GPUS FOR LATTICE FIELD THEORY

Kate Clark, NVIDIA

AGENDA

Introduction and QUDA Retrospective

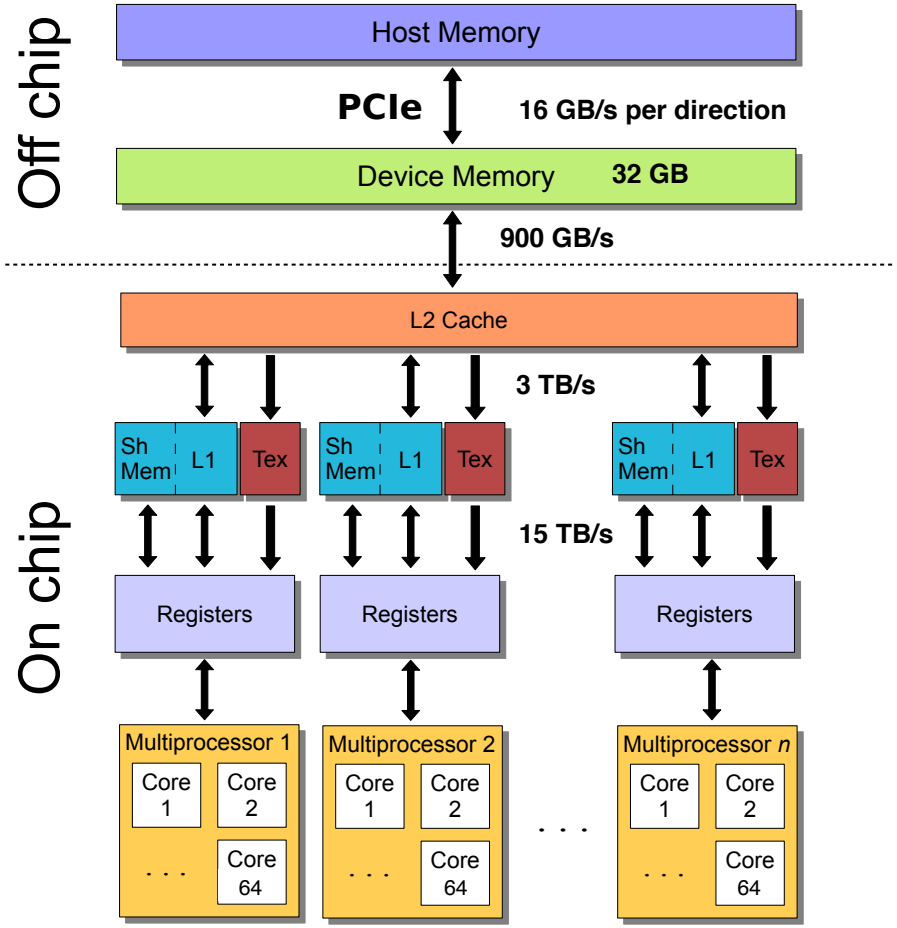
Recent Progress with Multigrid (on GPUs)

Strong scaling

The Community is Active

Some thoughts on the future

WHAT IS A GPU?



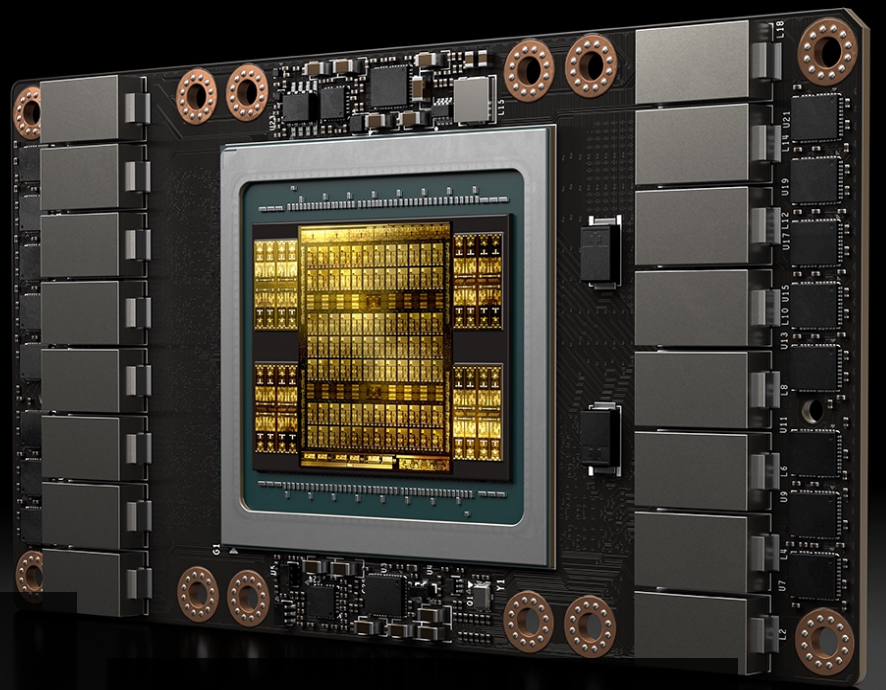
- Tesla V100 - Volta architecture (2017)
  - Massively threaded - 5120 processing cores
  - 7.5 FP64 /15 FP32 / 125 FP16 Gflops peak
- Deep memory hierarchy
  - As we move away from registers
    - Bandwidth decreases
    - Latency increases
- Inverse memory hierarchy
  - 40 MiB register file (up to 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`

NVIDIA POWERS WORLD'S FASTEST SUPERCOMPUTER

Summit Becomes First System To Scale The 100 Petaflops Milestone

122 PF
HPC

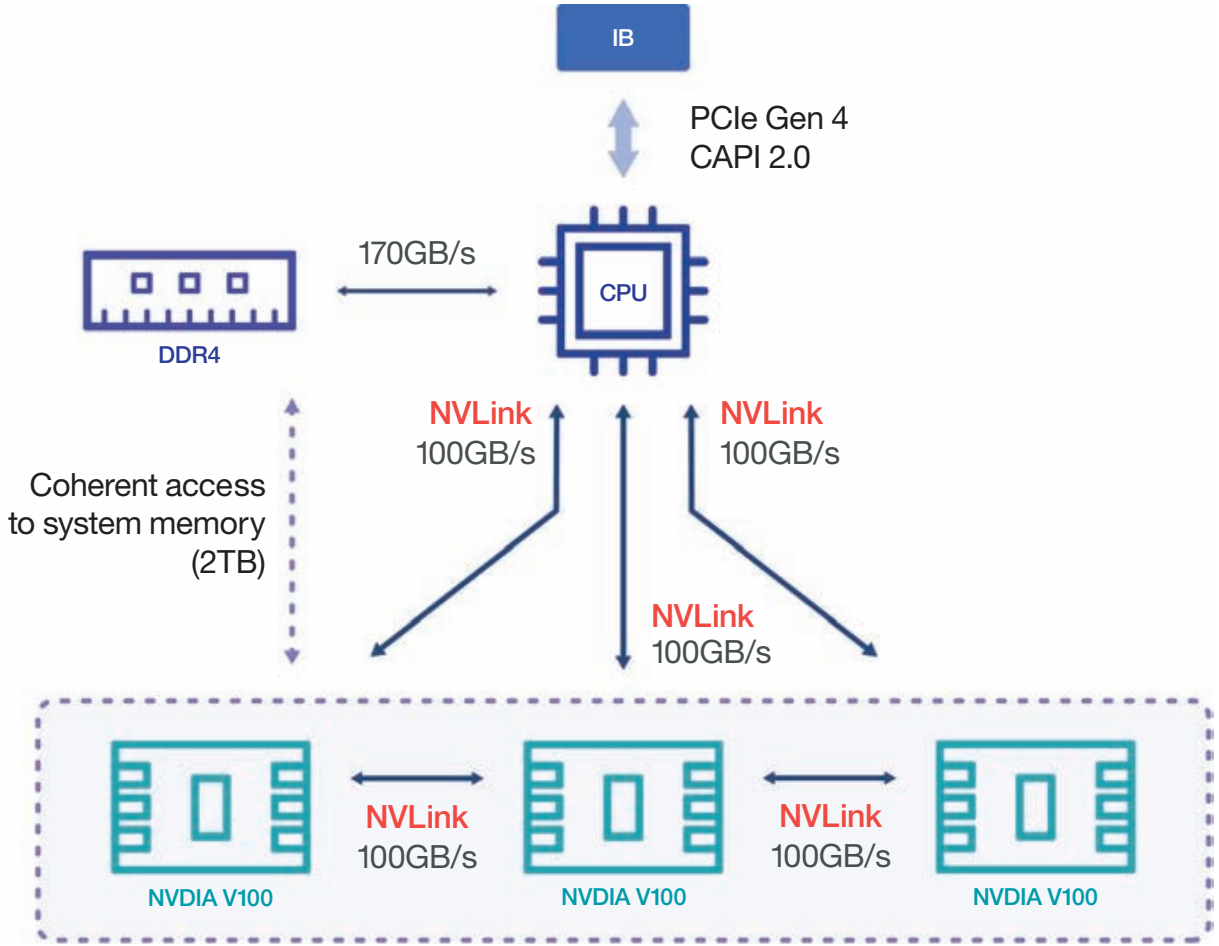
3 EF
AI



27,648
Volta Tensor Core GPUs

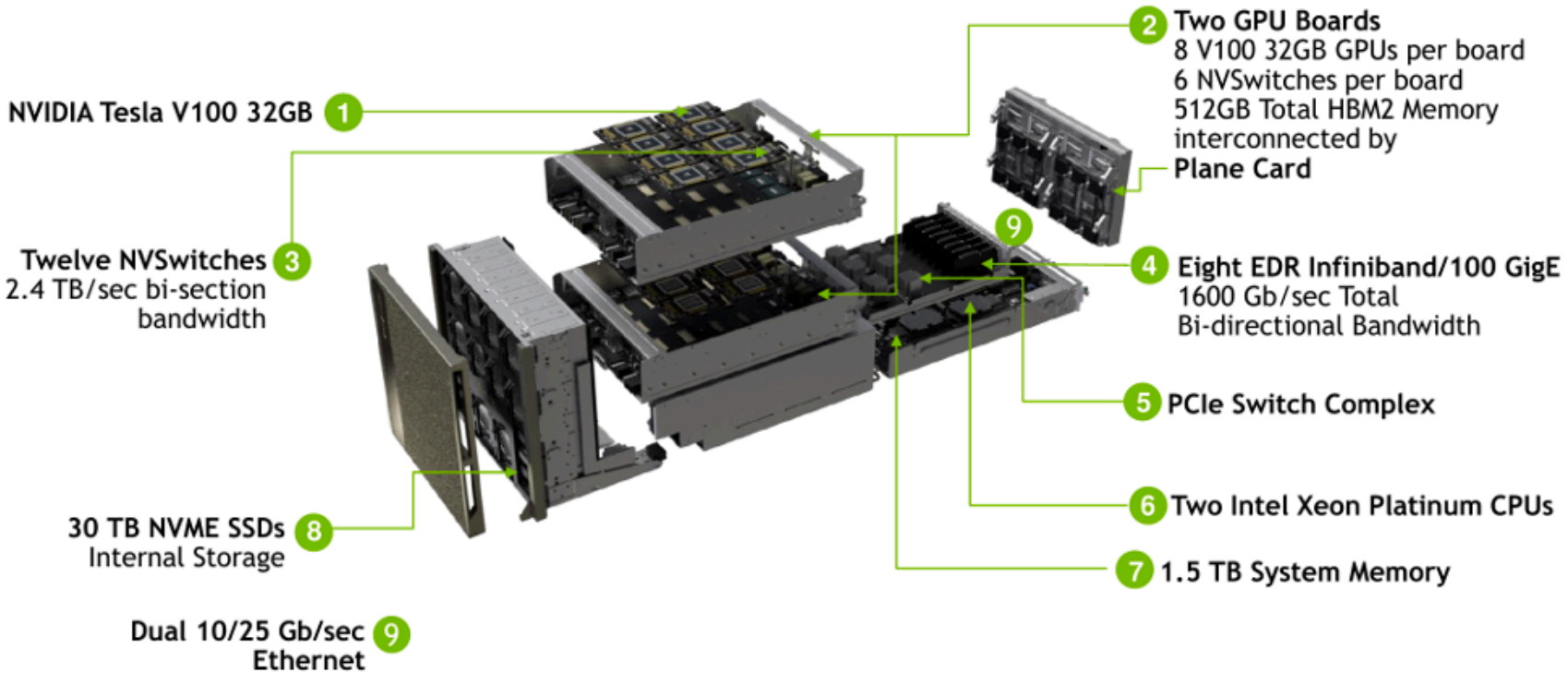
IBM AC 922

4/6 V100 GPUs
NVLink to GPU and P9
2 EDR IB



DGX-2

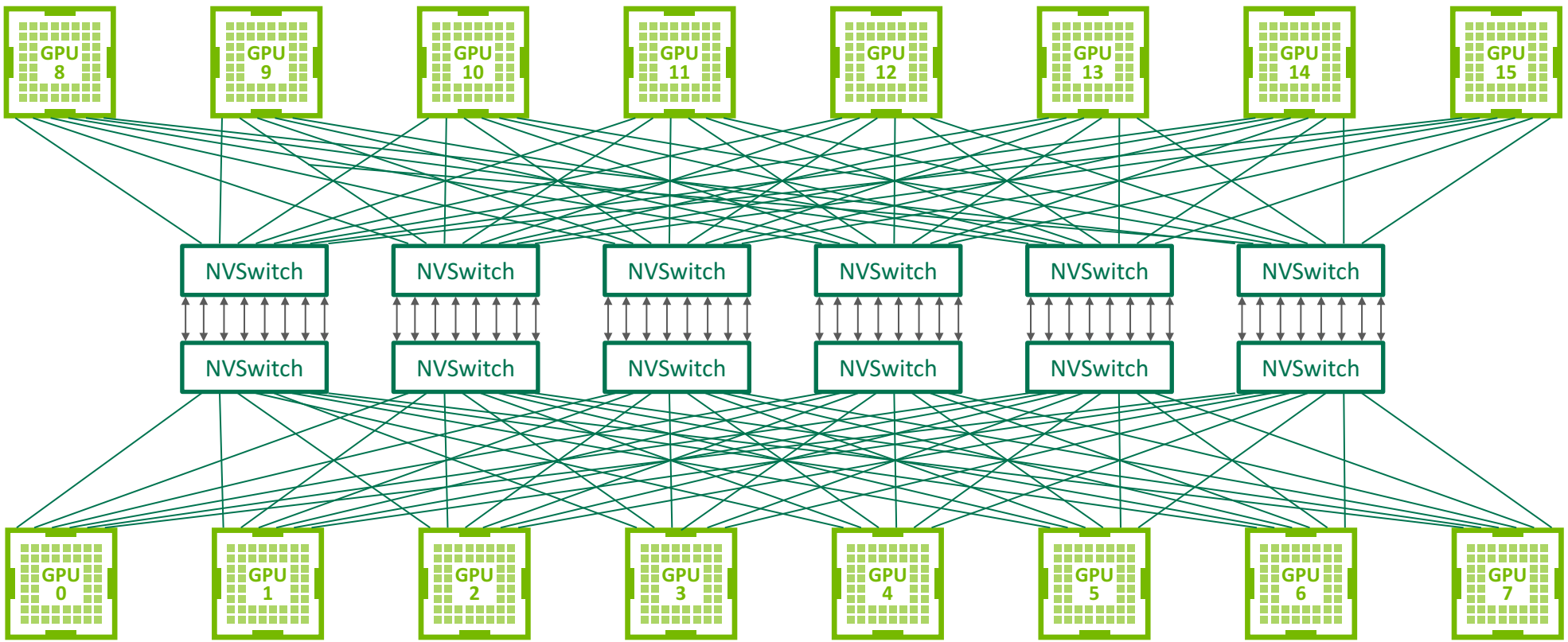
1 node supercomputer



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

DGX-2: FULL NON-BLOCKING BANDWIDTH

2.4 TB/s bisection bandwidth



SOME GPU CLUSTERS

Summit @ OLCF

#1 on TOP500

4600 IBM P9 nodes
6x Tesla V100 / node
100 GB/s NVLink
2x EDR IB / node

Piz Daint @ CSCS

#6 on TOP500

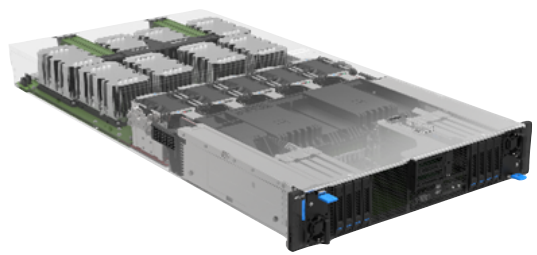
5272 Cray X50 nodes
1x Tesla P100 / node
Cray Aries

DGX SuperPOD @ NVIDIA

#22 on TOP500

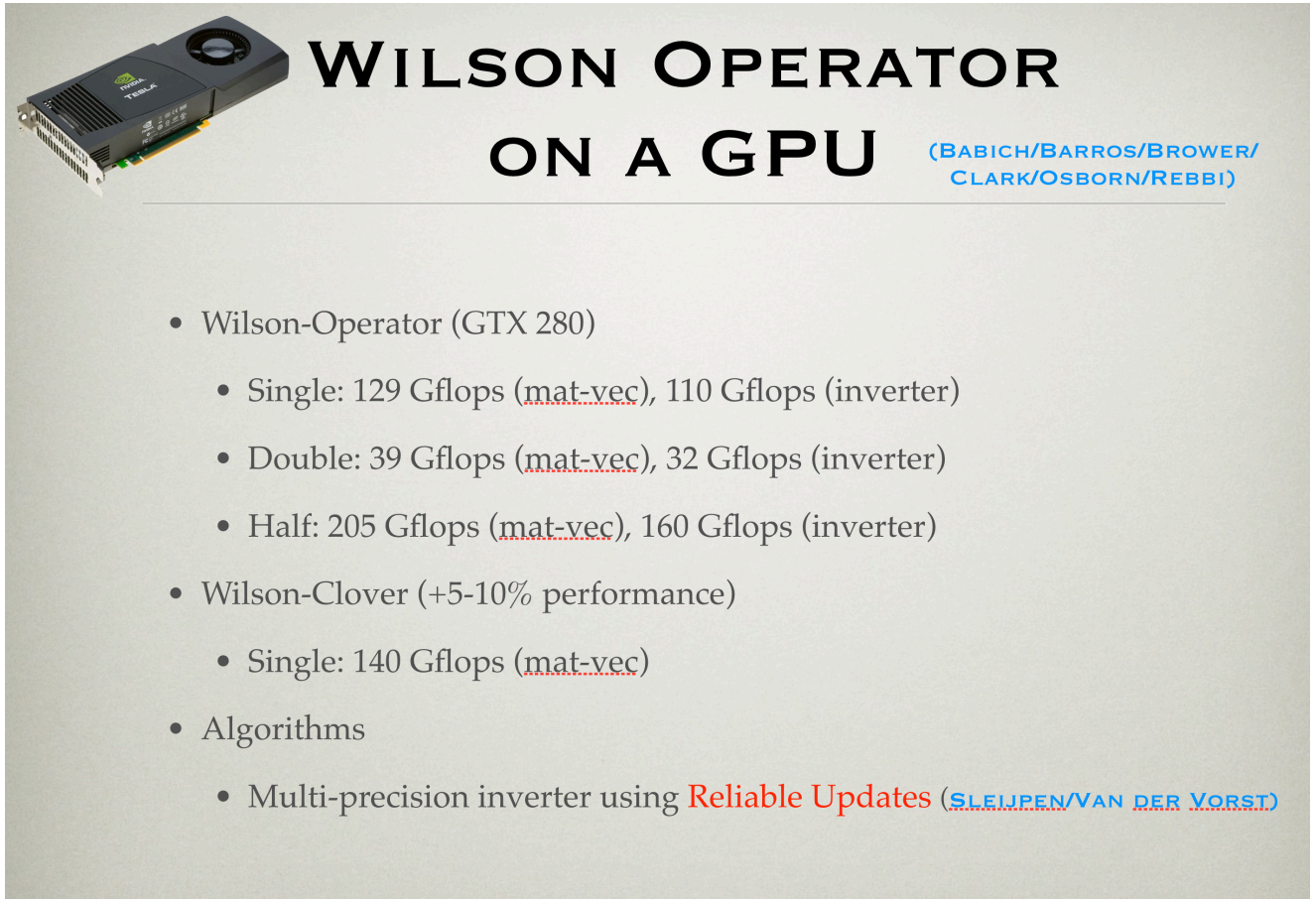
96 DGX-2 nodes
16x Tesla V100 / node
300 GB/s NVLink P2P
8x EDR IB / node

