O Balanceamento de Réplicas no HDFS frente a aplicações com uso intensivo de E/S, dados e CPU

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*—A replicação de dados é essencial para o sistema de arquivos distribuído do Apache Hadoop (HDFS). Para garantir alta confiabilidade, disponibilidade e desempenho, as réplicas precisam estar armazenadas de forma otimizada pelos nodos do cluster. Entretanto, quando os nodos armazenam quantidades desproporcionais de dados, o funcionamento do sistema é afetado. Visando mitigar os problemas inerentes do desbalanceamento de réplicas, o HDFS Balancer é a solução oficial disponibilizada para a redistribuição dos dados já armazenados no cluster. Neste trabalho, nós avaliamos a efetividade do HDFS Balancer e como o sistema de arquivos explora a localidade dos dados em diferentes situações. Para uma análise aprofundada, aplicações com comportamentos distintos foram consideradas. Os resultados demonstram que o balanceamento de réplicas possibilita otimizações de desempenho significativas no HDFS.**

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

I. Introdução

Sistemas de arquivos distribuídos são essenciais para suportar aplicações que lidam com grandes volumes de dados. Um dos sistemas de arquivos distribuídos mais amplamente utilizados é o *Hadoop Distributed File System* (HDFS) [1]. O HDFS é o principal mecanismo de armazenamento do Apache Hadoop e de seu ecossistema, sendo incorporado por diversas tecnologias e *frameworks* de processamento paralelo, tais como Apache Spark, Storm, Kafka e Hbase [2].

Um cluster HDFS segue uma arquitetura *server-worker*, composta por dois tipos de nodos: *NameNode* (NN) e *DataNode* (DN). O NN é o servidor mestre, que gerencia o *namespace* do sistema e controla o acesso e a distribuição dos arquivos. Os DNs, por sua vez, são os *workers* responsáveis por armazenar os dados e atender às solicitações de escrita e leitura dos clientes. Aplicações compatíveis com o HDFS, em geral, manipulam volumes massivos de dados e necessitam de garantias de confiabilidade, disponibilidade e alto *throughput* para acesso aos dados [3]. Desse modo, a replicação de dados é utilizada como o principal mecanismo de tolerância a falhas e o elemento central do modelo de armazenamento do HDFS.

Otimizar a forma como as réplicas são distribuídas através do *cluster* distingue o HDFS de grande parte dos demais sistemas de arquivos distribuídos [1]. Essa otimização é possibilitada por estratégias eficientes para o posicionamento das réplicas entre os DNs, visando aprimorar a disponibilidade e o desempenho do HDFS em atender operações de entrada/saída (E/S). Entretanto, com o passar do tempo, a distribuição dos dados replicados pode se tornar desbalanceada [2].

A solução nativa para analisar o posicionamento das réplicas já armazenadas e promover o balanceamento de réplicas através da redistribuição dos dados é o HDFS *Balancer* [4]. Além de garantir que o volume de dados mantido por cada DN fique em um intervalo controlado, o processo de balanceamento contribui com o bom funcionamento do HDFS, podendo evitar sobrecargas desnecessárias e otimizar a localidade dos dados mantidos no sistema de arquivos [5].

Este trabalho avalia a efetividade do HDFS *Balancer* e os potenciais ganhos de desempenho obtidos pelo balanceamento de réplicas. A investigação experimental conduzida considerou cenários com múltiplas fases que destacam como a distribuição de dados no HDFS afeta o comportamento do sistema ao longo do tempo. Para permitir uma visão mais ampla das otimizações promovidas pelo HDFS *Balancer* ao redistribuir as réplicas, os experimentos foram conduzidos com diferentes tipos de aplicações frequentemente executadas em *clusters* HDFS.

O trabalho está organizado em seis seções. A Seção II detalha o mecanismo de replicação no HDFS. A Seção III aborda o processo de balanceamento de réplicas e apresenta o HDFS *Balancer*. A Seção IV elenca os principais trabalhos relacionados. A Seção V descreve os cenários de testes e discute os resultados obtidos nos experimentos. Por fim, a Seção VI conclui o artigo e direciona os trabalhos futuros.

II. Replicação de dados no HDFS

O HDFS foi projetado para assegurar alta confiabilidade mesmo enquanto executa em *hardware* de baixo custo [6]. Uma forma comum de garantir alta confiabilidade e disponibilidade em ambientes distribuídos é através da replicação. No HDFS, os arquivos são automaticamente segmentados em blocos de dados de tamanho fixo (128MB por padrão), que são replicados com base em um Fator de Replicação (FR) [2].

O NN é responsável por monitorar ativamente a quantidade de réplicas de cada bloco armazenado no sistema, disparando a re-replicação dos blocos sub-replicados sempre que for preciso restaurar a conformidade com o FR configurado [3]. A seleção dos DNs para manter as réplicas – originadas tanto

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

da replicação inicial dos blocos quanto do processo de re-replicação – é um fator crítico para o funcionamento adequado do HDFS. Para fazer essa seleção, o NN segue uma Política de Posicionamento de Réplicas (PPR) [2].

