UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

VINICIUS ARANTES DE SOUZA

AVALIAÇÃO DE MÉTRICAS DE DESEMPENHO ENTRE WSL 2 E LINUX NATIVO NA EXECUÇÃO DE BENCHMARKS CIENTÍFICOS

PATO BRANCO

2025

VINICIUS ARANTES DE SOUZA

AVALIAÇÃO DE MÉTRICAS DE DESEMPENHO ENTRE WSL 2 E LINUX NATIVO NA EXECUÇÃO DE BENCHMARKS CIENTÍFICOS

Evaluation of Performance Metrics in WSL 2 and Native Linux during the Execution of Scientific Benchmarks

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Me. Luis Cassiano Goularte Rista

Coorientador: Prof. Dr. Fábio Favarim

PATO BRANCO

2025

VINICIUS ARANTES DE SOUZA

AVALIAÇÃO DE MÉTRICAS DE DESEMPENHO ENTRE WSL 2 E LINUX NATIVO NA EXECUÇÃO DE BENCHMARKS CIENTÍFICOS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 24/junho/2025

Luis Cassiano Goularte Rista
Mestrado
Universidade Tecnológica Federal do Paraná

Fábio Favarim
Doutorado
Universidade Tecnológica Federal do Paraná

Adriano Serckumecka
Mestrado
Universidade Tecnológica Federal do Paraná

Marcelo Teixeira
Doutorado
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2025

RESUMO

Este trabalho avalia o desempenho do *Windows Subsystem for Linux* versão 2 (WSL 2) em comparação com o ambiente Linux nativo, com foco na execução de aplicações científicas. A crescente adoção do WSL por pesquisadores e desenvolvedores justifica a análise de sua viabilidade sob cargas computacionais intensivas. A metodologia adotada baseou-se na aplicação de três benchmarks sintéticos amplamente reconhecidos — SA-HPL, STREAM e IOzone — para mensurar métricas específicas de desempenho: tempo de execução, operações por segundo (FLOPS), largura de banda de memória e *throughput* de I/O. Os testes foram realizados em *hardware* idêntico, com configurações padronizadas para garantir comparabilidade. Os resultados mostraram que o Linux nativo supera o WSL 2 em todas as métricas avaliadas, especialmente em cenários com alta concorrência e operações de escrita intensiva. O WSL 2, por sua vez, apresentou desempenho satisfatório em cargas leves e operações de leitura, destacando-se pela integração ao Windows e facilidade de uso. Conclui-se que, embora apresente limitações de desempenho, o WSL 2 pode ser uma alternativa viável para fins de ensino, testes e prototipação em ambientes científicos.

Palavras-chave: wsl; aplicações científicas; máquina virtual; benchmarks; desempenho computacional.

ABSTRACT

This work evaluates the performance of Windows Subsystem for Linux version 2 (WSL 2) in comparison with the native Linux environment, with a focus on the execution of scientific applications. The growing adoption of WSL by researchers and developers justifies the analysis of its viability under intensive computational workloads. The adopted methodology was based on the application of three widely recognized synthetic benchmarks — SA-HPL, STREAM, and IOzone — to measure specific performance metrics: execution time, floating-point operations per second (FLOPS), memory bandwidth, and I/O throughput. The tests were conducted on identical hardware, with standardized configurations to ensure comparability. The results showed that native Linux outperforms WSL 2 across all evaluated metrics, especially in scenarios involving high concurrency and intensive write operations. WSL 2, in turn, demonstrated satisfactory performance under light loads and read operations, standing out for its integration with Windows and ease of use. It is concluded that, although it presents performance limitations, WSL 2 can be a viable alternative for educational purposes, testing, and prototyping in scientific environments.

Keywords: wsl; scientific applications; virtual machine; benchmarks; computational performance.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

CPU	Unidade Central de Processamento
FLOPS	*Floating-Point Operations Per Second*
GPU	*Graphics Processing Unit*
HPL	*High Performance Linpack*
I/O	*Input/Output*
LU	*Lower-Upper*
NPB	NAS *Parallel Benchmarks*
NUMA	Acesso Não Uniforme a Memória
SA-HPL	*Self-Adaptive High Performance Linpack*
VM	Máquina Virtual
WSL	*Windows Subsystem for Linux*

SUMÁRIO

1 INTRODUÇÃO

Nesta seção será apresentada uma visão geral do tema, incluindo a contextualização dos principais conceitos, os objetivos do trabalho e a justificativa da escolha do tema. Serão descritas as motivações e contribuições que o estudo trará para a área.

1.1 Considerações iniciais

A execução de aplicações científicas frequentemente demanda alto poder computacional, seja para simulações numéricas, modelagens estatísticas ou análise de grandes volumes de dados. Esses cenários exigem não apenas capacidade de processamento da *Unidade Central de Processamento (CPU)* , mas também uma utilização eficiente da memória e do sistema de arquivos (RISTA *et al.*, 2017). Embora tais aplicações tenham sido historicamente associadas a *clusters* e supercomputadores, o uso de estações de trabalho pessoais para prototipagem, testes e execução local tem se tornado cada vez mais comum, especialmente em ambientes acadêmicos ou de pesquisa aplicada (XAVIER *et al.*, 2013).

Nesse contexto, tecnologias de virtualização leve têm se destacado por oferecerem ambientes flexíveis, com menor sobrecarga de sistema e maior praticidade, em comparação a soluções tradicionais de virtualização. Entre essas tecnologias, destaca-se o *Windows Subsystem for Linux (WSL)*, desenvolvido pela Microsoft, que permite a execução de distribuições Linux diretamente no ambiente Windows, sem a necessidade de uma máquina virtual completa. Sua proposta é oferecer compatibilidade com ferramentas do ecossistema Linux em um ambiente nativo do Windows, favorecendo a integração entre sistemas e facilitando o trabalho de desenvolvedores e pesquisadores (MICROSOFT, 2022).

Estudos exploraram como diferentes abordagens de virtualização afetam o desempenho de aplicações paralelas e intensivas em recursos. Al-hamouri *et al.* (2020) analisaram o impacto de virtualização em tarefas *multithread* e identificaram que a sobrecarga do sistema pode limitar a escalabilidade em cenários científicos. De forma semelhante, Walters *et al.* (2008) compararam contêineres e máquinas virtuais e demonstraram variações significativas de desempenho, principalmente em operações de memória e rede. Tais resultados reforçam a necessidade de se entender o comportamento de soluções híbridas como o WSL em contextos computacionalmente exigentes.

No entanto, apesar da crescente adoção do WSL, ainda há incertezas quanto à sua viabilidade para aplicações científicas que impõem grande demanda de recursos. Embora trabalhos como o de Kochberger, Tauber e Schrittwieser (2019) tenham investigado aspectos de desempenho e compatibilidade, não há uma avaliação sistemática que compare o WSL diretamente com ambientes Linux nativos sob cargas de aplicações científicas. Essa lacuna justifica a condução de estudos experimentais mais aprofundados, com foco em desempenho sob diferentes intensidades de carga e nas características específicas de aplicações científicas.

Dada essa lacuna, torna-se necessário não apenas comparar diretamente o WSL com o Linux nativo, mas também aplicar ferramentas que permitam mensurar com precisão o desempenho sob diferentes tipos de carga. Nesse sentido, *benchmarks* sintéticos amplamente utilizados na comunidade científica se mostram adequados por oferecerem métricas padronizadas para avaliar aspectos distintos do sistema, como processamento numérico, largura de banda de memória e desempenho de entrada e saída em disco.

1.2 Objetivos

Esta seção apresenta o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo geral

Avaliar o desempenho do WSL 2 em comparação com ambientes Linux nativos na execução de aplicações científicas, analisando seu comportamento sob diferentes cargas de trabalho e estabelecendo recomendações de uso com base nas métricas obtidas.

1.2.2 Objetivos específicos

- Definir métricas de desempenho relevantes e aplicar um conjunto de *benchmarks* representativos em ambos os ambientes;
- Coletar e analisar as métricas obtidas (CPU, memória e I/O), comparando os resultados do WSL 2 com os do Linux nativo;
- Avaliar o comportamento do WSL 2 sob diferentes intensidades de carga, identificando seus limites de desempenho;
- Elaborar recomendações sobre a viabilidade de uso do WSL 2 em aplicações científicas, com base nos dados experimentais.

1.3 Justificativa

A crescente popularidade do WSL, que permite a execução de distribuições Linux de forma integrada ao ambiente Windows, levanta questionamentos sobre sua viabilidade em aplicações científicas que exigem alto desempenho computacional. Em contextos acadêmicos e corporativos nos quais o Windows é o sistema predominante, o uso do WSL pode representar uma alternativa prática à instalação de sistemas *dual-boot* ou máquinas virtuais tradicionais. No entanto, há escassez de estudos sistemáticos que avaliem suas limitações e capacidades em cenários reais de carga.

Este trabalho busca preencher essa lacuna ao analisar, por meio de métricas padronizadas, o comportamento do WSL 2 em comparação com o Linux nativo, oferecendo subsídios técnicos para decisões mais informadas sobre sua adoção em ambientes de ensino, pesquisa e prototipação. A pesquisa se insere no contexto de computação de alto desempenho e virtualização leve, contribuindo para o entendimento dos *trade-offs* envolvidos no uso do WSL em aplicações científicas.

1.4 Estrutura do trabalho

Este trabalho está organizado em sete capítulos, estruturados de forma a proporcionar uma compreensão clara e sistemática do tema proposto e descritos a seguir:

- Introdução: apresenta o contexto do estudo, os objetivos geral e específicos, a justificativa para a realização da pesquisa e a delimitação do problema abordado;
- Referencial Teórico: discute os principais conceitos relacionados à virtualização, ao WSL e aos *benchmarks* utilizados (SA-HPL, STREAM e IOzone), fornecendo a base teórica para a análise comparativa;
- Trabalhos Relacionados: revisa estudos anteriores que abordam desempenho em ambientes virtualizados, destacando lacunas que justificam a realização deste trabalho;
- Abordagem Proposta: descreve os ambientes de teste, os procedimentos metodológicos adotados e as métricas utilizadas para a avaliação do desempenho nos dois sistemas;
- Resultados: apresenta e analisa os dados obtidos por meio da execução dos *benchmarks*, comparando o comportamento do WSL 2 e do Linux nativo com base em CPU, memória e I/O;
- Limitações do Trabalho: discute as particularidades e considerações metodológicas que delimitam o escopo dos resultados, contextualizando a interpretação dos dados;
- Conclusão: sintetiza os principais achados do estudo, verifica o alcance dos objetivos propostos e apresenta recomendações e perspectivas para trabalhos futuros;
- Referências: lista as fontes bibliográficas utilizadas ao longo do desenvolvimento da pesquisa, assegurando o embasamento teórico e científico do trabalho.

2 REFERENCIAL TEÓRICO

Nesta seção, são apresentados os fundamentos teóricos necessários para embasar o estudo, incluindo o conceito de máquinas virtuais, uma descrição detalhada do WSL e a explicação das ferramentas de *benchmark* utilizadas: *Self-Adaptive High Performance Linpack* (SA-HPL), STREAM e IOzone. Por fim, são abordados os trabalhos relacionados, destacando estudos que contribuem para o contexto e o desenvolvimento deste trabalho.

2.1 Máquinas Virtuais

Máquina Virtual (VM) é uma tecnologia amplamente utilizada para simular múltiplos sistemas operacionais em um único equipamento físico. Essa virtualização é viabilizada por meio de *hypervisors*, *softwares* responsáveis por criar e gerenciar VMs utilizando os recursos do sistema *host*. Existem dois tipos principais de *hypervisors*: Tipo 1, que opera diretamente no *hardware* físico (bare-metal), oferecendo maior eficiência para cargas computacionais exigentes; e Tipo 2, que funciona sobre um sistema operacional, sendo mais acessível para testes, desenvolvimento e ambientes educacionais (ARYOTEJO *et al.*, 2021).

Hypervisors de Tipo 1 são preferidos em contextos que exigem desempenho computacional mais elevado, como em aplicações científicas com uso intensivo de CPU e memória. Esse tipo de *hipervisor* é executado diretamente sobre o *hardware*, sem a necessidade de um sistema operacional *host*. Um exemplo utilizado é o VMware ESXi, componente da suíte vSphere, que atua como sistema operacional mínimo e oferece controle direto sobre os recursos físicos da máquina (VMWARE, 2025). Já os *hypervisors* de Tipo 2, como o VirtualBox — um *software* de virtualização completo voltado para *desktop*, *notebooks*, servidores e embarcados (ORACLE, 2025) — são instalados sobre um sistema operacional e oferecem maior simplicidade de uso. No entanto, por dependerem da camada adicional do *host*, tendem a apresentar desempenho inferior em tarefas intensivas quando comparados aos de Tipo 1 (ARYOTEJO *et al.*, 2021).

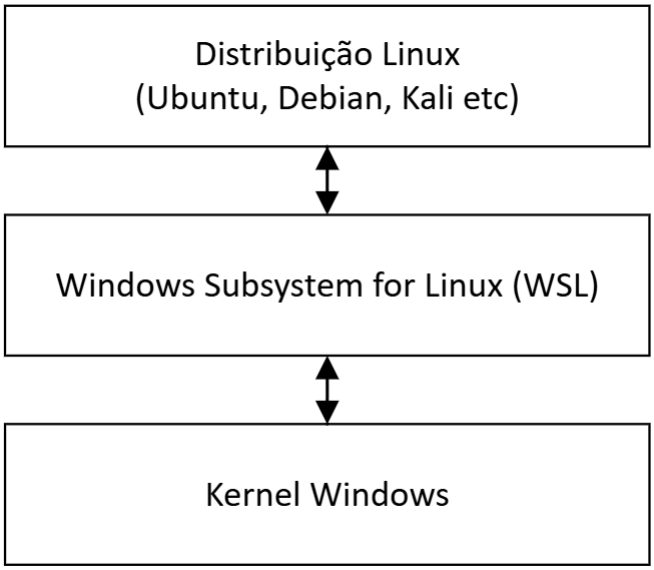
O principal benefício das VMs é a capacidade de executar diversos sistemas operacionais ou aplicações em um único *hardware*, reduzindo custos e aumentando a flexibilidade. Essa funcionalidade é fundamental para soluções modernas como a computação em nuvem e o desenvolvimento multiplataforma, além de permitir escalabilidade e isolamento de recursos. As VMs também facilitam o aprendizado remoto, como em cursos de redes de computadores, permitindo que estudantes utilizem seus próprios dispositivos para acessar ambientes virtualizados complexos sem a necessidade de laboratórios físicos (POKHARANA; GUPTA, 2023).

A compreensão dos princípios de virtualização é essencial para este estudo, uma vez que o WSL adota uma abordagem alternativa que combina elementos de compatibilidade entre sistemas operacionais e execução em nível de virtualização leve. A próxima seção abordará os detalhes técnicos e arquitetônicos do WSL, com ênfase em seu funcionamento e implicações para aplicações científicas.

2.2 WSL

O WSL é um recurso desenvolvido pela Microsoft que permite a execução de distribuições Linux diretamente no Windows. Diferentemente de máquinas virtuais tradicionais, o WSL utiliza uma camada de tradução de chamadas de sistema para executar binários Linux no ambiente Windows, sem a necessidade de inicialização dupla ou uso de um *hypervisor*, conforme ilustrado na Figura 1. Essa abordagem visa fornecer uma experiência integrada e produtiva, especialmente para desenvolvedores que trabalham em ambientes híbridos (MICROSOFT, 2022).

