arquivo e diretório está associado a um proprietário e a um grupo. O modelo de permissões do HDFS suporta leitura (r), gravação (w) e execução (x). Como não há nenhum conceito de execução de arquivo dentro do HDFS, a permissão x assume um significado diferente. Em termos simples, o atributo x indica permissão para acessar um diretório-filho de um

determinado diretório-pai. O proprietário de um arquivo ou diretório é a identidade do processo cliente que o criou. O grupo é o grupo do diretório-pai.

Capturas Instantâneas

O HDFS foi planejado originalmente para suportar capturas instantâneas que podem ser usadas para retroceder uma instância corrompida do HDFS a um estado anterior. No entanto, o suporte do HDFS para capturas instantâneas foi adiado por enquanto.

Resumo

O Hadoop é um sistema de arquivos distribuído do Apache Software Foundation e um projeto de gerenciamento de dados que se destina a armazenar e gerenciar grandes quantidades de dados. Ele usa um sistema de armazenamento denominado HDFS para conectar computadores pessoais padrão, conhecidos como nós, contidos em clusters nos quais os blocos de dados são distribuídos. É possível acessar e armazenar os blocos de dados como um sistema de arquivos contínuo que usa o modelo de processamento MapReduce.
O HDFS compartilha recursos comuns com outros sistemas de arquivos distribuídos enquanto suporta algumas diferenças importantes. Uma diferença significativa é o modelo WORM (write-once-read-many) do HDFS que afrouxa as exigências do controle de simultaneidade, simplifica a persistência de dados e habilita acesso de alto rendimento. Para disponibilizar um modelo otimizado de acesso a dados, o HDFS foi desenvolvido de modo a posicionar a lógica de processamento próxima aos dados, ao invés de posicionar os dados próximos ao espaço do aplicativo.