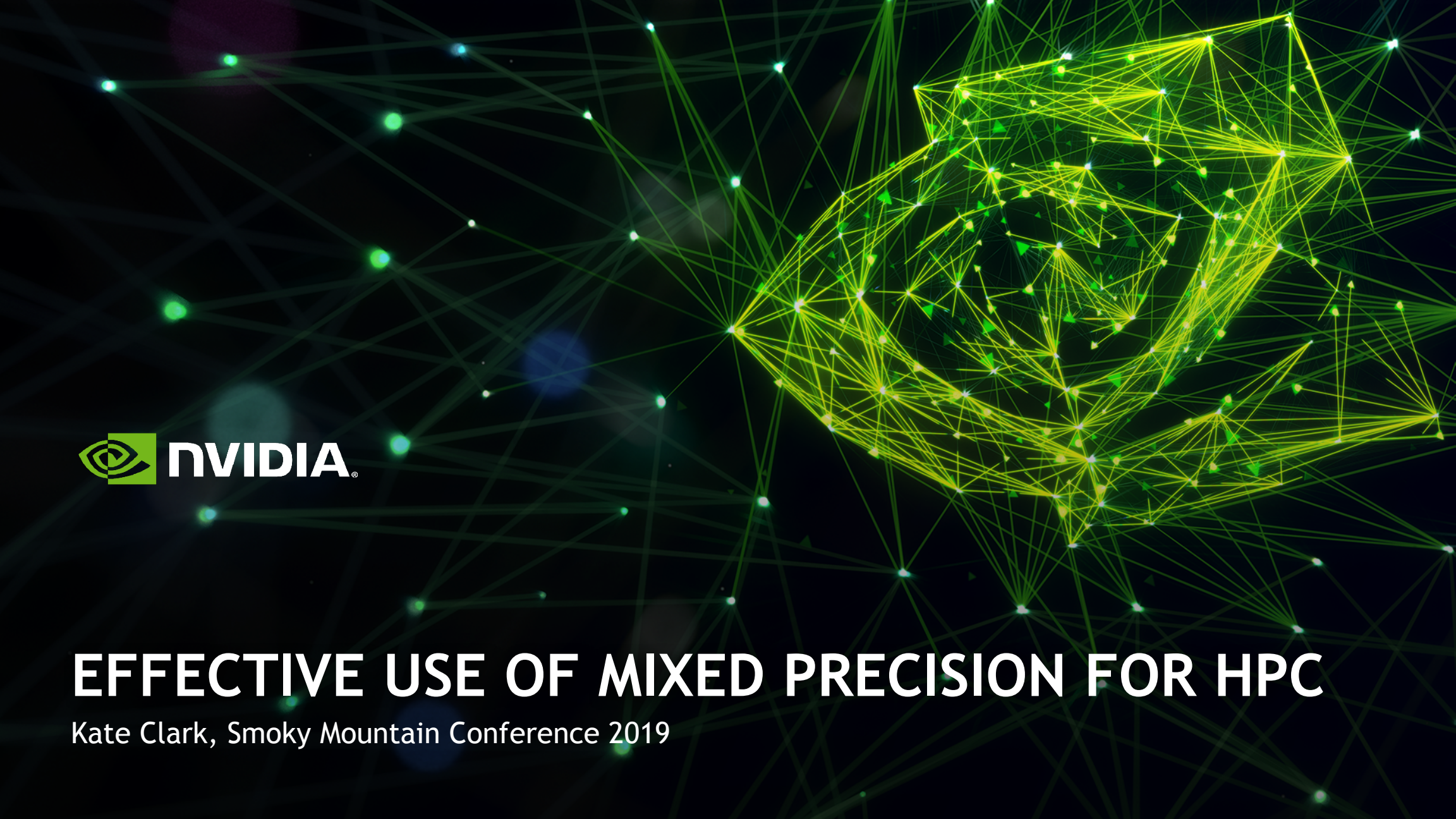


EFFECTIVE USE OF MIXED PRECISION FOR HPC

Kate Clark, Smoky Mountain Conference 2019

AGENDA

Why Mixed Precision

Lattice Quantum Chromodynamics

Mixed Precision and Krylov Solvers

Mixed Precision and Multigrid

Tensor cores

Future Challenges

Summary

NVIDIA.

WHY MIXED PRECISION?

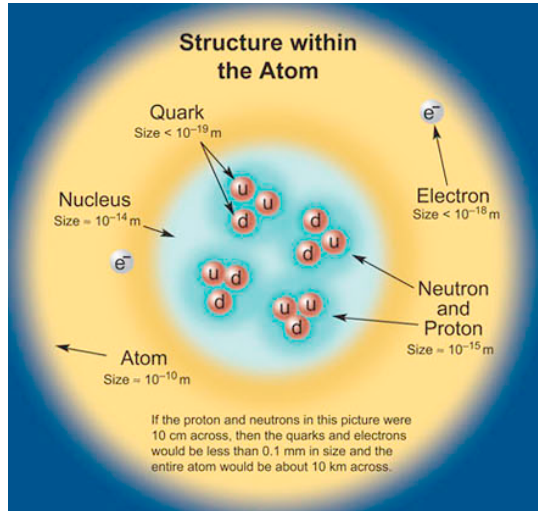
There are many reasons to consider different precisions

- Reduce memory traffic
- Reduce network traffic
- Reduce memory footprint
- Accelerated hardware in current architecture
- Suitable numerical properties for the algorithm at hand

Accelerate or even *improve* the algorithm without compromising the quality of science

LATTICE QCD

QUANTUM CHROMODYNAMICS



The strong force is one of the basic forces of nature
(along with gravity, em and weak)

It's what binds together the quarks and gluons in the proton and the neutron
(as well as hundreds of other particles seen in accelerator experiments)

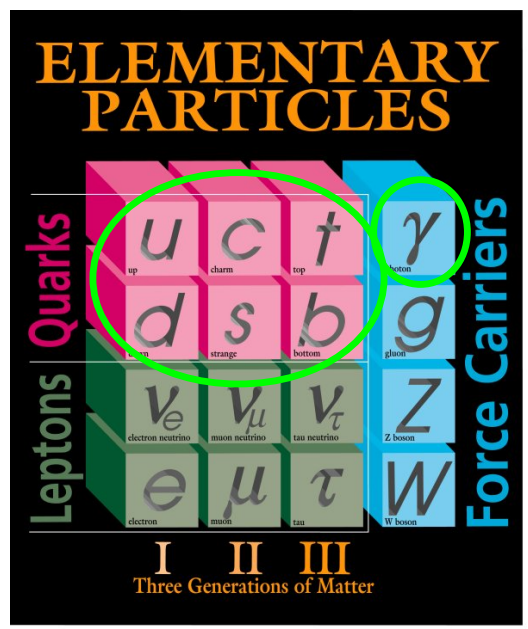
Responsible for the particle zoo seen at sub-nuclear scales (mass, decay rate, etc.)

QCD is the theory of the strong force

It's a beautiful theory…

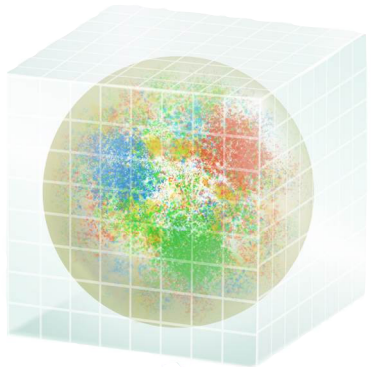
$$\langle \Omega \rangle = \frac{1}{Z} \int [dU] e^{-\int d^4x L(U)} \Omega(U)$$

…but



Fermi National Accelerator Laboratory

LATTICE QUANTUM CHROMODYNAMICS



Theory is highly non-linear ⇒ cannot solve directly

Must resort to numerical methods to make predictions

Lattice QCD

Discretize spacetime ⇒ 4-d dimensional lattice of size $L_x \times L_y \times L_z \times L_t$

Finite spacetime ⇒ periodic boundary conditions

PDEs ⇒ finite difference equations

Consumer of 10-20% of public supercomputer cycles

Traditionally highly optimized on every HPC platform for the past 30 years

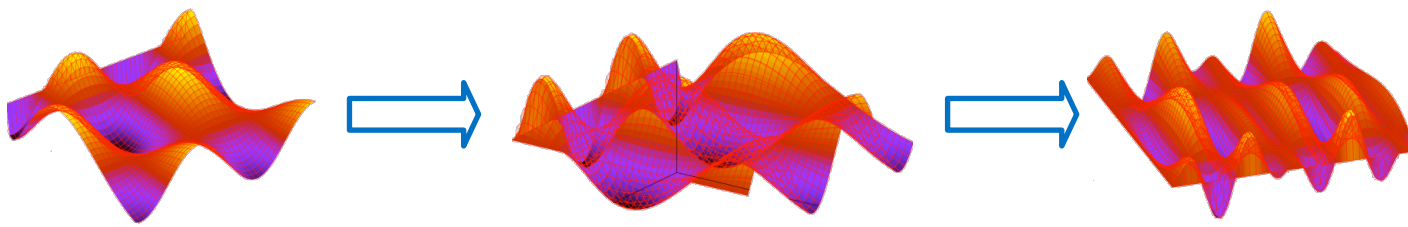
STEPS IN AN LQCD CALCULATION

$$D_{ij}^{\alpha\beta}(x,y;U)\psi_j^{\beta}(y) = \eta_i^{\alpha}(x)$$

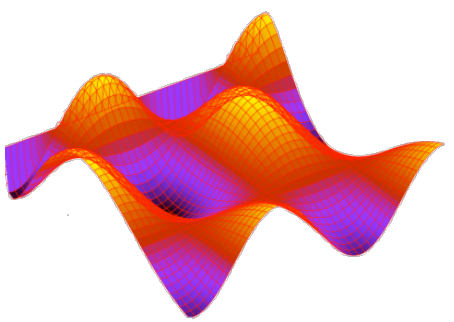
or Ax = b

1. Generate an ensemble of gluon field configurations “gauge generation”
   - Produced in sequence, with hundreds needed per ensemble
   - Strong scaling required with 100-1000 TFLOPS sustained for several months
   - 50-90% of the runtime is in the linear solver
   - O(1) solve per linear system
   - Target 16^4 per GPU

Simulation Cost ~ a^{-6} $V^{5/4}$

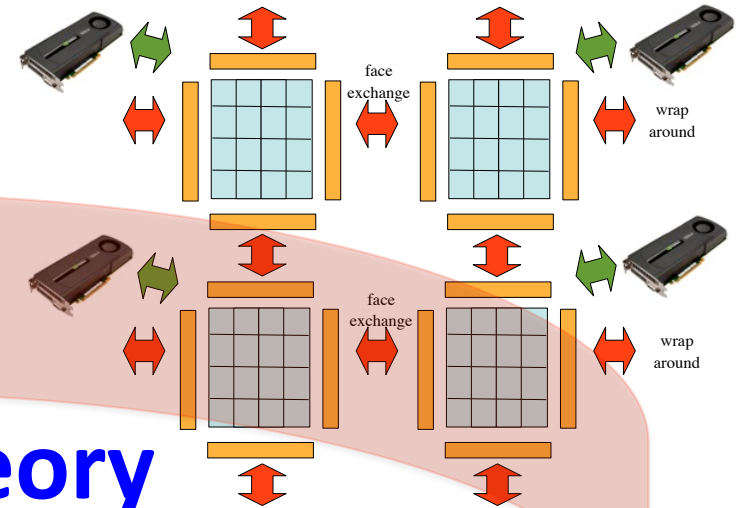


2. “Analyze” the configurations
   - Can be farmed out, assuming ~10 TFLOPS per job
   - Task parallelism means that clusters reign supreme here
   - 80-99% of the runtime is in the linear solver
     Many solves per system, e.g., $O(10^6)$
   - Target 24^4-32^4 per GPU

