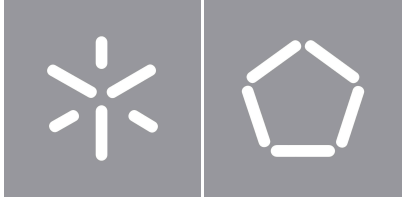
University of Minho
School of Engineering

Diogo Alexandre Correia Marques

Realistic Benchmarking of Data Deduplication and Compression Systems

september 2025



University of Minho
School of Engineering

Diogo Alexandre Correia Marques

Realistic Benchmarking of Data Deduplication and Compression Systems

Master's Dissertation in Informatics Engineering

Dissertation supervised by
João Tiago Medeiros Paulo

september 2025

Abstract

Data deduplication is a technique for identifying and removing duplicate content in storage systems, thereby contributing to better use of available space and consequent cost reduction. In fact, modern systems combine compression techniques to extract greater data density and store only what is strictly necessary.

By supporting various data manipulation techniques, the evaluation of these systems becomes increasingly complex, given that workloads need to meet a series of criteria that validate deduplication and compression simultaneously, without forgetting that the common characteristics between systems must remain the focus of the evaluation, in particular the spatial and temporal locality of accesses.

However, the benchmarks available by the community (**FIO**, vdbench) only allow partial manipulation of entropy and deduplication levels. Furthermore, trace simulation is too simplistic and becomes impractical in modern systems because they are too fast to complete the trace and there is no trivial way to extend it and preserve its characteristics.

Furthermore, in order to extract maximum performance, some systems only provide low-level **APIs**, such as **SPDK**, which makes it even more complicated to execute workloads, as the aforementioned benchmarks do not directly support such protocols for communication with the disk.

That said, this dissertation aims to develop a benchmark for storage systems, capable of supporting various **I/O** interfaces, as well as generating realistic workloads that allow the collection of relevant metrics for system evaluation, thus enabling the identification of performance bottlenecks and impacts associated with the characteristics of the system itself.

Keywords storage system, I/O interface, deduplication, compression, realistic workload.

Resumo

A deduplicação de dados corresponde a uma técnica para identificar e remover conteúdos duplicados em sistemas de armazenamento, contribuindo assim para uma melhor utilização do espaço disponível e consequente redução de custos. De facto, os sistemas modernos combinam técnicas de compressão para extrair maior densidade dos dados e armazenar o estritamente necessário.

Ao suportarem várias técnicas de manipulação de dados, a avaliação destes sistemas torna-se cada vez mais complexa, dado que as workloads necessitam de responder a uma série de critérios que validem a deduplicação e compressão em simultâneo, sem esquecer que as características comuns entres sistemas devem permanecer no alvo da avaliação, em particular a localidade espacial e temporal dos acessos.

No entanto, os benchmarks disponíveis pela comunidade (**FIO**, vdbench) apenas permitem uma manipulação parcial dos níveis de entropia e deduplicação. Ademais, a simulação de traces é demasiado simplista e torna-se impraticável em sistemas modernos por estes serem demasiado rápidos a concluir o trace e não existir uma forma trivial de o estender e preservar as suas características.

Além disso, no sentido de extrair o máximo de performance, alguns sistemas disponibilizam unicamente **APIs** de baixo nível, tal como **SPDK**, o que torna ainda mais complicada a execução de workloads, pois os benchmarks anteriormente referidos não suportam diretamente tais protocolos para comunicação com o disco.

Posto isto, esta dissertação tem por objetivo desenvolver um benchmark para sistemas de armazenamento, sendo este capaz de suportar diversas interfaces de **I/O**, bem como a geração de workloads realistas que permitam a recolha de métricas relevantes para a avaliação do sistema, permitindo assim a identificação de gargalos de desempenho e impactos associados às característica do sistema em si.

Palavras-chave sistema de armazenamento, interface de I/O, deduplicação, compressão, workload realista.

