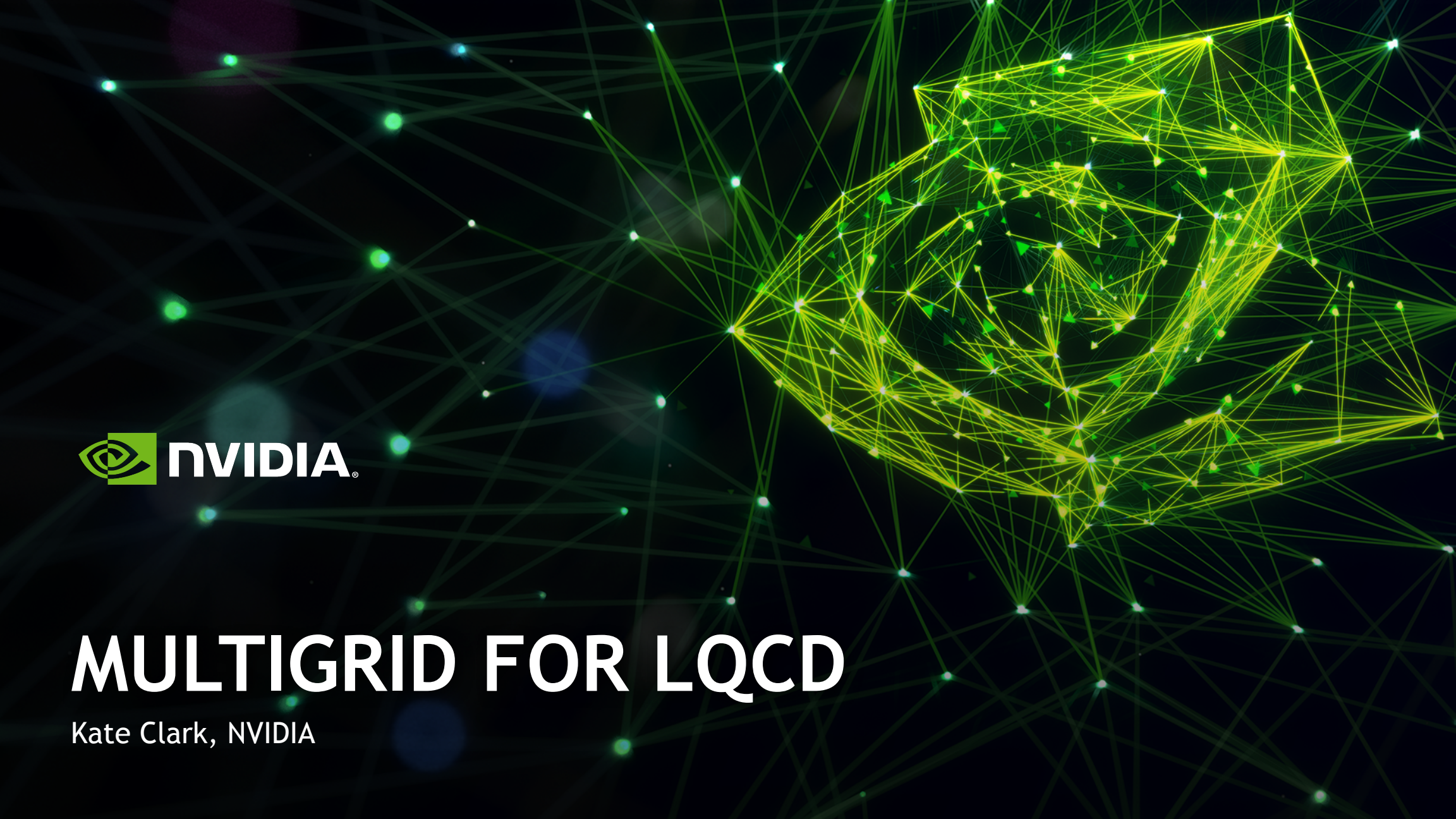


MULTIGRID FOR LQCD

Kate Clark, NVIDIA

AGENDA

Introduction to Multigrid

Multigrid on GPUs

Multigrid and HMC

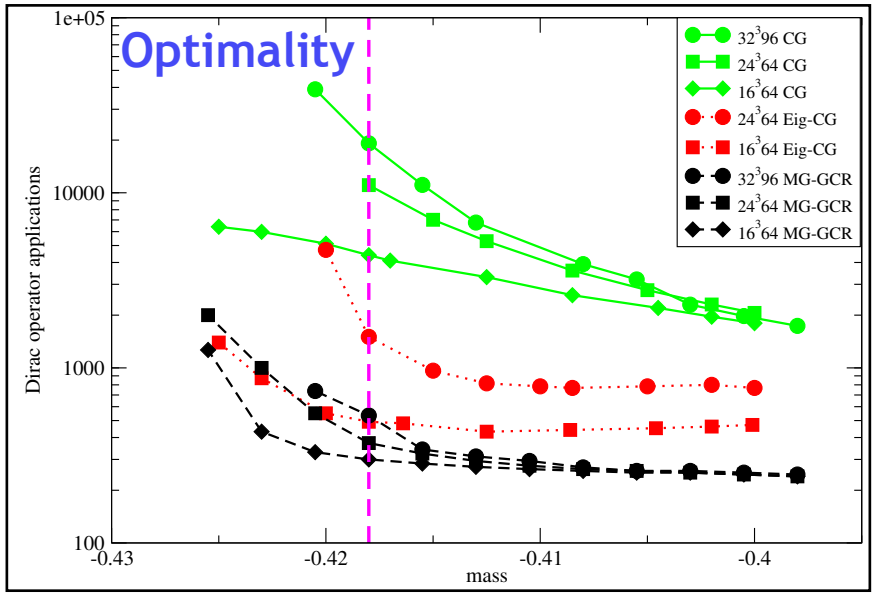
Twisted-Mass Multigrid

Staggered Multigrid

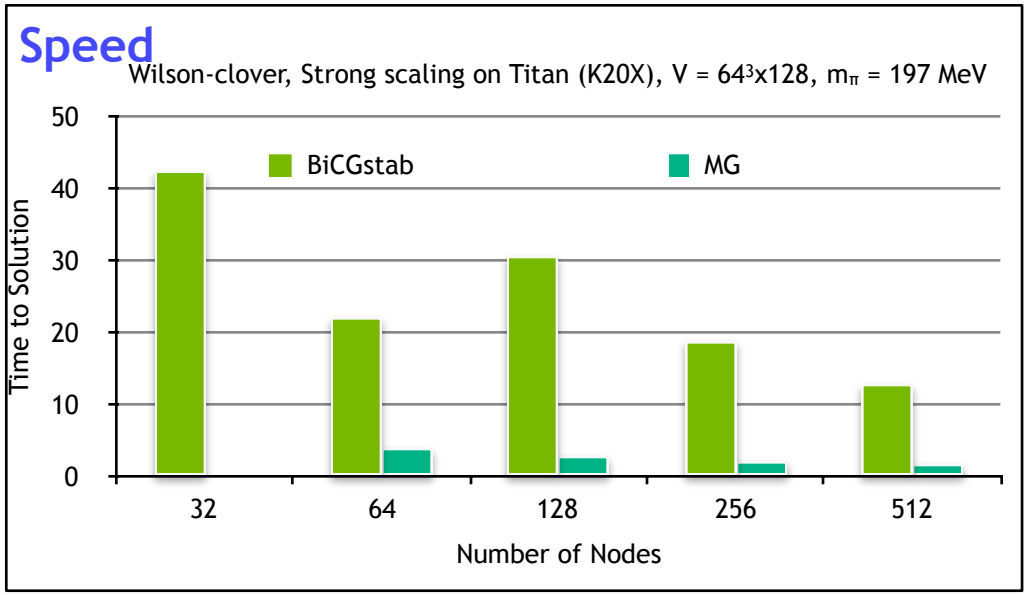
Ongoing and Future Challenges

INTRODUCTION TO MULTIGRID

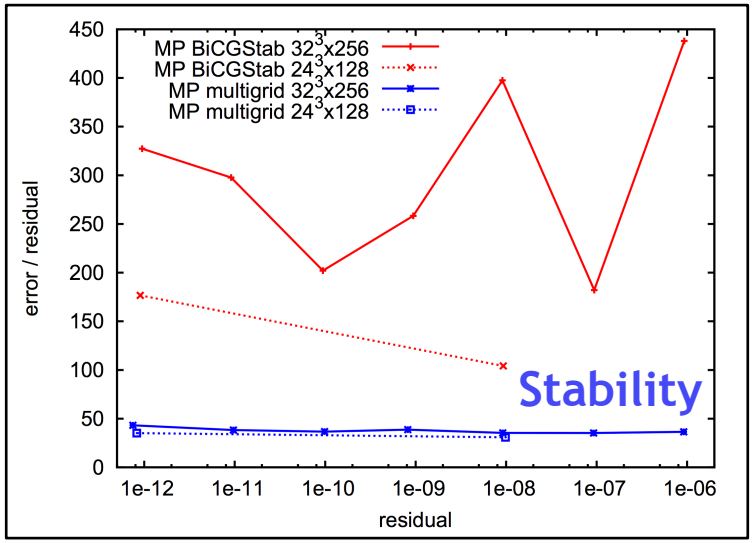
WHY MULTIGRID?



Babich *et al* 2010



Clark *et al* (2016)



Osborn *et al* 2010

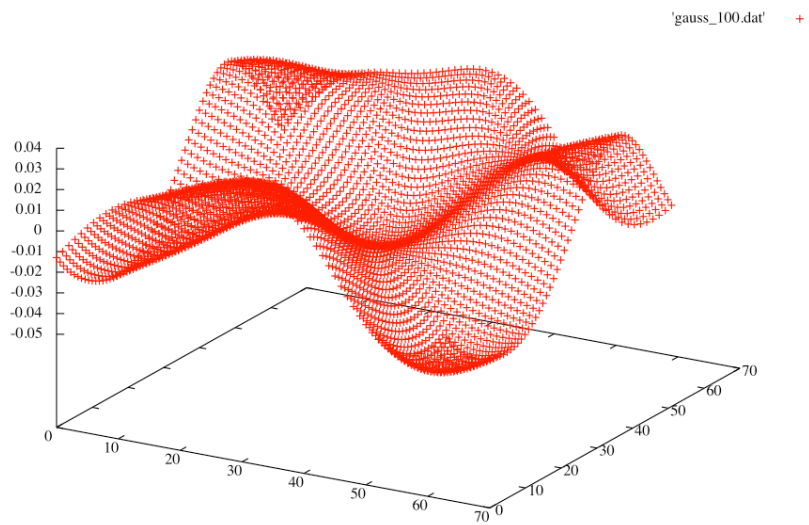
INTRODUCTION TO MULTIGRID

Stationary iterative solvers effective on high frequency errors

Minimal effect on low frequency error

Example

- Free Laplace operator in 2d
- $Ax = 0$, x_0 = random
- Gauss Seidel relaxation
- Plot error $e_i = -x_i$



INTRODUCTION TO MULTIGRID

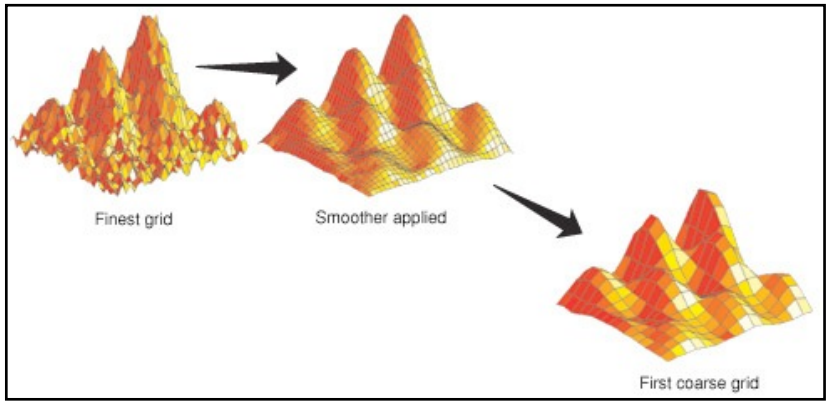
Low frequency error modes are smooth

Can accurately represent on coarse grid

Low frequency on fine
=> high frequency on coarse

Relaxation effective again on coarse grid

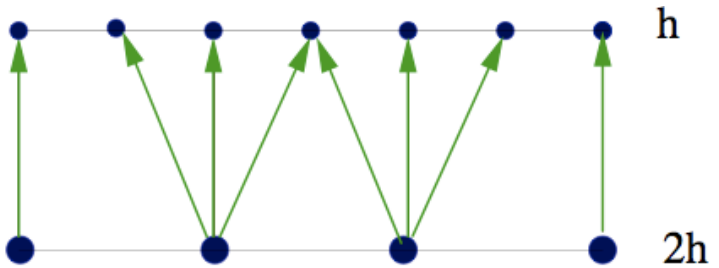
Interpolate back to fine grid



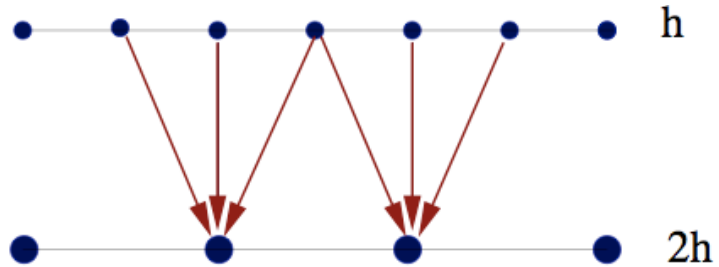
Falgout

INTRODUCTION TO MULTIGRID

Define the Prolongator P



Define the Restriction operator R = P†



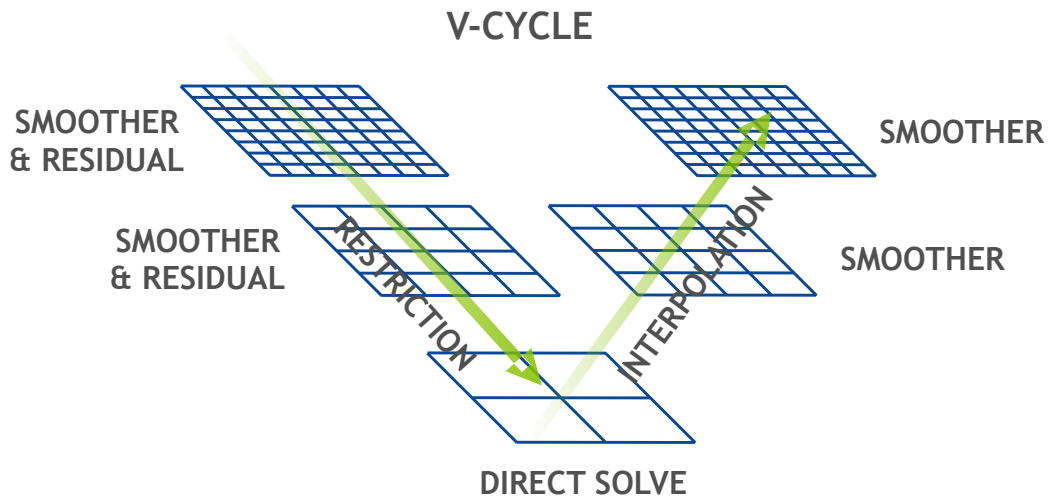
Operator on coarse space

$$A_c = P^\dagger A P$$

MULTIGRID V-CYCLE

Solve

1. Smooth
2. Compute residual
3. Restrict residual
4. Recurse on coarse problem
5. Prolongate correction
6. Smooth
7. If not converged, goto 1



Multigrid has optimal scaling

- *O(N)* Linear scaling with problem size
- Convergence rate independent of condition number

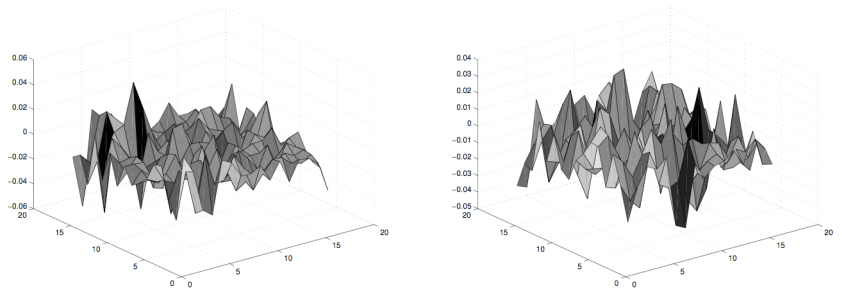
For LQCD, we do not know the null space components that need to be preserved on the coarse grid

FAILURE OF CLASSICAL MULTIGRID

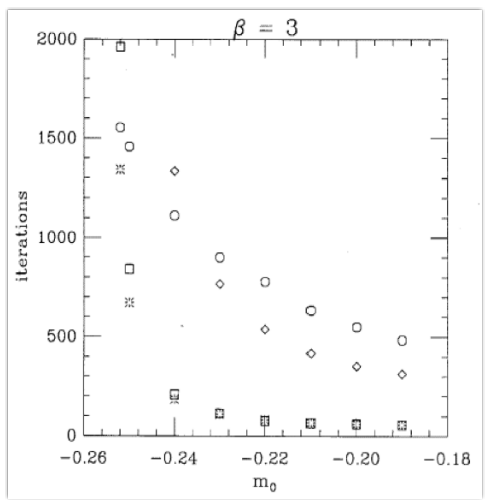
U field is not geometrically smooth for interacting case

Low frequency modes of Dirac operator oscillatory

e.g., 2d Wilson Dirac operator error after 200 Gauss-Seidel iterations



Geometric multigrid completely fails



LQCD and MG have long and painful history, e.g.,

- PTMG (Lauwers *et al*)
- Projective MG (Brower *et al*)
- RG approaches (de Forcrand *et al*)

Previous MG methods

- Work for smooth gauge fields ($\mu^{-1} < l_\sigma$)
- Fail as m->0 ($\mu^{-1} > l_\sigma$)

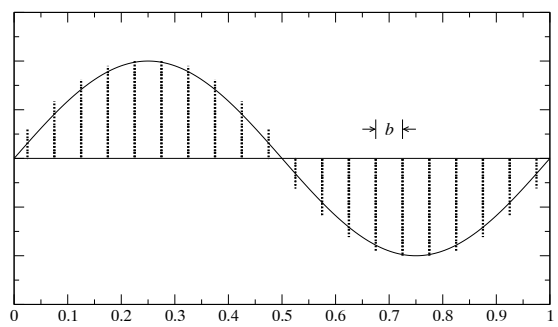
WHY DOES MULTIGRID WORK?

In free field theory, zero mode = constant

Exactly preserved by the projector $0 = (1 - P^\dagger P)\psi_0$

Near null space approximately preserved $0 \approx (1 - P^\dagger P)\psi_k$

- Weak approximation property (cf Lüscher "Local Coherence")
- Possible because eigenvectors are not locally orthogonal



Need P such that low modes space is preserved

- Adaptivity required for interacting gauge fields

Babich, Branich, Brower, Clark, Manteuffel, McCormick, Osborn, Rebbi (2009)

ADAPTIVE GEOMETRIC MULTIGRID

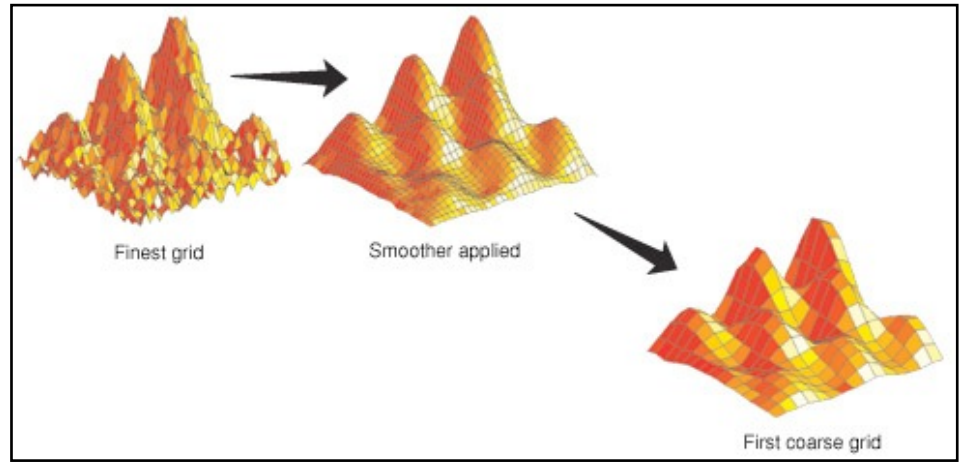
Based on “Adaptive Algebraic Smooth Aggregation Multigrid” (Brezina et al, 2003)

Adaptively find candidate null-space vectors

- Dynamically learn the null space and use this to define the prolongator
- Algorithm is self learning

Setup

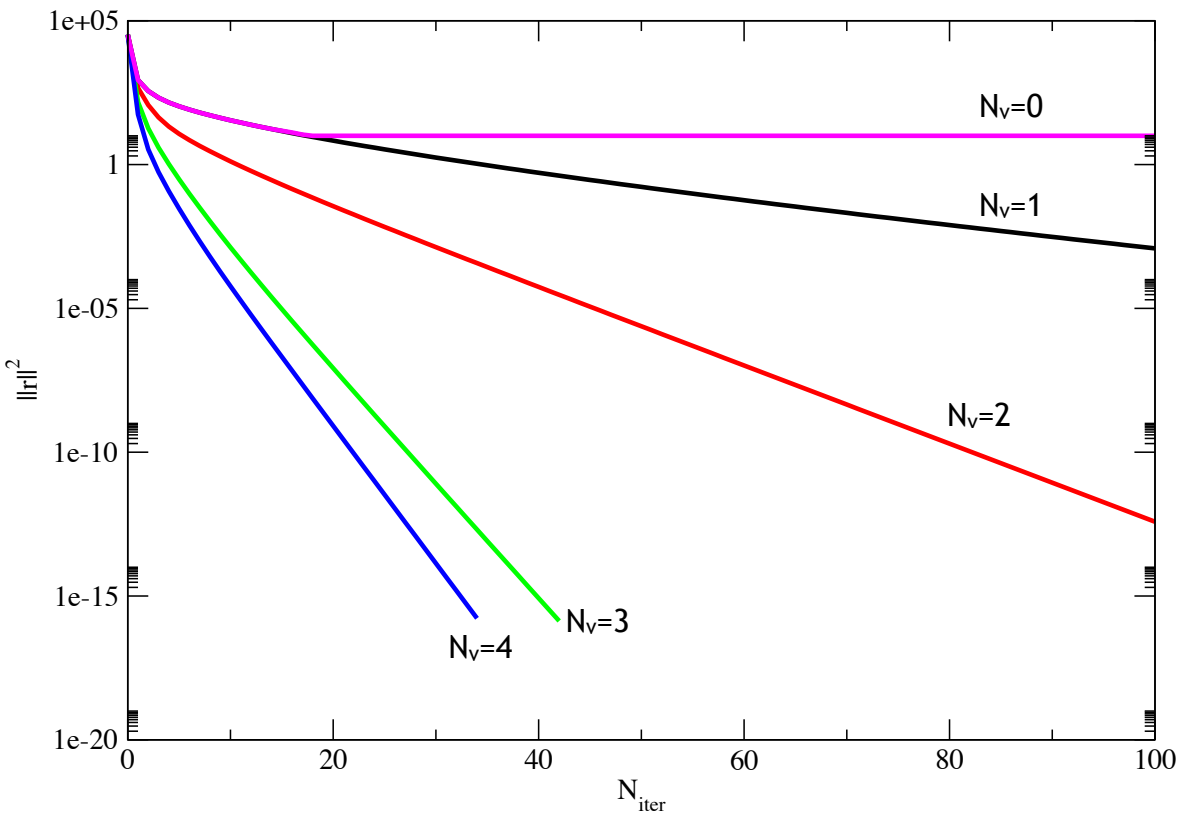
1. Set solver to be simple smoother
2. Apply current solver to random vector $v_i = P(D)\ \eta_i$
3. If convergence good enough, solver setup complete
4. Construct prolongator using fixed coarsening $(1 - P\ R)\ v_k = 0$
   - ➡ Typically use 4^4 geometric blocks
   - ➡ Preserve chirality when coarsening $R = \gamma_5\ P^\dagger\ \gamma_5 = P^\dagger$
5. Construct coarse operator ($D_c = R\ D\ P$)
6. Recurse on coarse problem
7. Set solver to be augmented V-cycle, goto 2



