Multi-GPU Solutions of Geophysical PDEs with Radial Basis Function-generated Finite Differences

Evan F. Bollig

Nov 6th, 2013

499 DSL, 4:30 pm – 6:30 pm

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

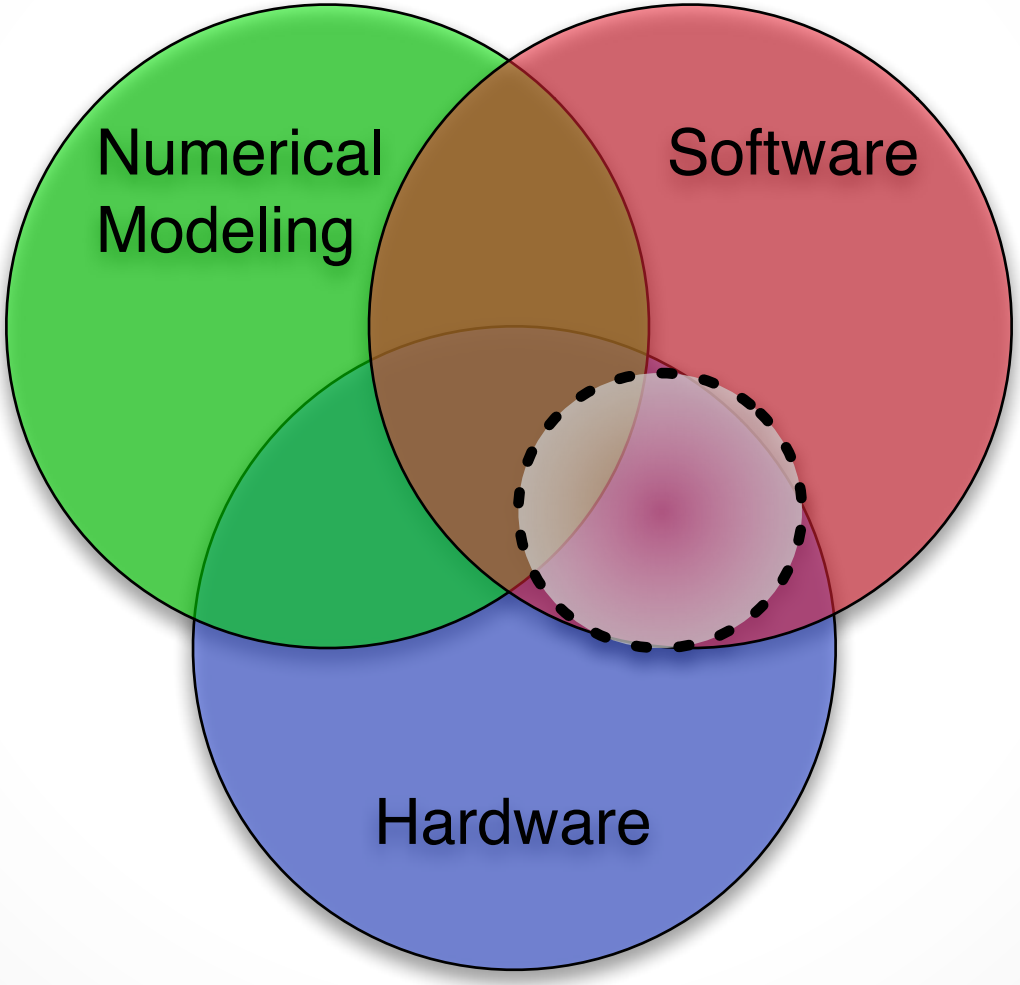
NSF

Acknowledgements

- This work is supported in part by NSF awards **DMS-#0934331** (FSU), **DMS-#0934317** (NCAR) and **ATM-#0602100** (NCAR).
- Computational resources provided by:
  - 

  - Keeneland Computing Facility at the Georgia Institute of Technology (**NSF Contract OCI-0910735**)

Investigation Focus

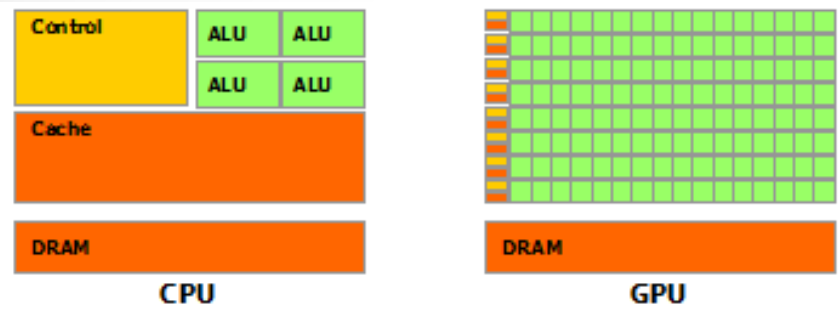


Overview

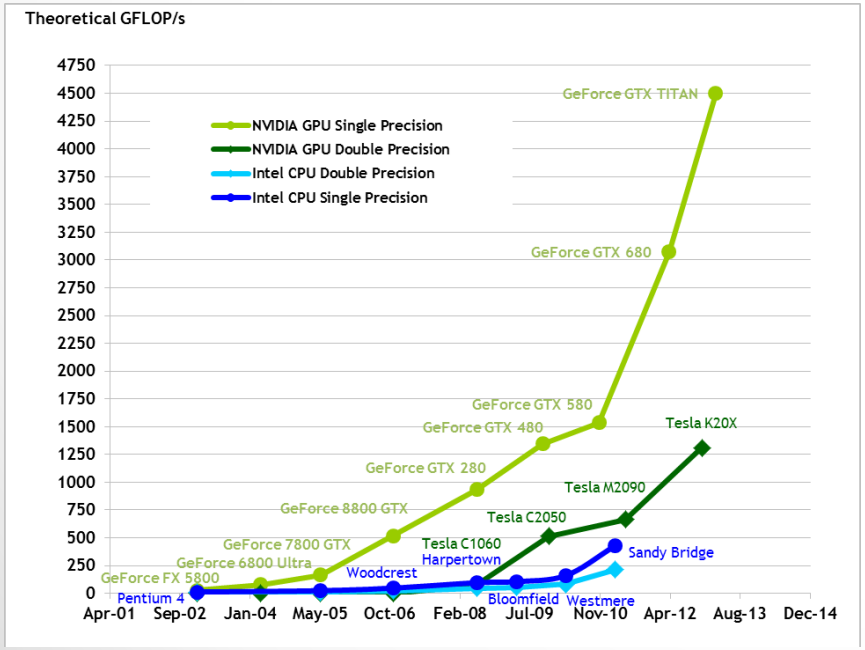
- Preliminaries
  - Introduction to GPUs
  - Introduction to RBF-FD
  - Code Verification Details
- A Fixed-Grid Method for RBF Methods
  - Space Filling Curves and Node Ordering
  - Lasting Impacts on Performance
- Single GPU RBF-FD
- Targeting Multi-GPU Clusters
  - MPI Tuning
  - Overlapping Communication and Computation
- Conclusions

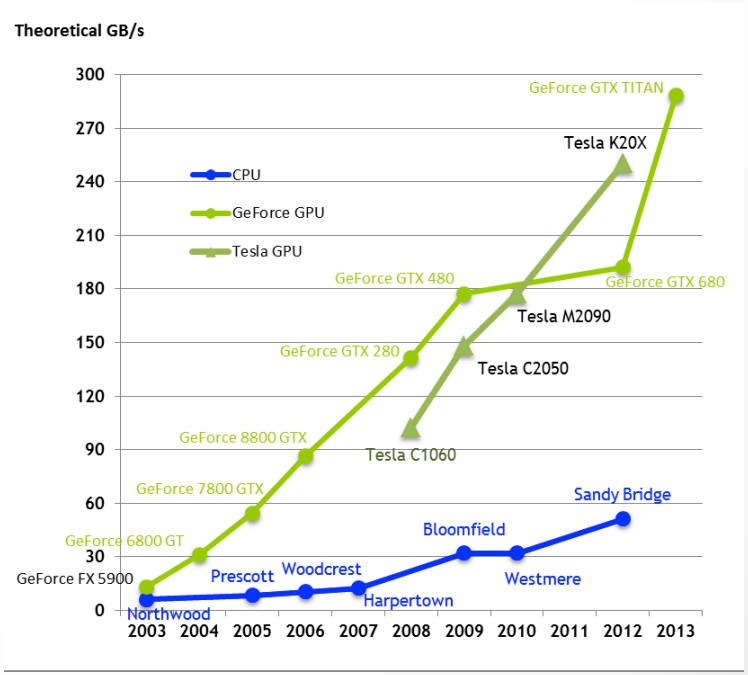
Images courtesy of NVidia

Why GPUs?

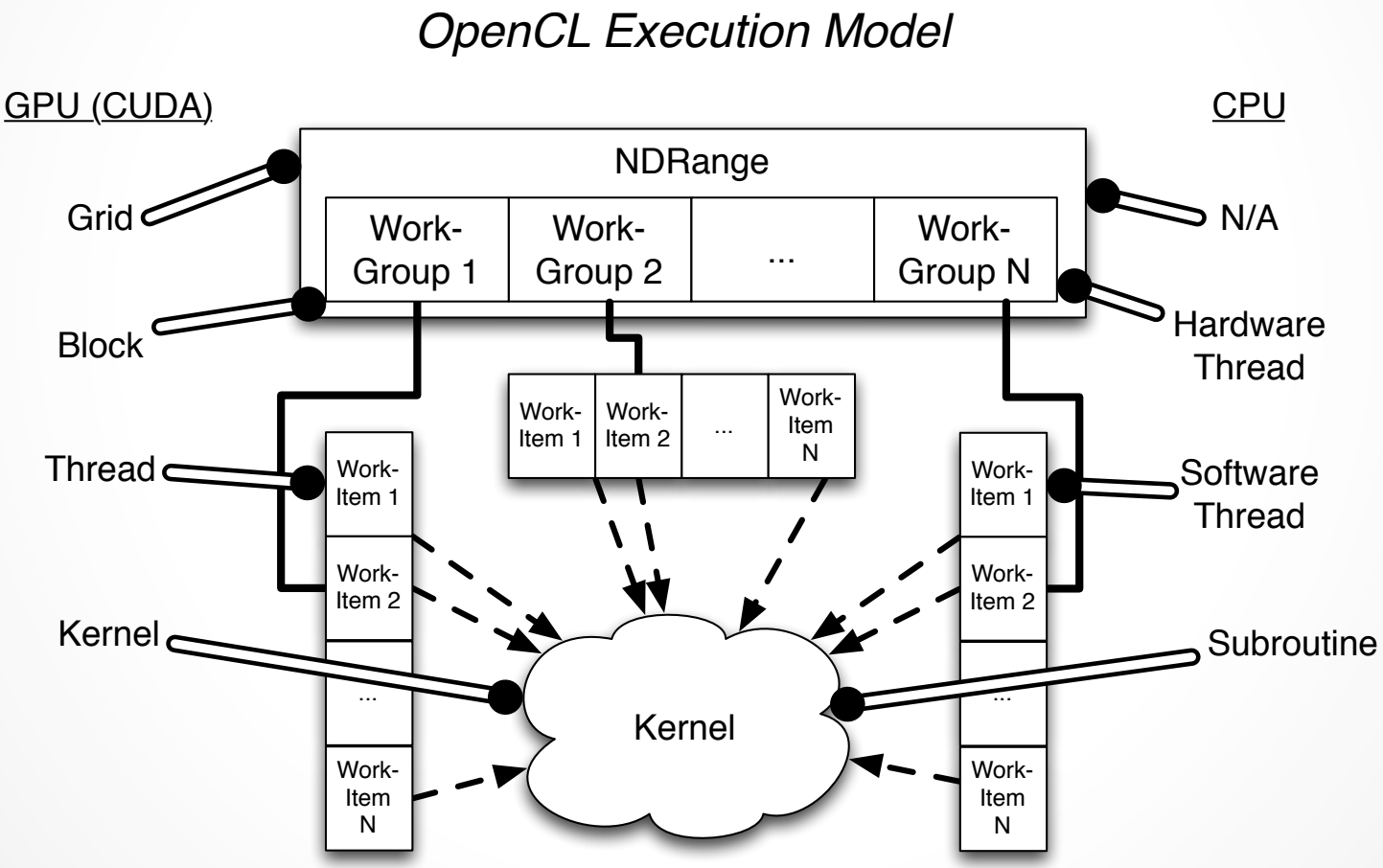


- Massive parallelism
- High GFLOP/sec
- Fast Memory



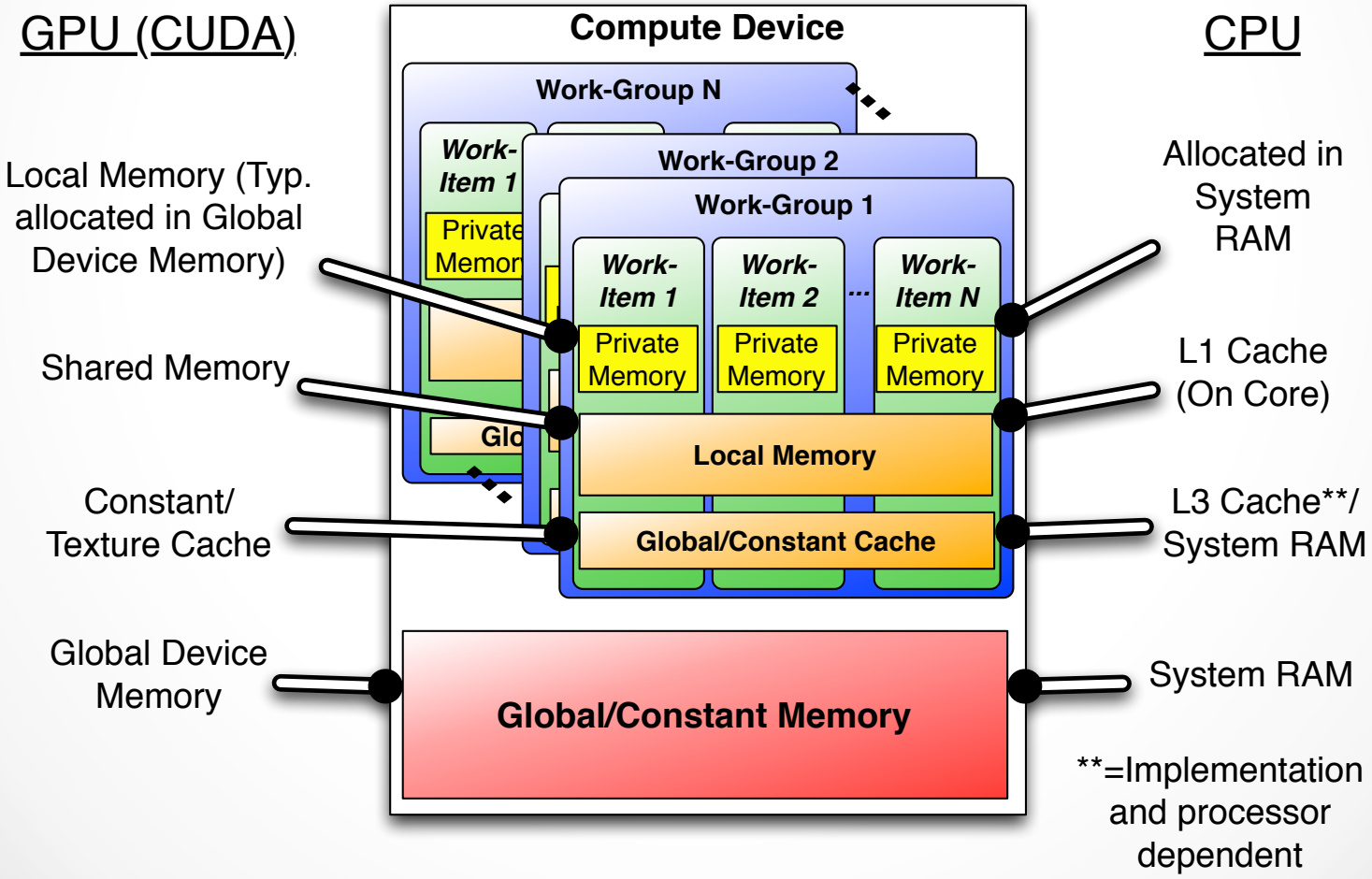


How to Target GPUs?



GPU Memory Model

OpenCL Memory Model



OpenCL Event Queues

- Queues allow non-blocking kernel launches
- Concurrent kernel execution (on Fermi and Kepler GPUs)
- Shared memory pool between queues.

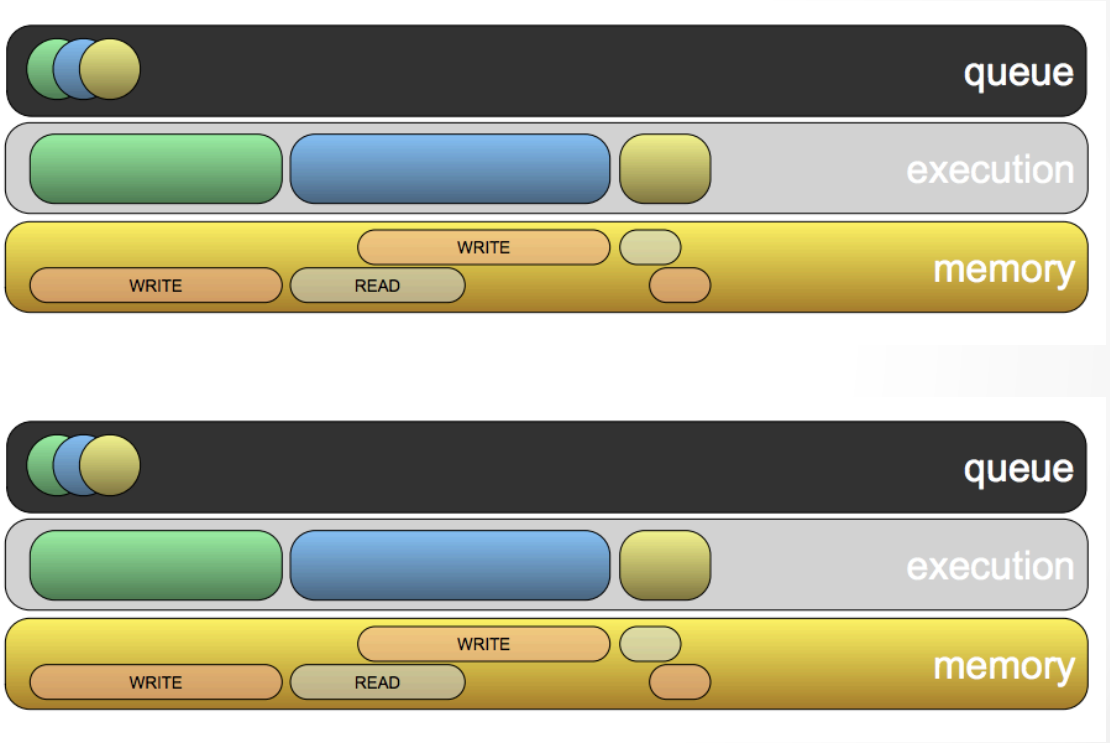
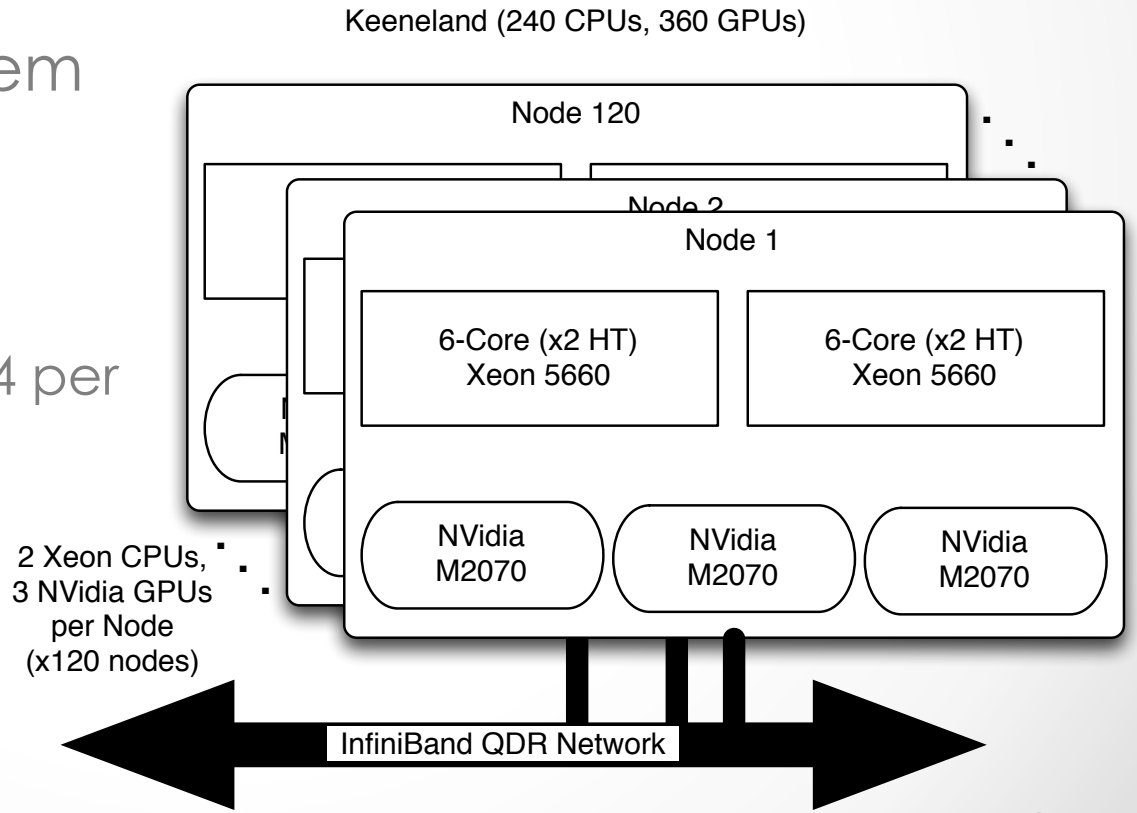


Image courtesy of Gerstmann, 2009. *Advanced OpenCL Event Model Usage* (tutorial slides)

Target: GPU Cluster

- Keeneland Initial Delivery (KID) System
  - 360 NVidia M2070
- Cascade
  - 32 NVidia M2070 (4 per node)
  - 8 NVidia K20



Multi-GPU Concerns

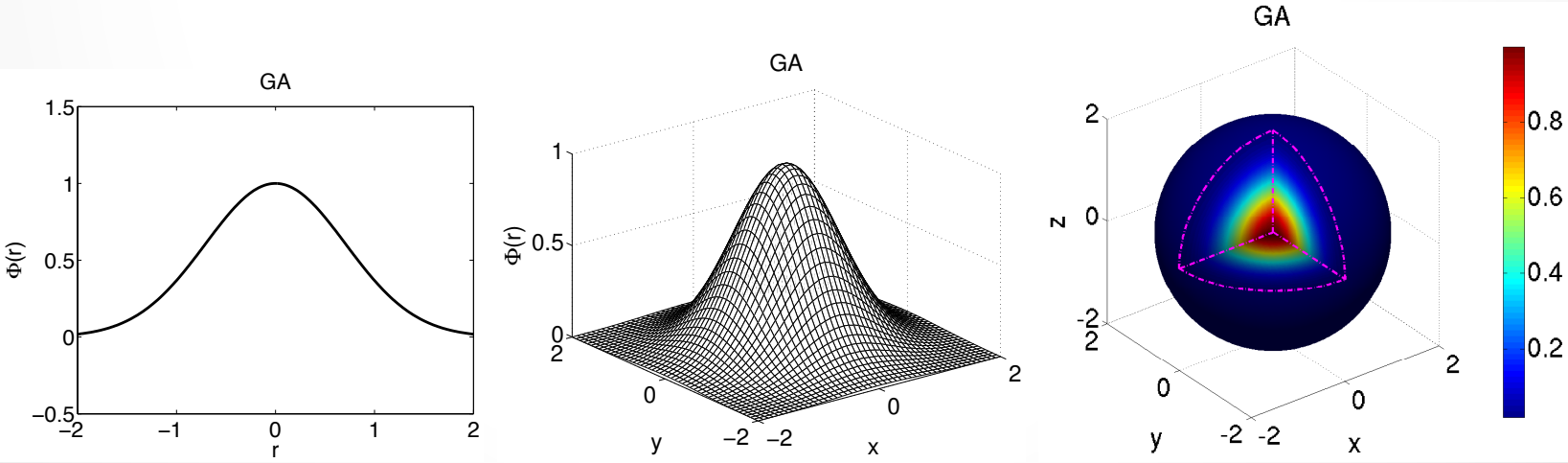
- How to partition and distribute our problem?
- Faster computation increases relative cost of communication. Can it be amortized/hidden?
- Multi-GPU OpenCL requires device ←→ host copies. Can they be hidden?

