

Response Letter to Topic Editor

High-Performance Geodynamics on GPUs Using the PETSc CUDA Backend (GAUZZ v1.0.0)

[Ms. Ref. No.: egusphere-2026-2528]

[Kyeong-Min Lee¹, Deok-Kyu Jang³, Cedric Thieulot⁴, Byung-Dal So^{1,2*}]

Text in ~~red strike through~~: deleted from the manuscript.

Text in **blue**: added or revised.

Text in *italic*: comments from the Topic Editor.

Dear authors,

Thank you for addressing the comments raised by the reviewers. I want to follow up on two points before making the final decision.

Authors' reply:

We thank Topic Editor Ludovic Räss for the careful assessment of our revised manuscript and the opportunity to address the two remaining points. We have performed the requested validation analyses and revised the manuscript, as detailed below.

1. It was emphasised during revision that memory bandwidth and achieved memory throughput play the key role in assessing performance given the algorithms are most likely memory bound. Running e.g. a memory copy or similar type of benchmark on both the CPU and GPU hardware used in this study would give you a practical upper bound in what can be achieved in terms of bandwidth and may better explain the observed speed-up on the GPU (as it may be actually challenging to squeeze out the ~200 GB/s of the CPU).

Authors' reply:

We thank the editor for this suggestion. We measured the sustained memory bandwidth of the CPU and GPU using copy, scale, add, and vector update operations (Deakin et al., 2018). The vector update $a = a + s \cdot b$ is representative of operations used in the Krylov solver. As shown in the new Table S1, the measured bandwidth for this operation was 113.2 GB s⁻¹ on the 32-core CPU and 824.0 GB s⁻¹ on the GPU, a 7.3-fold difference. The CPU achieved 74% of the theoretical peak bandwidth for its installed memory configuration. We have also corrected that peak value to 153.6 GB s⁻¹, reflecting that six of the CPU's eight memory channels are populated.

Table S1: Memory bandwidth measured using vector operations on the CPU and GPU. Percentages in parentheses indicate the fraction of the theoretical peak memory bandwidth.

Operation	32-core CPU [GB s ⁻¹]	GPU [GB s ⁻¹]	GPU/CPU ratio
Copy $c = a$	115.4 (75%)	800.9 (83%)	6.9
Scale $b = s \cdot c$	104.1 (68%)	800.3 (83%)	7.7
Add $c = a + b$	109.1 (71%)	822.2 (86%)	7.5
Update $a = a + s \cdot b$	113.2 (74%)	824.0 (86%)	7.3

Added Table S1

We also revised the manuscript as follows.

Lines 126–130; Introduction,

The Xeon Gold 6338 ~~supports~~ ~~provides eight DDR4-3200 channels, giving a theoretical memory bandwidth of 204.8 GB s⁻¹~~ eight DDR4-3200 memory channels, six of which are populated in the CPU workstation used here, giving a theoretical memory bandwidth of 153.6 GB s⁻¹. The theoretical bandwidth per NVIDIA RTX 6000 Ada Generation is 960 GB s⁻¹. ~~and each NVIDIA-RTX 6000 Ada Generation provides 960 GB s⁻¹~~. We measured the sustained bandwidth with a memory benchmark based on the vector update $a = a + s \cdot b$, following the approach of Deakin et al. (2018). The benchmark sustained 824.0 GB s⁻¹ on a single GPU (86% of the peak) and 113.2 GB s⁻¹ on the 32-core CPU (74% of the peak) (Table S1).

Lines 342–349; Section 4.2,

... The single GPU reached a throughput of 4.2×10^6 DOFs/s, compared with 5.3×10^5 DOFs/s on the 32-core CPU. ~~The two systems differ in theoretical memory bandwidth by $4.7 \times (204.8 \text{ GB s}^{-1}$ for the Xeon Gold 6338 and 960 GB s^{-1} for the NVIDIA RTX 6000 Ada Generation). However, the speedups measured across the Stokes benchmarks span $5-11 \times$, ranging from approximately the bandwidth ratio to higher values.~~ In the memory bandwidth benchmark (Table S1), the GPU sustains 824.0 GB s⁻¹ and the 32-core CPU sustains 113.2 GB s⁻¹, a ratio of $7.3 \times$. The sustained ratio exceeds the theoretical peak ratio of $6.25 \times$ (960 and 153.6 GB s⁻¹). The speedups measured across the Stokes benchmarks span $6-11 \times$ and cluster around the sustained ratio. The cost of PCG and GMRES is dominated by sparse matrix-vector products, data-parallel operations that map onto the GPU, where high memory bandwidth sustains the streaming access pattern. ~~The ratio of peak memory bandwidth between the GPU and CPU remains a poor predictor of the achievable speedup. The GPU and CPU deliver unequal fractions of peak bandwidth. The effective bandwidth ratio during the computation exceeds the nominal peak ratio.~~ The unequal fractions of peak bandwidth attained by the GPU and CPU (86% and 74%, respectively) widen the sustained ratio beyond the peak ratio. The CPU and GPU versions of BoomerAMG further adopt different coarsening strategies that shift the number of iterations. The measured speedup depends on the cost per iteration and on the number of iterations. ~~Therefore, the peak bandwidth ratio~~ The sustained ratio serves as an approximate reference rather than an upper bound.

