Análise do Impacto da Replicação de Dados Implementada pelo Apache Hadoop no Balanceamento de Carga

Rhauani Weber Aita Fazul, Paulo Vinicius Cardoso, Patricia Pitthan Barcelos

Análise do Impacto da Replicação de Dados Implementada pelo Apache Hadoop no Balanceamento de Carga

Rhauani Weber Aita Fazul[1], Paulo Vinicius Cardoso[2], Patrícia Pitthan Barcelos[2]

[1]Laboratório de Sistema de Computação (LSC)
[2]Pós-Graduação em Ciência da Computação (PGCC)
Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brasil

{rwfazul, pcardoso, pitthan}@inf.ufsm.br

Abstract. *Big Data processing tools, such as Apache Hadoop, should ensure data integrity and availability through fault tolerance mechanisms. The HDFS, Hadoop Distributed File System, implements several fault tolerance techniques, among them the traditional data replication. To deal with highly scalable clusters, there is a concern in validate if the replicated data is spread homogeneously among the computational nodes. In this paper, we analyze experimentally the behavior of HDFS in scenarios with and without the occurrence of failures in order to collect metrics of load balancing regarding the process of data replication adopted by Apache Hadoop. Additional experiments measure the performance achieved by balancing a cluster.*

Resumo. *Ferramentas especializadas em Big Data, como o Apache Hadoop, devem portar meios que garantam a integridade e a disponibilidade dos dados. Para tal, o HDFS, sistema de arquivos distribuído do Hadoop, faz uso de diversas técnicas de tolerância a falhas, dentre elas a replicação de dados. Tratando-se de clusters altamente escaláveis, há a preocupação em verificar se a distribuição dos dados replicados ocorre de forma homogênea. Este trabalho analisa experimentalmente o comportamento do HDFS em cenários com e sem a ocorrência de falhas, avaliando o desbalanceamento de carga resultante do processo de replicação de dados implementado pelo Apache Hadoop. Experimentos adicionais medem o desempenho alcançado ao balancear um cluster.*

1. Introdução

O conceito de *Big Data* refere-se a uma enorme quantidade de dados produzidos em uma velocidade extremamente alta, o que faz com que técnicas tradicionais não sejam suficientes para sua manipulação efetiva. Os cenários de *Big Data*, de modo geral, estão relacionados a altas demandas de escalabilidade, distribuição de dados, paralelismo de processamento e armazenamento de estruturas heterogêneas.

Ferramentas convencionais, mesmo com constantes melhorias e aprimoramentos, nem sempre são eficientes ao lidar com esse volume de dados massivo. Fortemente consolidado, o Apache Hadoop[1] é uma das principais tecnologias neste meio. Sendo um *framework open source* baseado em Java, o Hadoop permite o processamento de grandes

[1]https://hadoop.apache.org/

volumes de dados através de modelos de programação simples. Um componente fundamental do Hadoop é o seu sistema de arquivos distribuído (HDFS), um mecanismo confiável para o armazenamento de arquivos em *clusters* altamente escaláveis.

Um dos requisitos fundamentais para promover alta disponibilidade é a tolerância a falhas. Para tal, o HDFS implementa diversos mecanismos que buscam assegurar a confiabilidade e a disponibilidade do sistema. A replicação pode ser considerada um dos principais mecanismos nesse sentido, fazendo com que o HDFS distribua as réplicas dos dados em diversos nós do *cluster*. Por sua vez, não há garantias de que esta distribuição ocorra de forma equilibrada, sendo que alguns fatores, como a adição de um novo nó ao *cluster* ou a ocorrência de falhas, podem tornar o desbalanceamento acentuado.

Este trabalho propõe uma análise experimental da replicação de dados adotada pelo Apache Hadoop. Os testes foram efetuados de modo a examinar o comportamento do HDFS envolvendo o balanceamento na distribuição de blocos no *cluster*. Além disso, comparações de desempenho foram realizadas em cenários com e sem balanceamento de carga, de modo a evidenciar benefícios de uma distribuição de dados equilibrada.

O artigo está organizado em cinco seções. O *framework* Apache Hadoop e seus principais mecanismos de tolerância a falhas, concentrando-se na replicação de dados realizada pelo HDFS, são apresentados na Seção 2. A Seção 3 descreve a metodologia adotada nos experimentos. A Seção 4 exibe e discute os resultados obtidos. Por fim, a Seção 5 apresenta as considerações finais e direciona os trabalhos futuros.