What is a Radial Basis Function?

- A radially symmetric function centered at $\mathbf{x}_j$

$$\phi_j(\mathbf{x}) = \phi(\epsilon r(\mathbf{x}))$$

$$r(\mathbf{x}) = ||\mathbf{x} - \mathbf{x}_j||_2 = \sqrt{(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2}$$

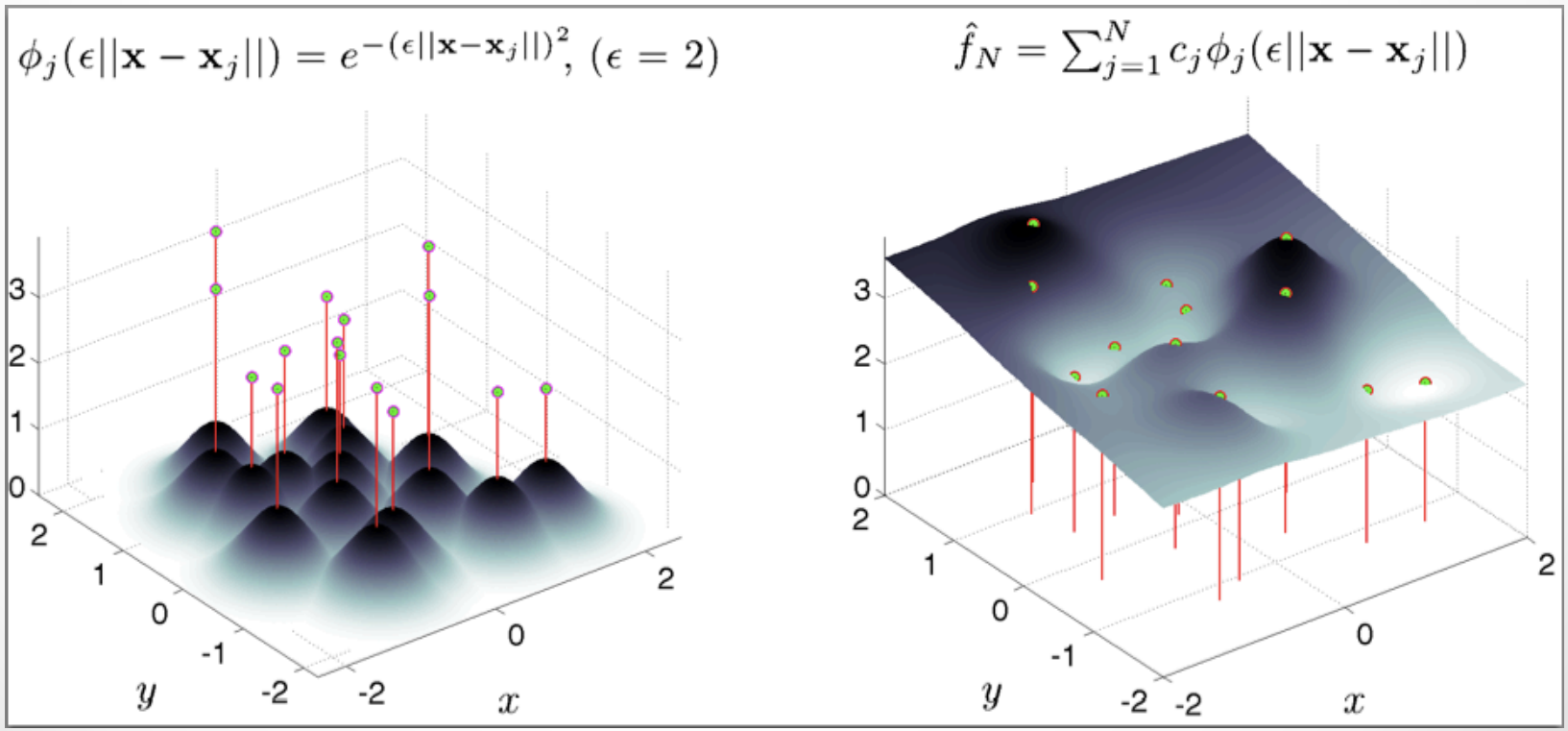


RBF-Interpolation

$$\sum_{j=1}^{N} c_j \phi_j(x) + \sum_{l=1}^{M} d_l P_l(x) = f(x), \qquad P_l(x) \in \Pi_m^d$$

$$\sum_{j=1}^{N} c_j P_l(x_j) = 0, \qquad l = 1, ..., M$$

$$\underbrace{\begin{bmatrix} \phi & P \\ P^T & 0 \end{bmatrix}}_{A} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$$



Pros/Cons of RBF Methods

- Pros
  - Function independent of dimension
  - Operate on unstructured meshes
  - High order convergence
  - Large time-steps
- Cons
  - Support parameter, ϵ
    - Uncertainty Relation (Schaback, 1995)
    - Can be overcome with RBF-QR (Fornberg and Piret, 2007), RBF-GA (Fornberg, Lehto and Powell, 2013)

RBF-generated Finite Differences (RBF-FD)

Classical FD (1-D*):

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \ddots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \mathcal{L}1\big|_{x=x_c} \\ \mathcal{L}x\big|_{x=x_c} \\ \mathcal{L}x^2\big|_{x=x_c} \\ \vdots \\ \mathcal{L}x^{n-1}\big|_{x=x_c} \end{pmatrix},$$

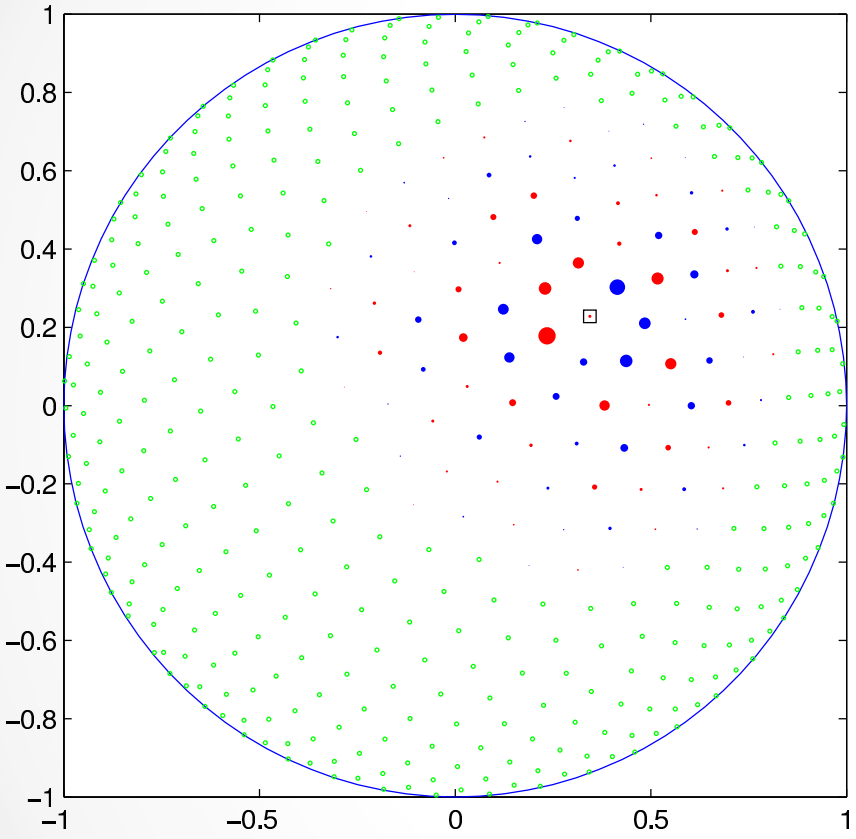
*Nonsingularity lost in 2-D; compose weights across dimensions

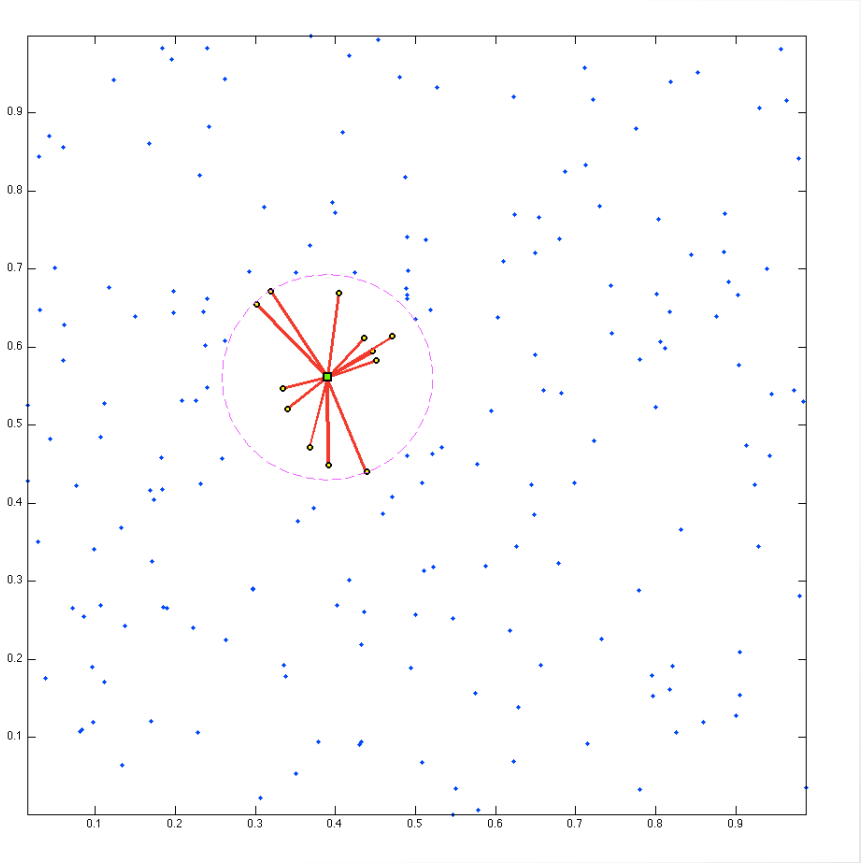
RBF-FD (d-D):

$$\begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) & 1 \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_n(x_2) & 1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \\ c_{n+1} \end{pmatrix} = \begin{pmatrix} \mathcal{L}\phi_1(x)\,|_{x=x_c} \\ \mathcal{L}\phi_2(x)\,|_{x=x_c} \\ \vdots \\ \mathcal{L}\phi_n(x)\,|_{x=x_c} \\ 0 \end{pmatrix}$$

$$\begin{bmatrix} \phi & P \\ P^T & 0 \end{bmatrix} \begin{pmatrix} c_{\mathcal{L}} \\ d_{\mathcal{L}} \end{pmatrix} = \begin{pmatrix} \phi_{\mathcal{L}} \\ 0 \end{pmatrix}.$$

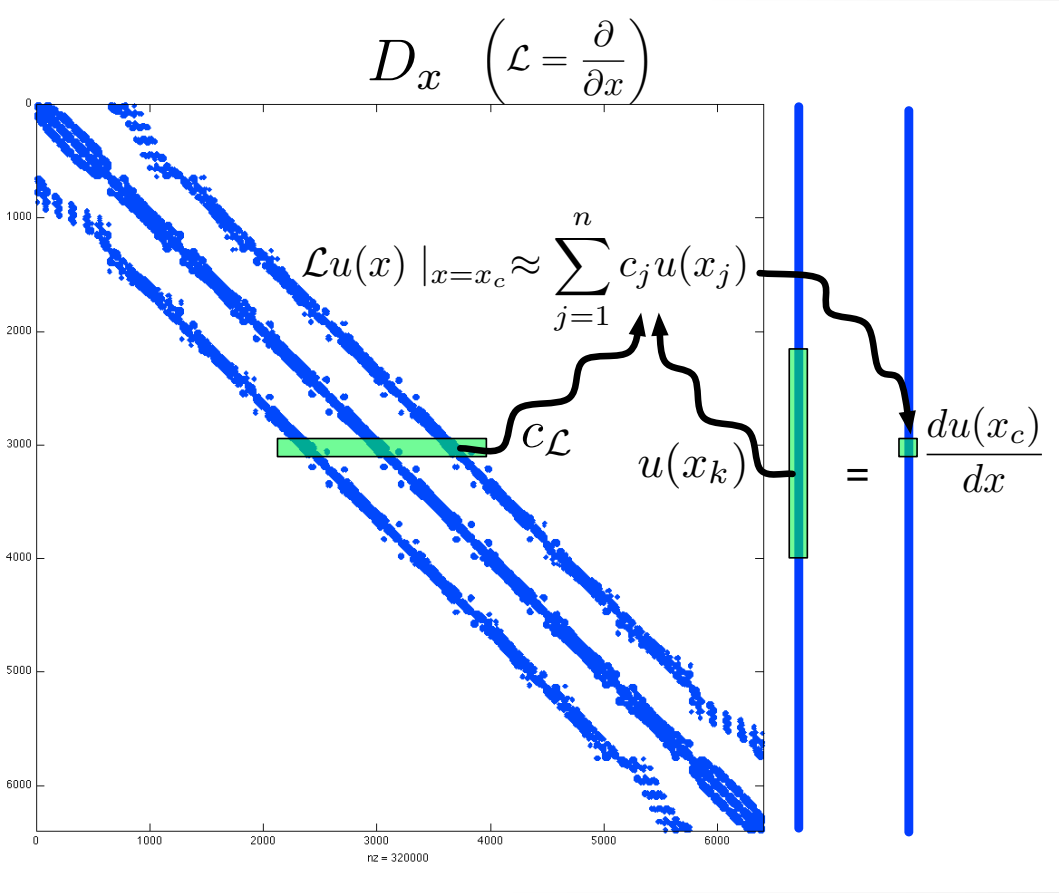
What Kind of Stencils?





Differentiation Matrix (DM)

- Sparse Matrix
  - n-nonzeros per row (stencil size)
  - Unsymmetric (unique weights per row)
- Sparse Matrix Vector (SpMV) multiply to get derivative across full domain



Multiple Linear Operators

$$\mathcal{L} = \nabla^2, \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \text{etc.}$$

$$\begin{bmatrix} \phi & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} c_{\nabla^2} & c_x & c_y & \cdots \\ d_{\nabla^2} & d_x & d_y & \ldots \end{bmatrix} = \begin{bmatrix} \phi_{\nabla^2} & \phi_x & \phi_y & \cdots \\ 0 & 0 & 0 & \cdots \end{bmatrix}.$$

Solve for weights with LAPACK dgesv or other efficient direct solver

$$\nabla^2 u \approx \left(D_{x^2} + D_{y^2}\right) u = D_{x^2} u + D_{y^2} u \ .$$

Analytically apply operators to the RBF or compose operators like classical FD

Some Weight Operators

First and Second Derivatives:

$$\frac{1}{r}\frac{d}{dr}\phi(r) = -2\epsilon^2\phi(r)$$

$$\frac{\partial^2\phi}{\partial r^2} = \epsilon^2(-2 + 4(\epsilon r)^2)\phi(r)$$

Cartesian Gradient:

$$\frac{\partial\phi}{\partial x} = \frac{\partial r}{\partial x}\frac{\partial\phi}{\partial r} = \frac{(x - x_j)}{r}\frac{\partial\phi}{\partial r}$$

Cartesian Laplacian:

$$\nabla^2\phi = \frac{\partial^2}{\partial r^2}\phi(r) + \frac{d-1}{r}\frac{\partial}{\partial r}\phi(r)$$

RBF-FD for Time-Dependent PDEs

Algorithm 3.1 A High-Level View of RBF-FD

Preprocessing Phase:

$\{x\}_{j=1}^{N} = \textsc{GenerateGrid}()$
for $j = 1$ to N **do**
  Stencil $\{S_{j,i}\}_{i=1}^{n} = \textsc{QueryNeighbors}(x_j)$
end for
$\{x\}_{j=1}^{N_p} = \textsc{DecomposeDomain}(D_{\mathcal{L}}, \{x\}_{j=1}^{N})$
for $j = 1$ to N_p **do**
  $\{w_{j,i}\}_{i=1}^{n} = \textsc{SolveForWeights}(\{S_j\})$
  $D_{\mathcal{L}}^{(j)} = \textsc{AssembleDM}(\{w_j\})$
end for

Application Phase:

$t = t_{min}$
while $t < t_{max}$ **do**
  $\{u'\} = \textsc{SolvePDE}(D_{\mathcal{L}}, \{u\})$
  $\{u\} = \textsc{UpdateSolution}(\{u\}, \{u'\}, \Delta t)$
  $t \mathrel{+}= \Delta t$
end while

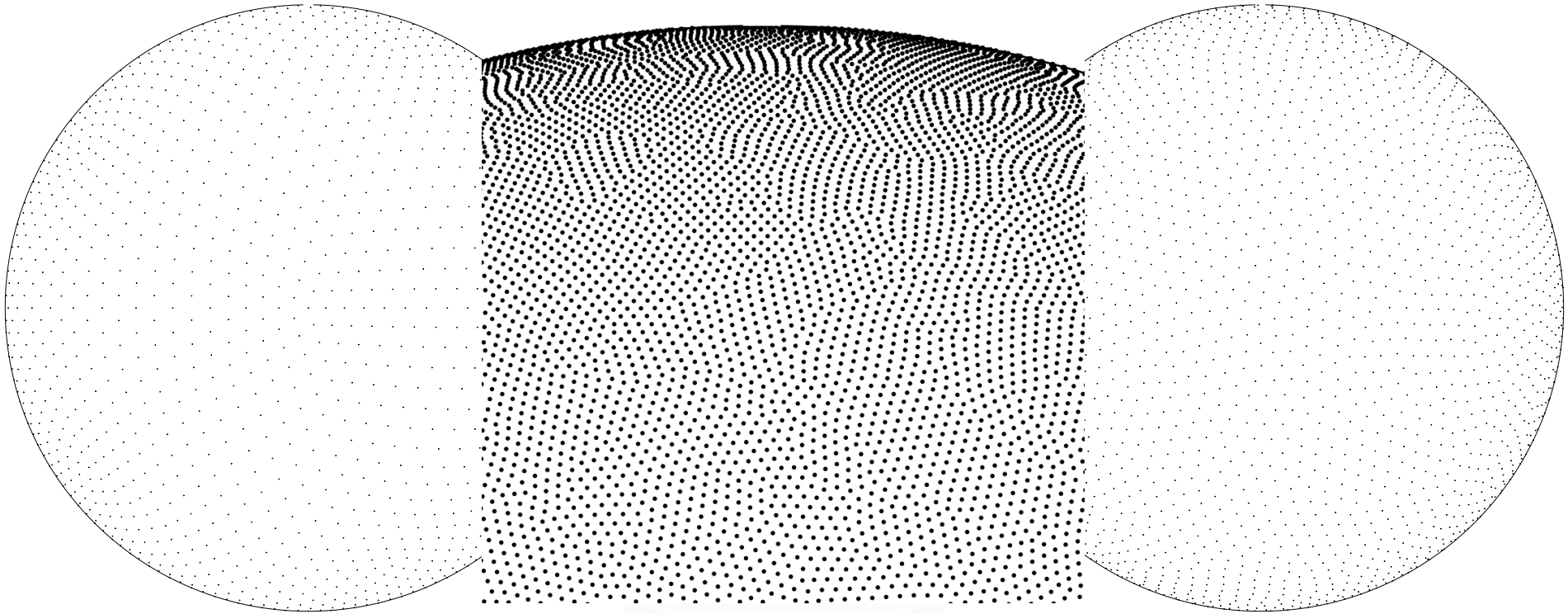
- Assume an input grid is provided.
- We focus on stencil generation, decomposition and the Application Phase.

