(TRADITIONAL) COMPUTING CHALLENGES IN LATTICE FIELD THEORY

Kate Clark @ MLT2021

OUTLINE

Introduction

GPUs for Lattice QCD

Scaling Challenges

Future Challenges

Caveat: my focus is solver, gauge generation and GPU focussed, and not representative of all LQCD challenges

If you have an LQCD-type problem that you're struggling to get working on GPUs: mclark@nvidia.com

INTRODUCTION

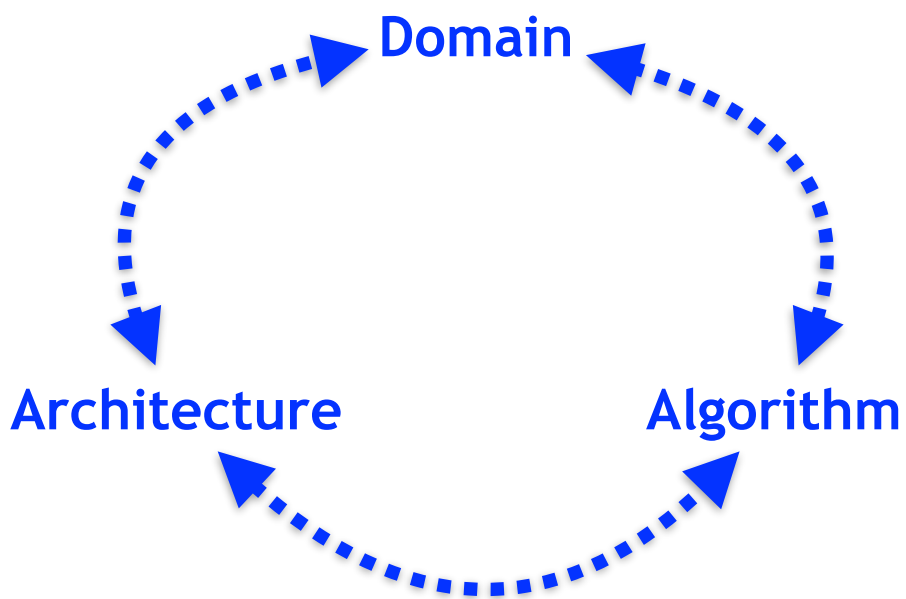
WHY APPLICATION CO-DESIGN?

What do I do?

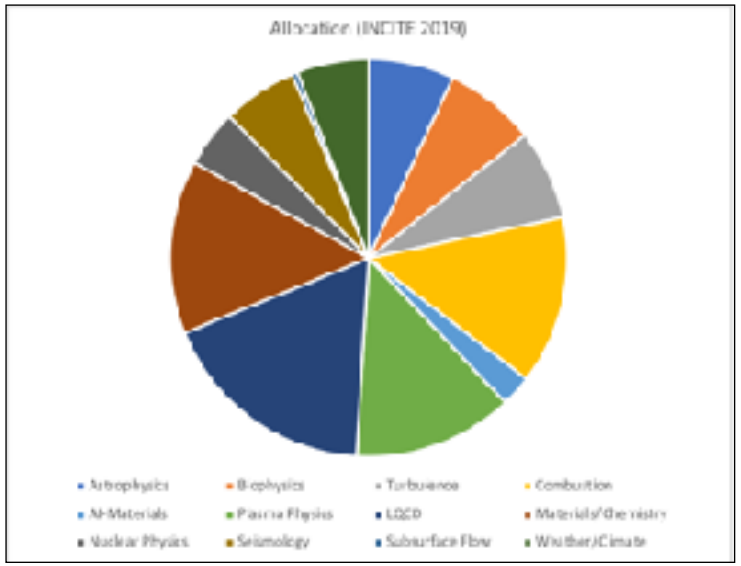
Understand what are the performance and scaling limiters today and *tomorrow*

Intercept disruption and seize ensuing opportunities

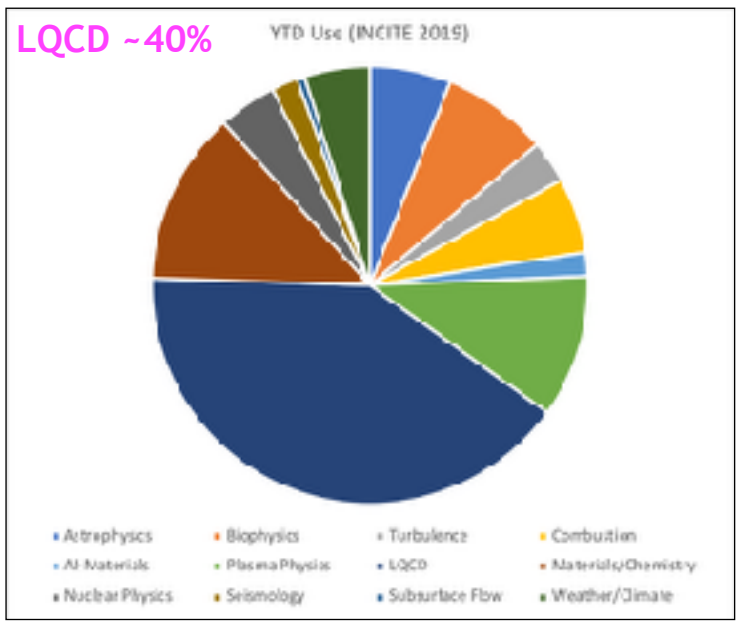
Ensure smooth transition while reducing time to science



LATTICE QCD IS HUNGRY

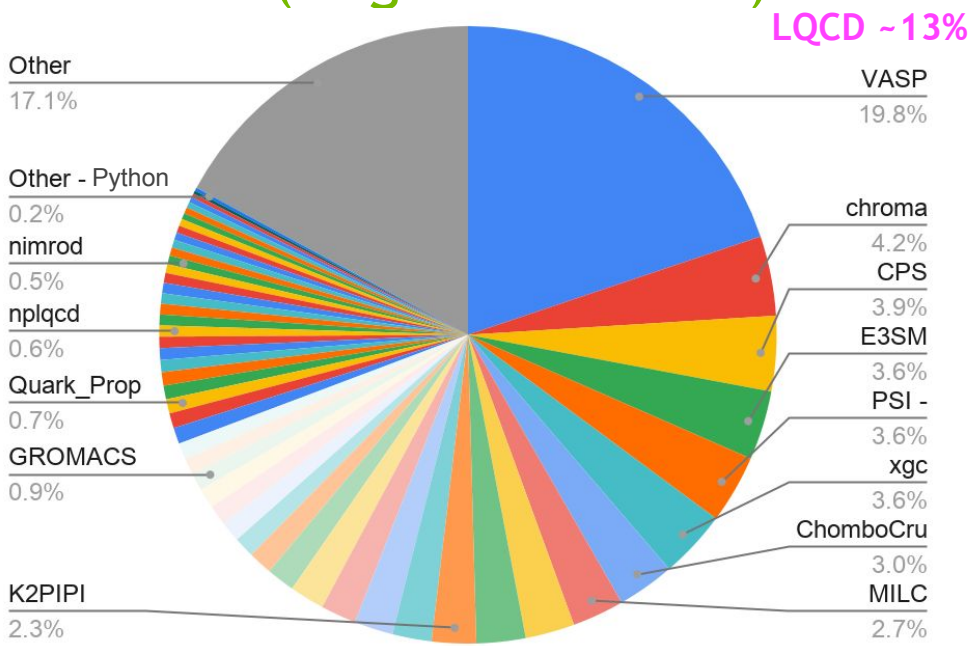


Summit cycle breakdown in INCITE allocation

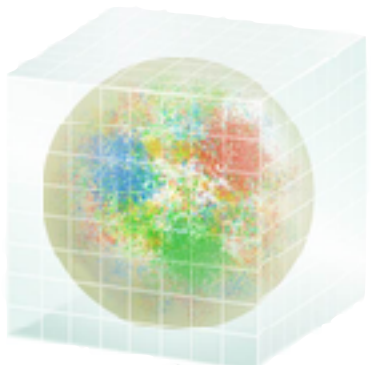


Summit cycle breakdown in INCITE use

NERSC Utilization (Aug '17 - Jul'18)



