Avaliação de Desempenho do Docker em Diferentes Sistemas Operacionais

Docker Performance Evaluation across Operating Systems

Maciej Sobieraj Daniel Kotynski

7 de julho de 2025

Applied Sciences 2024, Volume 14

Sumário

Introdução

Contexto e Motivação

- Docker é uma tecnologia de virtualização extremamente popular, usada por milhões de desenvolvedores.
- O desempenho da virtualização é fortemente influenciado pelo sistema operacional (SO) hospedeiro.
- O Docker no Linux utiliza recursos nativos do kernel, mas no Windows e macOS, requer uma camada de virtualização adicional (VM leve).
- **Questão principal:** Qual o impacto do SO no desempenho de contêineres Docker?

Objetivos do Estudo

- Avaliar o overhead de desempenho de diferentes SOs (Linux, Windows, macOS) ao hospedar contêineres Docker.
- Preparar um ambiente de teste justo para medições precisas.
- Projetar benchmarks variados para identificar gargalos de desempenho (CPU, I/O, Rede).
- Recomendar o melhor ambiente com base nos resultados obtidos.

Arquitetura do Docker

Docker no Linux

Utiliza recursos de virtualização de baixo nível do próprio kernel Linux para fornecer isolamento e gerenciamento de recursos.

Namespaces:

Control Groups (cgroups):

Docker no Linux

Utiliza recursos de virtualização de baixo nível do próprio kernel Linux para fornecer isolamento e gerenciamento de recursos.

Namespaces:

- PID (isola processos)
- Mount (isola sistema de arquivos)
- Network (isola interfaces de rede)
- User (mapeia UIDs para segurança)
- IPC (isola comunicação entre processos)

Control Groups (cgroups):

Docker no Linux

Utiliza recursos de virtualização de baixo nível do próprio kernel Linux para fornecer isolamento e gerenciamento de recursos.

Namespaces:

- PID (isola processos)
- Mount (isola sistema de arquivos)
- Network (isola interfaces de rede)
- User (mapeia UIDs para segurança)
- IPC (isola comunicação entre processos)

Control Groups (cgroups):

- Limita e gerencia o uso de recursos:
  - CPU
  - Memória (evita OOM Killer)
  - I/O de disco e rede
- Permite definir limites “soft” e “hard”.

Docker no Windows e macOS

Como o Docker depende de recursos do kernel Linux, outras plataformas precisam de uma camada de emulação. **Docker Desktop para Windows:**

- Utiliza o **Windows Subsystem for Linux 2 (WSL2)**.
- WSL2 é uma VM leve que roda sobre o Hypervisor **Hyper-V**.
- Usa **VPNKit** para gerenciar a conectividade de rede.

Docker Desktop para Mac:

- Utiliza o framework **Moby** para fornecer um ambiente Linux.
- Roda sobre o Hypervisor **HyperKit**.
- Também utiliza **VPNKit** para a rede.

Metodologia

Plataforma de Teste

Para garantir resultados comparáveis, todos os testes foram executados no mesmo hardware.

Hardware

- **Modelo:** Apple MacBook Pro 13 (2019)
- **CPU:** Intel i5-8257u @ 1.40 GHz (x86/64)
- **RAM:** 16 GB LPDDR3 2133 Mhz
- **Armazenamento:** 256 GB NVME SSD

Sistemas Operacionais

- **macOS:** Ventura 13.5.1
- **Windows 10:** 22H2 (via Boot Camp)
- **Ubuntu:** 22.04 (distribuição T2-Ubuntu para compatibilidade)

Benchmarks Utilizados

- **Cálculo de π:** Teste de CPU single-thread usando a fórmula de Leibniz.
- **Sysbench:** Ferramenta para medir desempenho de:
  - CPU (cálculo de números primos)
  - I/O de disco (leitura/escrita sequencial e aleatória)
- **Iperf3:** medição de desempenho de rede (TCP e UDP) entre contêineres e entre contêiner e host.
- **7zip:** benchmark de compressão/descompressão para avaliar o sistema de forma complexa.
- **pgbench:** Benchmark para o banco de dados PostgreSQL.
- **Ataque DoS (Slowloris):** Teste de resistência em um servidor web Apache para medir a disponibilidade sob estresse.