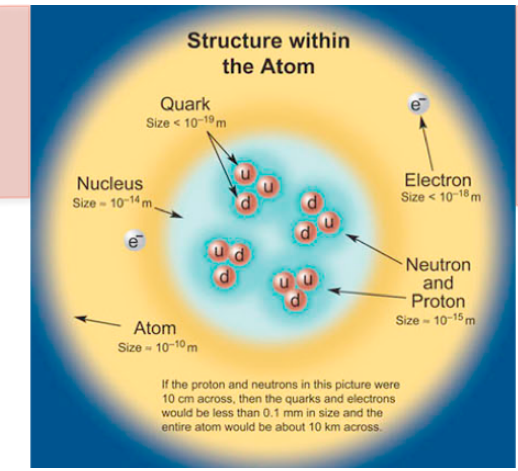


LATTICE QCD IN A NUTSHELL

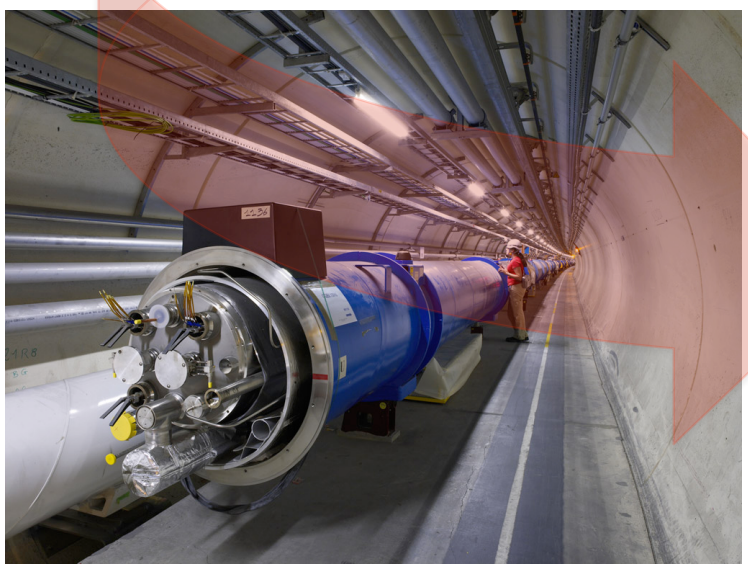
$$\langle \Omega \rangle = \frac{1}{Z} \int [dU] e^{-\int d^4x L(U)} \Omega(U)$$



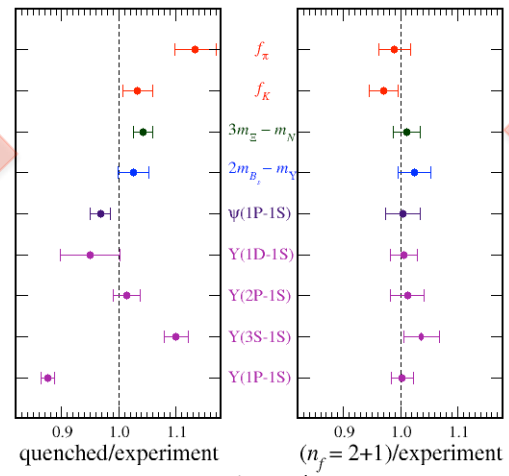
experiment



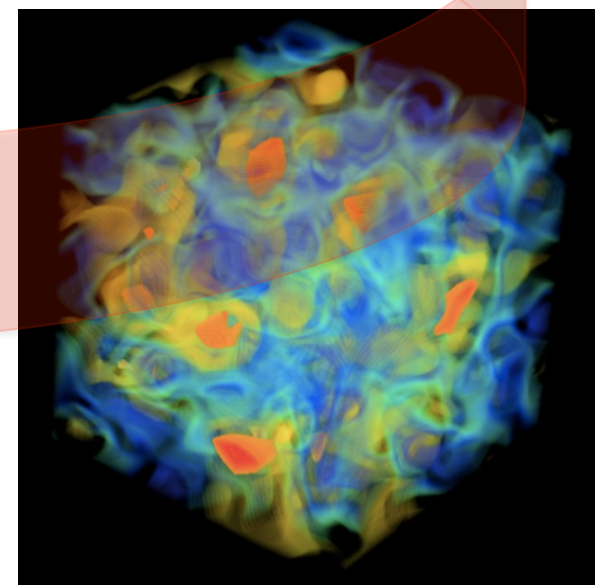
theory



Large Hadron Collider



Davies *et al*



Brookhaven National Laboratory

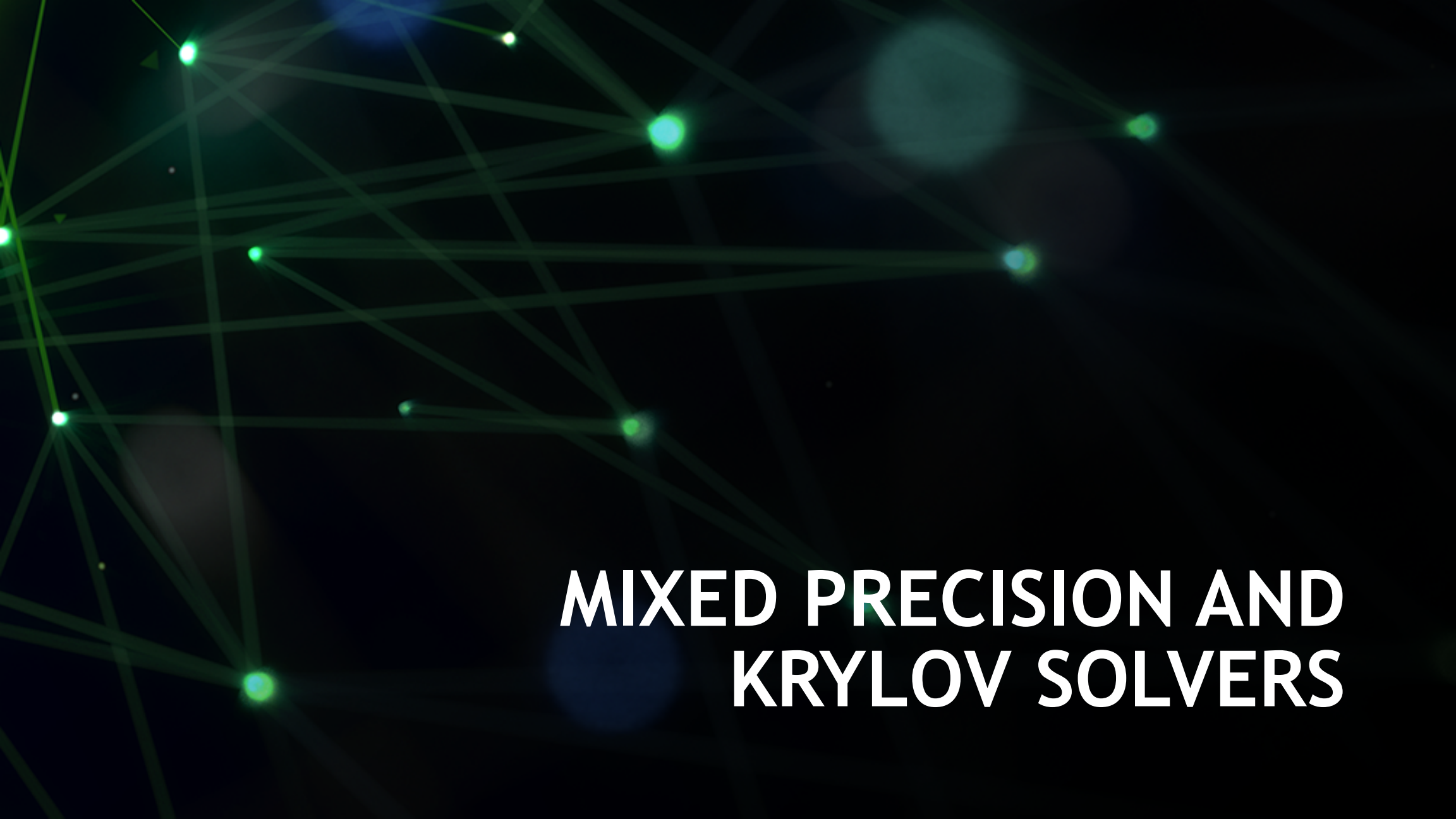
QUDA

- "QCD on CUDA" – http://lattice.github.com/quda (C++14, 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.
- Various solvers for all major fermionic discretizations, with multi-GPU support
- Maximize performance
  - **Mixed-precision methods (runtime specification of precision for maximum flexibility)**
  - Exploit physical symmetries to minimize memory traffic
  - 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-RHS solvers
  - Multigrid solvers for optimal convergence
- **A research tool for how to reach the exascale (and beyond)**

QUDA CONTRIBUTORS

10 years - lots of contributors

Ron Babich (NVIDIA)
Simone Bacchio (Cyprus)
Michael Baldhauf (Regensburg)
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)
Hauke Sandmeyer (Bielefeld)
Guochun Shi (NCSA -> Google)
Mario Schröck (INFN)
Alexei Strelchenko (FNAL)
Jiqun Tu (Columbia)
Alejandro Vaquero (Utah University)
Mathias Wagner (NVIDIA)
Evan Weinberg (NVIDIA)
Frank Winter (Jlab)


MIXED PRECISION AND
KRYLOV SOLVERS

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

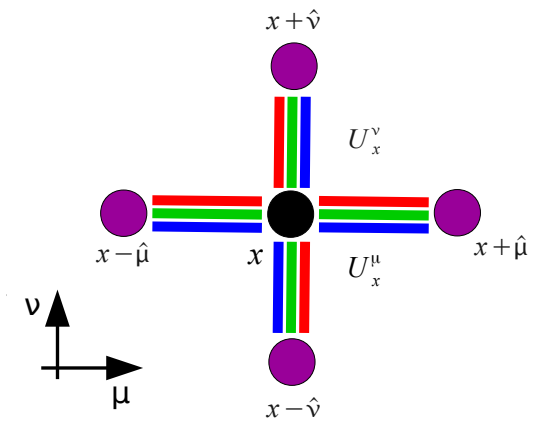
- Load the neighboring spinor (24 numbers x8)
- Load the color matrix connecting the sites (18 numbers x8)
- Do the computation
- Save the result (24 numbers)

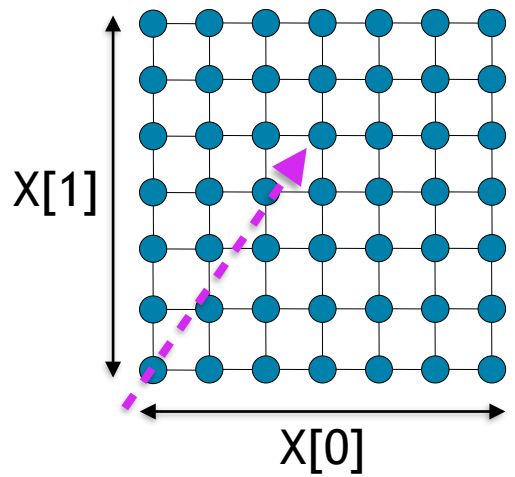
Each thread has (Wilson Dslash) 0.92 naive arithmetic intensity

QUDA reduces memory traffic

- Exact SU(3) matrix compression (18 => 12 or 8 real numbers)
- **Use reduced precision**

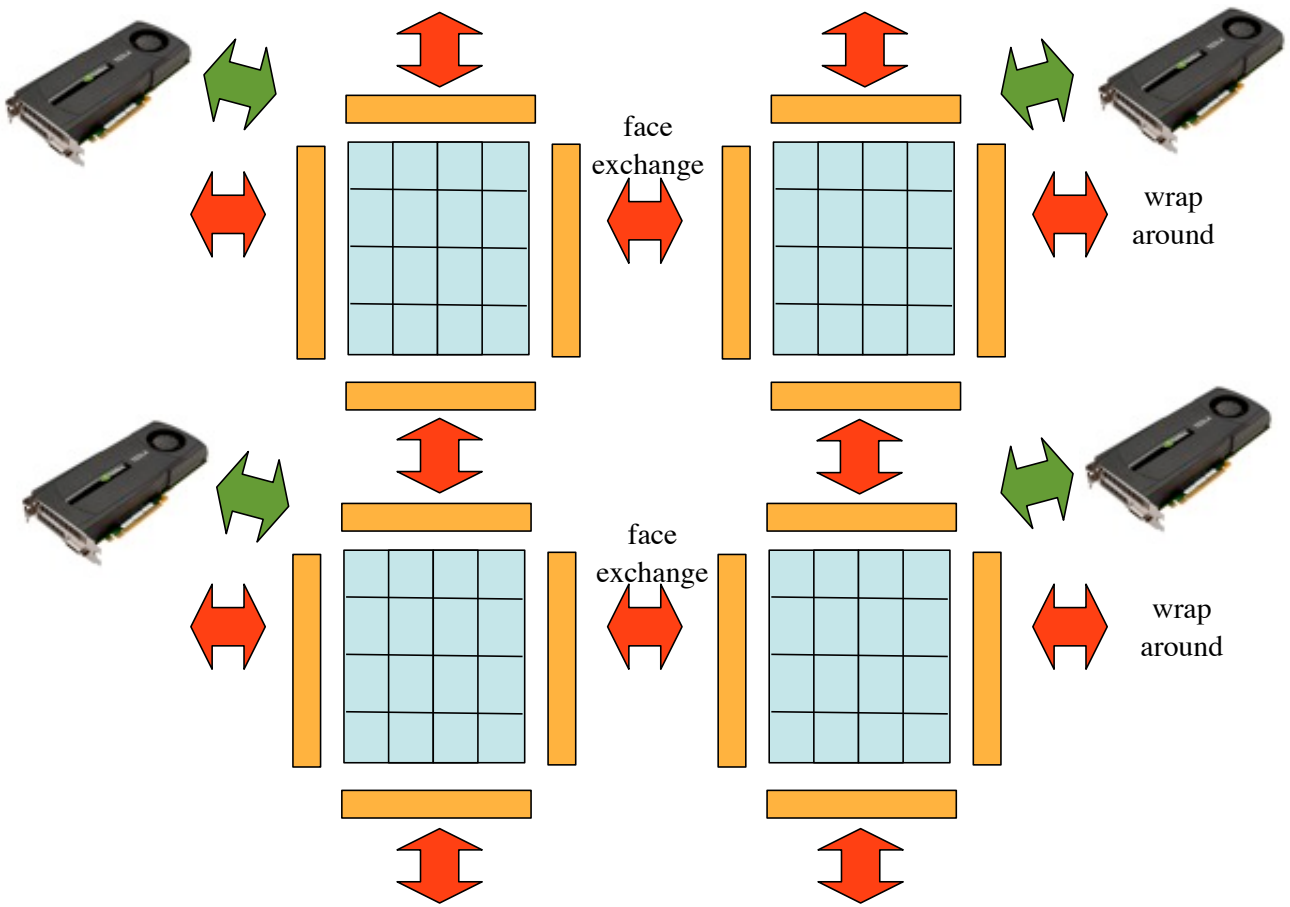
$$D_{x,x'} =$$





NVIDIA.

