Estratégia Reativa e Orientada a Eventos para o Balanceamento de Réplicas em um Cluster HDFS

Rhauani Weber Aita Fazul, Patrícia Pitthan Barcelos
Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS, Brasil
{rwfazul, pitthan}@inf.ufsm.br

***Resumo*—O HDFS Balancer é a solução oficial para o balanceamento de réplicas no sistema de arquivos distribuído do Hadoop (HDFS). Atualmente, é responsabilidade do administrador do cluster monitorar o estado do sistema e, quando julgar necessário, disparar a execução do balanceador. A configuração e disparo sob demanda do HDFS Balancer, entretanto, apresentam limitações e podem comprometer o propósito do processo. Este trabalho apresenta uma estratégia para o balanceamento reativo que toma ações automáticas baseadas no estado de atividade e inatividade dos nodos. Os resultados obtidos demonstraram que a solução é capaz de reagir adequadamente a eventos de adição e redução de nodos, promovendo o balanceamento de réplicas no HDFS.**

***Index Terms*—balanceamento de réplicas, estratégia reativa, replicação de dados, sistemas de arquivos distribuídos**

I. Introdução

O *framework* Apache Hadoop [1] é uma plataforma dedicada à gerência de *big data* em ambientes distribuídos. Um dos principais componentes de seu ecossistema é o *Hadoop Distributed File System* (HDFS): um sistema de arquivos altamente confiável e otimizado para o armazenamento de volumes massivos de dados. O HDFS é incorporado como camada de persistência por várias tecnologias além do Hadoop, tais como Apache Spark, Storm e HBase [2].

Um *cluster* HDFS segue uma arquitetura *master-worker*, em que o servidor mestre, denominado *NameNode* (NN), é responsável por manter o *namespace* do sistema e controlar a distribuição e o acesso aos arquivos. Os dados, por sua vez, são armazenados em *workers* conhecidos como *DataNodes* (DNs), que atendem as operações dos clientes. Os arquivos a serem escritos no sistema são automaticamente segmentados em blocos de dados de tamanho fixo (128MB por padrão). Para assegurar alta confiabilidade e disponibilidade mesmo enquanto executa sobre *commodity hardware*, os blocos são individualmente replicados e armazenados por diferentes DNs [3].

A otimização do posicionamento das réplicas dos blocos é uma característica que distingue o HDFS de outros sistemas de arquivos distribuídos [1]. Todavia, com o decorrer do tempo, a distribuição das réplicas pode se tornar desbalanceada [2], apresentando uma grande discrepância entre os volumes de dados mantidos pelos DNs, o que impacta na localidade dos dados e prejudica o desempenho geral do HDFS. Cabe ao administrador do *cluster* monitorar o estado do sistema e as situações que ocasionam o desbalanceamento e, quando necessário, disparar manualmente a execução do HDFS *Balancer* [4]: solução oficial para o balanceamento de réplicas no HDFS.

Este trabalho apresenta uma estratégia de balanceamento orientada a eventos que remove a necessidade de intervenção manual para configuração e disparo do HDFS *Balancer*. Para tal, nós consideramos dois eventos que contribuem com o desbalanceamento de réplicas no HDFS: a inserção de novos DNs ao *cluster* e a inatividade de DNs em decorrência de falhas. Em ambos os eventos, o HDFS não move automaticamente os dados existentes para equilibrar a distribuição das réplicas. A solução proposta, por sua vez, percebe a ocorrência dos eventos e a mudança de estado dos DNs e dispara a operação de balanceamento sempre que aplicável. Uma investigação experimental foi conduzida visando avaliar a efetividade da estratégia em cenários com adição e redução de nodos.

O artigo está organizado em 6 seções. A Seção II aborda o mecanismo de replicação de dados e o balanceamento de réplicas no HDFS. A Seção III aponta os principais trabalhos relacionados, com foco no HDFS *Balancer*. A Seção IV descreve a solução desenvolvida. A Seção V apresenta os experimentos realizados e detalha os resultados obtidos. Por fim, a Seção VI conclui o artigo e direciona os trabalhos futuros.

II. Replicação de Dados no HDFS

A replicação é o elemento central do modelo de armazenamento no HDFS. Com a replicação, cópias dos dados são mantidas em múltiplos nodos, de forma que um bloco possa ser acessado a partir de qualquer DN que mantenha uma de suas réplicas [2]. A quantidade de réplicas de um bloco é definida pelo Fator de Replicação (FR), que é configurado por arquivo e possui o valor padrão de três. Um FR de n evita a perda de dados ainda que $n-1$ DNs falhem simultaneamente.

Mesmo em cenários com falhas consecutivas, o HDFS deve manter a confiabilidade e a disponibilidade dos dados. Portanto, o NN deve controlar o número de réplicas existentes de cada bloco, garantindo que o FR seja respeitado [3]. Para tal, os processos dos DNs comunicam-se periodicamente com o NN através de mensagens *heartbeat* [1]: um mecanismo de tolerância a falhas que permite detectar falhas operacionais em DNs. Se o NN não receber *heartbeats* de um DN dentro de um período predefinido, esse marca o DN como inativo.