2. Apache Hadoop

Surgido como uma plataforma voltada ao armazenamento e ao processamento de dados em larga escala, o *framework* Apache Hadoop é uma tecnologia amplamente difundida e consolidada, tanto em termos comerciais quanto no ambiente acadêmico. A simplicidade de configuração e de uso da ferramenta, aliada a uma estrutura de desenvolvimento *open-source*, oferecem às organizações alta flexibilidade na manipulação de *Big Data*.

Conectando equipamentos de baixo custo para operar em paralelo, o Hadoop provê alta escalabidade em sistemas distribuídos. Seu modelo de programação MapReduce foi projetado para processar grandes volumes de dados, dividindo o trabalho em tarefas independentes, que podem ser executadas paralelamente em diferentes nós do *cluster*.

Entre os principais componentes do Hadoop, compondo a camada de gerenciamento de dados da ferramenta, estão o *Another Resource Negotiator* (YARN) e o *Hadoop Distributed File System* (HDFS) [White 2015]. O YARN foi introduzido a partir da versão 2 do Hadoop como uma plataforma de processamento de dados de uso geral, responsável por atribuir recursos computacionais para a execução de aplicações dentro do *framework*.

Outro componente chave do Apache Hadoop é o seu sistema de arquivos distribuído. O HDFS é um mecanismo confiável e tolerante a falhas para o armazenamento de arquivos em *clusters* altamente escaláveis, que permite que inúmeras aplicações - coordenadas pelo YARN - manipulem dados simultaneamente de forma segura e confiável. O HDFS emprega uma arquitetura mestre-escravo baseada em dois tipos de nós: $NameNode$ (NN) e $DataNode$ (DN). O $NameNode$ é o servidor mestre que gerencia o *namespace* (metadados) do sistema e controla o acesso e a distribuição dos arquivos. Enquanto isso, os $DataNodes$ realizam efetivamente o armazenamento dos dados

no HDFS. Múltiplas instâncias de $DataNodes$ possibilitam que a distribuição de dados ocorra em diferentes máquinas do *cluster*, conectadas por rede.

A alta escabilidade do Hadoop torna possível que os *clusters* possuam milhares de nós trabalhando em conjunto, responsáveis tanto por computação quanto por armazenamento de dados. Mesmo que alguns componentes de *hardware* falhem, é importante que o sistema mantenha sua consistência. O ecossistema do Hadoop possui diversos mecanismos voltados para a detecção e recuperação de falhas. Este artigo é focado nos mecanismos de tolerância a falhas implementados pelo HDFS.

2.1. Tolerância a Falhas no HDFS

Sendo voltado a cenários de *Big Data*, o HDFS oferece suporte ao armazenamento de dados na escala de petabytes. Para lidar com a quantidade massiva de dados, definiu-se um processo padrão para o uso do sistema de arquivos do Hadoop, ilustrado na Figura 1.

Ao armazenar algum arquivo no HDFS, ele é automaticamente dividido em arquivos menores (blocos). Todos os blocos (exceto o último) possuem o mesmo tamanho, o que possibilita calcular o número exato de blocos que podem ser armazenados em cada nó do sistema. A partir da versão 2 adotou-se 128MB como tamanho de bloco padrão.

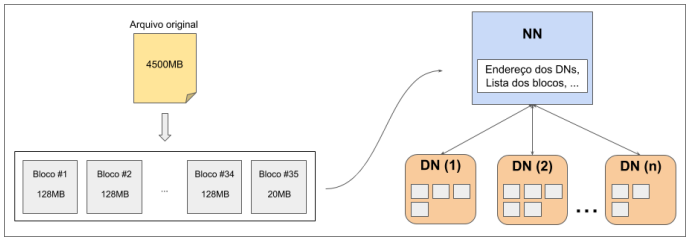


Figura 1. Processo de armazenamento de arquivos no HDFS.

Conforme exibe a Figura 1, os blocos são armazenados em um conjunto de nós escravos (DNs) determinado pelo NN. Se possível, cada parte do arquivo será armazenada em um DN diferente, o que permite um maior paralelismo na escrita e na leitura dos dados. Mesmo com a distribuição dos blocos, cabe ao HDFS assegurar confiabilidade e disponibilidade, de modo que nenhum dado seja perdido ou comprometido.