Para o caso padrão, quando o FR é três, a PPR armazena a primeira réplica no mesmo nodo do cliente (para clientes em execução fora do *cluster* um DN é determinado aleatoriamente). A segunda réplica, por sua vez, é armazenada em um outro DN, escolhido aleatoriamente em um *rack* diferente do da primeira réplica. Já a terceira réplica é armazenada em um DN diferente no mesmo *rack* remoto da segunda réplica.

A estratégia implementada pela PPR garante alta confiabilidade, pois, mesmo que um *rack* inteiro falhe, nenhum bloco de dados será perdido. Além disso, com seu modelo de distribuição *rack-aware*, é possível identificar o *rack* mais próximo do cliente que armazena a réplica solicitada [6]. A Seção III explica a importância do balanceamento de réplicas para o bom funcionamento do sistema de arquivos.

III. Balanceamento de Réplicas

Visando maximar o *throughput* durante as operações de leitura, um dos princípios do Hadoop é mover as tarefas de computação para onde as réplicas são mantidas e, se não for possível, para os nodos que têm um caminho de rede mais rápido para os DNs que armazenam os blocos solicitados. Com isso, evita-se o consumo elevado da largura de banda do *cluster* [1]. Este recurso, conhecido como otimização da localidade de dados [2], aumenta a taxa de transferência global do sistema ao processar grandes conjuntos de dados, além de minimizar a latência de leitura e o congestionamento da rede.

Uma distribuição desequilibrada das réplicas tende a afetar a localidade dos dados armazenados no sistema de arquivos, resultando em um aumento do número de transferências intra-*rack* e *off-rack*, uma vez que as tarefas atribuídas a nodos que não mantenham muitas réplicas possivelmente não terão acesso a dados locais. Além de aumentar o consumo de largura de banda da rede, o desbalanceamento de réplicas pode fazer com que alguns DNs esgotem sua capacidade de armazenamento. Isso impede que o DN receba novos blocos e reduz seu paralelismo de leitura [1].

As principais causas do desbalanceamento de réplicas são [3]: (i) a própria PPR que, em geral, não considera a utilização dos nodos e armazena dois-terços das réplicas no mesmo *rack*, não garantindo uma distribuição balanceada; (ii) o comportamento da aplicação cliente que, se executada diretamente em um DN, armazena uma das réplicas na máquina local para otimizar a operação de escrita, fazendo com que esse DN, em geral, termine com uma maior ocupação; (iii) o procedimento de re-replicação dos blocos, que também segue a PPR e contribui com o desequilíbrio no posicionamento das réplicas; (iv) a ocorrência de falhas de DNs, uma vez que DNs inativos resultam na re-replicação de múltiplos blocos; e (v) a adição de novos DNs ao sistema, uma vez que os blocos existentes não são movidos automaticamente.

Para manter o bom funcionamento do *cluster*, tirar melhor proveito dos recursos computacionais e evitar gargalos de desempenho, é necessário redistribuir as réplicas dos dados até que haja menos discrepância entre os volumes de dados armazenados nos DNs. Para tal, existe uma ferramenta, integrada na distribuição Hadoop, destinada ao balanceamento de réplicas: o HDFS *Balancer* [4]. O balanceador, que deve ser acionado sob demanda pelo administrador do *cluster*, opera iterativamente e é guiado por um *threshold* de balanceamento. Considerando $G_{i,t}$ como o grupo de dispositivos de armazenamento do tipo t de um DN i (por exemplo, disco ou SSD), o *threshold* limita a diferença máxima entre sua utilização ($U_{i,t}$) e a utilização média dos dispositivos de armazenamento do tipo t do *cluster* ($U_{\mu,t}$). Quando a utilização de cada DN estiver dentro desse limite, o *cluster* é considerado balanceado.

IV. Trabalhos Relacionados

Um estudo para determinar se o aumento do fator de replicação dos blocos aprimora o desempenho do HDFS é apresentado em [7]. Com a ajuda de um sistema de replicação adaptável, que incrementa o FR dos dados mais acessados, os autores conseguiram otimizar a disponibilidade dos dados e reduzir os tempos de execução de *jobs* no *cluster*. Em [8], os autores realizaram uma análise teórica de diferentes estratégias para a escrita de blocos de dados entre os DNs. O trabalho descreve as especificações técnicas, recursos e especialização para cada abordagem, juntamente com suas aplicações.

Um algoritmo aprimorado para balancear *racks* sobrecarregados é proposto em [9]. A estratégia atua principalmente no balanceamento de *racks*, reduzindo assim as chances de falha total de *rack* por sobrecarga e contribuindo para uma distribuição de dados mais uniforme no sistema. O trabalho de [10], por sua vez, otimiza o procedimento de redistribuição de réplicas aproveitando-se da capacidade de processamento dos nodos. A solução é especialmente projetada para ambientes heterogêneos e redistribui os blocos apenas para DNs específicos, determinados a partir de uma classificação inicial por sua heterogeneidade e desempenho.

