Avaliação de Desempenho de Sistemas de Arquivos *Linux*: Um Estudo Comparativo com *Benchmarks*

Keven E. V. Bilibio[1,2], Juliana S. Silva[1]

[1]Departamento de Computação – Campus Cuiabá – Cel. Octayde Jorge da Silva – Instituto Federal de Mato Grosso - R. Zulmira Canavarros, 95 – Centro, Cuiabá - MT, 78005-390 – Cuiabá-MT – Brasil.

[2]IFMT Maker FabLab, Instituto Federal do Mato Grosso – Campus Cuiabá – Cel. Octayde Jorge da Silva – Cuiabá – MT, Brasil

e.vaz@estudante.ifmt.edu.br, juliana.silva@cba.ifmt.edu.br

Abstract. *File systems are the foundation upon which we access and organize data, influencing everything from the boot speed of an operating system to the performance of more complex applications on Linux systems. Given this reality, this article presents a comparative study among the main Linux file systems (XFS, ZFS, EXT4, BTRFS and F2FS), using the Flexible Input/Output Tester (FIO) benchmark tool, to evaluate their performance under different workloads. From the results, it was possible to compare performance metrics across these file systems, such as transfer rate, latency, and input/output operations per second. Therefore, it is hoped that this study can assist people in selecting the most suitable file system option, according to their needs.*

Resumo: *Os sistemas de arquivos são a base para qual acessamos e organizamos os dados, influenciando desde a velocidade de inicialização de um sistema operacional até o desempenho de aplicações mais complexas em sistemas Linux. Diante dessa realidade, este artigo apresenta um estudo comparativo entre os principais sistemas de arquivos Linux (XFS, ZFS, EXT4, BTRFS e F2FS), utilizando a ferramenta de benchmark Flexible Input/Output Tester (FIO), para avaliar seu desempenho em diferentes cargas de trabalho. A partir dos resultados, foi possível comparar métricas de desempenho sobre esses sistemas de arquivos, como taxa de transferência, latência e operações de entrada/saída por segundo. Assim sendo, espera-se que este estudo possa auxiliar as pessoas na seleção da opção mais adequada de um sistema de arquivo, de acordo com a sua necessidade.*

1. Introdução

Os sistemas de arquivos são a base sobre a qual acessamos e organizamos dados, influenciando desde a velocidade de inicialização de um sistema operacional até o desempenho de aplicações (ArchWiki, 2024). Isso pode impactar desde usos mais simples até mais complexos, como bancos de dados e sistemas de armazenamento em nuvem. A escolha do sistema de arquivos pode depender de alguns fatores e necessidades do usuário, que vão desde o tipo de *hardware* (*Solid State Drive — SSD, Hard Disk Drive — HDD, Non-Volatile Memory Express — NVMe*), o sistema operacional utilizado, a carga de trabalho (leitura ou escrita intensiva, acesso aleatório) e os requisitos de segurança e de integridade dos dados (Almeida, 2009).

Por conta da transformação digital em diversas áreas, em função do surgimento de novas tecnologias, como Inteligência Artificial, Big Data e Internet das Coisas (IoT), emergiu a necessidade de estudo e desenvolvimento de sistemas de arquivos, cada vez mais eficazes, para gerenciar e armazenar uma alta quantidade de dados, de maneira mais segura, além de criar recursos mais modernos (Laureano, 2021).

Assim sendo, a realização de um estudo comparativo entre os diversos sistemas de arquivos Linux, é um assunto de grande relevância para orientar pessoas e/ou empresas na escolha entre os diferentes tipos de hardware e ambientes de uso. Trabalhos recentes, como o de Shaikh (2024), investigam os limites e o comportamento dos sistemas de arquivos Linux EXT4, XFS, BtrFS, ZFS e F2FS ao lidar com um número massivo de arquivos pequenos (até um bilhão), em armazenamento local, analisando métricas como throughput e degradação de desempenho. Estudos como esse exploram os limites dos sistemas de arquivos e abrem oportunidades para pesquisas focadas em melhorar o desempenho nesses sistemas.