O presente trabalho foi realizado com apoio do CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil.

Como o NN determina o mapeamento de blocos para DNs e rastreia constantemente quais blocos precisam ser replicados, esse dispara a re-replicação dos blocos sub-replicados sempre que necessário, restaurando a conformidade com o FR.

Tanto para a replicação inicial quanto para a re-replicação, o NN deve selecionar os DNs para o armazenamento das réplicas. Para tal, o NN segue uma Política de Posicionamento de Réplicas (PPR) [1], que utiliza *rack awareness* para aumentar a confiabilidade e o desempenho do HDFS de acordo com a estrutura do *cluster*. Considerando um FR padrão, a PPR garante que as réplicas sejam dispostas em três DNs distintos (se o cliente estiver executando em um DN, uma das réplicas é mantida localmente) e que, em um mesmo *rack*, estejam contidos no máximo dois terços das réplicas de um bloco [2]. Assim, mesmo se um *rack* falhar por completo, nenhum bloco será perdido, já que suas réplicas não ficam concentradas em um único *rack*. Além disso, uma maior disponibilidade permite explorar a largura de banda de múltiplos *racks* para atender requisições de entrada/saída (E/S).

A. Balanceamento de Réplicas

Um dos princípios do Hadoop é mover as tarefas de computação para onde as réplicas estão armazenadas e, se não for possível, para os nodos que tenham um caminho de rede mais rápido para os DNs que mantenham os blocos requisitados. Este recurso, conhecido como otimização da localidade de dados [2], aumenta o *throughput* geral do sistema ao processar grandes volumes de dados.

Com a replicação, uma tarefa computacional tende a processar a maioria dos blocos localmente. No entanto, uma distribuição desequilibrada afeta a localidade dos dados, resultando em um número elevado de transferências intra-*rack*, uma vez que tarefas atribuídas a nodos que não mantenham muitas réplicas possivelmente não terão acesso a dados locais. Além de aumentar o consumo da largura de banda do *cluster*, o desequilíbrio pode fazer com que alguns DNs fiquem sobrecarregados, impedindo-os de receber novas réplicas e, assim, reduzindo o paralelismo de leitura [1].

O desbalanceamento possui diferentes causas, nomeadamente [5]: (i) a própria PPR, que não garante uma distribuição equilibrada; (ii) o comportamento da aplicação do cliente que, se executar diretamente em um DN, sempre armazena uma das réplicas no nodo local; (iii) a ocorrência de falhas de DNs, uma vez que DNs inativos resultam em re-replicação, que também baseia-se na PPR; e (iv) a inserção de novos DNs, já que estes irão competir igualmente com os outros DNs ativos do *cluster* para o recebimento de novas réplicas [3]. A Seção III apresenta possíveis estratégias para o balanceamento no HDFS.

III. Trabalhos Relacionados

Uma forma de promover o balanceamento de réplicas no HDFS é agir no momento da escrita inicial dos blocos, fazendo com que a estratégia de posicionamento de réplicas considere o estado de utilização dos DNs. Nesse sentido, [6] propõem uma nova PPR que visa uma distribuição de réplicas totalmente balanceada entre os DNs, enquanto satisfaz as condições impostas pela PPR padrão. A política proposta é otimizada para instâncias HDFS executando em ambientes heterogêneos, uma vez que prioriza a disposição das réplicas com base na capacidade de processamento dos nodos.

Mesmo com soluções proativas, nem sempre é possível impedir o desequilíbrio na distribuição das réplicas. Em certas situações, como a adição de novos nodos no sistema, o rebalanceamento torna-se necessário. Em [7], por exemplo, é introduzido um algoritmo modificado (*LatencyBalancer*) que, além da utilização dos DNs, considera variações na latência de escrita e leitura dos discos de armazenamento dos nodos para a realocação dos dados armazenados no HDFS. Com isso, os DNs que apresentarem menor latência de disco são priorizados para o recebimento de mais blocos.

O Hadoop também disponibiliza uma solução nativa para o balanceamento reativo, o HDFS *Balancer* [4]. O *Balancer* é a ferramenta oficial destinada ao balanceamento de réplicas entre dispositivos de armazenamento no HDFS. A *daemon* do balanceador deve ser disparada sob demanda pelo administrador do *cluster* sempre que esse julgar necessário. A operação do HDFS *Balancer* consiste em levar a utilização de todos os DNs para um intervalo controlado por um *threshold* de balanceamento. Tendo $U_{i,t}$ como a porcentagem de utilização dos dispositivos de armazenamento do DN i do tipo t (e.g., HDD ou SDD) e $U_{\mu,t}$ como a utilização média dos dispositivos do tipo t do *cluster*, cada DN é classificado em [5]:

- superutilizado, se $U_{i,t} > U_{\mu,t} + \mathit{threshold}$;
- acima da média, se $U_{\mu,t} + \mathit{threshold} \geq U_{i,t} > U_{\mu,t}$;
- abaixo da média, se $U_{\mu,t} \geq U_{i,t} \geq U_{\mu,t} - \mathit{threshold}$; ou
- subutilizado, se $U_{\mu,t} - \mathit{threshold} > U_{i,t}$.