Unit Sphere Test Grids

Icosahedral (Geodesic) Grid

Spherical Centroidal Voronoi Tessellation

Maximum Determinant (MD)



Related Work on Multi-GPU

	Multi-CPU (⋆:MPI+OpenMP)		Single GPU			Multi-GPU (†:Non-overlapped, ‡:Overlapped)	
	OpenMP	MPI	CUDA	OpenCL	Other	CUDA	OpenCL
RBF-FD							[21]† [This Work]‡
RBF PDE Methods	[99]	[38, 39, 83, 84, 173]	[74, 76, 132, 133]				
Non-PDE RBF Methods		[145]⋆	[32, 33]		[22, 24, 30, 158]		
Non-RBF PDE Methods		[134]⋆	[31]			[67, 68, 69, 70, 98, 122, 149]† [96, 97, 110, 121, 174]‡	

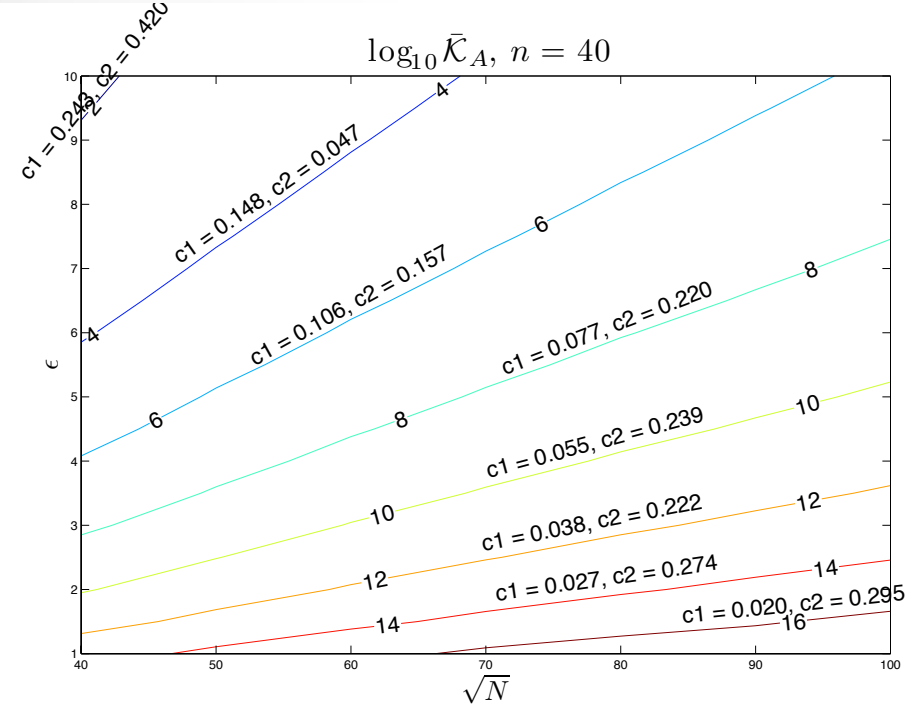
Verification

- Solve hyperbolic equations with DMs of the form:

$$D = \alpha \frac{\partial}{\partial \lambda} + \beta \frac{\partial}{\partial \theta}$$

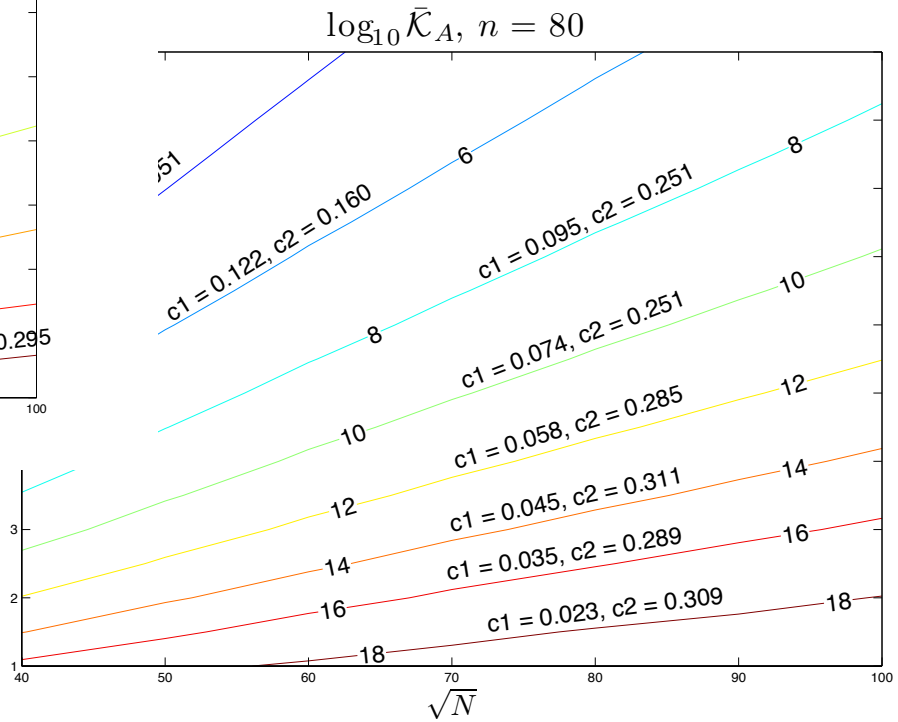
- Apply hyperviscosity (Fornberg and Lehto, 2011) to filter spurious eigenmodes
  - Operator: $\Delta^k \phi(r) = \epsilon^{2k} p_k(r) \phi(r)$
  - Applied as: $\frac{\partial u}{\partial t} = -\mathcal{L}u + Hu,$
  - With scaling: $H = \gamma \Delta^k = \gamma_c N^{-k} \Delta^k.$
- Tuned hyperviscosity parameters are based on trial and error

Support Parameter



Following Flyer et al. 2011, choose ϵ as a function of N:

$$\epsilon(\sqrt{N}) = c_1\sqrt{N} - c_2$$



$$\bar{\kappa}_A = \frac{1}{N}\sum_{i=1}^{N}(\kappa_A)_i$$

$$(\kappa_A)_i = ||A_i^{-1}||_2 \cdot ||A_i||_2$$

Vortex Roll-Up

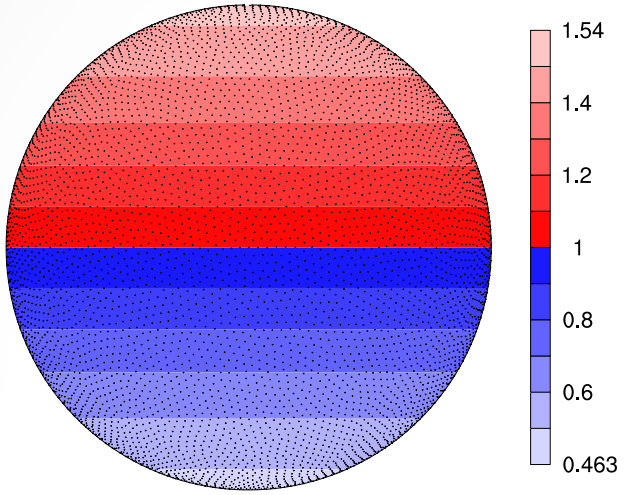
- Purpose: Spin a fluid into two diametrically opposed but stationary vortices.

$$\frac{\partial h}{\partial t} + \frac{u}{\cos\theta}\frac{\partial h}{\partial \lambda} = 0$$

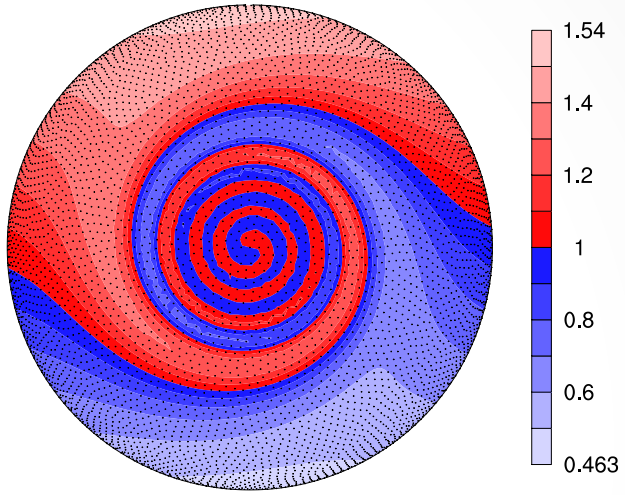
$$u = \omega(\theta)\cos\theta.$$

- Solve with Method of Lines:

$$\frac{d\mathbf{h}}{dt} = -\mathrm{diag}(\omega(\theta))D_\lambda \mathbf{h} + \gamma_c N^{-k} D_{\Delta^k}\mathbf{h},$$

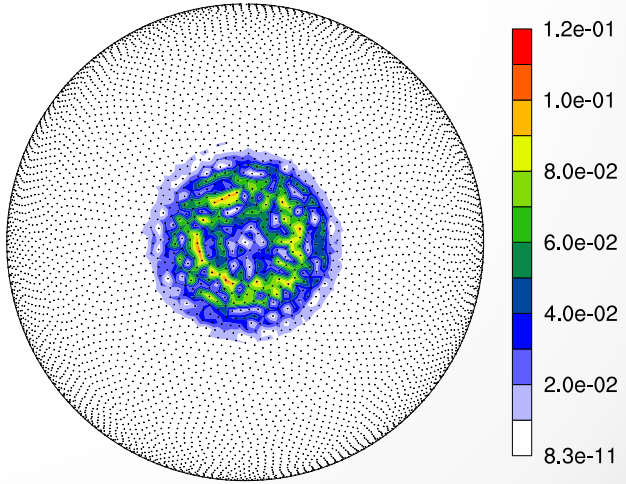


(a) Initial Condition



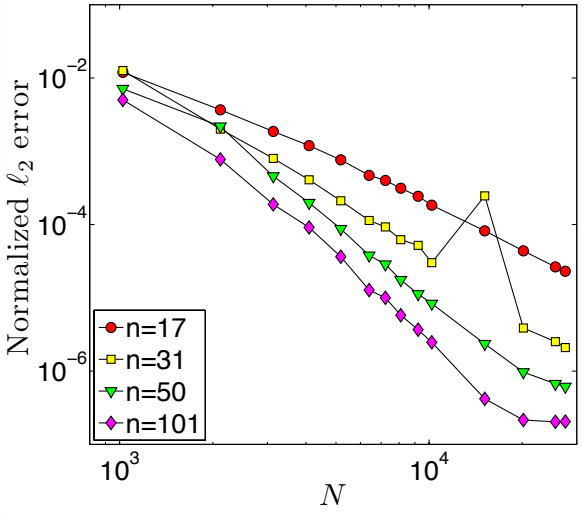
(b) Computed Solution ($t = 10$)

	$\epsilon = c_1\sqrt{N} - c_2$		$H = -\gamma_c N^{-k}\Delta^k$	
Stencil Size (n)	c_1	c_2	k	γ_c
17	0.026	0.08	2	8
31	0.035	0.1	4	800
50	0.044	0.14	4	145
101	0.058	0.16	4	40

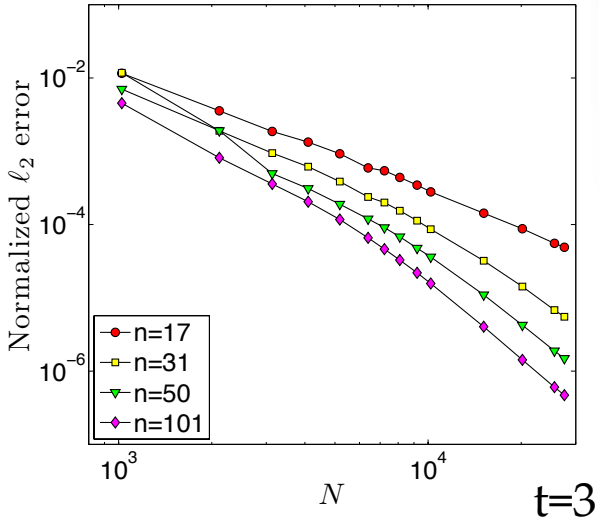


(c) Relative ℓ_2 Error

Convergence (Vortex Roll-Up)

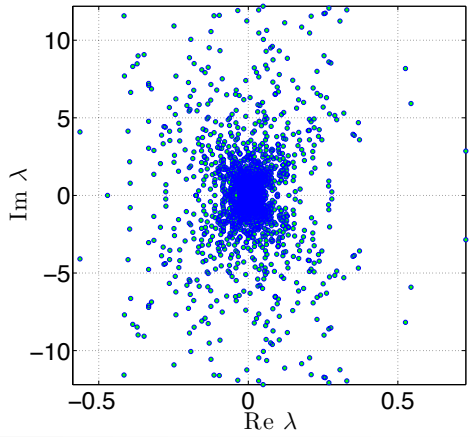


(a) No Hyperviscosity

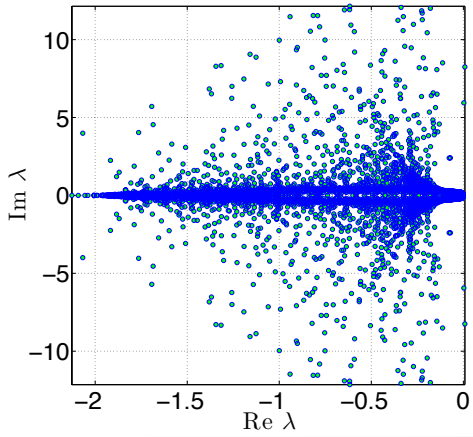


t=3

(b) With Hyperviscosity



(a) No Hypervsiscosity



(b) With Hyperviscosity ($k = 4$ and $\gamma_c = 40$)

Cosine Bell Advection

- Purpose: advect a solid body around the sphere

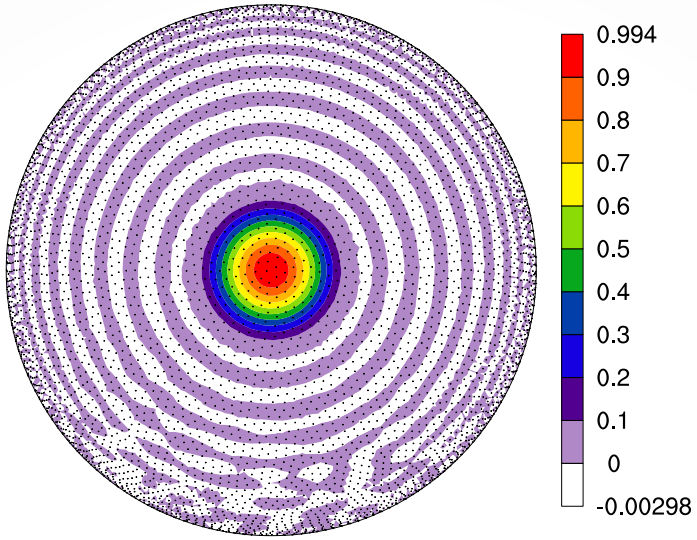
$$\frac{\partial h}{\partial t} + \frac{u}{\cos\theta}\frac{\partial h}{\partial \lambda} + v\frac{\partial h}{\partial \theta} = 0,$$

$$\begin{cases} u = u_0(\cos\theta\cos\alpha + \sin\theta\cos\lambda\sin\alpha), \\ v = -u_0(\sin\lambda\sin\alpha) \end{cases}$$

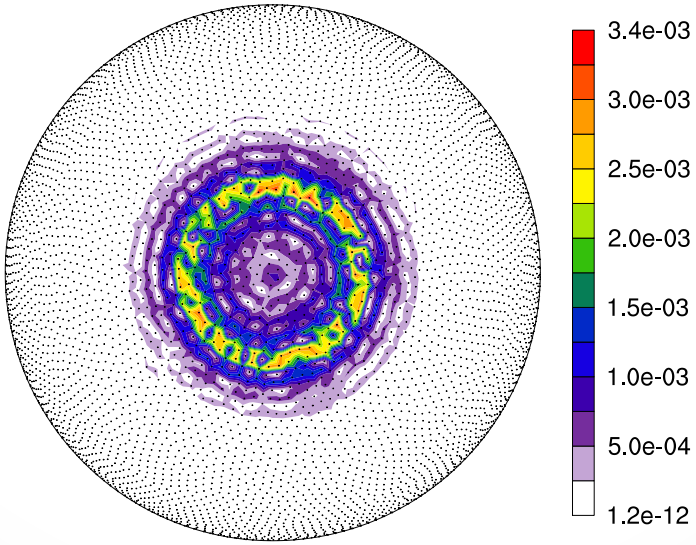
- Method of Lines with two DMs intentionally separated*:

$$\frac{d\mathbf{h}}{dt} = -\mathrm{diag}\left(\frac{u}{\cos\theta}\right) D_\lambda \mathbf{h} - \mathrm{diag}(v) D_\theta \mathbf{h}$$

- * See appendix for discussion on avoiding pole singularities

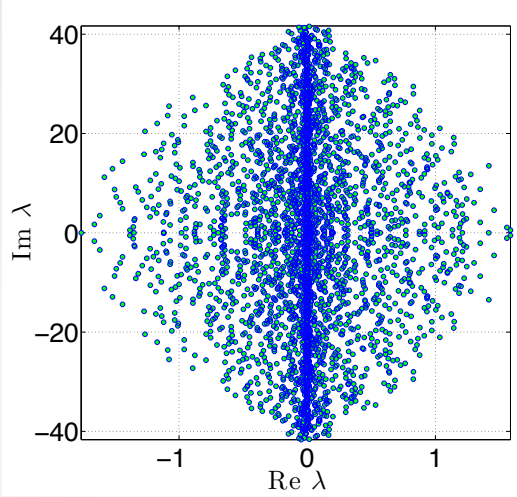


(a) 10 Revolutions

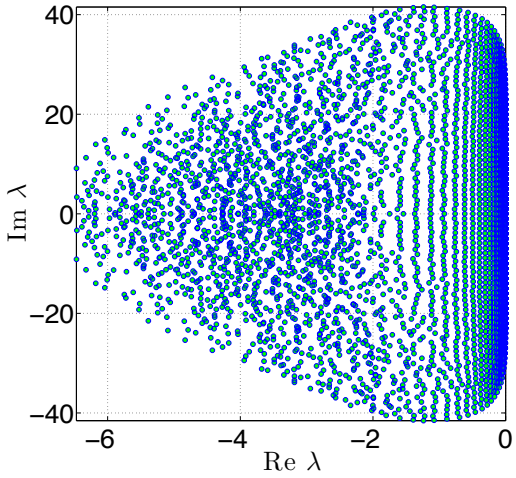


(b) Absolute Error at 10 Revolutions

Stability and Convergence of Cosine Bell Advection



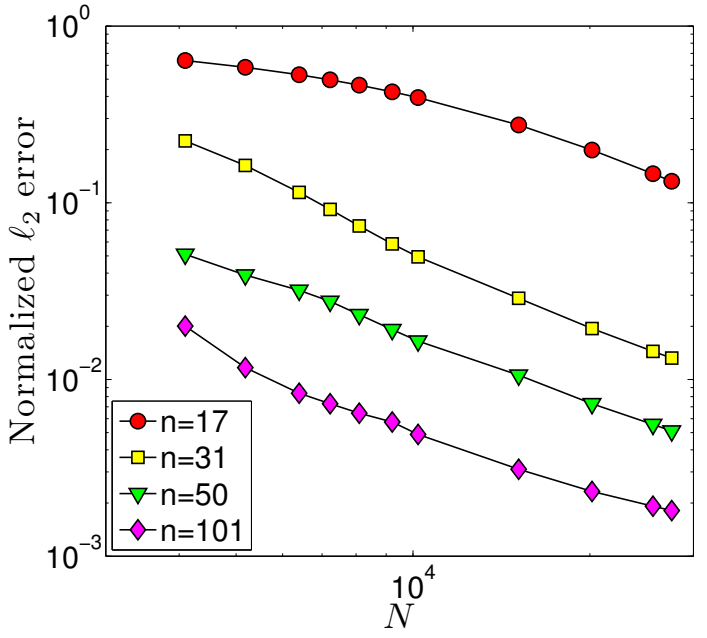
(a) No Hyperviscosity



(b) With Hyperviscosity ($k = 8$ and $\gamma_c = 5 * 10^{-2}$)

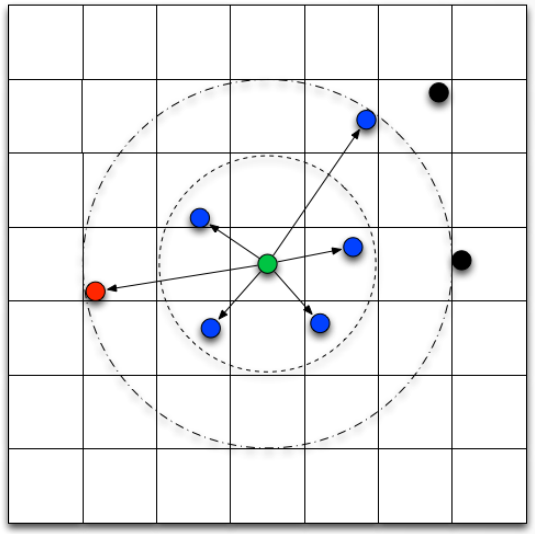
	$\epsilon = c_1\sqrt{N} - c_2$		$H = -\gamma_c N^{-k}\Delta^k$	
Stencil Size (n)	c_1	c_2	k	γ_c
17	0.026	0.08	2	$8 * 10^{-4}$
31	0.035	0.1	4	$5 * 10^{-2}$
50	0.044	0.14	6	$5 * 10^{-1}$
101	0.058	0.16	8	$5 * 10^{-2}$

10 revolutions

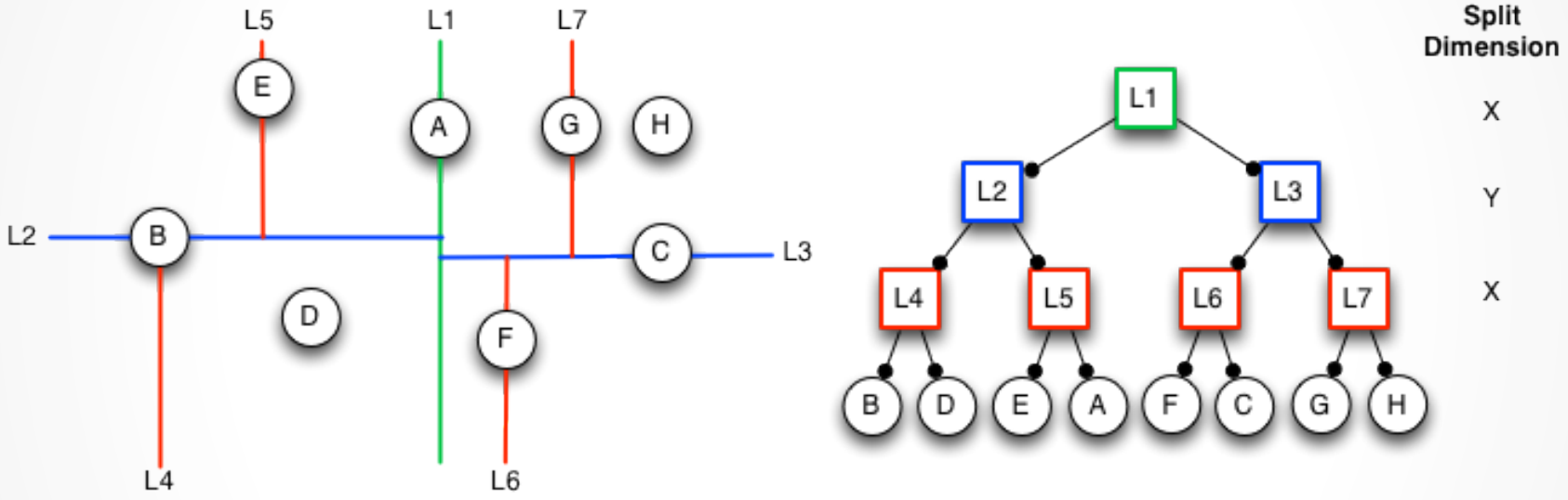


Neighbor Query

- We introduce the fixed-grid method to the RBF community for stencil generation
- Results in minimal gains for stencil generation but works well as preconditioning on nodes for SpMV performance.

