Combinando Abordagens de Balanceamento de Réplicas Proativo e Reativo no HDFS

Rhauani Weber Aita Fazul[1], **Odorico Machado Mendizabal**[1], **Patrícia Pitthan Barcelos**[2]

[1]Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brasil

[2]Laboratório de Sistemas de Computação (LSC)
Universidade Federal de Santa Maria (UFSM) – Santa Maria – RS – Brasil

rhauani.fazul@posgrad.ufsc.br, odorico.mendizabal@ufsc.br,
pitthan@inf.ufsm.br

Abstract. *As new data is loaded into the system, it is common for the distribution of the replicas among the nodes to become unbalanced. HDFS Balancer is the standard solution for data balancing by rearranging the replicas already stored in the cluster. However, its current balancing operation has manual dependency and does not consider the specific needs of applications in the cluster. To address such limitations, this work exploits a balancing solution that combines proactive and reactive approaches, acting both in the pre-operational stage and during the execution of the HDFS Balancer. The evaluation results demonstrate that the solution improves performance through replica rearrangement while considering reliability and availability attributes.*

Resumo. *À medida que novos dados são armazenados no HDFS, é comum que a distribuição das réplicas entre os nodos fique desequilibrada. O HDFS Balancer é a solução padrão para o balanceamento reativo no sistema de arquivos. No entanto, sua política de operação atual possui dependência manual e não considera as necessidades específicas das aplicações no cluster. Para endereçar tais limitações, este trabalho faz uso de solução de balanceamento que combina uma abordagem proativa e reativa, atuando tanto no momento pré-operacional quanto durante a execução do HDFS Balancer. Os resultados demonstram que a solução é capaz de utilizar do balanceamento para fornecer melhorias de desempenho enquanto considera atributos de confiabilidade e disponibilidade.*

1. Introdução

Conforme a produção de dados se intensifica, aumenta também a necessidade de sistemas computacionais capazes de armazenar e manipular tais dados de forma confiável e eficiente. Nesse contexto, são adotadas estratégias para a distribuição de dados e tarefas computacionais em ambientes distribuídos. Diversas ferramentas fazem uso das arquiteturas distribuídas para criar plataformas dedicadas ao processamento paralelo de alto desempenho e ao armazenamento eficiente de *big data*. O Apache Hadoop [Foundation 2022] é um exemplo proeminente desse segmento.

O Hadoop é uma plataforma *open source* que fornece um ecossistema consolidado de soluções baseadas em computação paralela e distribuída. Um de seus principais componentes é o *Hadoop Distributed File System* (HDFS) [Foundation 2022], um sistema de

arquivos distribuído, altamente escalável e tolerante a falhas, projetado para armazenar grandes volumes de dados de forma confiável mesmo quando executa sobre *commodity hardware*. Por oferecer um robusto sistema de gerenciamento de dados para *clusters* e *grids*, o HDFS é frequentemente empregado como motor de armazenamento por diversos *frameworks* de processamento paralelo [White 2015], tais como Apache Spark, Storm e Flink, além de ser compatível com Apache Kafka e HBase.

Um *cluster* HDFS implementa uma arquitetura *server-worker* composta por dois tipos de nós: *NameNode* (NN) e *DataNode* (DN) [Foundation 2022]. O NN é o servidor mestre que gerencia o *namespace* e os metadados do sistema, mantém a árvore de diretórios e controla o acesso e a distribuição dos arquivos. Os DNs são os *workers* que realizam o armazenamento e a recuperação dos dados. Múltiplas instâncias de DNs possibilitam que a distribuição de dados ocorra em diferentes máquinas do *cluster*.

O HDFS utiliza diversas estratégias para assegurar alta confiabilidade e disponibilidade, sendo a replicação de dados o principal mecanismo de tolerância a falhas (TF) implementado [White 2015]. A replicação garante confiabilidade e resiliência dos dados por meio da redundância. Além disso, o uso de réplicas também aumenta o desempenho do sistema, permitindo que as aplicações explorem a localidade dos dados de maneira otimizada. Nesse sentido, manter uma distribuição equilibrada das réplicas entre os nodos se torna fundamental, pois o posicionamento das réplicas influencia diretamente na confiabilidade e no desempenho do HDFS [Foundation 2022].

O HDFS *Balancer* [Shvachko et al. 2010] é a ferramenta oficial para realizar o balanceamento reativo no HDFS. No entanto, existem limitações que afetam sua flexibilidade e eficácia. A política de operação adotada não considera as características específicas do *cluster* e suas aplicações durante a redistribuição de réplicas, tornando o balanceamento menos eficiente e adaptável. Além disso, seu modelo de operação sob demanda depende de intervenção manual, uma vez que a configuração dos atributos de balanceamento e a escolha do momento para execução do balanceador ficam a critério do administrador, podendo afetar a confiabilidade e resultar em gargalos de desempenho.

Após entender as limitações, dificuldades e desafios em rearranjar réplicas no HDFS, em trabalhos anteriores, foi desenvolvido um conjunto de prioridades para personalizar a política de balanceamento padrão do HDFS *Balancer* [Fazul and Barcelos 2019, Fazul and Barcelos 2023] e de uma estrutura para automatizar a configuração e disparo do processo de balanceamento [Fazul and Barcelos 2020, Fazul and Barcelos 2021]. Neste trabalho, avalia-se o comportamento e a eficácia em unir as duas soluções desenvolvidas, dando origem a uma nova solução, caracterizada como proativa e reativa, para o balanceamento de réplicas no HDFS. Para tanto, idealizaram-se experimentos a fim de incorporar as modificações no fluxo de operação do HDFS *Balancer*, comparando os resultados obtidos com a abordagem padrão empregada pelo HDFS.

O trabalho está organizado em seis seções. A Seção 2 detalha o mecanismo de replicação de dados e o processo de balanceamento de réplicas no HDFS. A Seção 3 descreve os principais trabalhos relacionados. A Seção 4 apresenta a solução de balanceamento desenvolvida. A Seção 5 descreve os cenários de testes e os resultados obtidos nos experimentos. Por fim, a Seção 6 aponta as considerações finais.