Falgout

ADAPTIVE GEOMETRIC MULTIGRID

4-d Laplace operator



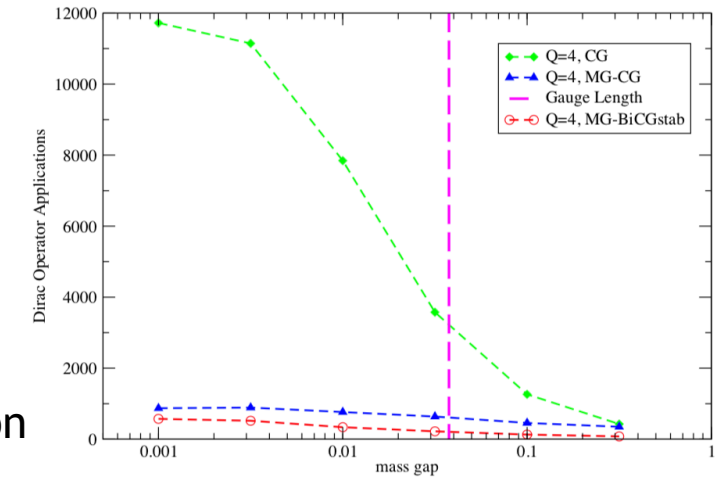
Typically 20-30 vectors needed to capture Dirac null space

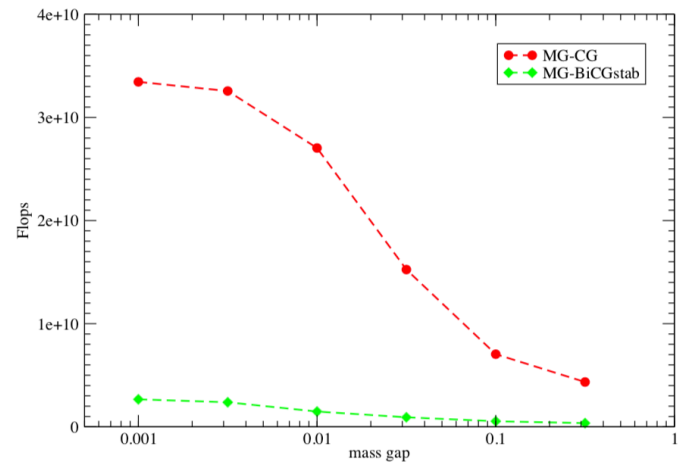
Brannick, Brower, Clark, Osborn, Rebbi (2007)

ADAPTIVE GEOMETRIC MULTIGRID

2-d Wilson

- 128 × 128 lattice, β = 6, 10, m = 0.001 – 0.5
- MG setup run at lightest mass only
- $D^\dagger D$-MG algorithm
  - 4×4(×2) blocking, 3 levels, N_V = 8 – Under-relaxed MR relaxation
  - Preconditioner for CG
- D-MG algorithm
  - 4×4 blocking, 3 levels, N_V = 4 – Under-relaxed MR relaxation
  - Preconditioner for BiCGstab
- Results
  - Critical slowing down virtually gone – Weak dependence on β
  - **D-MG superior to $D^\dagger D$-MG**

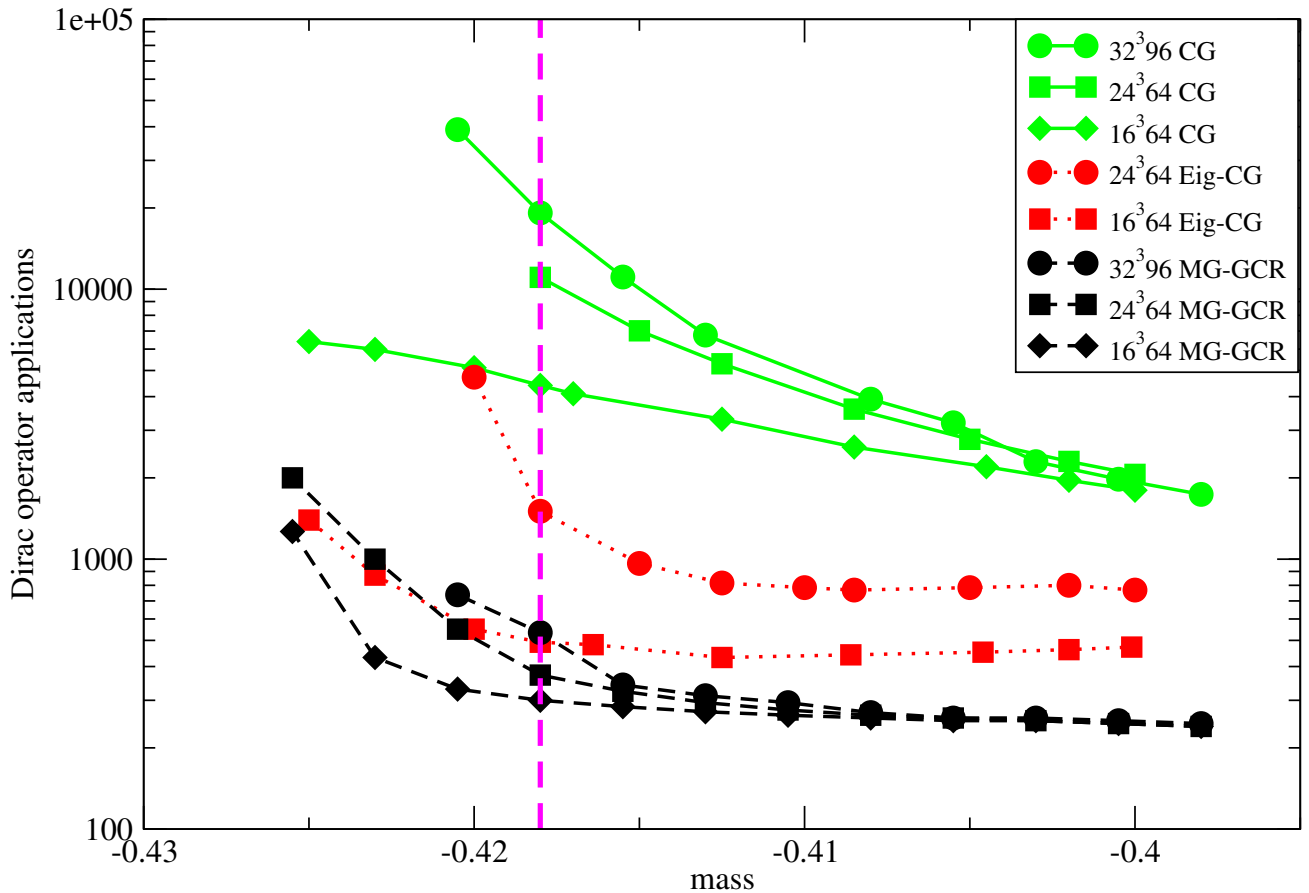




REAL LQCD MULTIGRID

Babich, Brannick, Brower, Clark, Manteuffel, McCormick, Osborn, Rebbi (2009)

CG vs Eig-CG vs MG (Anisotropic Wilson)



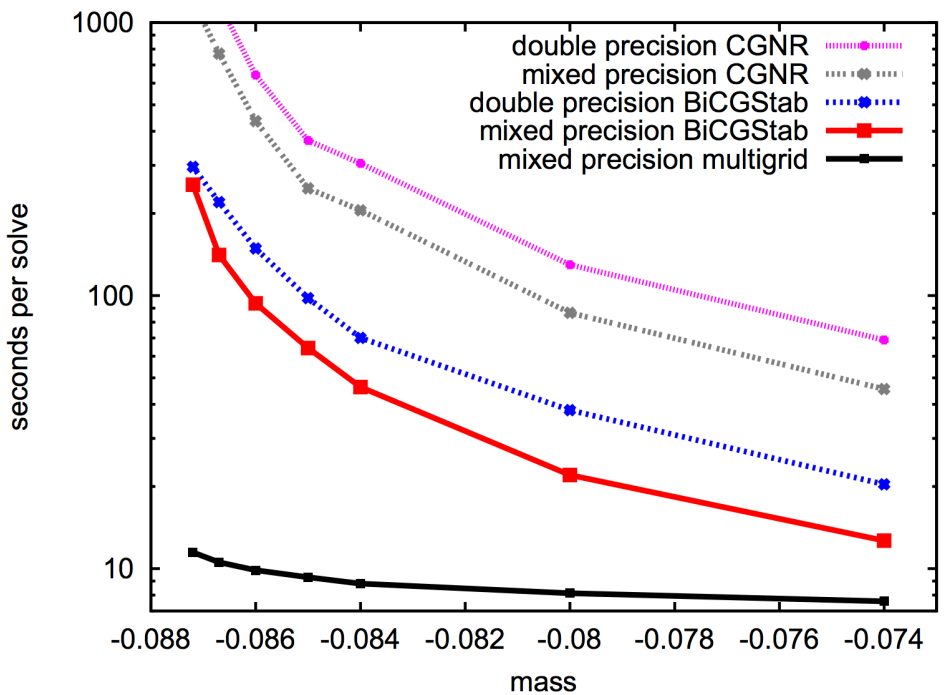
240 vectors

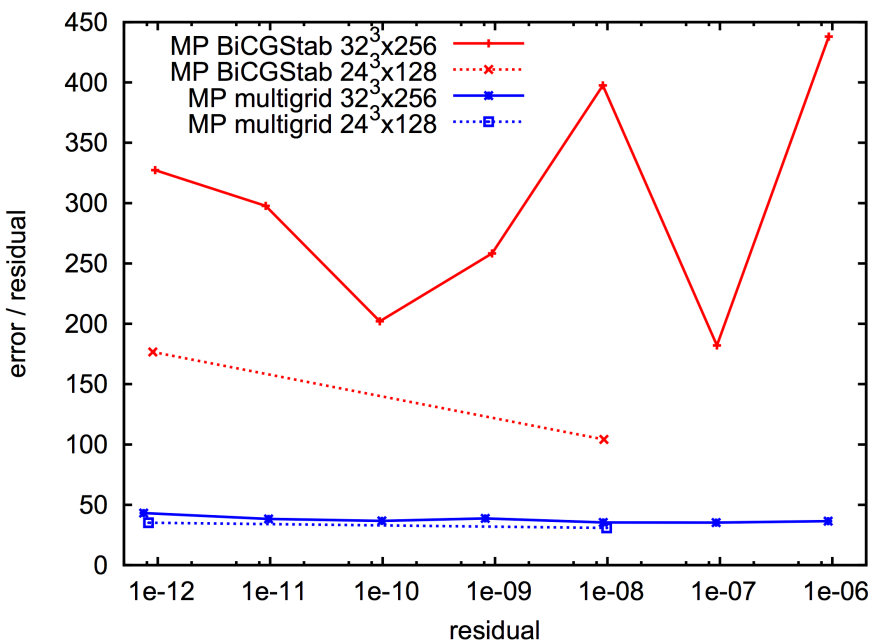
20 vectors

Osborn, Babich, Brannick, Brower, Clark, Cohen, Rebbi (2009)

CLOVER MULTIGRID

Combined multigrid with even-odd preconditioning





INEXACT DEFLATION

Lüscher (2007)

Removal of critical slowing down through *Local Coherence*

Closely related to adaptive multigrid: same building blocks, put together in a different order

1. Deflate RHS $\hat{b} = (1 - DPD_c^{-1}P^\dagger)b$
2. Solve deflated system $(1 - DPD_c^{-1}P^\dagger)D\hat{x} = \hat{b}$
3. Solve little Dirac operator $(P^\dagger D_c P)y = b$
4. Solution given by $x = \hat{x} + y$

Not scalable to multiple levels owing to subtraction of low modes

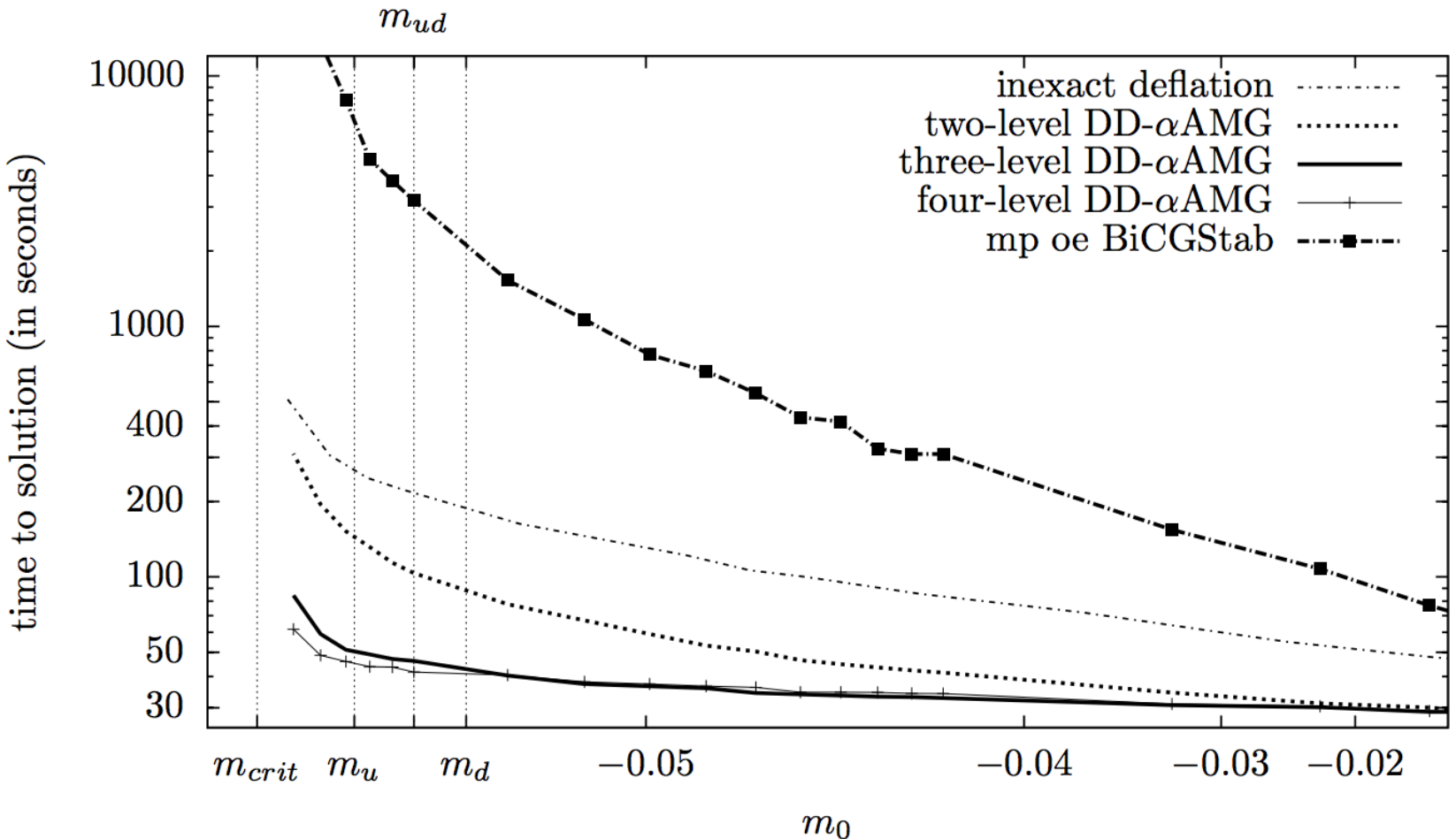
vs MG which uses a multiplicative preconditioner

Requires accurate solution of coarse grid operator

Wilson MG only requires a very loose stopping condition on each level

MULTIGRID VS INEXACT DEFLATION

Frommer, Kahl, Krieg, Leder and Rottmann (2014)



DD-αAMG

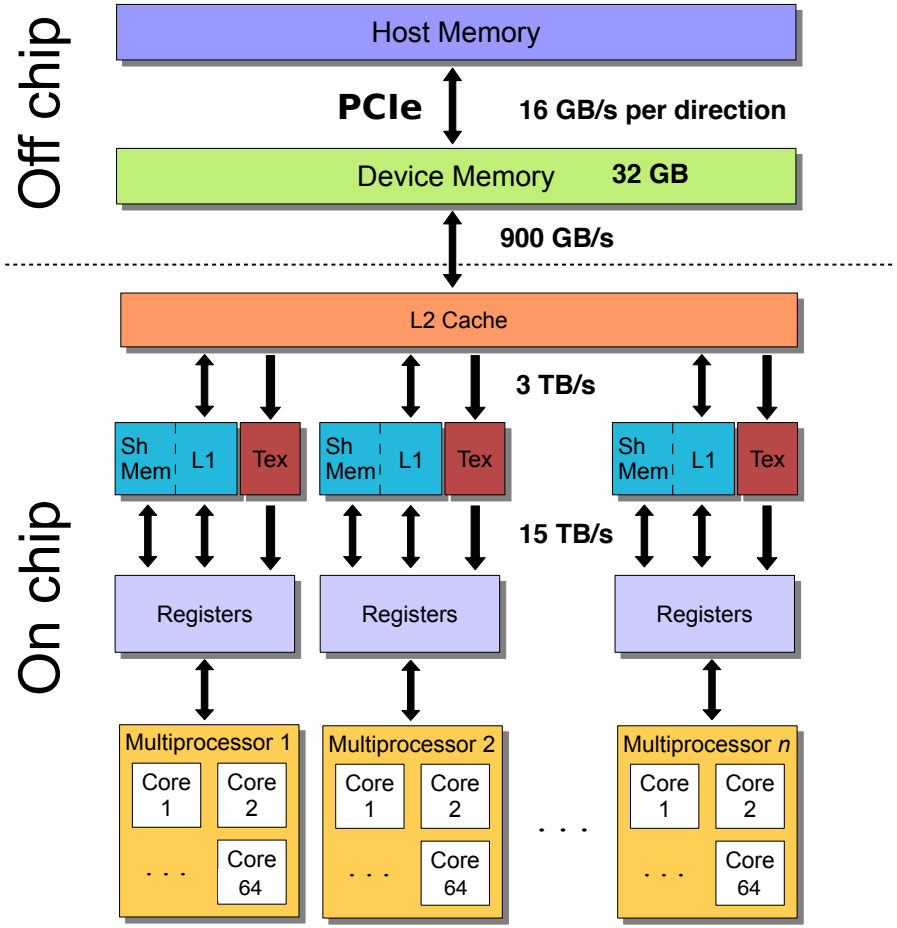
Frommer, Kahl, Krieg, Leder and Rottmann (2014)

Use SAP as a smoother for multigrid for improved scalability

	id 5, 128 cores			id 6, 256 cores		
	AMG-d	AMG-20	DD-αAMG	AMG-d	AMG-10	DD-αAMG
setup time	2424s	826s	896s	2464s	607s	656s
solve iter	14	22	10	13	21	11
solve time	45.4s	66.0s	57.1s	36.5s	50.4s	37.3s
	id 5, 8192 cores			id 6, 8192 cores		
	AMG-d	AMG-40	DD-αAMG	AMG-d	AMG-20	DD-αAMG
setup time	52.3s	24.6s	27.7s	89.9s	29.1s	32.3s
solve iter	14	16	10	13	16	11
solve time	4.75s	5.51s	1.82s	3.49s	3.43s	1.86s


MULTIGRID ON GPUS

WHAT IS A GPU?



- Tesla V100 - Volta architecture (2017)
  - Massively threaded - 5120 processing cores
  - 7.5 FP64 /15 FP32 / 125 FP16 Gflops peak
- Deep memory hierarchy
  - As we move away from registers
    - Bandwidth decreases
    - Latency increases
- Inverse memory hierarchy
  - 40 MiB register file (up to 255 registers / thread)
  - 10 MiB 128 KiB L1 / shared memory
  - 6 MiB coherent L2 cache
- Programmed using a diversity of approaches
  - CUDA C++ / Fortran / Python
  - OpenACC / OpenMP directives
  - Future: C++17 pSTL / Fortran 2018 `DO CONCURRENT`

QUDA

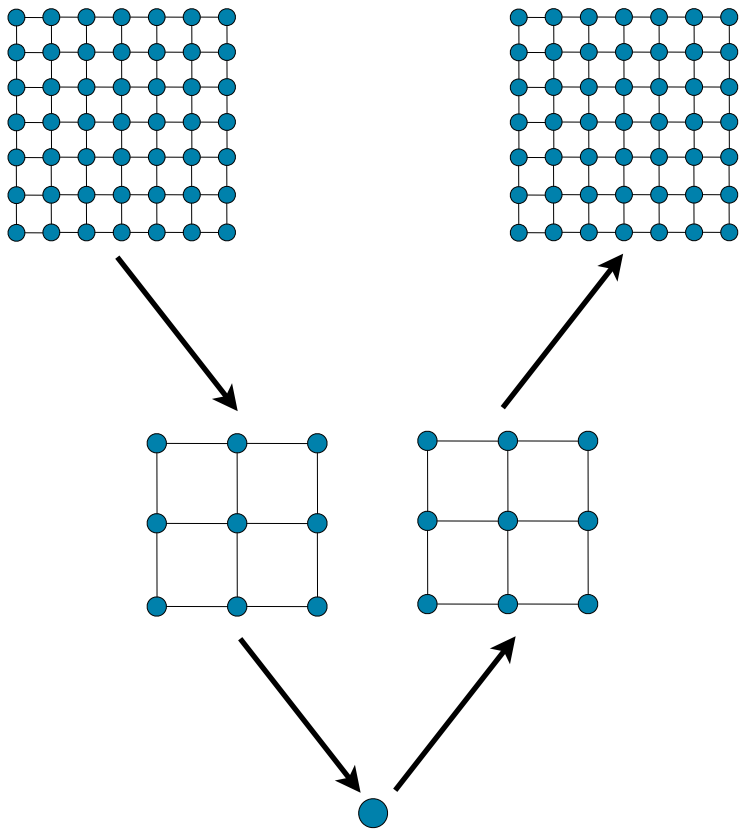
- "QCD on CUDA" – http://lattice.github.com/quda (open source, BSD license)
- Effort started at Boston University in 2008, now in wide use as the GPU backend for BQCD, Chroma, CPS, MILC, TIFR, etc.
- Provides:
  - Various solvers for all major fermionic discretizations, with multi-GPU support
  - Additional performance-critical routines needed for gauge-field generation
- Maximize performance
  - Exploit physical symmetries to minimize memory traffic
  - Mixed-precision methods
  - Autotuning for high performance on all CUDA-capable architectures
  - Domain-decomposed (Schwarz) preconditioners for strong scaling
  - Eigenvector and deflated solvers (Lanczos, EigCG, GMRES-DR)
  - Multi-source solvers
  - Multigrid solvers for optimal convergence
- A research tool for how to reach the exascale

QUDA CONTRIBUTORS

10+ years - lots of contributors

Ron Babich (NVIDIA)
Simone Bacchio (Cyprus)
Kip Barros (LANL)
Rich Brower (Boston University)
Nuno Cardoso (NCSA)
Kate Clark (NVIDIA)
Michael Cheng (Boston University)
Carleton DeTar (Utah University)
Justin Foley (Utah -> NIH)
Joel Giedt (Rensselaer Polytechnic Institute)
Arjun Gambhir (William and Mary)
Steve Gottlieb (Indiana University)
Kyriakos Hadjiyiannakou (Cyprus)
Dean Howarth (BU)
Bálint Joó (Jlab)
Hyung-Jin Kim (BNL -> Samsung)
Bartek Kostrzewa (Bonn)
Claudio Rebbi (Boston University)
Eloy Romero (William and Mary)
Hauke Sandmeyer (Bielefeld)
Guochun Shi (NCSA -> Google)
Mario Schröck (INFN)
Alexei Strelchenko (FNAL)
Jiqun Tu (Columbia)
Alejandro Vaquero (Utah University)
Mathias Wagner (NVIDIA)
André Walker-Loud
Evan Weinberg (NVIDIA)
Frank Winter (Jlab)

THE CHALLENGE OF MULTIGRID ON GPU



GPU requirements very different from CPU

- Each thread is slow, but O(10,000) threads per GPU

Fine grids run very efficiently

- High parallel throughput problem

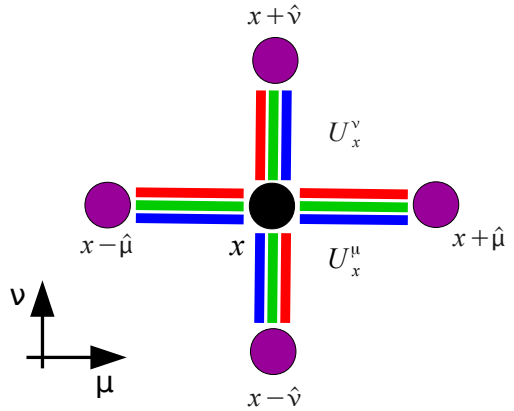
Coarse grids are worst possible scenario

