BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

HPGMG BoF

High Performance Geometric Multigrid
Birds of a Feather

Introduction	*Mark Adams (LBNL)*
4th order HPGMG-FV	*Samuel Williams (LBNL)*
HPC Benchmarking	*Vladimir Marjanovic (HLRS)*
GPU Implementation of HPGMG-FV	*Simon Layton (NVIDIA)*
Questions and Discussion	*all*

4^{th} Order HPGMG-FV Implementation

Samuel Williams
SWWilliams@lbl.gov
Lawrence Berkeley National Laboratory

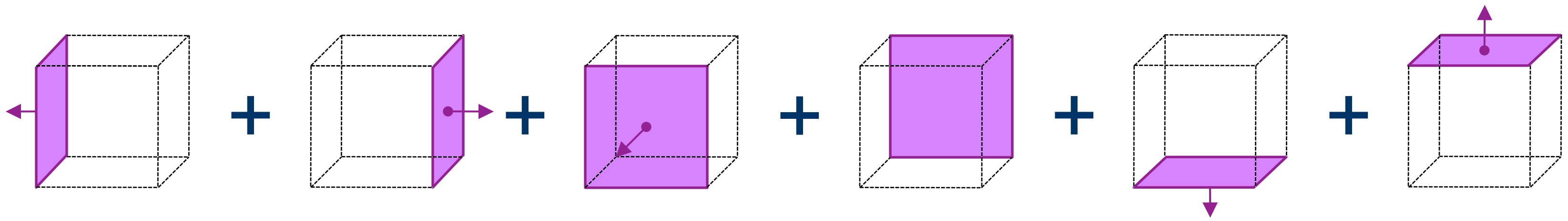
Observations of the 2nd order HPGMG-FV

- ‘Order’ describes the relationship between grid spacing and error.
- <=v0.2 implemented a 2nd order Finite Volume method.
- We found this method did not sufficiently stress HPC systems…
  - The 7pt operator and interpolation routines were heavily memory-bound on most machines (**STREAM-proxy on a single node**)
  - The Chebyshev smoother **did not stress most compilers**
  - On DDR-based architectures, the memory capacity:bandwidth balance allowed for huge problem sizes that **hid network communication**.
  - The simple 1st order boundary conditions could easily be fused with the operator and thus **sidestepped the desire to benchmark irregular parallelism and memory access**.

4^{th} order $\langle \nabla \cdot \beta \nabla u \rangle$

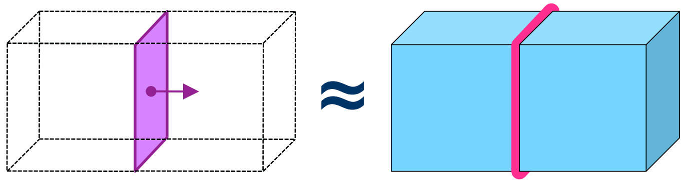
- Finite volume method expresses the average value of an operator over a cell’s volume ($\langle \nabla \cdot \beta \nabla u \rangle$) as an integral over the cell’s surface ($\langle \beta \nabla u \cdot N \rangle$).
- For 3D structured grids, each h^3 cell has 6 faces and we must calculate this term on each face….