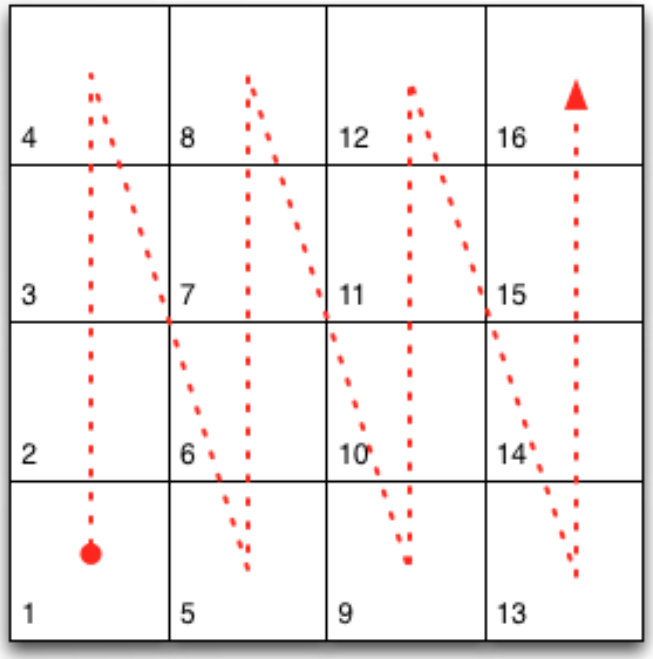


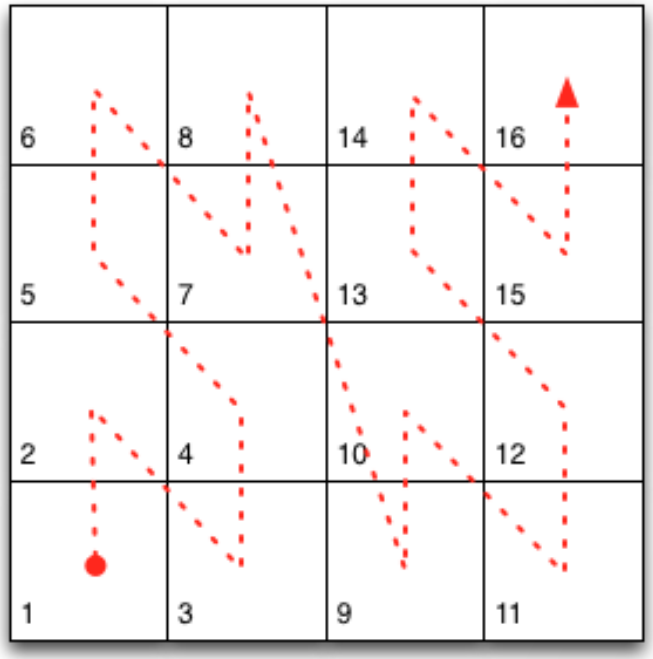
kD-Tree



$O(N \log N)$ to build, $O(N \log N)$ to query

Fixed Grid



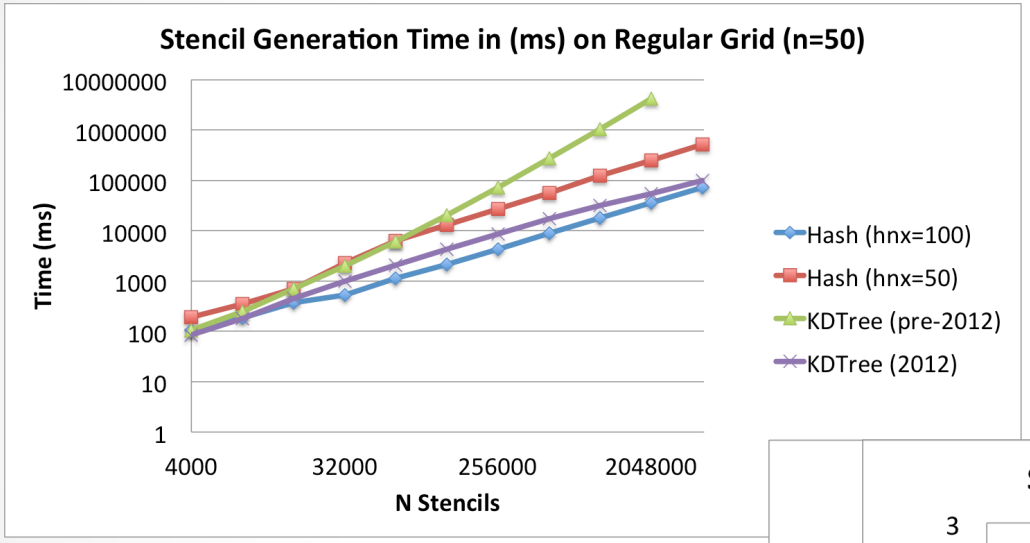


Space-filling curves optional

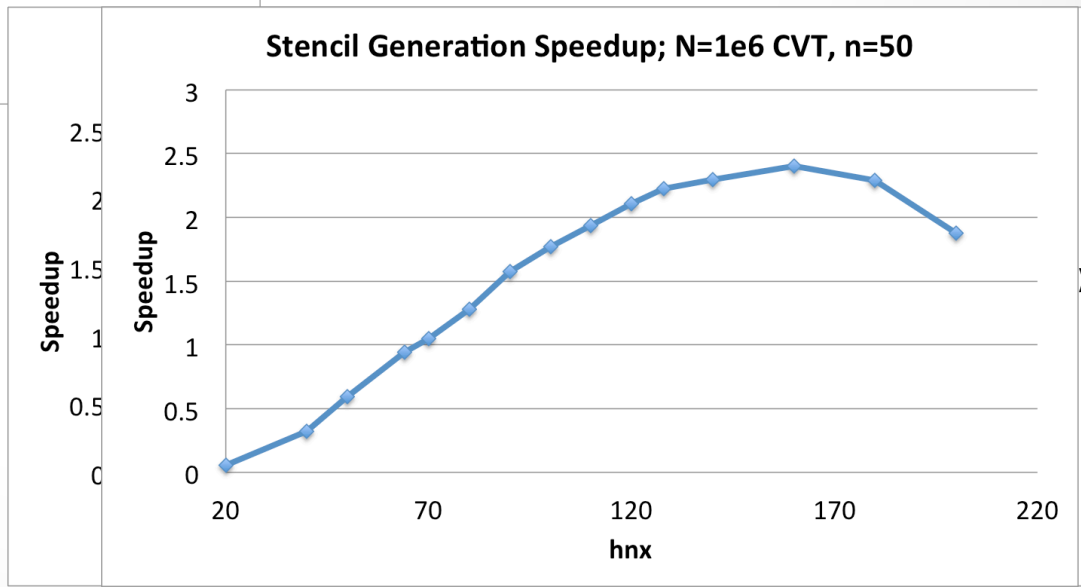
$O(N)$ to build, $O(N \log N)$ to query

Raster Order Fixed Grid Impact

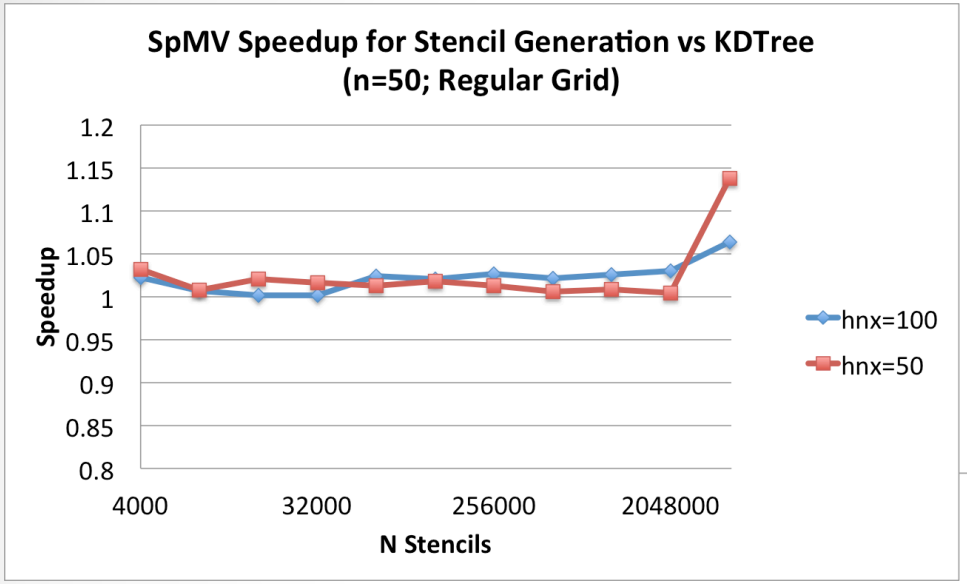
Up to 2x faster for h_n=100 on uniform nodes



Up to 2.5x faster for h_n=100 on random nodes

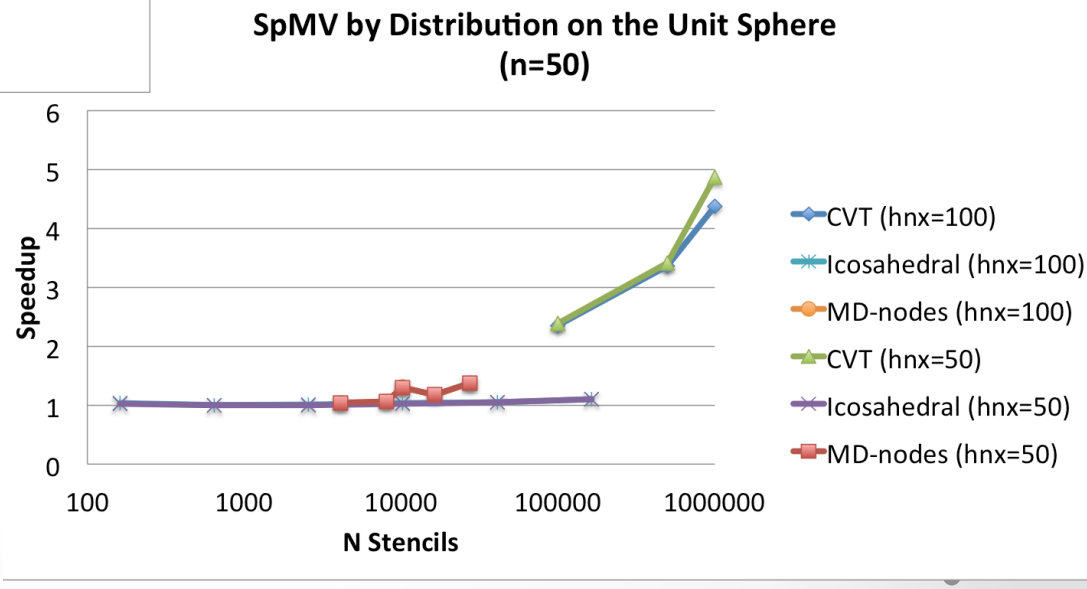


SpMV Impact of Raster Order Fixed-Grid

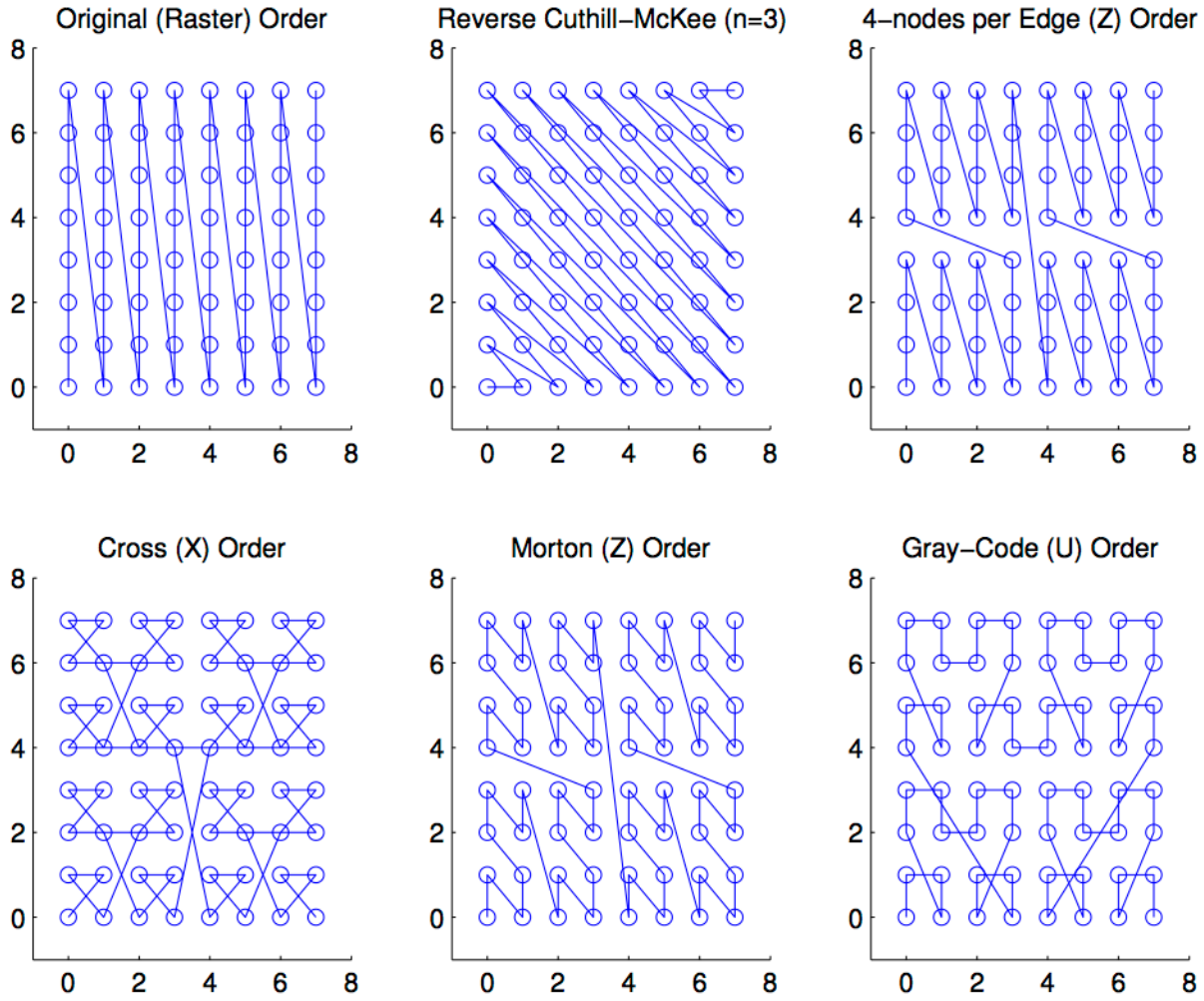


No negative impact on regular grids

Up to 5x speedup for random nodes



Orderings


Original (Raster) Order
Reverse Cuthill–McKee (n=3)
4–nodes per Edge (Z) Order
Cross (X) Order
Morton (Z) Order
Gray–Code (U) Order
0
2
4
6
8

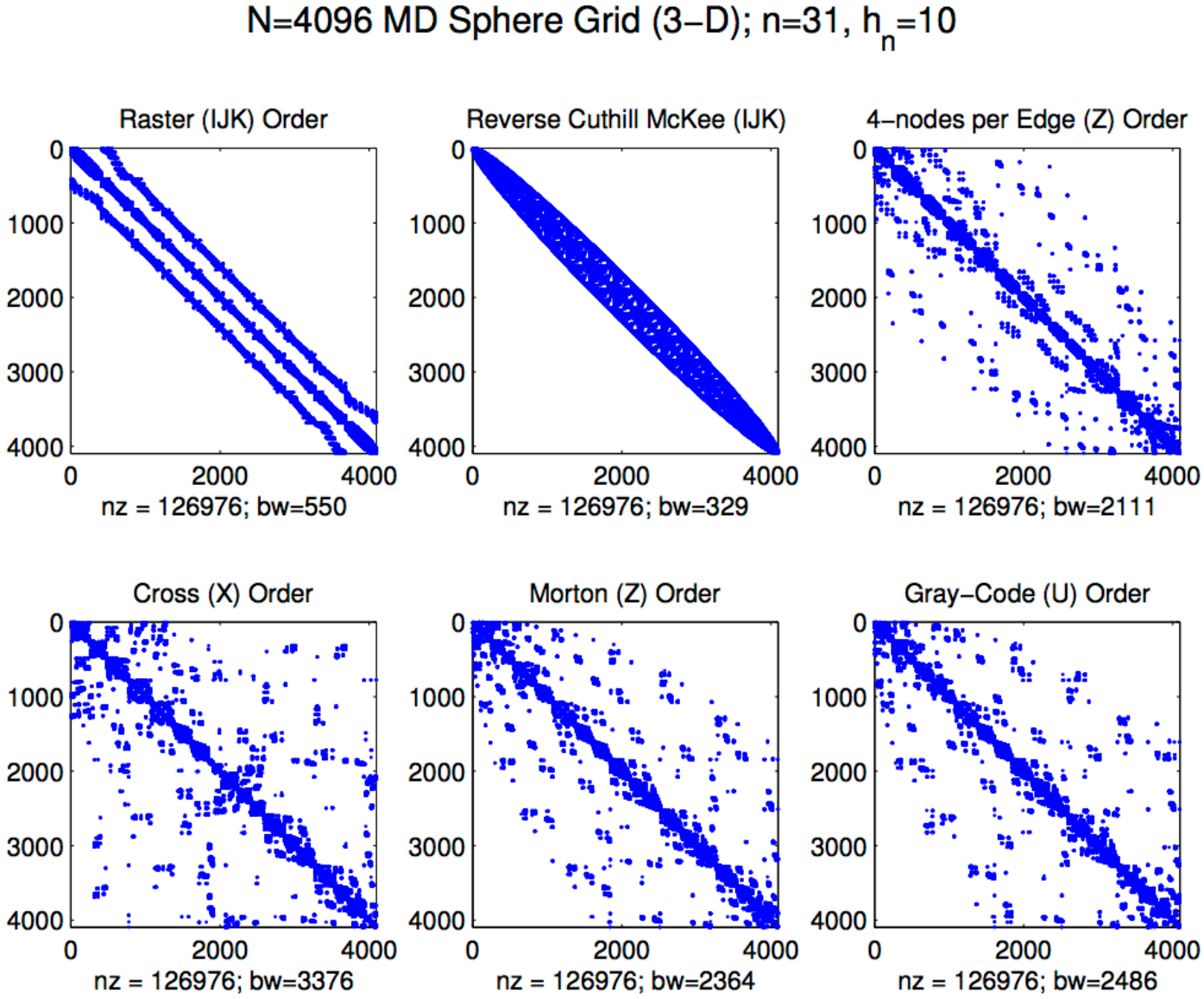
Integer Dilation and Interleaving

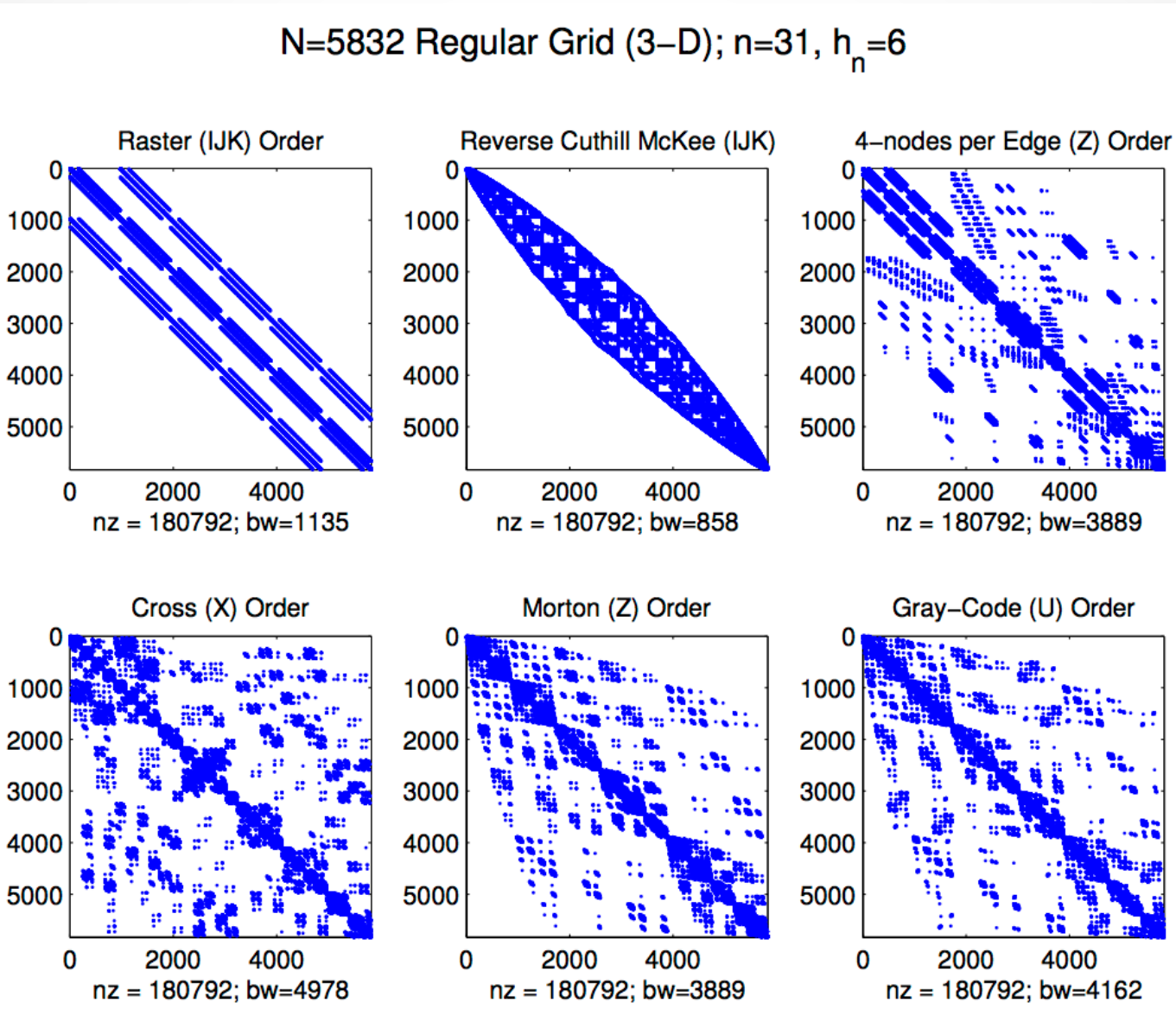
- Example:

$$(c_x, c_y) = (5, 3) = (0101_2, 0011_2)$$
$$(d_x, d_y) = (00010001_2, 00000101_2)$$
$$interleave(c_x, c_y) = (00100111_2) = 39.$$