2. Replicação de Dados no HDFS

O HDFS foi projetado para armazenar dados na escala de petabytes [Shvachko et al. 2010]. Para manipular grandes volumes de dados e suportar arquivos grandes, o HDFS implementa uma estrutura de armazenamento baseada em blocos. Os arquivos inseridos no sistema são automaticamente segmentados em blocos de tamanho fixo (128MB por padrão) e armazenados como unidades independentes. Ao executar sobre equipamento comum, as chances de falhas de DN em um *cluster* HDFS são altas [Turkington 2013]. Devido a essa condição, é necessário garantir alta disponibilidade e confiabilidade, para que os serviços permaneçam operacionais e nenhum dado seja definitivamente perdido.

A replicação é uma estratégia amplamente utilizada em sistemas distribuídos para garantir alta confiabilidade e disponibilidade de dados, e não é diferente no HDFS. Com a replicação, as cópias dos blocos são mantidas em vários nodos do *cluster*, permitindo o acesso aos blocos por qualquer DN que mantenha suas réplicas. Em caso de falhas, os blocos redundantes garantem que cópias dos dados estejam disponíveis [White 2015]. O número de réplicas é definido pelo Fator de Replicação (FR), que é configurável e tem um valor padrão de três réplicas por bloco [Achari 2015].

Além da replicação inicial dos blocos, realizada durante a escrita dos arquivos no sistema, o monitoramento ativo é um requisito vital para garantir resiliência e TF no HDFS. Mesmo em cenários com falhas consecutivas, o HDFS deve manter a confiabilidade e a disponibilidade dos dados. O NN monitora constantemente o estado e a quantidade de réplicas existentes de cada bloco, garantindo que o FR seja respeitado [Turkington 2013]. O NN toma as decisões relacionadas à replicação dos blocos, determinando como os blocos são organizados no sistema, selecionando os DNs para armazenar cada uma de suas réplicas. Essa seleção é feita durante a escrita de arquivos, bem como durante a re-replicação e a redistribuição de blocos já armazenados.

Um bom posicionamento de réplicas aproveita a estrutura de rede do *cluster* para melhorar a disponibilidade e confiabilidade dos dados, reduzir o consumo de largura de banda em operações de escrita e aumentar o desempenho de operações de leitura [White 2015]. Para distribuir dados em um *cluster* HDFS, o NN segue uma Política de Posicionamento de Réplicas (PPR) [Foundation 2022], que usa o conceito de *rack awareness* [Achari 2015] para melhorar a TF e o desempenho de acordo com a topologia do *cluster*. Por padrão, a PPR aloca as réplicas em três DNs distintos (se o cliente estiver em um DN, uma das réplicas será mantida na máquina local) e que um determinado *rack* não contenha mais do que dois terços das réplicas de um mesmo bloco [White 2015].

Em geral, a PPR padrão oferece bons níveis de confiabilidade, largura de banda de escrita e desempenho de leitura [White 2015]. A TF é endereçada colocando réplicas de um mesmo bloco em pelo menos dois *racks* diferentes, evitando a perda de dados se um *rack* inteiro falhar. O desempenho é otimizado reduzindo o consumo de largura de banda global e a latência de leitura, pois DNs de vários *racks* são candidatos para atender a solicitações de E/S [Turkington 2013].

2.1. Localidade dos Dados e Balanceamento de Réplicas

Para maximizar a vazão durante as operações de leitura, o Hadoop utiliza o princípio de mover tarefas computacionais para onde as réplicas de bloco estão armazenadas

[Foundation 2022]. Caso não seja possível, as tarefas são movidas para os nós que possuem um caminho de rede mais curto para os DNs que mantêm os dados necessários para a operação. Esse recurso, conhecido como otimização da localidade de dados [White 2015], melhora o desempenho de processamento de grandes conjuntos de dados, minimizando o consumo de largura de banda global, congestionamento de rede e latência de leitura, uma vez que o acesso aos dados é local. Com a replicação, as chances de que uma tarefa computacional consiga processar grande parte dos blocos localmente é alta. No entanto, em certas situações, o *cluster* pode ficar desbalanceado com uma grande discrepância no volume de dados armazenados em diferentes DNs [Turkington 2013]. Embora a distribuição das réplicas, seguindo as premissas PPR, garanta um balanceamento mínimo (réplicas de um mesmo bloco não são armazenadas no mesmo DN), ela não é, de fato, equilibrada e contribui com o desbalanceamento *inter-rack* [Shvachko et al. 2010].

O desbalanceamento de réplicas pode afetar a localidade dos dados, resultando em um aumento no número de transferências *intra-rack* e *off-rack*, já que tarefas atribuídas a um nodo que não mantenha muitas réplicas possivelmente não terão acesso a dados locais. Isso pode aumentar o consumo geral de largura de banda no *cluster* e sobrecarregar os DNs altamente utilizados [White 2015]. À medida que o armazenamento de alguns nodos se esgota, eles são impedidos de receber novos blocos de dados, reduzindo seu paralelismo de leitura e levando à degradação do desempenho. Portanto, o HDFS funciona melhor quando os blocos de dados são uniformemente distribuídos entre os DNs do *cluster*.

Os dados armazenados no HDFS podem ficar desbalanceados por diferentes razões [Cloudera, Inc. 2021], incluindo as premissas adotadas pelo algoritmo de seleção de DNs da PPR, o procedimento de re-replicação, a ocorrência de falhas de DN, o comportamento da aplicação cliente e a adição de novos DNs ao sistema. Para mitigar os problemas decorrentes do desequilíbrio de réplicas e manter a integridade do *cluster*, soluções de balanceamento são necessárias. Abordagens voltadas ao balanceamento de réplicas, que estão alinhadas com o contexto deste trabalho, são apresentadas na Seção 3.

3. Trabalhos Relacionados

De uma maneira geral, duas abordagens principais podem ser usadas para mitigar os problemas do desequilíbrio de réplicas no HDFS: proativas ou reativas. As abordagens proativas agem antes que ocorra o desbalanceamento, esforçando-se para preservar o equilíbrio de dados no *cluster*. As abordagens reativas, por sua vez, atuam como uma estratégia corretiva que permite tornar o posicionamento dos dados entre os nodos mais homogêneo através da redistribuição ou rebalanceamento de réplicas.