Com base na classificação dos DNs, o balanceador opera iterativamente movimentando réplicas de nodos com alta utilização (origem) para nodos com menos dados armazenados (destino) [2], até que o *cluster* não possua mais DNs classificados como superutilizados ou subutilizados. Dessa forma, garante-se que a utilização de cada DN fique dentro de um limite inferior ($U_{\mu,t} - \mathit{threshold}$) e superior ($U_{\mu,t} + \mathit{threshold}$).

IV. Solução Proposta

Uma limitação de uso do HDFS *Balancer* é a necessidade de configuração e disparo manual. Com isso, a tomada de decisão para escolha dos melhores momentos para execução do balanceador é de inteira responsabilidade do administrador do *cluster*. Em trabalhos anteriores, nós detalhamos o desenvolvimento de uma estrutura para a automatização do uso do HDFS *Balancer* e para a customização da política de balanceamento com base em prioridades [8]. Nessa estrutura, a coleta de informações é periódica, realizada por processos responsáveis por monitorar o estado do sistema e, com o passar do tempo, disparar a *daemon* do HDFS *Balancer* (i.e., solução baseada em uma arquitetura *time-triggered*).

Neste trabalho, por sua vez, nós desenvolvemos uma estratégia que remove a condição de disparo manual do HDFS *Balancer*, ao mesmo tempo em que evita o disparo sob tempo. Assim, introduz-se ao HDFS o conceito de um balanceamento de réplicas reativo e orientado a eventos (*event-*

triggered), em que diferentes situações que afetam o comportamento do HDFS são observadas e ocasionam ações adequadas ao contexto do sistema. Dois eventos foram escolhidos para observação: a adição e a redução de nodos no *cluster*.

Ao inserir novos DNs, os dados ficam desequilibrados e o HDFS não move automaticamente as réplicas existentes para balancear a distribuição de dados entre os nodos ativos. O NN simplesmente começa a usar esses DNs para armazenar novas réplicas. Já a redução de DNs é comumente ocasionada por falhas de nodo, o que leva o DN a um estado inativo e faz com que a re-replicação dos blocos seja iniciada. Ainda que esse processo seja realizado de forma transparente pelo HDFS, a re-replicação impacta o balanceamento do *cluster*, uma vez que a PPR não busca uma distribuição equilibrada.

Assim que for observado um evento que afete o número de DNs ativos no sistema, avalia-se a possibilidade de disparar uma execução do HDFS *Balancer*. Embora, em muitos casos, a sobrecarga do procedimento de balanceamento não seja alta, recomenda-se executar o balanceador durante os períodos em que o *cluster* está sendo pouco utilizado [5]. Desse modo, métricas do estado de ocupação e utilização do *cluster* são consultadas a fim de verificar se o HDFS está em um momento adequado ao balanceamento considerando a classificação de nodos baseada na $U_{\mu,t}$ e no *threshold*. O controle interno para essas definições foi realizado através do *framework* Apache ZooKeeper [9]: um projeto *open-source* que fornece serviços de coordenação confiáveis para ambientes distribuídos.

O ZooKeeper atua como um repositório central para a manutenção de informações de configuração. Com seu *namespace* compartilhado, estruturado hierarquicamente em uma árvore de *znodes* [10], é possível organizar as informações de ambiente, dos eventos de adição e redução de DNs observados e das operações de balanceamento realizadas no *cluster*. Assim, no momento em que o equilíbrio de réplicas se fizer necessário (dada a alta discrepância no volume de dados armazenado nos nodos), é possível disparar a *daemon* do HDFS *Balancer* por meio de *watches*: um poderoso mecanismo de notificação que permite aos clientes e sistemas observarem quaisquer mudanças associadas aos *znodes* monitorados, eliminando a necessidade de *polling* [9].

Desse modo, com o balanceamento baseado em eventos, remove-se do administrador a responsabilidade de disparar a execução do HDFS *Balancer* após a inserção de DNs e/ou a ocorrência de falhas em nodos no sistema. Além disso, é possível evitar a intrusividade resultante de um modelo de coleta periódica. A seguir, a Seção V apresenta os experimentos realizados para avaliar a solução de balanceamento proposta.