- Orderings:

Ordering	2-D	3-D
IJK	$(X * h_n) + Y$	$((X * h_n) + Y) * h_n + Z$
Z	$interleave(X, Y)$	$interleave(X, Z, Y)$
U	$interleave(X, X \oplus Y)$	$interleave(X, Z, Z \oplus Y)$
X	$interleave(X \oplus Y, X)$	$interleave(X \oplus Y, Z, Y)$
4-Node Z *	$(d_x \ll 2) \mid d_y$	$((d_x \ll 4) \mid (d_z \ll 2)) \mid d_y$


N=4096 MD Sphere Grid (3–D); n=31, h_n=10
Raster (IJK) Order
Reverse Cuthill McKee (IJK)
4–nodes per Edge (Z) Order
Cross (X) Order
Morton (Z) Order
Gray–Code (U) Order
0
1000
2000
3000
4000
nz = 126976; bw=550
nz = 126976; bw=329
nz = 126976; bw=2111
nz = 126976; bw=3376
nz = 126976; bw=2364
nz = 126976; bw=2486


N=5832 Regular Grid (3–D); n=31, h_n=6
Raster (IJK) Order
nz = 180792; bw=1135
Reverse Cuthill McKee (IJK)
nz = 180792; bw=858
4–nodes per Edge (Z) Order
nz = 180792; bw=3889
Cross (X) Order
nz = 180792; bw=4978
Morton (Z) Order
nz = 180792; bw=3889
Gray–Code (U) Order
nz = 180792; bw=4162
0
1000
2000
3000
4000
5000
0
2000
4000

Fixed Grid Improvements

	IJK	RCM	Z	X	U	4-node Z
Bandwidth	7885	6566	575606	819312	611513	513579
SpMV Speedup	1	1.09	1.06	1.06	1.04	0.94

- RCM works best given random distribution
- Bandwidth is not the key to optimal performance

GPU Comparison

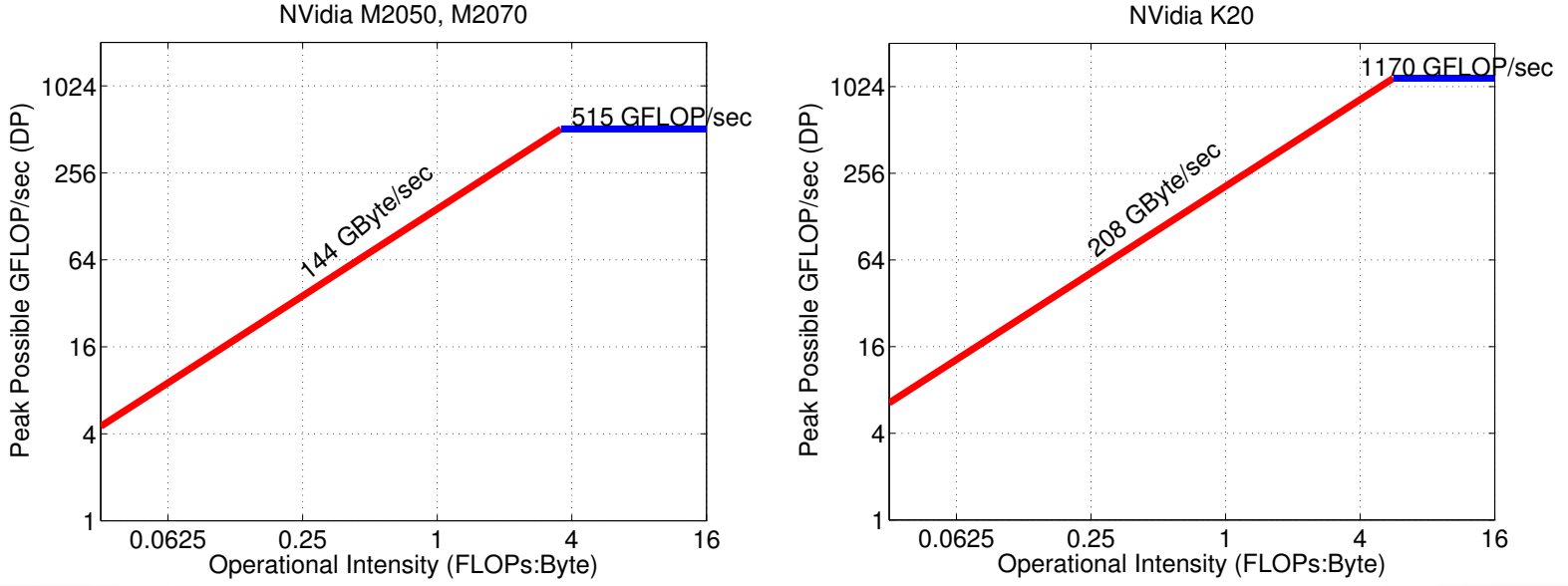
Fermi M2070

- Memory: 6GB
- Bandwidth: 148 GB/s
- DP Peak: 515 GFLOP/sec
- Multiprocessors: 14
- Cores: 448 (32 per Multiprocessor)

Kepler K20

- Memory: 5GB
- Bandwidth: 208 GB/s
- DP Peak: 1.17 TFLOP/s
- Multiprocessors: 13
- Cores: 2496 (192 per Multiprocessor)

Roofline Model (Williams et al. 2009)



(a) Roofline Model for NVidia Fermi class GPUs, M2050 and M2070. SpMV Peak: 36 GFLOP/sec.

(b) Roofline Model for NVidia Kepler K20. SpMV Peak: 52 GFLOP/sec.

- The roofline model predicts MAXIMUM possible performance on the hardware.
- RBF-FD SpMV is only 1/8 FLOPs:bytes (double precision)

Custom RK4

- RK4 time-scheme:

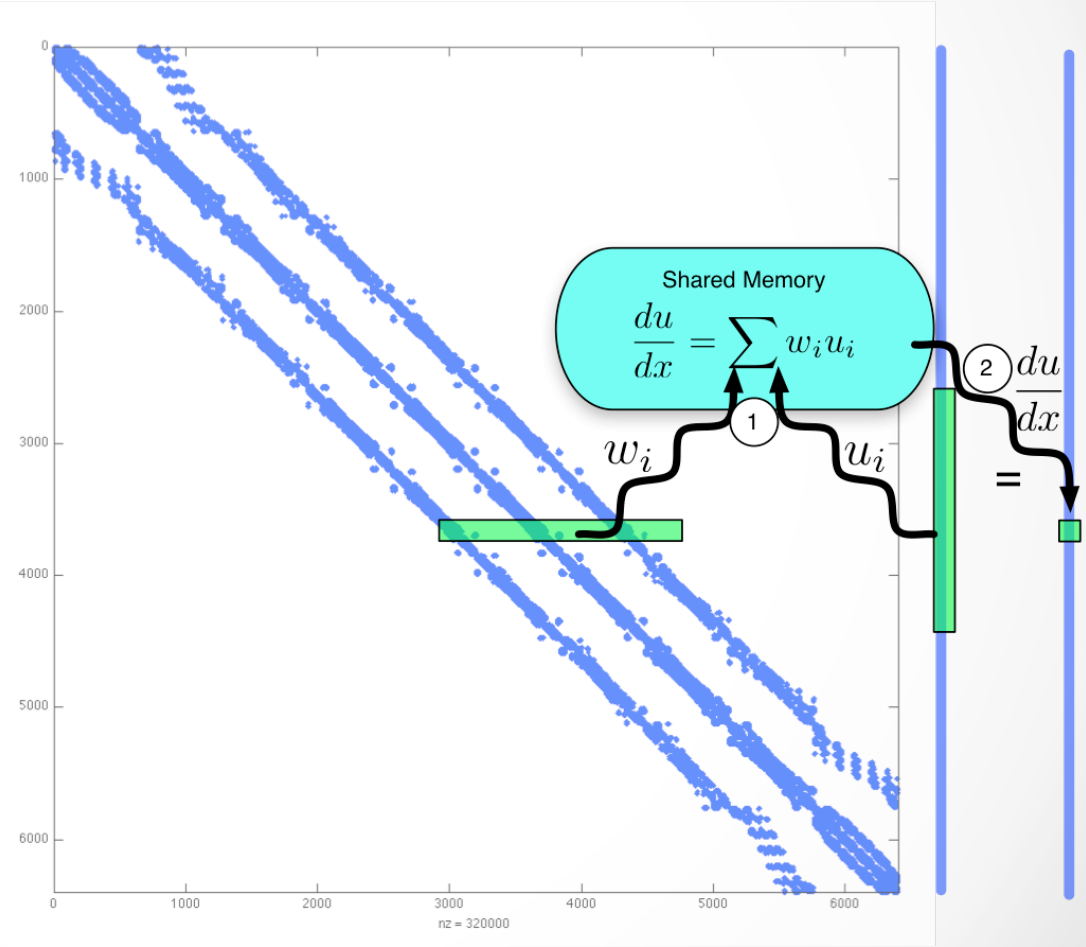
$$
\begin{aligned}
\mathbf{k}_1 &= \Delta t f(t_n, \mathbf{u}_n) \\
\mathbf{k}_2 &= \Delta t f(t_n + \frac{1}{2}\Delta t, \mathbf{u}_n + \frac{1}{2}\mathbf{k}_1) \\
\mathbf{k}_3 &= \Delta t f(t_n + \frac{1}{2}\Delta t, \mathbf{u}_n + \frac{1}{2}\mathbf{k}_2) \\
\mathbf{k}_4 &= \Delta t f(t_n + \Delta t, \mathbf{u}_n + \mathbf{k}_3) \\
\mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4),
\end{aligned}
$$

- Reusable kernels called as:

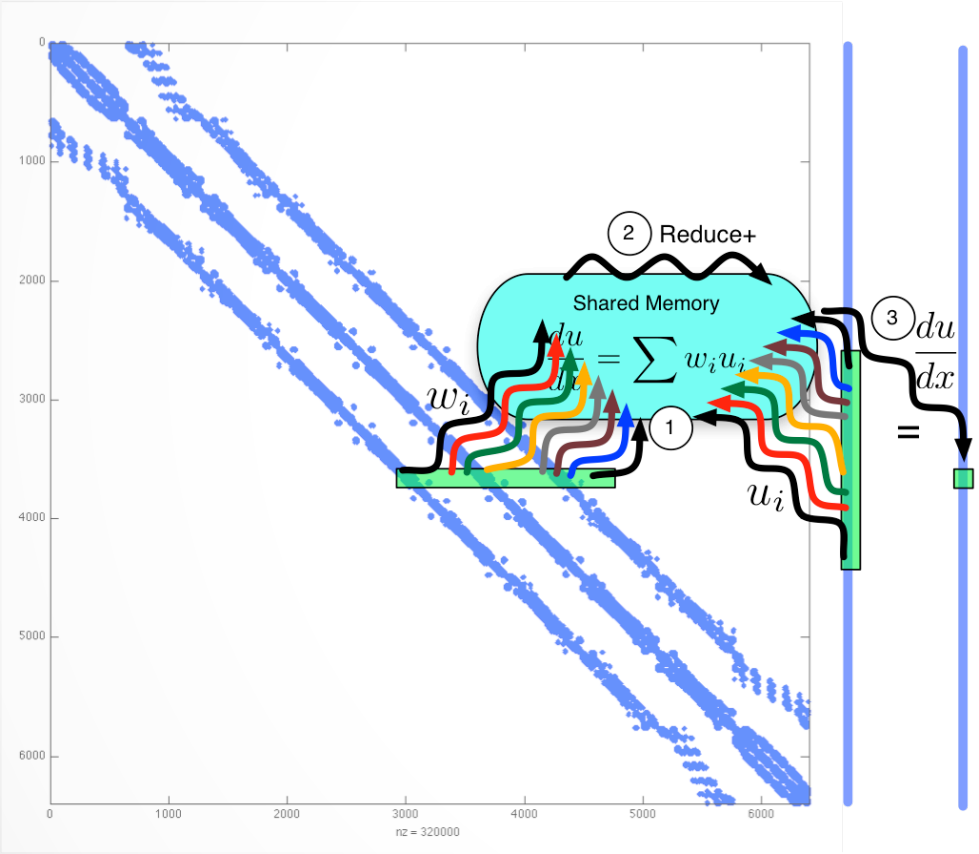
1. $\mathbf{k}_i = \Delta t f(t_n + \alpha_i \Delta t, \mathbf{u}_n + \alpha_i \mathbf{k}_{i-1}),$
2. $\mathbf{u}_n + \alpha_{i+1}\mathbf{k}_i$

One Thread per Stencil

- Pros:
  - All stencils have the same number of operations (perfect concurrency)
- Cons:
  - Threads may access disparate areas of memory
  - Large stencils

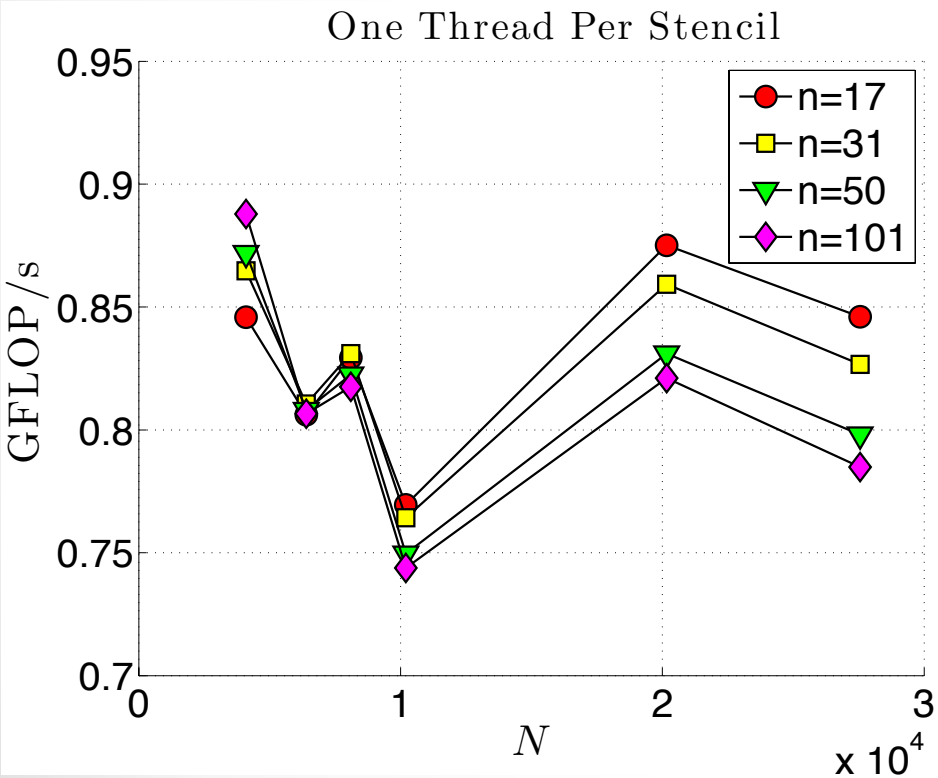


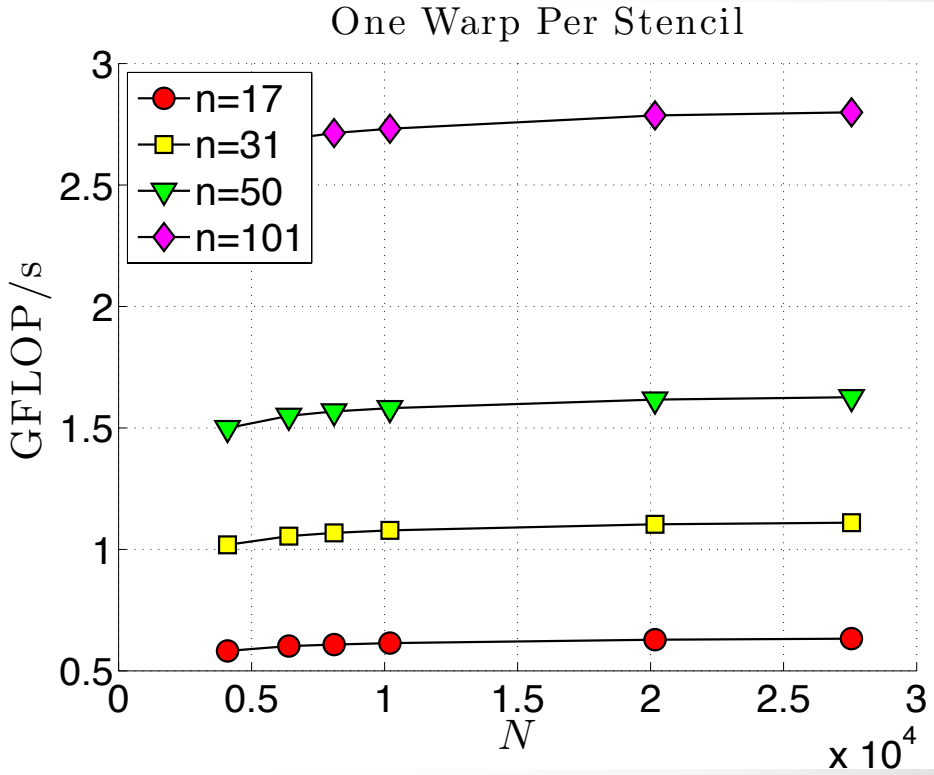
One Warp per Stencil



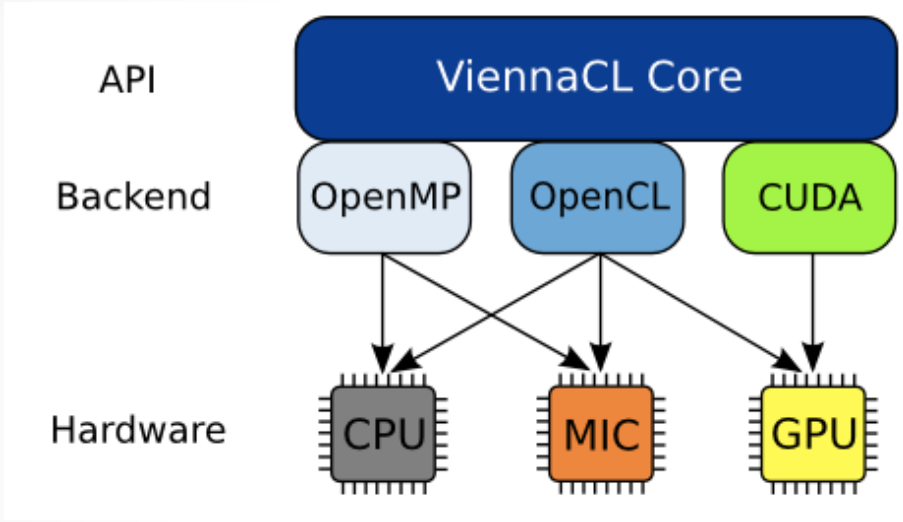
- Pros:
  - GPU operates in warps, so memory loads are shared
- Cons:
  - Only 32 threads queued per stencil on single multiprocessor

Custom Kernel Performance (Keeneland M2070)





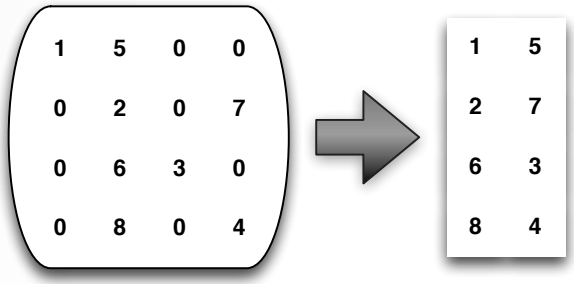
ViennaCL



Recently integrated into PETSc!