Resultados e Análise

Desempenho de CPU: Cálculo de π (single-thread)

- macOS e Linux com desempenho similar.
- Windows foi significativamente mais lento.
- *(Menor tempo de execução é melhor)*

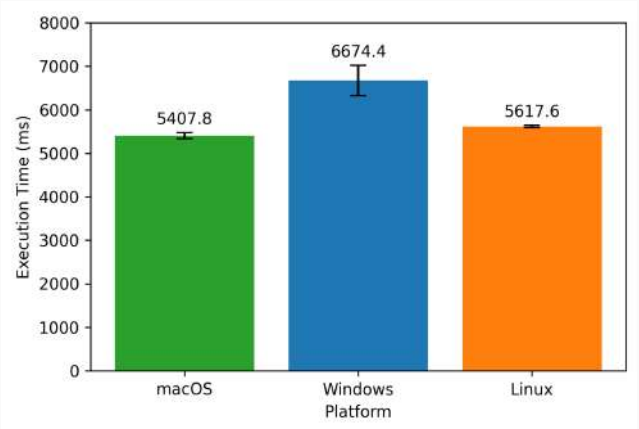

Figura 1: Tempo de execução para cálculo de π (s).

Desempenho de CPU: Sysbench CPU

Sysbench CPU (multi-thread)

- Linux apresentou a melhor desempenho.
- Windows ficou em segundo lugar.
- macOS teve o pior desempenho.
- *(Mais eventos por segundo é melhor)*

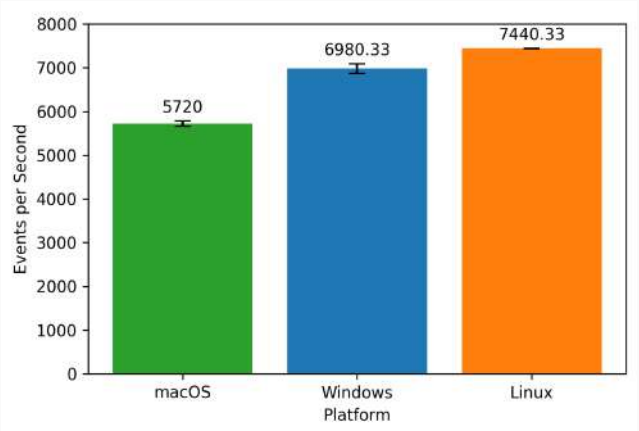

Figura 2: Eventos por segundo no Sysbench CPU.

Desempenho de I/O (Disco) com Sysbench – Leitura

Leitura (Sequencial e Aleatória)

- Todos os sistemas tiveram desempenho comparável.
- Leve vantagem para o Linux.

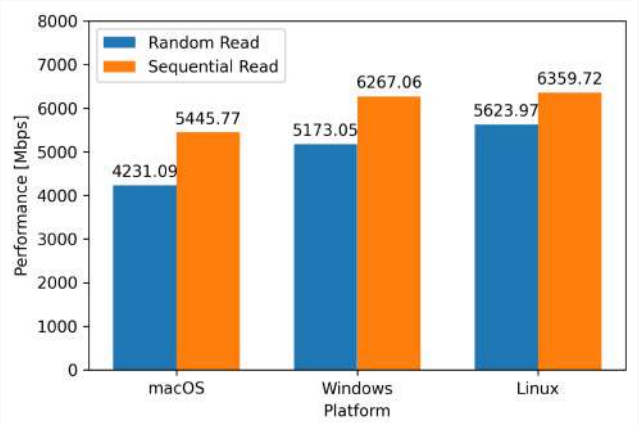

Figura 3: Leitura aleatória e sequencial (Mbps).

Desempenho de I/O (Disco) com Sysbench – Escrita

Escrita (Sequencial e Aleatória)

- **macOS** foi drasticamente superior.
- Linux e Windows tiveram baixo desempenho em escrita aleatória.
- Causa provável: drivers de SSD não otimizados.

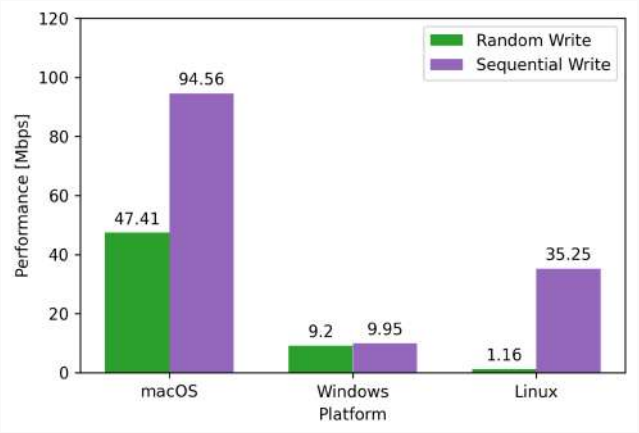

Figura 4: Escrita aleatória e sequencial (Mbps).