NSC3 @ CCNU



18 AGX-2 nodes
8 Tesla V100 / node
100 GB/s NVLink P2P

GPU COMPUTING FOR LQCD, CIRCA 2009



plenary Beijing 2009

GPUs for LQCD were bleeding edge

~100 GFLOPs per GPU

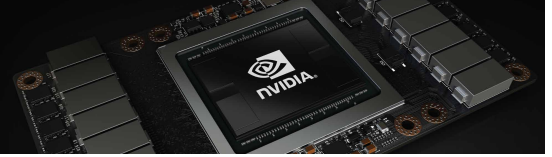
Single GPU only

Mixed-precision Krylov solvers were the state of the art

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

LATTICE QCD ON NVIDIA® TESLA® V100



Authors: Kate Clark, Mathias Wagner, Evan Weinberg, NVIDIA

QUDA 1.0

QUDA: A LIBRARY FOR QCD AND BEYOND ON GPUs

- QUDA is an open source community-developed and NVIDIA-supported library for performing LQCD and strongly coupled BSM calculations on GPUs, leveraging NVIDIA's CUDA platform. QUDA provides highly optimized mixed-precision linear solvers, eigenvector solvers, multi-grid algorithms, gauge-link fattening and fermion force algorithms
- Supported fermion types are: Wilson, Wilson-clover, twisted mass singlet and doublet, twisted mass clover, naïve staggered, improved staggered (ASQTAD or HISQ), domain-wall (4-d or 5-d) and möbius
- Use of multiple GPUs in parallel is supported throughout the library, with inter-GPU communication achieved using MPI or QMP. Several commonly used LQCD applications integrate support for QUDA as a compile-time option, including Chroma, MILC, CPS, BQCD, TIFR and tmLQCD
- QUDA is an unparalleled research tool for reaching the exascale. With its new high-level C++11 framework, QUDA enables testing new fermion discretizations and new algorithms performantly and at scale with relative ease

WHY GPUs?

- LQCD simulations are typically memory-bandwidth bound, and so run very efficiently on GPUs
- LQCD simulations have high degrees of data parallelism that can be expressed effectively using the single instruction multiple data (SIMD) paradigm. This makes LQCD ideal for GPU deployment
- Most LQCD calculations require only local communication on the 4-d lattice. This makes them suitable for deployment on multiple GPUs through partitioning the lattice into disjoint equal sub-lattices and distributing these between GPUs
- GPUs are prevalent on the fastest computing clusters and supercomputers in the world

ANNOUNCING QUDA 1.0

We are pleased to announce QUDA 1.0

- Complete re-write of all operators in a **new Dslash framework**
- **Jitify support:** huge reduction in compile time, enabling even more rapid algorithm development
- Improved Multi-GPU performance taking advantage of latest GPU Direct RDMA and NVLink technologies
- Adaptive Multigrid for Wilson, Wilson-clover, Twisted-mass, Twisted-clover, **Staggered, and Improved Staggered** fermions
- A **polynomial-accelerated Lanczos** as well as deflated CG, communication-avoiding (CA) CG, and SVD-deflated (CA-)GCR
- Significantly improved build system and automated unit tests
- Various other new routines and algorithms, code cleanup and bug fixes

QUDA REWRITE

- Old Dslash kernel code became increasingly limiting
- Rewrite brings a lot of benefits:
  - Extensibility, composability and maintainability: "one Wilson kernel to rule them all"
  - Ability to add new discretizations easily: e.g. Twisted clover doublet
  - Changing representation, N_c etc: Accessor abstraction
  - Larger local volumes
  - Ability to run on CPU: Future framework for all architectures??
- Same or better performance

ROAD TO THE FUTURE

- Block solvers for all fermion actions
- Eigenvector compression
- Continuous optimization of multi-grid algorithms
- Improved scaling: NVSHMEM
- C++ Interface
- Alternative compilation targets (e.g., C++17 pSTL)

TESLA VOLTA V100

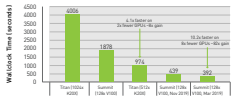
SUMMIT AND SIERRA ARE NOW LIVE

- IBM POWER9 + NVIDIA Volta V100 + NVLink
- Summit: 4608 nodes (6 GPUs/node): 200 PFLOPS peak
  - The most powerful computer in the world
- Sierra: 4320 nodes (4 GPUs/node): 125 PFLOPS peak
  - 2nd most powerful computer in the world

CHROMA HMC-MG ON SUMMIT

In collaboration with Bálint Joó – Chroma, Boram Yoon – Force gradient integrator, Frank Winter – QDP-JIT

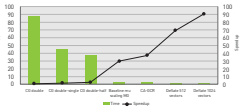
- From Titan running 2016 code to Summit running 2019 code we see >82x speedup in HMC throughput
- Multiplicative speedup coming from machine and algorithm innovation
- Highly optimized multigrid for gauge field evolution



DEFLATED ADAPTIVE MG AT THE PHYSICAL POINT

In collaboration with Dean Howarth (BU)

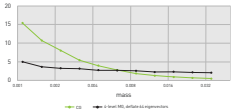
- Multigrid shifts the lowest eigenvalues to the coarse grids
- Some Dirac operators (staggered and twisted mass) end up with pathological coarse-grid spectrum
- For Twisted-clover operator, solution has been to add a fictitious heavy twist to the coarse operator to improve its condition number at the cost of decreased multigrid efficiency (*Alexandrou et al*)
- Instead we deflate the coarse grid operator recovering optimal MG convergence and a 3x speedup over "mu scaling"



Time to solution is measured running QUDA on 4 DGX-1 (32 V100) for solving the Twisted-mass + clover operator against a random source on a 64³x128 lattice, B = 1.778, κ = 0.139427, μ = 0.000720, solver tolerance 10^{-7}

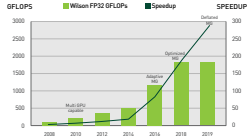
HISQ MULTIGRID

48³x96, B = 6.4 (quenched), HISQ fermions



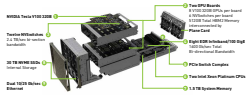
Time to solution is measured time for solving the HISQ operator against a random source. Runs were performed on 3 DGX-1 (24 V100 total).

QUDA NODE PERFORMANCE OVER TIME



Time to solution is measured for solving the Wilson operator against a random source on a 24x24x24x64 lattice, β=5.5, M= 4.16 MeV. One node is defined to be 3 GPUs.

NVIDIA DGX-2

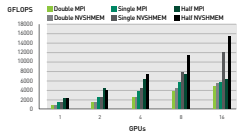


NVSHMEM

- Implementation of OpenSHMEM standard for GPUs
- GPU-centric communication
- Communication controlled in the kernels
  - Fine-grained load/store over NVLink
  - Bulk transfers over Infiniband using RDMA
- Reduce number of required kernels by enabling kernel fusion
- Remove API overheads and GPU-CPU synchronizations

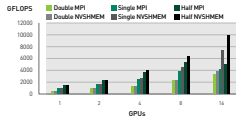
MULTI-GPU WILSON DSLASH

Global Volume = 32³, DGX-2, NVLink + NVSwitch



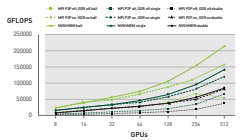
MULTI-GPU HISQ DSLASH

Global Volume = 32³, DGX-2, NVLink + NVSwitch



MULTI-NODE WILSON DSLASH

Global Volume = 64³x128, DGX1 with 8 V100, NVLink, 4 EDR IB



REFERENCES

https://lattice.github.io/quda
https://developer.nvidia.com/cuda-zone
https://devblogs.nvidia.com/parallelforall/inside-volta/

TALKS

Tuesday, 15:20, Algorithms & Machines: Leadership-Class Multi-Grid Algorithms for HISQ Fermions on GPUs.
Tuesday, 15:40, Algorithms & Machines: Breaking the latency barrier: Strong scaling LQCD on GPUs.
Thursday, 11:15, Plenary: GPUs for Lattice Field Theory.

NVIDIA

QUDA 1.0

This week we announced QUDA 1.0

Complete rewrite in modern C++

11 years in the making

See our poster for more details

Still many problems to solve in Lattice Field Theory…

Always wanting to build upon our collaboration with the community…come and join the fun

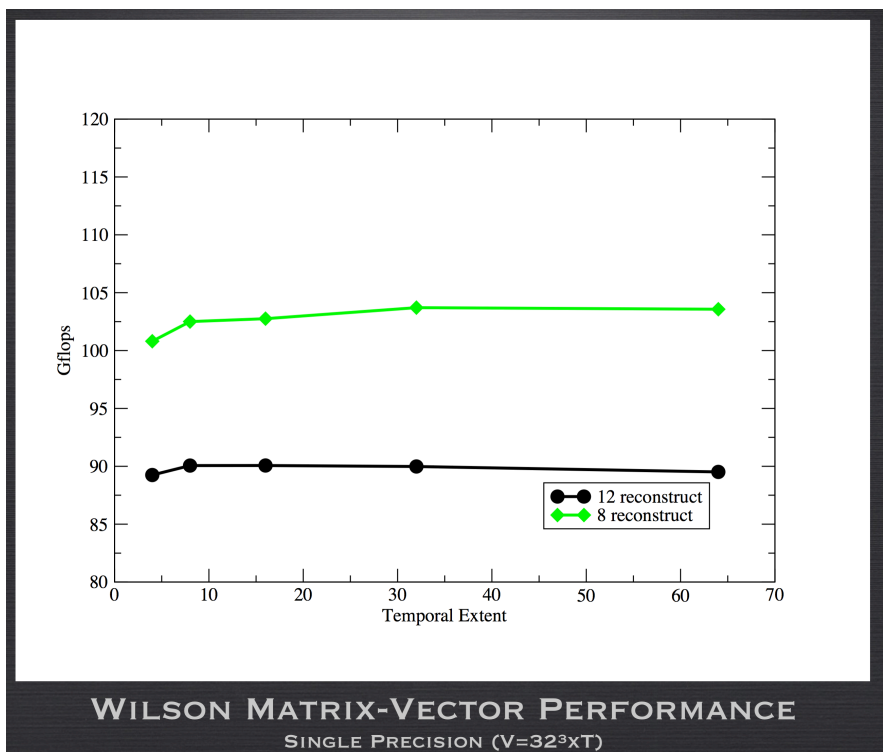
NVIDIA

QUDA CONTRIBUTORS

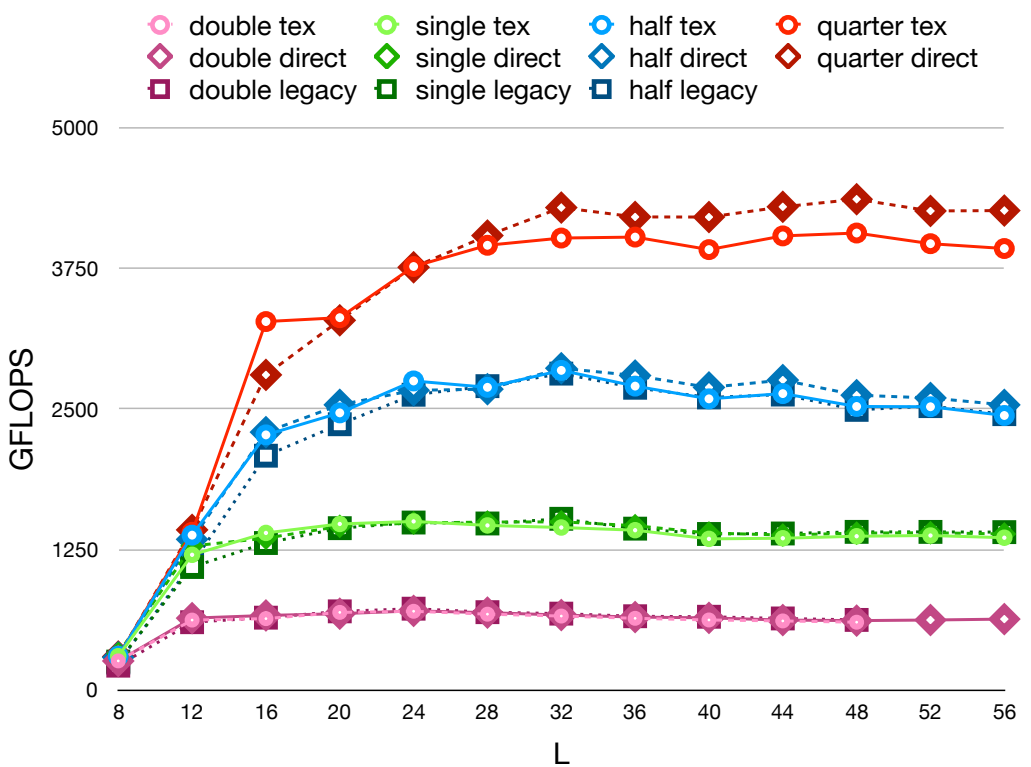
10+ years - lots of contributors

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)

RAW GPU PERFORMANCE

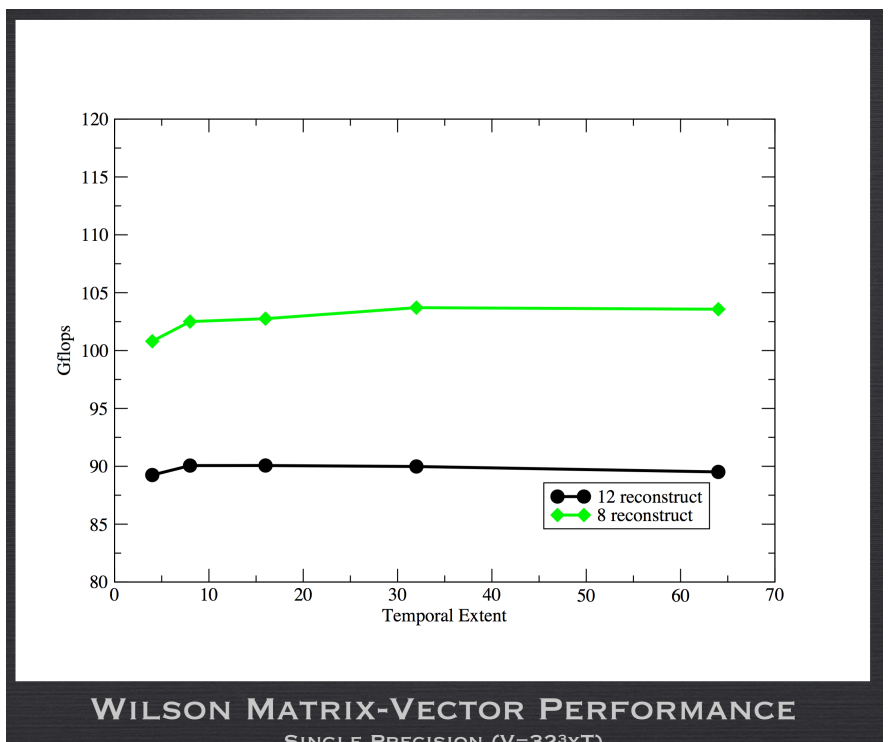


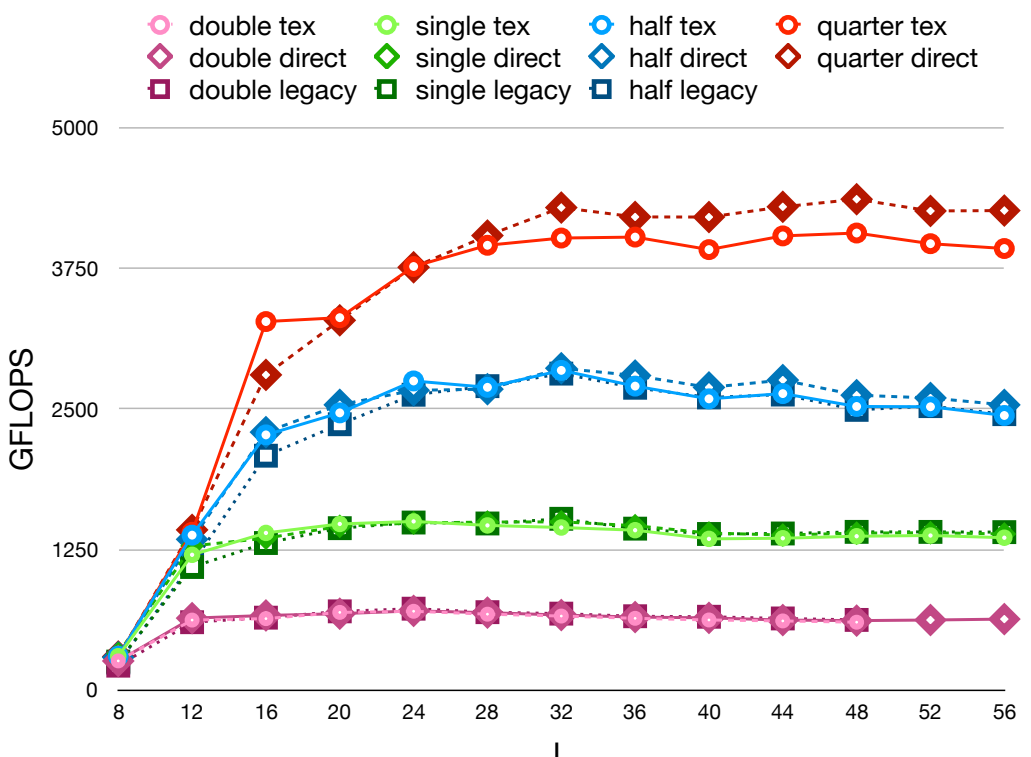
QUDA 0.1 on Tesla



QUDA 1.0 on Volta

RAW GPU PERFORMANCE OVER TIME

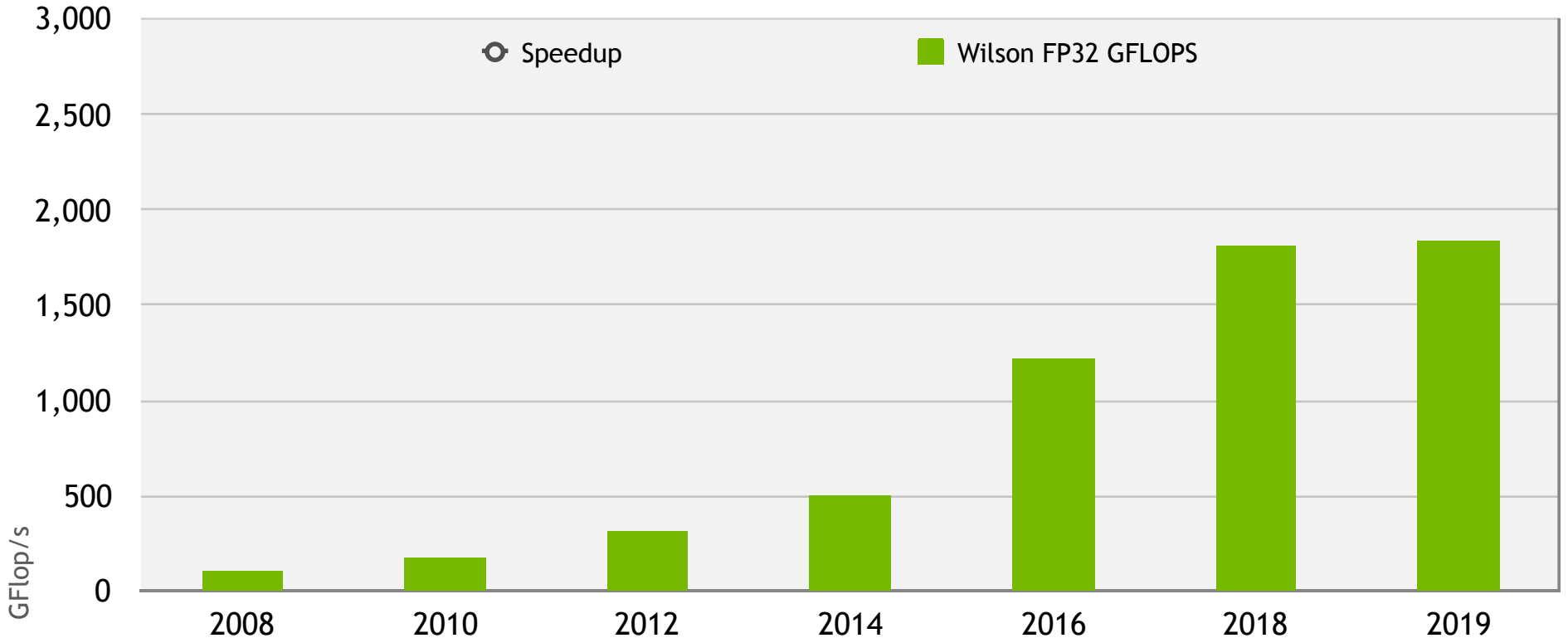




15x in 10 years
(Moore's law would have been >32x)

