TO THE EXASCALE AND BEYOND: COMPUTING CHALLENGES IN LATTICE QCD

Kate Clark @ Efficient Simulations on GPU hardware

OUTLINE

Introduction

GPUs for Lattice QCD

Scaling Challenges

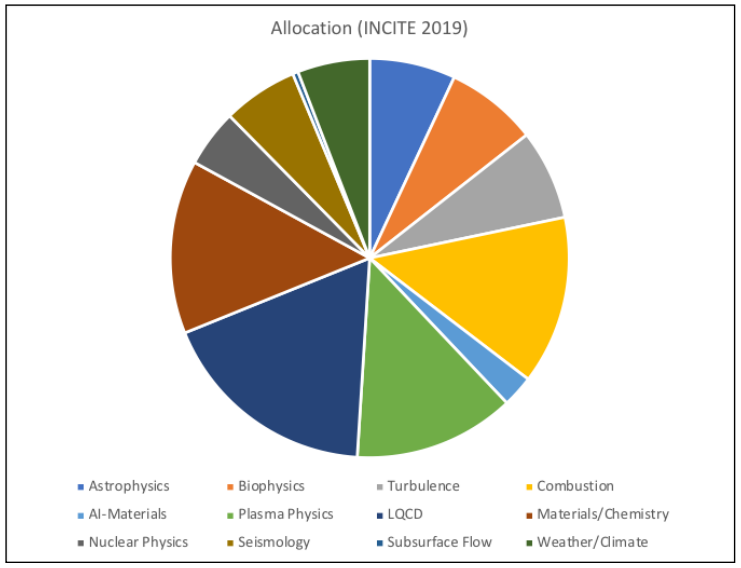
Parallelism and Locality

Future Challenges

INTRODUCTION

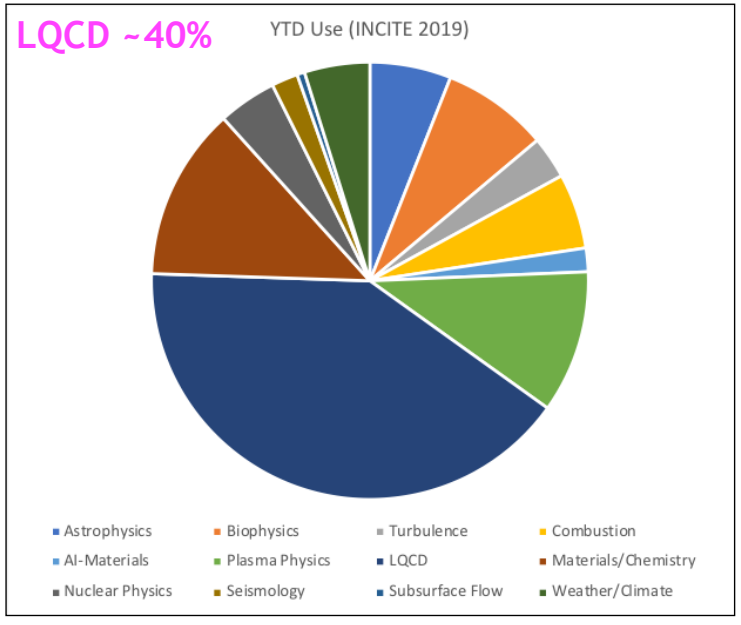
Jack Wells

LATTICE QCD IS HUNGRY



Summit cycle breakdown in INCITE allocation

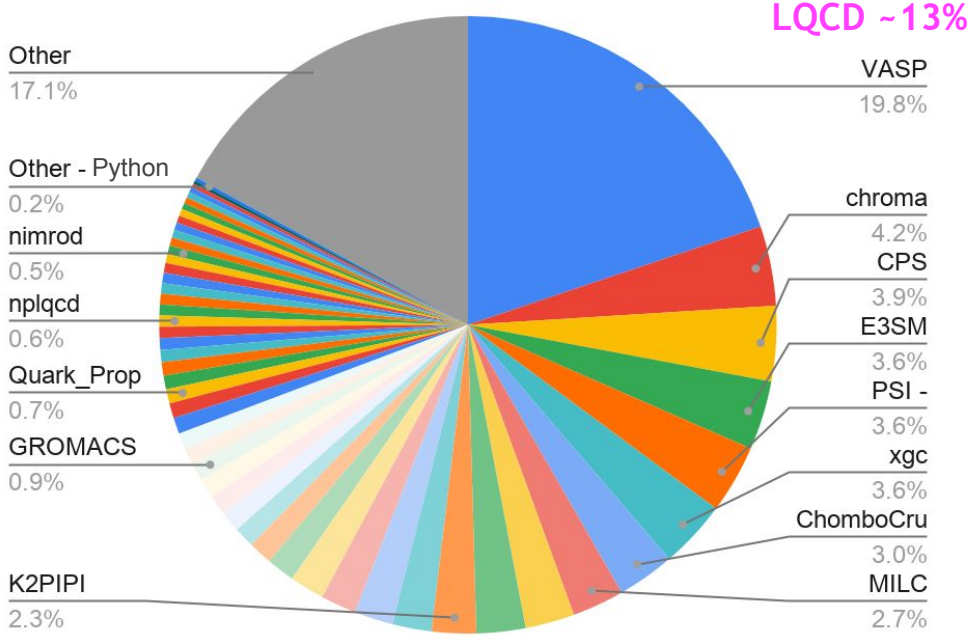
LQCD ~40%



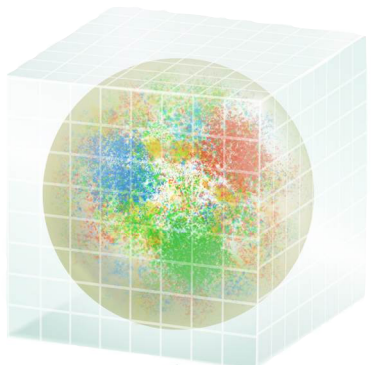
Summit cycle breakdown in INCITE use

NERSC Utilization (Aug ’17 - Jul’18)

LQCD ~13%



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 ⇒ **Linear solvers Ax = b**

Consumer of 10+% of public supercomputer cycles

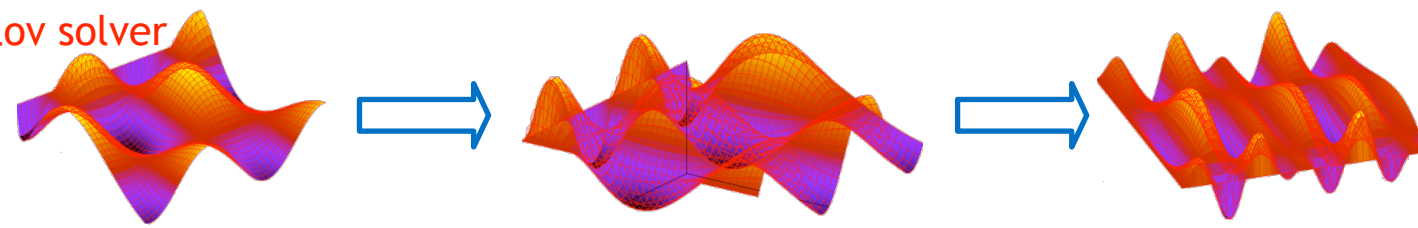
- Traditionally highly optimized on every HPC platform for the past 30 years
- Jobs often run at the 1000+ GPU scale

STEPS IN AN LQCD CALCULATION

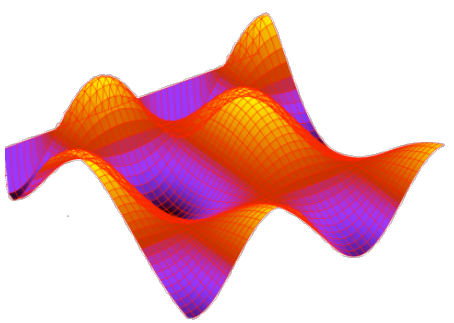
$$D_{ij}^{\alpha\beta}(x,y;U)\psi_j^{\beta}(y) = \eta_i^{\alpha}(x)$$

or Ax = b

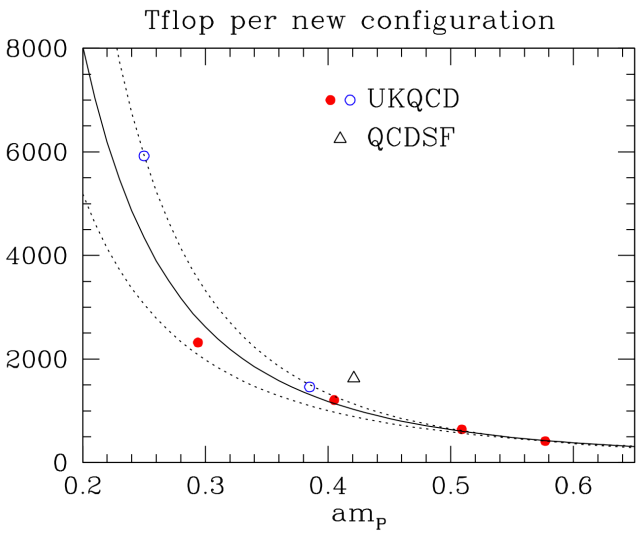
1. Generate an ensemble of gluon field configurations “gauge generation”
   - Hybrid Monte Carlo is the algorithm of choice
   - Produced in sequence, with hundreds needed per ensemble
   - Strong scaling required with 100-1000 TFLOPS sustained for several months
   - 70-90% of the runtime is in the Krylov solver
   - O(1) solve per linear system



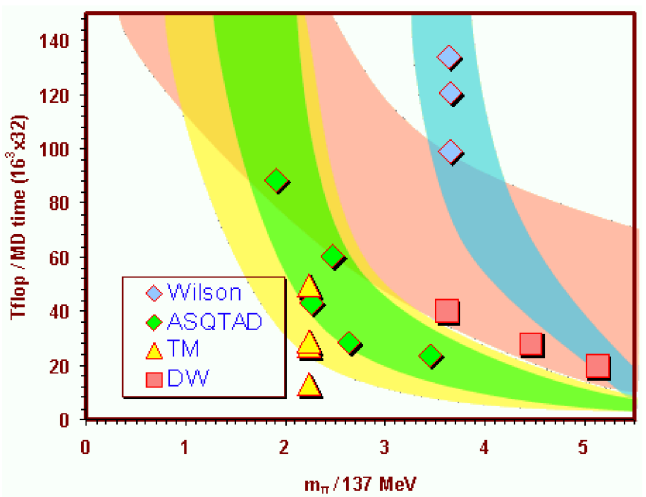
2. “Analyze” the configurations
   - Task parallelism means that clusters reign supreme here
   - 80-99% of the runtime is in the Krylov solver
   - Many solves per system, e.g., O(10^6)



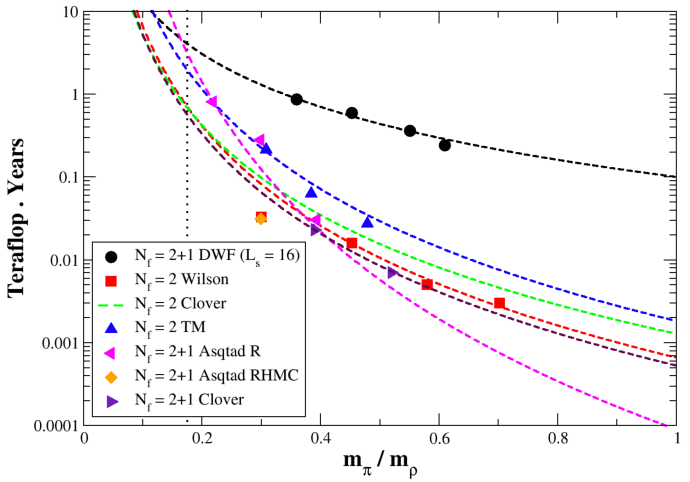
SCALING THE BERLIN WALL



Wittig *et al* 2001



Kennedy 2004



Clark 2006

Simulation Cost ~ $V^{\alpha} a^{\beta} m^{\gamma}$

$\alpha = 1.25$

$\beta \in -[3,6]$

$\gamma \sim -3$

(Early 2000s possible values)

SCALING THE BERLIN WALL

Metropolis Volume dependence V^{α}

- Scaling arises from holding stepwise errors with second-order Symplectic integrator
- Suppressed through use of fourth-order integrator $\alpha \rightarrow 1.125$

Kennedy, Silva and Clark, 2012

Linear solver critical critical slowing down

- Condition number diverges as we approach physical point
- (Adaptive) Multigrid removes the condition number and volume dependence

Lüscher 2007
Brannick *et al* 2007
Babbich *et al* 2010
Frommer *et al* 2013

Fermion force instability

- Instability in the MD integration due to low fermion modes requiring $\delta t \rightarrow 0$ as $m \rightarrow 0$
- Hasenbusch mass preconditioning / multiple pseudo-fermions dealt with step size instabilities

Hasenbusch 2001,
Urbach *et al* 2005,
Clark and Kennedy 2006

Autocorrelation length diverges as $a \rightarrow 0$

- Topology freezing…

Citations are illustrative,
not exhaustive


GPUS FOR LQCD

WHAT IS A GPU?

