Compressão de Dados para Redução de E/S em Simulações Sísmicas de Alto Desempenho *

Cristiano A. Künas[1], Gabriel Freytag[1], Thiago Araújo[1], Rodrigo Machado[1], Bruno Morales[1], Alexandre Sardinha[2], Philippe O. A. Navaux[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)

[2]Petroleo Brasileiro S.A., Rio de Janeiro, Brazil

{cakunas, gfreytag, tsaraujo, rcmachado, bmmorales, navaux}@inf.ufrgs.br

alexandre.sardinha@petrobras.com.br

Resumo. *Simulações sísmicas de alta fidelidade geram grandes volumes de dados, tornando a E/S um gargalo crítico em aplicações de HPC. Este trabalho avalia o impacto da compressão de dados na redução desse gargalo, aplicando ferramentas de compressão sobre conjuntos de dados gerados por um modelo sísmico em diferentes escalas. A análise considera taxa de compressão, tempo de compressão e descompressão, e impacto no tempo total de execução. Os resultados mostram que o fpzip oferece maior compressão, enquanto o lz4 apresenta melhor desempenho em tempo de execução, e o zfp fornece um equilíbrio entre ambos. Também discutimos os desafios para aceleração em GPU, destacando a maturidade da biblioteca nvCOMP para GPUs NVIDIA e a ausência de soluções equivalentes para o ecossistema AMD, indicando caminhos para futuras otimizações em ambientes heterogêneos.*

1. Introdução

O avanço dos sistemas de computação de alto desempenho (HPC) tem viabilizado simulações em larga escala em diversos campos científicos e de engenharia, incluindo a indústria de petróleo e gás. Entre essas aplicações, a Migração Reversa no Tempo (RTM) destaca-se como técnica fundamental para geração de imagens sísmicas na exploração geofísica. Contudo, sua execução produz volumes massivos de dados que demandam alta capacidade de armazenamento e largura de banda de entrada e saída (E/S). A alta intensidade computacional, associada às exigências de estabilidade numérica e controle de dispersão na solução da equação de onda discreta, torna a RTM um processo demorado e agrava os desafios de gerenciamento e movimentação de dados [Qawasmeh et al. 2017, Serpa et al. 2021, Barbosa and Coutinho 2022].

O manuseio ineficiente desses dados pode criar gargalos críticos de I/O, prolongando os tempos de simulação e aumentando os custos operacionais [Künas et al. 2024]. Para mitigar tais problemas, estratégias avançadas de compressão — tanto sem perdas (lossless) quanto com perdas controladas (lossy) — podem ser integradas a workflows de HPC. A compressão sem perdas garante a reconstrução exata dos dados originais, assegurando reprodutibilidade científica. Já a compressão com perdas, quando baseada em

*O presente trabalho foi realizado com apoio da Petrobras, concessão n.º 2020/00182-5, CNPq e Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Alguns experimentos deste trabalho utilizaram os recursos da infraestrutura PCAD, http://gppd-hpc.inf.ufrgs.br, no INF/UFRGS.

técnicas de controle de erro (error-bounded), permite sacrificar parte da precisão numérica de forma limitada e previsível, mantendo a validade científica dos resultados. Por exemplo, bibliotecas como ZFP e cuSZ permitem definir tolerâncias absolutas ou relativas de erro, de modo que o impacto introduzido pela compressão seja inferior ao ruído natural dos modelos ou aceitável para análises estatísticas subsequentes.

Além das soluções baseadas em CPU, o uso de GPUs abre novas possibilidades ao aplicar compressão diretamente na memória do dispositivo, reduzindo custos de movimentação de dados e potencialmente acelerando a execução. Contudo, esse cenário ainda apresenta desafios. No ecossistema CUDA/NVIDIA, bibliotecas como a nvCOMP oferecem suporte maduro à compressão acelerada em GPU. Já no ecossistema ROCm/AMD, ainda há carência de soluções equivalentes, limitando a portabilidade em ambientes heterogêneos de HPC.

Neste trabalho, o foco principal é a avaliação de ferramentas de compressão lossless em CPU aplicadas a dados científicos em ponto flutuante, com ênfase em três perfis distintos: **fpzip**, representando algoritmos especializados em taxa de compressão para dados científicos; **zfp**, oferecendo flexibilidade entre modos com e sem perdas, permitindo comparações futuras com compressão controlada; **lz4**, uma solução genérica de alta velocidade. Complementarmente, conduzimos experimentos preliminares com a biblioteca nvCOMP em GPUs NVIDIA, com o objetivo de mensurar ganhos práticos de aceleração. Embora limitados a esse ecossistema, esses resultados servem como referência inicial para futuras extensões em GPUs AMD e em cenários de compressão com perdas.

Dessa forma, este artigo contribui com: (i) uma avaliação experimental sistemática de ferramentas de compressão lossless em CPU no contexto de simulações sísmicas; (ii) uma análise de trade-offs entre taxa de compressão, custo computacional e impacto no tempo total de execução; e (iii) uma discussão preliminar sobre os desafios e oportunidades de aceleração em GPU, destacando lacunas existentes para ambientes heterogêneos.