- Started assisting on the ViennaCL project (Summer 2010)
  - Extensions for rectangular matrices, ILU0, distributed SpMV + GMRES
  - Integrated into RBF-FD code
  - Benefit from existing optimizations for SpMV (formats)

Memory Compression (Sparse Storage)



COO

O(3*n*N)

Value	1	5	2	7	6	3	8	4
Row	0	0	1	1	2	2	3	3
Col	0	1	1	3	1	2	1	3

CSR

O((2*n+1)*N)

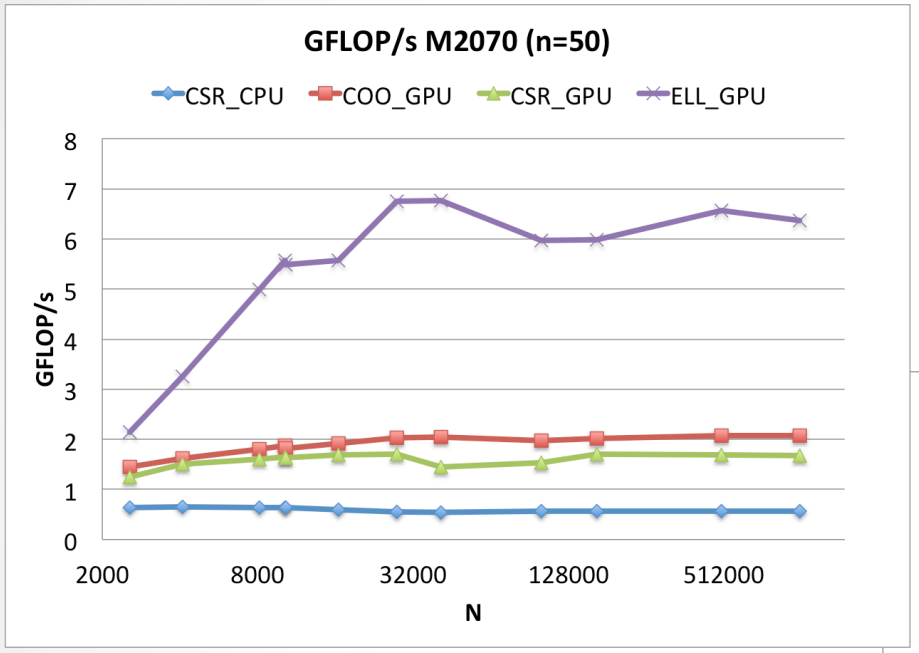
Value	1	5	2	7	6	3	8	4
Row Ptr	0	2	4	6				
Col	0	1	1	3	1	2	1	3

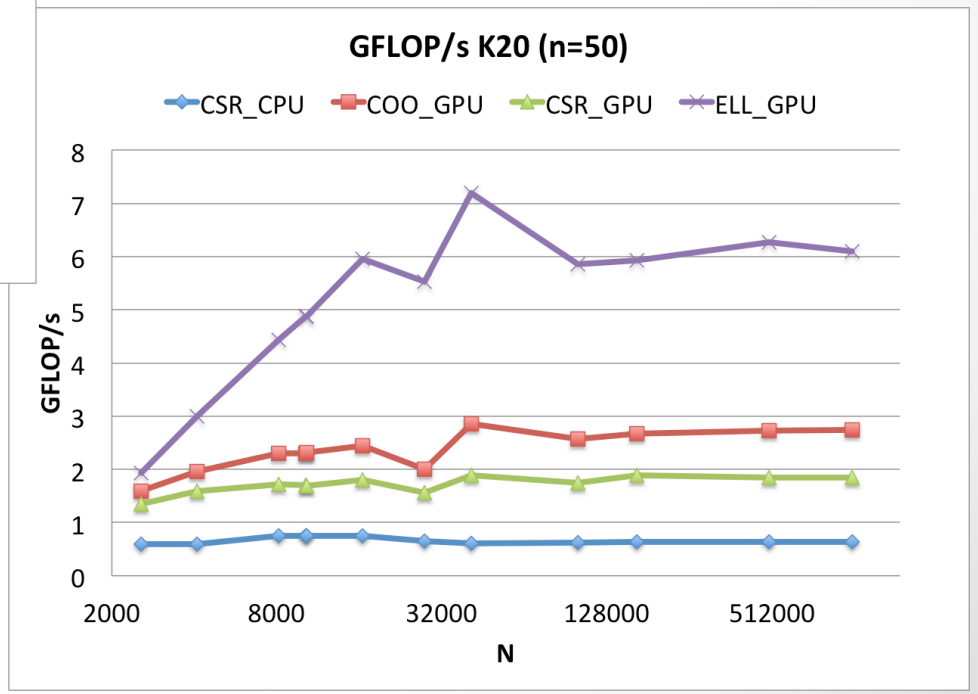
ELL

Value	1	5	2	7	6	3	8	4
Col	0	1	1	3	1	2	1	3

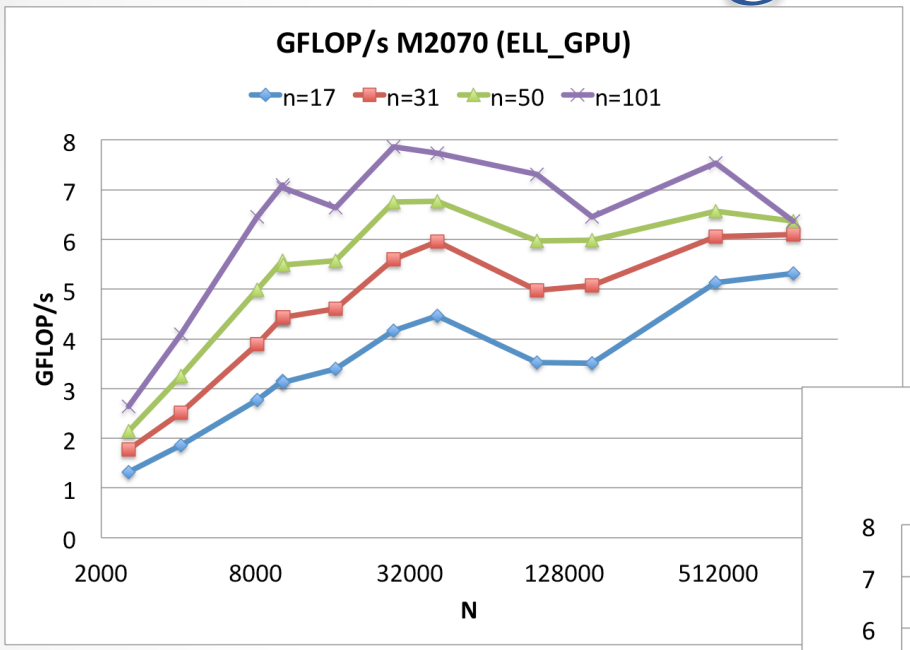
O(2*n*N)

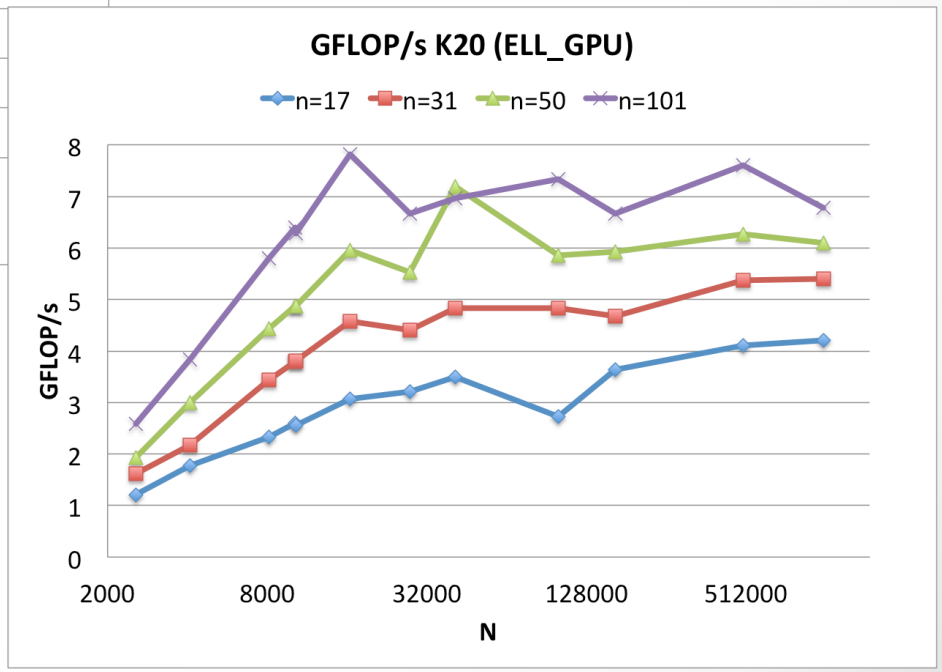
Single GPU


GFLOP/s M2070 (n=50)
CSR_CPU
COO_GPU
CSR_GPU
ELL_GPU
GFLOP/s
0
1
2
3
4
5
6
7
8
2000
8000
32000
128000
512000
N


GFLOP/s K20 (n=50)
CSR_CPU
COO_GPU
CSR_GPU
ELL_GPU
GFLOP/s
0
1
2
3
4
5
6
7
8
2000
8000
32000
128000
512000
N

Peak ELL GFLOP/sec on Single GPU



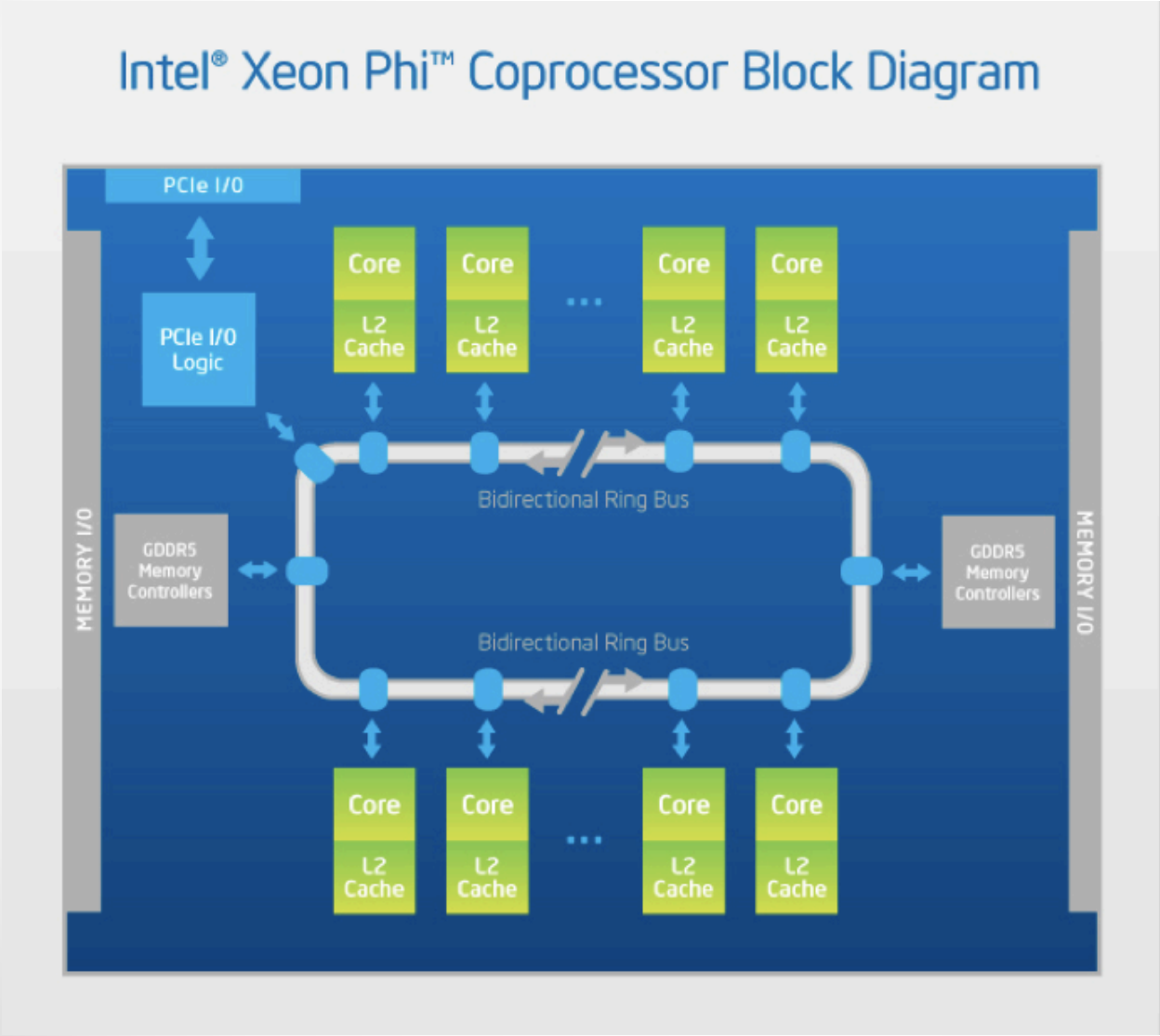


Fermi and Kepler perform the same.

Future Work for GPU SpMV Performance

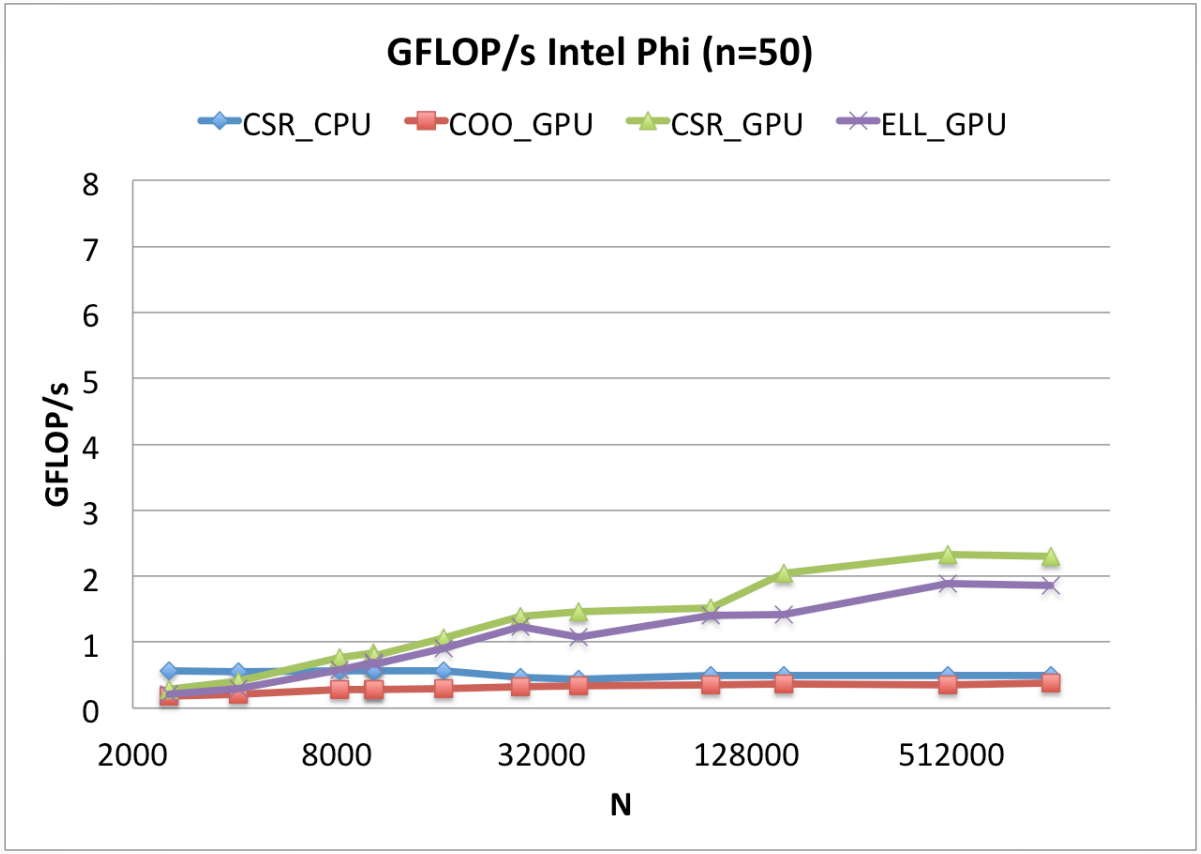
- Other Formats
  - BELL, SELL, SBELL, etc.
- Fewer memory loads
  - Ordering
  - Register packing (non-zero condensation)
- More operations (Erlebacher et al. 2013)

Performance Portability? Intel Phi



2013 OpenCL Beta Driver

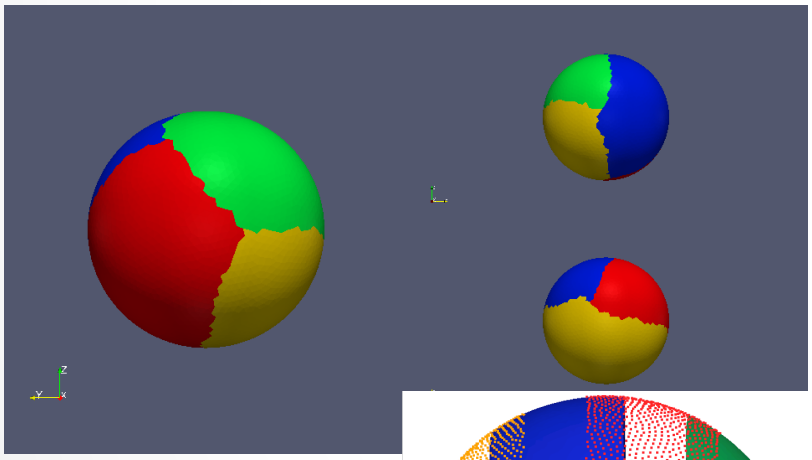
Preliminary Data: Intel Phi

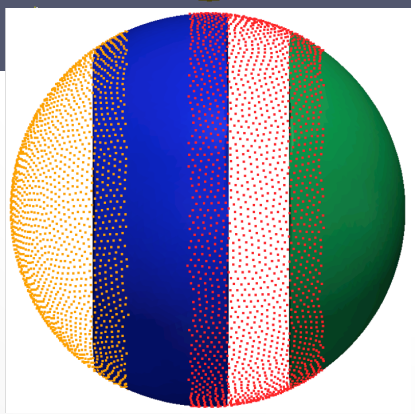


Domain Decomposition

$$\begin{aligned} \mathcal{L}u_1^{n+1} &= f \quad \text{in } \Omega_1, \\ \mathcal{B}u_1^{n+1} &= g \quad \text{on } \partial\Omega_1, \\ u_1^{n+1} &= u_2^n \quad \text{on } \Gamma_{12}, \end{aligned}$$

$$\begin{aligned} \mathcal{L}u_2^{n+1} &= f \quad \text{in } \Omega_2 \\ \mathcal{B}u_2^{n+1} &= g \quad \text{on } \partial\Omega_2, \\ u_2^{n+1} &= u_1^n \quad \text{on } \Gamma_{21}, \end{aligned}$$





- Restricted Additive Schwarz Method
- Graph partitioning with METIS for load balancing (Karypis and Kumar, 1999)
  - Requires symmetric adjacency matrix: $A + A^T$

Load Balancing with METIS

- Load balancing impacts parallel gains:

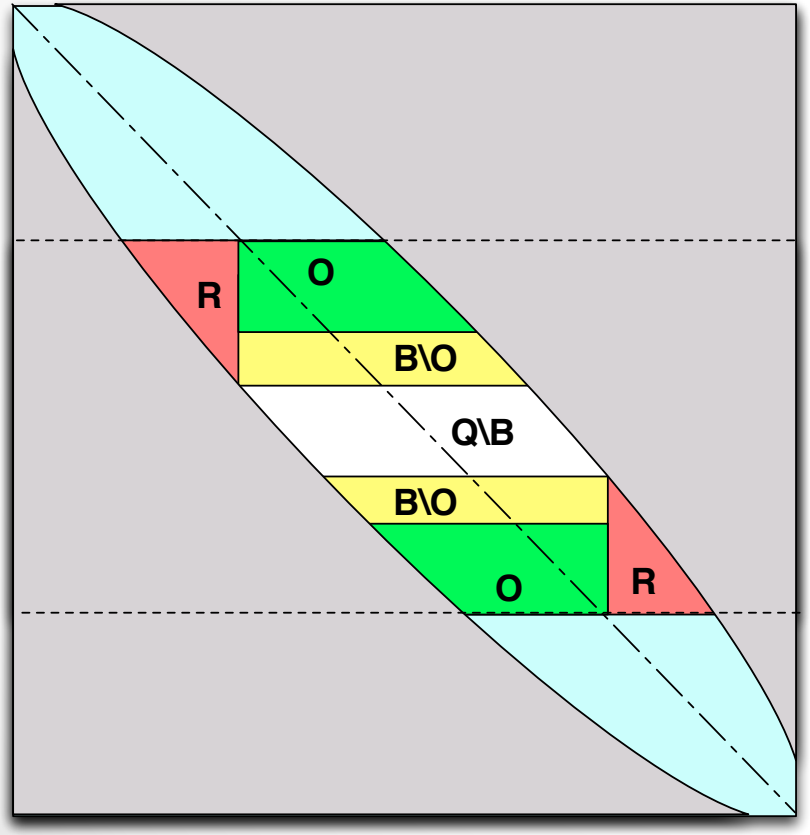
$$S_p = \frac{(p-1)W_1 + W_2}{W_2} = 1 + (p-1)\frac{W_1}{W_2}.$$

p	Partitioning	Total $(\frac{\min N_p}{\max N_p})$	Interior $(\frac{\min N_i}{\max N_i})$	Ghost Nodes $(\frac{\min N_r}{\max N_r})$
4	Linear	0.863	0.997	0.482
	METIS	0.986	0.971	0.913
16	Linear	0.551	0.980	0.264
	METIS	0.925	0.942	0.864

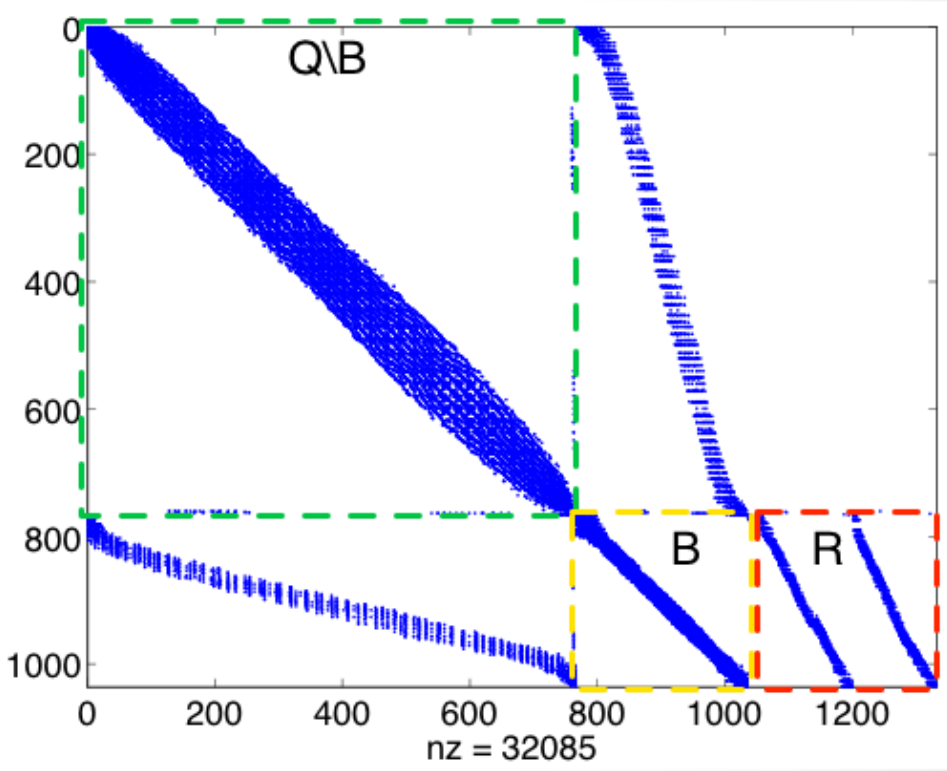
Ghost nodes are the killer!