Diante dessa realidade, o objetivo deste artigo é realizar um estudo comparativo entre os principais sistemas de arquivos Linux (XFS, ZFS, EXT4, BTRFS e F2FS), utilizando a ferramenta de benchmark Flexible Input/Output Tester (FIO), para avaliar seu desempenho em diferentes cargas de trabalho. E, a partir dos resultados dos testes, apresentar as vantagens e as desvantagens de cada sistema, na busca por um melhor entendimento do funcionamento de cada um. Por fim, espera-se que, a partir dessa experimentação, este artigo auxilie as pessoas na escolha de um sistema de arquivos mais apropriado, dependendo da necessidade, considerando suas capacidades e restrições.

Desta forma, este artigo está estruturado em cinco seções, incluindo esta introdução. A seção 2 aborda os conceitos fundamentais dos principais sistemas de arquivos Linux. A seção 3 aborda a metodologia aplicada, detalhando os testes conduzidos, o ambiente e as métricas avaliadas. Em seguida, a seção 4 apresenta os resultados obtidos e discute o desempenho de cada sistema de arquivos, sob diferentes cargas de trabalho. Por último, a seção 5 apresenta as considerações finais.

2. Referencial Teórico

Esta seção tem o objetivo de descrever, brevemente, alguns termos fundamentais para a compreensão desse estudo de caso, como sistema de arquivos *Linux*, suas características e quais são os principais sistemas de arquivos utilizados atualmente.

2.1 Sistemas de Arquivos *Linux*

Os sistemas de arquivos são uma parte essencial dos sistemas operacionais, eles são responsáveis pela maneira como os dados são acessados e organizados (ArchWiki, 2024). Assim, fornecem uma base sobre qual dados e programas são armazenados para o sistema operacional e para o usuário. A escolha de um sistema de arquivos apropriado pode afetar o desempenho, a segurança e a durabilidade do sistema (Almeida, 2009), podendo impactar desde a velocidade de inicialização do sistema operacional até o desempenho das aplicações.

Os sistemas de arquivos permitem que os usuários criem e armazenem dados de forma organizada, sem a necessidade do usuário gerenciar os endereços de memória, as partições do disco ou os espaços reservados na memória (Laureano, 2021).

O Sistema Operacional *Linux* oferece suporte a uma ampla variedade de sistemas de arquivos, permitindo até que, diferentes sistemas de arquivos, sejam montados em diretórios ou participações específicas, proporcionando modularidade, versatilidade e adaptabilidade (Laureano, 2021). Entre esses vários sistemas de arquivos disponíveis para *Linux*, este estudo abordará os mais utilizados atualmente: XFS, ZFS, EXT4, BTRFS e F2FS - os quais são apresentados a seguir.

2.1.1 Sistema de Arquivo X (*X File System* - XFS)

O Sistema de Arquivo X (*X File System - XFS*) foi desenvolvido pela *Silicon Graphics*, em 1994, para o sistema operacional IRIX 5.3, com o propósito de ser um sistema robusto e escalável, para gerenciar grandes quantidades de arquivos. Em 1999, foi apresentada uma versão adaptada para o *Linux* e, em 2004, foi adicionada ao *Kernel Linux* 2.4 (Almeida, 2009).

Uma das principais características do XFS é a utilização de um *buffer* para armazenar temporariamente os dados, antes de serem gravados no disco, ajudando a reduzir a fragmentação e a otimizar o desempenho das operações de leitura e escrita. Além disso, o XFS possui um *journaling* que auxilia na recuperação de dados, em caso de falhas. Ele também conta com uma ferramenta para a desfragmentação do disco, que reorganiza os arquivos, sem a necessidade de desmontar a partição (Rodeh; Bacik; Mason, 2019; ArchWiki, 2020).

No entanto, o XFS possui um problema de consumir mais recursos ao gerenciar arquivos pequenos, por conta do maior uso do *buffer*, ao gerir esses tipos de dados, aumentando a carga da CPU (Almeida, 2009). Adicionalmente, o XFS não permite a redução do tamanho das partições, após a sua criação, embora algumas alternativas sejam discutidas pela comunidade.

Uma vez que foi apresentado o referencial teórico desta pesquisa, a seção a seguir se dedica apresentar a metodologia utilizada neste estudo.

2.1.2 Sistema de Arquivos Zettabyte (*Zettabyte File System* - ZFS)

O Sistema de Arquivos Zettabyte (*Zettabyte File System* - ZFS) é um sistema de arquivos criado com a ideia de garantir integridade de dados, recuperação e facilidade na administração. Foi inicialmente desenvolvido pela empresa *Sun Microsystems* em 2001 e integrado ao *Solaris* em 2005; tem sido utilizado, principalmente, em *desktops* e servidores de banco de dados (Bonwick *et al.*, 2003). Projetado para superar problemas de sistemas tradicionais, evita perda de arquivos, corrupção de dados e dificuldades de gerenciamento.