Em relação às abordagens proativas, muitas das soluções complementares encontradas na literatura envolvem a criação de novas políticas de posicionamento de réplicas para o HDFS, assim agindo no momento da distribuição inicial dos blocos para impedir – ou reduzir as chances de – que o *cluster* fique desequilibrado. Nesse contexto, [Dai et al. 2017] analisaram o modelo matemático por trás do problema de partição de nós para o posicionamento de réplicas no HDFS e apresentaram uma política de posicionamento de réplicas aprimorada, projetada especificamente para ambientes heterogêneos. A política proposta satisfaz todas as premissas de seleção impostas pela PPR padrão, ao mesmo tempo em que se esforça para garantir uma distribuição de réplicas equilibrada. Já o trabalho de [Liu et al. 2021] apresentou uma nova política de posicionamento de

réplicas baseada em grupo para o processamento em larga escala e em *batch* de dados geoespaciais *raster 3D*. Essa política visa otimizar o posicionamento das réplicas armazenadas no HDFS para reduzir a sobrecarga de rede causada por arquivos de regiões adjacentes armazenados aleatoriamente em vários nodos.

As abordagens reativas, em contraste, concentram-se nos blocos já armazenados no sistema, visando promover o balanceamento *inter-rack* e/ou entre DNs através da redistribuição das réplicas. Nesse sentido, [Dharanipragada et al. 2017] introduziram um algoritmo modificado como parte do balanceador *Tula* 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 no HDFS. Com isso, os DNs que apresentarem uma menor latência de disco recebem um número maior de blocos. Em contraste, [Shah and Padole 2018] focaram em otimizar o processo de redistribuição das réplicas aproveitando-se da capacidade de processamento dos nodos. Em contraste com o balanceador *Tula*, que não considera variações nos recursos de processamento e de memória dos nodos, o algoritmo de balanceamento proposto é baseado na capacidade de computação dos DNs. Sendo voltado a instâncias do Hadoop executando em ambientes heterogêneos, os blocos são redistribuídos apenas para DNs específicos, determinados a partir de uma classificação pela heterogeneidade e desempenho de cada nó, o que provê uma melhor distribuição de carga entre os nodos e reduz o tempo gasto em transferências de dados.

O HDFS também fornece uma ferramenta oficial para o balanceamento reativo dos dados entre os dispositivos de armazenamento no *cluster*, o HDFS *Balancer* [Shvachko et al. 2010]. Para realizar a redistribuição dos dados, o HDFS *Balancer* opera iterativamente movendo réplicas de DNs com alta utilização para DNs com um menor volume de dados mantido em seus dispositivos de armazenamento, até que a utilização de todos os DNs do *cluster* fique dentro de um intervalo controlado por um *threshold* de balanceamento. Para exemplificar, considerando o *threshold* padrão de 10% e, supondo que a utilização média do *cluster* esteja em 50%, a ferramenta irá executar até que a utilização de todos os dispositivos de armazenamento de todos os DNs estejam com utilização entre 40% e 60%. A definição do *threshold*, assim como as configurações adicionais dos atributos de balanceamento, deve ser feita pelo administrador [Foundation 2022]. Além disso, a execução do HDFS *Balancer* deve ser acionada sob demanda pelo administrador do *cluster* a partir da linha de comando, sempre que julgar necessário [White 2015].

Mesmo que a arquitetura do HDFS seja compatível com esquemas de rebalanceamento automático de dados entre DNs, tal funcionalidade não é implementada nativamente [Foundation 2022], deixando o HDFS *Balancer* propenso a configurações ineficientes e escolhas inapropriadas para o momento de sua execução. Assim, o HDFS *Balancer* apresenta limitações que dificultam seu uso e que podem resultar em gargalos de desempenho. O administrador do HDFS deve ter um conhecimento aprofundado do comportamento do sistema e de suas aplicações a fim de tomar decisões eficientes, que garantam alta confiabilidade sem afetar o funcionamento do *cluster*. Além disso, por atuar de forma generalizada, o balanceador pode não ser capaz de atender às demandas de confiabilidade e disponibilidade durante a redistribuição das réplicas. Como o sistema de arquivos apresenta variações de comportamento devido a fatores como falhas, volume de dados e número de clientes, a política de balanceamento atual nem sempre é satisfatória para atender às necessidades das aplicações em execução no HDFS.

4. Abordagem Proativa e Reativa para o Balanceamento de Réplicas

De modo a promover otimizações no processo de balanceamento no HDFS, implementou-se uma solução baseada na personalização da política de operação do HDFS *Balancer* e na automatização por meio do monitoramento ativo do ambiente computacional. Ambas as estratégias, quando combinadas, permitem endereçar o problema do desbalanceamento de réplicas de forma proativa e reativa, como descrito nas Seções 4.1 e 4.2

4.1. PRBP: *Prioritized Replica Balancing Policy*

Com um profundo entendimento de como o HDFS *Balancer* funciona internamente, foi feito o mapeamento de diferentes possibilidades de melhorias a serem incorporadas ao seu algoritmo de balanceamento de réplicas [Fazul and Barcelos 2019]. Tal investigação foi a base para a implementação de novas funcionalidades que foram incorporadas à política padrão do HDFS *Balancer*, criando assim uma política personalizada denominada *Prioritized Replica Balancing Policy* (PRBP) [Fazul and Barcelos 2023].

A PRBP se caracteriza por permitir funcionalidades de priorização durante o processo de balanceamento. Além disso, a redistribuição de dados entre os DNs deixa de depender de aleatoriedade e passa a ser configurável via parâmetros de execução do HDFS *Balancer*. Para tanto, estruturou-se um sistema de prioridades, exibido na Tabela 1, que baseia-se na topologia física do *cluster* e em diferentes características de usabilidade do sistema de arquivos. As prioridades foram agrupadas em quatro categorias de acordo com suas principais características operacionais e similaridades de implementação. Os nomes em inglês das categorias e prioridades foram mantidos para referências futuras em trabalhos como [Fazul and Barcelos 2023].