MULTI GPU BUILDING BLOCKS



Halo packing Kernel

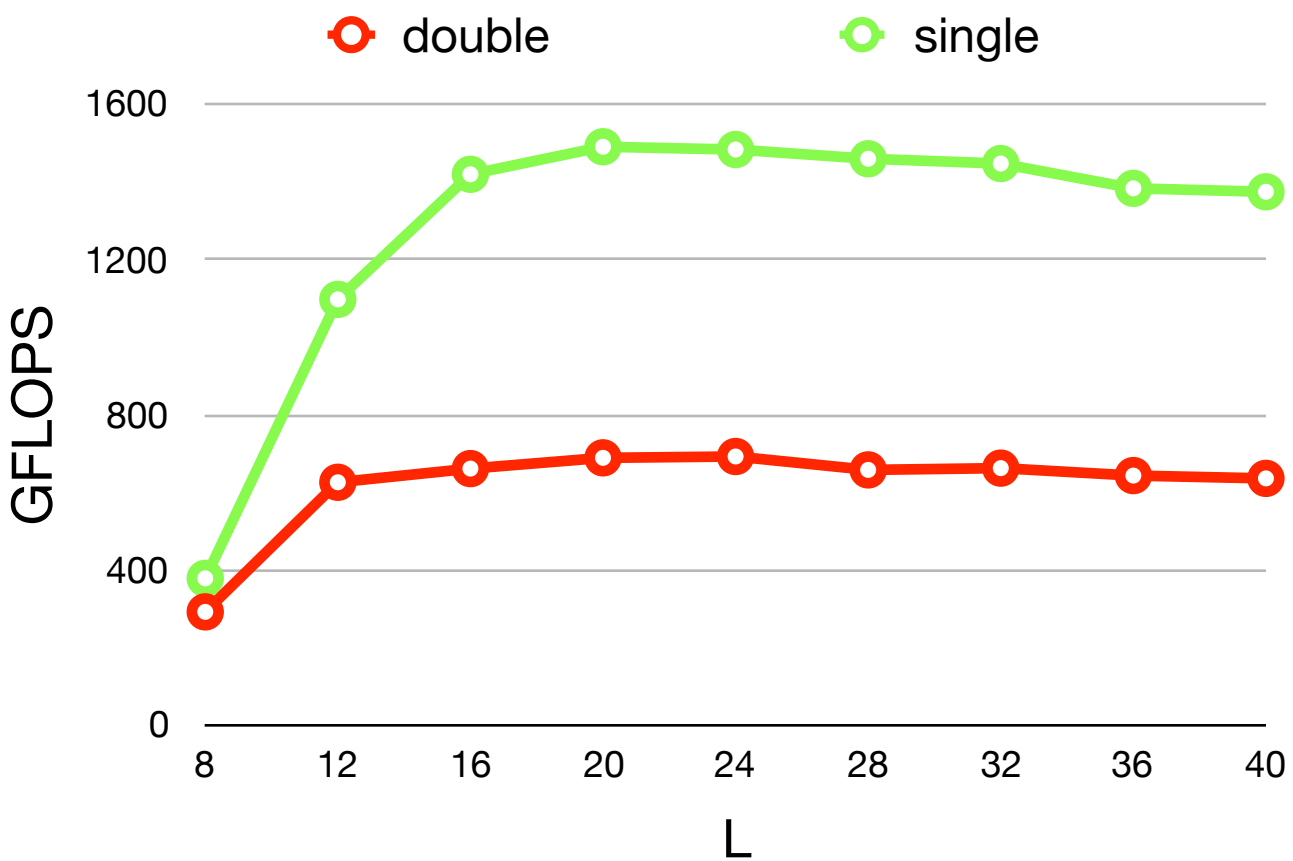
Interior Kernel

Halo communication

Halo update Kernel

SINGLE GPU PERFORMANCE

“Wilson Clover” stencil (Chroma)



Tesla V100
CUDA 10.1
GCC 7.3

QUDA'S 16-BIT FIXED POINT FORMAT

In production since 2009

Link field - Defines the sparse matrix elements

SU(3) matrices that live between all adjacent sites on the 4-d grid

All elements $\in [-1,1]$ => very natural to use 16-bit fixed-point representation

Fermion field - The vector that appears in the linear solver

Each 4-d grid point consists of a 12-component complex vector

No a priori bounds on the elements

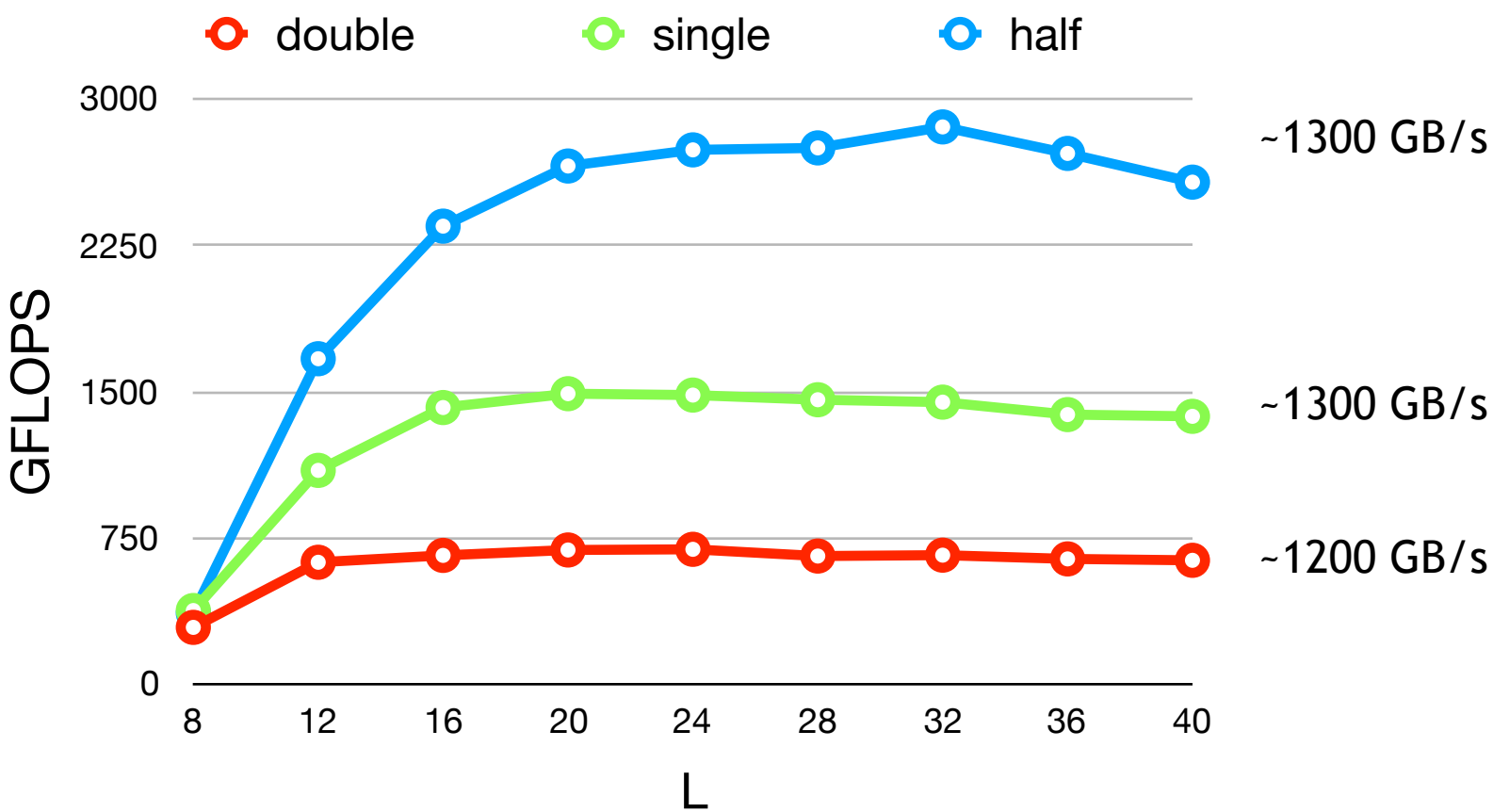
Use per-site L_{inf} norm to normalize the site vector and use 16-bit fixed point

Retain global dynamic range with local 16-bit mantissa

Low precision used only as a storage type and computation done in FP32

SINGLE GPU PERFORMANCE

“Wilson Clover” stencil (Chroma)



Tesla V100
CUDA 10.1
GCC 7.3

LINEAR SOLVERS

LQCD requires a range of sparse iterative linear solvers

CG, BiCGstab, GCR, Multi-shift solvers, etc.

Condition number inversely proportional to mass

Light (realistic) masses are highly singular

Naive Krylov solvers suffer from critical slowing down at decreasing mass

Entire solver algorithm must run on GPUs

Time-critical kernel is the stencil application

Also require BLAS level-1 type operations

```
while (|r_k|> ε) {
        β_k = (r_k,r_k)/(r_{k-1},r_{k-1})
        p_{k+1} = r_k - β_k p_k
    q_{k+1} = A p_{k+1}
        α = (r_k,r_k)/(p_{k+1}, q_{k+1})
        r_{k+1} = r_k - α q_{k+1}
        x_{k+1} = x_k + α p_{k+1}
        k = k+1
}
```

conjugate gradient

KC, Babich, Barros, Brower, Rebbi (2009)

*Sleijpen and Van der Worst, 1996

RELIABLE UPDATES FOR MIXED PRECISION

Traditional approach to mixed precision is to use iterative refinement

- Disadvantage: each restart means we discard the Krylov space

$$
\begin{aligned}
&\text{while } (|\mathbf{r}_k| > \varepsilon) \{ \\
&\quad \mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k \\
&\quad \text{solve } A\mathbf{p}_k = \mathbf{r}_k \\
&\quad \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k \\
&\}
\end{aligned}
$$

Instead we use Reliable Updates*

- As low-precision solver progresses, residual will drift
- Occasionally replace iterated residual with high-precision residual
- Retains the Krylov space information
- Maintain a separate partial-solution accumulator

$$
\begin{aligned}
&\text{if } (|\mathbf{r}_k| < \delta |\mathbf{b}|) \{ \\
&\quad \mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k \\
&\quad \mathbf{b} = \mathbf{r}_k \\
&\quad \mathbf{y} = \mathbf{y} + \mathbf{x}_k \\
&\quad \mathbf{x}_k = 0 \\
&\}
\end{aligned}
$$

Aside: reductions always done in fp64 regardless of data precision

(STABLE) MIXED-PRECISION CG

Three key ingredients

```
while (|r_k|> ε) {
        β_k = (r_k,r_k)/(r_k-1,r_k-1)
        p_k+1 = r_k - β_k p_k
      q_k+1 = A p_k+1
        α = (r_k,r_k)/(p_k+1, q_k+1)
        r_k+1 = r_k - α q_k+1
        x_k+1 = x_k + α p_k+1
        k = k+1
}
```

CG convergence relies on gradient vector being orthogonal to residual vector

- Re-project when injecting new residual (Strzodka and Gödekke, 2006)

α (stepsize) chosen to minimize $|e|_A$

- True regardless of underlying precision of process
- Solution correction is truncated if keep the solution vector in low precision
- Always keep the (partial) solution vectors in high precision