2. General proofreading may be advised given that some portions of the manuscript contain not very accurate information or do not read very well. Specifically, 1.84-92 in the revised manuscript are not very accurate. GPU and CPU core comparison cannot be done side by side, and noonday Nvidia GPUs have CUDA core. A paragraph briefly discussion discrepancies between CPU and GPU hardware design is welcome, but should be improved.

Authors' reply:

We thank the editor for pointing out these inaccuracies. We removed the direct comparison of CPU and CUDA core counts, as well as the bandwidth comparison with the Xeon Platinum 8568Y+, which was not used in this study. The revised paragraph now explains the architectural differences between CPUs, which prioritise low latency, and GPUs, which are designed for high throughput. We made the corresponding change to the first two sentences of the Abstract, corrected the description of the accelerated pseudo-transient method in the Introduction, and updated the hardware descriptions in Sections 3.2 and 5.2 for consistency. We also proofread the full manuscript. The relevant revisions are shown below.

Lines 84–91; Introduction,

~~In CPU architecture, individual cores offer high single-thread performance. However, in high-resolution simulations involving repeated access to large sparse matrices and associated vectors, the limited memory bandwidth frequently functions as the primary bottleneck, leading to significant performance degradation (May et al., 2015). In contrast, a graphics processing unit (GPU) provides~~

~~access to thousands of CUDA cores (e.g., NVIDIA RTX 6000 Ada: 18,176 CUDA cores) concurrently, enabling massively parallel execution of operations such as matrix-vector products, an architectural advantage that clearly distinguishes GPUs from conventional CPUs. Moreover, GPUs sustain memory bandwidth on the order of several hundred gigabytes per second, surpassing that of similarly priced CPUs (RTX 6000 Ada: 960 GB/s; Intel Xeon Platinum 8568Y+: 358 GB/s). Building on these hardware advantages, NVIDIA's Single Instruction Multiple Threads architecture provides efficient thread-level scheduling, further enhancing the performance of GPU-accelerated numerical modelling (Ta et al., 2015).~~ A GPU integrates thousands of arithmetic units designed for massively parallel computation (Ta et al., 2015). A CPU uses a few complex cores designed for fast sequential computation. The GPU also accesses memory at a higher bandwidth than the CPU, with theoretical peaks of 960 and 153.6 GB s⁻¹ for the NVIDIA RTX 6000 Ada Generation and the Xeon Gold 6338 workstation used in this study, respectively. Sparse matrix-vector products and vector updates dominate the cost of Krylov solvers (May et al., 2015). Both operations perform approximately one multiplication and one addition per value read from memory (Williams et al., 2009). The low ratio of arithmetic to memory traffic, termed arithmetic intensity, makes memory bandwidth, rather than the number of arithmetic units or cores, a strong control on the cost of the two operations. Preconditioning and other solver steps also contribute to the time per Krylov iteration.