$Lu = \langle \nabla \cdot \beta \nabla u \rangle =$



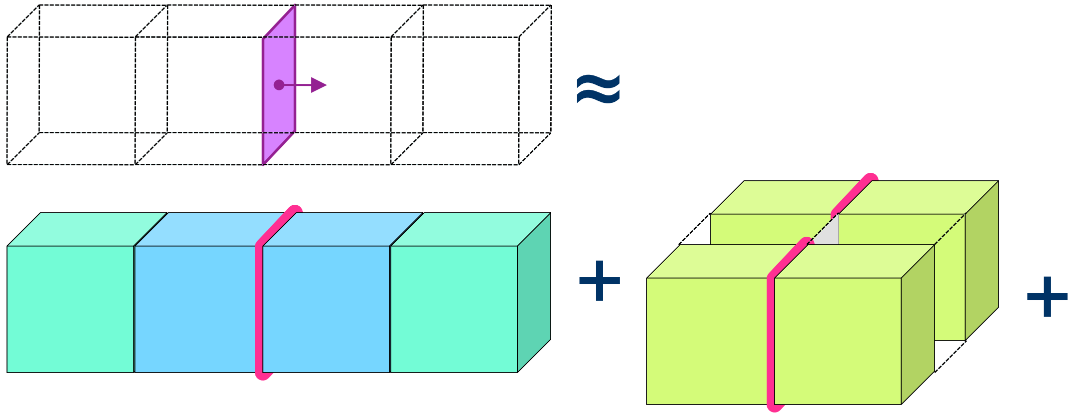
4^{th} Order Operator <∇•β∇u>

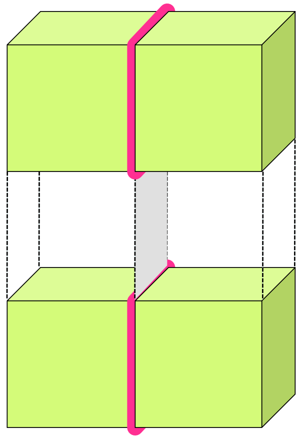
- To 2^{nd} order, we can approximate each of these flux terms as a 2-point weighted stencil…



- Hence, in 3D, 6 such terms form a 7-point variable-coefficient stencil.

- For 4^{th} order, additional terms are required…





- 25-point stencil…
  - 18 x 4-point stencils
  - 4x the floating-point operations
  - no extra DRAM data movement
  - 3x the neighbors (faces+edges)
  - 2-deep ghost zones

Choice of Smoother

- Whereas Chebyhev and Jacobi are easily SIMDized by most compilers today, we wanted a smoother to **challenge the compiler/ISA without sacrificing parallelism**.
- Out-of-place Gauss Seidel Red Black (**GSRB**) iteration…
  - Ping pong between two arrays (u and u_{new}) like Jacobi
  - Unlike Jacobi, only apply the stencil if the cell and iteration color match. Otherwise simply copy the old value to the new array.
  - Generally performs well mathematically and is insensitive to parallelism implementation choices (reproducible when threaded/vectorized)
  - Reference implementation includes stride-2, conditional, and vector variants
  - Two-pass wavefront (calculate fluxes, smooth) implementation is viable

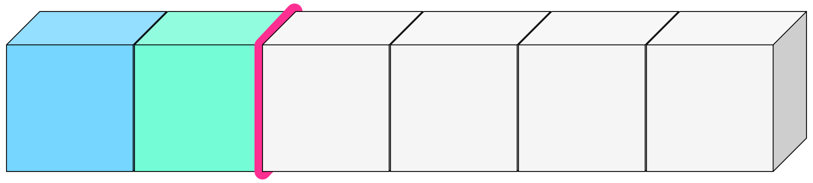
4^{th} Order Boundary Conditions

- In HPGMG-FV, as the boundary exists on cell faces, the boundary condition must be enforced prior to **every application of a stencil**.
- v0.2 and earlier used a simple, linear approximation for the zero Dirichlet boundary condition.
  - It was possible to fuse these boundary condition stencils into the operator itself
  - As such, one could **eliminate both reduced parallelism and irregular memory access** as one traverses the boundary.
  - This optimization is atypical of many real codes and undermines the benchmark's ability to evaluate the ISA/architecture/compiler/runtime response to challenging sub problems.
  - As such, it was eliminated in v0.3 and replaced with a **4^{th} order boundary condition…**

4th Order Boundary Conditions

- The 4th order boundary condition is realized by filling in ghost zone values extrapolated using interior values.
- Produces three basic families of boundary condition stencils…