O ZFS faz o uso de *checksums* e árvores de Merkle para detectar e corrigir falhas automaticamente, além de alocação dinâmica de espaço para otimizar o armazenamento. Além disso, possui suporte a *snapshots* e COW, que facilitam a criação de *backups* e versionamento de dados. O ZFS é bem flexível, podendo se adaptar em discos individuais ou, até mesmo, grandes *pools* de armazenamento. Ele também possui o mecanismo de atualizações atômicas, garantindo que escritas no sistema de arquivos sejam concluídas ou não ocorram, evitando a corrupção dos dados (Bonwick *et al.*, 2003).

Por outro lado, o ZFS tende a consumir mais recursos do sistema, especialmente de memória *RAM*, o que pode afetar *hardwares* mais simples (Lima, 2021). Sua configuração pode ser um pouco complexa e, embora esteja disponível para *Linux*, via "ZFS *on Linux*", o ZFS foi pensado para o Solaris e, por isso, nem todas as funcionalidades podem estar presentes. Além disso, a sua compatibilidade com outros sistemas operacionais, como o *Windows*, é limitada. Em ambientes virtualizados, o alto consumo de *RAM* pode afetar a estabilidade, sendo recomendável usá-lo com outros sistemas de arquivos (Lima, 2021).

2.1.3 Sistema de Arquivos *B-tree* (*B-tree File System* - BTRFS)

O Sistema de Arquivos *B-tree* (*B-tree File System* — BTRFS), é um sistema de arquivos que começou a ser desenvolvido em 2007, combinando conceitos do *ReiserFS*, *B-tree* e *Copy-On-Write* (COW), sendo mantido por empresas como Fujitsu, Intel, Oracle e SUSE, que buscam tornar o BTRFS o principal sistema de arquivos *Linux* (Rodeh; Bacik; Mason, 2013).

Sua principal característica é o uso de árvores B (*B-tree*) em conjunto ao COW, garantindo um bom desempenho ao longo do tempo. Diferente do EXT4, que atualiza arquivos diretamente, o BTRFS copia nós *B-tree* modificados para locais não alocados, resultando em uma alta taxa de escrita sequencial; no entanto, acaba gerando mais dados do que o necessário, para serem gravados. Ele também faz uso de *extents* para alocar blocos contíguos com maior eficiência, além de *checksums* para verificar a integridade dos dados e da técnica de alocação e desalocação de blocos de dados, conhecida como contagem de referências, para a otimização de espaço de armazenamento (Rodeh; Bacik; Mason, 2013).

O BTRFS possui algumas desvantagens, como ser considerado menos estável comparado a outros sistemas que possuem mais tempo de suporte e desenvolvimento. O uso do COW, apesar de ser bem eficiente, pode impactar no desempenho devido à alta

taxa de escrita. Além disso, *snapshots* tornam a desfragmentação mais complexa, pois mover os *extents* exige atualização de múltiplos ponteiros.

2.1.4 Quarto Sistema de Arquivo Estendido (*Fourth Extended File System* - EXT4)

O Quarto Sistema de Arquivo Estendido (*Fourth Extended File System - EXT4*) começou a ser desenvolvido em 2006 e foi incorporado ao código do *Kernel Linux* na versão 2.6.19, em 2008 (Almeida, 2009). O EXT4, é atualmente o sistema de arquivos mais conhecido e usado no *Linux*, principalmente por sua estabilidade e capacidade de gerenciar arquivos de até 16 *terabytes* (TB) e volumes de até 1 *Exabyte* (EB) - o que fez com que muitas distribuições adotassem esse sistema de arquivo como padrão.

O EXT4 possui o recurso *Journaling Block Device 2* (JBD2), uma camada de registro para a proteção dos dados, que registra as operações antes que elas sejam escritas no disco, protegendo os metadados contra falhas inesperadas (*Linux Kernel Organization*, 2025). Assim, é possível configurar o modo de funcionamento desse registro, para somente metadados ou dados e metadados, caso deseje mais desempenho ou mais segurança. Além disso, utiliza *extents*, que alocam de forma mais eficiente blocos contíguos, reduzindo a fragmentação do disco e facilitando a gestão de grandes arquivos.