V. Experimentação

Os experimentos foram realizados na plataforma GRID'5000[1], na qual foram configurados 15 nodos no *cluster grisou* do *site Nancy*. Cada nodo (Dell PowerEdge R630) executou uma distribuição Debian GNU/Linux 10 (*buster*) e possuía as seguintes configurações: 2 processadores Intel Xeon E5-2630 v3 (Haswell, 2.40GHz, 8 cores/CPU), 128GB de memória RAM, 558GB de capacidade de armazenamento HDD (SAS Seagate), uma conexão Ethernet de 1Gbps e quatro conexões de 10Gbps.

Neste ambiente, o Apache Hadoop (versão 2.9.2) foi configurado para operar em modo totalmente distribuído com 1 NN e 15 DNs (um DN por nodo) agrupados por diferentes *racks*. A carga dos dados foi realizada pelo *TestDFSIO* [2] (versão 1.8): um *benchmark* que testa o desempenho do HDFS através da execução de tarefas paralelas de E/S intensiva. Com o *TestDFSIO*, 30 arquivos de 35GB cada e com um FR padrão de 3 réplicas por bloco foram escritos no sistema, totalizando um volume de dados de 3,15TB.

De forma a avaliar os benefícios de realizar o balanceamento de réplicas após a adição e remoção de nodos no HDFS, os cenários de testes compararam o comportamento do sistema considerando uma distribuição de dados baseada inteiramente na PPR padrão (*baseline*) com uma distribuição obtida após o uso do HDFS *Balancer* disparado pela estratégia de balanceamento desenvolvida neste trabalho. Ambos os eventos de adição e redução de DNs no sistema de arquivos foram considerados. A Seção V-A apresenta os resultados obtidos com a adição de novos nodos. A Seção V-B, por sua vez, investiga a redução no número de nodos ativos no HDFS.

A. Adição de Nodos

Para os cenários com adição de DNs, a instância HDFS foi inicializada com 12 DNs (DN_{01} a DN_{12}) dispostos igualitariamente em 4 *racks* (R_1 a R_4). Dessa forma, com a escrita dos arquivos, a utilização média do *cluster* ($U_{\mu,t}$) ficou em, aproximadamente, $44{,}37\%$. Na sequência, 3 novos DNs foram adicionados ao sistema, fazendo com que o *cluster* ficasse com 15 DNs ativos. Considerou-se dois modelos para a inserção desses nodos, sendo eles, a inserção de DNs em um novo *rack* e a inserção dos DNs entre os *racks* já configurados no *cluster*. A seguir, as Seções V-A1 e V-A2 apresentam e discutem os resultados obtidos com os dois modelos de adição de DNs.

1) Inserção de Novo Rack: A Tabela I exibe o estado de ocupação em GB ($O_{i,DISK}$) e a porcentagem de utilização ($U_{i,DISK}$) de cada um dos DNs do *cluster*. Os nodos adicionados após a carga dos dados (DN_{13}, DN_{14} e DN_{15}) foram mapeados para um novo *rack* (R_5). Com a abordagem padrão do HDFS (*baseline*), percebe-se que não há redistribuição dos dados já armazenados para os novos DNs. Para tal, o administrador do *cluster* precisaria configurar e disparar, manualmente, o HDFS *Balancer*. Com a solução proposta nesse trabalho, por sua vez, o balanceamento de réplicas é realizado automaticamente de forma reativa, assim que o evento de inserção de DNs for observado no sistema de arquivos.

Além do desbalanceamento inerente da própria PPR do HDFS, conforme abordado na Seção II-A, a inserção de novos DNs agrava ainda mais o desequilíbrio no volume de dados armazenado pelos nodos através do *cluster*. Isso é evidenciado pelos desvios padrões elevados da ocupação e da utilização dos

[1]Os experimentos apresentados neste trabalho foram conduzidos na plataforma Grid'5000, 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).

Tabela I
ESTADO DO HDFS COM DNS ADICIONADOS NO MESMO RACK.

Rack	DataNode	*baseline*		*solução proposta*	
		$O_{i,DISK}$ (GB)	$U_{i,DISK}$ (%)	$O_{i,DISK}$ (GB)	$U_{i,DISK}$ (%)
R_1	DN_{01}	220,67	43,76	212,61	42,17
	DN_{02}	288,75	57,27	212,15	42,08
	DN_{03}	289,05	57,33	212,45	42,14
R_2	DN_{04}	238,93	47,39	212,03	42,05
	DN_{05}	327,10	64,87	225,88	44,80
	DN_{06}	178,41	35,38	198,04	39,28
R_3	DN_{07}	274,70	54,48	213,92	42,43
	DN_{08}	305,96	60,68	220,85	43,80
	DN_{09}	183,60	36,41	206,90	41,03
R_4	DN_{10}	208,91	41,43	212,57	42,16
	DN_{11}	243,61	48,31	229,12	45,44
	DN_{12}	419,05	83,11	220,85	43,80
R_5	$\mathbf{DN_{13}}$	0	0	215,38	42,72
	$\mathbf{DN_{14}}$	0	0	195,15	38,70
	$\mathbf{DN_{15}}$	0	0	191,12	37,90
Desvio padrão (σ)		125,15	24,82	10,69	2,12