Categoria	**Prioridade**	**Objetivo de uso**
Node Capacity	*Processing Capacity* *Storage Capacity*	Personalizar o balanceamento em ambientes heterogêneos a partir de diferenças de *hardware* dos DNs.
Node Status	*Node Utilization* *Node Classification* *Node Load*	Reduzir o custo de processamento e transferência de dados do balanceamento através de priorizações com base em métricas recuperadas em tempo de execução.
Rack Status	*Rack Reliability* *Rack Utilization*	Permitir que o balanceamento seja conduzido considerando características dos *racks* que agrupam os DNs do *cluster*.
Data Distribution	*Data Availability*	Priorizar movimentações de réplicas que permitam aumentar a disponibilidade final dos blocos armazenados.

Tabela 1. Sistema de Prioridades definido pela PRBP.

Em todas as oito prioridades de balanceamento reativo que foram implementadas, existe a garantia de que a variação máxima do volume de dados em cada DN, após o balanceamento, permaneça controlada pelo valor de *threshold*. Além disso, a disposição dos blocos após a execução do HDFS *Balancer* continua respeitando as premissas definidas pela PPR padrão do HDFS. Embora as prioridades possam ser usadas isoladamente, alguns cenários se beneficiam de uma operação de balanceamento guiada por um conjunto de prioridades distintas. Ao permitir que múltiplas prioridades sejam configuradas em simultâneo, a PRBP possibilita que o processo de balanceamento seja especializado e realizado de acordo com demandas específicas. Entre as aplicações que podem se beneficiar do balanceamento de réplicas com prioridades associadas estão aquelas voltadas para

classificação e agrupamento de dados, que se concentram em indexação e classificação e as que fazem uso intensivo de dados e E/S.

Cabe ao administrador do sistema definir qual prioridade – ou conjunto de prioridades – deve ser considerada durante o balanceamento. Para dar suporte a este processo e seguindo as regras de associação definidas, foram idealizadas cinco *guidelines* (GL_1 a GL_5) para suportar o processo de tomada de decisão na configuração da PRBP [Fazul and Barcelos 2023]. A Tabela 2 apresenta as *guidelines* para utilização da PRBP, contemplando: (i) as prioridades de balanceamento recomendadas para uso associado; (ii) o principal objetivo perseguido pela *guideline*; (iii) o custo esperado (tempo de balanceamento e consumo de largura de banda) para conduzir o rebalanceamento de réplicas com a associação; e (iv) a magnitude do aumento de desempenho obtido após a execução do HDFS *Balancer* com a PRBP. Ressalta-se que todas as *guidelines* proporcionam algum nível de ganho de desempenho após o rearranjo das réplicas, pois o balanceamento da distribuição dos blocos melhora o desempenho em cenários com replicação de dados.

Guideline	Prioridades	Objetivo principal	Custo do balanceamento	Ganho de desempenho
GL_1	*Node Utilization* *Node Classification* *Node Load*	Desempenho no balanceamento	Baixo	Baixo
GL_2	*Storage Capacity* *Data Availability*	Desempenho de escrita	Moderado	Moderado
GL_3	*Processing Capacity* *Data Availability*	Desempenho de leitura	Alto	Alto
GL_4	*Rack Utilization* *Data Availability*	Balanceamento *inter-rack*	Alto	Moderado
GL_5	*Rack Reliability* *Data Availability*	Tolerância a falhas	Alto	Moderado

Tabela 2. *Guidelines* de uso para conduzir o balanceamento de réplicas no HDFS.

A *guideline* GL_1 é voltada para cenários com aplicações sensíveis ao desempenho, minimizando o tempo e o número de transferências necessárias para equilibrar o *cluster*. A GL_2, por sua vez, é projetada para ambientes heterogêneos em que as operações de carga de dados são frequentes, priorizando os nodos com alta capacidade de armazenamento durante a distribuição de dados. Em contraste, a GL_3 é especializada para atender demandas de desempenho de aplicações em razão de diferenças nas capacidades de processamento dos nodos. Já a GL_4 é especialmente útil para cenários com instâncias HDFS dispersas em muitos *racks*, onde é necessário minimizar o consumo de largura de banda global durante as operações de leitura. Por fim, a GL_5 é apropriada para cenários em que atributos de confiabilidade e disponibilidade são essenciais, contribuindo para a TF e a resiliência dos dados armazenados no sistema de arquivos.

4.2. Estrutura para automatização do uso do HDFS Balancer

Enquanto a PRBP especializa o uso do HDFS *Balancer*, endereçando as limitações de sua política de operação generalizada, a estrutura descrita nesta seção foi concebida frente às limitações de seu modo de operação, visando otimizar a configuração e automatizar o disparo do processo de balanceamento. A fim de alcançar esses objetivos, o monitora-

mento ativo do ambiente computacional é essencial para a identificação de cenários favoráveis ao balanceamento de réplicas, que proporcionem um *trade-off* satisfatório entre confiabilidade e desempenho. Assim, implementou-se uma estratégia de monitoramento dos nodos em tempo real com base no modelo agente-servidor [Fazul and Barcelos 2020] Três módulos de monitoramento foram desenvolvidos, nomeadamente: *agente*, *gerente* e *controlador*. Para a manutenção das informações de configuração, controle interno e comunicação entre os módulos, utilizou-se o *framework* Apache ZooKeeper [Haloi 2015].

A Figura 1 apresenta uma visão em alto nível da estrutura de monitoramento e automatização do balanceamento no HDFS [Fazul and Barcelos 2021]. Cada nodo do *cluster*, que também executa um processo DN do HDFS, passa a ser monitorado por um *agente*. Os *agentes* são responsáveis pela coleta de informações relacionadas aos dispositivos de armazenamento e à instância HDFS em suas máquinas. Conforme ilustrado, os *agentes*, periodicamente, reportam as observações realizadas para um servidor *gerente*. Tendo uma visão global do estado do sistema, cabe ao *gerente* analisar e processar os *reports* dos *agentes* para tomar decisões acerca do balanceamento de réplicas no *cluster*, promovendo, quando necessário, mudanças nas configurações do HDFS *Balancer*.