Apesar das vantagens, o EXT4 tem algumas desvantagens, como o risco de perda de dados, caso o *journaling* não esteja configurado corretamente, algo já vivenciado por um dos proponentes desta pesquisa. Ele também não possui suporte nativo a *snapshots*, recurso útil para o *backup* e a restauração do sistema operacional.

2.1.5 Sistema de Arquivos *Flash*-Amigável (*Flash-Friendly File System* - F2FS)

O Sistema de Arquivos *Flash*-Amigável (*Flash-Friendly File System — F2FS*) é um sistema de arquivos criado e otimizado para dispositivos de armazenamento *flash*, como SSDs e NAND. Ele foi proposto pela Samsung, em 2012, e foi incorporado ao *Kernel Linux* 3.8, em 2013. Projetado para explorar as características desse tipo de armazenamento, superando o EXT4 em diversas situações.

O F2FS demonstrou desempenho até 3,1 vezes melhor em dispositivos móveis e reduziu o tempo de algumas tarefas em 40% (Lee *et al*., 2015). Em servidores, foi até 2,5 vezes mais rápido, com o uso de SSDs SATA, e 1,8 vezes melhor em SSDs PCIe (*Peripheral Component Interconnect Express*). Além disso, utiliza-se de política de *multi-head logging*, separando dados quentes (acessados mais frequentemente) e frios (acessados com menos frequência) para otimizar o desempenho, além de um gerenciamento segmentado para melhor organização do armazenamento (Lee *et al.,* 2015).

O sistema de arquivos utiliza uma estratégia de mapeamento híbrido para gerenciar as localizações dos dados no dispositivo de armazenamento, que melhora gravações aleatórias e mantém um alto desempenho, sem manter grandes tabelas de mapeamento na memória (Lee *et al.*, 2015). Apesar das vantagens, o F2FS ainda é relativamente novo, em constante desenvolvimento, e não oferece compatibilidade com outros sistemas de arquivos.

2.1.6 Quadro Resumo - Sistemas de Arquivos Analisados

A partir dessa revisão da literatura sobre os principais sistemas de arquivos *Linux* disponíveis atualmente, foi possível mapear características-chave sobre cada um deles, as quais estão dispostas no Quadro 1.

Quadro 1. Quadro Resumo dos Sistemas de Arquivos *Linux*

	XFS	**ZFS**	**EXT4**	**BTRFS**	**F2FS**
Ano	1994	2001	2008	2007	2013
Journaling	Metadados	Via COW	JBD2 Configuravel	Via COW	*Log-structured*
Copy-on-Write	Não	Sim	Não	Sim	Não
Snapshots	Não	Sim	Não	Sim	Não
Checksums	Metadados	Dados e Metadados	Metadados	Dados e Metadados	Metadados
Compressão Nativa	Não	Sim	Não	Sim	Sim
Caracteristicas	Escalabilidade, Grandes Arquivos, *Extents*, Estável	Integridade de dados, RAID-Z, Pools, COW, Snapshots	Padrão Popular, Estável, *Extents*, Equilibrado	COW, Snapshots, RAID Integrado, Flexivel	*Log-structured*, Otimizado para NAND/SSD, Dados Quentes/Frios

Uma vez que o referencial teórico do trabalho foi apresentado, a seção a seguir se dedica a apresentar a metodologia utilizada para o desenvolvimento do estudo de caso.

3. Metodologia

Para a realização dos *benchmarks* desse estudo comparativo, foi utilizado um *notebook* com processador Intel I7-1255U, 16GB de RAM (2x8GB, 3200MHz) e SSD NVMe de 512GB. O *Arch Linux* foi a opção escolhida para o Sistema Operacional, por conta da maior afinidade de um dos pesquisadores por esta distribuição. Assim sendo, o *Arch Linux* foi instalado pelo *script archinstall*, com perfil mínimo de instalação, particionamento padrão (sem *swap* e sem encriptação de disco), rodando o *Kernel Linux* 6.12.10 genérico - que é o mais recente disponível, até a data de realização dos testes, além de possuir o *BASH* 5.2.37. Para a transferência dos dados também foi instalado no *notebook* o pacote *android-tools*, além da ferramenta de *benchmark* FIO.