Node Ordering and Index Sets

$$\mathcal{G} = \{\mathcal{Q}\backslash\mathcal{B} \quad \mathcal{B}\backslash\mathcal{O} \quad \mathcal{O} \quad \mathcal{R}\},$$

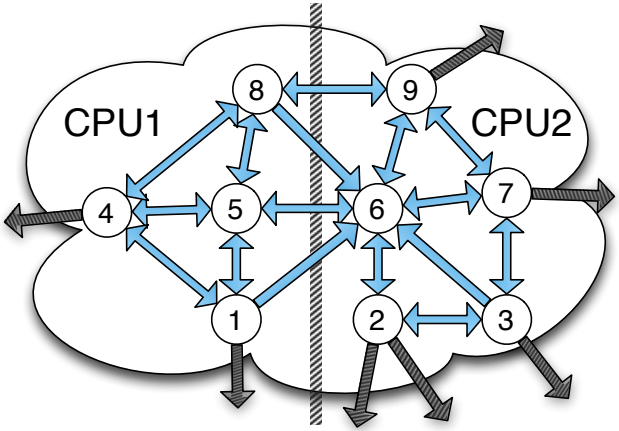


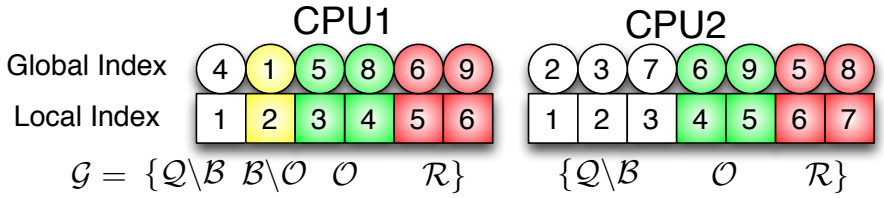
Global Index Space

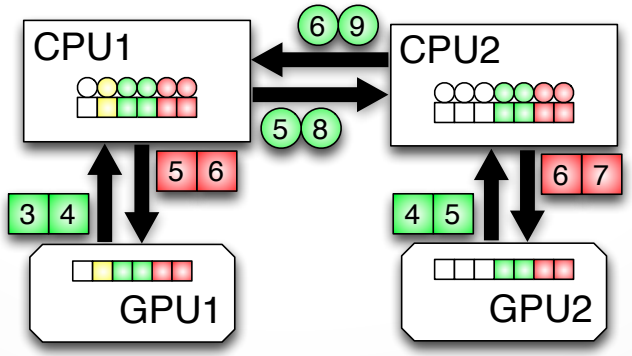


Local Index Space

Data Movement for Multi-GPU



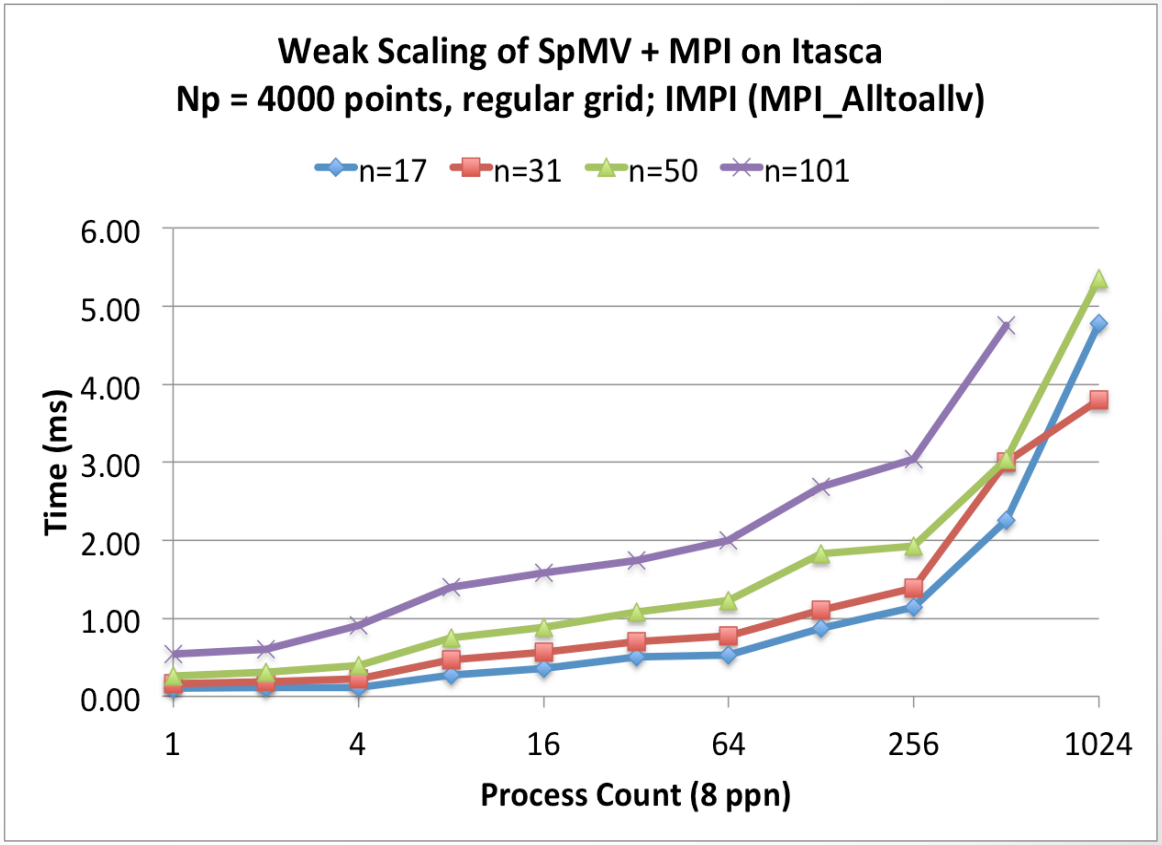




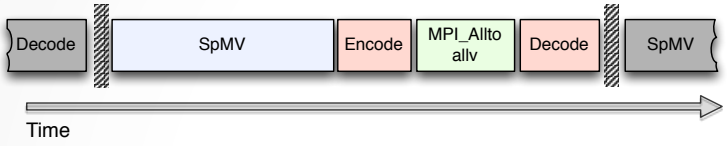
Multi-CPU Weak Scaling

- Tuned on Itasca (UMN)
  - 1,134 HP ProLiant blade servers
  - 2x quad-core Xeon 5660 ("Nehalem EP") processors per node
- N=160^3 nodes
- Impact on performance for fixed problem size per processor

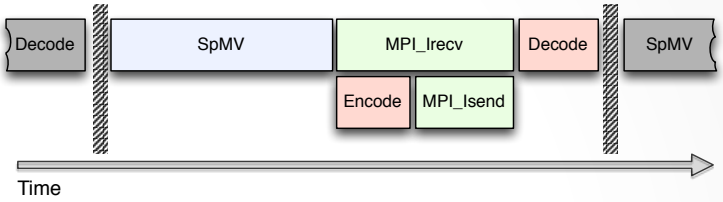
BEFORE (MPI_Alltoallv)



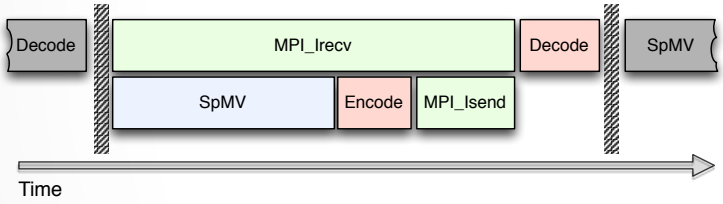
MPI Stages



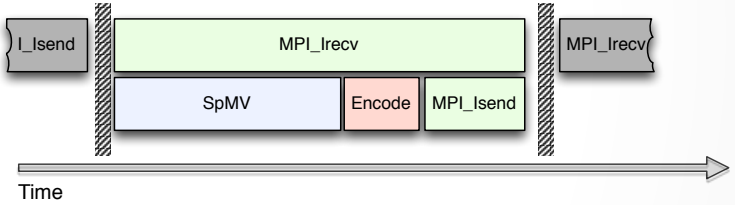
(a) MPI_Alltoallv



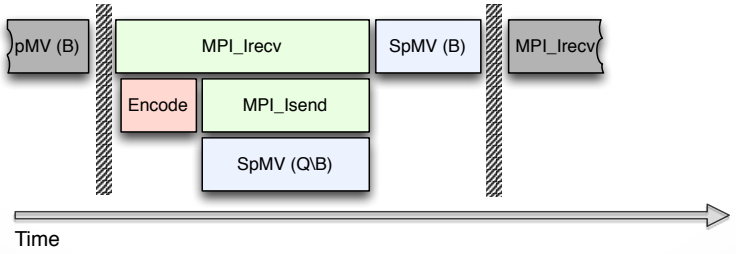
(b) MPI_Isend/MPI_Irecv



(c) Post an MPI_Irecv before SpMV ("Pre-Recv")



(d) No Decode

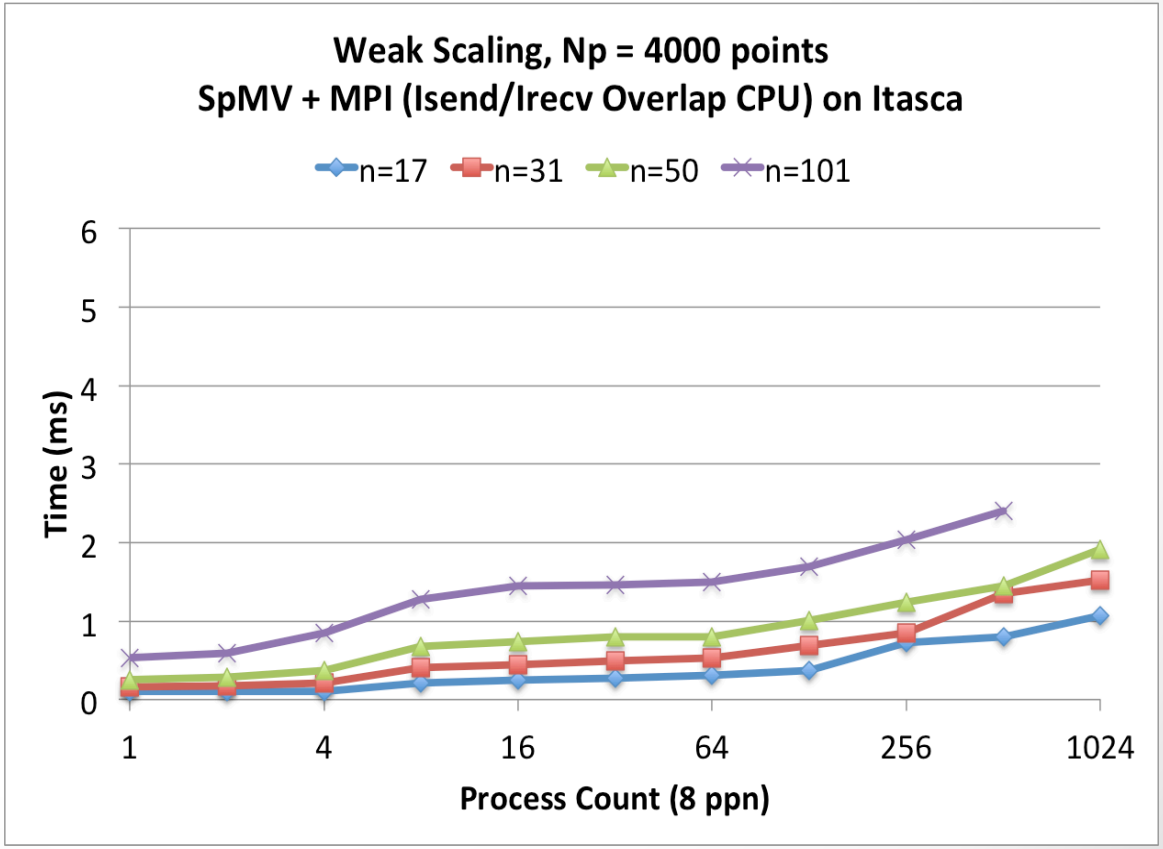


(e) Overlapping SpMV and Communication

Multi-CPU Weak Scaling

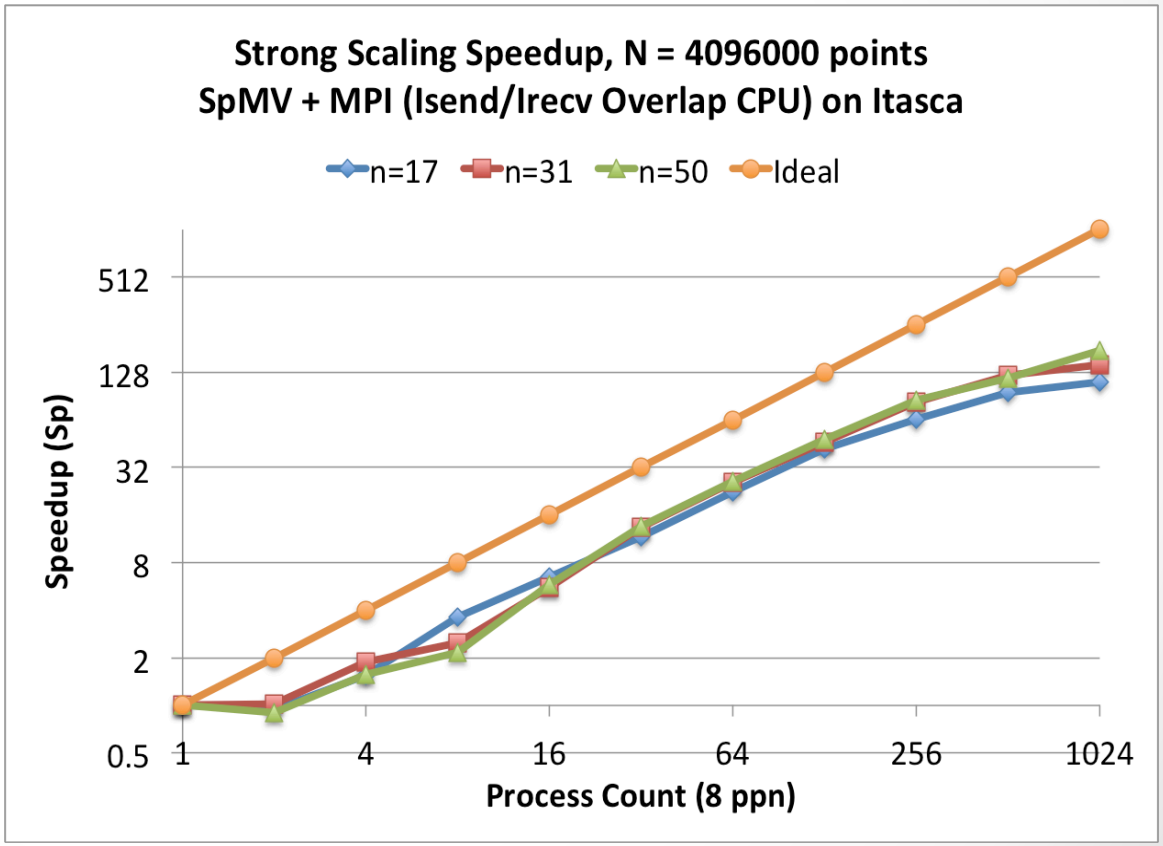
- 2x-5x improvement at p=1024 processors
- Larger N_p per processor would hide growth
- METIS load balancing fails for p=1024, n=17:

$$\frac{\min N_r}{\max N_r} = 0.39$$

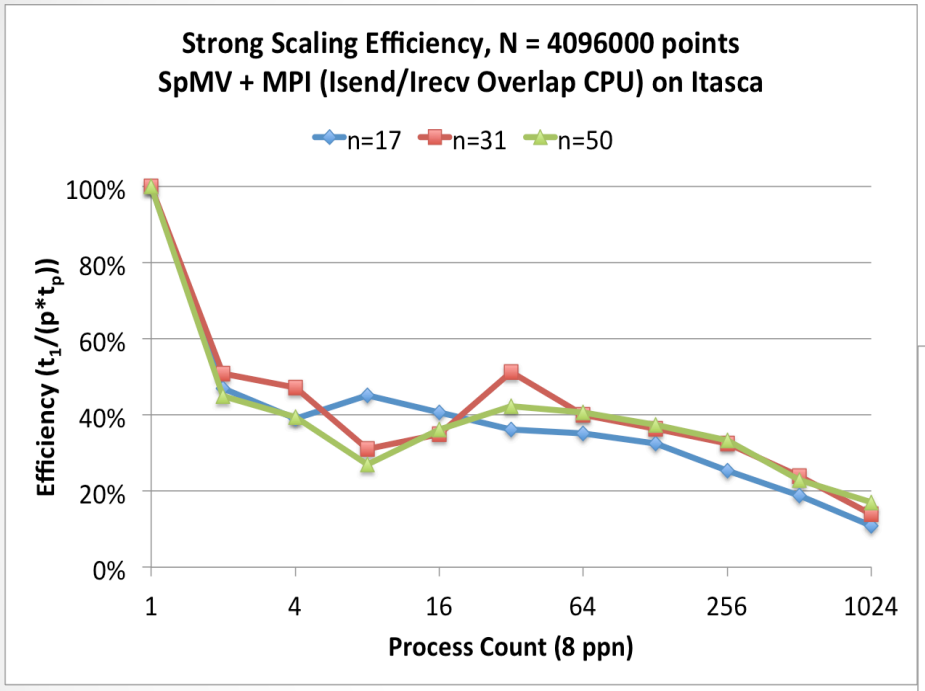


Multi-CPU Strong Scaling

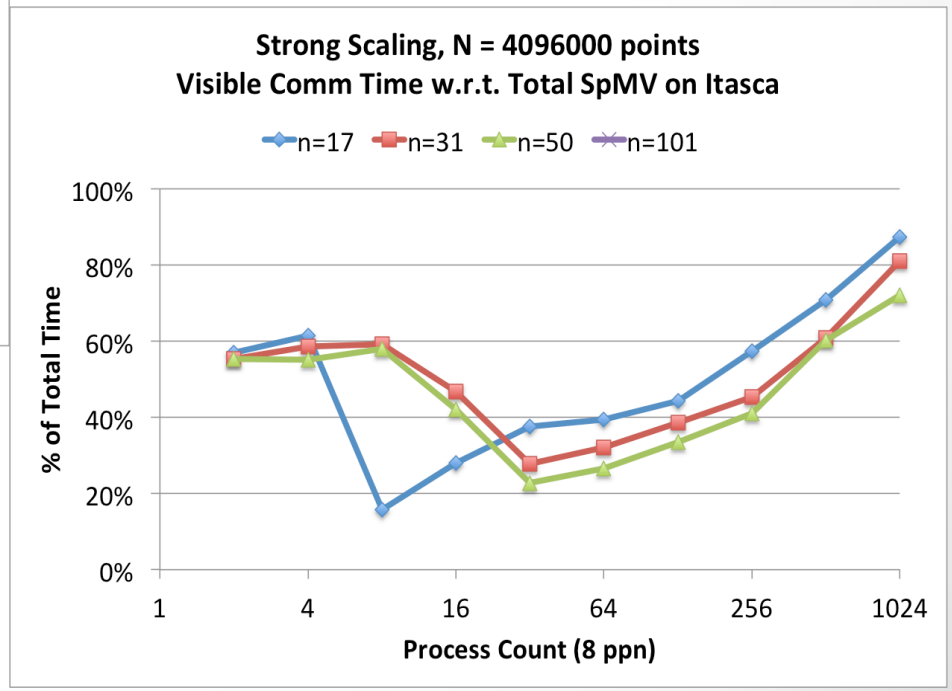
- Fixed global problem size with decreasing workloads
- MPI_Alltoallv is 2x faster when operating on one node. Slower otherwise.
- Near linear (ideal) scaling.



