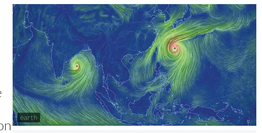
The arithmetic intensity of high-order discontinuous Galerkin methods for hyperbolic systems

Suyash Tandon and Eric Johnsen

Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, U.S.A

Large-scale computations of turbulent flows

Hi-order methods offer high resolution capabilities that are often required for numerical simulation of complex turbulent flows, which are characterized by chaotic and highly unsteady motion of diverse spatial and temporal scales.

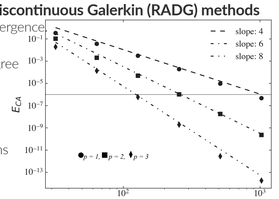


Streamlines showing turbulent wind flow along the surface of the earth. Data accessed from site: http://earth.nullschool.net/ on May 8, 2020.

Recovery-assisted discontinuous Galerkin (RADG) methods can achieve super convergence

2p+2 where ***p*** is degree of the polynomial basis.

Desired accuracy is achieved with lower floating-point operations (FLOP) at high ***p***.

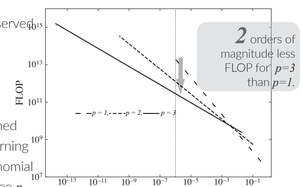


Superconvergence is observed for RADG schemes with order of accuracy given by 2p+2.

For the hyperbolic system,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}} = 0,$$

u(x,t) is a vector of conserved variables, and ***F(u)*** are vector-valued flux functions.



2 orders of magnitude less FLOP for ***p=3*** than ***p=1***.

*FLOP count required to achieve a desired accuracy for a 3D scalar advection problem with RADG for varying polynomial order **p**.*

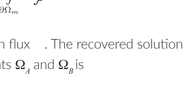
DG weak-form is obtained by multiplying the governing equation with the polynomial basis ϕ_m of at most degree ***p***, and integrating over each element Ωm with outward normal $\mathbf{n}^-$ at the element interface

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

Use recovery to evaluate the common flux $\widetilde{\mathcal{F}}$. The recovered solution $\mathbf{f(r)}$ over the union of adjacent elements Ω_A and Ω_B is

$$\mathbf{f}(\mathbf{r}) = \sum_{n=0}^{2K-1} \Psi^n(\mathbf{r}) \hat{\mathbf{f}}^n.$$

The recovery basis is a $2K$-dimension polynomial, where $\boldsymbol{K = (p+1)^D}$ in D-dimensions.



The recovered solution is weakly equivalent to numerical solutions U_A^h and U_B^h over the union with respect to $k = 0, 1, ..., K-1$ DG basis functions of both Ω_A and Ω_B, i.e.,

$$\int_{\Omega_A} \phi_A^k \mathbf{f}\, d\mathbf{x} = \int_{\Omega_A} \phi_A^k \mathbf{U}_A^h\, d\mathbf{x}, \quad \text{and} \quad \int_{\Omega_B} \phi_B^k \mathbf{f}\, d\mathbf{x} = \int_{\Omega_B} \phi_B^k \mathbf{U}_B^h\, d\mathbf{x}.$$

The recovery is exclusively used to calculate the interface quantity $\mathbf{f(0)}$ located at x_I, and this entire procedure is encapsulated in a matrix-vector multiplcation with recovery operator $\mathcal{R}$,

$$\mathbf{f}(\mathbf{r}=0) = \mathcal{R} \begin{bmatrix} \hat{\mathbf{U}}_A \\ \hat{\mathbf{U}}_B \end{bmatrix}.$$

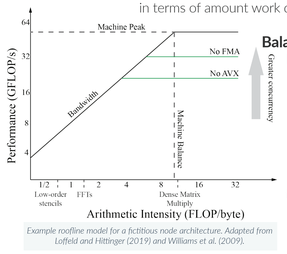
Pass the recovered solutions to a Riemann solver to calculate the flux

$$\widetilde{\mathcal{F}} = Riе(f_A|_{x_I} + f_B|_{x_I})$$

Methods

1 Roofline-model

Measures the performance of an algorithm on a given architecure in terms of amount work done per data transferred.



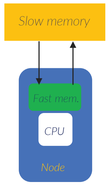
Example roofline model for a fictitious node architecture. Adapted from Lolfeld and Hittinger (2019) and Williams et al. (2009).

Balance point is the optimal location on a roofline plot

Low-order stencil calculations such as traditional methods used in simulations of turbulent flows have very low arithmetic intensity

Most mid-range high-performance computing (HPC) platforms have a machine balance of ***>5***

2 Simple machine model



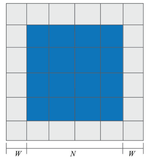
2 levels of memory: unlimited amount of slow memory limited amount of fast memory on the chip

Computation is performed by CPU on data residing in the fast memory

2 data transfers per computation: ***Load*** from slow memory, and ***store*** back in slow memory

Assumptions include decomposing vectorize-addition (VXA) and fused-multiply-add (FMA) operations into individual operations. Multithreading has no effect on the number of operations performed.