DNs obtidos com a abordagem *baseline*. Já a solução proposta atua de forma corretiva, balanceando a distribuição de blocos tanto nos novos DNs quanto nos DNs que já armazenavam dados no sistema. Dessa forma, garante-se que a utilização de todos os nodos fique dentro de um intervalo controlado pelos limites inferior ($U_{\mu,t} - threshold$) e superior ($U_{\mu,t} + threshold$) de balanceamento. Assim, conforme esperado, considerando a utilização média do *cluster* e o *threshold* padrão aplicado, a utilização dos DNs ($U_{i,DISK}$) com a solução proposta ficou entre $34{,}37\%$ ($44{,}37\% - 10\%$) e $54{,}37\%$ ($44{,}37\% + 10\%$).

Com o balanceamento, permite-se que as aplicações explorem de forma otimizada a localidade dos dados armazenados no HDFS. Para investigar as possíveis melhorias de desempenho, foram realizadas 10 execuções distintas do *benchmark TestDFSIO* voltadas à leitura de todos os arquivos armazenados no HDFS. A média aritmética do tempo necessário (em segundos) para a leitura dos dados – o que equivale ao tempo de execução total da aplicação – foi de $1.202{,}05$ segundos com a *baseline* e de $674{,}44$ segundos com a solução proposta. Considerando a variação percentual dada por $((T_b - T_a)/T_a \times 100)$, em que T_a e T_b equivalem, respectivamente, às médias dos tempos obtidos com a *baseline* e com a solução proposta, a variação foi de $-43{,}89\%$, indicando a redução alcançada pela estratégia implementada neste trabalho em relação à abordagem padrão do HDFS.

Além do tempo de leitura, outras métricas tornam-se passíveis de otimização em um *cluster* balanceado, tais como o *throughput* de leitura e a taxa média de E/S. O *throughput* médio foi de $48{,}95$MB/s com a *baseline* e de $81{,}31$MB/s com a solução proposta, equivalendo a uma variação percentual de $66{,}11\%$. Já a média da taxa de E/S foi de $53{,}89$MB/s com a *baseline* e de $89{,}42$MB/s com a solução proposta. A variação percentual foi de $65{,}93\%$, indicando o aumento alcançado pela estratégia desenvolvida.

Para possibilitar tais otimizações, que são impulsionadas pelo balanceamento de réplicas no *cluster*, a solução proposta deve observar o estado de atividade dos DNs e disparar a execução do HDFS *Balancer* quando necessário. Dessa forma, existe um custo inerente da operação do balanceador que precisa ser considerado. Nesse experimento foram necessárias 3 iterações de balanceamento, as quais movimentaram, ao total, um volume de dados de $641{,}75$GB. A redistribuição desses dados demandou $2.474{,}85$ segundos para ser completada. Embora o HDFS *Balancer* possa operar em segundo plano, sem impactar significativamente as demais aplicações em execução no *cluster*, a operação de balanceamento consome largura de banda para transferências de réplicas entre os nodos de diferentes *racks*, o que pode causar sobrecarga e degradação de desempenho momentânea no sistema de arquivos.

2) Inserção de DNs em Racks Existentes: No experimento desta seção, os 3 novos DNs, adicionados após a carga dos dados, foram distribuídos entre os *racks* já existentes no *cluster*, sendo o DN_{13} no *rack* R_1, o DN_{14} no R_2 e o DN_{15} no R_3. A Tabela II exibe o volume de dados mantido por cada nodo. Assim como no experimento da Seção V-A1, a *baseline* não redistribui as réplicas de forma automática, enquanto a solução proposta dispara a operação do HDFS *Balancer* para transferir os dados dos nodos superutilizados para os nodos subutilizados. O resultado dessas movimentações pode ser visto pela redução dos desvios padrões da ocupação e utilização dos DNs com a solução proposta.

Tabela II
ESTADO DO HDFS COM DNS ADICIONADOS EM MÚLTIPLOS RACKS.