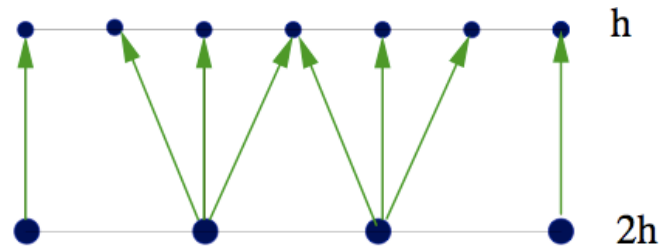
- More cores than degrees of freedom
- Increasingly serial and latency bound
- Little's law (bytes = bandwidth * latency)
- Amdahl's law limiter

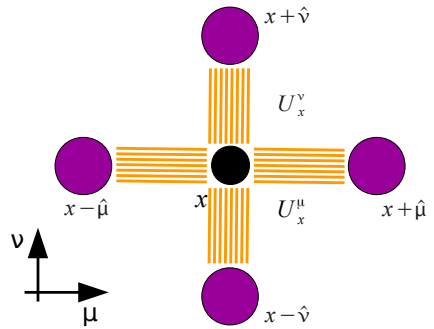
Multigrid exposes many of the problems expected at the Exascale

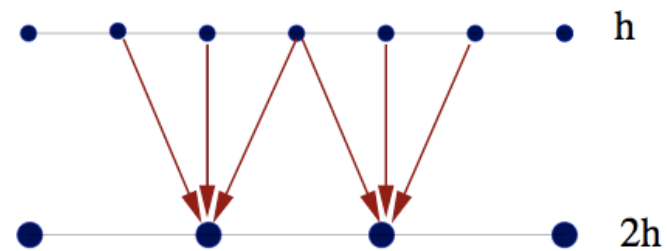
INGREDIENTS FOR PARALLEL ADAPTIVE MULTIGRID

- Multigrid setup
  - Block orthogonalization of null space vectors
  - Batched QR decomposition
- Smoothing (relaxation on a given grid)
  - Repurpose existing solvers
- Prolongation
  - interpolation from coarse grid to fine grid
  - one-to-many mapping
- Restriction
  - restriction from fine grid to coarse grid
  - many-to-one mapping
- Coarse Operator construction (setup)
  - Evaluate $R\,A\,P$ locally
  - Batched (small) dense matrix multiplication
- Coarse grid solver
  - Need optimal coarse-grid operator









MAPPING THE DIRAC OPERATOR TO CUDA

Finite difference operator in LQCD is known as Dslash

Assign a single space-time point to each thread

- V = XYZT threads, e.g., V = 24^4 => 3.3×10^6 threads

Looping over direction each thread must

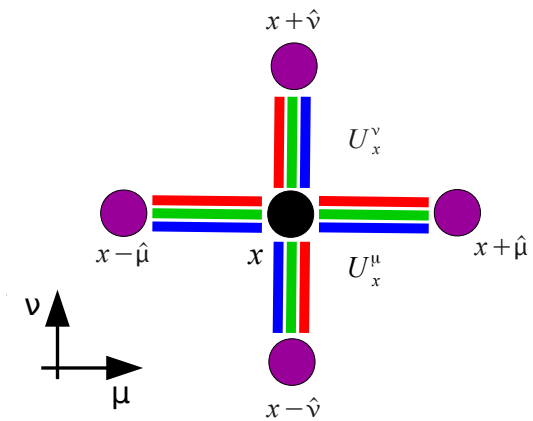
- Load the neighboring spinor (24 numbers x8)
- Load the color matrix connecting the sites (18 numbers x8)
- Do the computation
- Save the result (24 numbers)

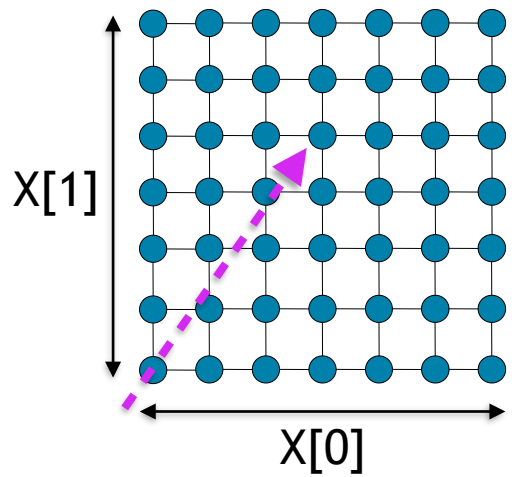
Each thread has (Wilson Dslash) 0.92 naive arithmetic intensity

QUDA reduces memory traffic

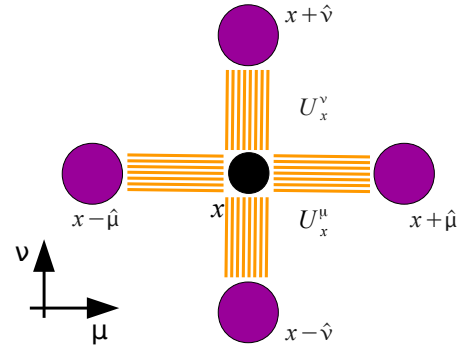
- Exact SU(3) matrix compression (18 => 12 or 8 real numbers)
- Use 16-bit fixed-point representation with mixed-precision solver

$$D_{x,x'} =$$





COARSE GRID OPERATOR

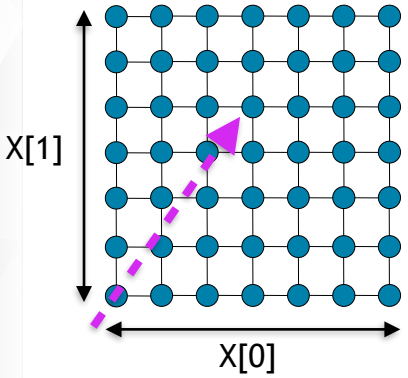


- Coarse operator looks like a Dirac operator (many more colors)
  - Link matrices have dimension $2N_v$ x $2N_v$ (e.g., 48 x 48)

$$\hat{D}_{\mathbf{i}\hat{s}\hat{c},\mathbf{j}\hat{s}'\hat{c}'} = -\sum_{\mu}\left[Y^{-\mu}_{\mathbf{i}\hat{s}\hat{c},\mathbf{j}\hat{s}'\hat{c}'}\delta_{\mathbf{i}+\mu,\mathbf{j}} + Y^{+\mu\dagger}_{\mathbf{i}s\hat{c},\mathbf{j}s'\hat{c}'}\delta_{\mathbf{i}-\mu,\mathbf{j}}\right] + \left(M - X_{\mathbf{i}\hat{s}\hat{c},\mathbf{j}\hat{s}'\hat{c}'}\right)\delta_{\mathbf{i}\hat{s}\hat{c},\mathbf{j}\hat{s}'\hat{c}'}\,.$$

- Fine vs. Coarse grid parallelization
  - Fine grid operator has plenty of grid-level parallelism
    - E.g., 16x16x16x16 = 65536 lattice sites
  - Coarse grid operator has diminishing grid-level parallelism
    - first coarse grid 4x4x4x4= 256 lattice sites
    - second coarse grid 2x2x2x2 = 16 lattice sites
- Current GPUs have up to >5000 processing cores
- Need to consider finer-grained parallelization
  - Increase parallelism to use all GPU resources
  - Load balancing

SOURCE OF PARALLELISM



1. Grid parallelism

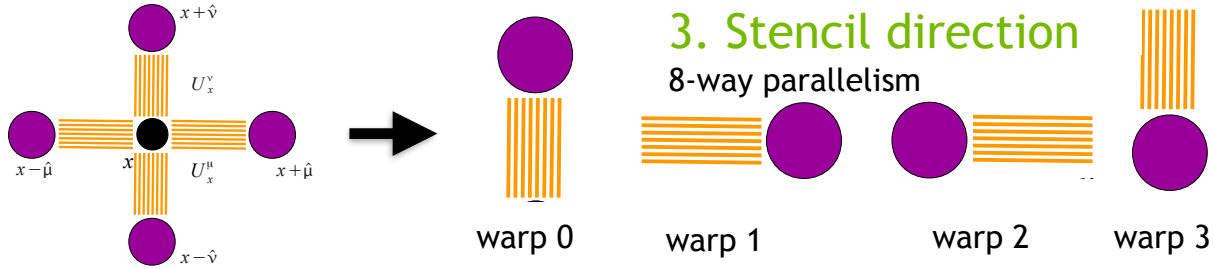
Volume of threads

thread y index

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} += \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

2. Link matrix-vector partitioning

2 N_{vec}-way parallelism
(spin * color)



3. Stencil direction

8-way parallelism

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \Rightarrow \begin{pmatrix} a_{00} & a_{01} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} + \begin{pmatrix} a_{02} & a_{03} \end{pmatrix} \begin{pmatrix} b_2 \\ b_3 \end{pmatrix}$$

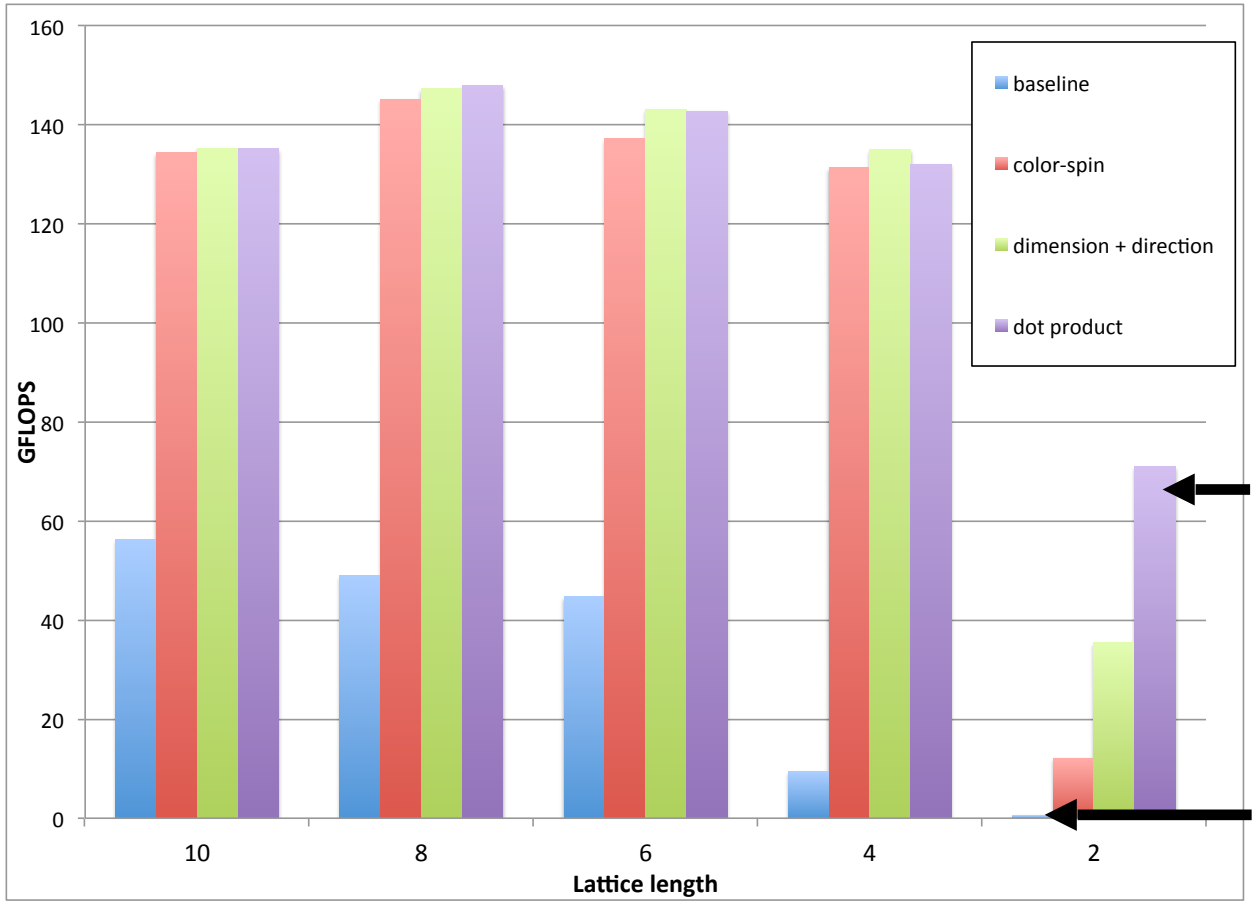
4. Dot-product partitioning

4-way parallelism

Clark, Joó, Strelchenko,
Cheng, Gambhir, Brower (2016)

COARSE GRID OPERATOR PERFORMANCE

Tesla K20X (Titan), FP32, N_{vec} = 24



24,576-way parallel

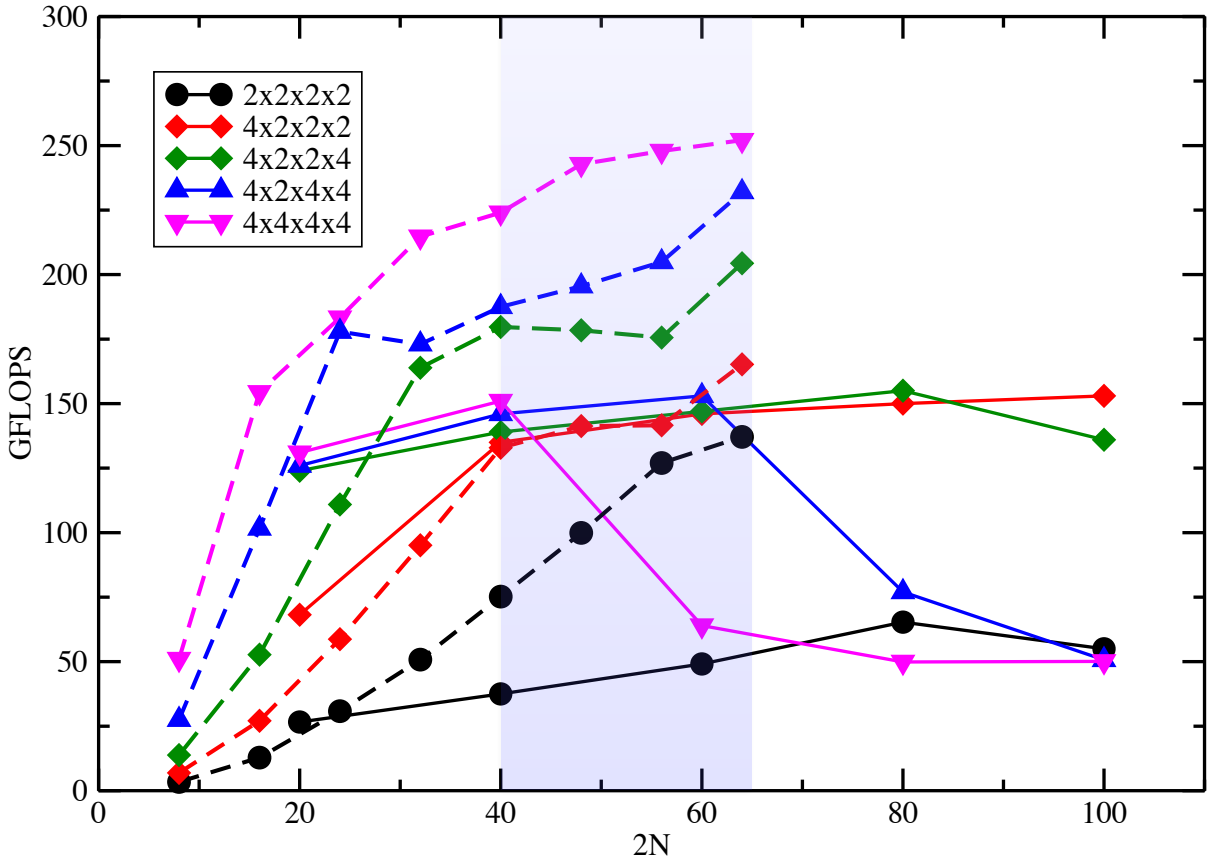
16-way parallel

NVIDIA

Clark, Joó, Strelchenko,
Cheng, Gambhir, Brower (2016)

COARSE GRID OPERATOR PERFORMANCE

8-core Haswell 2.4 GHz (solid line) vs M6000 (dashed lined), FP32

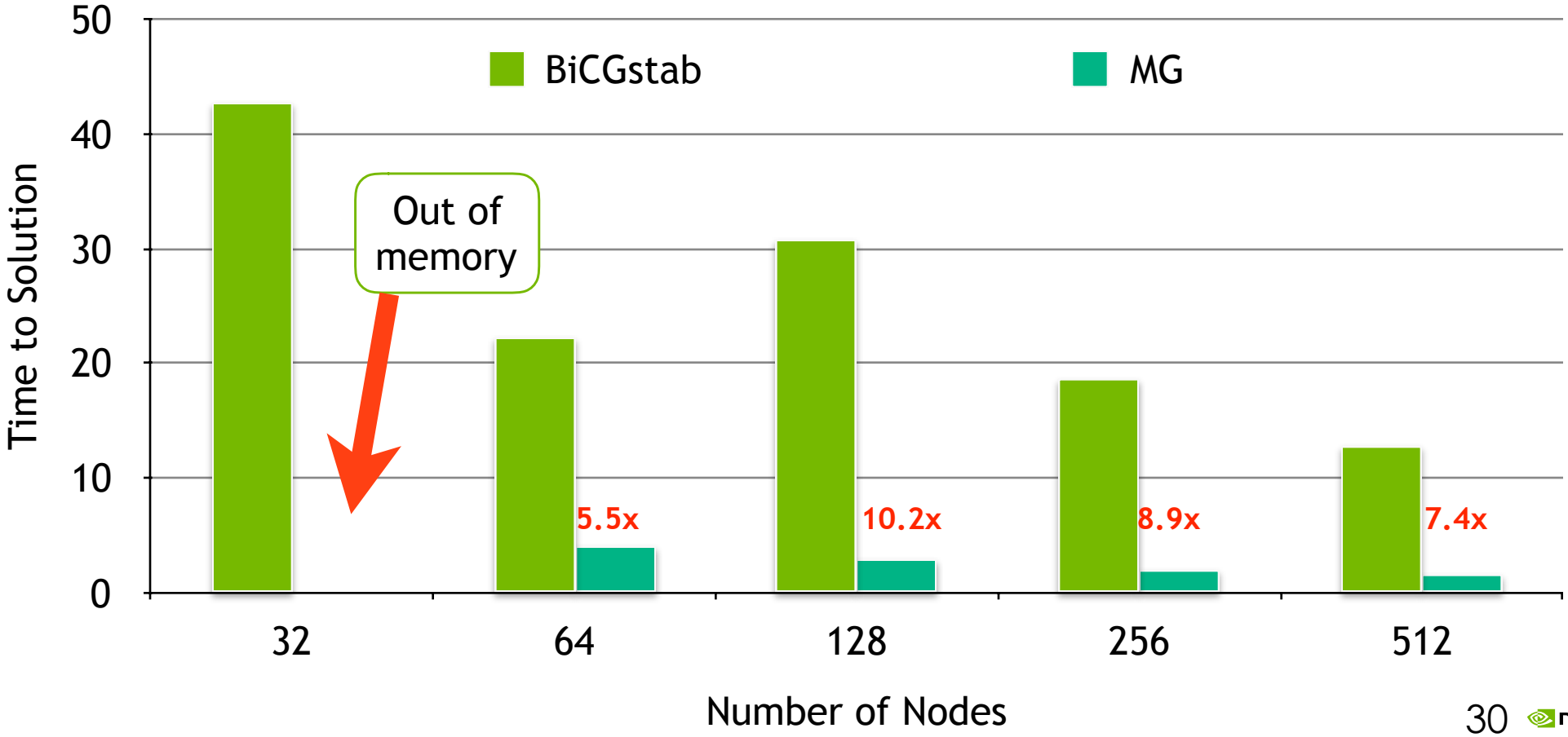


- Autotuner finds optimum degree of parallelization
  - Larger grids favor less fine grained
  - Coarse grids favor most fine grained
- GPU is nearly always faster than CPU
- Expect in future that coarse grids will favor CPUs
- For now, use GPU exclusively

Clark, Joó, Strelchenko,
Cheng, Gambhir, Brower (2016)

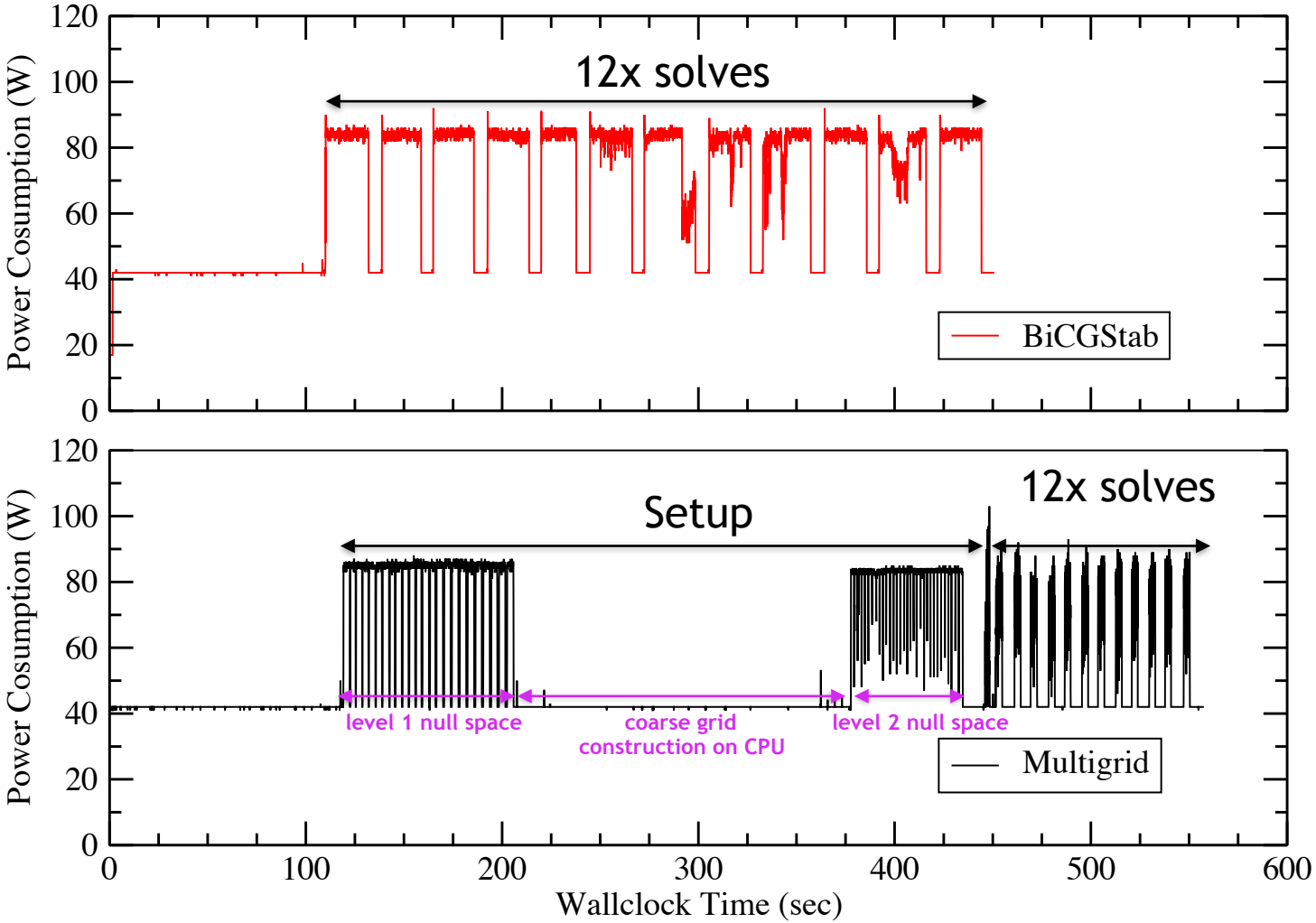
MULTIGRID VERSUS BICGSTAB

Wilson-clover, Strong scaling on Titan (K20X), V = 64^3x128, m_π = 197 MeV



Credit to Don Maxwell @ OLCF for helping with Power measurements on Titan

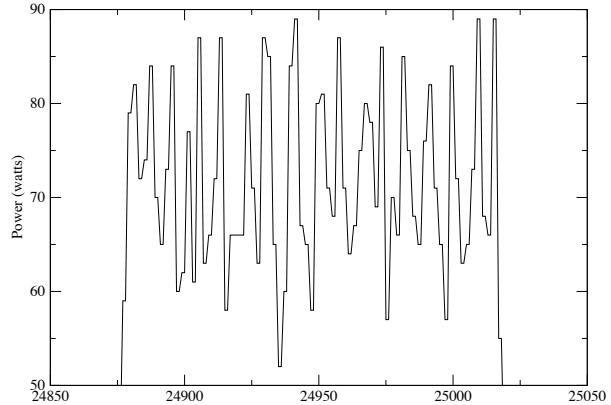
~~POWER EFFICIENCY~~ PROFILE



BiCGstab average power
~ 83 watts per GPU

MG average power
~ 72 watts per GPU

MG consumes less power and 10x faster



16-BIT FIXED-POINT FOR COARSE GRIDS

QUDA uses 16-bit precision as a memory traffic reduction strategy

Computation always done in FP32

Actually uses “block float” format

Uses 16-bit fixed point per grid point with single float to normalize

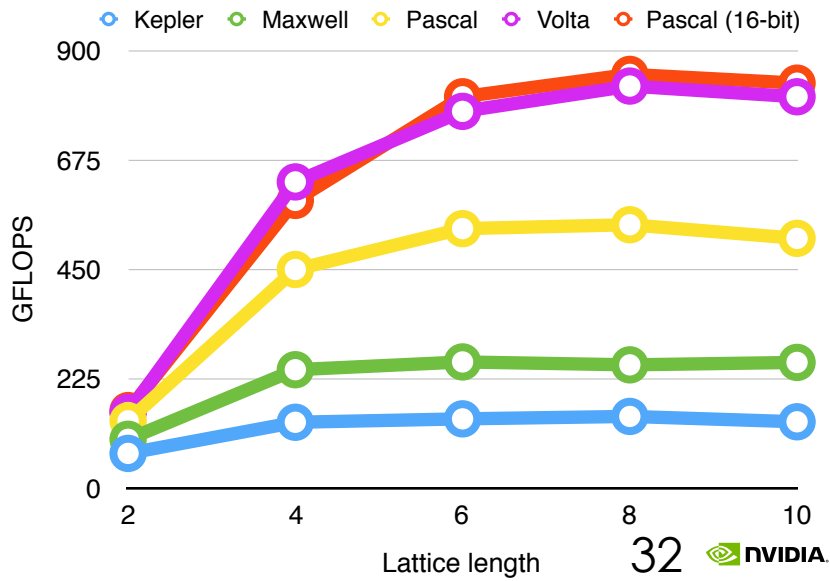
CG / BiCGStab has 5-10% hit in iteration count for overall ~1.7x

