

Which architecture is better suited for matrix-free finite-element algorithms: Intel Skylake or NVIDIA Volta?

Martin Kronbichler
Technical University of Munich
Boltzmannstr. 15
Garching, Germany 85748
kronbichler@lhm.mw.tum.de

Martin Ohlerich
Leibniz Computing Centre
Boltzmannstr. 1
Garching, Germany

Momme Allalen
Leibniz Computing Centre
Boltzmannstr. 1
Garching, Germany

Wolfgang A. Wall
Technical University of Munich
Boltzmannstr. 15
Garching, Germany

ABSTRACT

This work presents a performance comparison of highly tuned matrix-free finite element kernels from the deal.II finite element library on different contemporary computer architectures, NVIDIA V100 and P100 GPUs, an Intel Knights Landing Xeon Phi, and two multi-core Intel CPUs (Broadwell and Skylake). The algorithms are based on fast integration on hexahedra using sum factorization techniques. For small problem sizes when all data fits into CPU caches, Skylake is very competitive with Volta. For larger sizes, however, the GPU holds an advantage of approximately a factor of three over Skylake, because all architectures operate in the memory-bandwidth limited regime. A detailed performance analysis contrasts the throughput-oriented character of GPUs versus the more latency-optimized CPUs for the scenario of high-order finite element computations. Furthermore, the algorithms are ported to a multi-GPU case on NVIDIA DGX-1, where we find good scaling at large sizes, but a severe increase in the minimum time for a matrix-vector product from less than 10^{-4} seconds on a single GPU to almost 10^{-3} seconds on 8 GPUs.

CCS CONCEPTS

•Mathematics of computing →Mathematical software performance;

ACM Reference format:

Martin Kronbichler, Momme Allalen, Martin Ohlerich, and Wolfgang A. Wall. 2018. Which architecture is better suited for matrix-free finite-element algorithms: Intel Skylake or NVIDIA Volta?. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, Texas USA, November 2018 (SC’18)*, 3 pages. DOI:

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SC’18, Dallas, Texas USA

© 2018 Copyright held by the owner/author(s).
DOI:

1 INTRODUCTION

Matrix-free evaluation of weak forms in finite element methods, a technique originally established in the spectral element community, is the most competitive implementation variant for higher-order basis functions in a wide range of applications involving partial differential equations such as structural mechanics, biomechanics and fluid dynamics. Computing the integrals on the fly trades a lower transfer of data from main memory for some additional arithmetic operations as compared to conventional solvers using sparse matrices. Since sparse matrix algebra is heavily memory bandwidth bound with arithmetic intensities of 0.16 to 0.25 floating point operations per byte loaded from main memory, a matrix-free alternative can be much faster also when performing more computations [7, 9]. A competitive evaluation of integrals is provided by sum factorization, a technique that decomposes the interpolation and unit cell derivatives for tensor product shape functions and quadrature into a series of 1D operations. This approach achieves arithmetic intensities between 1 and 10 floating point operations per byte loaded from main memory [9].

Our poster considers algorithmic components that provide a high node-level performance, supported by detailed performance figures over five different processor architectures. The performance of the general-purpose kernels included in the open-source finite element library deal.II [1] is critically assessed in terms of the throughput at large sizes and the latency when there is less work per core. The analysis highlights the increasing importance of memory bandwidth also for algorithms with a higher FLOP/Byte ratio than traditional sparse matrix algebra. As long as all data fits into caches, CPUs are surprisingly competitive though. Overall, the NVIDIA V100 and P100 GPU have an edge over the competing architectures in terms of aggregate performance.

2 MATRIX-FREE ALGORITHM

In both explicit time integration schemes and a selection of iterative solvers, the matrix-vector product is the primary determinant for application performance. This includes state-of-the-art multigrid solvers based on Jacobi and polynomial smoothers [9, 11].