Em trabalhos anteriores [5], nós automatizamos o processo de tomada de decisão para configurar e executar a *daemon* do HDFS *Balancer*. A solução, avaliada através de aplicações com foco em E/S, atua de forma proativa e reativa para manter o equilíbrio das réplicas enquanto redistribui os dados de acordo com a propensão a falhas dos nodos.

De forma geral, até o momento, os trabalhos encontrados na literatura avaliam os benefícios do balanceamento de réplicas a partir de otimizações de desempenho proporcionadas a aplicações voltadas a E/S. Neste trabalho, por sua vez, visamos uma investigação aprofundada considerando o balanceamento frente a diferentes situações e classes de aplicações.

V. Experimentação e Discussão

De forma a investigar o comportamento e avaliar a efetividade do balanceamento de réplicas no HDFS, realizaram-se experimentos na plataforma GRID'5000[1]. O ambiente de

[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).

testes consistiu em 12 nodos configurados no *cluster paravance* do *site Rennes*. Cada nodo (Dell PowerEdge R630) possuía 2 CPUs Intel Xeon E5-2630 v3 (Haswell, 2.40GHz, 8 cores/CPU), 128GB de memória RAM, 558GB de capacidade de armazenamento HDD (SATA Seagate) e 2 conexões Ethernet de 10Gbps cada. Todos os nodos do *cluster* estavam executando uma distribuição Debian GNU/Linux 10 (*buster*).

O *framework* Apache Hadoop (versão 2.9.2) foi configurado em modo de operação totalmente distribuído com 1 NN e 12 DNs (um DN por nodo). Os DNs foram agrupados em quatro *racks* (R_1 a R_4), de forma que cada *rack* manteve 3 DNs. A capacidade de armazenamento total do *cluster* acessível ao HDFS era de, aproximadamente, 5,6TB.

Os cenários de testes foram construídos com base em aplicações com comportamentos distintos que possibilitam avaliar o real impacto do balanceamento de réplicas no HDFS. A Seção V-A apresenta e discute os resultados obtidos com uma aplicação que faz uso intensivo de dados e E/S (*I/O bound*), enquanto a Seção V-B considera uma aplicação que realiza tanto operações focadas em E/S quanto operações com uso de intensivo de CPU (*I/O bound* e *CPU bound*).

A. TestDFSIO

Os experimentos nesta seção tiveram como base um conhecido *benchmark* utilizado para testes de estresse no HDFS, o *TestDFSIO* [2]. Com um comportamento característico *I/O bound*, o *TestDFSIO* é um *benchmark* distribuído que mede o desempenho do HDFS através de operações de E/S intensivas, que são executadas em paralelo no *cluster* por meio do modelo MapReduce. De forma geral, o *TestDFSIO* permite avaliar o estado do sistema de arquivos em termos de E/S.

A Figura 1 exibe o fluxo de execução dos cenários de testes construídos com o *benchmark*. Três cenários, abrangendo múltiplas operações de escrita e leitura, foram idealizados ($C1$, $C2$ e $C3$). O cenário $C1$ é tido como *baseline* para comparação e considera a distribuição de dados padrão do HDFS, isto é, baseada no posicionamento das réplicas feito pela PPR durante a escrita dos arquivos. No cenário C_2, por sua vez, as réplicas são redistribuídas após a carga inicial dos dados. Já o cenário $C3$ realiza o balanceamento após cada operação de escrita.

Os cenários foram divididos em fases ($F1$ a $F6$). A carga inicial dos dados é realizada na fase $F1$ e, na fase $F2$, o balanceamento de réplicas é realizado nos cenários $C2$ e $C3$. Na sequência, na fase $F3$, o desempenho do HDFS é medido através de operações voltadas à leitura dos dados armazenados. Uma nova carga de dados é realizada na fase $F4$ e, na fase $F5$, o balanceamento é novamente conduzido no cenário $C3$ (momento em que o cenário $C3$ se distingue do cenário $C2$). Por fim, na fase $F6$ novas operações de leitura são realizadas sobre os arquivos armazenados no HDFS.

Na fase $F1$ é feita a carga inicial dos dados com o *TestDFSIO* (versão 1.8). Para tal, 25 arquivos de 25GB cada e com um FR padrão de 3 réplicas por bloco foram escritos no HDFS, totalizando 5.000 blocos de dados de 128MB cada (15.000 réplicas). Isso equivale a uma ocupação de 1,85TB e uma utilização média do *cluster* ($U_{\mu,DISK}$) de 33,01%.