Os mecanismos de tolerância a falhas mais expressivos (i.e. estratégias que possibilitam recuperação automática) no HDFS são o envio de mensagens *Heartbeat*, o estabelecimento de *checkpoints* e a replicação de blocos [White 2015].

Durante o funcionamento normal do HDFS, os DNs enviam mensagens periódicas ao NN, indicando que estão ativos. Estas mensagens, denominadas *Heartbeats*, possibilitam a comunicação entre os dois nodos, além de permitirem a detecção de falhas no HDFS. Se o NN não receber nenhuma mensagem de um determinado DN dentro de um limite de tempo previamente estipulado, o nó em questão será dado como inativo. Um DN inativo não recebe mais instruções de leitura ou escrita de dados.

Vale ressaltar que as mensagens *Heartbeat* também carregam informações sobre a capacidade e a utilização de disco dos DNs. Conjuntamente, é enviada a quantidade de transferências de dados ativas que estão sendo executadas pelo nodo em questão no momento do envio da mensagem. Estes dados são utilizados pelo NN para decisões envolvendo gerência de espaço e balanceamento de carga nos DNs que compõem o *cluster*.

Além das mensagens *Heartbeat*, o HDFS implementa um mecanismo de *checkpoint and recovery*. Este mecanismo salva o contexto do sistema em intervalos de tempo predefinidos, criando um *checkpoint*. Em caso de falha, utiliza-se o último *checkpoint* salvo como base para o *roolback*, i.e. recuperação do sistema para o último estado seguro.

Atuando como um importante mecanismo de tolerância a falhas e como uma estratégia para suprir altas demandas de disponibilidade, a replicação de dados também está presente no HDFS. Sendo esta solução o foco do presente trabalho, a Seção 2.1.1 detalha como o processo de replicação de blocos ocorre no HDFS.

2.1.1. Replicação de Blocos

A replicação de blocos é um mecanismo crucial para o funcionamento do HDFS. A ideia baseia-se na criação de cópias dos blocos de dados presentes no sistema (réplicas), de modo a aumentar a confiabilidade e a disponibilidade de dados com base em redundância.

A criação de uma réplica pode ocorrer em diferentes momentos no ciclo de vida de um arquivo, sendo o NN o responsável pelas decisões que dizem respeito à replicação. Como abordado anteriormente, os arquivos inseridos no HDFS são divididos em blocos de tamanho fixo e, posteriormente, armazenados em diferentes DNs. Caso uma falha ocorra, deve-se preservar a integridade e a consistência dos dados, de forma que, caso necessário, seja possível acessar o bloco a partir de outros nós.

Ao ser armazenado no HDFS, um bloco é replicado com base em um fator de replicação previamente definido (por padrão utiliza-se 3 réplicas por bloco). O fator pode ser especificado no momento da criação de cada arquivo, sendo possível alterá-lo posteriormente. A Figura 2 ilustra o processo de replicação de blocos considerando o armazenamento de um arquivo, dividido em 5 blocos e com um fator de replicação fixado em 3, em um *cluster* com 2 *racks* contendo 3 DNs cada.

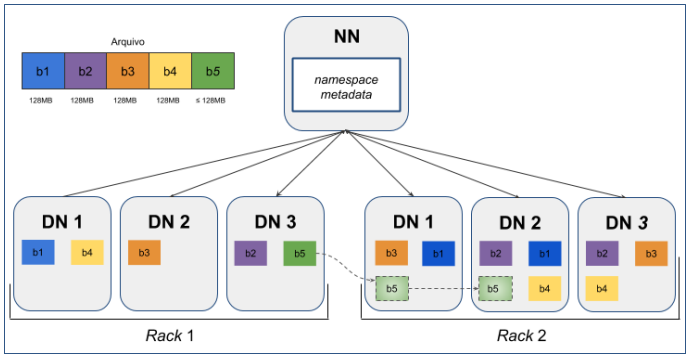


Figura 2. Mecanismo de Replicação de Dados ao armazenar um arquivo no HDFS.