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

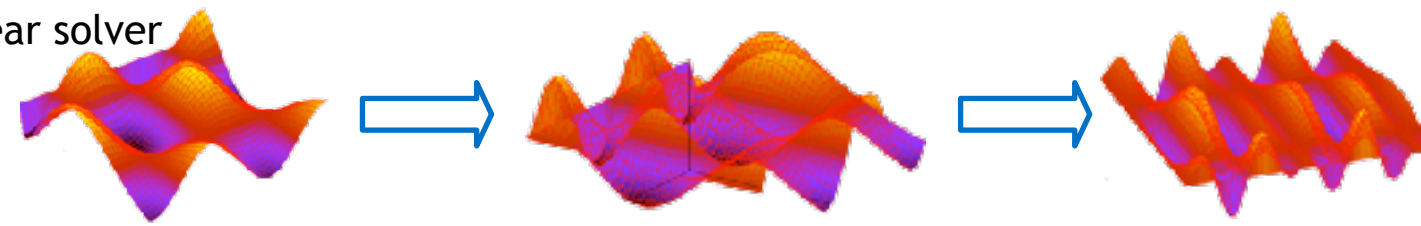
- Traditionally highly optimized on every HPC platform for the past 30 years
- Jobs often run at the 1000+ GPU scale

STEPS IN AN LQCD CALCULATION

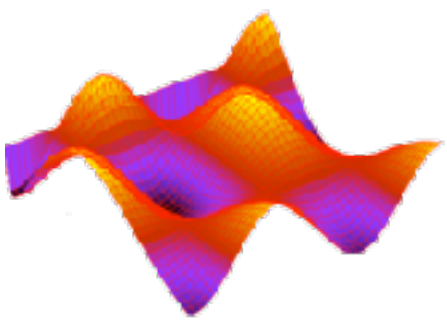
$$D_{ij}^{\alpha\beta}(x,y;U)\psi_j^{\beta}(y) = \eta_i^{\alpha}(x)$$

or Ax = b

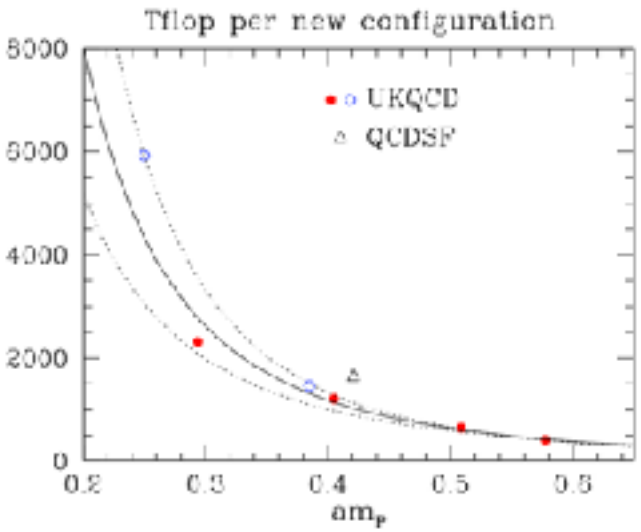
1. Generate an ensemble of gluon field configurations “gauge generation”
   - Hybrid Monte Carlo is the algorithm of choice
   - Produced in sequence, with hundreds needed per ensemble
   - Strong scaling required with 100-1000 TFLOPS sustained for several months
   - 50-90% of the runtime is in the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



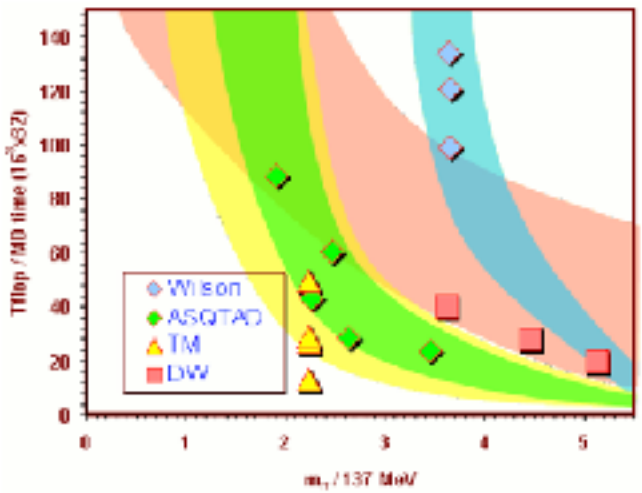
2. “Analyze” the configurations
   - Can be farmed out, assuming ~10 TFLOPS per job
   - Task parallelism means that clusters reign supreme here
   - 80-99% of the runtime is in the linear solver
     Many solves per system, e.g., O(10^6)



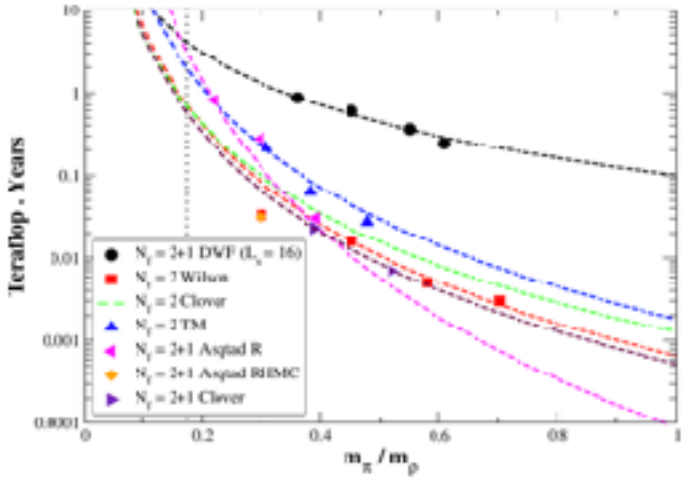
SCALING THE BERLIN WALL



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

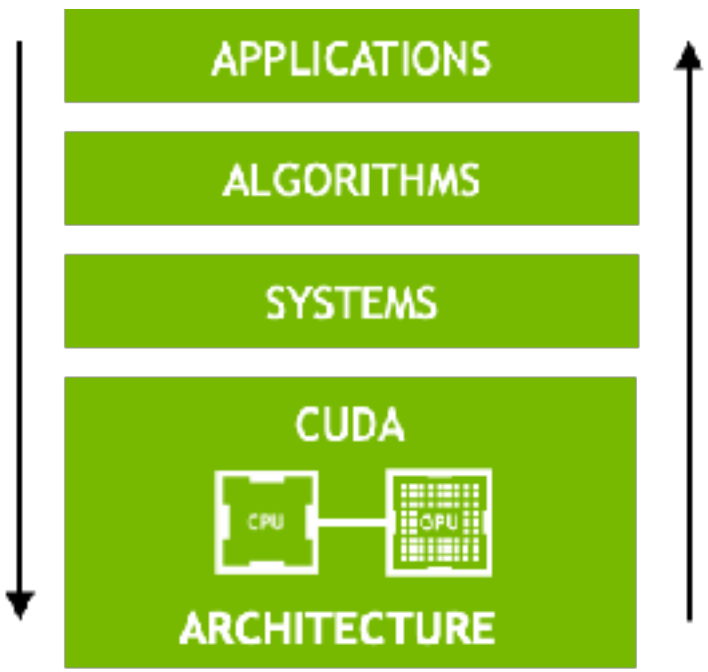
Autocorrelation length diverges as $a \rightarrow 0$