Imediatamente após a carga dos dados, na fase $F2$, o HDFS *Balancer* é executado com um *threshold* padrão de 10% nos cenários com balanceamento ($C2$ e $C3$). A operação movimentou 120GB de dados em 5 iterações de balanceamento, que demandaram 5166,85s para serem concluídas. Por serem baseados nas mesmas operações, os cenários $C2$ e $C3$ são observados conjuntamente até o momento em que o cenário $C3$ realiza uma nova operação de balanceamento (fase $F5$), esta que não é realizada no cenário $C2$.

A Tabela I exibe o estado do *cluster* ao final da fase $F2$ nos três cenários avaliados, considerando a ocupação em GB ($O_{i,DISK}$) e a porcentagem de utilização ($U_{i,DISK}$) dos DNs. Percebe-se como a distribuição das réplicas baseada na PPR inicial (cenário $C1$) resulta em uma grande discrepância no volume de dados mantido por cada nodo, a qual é evidenciada pelo desvio padrão (σ) elevado da ocupação e utilização dos DNs. Nos cenários com balanceamento ($C2$ e $C3$), por sua vez, a divergência é controlada pelo HDFS *Balancer*, que esforça-se em manter a utilização dos DNs dentro dos limites inferior e superior, isso é, $U_{\mu,DISK}$ − *threshold* (33,01% − 10%) e $U_{\mu,DISK}$ + *threshold* (33,01% + 10%).

Tabela I
ESTADO DE OCUPAÇÃO E UTILIZAÇÃO DOS DNS AO FINAL DA FASE F2.

Rack	***DataNode***	**Cenário C1**		**Cenários C2 e C3**	
		$O_{i,DISK}$	$U_{i,DISK}$	$O_{i,DISK}$	$U_{i,DISK}$
R_1	DN_{01}	140,88	27,94	144,37	28,63
	DN_{02}	143,71	28,50	140,47	27,86
	DN_{03}	139,53	27,67	147,65	29,28
R_2	DN_{04}	196,80	39,03	143,64	28,49
	DN_{05}	195,89	38,85	145,63	28,88
	DN_{06}	99,99	19,83	193,25	38,33
R_3	DN_{07}	218,94	43,42	194,77	38,63
	DN_{08}	216,22	42,88	121,07	24,01
	DN_{09}	74,88	14,85	167,81	33,28
R_4	DN_{10}	120,93	23,98	141,10	27,98
	DN_{11}	176,87	35,08	196,66	39,00
	DN_{12}	168,05	33,33	161,76	32,08
Desvio padrão (σ)		45,71GB	9,07%	24,88GB	4,93%

Na fase $F3$, o *TestDFSIO* foi utilizado para leitura de todos os arquivos armazenados no sistema de forma a investigar possíveis melhorias de desempenho e otimizações na localidade dos dados promovidas pelo balanceamento. O *benchmark* foi executado 15 vezes em cada cenário, coletando três métricas das operações de E/S, sendo elas: o tempo de leitura (ou seja, o tempo total de execução do *benchmark*), o *throughput* de leitura e a taxa média de E/S.

Em relação ao tempo de leitura, a média aritmética considerando as 15 execuções do *benchmark* foi de 517,35s no cenário $C1$ e de 436,22s nos cenários $C2$ e $C3$. Para avaliar a relação dos valores, considera-se a variação percentual dada por $((T_b - T_a)/T_a) \times 100$, onde T_a e T_b equivalem, respectivamente, ao valor médio da métrica analisada no cenário $C1$ (*baseline*) e no cenário com balanceamento ($C2$ ou $C3$). A variação do tempo de leitura foi de $-15{,}68\%$. A redução dos tempos de leitura representa o ganho de desempenho obtido nos cenários com balanceamento em relação à PPR padrão.

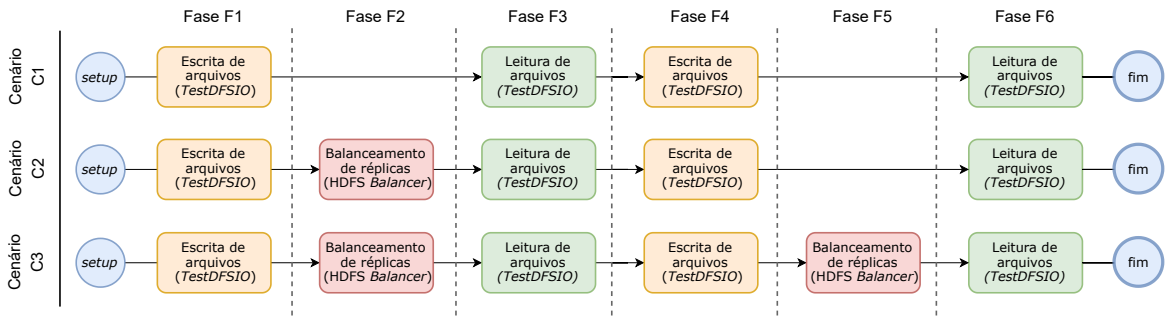

Figura 1. Fluxo de execução dos cenários de testes com o TestDFSIO.