A política de posicionamento de réplicas atual do HDFS (Apache Hadoop versão 2), baseada em um fator de replicação de 3, é armazenar a primeira réplica no mesmo nó onde a aplicação do cliente está executando (considerando que o cliente esteja dentro de um DN, caso contrário um DN é escolhido aleatoriamente). Como a chance de falha de um *rack* é menor que a chance de falha de um nodo, a segunda e a terceira réplicas são armazenadas em um mesmo *rack* remoto (diferente do local da primeira réplica), porém em dois DNs distintos. Essa distribuição pode ser observada na Figura 2.

Embora a replicação resulte em maior gasto no armazenamento, o HDFS tira proveito da redundância espacial dos dados para suprir altas demandas de acesso. Se um *cluster* HDFS estiver disposto em múltiplos *data centers*, as requisições de leitura serão atendidas a partir do *data center* mais próximo da origem da solicitação. De todo modo, as políticas de posicionamento de réplicas ainda estão sendo aprimoradas e a estratégia atual ainda é vista como um trabalho em desenvolvimento [Foundation 2018].

Em relação à sobrecarga no processo de armazenamento das réplicas, o HDFS usa uma técnica de *pipeline* na escrita das cópias. A partir do momento que o NN definir a lista dos três DNs que irão receber a cópia de um determinado bloco e o primeiro DN receber seu bloco de dados, este irá armazená-lo em seu disco local e transferi-lo para o segundo nó da lista, que irá transferi-lo para o terceiro DN e assim sucessivamente, conforme exemplificado no bloco *b5* da Figura 2. Dessa forma, um DN pode, simultaneamente, receber e encaminhar dados para os demais nós que compõem o *pipeline*.

Mesmo com estes aprimoramentos, a replicação inicial dos blocos no momento da escrita de um arquivo não é suficiente para garantir alta confiabilidade e alta disponibilidade. Em um sistema como o HDFS, que foi projetado para ser executado em *hardware* comum (i.e. não confiável), falhas de dados e de nodos são frequentes. Para realizar a replicação, pelo menos uma cópia do bloco deve existir. Logo, o NN precisa monitorar e controlar ativamente o número de réplicas de cada bloco presente no HDFS, de forma a proteger e manter a integridade do sistema em cenários onde ocorram falhas consecutivas em um curto intervalo de tempo.

O HDFS também usa o conceito de re-replicação, que pode ser motivada pelos seguintes fatores: o aumento do fator de replicação de um arquivo, a corrupção de uma réplica já existente, a ocorrência de uma falha de disco em um DN ou a indisponibilidade total de um nó [Foundation 2018]. Em todos os casos, o NN deve agir e reiniciar o processo de replicação em outros DNs para manter a conformidade com o fator mínimo de replicação definido. A estratégia desse processo é simples: para cada bloco que necessite de mais réplicas, o NN irá selecionar um DN que contenha uma das cópias originais do bloco e, em seguida, arbitrariamente definir um DN como destino do bloco re-replicado.

Ambos os processos de replicação e de re-replicação são realizados de forma transparente pelo HDFS. É natural que alguns DNs possuam mais blocos de dados em comparação com o restante dos nós do *cluster*, tendo em vista que não há nenhuma garantia de que a distribuição dos dados pelo NN seja uniforme. Ainda assim, falhas que resultem na re-replicação de blocos podem agravar a sobrecarga em DNs específicos e, com o passar do tempo, esse desbalanceamento pode se tornar ainda mais acentuado.

3. Metodologia

Com o objetivo de analisar o balanceamento na distribuição de dados dentro do HDFS, foram coletadas informações e estatísticas de uso do sistema de arquivos a partir da execução de um *benchmark*. Os experimentos foram realizados considerando a configuração do Hadoop em dois modos: pseudo-distribuído e totalmente distribuído.

No modo pseudo-distribuído (único nó), a interação entre os nodos é executada sobre a Máquina Virtual Java (JVM), onde cada *daemon* do Hadoop roda em uma instância distinta da JVM, simulando um *cluster* em pequena escala. Já em um modo totalmente distribuído, a execução é realizada em um *cluster* real, com múltiplos nós interconectados.

O contexto do ambiente deu-se por uma configuração do Hadoop (versão 2.9.1) com 1 NN e 10 DNs. Em ambos os modos, tomaram-se como base cenários com e sem a ocorrência de falhas de $DataNode$. As falhas de DN - representando nós com defeito no *cluster* - foram induzidas através do comando *kill* do Linux aos processos responsáveis pela gerência dos DNs selecionados em cada teste. A descoberta das falhas de DN é feita a partir de um dos mecanismos apresentados na Seção 2.1, as mensagens *Heartbeat*.