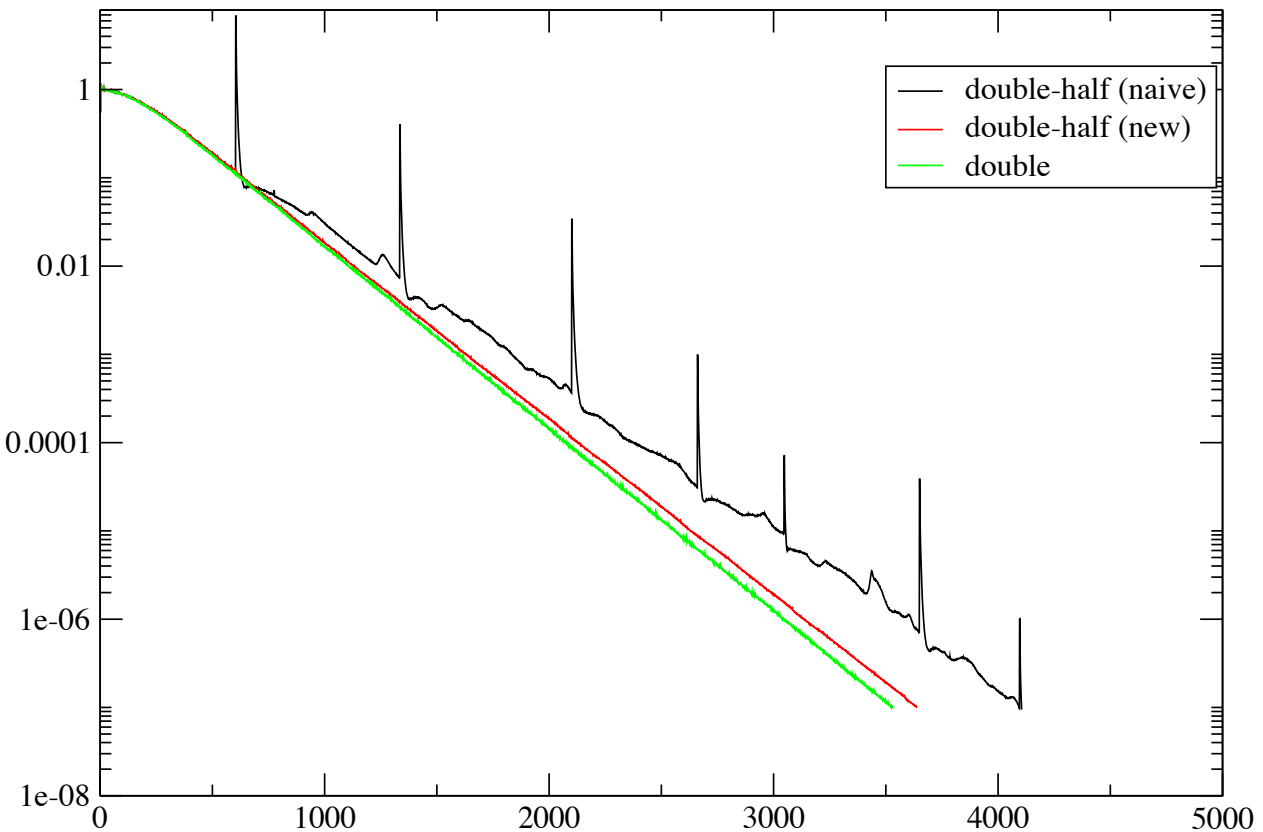
β computation relies on $(r_i, r_j) = |r_i|^2 \delta_{i,j}$

- Not true in finite precision
- Polak-Ribière form is equivalent and self-stabilizing

$$\beta_k = \frac{(r_k, (r_k - r_{k-1}))}{|r_{k-1}|^2}$$

MIXED-PRECISION MILC CG SOLVER

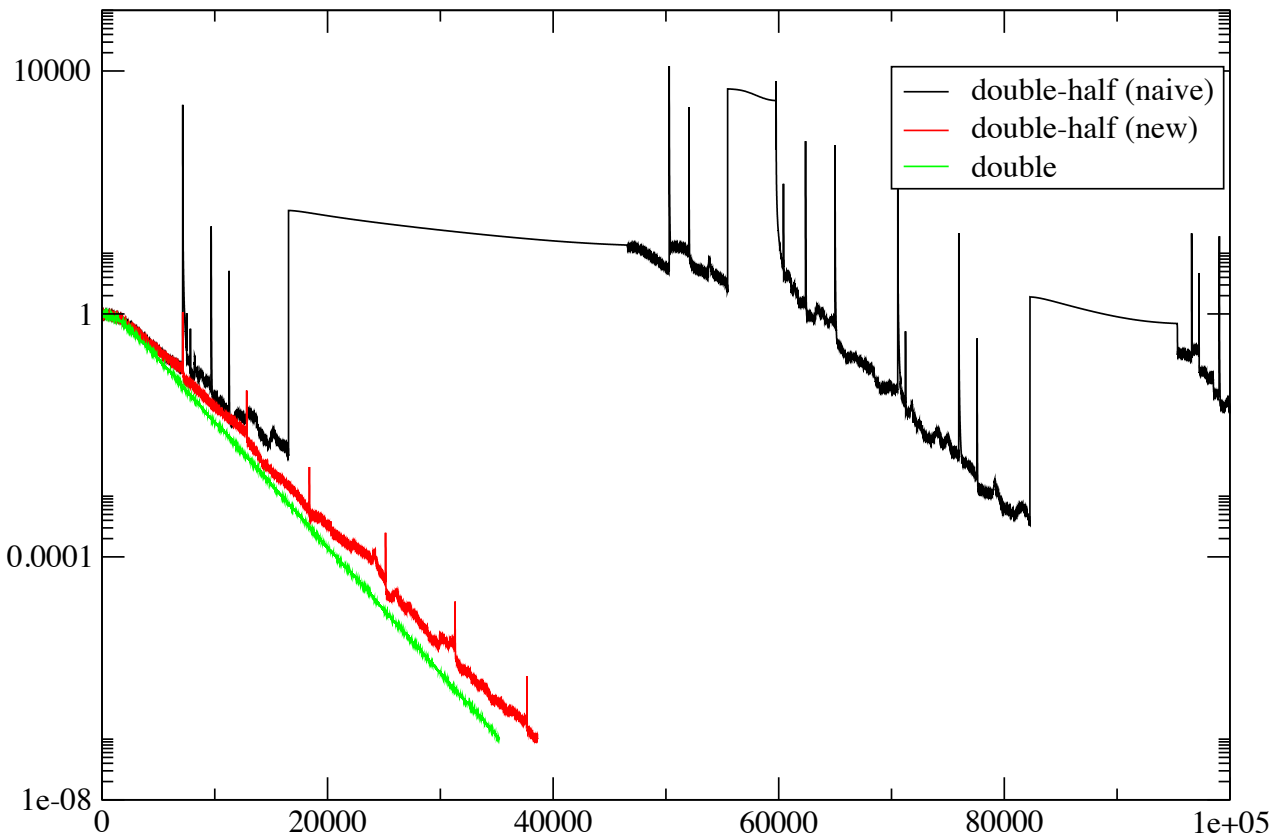
mass = 0.01 => $\kappa \sim 10^4$, $\delta = 0.1$



NVIDIA.

MIXED-PRECISION MILC CG SOLVER

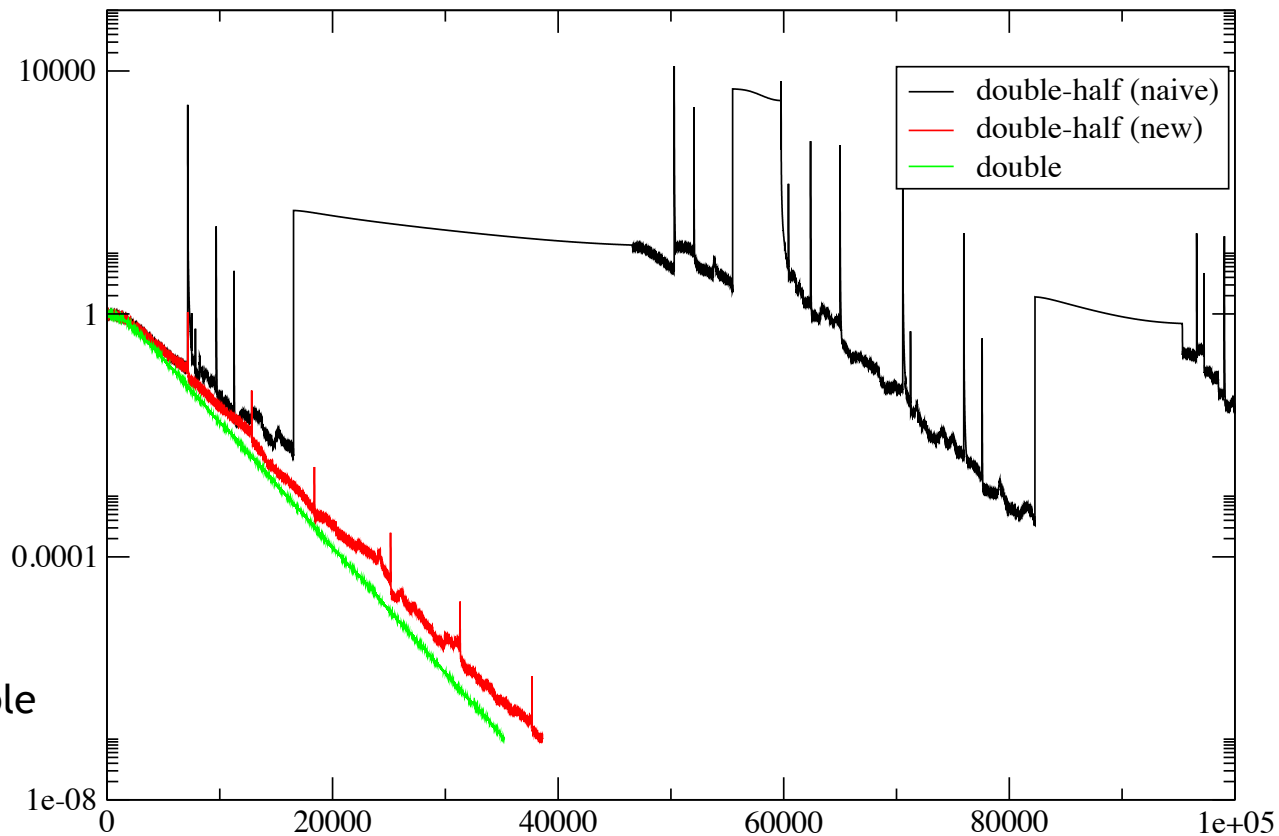
mass = 0.001, $\kappa \sim 10^6$, $\delta = 0.1$



NVIDIA.

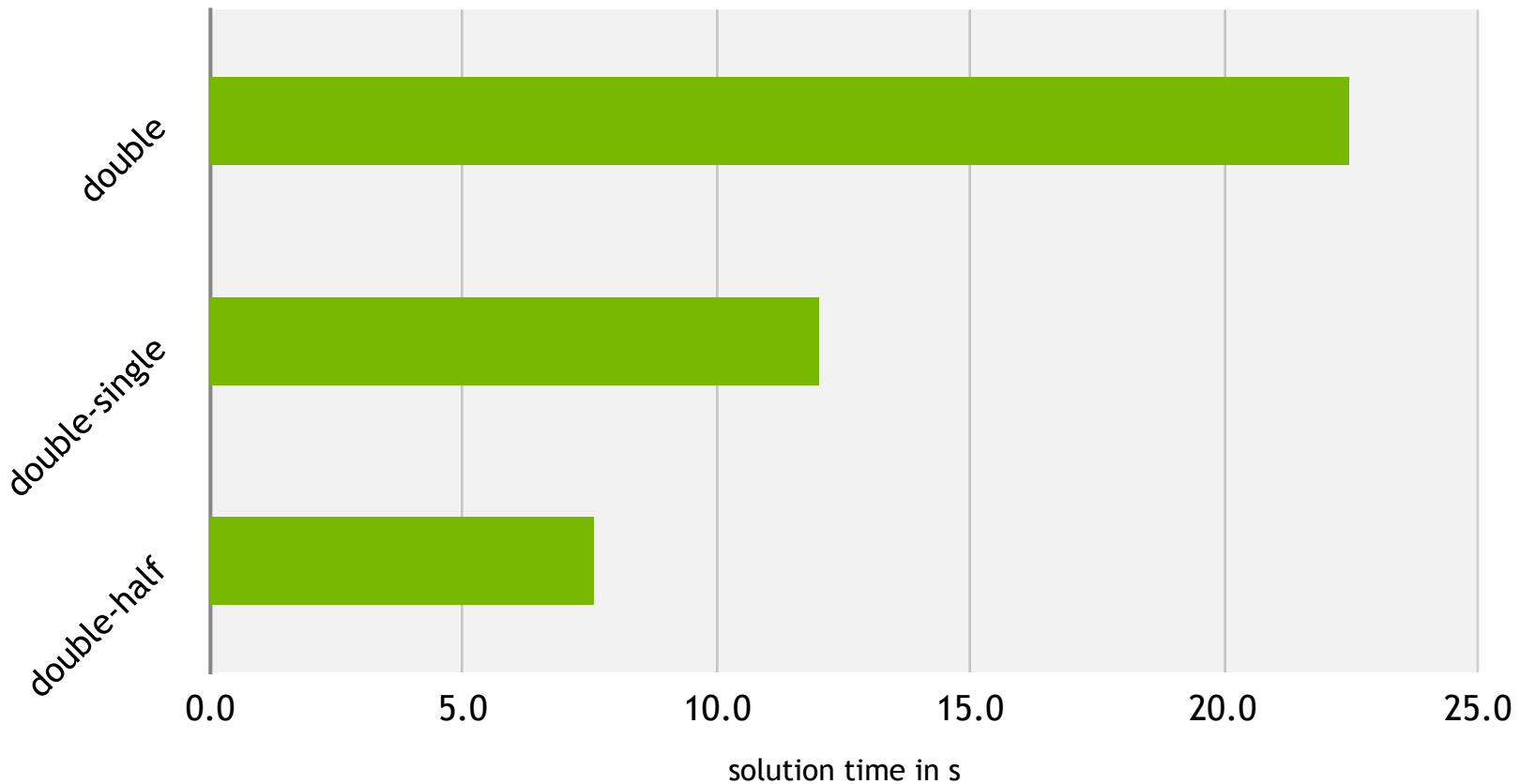
MIXED-PRECISION MILC CG SOLVER

mass = 0.001, $\kappa \sim 10^6$, $\delta = 0.1$



Looking at smarter reliable update triggers based on dynamic error estimates e.g. van der Vorst and Ye