O *throughput* de leitura foi de 84,78MB/s no cenário $C1$ e de 99,75MB/s nos cenários $C2$ e $C3$. Isso equivale a uma variação percentual de 17,66%, indicando o aumento no *throughput*. Já a taxa média de E/S foi de 110,65MB/s no cenário $C1$ e de 130,04MB/s nos cenários $C2$ e $C3$. A variação percentual foi de 17,52%, equivalendo ao aumento na taxa de transferência proporcionado pelo balanceamento de réplicas após a escrita inicial dos dados.

Ao avançar para a fase $F4$, novos dados foram carregados no HDFS. Um volume equivalente ao da fase $F1$ foi escrito no sistema de arquivos com o *TestDFSIO* (25 arquivos de 25GB cada). Com as duas cargas realizadas, 10.000 blocos de dados (30.000 réplicas) estavam armazenadas no sistema, totalizando 3,69TB ($U_{\mu,DISK}$ em 65,95%). Desse ponto em diante, os cenários $C2$ e $C3$ passam a ser avaliados separadamente já que o balanceamento de réplicas é realizado apenas uma única vez no cenário $C2$ (após a primeira carga dos dados), enquanto no cenário $C3$ é realizado duas vezes (após cada operação de escrita). Sendo assim, imediatamente após a segunda operação de escrita, o balanceamento de réplicas é conduzido na fase $F5$ com o cenário $C3$. A operação demandou 5578,19s para movimentar 87,88GB em 6 iterações de balanceamento.

A Tabela II exibe a ocupação e utilização dos DNs ao final da fase $F5$. No cenário $C1$, percebe-se como o desequilíbrio foi agravado ainda mais, o que é demonstrado pelo aumento dos desvios padrões da ocupação e utilização dos DNs quando comparados ao desvios após a primeira carga (Tabela I). No cenário $C2$, o desbalanceamento também foi acentuado, reforçando que uma única operação não foi suficiente para manter o *cluster* em um estado balanceado após novos arquivos serem escritos. O cenário $C3$, por sua vez, realiza uma nova operação de balanceamento, fazendo com que a utilização dos DNs volte a ficar em concordância com o *threshold*.

Em seguida, na fase $F6$, 15 novas operações de leitura foram realizadas com o *TestDFSIO*. Os arquivos lidos foram os escritos pelo *benchmark* na fase $F4$. No cenário $C1$, a média dos tempos de leitura foi de 1093,42s. No cenário $C2$, esse tempo foi reduzido para 583,32s, equivalendo a uma variação percentual de $-46{,}65\%$ em relação ao cenário $C1$. Enquanto isso, no cenário $C3$, o tempo de leitura dos arquivos foi de 426,97s, correspondendo a uma redução de $-60{,}95\%$ quando comparado à *baseline*.

Tabela II
ESTADO DE OCUPAÇÃO E UTILIZAÇÃO DOS DNS AO FINAL DA FASE F5.

Rack	***DataNode***	**Cenário C1**		**Cenário C2**		**Cenário C3**	
		$O_{i,DISK}$	$U_{i,DISK}$	$O_{i,DISK}$	$U_{i,DISK}$	$O_{i,DISK}$	$U_{i,DISK}$
R_1	DN_{01}	412,18	81,75	338,02	67,04	338,02	67,04
	DN_{02}	238,90	47,38	326,71	64,80	326,39	64,73
	DN_{03}	234,70	46,55	271,33	53,81	282,77	56,08
R_2	DN_{04}	291,08	57,73	312,56	61,99	312,56	61,99
	DN_{05}	410,28	81,37	322,79	64,02	322,77	64,01
	DN_{06}	316,84	62,84	310,18	61,52	310,17	61,51
R_3	DN_{07}	373,16	74,01	366,19	72,62	335,74	66,59
	DN_{08}	370,01	73,38	242,39	48,07	272,63	54,07
	DN_{09}	230,80	45,77	301,47	59,79	300,84	59,67
R_4	DN_{10}	297,94	59,09	290,55	57,62	315,71	62,61
	DN_{11}	282,06	55,94	414,32	82,17	355,77	70,56
	DN_{12}	322,77	64,01	285,60	56,64	310,42	61,57
Desvio padrão (σ)		64,86GB	12,86%	44,79GB	8,88%	23,15GB	4,59%

O *throughput* médio de leitura foi de 70,55MB/s no cenário $C1$. Nos cenários $C2$ e $C3$, houve um aumento para 105,51MB/s (variação percentual de 49,55%) e 123,53MB/s (75,1%), respectivamente. Já a taxa média de E/S foi de 106,52MB/s no cenário $C1$. Nos cenários $C2$ e $C3$ houve um aumento da taxa de transferência para, respectivamente, 153,72MB/s (44,31%) e 162,68MB/s (52,72%).