O NN considera inativos os DNs que não se comunicam dentro de um determinado intervalo. Um nó inativo acarreta na diminuição do fator de replicação dos blocos nele armazenados. Para prevenir a perda de dados, o NN responde as mensagens *Heartbeat* de algum DN que possua cópias destes blocos, enviando instruções para realizar uma nova replicação em outros nós, restaurando assim o fator de replicação e retornando o sistema para um estado seguro. Nos testes realizados, definiu-se que os DNs fossem marcados como inativos após aproximadamente 20 segundos sem o envio de mensagens *Heartbeat*.

Para analisar o balanceamento de carga no HDFS foram realizados testes de estresse por meio de um *benchmark* presente na distribuição do Hadoop. A aplicação usada foi o TestDFSIO [White 2015], que permite medir o desempenho do sistema a partir de operações de entrada e saída (i.e. escrita e leitura de dados). O *benchmark* foi usado para escrita de 15 arquivos. Os parâmetros de configuração referentes ao tamanho dos blocos de cada arquivo e do fator de replicação mínimo foram 128MB e 3, respectivamente.

Ao executar o Hadoop em um único nó, limita-se o paralelismo na execução das aplicações. Em contrapartida, no modo distribuído, com múltiplos nós trabalhando em conjunto, é possível manipular maiores quantidades de dados. Desta forma, por questões de limitações de *hardware*, o tamanho dos arquivos escritos durante os testes no modo pseudo-distribuído e totalmente distribuído foram distintos. Como o objetivo dos experimentos é avaliar a proporcionalidade na distribuição de dados nos DNs (independente de carga), os resultados não são afetados por essa disparidade.

Para ambos os modos de execução do Hadoop, foram realizados seis Casos de Teste (C.T.). Em cada teste, o número de falhas de DN induzidas durante a execução do *benchmark* TestDFSIO foi incrementada. A Tabela 1 demonstra a quantidade de falhas induzidas em cada C.T., onde o primeiro C.T. (cenário A) representa uma execução sem falhas de DN e o último (cenário F) uma execução com ocorrência de 50% de falhas, ou seja, execução com falhas em 5 dos 10 DNs inicialmente configurados.

Tabela 1. Casos de Teste adotados nos experimentos.

C.T.	Falhas	Ideal (%)
A	0	10
B	1	11.11
C	2	12.5
D	3	14.28
E	4	16.67
F	5	20

Além disso, para cada cenário, foi definido o valor ideal de ocupação de cada DN. Sendo DN_t o número total de $DataNodes$ configurados e DN_o o número de $DataNodes$ inativos, a fórmula para calcular os valores ideias (média para o balanceamento) é definida como: $Ideal(\%) = 100/(DN_t - DN_o)$. O valor obtido representa a porcentagem de dados a serem salvos em cada DN em relação ao volume total armazenado no sistema de

arquivos, de modo que a distribuição de dados possa ser considerada homogênea.

4. Experimentação

Esta Seção apresenta os experimentos realizados nos dois modos de execução do Apache Hadoop, conforme a metodologia descrita na Seção 3. Para aumentar a confiabilidade dos valores, em cada C.T. foi calculada a porcentagem de ocupação de disco de cada DN considerando a média aritmética de 10 execuções distintas. Por fim, na Seção 4.3 são apresentados os resultados alcançados após o balanceamento do *cluster*.

4.1. Modo pseudo-distribuído

No modo de operação pseudo-distribuído do Apache Hadoop, utilizou-se uma máquina com a seguinte configuração: processador Intel Core 2 Quad com frequência de 2.33GHz, 4GB de memória RAM e 750GB de disco rígido, executando um sistema operacional Ubuntu 16.04.1 com *Kernel* GNU/Linux 4.13.0-45-*generic*.

A execução do *benchmark* TestDFSIO deu-se pela escrita de 15 arquivos de 350MB cada. Sendo o fator de replicação definido como 3, 124 blocos de dados $((15 \times 350 \times 3) \div 128)$ foram escritos no HDFS em cada teste no modo pseudo-distribuído. A Tabela 2 apresenta os resultados referentes às porcentagens de ocupação de cada DN (em relação ao total armazenado no sistema de arquivos) após a execução do *benchmark*.