With multigrid, store everything in 16-bit fixed point that makes sense

➡ null-space vectors — already block orthonormal

➡ coarse-link matrices — estimate max element to set scale, e.g., $|U\,V|_{max} \sim |U|_{max}\,|V|_{max}$

Absolutely zero effect on multigrid convergence



HMC AND MULTIGRID

STARTING POINT

HMC-MG Team
Clark, Joó, Wagner, Weinberg
(+ Winter and Yoon)

2+1 flavour Wilson-clover fermions with Stout improvement running on Chroma

Physical parameters:

V = 64^3x128, ml=-0.2416, ms=-0.2050, a~0.09 fm, m_π~170 MeV

Performance measured relative to prior pre-MG optimal approach

- Essentially the algorithm that has been run on Titan 2012-2016
- 3 Hasenbusch ratios, with heaviest Hasenbusch mass = strange quark
  - Represented as 1 + 1 + 1 using multi-shift CG (pure double precision)
- 2-flavour solves: GCR + Additive Schwarz preconditioner (mixed precision)
- All fermions on the same time scale using MN5FV 4^{th} order integrator

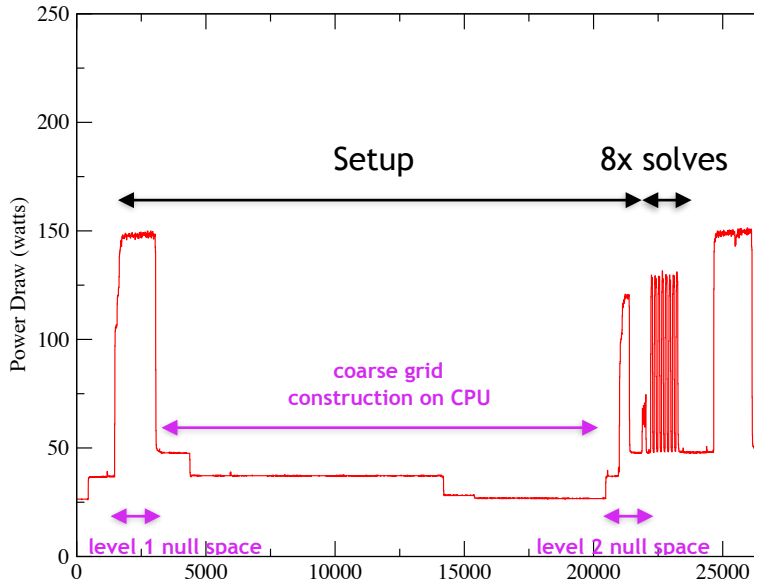
Benchmark Time: 1024 nodes of Titan = 4006 seconds

WHY HMC + MULTIGRID?

HMC typically dominated by solving the Dirac equation

However, much more challenging than analysis

- Few solves per linear system
- Can be bound by heavy solves (c.f. Hasenbusch mass preconditioning)



Build on top of pre-existing QUDA MG (arXiv:1612.07873)

Multigrid setup must run at speed of light since little scope for amortizing

Reuse and evolve multigrid setup where possible

MULTIGRID SETUP

Generate null vectors (BiCGStab, CG, etc. acting on homogenous system)

$$Ax_k = 0, \;\; k = 1 \ldots N, \quad \rightarrow \quad B = (x_1 x_2 \ldots x_n)$$

Block Orthogonalization of basis set

$$B^i = Q^i R^i = V^i B_c^i \qquad B = \sum_i B^i, V = \sum_i V^i$$

QR decomposition over each block

Coarse-link construction (Galerkin projection $D_c = P^\dagger D P$)

$$D_c = -\sum_\mu \left[Y_\mu^{-\mathrm{f}}(\hat{x}) + Y_\mu^{+\mathrm{b}\dagger}(\hat{x} - \mu) \right] + X\delta_{\hat{x},\hat{y}}$$

$$Y_\mu^{+\mathrm{b}}(\hat{x}) = \sum_{x \in \hat{x}} V^\dagger(x) P^{+\mu} U_\mu(x) A^{-1}(y) V(y) \delta_{x,y+\mu} \delta_{\hat{x},\hat{y}+\mu}$$ “backward link”

$$Y_\mu^{-\mathrm{f}}(\hat{x}) = \sum_{x \in \hat{x}} V^\dagger(x) A^{-1}(x) P^{-\mu} U_\mu(x) V(y) \delta_{x,y+\mu} \delta_{\hat{x},\hat{y}+\mu}$$ “forward link”

$$X(\hat{x}) = \sum_{x \in \hat{x},\mu} V^\dagger(x) \left(P^{+\mu} U_\mu(x) A^{-1}(y) + A^{-1}(x) P^{-\mu} U_\mu(x) \right) V(y) \delta_{x,y+\mu} \delta_{\hat{x},\hat{y}}$$ “coarse clover”

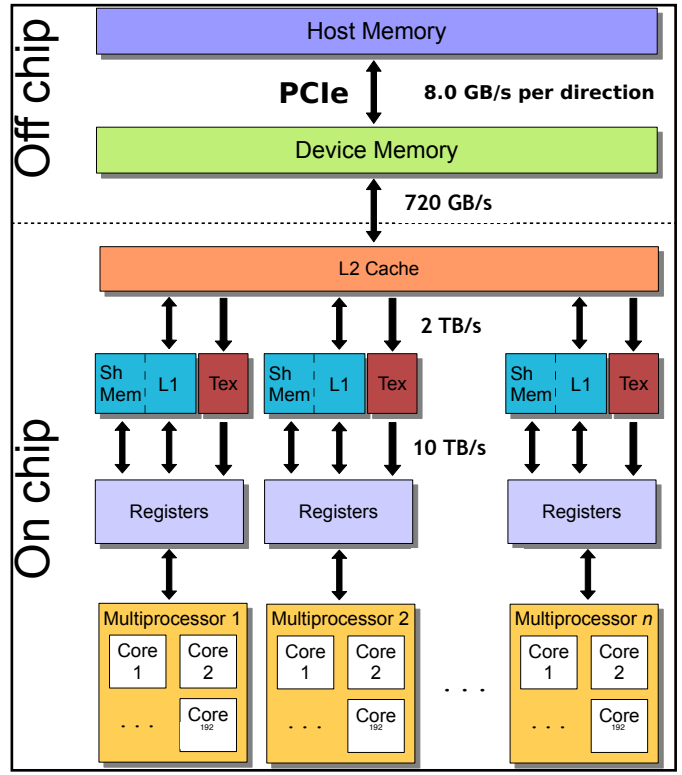
BLOCK ORTHOGONALIZATION

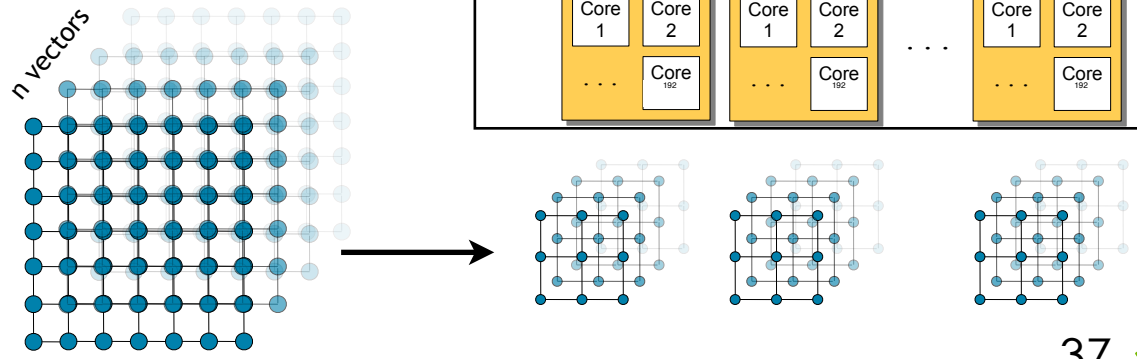
Forms the block orthonormal basis upon which we construct the coarse grid

QR on the set of null-space vectors within each multigrid aggregate

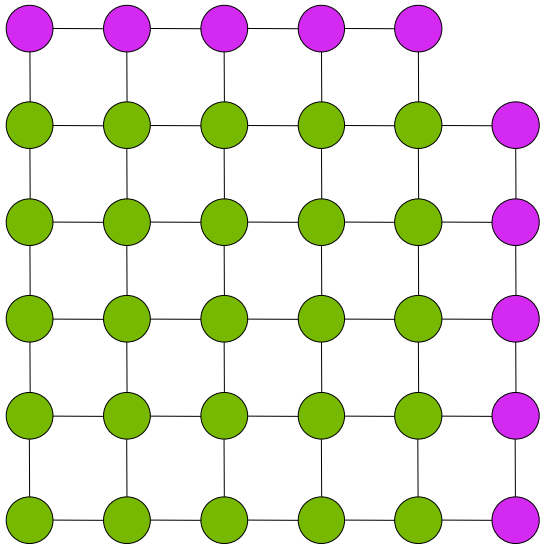
Assign each multigrid aggregate to a CUDA thread block

- All reductions are therefore local to a CUDA thread block
- Do the full block orthonormalization in a single kernel
- Minimizes total memory traffic





COARSE-LINK CONSTRUCTION



Employ fine-grained parallelization

- fine-grid geometry
- coarse-grid color

Each thread computes its assigned matrix elements

Atomically update the relevant coarse link field depending on thread location

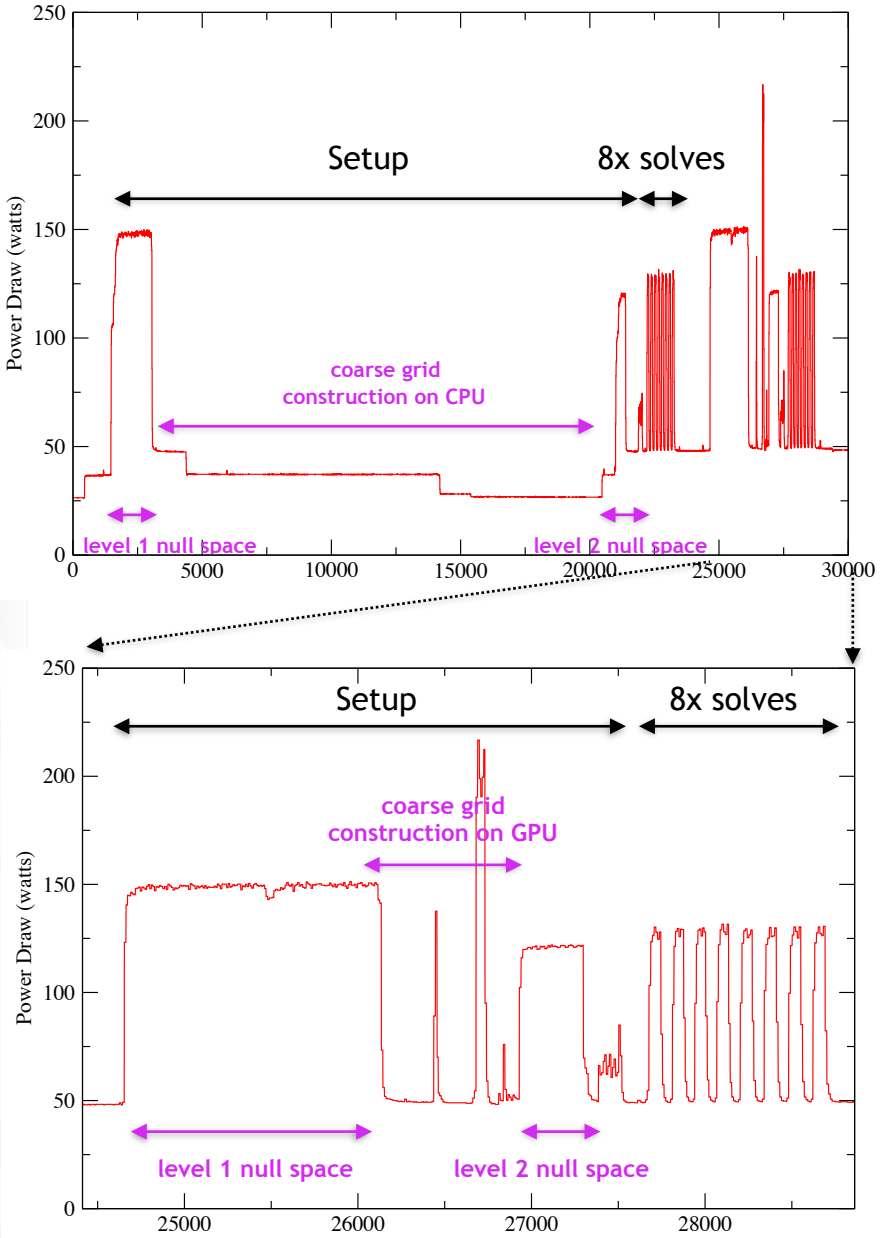
$$Y = \sum$$ 

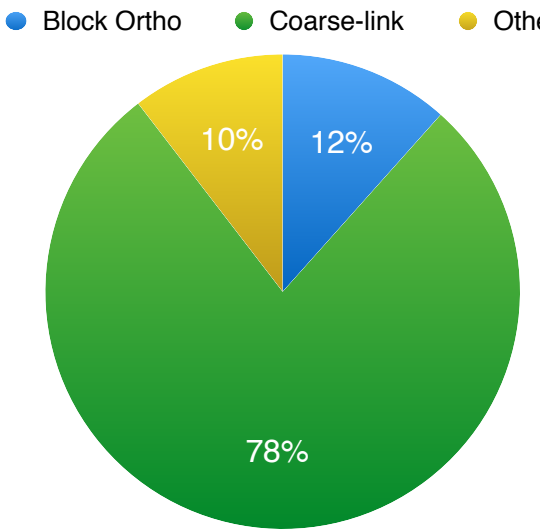
$$X = \sum$$ 

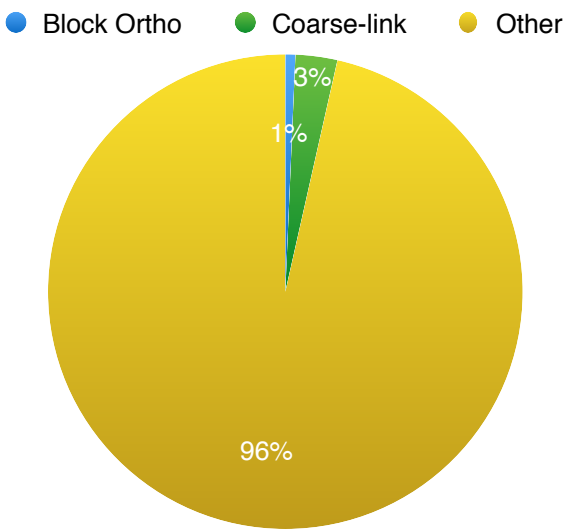
Atomic update is done in 32-bit integers

Finally, neighbour exchange boundary link elements

RESULTS







Null-space finding now dominates the setup process

Coarse-link construction runs at ~1 TFLOPS (P100)

Now dominated by null-space finding

This is a multi-RHS problem

HMC MULTIGRID ALGORITHM

Use the same null space for all masses (setup run on lightest mass)

- We use CG to find null-space vectors

Evolve the null space vectors as the gauge field evolves (Lüscher 2007)

- Update the null space when the preconditioner degrades too much on lightest mass

Parameters to tune

- Refresh threshold: at what point do we refresh the null space?
- Refresh iterations: how much work do we do when refreshing?

OPTIMIZATION AND TUNING STEPS

(far from exclusive)

Replace GCR+DD with GCR-MG

- Made Hasenbusch terms cheaper so add extra Hasenbsuch term and retuned
- Put heaviest fermion doublet onto the fine (gauge) time scale

Optimize mixed-precision multigrid method:

- 16-bit precision wherever it makes sense (null space, coarse link variables, halo exchange)

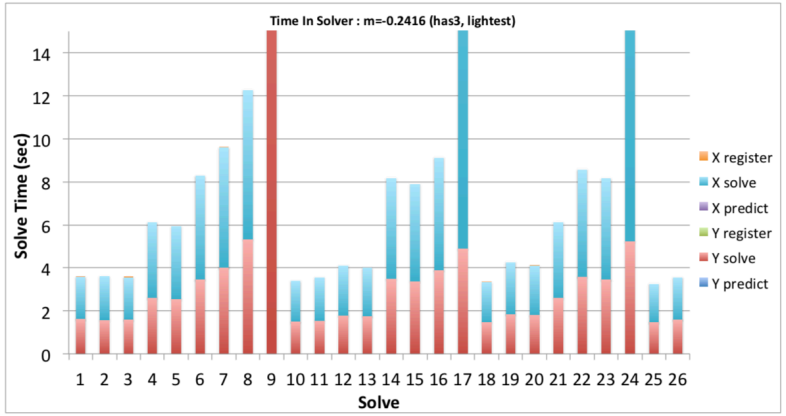
Volta 4x faster than Pascal for key setup routines: use multigrid for all 2-flavour solves

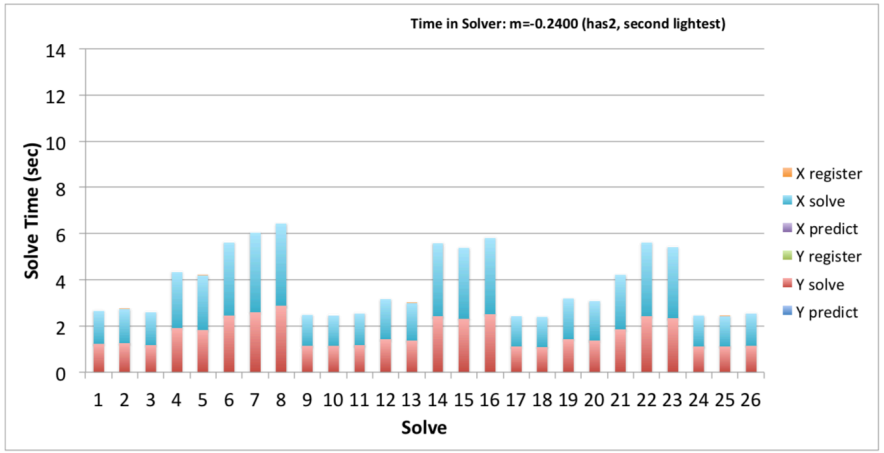
Replaced MN5FV integrator with Force Gradient integrator (Boram Yoon's Chroma implementation), tuned number of steps

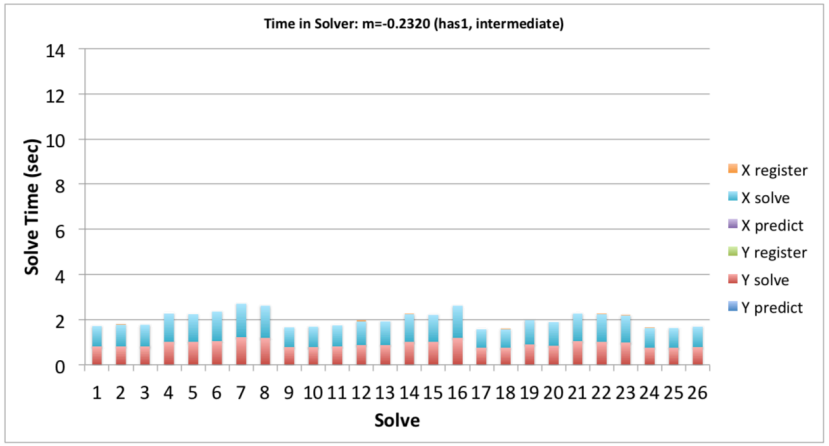
Multi-shift CG is expensive (no multigrid - yet…)

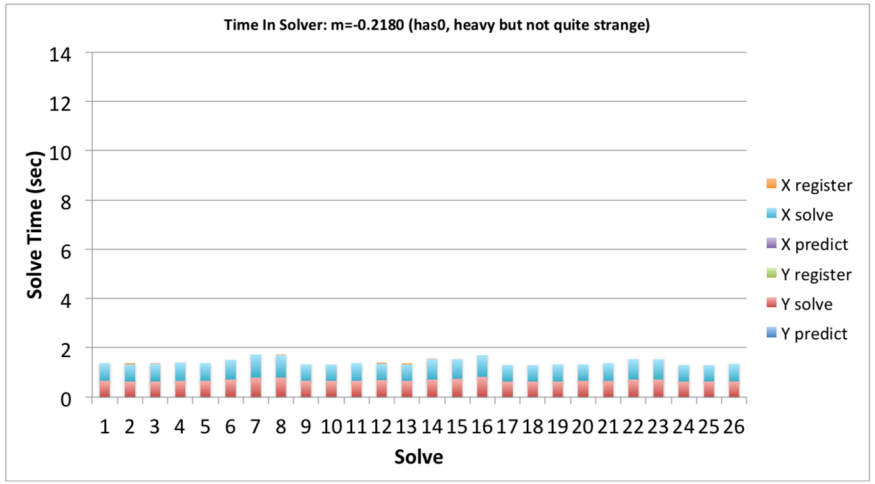
- Replace pure fp64 multi-shift CG with mixed-precision multi-shift CG and refinement: 1.5x faster

NULL-SPACE EVOLUTION

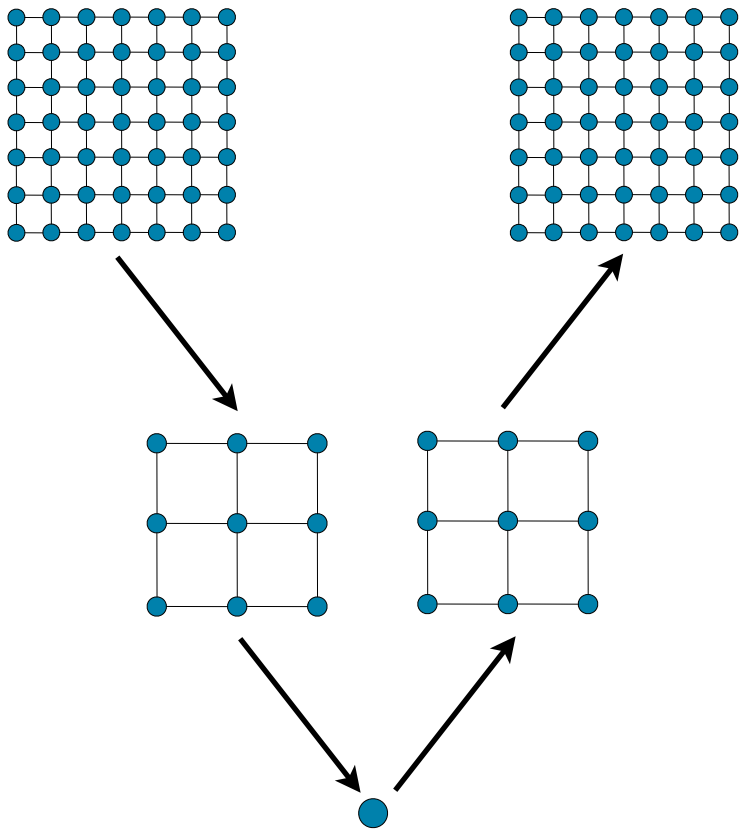








GLOBAL SYNCHRONIZATIONS IN LQCD MG



Non-Hermitian system

- No guarantee of convergence
- Use a K-cycle for solver stability

GCR solver deployed at every level

- N(N+1)/2 reductions required

Use MR as a smoother

- N reductions required

Example: 24x24x24x64 Wilson lattice @ κ_{crit}

- MR(0,8) smoother with GCR coarse grid solver
- **980 reductions to reach convergence**