Strong Scaling Efficiency and Communication Overlap



Efficiency: $E_p = {}^{t_1}/_{(p*t_p)} * 100\%.$

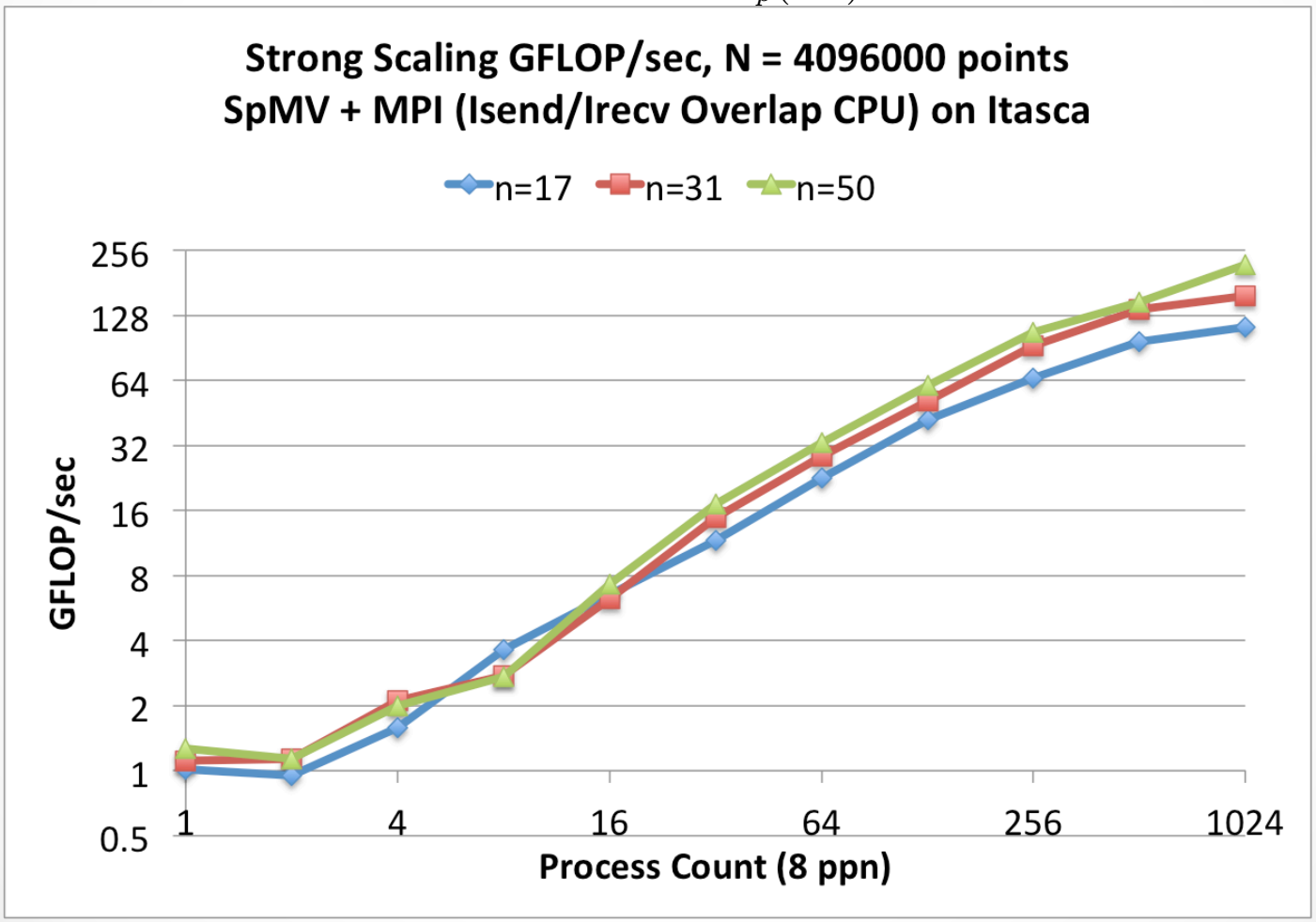


Visible Communication Time:

$$Vis = {}^{(t_{Comm} - t_{SpMV1})}/_{t_{Total}} * 100\%$$

Achieved GFLOP/sec on Itasca

$$GFLOP/sec = ({}^{N*n*2}/_{t_p(sec)}) * 10^{-9})$$

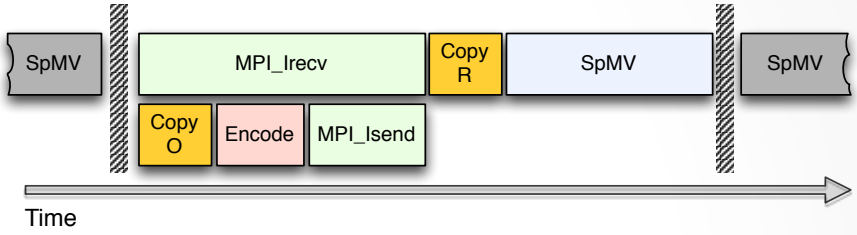


Multi-CPU Summary

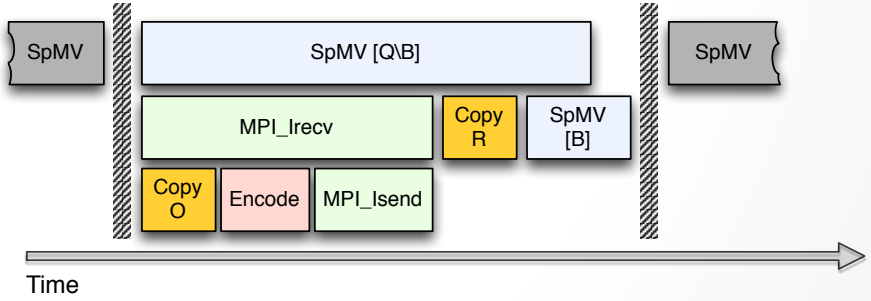
- Strong scaling
  - Scales well aside from effects of contention IB network and insufficient problem size for large p.
- Weak scaling
  - Code scales well for small stencil sizes.
  - Larger N_p necessary to hide increasing cost of communication.
- Future work:
  - Align messages properly for best performance
  - Possible FMM approach to limit communication between subddomains (Yokota et al. 2010)

Multi-GPU Communication

- GPU adds transfers between host←→device
- Run two OpenCL queues
  - One for SpMV[Q\B]
  - One for host←→device transfers and SpMV[B]
  - Allows overlap of communication and

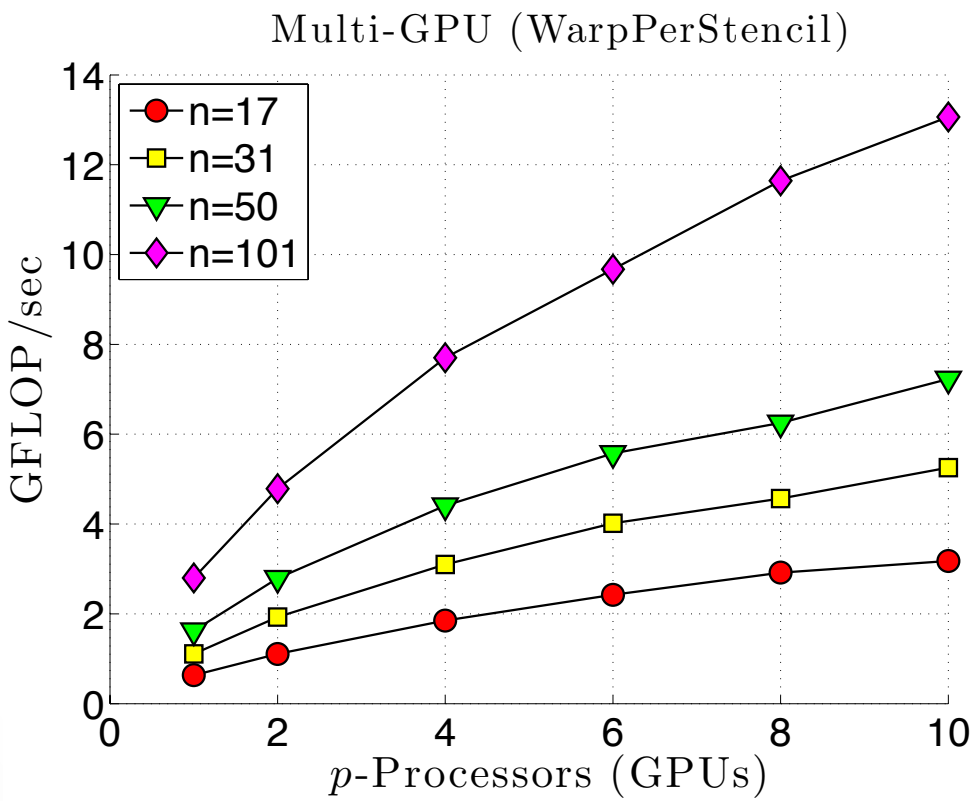


MPI_Alltoallv (No Overlap)



Overlapping Collective

Multi-GPU Strong Scaling Cosine Bell (Keeneland M2070)



CSR Matrix with MPI_Alltoallv

Assumes $(24Nn + 33N)$ FLOPs per RK4 step

Multi-GPU Strong Scaling (Cascade M2070s)

		Observed GFLOP/sec (Speedup over Non-overlapped)			
Compute Nodes	PPN	n=17	n=31	n=50	n=101
1	1	8.5 (1.0x)	8.4 (1.0x)	9.0 (1.0x)	–
2	1	13.8 (2.3x)	13.0 (1.9x)	13.1 (2.0x)	13.5 (1.8x)
4	1	13.1 (1.9x)	25.1 (2.8x)	24.6 (2.4x)	25.2 (1.9x)
8	1	24.5 (2.0x)	33.2 (2.3x)	41.2 (3.4x)	53.6 (2.3x)
1	2	11.3 (3.4x)	12.2 (3.0x)	12.1 (3.2x)	12.7 (3.0x)
2	2	13.0 (3.0x)	22.9 (4.2x)	23.1 (3.6x)	24.5 (3.0x)
4	2	25.1 (2.3x)	37.8 (4.1x)	50.1 (5.0x)	53.5 (3.8x)
8	2	35.8 (2.2x)	38.3 (2.5x)	59.6 (2.7x)	87.6 (3.7x)
1	4	14.1 (**4.3x**)	22.5 (**5.1x**)	24.4 (**5.4x**)	24.4 (**4.4x**)
2	4	19.6 (2.4x)	32.0 (4.2x)	37.2 (3.9x)	50.8 (4.6x)
4	4	27.5 (2.2x)	38.6 (3.0x)	57.0 (3.0x)	81.3 (4.5x)
8	4	50.9 (2.9x)	61.6 (2.2x)	88.8 (3.2x)	130.8 (3.5x)

Known issue on Cascade: a shared I/O hub between 4 GPUs serializes data transfers. Overlapping bypasses the problem.

Multi-GPU Summary

- Bottleneck on multi-GPUs of same node is avoided with overlapping queues.

Conclusions

- Presented the first distributed RBF-FD implementation
  - Tuned MPI for decent scaling, still more optimizations possible
  - Ghost nodes are the source workload imbalance in subdomains
  - First investigation for RBF-FD on GPU and Multi-GPU
- ELL format is appropriate for RBF-FD with fixed stencil size and achieves up to 8 GFLOP/sec (double precision) for a single SpMV (~43% of the roofline estimate for SpMV).
- Overlapping communication and computation is essential
  - Bottleneck on I/O-Hub connecting multiple-GPUs on same node is avoided with overlapping queues.

Conclusions II

- Hyperviscosity parameters stabilize RBF-FD solutions to hyperbolic equations.
  - Convergence is demonstrated as function of N.
- Fixed-grid method can be 2.5x faster than k-D Tree (one time cost) and can result in 5x faster SpMV for quasi-random node distributions.
  - Space-filling curves and Reverse Cuthill-McKee do not make significant difference so we proceed with IJK (Raster) Ordering.

Community Outreach

- Erlebacher, G., Saule, E., Flyer, N., and Bollig, E. Sparse Matrix Vector Multiplication with Multiple vectors and Multiple Matrices on the MIC Architecture. Submitted to IEEE Parallel & Distributed Processing Symposium (IPDPS), 2014, October 2013
- Bollig, E. F., Flyer, N., and Erlebacher, G. Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs. *Journal of Computational Physics 231*, 21 (2012), 7133 – 7151
- Bollig, E. F. Multi-GPU Solutions of Hyperbolic and Elliptic PDEs with RBF-FD. Contributed Talk. SIAMSEAS Annual Meeting. Huntsville, AL, March 2012
- Bollig, E. F. RBF-generated Finite Differences for Elliptic PDEs on Multiple GPUs. Poster. FSU Dept. of Sci. Comp. Computational Xpo. Tallahassee, FL, March 2013. Also presented at: the Minnesota Supercomputing Institute Research Exhibition, Minneapolis, MN, April 2013
- Bollig, E. F. Parallel Algorithms for RBF-FD Solutions to PDEs on the Sphere. Poster. FSU Dept. of Sci. Comp. Computational Xpo. Tallahassee, FL, March 2012
- Bollig, E. F. A Multi-CPU/GPU implementation of RBF-generated finite differences for PDEs on a Sphere. Poster. American Geophys. Union. San Francisco, CA, December 2011
- Bollig, E. F. Accelerating RBF-FD in Parallel Environments. Student Presentation. NSF CBMS Workshop: Radial Basis Functions Mathematical Developments and Applications. Darmouth, MA, June 2011

Survey of Technologies Used

- Matlab
- Mathematica
- OpenCL
- CUDA
- ParaView
- NCAR Command Language (NCL) and NCAR Graphics

- C++
  - Armadillo (http://arma.sf.net)
  - ViennaCL (http://viennacl.sf.net)
  - CUSP (https://code.google.com/p/cusp-library/)
  - BOOST::uBLAS (http://www.boost.org/)
  - METIS (http://cs.umn.edu/~metis/)
  - MPI (http://mpi-forum.org)

Current and Future Work

- Return to CUDA
  - CUDA v5.5 features supports MPI with GPUDirect®
- SpMV Improvements:
  - Auto-tuned kernels with clSpMV (Su and Keutzer 2012)
    - 50+ GFLOP/sec when running in single precision
  - New formats:
    - BELL, SELL, SBELL (Su and Keutzer 2012)
    - ELLPACK-R, pJDS, etc. (Kreutzer et al. 2012)
  - Applications with Multi-Matrix SpMM (Erlebacher et al. 2013)

Current and Future Work II

- Multiple subdomains per GPU (to saturate Fermi and Kepler cards)
- Improve communication by properly aligning/ padding MPI messages
- Consider FMM approach for RBF-FD

Continued and Future Work III

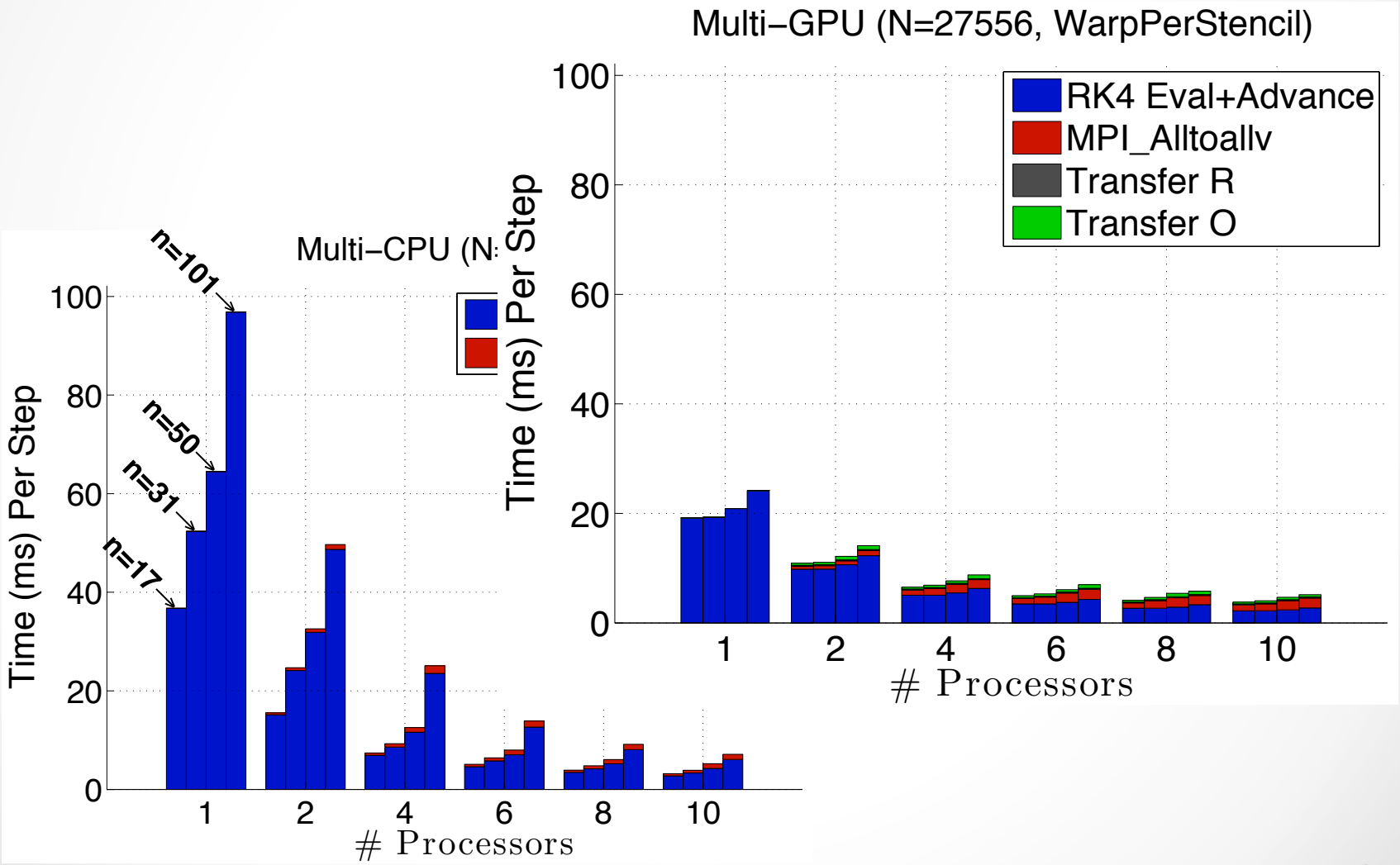
- Intel Phi vs NVidia Kepler
  - Latest hardware from Nvidia? K40*
- Multi-GPU preconditioned GMRES solutions of RBF-FD for Stokes flow
  - Core operation of GMRES: SpMV
  - Current work is incomplete
- $\epsilon \to 0$ with RBF-GA

Questions?

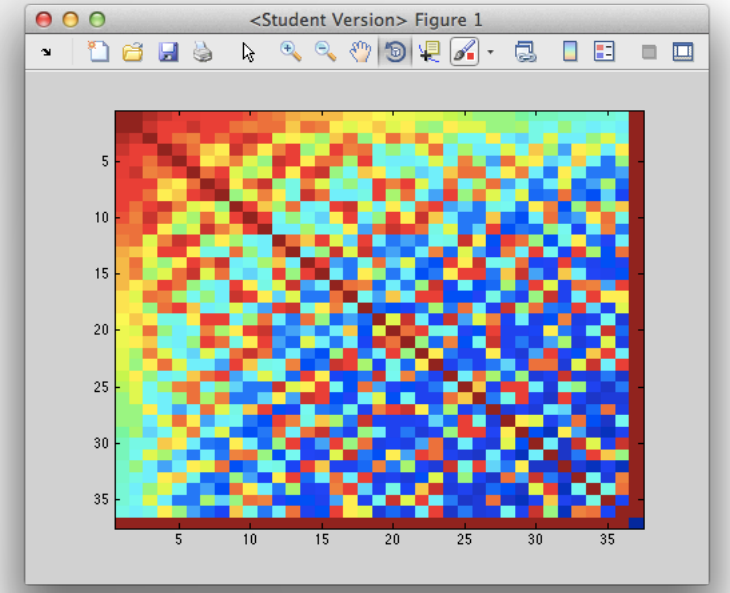
Remember: this is research. If we had all the answers we wouldn't be doing it.

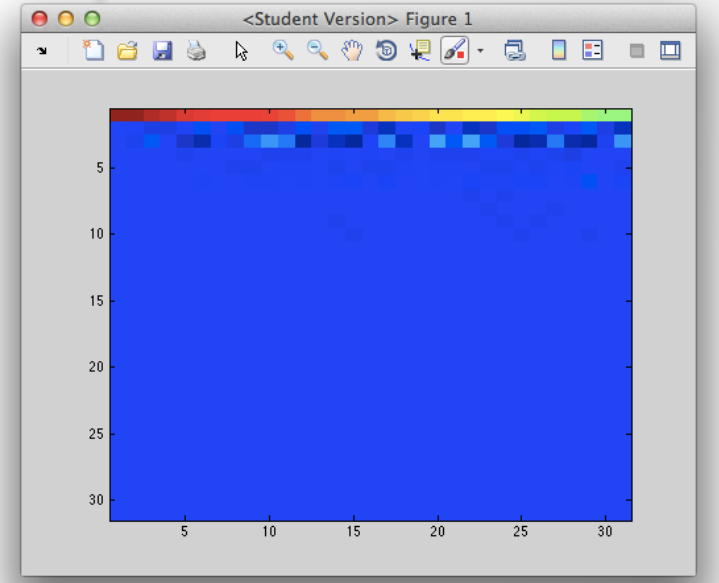
Extras

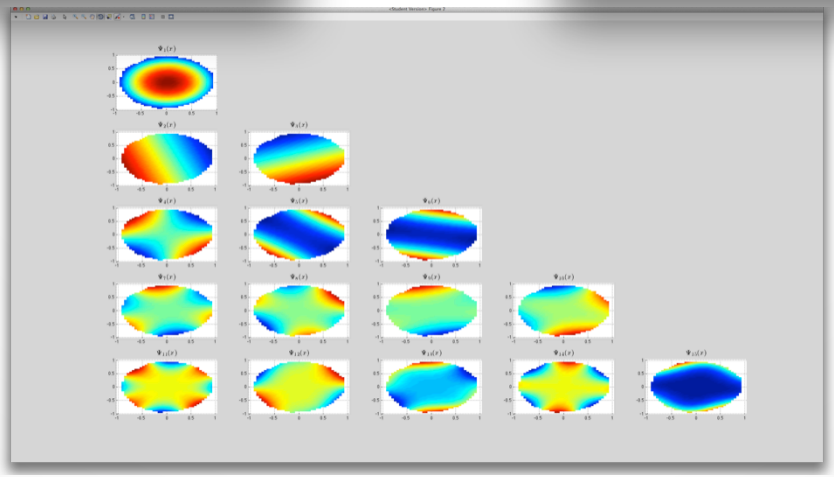
This is science, not politics! Don't you know the difference? A politician starts in the middle with "$= b$" and draws conclusions based on smoke and mirrors. In science you have no choice but to start from the beginning with something like "$Ax = b$". (Gordon Erlebacher, c. 2010).


Multi−GPU (N=27556, WarpPerStencil)
RK4 Eval+Advance
MPI_Alltoallv
Transfer R
Transfer O
Time (ms) Per Step
Processors
Multi−CPU (N
n=101
n=50
n=31
n=17

RBF-GA (Preliminary Results)







Keeneland Cosine Custom Kernel GFLOP/sec