Figura 1 – Arquitetura do WSL



Fonte: Adaptado de Developer (2020).

O WSL funciona como um aplicativo do Windows que permite aos desenvolvedores executar ambientes Linux no mesmo computador, sem a necessidade de instalar um segundo sistema operacional ou uma máquina virtual. Essa funcionalidade facilita a troca de dados entre programas que operam nos dois sistemas e possibilita a execução de diversos *scripts* Linux diretamente no Windows (ZUNIN; STEMPKOVSKY; SOLOVYEV, 2024).

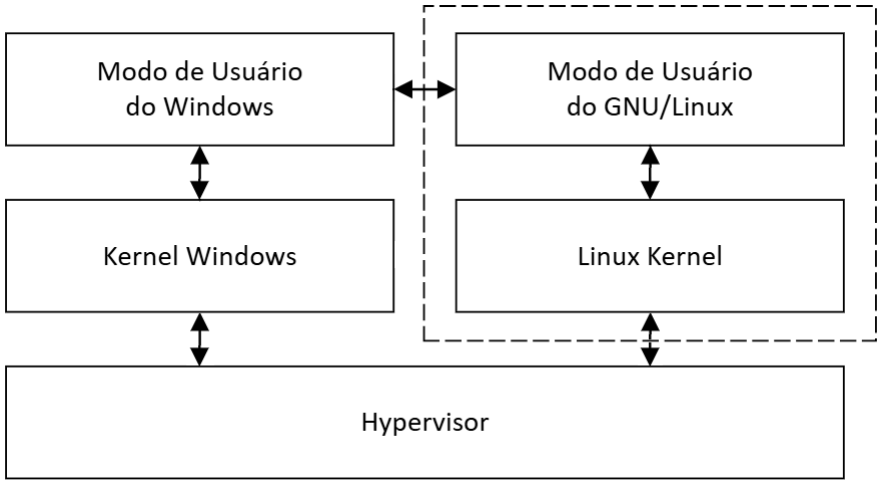
De acordo com a documentação oficial (MICROSOFT, 2022), o WSL oferece diversas funcionalidades, incluindo:

- Instalação e execução de distribuições Linux (como Ubuntu, Debian e Kali) diretamente a partir da Microsoft Store;
- Execução de ferramentas de linha de comando comuns do Linux;
- Suporte a linguagens de programação como Python, Node.js, C++ e Rust;
- Execução de aplicativos gráficos Linux integrados à interface do Windows;

- Suporte ao uso da *Graphics Processing Unit (GPU)* para aceleração de cargas de trabalho computacionais, como aprendizado de máquina, modelagem científica e análise de dados.

A versão WSL 2, lançada posteriormente, introduziu mudanças estruturais significativas, incluindo a adoção de um *kernel* Linux completo, mantido pela própria Microsoft. Essa mudança aumentou substancialmente a compatibilidade e o desempenho, especialmente em tarefas que envolvem operações de I/O intensivo e acesso direto ao sistema de arquivos. Diferentemente da primeira versão, o WSL 2 utiliza uma máquina virtual leve para executar esse *kernel*, o que garante maior eficiência em cenários computacionais exigentes (AL-HAMOURI *et al.*, 2020). A Figura 2 exemplifica a arquitetura utilizada no WSL 2.

Figura 2 – Arquitetura do WSL 2



Fonte: Adaptado de Developer (2020).

O WSL 2 oferece desempenho superior em operações como atualização de pacotes, compilação de código e manipulação de arquivos, superando em diversos aspectos o seu antecessor. Estudos demonstram que, em determinadas tarefas de entrada/saída, o WSL 2 pode ser até 20 vezes mais rápido do que sua versão anterior (AL-HAMOURI *et al.*, 2020). No entanto, essa melhoria vem acompanhada de um maior consumo de memória, especialmente em sessões prolongadas ou manipulação de grandes volumes de dados (MICROSOFT, 2022).

Com essas melhorias, o WSL 2 torna-se uma ferramenta robusta não apenas para desenvolvedores, mas também para pesquisadores e profissionais que executam aplicações científicas no ambiente Windows. Sua arquitetura híbrida, aliando a flexibilidade do Linux à integração com o sistema operacional Windows, viabiliza o uso do subsistema em atividades como simulações numéricas, análise computacional de dados e outras tarefas típicas de aplicações científicas (AL-HAMOURI *et al.*, 2020).

Este trabalho utiliza exclusivamente o WSL 2, por introduzir mudanças estruturais importantes em relação à versão anterior. No entanto, ao longo do texto, para fins de simplificação e padronização visual, será utilizada a nomenclatura WSL para se referir ao WSL 2.

2.3 Benchmark

Um *benchmark* é definido como um conjunto de testes padronizados utilizados para avaliar e comparar o desempenho de sistemas computacionais. Ele serve como ferramenta essencial para identificar capacidades, limitações e gargalos de *hardware* e *software*. Esses testes podem medir diferentes aspectos de um sistema, incluindo processamento, memória e desempenho de *Input/Output (I/O)* (JAIN, 1991).

De acordo com Jain (1991), *benchmarks* podem ser divididos em diferentes categorias, como programas sintéticos, que simulam cargas de trabalho reais; *kernels*, que representam partes essenciais de aplicações maiores; e *benchmarks* de aplicação, que incluem funções de sistemas reais como bancos ou sistemas de reservas de companhias aéreas. A escolha de qual tipo utilizar depende dos objetivos da análise e do sistema a ser avaliado. A importância dos *benchmarks* está no fato de que permitem realizar medições replicáveis e controladas, fundamentais para análises comparativas de desempenho. Além disso, eles possibilitam a experimentação em diferentes configurações de *hardware* e *software*, facilitando a tomada de decisão em projetos de alto desempenho computacional.

Diversas ferramentas de *benchmark* são utilizadas na avaliação de desempenho computacional, entre elas o NAS *Parallel Benchmarks* (NPB), Sysbench, Geekbench e Phoronix Test Suite. O NPB é aplicado em avaliações de desempenho em ambientes paralelos (XAVIER *et al.*, 2013), o Geekbench foi utilizado por Lin, Barker e Thomson (2020) em medições de desempenho de máquinas virtuais em nuvem, enquanto o Sysbench foi empregado por Pokharana e Gupta (2023) em testes comparativos entre VMs em *hypervisores* do tipo 2. O Phoronix Test Suite, por sua vez, foi adotado por Dordevic, Jovanovic e Timcenko (2014) em estudos de desempenho em plataformas de nuvem como AWS e Azure.

Neste trabalho, foram selecionados os *benchmarks* SA-HPL, STREAM e IOzone por sua natureza de código aberto, ampla aceitação na comunidade científica e aderência às necessidades do estudo, sendo responsáveis por analisar o desempenho do CPU, memória e sistemas de arquivo, respectivamente.

Esses *benchmarks* foram escolhidos por abordarem aspectos complementares do desempenho computacional, em cenários sujeitos a variabilidades comuns em aplicações científicas, como desbalanceamento de carga de CPU, limitações de largura de banda da memória e intensidade variável de operações de entrada e saída. Juntas, essas condições tornam desafiador antecipar o desempenho, e a combinação dessas três ferramentas permite uma avaliação mais abrangente e realista.

2.3.1 SA-HPL

O SA-HPL é uma versão aprimorada do *benchmark High Performance Linpack* (HPL), projetada para avaliar o desempenho de sistemas paralelos de memória compartilhada. Diferen-

temente do HPL tradicional, que exige ajustes manuais complexos, o SA-HPL introduz recursos auto-adaptativos que otimizam a execução em arquiteturas *multicore*. Essa evolução simplifica o processo de configuração, reduzindo o esforço necessário para ajustar parâmetros como o tamanho de blocos e a distribuição de tarefas (RISTA; TEIXEIRA; FONSECA, 2024).

Uma das principais características do SA-HPL é o uso de escalonadores dinâmicos para gerenciar o paralelismo de tarefas, permitindo que o *benchmark* se adapte automaticamente às condições de carga do sistema. Essa abordagem melhora a utilização dos recursos de *hardware*, reduzindo gargalos e aumentando a eficiência global. Além disso, o SA-HPL utiliza a biblioteca Eigen — uma biblioteca C++ otimizada para computação matricial e álgebra linear (GUENNEBAUD G., 2024) — para realizar fatorações *Lower-Upper (LU)*, dividindo os sistemas lineares em subproblemas menores que podem ser resolvidos paralelamente (RISTA; TEIXEIRA; FONSECA, 2024).

Os resultados obtidos com o SA-HPL demonstram sua superioridade em relação ao HPL tradicional. De acordo com Rista, Teixeira e Fonseca (2024) os ganhos foram especialmente evidentes em sistemas com carga de trabalho desbalanceada, em que os escalonadores dinâmicos mostraram-se particularmente eficazes.

2.3.2 STREAM

O STREAM é um *benchmark* sintético amplamente utilizado para medir o desempenho sustentável de largura de banda de memória (em MB/s) e a taxa correspondente de operações computacionais em *kernels* de vetor simples. Foi projetado especificamente para avaliar sistemas de memória, ignorando efeitos transitórios de *cache*, e simular cenários reais encontrados em aplicações científicas e computacionais de alto desempenho (MCCALPIN, 2016).

A relevância do STREAM reside na crescente disparidade entre a velocidade de processamento das CPUs e a capacidade dos sistemas de memória. Essa diferença resulta em situações em que a limitação de desempenho está relacionada à largura de banda de memória, e não à capacidade computacional do processador. Para abordar esse desafio, o STREAM opera com conjuntos de dados significativamente maiores que o cache disponível em qualquer sistema, garantindo resultados representativos de aplicações intensivas em transferência de dados (MCCALPIN, 2016).

Segundo McCalpin (2016), o *benchmark* executa quatro operações principais:

- COPY (cópia de valores entre vetores);
- SCALE (cópia com escala);
- ADD (soma de dois vetores em um terceiro);
- TRIAD (soma e escala combinadas).

Essas operações refletem os padrões de acesso à memória comuns em cargas de trabalho científicas e são fundamentais para avaliar a eficiência de acesso à memória em sistemas modernos.

2.3.3 IOzone

O IOzone é uma ferramenta de *benchmark* amplamente utilizada para testar e analisar o desempenho de sistemas de arquivos, medindo o desempenho em operações como leitura, escrita, re-leitura, reescrita, leitura aleatória e escrita assíncrona. Essa variedade de testes permite avaliar tanto o *throughput* quanto a latência em diferentes condições operacionais, tornando-o especialmente útil para identificar gargalos e otimizar a eficiência de plataformas de armazenamento. Sua abrangência o torna indispensável em ambientes computacionais que exigem alto desempenho, como na execução de aplicações científicas intensivas em entrada e saída de dados (IOZONE, 2016).

De acordo com Zhang e Nie (2016), entre as características avançadas do IOzone, destaca-se sua capacidade de realizar medições em operações como escrita assíncrona, múltiplos fluxos e testes de servidores distribuídos. Ele também é compatível com sistemas de 64 bits e oferece recursos para gerar gráficos interpretáveis em ferramentas como o Excel. Essas funcionalidades tornam a ferramenta essencial para análises detalhadas de sistemas de arquivos em ambientes computacionais que exigem alta performance, como na execução de cargas intensivas de leitura e escrita.

3 TRABALHOS RELACIONADOS

O trabalho de Kochberger, Tauber e Schrittwieser (2019) é um dos poucos estudos que analisam diretamente o WSL, avaliando sua transparência e desempenho em comparação com sistemas Linux nativos, como o Ubuntu. Utilizando *benchmarks* específicos, o artigo examina a tradução de chamadas de sistema e a eficiência geral do subsistema. Embora o foco principal esteja na identificação do ambiente de execução, o estudo oferece uma visão aprofundada sobre os desafios enfrentados pelo WSL ao buscar compatibilidade com desempenho.

Al-hamouri *et al.* (2020) exploram como a virtualização afeta o desempenho de aplicações paralelas com uso intensivo de *threads*. A análise se baseia em métricas como latência, uso de CPU e eficiência *multithread*. As conclusões do artigo reforçam que ambientes virtualizados podem introduzir limitações em aplicações sensíveis à sincronização e ao *throughput* de memória — um ponto importante na análise de desempenho do WSL.

Xavier *et al.* (2013) apresentam uma análise detalhada do desempenho de virtualização baseada em contêineres em comparação com sistemas bare-metal. Por meio de *benchmarks* como LINPACK, STREAM e IOzone, os autores demonstram que, em determinadas cargas computacionais, os contêineres podem alcançar desempenho próximo ao nativo. A discussão sobre uso de CPU, eficiência de memória e operações de I/O é especialmente relevante para este estudo, uma vez que o WSL compartilha características com soluções baseadas em contêineres, como leveza e sobrecarga reduzida.

Regola e Ducom (2010) fornecem recomendações práticas para o uso de virtualização em contextos computacionais exigentes. Embora o estudo seja anterior ao WSL, ele aborda tecnologias de virtualização como máquinas virtuais, contêineres e soluções híbridas, destacando a dificuldade de equilibrar flexibilidade e desempenho. Tais desafios continuam presentes no WSL, que busca entregar uma experiência Linux integrada ao Windows, com desempenho aceitável para aplicações técnicas.

Walters *et al.* (2008) compara diversas tecnologias de virtualização, incluindo máquinas virtuais e contêineres, avaliando seu impacto no desempenho em ambientes de alto processamento. A análise envolve *benchmarks* de memória, CPU e rede, e evidencia como configurações de *hardware* e arquitetura de virtualização influenciam diretamente os resultados. O estudo é relevante por oferecer uma base comparativa ampla que ajuda a entender o posicionamento técnico do WSL em relação a soluções mais tradicionais.

Além dos estudos comparativos de desempenho, este trabalho também se fundamenta em propostas específicas de *benchmarks* desenvolvidos para avaliação de sistemas *multicore*. Rista, Teixeira e Fonseca (2024) propõem o SA-HPL, uma versão auto-adaptativa do *benchmark* HPL voltada à avaliação de arquiteturas paralelas com memória compartilhada. O *benchmark* utiliza escalonamento dinâmico para ajustar a execução conforme as características do sistema, simplificando a configuração e permitindo mensurar tempo de execução, *throughput* e eficiência de paralelismo de forma otimizada.

No contexto da avaliação de subsistemas de memória, destaca-se o trabalho de McCalpin (1995), que introduz o *benchmark* STREAM, amplamente utilizado para medir a largura de banda de memória de forma sustentável, com base em operações vetoriais simples. O estudo define métricas como eficiência de uso da memória e balanço de máquina, utilizadas neste trabalho para avaliar o desempenho do subsistema de memória principal.

Para a análise do subsistema de entrada e saída (I/O), este estudo baseia-se em diferentes pesquisas da literatura, entre as quais se destaca o trabalho de Diaz *et al.* (2014). Os autores utilizam o IOzone para avaliar o desempenho de I/O em ambientes virtualizados, considerando diferentes níveis de concorrência. O estudo demonstra que o *benchmark* é sensível a variações de infraestrutura, sendo eficaz para identificar gargalos em operações de leitura e escrita, o que justifica sua aplicação neste trabalho.