Rack	DataNode	*baseline*		*solução proposta*	
		$O_{i,DISK}$ (GB)	$U_{i,DISK}$ (%)	$O_{i,DISK}$ (GB)	$U_{i,DISK}$ (%)
R_1	DN_{01}	150,42	29,83	173,30	32,98
	DN_{02}	261,03	51,77	213,29	42,30
	DN_{03}	412,96	81,90	259,02	51,37
	$\mathbf{DN_{13}}$	0	0	198,68	39,40
R_2	DN_{04}	180,91	35,88	179,58	37,00
	DN_{05}	429,34	85,15	232,69	46,15
	DN_{06}	172,59	34,23	183,30	36,35
	$\mathbf{DN_{14}}$	0	0	199,86	39,64
R_3	DN_{07}	203,96	40,45	208,25	41,30
	DN_{08}	307,90	61,06	212,15	42,08
	DN_{09}	338,76	67,19	212,66	42,18
	$\mathbf{DN_{15}}$	0	0	205,07	40,67
R_4	DN_{10}	237,60	47,12	225,51	44,72
	DN_{11}	235,21	46,65	235,21	46,65
	DN_{12}	244,02	48,40	244,03	48,40
Desvio padrão (σ)		135,94	26,96	24,12	4,78

Foram realizadas 10 novas execuções do *TestDFSIO* com as duas abordagens. A média dos tempos foi de $1.523{,}68$ segundos com a abordagem *baseline* e de $754{,}69$ segundos com a solução proposta, o que equivale a uma variação percentual de

$-50{,}47\%$. Em relação aos tempos obtidos no cenário com a inserção de nodos em um mesmo *rack*, percebe-se um aumento do tempo de execução médio em ambas as abordagens neste experimento. Isso é justificável pela capacidade do HDFS em utilizar a largura de banda de um *rack* adicional (R_5 no experimento da Seção V-A1) para atender as operações de leitura sobre os dados armazenados no *cluster*.

O *throughput* médio alcançado com a *baseline* e a solução proposta, foi de, respectivamente, $38{,}59$MB/s e $82{,}71$MB/s, o que representa um aumento de $114{,}33\%$. A média da taxa de E/S, por sua vez, foi de $44{,}33$MB/s e $88{,}08$MB/s. Isso resultou em uma variação percentual de $98{,}69\%$, indicando a otimização impulsionada pela solução proposta.

Ao total, $627{,}38$GB de dados foram movimentados em 3 iterações de balanceamento com a solução proposta. A movimentação desses dados demandou $2.063{,}01$ segundos para ser completada. Assim como no cenário da Seção V-A1, a operação do HDFS *Balancer* apresenta um custo elevado por precisar movimentar, no mínimo, o volume de dados necessário para levar a utilização dos novos DNs (subutilizados) para o limite inferior ($U_{\mu,t} - \textit{threshold}$), de forma que eles sejam classificados como abaixo da média.

Tabela III
ESTADO DO HDFS COM DNS FALHANDO EM UM MESMO RACK.

Rack	**DataNode**	***baseline***		***solução proposta***	
		$O_{i,DISK}$ (GB)	$U_{i,DISK}$ (%)	$O_{i,DISK}$ (GB)	$U_{i,DISK}$ (%)
R_1	DN_{01}	233,34	46,28	233,32	46,27
	DN_{02}	236,77	46,96	245,29	48,65
	DN_{03}	237,50	47,10	249,73	49,53
R_2	DN_{04}	307,39	60,96	307,39	60,96
	DN_{05}	327,80	65,01	275,90	54,72
	DN_{06}	209,63	41,58	265,07	52,57
R_3	DN_{07}	307,32	60,95	267,96	53,14
	DN_{08}	280,62	55,66	265,19	52,59
	DN_{09}	381,47	75,66	274,01	54,34
R_4	DN_{10}	220,43	43,72	268,06	53,16
	DN_{11}	215,41	42,72	267,88	53,13
	DN_{12}	217,54	43,14	264,94	52,54
R_5	$\mathbf{DN_{13}}$	×	×	×	×
	$\mathbf{DN_{14}}$	×	×	×	×
	$\mathbf{DN_{15}}$	×	×	×	×
Desvio padrão (σ)		55,32	10,97	18,21	3,61

B. Redução de Nodos

No cenário com redução de DNs, o HDFS foi iniciado com 15 DNs (DN_{01} a DN_{15}) agrupados em 5 *racks* (R_1 a R_5). A utilização média do *cluster* ($U_{\mu,t}$) após a carga dos dados ficou em, aproximadamente, $55{,}40\%$. Em sequência, um comportamento falho foi emulado através da introdução de falhas de *crash* em 3 DNs, fazendo com que o *cluster* ficasse com 12 DNs ativos. As falhas foram induzidas com o comando *kill* do Linux nos processos dos DNs selecionados. Na Seção V-B1, as falhas ocorrem em DNs mantidos em um mesmo *rack*, resultando em falha total de *rack*. Já na Seção V-B2, as falhas ocorrem em DNs de múltiplos *racks*.

1) Falha Total de Rack: A Tabela III mostra o estado do *cluster* após a indução das falhas nos nodos do *rack* R_5 (DN_{13}, DN_{14} e DN_{15}) e como a distribuição das novas réplicas afetou o estado de ocupação e utilização do sistema de arquivos. Nesse cenário, mesmo com a abordagem *baseline*, os blocos originalmente mantidos nos nodos falhos são automaticamente re-replicados para os DNs ativos restantes no *cluster*.