Lines 92–102; Introduction,

~~To date, in geodynamic modelling, GPU-based solutions of the Stokes equations have primarily focused on pseudo-transient (PT) approaches, in which a pseudo-inertial term is added to enable explicit time stepping (Räss et al., 2022; Wang et al., 2022; Alkhimenkov and Podladchikov, 2025). PT methods reduce memory requirements by avoiding full implicit solves, making them attractive for GPU implementations. Subsequent developments led to the accelerated PT method, which allows flexible time steps and preconditioning strategies. By exploiting the mass matrix to treat the stiffness matrix explicitly, the PT method achieves a substantial reduction in memory usage. However, explicit pseudo-time stepping can require a large number of iterations to converge, particularly for problems with strong viscosity contrasts (Räss et al., 2022).~~ To date, GPU-based solutions of the Stokes equations in geodynamic modelling have often used pseudo-transient (PT) methods (Räss et al., 2022), which have also been extended to fluid flow and to elastic, viscoelastic, and hydromechanical problems (Wang et al., 2022; Alkhimenkov and Podladchikov, 2025). PT methods augment the governing equations with pseudo-time derivatives and integrate them explicitly until the residuals are reduced below a prescribed tolerance. In the cited Stokes implementations, the explicit update applies local finite-difference stencils on regular staggered grids, requiring no assembly of global matrices. The matrix-free formulation keeps the memory footprint small, which suits GPUs with limited device memory. The accelerated PT method adds pseudo-time terms that give rise to damped-wave dynamics (Räss et al., 2022). With near-optimal iteration parameters, the iteration count grows approximately linearly, rather than quadratically, with the number of grid cells along one dimension. Räss et al. (2022) demonstrated convergence for viscosity contrasts of up to nine orders of magnitude. Convergence slowed in some of the high-contrast cases. In contrast, GPU implementations of Krylov solvers with algebraic multigrid preconditioning, the standard approach in finite-element geodynamic codes, have received less attention.

Abstract, first two sentences,

~~A GPU combines thousands of cores with high memory bandwidth.~~ GPUs provide much higher memory bandwidth than CPUs. Data-parallel computations with regular memory access patterns ~~attain the performance of this architecture~~ exploit the GPU architecture efficiently.

Lines 253–255; Section 3.2,

We optimised the BoomerAMG preconditioner settings by considering architectural differences between a 32-core CPU and a GPU with thousands of ~~parallel processing cores~~ arithmetic units.

Lines 619–624; Section 5.2,

~~Krylov subspace solvers for the Stokes system benefit from high memory bandwidth, large device memory, and FP64 performance. The RTX 6000 Ada delivers an FP64 peak near 1.4 TFLOPS, while HPC-class GPUs such as the H200 provide much higher FP64 performance and larger memory capacity. For example, the H200 delivers 34 TFLOPS, roughly 24 times higher than the RTX 6000 Ada. Using H200-class systems could further improve Stokes solver performance at high resolution. Such scaling enables higher resolution simulations of global mantle convection and complex crustal deformation, demonstrating how GPU system scaling shapes next-generation geodynamics.~~ Krylov solvers for the Stokes system benefit mainly from high memory bandwidth and large device memory. HPC-class GPUs such as the NVIDIA H200 provide 4.8 TB s⁻¹ and 141 GB for the two quantities, approximately 5× and 2.9× the values of the RTX 6000 Ada, respectively. The higher bandwidth of the H200 is expected to lower the cost of operations limited by memory bandwidth. The overall gain for the Stokes solver requires direct measurement on the H200.

While working on this minor revision items, please also address the points raised during file validation and the reference validation ones.

Authors' reply:

We thank the editor for this reminder. We addressed the file validation and reference validation reports as follows.

File validation

Notification to the authors: Your "Short summary" text in the database (MS records) contains several abbreviations. Please provide at least one written-out version of each abbreviation to make it better understandable for non-experts. Please remember that there is a character-limitation for the short summary text of max. 500 characters (including spaces).

Authors' reply:

We thank the technical editor for this notification. We rewrote the short summary in plain words, replacing all abbreviations (e.g., GPU and CPU) with written-out terms within the 500-character limit.

Short Summary

~~Geodynamic simulations of mantle flow and plate tectonics repeatedly solve the Stokes equations, requiring substantial memory and computational time. We developed a GPU-accelerated solver by coupling FEniCS with PETSc while keeping matrices and vectors resident in GPU memory during iterative solves. A single GPU achieved 5–11× shorter execution times than a 32-core CPU across the tested benchmarks, while multiple GPUs increased throughput and enabled larger three-dimensional simulations.~~ Geodynamic simulations of mantle flow repeatedly solve the Stokes equations at high memory and computational cost. We developed a Stokes solver for graphics processing units by coupling two open-source libraries. A single graphics processing unit solved the Stokes equations 6–11 times faster than a 32-core central processor, with the same

accuracy. Two units further shortened the solution time of large problems. The doubled memory extended the maximum size of three-dimensional simulations.

Reference validation

This report contains any reference with DOI or URL where our reference verification found mismatches with the allocated original sources. Please review the references and correct them in your manuscript before sending it back to us.

Reference 93 [No DOI/URL]: Uzawa, H.: Iterative methods for concave programming, in: Studies in Linear and Non-linear Programming, edited by: Arrow, K. J., Hurwicz, L., and Uzawa, H., Stanford University Press, Stanford, CA, 154–165, 1958.