Os trabalhos relacionados apresentados abordam temas fundamentais como virtualização, desempenho computacional e o uso do WSL em diferentes cenários de execução, além de embasar as métricas de avaliação utilizadas neste estudo. No entanto, nenhuma dessas abordagens realiza uma comparação direta e abrangente entre o WSL e sistemas Linux nativos no contexto de aplicações científicas com alta demanda de recursos. Este trabalho busca preencher essa lacuna ao conduzir uma análise específica e detalhada, utilizando ferramentas reconhecidas como SA-HPL, STREAM e IOzone para mensurar largura de banda de memória, desempenho de CPU e latência de I/O de forma sistemática.

4 ABORDAGEM PROPOSTA

Este capítulo apresenta os recursos utilizados, a configuração dos ambientes de teste e a metodologia aplicada para avaliar o desempenho do WSL em comparação com o Linux nativo, com foco na execução de aplicações científicas.

4.1 Ambientes e Ferramentas Utilizadas

Os experimentos foram realizados em uma estação de trabalho com as seguintes especificações:

- Processador: AMD Ryzen 7 5700X (8 núcleos / 16 *threads*);
- Memória RAM: 32 GB DDR4, 3200 MHz;
- Armazenamento: SSD NVMe de 1 TB;
- Sistema Operacional *Host*: Windows 11 Education – Compilação 26100.4061.

Foram configurados dois ambientes de teste sobre o mesmo hardware:

- Linux nativo, com Ubuntu 22.04 LTS instalado em *dual-boot*;
- WSL 2, com Ubuntu 22.04 LTS instalado via Microsoft PowerShell.

Optou-se por utilizar a versão 22.04 LTS em ambos os ambientes, mesmo havendo uma versão mais recente disponível (24.04 LTS), a fim de evitar possíveis interferências de compatibilidade ou instabilidade decorrentes de atualizações recentes. Essa decisão visou garantir maior previsibilidade e comparabilidade entre os resultados obtidos nos dois sistemas. Ambos os sistemas utilizaram as mesmas versões de bibliotecas, pacotes e ferramentas, com o objetivo de manter um ambiente de comparação justo e equilibrado.

Os *benchmarks* utilizados foram:

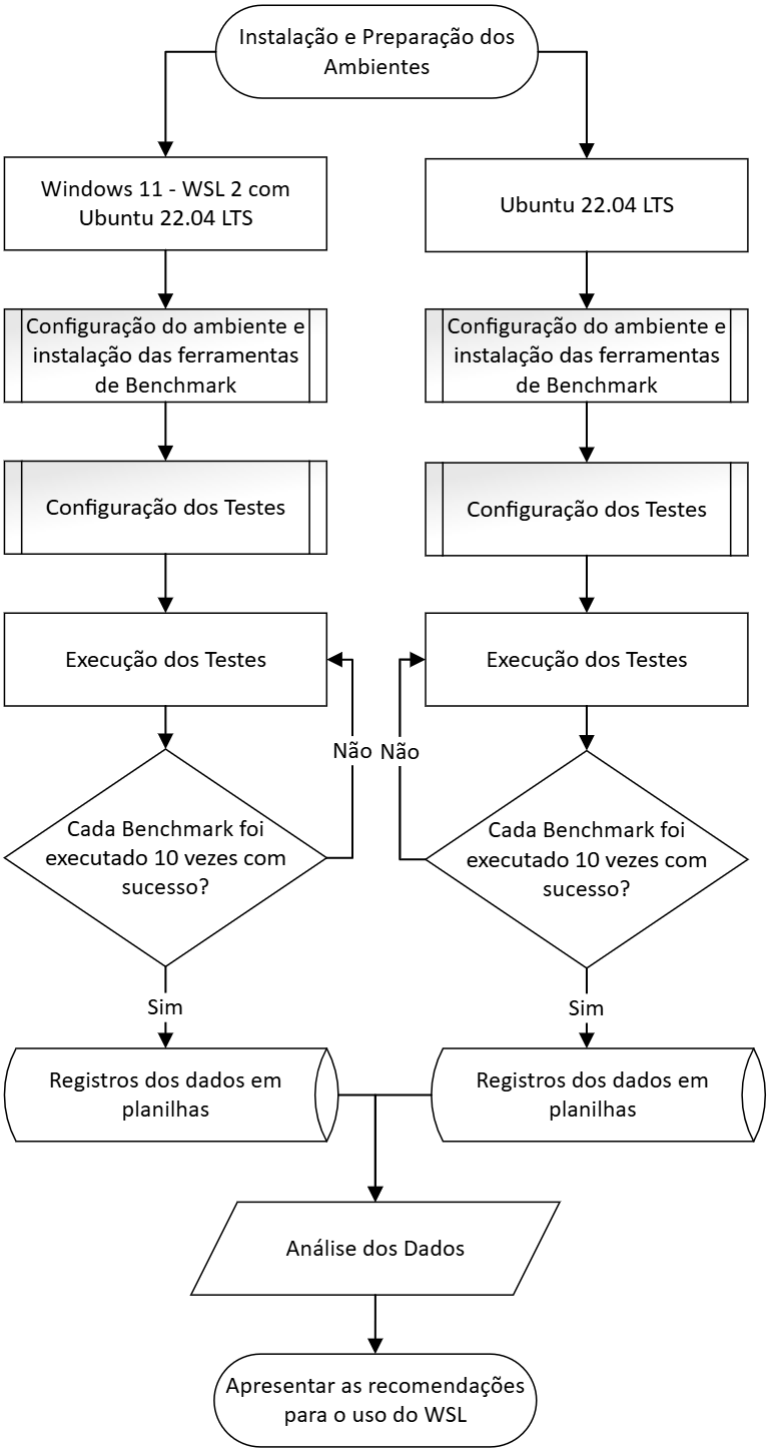
- SA-HPL, para medir o desempenho computacional em cálculos numéricos (CPU);
- STREAM, para avaliar a largura de banda de memória;
- IOzone, para medir o desempenho do sistema de arquivos (entrada e saída em disco).

Todos os *benchmarks* são de código aberto, amplamente utilizados em pesquisas científicas e foram obtidos diretamente de seus repositórios oficiais. As distribuições Linux (tanto no WSL quanto no ambiente nativo) foram previamente atualizadas com os comandos de atualização padrão do sistema.

4.2 Procedimentos Metodológicos

Esta seção descreve as configurações aplicadas aos ambientes de teste, os critérios de execução dos *benchmarks* e os procedimentos adotados para registro e organização dos dados. A Figura 3 ilustra a metodologia utilizada.

Figura 3 – Fluxograma



Fonte: Elaborado pelo autor (2025).

A Figura 3 apresenta uma visão geral da metodologia adotada, organizada em etapas que estruturam desde a preparação dos ambientes até a análise dos dados e formulação das recomendações finais. Cada uma das etapas ilustradas no fluxograma é detalhada a seguir.

A instalação e configuração dos ambientes de teste, incluindo o sistema operacional e ferramentas de *benchmark*, corresponde à seção 4.1. A configuração dos testes, bem como os ajustes de desempenho e controle do ambiente, são descritos nas seções 4.2.1 e 4.2.2. O processo de execução automatizada dos *benchmarks* com dez repetições é abordado também nessas seções, enquanto o registro e organização dos dados é tratado na seção 4.2.3. A análise estatística e visualização dos resultados são aprofundadas nos capítulos 5 e 7, onde se encontram a interpretação dos dados e as recomendações para uso do WSL.

4.2.1 Ambiente Linux Nativo

No Ubuntu, o modo de desempenho força a CPU a operar continuamente em sua frequência máxima, mesmo em situações de baixa carga, ao contrário dos modos padrão equilibrado ou economia de energia, que reduzem dinamicamente o *clock* para poupar energia e diminuir o aquecimento (UBUNTU, 2025). Como o Ubuntu costuma priorizar economia de energia, ele não mantém automaticamente a frequência máxima, embora esse seja o comportamento desejado em contextos de *benchmark*, no qual buscamos máxima previsibilidade e desempenho sustentado.

Neste sentido, para garantir que a CPU atue sempre em sua capacidade plena, foram aplicadas configurações que definem o modo de desempenho como *performance*, desabilitando o serviço padrão do Ubuntu. Além disso, o *kernel* do sistema foi atualizado para uma versão mais recente que inclui suporte completo ao processador AMD Ryzen 5700X, especialmente em relação ao *driver* de escalonamento de frequência. Isso assegura que todas as frequências operacionais sejam reconhecidas corretamente e aplicadas conforme o modo configurado, evitando limitações causadas por suporte incompleto no *kernel* anterior.

Para conduzir os testes, foi utilizado um *script* desenvolvido especificamente para este trabalho, responsável por aplicar o *clock* máximo a todos os *threads* da CPU antes de cada execução, iniciar os *benchmarks* de forma sequencial e repetir cada teste dez vezes consecutivas. Essa automação garantiu repetibilidade, controle total do ambiente e eliminação de interferências manuais durante as medições.

Todo o processo foi realizado em ambiente texto, com o sistema inicializado sem interface gráfica ou serviços desnecessários, garantindo que apenas os processos do sistema operacional e dos *benchmarks* estivessem ativos, minimizando ruídos que pudessem afetar os resultados das medições.

4.2.2 Ambiente WSL 2

No ambiente WSL 2, o sistema Windows 11 foi configurado para utilizar o plano de energia Alto Desempenho, com o objetivo de evitar o escalonamento dinâmico da frequência da CPU durante a execução dos testes. Essa medida é essencial, pois o gerenciamento de energia do Windows pode reduzir automaticamente o *clock* do processador sob demanda, o que comprometeria a consistência dos resultados em cargas computacionais repetitivas.

Mantendo a padronização metodológica entre os dois ambientes, o mesmo *script* desenvolvido para o Linux nativo foi adaptado ao WSL 2. Esse *script* automatizou a execução sequencial dos *benchmarks*, com dez repetições consecutivas para cada ferramenta. Assim como no Ubuntu, sua utilização garantiu repetibilidade, controle do ambiente e eliminação de qualquer interferência manual durante as medições.

Durante os testes, apenas o terminal do WSL foi mantido ativo, com todos os demais aplicativos do Windows encerrados, de modo a minimizar a interferência de processos em segundo plano e garantir maior controle sobre o ambiente de execução. Esse conjunto de ações visou aproximar as condições operacionais do WSL àquelas aplicadas no Linux nativo, assegurando um cenário de comparação justo e tecnicamente equilibrado.

4.2.3 Registro e Análise dos Dados

Após a execução dos testes em ambos os ambientes, os resultados gerados pelos *benchmarks* foram armazenados em arquivos de texto. Esses arquivos continham os valores brutos referentes ao *throughput*, tempo de execução e demais métricas associadas a cada *benchmark*.

Os dados foram processados por meio de *scripts* em Python, que realizaram a leitura, filtragem e organização das informações relevantes. Em seguida, os resultados foram consolidados em planilhas Excel, permitindo uma estrutura padronizada para análise posterior.

As planilhas foram importadas para o Power BI, ferramenta utilizada para o cálculo das medidas estatísticas e a construção dos gráficos comparativos entre os ambientes testados. Esse processo permitiu uma visualização clara das diferenças de desempenho e contribuiu diretamente para a interpretação dos resultados.

A análise dos dados, apresentada no capítulo 5, foi fundamentada nas métricas descritas na seção 4.3, que estabelece critérios utilizados para mensurar o desempenho dos *benchmarks* por ambiente. Dessa forma, a seção 4.3 complementa metodologicamente esta etapa, oferecendo os parâmetros conceituais que guiaram a avaliação comparativa dos testes.

4.3 Métricas de Desempenho

As métricas avaliadas foram definidas conforme a natureza de cada ferramenta de *benchmark*, com foco em identificar características de desempenho associadas à execução de aplicações científicas.

4.3.1 SA-HPL

O *benchmark* SA-HPL foi utilizado neste trabalho com o objetivo de avaliar o desempenho de sistemas *multicore* em operações intensivas de ponto flutuante, considerando diferentes tamanhos de matriz e níveis de paralelismo (RISTA; TEIXEIRA; FONSECA, 2024). As métricas selecionadas estão descritas nas seções a seguir.

4.3.1.1 Tempo Total de Execução

Corresponde ao tempo necessário para resolver um sistema de equações lineares. No SA-HPL, é realizado por meio da decomposição LU com pivotamento parcial, utilizando a biblioteca Eigen (GUENNEBAUD G., 2024). Dessa forma, o tempo de execução reflete diretamente o desempenho da CPU em operações de álgebra linear sob diferentes níveis de paralelismo.

4.3.1.2 *Floating-Point Operations Per Second* (FLOPS)

A métrica de *Floating-Point Operations Per Second* (FLOPS) representa o número de operações de ponto flutuante realizadas por segundo. No SA-HPL, esse valor é calculado com base na resolução do sistema linear via decomposição LU, considerando o tempo de execução e a complexidade algébrica da operação. Trata-se de uma medida direta da capacidade computacional da CPU em cenários com alta demanda matemática, frequentemente utilizada em *benchmarks* científicos.

4.3.1.3 Eficiência de Processamento e Paralelismo

Avalia o desempenho da aplicação em ambientes *multicore* ao utilizar múltiplas *threads*. O *speed-up* representa o ganho obtido com a paralelização, calculado pela razão entre o tempo de execução com uma *thread* (T_1) e o tempo com n *threads* (T_n), apresentado na Equação (1):

$$\text{Speed-Up} = \frac{T_1}{T_n} \tag{1}$$

A partir do *speed-up*, calcula-se a eficiência de paralelismo, que indica o aproveitamento dos recursos paralelos disponíveis. É definida de acordo com a Equação (2):

$$\text{Eficiência} = \frac{\text{Speed-Up}(n)}{n} \tag{2}$$

Essas métricas permitem avaliar a escalabilidade da aplicação e identificar perdas de desempenho decorrentes de *overhead* de sincronização, divisão desigual de carga ou limitações da arquitetura. No SA-HPL, a eficiência está relacionada ao comportamento dos escalonadores dinâmicos durante a distribuição das tarefas entre os núcleos.

4.3.2 STREAM

O STREAM é utilizado para avaliar a largura de banda de memória, medindo:

- *Throughput* de memória (MB/s): taxa de transferência em operações de vetor;
- Operações principais: Desempenho em operações de cópia, soma, escala e operações combinadas (TRIAD), refletindo padrões típicos de acesso à memória em computação científica.

Desta forma, as métricas para o STREAM estão descritas nas seções a seguir.