COMMUNICATION-AVOIDING GCR

source vector b, solution vector x

while (i<N) {

p_{i+1} <- A*p_i // build basis (N mat-vecs)

$q_i = p_{i+1}$

}

// minimize residual solving (one “blas-3” reduction)

$\psi = (q, q)^{-1} (q, b)$

// update solution vector (one “blas-2” kernel)

$x = \Sigma_k \psi_k p_k$

Similar to CA-GMRES (see Mark Hoemmen’s thesis)

GCR(N) uses modified Gram Schmidt to orthonormalize the basis at every step

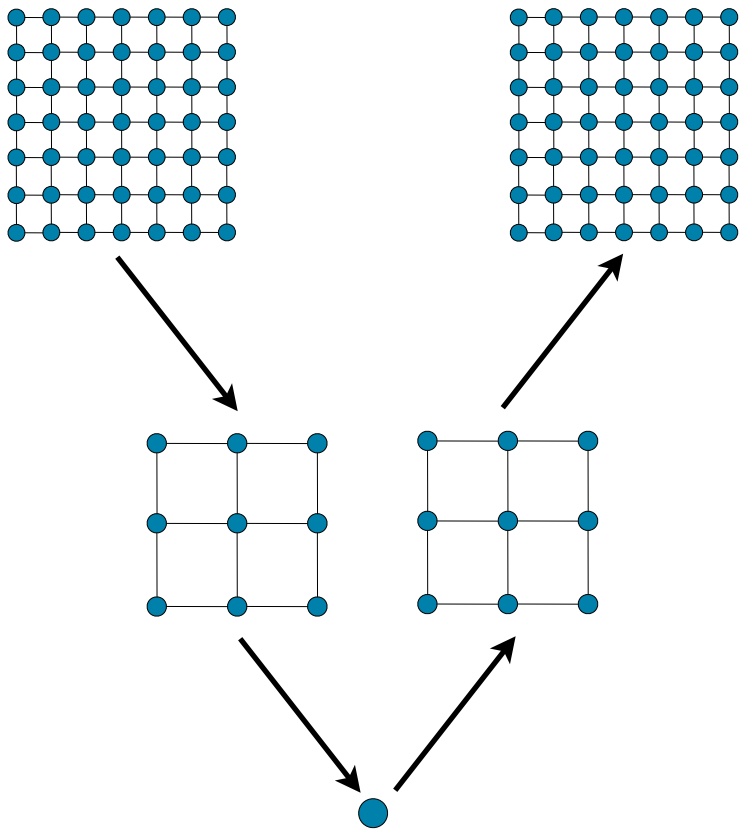
- Hence N(N-1)/2 reductions

Instead use classical Gram Schmidt and orthonormalize every N steps

- One reduction every N steps

Strong smoother than MR

GLOBAL SYNCHRONIZATIONS IN LQCD MG



Example: 24x24x24x64 Wilson lattice @ κ_{crit}

MR(0,8) smoother with GCR coarse grid solver

- 980 reductions to reach convergence

MR(0,8) smoother, with pipelined GCR

- 829 reductions to reach convergence

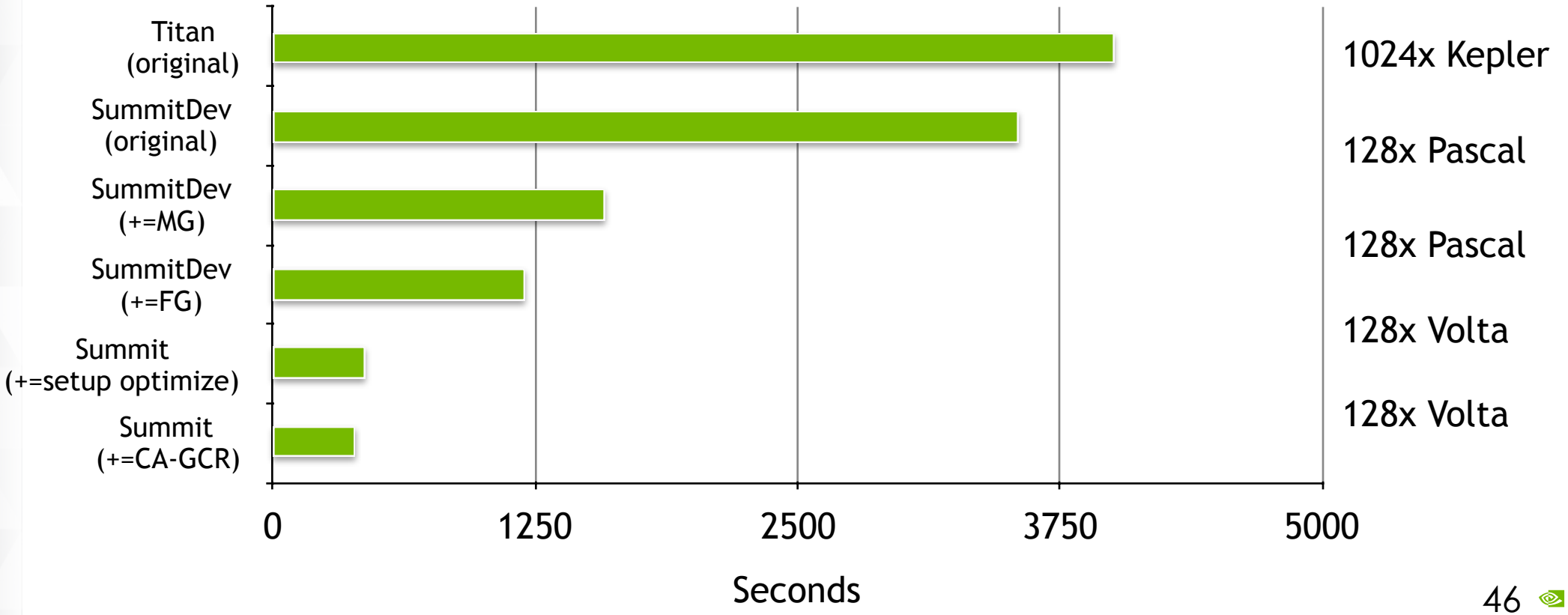
CA-GCR(0,8) for smoother and coarse-grid

- 153 reductions to reach convergence
- >6x reduction in reductions
- 20% faster on a single workstation

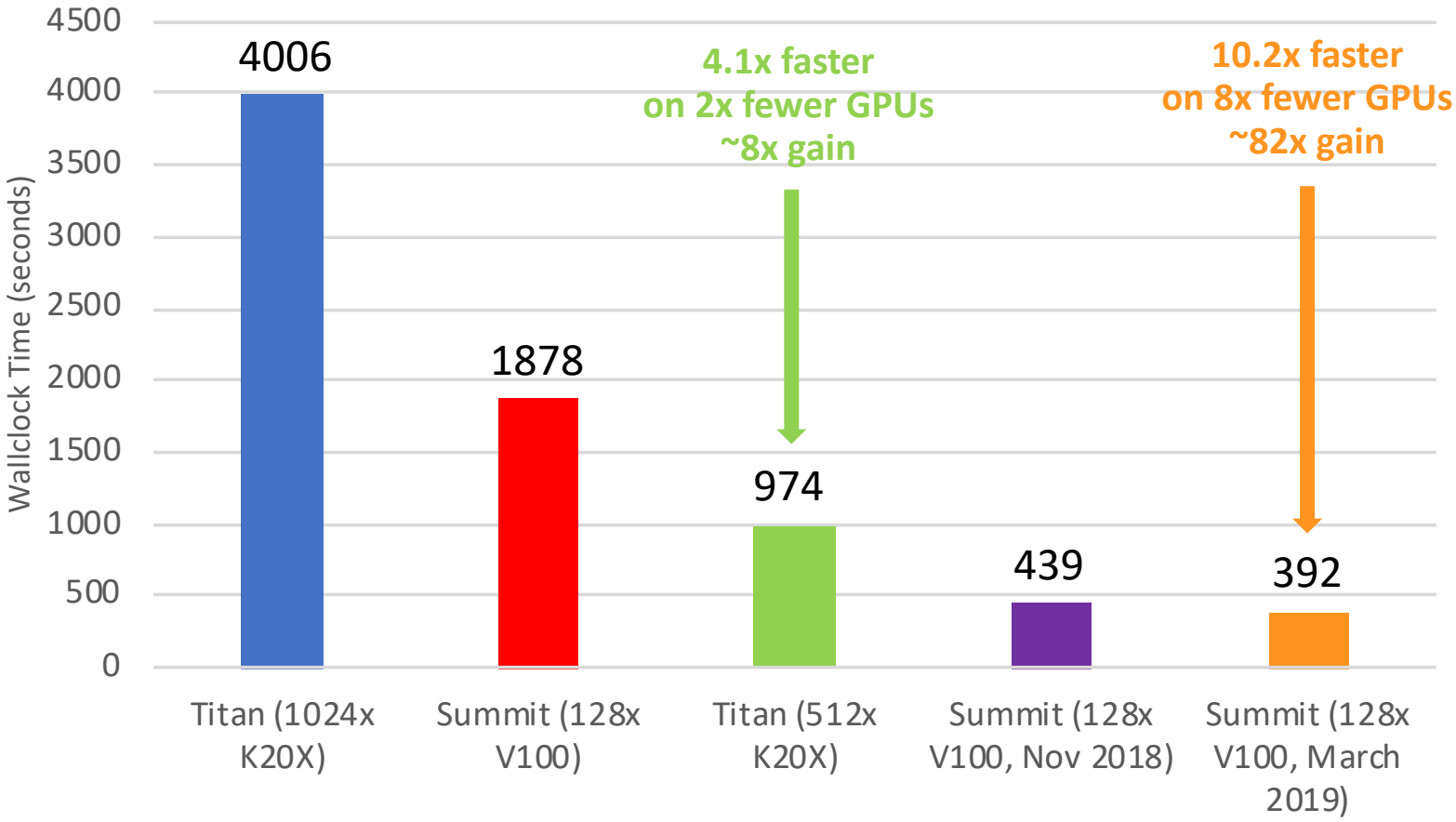
Reference $V=64^3x128$ problem

- Solver is 40% faster on Titan on 512 nodes

HMC SPEEDUP PROGRESSION



LATEST RESULTS



HMC MULTIGRID SUMMARY

2019 Chroma gauge generation close to 100x increase in throughput vs 2016

- MG solver, FG integrator, Titan -> Summit (Kepler to Volta)
- Speedup = machine x algorithm

Ongoing work to go beyond 100x

- Inability to coarsen past 2^4 coarse-grid per MPI process presenting hard limit on scaling
- Use multi-rhs null-space generation, e.g., 24x CG => 1x block CG on 24 rhs

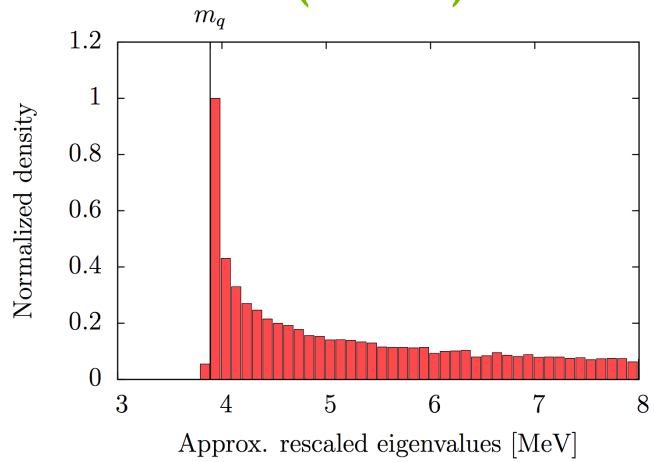

TWISTED-MASS
MULTIGRID

TWISTED-MASS MULTIGRID

Alexandrou, Bacchio, Finkenrath, Frommer, Kahl, Rottmann (2016)

Twisted-mass operator has pathological spectrum on the coarse grid

> "Although the dimension of the coarse grid operator is reduced, it can develop a large number of small eigenvalues close to μ. This can critically slow down the convergence of a standard Krylov solver to be used on the coarsest grid such that the time spend in the coarsest operator inversions dominates…"



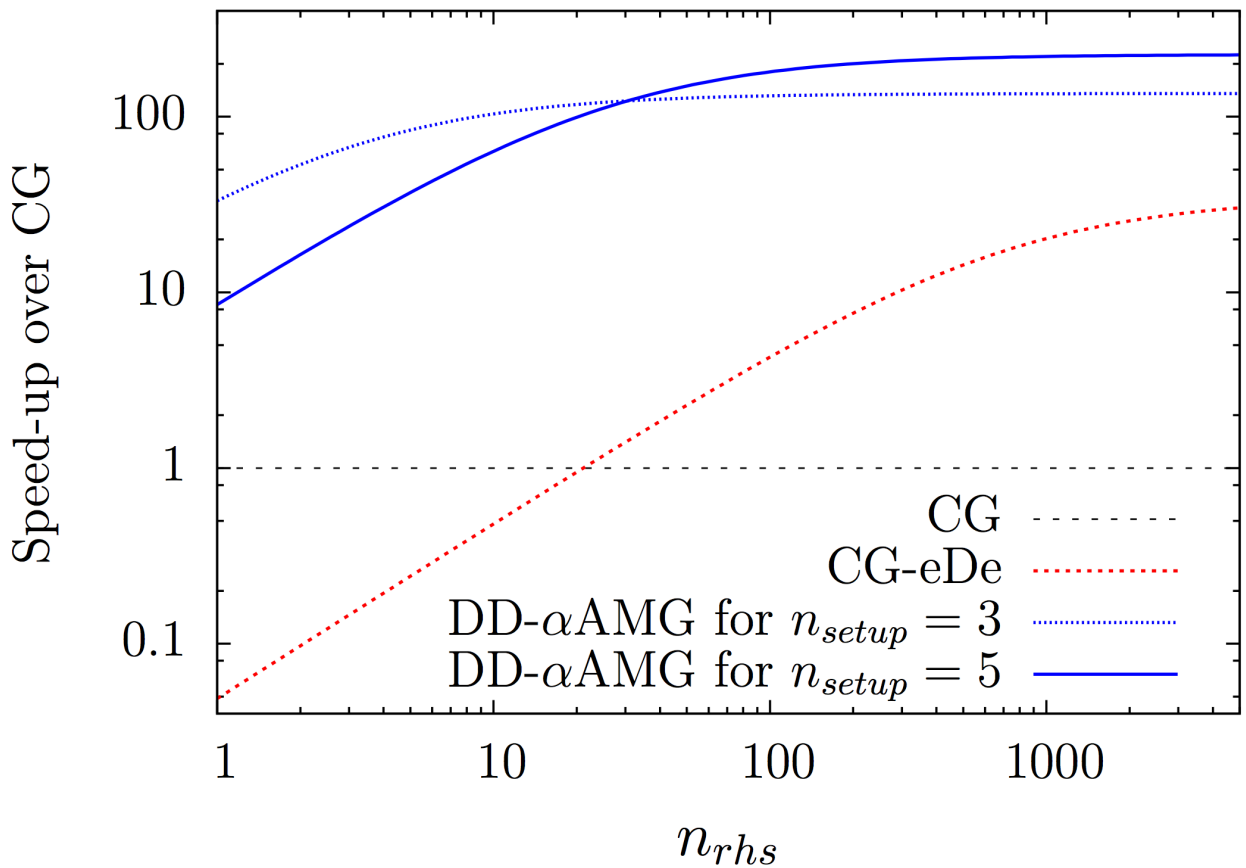
For Twisted-mass, solution is to add a fictitious heavy twist to the coarsest operator

$$D_c(\mu, \delta) = D_c + i\delta\mu \cdot \Gamma_{5,c}$$

Improves condition number and fast coarse-grid convergence restored

TWISTED-MASS MULTIGRID

Alexandrou, Bacchio, Finkenrath, Frommer, Kahl, Rottmann (2016)



DEFLATED MG AT THE PHYSICAL POINT

Clark, Howarth and Weinberg

"μ-scaling", while recovering a viable solver, impacts the quality of MG convergence

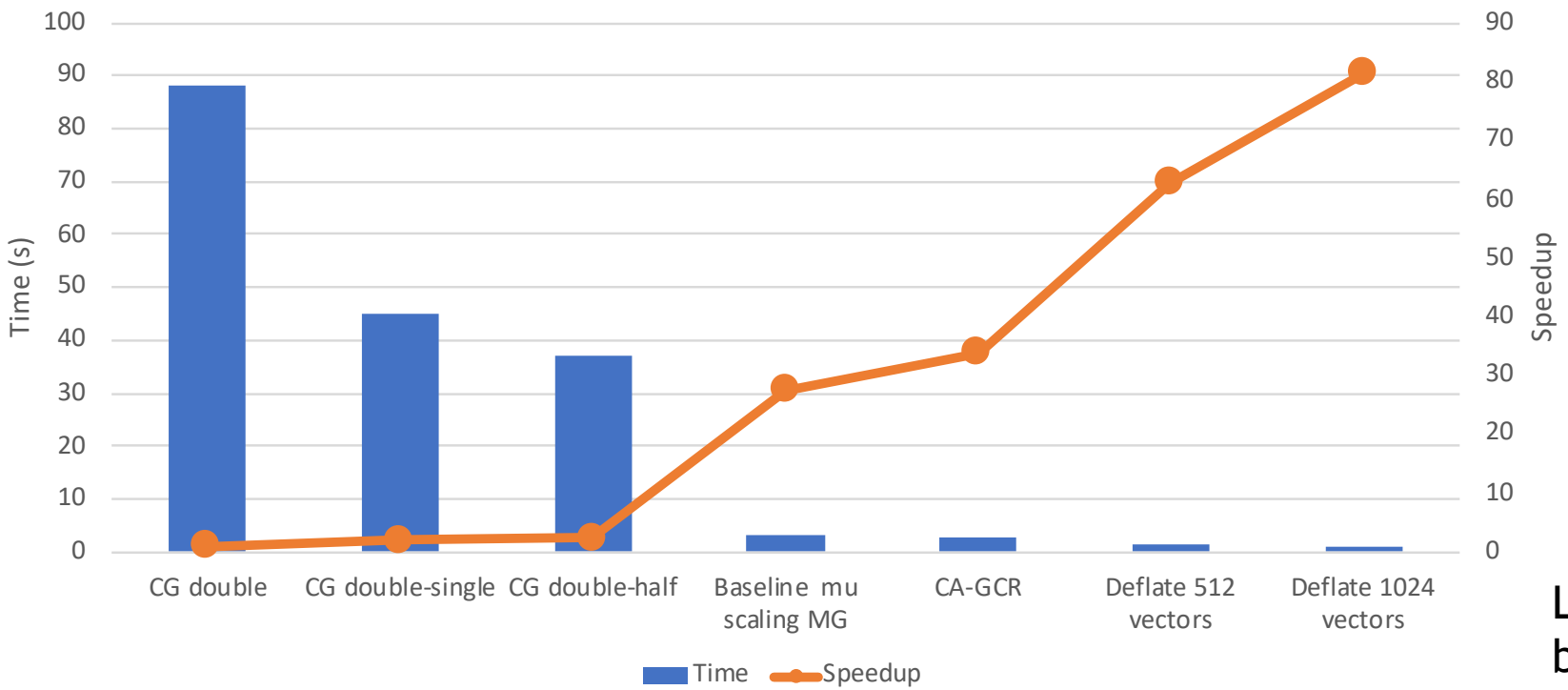
Realized that coarse-grid operator actually becomes indefinite
- Lowest eigenvalues can drift over to negative real on coarse grid
- **Pathological spectrum on the coarse grid**

Instead we deflate the coarse grid operator
- Remove troublesome modes directly
- Coarse-grid is small, so cost of deflation is negligible

Recover optimal MG convergence and a 3x speedup over "μ-scaling"

DEFLATED MG AT THE PHYSICAL POINT

Clark, Howarth and Weinberg



Lattice provided by ETMC

Time to solution is measured running QUDA on 4x DGX-1V nodes (32 GPUs) for solving the Twisted-mass + clover operator against a random source on a 64^3x128 lattice, $\beta = 1.778$, $\kappa = 0.139427$, $\mu = 0.000720$, solver tolerance 10^{-7}

(WILSON) FERMION SOLVERS

Combination of algorithm (multigrid) and machine (GPUs)

A single GPU can solve at 1 second per Wilson solve with local volume of $V=32^3x64$ per GPU

A single node (DGX-2) can solve solve $V=64^3x128$ at one second per solve

16 nodes of DGX-2 can solve $V=128^3x256$ at one second per solve

Fermion solvers are not the challenge they used to be (caveats unbound)

MULTIGRID FOR STAGGERED FERMIONS

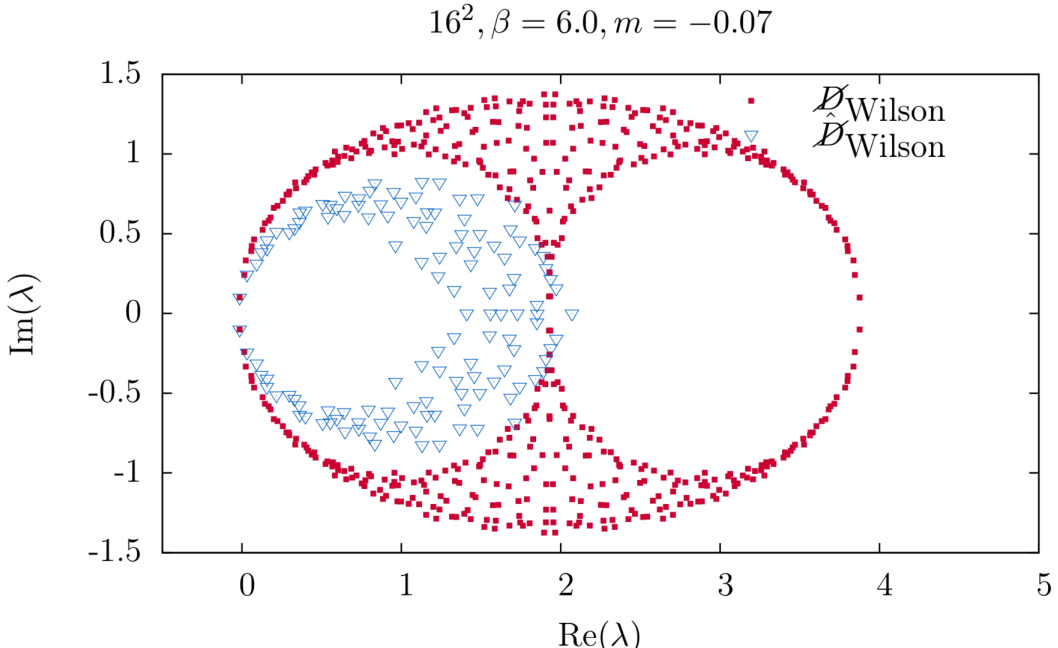
arXiv:1801.07823

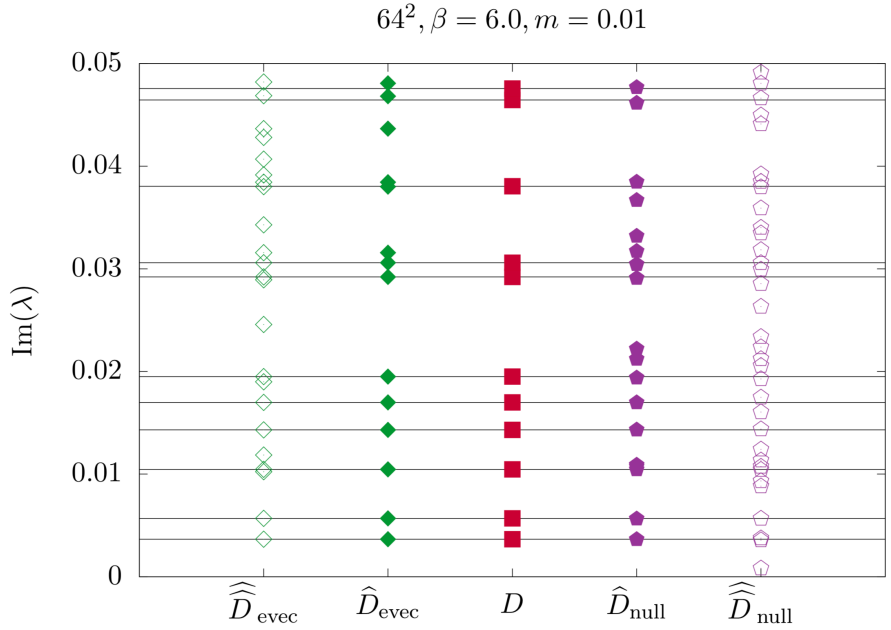
WHY IS STAGGERED MG HARD?