Authors' reply:

We thank the technical editor for this notification. The chapter of Uzawa (1958) lacks a DOI and a URL. We replaced the chapter with its reprint in Uzawa (1989), which provides a DOI. We also checked the entire reference list of the revised manuscript. The list contains 108 references, consisting of 98 journal articles, 4 conference papers, 3 book chapters, 1 technical report, 1 preprint, and 1 software archive. All 108 references include a DOI, and all DOIs resolve to the cited works. All references in the list are cited in the main text, and all citations in the main text appear in the list.

Line 60; Introduction,

... which can be solved iteratively (Uzawa, ~~1958~~1989; Moresi et al., 1996; Leng and Zhong, 2008).

Reference list,

~~Uzawa, H.: Iterative methods for concave programming, in: Studies in Linear and Non-linear Programming, edited by: Arrow, K. J., Hurwicz, L., and Uzawa, H., Stanford University Press, Stanford, CA, 154–165, 1958.~~

Uzawa, H.: Iterative methods for concave programming, in: Preference, Production and Capital: Selected Papers of Hirofumi Uzawa, Cambridge University Press, Cambridge, 135–148, <https://doi.org/10.1017/CBO9780511664496.011>, 1989.

We sincerely thank the editor for the careful handling of our manuscript and for the constructive comments. The request for a measured memory bandwidth and the remark on the CPU and GPU core comparison made the revision more accurate. We also thank the technical editor for the file and reference validation reports.

With best regards,

Kyeong-Min Lee and Byung-Dal So

References of this response letter

- Alkhimenkov, Y. and Podladchikov, Y. Y.: Accelerated pseudo-transient method for elastic, viscoelastic, and coupled hydromechanical problems with applications, *Geoscientific Model Development*, 18, 563–583, <https://doi.org/10.5194/gmd-18-563-2025>, 2025.
- Deakin, T., Price, J., Martineau, M., and McIntosh-Smith, S.: Evaluating attainable memory bandwidth of parallel programming models via BabelStream, *International Journal of Computational Science and Engineering*, 17, 247–262, <https://doi.org/10.1504/IJCSE.2018.095847>, 2018.
- Leng, W. and Zhong, S.: Viscous heating, adiabatic heating and energetic consistency in compressible mantle convection, *Geophysical Journal International*, 173, 693–702, <https://doi.org/10.1111/j.1365-246X.2008.03745.x>, 2008.
- May, D. A., Brown, J., and Le Pourhiet, L.: A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow, *Computer Methods in Applied Mechanics and Engineering*, 290, 496–523, <https://doi.org/10.1016/j.cma.2015.03.014>, 2015.
- Moresi, L., Zhong, S., and Gurnis, M.: The accuracy of finite element solutions of Stokes’s flow with strongly varying viscosity, *Physics of the Earth and Planetary Interiors*, 97, 83–94, [https://doi.org/10.1016/0031-9201\(96\)03163-9](https://doi.org/10.1016/0031-9201(96)03163-9), 1996.
- Räss, L., Utkin, I., Duretz, T., Omlin, S., and Podladchikov, Y. Y.: Assessing the robustness and scalability of the accelerated pseudo-transient method, *Geoscientific Model Development*, 15, 5757–5786, <https://doi.org/10.5194/gmd-15-5757-2022>, 2022.
- Ta, T., Choo, K., Tan, E., Jang, B., and Choi, E.: Accelerating DynEarthSol3D on tightly coupled CPU-GPU heterogeneous processors, *Computers & Geosciences*, 79, 27–37, <https://doi.org/10.1016/j.cageo.2015.03.003>, 2015.
- Uzawa, H.: Iterative methods for concave programming, in: *Studies in Linear and Non-linear Programming*, edited by: Arrow, K. J., Hurwicz, L., and Uzawa, H., Stanford University Press, Stanford, CA, 154–165, 1958.
- Uzawa, H.: Iterative methods for concave programming, in: *Preference, Production and Capital: Selected Papers of Hirofumi Uzawa*, Cambridge University Press, Cambridge, 135–148, <https://doi.org/10.1017/CBO9780511664496.011>, 1989.
- Wang, L. H., Yarushina, V. M., Alkhimenkov, Y., and Podladchikov, Y.: Physics-inspired pseudo-transient method and its application in modelling focused fluid flow with geological complexity, *Geophysical Journal International*, 229, 1–20, <https://doi.org/10.1093/gji/ggab426>, 2022.
- Williams, S., Waterman, A., and Patterson, D.: Roofline: an insightful visual performance model for multicore architectures, *Communications of the ACM*, 52, 65–76, <https://doi.org/10.1145/1498765.1498785>, 2009.