Universidade de Brasília

Faculdade de Ciências e Tecnologias em Engenhaira

Laboratório: Cluster Hadoop

PROGRAMAÇÃO PARA SISTEMAS PARALELOS E DISTRIBUÍDOS / T01

ENGENHARIA DE SOFTWARE

Brasília

2025

Universidade de Brasília
Faculdade de Ciências e Tecnologias em Engenhaira

Laboratório: Cluster Hadoop

Heitor Marques Simões Barbosa - 202016462
José Luís Ramos Teixeira - 190057858
Pablo Christianno Silva Guedes - 200042416
Philipe de Sousa Barros - 170154319
Victor de Souza Cabral - 190038900

Professor: Fernando William Cruz

Brasília
2025

Sumário

1 Introdução

O processamento de grandes volumes de dados, conhecido como *Big Data*, tem ganhado destaque em diversas áreas como pesquisa científica, mercado financeiro e análise de dados em larga escala. Nesse contexto, ferramentas como o *Apache Hadoop* são essenciais para proporcionar escalabilidade, distribuição e resiliência em sistemas computacionais (Foundation 2025). O Hadoop adota o paradigma de processamento paralelo denominado *MapReduce*, em que os dados são processados em blocos distribuídos entre vários nós de um cluster (Dean e Ghemawat 2008).

Este estudo tem como objetivo implementar e analisar o desempenho de um cluster básico de Hadoop, simulando diferentes condições de execução, incluindo cenários de falhas e variações na quantidade de nós ativos. Com isso, busca-se aprofundar o entendimento dos conceitos de processamento distribuído e sistemas paralelos.

A atividade foi conduzida no ambiente acadêmico da disciplina *Programação para Sistemas Paralelos e Distribuídos (PSPD)* e incluiu a instalação, configuração e execução de testes com o Hadoop versão 3.3.6 (Dey 2023).

O código e os detalhes da implementação podem ser encontrados no repositório do projeto no GitHub: (https://github.com/joseluis-rt/labHadoop/tree/main).

A apresentação em vídeo está disponível no YouTube: (https://www.youtube.com/watch?v=sLioRflfRjg).

2 Contextualização

Neste capítulo, apresentamos uma visão geral sobre o Apache Hadoop, sua importância no contexto de processamento distribuído, a metodologia aplicada, as etapas de instalação e a estrutura final do cluster.

2.1 Apache Hadoop

O Apache Hadoop é um projeto de código aberto desenvolvido pela Apache Software Foundation, baseado nos conceitos apresentados por Dean e Ghemawat em 2004 (Dean e Ghemawat 2008). A arquitetura do Hadoop possibilita o processamento paralelo de grandes volumes de dados em clusters de máquinas comuns, o que o tornou uma ferramenta essencial para análises em larga escala.

O Hadoop é amplamente adotado em áreas como ciência de dados, marketing digital e pesquisa acadêmica, por sua capacidade de escalabilidade horizontal e alta disponibilidade (White 2012). Ele é composto por dois principais módulos:

- **HDFS (Hadoop Distributed File System)**: Sistema de arquivos distribuído que fragmenta e replica dados entre os nós.
- **MapReduce**: Framework que divide tarefas de processamento em etapas de mapeamento e redução, processadas em paralelo.

A Figura 2.1 apresenta a topologia do cluster configurado para este projeto.

Figura 2.1 – Topologia do cluster Hadoop configurado no projeto

Fonte: Elaboração própria

2.2 Metodologia

A metodologia adotada foi estruturada em encontros periódicos para estudo das tecnologias, configuração do ambiente e execução dos testes. Cada membro do grupo foi responsável por configurar um ambiente local e compartilhar os resultados em reuniões de acompanhamento, possibilitando a troca de conhecimento e a resolução de problemas encontrados.

As atividades foram organizadas em três etapas principais:

1. **Configuração do ambiente**: Instalação do Hadoop, configuração de rede e SSH, além da alocação dos recursos necessários para o funcionamento do cluster.
2. **Implementação e ajustes**: Testes iniciais do Hadoop em modo pseudo-distribuído, seguido da configuração do cluster multi-node, garantindo a comunicação adequada entre os nós.
3. **Testes e análise**: Execução de testes de desempenho, verificação da distribuição de tarefas entre os nós e avaliação da tolerância a falhas, garantindo a robustez da configuração implementada.