2. Compressão de Dados

A compressão de dados é uma estratégia fundamental para mitigar gargalos de armazenamento e E/S em aplicações de HPC que manipulam grandes volumes de dados, como simulações para exploração de petróleo. O objetivo é reduzir o tamanho dos arquivos removendo ou diminuindo de redundâncias, acelerando transferências e otimizando o uso de recursos computacionais sem comprometer a integridade científica [Chen et al. 2024].

2.1. Fundamentos da Compressão

A compressão de dados consiste em converter um fluxo de entrada em um fluxo menor, explorando padrões e redundâncias para gerar uma representação compacta. O processo pode ser aplicado tanto em arquivos de disco quanto em *buffers* de memória, sendo especialmente relevante em *workflows* científicos que produzem e transferem grandes volumes de dados. Existem duas classes principais de algoritmos de compressão:

- **Compressão *lossless*** (sem perdas): preserva exatamente todos os valores originais, permitindo reconstrução perfeita dos dados. É a abordagem preferida em aplicações científicas críticas, onde a reprodutibilidade e a precisão numérica absoluta são requisitos fundamentais.
- **Compressão *lossy*** (com perdas): sacrifica parte da informação para atingir taxas de compressão mais agressivas. A confiabilidade científica pode ser preservada

quando são empregados modelos de erro controlado (error-bounded), que garantem que o erro introduzido esteja abaixo de um limite especificado. Nesses casos, a qualidade dos resultados permanece dentro da margem de tolerância definida pelo domínio da aplicação. [Elakkiya and Thivya 2022].

Em *workflows* científicos críticos, a preferência normalmente recai sobre métodos sem perdas ou quase sem perdas, garantindo a reprodutibilidade dos resultados.

2.2. Desafios em Ambientes Heterogêneos

O crescente uso de arquiteturas heterogêneas em HPC introduz novos desafios para compressão de dados científicos. Com a popularização das GPUs, surgiram bibliotecas voltadas à compressão acelerada por hardware. O principal destaque atual é a nvCOMP, desenvolvida pela NVIDIA, que disponibiliza compressão com e sem perdas para GPUs CUDA, permitindo compressão diretamente na memória da GPU e integração com *pipelines* paralelos [NVIDIA 2025]. Estudos de caso indicam que o uso de nvCOMP pode resultar em *throughput* várias vezes superior ao obtido com ferramentas em CPU, principalmente em grandes volumes de dados [Peñaranda et al. 2024, Azami et al. 2025].

Por outro lado, o ecossistema ROCm da AMD [AMD 2025] ainda carece de uma biblioteca amplamente consolidada e de alto desempenho que ofereça suporte estável a compressão de dados científicos. Embora existam iniciativas emergentes para dispositivos AMD, essas soluções ainda não atingem o nível de maturidade e integração encontrado na *stack* CUDA. Isso representa um gargalo técnico significativo para aplicações que buscam portabilidade entre arquiteturas ou executam em ambientes heterogêneos—uma tendência crescente em HPC e *cloud computing*. A escolha da ferramenta de compressão depende fundamentalmente do equilíbrio entre taxa de compressão, desempenho computacional e tolerância a erros, considerando as características específicas dos dados científicos e as restrições do ambiente de execução.

3. Trabalhos Relacionados

A compressão de dados científicos em ponto flutuante tem sido amplamente investigada como estratégia essencial para mitigar gargalos de armazenamento e E/S em aplicações de HPC. Essa área abrange tanto algoritmos *lossless* quanto *lossy*, executados em CPUs e GPUs, com foco crescente em portabilidade e aceleração em tempo real. A seguir, revisamos trabalhos relevantes organizados por temática, destacando lacunas que motivam este estudo.

3.1. Algoritmos *Lossless* para HPC

Diversas soluções foram propostas para compressão sem perdas de dados científicos. O **fpzip** [Lindstrom 2017] destaca-se como uma solução especializada para *arrays* multidimensionais de ponto flutuante, aplicando codificação preditiva e algoritmos Huffman otimizados para padrões científicos. Sua eficácia é particularmente notável em conjuntos de dados com correlação espacial ou temporal. O **zfp** [Lindstrom et al. 2024] oferece versatilidade através de modos adaptativos com e sem perdas, proporcionando controle granular sobre a relação entre taxa de compressão e precisão. Sua arquitetura baseada em blocos multidimensionais o torna adequado para diversos tipos de simulações científicas. O **lz4** [Collet 2025] embora não otimizado para características científicas, é amplamente utilizado como baseline pela sua extrema velocidade, sendo útil em cenários de E/S intensiva.

Estudos comparativos mostram ainda a adoção de alternativas como zstd e zlib, que fornecem compressão lossless genérica. Entretanto, análises recentes [Sruthi and Meena 2025] indicam que, embora possam oferecer taxas de compressão razoáveis, esses algoritmos não são competitivos em desempenho para dados científicos em larga escala. Assim, fpzip, zfp e lz4 permanecem como representativos de três perfis distintos: taxa de compressão, maior versatilidade e maior velocidade.

3.2. *Benchmarks* e análises comparativas

Para orientar a escolha de algoritmos em HPC, surgiram *benchmarks* dedicados. O **FC-Bench** [Chen et al. 2023a] oferece uma avaliação abrangente de métodos *lossless* em dados científicos, considerando taxas de compressão, estabilidade e até métricas energéticas. Já Chen et al. [Chen et al. 2023b] expandiram a análise para aspectos além da compressão, como integração em workflows complexos, reforçando a importância de avaliar impacto no tempo total de execução e não apenas em métricas isoladas.