Desempenho de Rede: Comunicação Inter-Contêiner

- Testes realizados com **Iperf3**.
- **Windows** apresentou o melhor vazão, ideal para aplicações distribuídas.
- Linux e macOS tiveram desempenho similar entre si, com 0% de perda de pacotes UDP.

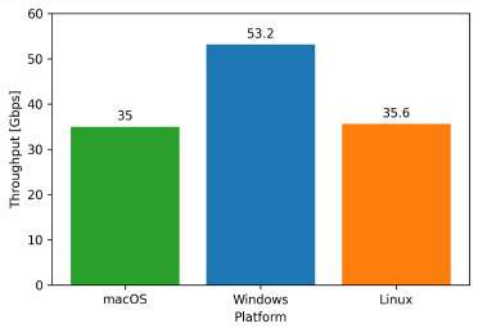

Figura 5: Vazão TCP Inter-Contêiner (Gbps).

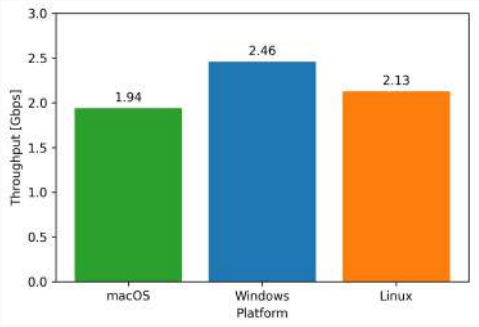

Figura 6: Vazão UDP Inter-Contêiner (Gbps).

Desempenho de Rede: Comunicação Host-Contêiner

- Teste de vazão TCP com **Iperf3**.
- **Linux** foi esmagadoramente superior.
- A arquitetura nativa do Docker no Linux evita as camadas de virtualização de rede (VPNKit) presentes no macOS e Windows.
- Ideal para serviços expostos ao host.

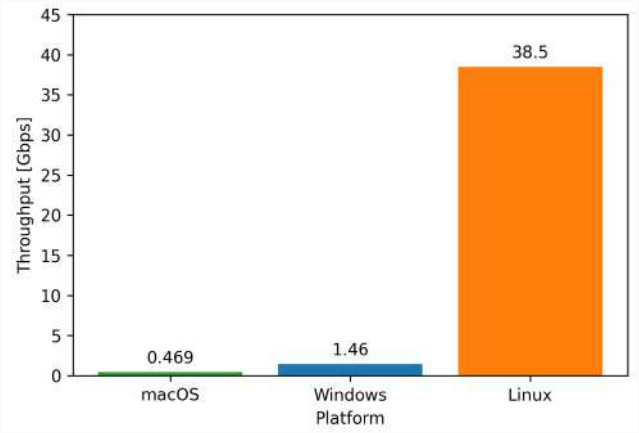

Figura 7: Vazão TCP Host-Contêiner (Gbps).

Benchmark Complexo: 7zip (Compressão)

- Avalia CPU e I/O de forma integrada.
- Resultados muito próximos entre os três sistemas.
- Linux mostrou uma pequena vantagem ao usar todos os 8 threads lógicos.
- O overhead da virtualização no macOS e Windows é mínimo para esta tarefa.

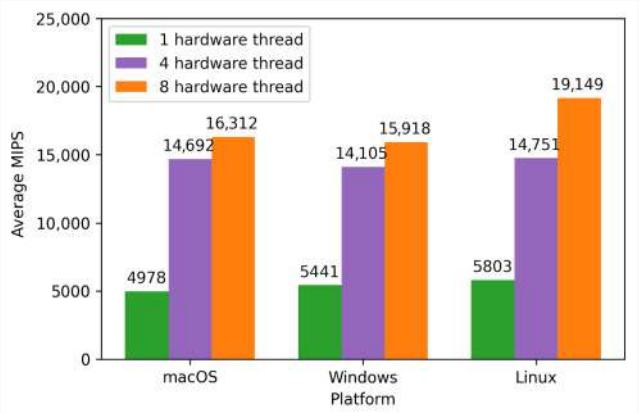

Figura 8: Desempenho do 7zip (MIPS) com 1, 4 e 8 núcleos.