Tabela 2. Porcentagem de ocupação dos DNs no modo pseudo-distribuído.

DN	C.T.					
	A	**B**	**C**	**D**	**E**	**F**
1	9.55	X	X	X	X	X
2	**4.88**	12.85	X	X	X	X
3	5.63	12.13	9.99	X	X	X
4	9.89	10.47	9.12	16.65	X	X
5	11.56	8.48	**7.24**	12.93	14.63	X
6	10.77	11.79	10.39	**18.27**	18.78	**16.18**
7	8.58	**4.26**	15.06	**9.47**	**19.5**	20.65
8	12.94	13.52	9.74	14.8	17.51	19.64
9	**15.77**	11.85	**19.71**	15.46	16.02	19.74
10	10.43	**14.65**	18.75	12.42	**13.56**	**23.79**
Total	100					

De modo a evidenciar os valores discrepantes em relação a *baseline* estabelecida (valores ideais apresentados na Tabela 1), os valores mínimos e máximos de ocupação dos DNs durante cada Caso de Teste foram destacados em negrito. Como pode ser observado, há uma variação significativa na quantidade de dados armazenada em cada DN. Dessa forma, a política de posicionamento de réplicas empregada pelo HDFS fez com que, em cada teste, as operações de escrita de blocos realizadas ocorressem com uma frequência diferente em certos DNs, ocasionando em um desbalanceamento de carga.

4.2. Modo totalmente distribuído

Para a execução do Apache Hadoop no modo totalmente distribuído, foram configurados 10 nós na plataforma Grid'5000[2], cada um com dois processadores Intel Xeon E5520

[2]Grid'5000 é uma plataforma para experimentos apoiada por um grupo de interesses científicos hospedado por Inria e incluindo CNRS, RENATER e diversas Universidades, bem como outras organizações (mais detalhes em `https://www.grid5000.fr`).

(quatro cores por CPU) com frequência de 2.27GHz, 32GB de memória RAM e 557GB de disco rígido, com uma distribuição Debian 9.5 com *Kernel* 4.9.0-8-amd64. Neste modo de execução, cada um dos 10 nós foi responsável pela execução de um DN, já o NN foi configurado para executar no mesmo nó que o primeiro DN.

A execução do *benchmark* TestDFSIO consistiu na escrita de 15 arquivos de 10GB cada. Considerando o fator de replicação definido como 3, foram escritos 3.600 blocos de dados $((15 \times 10240 \times 3) \div 128)$ no HDFS em cada C.T.. A Tabela 3 exibe os resultados relacionados às porcentagens de ocupação de cada DN após a experimentação.

Tabela 3. Porcentagem de ocupação dos DNs no modo totalmente distribuído.

DN	C.T.					
	A	**B**	**C**	**D**	**E**	**F**
1	**17.38**	X	X	X	X	X
2	16.17	8.71	X	X	X	X
3	14.76	10.27	10.08	X	X	X
4	7.46	9.3	14	10.95	X	X
5	7.58	11.88	**17.49**	18.58	15.4	X
6	7.37	14.6	**9.1**	15.24	**21.25**	**25.35**
7	**6.92**	**14.85**	13.25	13.91	15.82	21.16
8	7.4	9.02	12.62	**10.76**	15.45	18.7
9	7.35	13.71	10.55	11.36	17.92	**16.87**
10	7.61	**7.66**	12.91	**19.2**	**14.16**	17.92
Total	100					

Os valores em negrito destacam os valores extremos (mínimos e máximos) obtidos em cada cenário. Percebe-se que, assim como no modo pseudo-distribuído, há um desequilíbrio considerável no que diz respeito à distribuição de dados entre os DNs. Logo, para se aproximar dos valores ideais para uma distribuição homogênea (vide Tabela 1), surge a necessidade do balanceamento do *cluster*.