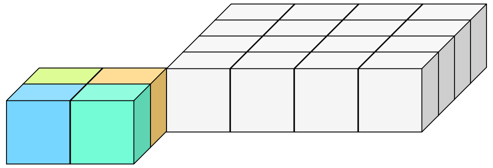
Faces



2 ghost zones x 6 symmetries

= 12 different stencil types

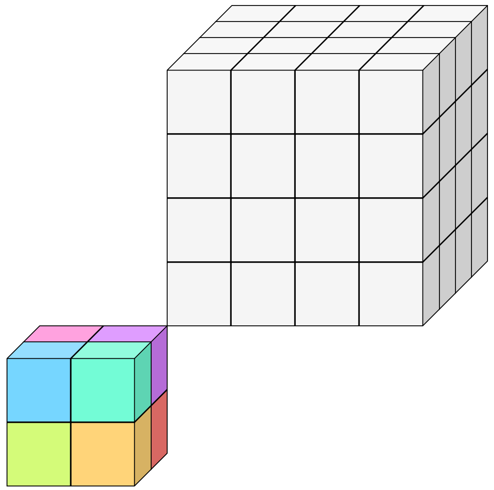
Edges



4 ghost zones x 12 symmetries

= 48 different stencil types

Corners



8 ghost zones x 8 symmetries

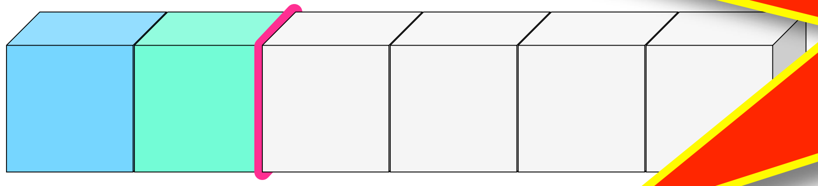
= 64 different stencils

4^{th} Order Boundary Conditions

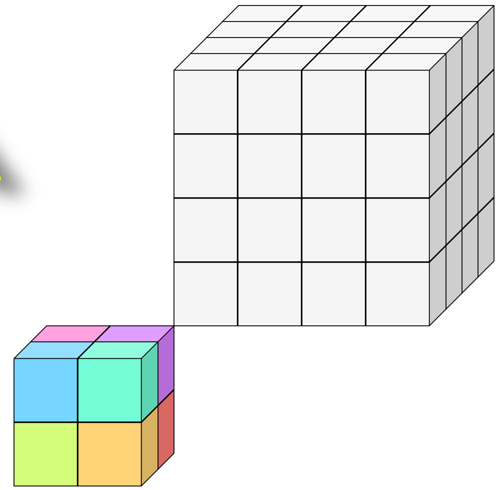
- The 4^{th} order boundary condition realiz d by filling in ghost zone values extrapolate usin alues.
- Produces thr basic fa ondition stencils…

Each ApplyOp( ) requires over 120 stencils !

Faces



Corners



2 ghost zones x 6 symmetries
= 12 different stencil types

4 ghost zones x 12 symmetries
= 48 different stencil types

8 ghost zones x 8 symmetries
= 64 different stencils

High Order Interpolation

- For 2nd order, we used …
  - Piecewise Constant interpolation (1pt stencil) in the V-Cycles
  - Piecewise Linear interpolation (8pt stencil) for FMG's F-Cycle.
- These operations …
  - **were strongly memory-bound**
  - stressed neither core architecture nor the compiler.

- For 4th order, we now use …
  - Quadratic interpolation (**27pt stencil**) in the V-Cycles
  - Quartic interpolation (**125pt stencil**) in FMG's F-Cycle
- These operations …
  - Require communication and BCs
  - Are potentially compute-bound
  - Exercise architecture and compilers (complex symmetries can be exploited)

MPI Communication

- The operator requires communicating with face and edge neighbors
  - 3x the messages per applyOp( )
  - 2x the message size (2-deep ghost zone)
- Moreover, all interpolations now require communication with face, edge, and corner neighbors (at least 26 neighbors)

- **process0 still performs O($\log^2(P)$) ghost zone exchanges**.
- As such, at large scale, communication and coarse grid operations are relatively expensive.