4.3.2.1 Balanço de Máquina

De acordo com McCalpin (1995), o balanço de máquina é a relação entre a capacidade máxima de processamento em operações de ponto flutuante (FLOPS) e a largura de banda máxima da memória (Equação (3)). Essa métrica representa o equilíbrio entre o desempenho computacional do processador e a taxa de transferência de dados da memória. Um valor elevado indica que o sistema pode estar limitado pelo subsistema de memória, especialmente em aplicações com baixa densidade computacional.

$$\text{Balanço de Máquina} = \frac{\text{Pico de FLOPS (operações/s)}}{\text{Largura de banda teórica da memória (byte/s)}} \tag{3}$$

4.3.2.2 Largura de Banda Sustentável

A largura de banda sustentável, definida por McCalpin (1995) e calculada através da Equação (4), refere-se à taxa efetiva de transferência de dados medida durante a execução do *benchmark* STREAM. Diferentemente da taxa teórica, essa métrica ignora efeitos de cache ao operar com vetores de grande dimensão, refletindo a real capacidade do sistema em transferir dados de forma contínua e sob carga realista.

$$\text{Largura de Banda Sustentável} = \frac{\text{Quantidade de dados transferidos (bytes)}}{\text{Tempo de execução (segundos)}} \tag{4}$$

Para comparar os resultados obtidos nos dois ambientes, foi calculada a variação percentual de desempenho entre o Linux nativo e o WSL (Equação (5)). Essa métrica permite quantificar a diferença relativa entre os valores medidos, independentemente de qual ambiente apresentou maior desempenho.

$$\text{Variação (\%)} = \left(\frac{\text{Linux nativo} - \text{WSL}}{\text{Linux nativo}} \right) \times 100 \tag{5}$$

4.3.2.3 Eficiência

A eficiência representa a razão entre a largura de banda de memória observada durante a execução do *benchmark* STREAM e a largura de banda teórica máxima do sistema, definida pela Equação (6). Esse indicador quantifica o grau de aproveitamento do subsistema de memória em relação ao seu potencial máximo. O conceito foi originalmente discutido por McCalpin (1995) como parte da avaliação da largura de banda sustentável, e utilizado por Nabi e Vanderbauwhede (2018) na análise de desempenho de memória em dispositivos heterogêneos.

$$\text{Eficiência (\%)} = \left(\frac{\text{Largura de banda sustentável (GB/s)}}{\text{Largura de banda teórica (GB/s)}} \right) \times 100 \tag{6}$$

4.3.2.4 Tempo por Operação de Kernel

O tempo por operação de *kernel* permite inferir a latência de acesso à memória, especialmente em arquiteturas com Acesso Não Uniforme a Memória (NUMA). Conforme demonstrado por Bergstrom (2011), variações no tempo mínimo de execução de operações vetoriais do *benchmark* STREAM — como o TRIAD — podem evidenciar sobrecarga ou contenção no subsistema de memória à medida que o número de *threads* aumenta. A Equação (7) define o tempo por operação de *kernel*.

$$\text{Tempo por Kernel} = \min(\text{tempo de execução do } \textit{kernel} \text{ em todas as repetições}) \tag{7}$$

4.3.2.5 Escalabilidade

A escalabilidade mede a variação do desempenho de um sistema conforme o número de *threads* aumenta (Equação (8)). Com base em McCalpin (1995), essa métrica é utilizada para avaliar o quanto o sistema é capaz de aproveitar recursos adicionais de paralelismo. A escalabilidade ideal ocorre quando o *throughput* cresce proporcionalmente ao número de *threads*.

$$\text{Escalabilidade} = \frac{\textit{Throughput} \text{ com } n \textit{ threads}}{\textit{Throughput} \text{ com 1 } \textit{thread}} \tag{8}$$

4.3.3 IOzone

Com foco para medir o desempenho do sistema de arquivos, o IOzone mede:

- *Throughput* de I/O: Taxa de transferência de dados durante operações de leitura e escrita sequenciais e aleatórias;
- Latência: Tempo de resposta do sistema em operações de entrada e saída;
- Eficiência em múltiplos fluxos: Avaliação do desempenho em cenários com operações simultâneas de leitura e escrita.

Neste sentido, as métricas para o IOzone estão descritas nas seções a seguir.

4.3.3.1 Throughput por operação

O *throughput* por operação representa a taxa de transferência efetiva obtida durante diferentes tipos de operações sequenciais de entrada e saída. No *benchmark* IOzone, essa taxa é medida em megabytes por segundo (MB/s) e corresponde ao volume de dados lido ou escrito por unidade de tempo. Essa métrica permite a comparação direta do desempenho do subsistema de armazenamento sob diferentes tipos de carga sequencial (XAVIER *et al.*, 2013).

4.3.3.2 Variação de Desempenho de I/O por Nível de Paralelismo

A variação percentual por operação de entrada e saída, calculada através da Equação (9), expressa a diferença relativa entre os valores de *throughput* obtidos nos dois ambientes, considerando separadamente cada tipo de operação sequencial. A métrica é baseada na taxa de transferência medida em *megabytes* por segundo (MB/s) e permite quantificar a discrepância de desempenho entre os sistemas. Essa abordagem é utilizada por Mavridis e Karatza (2017) na comparação entre ambientes virtualizados e sistemas nativos.

$$\text{Variação (\%)} = \left(\frac{\text{Linux nativo} - \text{WSL}}{\text{Linux nativo}} \right) \times 100 \tag{9}$$

Além da diferença entre ambientes, essa métrica permite observar a influência do nível de paralelismo, ou seja, como o *throughput* se comporta com o aumento do número de *threads* utilizadas. Essa análise indica a capacidade do sistema de explorar o paralelismo disponível na arquitetura para intensificar a taxa de transferência em operações de entrada e saída.

O *benchmark* IOzone oferece suporte à execução *multithread* por meio da definição da quantidade de fluxos paralelos, possibilitando a medição da resposta do sistema sob diferentes níveis de concorrência. Diaz *et al.* (2014) analisa a escalabilidade do *throughput* em ambientes *multicore* com múltiplos fluxos de I/O, enquanto Johnson e Valles (2018) utilizam essa métrica para comparar o comportamento de diferentes sistemas de arquivos sob carga concorrente.

5 RESULTADOS

Este capítulo apresenta os dados obtidos a partir da execução dos *benchmarks* nos dois ambientes avaliados: Linux nativo e WSL 2. Os resultados são organizados por ferramenta de teste, de acordo com as métricas definidas, e acompanhados por uma análise crítica sobre o desempenho observado em cada caso.

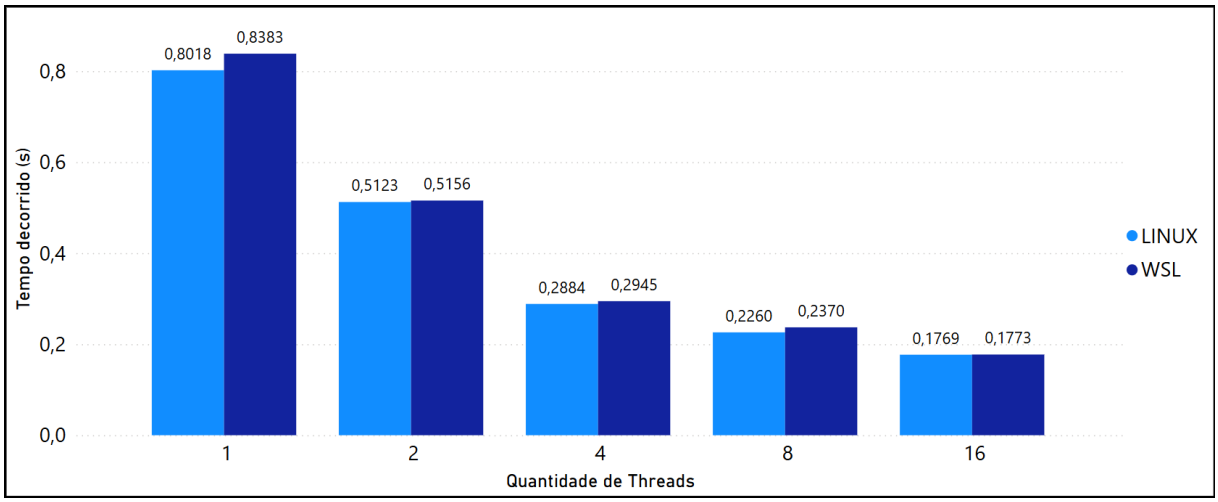
5.1 SA-HPL

O *benchmark* SA-HPL foi utilizado para avaliar o desempenho computacional em operações intensivas de ponto flutuante, com foco em tempo de execução, ganho de desempenho (*speed-up*), operações por segundo (MFLOPS) e eficiência de paralelismo. Os testes foram realizados com 1, 2, 4, 8 e 16 *threads*, aplicados a três tamanhos de matriz: 2000, 4000 e 8000. Cada combinação de configuração foi executada dez vezes para garantir confiabilidade estatística. A seguir, são apresentados os resultados obtidos para cada métrica.

5.1.1 Tempo Total de Execução

Para o menor tamanho de matriz (2000 × 2000), ambos os sistemas apresentaram tempos de execução inferiores a 1 segundo, conforme Figura 4. O Linux obteve um tempo de execução inferior ao WSL em todos os *threads*. Com o aumento do número de *threads*, houve uma redução do tempo de execução, o que indica boa escalabilidade para ambos os ambientes.

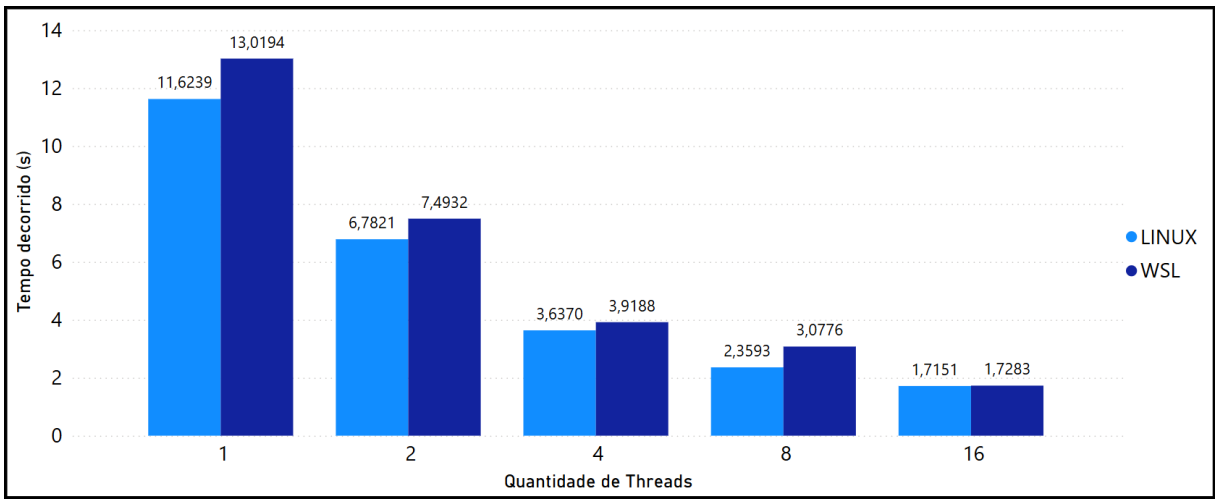
Figura 4 – Tempo de execução com matriz quadrada de 2000 por threads



Fonte: Elaborado pelo autor (2025).

No caso da matriz 4000 × 4000, a Figura 5 apresenta que os tempos de execução aumentam de forma significativa em relação à matriz anterior. Embora os valores sejam próximos, o Linux manteve um tempo de execução inferior em todos os *threads*.

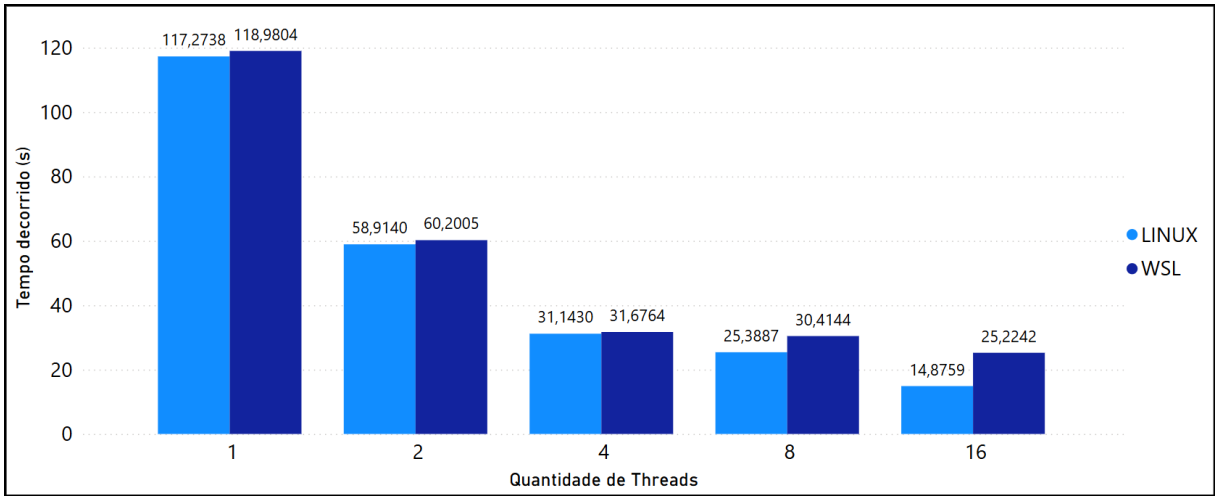
Figura 5 – Tempo de execução com matriz quadrada de 4000 por threads



Fonte: Elaborado pelo autor (2025).

Com a matriz de maior dimensão (8000 × 8000), apresentada na Figura 6, os dois sistemas demonstram desempenho semelhante com até 4 threads, com diferenças aproximadas de 1 segundo. A partir de 8 *threads*, no entanto, o Linux passa a apresentar ganhos mais significativos, com tempo de execução inferior a 15 segundos em 16 *threads*, contra mais de 25 segundos no WSL — uma diferença de 69%. Esse resultado sugere que, sob cargas maiores e elevado grau de paralelismo, o Linux nativo é mais eficiente na escalabilidade dos recursos, enquanto o WSL encontra limitações que impactam o aproveitamento de múltiplos núcleos.

Figura 6 – Tempo de execução com matriz quadrada de 8000 por threads



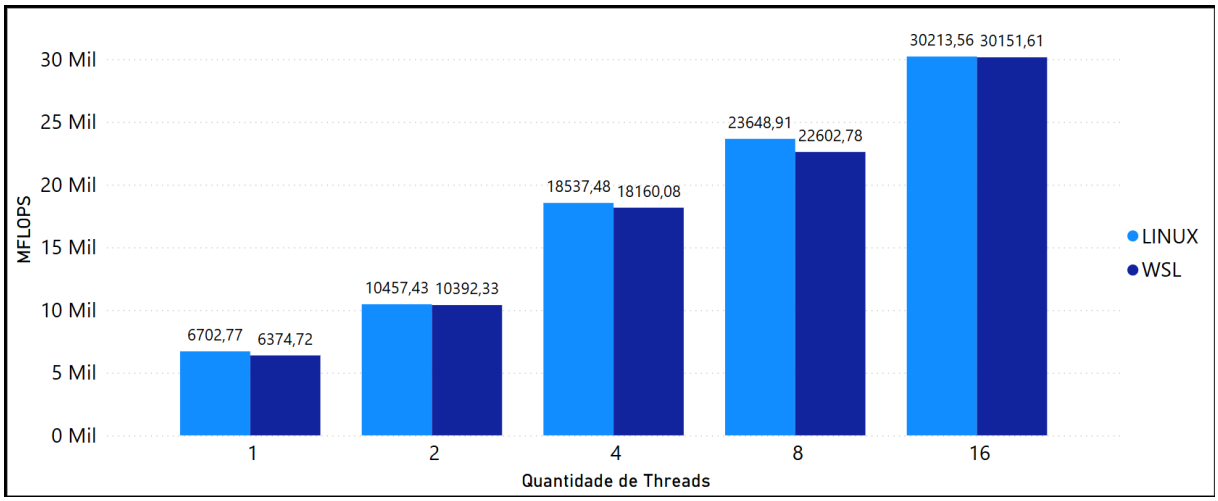
Fonte: Elaborado pelo autor (2025).