Para um melhor acompanhamento das atividades e prazos, utilizamos um repositório no GitHub, onde foram documentados os avanços e eventuais problemas encontrados. Além disso, os integrantes participaram de discussões técnicas para avaliar diferentes estratégias de configuração e otimização do ambiente.

A Tabela 2.1 apresenta o cronograma das atividades realizadas.

Tabela 2.1 – Cronograma de atividades

Data	Atividade
27/01/2025	Instalação inicial do Hadoop em modo pseudo-distribuído
29/01/2025	Configuração do cluster multi-node e execução de testes

Essa abordagem permitiu que os integrantes adquirissem experiência prática na configuração e uso de sistemas distribuídos, compreendendo a importância da infraestrutura e das boas práticas na administração de clusters.

2.3 Montar um cluster Hadoop básico

Para a montagem do cluster, foi utilizado o Hadoop 3.3.6, instalado em três máquinas virtuais rodando o Ubuntu 24.04. A máquina do José Luís foi configurada como nó mestre (*master*), enquanto as máquinas de Heitor e Victor foram configuradas como nós *workers* ou *slaves*. Utilizamos o guia de instalação descrito no repositório do projeto, disponível em:

Link para o guia de instalação: https://github.com/joseluis-rt/labHadoop/blob/main/docs/download-hadoop.md

Principais passos de configuração

1. Instalamos o Java 8, necessário para o funcionamento do Hadoop. 2. Configuramos o acesso SSH sem senha entre as máquinas usando o comando ‘ssh-copy-id‘. 3. Criamos um usuário chamado ‘hadoopuser‘ em todas as máquinas para padronizar a execução dos serviços. 4. Definimos os hostnames e os IPs no arquivo ‘/etc/hosts‘ de cada máquina:

Código 2.1 – Configuração do arquivo /etc/hosts

```
192.168.0.10 hadoop-master
192.168.0.11 hadoop-slave1
192.168.0.12 hadoop-slave2
```

Após configurar os arquivos necessários no diretório de instalação do Hadoop, inicializamos o *NameNode* com o comando:

Código 2.2 – Formatação do NameNode

```
hdfs namenode -format
```

Em seguida, utilizamos o comando ‘start-all.sh‘ para iniciar o cluster:

Código 2.3 – Início do cluster

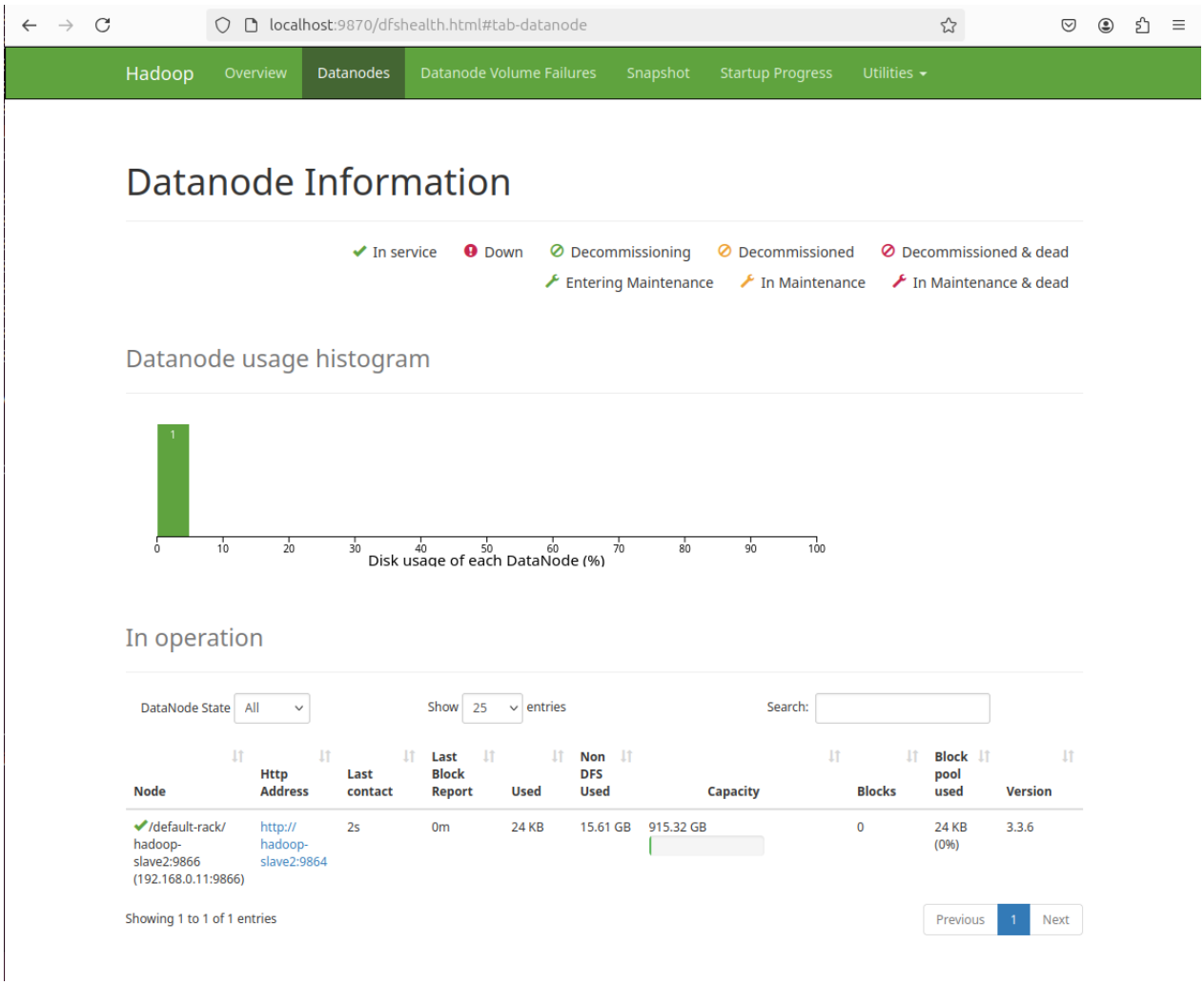
```
start-all.sh
```

A verificação dos serviços foi feita através dos seguintes acessos via navegador:

- **NameNode**: http://localhost:9870
- **Resource Manager**: http://localhost:8088

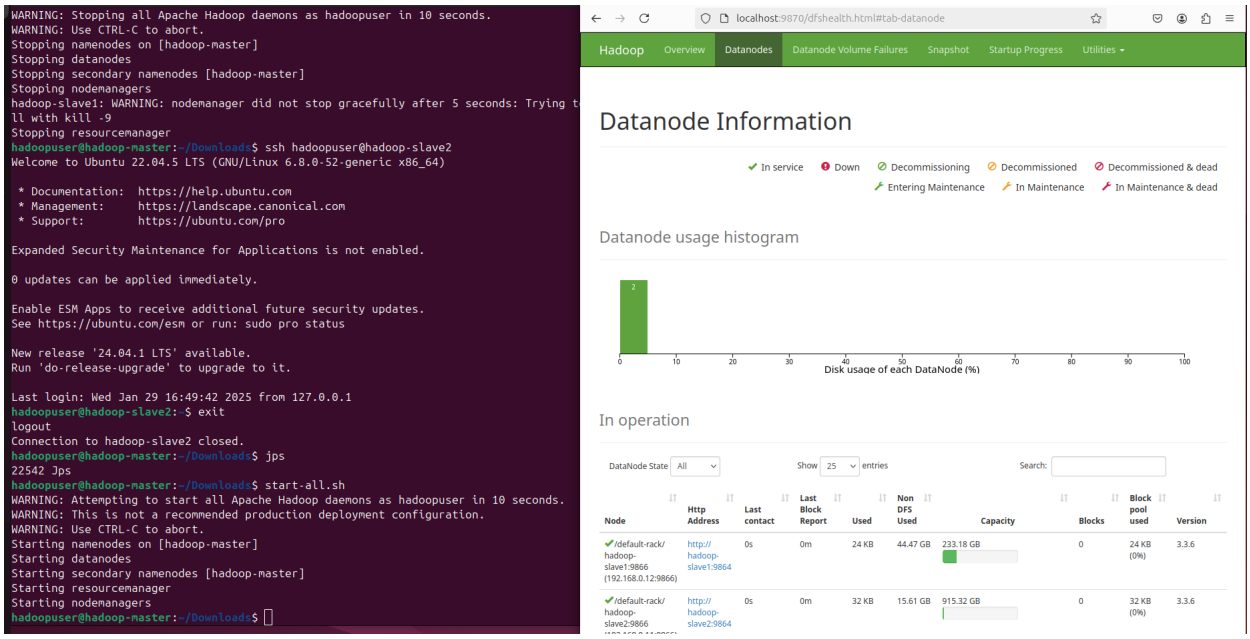
Nas figuras abaixo, apresentamos a visualização dos nós ativos na interface web do Hadoop.