Brower, Clark, Howarth, Strelchenko, Weinberg

Naïve Galerkin projection does not work

- Spurious low modes on coarse grids
- System gets worse conditioned as we progressively coarsen





Compare to Wilson MG which preserves low modes with no cascade

arXiv:1801.07823

OUR SOLUTION

Staggered fermions distribute d fermions over 2^d sites

Each 2^d block is a supersite

or flavour representation or Kahler-Dirac block (arXiv:0509026 Dürr)

$$\mathcal{S} = b^4 \sum_{X,\mu} \bar{q}(X) \left[\nabla_\mu \left(\gamma_\mu \otimes 1 \right) - \frac{b}{2} \triangle_\mu \left(\gamma_5 \otimes \tau_\mu \tau_5 \right) + m \left(1 \otimes 1 \right) \right] q(X)$$

$$\equiv b^4 \sum_{X,\mu} \bar{q}(X) \left[\not{D} + m \right] q(X)$$

$$\left(\nabla_\mu q \right)(X) = \frac{q(X + b\hat{\mu}) - q(X - b\hat{\mu})}{2b}$$

$$\left(\triangle_\mu q \right)(X) = \frac{q(X + b\hat{\mu}) - 2q(X) + q(X - b\hat{\mu})}{b^2}$$

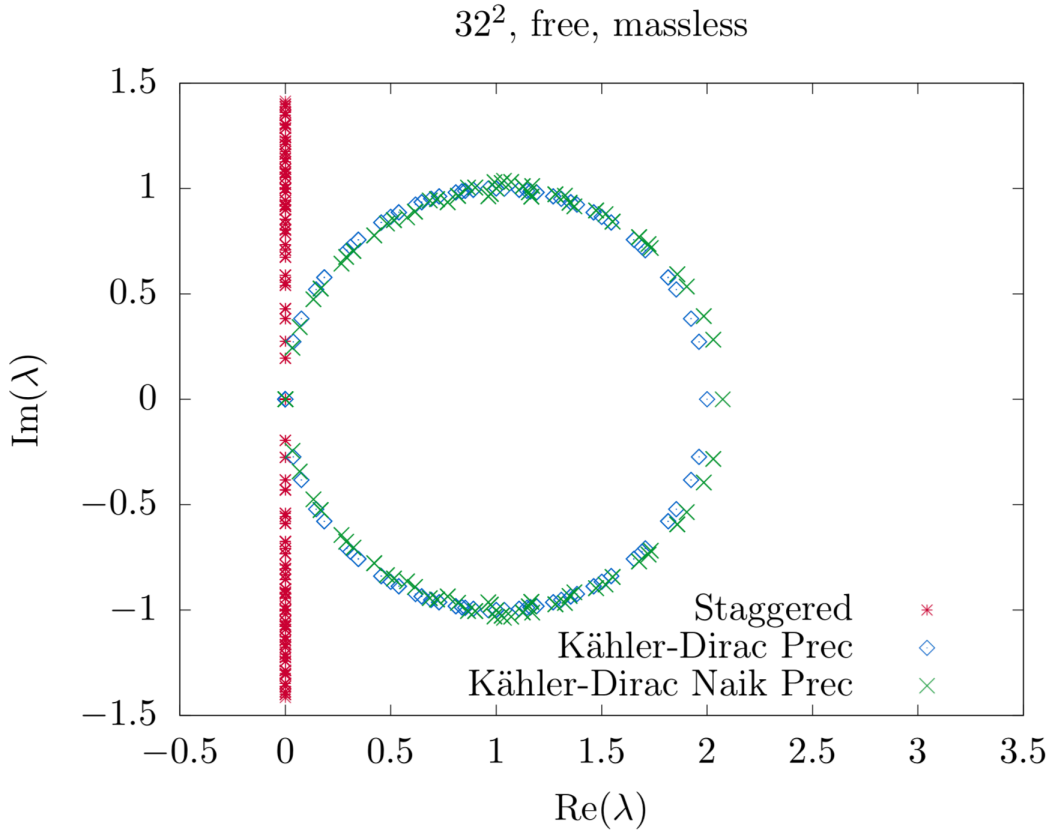
arXiv:1801.07823

OUR SOLUTION

Transform into Kahler-Dirac form through unitary transformation

$$\begin{pmatrix} m & 0 & -\frac{1}{2}U_x(2\vec{n}) & -\frac{1}{2}U_y(2\vec{n}) \\ 0 & m & -\frac{1}{2}U_y^\dagger(2\vec{n}+\hat{x}) & \frac{1}{2}U_x^\dagger(2\vec{n}+\hat{y}) \\ \frac{1}{2}U_x^\dagger(2\vec{n}) & \frac{1}{2}U_y(2\vec{n}+\hat{x}) & m & 0 \\ \frac{1}{2}U_y^\dagger(2\vec{n}) & -\frac{1}{2}U_x(2\vec{n}+\hat{y}) & 0 & m \end{pmatrix}$$

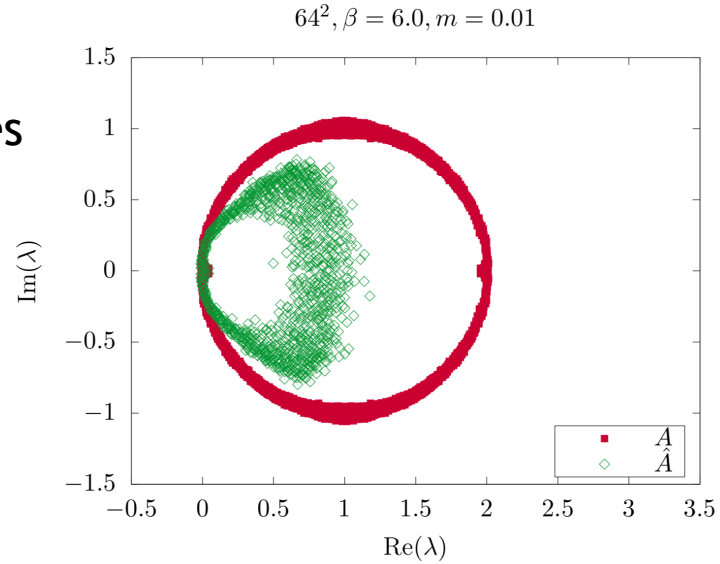
“Precondition” the staggered operator by the Kahler-Dirac block

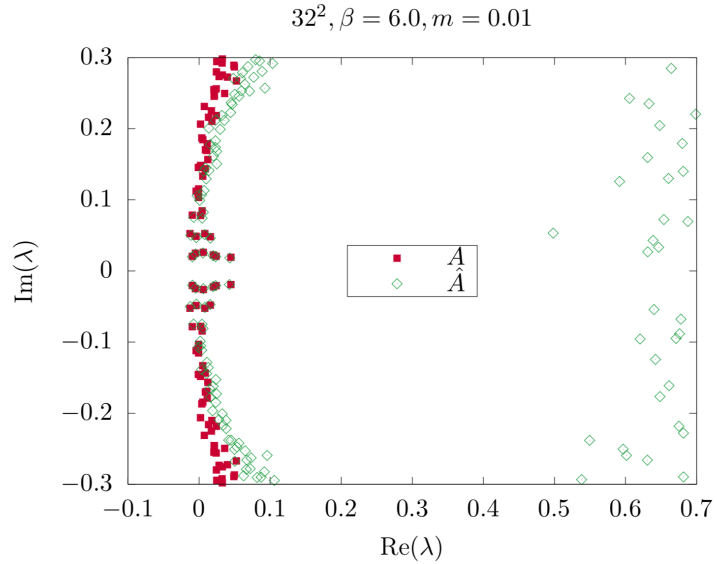


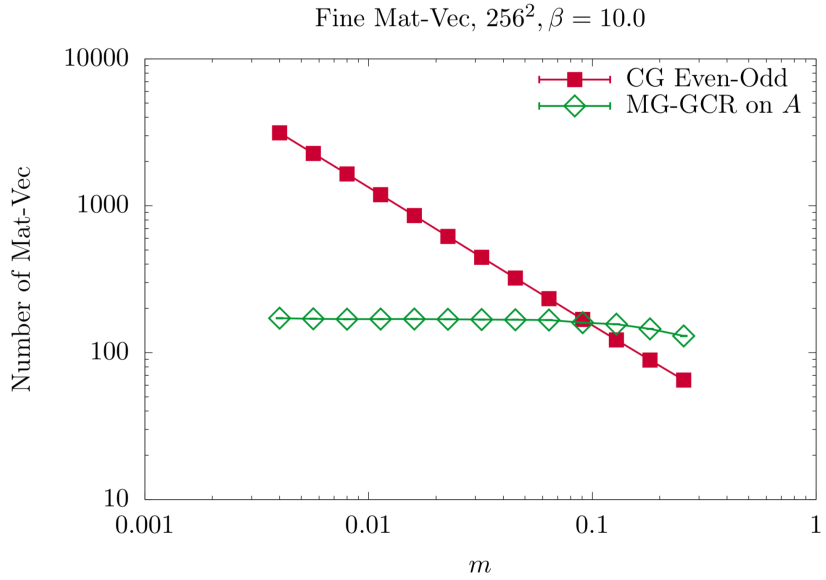
NVIDIA

arXiv:1801.07823

No spurious low modes as we coarsen







Removal of critical slowing down

GOING TO 4D AND HISQ FERMIONS

Block-preconditioned operator is no longer an exact circle

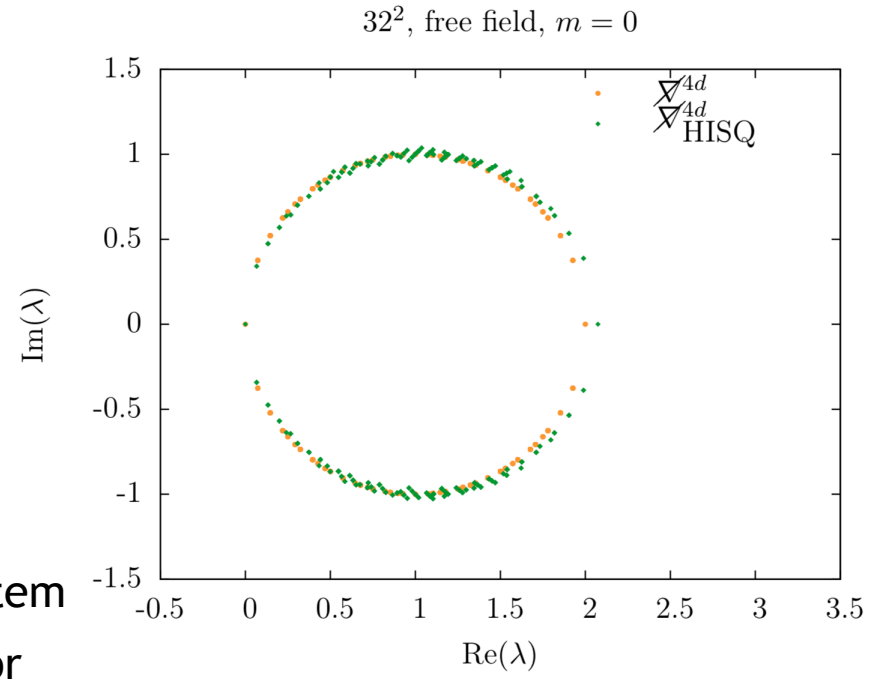
Prescription is *almost* identical to 2-d method

Drop Naik contribution from block preconditioner

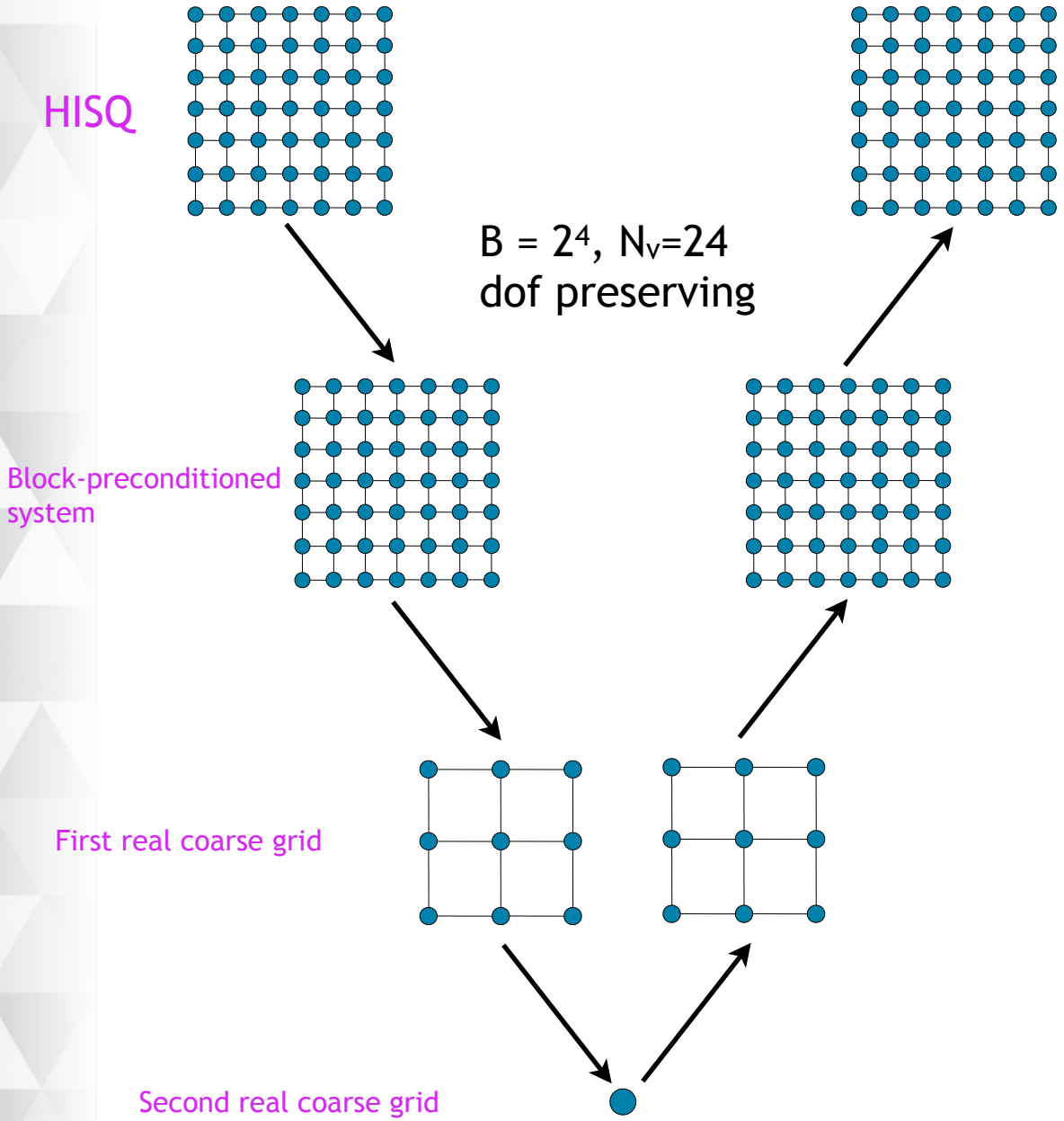
- No longer a unitary transformation
- No longer an exact Schur complement

Iterate between HISQ operator and block-preconditioned system

- Effectively apply MG to fat-link truncated HISQ operator only



HISQ MG



Level 1: 3 dof per site.
Solver: GCR, tolerance 10^{-10}

Smoother: CA-GCR(0,8)

Staggered has 4-fold degeneracy

- Need ~4x null space vectors
- Much more memory intensive

Level 2: 48 dof per site.
Solver: GCR, tolerance 0.25, max 16 iterations
Operator: Left-block Schur, 16-bit precision

Smoother: CA-GCR(0,2)

Level 3: **128 dof per site.** Solver: GCR, tolerance 0.25, max 16 iterations
Operator: Left-block Schur, 16-bit precision

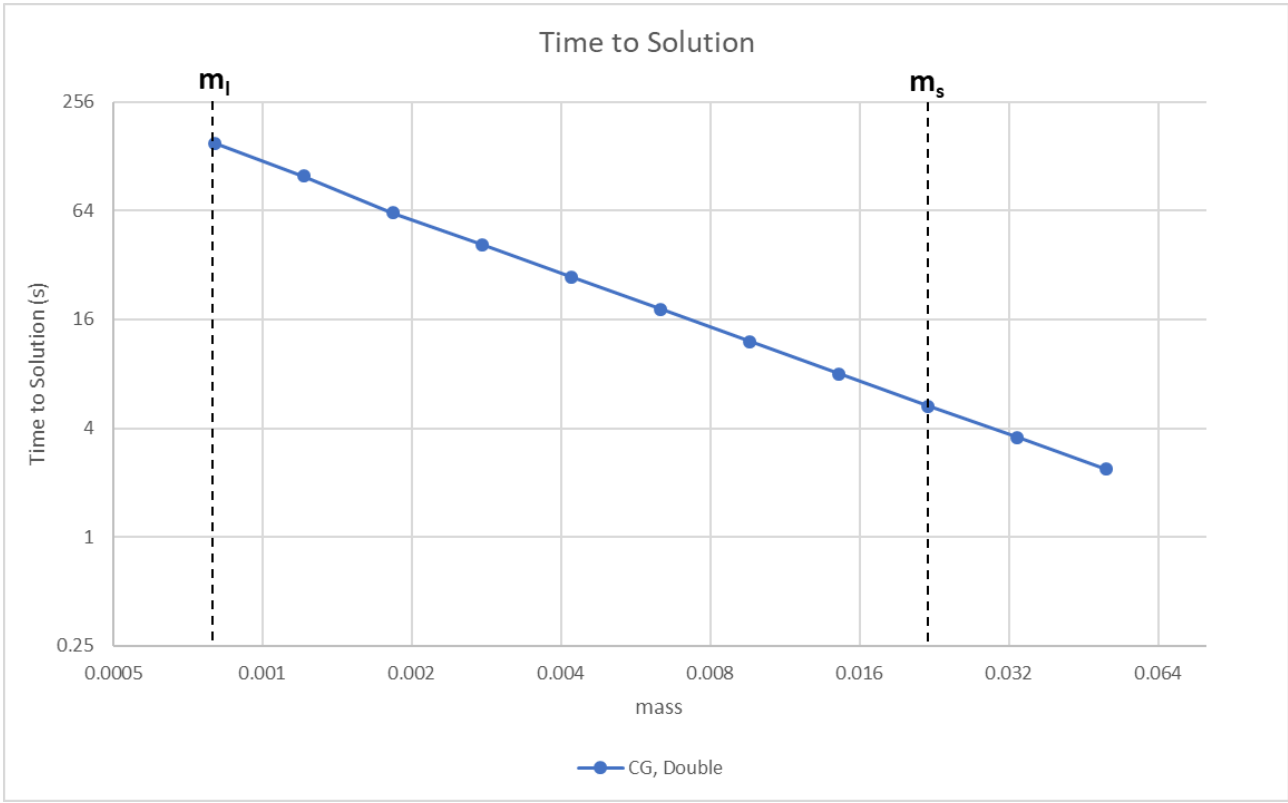
Smoother: CA-GCR(0,2)

Level 4: **192 dof per site.**
Solver: CA-GCR(16)
Operator: Left-block Schur, 16-bit precision

OLD SCHOOL: CG

Schur system: $\left(m^2 - D_{eo}^{HISQ} D_{oe}^{HISQ}\right)\overrightarrow{x_e} = m\overrightarrow{b_e} - D_{eo}^{HISQ}\overrightarrow{b_o}$ to tolerance $m10^{-10}$

Pure double precision solve, reconstruct-9 (long links can be encoded by 9 numbers)



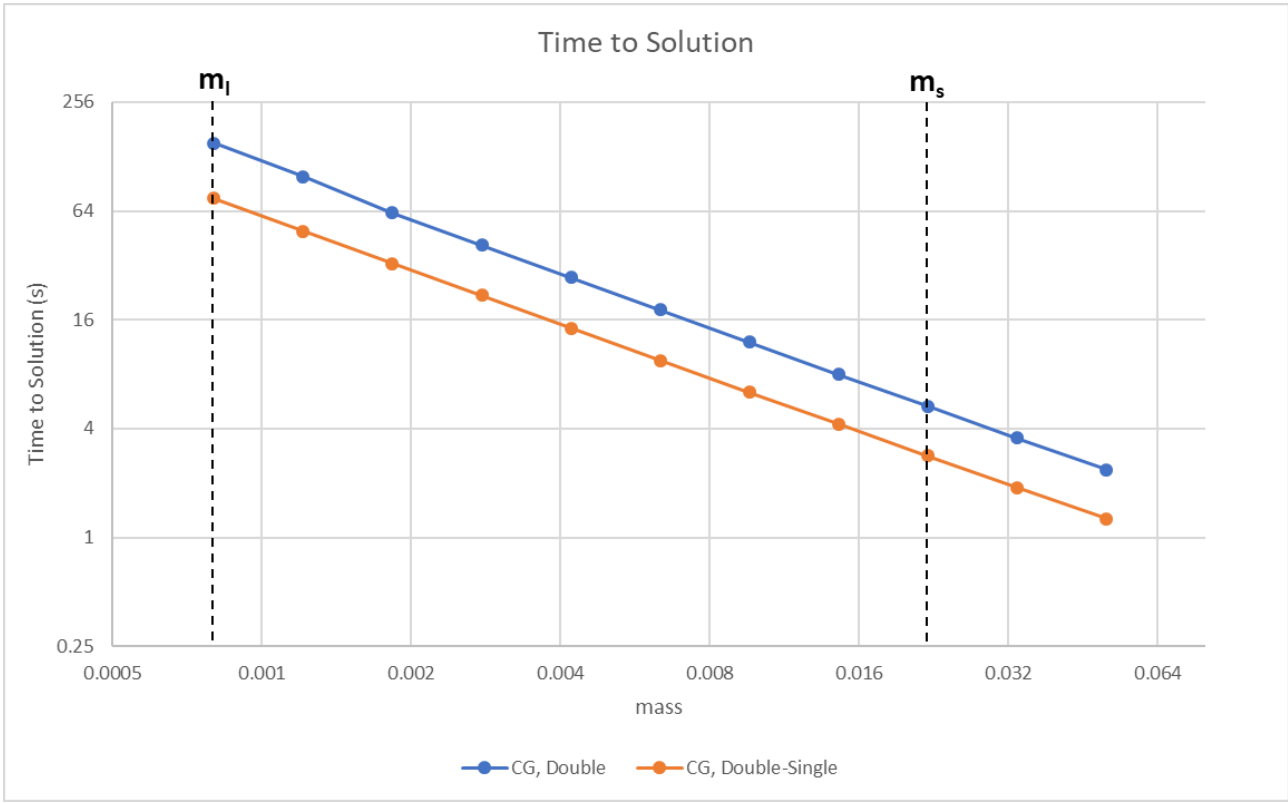
Thanks to MILC for lattices

72x Summit nodes, random source, V=96^3x192 lattice, $\beta = 6.72$, $a = 0.06$, $m_l = 0.0008$, $m_s = 0.022$, $||r|| = 10^{-10}$

OLD SCHOOL: CG

Schur system: $\left(m^2 - D_{eo}^{HISQ} D_{oe}^{HISQ}\right)\overrightarrow{x_e} = m\overrightarrow{b_e} - D_{eo}^{HISQ}\overrightarrow{b_o}$ to tolerance $m10^{-10}$