QUDA NODE PERFORMANCE OVER TIME

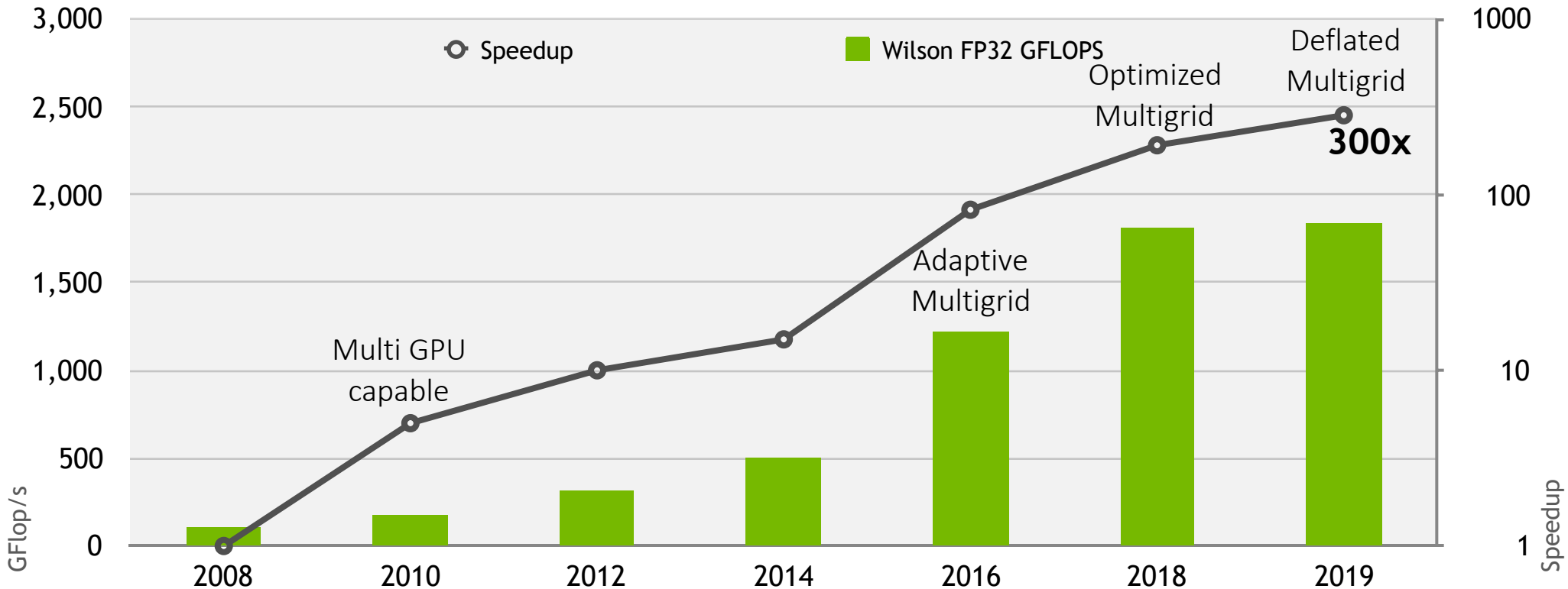
Multiplicative speedup through software and hardware



Speedup determined by measured time to solution for solving the Wilson operator against a random source on a $V=24^3 64$ lattice, $\beta=5.5$, $M_\pi = 416$ MeV. One node is defined to be 3 GPUs

QUDA NODE PERFORMANCE OVER TIME

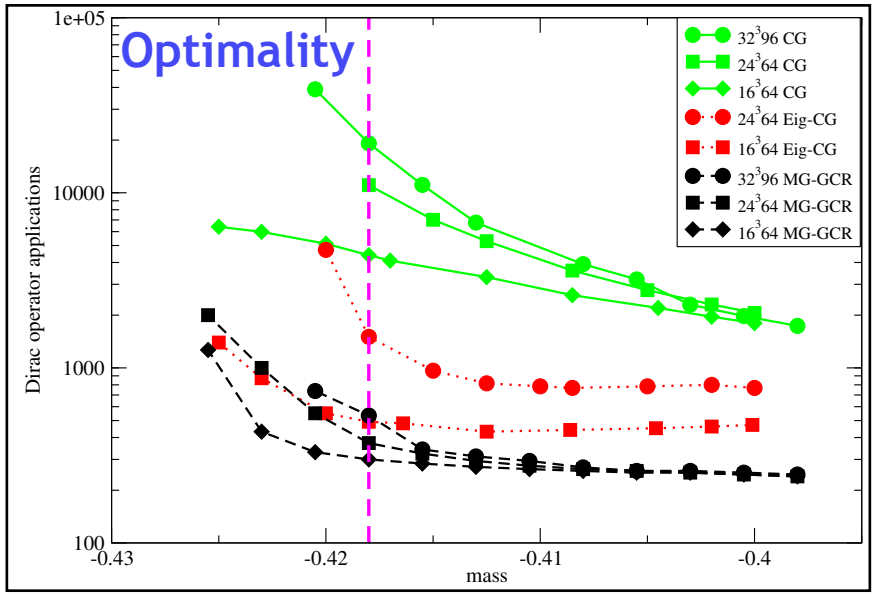
Multiplicative speedup through software and hardware



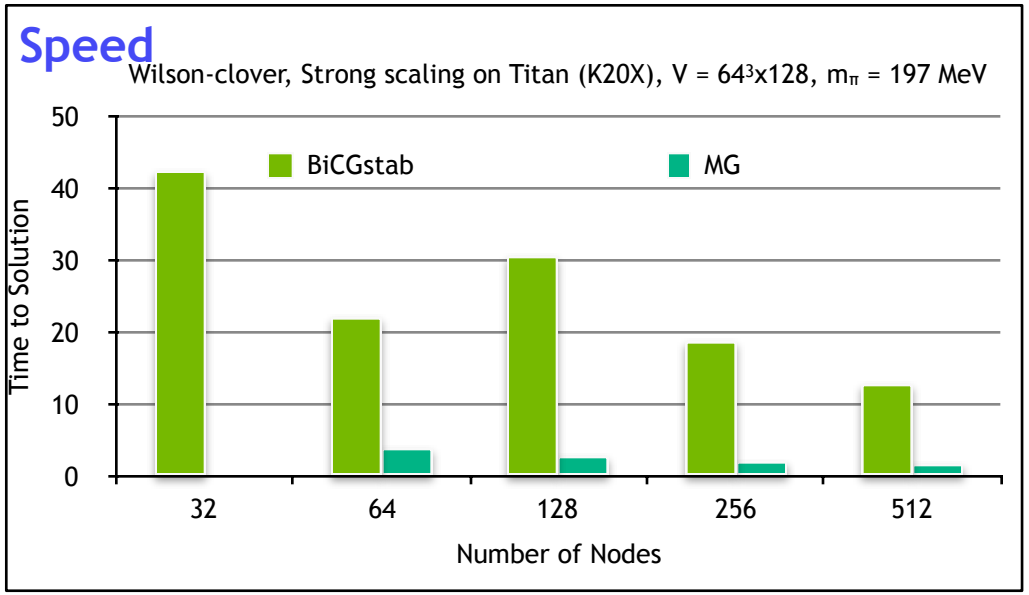
Speedup determined by measured time to solution for solving the Wilson operator against a random source on a $V=24^3 64$ lattice, $\beta=5.5$, $M_\pi = 416$ MeV. One node is defined to be 3 GPUs


RECENT PROGRESS WITH
MULTIGRID SOLVERS

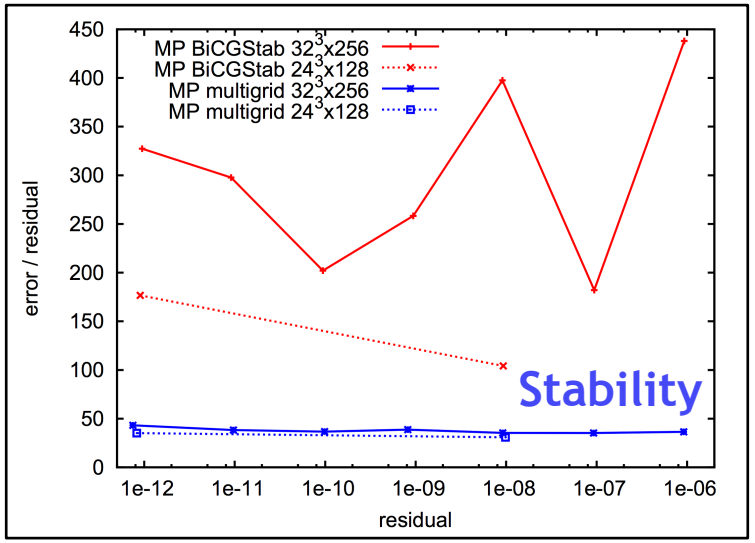
WHY MULTIGRID?



Babich *et al* 2010



Clark *et al* (2016)



Osborn *et al* 2010

CHROMA HMC ON SUMMIT

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

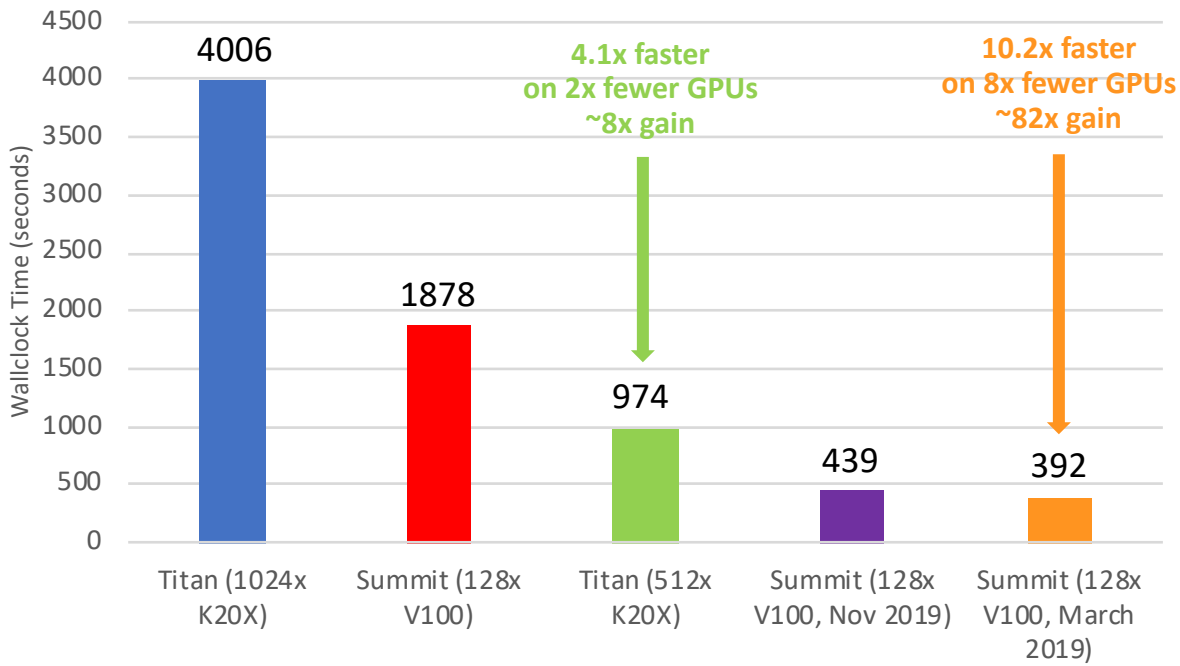
From Titan running 2016 code to Summit running 2019 code we see >82x speedup in HMC throughput

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 – preconditioner
- int32 – deterministic parallel coarsening



DEFLATED MG AT THE PHYSICAL POINT

KC, Dean Howarth and Evan Weinberg

Multigrid shifts the lowest eigenvalues to the coarse grids

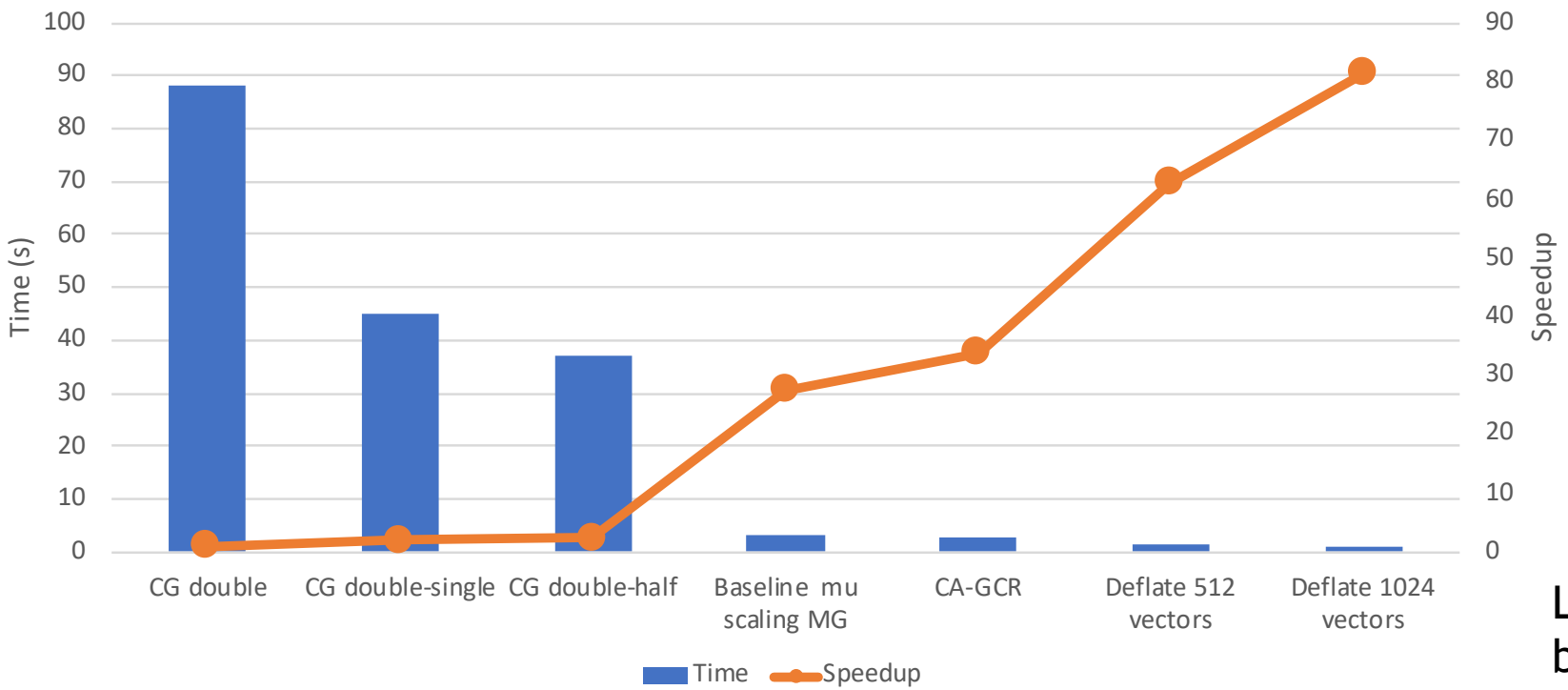
Some Dirac operators (e.g., staggered and twisted mass) end up with pathological coarse-grid spectrum

For Twisted-clover operator, solution has been to add a fictitious heavy twist to the coarse operator to improve its condition number at the cost of decreased multigrid efficiency [*Alexandrou et al]*

Instead we deflate the coarse grid operator recovering optimal MG convergence and a 3x speedup over “mu scaling”