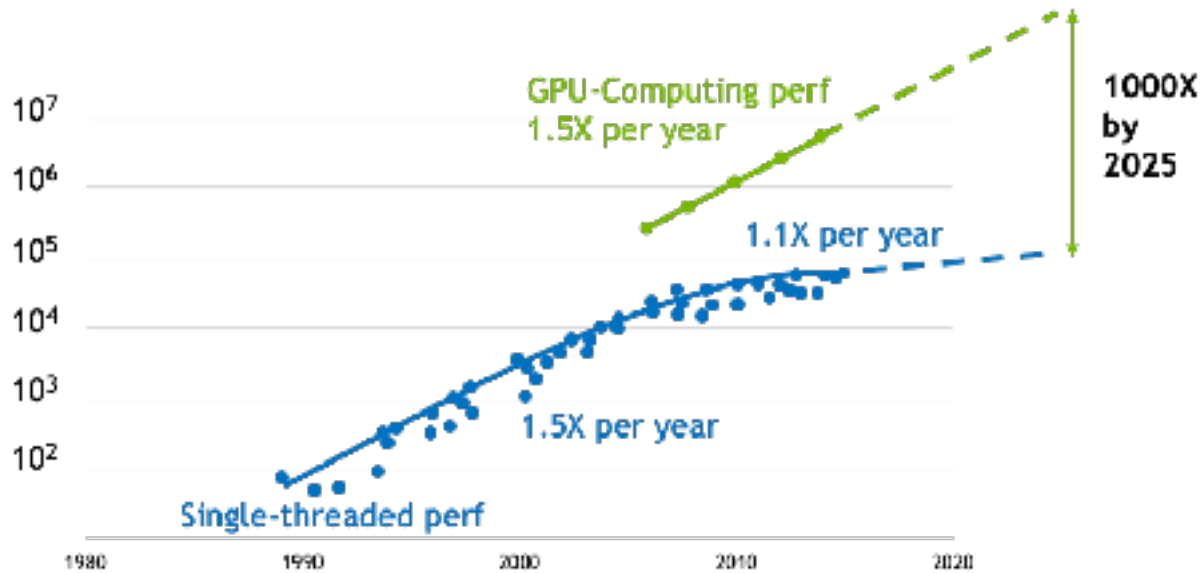
- Topology freezing…

Citations are illustrative,
not exhaustive


GPUS FOR LQCD

RISE OF GPU COMPUTING





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

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

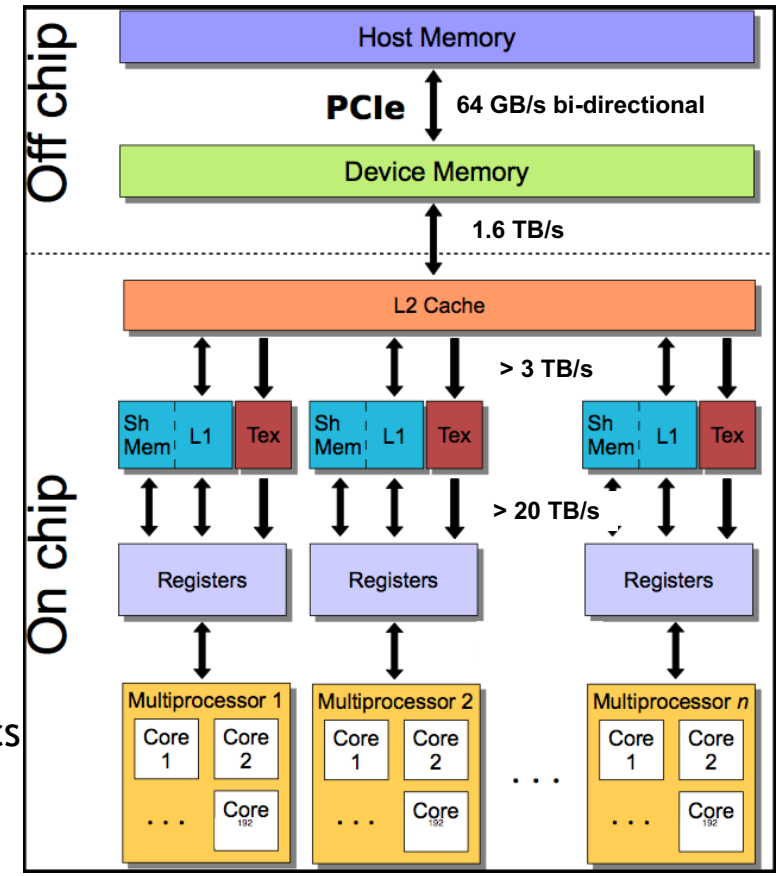
Increasingly coupled instruction hierarchy

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

Synchronization possible at many levels

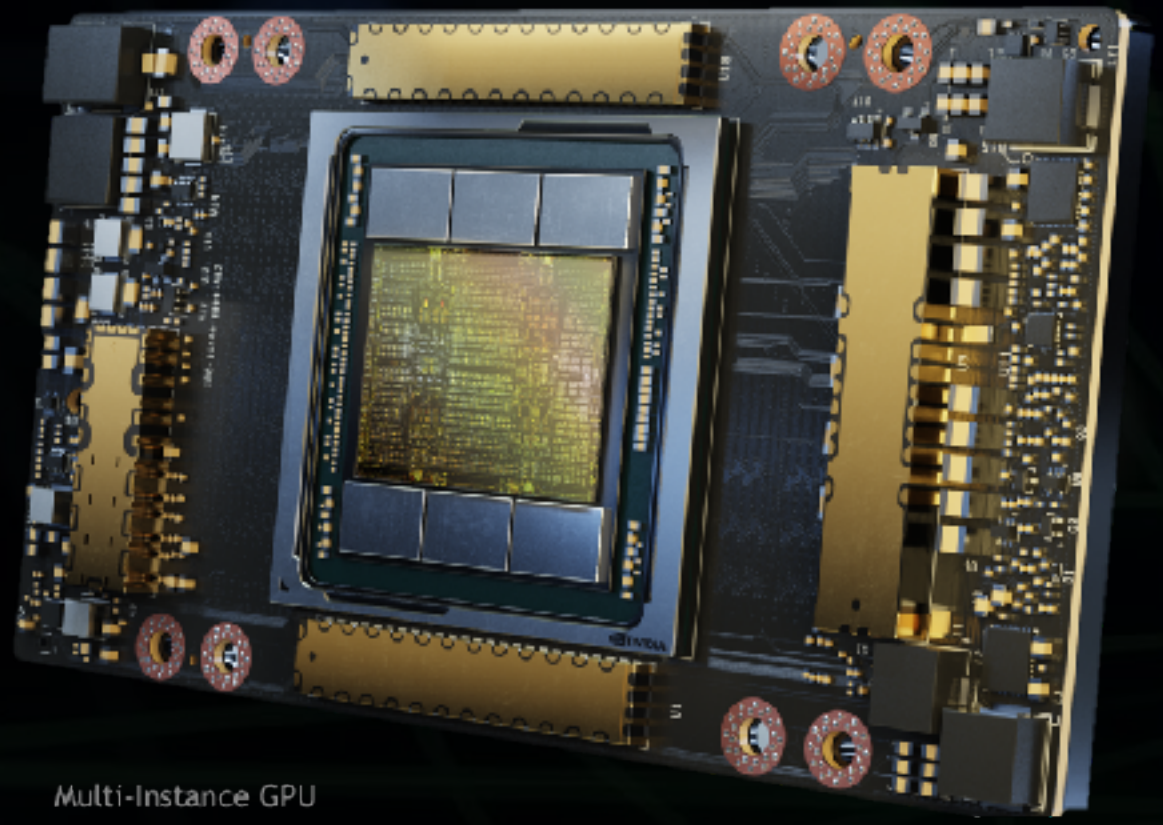
(Sub-)Warp <-> Thread Block <-> Grid <-> Node <-> Cluster