Com base nos resultados, percebe-se como o desbalanceamento inerente do cenário $C1$ acarretou em uma degradação de desempenho significativa na fase $F6$ em relação às operações de leitura realizadas na fase $F3$. Em contraste, nos cenários com balanceamento, o desempenho alcançado para a leitura dos arquivos na fase $F6$ foi mantido em níveis similares ao da fase $F3$. Além disso, percebe-se como a localidade dos dados pode ser melhor explorada em *clusters* balanceados, sendo que o cenário $C3$, com duas operações de balanceamento, foi o cenário que proporcionou maiores otimizações de desempenho. Ressalta-se, entretanto, que a única operação de balanceamento efetuada no cenário $C2$ na fase $F2$ manteve ganhos interessantes mesmo após novas réplicas serem escritas e distribuídas através do sistema de arquivos.

B. TeraSort

Os experimentos desta seção foram realizados com um conjunto de *benchmarks* amplamente utilizado no ecossistema

do Hadoop, o *TeraSort* [2]. Esse é empregado para testes de estresse das camadas do HDFS e do *framework* MapReduce e consiste em três componentes: *TeraGen*, *TeraSort* e *TeraValidate*. Inicialmente, dados aleatórios, no formato de linhas de dados binários de 100 bytes, são gerados com o *TeraGen*. Com base na saída do *TeraGen*, o *TeraSort* ordena os dados por meio de um programa MapReduce. Por fim, o *TeraValidate* valida se a saída do *TeraSort* está corretamente ordenada.

O *TeraSort* mostra-se como uma alternativa interessante para avaliar o balanceamento de réplicas em *clusters* HDFS pelo seu comportamento misto, uma vez que a operação do *benchmark* tem uso predominante de CPU durante a fase mapeamento e foco em E/S durante a fase de redução. Naturalmente, os ganhos promovidos pelo balanceamento são revelados, em maior parte, durante as operações voltadas a E/S. A Figura 2 exibe os cenários de testes construídos com base no *benchmark TeraSort*. Assim como na experimentação realizada com o *benchmark TestDFSIO*, o cenário $C1$ consiste na distribuição de dados seguindo a PPR inicial (i.e., sem balanceamento), enquanto os cenários $C2$ e $C3$ fazem uso do HDFS *Balancer* após os dados serem gerados pelo *TeraGen*.

O experimento foi divido em 6 fases, sendo que na fase $F1$ o conjunto de trabalho inicial é gerado com o *TeraGen* e, na fase $F2$, o HDFS *Balancer* é executado nos cenários com balanceamento ($C1$ e $C2$). Na fase $F3$ o *TeraSort* e o *TeraValidate* são executados em sequência. Uma nova carga é gerada na fase $F4$, que é balanceada na fase $F5$ no cenário $C3$. Por fim, novas operações destinadas à ordenação e à validação dos dados são realizadas na fase $F6$.

Inicialmente, na fase $F1$, o *TeraGen* foi utilizado para gerar os dados. De forma a resultar em um volume similar ao experimento realizado com o *TestDFSIO*, 6.709.254.709 de linhas de 100 bytes cada foram geradas. Esse valor equivale a 5.000 blocos de dados de 128MB cada e, dado o FR padrão, 15.000 réplicas contidas em 6 arquivos. Um volume de dados de, aproximadamente, 1,85TB foi armazenado no sistema de arquivos ao total ($U_{\mu,DISK}$ em 32,95%).

Seguindo para a fase $F2$, o HDFS *Balancer* foi executado nos cenários $C2$ e $C3$ com o *threshold* padrão de 10%. A operação foi realizada em 4 iterações que movimentaram 80,03GB de dados em 4135,72s. A III exibe a ocupação em GB ($O_{i,DISK}$) e a porcentagem de utilização ($U_{i,DISK}$) dos DNs ao final da fase $F2$ no cenário $C1$ (sem balanceamento) e nos cenários $C2$ e $C3$ (com balanceamento).

Mesmo com menos arquivos sendo gerados pelo *TeraGen* (6 arquivos com 834 blocos cada) em relação ao *TestDFSIO* (25 arquivos com 200 blocos cada), o estado de ocupação do *cluster* após a carga inicial dos dados foi similar com ambos os *benchmarks*. Nesse sentido, o cenário $C1$ apresentou o maior nível de desbalanceamento após a carga dos dados com o *TeraGen*, o que é evidenciado pelo desvio padrão e pela diferença máxima do volume de dados mantidos nos DNs, em específico no DN_{01} (mais de 200GB) e no DN_{08} (menos de 90GB). Isso atesta que a PPR não considera a utilização dos nodos ao selecionar os DNs para armazenar as réplicas e, portanto, não garante uma distribuição balanceada dos dados.

Tabela III
ESTADO DE OCUPAÇÃO E UTILIZAÇÃO DOS DNS AO FINAL DA FASE F2.