No campo CPU-GPU, Azami et al. [Azami et al. 2025] investigaram a portabilidade entre implementações, mostrando que escolhas arquiteturais afetam significativamente o desempenho. Esses resultados motivam análises específicas em cenários como simulações sísmicas, onde a compressão pode representar fração significativa do tempo de execução.

Embora ricos em métricas, esses benchmarks raramente exploram dados sísmicos em larga escala. O presente trabalho se diferencia ao adotar uma aplicação real de RTM como caso de estudo, oferecendo evidências práticas para o domínio de geofísica.

3.3. Compressão *lossy* controlada

A compressão com perdas, quando controlada por bounds de erro, tem atraído atenção em HPC por viabilizar reduções mais agressivas. O **cuSZ** [Liu et al. 2023] consolidou-se como referência em GPUs, permitindo controle absoluto/relativo do erro. Extensões como cuSZ-I, FZ-GPU e FRaZ [Zhao et al. 2022, Chen et al. 2023c] buscaram melhorar a eficiência através de técnicas como interpolação multiescala e compressão hierárquica.

Estudos de casos [Liu et al. 2020, Liu et al. 2022] demonstram que erros introduzidos podem ser toleráveis ou até irrelevantes em aplicações como visualização, checkpointing e análises estatísticas. No entanto, esses trabalhos concentram-se em domínios de clima e cosmologia, ainda pouco explorando cenários sísmicos — onde pequenas alterações podem impactar a interpretação geológica. Esse é um ponto que diferencia nosso foco, que neste estágio restringe-se à compressão lossless.

3.4. Soluções em GPU e lacunas atuais

O crescimento do uso de GPUs em HPC impulsionou bibliotecas como o nvCOMP [NVIDIA 2025], que suporta compressão lossless e lossy diretamente na memória da GPU com elevado throughput. Outras propostas, como ndzip-gpu [Knorr et al. 2021], exploram transformações matemáticas otimizadas para paralelismo massivo.

Entretanto, tais soluções permanecem fortemente atreladas ao ecossistema NVIDIA. No caso das GPUs AMD, o ecossistema ROCm ainda carece de bibliotecas maduras para compressão científica, criando um gap relevante para aplicações heterogêneas. Trabalhos como Peñaranda et al. [Peñaranda et al. 2024] discutem bibliotecas híbridas CPU-GPU, mas ainda não cobrem adequadamente o ambiente AMD. Essa lacuna é diretamente relacionada à motivação do presente estudo, que visa mapear o estado atual e indicar caminhos para futuras contribuições voltadas à portabilidade.

4. Compressão de Dados em Simulações Sísmicas

A metodologia adotada neste trabalho foi estruturada em múltiplas etapas, com o objetivo de avaliar o desempenho de diferentes ferramentas de compressão de dados aplicadas a simulações sísmicas. O processo envolveu a seleção de uma aplicação representativa do domínio de extração de petróleo, a geração de conjuntos de dados em ponto flutuante sob diferentes condições, a aplicação de algoritmos de compressão e a análise de métricas de desempenho associadas.

4.1. Aplicação Alvo e Geração dos Dados

A aplicação escolhida para compor o cenário experimental é o método Fletcher, uma técnica computacional amplamente utilizada em exploração sísmica [Fletcher et al. 2009, Künas et al. 2024]. O método simula a propagação de ondas sísmicas no subsolo terrestre, levando em conta variações na velocidade das ondas em diferentes camadas geológicas. O processo de aquisição de dados sísmicos começa com o envio de ondas acústicas por embarcações equipadas com cabos com fontes e receptores. Essas ondas se propagam pelas camadas subterrâneas, sofrendo reflexões e refrações, e os sinais retornados são capturados, oferecendo informações para a identificação de reservas de petróleo e gás.

No núcleo da aplicação está a solução numérica da equação de ondas, que considera propriedades elásticas do meio, como a velocidade de propagação. Essa solução é computada sobre uma malha tridimensional e envolve um processo iterativo: a cada etapa, o algoritmo estima o estado das ondas em cada ponto da malha com base em estados anteriores, capturando a complexa interação das ondas com as diferentes camadas geológicas, incluindo efeitos de dispersão e difração.

Com base na aplicação de simulação sísmica, a geração dos dados experimentais foi conduzida para representar diferentes cargas de trabalho e níveis de complexidade da simulação. Três conjuntos de dados foram produzidos, classificados conforme o tamanho:

- **Pequeno**: malha de 216×216×216 células, com 10 operações de escrita. Essa configuração corresponde a uma simulação menos complexa, gerando cerca de **0,74 GB** de dados.
- **Médio**: mesma malha de 216×216×216, mas com 100 operações de escrita, resultando em aproximadamente 6,78 GB.
- **Grande**: malha expandida para 408×408×408 células, com 200 operações de escrita, gerando cerca de 72,3 GB de dados.

Essa variação permitiu avaliar o desempenho das ferramentas de compressão sob diferentes cenários de volume e granularidade dos dados, simulando desde execuções rápidas até cargas realistas de simulações geofísicas em grande escala.