Para a consistência dos testes, o *notebook* foi formatado entre cada sistema de arquivo testado, depois da realização de todos os testes, garantindo um sistema limpo e dedicado. A única exceção foi o ZFS, que não estava disponível no instalador padrão, não sendo possível executar o sistema operacional nativamente. Então, para os testes, o ZFS foi implementado utilizando os pacotes experimentais *zfs-utils*, *zfs-linux* e *zfs-linux-headers*, fornecidos pelo *archzfs*, criando um ponto de montagem em partição separada, utilizando o comando "`zpool create -f -o ashift=12 -m /mnt/pool zfs /dev/nvme0n1p3`".

O *benchmark* utilizou o FIO 3.38 com duas cargas de trabalho, pesada e leve, a fim de simular um cenário mais leve e outro mais pesado, bem como avaliar o desempenho do sistema de armazenamento. A carga pesada simulou um ambiente de alta demanda, utilizando os parâmetros "`ioengine=libaio`", que permite operações de I/O assíncronas e otimiza o uso dos recursos de *hardware*, e "`direct=1`", para as

operações serem realizadas diretamente, sem que o *cache* do sistema tenha uma influência no teste. O tamanho do arquivo escolhido foi de 60 GB, o tempo de execução de 360 segundos, 10 *jobs* simultâneos e uma profundidade de fila (`iodepth`) de 32, que simula um cenário parecido com um servidor de banco de dados ou com a virtualização de um sistema com múltiplas requisições simultâneas. Os resultados obtidos foram unificados por meio da opção "`group_reporting`". A carga leve simulou um ambiente de baixa demanda, mantendo o tempo de execução de 360 segundos, mas com arquivos de 10 GB, 4 *jobs* e um "`iodepth`" de 4, simulando cenários de uso pessoal. Em ambos os cenários foram executados teste de escrita/leitura sequencial (com tamanhos de blocos de 128k) e aleatória (com tamanho de blocos de 4k); cada teste foi executado 2 vezes, gerando 4 arquivos de resultado, em *json*, para cada sistema de arquivo.

Os dados de cada teste foram armazenados no formato *json*, o que permitiu a criação de um *script* em *python* para a análise e a geração dos gráficos, de forma automática. A partir de então, foram gerados 6 gráficos, baseados no tipo de carga (leve/pesada) e as seguintes métricas analisadas: (i) ***throughput*** (medida em MB/s), que mede a quantidade de dados transferidos por unidade de tempo, nesse caso em MB/s; (ii) **IOPS** (*input/output operations per second*), que mede o número de operações de entrada/saída por segundo; e (iii) **latência**, que mede o tempo médio de resposta dos sistema de arquivos, em milissegundos - os resultados desses testes são apresentados a seguir.

Os arquivos de configuração para os testes (leve e pesado), além do *script* em *python* (para a plotagem dos gráficos), encontram-se disponíveis no seguinte endereço eletrônico: https://github.com/Vaz15K-Faculdade/estudo_sistemas_arquivos_linux.

4. Estudo Comparativo

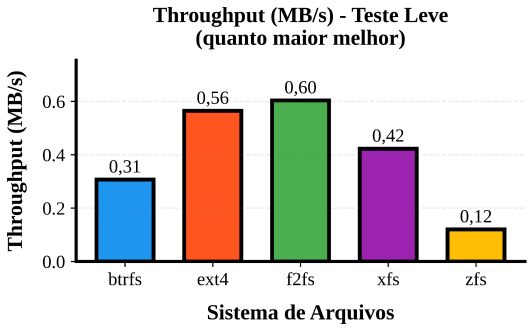
Com os testes realizados utilizando a ferramenta de *benchmark* FIO, nos sistemas de arquivos EXT4, XFS, BTRFS, F2FS e ZFS, foi possível obter alguns resultados sobre o desempenho de cada sistema. Com base nas métricas definidas, a seguir é descrita uma análise mais detalhada dos resultados obtidos, sob condições de carga leve e pesada.

Em condições leves (Figura 1A), o F2FS se destacou com o melhor desempenho, devido principalmente às suas otimizações para dispositivos *flash*, mostrando a sua excelente capacidade para gerenciar operações I/O com arquivos menores. Na sequência, o EXT4 e XFS apresentaram bons resultados em função do uso de *extents*, *buffers* e *journaling*, que contribuem para a organização e velocidade dos dados, mostrando-se eficientes, mesmo em cenários de baixa demanda, sem muitas complicações.