MIXED-PRECISION MILC CG SOLVER



NVIDIA

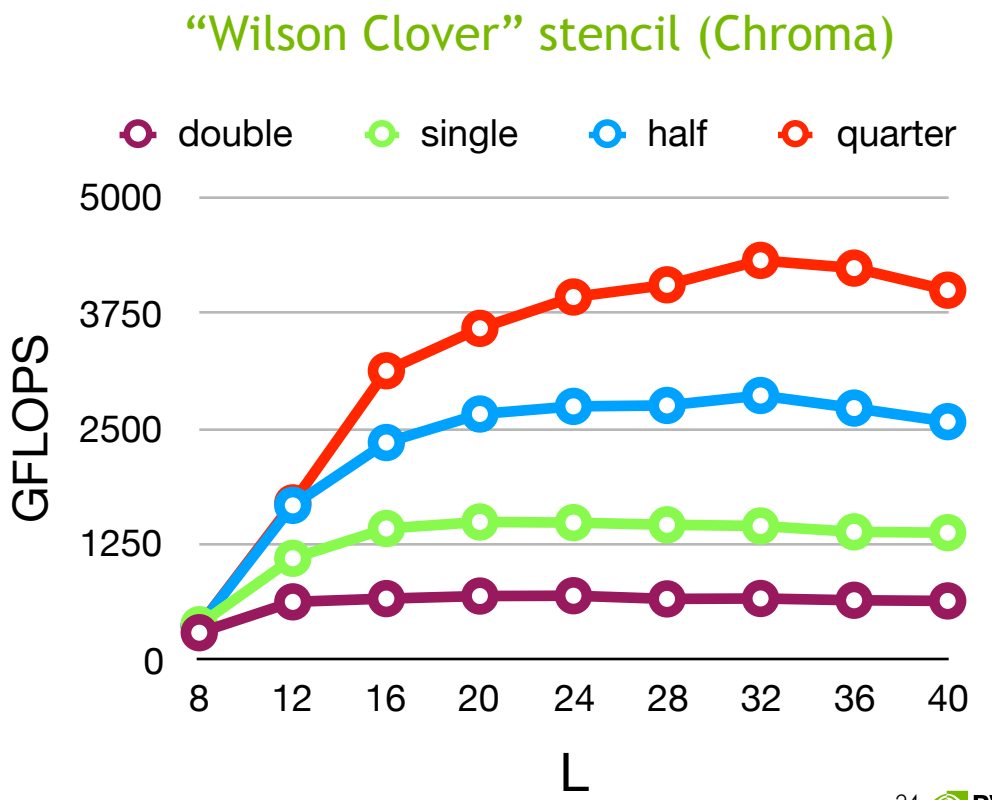
HOW LOW CAN YOU GO?

Easily extend to 8-bit fixed-point format

(Aside) finally becoming compute bound

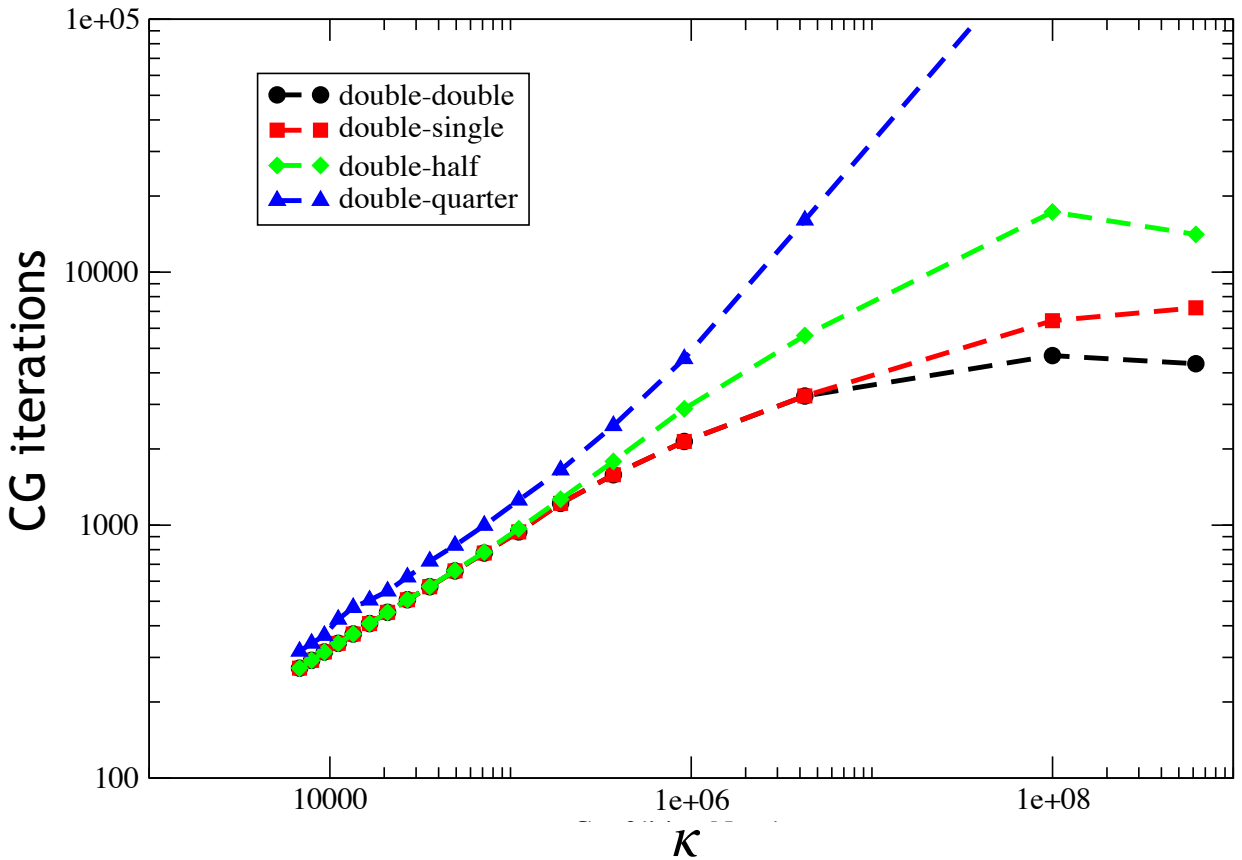
How much precision (information) is needed to converge the solver?

Condition number dictates the precision and range required



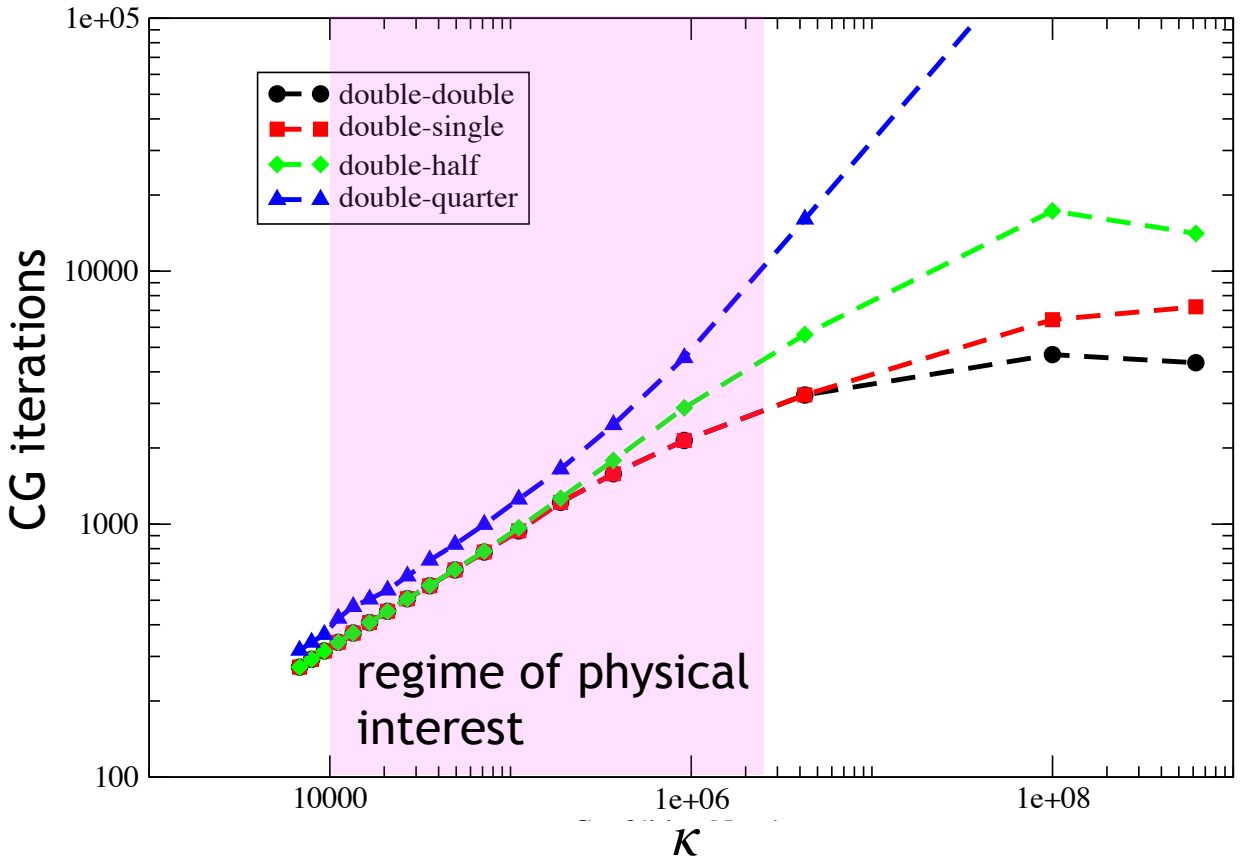
HOW LOW CAN YOU GO?