DEFLATED MG AT THE PHYSICAL POINT

KC, Dean Howarth and Evan 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}

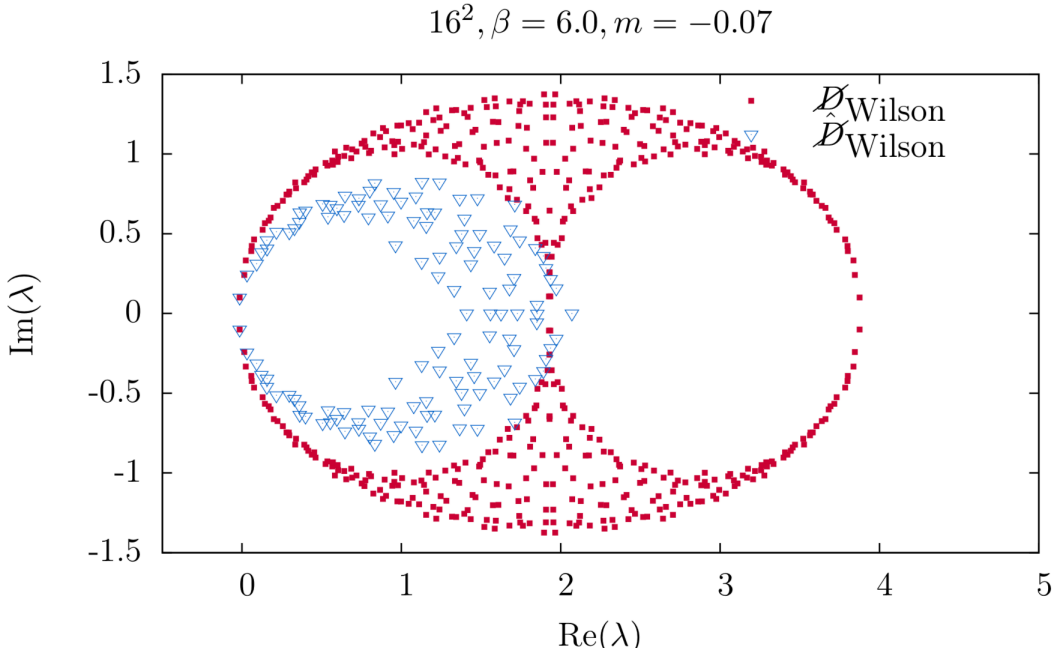
arXiv:1801.07823

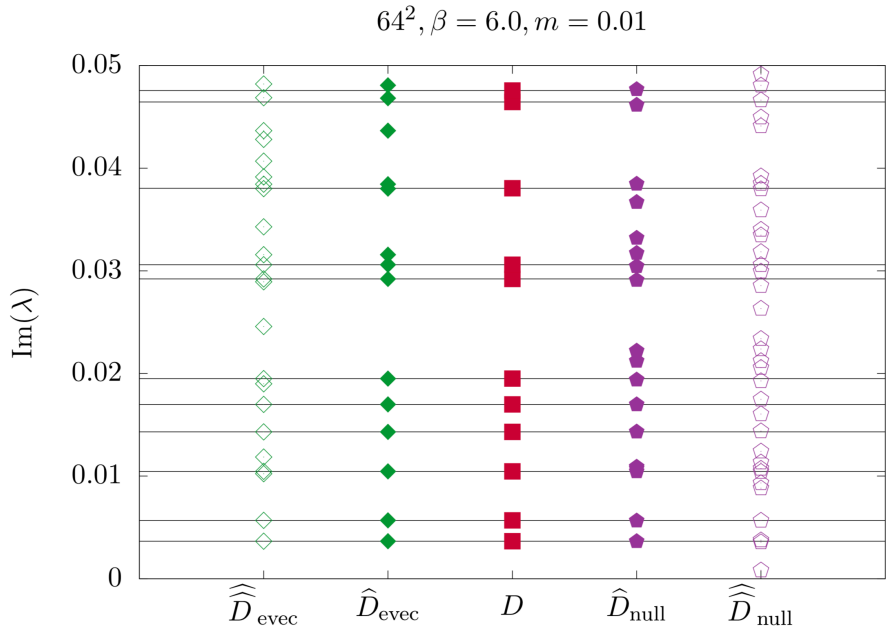
STAGGERED MULTIGRID

Rich Brower, KC, Dean Howarth, Alexei Strelchenko, **Evan Weinberg** (Tuesday 15:20)

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

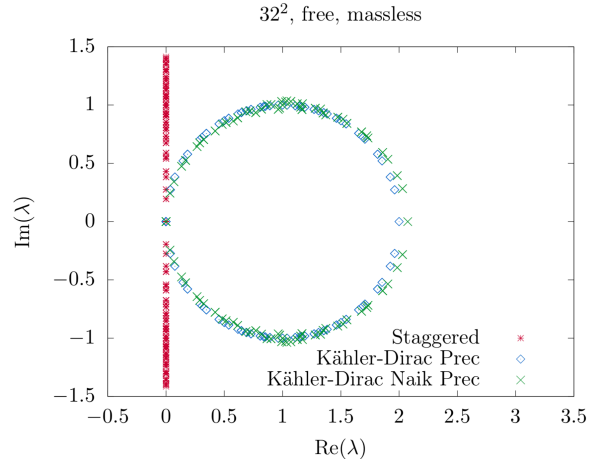
STAGGERED MULTIGRID

Rich Brower, KC, Dean Howarth, Alexei Strelchenko, **Evan Weinberg** (Tuesday 15:20)

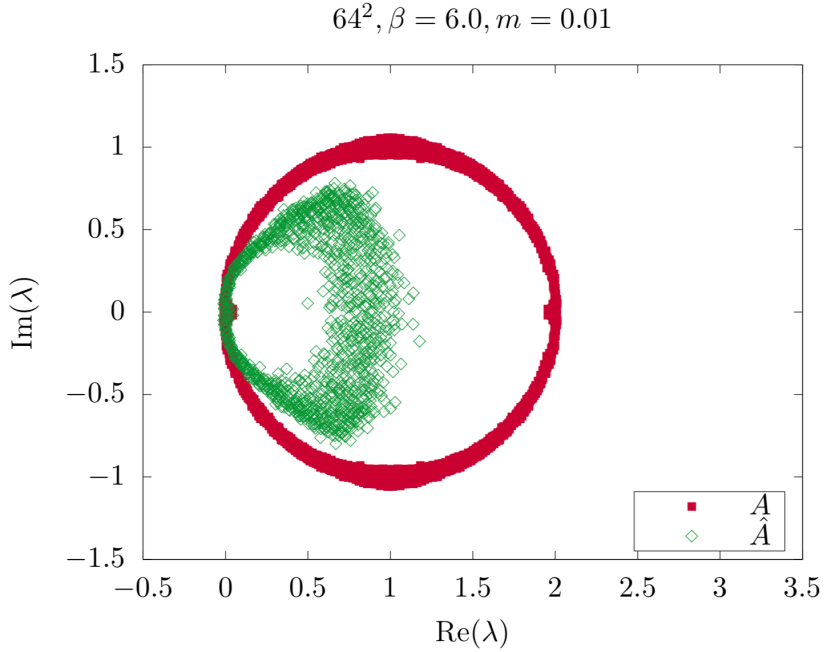
Transform into Kahler-Dirac form (arXiv: 0509026 Dürr) through unitary transformation

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

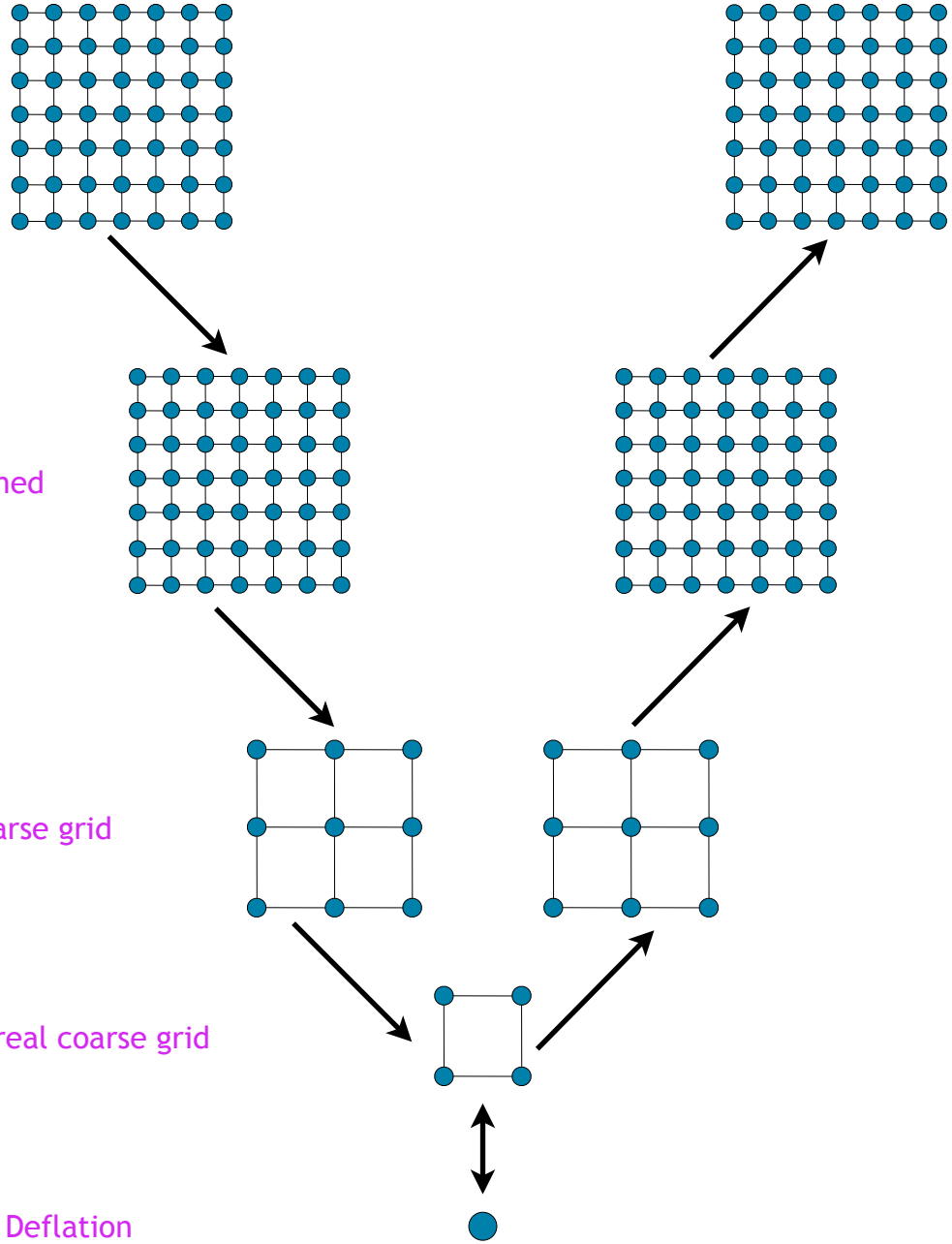
"Precondition" the staggered operator by the Kahler-Dirac block



No spurious low modes as we coarsen



HISQ



KD preconditioned system "pseudo fine"

First real coarse grid

Second real coarse grid

Deflation

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

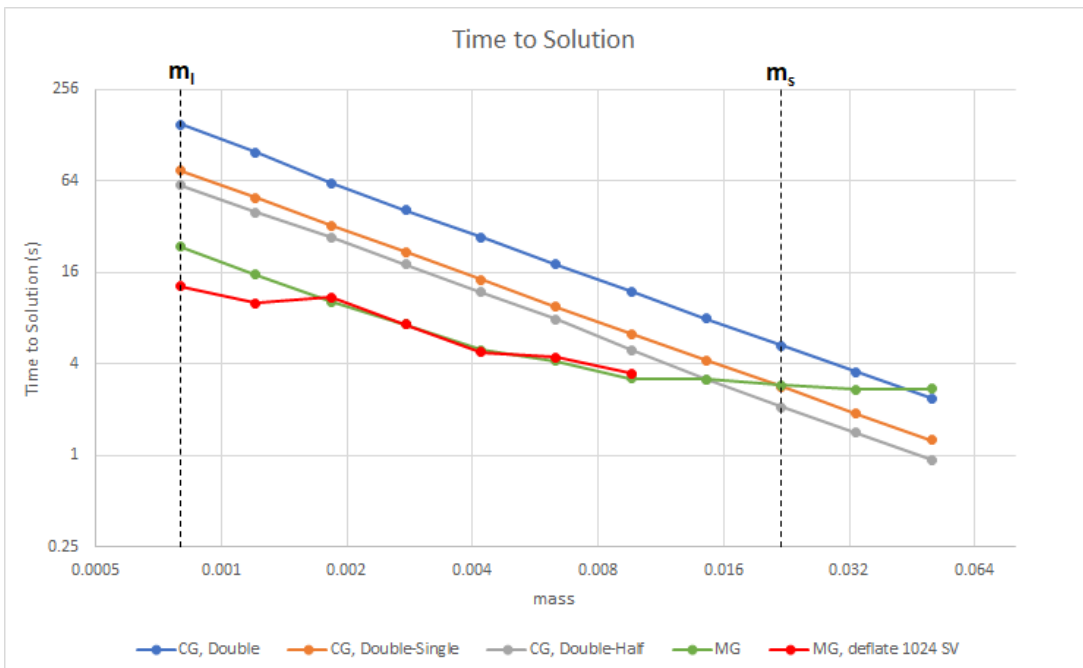
Smoother: CA-GCR(0,2)

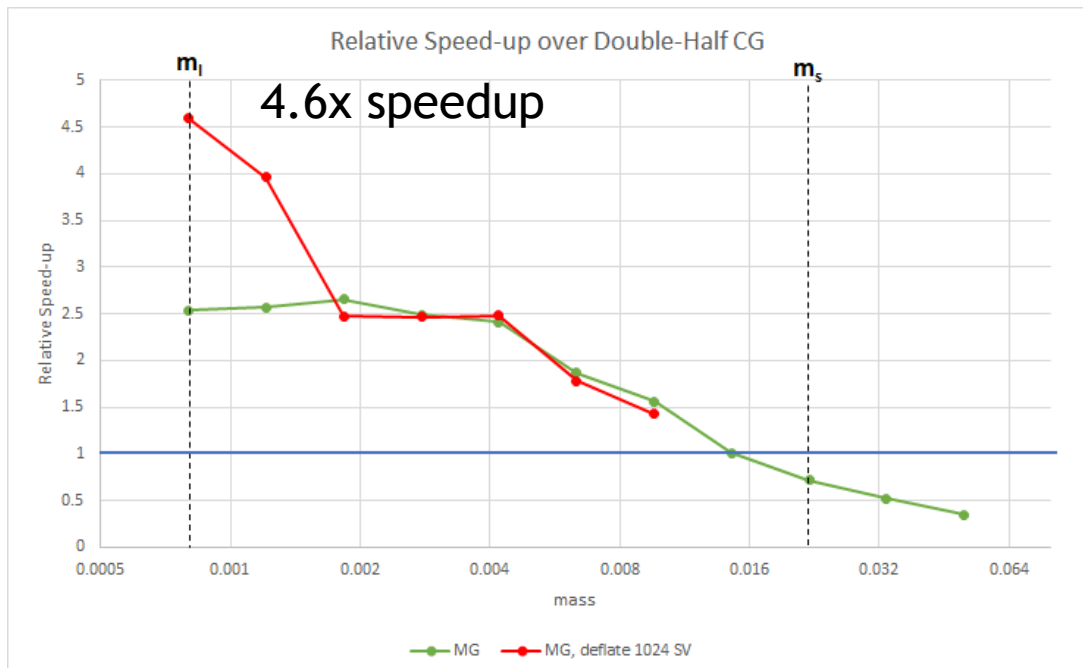
Level 4: **192 dof per site.**
Solver: CA-GCR(16)
Operator: Left-block Schur, 16-bit precision

Level 5: 1024 vector SVD Deflation

HISQ MULTIGRID

Rich Brower, KC, Dean Howarth, Alexei Strelchenko, **Evan Weinberg** (Tuesday 15:20)





Lattice provided by MILC

Time to solution is measured running QUDA on 72x Summit nodes (432x GPUs) for solving the HISQ operator against a random source on a 96^3x192 lattice, $\beta = 6.72$, $a = 0.06$, $m_l = 0.0008$, $m_s = 0.022$, solver tolerance 10^{-10}

FERMION SOLVERS

Combination of algorithm (multigrid) and machine (GPUs)

A single Volta 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

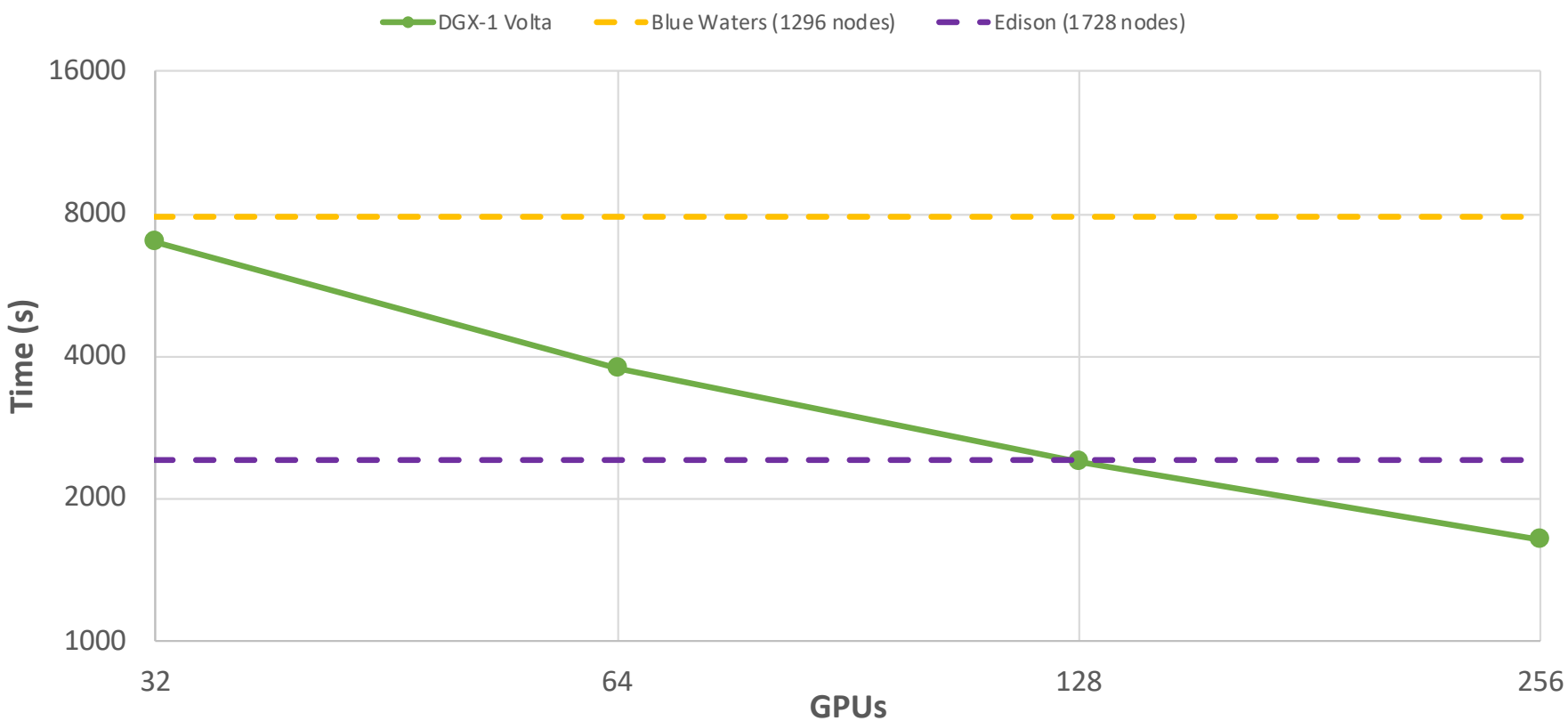
Fermion solvers are not the challenge they used to be (caveats unbound)

Next frontier for: combining multi-rhs with multigrid

STRONG SCALING

MILC APEX LARGE BENCHMARK

Strong scaling HISQ RHMC V = $72^3 x144$



Mathias Wagner
(Tuesday 15:40)

STRONG SCALING

Multiple meanings

- Same problem size, more nodes, more GPUs
- Same problem, next generation GPUs
- Multigrid - strong scaling within the same run
- We (the LQCD community) care about all of the above