5.1.2 *Floating-Point Operations Per Second* (FLOPS)

Para o menor tamanho de matriz, ilustrado na Figura 7, o desempenho em MFLOPS foi semelhante entre os dois ambientes, com ligeira vantagem para o Linux em todas as configu-

rações de *threads*. Esse equilíbrio sugere que, sob cargas leves, tanto o Linux nativo quanto o WSL 2 são capazes de entregar desempenho computacional semelhante.

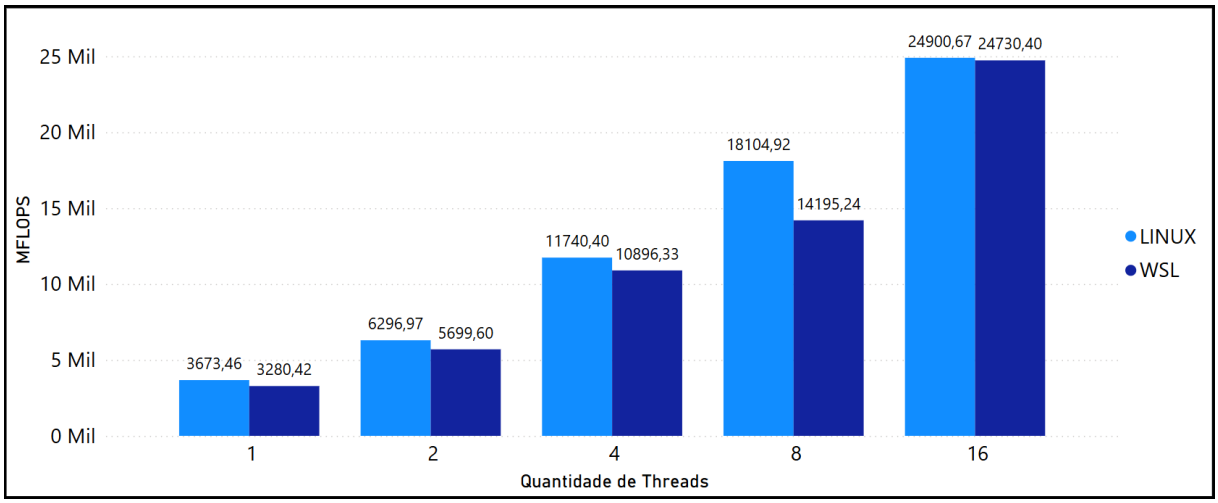
Figura 7 – Total de MFLOPS com matriz quadrada de 2000 por threads



Fonte: Elaborado pelo autor (2025).

Com a matriz de tamanho 4000 × 4000, o Linux superou o WSL em todas as configurações até 8 *threads*, com destaque para este último caso: 18.104,92 MFLOPS no Linux contra 14.195,24 no WSL, uma diferença de aproximadamente 21% (Figura 8). Contudo, esse comportamento se inverte com 16 *threads*, quando ambos os ambientes alcançam resultados praticamente idênticos. Esse padrão indica uma oscilação no desempenho do WSL em níveis intermediários de paralelismo, com uma recuperação inesperada na configuração máxima.

Figura 8 – Total de MFLOPS com matriz quadrada de 4000 por threads

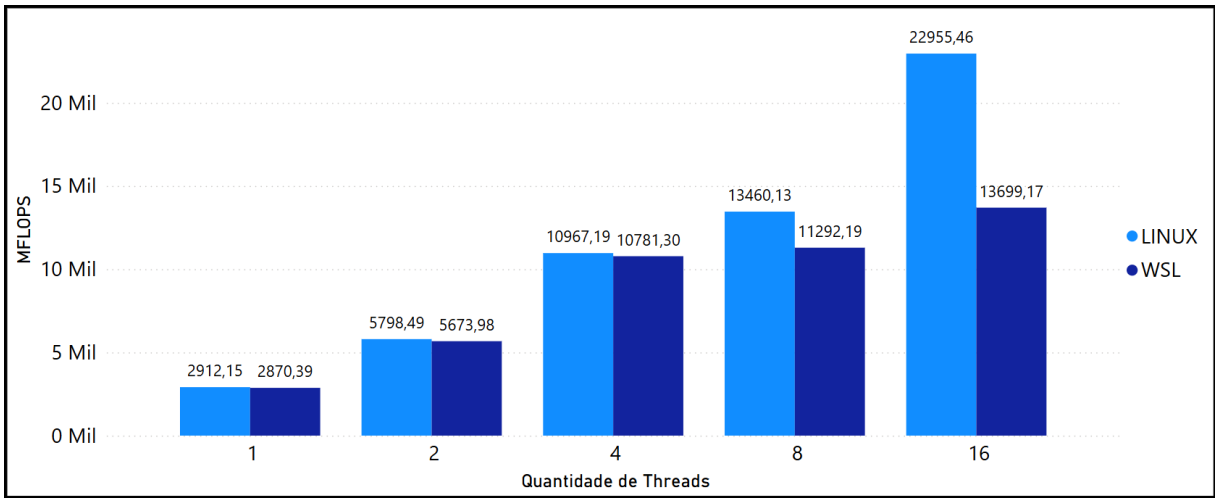


Fonte: Elaborado pelo autor (2025).

Para a maior matriz, as diferenças tornam-se mais evidentes à medida que se aumenta o número de *threads*, conforme Figura 9. A partir de 8 *threads* a diferença de desempenho do LINUX em relação ao WSL supera os 16% (13.460,13 MFLOPS contra 11.292,19). Já com 16 *threads*, o Linux alcança 40% a mais de desempenho em relação ao WSL (22.955,46 MFLOPS

contra 13.699,17). Esse resultado evidencia que, sob cargas computacionais mais intensas, o Linux nativo é capaz de aproveitar melhor os recursos de paralelismo disponíveis, enquanto o WSL demonstra limitação no ganho de desempenho proporcional ao aumento de *threads*.

Figura 9 – Total de MFLOPS com matriz quadrada de 8000 por threads

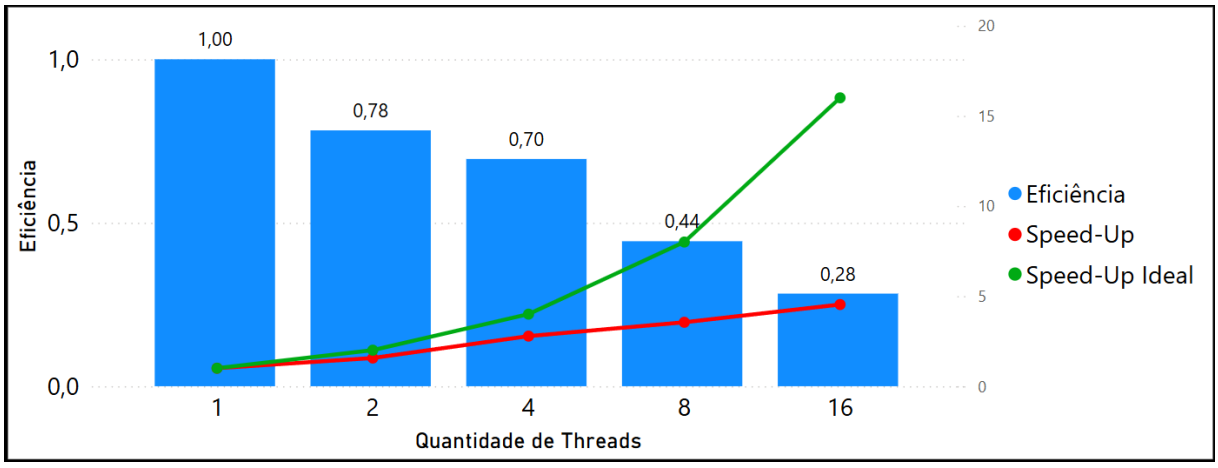


Fonte: Elaborado pelo autor (2025).

5.1.3 Eficiência de Processamento e Paralelismo

Para a matriz com dimensão 2000x2000, no ambiente Linux, a eficiência cai progressivamente com o aumento do paralelismo, conforme apresentado na Figura 10. O sistema apresenta boa escalabilidade até 4 *threads*, mas a partir de 8 há perdas significativas, típicas de aplicações com baixa carga computacional, em que o *overhead* da paralelização supera os ganhos.

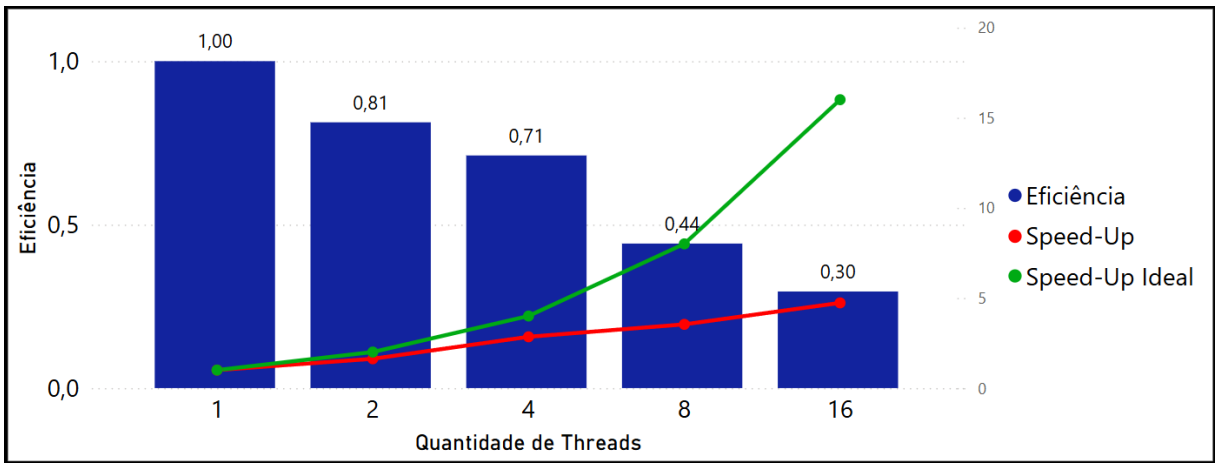
Figura 10 – Speed-Up com matriz quadrada de 2000 por threads para o Linux



Fonte: Elaborado pelo autor (2025).

Já com relação ao WSL, o comportamento é similar (Figura 11). O WSL acompanha de perto o Linux, com desempenho ligeiramente melhor em algumas configurações intermediárias.

Figura 11 – Speed-Up com matriz quadrada de 2000 por threads para o WSL

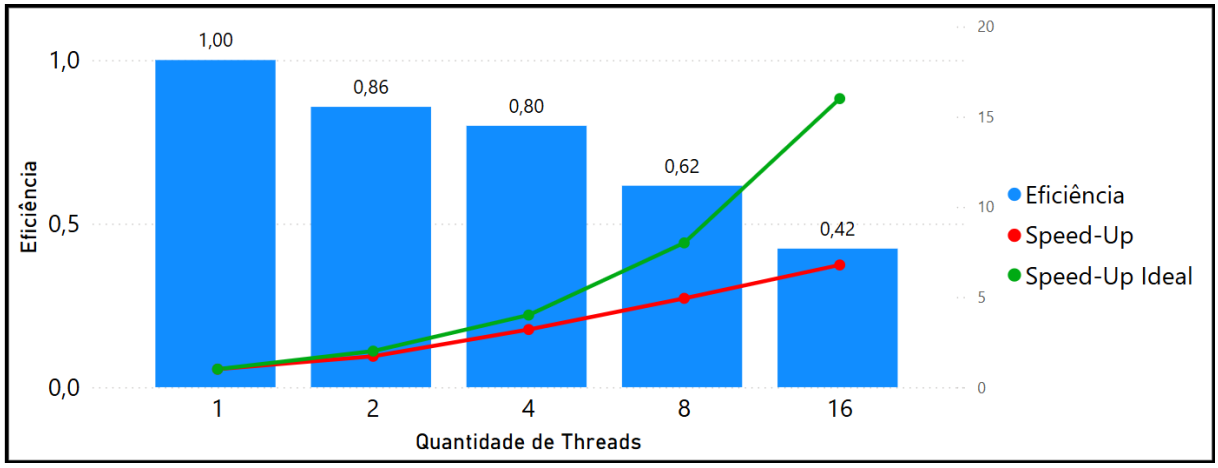


Fonte: Elaborado pelo autor (2025).

Comparando os dois sistemas para a matriz dimensão 2000 × 2000, ambos escalam bem até 4 *threads*, mas enfrentam as mesmas limitações a partir de 8. O WSL obteve uma eficiência um pouco superior com 2 e 16 *threads*, mas a diferença é pequena e não representa vantagem real em desempenho absoluto. Isso mostra que, com cargas leves, os dois ambientes se comportam de forma muito próxima em termos de escalabilidade relativa.

Quando se altera a dimensão da matriz para 4000x4000 no ambiente Linux, a eficiência mantém-se alta até 8 *threads* (Figura 12). Esse comportamento indica que o sistema consegue explorar plenamente os recursos de paralelismo até o limite dos 8 núcleos físicos do processador. A partir desse ponto, o desempenho começa a declinar, refletindo as limitações impostas pela transição para *threads* lógicas, que geralmente não oferecem o mesmo ganho de desempenho que os núcleos físicos.

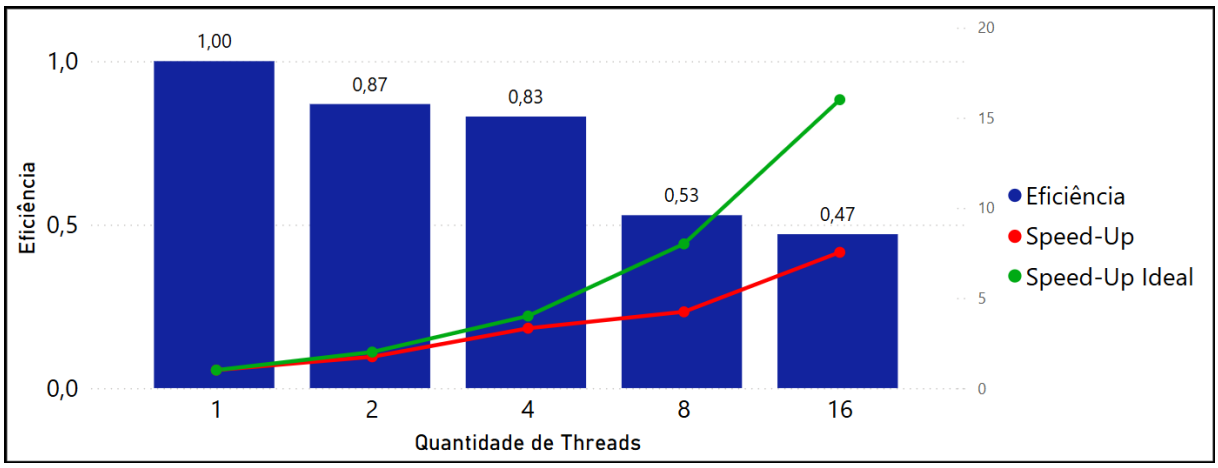
Figura 12 – Speed-Up com matriz quadrada de 4000 por threads para o Linux



Fonte: Elaborado pelo autor (2025).