A100 sketch

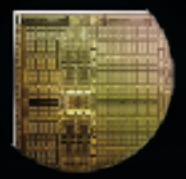


ANNOUNCING A100

Unprecedented Acceleration at Every Scale



Ampere Architecture

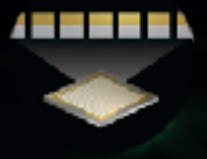


7nm Process
40GB

3rd Generation Tensor Cores

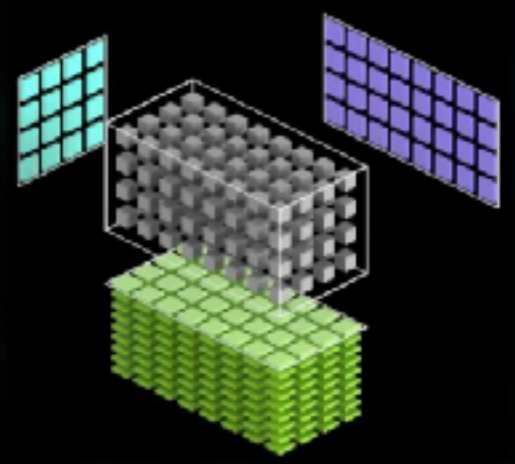
Performance Increase
TF32: 20X AI
FP64: 2.5X HPC

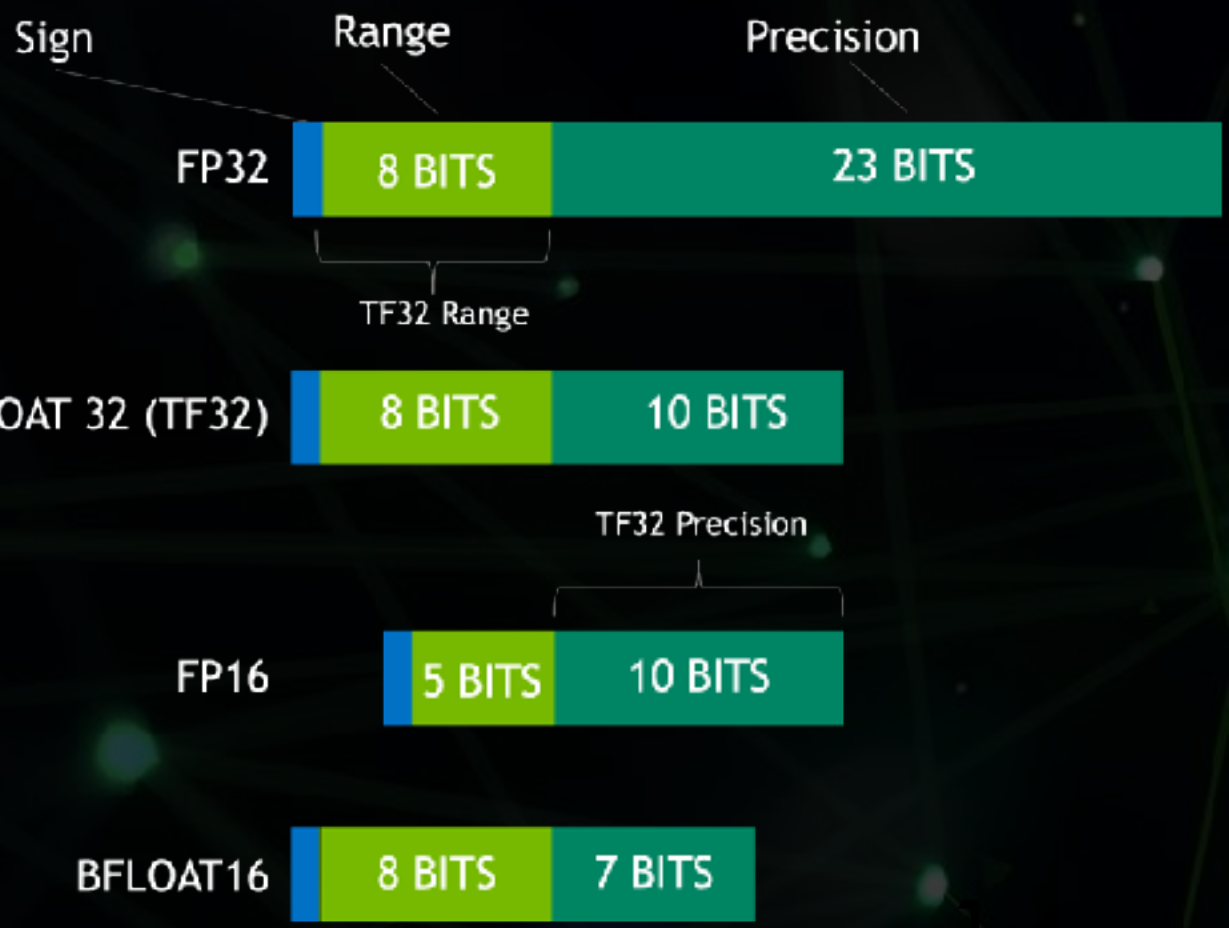
Multi-Instance GPU



Optimal Utilization
With Right Size GPU

AMPERE TENSOR CORES





- Wide variety of precisions and formats supported
  - 312 TFOPS: FP16, BFLOAT16
  - 156 TFOPS: TF32
  - 624 TOPS: INT8
  - 1248 TOPS: INT4
  - 19.5 TFLOPS: FP64

**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
  - Domain-decomposed (Schwarz) preconditioners for strong scaling
  - Multigrid solvers for optimal convergence
  - NVSHMEM for improving strong scaling
- **A research tool for how to reach the exascale (and beyond)**
  - Optimally mapping the problem to hierarchical processors and node topologies

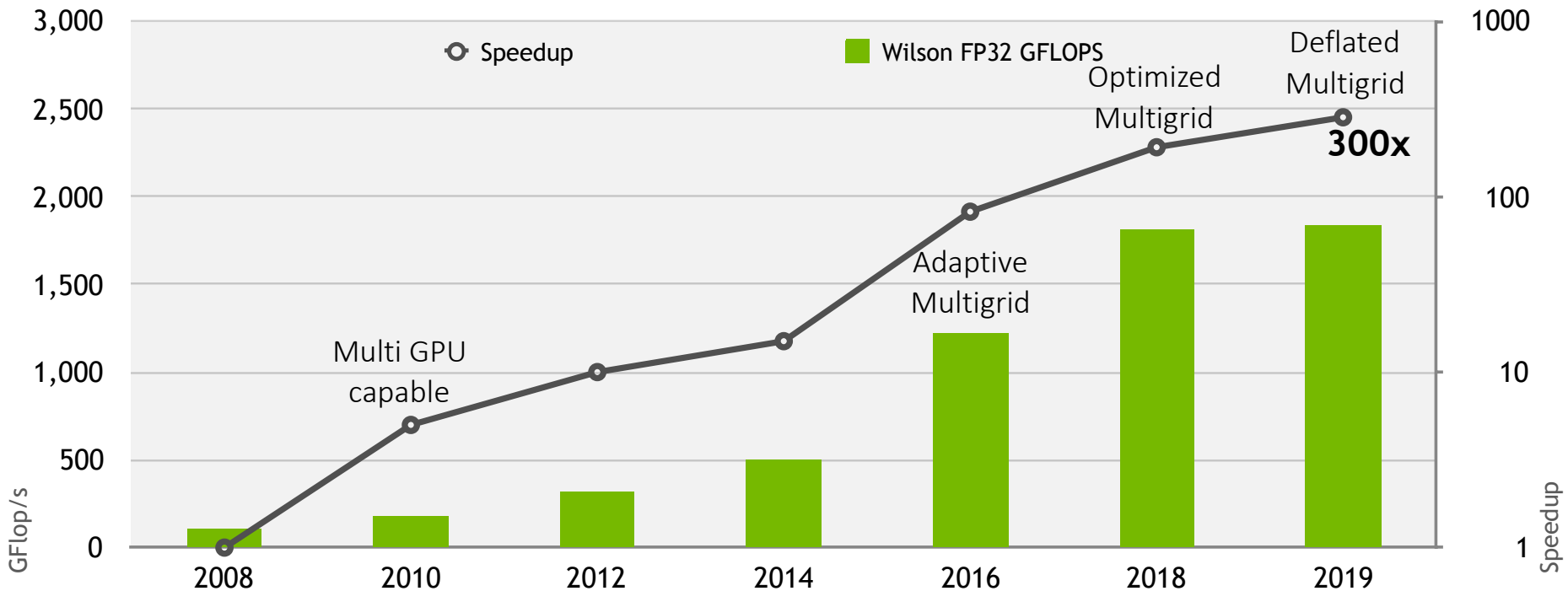
QUDA CONTRIBUTORS

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)
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 (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