Use mixed precision instead: double-single

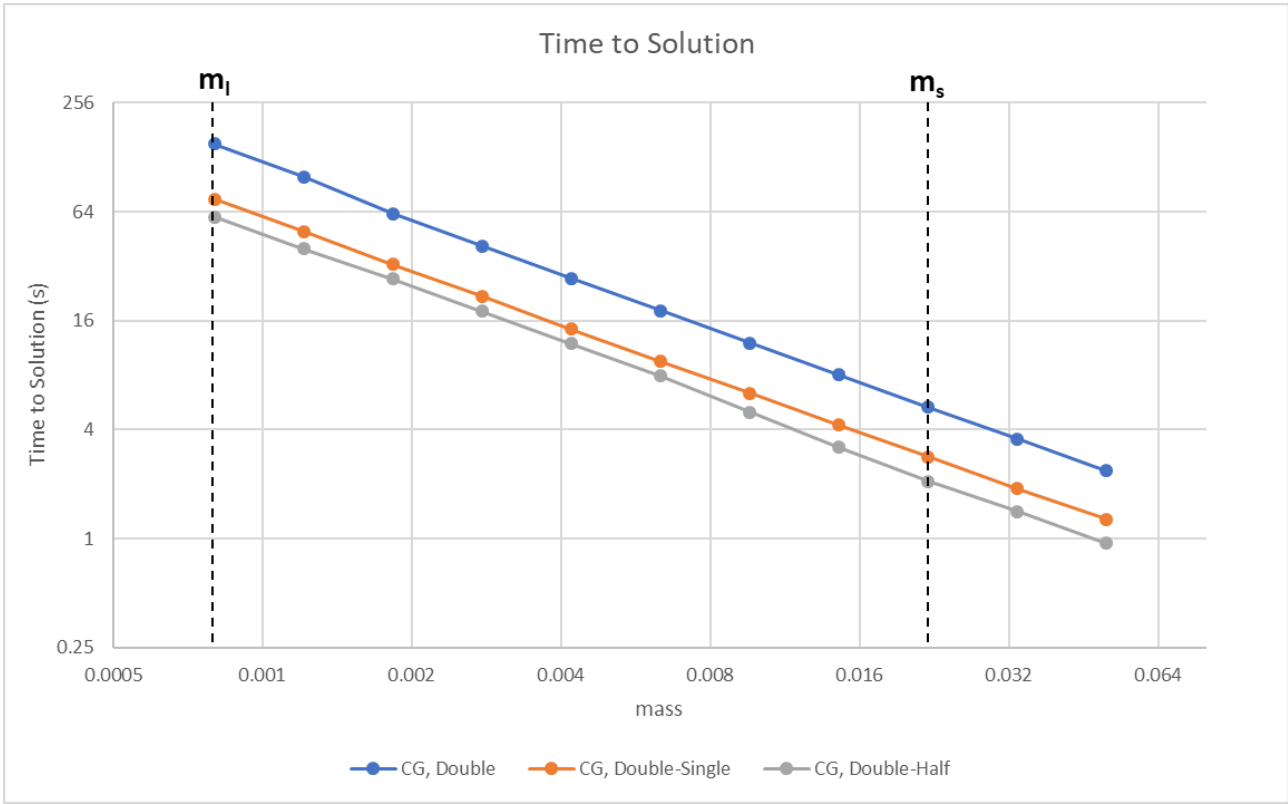


72x Summit nodes, random source, V=96^3x192 lattice, $\beta = 6.72$, $a = 0.06$, $m_l = 0.0008$, $m_s = 0.022$, $||r|| = 10^{-10}$

OLD SCHOOL: CG

Schur system: $\left(m^2 - D_{eo}^{HISQ} D_{oe}^{HISQ}\right)\overrightarrow{x_e} = m\overrightarrow{b_e} - D_{eo}^{HISQ}\overrightarrow{b_o}$ to tolerance $m10^{-10}$

Use mixed precision instead: double-single, **double-half**

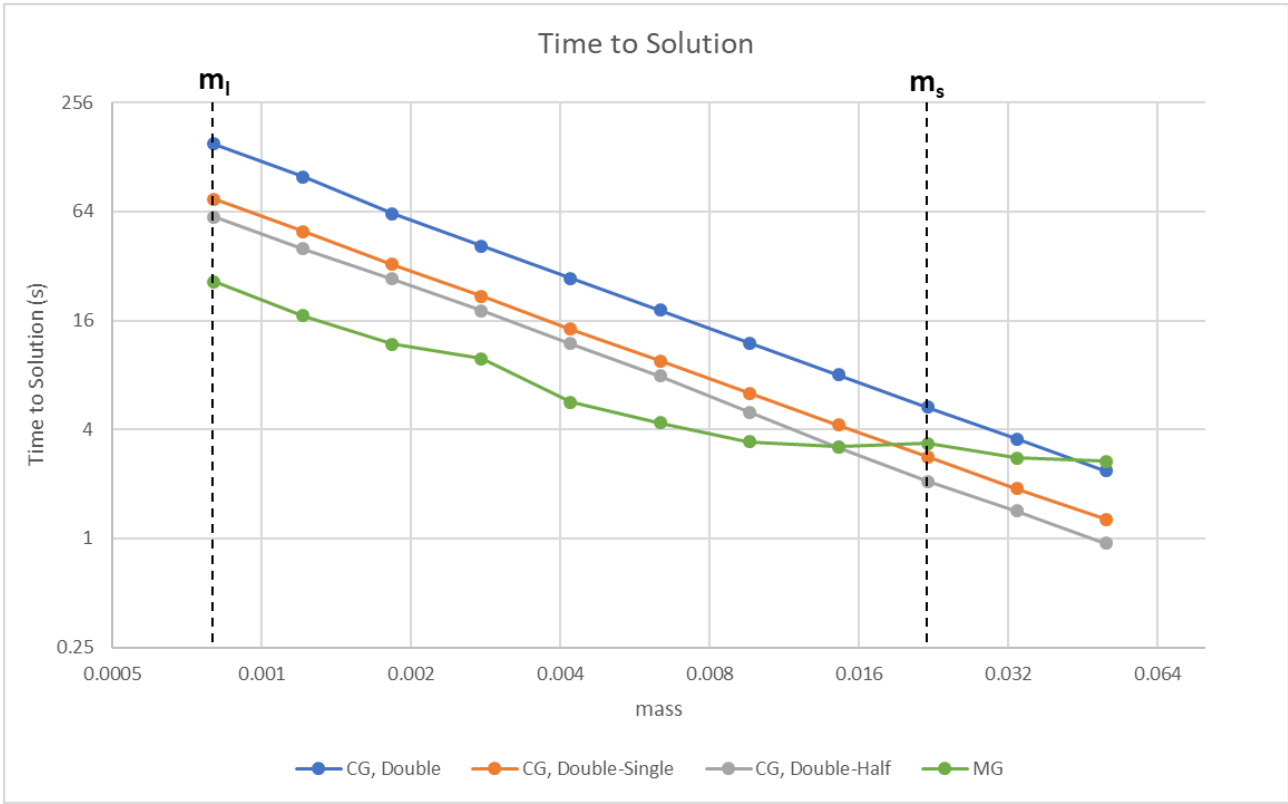


72x Summit nodes, random source, V=96^3x192 lattice, $\beta = 6.72$, $a = 0.06$, $m_l = 0.0008$, $m_s = 0.022$, $||r|| = 10^{-10}$

WITH MULTIGRID

Schur system: $\left(m^2 - D_{eo}^{stag} D_{oe}^{stag}\right)\overrightarrow{x_e} = m\overrightarrow{b_e} - D_{eo}\overrightarrow{b_o}$ to tolerance $m10^{-10}$

Note: re-uses near-null vectors generated at m_l for all masses

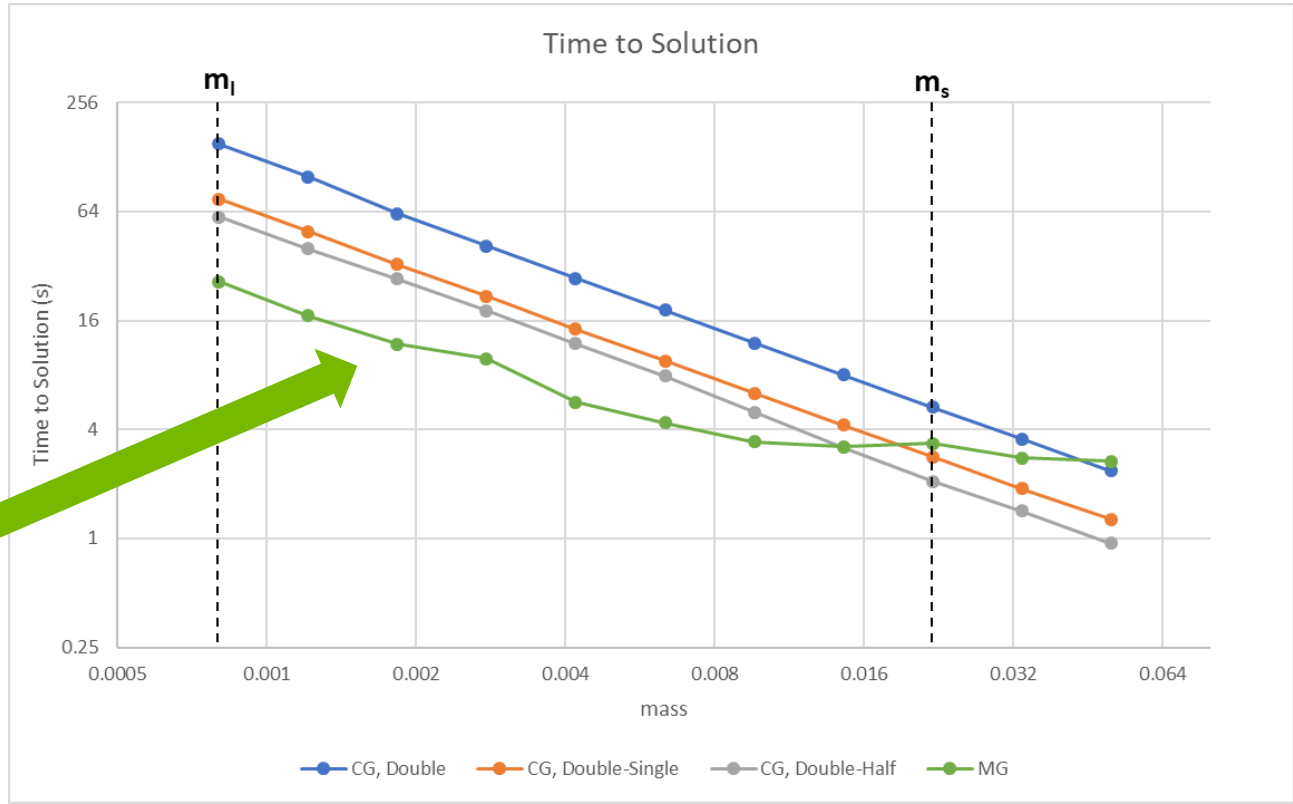


72x Summit nodes, random source, V=96^3x192 lattice, $\beta = 6.72$, $a = 0.06$, $m_l = 0.0008$, $m_s = 0.022$, $||r|| = 10^{-10}$

WITH MULTIGRID

Schur system: $\left(m^2 - D_{eo}^{stag} D_{oe}^{stag}\right)\overrightarrow{x_e} = m\overrightarrow{b_e} - D_{eo}\overrightarrow{b_o}$ to tolerance $m10^{-10}$

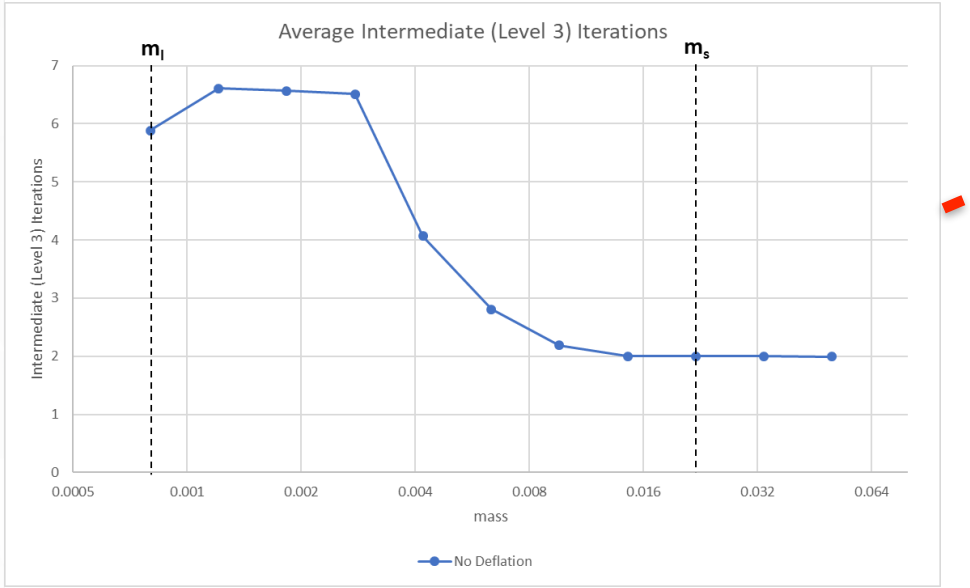
Note: re-uses near-null vectors generated at m_l for all masses



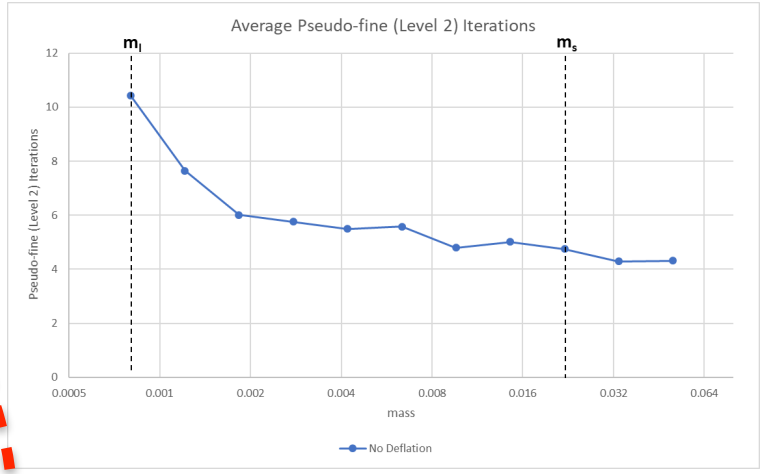
Still critical slowing down!

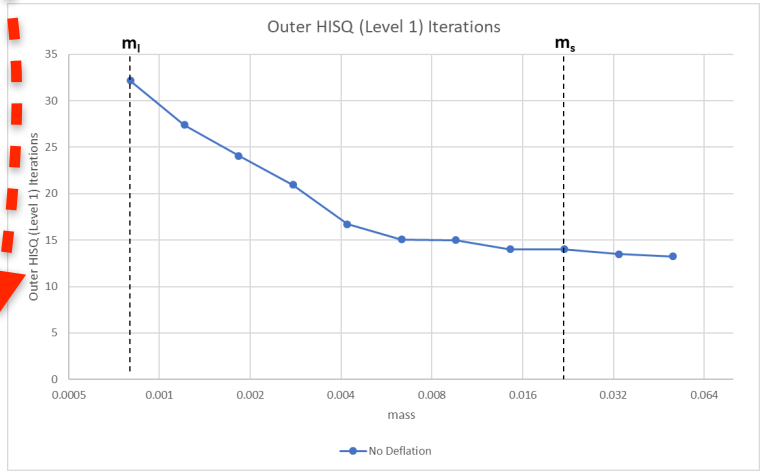
72x Summit nodes, random source, $V=96^3x192$ lattice, $\beta = 6.72$, $a = 0.06$, $m_l = 0.0008$, $m_s = 0.022$, $||r|| = 10^{-10}$

WHAT'S HAPPENING?



Percolates back up to fine grid

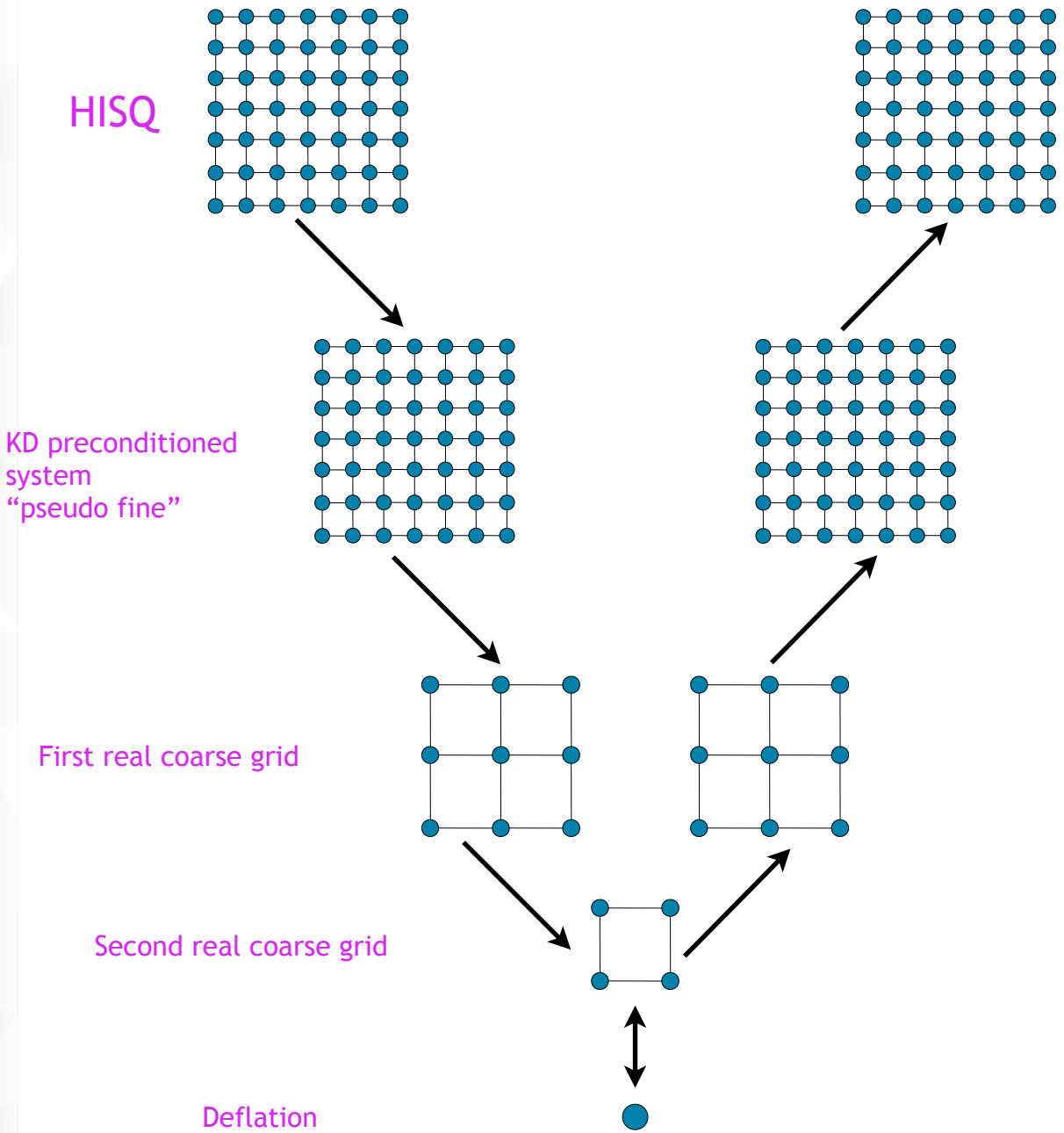




Not doing a good job preconditioning the next-coarsest level

=> Not solving the coarsest level well enough

DEFLATED STAGGERED MG



Level 1: 3 dof per site.
Solver: GCR, tolerance 10^{-10}

Smoother: CA-GCR(0,8)

Level 2: 48 dof per site.
Solver: GCR, tolerance 0.25, max 16 iterations
Operator: Left-block Schur, 16-bit precision

Smoother: CA-GCR(0,2)

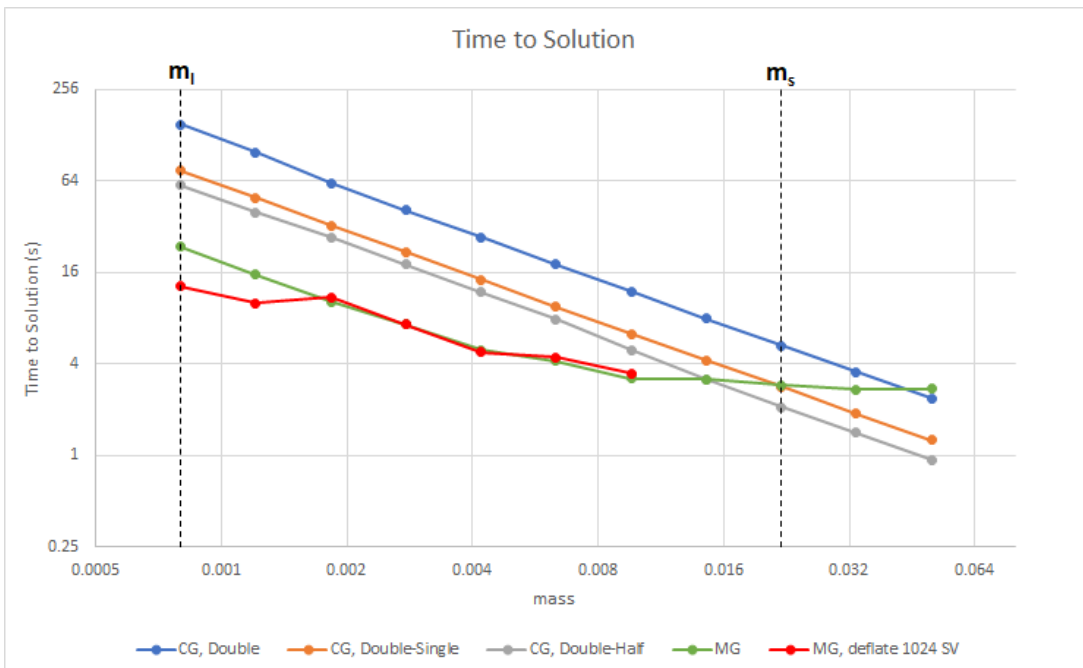
Level 3: **128 dof per site.** Solver: GCR, tolerance 0.25, max 16 iterations
Operator: Left-block Schur, 16-bit precision

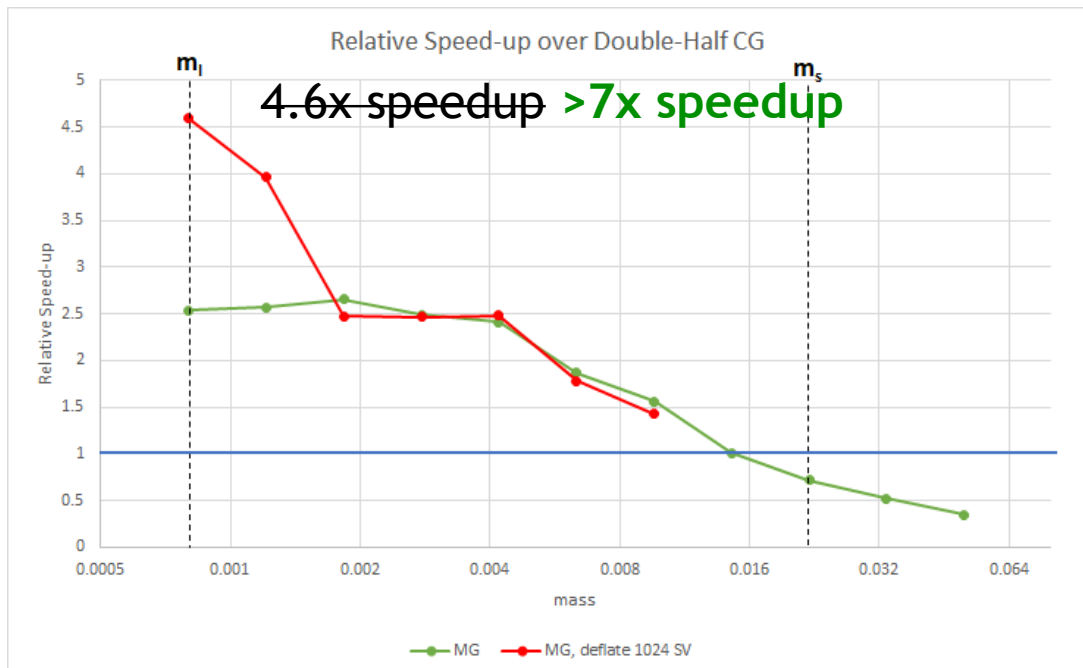
Smoother: CA-GCR(0,2)

Level 4: **192 dof per site.**
Solver: CA-GCR(16)
Operator: Left-block Schur, 16-bit precision