Benchmark Complexo: PostgreSQL (pgbench)

- **macOS** teve um desempenho absurdamente superior (200x melhor).
- Este resultado está fortemente correlacionado com o desempenho de **escrita aleatória** em disco.
- O gargalo de I/O no Linux e Windows torna-os inadequados para bancos de dados com escrita intensiva nesta plataforma.

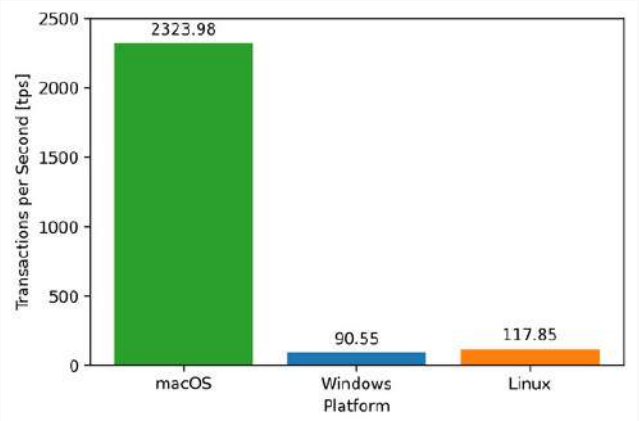

Figura 9: Desempenho do pgbench (transações por segundo).

Benchmark de Resistência: Ataque DoS (Slowloris)

- Um servidor web Apache foi atacado para medir a disponibilidade.
- O servidor no **Linux** foi o mais afetado.
- Isso indica uma maior capacidade da rede em aceitar conexões (o que, paradoxalmente, facilitou o ataque).
- O overhead de rede no macOS e Windows limitou o número de conexões maliciosas.

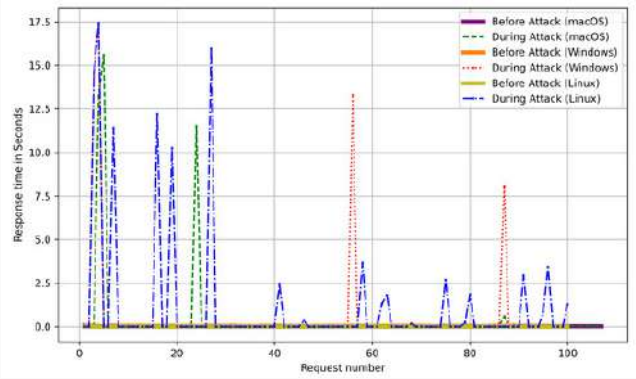

Figura 10: Tempo de resposta do Apache antes e durante o ataque.

Conclusões e Trabalhos Futuros

Veredito Final

O sistema operacional tem uma influência clara e significativa no desempenho dos contêineres Docker.

macOS
Considerado o sistema **mais versátil** para um fluxo de trabalho geral com Docker. Não sofreu com gargalos críticos, exceto por uma velocidade de rede host-contêiner mediana. Excelente para cenários de I/O intensivo como bancos de dados.

Linux
Ótima escolha para aplicações que dependem de CPU e rede (especialmente host-contêiner), mas que **não realizam operações de escrita intensiva em disco**. Ideal para bancos de dados em memória, streaming e sites estáticos.

Windows
Destaca-se no vazão de rede **inter-contêiner**, sendo uma boa opção para sistemas distribuídos. Similar ao Linux, não é recomendado para contêineres com alto uso de disco devido ao baixo desempenho de escrita.

Trabalhos Futuros

O estudo sugere várias direções para pesquisas futuras:

- Replicar os testes em um computador Apple sem o chip T2 para usar uma distribuição Linux oficial.
- Mudar a plataforma de teste para hardware não-Apple (usando um "Hackintosh"), embora isso possa introduzir outros problemas de compatibilidade.
- Realizar testes em uma plataforma com arquitetura **ARM** (ex: Apple Silicon M1/M2).
- Utilizar múltiplas plataformas de teste para validar os resultados.
- Comparar o desempenho não apenas com o SO base, mas também com outras tecnologias de contêineres e VMs tradicionais.
- Incluir benchmarks mais complexos que estressem CPU, I/O e rede simultaneamente.

Obrigado!

Perguntas?