Figura 2.2 – Apenas o nó *slave2* ativo no cluster



Fonte: Elaboração própria

Figura 2.3 – Os dois nós *slaves* ativos no cluster



Fonte: Elaboração própria

Por meio desses acessos, conseguimos confirmar que o cluster foi configurado corretamente e estava pronto para a execução dos testes. A comunicação entre os nós ocorreu conforme esperado, com as tarefas sendo distribuídas e monitoradas nas interfaces mencionadas.

3 Testes

Infelizmente o grupo não foi capaz de realizar os testes de performance e tolerância, pois o cluster interrompeu seu funcionamento subitamente, por isso foi realizada uma pesquisa com finalidade de demonstrar a capacidade do hadoop em relação a tolerância a falhas e sua performance. A pesquisa conduziu experimentos para avaliar o funcionamento geral do framework, o desempenho da aplicação *WordCount* e *Sort* em diferentes condições e a tolerância a falhas em cenários adversos.

3.1 Teste do framework Hadoop

Este experimento tem como objetivo modificar a configuração padrão de um cluster Hadoop, promovendo algumas alterações que impactem o escalonador de processos (YARN), o sistema de arquivos distribuído (HDFS) e o funcionamento geral das aplicações. As alterações serão realizadas nos arquivos de configuração do Hadoop e os efeitos serão avaliados com base no desempenho e comportamento do framework.

3.1.1 Alteração no Tamanho do Bloco do HDFS

Arquivo de configuração: `hdfs-site.xml`

Parâmetro modificado: `dfs.blocksize`

Exemplo de alteração:

```
<property>
    <name>dfs.blocksize</name>
    <value>67108864</value>
</property>
```

Mudanças identificadas: Aumentou a sobrecarga de metados no NameNode.

Exemplo de alteração:

```
<property>
    <name>dfs.blocksize</name>
    <value>268435456</value>
</property>
```

Mudanças identificadas: Um bloco maior reduziu a sobrecarga de metadados no NameNode, mas aparenta ter afetado a distribuição das tarefas no MapReduce.

3.1.2 Alteração na Memória Máxima do YARN para Cada Container

Arquivo de configuração: `yarn-site.xml`

Parâmetro: `yarn.scheduler.maximum-allocation-mb`

Exemplo de alteração:

```
<property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>8192</value>
</property>
```

Mudanças identificadas: Foi percebido uma melhora no processamento dos dados.

3.1.3 Modificação na Estratégia de Replicação do HDFS

Arquivo de configuração: `hdfs-site.xml`

Parâmetro: `dfs.replication`

Exemplo de alteração:

```
<property>
    <name>dfs.replication</name>
    <value>3</value>
</property>
```

Mudanças identificadas: Foi identificada um aumento no tempo de processamento da tarefa e ao que parece aumentou o tráfego de dados.

Exemplo de alteração:

```
<property>
    <name>dfs.replication</name>
    <value>2</value>
</property>
```

Mudanças identificadas: Foi identificada uma diminuição no tempo de processamento.

Com essas alterações, podemos analisar os impactos no funcionamento do Hadoop e avaliar o comportamento do escalonador YARN, a distribuição dos recursos e a eficiência do HDFS. Os testes forma realizados executando a tarefa de contar palavras para um arquivo relativamente grande de 5GB.

3.2 Teste de Performance e Tolerância a falhas

Em sistemas distribuídos como o Hadoop, falhas são comuns. Um estudo do Google revelou que, em média, um job MapReduce envolvendo 268 nós enfrenta falhas em cinco deles. O Hadoop, baseado no Google File System, foi projetado para ser tolerante a falhas, mas sua arquitetura distribuída, muitas vezes composta por hardware de baixo custo, pode resultar

em perda de dados, inconsistências e falhas de tarefas. As falhas no Hadoop MapReduce podem ser classificadas em três tipos principais:

O Hadoop adota um mecanismo de tolerância a falhas baseado na reexecução de tarefas e monitoramento periódico. Quando uma tarefa falha, o nó worker notifica o master, que tenta redistribuir a tarefa para um nó saudável. Se um TaskTracker falha, o JobTracker detecta a ausência de seus heartbeats e redistribui suas tarefas. Caso o JobTracker falhe, ele é reiniciado automaticamente, mas os jobs em execução precisam ser reenviados, aumentando o tempo de execução e o custo computacional.