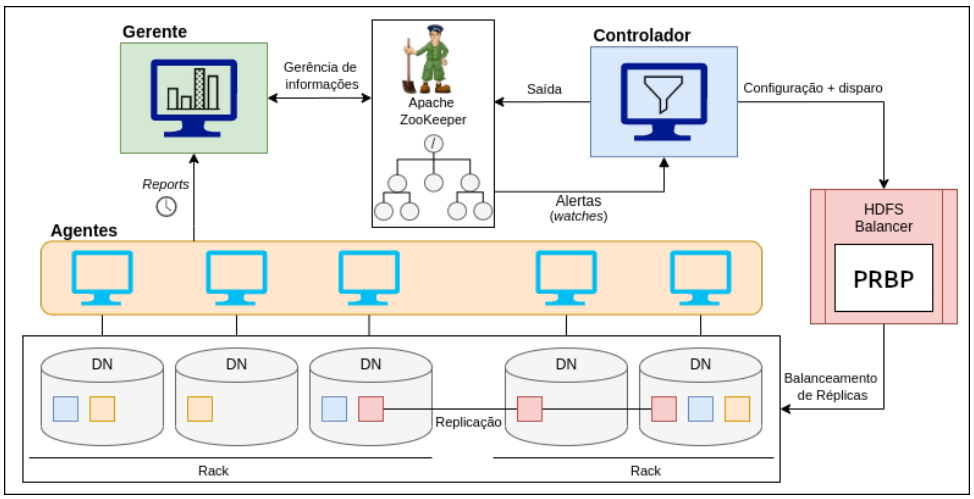

Figura 1. Estrutura para automação do balanceamento de réplicas.

De modo a determinar se o *cluster* está desbalanceado, o *gerente* baseia-se na metodologia padrão utilizada pelo HDFS *Balancer*, *i.e.*, divergência entre a utilização dos dispositivos de armazenamento e a média de utilização do *cluster*. Se o *gerente* perceber um desequilíbrio de réplicas acentuado no HDFS, ele inicia um procedimento para a definição de atributos de balanceamento de acordo com o contexto do sistema, que são posteriormente mapeados para parâmetros de execução do balanceador. A definição dos atributos foi inspirada nas configurações recomendadas para os modos de balanceamento *default*, *brackgroud* e *fast*, apresentados em [Cloudera, Inc. 2021]. Além da aplicação do modo de balanceamento, o *threshold* de balanceamento é configurado automaticamente, aplicando variações em torno do valor padrão de 10%. O modo e o *threshold* de balanceamento a serem utilizados são influenciados pela *guideline* da PRBP que será empregada durante o processo de balanceamento de réplicas.

Após definir os atributos de balanceamento a serem utilizados, o *gerente* salva as informações de controle na árvore do ZooKeeper e notifica o *controlador*. A comunicação

entre o *gerente* e o *controlador* é realizada através do *namespace* do Apache ZooKeeper, por meio do mecanismo de *watches* [Haloi 2015]. Ao receber a notificação, cabe ao *controlador* decidir se o balanceamento deve ser realizado no momento atual ou postergado, de modo a evitar sobrecarga desnecessária no sistema. Se o *controlador* julgar que a redistribuição de réplicas se faz necessária no sistema de arquivos, ele dispara a execução do HDFS *Balancer* – passando os parâmetros de execução definidos previamente pelo *gerente*. É responsabilidade do *controlador* aguardar pelo término da execução da *daemon* do balanceador a fim de coletar a saída da operação, que é salva no *namespace* do ZooKeeper. As informações históricas podem ser utilizadas posteriormente pelo *gerente* para a definição de atributos de balanceamento otimizados.

5. Experimentos e Discussão

Visando um relato focado de experiência prática, através da investigação do comportamento da estrutura de configuração e disparo automático do HDFS *Balancer* (abordagem proativa) em conjunto com a personalização da política de balanceamento fornecida pela PRBP (abordagem reativa), duas avaliações experimentais foram conduzidas, ambas na plataforma Grid'5000[1]. A seguir, a Seção 5.1 apresenta e discute os resultados com a *guideline* GL_1 e a Seção 5.2 com a *guideline* GL_5.

5.1. Experimentação incorporando a *guideline* GL_1 à estrutura de automatização

O experimento com a *guideline* GL_1 considerou um ambiente de teste configurado com 10 nodos no *cluster grisou* do *site Nancy*, no qual cada nodo (Dell PowerEdge R630) executou uma distribuição Debian GNU/Linux 10 (*buster*) e possuía as seguintes configurações: 2 CPUs Intel Xeon E5-2630 v3 (Haswell, 2.40GHz, 8 cores/CPU), 128GB de memória RAM, 558GB de capacidade de armazenamento HDD (SCSI Seagate) e uma conexão Ethernet de 1Gbps mais quatro conexões de 10Gbps.

O Apache Hadoop (versão 2.9.2) foi configurado em modo distribuído com 1 NN e 10 DNs (um por nodo). Para o cenário com a solução proposta, um módulo *agente* foi executado em cada nodo e uma instância replicada do Apache ZooKeeper (versão 3.6.1) foi utilizada para a comunicação entre os módulos *gerente* e *controlador*. Para a carga de dados no sistema, utilizou-se o *benchmark TestDFSIO* [White 2015] (versão 1.8), com o qual foram escritos 30 arquivos de 30GB cada, com um FR padrão de 3 réplicas por bloco, o que totalizou um volume de dados de aproximadamente $2{,}64$TB (21.600 réplicas de 128MB). Após a escrita, a utilização média do *cluster* ($U_{\mu,DISK}$) ficou em $57{,}01\%$.

A Figura 2 exibe o estado de utilização dos DNs do *cluster* em três cenários: (i) PPR padrão, com a distribuição dos dados baseada na PPR do HDFS (sem balanceamento); (ii) balanceador padrão, considerando o disparo manual (após a carga dos dados) do HDFS *Balancer* com as configurações padrão; e (iii) solução proposta, considerando o emprego da estratégia de automatização e personalização do balanceamento de réplicas. A linha pontilhada em azul representa a $U_{\mu,DISK}$ após a carga dos dados. Para cada DN é apresentada a sua carga calculada (L') com base nas definições da *guideline* GL_1, que se esforça para minimizar o tempo e o número de transferências de dados necessárias para

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

equilibrar o *cluster*, evitando transferir dados entre nodos vistos como ocupados, ou seja, que possam causar sobrecarga em outras tarefas em execução no sistema durante o balanceamento. O valor de L' varia de 0 a 1 sendo que, quanto mais próximo de 1, maior o tráfego de comunicação (número de conexões abertas) observado no nodo.

