The arithmetic intensity of high-order discontinuous Galerkin methods for hyperbolic systems

Suyash Tandon and Eric Johnsen
Department of Mechanical Engineering, University of Michigan, Ann Arbor, Michigan, U.S.A
suyashtn@umich.edu

***Abstract*—High-fidelity numerical simulations of complex flow problems require high-resolution capabilities, which can be achieved by employing high-order methods. A class of recovery-assisted discontinuous Galerkin (RADG) methods can achieve high-orders of accuracy by strategically combining degrees of freedom from neighboring cells; the order of accuracy can be increased by increasing the polynomial degree p of the solution representation. However, an increase in p increases the number of degrees of freedom, thereby significantly increasing the memory footprint due to I/O operations and floating-point operations. In this study, the arithmetic intensity, which is the amount of work done per data transferred, of a class of RADG methods for hyperbolic systems of conservation laws is theoretically analyzed for $p = 1$ through 6. Different data cache models are considered, and numerical experiments demonstrate that RADG methods have high arithmetic intensity, thus more effectively utilizing on-node floating-point capabilities on modern high-performance computing (HPC) platforms.**

***Index Terms*—high-order methods, arithmetic intensity, roofline analysis, high-performance computing**

I. Introduction

The next generation computational models will enable advances in the fields of climate science, ocean flow behavior, space sciences, complex materials and other areas of scientific research [1]. To meet the growing needs in scientific discovery, the computational capability of the fastest supercomputers must continue to grow. Driven by limitations in power consumption, memory bandwidth and latency [2], and high demand in accuracy, the next generation HPC sites for exascale computing are expected to feature heterogeneous architectures [3]. On these new systems, the decreasing power cost of floating-point operations (FLOP) has exposed the power cost of data motion. Thus, FLOP–if concurrency can be harnessed–are no longer the primary (on-node) cost factor for numerical simulation. A new trade-off must be made between data motion, memory usage, and operations [4], [5].

High-order methods such as the discontinuous Galerkin method (DG) [6] have the potential to address the above issues. High orders of accuracy can be achieved by adding degrees of freedom to each element to represent the solution as a high-order polynomial of degree p. Since the solution is allowed to be discontinuous across elements, borrowing from the finite volume method, the flux between immediately adjacent elements is used to preserve stability. The method maintains a compact stencil [7] and exhibits good parallel scaling on modern HPC platforms [8]–[10]. The recovery-assisted DG method (RADG [11]) uses the information from a pair of neighboring elements to recover a high-order DG approximation of the flux at the interface of the two neighboring elements. The recovery procedure results in order of accuracy $2p+2$, which is higher than conventional DG methods. While the DG methods present many advantages, a large number of degrees of freedom inside each element, particularly for RADG methods, increases the on-node data motion, memory usage, and FLOP.

This work aims to determine the trade-off between FLOP and bytes transferred for RADG discretizations of hyperbolic systems of conservation laws that are used to describe the complex flow problems. Theoretical bounds for solution polynomial orders one through six, for both the FLOP count and the arithmetic intensity, are evaluated. These two quantities are used to determine the on-node performance based on the roofline model [12].

II. RADG Discretization of Hyperbolic systems

Consider the hyperbolic system of conservation laws

$$\frac{\partial u}{\partial t} + \nabla.\mathcal{F}(u) = 0, \tag{1}$$

where $u(\mathbf{x}, t)$ is a vector of conserved variables, and $\mathcal{F}(u)$ are the vector-valued flux functions. In RADG discretization, the computational domain $\mathbf{\Omega}$ is partitioned into M non-overlapping elements Ω_m, such that $\cap_{m=1}^{M}\Omega_m = \emptyset$ and $\cup_{m=1}^{M}\Omega_m = \mathbf{\Omega}$, and the elemental boundary is denoted by $\partial\Omega_m$. Each element's set of basis functions ϕ_m is a polynomial basis of at most degree p in a given direction and contains $K = (p+1)^D$ members. The number of quadrature points in the interior of each element is $Q_v = K$, and that on each interface of the element is $Q_s = (p+1)^{D-1}$.