3 Problem description



*Schematic of a 2D box-structured grid employed for RADG schemes. Blue elements constitute the domain of interest and the outer grey elements form the ghost cell region. For RADG, the ghost cell layer width is **W = 1** on each side per direction.*

Box structured grid with ***N*** elements and layer of ghost cells of width ***2W*** in each direction

For RADG schemes ***W = 1***, giving total ***N+2*** elements per direction

Arithmetic Intensity

Algorithm design that maintains trade-off between data motion, memory usage, and operations is essential to extract benefits from modern supercomputers with heterogeneous architectures

1 Floating-point operation (FLOP) count

For RAGD schemes, the residual evaluation given by the right-hand side of DG weak-form as,

$$R_m = \int_{\Omega_m} (\nabla \phi_m^k) \cdot \mathcal{F}(U_m^h) d\mathbf{x} - \int_{\partial\Omega_m} \phi_m^k (\widetilde{\mathcal{F}} \cdot \mathbf{n}^-) ds,$$

is broken down into steps. The FLOP count of each step is evaluated, which increases with increase in ***p*** and ***N***.

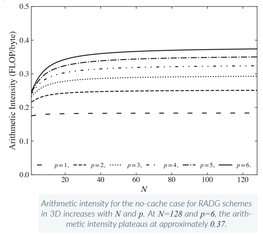
2 No-cache case

Degenerate case that mimics poor cache utilization

Provides a lower limit on the performance

Cache-size is small to hold data for ***1*** operation per step

Large number of data transfers reduce the arithmetic intensity to < ***0.4***



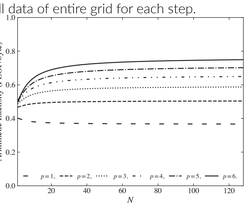
Arithmetic intensity for the no-cache case for RADG schemes in 3D increases with N and p. At N=128 and p=6, the arithmetic intensity plateaus at approximately 0.37.

3 Infinite-cache case with intermediate data transfers

Cache is large enough to hold all data of entire grid for each step.

Intermediate data, at each step are transferred back in memory.

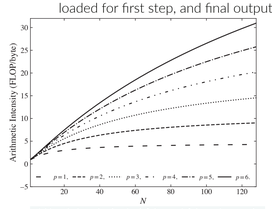
The rate of increase in FLOP is overshadowed



Arithmetic intensity for the infinite-cache case for RADG schemes in 3D where the data for each step of residual evaluation is transferred. The arithmetic intensity increases with N and p, such that at N=128 and p=6, the arithmetic intensity plateaus at approximately 0.76.

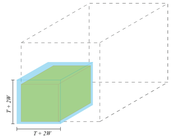
Infinite- & finite-cache case w/o intermediate data transfers 4

Improve performance by reducing data transfers such that data are loaded for first step, and final output is stored after last step with no other data transfers between steps.



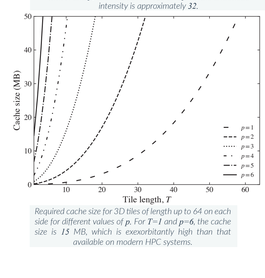
Arithmetic intensity for the infinite- and finite-cache case for RADG schemes in 3D where the data for each step of residual evaluation is kept in the cache. The arithmetic intensity increases with N and p, such that at N=128 and p=6, the arithmetic intensity is approximately 32.

In infinite-cache case, data of entire grid are loaded at once



In finite-cache case, data are broken into cubic tiles of length ***T*** such that, $\boldsymbol{N = m \times T}$, for $\boldsymbol{m > 0}$

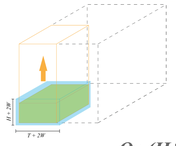
RADG schemes can achieve arithmetic intensity > ***10*** at large values of $\boldsymbol{N}$



*Required cache size for 3D tiles of length up to 64 on each side for different values of **p**. For **T=1** and **p=6**, the cache size is **15 MB**, which is exexorbitantly high than that available on modern HPC systems.*

For ***p>2***, RADG schemes are limited by cache size . The recovery matrix $\mathcal{R}$ requires the most memory. In practice modern HPC platforms have ***~2.5MB*** of cache per core.

Improving the cache size requirement for RADG schemes 5



Choose flattened rectangular tiles of size $\boldsymbol{T \times T \times H}$ where ***H*** is height of the tile.

Cache space is evaluated as

$$\boldsymbol{Qv\,(H+2)(T+2)^2 + Qv\,(HT^2) + Qs \times N_F}$$

where N_F is total number of faces in a rectangular tile.

Cache space reduces to ***2MB*** for rectangular tile ***8x8x2*** with ***p = 3*** from ***15***MB for cubic tile of size ***8³***.