No WSL, a eficiência é mantida até 4 *threads* (Figura 13). Porém, a queda é mais acentuada a partir de 8 *threads*. O comportamento é semelhante ao do Linux nas configurações iniciais, mas com eficiência inferior com 8 *threads* e levemente superior com 16.

Figura 13 – Speed-Up com matriz quadrada de 4000 por threads para o WSL

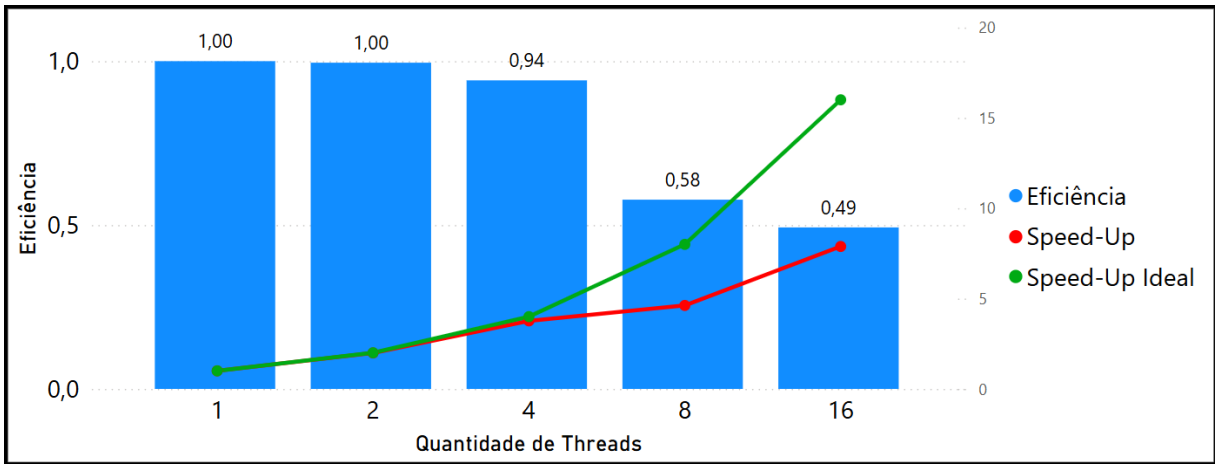


Fonte: Elaborado pelo autor (2025).

Com matriz de dimensão 4000x4000 ambos os sistemas apresentam boa escalabilidade até 4 *threads*, com desempenho praticamente equivalente. Com 8 *threads*, o Linux possui vantagem significativa, porém com 16 *threads* o WSL recupera ligeiramente a vantagem. Isso mostra que, apesar das quedas naturais de eficiência em ambos os casos, o Linux tende a ter melhor aproveitamento do paralelismo em níveis médios, enquanto o WSL se comporta de forma mais estável nas extremidades, sem que isso represente superioridade clara em desempenho.

Para a matriz de maior dimensão (8000x8000), o Linux possui eficiência máxima com 1 e 2 *threads*, mostrando aproveitamento ideal do paralelismo inicial. Com 4 *threads*, a eficiência ainda é alta, mas a partir de 8 *threads* a queda se torna mais acentuada. Apesar da redução, o sistema mantém bom aproveitamento mesmo com maior nível de paralelismo, o que indica escalabilidade consistente sob carga computacional mais intensa.

Figura 14 – Speed-Up com matriz quadrada de 8000 por threads para o Linux

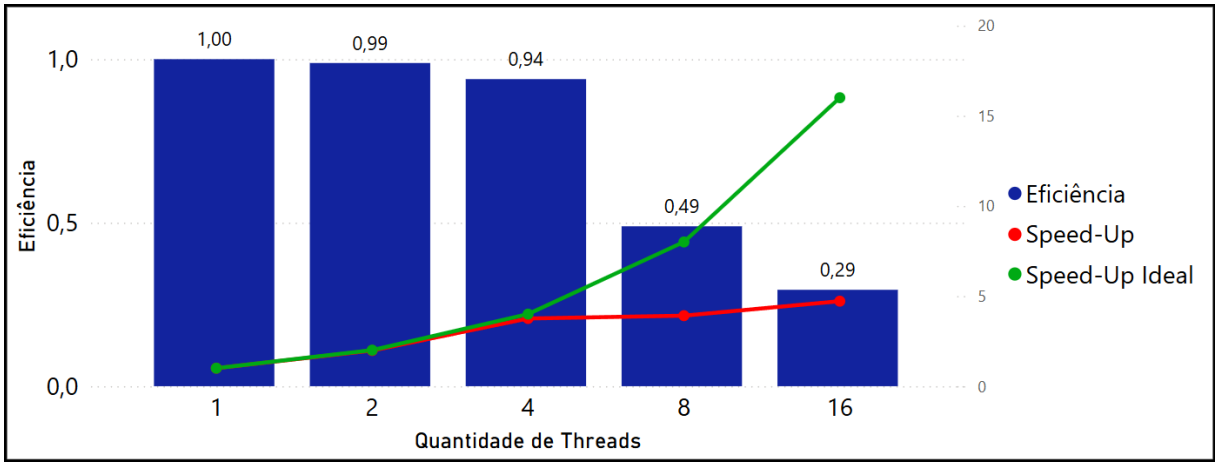


Fonte: Elaborado pelo autor (2025).

Assim como no Linux, no WSL a eficiência começa alta para 1 e 2 *threads*. Com 4 *threads*, iguala o desempenho do Linux, mas a partir de 8 *threads* a eficiência diminui fortemente.

Essa queda mostra que o WSL tem maior dificuldade em manter um bom aproveitamento de paralelismo em cargas pesadas.

Figura 15 – Speed-Up com matriz quadrada de 8000 por threads para o WSL



Fonte: Elaborado pelo autor (2025).

Comparando os dois sistemas para a matriz de dimensão 8000x8000, ambos apresentam excelente eficiência até 4 *threads*, com desempenho praticamente idêntico. A partir de 8 *threads*, no entanto, o Linux se destaca por manter uma eficiência consideravelmente maior. Enquanto o Linux ainda consegue aproveitar cerca de metade do potencial com 16 *threads*, o WSL cai para menos de um terço. Isso evidencia que, sob carga computacional mais elevada, o Linux tem maior capacidade de escalar com eficiência, enquanto o WSL sofre mais com limitações de paralelização.

5.1.4 Síntese dos resultados do SA-HPL

Os resultados do SA-HPL indicam que o WSL 2 acompanha de forma satisfatória o desempenho do Linux nativo em cargas leves e intermediárias, especialmente com matrizes de até 4000 × 4000, independentemente do número de *threads*. Com 8 *threads*, por exemplo, os dois ambientes exibem desempenho semelhante nessas cargas. No entanto, com o aumento da carga computacional — como na matriz de 8000 × 8000 — as diferenças se acentuam, e o Linux passa a demonstrar vantagem clara, principalmente nas configurações com maior paralelismo. Esse comportamento está alinhado com a arquitetura do processador utilizado, que possui 8 núcleos físicos, favorecendo o Linux na exploração de paralelismo em alto desempenho. Já o WSL 2 apresenta limitações mais evidentes em cargas intensas, ainda que mantenha desempenho viável em cenários moderados.

5.2 STREAM

O *benchmark* STREAM foi utilizado para avaliar o desempenho de memória, com foco na largura de banda efetiva, escalabilidade, e latência mínima por operação vetorial. Os testes foram realizados com 1, 2, 4, 8 e 16 *threads*, repetidos dez vezes por configuração. A seguir, são apresentados os resultados obtidos para cada métrica.

5.2.1 Balanço de Máquina

O cálculo do desempenho teórico em operações de ponto flutuante (FLOPS) é obtido pela multiplicação entre a frequência máxima da CPU, o número de núcleos e a quantidade de operações de ponto flutuante realizadas por ciclo em cada núcleo. No caso do processador AMD Ryzen 7 5700X, com arquitetura Zen 3, considera-se uma frequência de 4,6 GHz, 8 núcleos físicos e a capacidade de executar até 16 operações de ponto flutuante por ciclo por núcleo, resultando em um pico teórico de desempenho aproximado de 588,8 GFLOPS (AMD, 2025).

Para o cálculo do balanço de máquina, foi utilizada a operação TRIAD do *benchmark* STREAM, por envolver três acessos à memória por ciclo (dois de leitura e um de escrita), sendo a mais representativa em termos de uso intensivo de memória - resultado disponível na Seção 5.2.5. Os resultados considerados correspondem à execução com 1 *thread*, pois refletem o desempenho sequencial da arquitetura, sem interferência de escalonamento entre núcleos.

$$\text{Balanço de Máquina}_{\text{Linux}} = \frac{588{,}8 \text{ GFLOPS}}{28.671{,}82 \text{ MB/s}} \approx 20{,}57 \text{ FLOP/Byte}$$

$$\text{Balanço de Máquina}_{\text{WSL}} = \frac{588{,}8 \text{ GFLOPS}}{27.071{,}65 \text{ MB/s}} \approx 21{,}75 \text{ FLOP/Byte}$$

Observa-se que o valor do balanço de máquina é elevado em ambos os ambientes, o que indica que o sistema é limitado pela taxa de transferência da memória — um comportamento característico de aplicações *memory bound*. O valor ligeiramente superior registrado no WSL em relação ao Linux nativo sugere uma dependência ainda maior da largura de banda de memória, reforçando a presença de gargalos mais acentuados nesse subsistema.

5.2.2 Largura de Banda Sustentável

A Tabela 1 apresenta a variação percentual de desempenho entre os ambientes, calculada a partir da operação TRIAD do *benchmark* STREAM para diferentes quantidades de *threads*. Observa-se que o WSL apresentou desempenho inferior ao Linux nativo em todas as configurações, com variações entre 0,92% e 5,58%. A maior diferença ocorreu com 1 *thread*, indicando limitações no desempenho sequencial, possivelmente causadas pelo *overhead* da virtualização. Com o aumento do número de *threads*, a diferença diminui, sugerindo que o pa-

ralelismo ajuda a atenuar essas limitações. Esse comportamento em curva decrescente aponta para um equilíbrio entre custo e ganho no uso de múltiplos núcleos. A média geral foi de 2,57%, evidenciando uma leve desvantagem do WSL na largura de banda sustentável.

Tabela 1 – Variação percentual de desempenho da operação TRIAD do Linux em relação ao WSL por número de threads

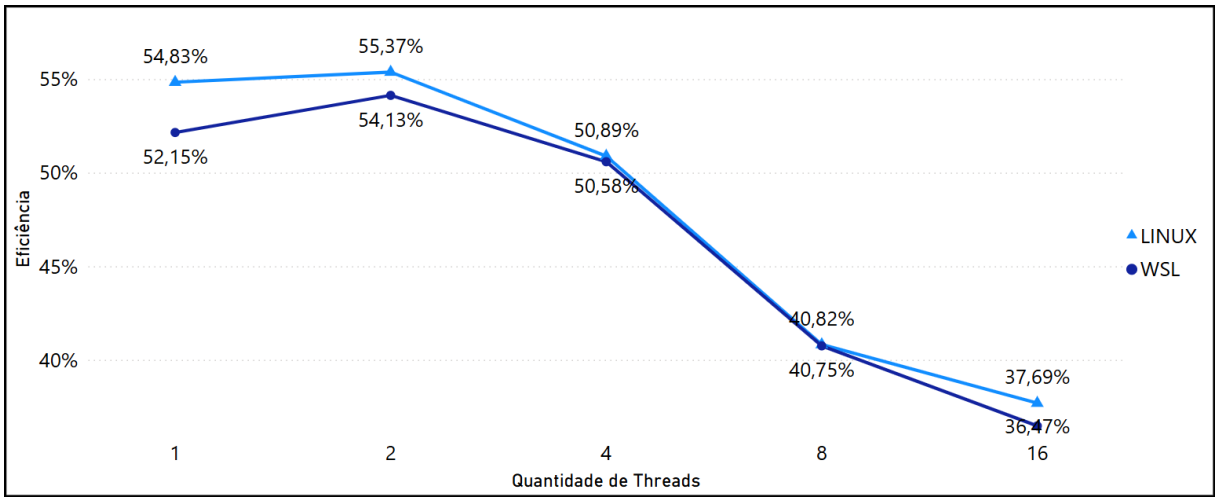
Threads	Variação (%)
1	5,58
2	2,06
4	1,50
8	0,92
16	2,16
Média Geral	**2,57**

Fonte: Elaborado pelo autor (2025).

5.2.3 Eficiência

A Figura 16 apresenta a eficiência da largura de banda de memória para os ambientes Linux nativo e WSL, considerando diferentes quantidades de *threads*. A métrica foi calculada como a razão entre a largura de banda sustentável, obtida a partir da operação TRIAD do STREAM, e a largura de banda teórica da memória. Esta, por sua vez, é obtida com a multiplicação da frequência, largura do barramento e do número de canais. Para a memória utilizada nos testes, com frequência de 3200 MHz, largura de barramento de 8 bytes (64 bits) e dois canais, tem-se um valor teórico de 51,2 GB/s.

Figura 16 – Eficiência da largura de banda por número de threads



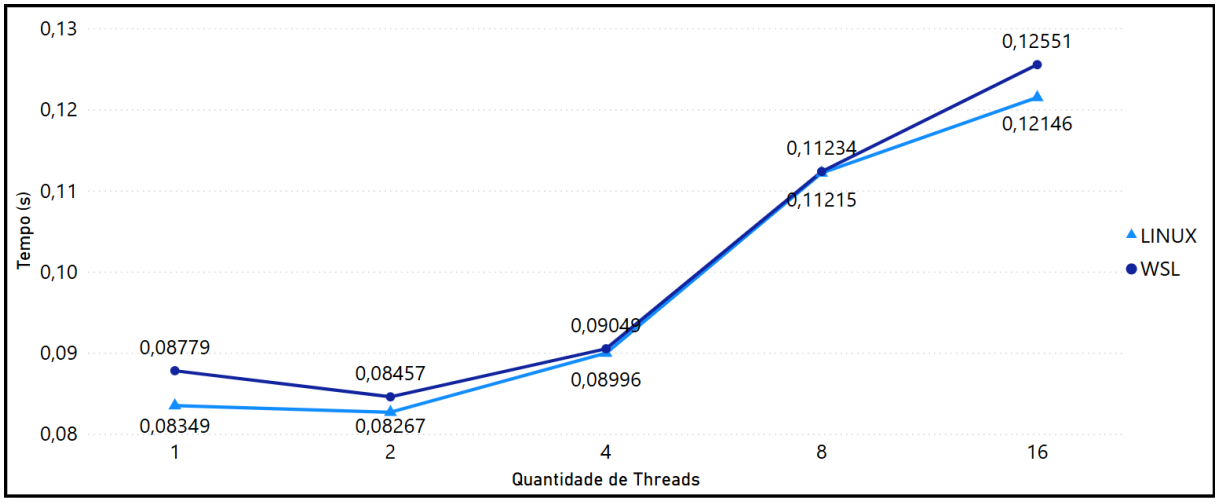
Fonte: Elaborado pelo autor (2025).