To tame strong scaling we have to understand the limiters

- Bandwidth or latency

Look at scaling of Dslash on one DGX-1: half precision with 16^4 local volume

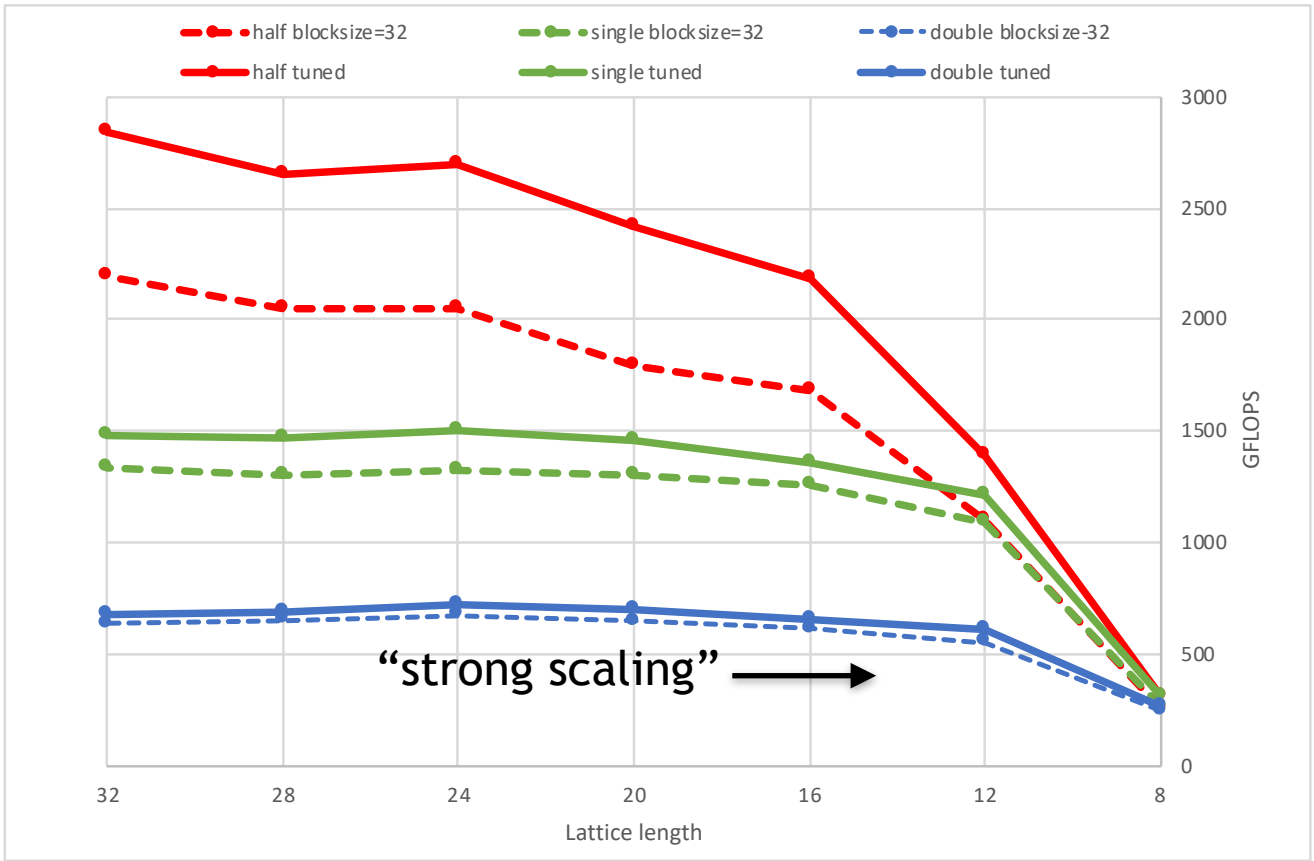
SINGLE GPU PERFORMANCE

Wilson Dslash

1312 GB/s

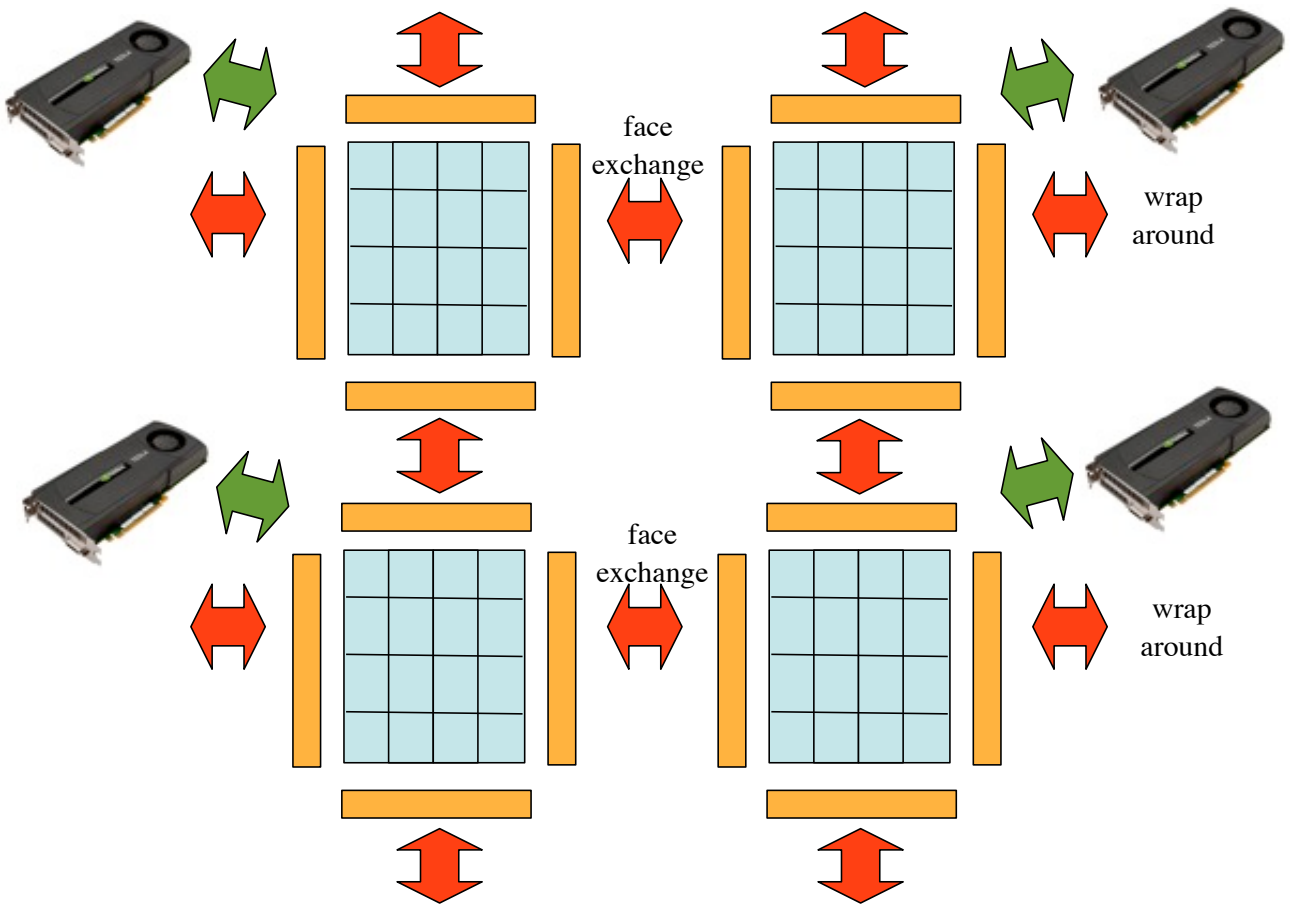
1291 GB/s

1180 GB/s



Tesla V100
CUDA 10.1
GCC 7.3

MULTI GPU BUILDING BLOCKS



Halo packing Kernel

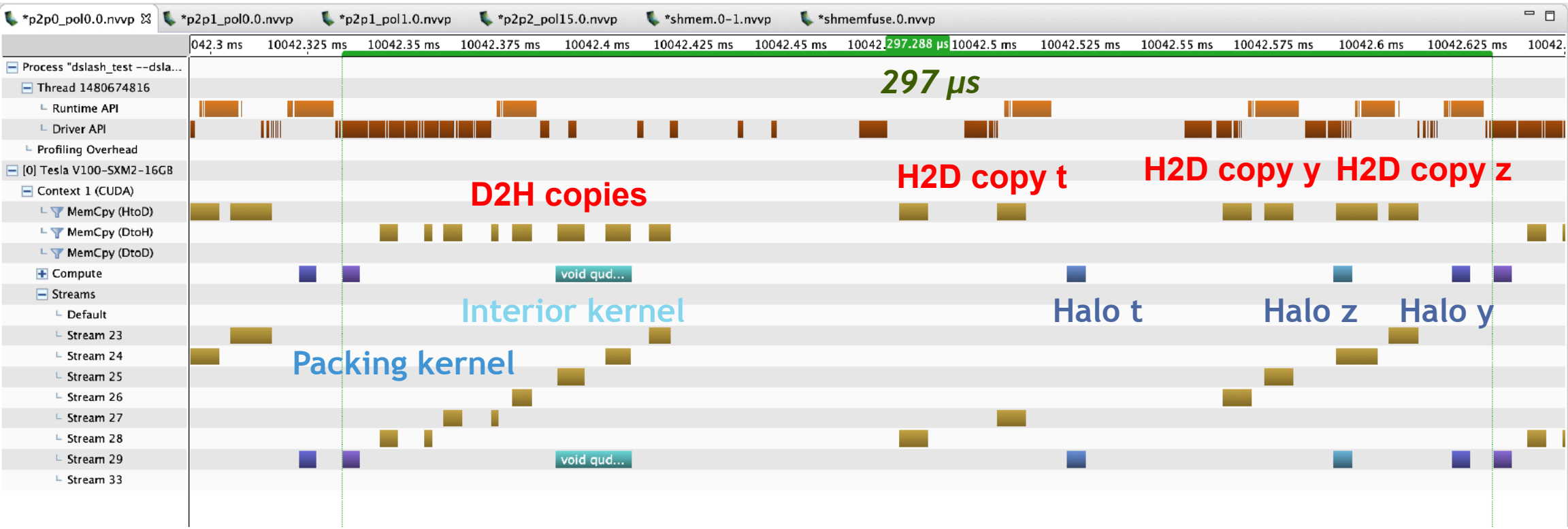
Interior Kernel

Halo communication

Halo update Kernel

WHAT IS LIMITING STRONG SCALING

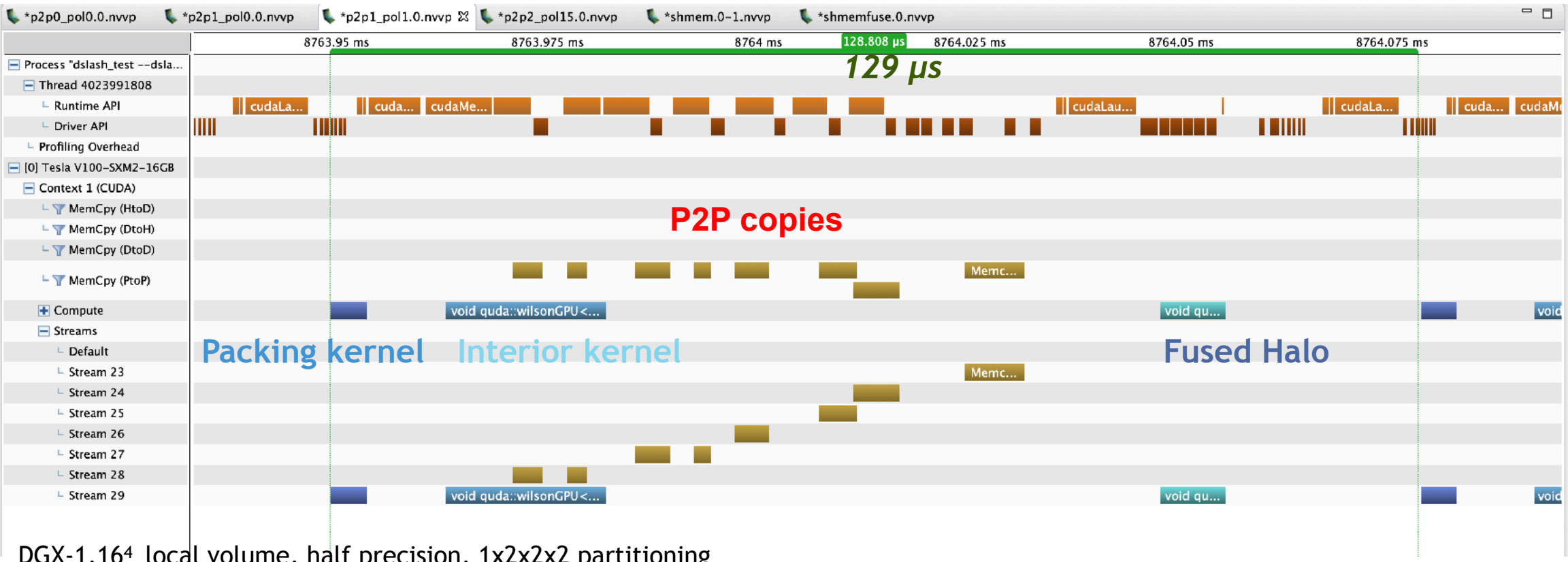
classical host staging



DGX-1,16^4 local volume, half precision, 1x2x2x2 partitioning

USING NVLINK AND FUSING KERNELS

fewer copies with higher bandwidth, fewer kernels, less API overhead



DGX-1,16^4 local volume, half precision, 1x2x2x2 partitioning

NVSHMEM

GPU-centric communication

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

NVSHMEM features

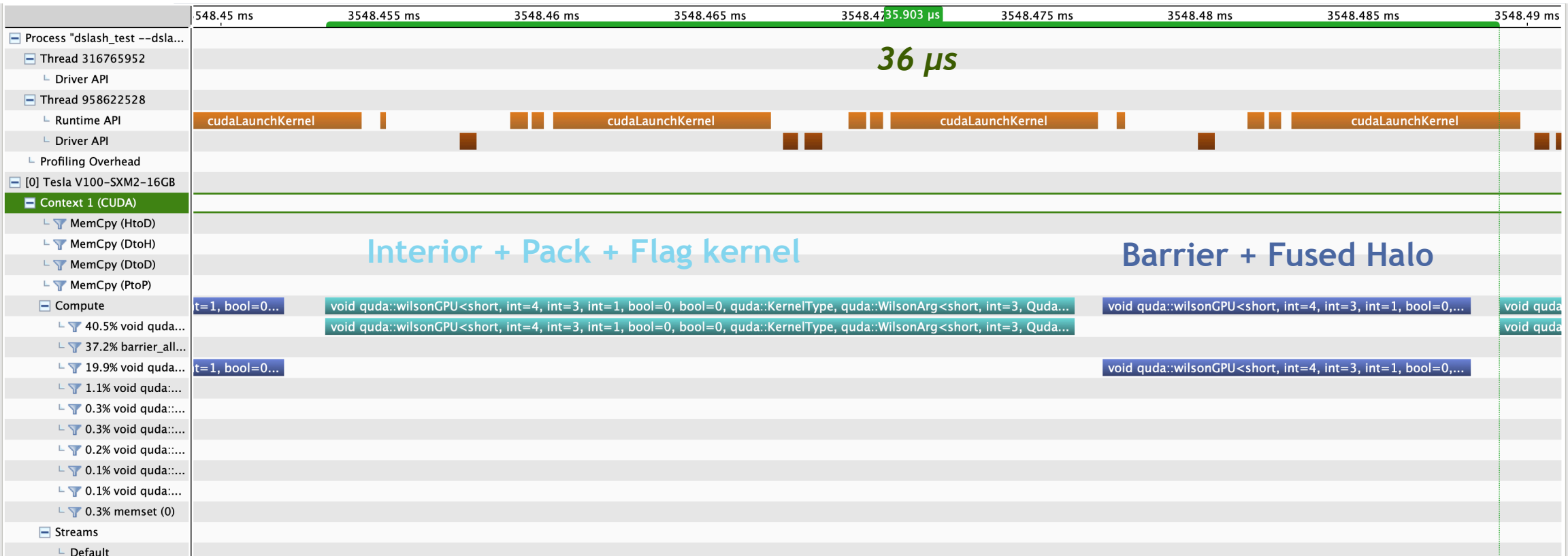
- Allows kernel-side communication (API and LD/ST) between GPUs
- NVLink and PCIe support (intranode), InfiniBand support (internode)
- X86 and Power9 support
- Interoperability with MPI and OpenSHMEM libraries

NVSHMEM has been developed as an NVIDIA internal co-design with QUDA

Early access (EA2) available – please reach out to nvshmem@nvidia.com

NVSHMEM + FUSING KERNELS

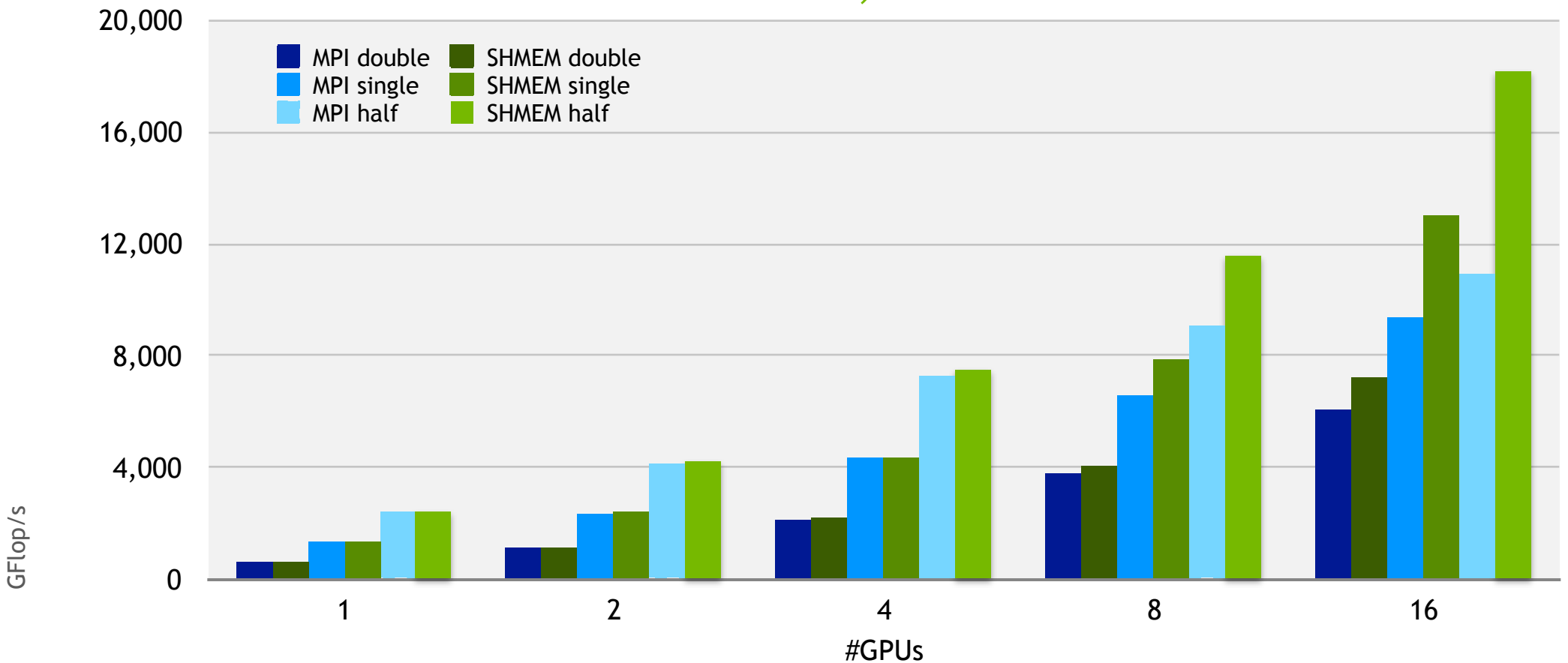
no extra packing and barrier kernels needed