Finite difference operator in LQCD is known as Dslash

Assign a single space-time point to each thread

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

Looping over direction each thread must

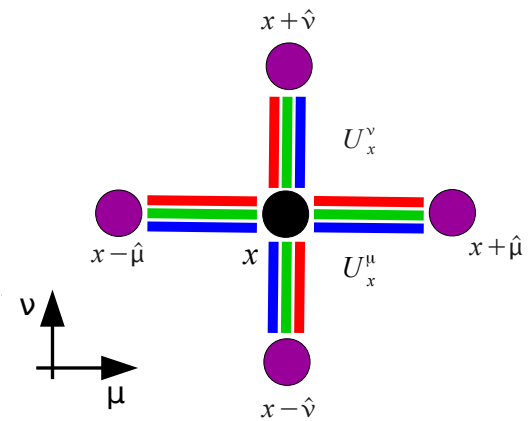
- Load the neighboring spinor (24 numbers x8)
- Load the color matrix connecting the sites (18 numbers x8)
- Do the computation
- Save the result (24 numbers)

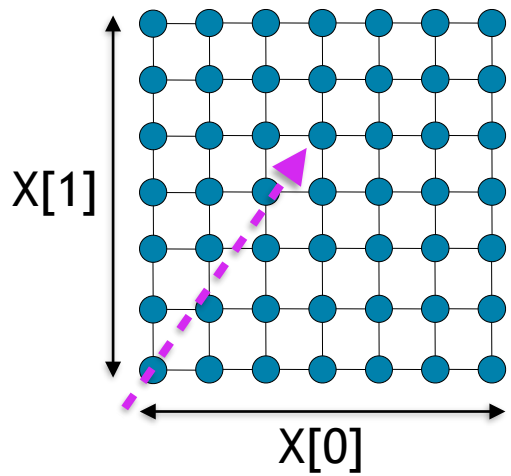
Each thread has (Wilson Dslash) 0.92 naive arithmetic intensity

QUDA reduces memory traffic

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

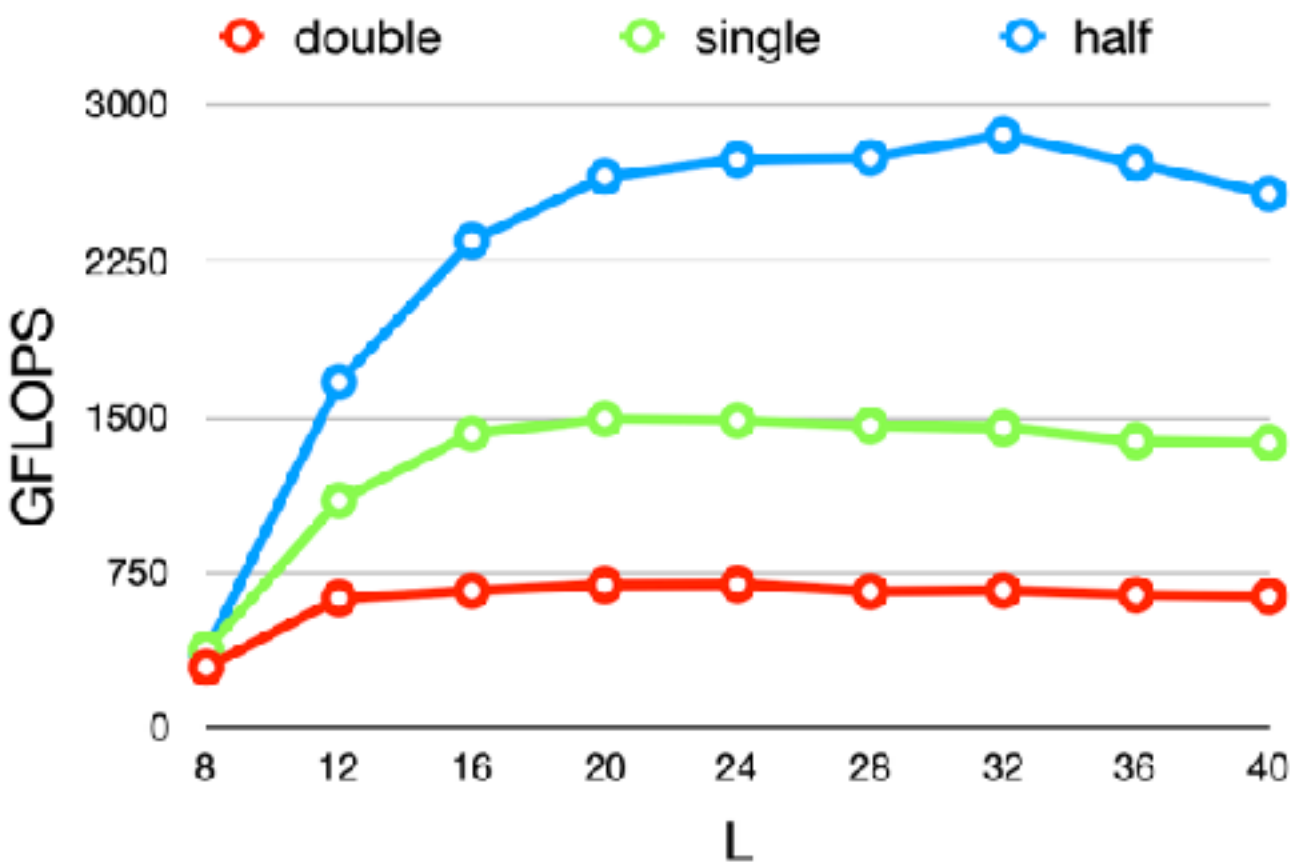
$$D_{x,x'} =$$





SINGLE GPU PERFORMANCE

“Wilson-clover” stencil (Chroma, V100)