Rack	***DataNode***	**Cenário C1**		**Cenários C2 e C3**	
		$O_{i,DISK}$	$U_{i,DISK}$	$O_{i,DISK}$	$U_{i,DISK}$
R_1	DN_{01}	219,98	43,63	142,88	28,34
	DN_{02}	126,11	25,01	144,64	28,69
	DN_{03}	126,27	25,04	139,24	27,61
R_2	DN_{04}	175,65	34,84	169,96	33,71
	DN_{05}	180,19	35,74	185,98	36,88
	DN_{06}	180,97	35,89	121,34	24,06
R_3	DN_{07}	191,38	37,96	179,16	35,53
	DN_{08}	89,32	17,71	197,30	39,13
	DN_{09}	188,37	37,36	118,93	23,59
R_4	DN_{10}	137,08	27,19	179,41	35,58
	DN_{11}	133,56	26,49	188,01	37,29
	DN_{12}	140,88	27,94	127,93	25,37
Desvio padrão (σ)		37,26GB	7,39%	28,30GB	5,61%

Em contraste, nos cenários $C2$ e $C3$, conforme esperado, a utilização dos DNs foi mantida dentro do intervalo controlado em função da $U_{\mu,DISK}$ e do *threshold* de balanceamento.

Para avaliar o desempenho do sistema, o *TeraSort* e o *TeraValidate* foram executados 15 vezes em sequência na fase $F3$. A métrica de interesse avaliada com os *benchmarks* é o tempo de execução. A média aritmética dos tempos de execução do *TeraSort* no cenário $C1$ foi de 242,13s. Nos cenários $C2$ e $C3$ a média foi de 228,73s, o que equivale a uma variação percentual de $-5{,}53\%$. Em relação ao *TeraValidate*, a validação foi realizada com sucesso após cada operação de ordenação. Os tempos para a validação, entretanto, não sofreram variações significativas entre os cenários, sendo de 19,33s no cenário $C1$ e de 19,53s nos cenários $C2$ e $C3$.

Na sequência, novos dados foram carregados no sistema de arquivos com o *TeraGen* na fase $F4$. O número de linhas escritas se manteve o mesmo da carga inicial da fase $F1$. A ocupação total do *cluster* após a segunda carga foi de 3,69TB, levando a $U_{\mu,DISK}$ para, aproximadamente, $65{,}91\%$. Após, na fase $F5$, uma nova operação de balanceamento foi conduzida no cenário $C3$. O HDFS *Balancer* foi executado com o *threshold* padrão e movimentou 110,02GB de dados em 5 iterações que demandaram 5167,96s para serem concluídas.

A Tabela IV exibe as ocupações e utilizações dos DNs ao final da fase $F5$. Com os novos dados sendo distribuídos com base na PPR, o desbalanceamento foi acentuado no cenário $C1$. O mesmo ocorre no cenário $C2$, em que a única operação de balanceamento realizada na fase $F2$ não foi capaz de manter o *cluster* equilibrado após a nova carga realizada na fase $F4$, o que é demonstrado pelo aumento dos desvios padrões. Já no cenário $C3$, a nova operação de balanceamento foi capaz de reduzir significativamente a discrepância no volume de dados mantido por cada nodo e fazer com que a utilização dos DNs ficasse em concordância com o *threshold*.

Ao fim do fluxo, na fase $F6$, 15 novas operações distintas do *TeraSort* e do *TeraGen* foram realizadas em sequência. No cenário $C1$, o tempo médio para a ordenação dos dados foi de 307,6s. No cenário $C2$ esse tempo foi reduzido para 285,8s (variação percentual de $-7{,}09\%$). Já no cenário $C3$, a

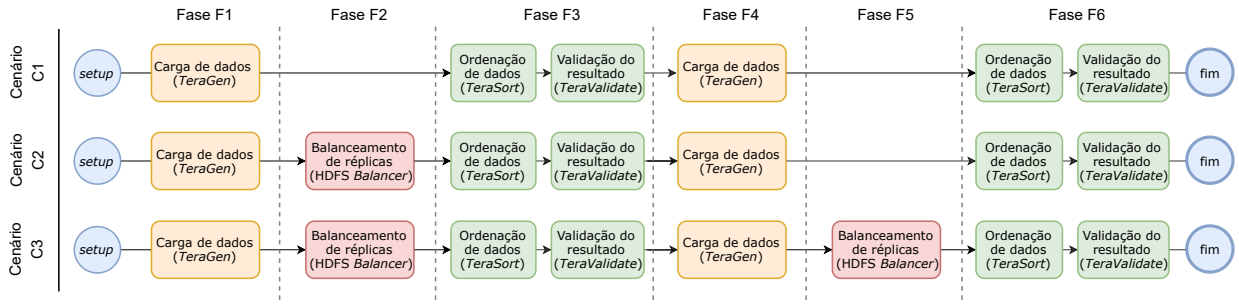

Figura 2. Fluxo de execução dos cenários de testes com o TeraSort.

Tabela IV
ESTADO DE OCUPAÇÃO E UTILIZAÇÃO DOS DNS AO FINAL DA FASE F5.