GPUs are extreme hierarchical processors

Many-core processor programmed using a massively threaded model

Threads arranged as Cartesian hierarchy of grids

Deep memory hierarchy

Registers <-> L1 <-> L2 <-> Device Memory <-> Host Memory

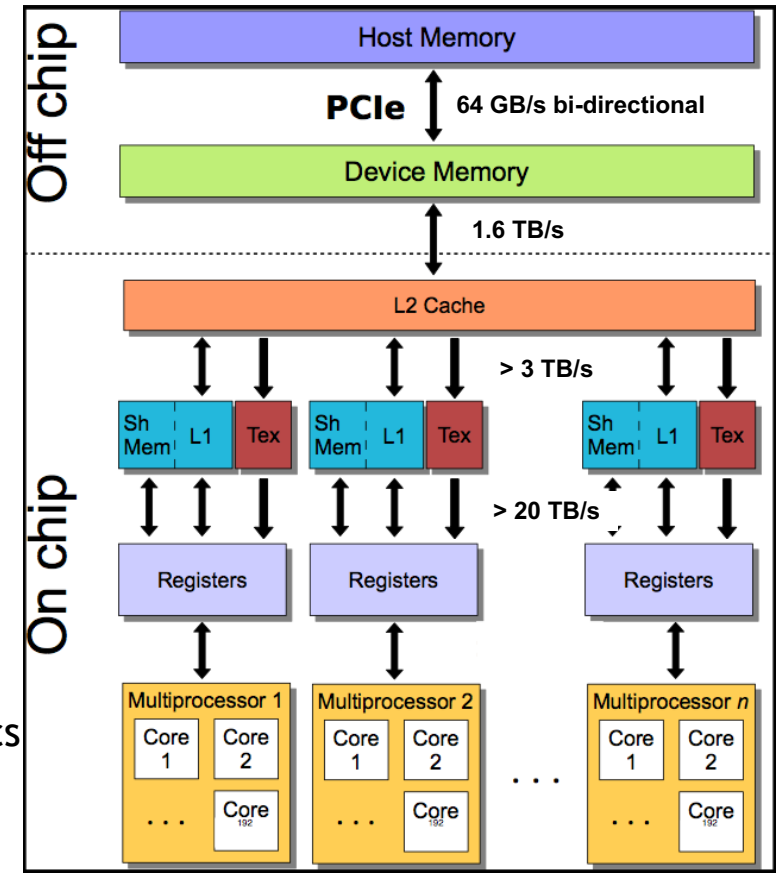
Increasingly coupled instruction hierarchy

Tensor cores <-> CUDA cores <-> shared mem atomics <-> L2 atomics

Synchronization possible at many levels

(Sub-)Warp <-> Thread Block <-> Grid <-> Node <-> Cluster

A100 sketch



ANNOUNCING H100

Unprecedented Performance, Scalability, and Security for Every Data Center

HIGHEST AI AND HPC PERFORMANCE

4PF FP8 (6X)| 2PF FP16 (3X)| 1PF TF32 (3X)| 60TF FP64 (3X)
3TB/s (1.5X), 80GB HBM3 memory

TRANSFORMER MODEL OPTIMIZATIONS

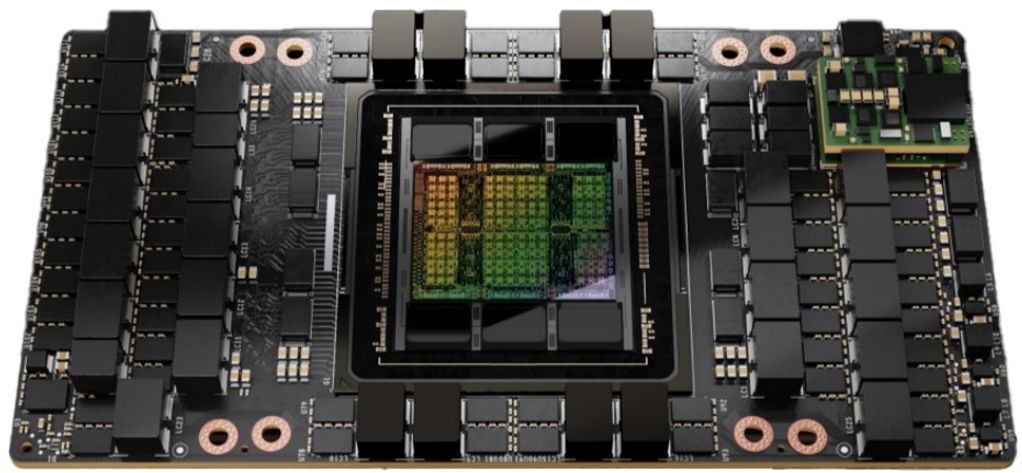
6X faster on largest transformer models

HIGHEST UTILIZATION EFFICIENCY AND SECURITY

7 Fully isolated & secured instances, guaranteed QoS
2nd Gen MIG | Confidential Computing

FASTEST, SCALABLE INTERCONNECT

900 GB/s GPU-2-GPU connectivity (1.5X)
up to 256 GPUs with NVLink Switch | 128GB/s PCIe Gen5



Custom 4N TSMC Process | 80 billion transistors

**ECP benchmarks apps

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 solvers for all major fermionic discretizations, with multi-GPU support
- Maximize performance
  - Mixed-precision methods
  - Autotuning for high performance on all CUDA-capable architectures
  - Multigrid solvers for optimal convergence
  - NVSHMEM for improving strong scaling
- Portable: HIP (merged), SYCL (in review) and OpenMP (in development)
- **A research tool for how to reach the exascale (and beyond)**
  - Optimally mapping the problem to hierarchical processors and node topologies

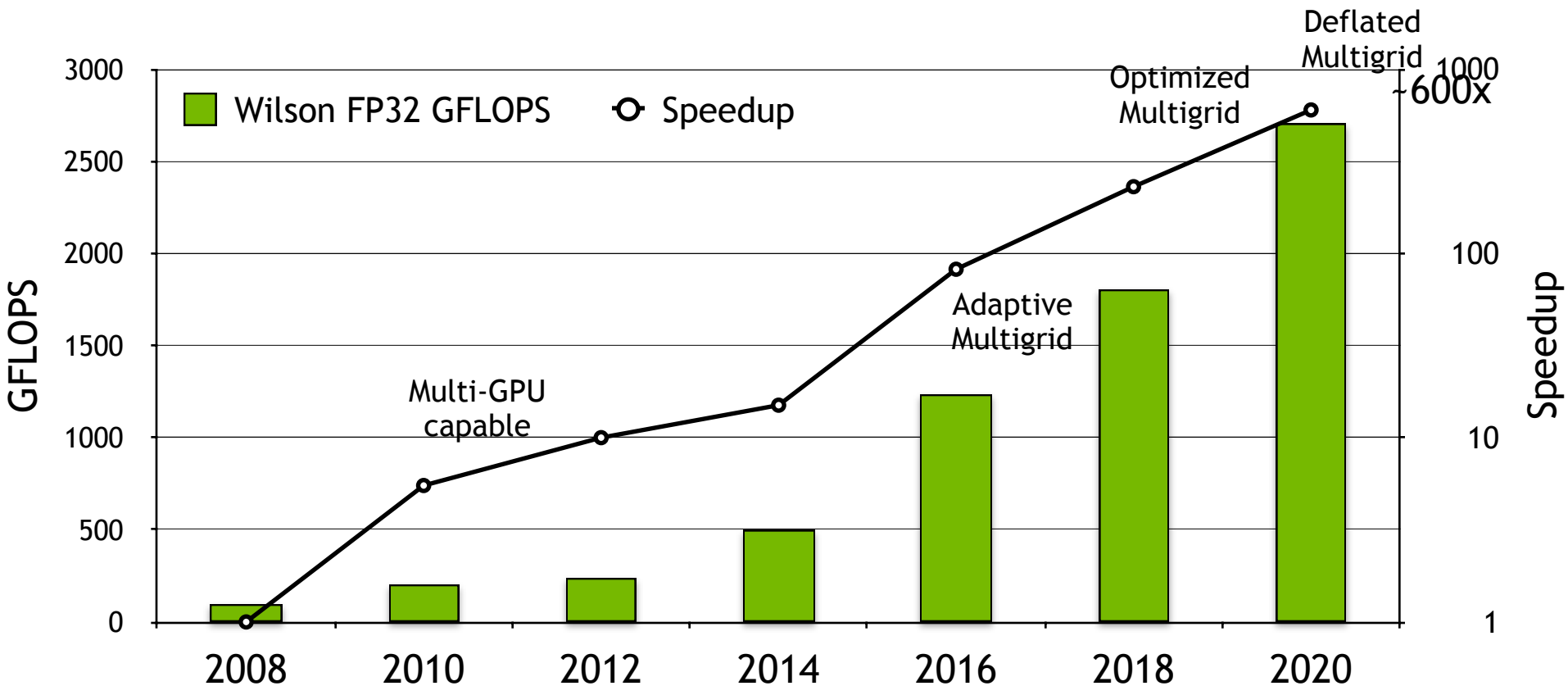
QUDA CONTRIBUTORS

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 (LLNL)
Xiao-Yong Jin (ANL)
Bálint Joó (Jlab)
Hyung-Jin Kim (BNL -> Samsung)
Bartek Kostrzewa (Bonn)
James Osborn (ANL)
Claudio Rebbi (Boston University)
Eloy Romero (William and Mary)
Hauke Sandmeyer (Bielefeld)
Guochun Shi (NCSA -> Google)
Mario Schröck (INFN)
Alexei Strelchenko (FNAL)
Jiqun Tu (NVIDIA)
Alejandro Vaquero (Utah University)
Mathias Wagner (NVIDIA)
André Walker-Loud (LBL)
Evan Weinberg (NVIDIA)
Frank Winter (Jlab)
Yi-bo Yang (CAS)

QUDA NODE PERFORMANCE OVER TIME

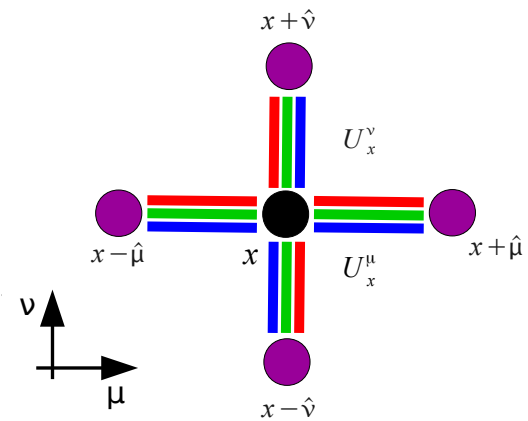
Multiplicative speedup through software and hardware



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.

MAPPING THE DIRAC OPERATOR TO GPUS

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



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 => $3.3\text{x}10^6$ threads

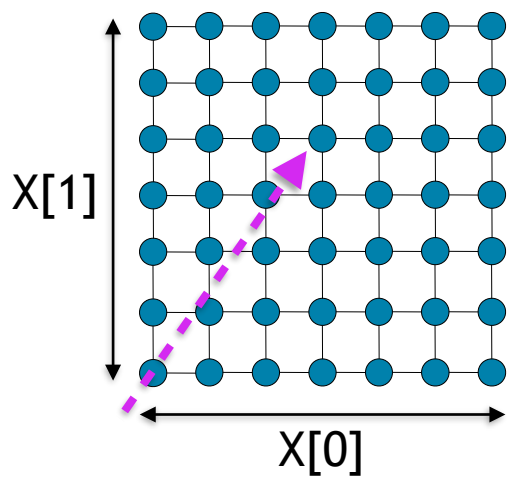
Looping over direction each thread must

- 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

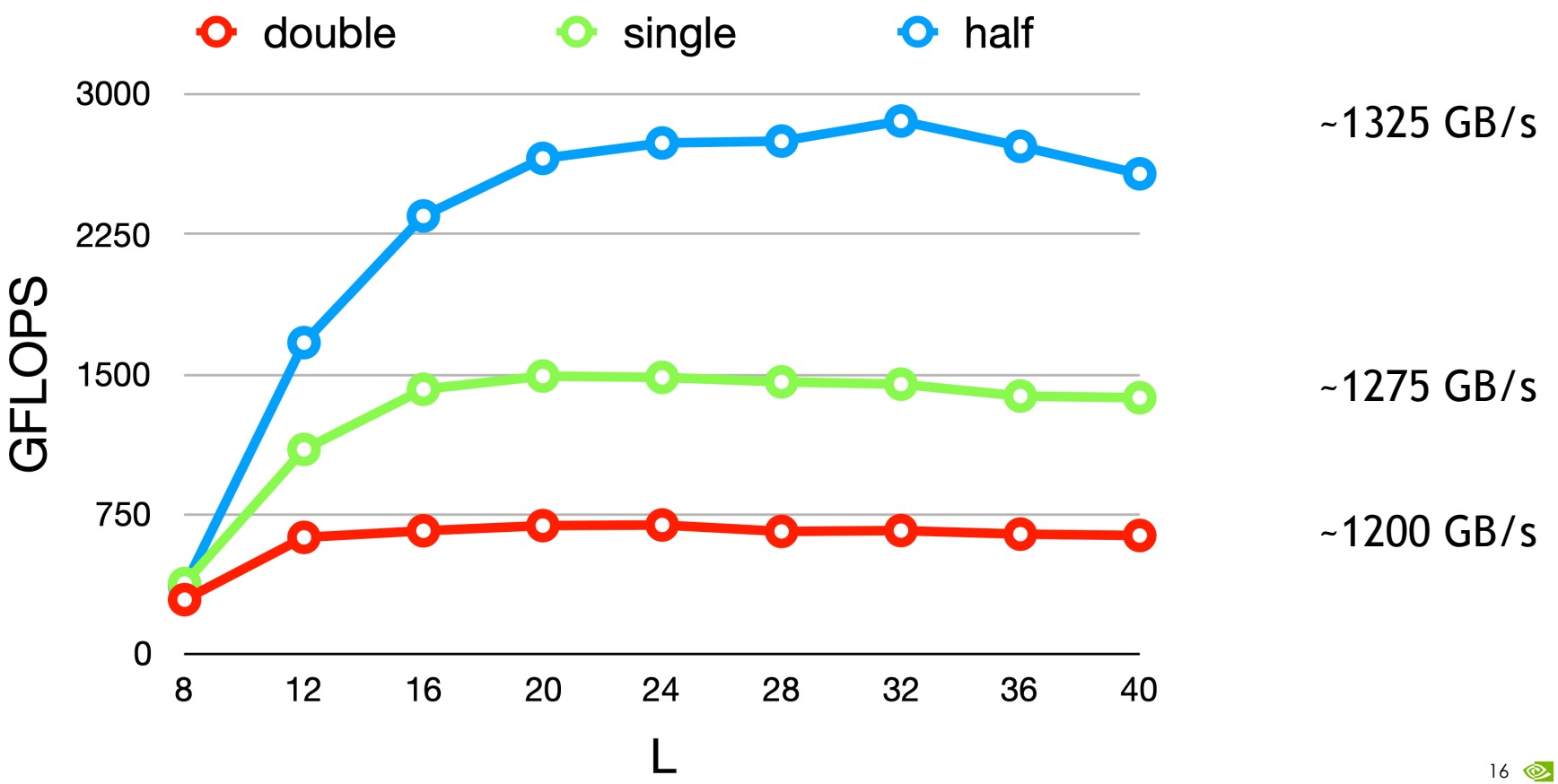
QUDA reduces memory traffic

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



SINGLE GPU PERFORMANCE

“Wilson-clover” stencil (Chroma, V100)