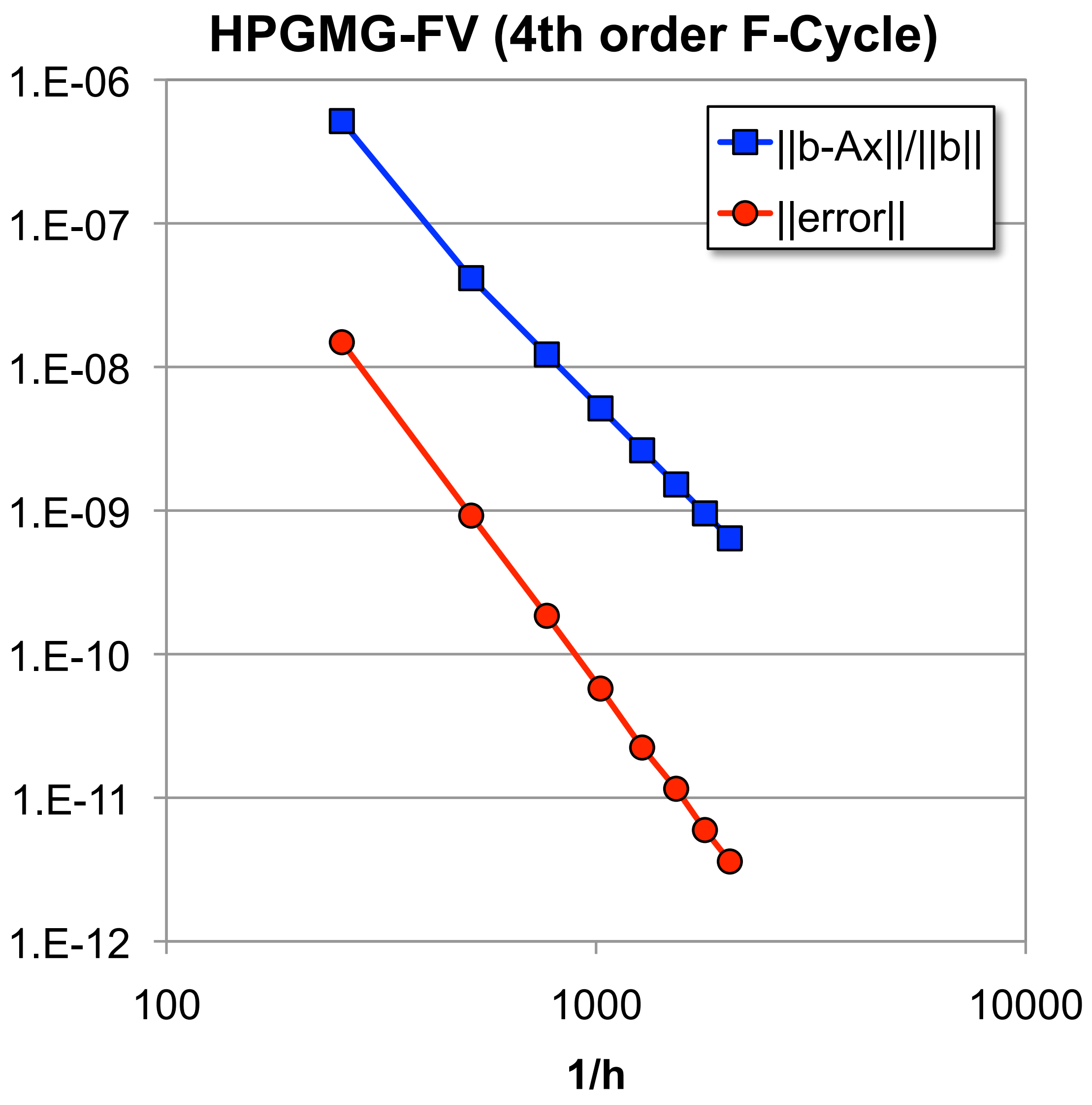
Overall Performance Implications

- The new 4th order HPGMG should …
  - perform 4x the FP operations
  - send 3x the MPI messages
  - double the MPI message size
  - move no more data from DRAM
  - attain 4th order accuracy
  - attain lower relative residual ($\sim 10^{-9}$)