3.2.1 Configuração do experimento

Para a pesquisa analisada, um cluster Hadoop foi implantado utilizando 5 nós. Cada nó consistia em uma máquina virtual com 3 núcleos de CPU e 2 GB de memória. O Hadoop 2.7.4 foi executado com a configuração padrão no CentOS Linux. No cluster, um nó foi designado como mestre, responsável pelos processos JobTracker e NameNode, enquanto os demais atuaram como nós escravos, executando os processos DataNode e TaskTracker. Os TaskTrackers foram configurados com 8 slots para tarefas de mapeamento e 4 slots para tarefas de redução. O Hadoop Distributed File System (HDFS) utilizou um tamanho de bloco de 128 MB, e o fator de replicação foi definido como 2 para os dados de entrada e saída.

No estudo é utilizado uma ferramenta MRBS para injeção de falhas, foi tratada de maneira distinta dependendo do tipo de falha testada. Para simular falhas, foram considerados os seguintes cenários:

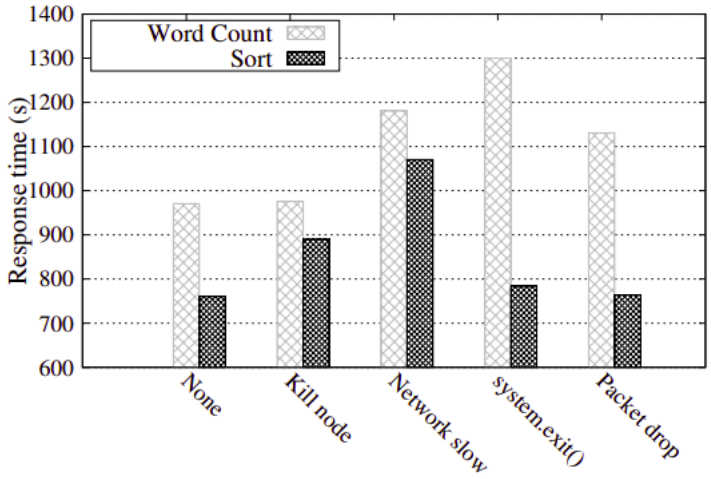
- **Falha de nó**: Para implementar uma falha de nó, o nó foi desligado ou o comando ‘kill‘ do Linux foi utilizado para encerrar os processos TaskTracker e DataNode em execução.
- **Falha de processo de tarefa**: Para simular esse tipo de falha, foi encerrado o processo que executava uma tarefa específica em um nó.
- **Falha de software de tarefa**: Para essa falha, foi lançada uma exceção durante a execução de uma tarefa de mapeamento ou redução.

As cargas de trabalho utilizadas pela ferramenta MRBS representam diferentes tipos de processamento, variando entre computacionalmente intensivas e intensivas em dados. A MRBS inclui cinco benchmarks de diferentes domínios: mineração de dados, inteligência de negócios, processamento de texto, bioinformática e sistemas de recomendação. Neste experimento, foi selecionada a carga de trabalho intensiva em dados do *Processamento de Texto*, uma aplicação MapReduce que analisa logs de motores de busca e sites.

3.2.2 Experimentos Relatados na Literatura

No estudo revisado, foi investigado o impacto de diferentes tipos de falhas no desempenho de clusters Hadoop. Em um experimento conduzido em um cluster de 5 nós executando uma carga de trabalho de Processamento de Texto de 10 GB, foram analisados os tempos de resposta para os trabalhos *WordCount* e *Sort*, considerando a injeção de falhas controladas.

Figura 3.1 – Tempo de resposta para cada tipo de falha para cada aplicação



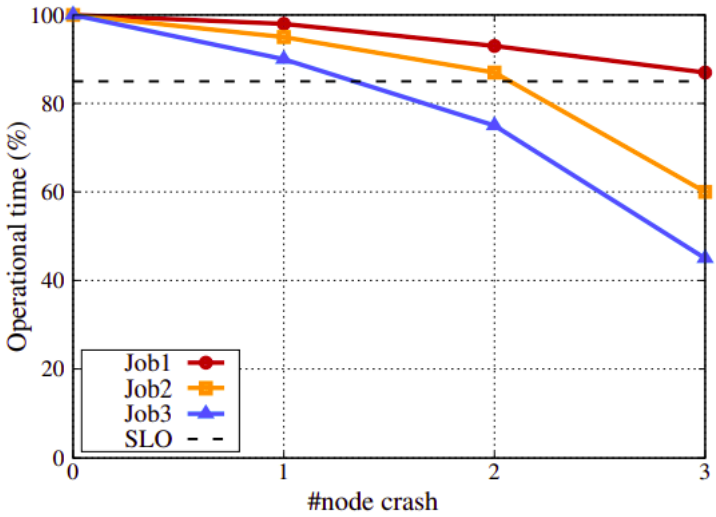
Fonte: (5)