“Chroma” linear system, 3x10⁶ dof

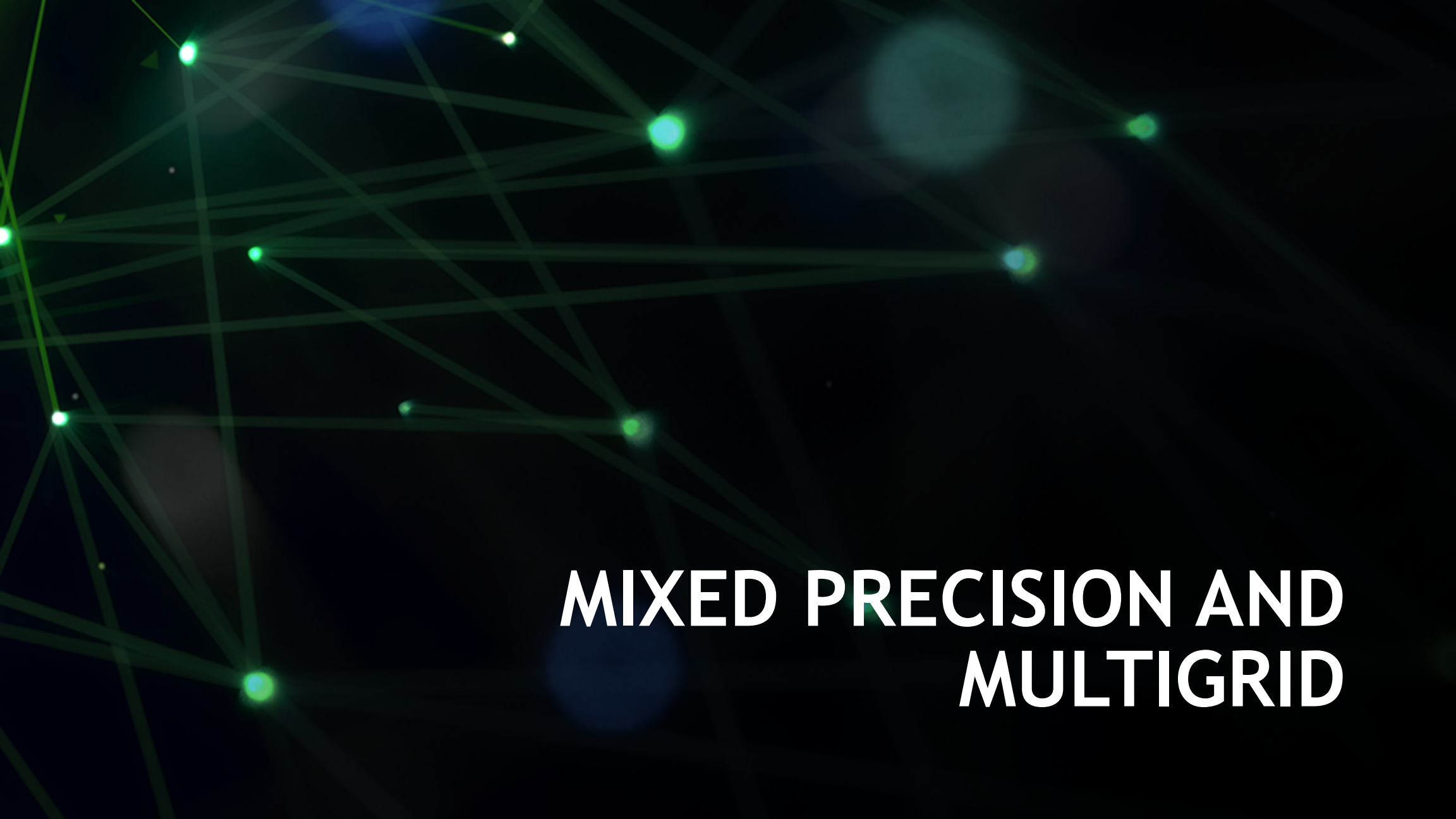


HOW LOW CAN YOU GO?

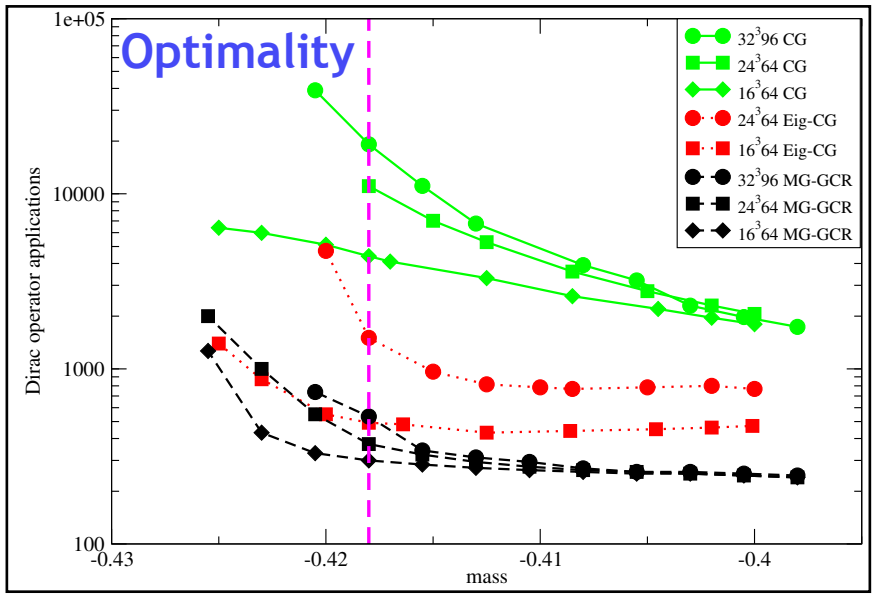
“Chroma” linear system, 3x10⁶ dof



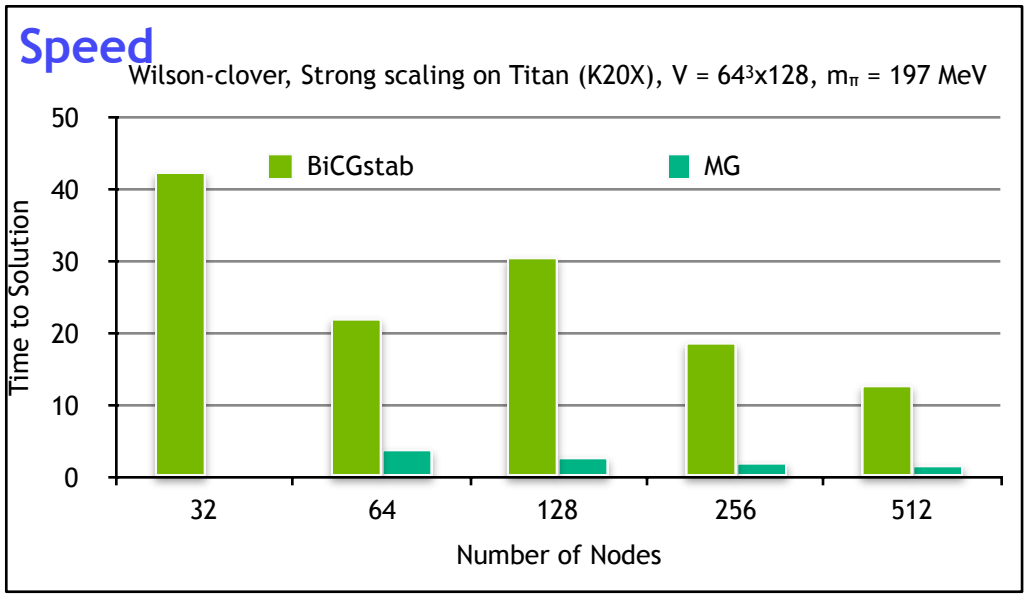
8-bit is a bridge too far


MIXED PRECISION AND
MULTIGRID

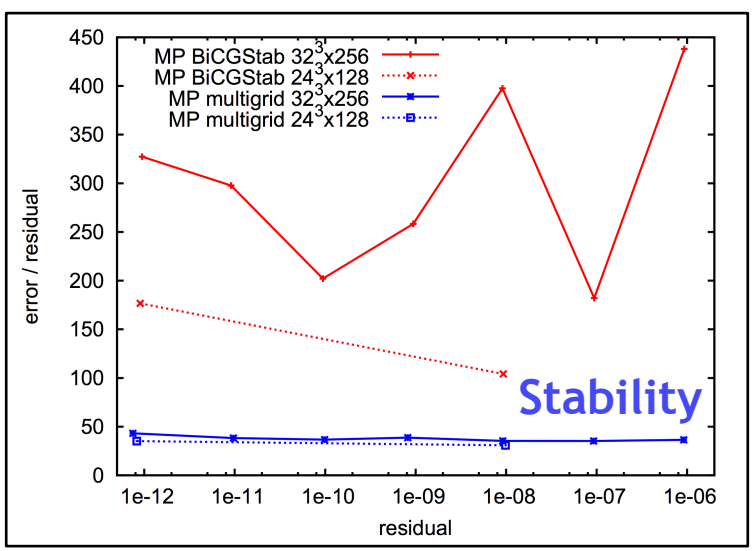
WHY MULTIGRID?



Babich *et al* 2010



Clark *et al* (2016)



Osborn *et al* 2010

Babich, Branich, Brower, **KC**, Manteuffel, McCormick, Osborn, Rebbi (2009)

ADAPTIVE GEOMETRIC MULTIGRID

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

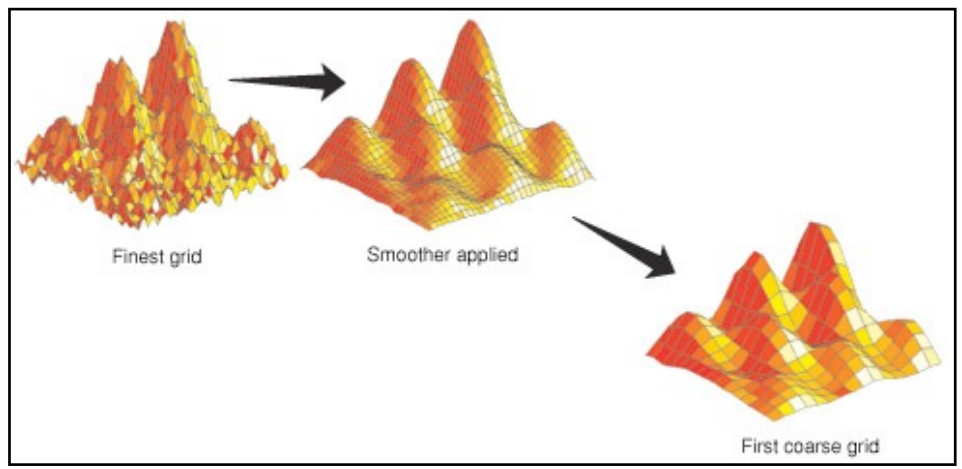
Adaptively find candidate null-space vectors

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

Specialized form of adaptive multigrid adapted for non-Hermitian LQCD problem with geometric coarsening

NVIDIA.

MIXED-PRECISION MULTIGRID SOLVER

Ideal algorithm for mixed precision

- Each MG pass only reduces the residual / error by an order of magnitude
- Deploy as a preconditioner for an outer Krylov solver

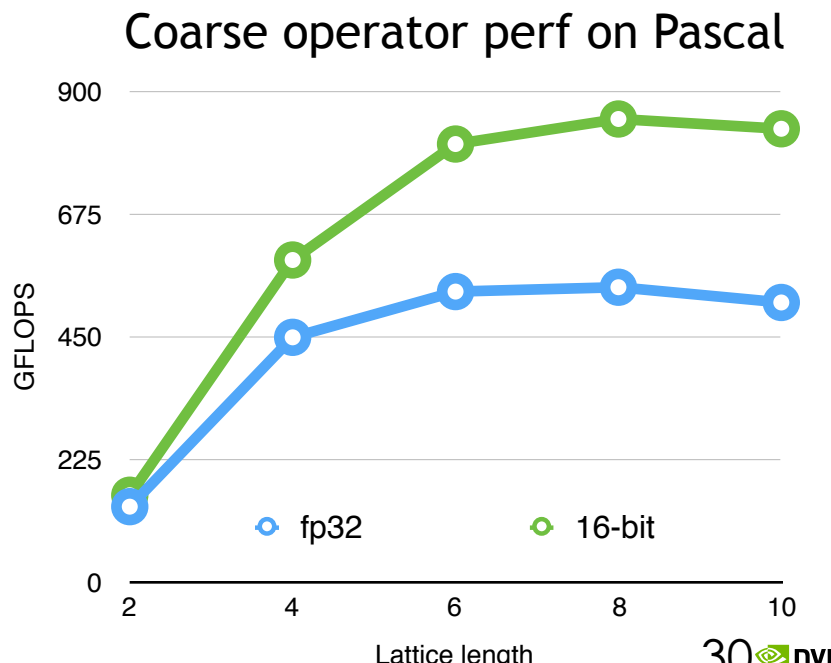
Method

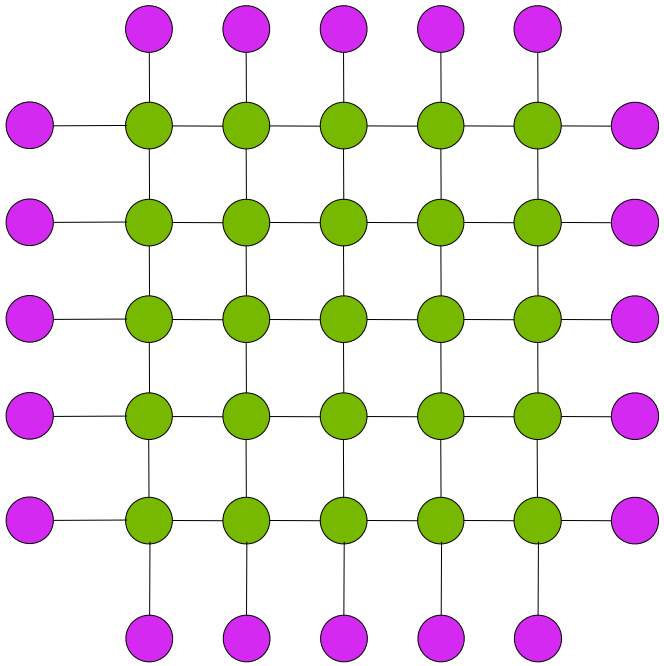
- Do entire MG cycle in 16-bit precision
- Wrapped by a single-precision GCR solver
- Use double-precision restarts to ensure convergence