Contents

1 Introdução 1
1.1 Definição do Problema e Desafios 2
1.2 Objetivos e Contribuições 3
1.3 Estrutura do Documento 3

List of Figures

List of Tables

Acronyms

I/O Input/Output.

ZFS Zettabyte File System.

FIO Flexible I/O Tester.

API Applications Programming Interface.

SPDK Storage Performance Development Kit.

bdev Block Device.

NVMe Non-Volatile Memory Express.

Capítulo 1

Introdução

Com o aumento das aplicações de inteligência artificial, a necessidade de processar e armazenar grandes quantidades de dados tornou-se cada vez mais relevante, consequentemente os sistemas de armazenamento evoluíram no sentido de oferecer uma maior eficiência de acessos e densidade dos dados.

Sistemas de armazenamento modernos, como o **ZFS**, disponibilizam uma série de recursos que procuram melhorar a performance das aplicações. Em particular, destaca-se a deduplicação - geralmente abreviada para dedup - que procura reduzir o espaço de armazenamento utilizado ao não reescrever dados que já existam. Por outro lado, a compressão também exerce um papel relevante neste sistema, permitindo aumentar a entropia dos dados ao eliminar aqueles que de algum modo podem ser obtidos através de uma amostra menor.

Neste caso em particular, estamos interessados num sistema orientado ao bloco, portanto a técnica de deduplicação tem como unidade básica um bloco de bytes, geralmente 4096 bytes, mas este valor pode variar conforme o sistema em questão. Da mesma forma, as técnicas de compressão são aplicadas ao nível do bloco (intrabloco), no entanto é possível que alguns sistemas realizem outra análise de entropia entre blocos (interbloco) para obter mais densidade.

Por outro lado, e com o objetivo de ultrapassar as limitações impostas pela stack de **I/O** do kernel, nomeadamente as recorrentes mudanças de contexto, interrupções e cópias entre user e kernel space que tornam evidentes o gargalo de desempenho, surgiram várias **APIs** que visam resolver esses mesmos problemas e geralmente funcionam sobre runtimes assíncronos.

Tendo em consideração a diversidade de técnicas que podemos encontrar num sistema de armazenamento e interfaces de **I/O** existentes para interagir com o disco, torna-se difícil avaliar um qualquer sistema nos seus pontos específicos de funcionamento, isto porque nem todos os sistemas foram desenvolvidos para servir o mesmo fim, devendo assim procurar ajustar o benchmark e workload àquilo que o sistema oferece, pois somente assim será alcançada uma análise justa e correta do seu funcionamento.

1.1 Definição do Problema e Desafios

Devido ao baixo nível de manipulação em termos de deduplicação e compressão, os benchmarks atuais não proporcionam uma avaliação correta dos sistemas de armazenamento orientados a estas técnicas, contribuindo para análises incorretas dos mesmos. Entre os fatores que justificam esta deficiência convém destacar os seguintes:

1. A geração de conteúdo duplicado deve seguir uma determinada distribuição, o que por sua vez acarreta custos ao nível da seleção dos índices e transferência de dados entre buffers, consequentemente obtemos um menor débito, e no pior dos casos não somos capazes de saturar o disco.
2. Os sistemas de armazenamento comportam-se de modo diferente conforme a distribuição de duplicados e taxas de compressão, no entanto os padrões oferecidos pelo **FIO** são bastante simplistas por realizarem deduplicação sobre o mesmo bloco, o que aumenta a localidade temporal e espacial, beneficiando indevidamente o sistema de armazenamento.
3. Embora o **FIO** suporte a várias interfaces de **I/O** síncronas e assíncronas, os restantes benchmarks oferecem uma gama muito limitada de **APIs**, tornando-se a utilização do **FIO** praticamente obrigatória, no entanto este não disponibiliza de forma direta suporte para **SPDK** e portanto dificulta a avaliação de sistemas que funcionem diretamente sobre um **NVMe**, ou seja, com bypass da stack de **I/O**.
4. Os sistemas de armazenamento evoluíram imenso nos últimos anos, consequentemente a replicação de traces antigos tornou-se praticamente instantânea e como tal não permite uma avaliação correta do sistema, além disso não existe um método conhecido para estender o trace à medida que o benchmark é executado, sendo que a extensão deverá salvaguardar as propriedades de deduplicação e compressão do trace original. No fundo isto resume-se a seguir dados reais enquanto for possível, e depois gerar sinteticamente para manter a avaliação durante o tempo que for necessário.
5. Perante a impossibilidade de acesso a dados reais, o benchmark é obrigado a seguir uma distribuição que defina padrões de acessos e operações a realizar, no entanto a maior parte das soluções é demasiado simplista e não permite testar padrões do género: `READ`, `WRITE`, `NOP`, `READ`, `FSYNC`.