DGX-1, 16^4 local volume, half precision, 1x2x2x2 partitioning

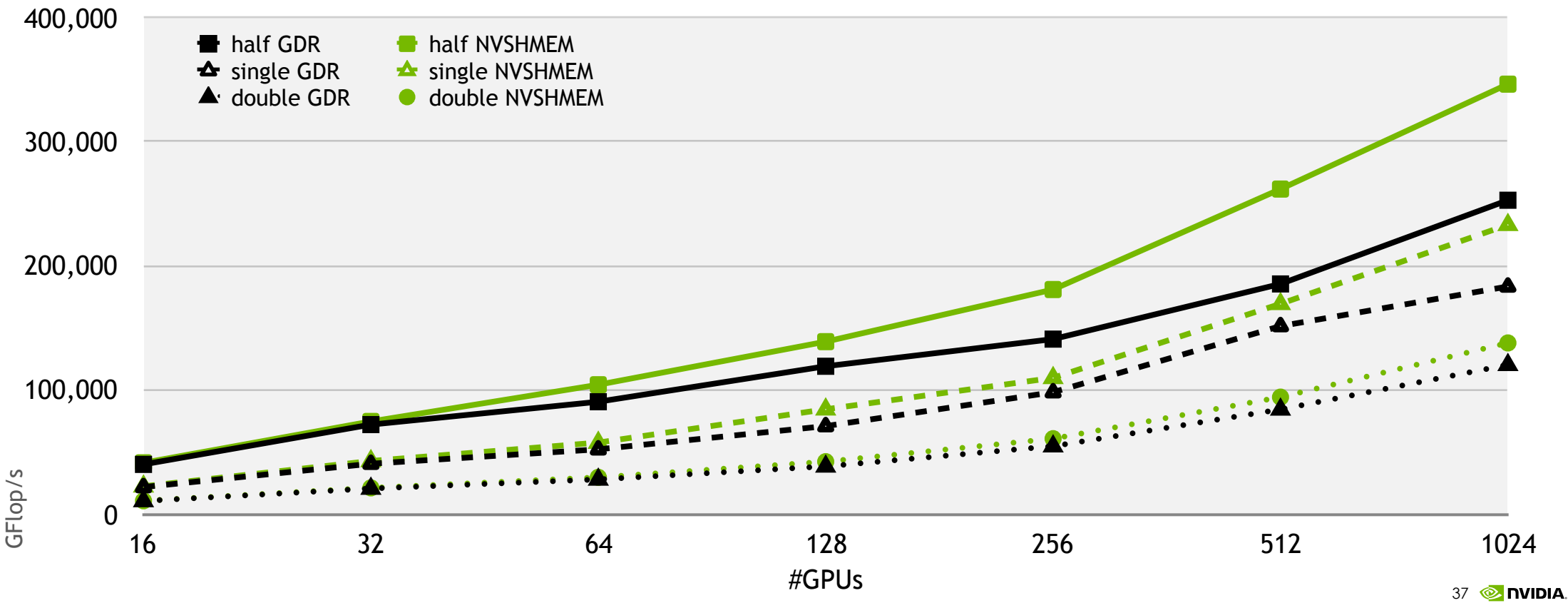
DGX-2 STRONG SCALING

Global Volume 32^4, Wilson-Dslash



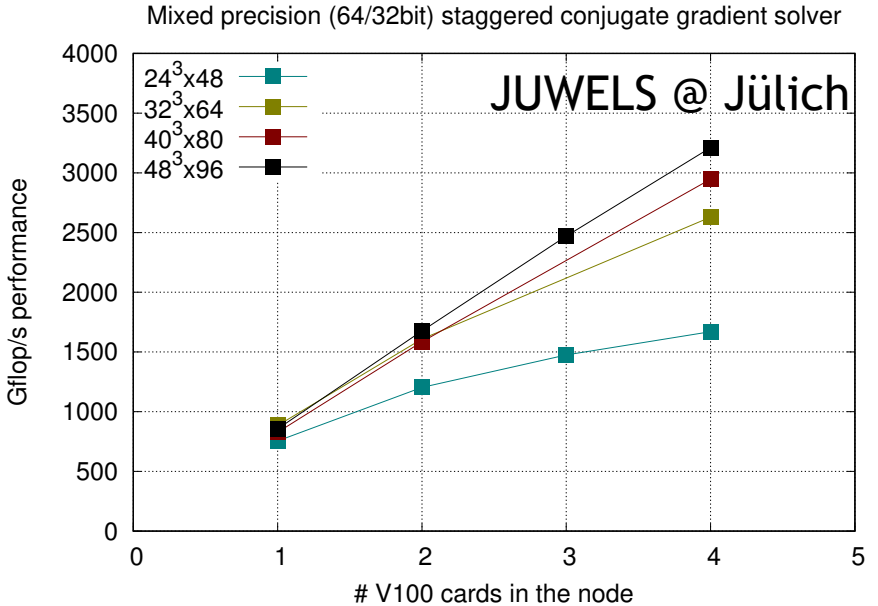
MULTI-NODE STRONG SCALING

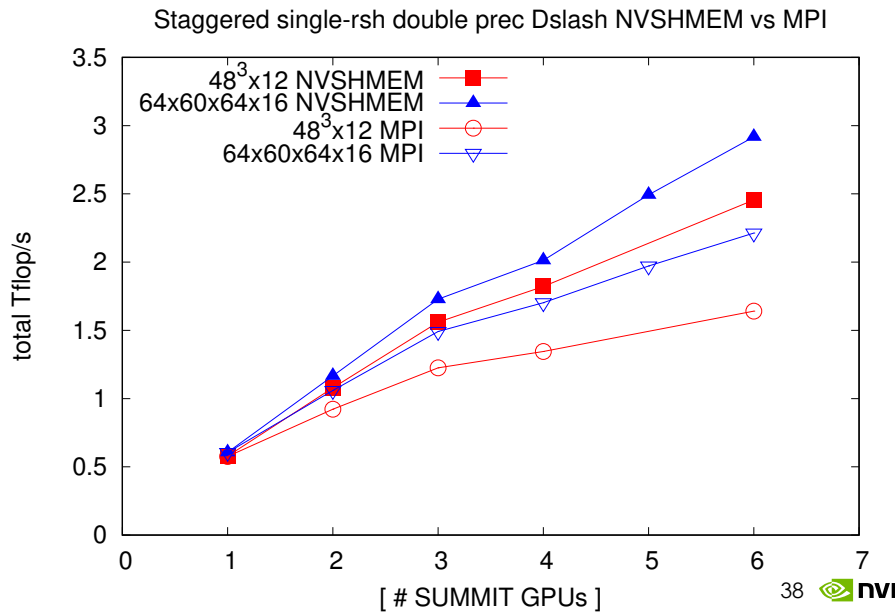
DGX SuperPOD, Wilson, 64^3x128 global volume



NVSHMEM IS NOT JUST FOR QUDA

BMW collaboration (Szabolcs Borzani)





THE COMMUNITY IS ACTIVE

Robert Mawhinney

(Thursday 14:00)

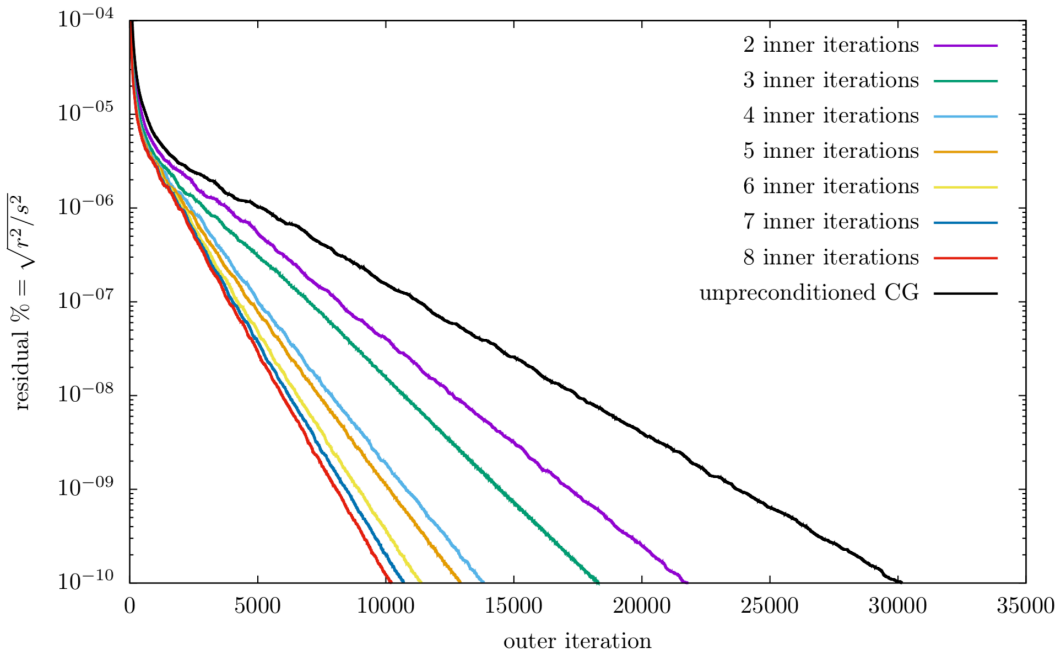
LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, **Jiqun Tu**

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

Multi-splitting Preconditioned CG

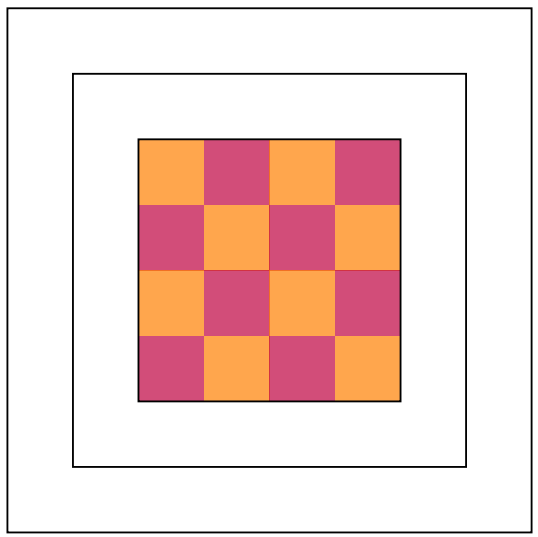
- Motivated by lack of network bandwidth in supercomputing centers (e.g., Summit)
- Block Jacobi preconditioner for (M)DWF that correctly applies the Dirichlet boundary condition for the e/o normal op
- Significantly reduces outer iterations
- Local preconditioner becomes prohibitively expensive



LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, **Jiqun Tu**

before 1st hopping term

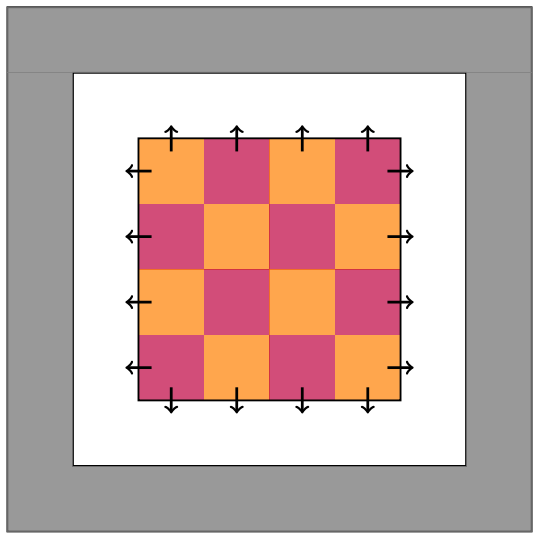


$$A = [1 - \kappa_b M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger}][1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi]$$

LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

before 1st hopping term

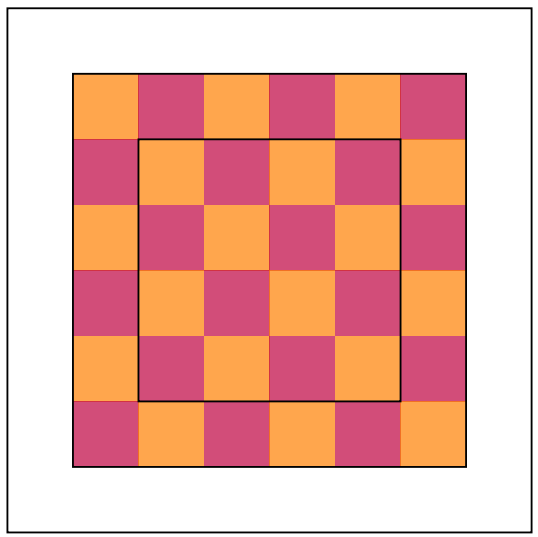


$$A = [1 - \kappa_b M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger}][1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi]$$

LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

after 1st hopping term

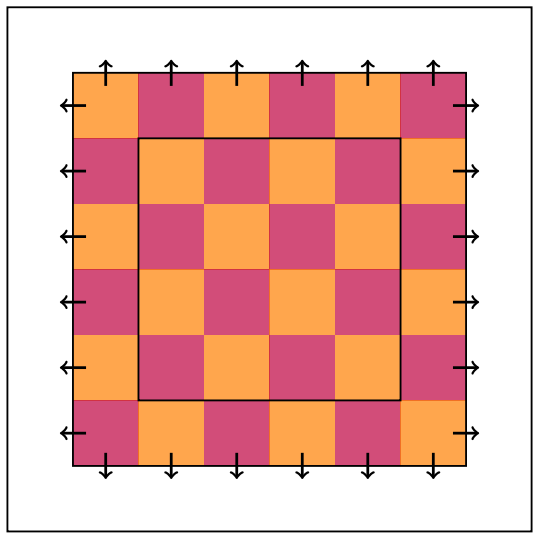


$$A = [1 - \kappa_b M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger}][1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi]$$

LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

before 2ed hopping term

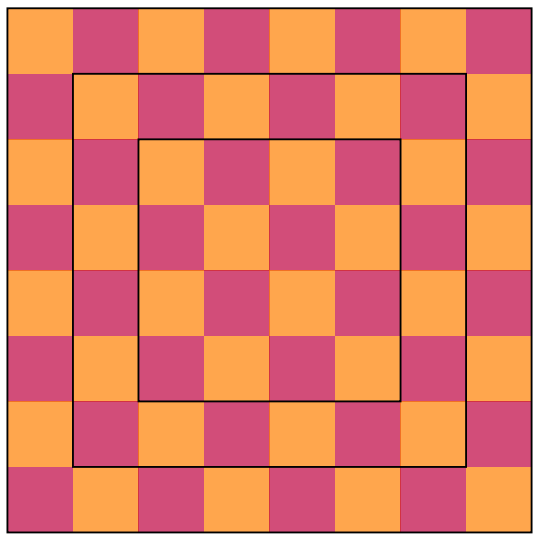


$$A = [1 - \kappa_b M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger}][1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi]$$

LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

after 2ed hopping term

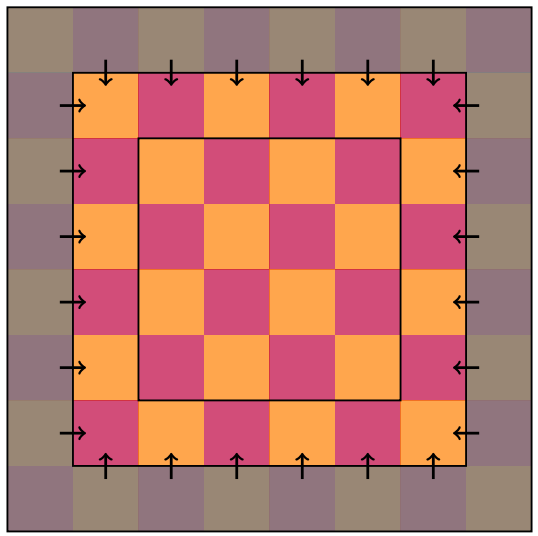


$$A = [1 - \kappa_b M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger}][1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi]$$

LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

before 3rd hopping term

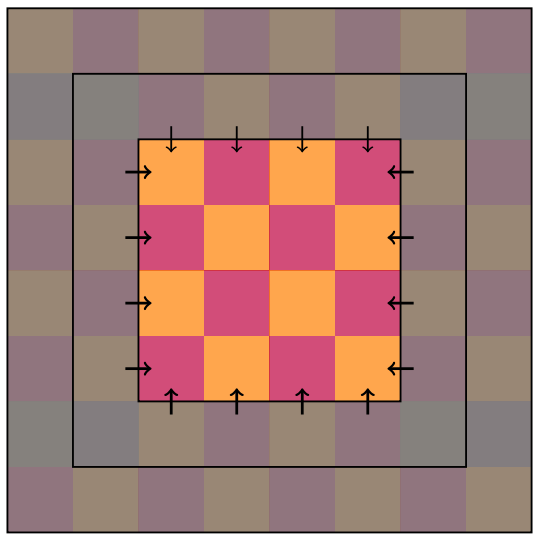


$$A = [1 - \kappa_b M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger}][1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi]$$

LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

before 4th hopping term



$$A = [1 - \kappa_b M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger}][1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi]$$

LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, **Jiqun Tu**

Modern GPU have high throughput tensor-core functionality

$$D = \underbrace{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,\dots} & A_{0,15} \\ A_{1,0} & A_{1,1} & A_{1,\dots} & A_{1,15} \\ A_{\dots,0} & A_{\dots,1} & A_{\dots,\dots} & A_{\dots,15} \\ A_{15,0} & A_{15,1} & A_{15,\dots} & A_{15,15} \end{pmatrix}}_{\text{FP16 or FP32}} \underbrace{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,\dots} & B_{0,15} \\ B_{1,0} & B_{1,1} & B_{1,\dots} & B_{1,15} \\ B_{\dots,0} & B_{\dots,1} & B_{\dots,\dots} & B_{\dots,15} \\ B_{15,0} & B_{15,1} & B_{15,\dots} & B_{15,15} \end{pmatrix}}_{\text{FP16}} + \underbrace{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,\dots} & C_{0,15} \\ C_{1,0} & C_{1,1} & C_{1,\dots} & C_{1,15} \\ C_{\dots,0} & C_{\dots,1} & C_{\dots,\dots} & C_{\dots,15} \\ C_{15,0} & C_{15,1} & C_{15,\dots} & C_{15,15} \end{pmatrix}}_{\text{FP16 or FP32}}$$

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

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

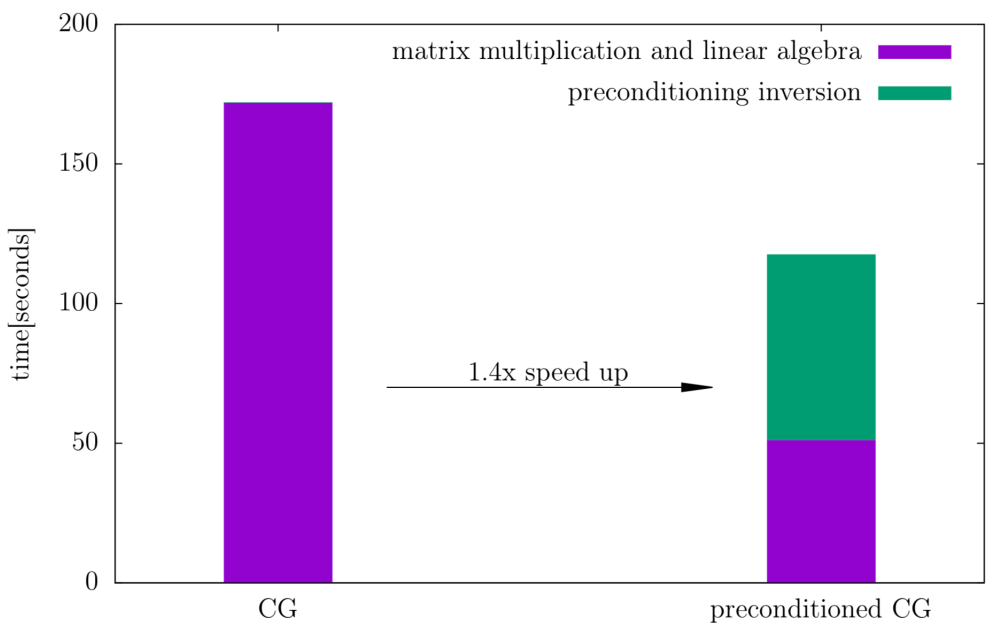
LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