Ao notar as falhas nos DNs pelo não recebimento de *heartbeats*, o NN dispara a criação de novas cópias dos blocos sub-replicados até que a quantidade de réplicas desses blocos volte a ficar em concordância com o FR estipulado. Desse modo, a diferença entre as abordagens reside no balanceamento do *cluster* após a redução de 15 para 12 DNs ativos, situação em que a solução proposta mantém um maior equilíbrio.

A média dos tempos para leitura dos dados em 10 execuções do *TestDFSIO* foi de $1.332{,}71$ segundos com a *baseline* e de $985{,}96$ segundos com a solução proposta. A variação percentual foi de $-26{,}02\%$, representando a redução alcançada. O aumento geral dos tempos de leitura com a solução proposta em relação aos valores obtidos nos cenários com inserção de DNs (Seção V-A) é devido ao número reduzido de DNs ativos (menos paralelismo e largura de banda disponível) decorrente das falhas. Com a *baseline*, por outro lado, os tempos se mantém similares aos cenários com inserção, já que – embora existissem 15 DNs ativos no *cluster* – o HDFS não explorava a largura de banda dos 3 DNs adicionados com essa abordagem, pois eles não possuíam dados armazenados. Dessa forma, assim como o experimento desta seção, as operações de leitura somente operam sobre as réplicas mantidas em 12 nodos.

A média dos valores de *throughput* foi de $41{,}54$MB/s com a abordagem *baseline* e de $50{,}68$MB/s com a solução proposta (variação percentual de 22%). A média das taxas de E/S, por sua vez, foi de $44{,}5$MB/s e de $54{,}97$MB/s, respectivamente (variação percentual de $23{,}53\%$).

O equilíbrio de réplicas proporcionado pela solução proposta movimentou $213{,}13$GB de dados em 3 iterações de balanceamento que demandaram $1.056{,}52$ segundos para serem concluídas. Essa movimentação é ocasionada pelo disparo do HDFS *Balancer* assim que a solução proposta percebe que o número de DNs ativos do *cluster* foi reduzido. Ao realizar o balanceamento neste momento, é possível corrigir o desequilíbrio na distribuição das réplicas originado tanto da replicação inicial (escrita dos arquivos durante a carga dos dados) quanto da re-replicação pós-falhas, uma vez que ambos os procedimentos são baseados na PPR padrão.

2) Falha de DNs em Múltiplos Racks: No experimento desta seção, o *cluster* possui quatro *racks* (R_1 a R_4), cada *rack* contendo, respectivamente, 4, 4, 4 e 3 DNs. As falhas ocorrem em três nodos, cada um pertencendo a um *rack* distinto. A Tabela IV exibe o estado do *cluster* após o procedimento de re-replicação dos dados previamente armazenados nos nodos falhos (DN_{13} em R_1, DN_{14} em R_2 e DN_{15} em R_3) com ambas as abordagens. De forma similar ao experimento da Seção V-B1, a solução proposta mantém a utilização dos DNs em um

intervalo controlado em função do *threshold* de balanceamento aplicado (10%) ao executar o HDFS *Balancer*.

Tabela IV
ESTADO DO HDFS COM DNS FALHANDO EM MÚLTIPLOS RACKS.

Rack	DataNode	*baseline*		*solução proposta*	
		$O_{i,DISK}$ (GB)	$U_{i,DISK}$ (%)	$O_{i,DISK}$ (GB)	$U_{i,DISK}$ (%)
R_1	DN_{01}	226,64	44,95	241,38	47,87
	DN_{02}	237,60	47,12	242,89	48,17
	DN_{03}	234,33	46,47	265,75	52,70
	$\mathbf{DN_{13}}$	×	×	×	×
R_2	DN_{04}	346,07	68,63	279,55	55,44
	DN_{05}	298,70	59,24	298,70	59,24
	DN_{06}	207,24	41,10	265,07	52,57
	$\mathbf{DN_{14}}$	×	×	×	×
R_3	DN_{07}	189,85	37,65	269,04	53,36
	DN_{08}	292,65	58,04	264,31	52,42
	DN_{09}	301,98	59,89	264,18	52,39
	$\mathbf{DN_{15}}$	×	×	×	×
R_4	DN_{10}	219,96	43,62	241,75	47,94
	DN_{11}	403,01	79,93	310,80	61,64
	DN_{12}	216,69	42,97	245,70	48,73
Desvio padrão (σ)		64,24	12,74	22,19	4,40

A média dos tempos de execução em 10 operações do *TestDFSIO* em modo leitura foi de $1.371{,}30$ segundos com a abordagem *baseline* e de $1.069{,}87$ segundos com a solução proposta, o que possibilitou uma redução de $-21{,}98\%$. A média do *throughput* com a *baseline* e com a solução proposta foi de, respectivamente, $42{,}16$MB/s e de $50{,}47$MB/s (aumento de $19{,}71\%$). A taxa de E/S, por sua vez, teve um valor médio de $42{,}50$MB/s com a *baseline* e de $51{,}06$MB/s com a solução proposta, equivalendo a uma variação percentual de $20{,}14\%$.