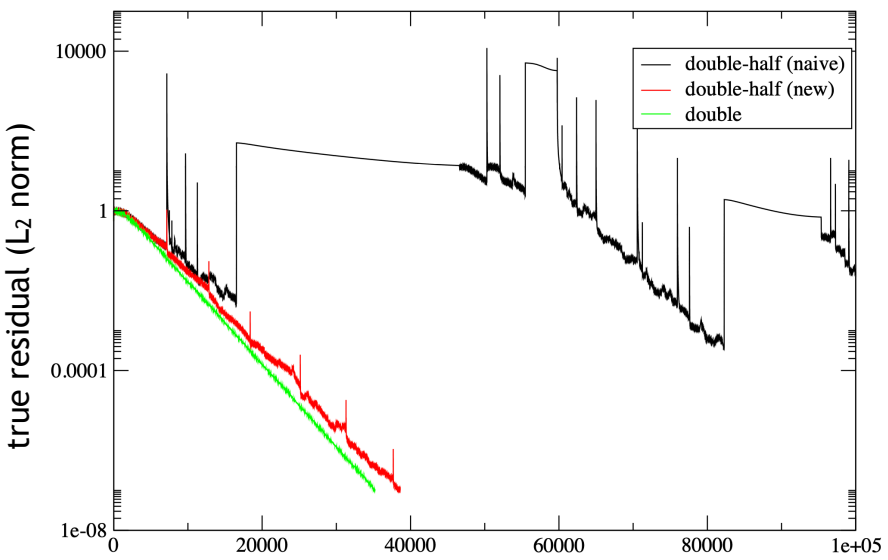


Tesla V100,
CUDA 10.1,
GCC 7.3,
QUDA 1.0

MIXED PRECISION

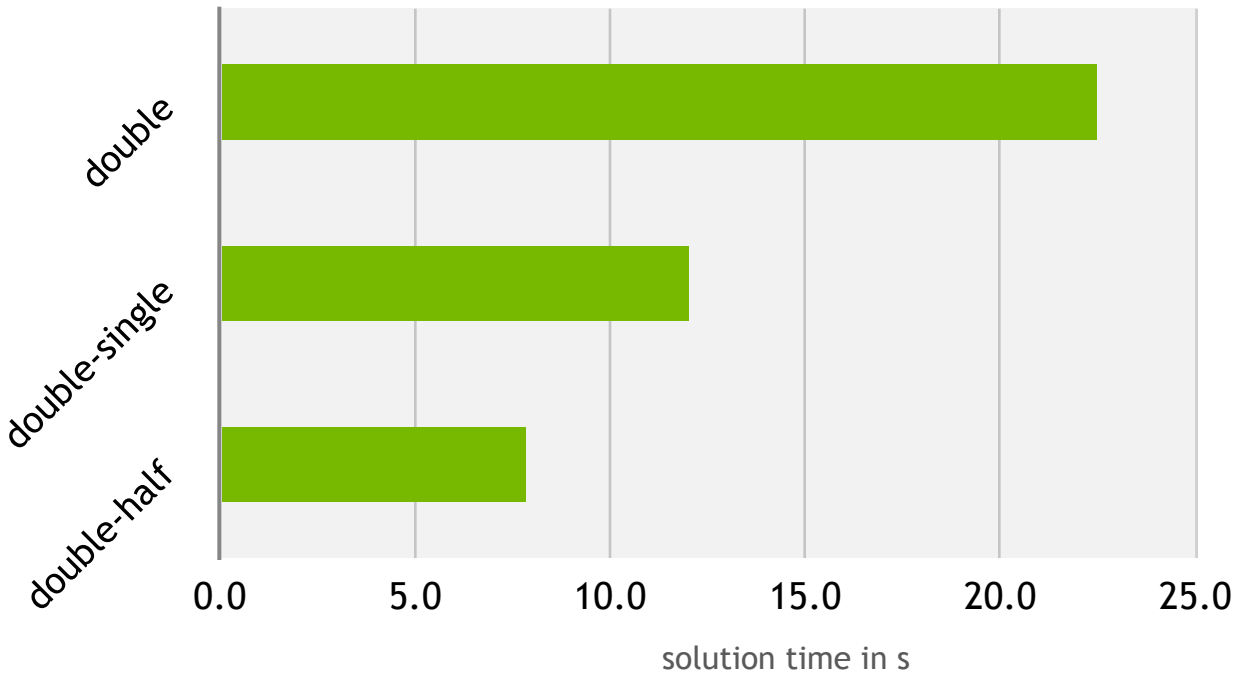
Using your bits wisely

MILC/QUDA HISQ CG, mass = 0.001 => κ~10^6

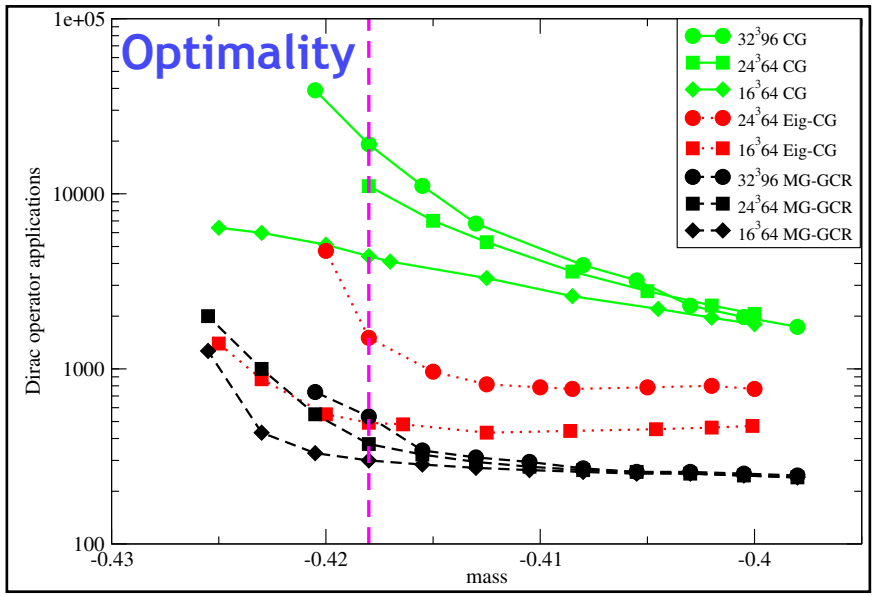


Tesla V100,
CUDA 10.1,
GCC 7.3,
QUDA 1.0

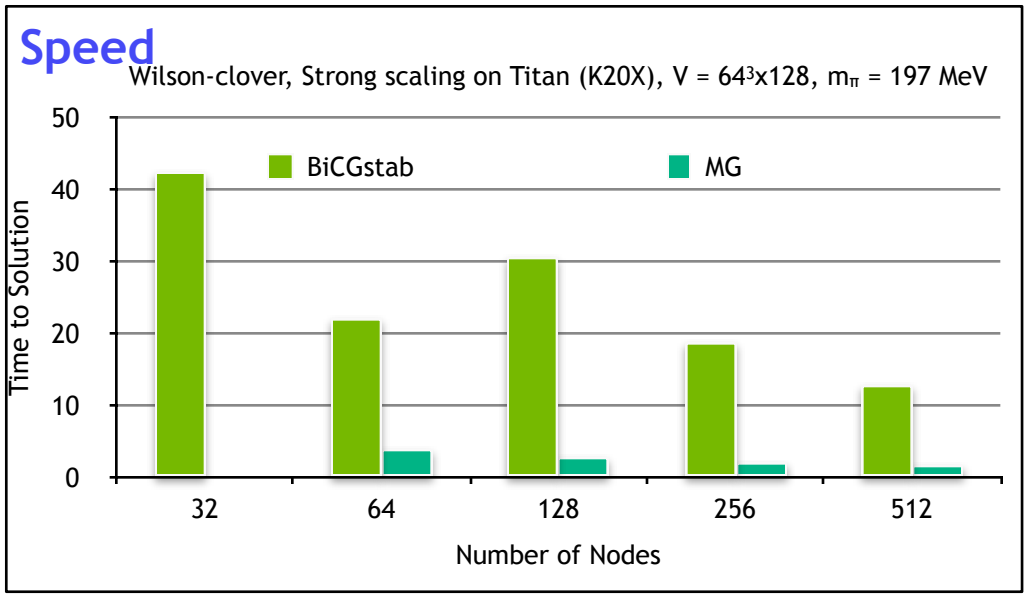
MILC/QUDA HISQ CG solver



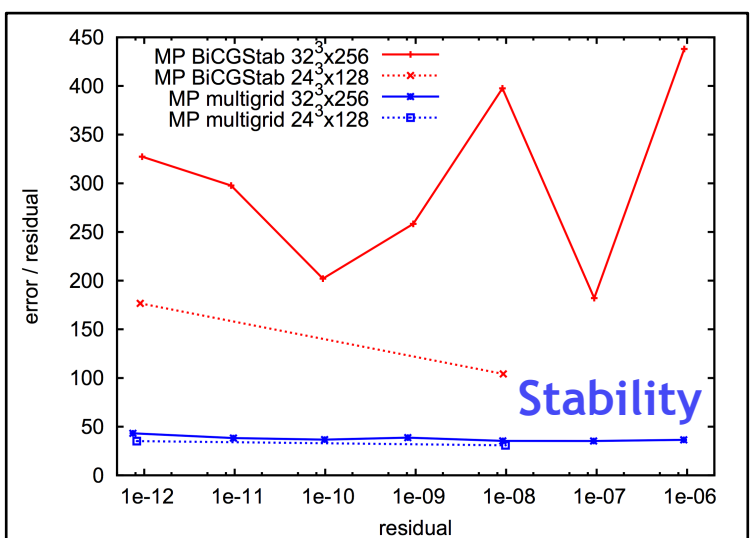
WHY MULTIGRID?



Babich *et al* 2010



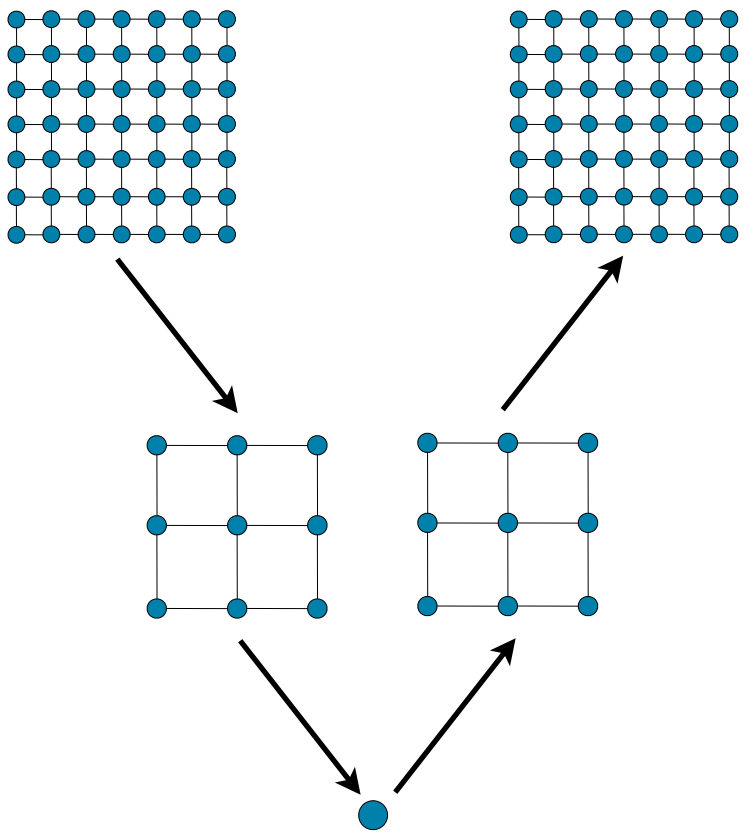
Clark *et al* (2016)



Osborn *et al* 2010

MULTIGRID

The *optimal* method for solving PDE-based linear systems



GPU requirements very different from CPU

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

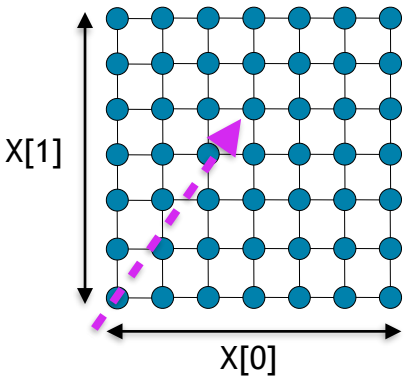
Fine grids run very efficiently

- High parallel throughput problem

Coarse grids are worst possible scenario

- More cores than degrees of freedom
- Increasingly serial and latency bound
- Amdahl's law limiter

SOURCE OF PARALLELISM



1. Grid parallelism

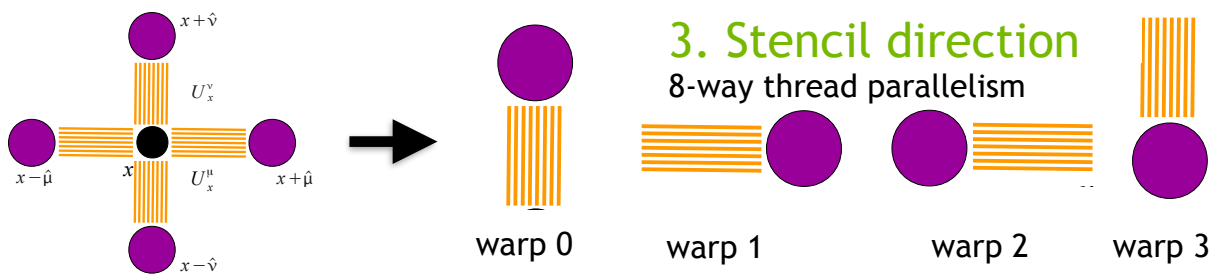
Volume of threads

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 thread parallelism (spin * color)