- As a result, HPGMG should be more sensitive to…
  - core/cache architectural parameters
  - compiler optimizations
  - messaging overheads and network latencies
  - network injection and bisection bandwidths

Mathematical Performance

- Examine error and relative residual as a function of 1/h (e.g. problem dimension of up to $2K^3$)…
  - Error is strongly 4th order with 3 GSRB presmooths + 3 GSRB postsmooths.
  - Residual is quickly reduced ($<10^{-9}$ in one F-Cycle)
- Mathematical properties are independent of parallelism choices (processes, threads, box size, etc…)



Initial 4th Order HPGMG-FV Results

HPGMG Rank	System Name	Site	DOF/s (h)	DOF/s (2h)	DOF/s (4h)	MPI	OMP	Acc	DOF per Process	Top500 Rank
1	Mira	ALCF	5.00e+11	3.13e+11	1.07e+11	49152	64	0	36M	5
			3.95e+11	2.86e+11	1.07e+11					
2	Edison	NERSC	2.96e+11	2.46e+11	1.27e+11	10648	12	0	128M	34
3	Titan (CPU-only)	OLCF	1.61e+11	8.25e+10	2.37e+10	36864	8	0	48M	2
4	Hopper	NERSC	7.26e+10	5.45e+10	2.74e+10	21952	6	0	16M	62
5	SuperMUC	LRZ	7.25e+10	5.25e+10	2.80e+10	4096	8	0	54M	20
6	Hazel Hen	HLRS	1.82e+10	8.73e+09	2.02e+09	1024	12	0	16M	-
7	SX-ACE	HLRS	3.24e+09	1.77e+09	7.51e+08	256	1	0	32M	-
8	Babbage (MIC-only)	NERSC	7.62e+08	3.16e+08	9.93e+07	256	45	0	8M	-

Notes:
- v0.3 was made available only 3 months ago
- Mira and Babbage used optimized implementations (alignment intrinsics, loop fission to reduce prefetcher contention, OMP4 SIMD pragmas, …)
- Only 22% of SUPER MUC was available
- Babbage (KNC): 4 MPI per MIC. MPI performance was poor. Network scalability was very poor. Very Sensitive to coarse grid operations.
- Each solve performs approximately 1200 FP operations per DOF.

Initial 4th Order HPGMG-FV Results

~600 TF/s (6% of peak)

Problem sizes (N,N/8,N/64)

~355 TF/s (14% of peak)

~3.9 TF/s (24% of peak)

4K nodes is only 22% of the system

512 nodes is only 6% of the system

HPGMG Rank	System Name	Site	DOF/s (h)	DOF/s (2h)	DOF/s (4h)	MPI	OMP	Acc	DOF per Process	Top500 Rank
1	Mira	ALCF	5.00e+11	3.13e+11	1.07e+11	49152	64	0	36M	5
			3.95e+11	2.86e+11	1.07e+11					
2	Edison	NERSC	2.96e+11	2.46e+11	1.27e+11	10648	12	0	128M	34
3	Titan (CPU-only)	OLCF	1.61e+11	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	48M	2
4	Hopper	NERSC	7.26e+10	5.45e+10	2.74e+10	21952	6	0	16M	62
5	SuperMUC	LRZ	7.25e+10	5.25e+10	2.80e+10	4096	8	0	54M	20
6	Hazel Hen	HLRS	1.82e+10	8.73e+09	2.02e+09	1024	12	[illegible]	[illegible]	-
7	SX-ACE	HLRS	3.24e+09	1.77e+09	7.51e+08	256	1	[illegible]	[illegible]	-
8	Babbage (MIC-only)	NERSC	7.62e+08	3.16e+08	9.93e+07	256	45	0	8M	-