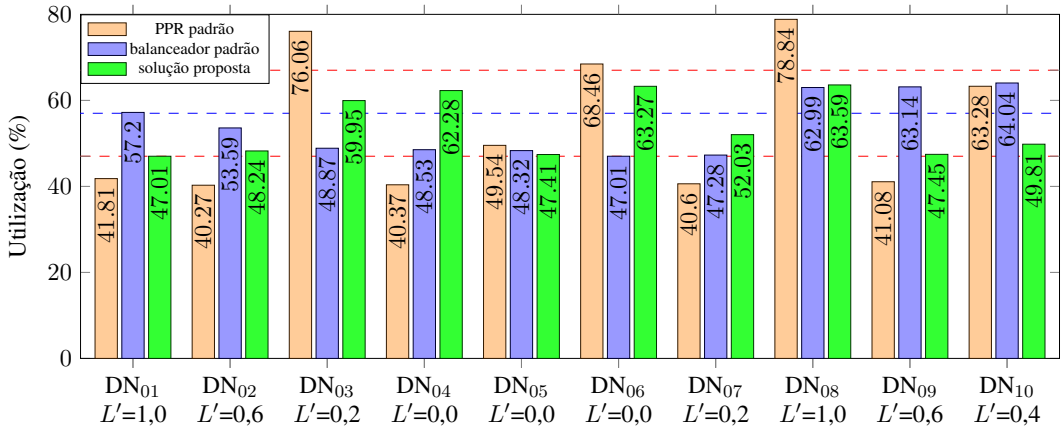

Figura 2. Utilização final dos DNs nos cenários analisados.

Com a PPR padrão, há uma grande discrepância no volume de dados mantido entre os DNs, que é evidenciada pelo desvio padrão elevado da utilização dos DNs ($15{,}94\%$). Conforme descrito na Seção 2, a razão disso reside na própria PPR, que não garante uma distribuição realmente equilibrada das réplicas. Já no cenário com o balanceador padrão, a utilização de cada DN se manteve dentro dos limites inferior ($U_{\mu,DISK}$ − *threshold*) e superior ($U_{\mu,DISK}$ + *threshold*) de balanceamento (linhas pontilhadas em vermelho), ou seja, $47{,}01\%$ ($57{,}01\% - 10\%$) e $67{,}01\%$ ($57{,}01\% + 10\%$), respectivamente. O desvio padrão das utilizações dos DNs nesse cenário foi de $7{,}13\%$. Todavia, com o balanceador padrão, a carga dos nodos não é considerada como critério para a redistribuição das réplicas. Com isso, DNs com alto tráfego de comunicação em seus nodos (*i.e.*, com L' igual ou próxima a $1{,}0$), tais como DN_{01} e DN_{08}, tiveram de despender tempo com processamento e transferência de dados durante o balanceamento. Ao total, $210{,}12$GB de dados foram movimentados em 12 iterações de balanceamento, que demandaram 2 horas e 40 minutos para serem concluídas. Vale ressaltar que, com a abordagem padrão, a execução do HDFS *Balancer* ficou dependente do disparo manual pelo administrador.

No cenário com a solução proposta, por sua vez, todas as decisões acerca do equilíbrio de réplicas no *cluster* são feitas pela estrutura de monitoramento. Assim como no cenário com o balanceador padrão, a utilização de cada DN também ficou dentro dos limites de balanceamento, resultando em um desvio padrão na utilização dos DNs de $7{,}24\%$. Além de automatizar a configuração e o disparo do balanceador, a solução empregando a *guideline* GL_1 faz com que o HDFS *Balancer* priorize a movimentação dos dados de acordo com o estado dos nodos, minimizando a sobrecarga em DNs que já estejam com um alto tráfego de comunicação em seus nodos. Tal comportamento pode ser observado, por exemplo, ao considerar que o DN_{01} (subutilizado) recebeu apenas a quantidade de blocos necessária para levar sua utilização até o limite inferior. Já o DN_{08}, por estar dentro do *threshold* e ter L' em $1{,}0$, não foi pareado em nenhuma iteração de balanceamento (*i.e.*, não enviou nem recebeu dados). Por operar visando o balanceamento mínimo do sistema e evitar o consumo desnecessário da largura de banda do *cluster*, o

emprego da *guideline* permitiu reduzir para 2 horas e 9 minutos o tempo de execução do HDFS *Balancer*. Ao total, 170GB de dados foram movimentados em 9 iterações. Desse modo, conforme evidenciado pelos desvios-padrão da utilização dos DNs obtidos nos três cenários, a estrutura de automatização é capaz de manter o estado de equilíbrio do *cluster* em um nível similar ao do HDFS *Balancer* padrão ao mesmo tempo que evita sobrecarga adicional causada pelo processo de balanceamento de réplicas.

Visando investigar possíveis melhorias de desempenho em aplicações de E/S possibilitadas pelo equilíbrio das réplicas em um momento pós-balanceamento, foram realizadas 20 execuções do *TestDFSIO* em modo leitura nos três cenários analisados. A média aritmética dos tempos de leitura foi de 1.158,77s no cenário com a PPR padrão, 1.097,59s com o balanceador padrão e de 1.029,11s com a estrutura de automatização. Considerando o cenário com a PPR padrão em relação ao cenário com a solução proposta, a variação percentual obtida foi de $-11{,}19\%$, que representa a redução no tempo necessário para a leitura dos dados com a estratégia apresentada neste trabalho. Comparando ao cenário baseado no HDFS *Balancer* padrão, por sua vez, a variação percentual foi de $-6{,}24\%$. Sendo assim, além de reduzir o tempo e a largura de banda necessários para o balanceamento, o emprego da estrutura de automatização neste experimento permitiu uma exploração otimizada da localidade espacial dos dados armazenados no HDFS.

5.2. Experimentação incorporando a *guideline* GL_5 à estrutura de automatização

A validação com a *guideline* GL_5 considerou um ambiente de teste com 21 nodos (Dell PowerEdge R630) configurados no *cluster paravance* do *site Rennes*, executando sobre uma distribuição Debian GNU/Linux 10 (*buster*). Cada nodo possuía 2 CPUs Intel Xeon E5-2630 v3 (Haswell, 2.40GHz, 8 cores/CPU), 128GB de memória RAM, 600GB de capacidade de armazenamento HDD (SATA Seagate) e 2 conexões Ethernet de 10Gbps.