3. Stencil direction

8-way thread parallelism

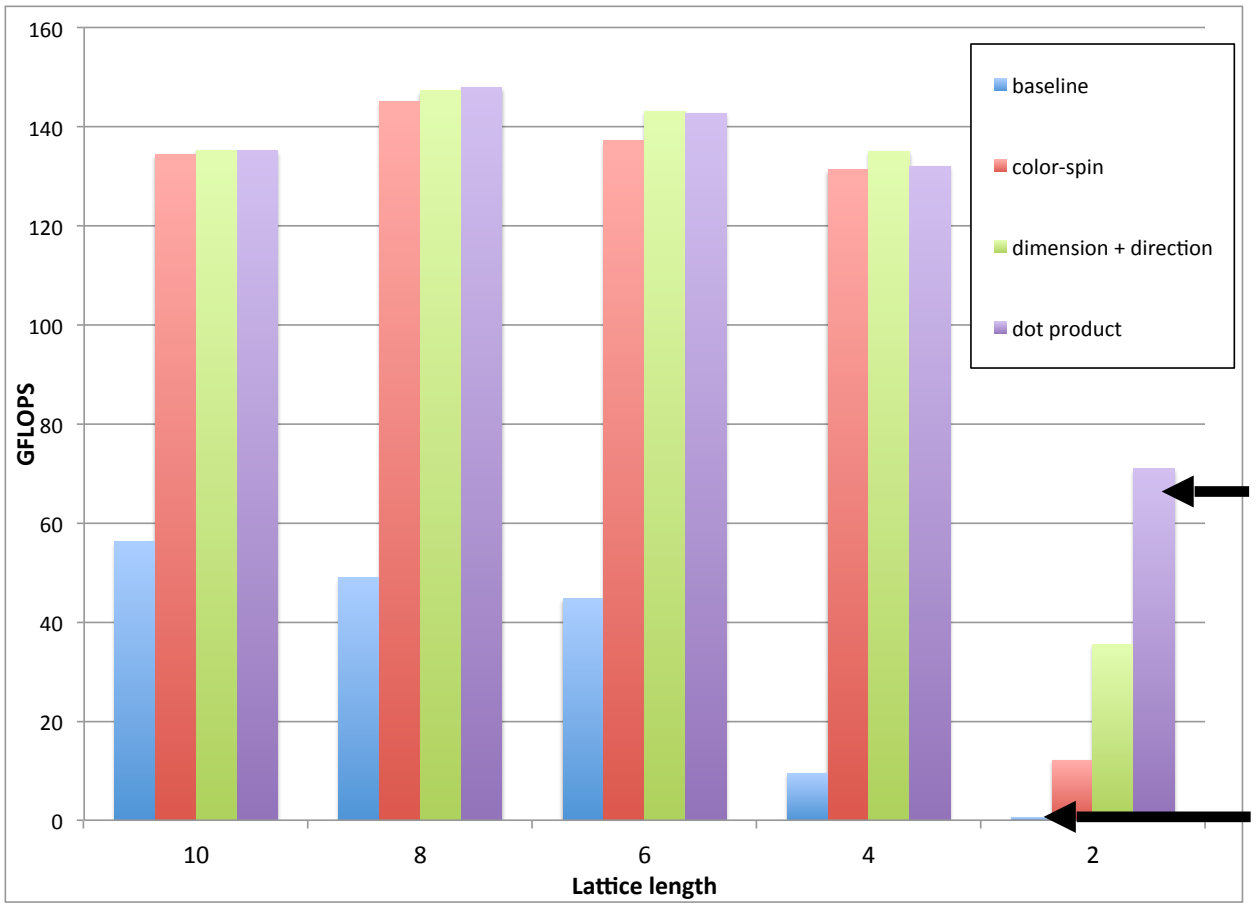
$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \Rightarrow \begin{pmatrix} a_{00} & a_{01} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} + \begin{pmatrix} a_{02} & a_{03} \end{pmatrix} \begin{pmatrix} b_2 \\ b_3 \end{pmatrix}$$

4. Dot-product partitioning

4-way thread parallelism + ILP

COARSE GRID OPERATOR PERFORMANCE

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



c.f. Fine grid operator
~300-400 GFLOPS

24,576-way parallel

16-way parallel

CHROMA HMC ON SUMMIT

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

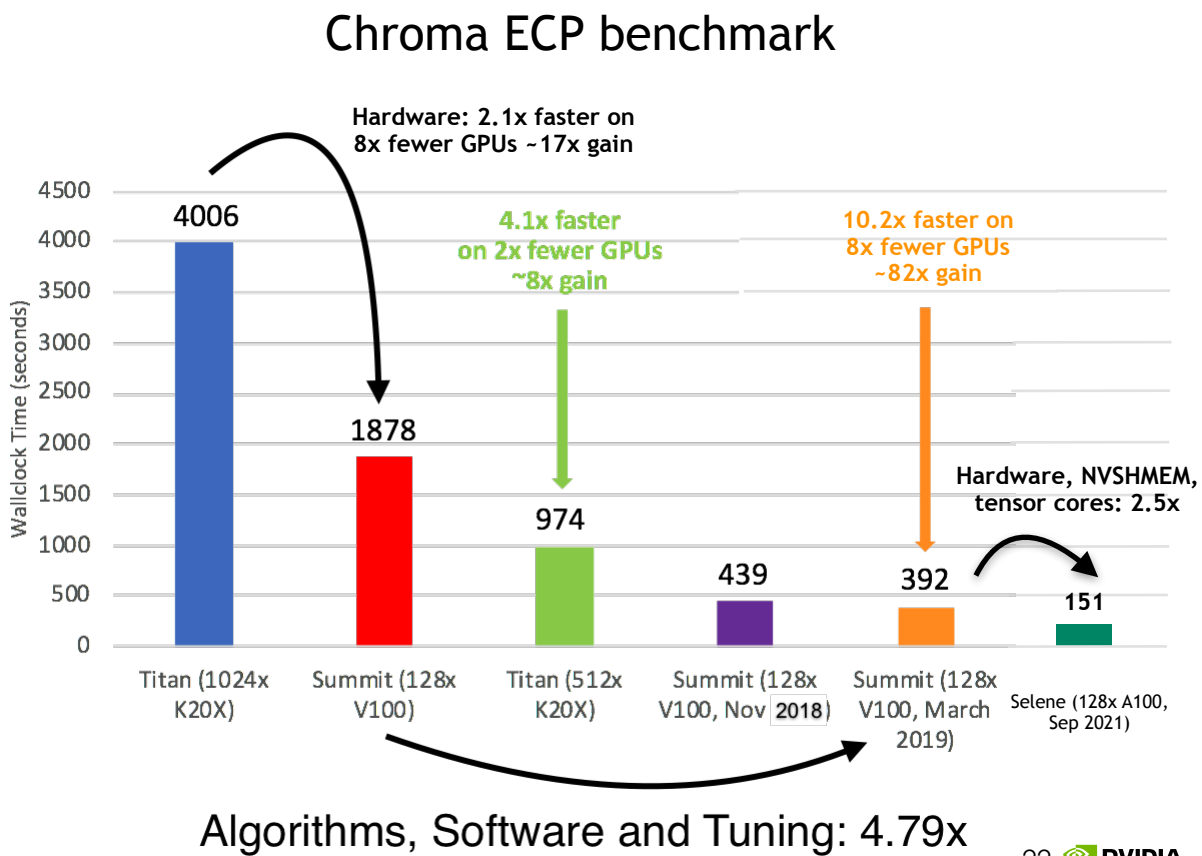
From Titan running 2016 code to Summit running 2019 code we see >82x speedup in HMC throughput

Multiplicative speedup coming from mapping hierarchical algorithm to hierarchical machine

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



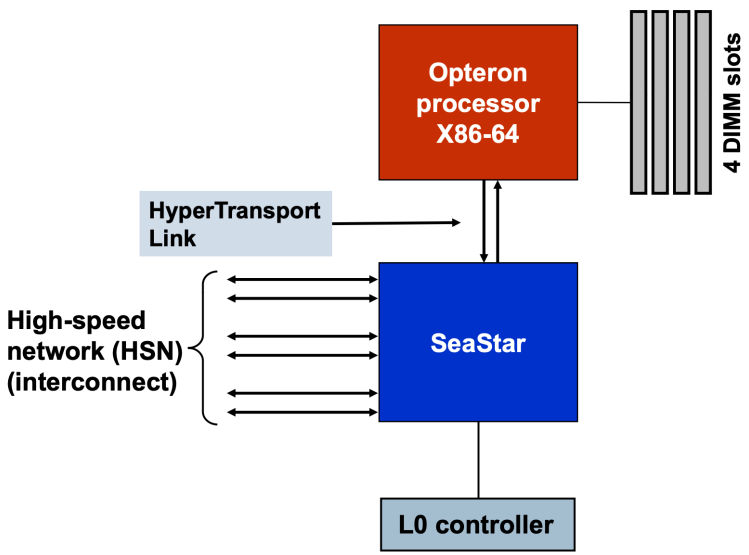

SCALING CHALLENGES

HPC IS GETTING MORE HIERARCHICAL

What does a node even mean?

Cray XT4 (2007)

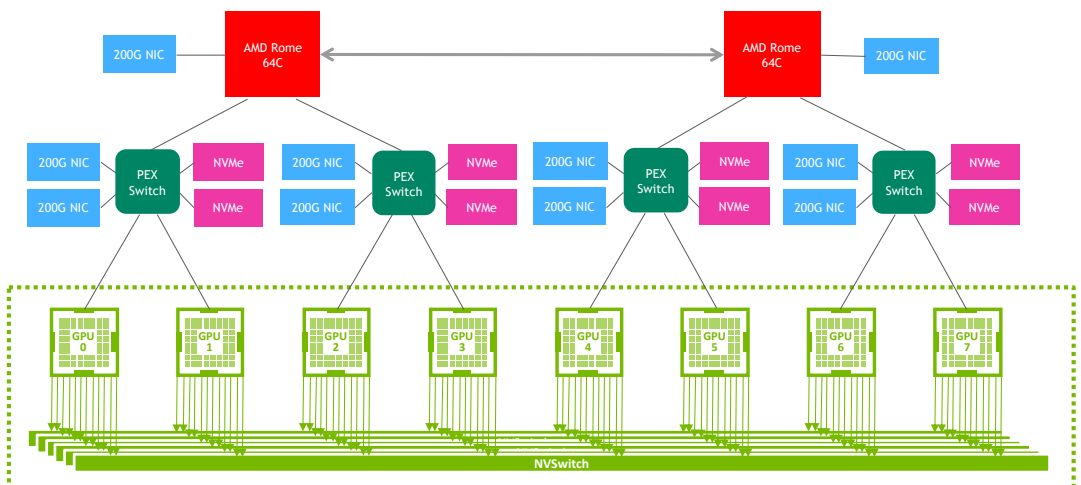
https://www.nersc.gov/assets/NUG-Meetings/NERSCSystemOverview.pdf



Legacy

NVIDIA DGX-A100 (2020)

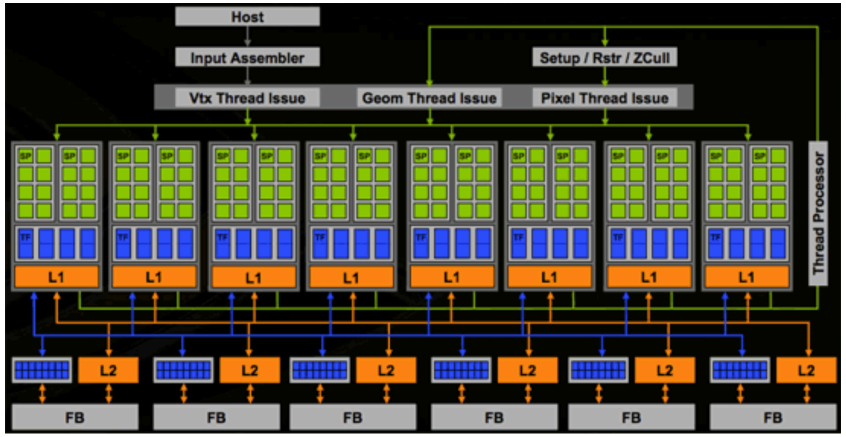
https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf



Current

EVOLVING GPU HIERARCHY

2006



NVIDIA G80 (Tesla)
128 FP32 elements
681M transistors

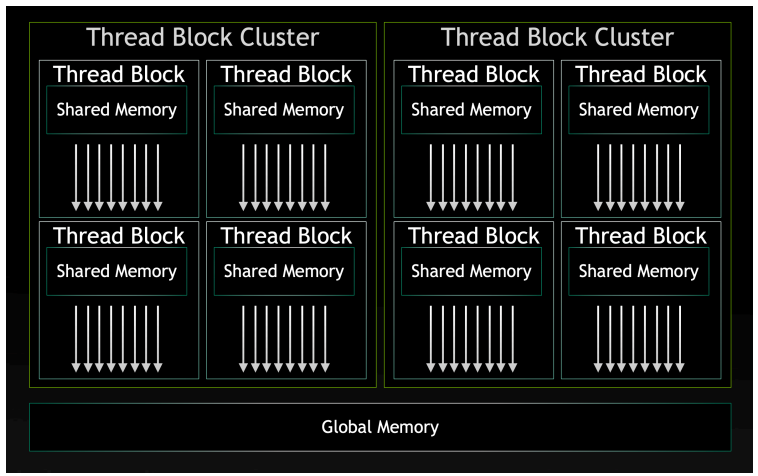
2022

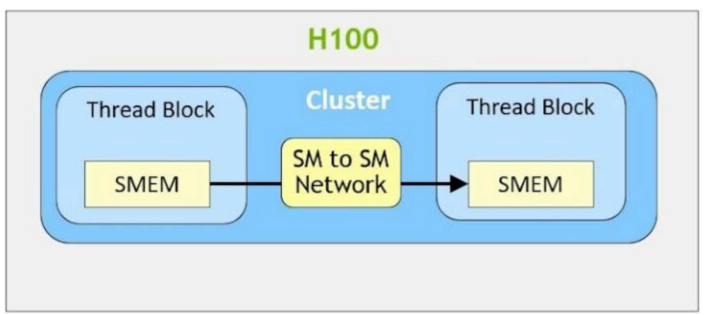
NVIDIA GH100 (Hopper)
18432 FP32 elements
80B transistors

Entire G80 ~ single H100 SM

INTRODUCING CLUSTERS

Synchronization at all levels





Warp

Communication through lane shuffling

Thread block

Communication through shared memory

Thread block cluster

New!