4.3. Modo totalmente distribuído com balanceamento de carga

Conforme já citado (e atestado pelos resultados apresentados anteriormente), não há garantias de que uma distribuição de dados homogênea entre os DNs seja realizada de forma automática pelo HDFS. De todo modo, o balanceamento do *cluster* é um aspecto importante a ser considerado. A medida que o desbalanceamento fica acentuado, DNs com mais dados podem ficar sobrecarregados devido a um maior número de operações de E/S sendo realizadas em seus nodos e, consequentemente, DNs com menos dados podem ficar subutilizados. Sendo assim, espera-se que uma distribuição uniforme dos blocos de arquivos pelo *cluster* otimize o uso do HDFS [White 2015].

Em geral, o sistema de arquivos do Hadoop precisa lidar com um alto fluxo de dados constantemente, o que faz do balanceamento uma tarefa complexa. Para auxiliar neste processo, o Hadoop disponibiliza uma ferramenta que realiza o balanceamento do *cluster*, o HDFS *Balancer* [Hortonworks 2018]. Este balanceador analisa como os blocos estão posicionados entre os DNs e, até que o cluster possa ser considerado balanceado, move os dados dos DNs superutilizados para os DNs subutilizados.

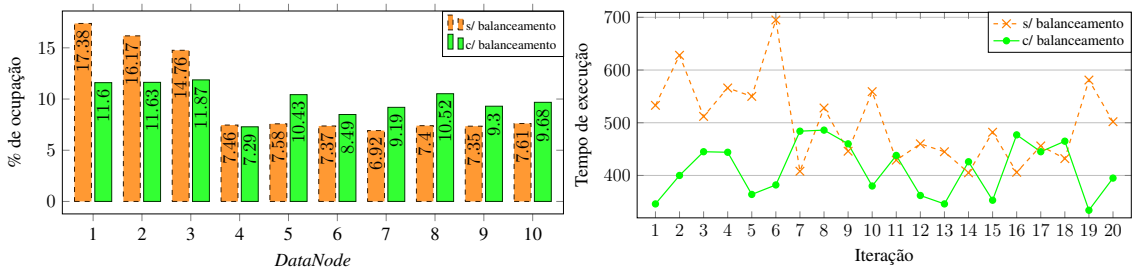
Um *cluster* é dado como balanceado quando o uso interno de seus DNs estiver dentro do limite (*threshold*) estipulado no início da execução do balanceador. Esse limite define a diferença máxima que a carga dos DNs (espaço utilizado no nodo em relação a sua capacidade total de armazenamento) e a carga total do *cluster* (espaço utilizado no *cluster*

em relação a sua capacidade total) pode assumir para que o *cluster* possa ser classificado como equilibrado. Isto proporciona certa flexibilidade no balanceamento de dados dentro do HDFS, permitindo que os DNs armazenem quantidades similares de dados, mas que ainda podem diferir moderadamente entre si.

O algoritmo do balanceador utiliza uma série de técnicas durante o deslocamento de dados entre DNs. Seu detalhamento pode ser encontrado em [Hortonworks 2018]. Para os fins deste trabalho, basta notar que a distribuição de blocos gerada a partir do balanceador adere à política de posicionamento de réplicas do HDFS apresentada na Seção 2.1.1.

Com o objetivo de medir o desempenho proporcionado pelo balanceamento do HDFS, realizou-se a execução do *benchmark* TestDFSIO em modo leitura antes e após o uso do balanceador. Tendo em vista que os possíveis benefícios de desempenho de um *cluster* balanceado originam-se ao explorar o paralelismo de operações em múltiplos nodos computacionais (otimizando a utilização dos recursos), os experimentos foram executados exclusivamente no Hadoop em modo de execução totalmente distribuído. Para ilustrar os resultados obtidos, foram escolhidos dois dos C.T. apresentados anteriormente.

A Figura 3a apresenta a porcentagem de uso de cada DN, antes e após a execução do balanceador e com um *threshold* definido em 3, em um cenário sem falhas (C.T. A). Observa-se que a disparidade de carga entre os DNs foi reduzida, com a ocupação aproximando-se do valor considerado ideal para cada DN (10% para este Caso de Teste).