Notes:

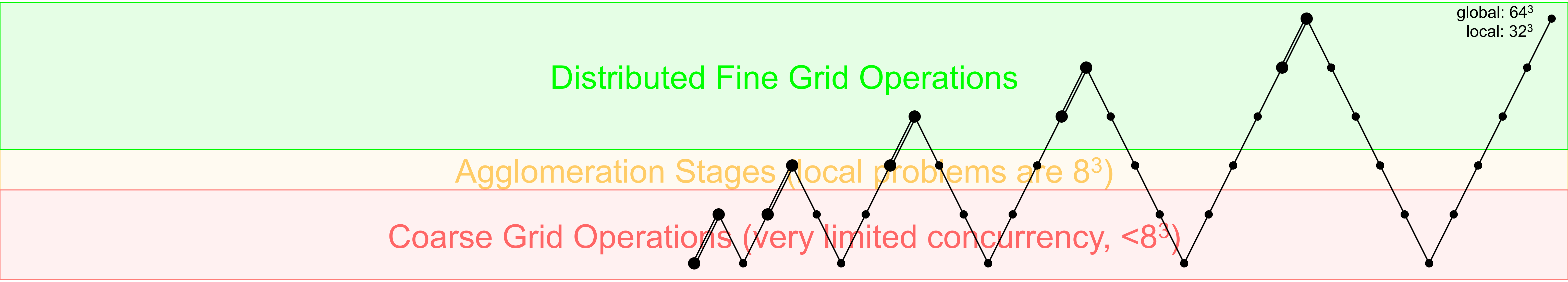
- v0.3 was made available only 3 months ago
- Mira and Babbage used optimized implementations (alignment intrinsics, loop fission to reduce prefetcher contention, OMP4 SIMD pragmas, …)
- Only 22% of SUPER MUC was available
- Babbage (KNC): 4 MPI per MIC. MPI performance was poor. Network scalability was very poor. Very Sensitive to coarse grid operations.
- Each solve performs approximately 1200 FP operations per DOF.

FMG Weak Scaling Challenge

- As one weak scales HPGMG-FV, coarse grid operations become an increasingly dominate fraction of the execution…

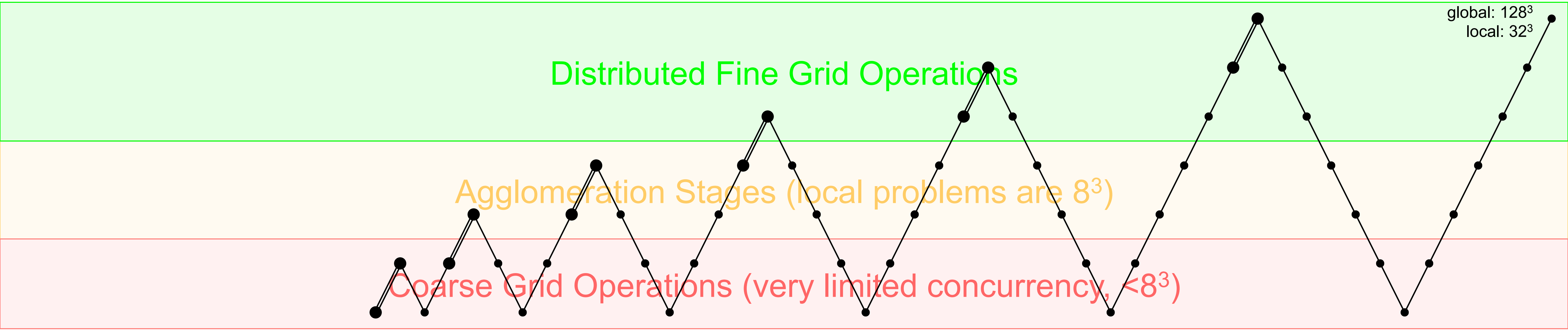
FMG Weak Scaling Challenge

- As one weak scales HPGMG-FV, coarse grid operations become an increasingly dominate fraction of the execution…