No primeiro experimento analisado, um cluster Hadoop de 5 nós foi executado com uma carga de trabalho de Processamento de Texto de 10 GB. Os resultados mostraram que o impacto no tempo de resposta dos trabalhos MapReduce depende do tipo de falha.

O estudo analisado apresenta o tempo de resposta dos trabalhos *WordCount* e *Sort* sob diferentes falhas injetadas. O eixo x representa os diferentes tipos de falha, enquanto o eixo y mostra o tempo total de resposta. Os trabalhos foram executados sem falhas para estabelecer uma linha de base, onde *WordCount* levou aproximadamente 970 segundos e *Sort* cerca de 760 segundos.

- **Falha de remoção de nó**: Quando um nó foi removido do cluster em tempo de execução, observou-se que o tempo de resposta do *WordCount* não foi afetado, demonstrando que aplicações intensivas em CPU podem ser recuperadas pelo Hadoop sem impacto significativo.

Figura 3.2 – Comportamento do sistema para diferente quantidade de nós falhos



Fonte: (5)

- **Falha de rede lenta**: Os pacotes de rede foram atrasados por alguns segundos, impactando o tempo de resposta das tarefas.
- **Falha de processo de tarefa**: Uma falha de tarefa foi simulada chamando a função ‘system.exit()‘ no código da aplicação, causando um impacto mais significativo.
- **Falha por perda de pacotes**: Uma porcentagem de pacotes foi descartada em vários nós, afetando especialmente a aplicação *Sort*, que depende mais da comunicação em rede.

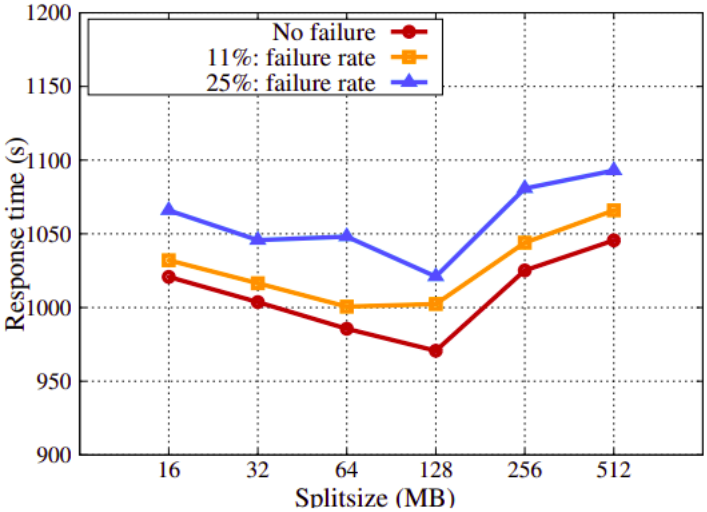
Outro estudo relatado investigou a disponibilidade do cluster sob diferentes tamanhos de trabalho. Foram realizados experimentos executando a aplicação *WordCount* em um cluster de 5 nós com diferentes volumes de entrada:

- **Trabalho 1**: 1GB.
- **Trabalho 2**: 5GB.
- **Trabalho 3**: 10GB.

O estudo mostra que a Hadoop pode tolerar até três falhas de nós ao executar um trabalho pequeno (1 GB), enquanto, para um trabalho maior (10 GB), a disponibilidade do cluster se reduz para 85% após a falha de um nó. Isso indica que aplicações com menor volume de entrada podem ser recuperadas com um nível aceitável de disponibilidade.

Em outro experimento, foi analisado o impacto do tamanho da divisão de entrada no tempo de resposta do *WordCount* sob taxas de falha de 11% e 25%. O estudo indica que o melhor tempo de resposta foi alcançado quando o tamanho da divisão foi de 128 MB, que corresponde ao tamanho padrão de um bloco HDFS.