Rack	*DataNode*	Cenário C1		Cenário C2		Cenário C3	
		$O_{i,DISK}$	$U_{i,DISK}$	$O_{i,DISK}$	$U_{i,DISK}$	$O_{i,DISK}$	$U_{i,DISK}$
R_1	DN_{01}	364,89	72,37	241,66	47,93	271,98	53,94
	DN_{02}	270,52	53,65	339,57	67,35	329,54	65,36
	DN_{03}	266,03	52,76	336,24	66,69	316,01	62,67
R_2	DN_{04}	273,75	54,29	267,93	53,14	298,12	59,12
	DN_{05}	373,20	74,02	382,95	75,95	352,60	69,93
	DN_{06}	377,76	74,92	317,71	63,01	317,63	62,99
R_3	DN_{07}	386,99	76,75	297,42	58,99	297,37	58,98
	DN_{08}	285,61	56,64	314,75	62,42	314,79	62,43
	DN_{09}	283,04	56,13	329,95	65,44	329,89	65,43
R_4	DN_{10}	332,79	66,00	303,35	60,16	314,51	62,38
	DN_{11}	229,61	45,54	403,83	80,09	353,43	70,09
	DN_{12}	335,77	66,59	243,90	48,37	284,08	56,34
Desvio padrão (σ)		53,09GB	10,53%	49,44GB	9,81%	24,74GB	4,91%

nova operação de balanceamento possibilitou reduzir a média do tempo de execução do *TeraSort* para 242,53s (variação percentual de $-21{,}15\%$). O *TeraValidate*, por sua vez, manteve tempos similares independente do cenário (18,13s, 18,73s e 18,07s, nos cenários $C1$, $C2$ e $C3$, respectivamente). Sendo assim, com base nos resultados obtidos com o *TeraSort*, percebe-se que o balanceamento de réplicas promoveu melhorias de desempenho relevantes, embora em escalas menores quando comparadas com *benchmarks* focados exclusivamente em E/S, tal como o *TestDFSIO*.

VI. CONSIDERAÇÕES FINAIS

Otimizar o posicionamento das réplicas é um fator fundamental para o funcionamento do HDFS. O HDFS *Balancer* é a solução oficial para equilibrar o volume de dados mantidos nos nodos, redistribuindo as réplicas entre seus dispositivos de armazenamento. Este trabalho avaliou o balanceamento de réplicas e a efetividade do uso do HDFS *Balancer* sob diferentes perspectivas. Para a investigação experimental, os cenários de testes construídos consideraram múltiplas situações que influenciam o estado de balanceamento e o desempenho do sistema em servir aplicações com comportamentos distintos.

Os experimentos demostraram que, a medida que o desbalanceamento se intensifica, a localidade dos dados é afetada e os recursos deixam de ser explorados de forma otimizada pelas aplicações, sejam elas focadas inteiramente em E/S ou com operações voltadas tanto para dados quanto para CPU. De forma geral, as maiores otimizações foram observadas nas aplicações com uso intensivo de E/S, ainda que aplicações com comportamento misto – mesmo que em proporções menores – também sejam beneficiadas pelo uso do HDFS *Balancer*. Em relação à conservação do balanceamento no *cluster*, percebeu-se que é possível manter determinado nível de equilíbrio mesmo após novas cargas de dados serem realizadas no sistema, embora os maiores ganhos de desempenho tenham sido alcançados em cenários onde o balanceamento é conduzido frequentemente após a escrita de novos arquivos.

Trabalhos futuros envolvem uma investigação aprofundada do comportamento do HDFS *Balancer* considerando *clusters* HDFS executando sobre ambientes heterogêneos e com ocorrência de falhas em nodos do sistema de arquivos. Além disso, as observações constatadas nos experimentos conduzidos neste trabalho serão incorporadas a uma arquitetura dinâmica para o balanceamento de réplicas no HDFS, capaz de adaptar configurações e parâmetros de operação do HDFS *Balancer* conforme o comportamento das aplicações executadas no *cluster* ao longo do tempo.

REFERÊNCIAS

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

[6] S. Achari, *Hadoop Essentials*, 1st ed. Packt Publishing Ltd, 2015.

[7] H. E. Ciritoglu *et al.*, "Investigation of replication factor for performance enhancement in the hadoop distributed file system," in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, 2018, pp. 135–140.

[8] E. S. Abead, M. H. Khafagy, and F. A. Omara, "A comparative study of hdfs replication approaches," *International Journal in IT and Engineering (IJITE)*, vol. 3, pp. 5–11, 2015.

[9] K. Liu, G. Xu, and J. Yuan, "An improved hadoop data load balancing algorithm," *Journal of Networks*, vol. 8, no. 12, pp. 2816–2822, 2013.

[10] A. Shah and M. Padole, "Load balancing through block rearrangement policy for hadoop heterogeneous cluster," in *2018 Int. Conference on Advances in Computing, Communications and Informatics (ICACCI)*. Bangalore: IEEE, 2018, pp. 230–236.