Seguindo para o teste de carga pesada (Figura 1B), o EXT4 assumiu a liderança, mostrando que seu longo tempo de suporte o torna bom para gerenciar grandes quantidades de dados, podendo até ser usado em banco de dados e servidores. O F2FS e o XFS mantiveram um bom desempenho, com taxas de transferência bem próximas às do EXT4, mostrando que os dois sistemas conseguem lidar com altas quantidades de dados, de forma simultânea.

Por outro lado, o BTRFS e ZFS apresentaram quedas significativas de *throughput*, tanto no teste de carga leve (Figura 1A), quanto no teste de carga pesada

(Figura 1B), mostrando que os recursos como COW, *snapshots* e *checksums*, embora sejam muito eficazes na parte de segurança, comprometem o desempenho da transferência dos dados - isso se mostra mais visível em ambientes mais intensos.



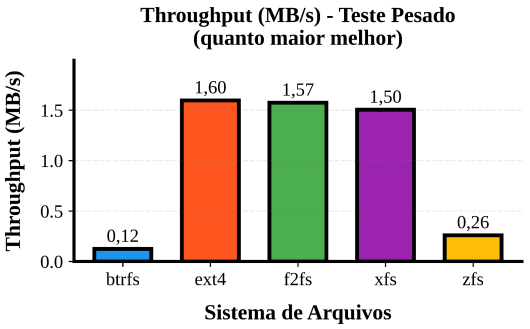


Figura 1A. Teste leve *throughput*

Figura 1B. Teste pesado *throughput*

Com relação aos testes de IOPS, observou-se um padrão de desempenho quando comparado ao teste de *throughput*. Em condições de carga leve (Figura 2A), o F2FS se destacou novamente, demonstrando a sua capacidade de processar várias operações de I/O, em ambientes de baixa demanda, que fazem acessos aleatórios e frequentes. O EXT4 e XFS também apresentaram bons resultados, mostrando novamente que as técnicas de *buffer* e *extents* minimizam o tempo de espera entre as operações, afetando diretamente o IOPS.

Seguindo para o teste em carga pesada (Figura 2B), o EXT4 assumiu novamente a liderança, consolidando a sua eficiência nas operações de I/O em grande escala, reforçando, assim, a ideia de ser uma boa escolha em ambientes que demandam muitas operações simultâneas, como banco de dados ou sistemas virtualizados. O F2FS e o XFS novamente mantiveram seu ótimo desempenho, não ficando atrás do EXT4 ao gerenciar operações de trabalho intenso e operações simultâneas.

O BTRFS e o ZFS apresentaram, mais uma vez, os piores resultados nos dois testes, confirmando que os recursos de integridade e segurança, comprometem de forma significativa o desempenho, ao gerenciar altas taxas de operação ou muitos arquivos em simultâneo. Isso demonstra que esses sistemas podem ser menos indicados em ambientes de uso intensivo de carga, apesar de suas vantagens, em termos de segurança dos dados.

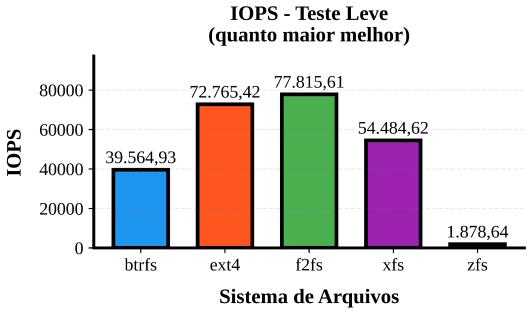


Figura 2 A. Teste leve IOPS

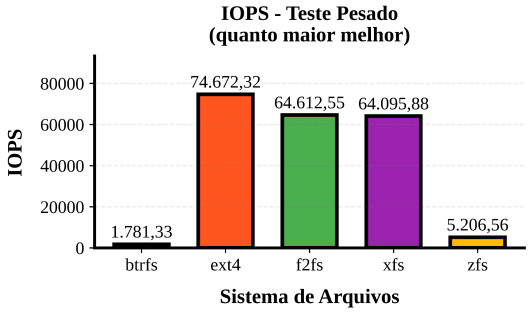


Figura 2 B. Teste pesado IOPS

Prosseguindo para os testes de latência, em carga leve (Figura 3A), o F2FS e o EXT4 apresentaram os menores tempos de resposta, demonstrando novamente a eficiência na rapidez ao gerenciar operações I/O - uma característica essencial para aplicações que requerem uso em tempo real. O XFS também apresentou uma boa latência, mostrando que o gerenciamento de blocos e o uso de *buffer* ajudam a manter um bom tempo de resposta.