Ambos os ambientes apresentaram comportamento semelhante, com as menores diferenças observadas em 4 e 8 *threads* — o que indica uma zona de equilíbrio sob cargas moderadas. O melhor desempenho foi registrado no Linux com duas *threads* (55,37% de eficiência), enquanto a menor eficiência ocorreu com 16 *threads* em ambos os casos, refletindo a perda de aproveitamento da largura de banda em cenários altamente concorrentes. Embora o Linux tenha se mantido levemente à frente em praticamente todos os pontos, os resultados reforçam que o WSL consegue acompanhar seu desempenho de forma próxima em boa parte das configurações testadas.

5.2.4 Tempo por Operação de Kernel

A Figura 17 apresenta os menores tempos de execução registrados para a operação TRIAD do *benchmark* STREAM, em diferentes quantidades de *threads*, considerando os ambientes Linux nativo e WSL.

Figura 17 – Tempo mínimo da operação TRIAD por threads



Fonte: Elaborado pelo autor (2025).

Observa-se que os tempos de execução aumentam progressivamente a partir de 2 *threads*, comportamento típico de contenção no acesso à memória. O Linux apresentou tempos menores em todas as configurações, com destaque para a diferença mais acentuada em 1 *thread*, indicando maior impacto do *overhead* no WSL sob cargas sequenciais. A partir de 4 *threads*, os dois sistemas apresentam resultados muito próximos, sugerindo que, sob cargas intermediárias, o paralelismo é suficiente para compensar parte das limitações do ambiente virtualizado. Com 16 *threads*, a diferença volta a crescer, refletindo o impacto da concorrência intensificada sobre o acesso à memória no WSL, enquanto o Linux mantém desempenho mais estável devido ao acesso nativo aos recursos de hardware.

A Tabela 2 apresenta o módulo da variação percentual dos tempos mínimos do Linux em relação ao WSL, com base nos dados da Figura 17. Os resultados mostram que o tempo mínimo

registrado no WSL foi, em média, 2,31% mais alto do que no Linux nativo. A maior diferença percentual ocorreu com 1 *thread* (5,14%), enquanto os menores desvios apareceram com 4 e 8 *threads*, ambos abaixo de 1%. Esses dados indicam uma tendência de latência ligeiramente maior no WSL, mesmo considerando apenas as execuções mais rápidas registradas.

Tabela 2 – Variação percentual do tempo mínimo da operação TRIAD do Linux em relação ao WSL por número de threads

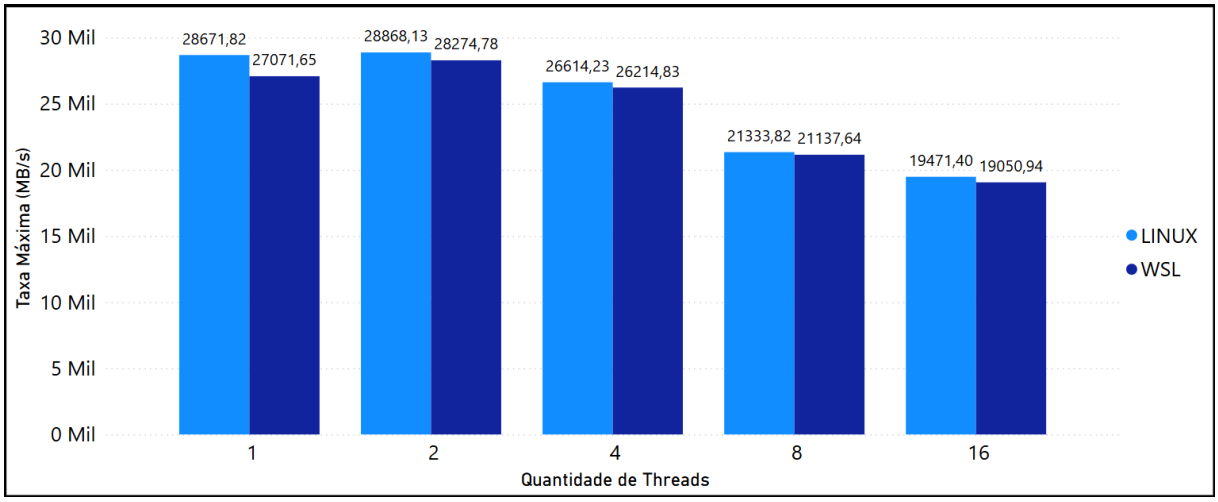
Threads	Variação (%)
1	5,14
2	2,30
4	0,59
8	0,17
16	3,33
Média Geral	**2,31**

Fonte: Elaborado pelo autor (2025).

5.2.5 Escalabilidade

A Figura 18 apresenta a taxa máxima de transferência (MB/s) obtida na operação TRIAD do *benchmark* STREAM em função do número de *threads*, para os ambientes Linux nativo e WSL.

Figura 18 – Escalabilidade por threads



Fonte: Elaborado pelo autor (2025).

Observa-se que ambos os ambientes apresentam comportamento típico de escalabilidade limitada: a taxa de transferência cresce entre 1 e 2 *threads*, estabiliza em 4 e passa a diminuir gradualmente a partir de 8. Embora os valores absolutos do Linux sejam superiores

em todas as configurações, a diferença entre os sistemas é pequena, o que indica que o WSL consegue manter desempenho comparável em termos de *throughput* de memória. No entanto, nenhum dos ambientes apresenta escalabilidade ideal, com queda perceptível de desempenho a partir de 8 *threads*.

A Tabela 3 apresenta os valores exatos da taxa máxima de transferência e os respectivos desvios padrão por quantidade de *threads* obtidos na operação TRIAD, detalhando o comportamento observado na Figura 18.

Tabela 3 – Taxa máxima de transferência e desvio padrão por número de threads

Threads	Linux Taxa Máxima (MB/s)	Linux Desvio Padrão (MB/s)	WSL Taxa Máxima (MB/s)	WSL Desvio Padrão (MB/s)
1	28.671,82	44,55	27.071,65	213,58
2	28.868,13	76,10	28.274,78	94,91
4	26.614,23	38,36	26.214,83	296,79
8	21.333,82	34,02	21.137,64	111,03
16	19.471,40	134,68	19.050,94	44,98
Média	**24.991,88**	**65,54**	**24.349,97**	**152,26**

Fonte: Elaborado pelo autor (2025).

Os valores apresentados confirmam a vantagem do Linux em todas as configurações testadas, com taxas máximas de transferência superiores e desvios padrão consistentemente menores. Esse padrão indica que o Linux não apenas alcança maiores velocidades, como também mantém um desempenho mais estável e previsível ao longo das execuções. Por outro lado, o WSL apresentou variações elevadas, especialmente com 1, 4 e 8 *threads*, sugerindo que seu desempenho é mais suscetível a flutuações internas — possivelmente relacionadas ao gerenciamento de memória compartilhada e à concorrência entre processos no ambiente virtualizado.

Um exemplo ocorre com 4 threads, onde o desvio padrão no WSL atinge 296,79 MB/s, contra apenas 38,36 MB/s no Linux. Isso representa uma variação quase oito vezes maior, o pode afetar a performance de aplicações sensíveis à consistência do *throughput*. Ainda assim, o WSL manteve valores médios próximos aos do Linux, o que reforça sua viabilidade em cenários onde pequenas oscilações de desempenho são toleráveis.

5.2.6 Síntese dos resultados do STREAM

Os testes com o *benchmark* STREAM revelaram que o WSL 2 apresenta desempenho muito próximo ao do Linux nativo em diversas métricas, especialmente nas configurações intermediárias de paralelismo. Com 4 e 8 *threads*, as diferenças entre os dois ambientes foram mínimas em termos de largura de banda, latência e *throughput*, indicando uma zona de equilíbrio entre carga computacional e concorrência. Já em configurações com 1 ou 16 *threads*,

as variações foram mais significativas, sugerindo maior impacto da limitação sequencial e da contenção por memória no ambiente virtualizado.

A eficiência da largura de banda foi semelhante nos dois sistemas, com pequenas vantagens para o Linux, principalmente nas configurações com menor ou maior número de *threads*. A curva de variação percentual apresentou formato parabólico, com a menor diferença entre os ambientes justamente nas zonas intermediárias, reforçando o padrão observado. Esses resultados indicam que, para aplicações com uso moderado de memória e paralelismo, o WSL 2 consegue manter desempenho competitivo. Entretanto, em situações de maior acesso à memória ou concorrência elevada, o Linux ainda demonstra maior consistência e estabilidade.

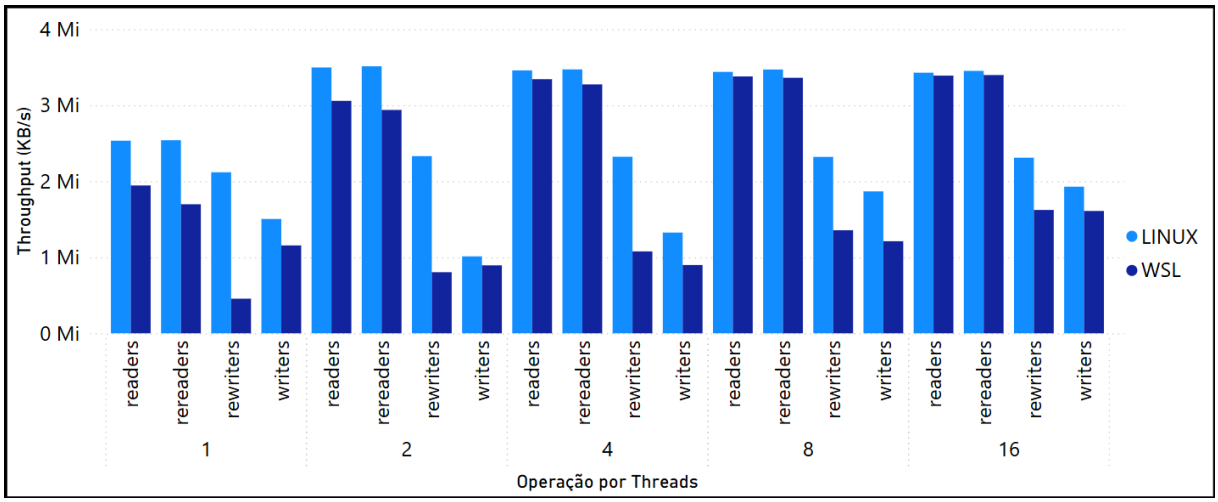
5.3 IOzone

O *benchmark* IOzone foi utilizado para avaliar o desempenho do subsistema de entrada e saída (I/O), com foco na taxa de transferência por operação, variação percentual entre ambientes e escalabilidade. Os testes foram realizados com 1, 2, 4, 8 e 16 *threads* e repetidos dez vezes por configuração. A seguir, são apresentados os resultados obtidos para cada métrica.

5.3.1 *Throughput* por operação

A Figura 19 apresenta o *throughput* médio (KB/s) obtido nas operações de leitura (*readers*), releitura (*rereaders*), reescrita (*rewriters*) e escrita (*writers*) para os ambientes Linux nativo e WSL, com 1, 2, 4, 8 e 16 *threads*.

Figura 19 – Throughput por operação por thread



Fonte: Elaborado pelo autor (2025).

Observa-se que o Linux apresentou desempenho superior em praticamente todas as combinações testadas, com destaque para as operações de escrita e reescrita, especialmente com 1 e 2 *threads*. Nessas configurações, o WSL teve desempenho significativamente inferior,

principalmente na operação de *writers*, em que os valores ficaram bem abaixo dos observados no ambiente nativo.

Com o aumento do número de *threads*, o WSL se aproxima do desempenho do Linux nas operações de leitura e releitura, atingindo *throughput* similar a partir de 8 *threads*. Ainda assim, o Linux mantém maior estabilidade e regularidade entre todas as operações e níveis de paralelismo. O WSL, embora competitivo em alguns cenários, apresenta maior variabilidade e limitações evidentes em operações de escrita, o que pode afetar sua previsibilidade em cargas de I/O mistas.

5.3.2 Variação de Desempenho de I/O por Nível de Paralelismo

A Tabela 4 apresenta a variação percentual do desempenho do Linux em relação ao WSL para cada tipo de operação de I/O, considerando diferentes quantidades de *threads*. Os valores refletem o quanto o WSL foi mais lento em relação ao ambiente nativo em cada cenário testado.

Tabela 4 – Variação percentual do desempenho do Linux em relação ao WSL por operação de I/O e número de threads — IOzone

Operação	1 Thread	2 Threads	4 Threads	8 Threads	16 Threads
readers	23,24%	12,56%	3,34%	1,74%	1,13%
rereaders	33,16%	16,35%	5,68%	3,15%	1,58%
rewriters	78,50%	65,52%	53,61%	41,57%	29,75%
writers	23,16%	11,64%	32,21%	35,13%	16,55%

Fonte: Elaborado pelo autor (2025).

Os resultados mostram que o WSL apresentou desvantagem em todas as operações, com maior impacto nas operações de escrita e reescrita. A operação *rewriters* concentrou as maiores perdas, chegando a 78,50% com 1 *thread*, e mantendo variações acima de 40% mesmo com 8 e 16 *threads*. As operações *writers* também apresentaram variações expressivas, com destaque para 32,21% com 4 *threads* e 35,13% com 8 *threads*.

Por outro lado, as operações de leitura (*readers* e *rereaders*) demonstraram diferença percentual significativamente menor, especialmente nas execuções com maior paralelismo. A partir de 8 *threads*, as variações ficaram abaixo de 5%, com valores inferiores a 2% em *readers* com 16 *threads*. Esse comportamento indica que o WSL consegue escalar melhor em tarefas de leitura, mas continua limitado em operações de escrita, mesmo sob condições de concorrência elevada. De modo geral, os dados reforçam que o WSL apresenta desempenho aceitável em operações de leitura, porém com prejuízo acentuado em escrita — o que compromete sua previsibilidade e eficiência em cargas mistas de I/O.

5.3.3 Síntese dos resultados do IOzone

Os resultados obtidos com o IOzone evidenciam que o Linux nativo mantém vantagem geral sobre o WSL 2 nas operações de entrada e saída, especialmente em tarefas de escrita e reescrita. Ainda assim, o WSL apresentou desempenho competitivo em operações de leitura, sobretudo em configurações com maior número de *threads*, onde as diferenças entre os ambientes foram menos relevantes. Esse comportamento indica que o WSL 2 pode ser uma alternativa viável para cenários de leitura com carga moderada, mas tende a apresentar limitações importantes quando submetido a operações de escrita intensiva ou que demandam maior previsibilidade no *throughput*.

6 DISCUSSÕES E LIMITAÇÕES

Este estudo seguiu critérios rigorosos para garantir comparabilidade entre os ambientes Linux nativo e WSL 2. No Linux, foram necessárias configurações adicionais para forçar o modo de desempenho máximo, como desativar a interface gráfica, ajustar os *clocks* da CPU e atualizar o *kernel* para garantir suporte total ao processador. Uma distribuição como o Ubuntu Server poderia ter simplificado essa preparação.