The governing equation 1 is satisfied in the weak sense in each element by expanding the numerical solution U^h within a K-dimensional solution space ϕ_m and integrating as,

$$\int_{\Omega_m} \phi_m^k \frac{\partial}{\partial t} U_m^h d\mathbf{x} = \int_{\Omega_m} (\nabla\phi_m^k).\mathcal{F}(U_m^h) d\mathbf{x} - \int_{\partial\Omega_m} \phi_m^k (\widetilde{\mathcal{F}}.\mathbf{n}^-) ds. \tag{2}$$

Here, $k \in \{0, 1, 2, \ldots, K-1\}$, $\widetilde{\mathcal{F}}$ is the common flux value at the element-element interface and $\mathbf{n}^-$ is the element's outward normal.

This work was supported in part by the Michigan Institute for Computational Discovery and Engineering. Additionally, this research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

TABLE I: Floating-point operations (FLOP) and data transfers per step for RADG discretization.

#	Calculation	FLOP	data transfers[§]	
			no-cache data transfer	finite-cache data transfer
1	U_L or U_R, at interface	$2(Q_s \times 4K) \times N_v^{\dagger}$	$2(\mathcal{R} + 2Q_v + Q_s) \times N_v$	$Q_v \times (T^D + (T+2W)^D) + \mathcal{R} \times 2D(T+2W)^{D-1}$
2	$\tilde{\mathcal{F}}(U_L, U_R)$, at interface	$(f_R \times Q_s) \times N_v$	$(3Q_s) \times N_v$	-
3	U^h at interior	$(Q_v \times 2K) \times N^D$	$(2Q_v) \times N^D$	-
4	$\mathcal{F}(U^h)$ at interior	$(f_F \times D \times Q_v) \times N^D$	$(2D+1) \times Q_v \times N^D$	$Q_v \times T^D$

$^{\dagger}N_v = D(N+2W+1)(N+2W)^{D-1}$ is the total number of interfaces in the block structured grid of size $(N+2)^D$.

§ each transfer is 8 Bytes for double-precision floating-point value.

* f_R and f_F are cost of Riemann solver and flux function.

Consider the union of two adjacent elements, $\mathcal{U} = \Omega_A \cup \Omega_B$ with same polynomial order p. A recovery basis $\mathbf{\Psi}$, supported over $\mathcal{U}$, is defined with the recovery coordinate $\mathbf{r}$. The recovered solution over the union is defined as a $2K-$dimensional polynomial expansion in the recovery basis:

$$f(\mathbf{r}) = \sum_{n=1}^{2K} \Psi^n(\mathbf{r}) \hat{f}^n \tag{3}$$

The recovery operation on each interface can be cast in a matrix-vector form [7] for a given variable. Equation 2 can be conveniently re-written, where the term on the left-hand side is expanded to give a $K \times K$ mass matrix M_m of the element Ω_m that depends only on the element's geometry and set of basis functions ϕ_m. Additionally, the spatial residual terms on the right for a given element can be collected in a single residual vector R_m. The mass matrix can be inverted to directly calculate the temporal derivatives of $\hat{\mathbf{U}}_m$,

$$\frac{d}{dt}\hat{U}_m = M_m^{-1} R_m(U^h). \tag{4}$$

The evaluation of the residual term at every time step is split into various operations listed in table I.

III. Arithmetic Intensity

A box-structured grid with N uniform elements in each direction is employed for the present analysis. At each step, box-grid is surrounded by a layer of ghost cells containing the neighboring data upon which the box is dependent, allowing computation in boxes to be done independently. This gives the total number of elements in each direction as $N+2$, and the total number of interfaces between adjacent cells is given as N_v. For the purpose of calculating arithmetic intensity, which is the measure of the the amount of work done per data transferred, an abstract machine model is considered that is composed of a processor and two levels of memory: an unlimited amount of slow memory and a limited amount of fast memory. At the start of processing, data are stored in the slow memory and the fast memory is empty. Computation is only performed by the processor on data residing in the fast memory, so any data to be operated on must first be transferred from slow memory. After the computation, the result are returned to the fast memory and then transferred to the slow memory.