O Apache Hadoop (versão 2.9.2) foi configurado para operação em modo distribuído com 1 NN e 21 DNs (um por nodo) dispostos em 5 *racks* distintos (R_1 a R_5), cada um mantendo, inicialmente, 3, 3, 4, 5 e 6 DNs, respectivamente. Uma instância replicada do Apache ZooKeeper (versão 3.6.1) foi configurada para o cenário com a solução proposta. A carga dos dados foi realizada com o *TestDFSIO* [White 2015] (versão 1.8), com o qual 40 arquivos de 30GB cada e com um FR padrão foram escritos no HDFS, totalizando um volume de dados de 3,52TB (28.800 réplicas de 128MB cada). A utilização média do *cluster* ($U_{\mu,t}$) após a escrita ficou em $50{,}64\%$. Considerou-se um cenário de teste com ocorrência de falhas de DN durante a operação de escrita. Inicialmente, foram introduzidas falhas – através do comando *kill* do Linux e com um intervalo de 60 segundos entre cada falha – nos processos de 6 DNs, sendo 1 deles pertencente ao *rack* R_3, 2 ao R_4 e 3 ao R_5. Assim, após a escrita, cada *rack* do *cluster* mantinha 3 DNs ativos.

A Tabela 3 exibe a utilização média dos *racks* do *cluster* em dois momentos (Δ_0 e Δ_1). O momento Δ_0 considera a distribuição dos dados ao final da execução do *TestDFSIO*, dividido entre o cenário com a PPR padrão do HDFS e com solução proposta. O momento Δ_1 considera uma falha posterior, introduzida no *rack* R_5, em ambos os cenários. Para cada *rack* é exibido o seu fator de confiança (T'), que é um parâmetro calculado pela *guideline* GL_5 com a finalidade de priorizar a operação de balanceamento. O fator de confiança de um *rack* é dado em razão de sua taxa de falhas (*i.e.*, proporção de DNs com falhas em relação ao total de DNs no *rack* em questão) normalizada e invertida, de forma que *racks* com alta taxa de falhas tenham um baixo T' e, analogamente, *racks* com uma

baixa taxa de falhas tenham um alto T'. O valor de T' varia de 0 a 1 sendo que, quanto mais próximo de 1, maior a confiança calculada para o *rack*.

Considerando o estado do *cluster* em Δ_0, o desvio padrão da utilização dos DNs com a PPR padrão foi de $11{,}40\%$, ao passo que esse valor foi reduzido para $3{,}76\%$ com o emprego da solução proposta, demonstrando sua atuação na conservação do balanceamento. Além disso, percebe-se como, com a PPR padrão, o fator de confiança não é determinante para o posicionamento das réplicas, de modo que *racks* com baixo T' podem vir a armazenar um alto volume de dados. Em contraste, a incorporação da *guideline* GL_5 pela estrutura de automatização faz com que o volume de dados armazenado em cada *rack* respeite seus respectivos fatores de confiança. Dessa forma, os *racks* com maior T' (R_1 e R_2) do *cluster* (*i.e.*, menor incidência de falhas de DNs) ficaram responsáveis por manter uma maior quantidade de dados. Analogamente, os *racks* com menor T' tiveram uma redução no volume de dados armazenado em seus DNs. Assim, conforme observado, a utilização média dos *racks* respeita o valor de T' no cenário com a solução proposta.

Δ_0				Δ_1			
Rack	**T′**	**PPR padrão**	**solução proposta**	**Rack**	**T′**	**PPR padrão**	**solução proposta**
R_1	1,0	41,56%	51,21%	R_1	1,0	54,94%	60,43%
R_2	1,0	41,30%	51,87%	R_2	1,0	54,62%	60,87%
R_3	0,5	55,12%	48,65%	R_3	0,4	71,77%	59,53%
R_4	0,2	43,76%	45,09%	R_4	0,0	58,08%	58,61%
R_5	0,0	57,81%	43,13%	-	-	-	-

Tabela 3. Utilização média após a carga dos dados (Δ_0) e após falha em R_5 (Δ_1).

Outra diferença no comportamento das abordagens pode ser observada pelo número de *racks* em que as réplicas de um mesmo bloco residem. Tendo em vista que a PPR, por padrão, armazena duas réplicas em um mesmo *rack*, todos os blocos estavam dispostos em exatamente dois *racks*. No cenário com a solução proposta, por sua vez, a *guideline* GL_5 prioriza as transferências que permitem aumentar a disponibilidade final dos dados durante o balanceamento. Com o utilitário *fsck*, constatou-se que, dos 1.927 blocos redistribuídos pelo HDFS *Balancer* com a solução proposta, 1.763 alcançaram uma disponibilidade máxima. Isso demonstra que $91{,}49\%$ dos blocos movimentados foram posicionados na maior quantidade de *racks* possível, dado o FR configurado.

Ao aumentar a disponibilidade, a TF do sistema é aprimorada, evitando que dados sejam perdidos mesmo se *racks* inteiros venham a falhar. Para possibilitar tal otimização, entretanto, um maior número de réplicas precisam ser transferidas entre os *racks* do *cluster*, o que demanda um maior esforço (tempo e consumo de banda) para efetivar o balanceamento. Ao total, o HDFS *Balancer* movimentou $317{,}38$GB de dados em 3 execuções fazendo com que a operação de escrita levasse $2.936{,}92$s para ser concluída. Em contraste, a escrita com a PPR padrão levou apenas $2.435{,}17$s. Sendo assim, a manutenção do balanceamento de réplicas com a solução proposta acarretou em um aumento de $17{,}08\%$ no tempo necessário para a escrita dos arquivos no HDFS.

Para investigar o impacto do processo de re-replicação após a carga inicial dos dados, uma falha total de *rack* foi simulada no *rack* com o menor fator de confiança do *cluster*. Assim, introduziu-se falhas em todos os DNs restantes no *rack* R_5. O momento Δ_1 na Tabela 3 exibe o novo estado de utilização dos *racks* do HDFS assim que processo

de re-replicação dos blocos armazenados em R_5 foi concluído. Ressalta-se que, com a redução de 15 para 12 DNs ativos no sistema, a utilização média do *cluster* ($U_{\mu,t}$) após as falhas ficou em $63{,}19\%$. Além disso, o fator de confiança (T'), que é um valor relativo e normalizado, foi recalculado apenas considerando os *racks* ativos do *cluster* (R_1 a R_4), com base na taxa de falhas do DNs desses *racks*.