No WSL 2, adotou-se o plano de energia Alto Desempenho do Windows e buscou-se minimizar interferências de processos em segundo plano. Ainda assim, certas anomalias foram observadas, como tempos negativos no STREAM (Apêndice A) e throughputs excessivamente altos no IOzone (Apêndice B), possivelmente relacionados à virtualização e ao gerenciamento de disco e memória pelo sistema *host*.

Embora o Linux tenha, em geral, apresentado desempenho superior, o WSL mostrou resultados semelhantes que pode ser considerado como um empate técnico. Isso ocorreu, por exemplo, nas execuções com 4 e 8 *threads* no STREAM (tempo mínimo da operação Triad) e com 16 *threads* nos testes de MFLOPS com matrizes menores (2000 e 4000). Esse tipo de comportamento é coerente com observações feitas por Xavier *et al.* (2013), que relatam que ambientes virtualizados leves podem se beneficiar de técnicas internas de otimização, como *caching* agressivo ou pré-alocação de memória.

Os testes foram realizados com *benchmarks* sintéticos amplamente utilizados para medir CPU, memória e I/O. Embora confiáveis, essas ferramentas simulam cenários específicos. Assim, os resultados aqui obtidos devem ser compreendidos dentro do escopo definido neste estudo, e não como representação absoluta de todas as possíveis aplicações científicas.

Em resumo, os dados indicam que o Linux nativo oferece melhor desempenho sob cargas elevadas e maior concorrência, mas o WSL 2 se mostrou competitivo em cenários menos exigentes. Esses resultados reforçam seu potencial como ambiente viável para aplicações científicas leves ou integradas ao ecossistema Windows.

7 CONCLUSÃO

Este trabalho teve como objetivo avaliar o desempenho do WSL 2 em comparação com o ambiente Linux nativo na execução de aplicações científicas, utilizando os *benchmarks* SA-HPL, STREAM e IOzone. Foi adotada uma metodologia padronizada, com ambientes controlados e repetição dos testes, buscando garantir condições comparáveis de análise entre os sistemas.

O referencial teórico abordou os conceitos de virtualização e desempenho computacional, destacando a arquitetura do WSL 2 como solução híbrida que combina integração com o sistema Windows e execução de um *kernel* Linux real. A escolha dos *benchmarks* permitiu avaliar aspectos distintos do desempenho: operações de ponto flutuante (SA-HPL), acesso à memória (STREAM) e sistema de arquivos (IOzone).

Os resultados obtidos demonstraram que o Linux nativo apresentou desempenho superior nas três métricas, especialmente sob cargas intensas e alto nível de paralelismo. O *benchmark* SA-HPL evidenciou melhor escalabilidade do Linux na utilização de múltiplos núcleos. O STREAM mostrou uma eficiência de memória levemente superior e mais estável no ambiente nativo. Já o IOzone revelou diferenças mais acentuadas, com o Linux apresentando vantagem significativa em operações de escrita e reescrita, enquanto o WSL se mostrou mais competitivo em leituras com paralelismo moderado.

Apesar da diferença de desempenho observada, o WSL 2 se mostrou funcional e tecnicamente viável em diversos cenários. Sua facilidade de uso e integração ao sistema Windows o tornam atrativo para atividades de prototipagem, ensino e testes não críticos. Esses resultados reforçam os resultados discutidos no capítulo 6, ao mesmo tempo em que apontam caminhos promissores para sua adoção consciente.

Conclui-se que os objetivos definidos foram plenamente atingidos, permitindo caracterizar o comportamento do WSL 2 frente ao Linux nativo com base em métricas específicas e resultados empíricos. O estudo contribui para a compreensão dos limites e possibilidades do WSL no contexto da computação científica, oferecendo subsídios para sua adoção consciente em diferentes cenários. Como continuidade, recomenda-se a execução dos testes com distribuições voltadas a desempenho, como o Ubuntu Server, bem como a inclusão de cargas reais e a replicação dos experimentos em diferentes arquiteturas de *hardware*. Além disso, sugere-se a comparação direta do WSL 2 com outras soluções de virtualização leve, como *containers*, de modo a ampliar o escopo da análise e avaliar sua competitividade em ambientes mais próximos da produção científica moderna.

REFERÊNCIAS

AL-HAMOURI, R. *et al.* Measuring the impacts of virtualization on the performance of thread-based applications. *In*: **2020 Seventh International Conference on Software Defined Systems (SDS)**. [*S.l.*: *s.n.*], 2020. p. 131–138.

AMD. **AMD Ryzen 7 5700X**. 2025. https://www.amd.com/en/support/downloads/drivers.html/processors/ryzen/ryzen-5000-series/amd-ryzen-7-5700x.html.

ARYOTEJO, G. *et al.* Performance of virtual machine managers for computer network learning. *In*: **2021 5th International Conference on Informatics and Computational Sciences (ICICoS)**. [*S.l.*: *s.n.*], 2021. p. 155–159.

BERGSTROM, L. **Measuring NUMA effects with the STREAM benchmark**. 2011. Disponível em: https://arxiv.org/abs/1103.3225.

DEVELOPER, M. **The new Windows subsystem for Linux architecture: a deep dive - BRK3068**. 2020. https://www.youtube.com/watch?v=lwhMThePdIo [Acessado em: (31/01/2025)].

DIAZ, C. O. *et al.* Performance evaluation of an iaas opportunistic cloud computing. *In*: **2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing**. [*S.l.*: *s.n.*], 2014. p. 546–547.

DORDEVIC, B. S.; JOVANOVIC, S. P.; TIMCENKO, V. V. Cloud computing in amazon and microsoft azure platforms: Performance and service comparison. *In*: **2014 22nd Telecommunications Forum Telfor (TELFOR)**. [*S.l.*: *s.n.*], 2014. p. 931–934.

GUENNEBAUD G., J. B. e. a. **Eigen: C++ templates for linear algebra**. 2024. http://eigen.tuxfamily.org/ [Acessado em: (01/05/2025)].

IOZONE. **IOzone Filesystem Benchmark**. 2016. https://www.iozone.org/ [Acessado em: (19/12/2024)].

JAIN, R. **The Art of Computer Systems Performance Analysis**: Techniques for experimental design, measurement, simulation, and modeling. [*S.l.*]: Wiley, 1991. ISBN 9780471503361.

JOHNSON, D. A.; VALLES, D. An initial scale-factor linear polynomial regression model approach for hardware performance on an hpc compute-node. *In*: **2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)**. [*S.l.*: *s.n.*], 2018. p. 661–666.

KOCHBERGER, P.; TAUBER, A.; SCHRITTWIESER, S. Assessment of the transparency of the windows subsystem for linux (wsl). *In*: **2019 International Conference on Software Security and Assurance (ICSSA)**. [*S.l.*: *s.n.*], 2019. p. 60–69.

LIN, Y.; BARKER, A.; THOMSON, J. Modelling vm latent characteristics and predicting application performance using semi-supervised non-negative matrix factorization. *In*: **2020 IEEE 13th International Conference on Cloud Computing (CLOUD)**. [*S.l.*: *s.n.*], 2020. p. 470–474.

MAVRIDIS, I.; KARATZA, H. Performance and overhead study of containers running on top of virtual machines. *In*: **2017 IEEE 19th Conference on Business Informatics (CBI)**. [*S.l.*: *s.n.*], 2017. v. 02, p. 32–38.

MCCALPIN, J. D. Memory bandwidth and machine balance in current high performance computers. **IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter**, p. 19–25, dez. 1995.

MCCALPIN, J. D. **STREAM: Sustainable Memory Bandwidth in High Performance Computers**. 2016. https://www.cs.virginia.edu/stream [Acessado em: (18/12/2024)].

MICROSOFT. **Windows Subsystem for Linux Documentation**. 2022. https://learn.microsoft.com/en-us/windows/wsl/ [Acessado em: (18/12/2024)].

NABI, S. W.; VANDERBAUWHEDE, W. Mp-stream: A memory performance benchmark for design space exploration on heterogeneous hpc devices. *In*: **2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [*S.l.*: *s.n.*], 2018. p. 194–197.

ORACLE. **VirtualBox**. 2025. http://www.virtualbox.org/ [Acessado em: (02/05/2025)].

POKHARANA, A.; GUPTA, R. Using sysbench, analyze the performance of various guest virtual machines on a virtual box hypervisor. *In*: **2023 2nd International Conference for Innovation in Technology (INOCON)**. [*S.l.*: *s.n.*], 2023. p. 1–5.

REGOLA, N.; DUCOM, J.-C. Recommendations for virtualization technologies in high performance computing. *In*: **2010 IEEE Second International Conference on Cloud Computing Technology and Science**. [*S.l.*: *s.n.*], 2010. p. 409–416.

RISTA, C. *et al.* Improving the network performance of a container-based cloud environment for hadoop systems. *In*: **2017 International Conference on High Performance Computing Simulation (HPCS)**. [*S.l.*: *s.n.*], 2017. p. 619–626.

RISTA, C.; TEIXEIRA, M.; FONSECA, M. A self-adaptive hpl-based benchmark with dynamic task parallelism for multicore systems. *In*: ____. [*S.l.*: *s.n.*], 2024. p. 242–251. ISBN 978-3-031-60226-9.

UBUNTU. **CPU Governors and the cpupower tool**. 2025. https://documentation.ubuntu.com/server/explanation/performance/perf-tune-cpupower/.

VMWARE. **VMware vSphere | Virtualization Platform**. 2025. https://www.vmware.com/products/cloud-infrastructure/vsphere [Acessado em: (28/06/2025)].

WALTERS, J. P. *et al.* A comparison of virtualization technologies for hpc. *In*: **22nd International Conference on Advanced Information Networking and Applications (aina 2008)**. [*S.l.*: *s.n.*], 2008. p. 861–868.

XAVIER, M. G. *et al.* Performance evaluation of container-based virtualization for high performance computing environments. *In*: **2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing**. [*S.l.*: *s.n.*], 2013. p. 233–240.

ZHANG, H.; NIE, J. Performance test of taurus hpc system. *In*: **2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)**. [*S.l.*: *s.n.*], 2016. p. 185–188.

ZUNIN, V. V.; STEMPKOVSKY, A. L.; SOLOVYEV, R. A. Cad architecture for expansion of wsl-based combinational circuits dataset. *In*: **2024 International Russian Smart Industry Conference (SmartIndustryCon)**. [*S.l.*: *s.n.*], 2024. p. 294–298.

APÊNDICE A – Saídas Anômalas do Benchmark STREAM

Este apêndice apresenta exemplos de saídas anômalas obtidas durante a execução do *benchmark* STREAM no ambiente WSL. As tabelas a seguir ilustram casos em que os tempos médios e mínimos foram registrados com valores negativos ou incoerentes, resultando em taxas de transferência inválidas. Esses dados foram desconsiderados nas análises principais apresentadas no Capítulo 5, mas são incluídos aqui como documentação complementar às limitações discutidas no Capítulo 6.

Tabela 5 – Primeiro exemplo de saída anômala do benchmark STREAM no ambiente WSL 2

Função	Best Rate (MB/s)	Avg Time (s)	Min Time (s)	Max Time (s)
COPY	38060.8	0.042367	0.042038	0.042933
SCALE	24516.6	0.066750	0.065262	0.068861
ADD	-0.2	-1199.874228	-10799.598152	0.092501
TRIAD	26563.1	1200.057028	0.090351	10799.782246

Fonte: Elaborado pelo autor (2025).

Tabela 6 – Segundo exemplo de saída anômala do benchmark STREAM no ambiente WSL 2

Função	Best Rate (MB/s)	Avg Time (s)	Min Time (s)	Max Time (s)
COPY	39069.0	0.041109	0.040953	0.041570
SCALE	25372.7	1200.029165	0.063060	10799.754543
ADD	27919.0	0.087102	0.085963	0.088688
TRIAD	-0.2	-1199.880432	-10799.604851	0.085904

Fonte: Elaborado pelo autor (2025).

Tabela 7 – Terceiro exemplo de saída anômala do benchmark STREAM no ambiente WSL 2

Função	Best Rate (MB/s)	Avg Time (s)	Min Time (s)	Max Time (s)
COPY	33950.1	0.047294	0.047128	0.047475
SCALE	17359.2	0.093661	0.092170	0.096194
ADD	18982.4	1200.092604	0.126433	10799.818594
TRIAD	-0.2	-1199.839439	-10799.565716	0.126865

Fonte: Elaborado pelo autor (2025).

Tabela 8 – Quarto exemplo de saída anômala do benchmark STREAM no ambiente WSL 2

Função	Best Rate (MB/s)	Avg Time (s)	Min Time (s)	Max Time (s)
COPY	33940.7	0.047268	0.047141	0.047377
SCALE	-0.1	-1199.872124	-10799.598370	0.094070
ADD	18977.3	1200.092748	0.126467	10799.815311
TRIAD	19056.2	0.126652	0.125943	0.127837

Fonte: Elaborado pelo autor (2025).

APÊNDICE B – Saídas Anômalas do Benchmark lOzone

Este apêndice apresenta exemplos de saídas anômalas obtidas durante a execução do *benchmark* IOzone no ambiente WSL. As tabelas a seguir ilustram casos em que as taxas de transferência foram registradas com valores extremamente elevados e incompatíveis com a capacidade real do sistema, indicando inconsistências no processo de medição. Esses dados não foram considerados nas análises estatísticas principais apresentadas no Capítulo 5, mas são documentados aqui como evidência técnica complementar às limitações discutidas no Capítulo 6.

Tabela 9 – Primeiro exemplo de saída anômala do benchmark IOzone no ambiente WSL 2

Operação	Children Throughput (kB/s)	Parent Throughput (kB/s)
initial writers	4.031.263,00	1.331.286,71
rewriters	1.048.576.000.000,00	1.048.576.000.000,00
readers	17.340.758,00	13.463.649,25
re-readers	17.342.194,00	15.088.321,07

Fonte: Elaborado pelo autor (2025).

Tabela 10 – Segundo exemplo de saída anômala do benchmark IOzone no ambiente WSL 2

Operação	Children Throughput (kB/s)	Parent Throughput (kB/s)
initial writers	2.088.960.000.000,00	2.088.960.000.000,00
rewriters	7.630.748,25	747.036,35
readers	30.623.967,00	24.350.937,92
re-readers	30.289.075,00	26.414.407,89

Fonte: Elaborado pelo autor (2025).

Tabela 11 – Terceiro exemplo de saída anômala do benchmark IOzone no ambiente WSL 2

Operação	Children Throughput (kB/s)	Parent Throughput (kB/s)
initial writers	7.690.885,00	511.648,70
rewriters	2.091.008.000.000,00	2.091.008.000.000,00
readers	29.821.503,00	26.484.578,73
re-readers	30.060.509,00	25.316.573,49

Fonte: Elaborado pelo autor (2025).

Tabela 12 – Quarto exemplo de saída anômala do benchmark IOzone no ambiente WSL 2

Operação	Children Throughput (kB/s)	Parent Throughput (kB/s)
initial writers	664.727,58	533.907,45
rewriters	4.031.488.000.000,00	4.031.488.000.000,00
readers	36.882.280,00	33.052.761,58
re-readers	10.617.389,25	10.383.894,27

Fonte: Elaborado pelo autor (2025).