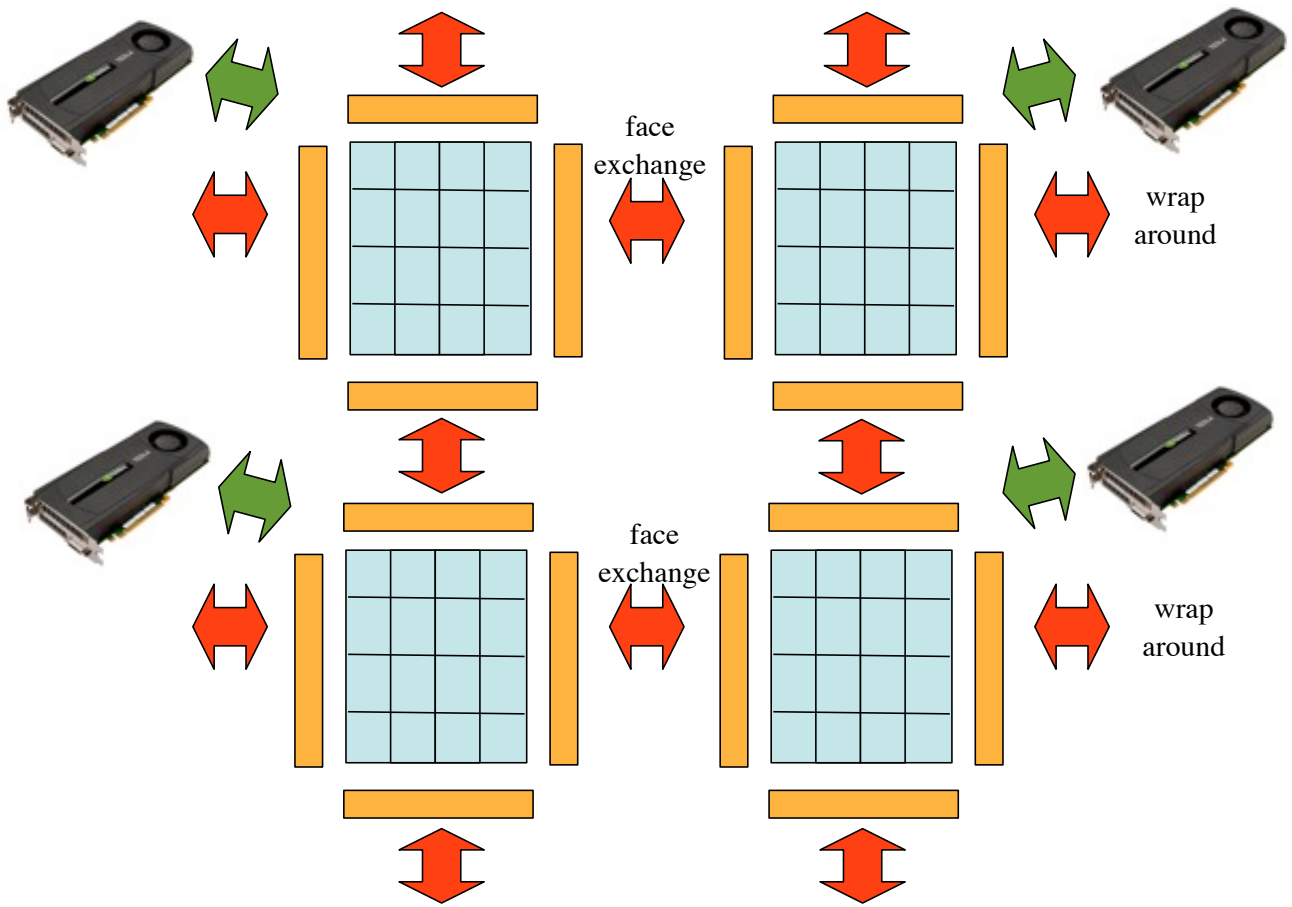
Communication through distributed shared memory

Global

Communication through global memory

Kernel boundary or grid synchronization

MULTI-GPU BUILDING BLOCKS



Halo packing Kernel

Interior Kernel

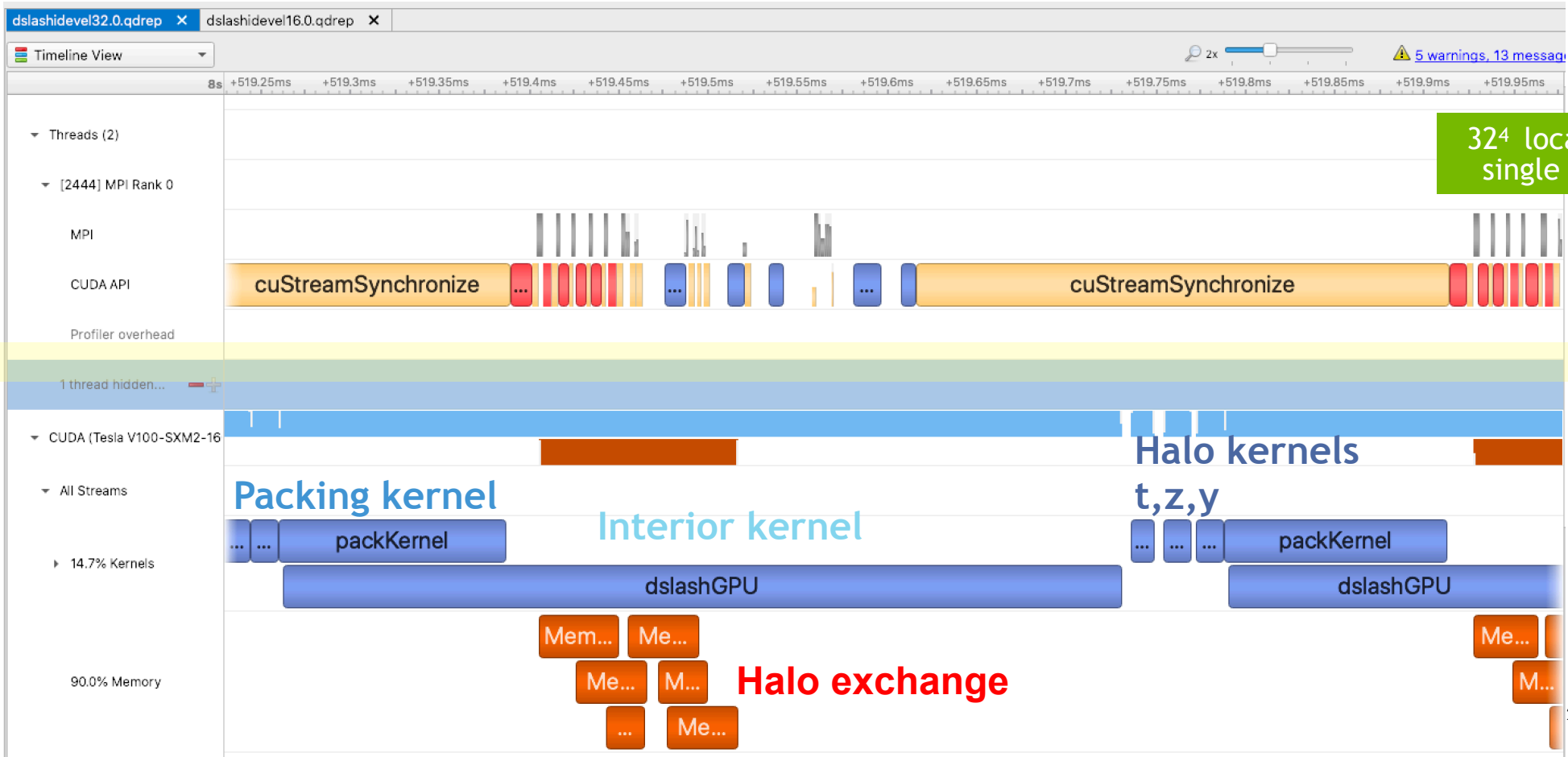
Halo communication

Overlap

Halo update Kernel

MULTI-GPU PROFILE

overlapping comms and compute

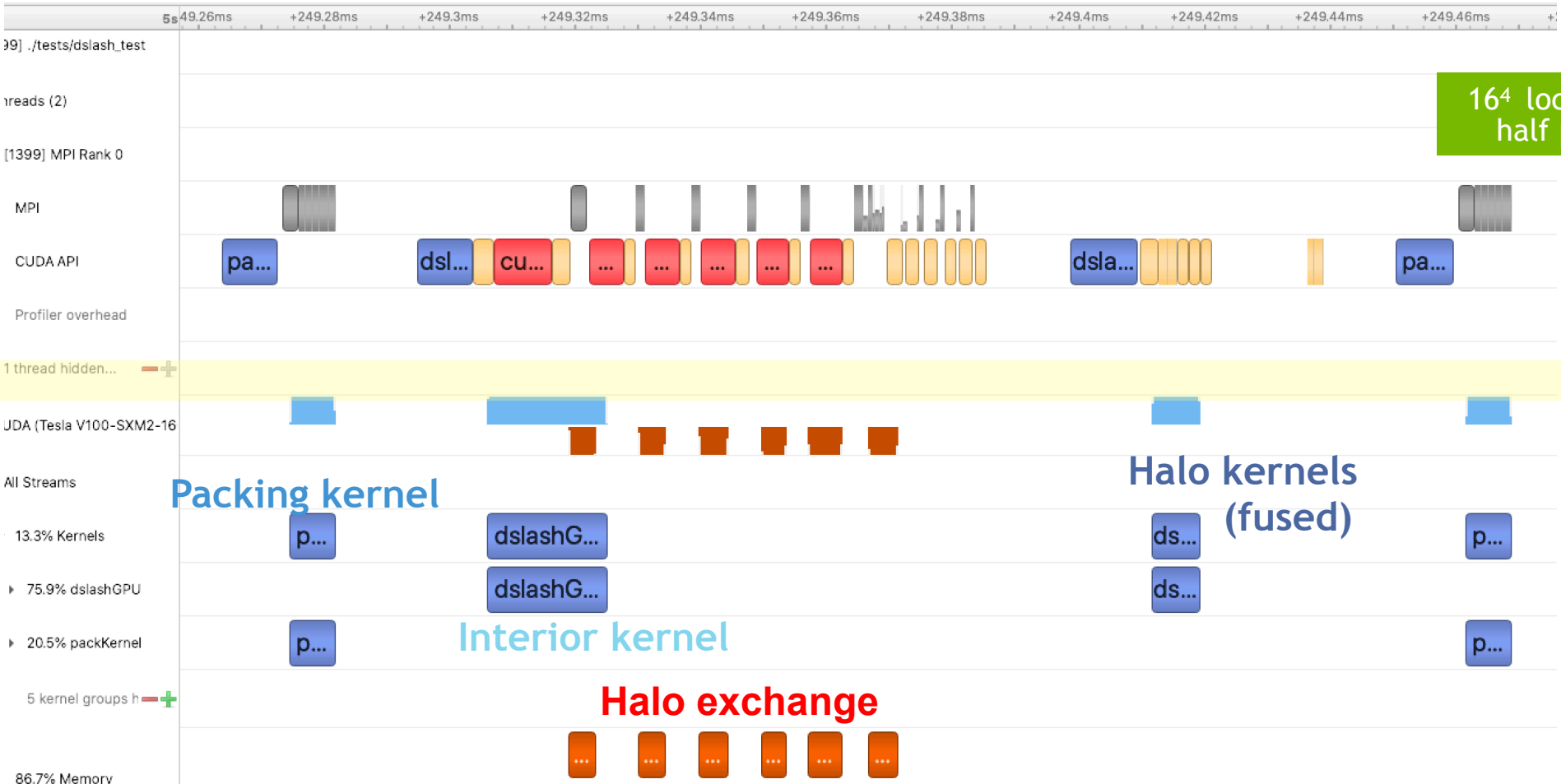


32^4 local volume, single precision

DGX-1V, 1x2x2x2 partitioning

STRONG SCALING PROFILE

~~overlapping~~ comms and compute



16^4 local volume, half precision

DGX-1V, 1x2x2x2 partitioning

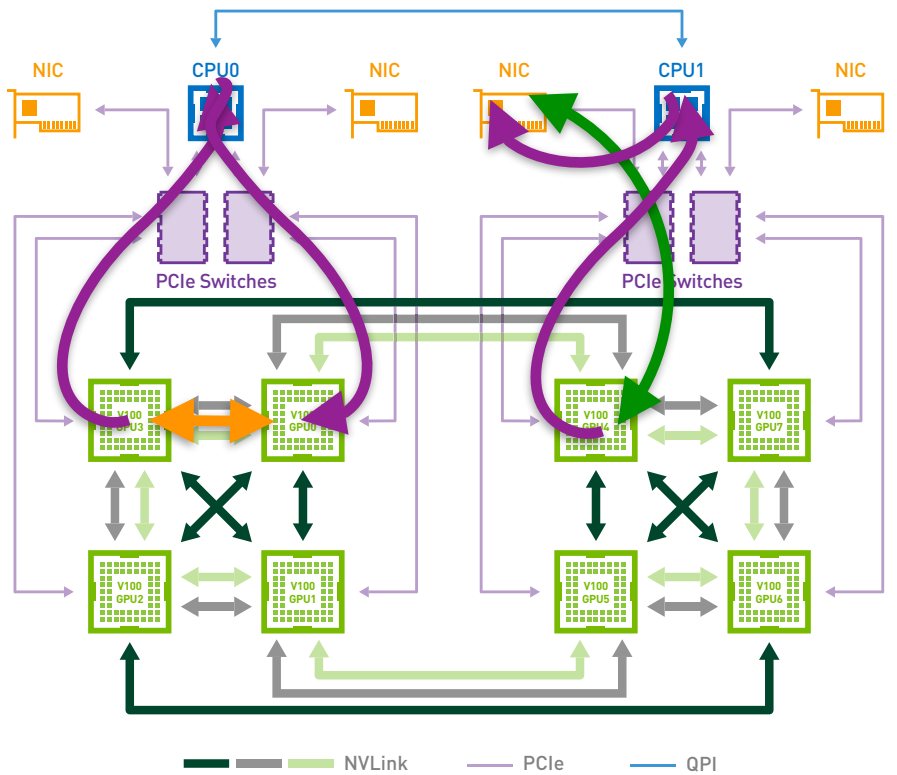
WHAT IS THE PROBLEM?