4.2. Ferramentas de Compressão e Métricas Avaliadas

Para a compressão dos dados gerados, foram selecionadas três ferramentas amplamente utilizadas na comunidade científica para compressão em CPU: **fpzip**, **zfp** e **lz4**. As duas primeiras ferramentas são especializadas em dados de ponto flutuante, enquanto a última é um algoritmo genérico de alta performance. Essas ferramentas foram escolhidas por representarem diferentes estratégias de compressão e atenderem a distintos perfis de aplicação — desde cenários que exigem precisão absoluta até aqueles em que o desempenho computacional é o fator prioritário.

A avaliação das ferramentas de compressão foi conduzida com base em métricas que refletem os principais compromissos em aplicações de HPC. As métricas selecionadas permitem uma análise abrangente do impacto da compressão tanto em termos de eficiência quanto de viabilidade para uso em simulações científicas.

- **Taxa de compressão**: representa a relação entre o tamanho original dos dados e o tamanho após a compressão. Quanto maior o valor, maior a economia de espaço em disco e potencial redução no tráfego de E/S. É uma métrica central em ambientes onde armazenamento e transferência de dados são gargalos críticos.
- **Tempo de compressão e descompressão**: em segundos, indica o custo computacional da aplicação dos algoritmos. Como todas as ferramentas avaliadas operam em CPU, essa métrica revela a sobrecarga imposta ao pipeline da simulação.

Essas métricas fornecem uma base sólida para caracterizar o comportamento de cada ferramenta em diferentes cenários de carga e complexidade. Além disso, estabelecem um referencial que pode ser estendido para comparar implementações aceleradas por GPU ou híbridas, onde o balanceamento entre compressão, tempo de execução e uso de recursos computacionais torna-se ainda mais crítico.

4.3. Ambiente Computacional e Execução dos Testes

Os experimentos foram conduzidos em uma plataforma computacional baseada em arquitetura x86 com CPU AMD, equipada com processador Ryzen Threadripper PRO 5965WX (24 núcleos, 48 threads), 256 GB de memória RAM e dois SSDs NVMe de 1 TB, utilizados para isolar os dados de entrada/saída e minimizar gargalos de E/S do sistema de arquivos. Cada experimento foi executado 10 vezes, e os tempos reportados correspondem à média com intervalo de confiança de 95%, estimado pela distribuição t de Student. Esse procedimento garante robustez estatística aos resultados, permitindo comparações confiáveis entre as ferramentas e cenários de carga.

5. Resultados Experimentais

Nesta seção, apresentamos os resultados experimentais das ferramentas fpzip, lz4 e zfp em três conjuntos de dados de tamanhos distintos: grande ($\approx 72,3$GB), médio ($\approx 6,78$GB) e pequeno ($\approx 0,74$GB). Além disso, todos os experimentos foram realizados com configuração sem perdas (*lossless*). Para cada ferramenta, medimos o tempo médio de compressão e descompressão, os tamanhos dos dados antes e após a compressão, juntamente com a respectiva taxa de compressão. A seguir, discutimos os resultados obtidos.

5.1. Avaliação de Compressão *Lossless* em CPU

A Figura 1 apresenta o tempo de execução de cada ferramenta nos respectivos tamanhos dos dados de entrada. Para os dados de tamanho grande ($\approx 72,3$ GB), o **fpzip** obteve a maior taxa de compressão ($\approx 1,865$), reduzindo o volume para $\approx 38,75$GB. Em contrapartida, apresentou um tempo significativamente alto de compressão (633s) e descompressão (684s). Em contraste, o **lz4** destacou-se pela velocidade — 89s para compressão e 75s na descompressão - mas menor taxa ($\approx 1,373$), resultando em $\approx 52,64$GB pós-compressão. O **zfp**, com taxa similar ($\approx 1,38$), teve o maior tempo (quase 1195s), embora com descompressão mais rápida que o fpzip (127s), mas mais lento que lz4.

O padrão geral se mantém nos dados médios ($\approx 6,78$GB), embora diferenças de tempo tenham sido menores em termos absolutos. O **fpzip** novamente obteve a melhor taxa de compressão ($\approx 1,734$), reduzindo os dados para $\approx 3,91$GB, com tempos de