Increasingly beneficial as one strong scales

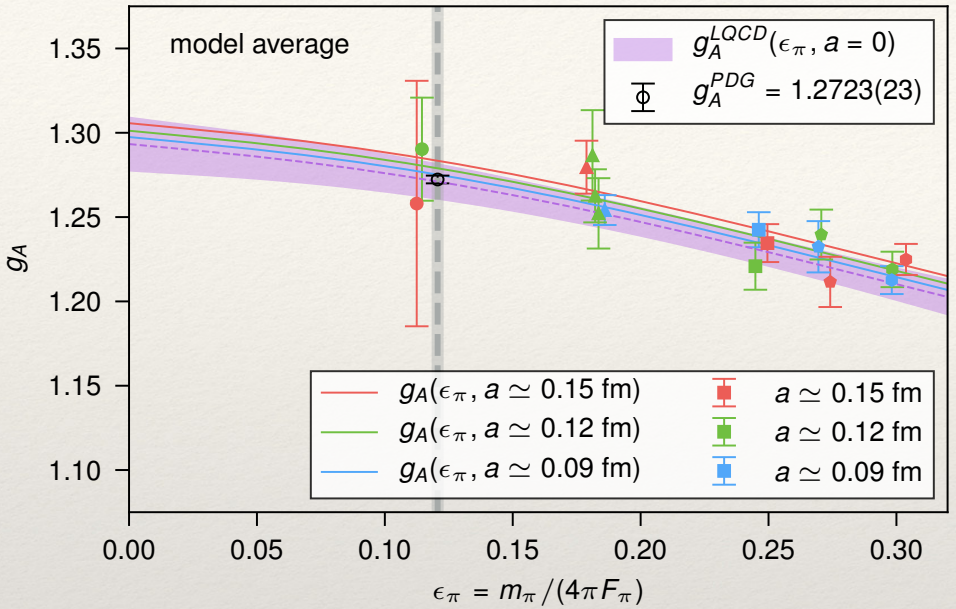
nodes	local volume	solver	inner iter.	(outer) iter.	r.u.	performance/node	time	speed up
256	$16 \cdot 24 \cdot 12 \cdot 24$	CG	–	42133	471	4.66	486.3	
		MSPCG	05	16903	195	1.56(01)/5.45(35)/37.29(53)	456.0	
		MSPCG	06	14860	173	1.56(01)/5.51(31)/37.60(58)	442.6	1.10x
		MSPCG	07	13787	161	1.56(01)/5.48(28)/37.49(60)	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)



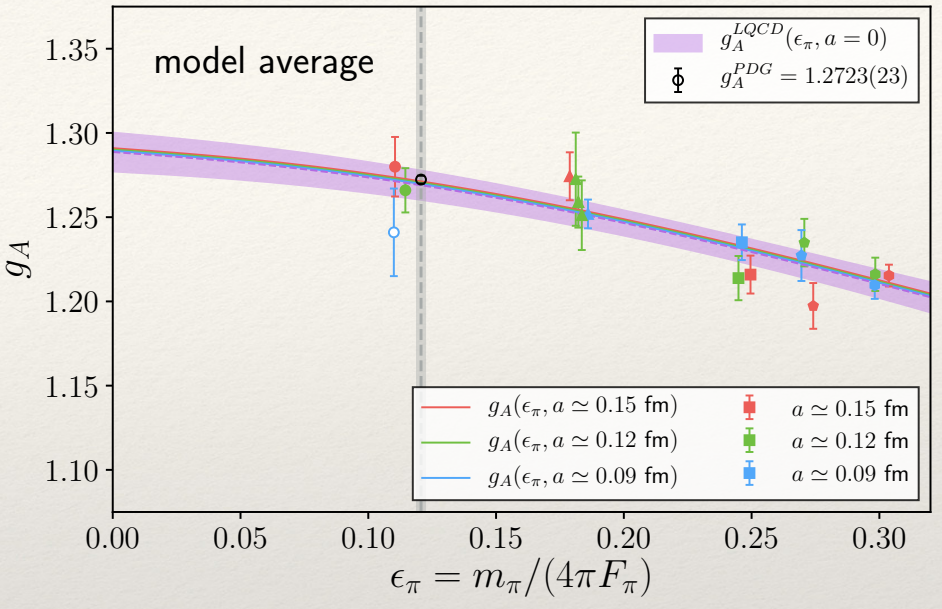
The Neutron Lifetime on Sierra Early Science

1 year on Titan (ORNL) + 2 years on GPU machines at LLNL



Nature 558 (2018) no. 7708, 91-94

PRELIMINARY



Sierra Early Science

- In 2.5 weekends on Sierra - we accomplished 5x more than in 1 year on Titan
- Machine-to-machine, compared to Titan @ ORNL (Titan is 18,688 nodes 16-core AMD + 1 K20X/node)
  For our research
  Sierra (~4300) is **~10 times** faster than Titan
  Summit (~4600) is **~15 times** faster than Titan
- These machines are **disruptively** faster than previous computers

Simulating the *weak* death of the neutron in a femtoscale universe with near-Exascale computing
Berkowitz, Clark, Gambhir, McElvain, Nicholson, Rinaldi, Vranas, Walker-Loud, Chang, Joó, Kurth, Orginos
https://www.olcf.ornl.gov/2018/09/17/uncharted-territory/

- Developers: Boyle, Cossu, Filaci, Portelli, Yamaguchi
- + Richtman, O'Haigan, Kettle, Erben, Marshall, Murphy, Jung, Lehner

Develop branch: excellent multicore CPU performance

- Contains functionality equivalent to QMP, QIO, QDP++, and much of Chroma
- SU(N), U(1), multiple fermion representations
- Gauge evolution: HMC, RHMC, EOFA
- Wilson, Clover, Domain wall, Mobius, + various 5D overlap
  - Improved staggered valence action
- Multiple solvers
  - Mixed precision Krylov solvers (CG, GMRES, BiCGstab …)
  - Block solvers
  - Polynomial preconditioned Lanczos & deflation
  - Experimental multigrid
- Split Grids for valence solves - avoid communication
  - NB: Not the same as trivial parallelism:
    - Eigenvectors held globally, deflate many RHS
    - Solves are local. 3x gain on Cori

GRID

Peter Boyle *et al*

First-class C++11 application with broad coverage for LQCD (and beyond) workflow

Embrace modern programming paradigms

GRID

Peter Boyle *et al*

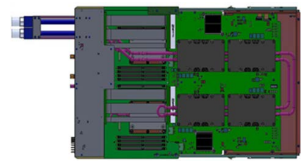
Single source high performance on both CPU and GPU

High-level C++ application code

feature/gpu-port

- **Aim to target multiple accelerators (Nvidia, AMD, Intel Aurora system)**
- **Presently compile all C++ files through NVCC (no .cu files)**
  - **Minimise vendor specific code to a few macros**
    - **Study HIP, OpenMP 5.0, SyCL options for other targets**
  - **Assume host and device have a unified memory space (UVM, SVM)**
- **(recent) Single source high performance kernels:**
  - **same kernel gives good performance on GPU *and* CPU**
- **All Fermion operators now accelerated**
  - **Wilson, Wilson5D, Improved Staggered**
  - **5D Mobius terms also**
- **Entire CG runs on GPU. Full (R)HMC next target.**
- **Intra node GPU-GPU communication uses NVlink ; multi node under test**

HPE ICE XA780i Quad Volta blade
24^4 5D single precision Wilson operator
1GPU 1.45 TF/s
4GPU 4.5 TF/s



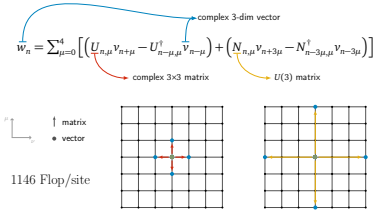
Performance of Staggered Fermion Kernels using Grid

Patrick Steinbrecher[1], Swagato Mukherjee[1]
[1] Brookhaven National Laboratory - USA

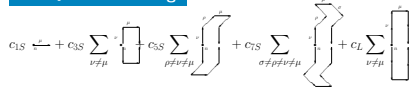
HISQ Dslash Operator

The key operation in many Lattice QCD simulations is the inversion of the fermion matrix. It requires a 4-dimensional stencil which calculates the product of a vector ν by a sparse matrix known as the Dslash operator and stems from a discretized 4-dimensional derivative.

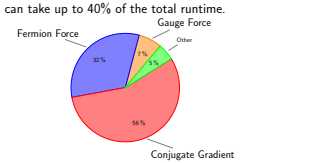
$$w_n = \sum_{\mu=0}^{4} \left[\left(U_{n,\mu} \nu_{n+\mu} - U^{\dagger}_{n-\mu,\mu} \nu_{n-\mu} \right) + \left(N_{n,\mu} \nu_{n+3\mu} - N^{\dagger}_{n-3\mu,\mu} \nu_{n-3\mu} \right) \right]$$



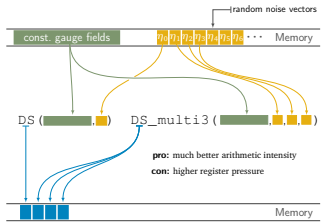
HISQ Link Smearing



The HISQ action suppresses unphysical interactions of quarks by smoothing each link with a weighted sum of neighboring links referred to as smearing. It is mainly used in the force calculation for the Hybrid Monte Carlo algorithm and can take up to 40% of the total runtime.

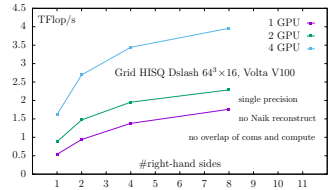


Multiple Right-hand Side Dslash



For many Lattice QCD applications, a large number of fermion matrix inversions are performed on a single gauge field. In order to exploit reuse of these gauge fields, we can apply the Dslash operation for multiple right-hand sides (rhs) at once. Increasing the number of rhs from one to four more than doubles the arithmetic intensity (Flop/byte) of the Dslash operation.

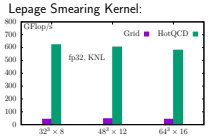
Multi-GPU Dslash Performance



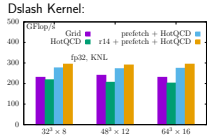
- Grid [1] available at: **github.com/paboyle/Grid**
  - architecture independent code
  - high and low level interface written in C++
  - support for multiple Lattice actions

Solvable Performance Problems

Lepage Smearing Kernel:

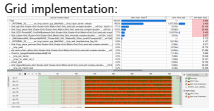


Dslash Kernel:



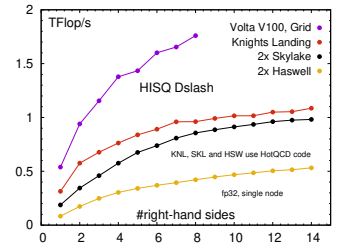
HotQCD implementation:

Grid implementation:



- Grid Dslash achieves optimal performance
- Grid Lepage smearing performance is limited by shortcomings of parallel transport approach
  - one parallel region for each link multiply in a staple
  - too many synchronization points
  - unnecessary loads and stores of intermediate results

Single-GPU Dslash Performance Comparison



#rhs	1	2	3	4	5	6
Flop/byte	0.73	1.16	1.45	1.65	1.80	1.91

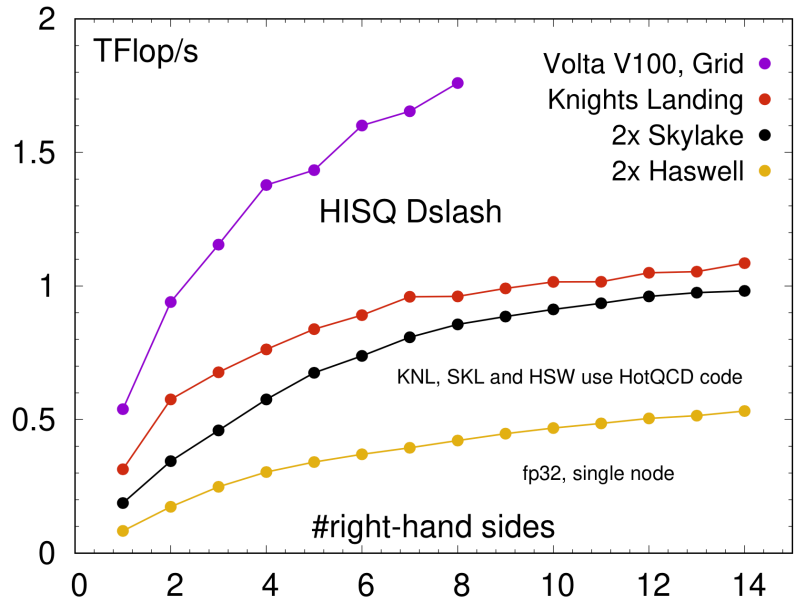
[1] P. Boyle et al., Grid: A next generation data parallel C++ QCD library. arXiv:1512.03487

SciDAC Scientific Discovery through Advanced Computing · NESAP · ECP Exascale Computing Project

GRID FOR HISQ

Patrick Steinbrecher (Poster)

Single-GPU Dslash Performance Comparison



#rhs	1	2	3	4	5	6
Flop/byte	0.73	1.16	1.45	1.65	1.80	1.91

Nim metaprogramming for performance, portability, and productivity

Xiao-Yong Jin and James C. Osborn
Computational Science Division, Argonne National Laboratory

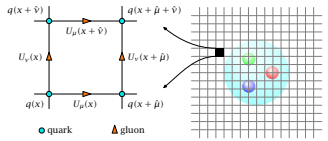
Abstract

The Nim programming language has an extensive metaprogramming capability. With the help of Nim's AST-based metaprogramming, we are building a framework where the application code is as high-level as in Python, and the low-level architecture specific code is compartmentalized. This allows us to target OpenMP, CUDA, and OpenCL for supported architectures. Here we describe our ongoing process of converting our CPU only lattice field theory framework QEX (Quantum EXpressions) [1] as an effort of the ECP application team, and compare the performance of key math kernels with multiple backends, including OpenMP with target device offloading, CUDA, and OpenCL. We show how a LISP-like macro system in Nim, combined with its feature rich type system, helps keep us mostly at a high level, while allowing us to dig in to the performance kernels and influence the C or C++ code generated by Nim.

[1] https://github.com/jcosborn/qex

Lattice QCD

A 4D lattice of space-time, where quarks (vectors) live on each site and gluons (matrices) live on each link.



- Large 4D (5D) grid of small vectors/matrices, homogeneous stencil operations—large sparse linear algebra.
- Most time consuming—solving Dirac equation.
- Most of the lattices too large to fit in a single node.
- Typically small volume to surface ratio per node.

Nim

- Modern language started in 2006, "efficient, expressive, and elegant".
- Feel of high level scripting language (Python) with extensive type inference.
- Statically typed systems language (full access to low-level objects & code).
- Choice of runtime with different types of GC or without GC at all.
- Generates C or C++ code, then compile with any compiler.
- Easy integration with C/C++: intrinsics (simd), pragmas (OpenMP) .
- LLVM-IR backend work in progress (passes 95% tests).
- Integrated build system (no Makefile necessary): copy main program, modify, compile.
- Extensive meta-programming support (nearly full language available at compile time).
- Transform any Nim code to new Nim code using Nim code.
- Website: `http://nim-lang.org`.

C++	Nim
Preprocessor macros	Templates: inline code substitutions
	allows overloading, choice of hygiene
Templates	Generics: including type classes and concepts
	applies to types, procedures, templates, and macros
N/A	Macros: LISP style, works on typed/untyped AST
	compile time evaluation and AST transformation

High level code example

```
proc test(V, M: static[int]; N: int) =
  var
    x = newColorMatrixArray(V,M,N)        Define complex
    y = newColorMatrixArray(V,M,N)        3×3 matrix field
    z = newColorMatrixArray(V,M,N)

  threads:    CPU threads
    x := 0
    y := 1    Set diagonal elements
    z := 2                                x^l_ij ← x^l_ij + y^l_ik × z^l_kj
    timeit "CPU": x += y * z              Benchmark on CPU

  timeit "GPU5":
    onGpu(N, 32):                         Run statement block on GPU
      x += y * z
  timeit "GPU6": onGpu(N, 64): x += y * z    Rerun with
  timeit "GPU7": onGpu(N, 128): x += y * z   different T/B

  threads: timeit "CPU": x += y * z       Back to CPU threads
```

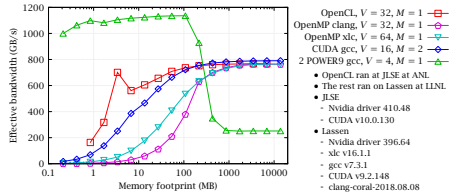
- A field holds pointers to memory regions on both CPU and GPU.
- `threads` runs the enclosed code block with OpenMP parallel.
- `onGpu` runs the enclosed code block by generating either a CUDA. kernel or an OpenMP target region, specified at compile time.
- GPU memory gets allocated only before the field is accessed the first time on GPU.
- Explicit memory copy is issued only when necessary.

OpenCL code generator

```
let s = toOpencl:
  proc matmul(A: global[ptr UncheckedArray[Cmat]],
              B: global[ptr UncheckedArray[Cmat]],
              C: global[ptr UncheckedArray[Cmat]]) {.oclkernel.} =
    let idx = get_global_id(0) div nv.toLit
    let vLane = get_global_id(0) mod nv.toLit
    for i in 0..<3:
      for j in 0..<3:
        var t: Cmplx
        t := A[idx][i][0] * B[idx][0][j]
        for k in 1..<3:
          t += A[idx][i][k] * B[idx][k][j]
        C[idx][i][j] := t
```

- `toOpencl` transforms the statically type checked Nim procedure definition to an OpenCL code string, which is then compiled at runtime.
- The vector lane, `vLane`, is detected and used when fields are accessed.

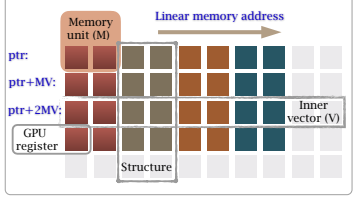
Performance on a V100 and two POWER9



OpenCL, $V = 32$, $M = 1$
OpenMP clang, $V = 32$, $M = 1$
OpenMP xlc, $V = 64$, $M = 1$
CUDA gcc, $V = 16$, $M = 2$
2 POWER9 gcc, $V = 4$, $M = 1$

- OpenCL ran at JLSE at ANL
- The rest ran on Lassen at LLNL
- JLSE
  - Nvidia driver 410.48
  - CUDA v10.0.130
- Lassen
  - Nvidia driver 396.64
  - xlc v16.1.1
  - gcc v7.3.1
  - CUDA v9.2.148
  - clang-coral-2018.08.08

Coalesced memory layout



Status & plans

- Optimize OpenMP implementation, improve abstraction and include new OpenMP 5 features.
- Improve OpenCL code generator, integrate the OpenCL backend to high level.
- Experiment with SYCL backend.
- Evaluate different backends across architectures and determine best match for current and future machines.