To derive the bounds on the arithmetic intensity for RADG discretization applied to hyperbolic systems, the FLOP count for each step of the residual evaluation and the data transfer necessary for the cache strategies considered are evaluated as listed in table I. The results are summarized as,

- *no-cache:* A degenerate case that is analogous to poor cache utilization [13] and provides an upper bound on the number of data transfers that must be performed, and a lower bound on the arithmetic intensity. In $3D$ with $N \leq 128$, the arithmetic intensity for RADG is < 0.4. Current machines have a flops-to-byte ratio of five or greater [12]. As expected, without a cache, a significant arithmetic intensity is not achievable, see fig. 1.
- *finite-cache:* The box of size $(N+2)^D$ is divided into cubic tiles of length $T = N/m$ for some $m > 0$. Cubic tiling improves the balance of operations and results in high arithmetic intensity (> 10) at large T, which is approximately the machine balance for state-of-the-art machines like OLCF's Summit [14]. Fig. 2 shows the arithmetic intensity for the finite-cache case in $3D$. The constraint on the size of the tile limits the maximum achievable arithmetic intensity. Small T results in data transfer overheads with more number of data transfers required. The finite-cache utilization is further improved by exploring techniques to reduce the storage of large recovery matrices and using a vertical rectangular tiling strategy [15].

IV. conclusions

On-node performance of an application needs to be emphasized along with parallel scalability for optimal algorithm design of scientific applications. With data management contributing more towards the cost of numerical simulations, the arithmetic intensity, which measures the amount of work done per byte of data transferred is an important design criterion. The on-node performance of a class of recovery-assisted discontinuous Galerkin (RADG) schemes, used for spatial discretization of conservation laws of fluid dynamics, is investigated. Estimates for different cache models are developed, including the lower bound case of no-cache and a finite-sized cache. Theory and numerical experiments predict that RADG schemes can achieve high arithmetic intensity (> 10), which is necessary to better utilize on-node floating-point capability of modern high-performance computing (HPC) platforms.

References

[1] D. Brown, P. Messina, D. Keyes, J. Morrison, R. Lucas, J. Shalf, P. Beckman, R. Brightwell, A. Geist, J. Vetter *et al.*, "Scientific grand

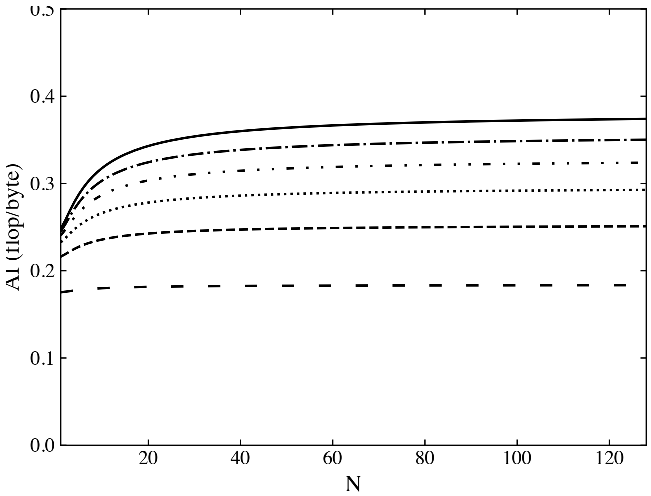

Fig. 1: Arithmetic intensity for no-cache model with N with varying polynomial order:— —, $p = 1$; - - - -, $p = 2$; ⋯⋯, $p = 3$; - · · · -, $p = 4$; -·-·-, $p = 5$; ——, $p = 6$.