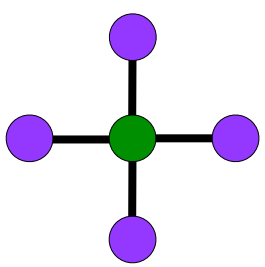
LQCD adaptive MG requires a lot of “state”

- 33% reduction in peak memory - can run on less nodes

Absolutely zero effect on multigrid convergence

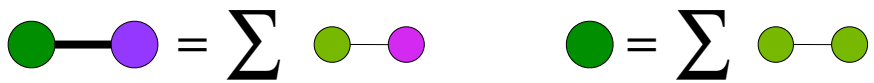






MIXED-PRECISION MULTIGRID SETUP

Need to evaluate RAP - pseudo batched triple matrix product



Coarse operator off diagonals Coarse operator diagonals

Need to employ fine-grained parallelization

- Perform each batched matrix product in parallel
- Each coarse matrix element has many contributions (many-to-one)

Cannot pose as a reduction so atomically update the coarse matrix elements

Floating point atomics are non-deterministic

Use 32-bit integer atomics for associativity and determinism

CHROMA HMC ON SUMMIT

KC, **Bálint Joó**, Mathias Wagner, Evan Weinberg, Frank Winter, Boram Yoon

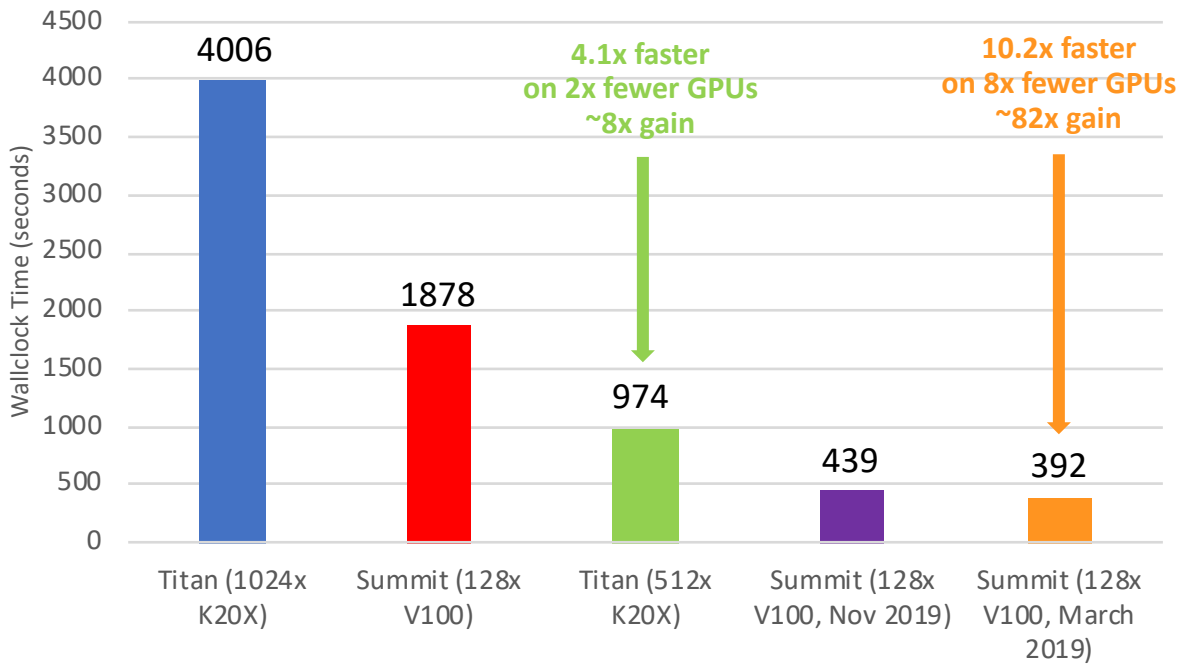
From Titan running 2016 code to Summit running 2019 code we see >82x speedup in HMC throughput

Multiplicative speedup coming from machine and algorithm

Highly optimized multigrid for gauge field evolution

Mixed precision an important piece of the puzzle

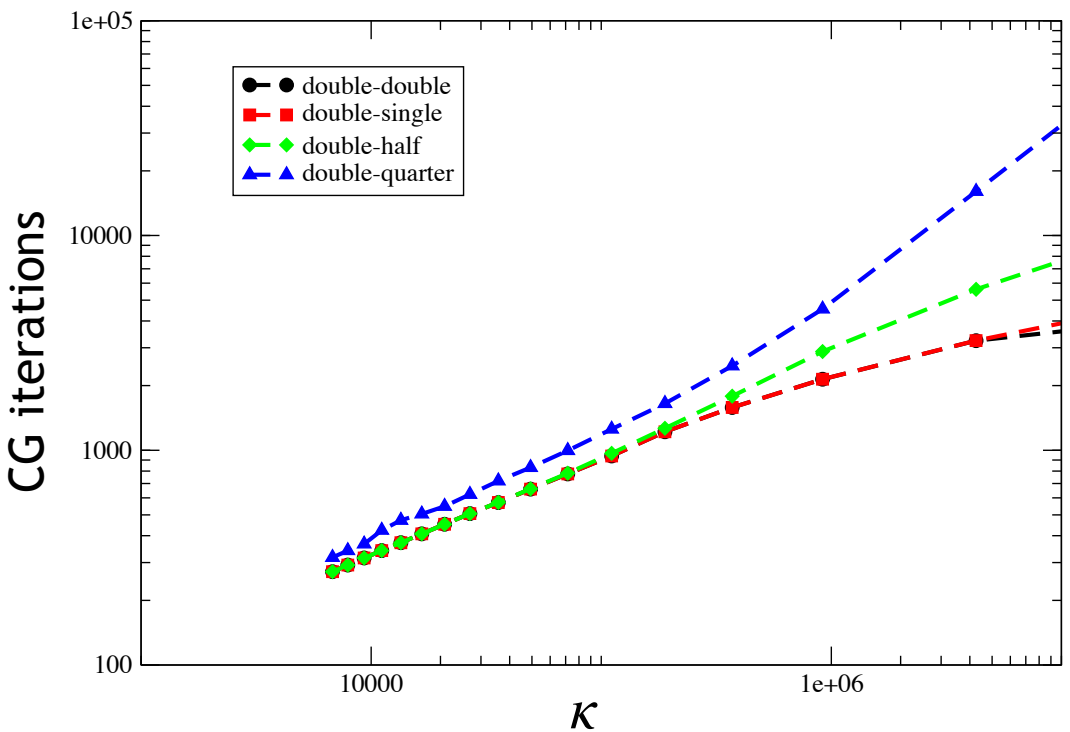
- double – outer defect correction
- single – GCR solver
- half – MG preconditioner
- int32 – deterministic parallel coarsening

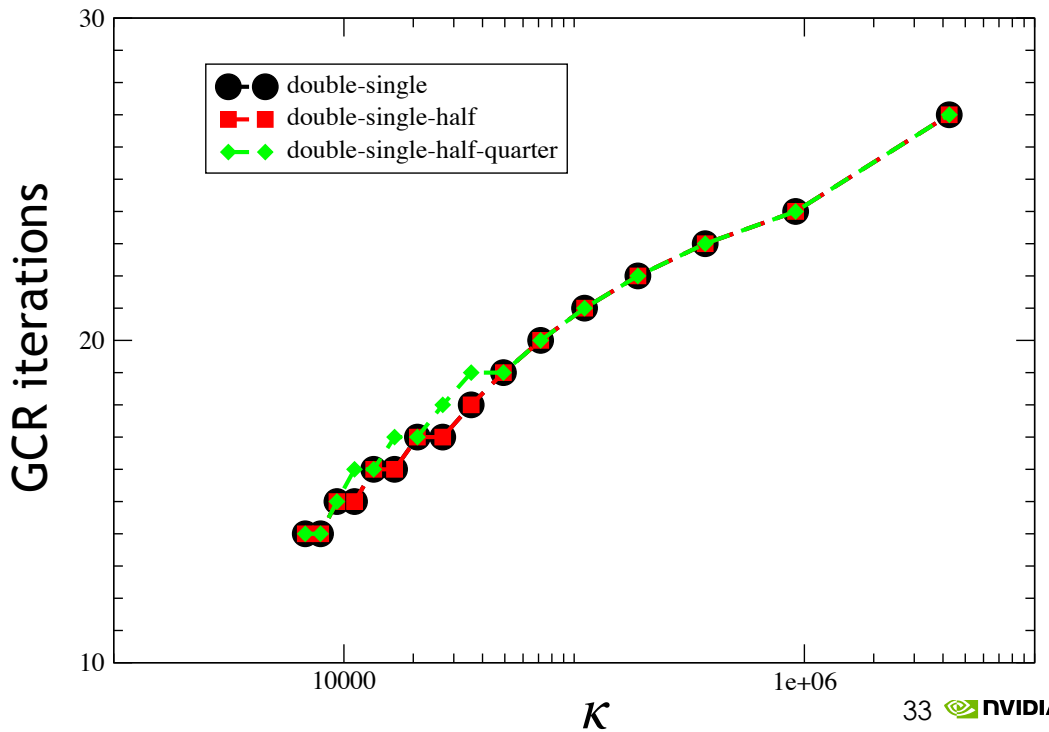


Data from B. Joó (Jefferson Lab). Chroma w/ QDP-JIT (F. Winter, Jefferson Lab) and QUDA. B. Joó gratefully acknowledges funding through the US DOE SciDAC program (DE-AC05-06OR23177)

HOW LOW CAN MULTIGRID GO?

Unlike CG, Multigrid stable with an 8-bit smoother





ONGOING AND FUTURE WORK

Plumb 8-bit into the full MG hierarchy

Use tensor-core accelerated direct solve for coarse grid (cf Dongarra)

- Motivation is a latency optimization
- Current: iterative local communication and computation => latency bound
- Future: single all gather collective and local direct solve


TENSOR CORES

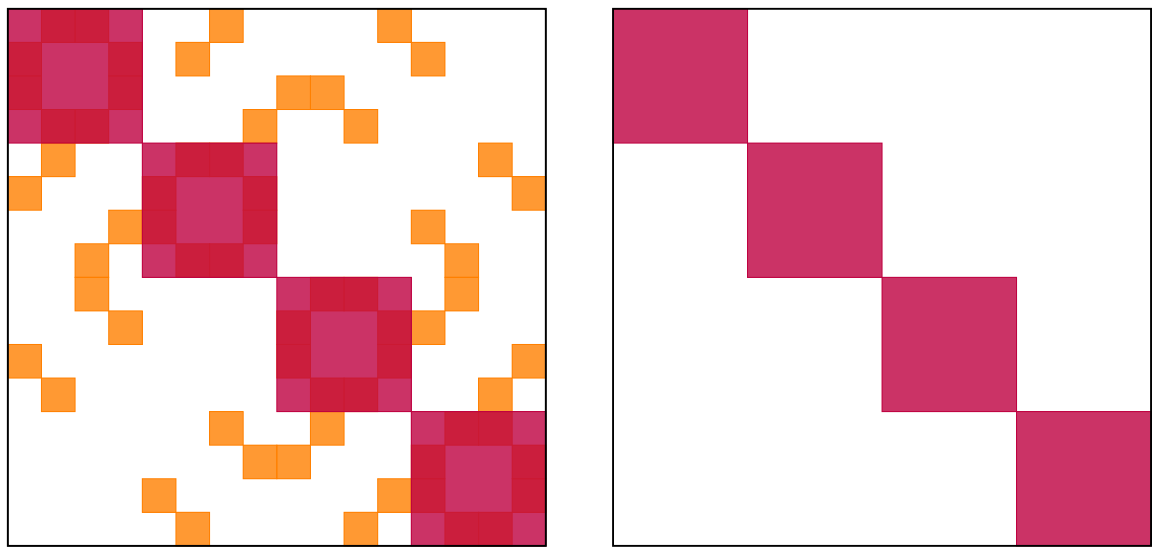
LATTICE QCD WITH TENSOR CORES

KC, Jung, Mawhinney, Tu

Follow up work from arXiv:1804.08593 (Guo, Mawhinney and Tu)

Multi-splitting Preconditioned CG

- Motivated by lack of network bandwidth in supercomputing centers (e.g., Summit)
- Block Jacobi preconditioner for (M)DWF that correctly applies the Dirichlet boundary condition for the red-black normal op