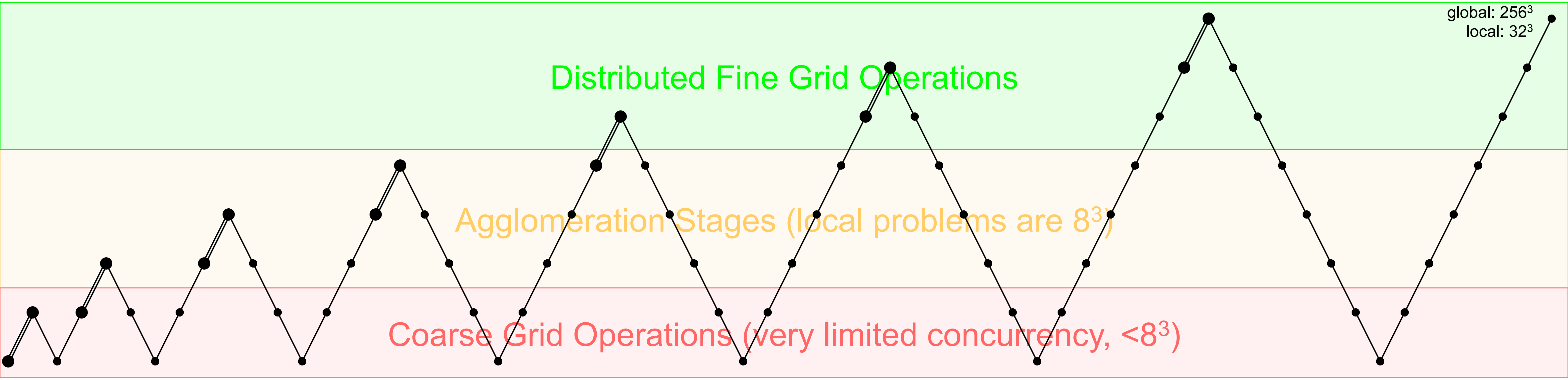
FMG Weak Scaling Challenge

- As one weak scales HPGMG-FV, coarse grid operations become an increasingly dominate fraction of the execution…