Neste cenário, 4 iterações de balanceamento foram realizadas, movimentando $223{,}13$GB de dados em $1.278{,}42$ segundos. Embora continue existindo um *trade-off* entre o ganho de desempenho em um *cluster* balanceado e o custo da execução do HDFS *Balancer*, percebe-se que as operações de balanceamento nos experimentos com falhas de DNs foram mais rápidas e movimentaram menos dados quando comparadas às operações nos cenários com inserção de nodos. Isso ocorre pois, no caso de inserção de novos DNs, o balanceador precisa movimentar um maior volume de dados para equilibrar os nodos, já que eles iniciam com utilização zerada. Por outro lado, em cenários com redução de nodos, o *cluster* encontra-se mais equilibrado (observa-se como, mesmo com a abordagem *baseline*, os desvios padrões da ocupação e utilização dos DNs obtidos nos experimentos da Seção V-A são maiores que os da Seção V-B), assim demandando um menor esforço da solução proposta para manter o balanceamento de réplicas no HDFS.

VI. CONSIDERAÇÕES FINAIS

O desequilíbrio na distribuição dos dados é uma condição prejudicial ao HDFS. O HDFS *Balancer* é a solução oficial utilizada para o balanceamento de réplicas. Embora seja uma boa prática executar o balanceador regularmente, cabe ao administrador observar o *cluster* e definir os melhores momentos para o seu disparo. Tendo em vista que diferentes situações podem influenciar o estado de balanceamento do sistema, essa tarefa é propensa a escolhas inadequadas.

Este trabalho apresentou uma solução para otimizar o balanceamento de réplicas no HDFS. A estratégia desenvolvida reage a eventos que alteram a quantidade de nodos ativos e inativos no sistema de arquivos, sendo eles a adição de novos nodos ao *cluster* e a remoção de nodos, que acontece principalmente em decorrência de falhas.

Os resultados obtidos demonstram que a solução proposta é capaz de observar os eventos e, automaticamente, executar o HDFS *Balancer* para rebalancear a distribuição das réplicas no sistema, sem necessidade de intervenção manual por parte do administrador. Nos cenários com adição de nodos, foi possível impulsionar melhorias de desempenho expressivas ao otimizar a localidade de dados com o balanceamento. Já nos cenários com remoção de nodos as otimizações de desempenho ocorreram em uma escala menor, porém o *overhead* causado pela operação do HDFS *Balancer* foi significativamente reduzido.

A estrutura de balanceamento baseada em eventos abre precedentes para observação de diferentes parâmetros que afetem o estado do HDFS. Neste sentido, em trabalhos futuros, pretende-se desenvolver uma solução sensível ao contexto, de forma que o balanceamento deixe de ser um procedimento genérico e passe a considerar os parâmetros fornecidos pelas aplicações em execução no *cluster*. Esta abordagem sensível ao contexto será a base para a consolidação de uma arquitetura dinâmica de balanceamento proativo e reativo para sistemas de arquivos distribuídos que baseiam-se na replicação de dados.

REFERÊNCIAS

[1] Apache Software Foundation. (2021) Apache hadoop. [Online]. Available: https://hadoop.apache.org/docs/r3.3.1/. [Acesso: Junho, 2021].

[2] T. White, *Hadoop: The Definitive Guide*, 4th ed. Sebastopol: O'Reilly Media, Inc., 2015.

[3] G. Turkington, *Hadoop Beginner's Guide*, 1st ed. Birmingham: Packt Publishing Ltd, 2013.

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Symposium on Mass Storage Systems and Technologies*. IEEE, 2010, pp. 1–10.

[5] Cloudera, Inc. (2021) Managing data storage: Balancing data across an hdfs cluster. [Online]. Available: https://docs.cloudera.com/runtime/7.2.7/scaling-namespaces/topics/hdfs-balancing-data-across-hdfs-cluster.html. [Acesso: Junho, 2021].

[6] W. Dai, I. Ibrahim, and M. Bassiouni, "An improved replica placement policy for hadoop distributed file system running on cloud platforms," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing*. IEEE, 2017, pp. 270–275.

[7] J. Dharanipragada, S. Padala, B. Kammili, and V. Kumar, "Tula: A disk latency aware balancing and block placement strategy for hadoop," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 2853–2858.

[8] R. W. A. Fazul and P. P. Barcelos, "Automation and prioritization of replica balancing in hdfs," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 35–38.

[9] S. Haloi, *Apache Zookeeper Essentials*, 1st ed. Birmingham: Packt Publishing Ltd, 2015.

[10] F. Junqueira and B. Reed, *ZooKeeper: Distributed Process Coordination*, 1st ed. O'Reilly Media, Inc., 2013.