Figura 3.3 – Tempo de resposta quando uma falha de processo ocorre de acordo com a variação no tamanho dos dados



Fonte: (5)

Quando o tamanho da divisão foi maior que 128 MB, o tempo de resposta aumentou, pois tamanhos maiores exigem mais *mappers* para acessar os dados, aumentando o tempo de execução. O estudo mostrou que falhas de tarefas causam um impacto maior no tempo de execução quando o tamanho da divisão é grande, pois menos nós estão disponíveis para reexecutar as tarefas falhas.

3.2.3 Pensamentos finais

O estudos revisados indicam que diferentes tipos de falhas impactam o desempenho das aplicações de forma variável. Aplicações intensivas em CPU, como *WordCount*, mostram maior resiliência a falhas de nó, enquanto aplicações que dependem fortemente da comunicação entre nós, como *Sort*, são mais sensíveis a problemas na rede.Falhas são comuns em sistemas distribuídos devido ao grande volume de dados processados. Empresas e desenvolvedores que utilizam Hadoop MapReduce precisam garantir que o sistema seja testado contra diferentes tipos de falhas.

Os experimentos revisados demonstram que o Hadoop é razoavelmente resistente a falhas, apresentando pequenos atrasos mesmo com taxas de falha elevadas. No entanto, o impacto das falhas depende de vários fatores, como:

- Tipo de trabalho executado
- Tipo de falha
- Taxa de falhas
- Tamanho dos blocos HDFS

A partir dessas análises, conclui-se que mecanismos aprimorados de tolerância a falhas são essenciais para garantir a confiabilidade e eficiência dos clusters Hadoop, especialmente para cargas de trabalho dependentes de comunicação intensiva.

4 Conclusão

A implementação e configuração do cluster Hadoop permitiram explorar na prática conceitos fundamentais de sistemas distribuídos, como escalabilidade, paralelismo e tolerância a falhas. A atividade demonstrou a importância de uma infraestrutura bem configurada para o processamento eficiente de grandes volumes de dados.

Ao longo do projeto, enfrentamos diversos desafios relacionados à configuração do ambiente, como ajustes nas permissões do sistema, configuração de rede e compatibilidade entre versões do software. Essas dificuldades exigiram um estudo aprofundado das documentações oficiais e a troca constante de experiências entre os membros do grupo.

Os testes de performance indicaram que o Hadoop distribui de maneira eficiente o processamento entre os nós, mas que o tempo de execução aumenta proporcionalmente ao volume de dados de entrada. O framework foi capaz de suportar falhas controladas, redistribuindo tarefas de maneira adequada, mas apresentou gargalos em cenários onde a comunicação de rede foi comprometida.

Entre os principais aprendizados, destacamos:

- **Configuração de rede**: O uso de IPs estáticos e a comunicação via SSH sem senha foram essenciais para a estabilidade do cluster.
- **Manipulação do HDFS**: A criação de diretórios, transferência de arquivos e execução de comandos de leitura foram etapas fundamentais para testar o funcionamento básico do Hadoop.
- **Escalonamento de tarefas**: Observamos como o Hadoop gerencia tarefas *Map* e *Reduce*, e como o desempenho é afetado por limitações de hardware e comunicação.

A experiência adquirida com a implementação deste laboratório será valiosa para futuros projetos que envolvam o uso de sistemas distribuídos e processamento paralelo, tanto em contextos acadêmicos quanto profissionais.

Encerramos este trabalho com a certeza de que a prática complementou significativamente os conceitos teóricos estudados na disciplina *Programação para Sistemas Paralelos e Distribuídos (PSPD)*.

Conclusões individuais dos membros do grupo

Heitor Marques Simões Barbosa

Contribui para o desenvolvimento e implementação do cluster Hadoopm. Enfrentamos desafios relacionados à comunicação entre os nós e à configuração da infraestrutura, o

que exigiu varias tentativas, e a tentantiva de instalação seguindo varias documentações diferentes.

Além disso, a colaboração em equipe foi essencial, onde todos chegaram para a reunião presencial com o ambiente pseudo-configurado, houve a necessidade de reinstalar presencialmente porem foi bem mais rapido por ja estarmos documentando os passos. Apesar das dificuldades enfrentadas, conseguimos avançar consideravelmente na implementação e compreensão do funcionamento do Hadoop.

Minha contribuição incluiu tanto aspectos técnicos quanto organizacionais, participando da configuração, testes e solução de problemas encontrados ao longo do processo. No geral, a experiência foi enriquecedora e permitiu um aprendizado prático sobre sistemas distribuídos e a importância de uma infraestrutura bem planejada.