Our matrix-free implementation considers a partial differential equation in weak form as the evaluation of the differential operator for a given input vector u in terms of the underlying integrals. In

the case of the Laplacian, we express the matrix-vector product $v = Au$ as

$$v_i = (Au)_i = \int_{\Omega} \nabla \varphi_i \cdot \nabla u^h dx,$$

where the assembled system matrix A is substituted by the underlying integrals with φ_i the i -th test function and u^h the piecewise polynomial representation associated with the input vector. In sum factorization, the matrix A is neither built globally nor on the element level, and only the integrals on the right hand side are evaluated by numerical quadrature in a loop over all cells in the mesh. The arithmetic work is spent in the interpolation of the unit cell derivatives $\hat{\nabla} u^h$ given the vector entries u_j on the cell and the multiplication and summation by $\hat{\nabla} \varphi_i$, see [7] and references therein. Optimal-complexity schemes are enabled by sum factorization, which utilizes the tensor product structure in the shape functions and the quadrature formula by factoring out common factors along the respective directions and can thus apply a series of one-dimensional interpolation kernels along each coordinate direction. This concept, dating back to [12], reduces the operational complexity from $O(p^d)$ operations per degree of freedom for p -th degree polynomials in d dimensions for the final matrix stencil to $O(dp)$ for all spatial dimensions. At degrees $p > 3$ in 3D, sum factorization does not only reduce the memory transfer, but also the number of arithmetic operations over matrices [7, 9].

In this work, we evaluate the implementation of sum factorization within the deal.II finite element library [1] which has a particularly well-tuned CPU version that runs close to hardware limits [7–9]. The CPU version uses SIMD vectorization by C++ wrapper classes around intrinsics over several cells and gather/scatter operations for the vector access which are specifically tuned for AVX-512. Despite the high level of optimization, the kernels can be accessed in a generic way that can also be leveraged in complex application codes [2, 5, 6]. The CPU code is parallelized with pure MPI, which we found this to outperform thread-based implementations that schedule the integration tasks according to dependencies as a means to avoid race conditions [4, 8].

For the GPUs, the algorithms from [10, 11] are extended to multiple GPUs with data exchange similar to the MPI implementation for the CPU [7]. The GPU implementation uses a more fine-grained parallelization with a separate thread for each degree of freedom on a cell that is combined with a parallel for loop over elements. Race conditions of access to the same degree of freedom from several threads are avoided by NVIDIA's built-in atomic instructions.

3 PERFORMANCE COMPARISON

Fig. 1 displays the performance of the evaluation of the scalar Laplacian for polynomial degree $p = 4$ on a setup similar to the CEED bakeoff problem BK5 [3]. The GPUs reach a clearly higher throughput than the CPUs for large problem sizes. However, the CPUs have a better throughput at small sizes where the latency becomes important.

Fig. 2 displays a weak scaling experiment with 1 and 10 million degrees of freedom per GPU over multiple GPUs on both the Pascal and the Volta architecture. We see that the latency limit of the single-GPU case gets exacerbated for multiple GPUs and throughput reduces for the smaller problem size. This decrease in efficiency

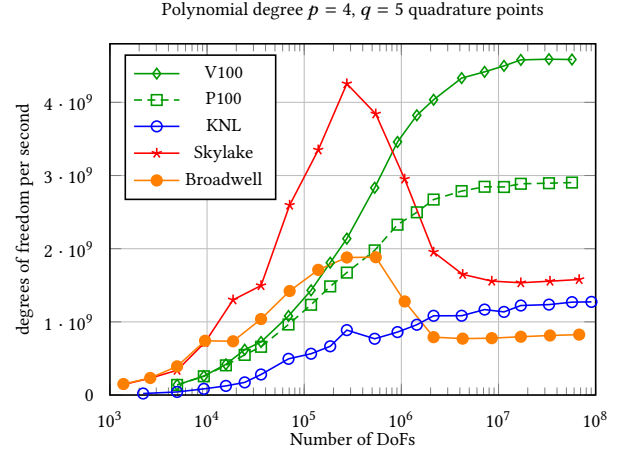


Figure 1: Throughput of Laplace operator evaluation on a curved mesh.