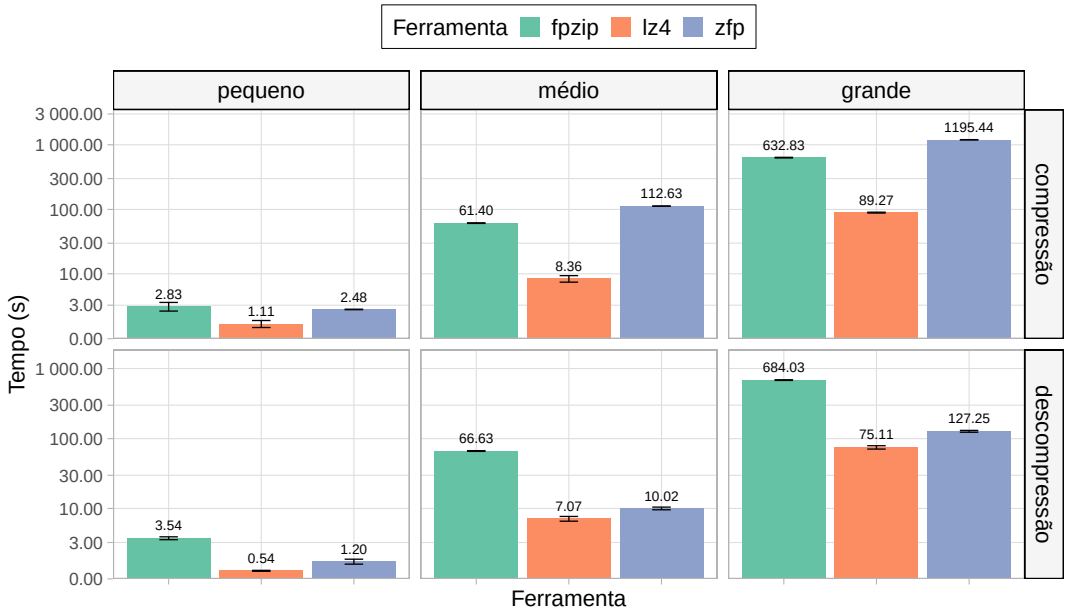


Figura 1. Desempenho de compressão e descompressão de fpzip, lz4 e zfp em conjuntos de dados grandes, médios e pequenos.

compressão e descompressão de 61s e 67s, respectivamente. O **lz4** concluiu ambas as etapas significativamente mais rápido — pouco menos de 8 s e 7 s — mas com taxa inferior ($\approx 1,342$). O **zfp** foi intermediário, com taxa de $\approx 1,305$, compressão mais lenta ($\approx$ 113s) e descompressão mais rápida que o fpzip ($\approx$ 10s). O mesmo trade-off é evidente: o lz4 prioriza velocidade, o fpzip prioriza compressão e o zfp mantém um meio-termo.

Nos conjuntos menores ($\approx 0,74$ GB), todos os tempos de execução foram baixos, mas os padrões anteriores persistem. O **fpzip** atingiu a maior taxa ($\approx 11,923$), reduzindo os dados para $\approx$ 0,062GB, em 2,84s (compressão) e 3,54s (descompressão). O **lz4** foi o mais rápido (1,10s/0,54s), com taxa levemente inferior ($\approx 11,243$), gerando um arquivo pós-compressão de 0,066GB. O **zfp** apresentou a menor taxa ($\approx 8,035$), com tempos intermediários (2,48s/1,20s). Esses resultados indicam que, mesmo em volumes pequenos, as características de cada ferramenta são consistentes.

A Figura 3 apresenta o trade-off entre taxa de compressão e tempo de execução. O eixo vertical representa a taxa de compressão (quanto maior, menor o tamanho pós-compressão dos dados), enquanto que o eixo horizontal mostra o tempo médio de compressão em escala logarítmica (quanto menor, mais rápida a execução da compressão). Como é possível observar, **fpzip** aparece consistentemente na região de maior taxa de compressão, mas deslocado para tempos de execução maiores. Isso confirma seu perfil para economia de espaço, mas à custa de maior sobrecarga computacional, sobretudo em dados médios e grandes. Já o **lz4** ocupa o extremo oposto, nas posições de menor tempo, ou seja, é significativamente mais eficiente em termos de tempo de compressão, mas com taxas de compressão ligeiramente inferiores.

Para cada combinação de tamanho (pequeno, médio, grande) e operação (compressão, descompressão), identificamos o pior desempenho em tempo de operação e taxa de compressão. A partir disso, calculamos a razão de melhoria relativa:

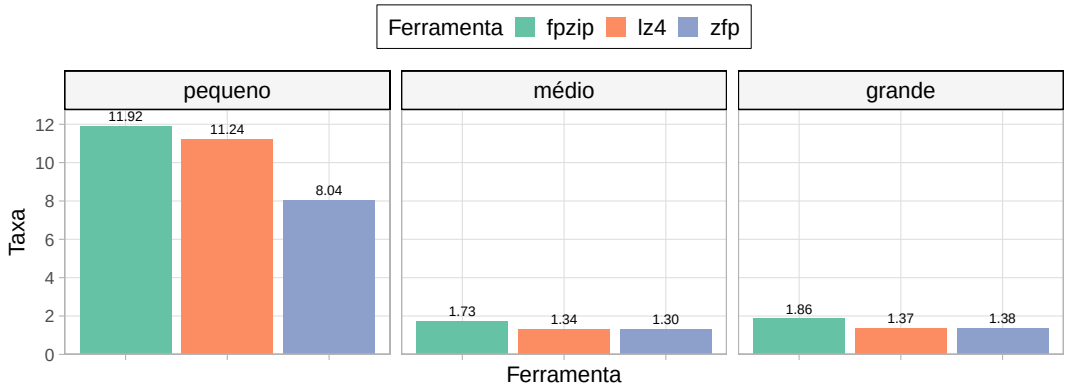


Figura 2. Comparação da taxa de compressão de fpzip, lz4 e zfp para conjuntos de dados grandes, médios e pequenos.