Level 5: 1024 vector SVD Deflation

DEFLATED HISQ MG





STAGGERED MG SUMMARY

Workable algorithm for applying multigrid to staggered fermions

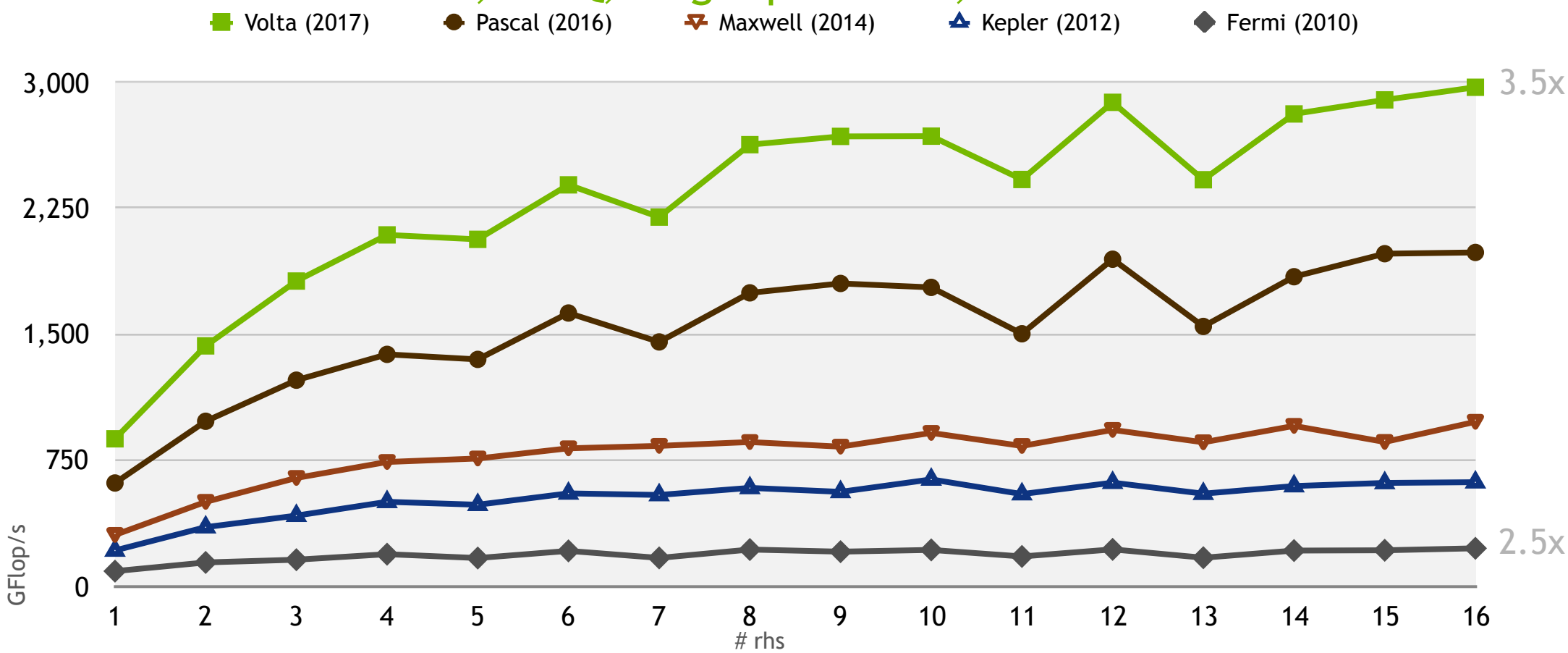
Significant speedups possible (>7x vs mixed-precision CG)

Requires a significant number of null-space vectors

HMC not ***yet*** workable

MULTIPLE RIGHT-HAND SIDES

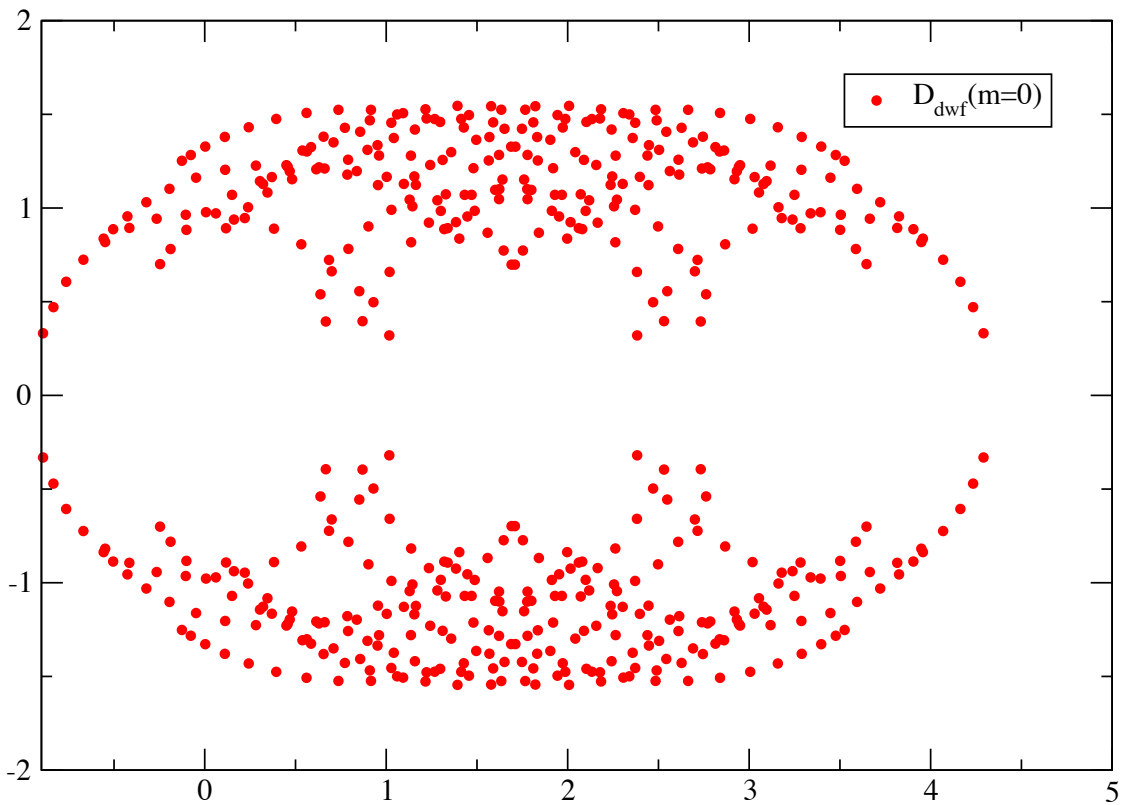
48^3x12, HISQ, single precision, one code



ONGOING AND FUTURE CHALLENGES

DOMAIN-WALL MULTIGRID

The problem is the spectrum



2-d Shamir operator

DOMAIN-WALL MULTIGRID

Multigrid for Domain Wall (Cohen *et al*, 2012)

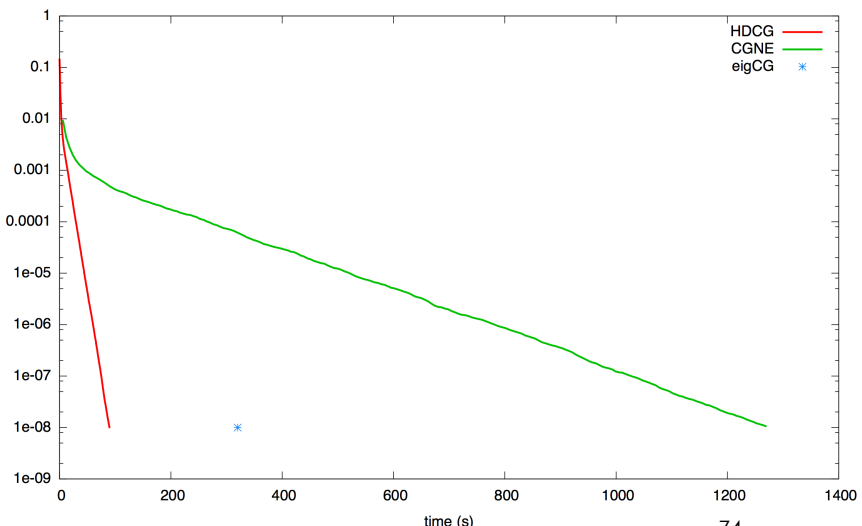
- Applied adaptive MG to normal op
- Removal of critical slowing down
- No actual speedup

Hierarchically Deflated CG (Boyle, 2014)

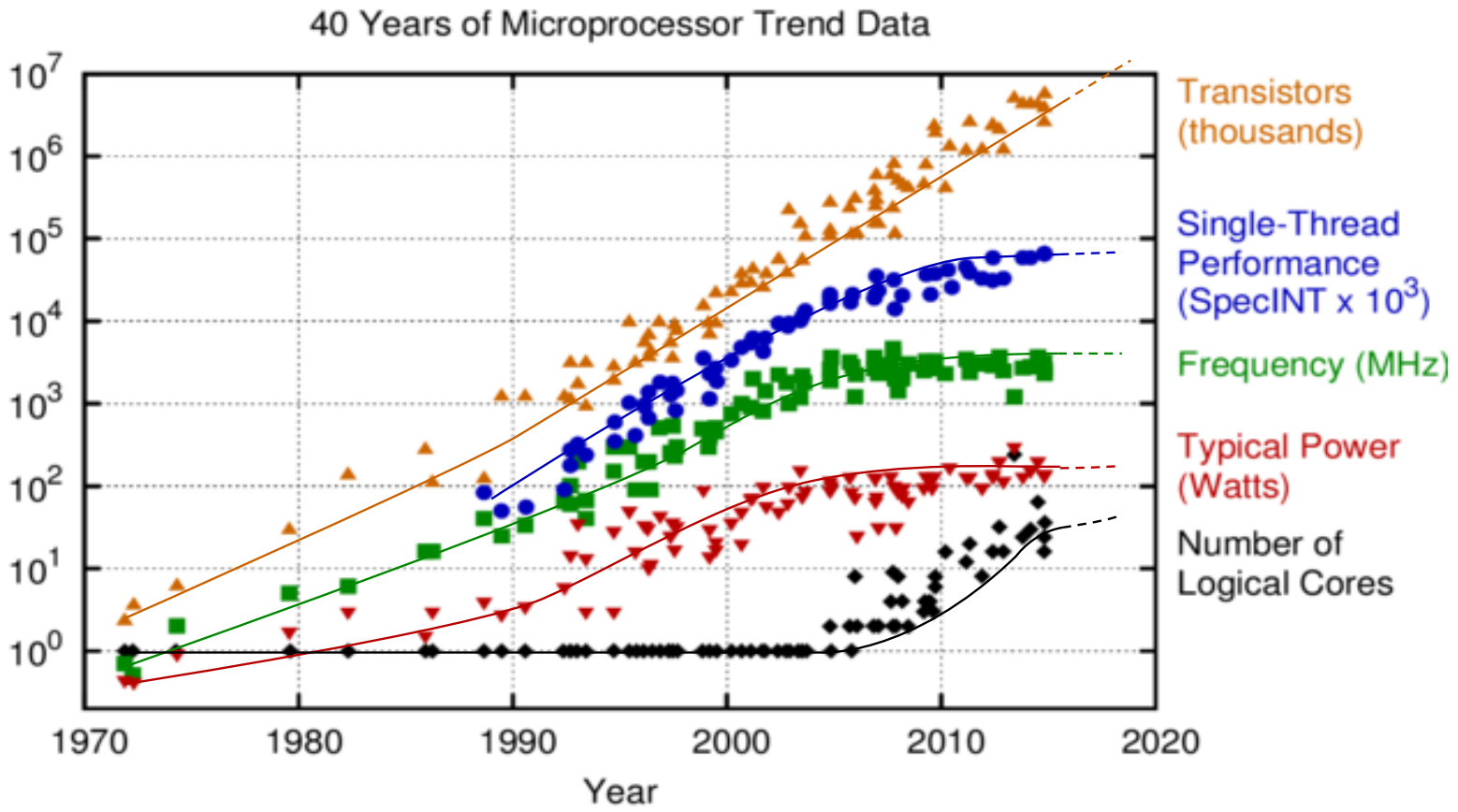
- Applied Inexact Deflation to normal op
- Removes critical slowing down
- Good solver speedup
- Expensive setup makes it unsuitable for HMC

Multigrid for Overlap (Brannick *et al*, 2014)

- Use Wilson as the preconditioner
- Run MG on this system
- Works as we approach continuum limit



THE PROTRACTED DEATH OF MOORE'S LAW



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

MULTIGRID AT THE EXASCALE

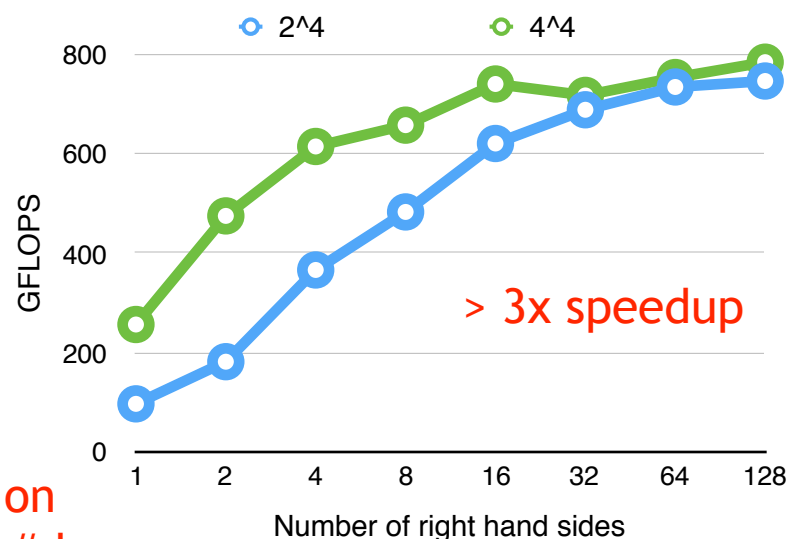
Cannot weak scale to infinite volume $C \sim m^{-1} a^{-6} V^{9/8}$

Even LQCD could be running out of parallelism

- Multigrid is pathological

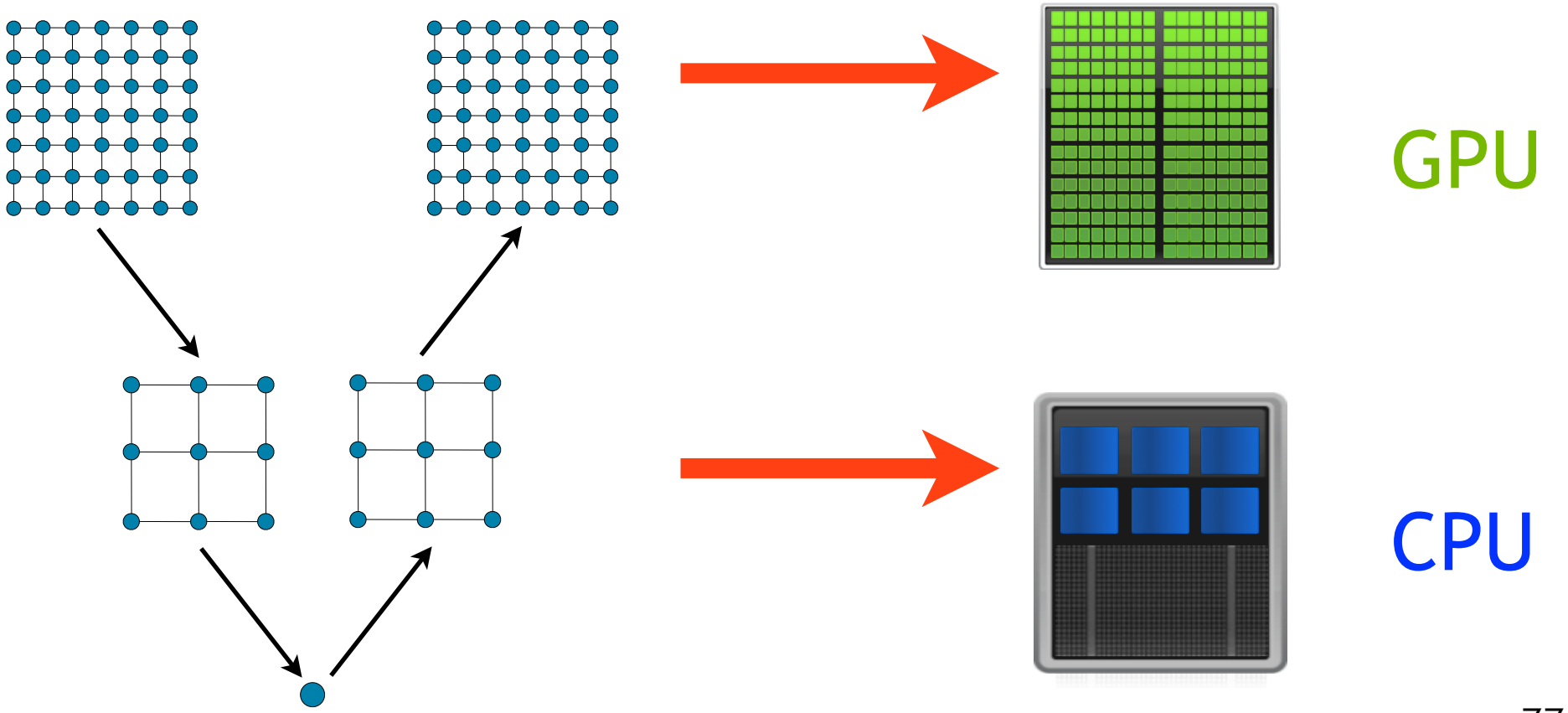
Multi-src solvers are a solution

- More parallelism and locality
- Bigger messages



Coarse dslash on M6000 GPU vs #rhs

HIERARCHICAL ALGORITHMS ON HETEROGENEOUS ARCHITECTURES



SUMMARY

Multigrid methods remove critical slowing down for most fermion formulations

Multigrid methods and GPUs are a potent combination

Ongoing challenges for chiral fermion formulations


NVIDIA

DOMAIN-DECOMPOSITION SMOOTHERS

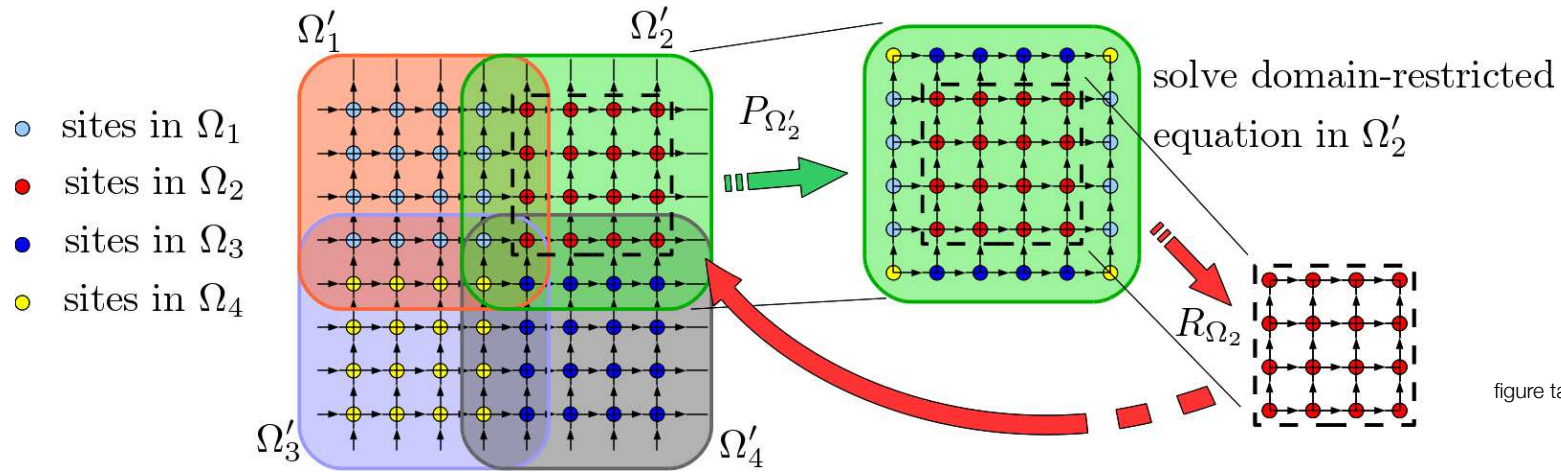


figure taken from Osaki and Ishikawa

Domain-decomposition smoothers are effective smoothers for QCD MG (Frommer *et al*)

QUDA now has support for both additive and multiplicative Schwarz smoothing

Enable at any level and / or combine with even/odd preconditioning at any level

Dramatic reduction in communication important on systems with weak networks

E.g., Piz Daint vs. Saturn V

https://github.com/JeffersonLab/qdp-jit

CHROMA + QDP-JIT/LLVM

QDP-JIT/PTX: implementation of QDP++ API for NVIDIA GPUs by Frank Winter (arXiv:1408.5925)

- Chroma builds unaltered and offloads evaluations to the GPU automatically
- Direct device interface to QUDA to run optimized solves

Prior publication covers earlier with direct PTX code generator

- Now use LLVM IR code generator and can target any architecture that LLVM supports

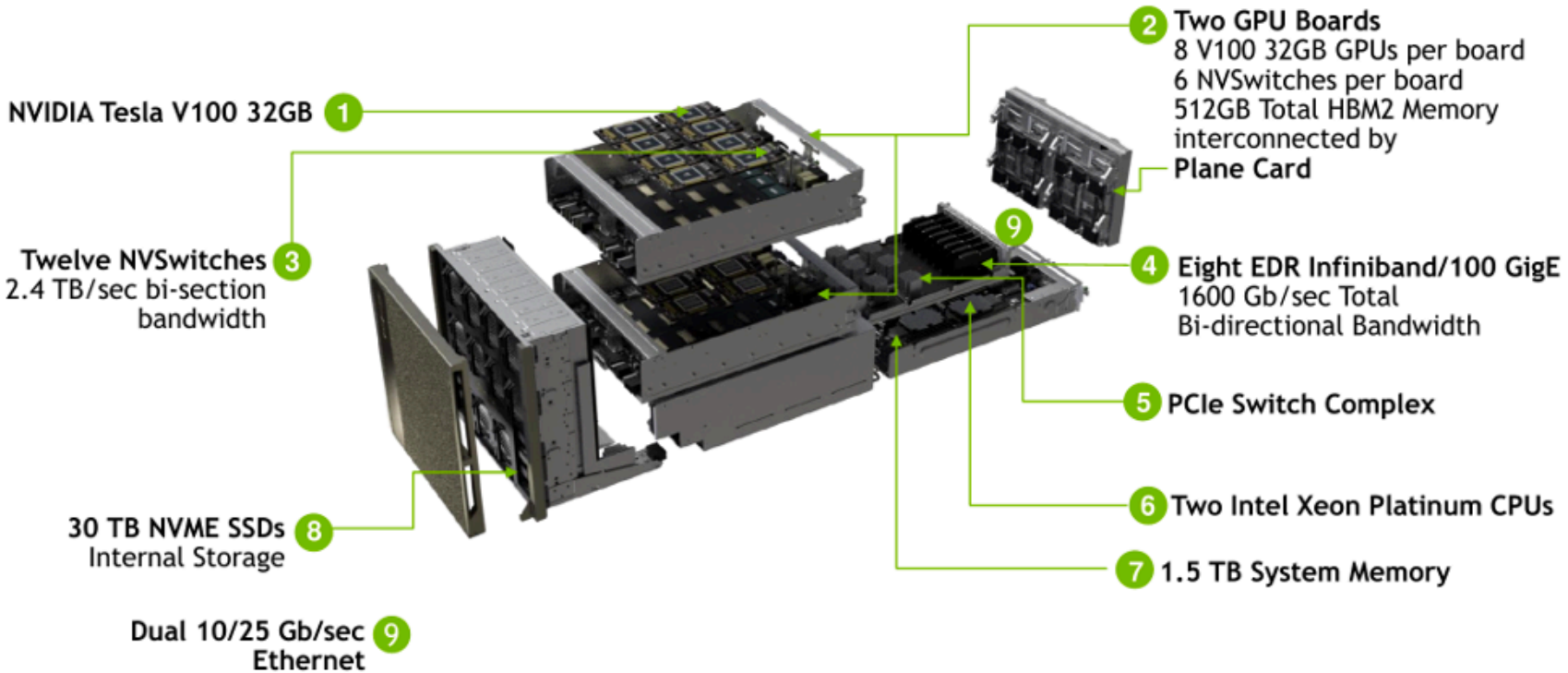
Chroma/QDP-JIT: Clover HMC in production on Titan and newer machines

Latest improvements:

- Caching of PTX kernels to eliminate overheads
- Faster startup times making the library more suitable for all jobs

DGX-2

1 node supercomputer



2 PFLOPS | 512GB HBM2 | 10 kW | 350 lbs

DGX-2: FULL NON-BLOCKING BANDWIDTH

2.4 TB/s bisection bandwidth