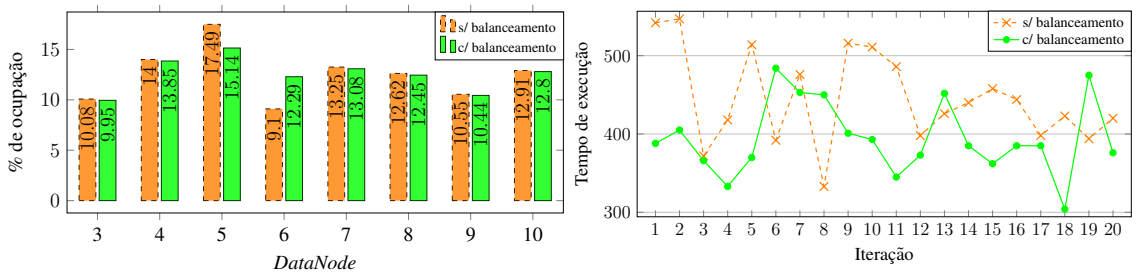
(a) Balanceamento de carga. **(b)** Desempenho obtido.

Figura 3. Uso do balanceador em um cenário sem falhas.

Baseando-se neste cenário, foram medidos os tempos para a conclusão do *benchmark* em 20 operações de leitura distintas. Os resultados alcançados podem ser observados na Figura 3b. A média aritmética dos tempos com o *cluster* em seu estado padrão (sem balanceamento) foi de 501,15 segundos. Após o balanceamento, o valor médio foi reduzido para 411,6 segundos. Isto representa um ganho de aproximadamente 18% no desempenho durante a leitura dos dados armazenados.

Para ilustrar um cenário com falhas, a Figura 4a apresenta a porcentagem de uso de cada DN antes e após o balanceamento, considerando um cenário com duas falhas (C.T. C). Com a execução do balanceador, o *cluster* é equilibrado com um valor que aproxima-se do ideal para cada DN (12.5% para este Caso de Teste).

A Figura 4b exibe os valores obtidos neste teste. Com o balanceador, o tempo médio de execução diminuiu de 445,4 para 394,25 segundos, proporcionando um ganho de desempenho de aproximadamente 11% durante a leitura dos dados armazenados.



(a) Balanceamento de carga. **(b)** Desempenho obtido.

Figura 4. Uso do balanceador em um cenário com falhas.

5. Considerações Finais

Este trabalho analisou o comportamento do HDFS durante a distribuição de blocos replicados entre DNs. Os experimentos atestaram que o desbalanceamento de carga é uma realidade presente no sistema de arquivos. Para resolver este problema, utilizou-se uma ferramenta disponibilizada pelo Hadoop, o HDFS *Balancer*, que oferece uma solução reativa ao desbalanceamento. Com o uso do balanceador, demonstrou-se que melhorias significativas de desempenho podem ser alcançadas mediante o equilíbrio de carga.

Trabalhos futuros envolvem a exploração de abordagens alternativas que eliminem a necessidade de execução de ferramentas remediadoras como o balanceador. Exemplos conhecidos, como [Shwe and Aritsugi 2018, Dai et al. 2016], partem da criação de uma nova política de posicionamento de réplicas para o HDFS, focada em otimizar a distribuição dos blocos replicados com métricas de uso do *cluster*. Já [Lin and Lin 2015] criam o papel de um novo nodo no sistema, dedicado exclusivamente ao balanceamento. Estudos adicionais, compreendendo a execução de novos *benchmarks* e a análise do comportamento do HDFS mediante o aumento de carga (*stress testing*), serão conduzidos.

Referências

Dai, W., Ibrahim, I., and Bassiouni, M. (2016). A new replica placement policy for hadoop distributed file system. In *IEEE Int. Conf. on Big Data Security on Cloud (BigDataSecurity), IEEE Int. Conf. on High Performance and Smart Computing (HPSC), and IEEE Int. Conf. on Intelligent Data and Security (IDS)*, pages 262–267. IEEE.

Foundation, A. (2018). "Replica Placement: The First Baby Steps". `https://hadoop.apache.org/docs/r2.9.1/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html`. Setembro.

Hortonworks (2018). "HDFS Administration". `https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_hdfs-administration/content/ch_balancing-in-hdfs.html`. Outubro.

Lin, C.-Y. and Lin, Y.-C. (2015). A load-balancing algorithm for hadoop distributed file system. In *International Conference on Network-Based Information Systems*, pages 173–179. IEEE.

Shwe, T. and Aritsugi, M. (2018). A data re-replication scheme and its improvement toward proactive approach. *ASEAN Engineering Journal*, 8(1):36–52.

White, T. (2015). *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 4th edition.