Para o testes em carga pesada (Figura 3B), o EXT4, F2FS e o XFS mantiveram seu excelente desempenho, com baixos tempos de latência, mesmo quando estão gerenciando grandes quantidades de dados e sendo submetidos a múltiplas operações simultâneas, sem comprometer a velocidade de resposta - o que é essencial no desempenho, quando muitas aplicações podem estar rodando em paralelo e requerendo uma resposta imediata.

Em seguida, o BTRFS e o ZFS apresentaram altos tempos de latência, principalmente no teste de carga pesada (Figura 3 B); e, o BTRFS, obteve o maior tempo do teste. Este resultado confirma, mais uma vez, que os mecanismos de segurança desses sistemas de arquivos, podem ser prejudiciais quando se precisar de uma resposta mais rápida do sistema, ao lidar com operações de I/O, podendo ser um fator decisivo na hora de se escolher esses sistemas para uso, ainda mais que eles requerem mais recursos de *hardware*.

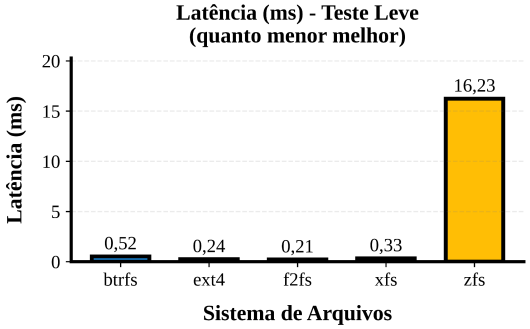


Figura 3A. Teste leve de latência

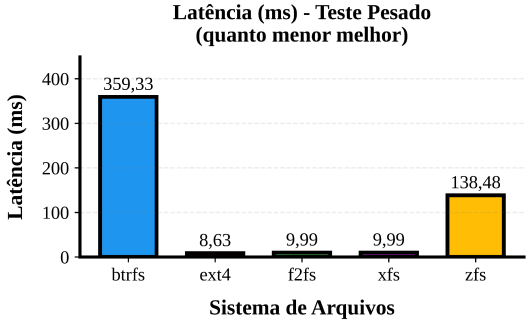


Figura 3B. Teste pesado de latência

5. Considerações Finais

Este estudo apresentou uma análise comparativa entre os sistemas de arquivos *Linux* mais populares: EXT4, XFS, BTRFS, F2FS e ZFS - destacando suas principais características, vantagens e desvantagens. Por meio do uso da ferramenta de *benchmark* FIO, foi possível realizar testes em carga leve e pesada e, a partir de então, avaliar as métricas essenciais para este estudo, como a taxa de transferência (*throughput*), número de operações I/O por segundo (IOPS) e latência.

De acordo com os resultados, deste estudo de caso, o EXT4 pode ser considerado como o sistema mais equilibrado, pois obteve um bom desempenho em todos os testes realizados: baixa latência, altas taxas de IOPS e *throughput* - mostrando-se confiável para diversas situações, como, por exemplo, para uso diário ou empresarial, mesmo que não tenha muitos recursos de segurança, se comparado aos demais sistemas de arquivo. Igualmente, o F2FS se destacou em testes de carga leve e obteve resultados próximos aos do EXT4, em carga pesada, devido ao uso de SSD NVMe, para executar o sistema operacional. Já o XFS também se mostrou muito equilibrado e promissor, principalmente para gerenciar grandes arquivos, embora não tenha superado o EXT4, nos testes de carga pesada.

Por outro lado, embora o BTRFS ofereça recursos avançados, como os *snapshots* e *checksums* (que são de grande valia para a segurança dos dados, evitando corrupções de arquivos), tais recursos impactaram negativamente o desempenho geral do sistema, ainda mais nos testes de cargas pesadas, quando se está trabalhando com grandes volumes de dados. Comparado ao BTRFS, o ZFS possui praticamente os mesmos mecanismos de segurança dele e obteve o pior resultado, em praticamente todos os testes. Esse resultado pode se dar em função da forma com que esse sistema de arquivos foi instalado e testado, sendo possível que performe melhor quando testado em um sistema operacional que já possui suporte nativo.