6 Numerical Experiment

A 3D cube of length $L = 2\pi$ is initialized with velocity fields ***(u, v, w)*** in the ***x, y*** and ***z*** directions respectively as,

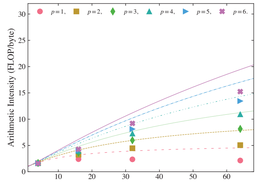
$$u = \sin(x/L)\cos(y/L)\cos(z/L),$$
$$v = w = 0.$$

A simple linear advection system for single component ***u*** is formulated as

$$\frac{\partial u}{\partial t} + \frac{\partial \mathcal{F}}{\partial x} = 0,$$

where $\mathcal{F} = a.u$ with advective speed $\boldsymbol{a = (\pi, 0, 0)^T}$.

Four cases are considered with $\boldsymbol{N = 4, 16, 32}$, and ***64***.



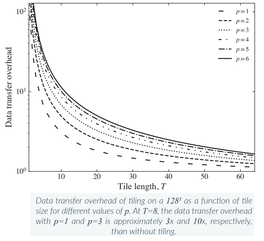
Measured (symbols) versus predicted (lines) arithmetic intensity for RADG schemes on cubic tiles.

Hardware performance counters on the nodes of the Intel-Knights-Landing-based XSEDE ***Stampede2*** system measure the FLOP and data transfers. The counters are read through the Intel Advisor XE. Measurements correspond to the residual evaluation for single time-step with no time-marching.

Measured arithmetic intensity is significantly lower than predicted estimates.

Reasons include, high cache size requirement for ***N>8*** and ***p>2***, which leads to tiling.

For tile lengths ***T<32***, the overheads are high that reduces the arithmetic intensity.



*Data transfer overhead of tiling on a **128³** as a function of tile size for different values of **p**. At **T=8**, the data transfer overhead with **p=1** and **p=3** is approximately **3x** and **10x**, respectively, than without tiling.*

Conclusions

1. It is important to emphsize on-node performance of algorithms along with parallel scalability, and therefore, assessing arithmetic intensity must be considered as an important design criterion.
2. Three data cache strategies are investigated - lower bound no-cache case, ideal infinite-cache and a finite-cache which is more representative of reality.
3. While the RADG schemes are predicted to require high cache size that is mostly dominated by the large recovery operator, the cache size can be relaxed by using a flattened rectangular tiling.
4. RADG schemes are predicted to achieve high arithmetic intensity, which is necessary to better utilize on-node floating-point capabilities of modern HPC platforms.
5. Numerical experiment shows that measured values of arithmetic intensity for RADG are lower than the predicted estimates mostly due to data tiling overheads at large values of ***N*** and ***p***.
6. Further investigation is needed to modify the implementation of recovery operator that reduces the storage requirement and increases the arithmetic intensity if RADG in practical applications.

Table: Floating-point operations (FLOP) count per time-step for RADG discretization of hyperbolic systems.

$Qs = (p+1)^{D-1}$ is the number of quadrature points along an element interface, $K = Qv = (p+1)^D$ is the number of degrees of freedom in an element, N is the number of elements in each direction with $W=1$, $Nv = D(N+2W+1)(N+2W)^{D-1}$ is the total number of interfaces in D-dimensions, $f_F = DQv$, and for Rusanov flux $f_R = 4DQs$.

#	Steps in the residual evaluation	FLOP count	No-cache case data transfers	Infinite-cache case with intermed. data transfers	Infinite- and finite-cache case without intermed. data transfers
1	U_L and U_R, at interface quadrature points	$8Qs\,KNv$	$2f(R+2Qv+Qs)Nv$	$Qv\,(N+2)^D + (R+2Qs)Nv$	$Qv\,(N^D + (N+2)^D) + 2DR(N+2)^{D-1}$
2	$\widetilde{\mathcal{F}}$, at interface quadrature points	$f_R\,Nv$	$3Qs\,Nv$	$3Qs\,Nv$	--
3	U^h, at interior quadrature points	$2Qv\,KN^D$	$2Qv\,N^D$	$2Qv\,N^D$	--
4	$\mathcal{F}(U^h)$, at interior quadrature points	$f_F N^D$	$(2D+1)Qv\,N^D$	$(2D+1)Qv\,N^D$	$Qv\,X\,N^D$

Acknowledgements

This work was supported in part by the Michigan Institute for Computational Discovery and Engineering (MICDE). Additionally, this research used the Extreme Science and Engineering Discovery Environment's Stampede2 system at the Texas Advanced Computing Center (TACC), which is supported by the NSF grant number ASC-130005.

References

[1] D. Brown, P. Messina, D. Keyes, et al., "Scientific grand challenges: Crosscutting technologies for computing at the exascale," U.S. Department Of Energy, Pacific Northwest National Laboratory, Tech. Rep., 2010.
[2] J. Dongarra, J. Hittinger, J. Bell, et al., "Applied Mathematics Research for Exascale Computing," Tech. Rep., 2014.
[3] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, "The Development of Discontinuous Galerkin Methods," Discontinuous Galerkin Methods, pp. 77–88, 2000.
[4] P. E. Johnson and E. Johnsen, "The Compact Gradient Recovery Discontinuous Galerkin Method for Diffusion Problems," J. Comp. Phys., vol. 398, p. 108872, 2019.
[5] 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.
[6] 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.
[7] J. Lofeld 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.