It’s not just data movement we need to minimize

Task marshaling from a lower level of hierarchy (e.g., host) adds latency

Data consistency requires synchronization between CPU and GPU

Ideally: offload of task marshaling to GPU thread to have same locality as data



Available in NVIDIA HPC toolkit

NVSHMEM

Implementation of OpenSHMEM1, a Partitioned Global Address Space (PGAS) library

NVSHMEM features

- Symmetric memory allocations in device memory
- Communication API calls on CPU (standard and stream-ordered)
- Kernel-side communication (API and LD/ST) between GPUs

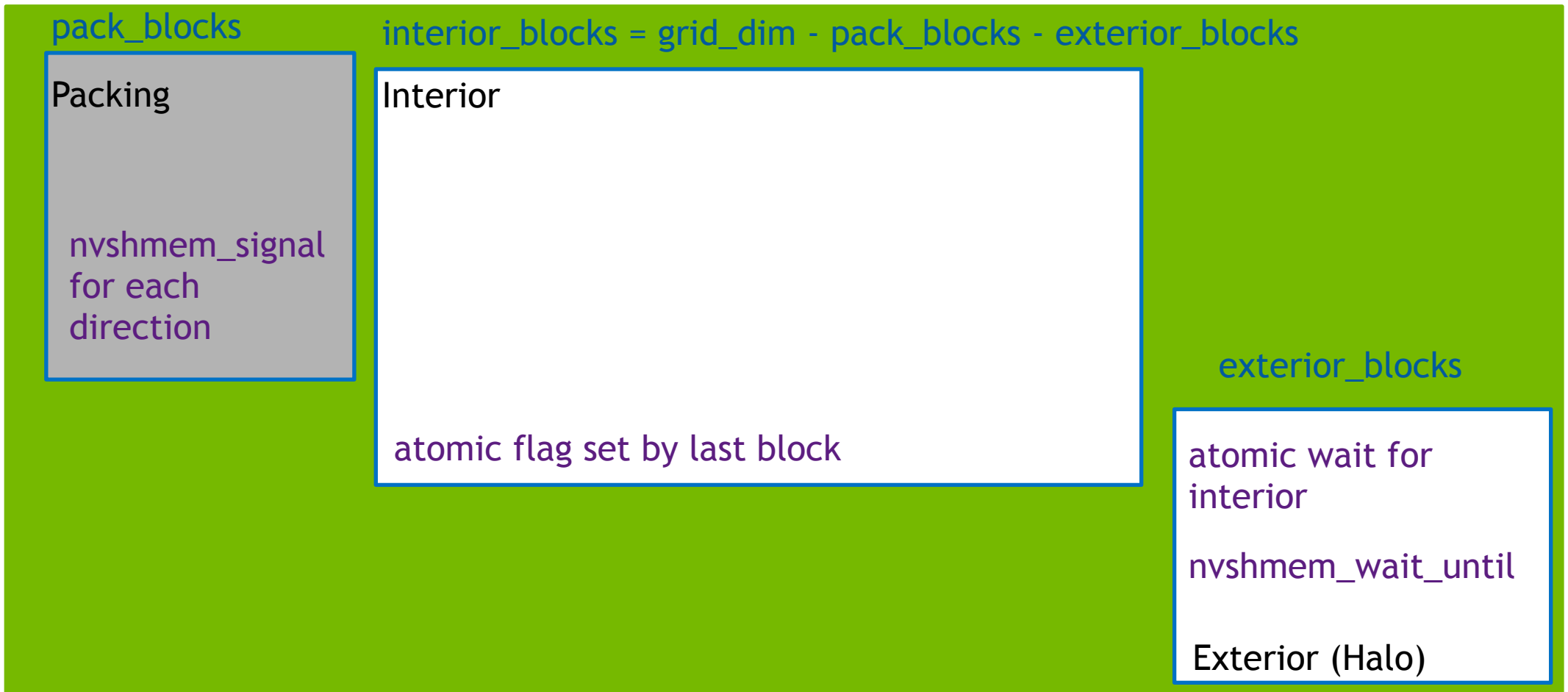
NVLink and PCIe support (intra-node)

InfiniBand support (inter-node)

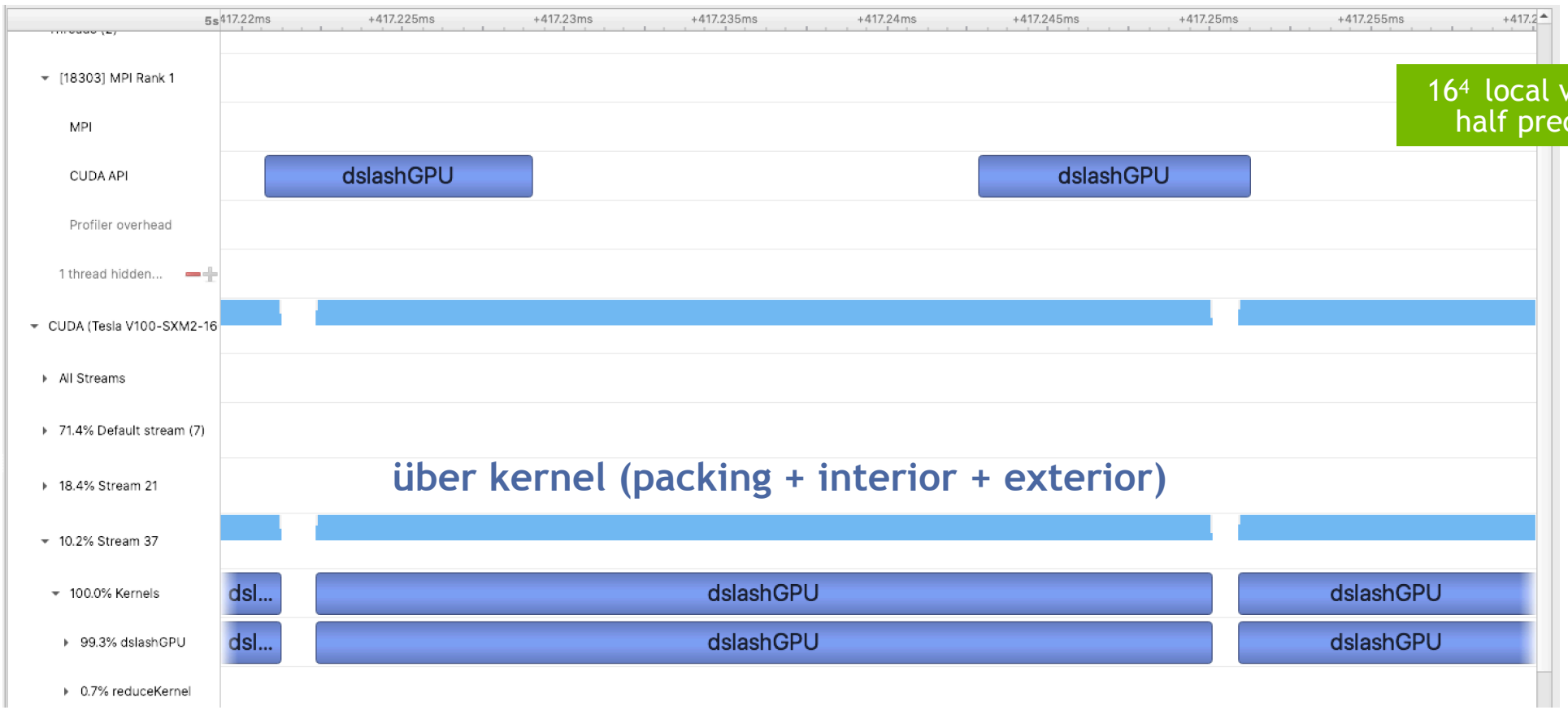
Interoperability with MPI and OpenSHMEM libraries

[1] *SHMEM from Cray's "shared memory" library, https://en.wikipedia.org/wiki/SHMEM*