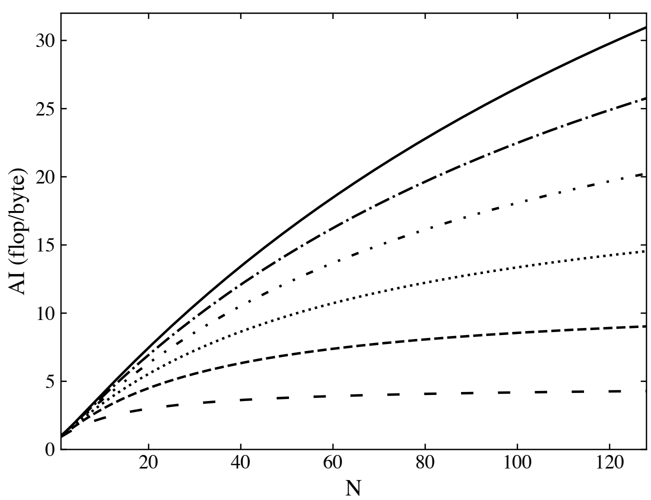

Fig. 2: Arithmetic intensity for finite-cache model with N with varying polynomial order:— —, $p = 1$; - - - -, $p = 2$; ⋯⋯, $p = 3$; - · · · -, $p = 4$; -·-·-, $p = 5$; ——, $p = 6$.

challenges: Crosscutting technologies for computing at the exascale," U.S. Department Of Energy, Pacific Northwest National Laboratory, Tech. Rep., 2010.

[2] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of Ion-implanted Mosfet's With Very Small Physical Dimensions," *IEEE J. Solid-State Cir.*, vol. 9, no. 5, pp. 256–268, 1974.

[3] J. Dongarra, J. Hittinger, J. Bell, L. Chacon, R. Falgout, M. Heroux, and others., "Applied Mathematics Research for Exascale Computing," Tech. Rep., 2014.

[4] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford *et al.*, "The Opportunities and Challenges of Exascale Computing," U.S. Department Of Energy, Office of Science, Advanced Scientific Computing Advisory Committee, Tech. Rep., 2010.

[5] R. Lucas, J. Ang, K. Bergman, S. Borkar, W. Carlson, L. Carrington *et al.*, "Top Ten Exascale Research Challenges," U.S. Department Of Energy, Office of Science, Advanced Scientific Computing Advisory Committee, Tech. Rep., 2014.

[6] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, "The Development of Discontinuous Galerkin Methods," *Discontinuous Galerkin Methods*, pp. 77–88, 2000.

[7] P. E. Johnson and E. Johnsen, "The Compact Gradient Recovery Discontinuous Galerkin Method for Diffusion Problems," *J. Comp. Phys.*, vol. 398, p. 108872, 2019.

[8] K. S. Bey, A. Patra, and J. T. Oden, "Hp-version Discontinuous Galerkin Methods for Hyperbolic Conservation Laws: a Parallel Adaptive Strategy," *Int. J. Num. Meth. Engg.*, vol. 38, no. 22, pp. 3889–3908, 1995.

[9] A. Heinecke, A. Breuer, S. Rettenberger, M. Bader, A.-A. Gabriel, C. Pelties *et al.*, "Petascale High Order Dynamic Rupture Earthquake Simulations on Heterogeneous Supercomputers," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2014, pp. 3–14.

[10] T. Houba, A. Dasgupta, S. Gopalakrishnan, R. Gosse, and S. Roy, "Supersonic Turbulent Flow Simulation Using a Scalable Parallel Modal Discontinuous Galerkin Numerical Method," *Scientific reports*, vol. 9, no. 1, pp. 1–19, 2019.

[11] P. E. Johnson, L. H. Khieu, and E. Johnsen, "Analysis of Recovery-Assisted Discontinuous Galerkin Methods for the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, p. 109813, 2020.

[12] S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[13] C. Olschanowsky, M. M. Strout, S. Guzik, J. Loffeld, and J. Hittinger, "A Study on Balancing Parallelism, Data Locality, and Recomputation in Existing PDE Solvers," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2014, pp. 793–804.

[14] A. YarKhan, J. Kurzak, A. Abdelfattah, and J. Dongarra, "An Empirical View of SLATE Algorithms on Scalable Hybrid Systems," 2019.

[15] J. Loffeld and J. A. F. Hittinger, "On the Arithmetic Intensity of High-Order Finite-Volume Discretizations for Hyperbolic Systems of Conservation Laws," *International Journal of High Performance Computing Applications*, vol. 33, no. 1, pp. 25–52, 2019.