~1325 GB/s

~1275 GB/s

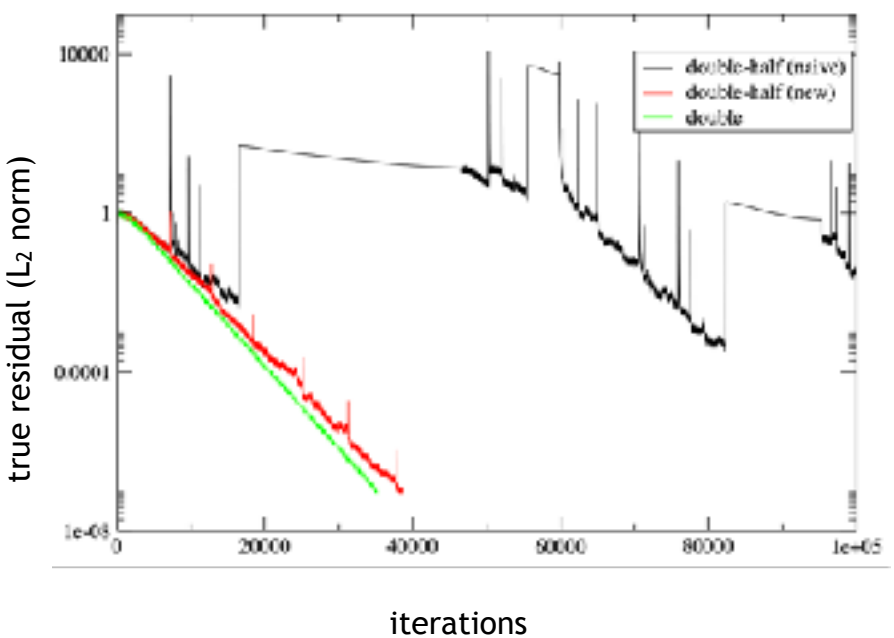
~1200 GB/s

Tesla V100,
CUDA 10.1,
GCC 7.3,
QUDA 1.0

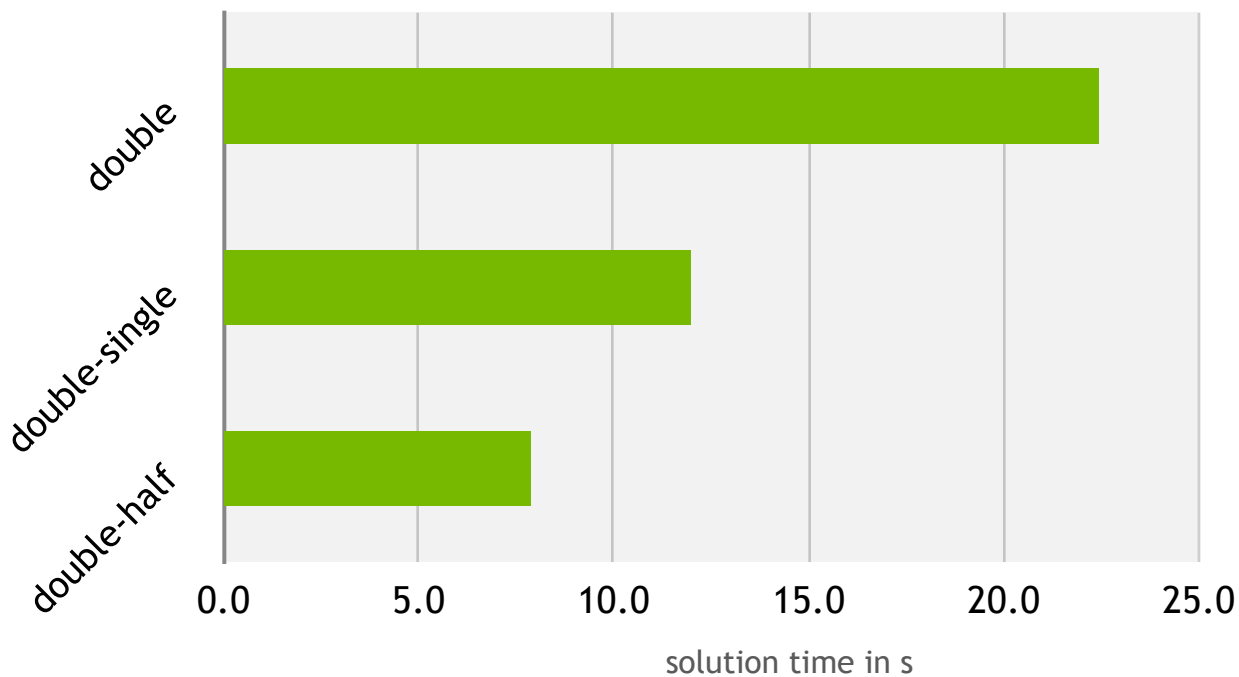
MIXED PRECISION

Using your bits wisely

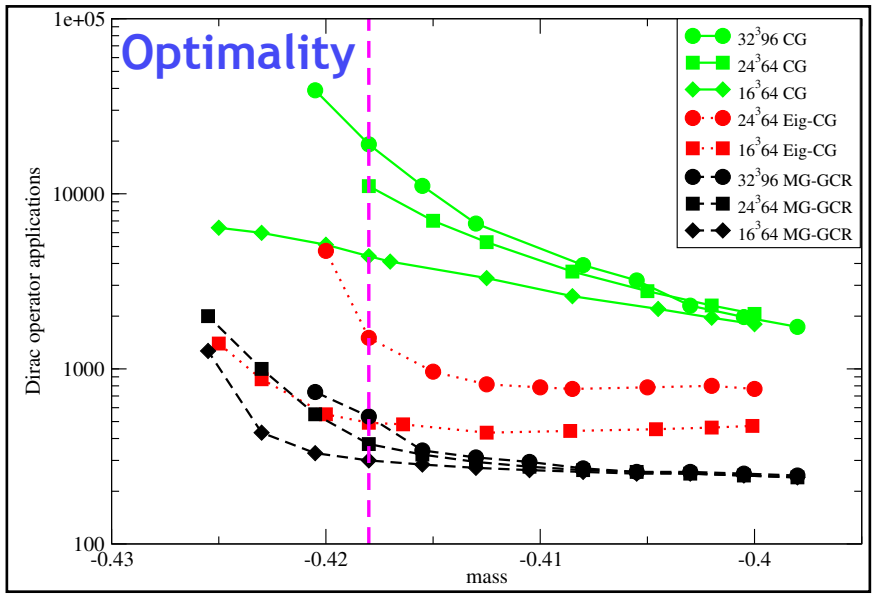
MILC/QUDA HISQ CG, mass = 0.001 => κ~10^6



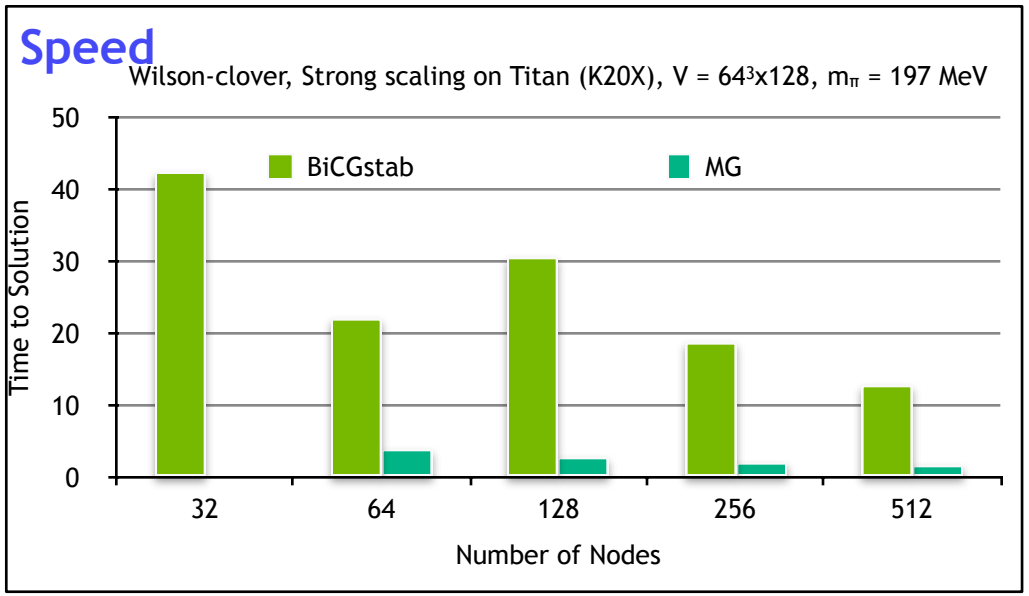
MILC/QUDA HISQ CG solver



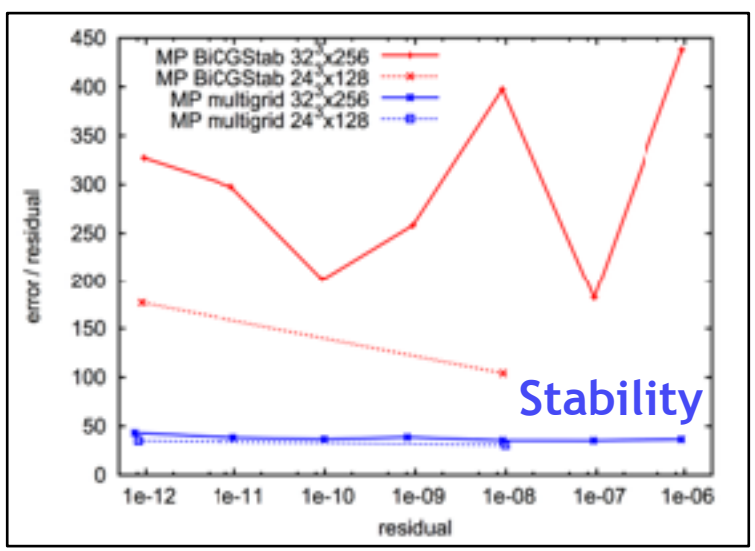
WHY MULTIGRID?