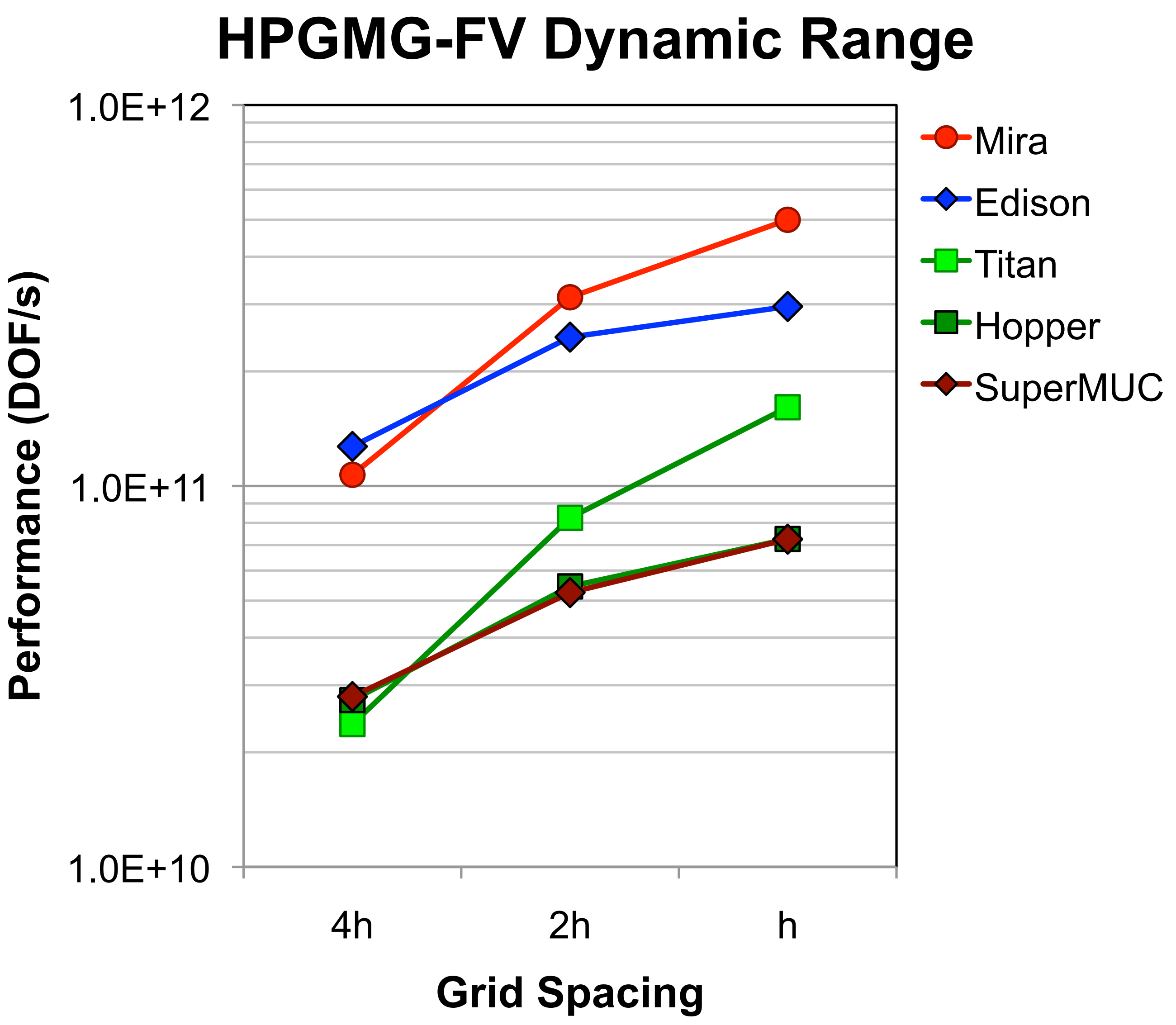
FMG Weak Scaling Challenge

- As one weak scales HPGMG-FV, coarse grid operations become an increasingly dominate fraction of the execution…
- Petascale machines can have 13+ levels !



Observations on Dynamic Range

- Dynamic Range gauges performance as a function of problem size (memory/node)
  - ‘h’ is the largest problem
  - ‘4h’ is a problem 64x (4^3) smaller
- Titan (Gemini) was found to be particularly sensitive to problem size
  - CPU-only data (apples-to-apples)
  - **7x lower performance at 4h**
- Conversely, Edison (Aries) saw only a 2.3x loss in performance at 4h
- Suggests systems are now sensitive to network architecture and memory usage
- We eagerly await HBM and GDDR-based results (lower capacity / more bandwidth)



Acknowledgements

- This research used resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231.
- This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.
- This research used resources of the Oak Ridge Leadership Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Questions?

HPGMG-FV is available for download:

https://bitbucket.org/hpgmg/hpgmg/

Submission Guidelines:

http://crd.lbl.gov/departments/computer-science/PAR/research/hpgmg/submission/

HPC Benchmarking

Vladimir Marjanovic
High Performance Computing Center Stuttgart (HLRS)

BERKELEY LAB

LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

GPU Implementation of HPGMG-FV

Simon Layton

NVIDIA

Questions and Discussion