Em suma, a partir dos resultados apresentados neste estudo comparativo, pode-se sugerir alguns sistemas de arquivos para usos práticos. Por exemplo, para uso cotidiano, o F2FS demonstrou um grande potencial, principalmente quando o equipamento estiver com um dispositivo de memória *flash* - já que o mesmo possui otimização para esse tipo de equipamento. Ademais, o EXT4 se mostra uma opção muito sólida e equilibrada, combinando a sua estabilidade e performance consistente. Já para ambientes de virtualização, servidores ou bancos de dados, onde acontecem muitas operações de arquivos, o EXT4 (como sugerem os testes de *throughput* e IOPS), o XFS e o F2FS se mostraram muito competitivos para essa demanda. Por fim, para aplicações onde se necessita de segurança nos dados (como em sistemas de *backups*) o BTRFS, oferece um conjunto de funcionalidades muito interessantes, como o *COW* e os *snapshots*. No entanto, a escolha ideal, ainda deve-se levar em conta as funcionalidades desejadas no seu ambiente e o foco desse ambiente, seja ele para performance, segurança, estabilidade ou outras cargas mais específicas.

Como perspectivas de trabalhos futuros, sugere-se a realização de testes com outros sistemas de arquivos, em diferentes configurações de *hardware* e com outros sistemas operacionais (como o *Ubuntu* ou *Fedora)*, ou até em sistemas virtualizados, podendo incluir outras métricas como o uso da memória RAM, do disco rígido e do processador. Esses diferentes cenários, poderão fornecer uma visão otimizada sobre o

quanto um sistema de arquivos pode se adaptar ao *hardware*, sendo mais abrangente ou específico.

Referências

ALMEIDA, Théo Rodrigues . **Sistemas de Arquivo**: Análise de Desempenho. Itatiba, São Paulo, Brasil, Dezembro de 2009. Monografia apresentada à disciplina de Conclusão de Curso, do Curso de Engenharia da Computação da Universidade São Francisco.

ARCH LINUX. **File systems** (Português). Disponível em: https://wiki.archlinux.org/title/File_systems_(Portugu%C3%AAs). Acesso em: 9 fev. 2025.

ARCH LINUX. **XFS** (Português). Disponível em: https://wiki.archlinux.org/title/XFS_(Portugu%C3%AAs). Acesso em: 1 fev. 2025.

ARCHZFS. **Archzfs**. Disponível em: https://github.com/archzfs/archzfs. Acesso em: 01 fev. 2025.

BONWICK, Jeff; AHRENS, Matt; HENSON, Val; MAYBEE, Mark; SHELLENBAUM, Mark. **The Zettabyte File System**. In: USENIX CONFERENCE ON FILE AND STORAGE TECHNOLOGIES, 2., 2003, Berkeley. **Proceedings** […]. Berkeley: USENIX, 2003.

CONWAY, Alex *et al.* **File System Aging**. 18 jan. 2024. Disponível em: https://arxiv.org/pdf/2401.08858. Acesso em: 13 out. 2024.

LAUREANO, Gabriel Oliveira. **Esteganografia em sistema de arquivos do sistema operacional Linux**. 2021. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Pontifícia Universidade Católica de Goiás, Goiânia, 2021. Disponível em: https://repositorio.pucgoias.edu.br/jspui/handle/123456789/2767. Acesso em: 13 out. 2025.

LEE, Changman; SIM, Dongho; HWANG, Joo-Young; CHO, Sangyeun. **F2FS: a new file system for flash storage**. In: USENIX CONFERENCE ON FILE AND STORAGE TECHNOLOGIES, 13., 2015, Santa Clara. **Proceedings** […]. Santa Clara: USENIX Association, 2015. p. 273–286.

LIMA, Marcos Laerte Gomes. **Virtualização de laboratórios educacionais com distribuição eficiente de imagens de disco**. 2021. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de São Carlos, São Carlos, 2021.

RODEH, Ohad; BACIK, Josef; MASON, Chris. BTRFS: the Linux B-tree filesystem. **ACM Transactions on Storage**, v. 9, n. 3, p. 1–32, 2013.

SHAIKH, Sohail. **Billion-files File Systems (BfFS)**: A Comparison. 3 ago. 2024. Disponível em: https://arxiv.org/abs/2408.01805. Acesso em: 16 Abr. 2025.

THE LINUX KERNEL DOCUMENTATION. **Filesystems in the Linux kernel**. Disponível em: https://www.kernel.org/doc/html/latest/filesystems/index.html. Acesso em: 18 Jan. 2025.