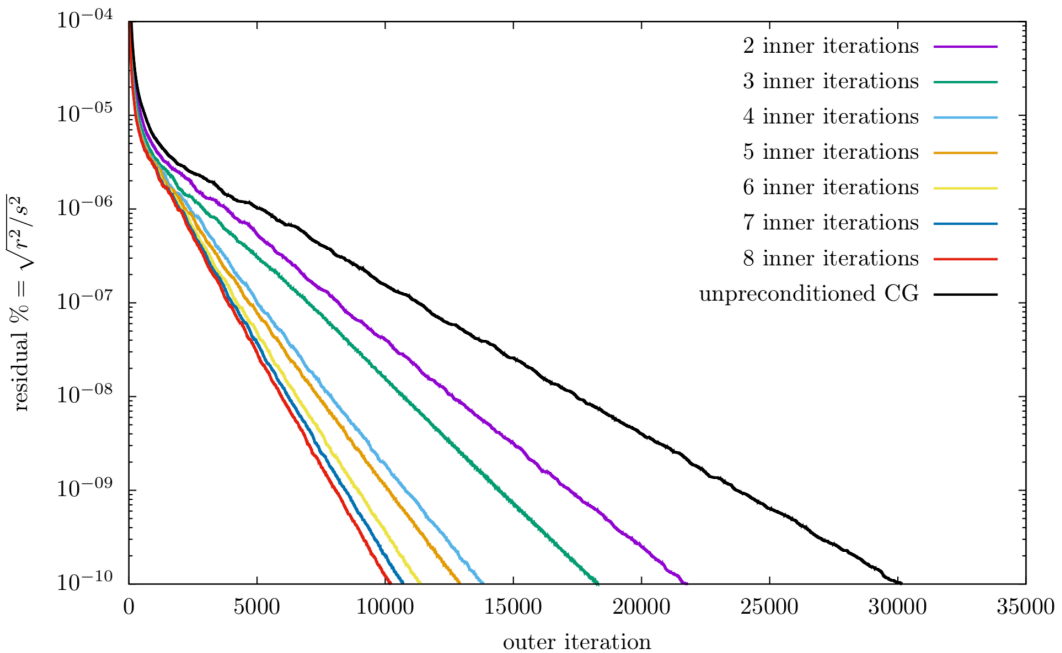


A $\quad P = \bigoplus_s A_s$

LATTICE QCD WITH TENSOR CORES

KC, Jung, Mawhinney, Tu

Significantly reduces outer iterations…

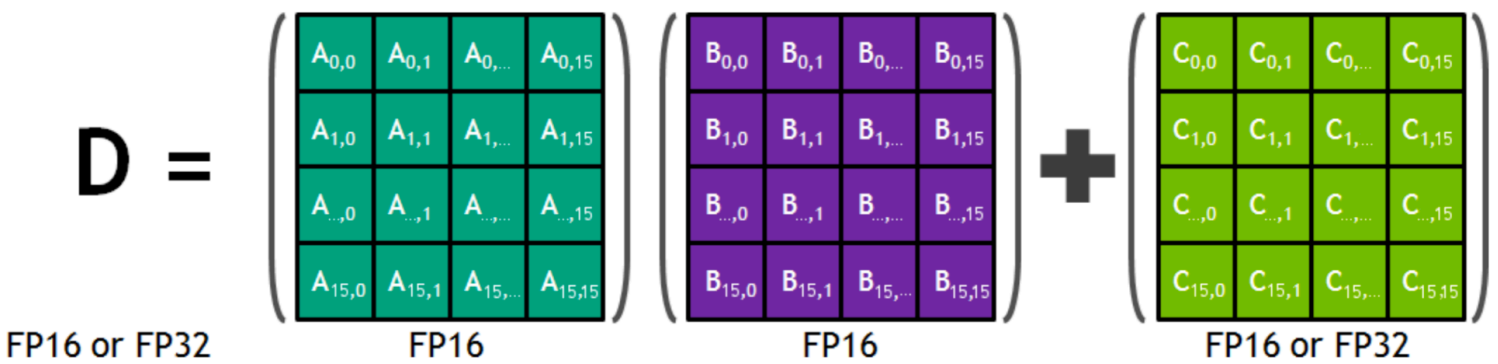


…but local preconditioner becomes prohibitively expensive

LATTICE QCD WITH TENSOR CORES

KC, Jung, Mawhinney, Tu

Modern GPU have high throughput tensor-core functionality



Tesla V100
FP64: 7.5 TFLOPS
FP32: 15 TFLOPS
Tensor: 125 TFLOPS

$$\left[1-\kappa_b^2 \underbrace{M_\phi^\dagger D_w^\dagger}_{\text{fuse}} \underbrace{M_5^{-\dagger} M_\phi^\dagger D_w^\dagger}_{\text{fuse}} \underbrace{M_5^{-\dagger}}_{\text{fuse}}\right]\left[1-\kappa_b^2 \underbrace{M_5^{-1} D_w}_{\text{fuse}} \underbrace{M_\phi M_5^{-1} D_w}_{\text{fuse}} M_\phi\right]$$

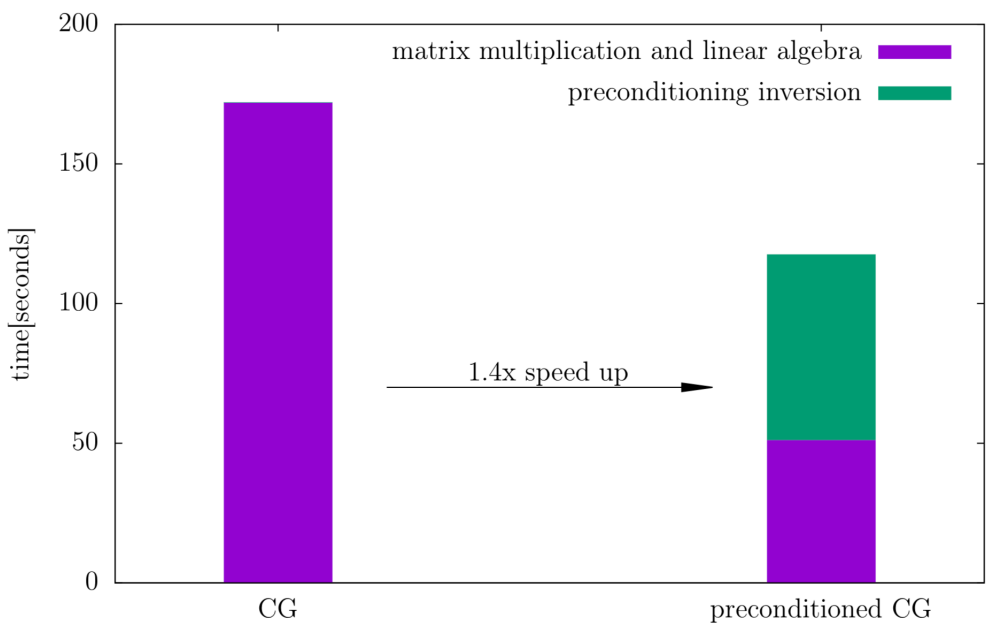
LATTICE QCD WITH TENSOR CORES

KC, Jung, Mawhinney, Tu

Increasingly beneficial as one strong scales

nodes	local volume	solver	inner iter.	(outer) iter.	r.u.	performance/node	time	speed up
256	$16 \cdot 24 \cdot 12 \cdot 24$	CG	–	42133	471	4.66	486.3	
		MSPCG	05	16903	195	1.56(01)/5.45(35)/37.29(53)	456.0	
		MSPCG	06	14860	173	1.56(01)/5.51(31)/37.60(58)	442.6	1.10x
		MSPCG	07	13787	161	1.56(01)/5.48(28)/37.49(60)	460.2	
		MSPCG	08	12922	151	1.56(01)/5.44(26)/37.55(63)	469.5	
512	$16 \cdot 12 \cdot 12 \cdot 24$	CG	–	42427	474	3.85	296.6	
		MSPCG	05	17625	203	1.26(01)/4.54(37)/36.21(52)	271.0	
		MSPCG	06	15425	179	1.27(01)/4.55(33)/36.26(57)	262.1	1.13x
		MSPCG	07	14409	168	1.26(01)/4.57(30)/36.39(60)	268.3	
		MSPCG	08	13597	159	1.27(01)/4.53(28)/36.35(63)	276.0	
1024	$16 \cdot 12 \cdot 12 \cdot 12$	CG	–	42482	474	2.93	195.2	
		MSPCG	05	18250	210	1.00(01)/3.68(34)/34.62(45)	183.3	
		MSPCG	06	15959	185	1.01(01)/3.68(35)/34.79(54)	159.7	1.22x
		MSPCG	07	14985	174	1.01(01)/3.68(32)/35.06(58)	163.6	
		MSPCG	08	14287	167	1.00(01)/3.69(29)/34.76(61)	169.1	

6144 GPUs (Summit)




FUTURE CHALLENGES

HOW HIGH DO YOU NEED TO GO?

Present state of art LQCD grid size ~128^3x256 => dof ~ 10^{10}

LQCD Hybrid Monte Carlo algorithms utilize a Metropolis acceptance test for accept/reject

- Involves a difference measurement of $\delta S = \psi^\dagger A_t^{-1} \psi - \psi^\dagger A_0^{-1} \psi$
- This is an extensive quality
- Presently require solver tolerance of ~10^{-12} to ensure high Metropolis acceptance rate

Not long before we have to consider beyond double precision

Already putting double-double (pseudo-quad) precision in QUDA to cover this eventuality

SCALING FURTHER

NVSHMEM gets the CPU out of the way

Increasingly latency limited when we reduce precision (and GPUs get faster)

- Overhead from calling MPI routines
- Halo-region updates do not saturate the GPU

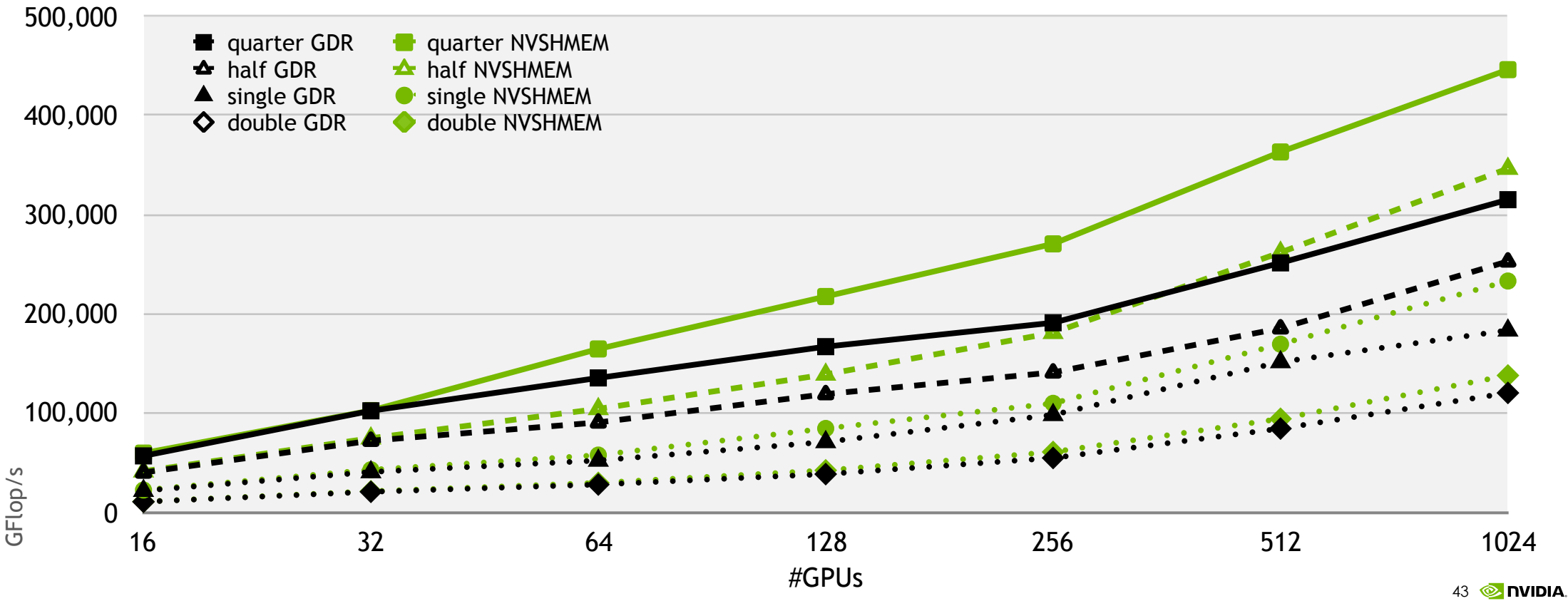
NVSHMEM: Implementation of OpenSHMEM, a Partitioned Global Address Space (PGAS) library

- Removing reliance on CPU for communication
- Parallelism for implicit compute – communication overlap
- Allows kernel-side communication (API and LD/ST) between GPUs
- Interoperability with MPI and OpenSHMEM libraries
- Improving performance while making it easier to program

Disclaimer: Results from an first implementation in QUDA with a pre-release version of NVSHMEM

MULTI-NODE STRONG SCALING

SuperPOD, Chroma stencil, 64^3x128 global volume



NVIDIA.

SUMMARY

SUMMARY

The state of the art for LQCD linear solver requires the use of mixed precision

Optimal Hierarchical solvers can require three or more different precisions

Some algorithms more amenable than others

New algorithms possible that were unfeasible before

Judicious choice of precision can lead to a significant speedup: more science


NVIDIA

RECOMPILE AND RUN