DSLASH ÜBER KERNEL



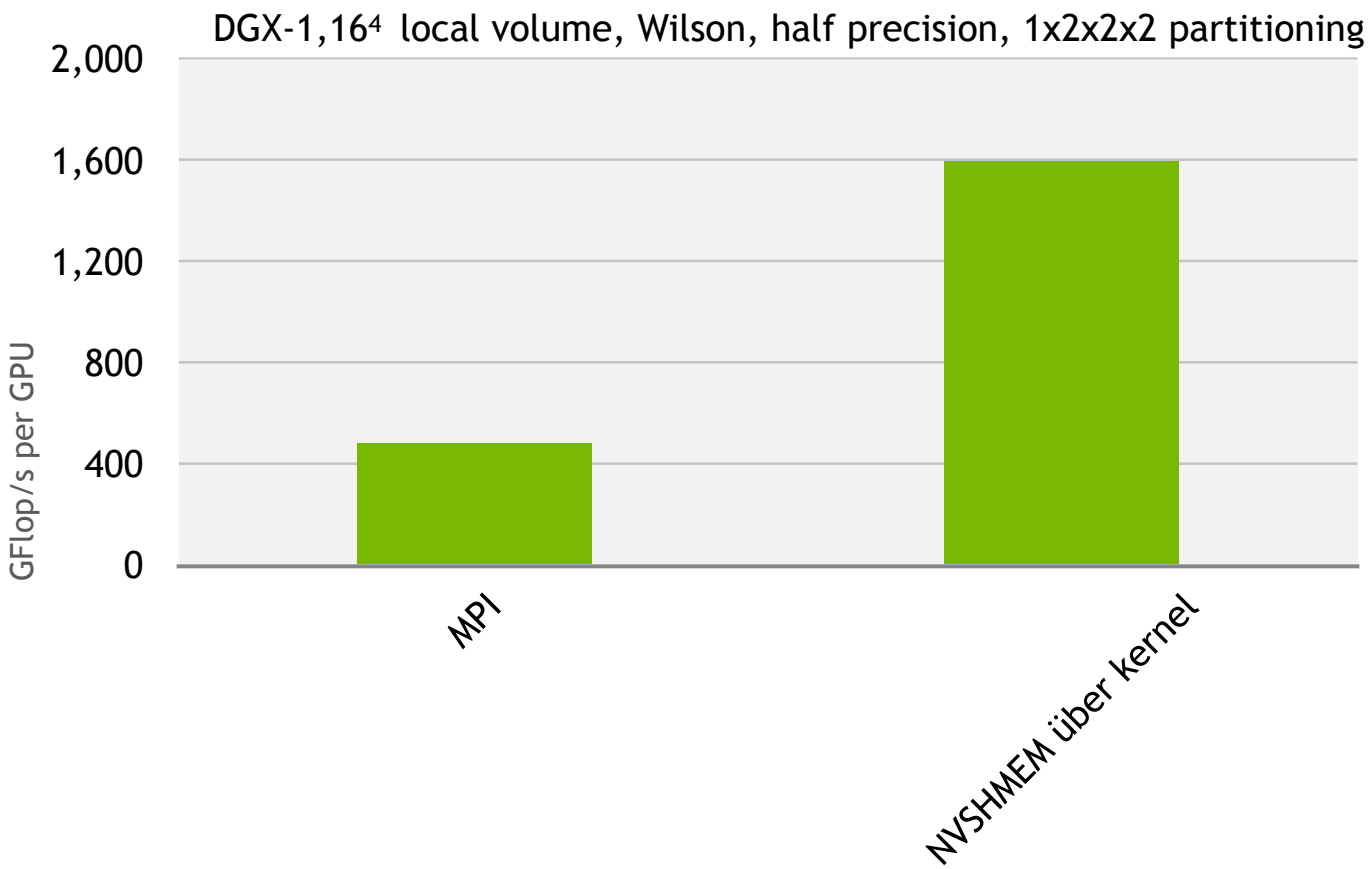
ÜBER KERNEL



16^4 local volume, half precision

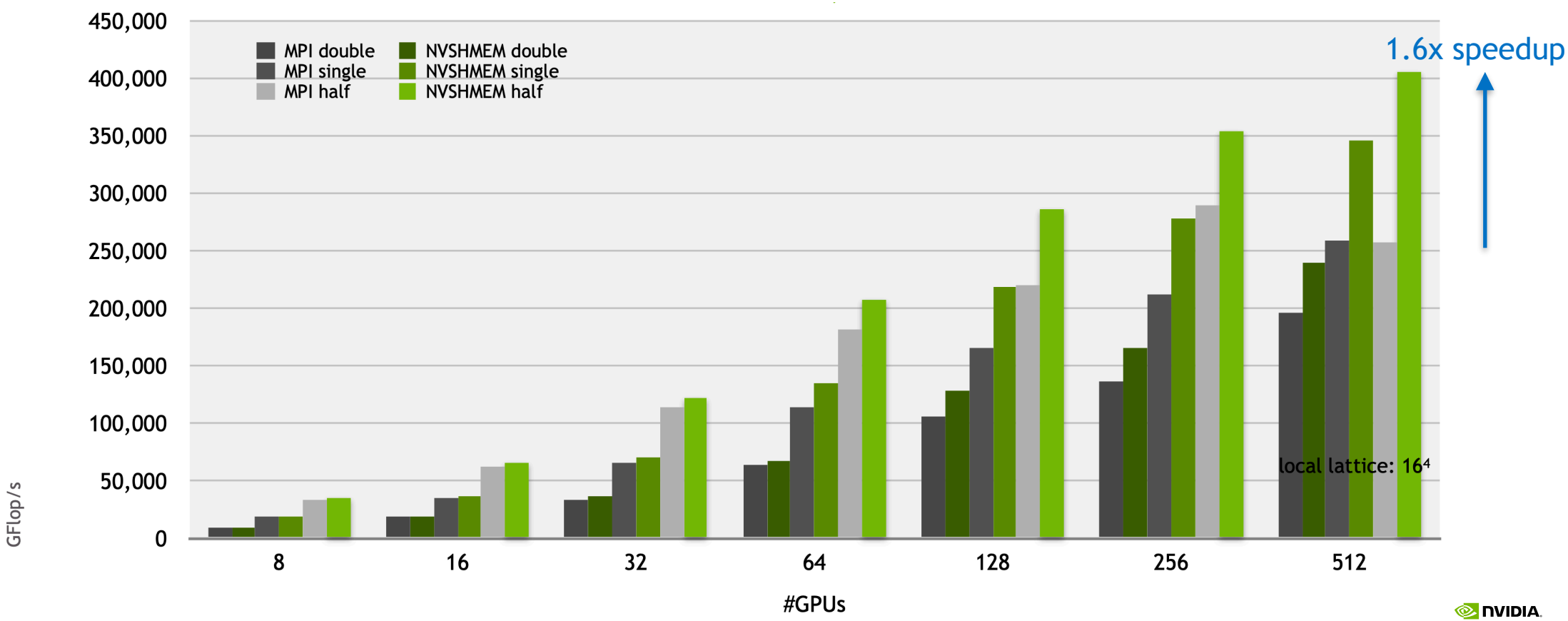
DGX-1, 1x2x2x2 partitioning

LATENCY REDUCTIONS



SELENE STRONG SCALING

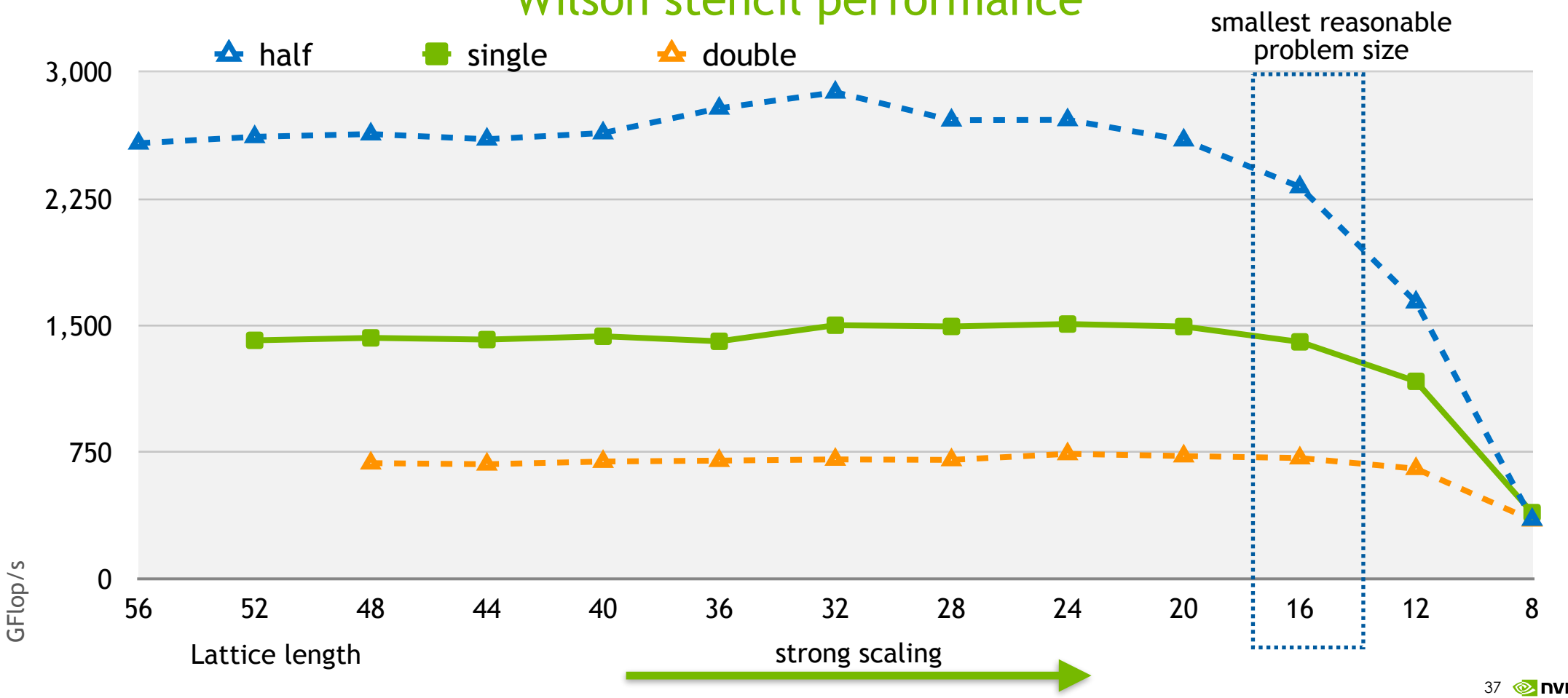
Global volume 64^3x128



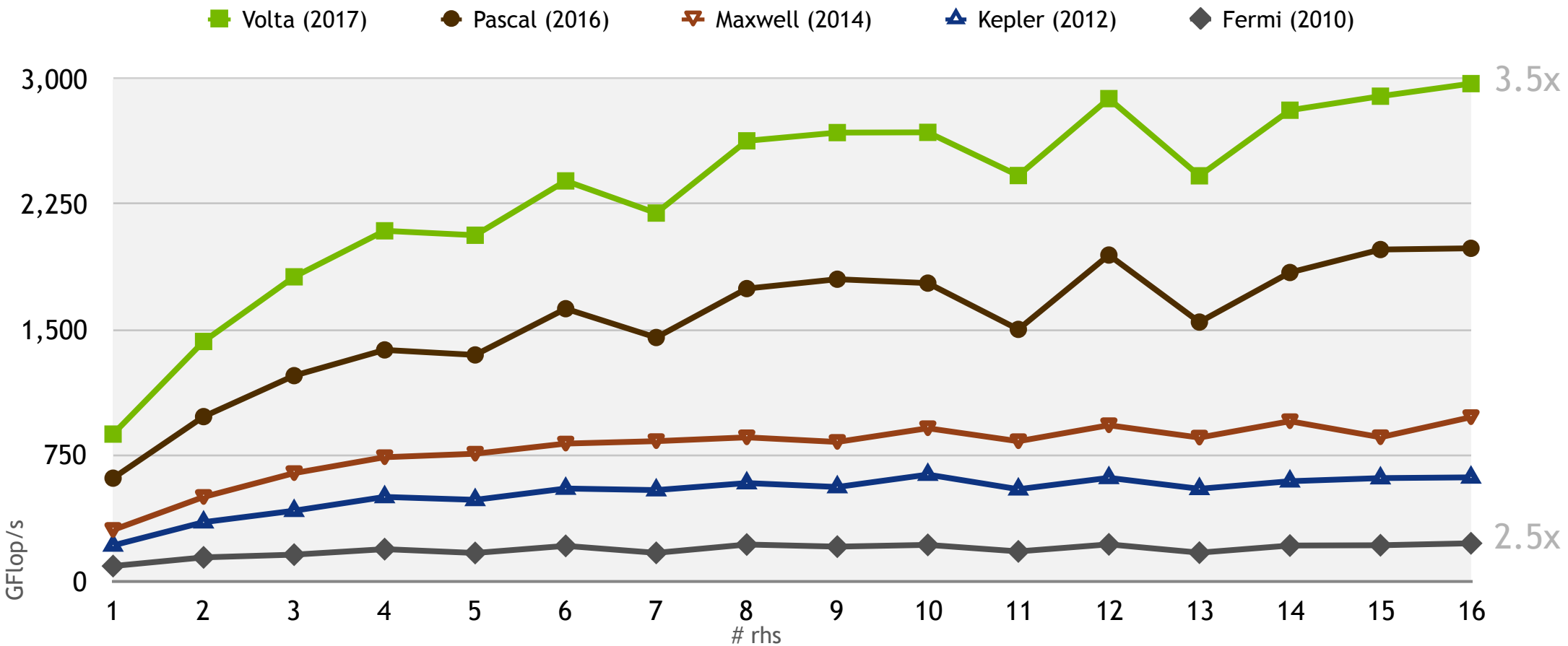
PARALLELISM AND LOCALITY

PARALLELISM ISN'T INFINITE...