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

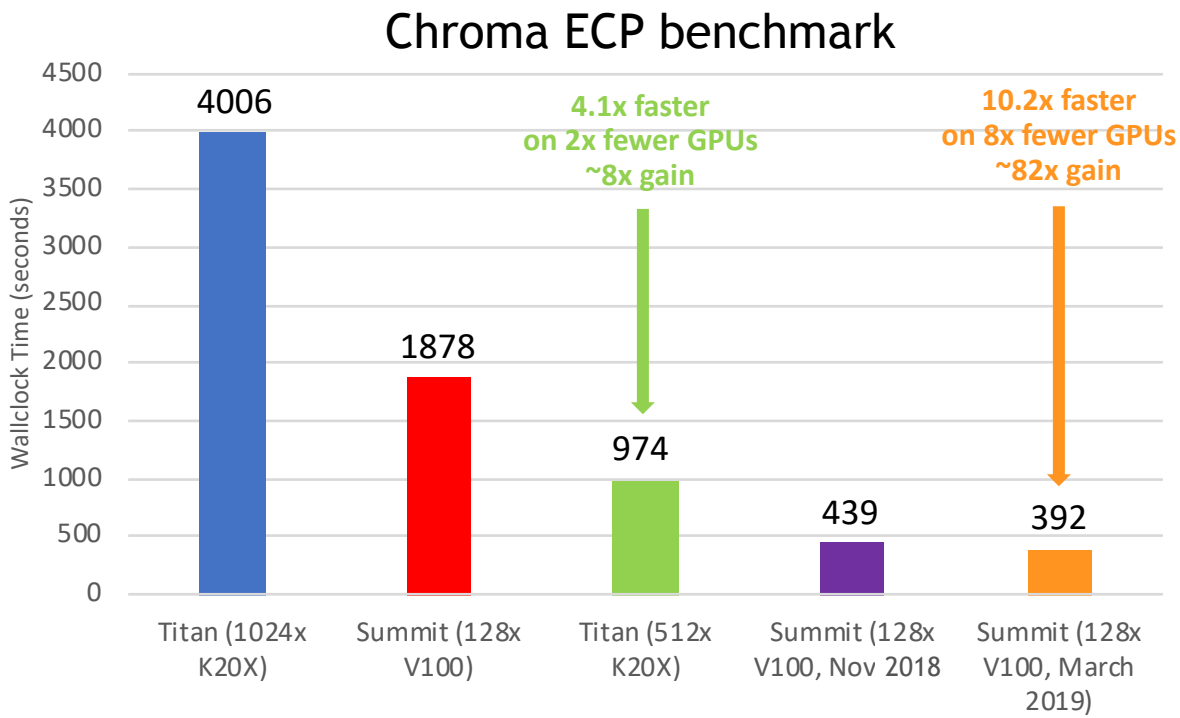
From Titan running 2016 code to Summit running 2019 code we see >82x speedup in HMC throughput

Multiplicative speedup coming from 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



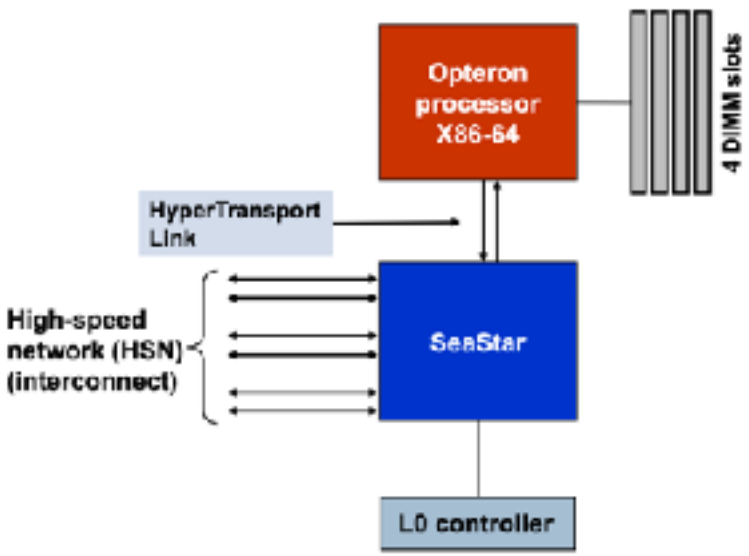

SCALING CHALLENGES

HPC IS GETTING MORE HIERARCHICAL

What does a node even mean?

Cray XT4 (2007)

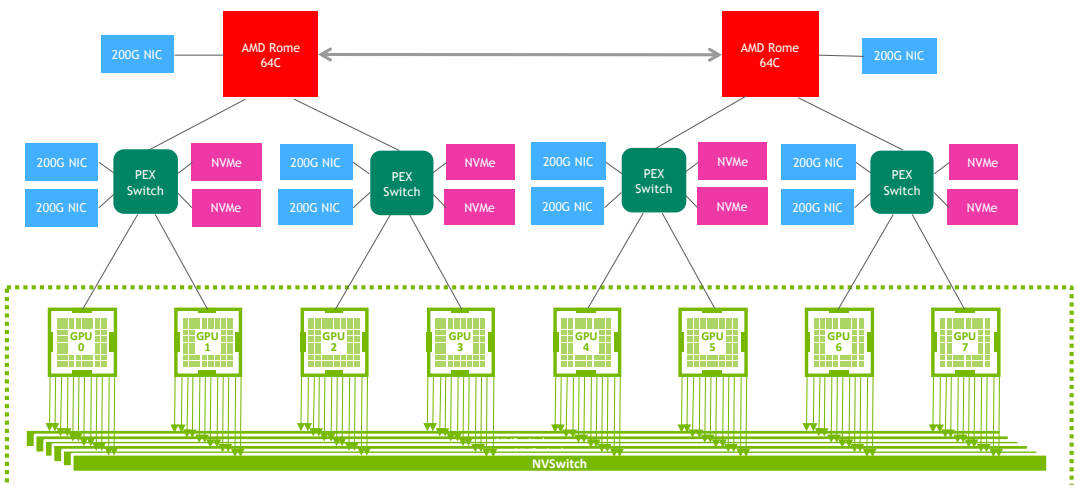
https://www.nersc.gov/assets/NUG-Meetings/NERSCSystemOverview.pdf



Legacy

NVIDIA DGX-A100 (2020)

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



Current

MULTI-NODE SCALABILITY

GPU clusters have reputation for lack of scalability

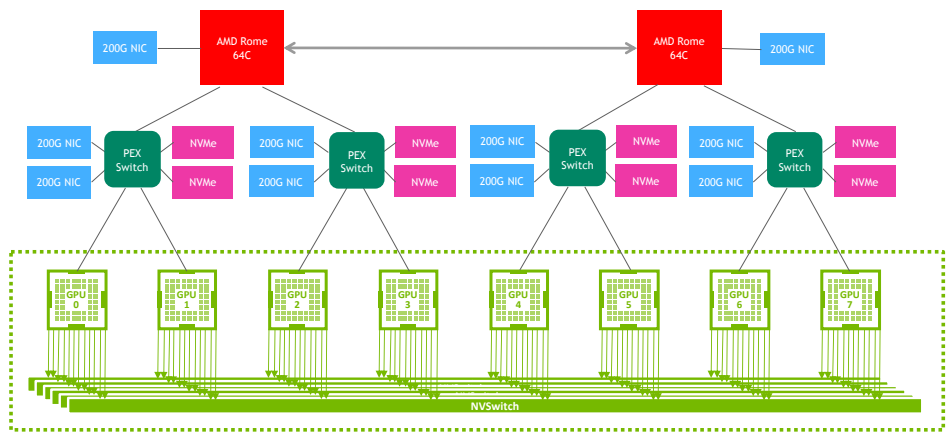
Summit vs BlueGene Q

Supercomputers are built for a broad range of applications, of which LQCD is an outlier

But is there more to it than simply NIC bandwidth?