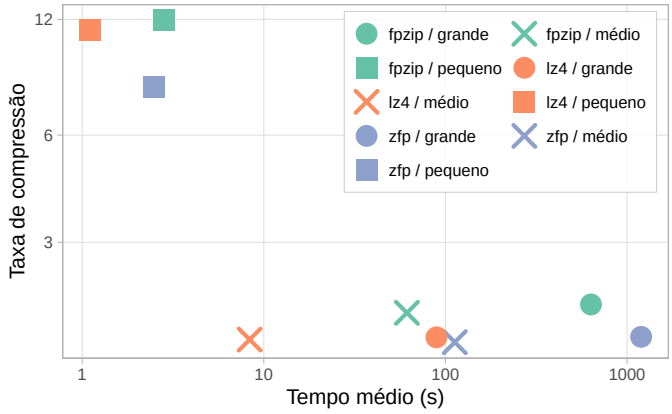


Figura 3. Trade-off entre taxa e tempo de compressão

$$\text{Melhor(tempo)} = \frac{\text{Pior tempo do grupo}}{\text{Tempo da ferramenta}}, \text{Melhor(taxa)} = \frac{\text{Taxa da ferramenta}}{\text{Pior taxa do grupo}}$$

A Tabela 1 apresenta os resultados desses cálculos, onde valores maiores que 1 indicam quantas vezes uma ferramenta é mais veloz ou mais eficiente em dada operação. Após analisarmos a coluna de melhor tempo e melhor taxa de compressão, fica claro a posição de destaque para a ferramenta **lz4** no quesito eficiência de tempo e para a ferramenta **fpzip** no quesito taxa de compressão nos três conjuntos distintos de dados avaliados e em ambas as operações.

5.2. Avaliação Preliminar de Compressão *Lossless* em GPU

Além das análises conduzidas em CPU, realizamos experimentos preliminares em uma GPU NVIDIA RTX 4090, utilizando a biblioteca nvCOMP (LZ4) para compressão diretamente na memória do dispositivo. O objetivo foi quantificar os ganhos em tempo total de execução e no volume de saída, considerando os três cenários distintos de simulação sísmica anteriores: pequeno, médio e grande. Cada configuração foi novamente executada 10 vezes, e os valores reportados correspondem às médias.

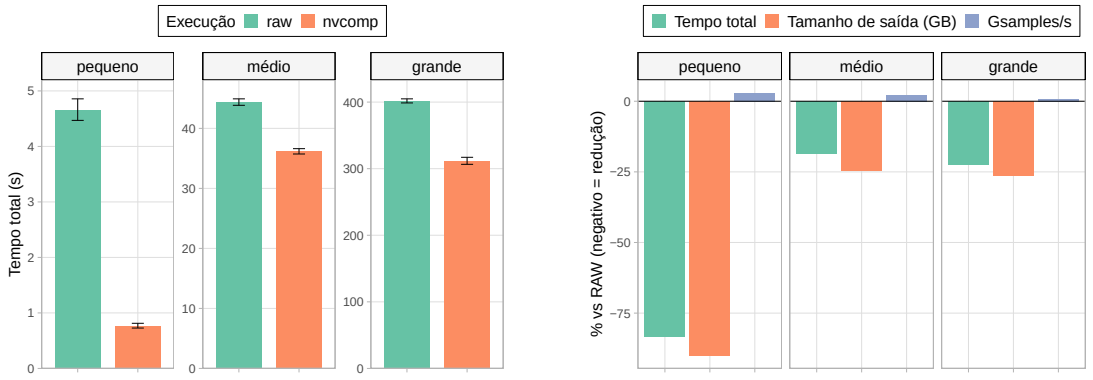
Os resultados preliminares com a biblioteca nvCOMP em GPU RTX 4090 demonstram ganhos expressivos tanto em tempo de execução para compressão quanto em tamanho final dos dados comprimidos. Observa-se que, para o caso pequeno, o tempo total foi reduzido em aproximadamente 83% e o volume de dados escrito caiu em cerca

Tabela 1. Resumo de tempo e taxa de compressão ($\times$ melhor comparado ao pior)

Tamanho	Operação	Ferramenta	Tempo (s)	Melhor(tempo)	Taxa	Melhor(taxa)
grande	**compressão**	fpzip	632.83	1.89	1.865	**1.36**
grande	compressão	lz4	89.27	**13.39**	1.373	1.00
grande	compressão	zfp	1195.44	1.00	1.380	1.01
grande	**descompressão**	fpzip	684.03	1.00	1.865	**1.36**
grande	descompressão	lz4	75.11	**9.10**	1.373	1.00
grande	descompressão	zfp	127.25	5.38	1.380	1.01
médio	**compressão**	fpzip	61.40	1.84	1.734	**1.33**
médio	compressão	lz4	8.36	**13.47**	1.342	1.03
médio	compressão	zfp	112.63	1.00	1.305	1.00
médio	**descompressão**	fpzip	66.63	1.00	1.734	**1.33**
médio	descompressão	lz4	7.07	**9.42**	1.342	1.03
médio	descompressão	zfp	10.02	6.65	1.305	1.00
pequeno	**compressão**	fpzip	2.84	1.00	11.923	**1.48**
pequeno	compressão	lz4	1.11	**2.56**	11.243	1.40
pequeno	compressão	zfp	2.48	1.14	8.035	1.00
pequeno	**descompressão**	fpzip	3.54	1.00	11.923	**1.48**
pequeno	descompressão	lz4	0.54	**6.60**	11.243	1.40
pequeno	descompressão	zfp	1.20	2.96	8.035	1.00

de 90%, evidenciando o forte impacto da compressão em cenários de baixa carga. No conjunto médio, a redução no tempo foi de cerca de 18% e no volume de dados de 25%, enquanto que no conjunto grande os ganhos chegaram a 22% no tempo e 26% no volume armazenado. Esses resultados, consolidados nas Figuras 4(a) e 4(b), reforçam que a compressão em GPU pode trazer ganhos expressivos tanto em tempo total de execução da aplicação quanto em eficiência de E/S em aplicações de simulação petrolífera da indústria, ainda que a magnitude varie conforme o tamanho do problema. Além destes ganhos, também houve uma leve melhora no desempenho em GSamples/s computados de até 3%.