Wilson stencil performance

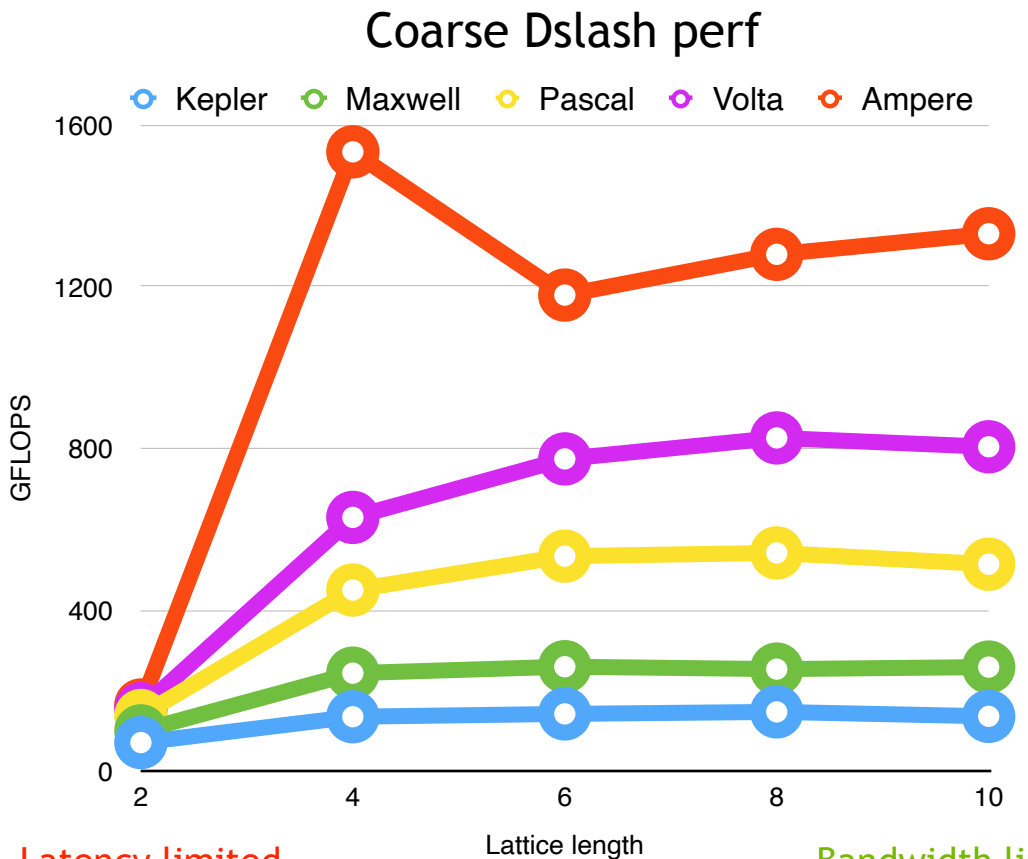


...UNLESS WE MAKE IT

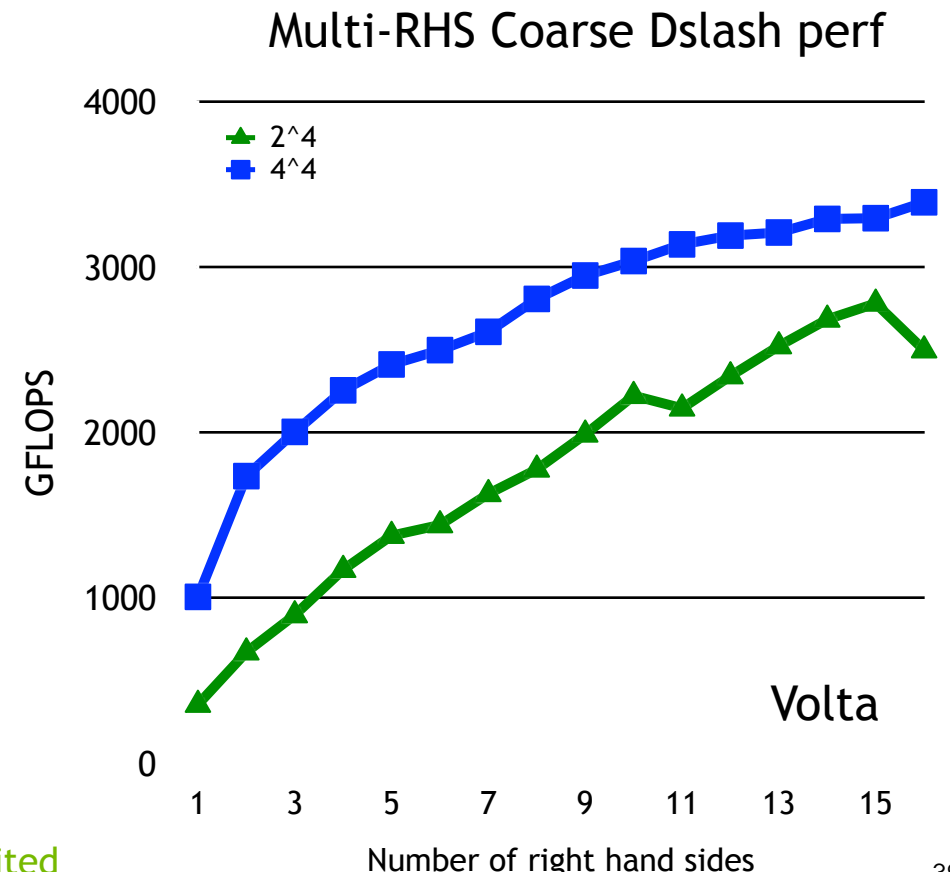


MULTIGRID

Gets harder with every generation



Latency limited

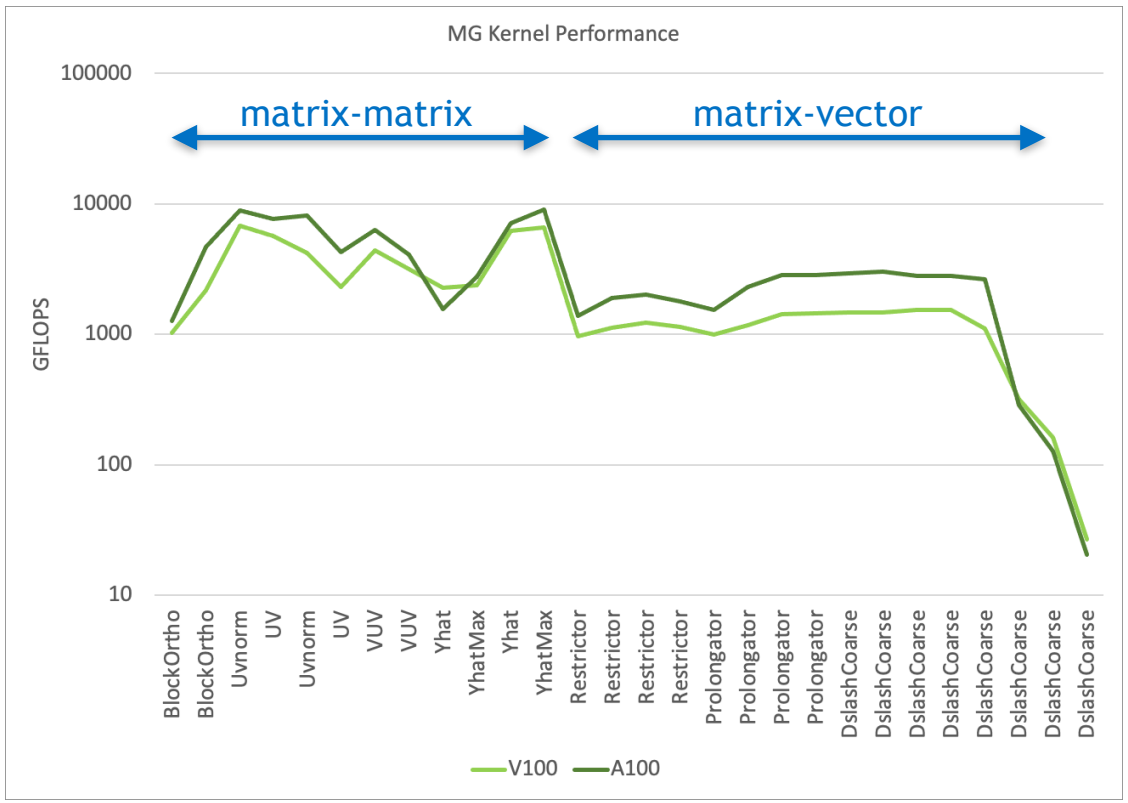


Bandwidth limited

MULTIGRID COMPONENTS

Wilson-clover solve (Chroma), $V=32^4$

Note: log scale



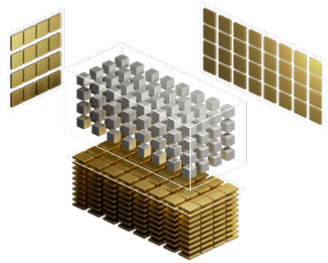
Classify multigrid kernels

- Matrix-matrix: flops / cache bound
- Matrix-vector: bandwidth bound

"Structured" GEMMS pervade the MG setup

- Coarse operator construction
- Block orthogonalization

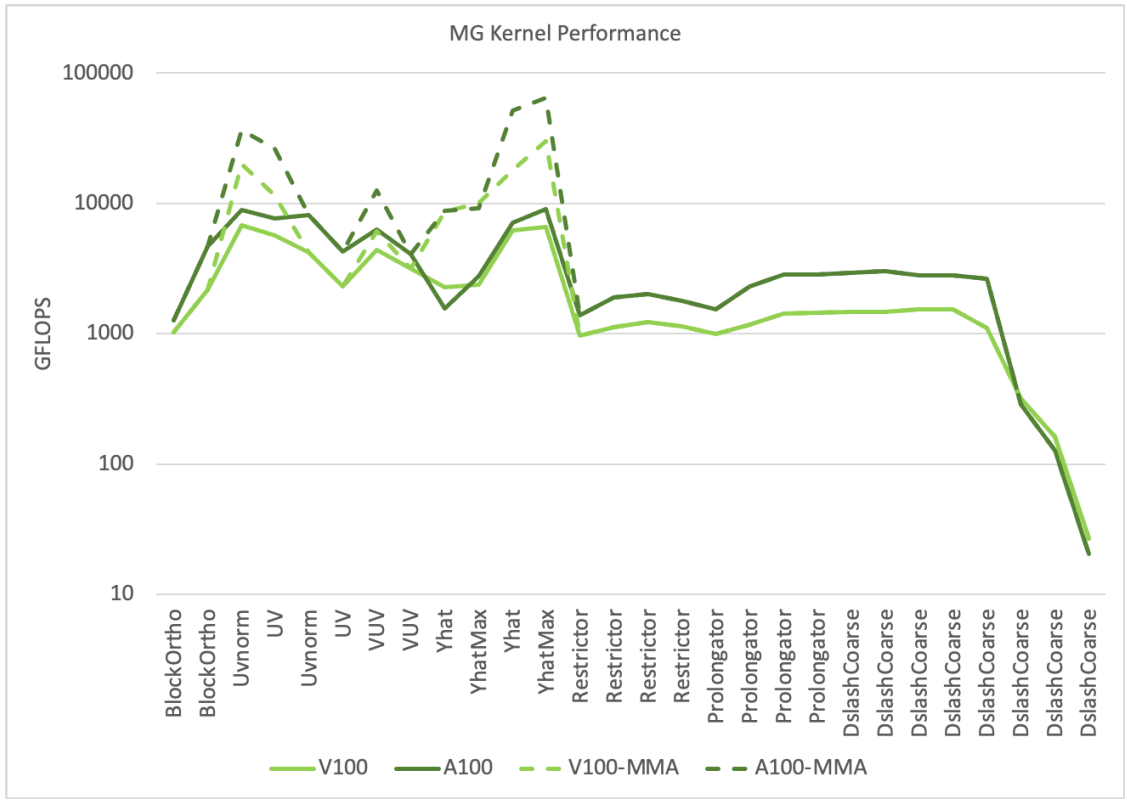
Tensor cores?



MULTIGRID COMPONENTS

Wilson-clover solve (Chroma), $V=32^4$

Note: log scale



MG is a preconditioner

- 16-bit precision is perfectly adequate
- No impact on convergence rate

Majority of MG setup kernels now implemented using tensor cores

- 1.5x-10x kernel speedups observed

Future work: reworking the pipeline to expose more in explicit matrix-matrix form

REWORKING THE LQCD PIPELINE

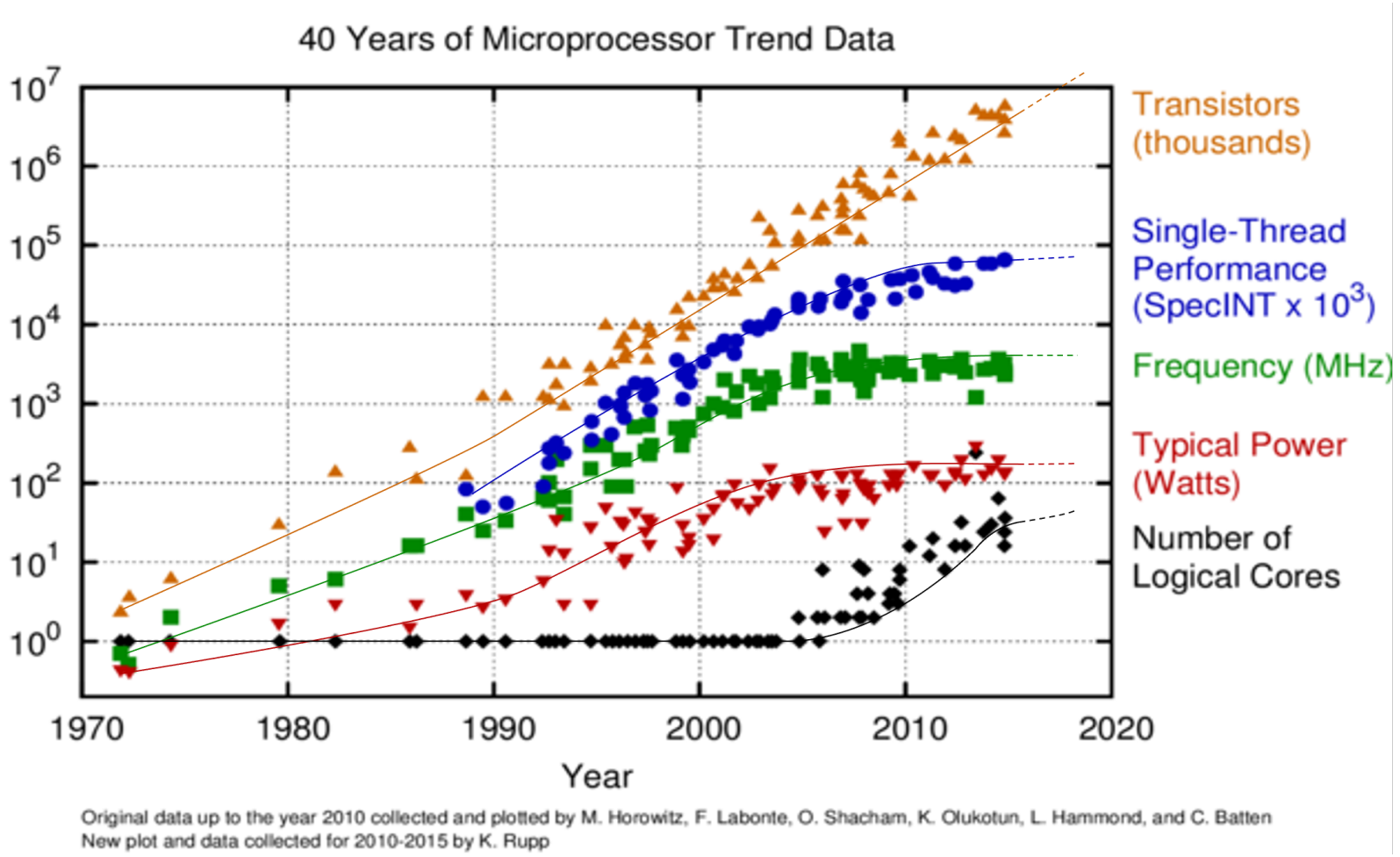
slaphnn collaboration

2 nucleon (2 baryon) and 2 hadron (ππ, Kπ) and meson-baryon catering 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 ~ $(N_\psi^2)/(2N_\psi)$

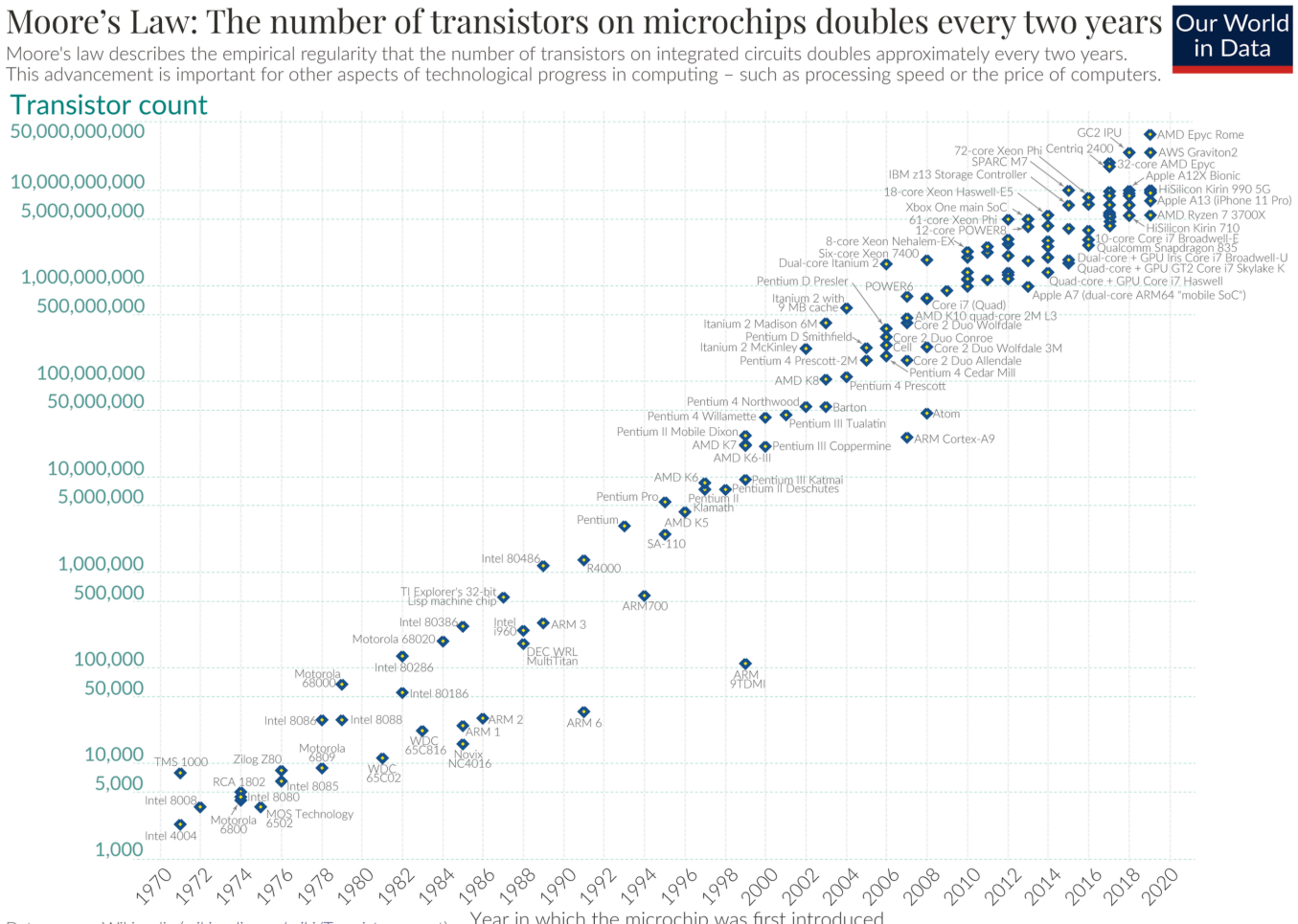

FUTURE CHALLENGES

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

PROTRACTED DEATH OF MOORE'S LAW



EXASCALE IS FINALLY HERE

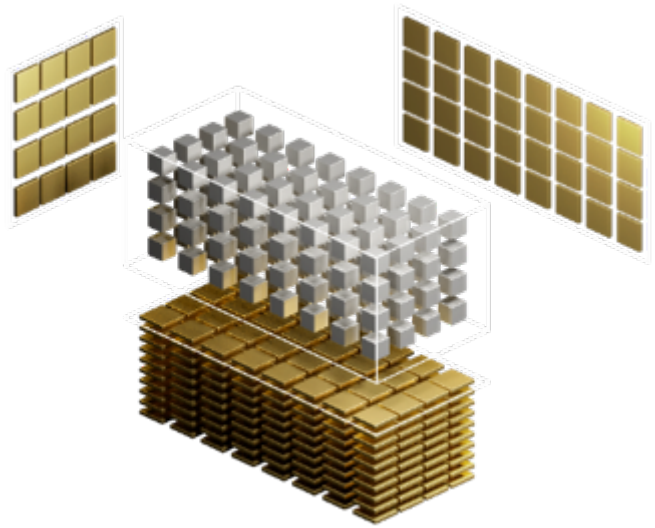
Zettascale will be even harder

Matrix and tensor operations required to saturate the machine

Low precision will go much faster

Extreme parallelism required

Hierarchy and Locality must be considered



Follow trends towards future architectures and seize disruptive opportunities

WHAT COULD LQCD DO WITH 100X MORE?

Getting nowhere even faster?

Can we get significantly more science with 100x more *specialized* compute?

Can we bludgeon our way past critical slowing down with HMC?

Or solve it with an algorithmic evolution (sMD, Fourier acceleration, etc.)

Or do we need a *completely different* approach…

That can more naturally use all those AI flops that are coming?


NVIDIA