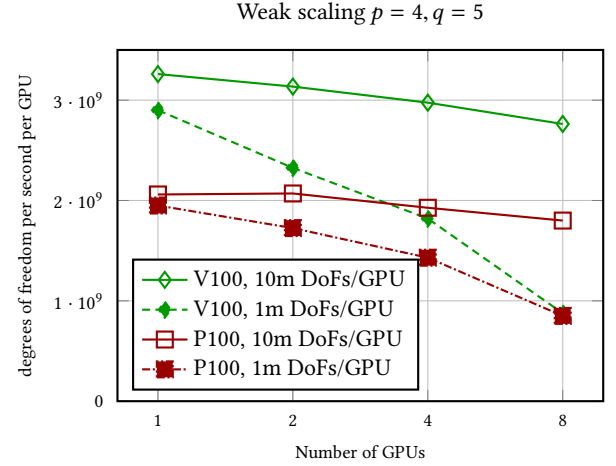


Figure 2: Weak scaling of multi-GPU setup.

can be explained by the fact that the 8-GPU setup does not go below $8 \cdot 10^{-4}$ seconds per operator evaluation, as compared to $5 \cdot 10^{-5}$ seconds on the single-GPU setup.

4 CONCLUSIONS AND OUTLOOK

Our experiments show the great capabilities of the V100 and P100 GPUs for throughput-dominated setups. While we observe Intel CPUs to reach a similar performance on small and moderate sizes that fit into the processors' caches, the V100 is able to considerably outperform the former in case memory bandwidth is important, which is a large share of matrix-free applications. Our results also show that further research is needed to improve the throughput at moderate problem sizes per GPU which does currently not optimally use the available resources. Future work will investigate better schemes for overlapping data transfer and computation, similar to the CPU implementation [7].

REFERENCES

- [1] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoeller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. 2018. The deal.II Library, Version 9.0. *J. Numer. Math.* (2018). <https://doi.org/10.1515/jnma-2018-0054>
- [2] N. Fehn, W. A. Wall, and M. Kronbichler. 2018. Efficiency of high-performance discontinuous Galerkin spectral element methods for under-resolved turbulent incompressible flows. *Int. J. Numer. Meth. Fluids* 88, 1 (2018), 32–54. doi:10.1002/fld.4511.
- [3] P. Fischer, Tz. Kolev, M. Min, and V. Dobrev. 2018. CEED bake-off problems for matrix-free operator evaluation. (2018). <http://ceed.exascaleproject.org/bps/>.
- [4] K. Kormann and M. Kronbichler. 2011. Parallel Finite Element Operator Application: Graph Partitioning and Coloring. In *Proceedings of the 7th IEEE International Conference on eScience*. 332–339.
- [5] B. Krank, N. Fehn, W. A. Wall, and M. Kronbichler. 2017. A high-order semi-explicit discontinuous Galerkin solver for 3D incompressible flow with application to DNS and LES of turbulent channel flow. *J. Comput. Phys.* in press (2017). <https://doi.org/10.1016/j.jcp.2017.07.039>
- [6] M. Kronbichler, A. Diagne, and H. Holmgren. 2018. A fast massively parallel two-phase flow solver for the simulation of microfluidic chips. *Int. J. High Perf. Comput. Appl.* 32, 2 (2018), 266–287. <https://doi.org/10.1177/1094342016671790>
- [7] M. Kronbichler and K. Kormann. 2012. A generic interface for parallel cell-based finite element operator application. *Comput. Fluids* 63 (2012), 135–147. <https://doi.org/10.1016/j.compfluid.2012.04.012>
- [8] M. Kronbichler, K. Kormann, I. Pasichnyk, and M. Allalen. 2017. Fast Matrix-Free Discontinuous Galerkin Kernels on Modern Computer Architectures. In *ISC High Performance 2017, LNCS 10266*, J. M. Kunkel, R. Yokota, and D. Balaji, P. Keyes (Eds.). Springer, Cham, 237–255. <https://doi.org/10.1007/978-3-319-58667-013>
- [9] M. Kronbichler and W. A. Wall. 2018. A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers. *SIAM J. Sci. Comput.* in press (2018). <https://doi.org/10.1137/16M110455X>
- [10] K. Ljungkvist. 2017. Matrix-Free Finite-Element Computations on Graphics Processors with Adaptively Refined Unstructured Meshes. In *HPC '17: Proceedings of the 25th High Performance Computing Symposium*. Society for Computer Simulation International, San Diego, CA, USA.
- [11] K. Ljungkvist and M. Kronbichler. 2017. *Multigrid for Matrix-Free Finite Element Computations on Graphics Processors*. Technical Report 2017-006. Department of Information Technology, Uppsala University.
- [12] S. A. Orszag. 1980. Spectral Methods for Problems in Complex Geometries. *J. Comput. Phys.* 37 (1980), 70–92.