This research uses resources at the Livermore Computing Center at Lawrence Livermore National Laboratory, which is supported by the National Nuclear Security Administration of the US DOE under Contract DE-AC52-07-NA27344. This research uses resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory. This research is supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

NIM

Xiao-Yong Jin and James Osborn

Express LQCD using the Nim language

High-level expressions transformed into different target

- CUDA
- OpenCL
- OpenMP

In production by the LSD collaboration

First Study of Nf=2+1+1 Lattice QCD with Physical Domain-Wall Quarks
Ting-Wai Chiu (Thursday, Hadron Spectroscopy, 14:40)

- Quarks: optimal DWF with $N_s = 16$, $\lambda_{\max} / \lambda_{\min} = 6.20 / 0.05$.
  Gluons: plaquette gauge action at $\beta = 6/g^2 = 6.20$.
- Lattice size: $64^3 \times 64$, a ≈ 0.062 fm, $M_\pi L \approx 3$, $L \approx 4$ fm.
- For the one-flavor, use the Exact One-Flavor pseudofermion Action (EOFA).
  [Y.C. Chen & TWC, Phys. Lett. B738 (2014) 55; TWC, Phys. Lett. B744 (2015) 95]
- For the 2-flavor, use the two-flavor algorithm for DWF.
  [TWC, T.H. Hsieh, Y.Y. Mao, Phys. Lett. B702 (2012) 131]
- Thermalization: 1 unit of DGX-1 (≈ 8 months), TWC, arXiv: 1811.08095
  Production runs: 4 DGX-1 (≈ 2 months) $\rightarrow$ 25 DGX-1 (≈ 2 months).
  Generated ≈ 1250 trajectories $\rightarrow$ ≈ 250 configurations.
- Topological charge is measured, and χ_{top} is determined.
- Mass spectra of $\pi^{\pm}, K^{\pm}, D^{\pm}, \bar{c}s, \bar{c}c, \Omega, \Omega_c, \Omega_{cc}, \Omega_{ccc}$

OPTIMAL DWF

Ting-Wai Chiu

Thursday 14:40

Moderate dense-node GPU clusters well suited for HMC

NUMERICAL STOCHASTIC PERTURBATION

Kenichi Ishikawa (Wednesday 9:20)

High-order numerical stochastic perturbation theory of the twisted Eguchi-Kawai model (Batched GEMMs and FFTs)

4x Pascal P100 | Dual socket SkyLake Gold

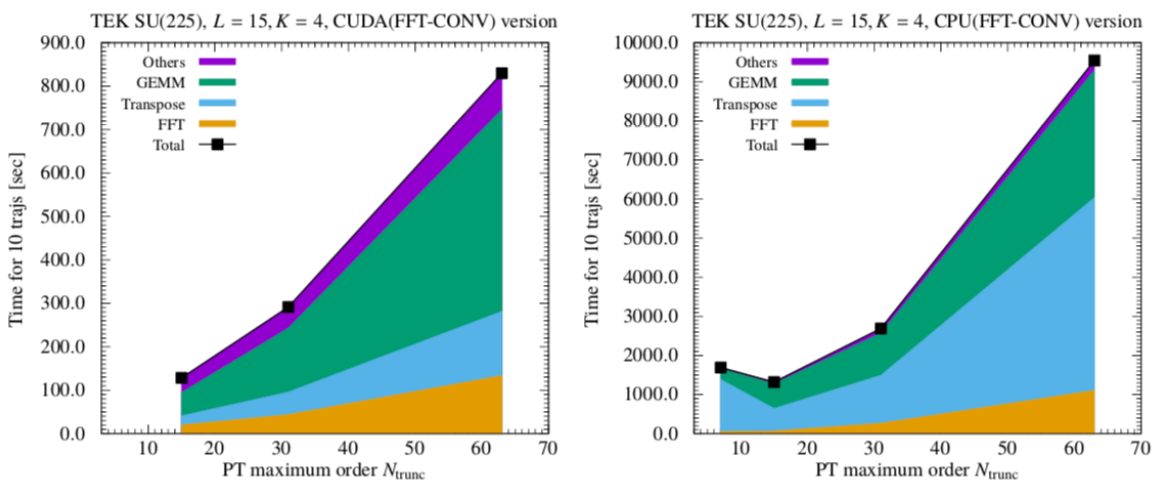


Fig. 2 Performance comparison from CPU and GPU versions ($N = 225$). Left: CUDA(FFT-CONV) version, Right : CPU(FFT-CONV) version.

MACHINE LEARNING QCD ON GPUS

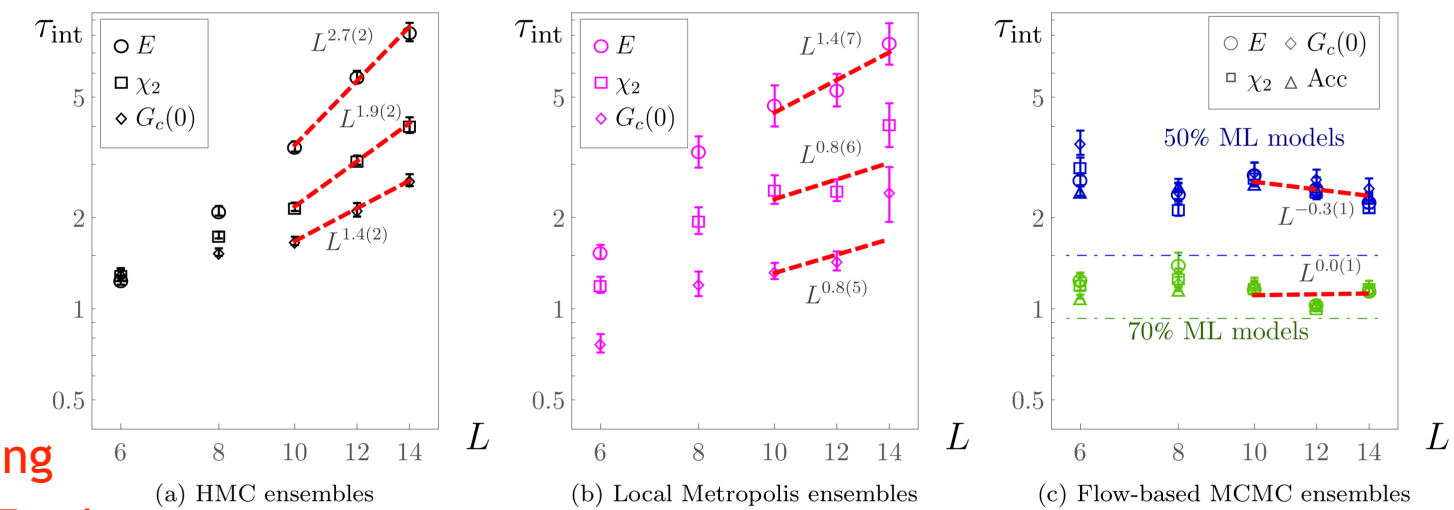
Albergo, Kanwar, Shanahan (Thursday, A&M 15:00)

Proof of concept study using machine-learned generative flow model
to propose configurations in a Markov chain (Metropolis-Hastings) (arXiv:1904.12072)

Removal of critical slowing down in φ^4 theory

Work in progress to extend to gauge theories

Trained the neural network using an 80x GPU cluster running PyTorch



(a) HMC ensembles (b) Local Metropolis ensembles (c) Flow-based MCMC ensembles

SOME THOUGHTS ON THE FUTURE

Slide courtesy of
Nick Wright @ NERSC

NEXT GENERATION

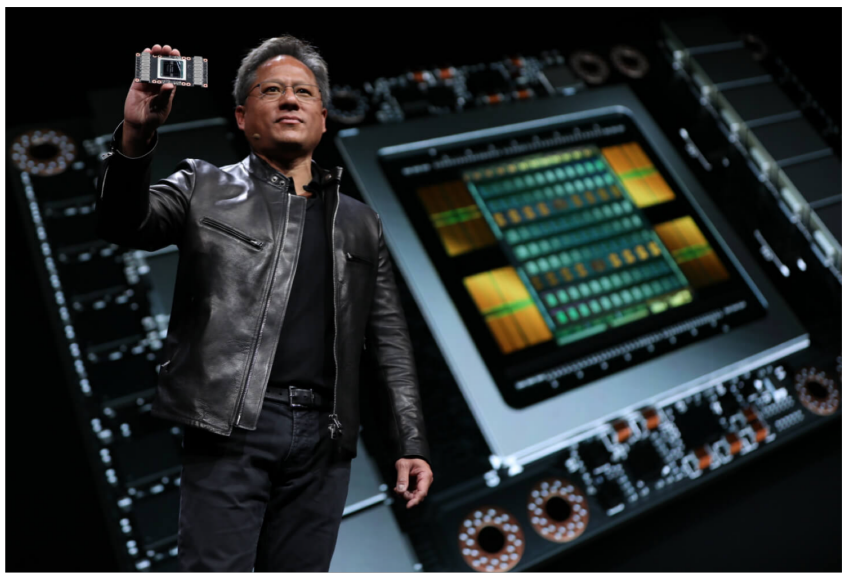
GPU nodes

4x NVIDIA “Volta-next” GPU
- > 7 TF
- > 32 GiB, HBM-2
- NVLINK

(Volta specs)

1x AMD CPU
4 Slingshot connections
- 4x25 GB/s

GPU direct, Unified Virtual Memory (UVM)
2-3x Cori

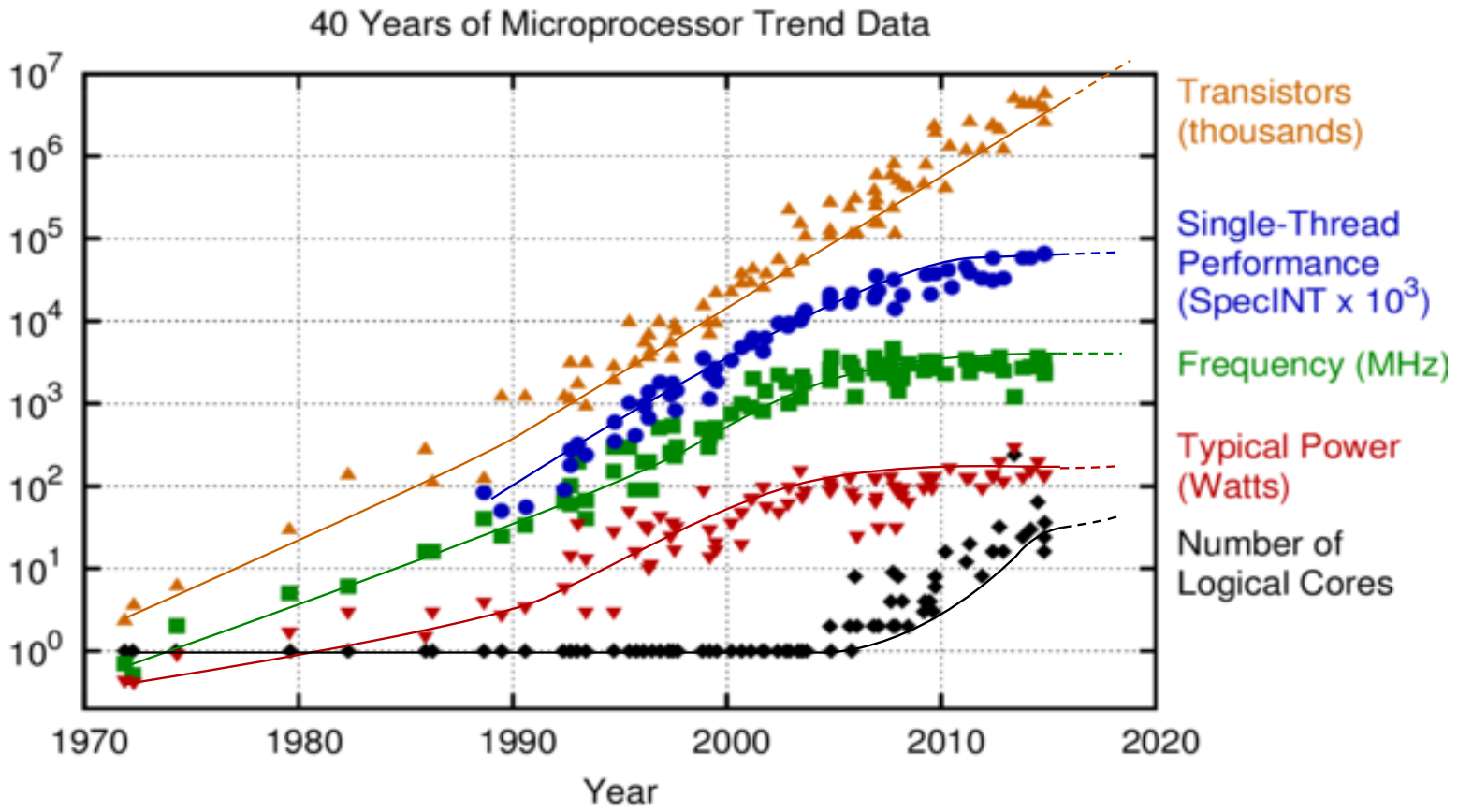


Perlmutter (2020)

More flops
* more parallelism

More memory
* bandwidth
* capacity

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

BEYOND THE EXASCALE

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

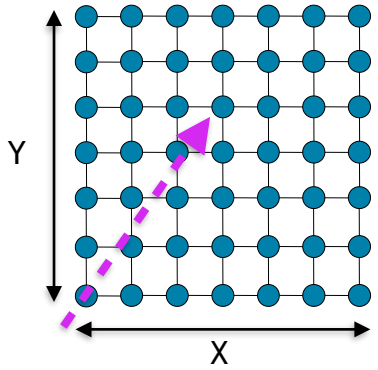
Even LQCD could be running out of parallelism

- Expose all sources parallelism

Pose everything as a matrix or batch problem where possible, e.g.,

- Block solvers
- Block contractions

SOURCES OF PARALLELISM



1. Grid parallelism

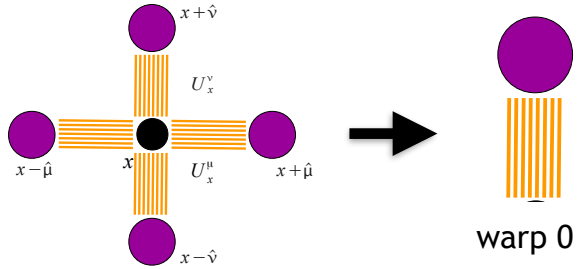
Volume of threads

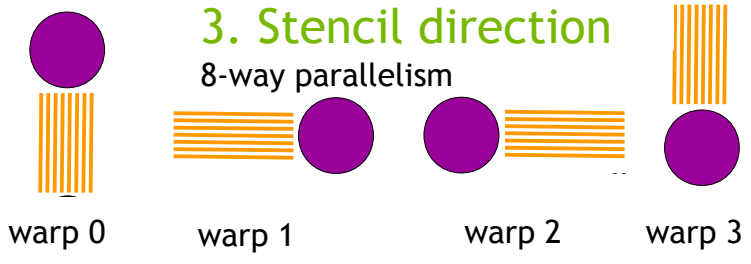
thread y index

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

2. Link matrix-vector partitioning

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





3. Stencil direction

8-way parallelism

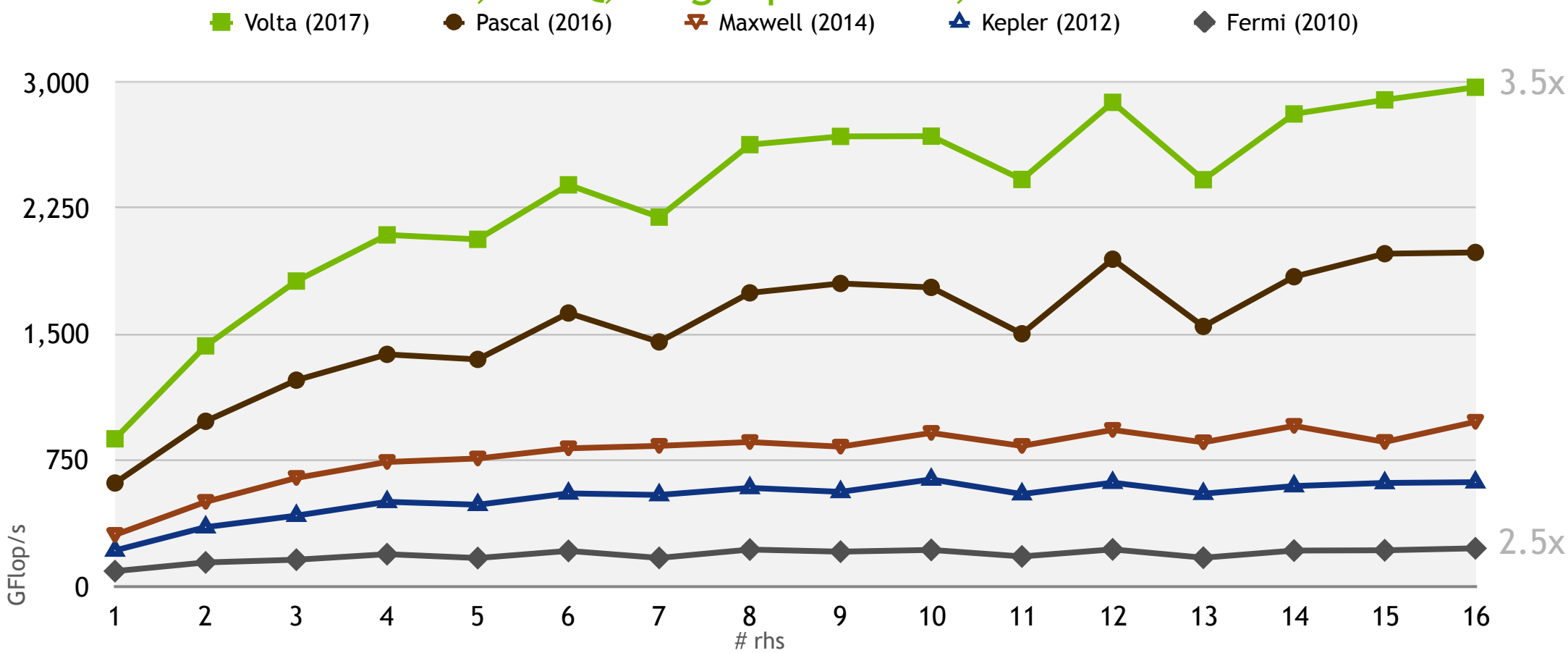
$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \Rightarrow \begin{pmatrix} a_{00} & a_{01} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} + \begin{pmatrix} a_{02} & a_{03} \end{pmatrix} \begin{pmatrix} b_2 \\ b_3 \end{pmatrix}$$

4. Dot-product partitioning

4-way parallelism

MULTIPLE RIGHT-HAND SIDES

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



REWORKING THE LAPH PIPELINE

KC, Ben Hoerz, Dean Howarth, Colin Morningstar, André Walker-Loud, *et al*

2 nucleon (2 baryon) and 2 hadron (ππ, Kπ) and meson-baryon scattering cross sections

	Classical approach	Parallelism / Intensity	Modern approach	Parallelism / Intensity
3-d Laplace eigenvectors	Lanczos	$T \times V_3$ AI ~ 1	Batched-Block-Lanczos	$B \times T \times V_3$ / AI ~ B
Clover-fermion solves	Sequential multigrid	V_4 AI ~ 1	Block multigrid	$N_\psi \times V_4$ / AI ~ N_{rhs}
Sink projections	Sequential inner products	$T \times V_3$ / AI ~ 1	Blocked inner productions => Matrix multiply	$N_\varphi \times N_\psi \times T \times V_3$ AI ~ $(N_\varphi \times N_\psi)/(N_l + N_\psi)$
Current Insertions	Sequential insertions (morally inner products)	$T \times V_3$ / AI ~ 1	Blocked insertions => Matrix multiply	$N_\psi^2 \times T \times V_3$ AI ~ $(N_\psi^2)/(2N_\psi)$

Goal is a single pipeline with no intermediate storage

SUMMARY

GPUs are the *de facto* platform for deploying Lattice Field Theory computations

Multigrid methods and GPUs are a potent combination

The community continues to be active in developing methods and software

Many ways to run on GPUs: QUDA, GRID, Nim, OpenACC/OpenMP, etc.

GPU-centric computing is the scalable future

Lattice Field Theory is a well-placed application to saturate post-Exascale computing


NVIDIA