Posto isto, nenhum benchmark é versátil o suficiente para permitir ao utilizador definir as suas distribuições de acesso ou taxas de duplicados, e assim avaliar corretamente as características alvo do sistema de armazenamento.

1.2 Objetivos e Contribuições

Perante os problemas mencionados anteriormente, esta dissertação tem como principal objetivo melhorar a eficiência e dotar duma maior flexibilidade os algoritmos para geração de conteúdo, em particular o respeito pelas taxas de deduplicação e compressão num sistema orientado ao bloco.

No entanto, tal condição é insuficiente para alcançar workloads realistas que efetuem uma avaliação correta do sistema, para isto é necessário replicar as propriedades de traces extraídos a partir de ambientes em produção, pois somente estes oferecem informações acerca das cargas reais.

Assim sendo, o protótipo do benchmark reúne todas as contribuições da dissertação numa arquitetura que torna as **APIs** completamente independentes da geração de conteúdo, deste modo o utilizador final usufrui das seguintes vantagens:

1. O funcionamento do benchmark não depende de implementações concretas das interfaces de **I/O**, o sistema apenas define uma interface para operações de **I/O**, consequentemente qualquer programador pode estender o benchmark para usufruir de uma **API** totalmente customizada.
2. A geração de conteúdo é dividida em vários módulos, cada um respeitante aos parâmetros da operação de **I/O** (offset, tipo de operação e conteúdo), assim diferentes estratégias e distribuições dos parâmetros são combinadas para extrair mais versatilidade das workloads.
3. Cada um dos módulos definidos anteriormente define uma interface, portanto é permitido misturar dados sintéticos com traces, ou seja, uma workload pode replicar os bytes de um traces enquanto os restantes parâmetros da operação de **I/O** são gerados sinteticamente enquanto o benchmark decorre.

Tendo um protótipo com estas características, contribuímos para que a avaliação dos sistemas de armazenamento seja efetuada como mais critério, afinal o utilizador tem a possibilidade de testar várias **APIs** e para cada uma selecionar workloads que avaliem determinadas características do sistema, tudo com o maior realismo possível, e não através de métodos simplistas como praticam grande parte das soluções atuais.

1.3 Estrutura do Documento

Os capítulos estão com o link errado, não esquecer de trocar

Este documento encontra-se dividido em três capítulos, o **Capitulo 1** serve de introdução ao problema abordado na dissertação, procurando desvendar os desafios inerentes ao mesmo, sendo ainda apontadas as contribuições que se pretendem alcançar.

Já o **Capítulo 2** apresenta o background relativo à deduplicação e compressão, em particular as técnicas aplicadas para gerar conteúdo com estas propriedades, além disso a stack de **I/O** é explorada para justificar as diferenças entre **APIs** e perceber os pontos de melhoria em soluções de benchmark já estabelecidas (DEDISBench e DEDISBench++).

Por fim, o **Capitulo 3** corresponde a uma visão geral da arquitetura da solução, especificando os fluxos entre componentes e configurações necessárias à descrição da workload por parte do utilizador, concluindo-se com a descrição do plano para o restante da dissertação.