Além da redução em tempo de execução, a taxa de compressão é significativa. No conjunto pequeno, a taxa de compressão é semelhante ao observado em CPU, atingindo uma taxa de $\approx 9,94$, que explica a redução no volume de dados e no tempo total. Nos tamanhos médio e grande, as taxas de compressão foram mais modestas ($\approx 1,33$ e $\approx 1,36$), mas ainda suficientes para gerar reduções no armazenamento. Mesmo nos casos



(a) Tempo total de simulação. (b) % vs RAW

Figura 4. Impacto da compressão no tempo total de simulação, tamanho final do binário (em GB) e desempenho em GSamples.

de menor taxa de compressão, os ganhos observados em GPU indicam que a sobrecarga do processo é compensada pela economia de E/S, tornando a estratégia vantajosa em diferentes escalas de simulação.

5.3. Discussão

Os resultados apresentados evidenciam que não existe uma solução única e universal para compressão em workflows de HPC; a escolha da ferramenta depende diretamente das restrições do problema e do ambiente de execução.

No contexto da simulação sísmica avaliada, observamos três perfis distintos:

- **fpzip** privilegia a redução do volume de dados, alcançando as maiores taxas de compressão, mas com custo computacional elevado, sobretudo em conjuntos médios e grandes.
- **lz4** se destaca pela velocidade, sendo até 13× mais rápido que o pior caso em determinados cenários, mas entrega taxas de compressão mais modestas.
- **zfp**, embora versátil por suportar também modos lossy, apresentou desempenho intermediário, equilibrando taxa e tempo, mas sem superar as demais opções em lossless.

Essa heterogeneidade fica clara quando analisamos o **trade-off taxa × tempo de compressão**. A fronteira de Pareto mostra que fpzip domina quando o objetivo é maximizar a economia de armazenamento, enquanto lz4 domina quando a prioridade é minimizar a sobrecarga de execução. O zfp aparece em pontos intermediários, sugerindo que sua maior utilidade emerge em cenários onde compressão com perdas é aceitável. Assim, a decisão prática depende se o gargalo está em E/S/armazenamento (caso em que fpzip é preferível) ou em tempo de execução/quase tempo real (caso em que lz4 se torna mais vantajoso).

Outro ponto crucial é o impacto no tempo total da simulação. Nos experimentos em CPU, a compressão adiciona uma fração não desprezível ao pipeline, especialmente para conjuntos grandes, o que pode reduzir os benefícios líquidos em aplicações de longa duração. Já nos testes preliminares em GPU com a nvCOMP, a sobrecarga de compressão é amplamente compensada, levando a reduções de até 83% no tempo total e 90% no volume de saída. Esses resultados reforçam o potencial da aceleração por GPU em tornar a compressão vantajosa mesmo em cenários de alta carga.

Embora os dados usados sejam do domínio sísmico, as tendências observadas se generalizam. Dados científicos em ponto flutuante de outras áreas — como simulações de dinâmica de fluidos, modelos climáticos ou cosmológicos — apresentam padrões semelhantes de correlação espacial e temporal, o que implica que os trade-offs entre taxa e tempo de compressão observados aqui podem ser extrapolados, ainda que as magnitudes variem. Isso sugere que as conclusões deste estudo têm aplicabilidade além da geofísica.

6. Conclusões e Trabalhos Futuros

Este estudo avaliou o desempenho de três ferramentas de compressão amplamente utilizadas em CPU — fpzip, lz4 e zfp — em conjuntos de dados de ponto flutuante gerados por um aplicativo de simulação de ondas sísmicas na área de petróleo. Os resultados destacam compensações distintas entre taxa de compressão, eficiência computacional e requisitos do aplicativo.

O fpzip alcançou consistentemente as maiores taxas de compressão (até ≈ 11.92), especialmente para grandes conjuntos de dados, reduzindo significativamente as demandas de armazenamento. No entanto, seus tempos de compressão e descompressão foram notavelmente maiores, tornando-o menos adequado para aplicações em tempo real. O lz4 se destacou em velocidade (até $13\times$ melhor), proporcionando a compressão e descompressão mais rápidas em todos os tamanhos de conjuntos de dados, embora com uma taxa de compressão ligeiramente menor. Isso torna o lz4 ideal para fluxos de trabalho que priorizam a taxa de transferência em vez de reduções extremas de armazenamento. O zfp ofereceu um equilíbrio entre taxa de compressão e velocidade, mas seu desempenho foi menos competitivo em cenários sem perdas. Sua versatilidade, no entanto, o torna valioso para casos em que precisão controlada ou compressão com perdas são aceitáveis.

Além das análises conduzidas em CPU, este estudo reforça a necessidade de explorar o uso de compressão em ambientes heterogêneos com aceleração por GPU. Embora preliminares, os resultados demonstram que a compressão acelerada em GPU pode reduzir tanto o tempo de execução (até 83%) quanto o volume de saída (até 90%), sugerindo ganhos expressivos em cenários de produção.