Observa-se que, com a PPR padrão, grande parte do volume de dados anteriormente mantido pelo *rack* R_5 foi redistribuído para o *rack* R_3. O desvio-padrão da utilização dos DNs ativos do *cluster* foi de $9{,}21\%$. Esse desbalanceamento ocorre pois a re-replicação, que também segue a PPR padrão do HDFS, não considera a utilização dos nodos. A estrutura de automatização da solução proposta, por sua vez, detecta o desequilíbrio resultante das falhas dos DNs e automaticamente inicia o processo de balanceamento. Assim, com um *threshold* adaptado para 5%, foi possível reduzir o desvio-padrão para $1{,}33\%$. Além disso, a *guideline* GL_5 baseia-se nos fatores de confiança para determinar o destino dos blocos, logo, a utilização final de cada *rack* passa a estar em concordância com o respetivo T' do *rack*. Ao total, 270 blocos foram movimentados e, em absoluto, reposicionados para o maior número de *racks* permitido pelo FR, aprimorando a confiabilidade e a disponibilidade dos dados no HDFS.

6. Considerações Finais

Utilizar o HDFS *Balancer* de forma eficiente apresenta-se como um desafio, uma vez que diversos fatores podem interferir no seu comportamento e afetar severamente o processo de balanceamento de réplicas no HDFS. Com um profundo conhecimento das estratégias de replicação e balanceamento,estabeleceram-se as bases para otimizar e especializar o processo de redistribuição de réplicas no HDFS. Para tanto, nesse trabalho, duas soluções previamente validadas foram combinadas de forma que o processo de balanceamento de réplicas passe a ser conduzido por meio de uma abordagem proativa e reativa.

Para a abordagem proativa, foi utilizada uma estrutura baseada no monitoramento ativo do *cluster*. Tal estrutura automatiza processo de tomada de decisão para a configuração e o disparo do HDFS *Balancer*. Já para a abordagem reativa, a política de operação do HDFS *Balancer* foi personalizada com base na PRBP. A PRBP e suas *guidelines* permitem especializar o balanceamento para contextos especializados, atuando como uma estratégia para melhorar o desempenho (seja durante o balanceamento, escrita ou leitura de dados), alcançar o equilíbrio em nível de *rack* ou fornecer tolerância a falhas. Com ambas as abordagens combinadas, o balanceamento de réplicas passa a ser conduzido com base no monitoramento em tempo real do ambiente computacional, removendo assim a dependência de configuração e disparo manual do HDFS *Balancer*.

Os resultados obtidos com a experimentação demostram que a solução combinada foi capaz de tornar transparente o processo de balanceamento e prover melhorias significativas de desempenho, considerando perspectivas de confiabilidade e disponibilidade. Sendo assim, com uma visão global do estado dos elementos do sistema e com o suporte do Apache ZooKeeper, foi possível automatizar o processo de tomada de decisão para a configuração e o disparo do HDFS *Balancer*. Além disso, o conceito da priorização do balanceamento de réplicas, por meio da PRBP, também foi incorporado com sucesso à estrutura construída com o ZooKeeper.

Trabalhos futuros envolvem a avaliação do custo do balanceamento conside-

rando a frequência de ações subsequentes que se fazem necessárias para a manutenção e conservação do *cluster* em um estado balanceado. Isso possibilitará uma compreensão aprofundada do *trade-off* entre o esforço para a redistribuição de réplicas em diferentes momentos e os ganhos de confiabilidade, disponibilidade e desempenho obtidos.

Referências

Achari, S. (2015). *Hadoop Essentials*. Packt Publishing Ltd, 1 edition.

Cloudera, Inc. (2021). Managing data storage. `https://docs.cloudera.com/runtime/7.2.12/scaling-namespaces/topics/hdfs-balancing-data-across-hdfs-cluster.html`. Março.

Dai, W., Ibrahim, I., and Bassiouni, M. (2017). An improved replica placement policy for hadoop distributed file system running on cloud platforms. In *4th Int. Conf. on Cyber Security and Cloud Computing (CSCloud)*, pages 270–275, New York. IEEE.

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

Fazul, R. and Barcelos, P. (2020). O apache zookeeper como estratégia de monitoramento ativo para manter o balanceamento de réplicas no hdfs. In *Anais do XXI Workshop de Testes e Tolerância a Falhas*, pages 1–14, Porto Alegre, RS, Brasil. SBC.

Fazul, R. and Barcelos, P. P. (2019). Política customizada de balanceamento de réplicas para o hdfs balancer do apache hadoop. In *Anais do XX Workshop de Testes e Tolerância a Falhas*, pages 90–103, Porto Alegre, RS, Brasil. SBC.

Fazul, R. W. A. and Barcelos, P. P. (2021). Automation and prioritization of replica balancing in hdfs. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, SAC '21, page 235–238, New York, NY, USA. ACM.

Fazul, R. W. A. and Barcelos, P. P. (2023). Prbp: A prioritized replica balancing policy for hdfs balancer. *Software: Practice and Experience*, 53(3):600–630.

Foundation, A. S. (2022). "HDFS Architecture". `https://hadoop.apache.org/docs/r3.3.4/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html`. Fevereiro.

Haloi, S. (2015). *Apache Zookeeper Essentials*. Packt Publishing Ltd, 1 edition.

Liu, Z., Hua, W., Liu, X., Liang, D., Zhao, Y., and Shi, M. (2021). An efficient group-based replica placement policy for large-scale geospatial 3d raster data on hadoop. *Sensors*, 21(23):8132.

Shah, A. and Padole, M. (2018). Load balancing through block rearrangement policy for hadoop heterogeneous cluster. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 230–236, Bangalore. IEEE.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Symposium on Mass Storage Systems and Technologies*, pages 1–10, Incline Village. IEEE.

Turkington, G. (2013). *Hadoop Beginner's Guide*. Packt Publishing Ltd, 1 edition.

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