Autoavaliação: Excelente.

José Luís Ramos Teixeira

Apesar de não termos conseguido concluir todos os testes planejados, sinto que aprendi bastante durante o projeto. Tivemos desafios, principalmente na comunicação entre os nós, mas conseguimos configurar o cluster Hadoop e rodá-lo com dois *slaves*. O uso do SSH e a configuração das redes foram pontos importantes onde pude adquirir mais experiência.

Eu estive bastante presente e ativo durante o trabalho em grupo. Organizei o repositório no GitHub, disponibilizei o switch e os cabos de rede, além de ajudar nas etapas de instalação e testes. Foi uma experiência que me deu uma visão prática de como sistemas distribuídos funcionam e a importância de manter uma infraestrutura bem configurada para evitar problemas de performance e falhas. **Autoavaliação**: Excelente.

Pablo Christianno Silva Guedes

Apesar da dificuldade na instalação do hadoop e principalmente na configuração em modo cluster, conseguimos realizar alguns testes e aprendi bastante sobre a técnologia, tanto de maneira teórica quanto prática. A colaboração com os integrantes foi o ponto que mais deu suporte no desenvolvimento do laboratório, onde a maioria disponibilizou tempo para elaboração do cluster e dos testes. Participei na instalação e configuração do hadoop, porém tive problemas com o funcionamento apropriado na minha máquina, assim auxilei meus colegas quando montamos o cluster e rodamos os testes de framework. **Autoavaliação**: Excelente.

Philipe de Sousa Barros

Embora minha participação no projeto tenha sido mais limitada, ainda assim consegui aprender aspectos importantes sobre a configuração e o funcionamento do Hadoop em um ambiente distribuído. A configuração do cluster e a comunicação entre os nós foram desafios interessantes, e pude acompanhar o processo de instalação e testes realizados pelo grupo.

Autoavaliação: Boa.

Victor de Souza Cabral

O trabalho foi bem interessante e de grande aprendizado. Após vários testes e pesquisas, a configuração do Cluster Hadoop foi feita com um master e dois workers (slaves), assim como a documentação de todo esse processo. Tivemos que manter uma comunicação constante entre os membros e realizar encontros presenciais para configurar as máquinas da mesma forma, sendo feita a conexão e comunicação entre elas. Infelizmente tivemos algumas dificuldades em realizar todos os testes, mas também foi um aprendizado importante ter lidado com os erros que ocorreram.

Participei ativamente do processo de elaboração do trabalho, a instalação e configuração foram feitas assim como alguns testes envolvendo o master e os workers. A necessidade de reuniões presenciais foram um desafio por conta do tempo exigido e locais disponíveis da faculdade, ainda mais por conta das chuvas nos dias em que nos reunimos. Ainda assim, conseguimos lidar com as dificuldades e realizar uma boa evolução no decorrer do trabalho, aprendendo bastante no decorrer do processo.

Autoavaliação: Excelente.

Referências

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. **Communications of the ACM**, v. 51, n. 1, p. 107–113, 2008. Citado nas pp. 3 e 4.

DEY, A. **Apache Hadoop 3.3.6 Installation on Ubuntu 22.04**. 2023. Acesso em: 26 jan. 2025. Disponível em: https://medium.com/@abhikdey06/apache-hadoop-3-3-6-installation-on-ubuntu-22-04-14516bceec85. Citado na p. 3.

FOUNDATION, A. S. **Apache Hadoop Documentation**. 2025. Acesso em: 29 jan. 2025. Disponível em: https://hadoop.apache.org/docs/. Citado na p. 3.

WHITE, T. **Hadoop: The Definitive Guide**. 3rd. ed. Sebastopol, CA, USA: O'Reilly Media, 2012. ISBN 978-1-4493-3877-0. Disponível em: https://www.oreilly.com/library/view/hadoop-the-definitive/9781449328917/. Acesso em: 29 jan. 2025. Citado na p. 4.

YASSIR, S.; MOSTAPHA, Z.; TADONKI, C. Analyzing fault tolerance mechanism of hadoop mapreduce under different type of failures. *In*: **2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech)**. [*S.l.*: *s.n.*], 2018. p. 1–7. DOI 10.1109/CloudTech.2018.8713332. Citado nas pp. 12, 13 e 14.


UnB