Trabalhos futuros incluirão: (i) avaliar diferentes algoritmos da nvCOMP além do LZ4, (ii) explorar cargas de simulação maiores e (iii) comparar com soluções emergentes no ecossistema AMD/ROCm, como hipCOMP, hipLZ4 e zfp com suporte HIP, de modo a avançar em direção à portabilidade em ambientes heterogêneos. Além disso, pretendemos avançar nesta pesquisa para avaliar a compressão com perdas e o trade-off entre perda de precisão e desempenho de compressão.

Referências

AMD (2025). Amd rocm documentation. `https://rocm.docs.amd.com/en/latest/`. Accessed: 2025-06-20.

Azami, N., Fallin, A., and Burtscher, M. (2025). Efficient lossless compression of scientific floating-point data on cpus and gpus. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 395–409.

Barbosa, C. H. and Coutinho, A. L. (2022). Multi-gpu 3-d reverse time migration with minimum i/o. In *Latin American High Performance Computing Conference*, pages 160–173. Springer.

Chen, X., Tian, J., Beaver, I., Freeman, C., Yan, Y., Wang, J., and Tao, D. (2023a). Fcbench: Cross-domain benchmarking of lossless compression for floating-point data. *Proceedings of the VLDB Endowment*, 17(6):1418–1431.

Chen, X., Tian, J., Beaver, I., Freeman, C., Yan, Y., Wang, J., and Tao, D. (2024). Fcbench: Cross-domain benchmarking of lossless compression for floating-point data. *Proceedings of the VLDB Endowment*, 17(6):1418–1431.

Chen, Z., Di, S., Tao, D., and Cappello, F. (2023b). Beyond compression: A comprehensive evaluation of lossless floating-point compression. *Journal of Parallel and Distributed Computing*.

Chen, Z., Tao, D., and Cappello, F. (2023c). Fraz: A generic high-fidelity fixed-ratio lossy compression framework for scientific floating-point data. In *IEEE IPDPS*.

Collet, Y. (2025). lz4: Extremely fast compression algorithm. GitHub Repository. `https://github.com/lz4/lz4`.

Elakkiya, S. and Thivya, K. (2022). Comprehensive review on lossy and lossless compression techniques. *Journal of The Institution of Engineers (India): Series B*, 103(3):1003–1012.

Fletcher, R. P., Du, X., and Fowler, P. J. (2009). Reverse time migration in tilted transversely isotropic (TTI) media. *Geophysics*, 74(6):WCA179–WCA187.

Knorr, F., Thoman, P., and Fahringer, T. (2021). ndzip-gpu: Efficient lossless compression of scientific floating-point data on gpus. In *Proceedings SC21: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13.

Künas, C. A., Freytag, G., and Navaux, P. O. (2024). Enhancing reverse time migration simulations in hpc systems through i/o and computation overlapping. In *Latin American High Performance Computing Conference*. Springer.

Lindstrom, P., Hittinger, J., Diffenderfer, J., Fox, A., Osei-Kuffuor, D., and Banks, J. (2024). Zfp: A compressed array representation for numerical computations. *The International Journal of High Performance Computing Applications*.

Lindstrom, P. G. (2017). Fpzip. Technical report, Lawrence Livermore National Laboratory (LLNL), Livermore, CA (United States).

Liu, J., Tao, D., and Cappello, F. (2022). Understanding gpu-based lossy compression for extreme-scale cosmological simulations. *Computing in Science & Engineering*.

Liu, J., Tian, J., Wu, S., Di, S., Zhang, B., Huang, Y., Zhao, K., Li, G., Tao, D., Chen, Z., and Cappello, F. (2023). cusz: An efficient gpu-based error-bounded lossy compression framework for scientific data. *arXiv preprint arXiv:2312.05492*.

Liu, K., Di, S., Tao, D., and Cappello, F. (2020). Use cases of lossy compression for floating-point data in scientific data sets. *Computing in Science & Engineering*.

NVIDIA, C. (2025). The nvcomp library. `https://developer.nvidia.com/nvcomp`. Accessed: 2025-06-18.

Peñaranda, C., Reaño, C., and Silla, F. (2024). Hybrid-smash: a heterogeneous cpu-gpu compression library. *IEEE Access*, 12:32706–32723.

Qawasmeh, A., Hugues, M. R., Calandra, H., and Chapman, B. M. (2017). Performance portability in reverse time migration and seismic modelling via openacc. *The International Journal of High Performance Computing Applications*, 31(5):422–440.

Serpa, M. S., Pavan, P. J., Cruz, E. H., Machado, R. L., Panetta, J., Azambuja, A., Carissimi, A. S., and Navaux, P. O. (2021). Energy efficiency and portability of oil and gas simulations on multicore and graphics processing unit architectures. *CCPE*, 33(18):e6212.

Sruthi, S. and Meena, S. (2025). Comparative study of data compression algorithms: Zstandard, zlib & lz4. In *2025 International Conference on Data Engineering and Applications*.

Zhao, K., Liu, J., Tao, D., and Cappello, F. (2022). Fz-gpu: A fast and high-ratio lossy compressor for scientific computing applications on gpus. In *IEEE Cluster Conference*.