What happens when we build a balanced machine?

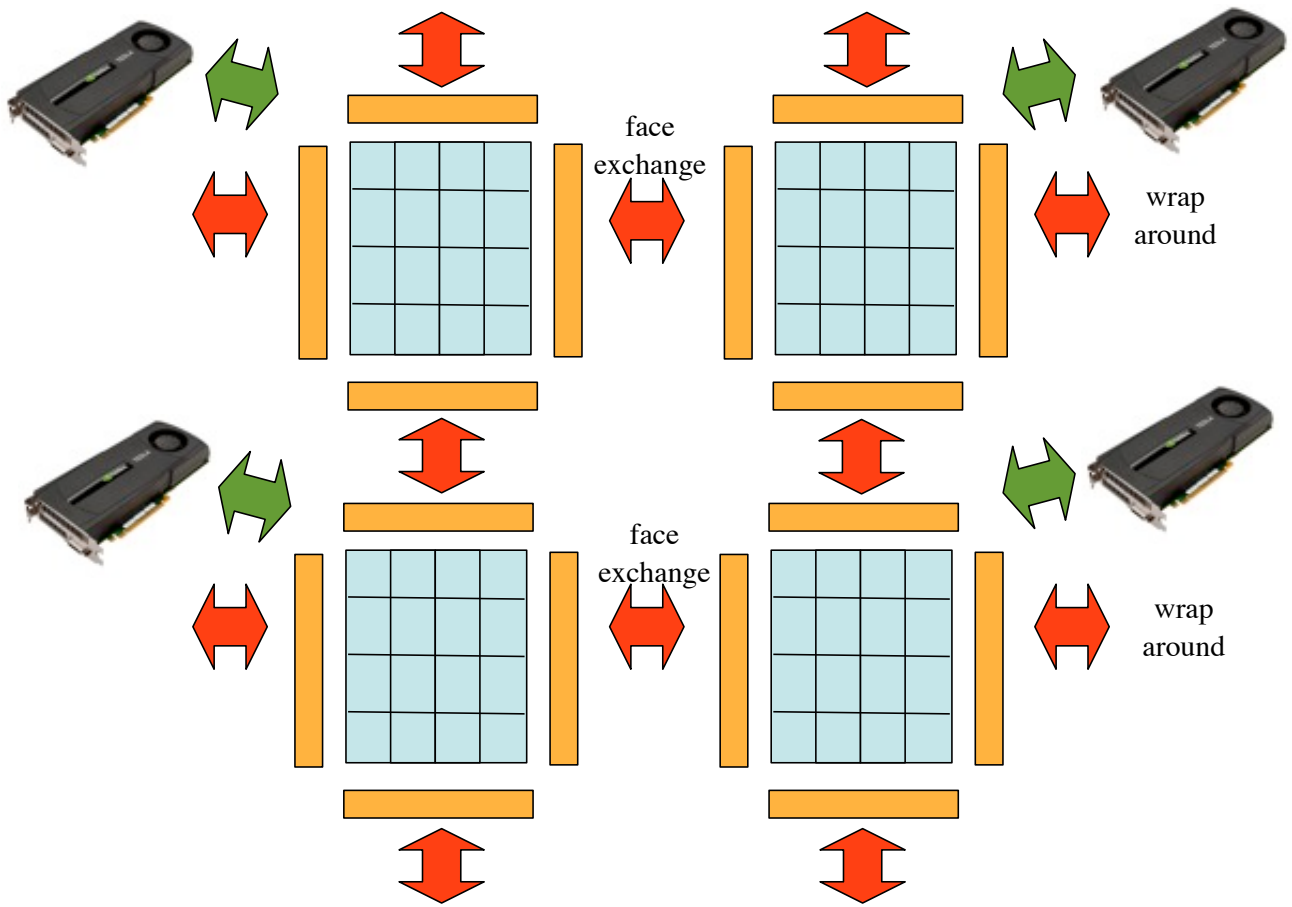
e.g., Selene, Juelich booster



Selene = 560 x DGX A100

640 Gb/s NIC / GPU 4-d neighbor bi-dir bandwidth

MULTI-GPU BUILDING BLOCKS



Halo packing Kernel

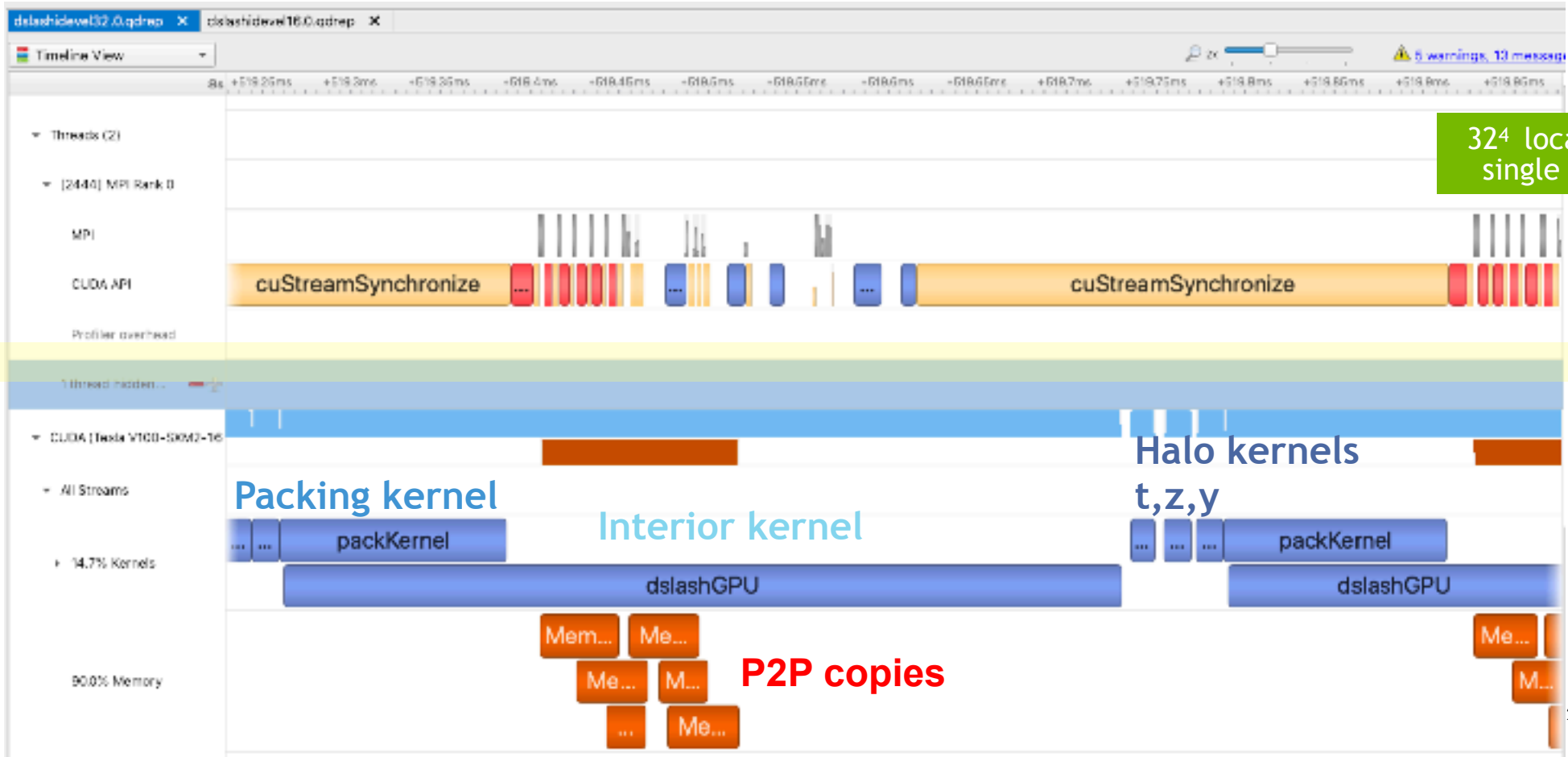
Interior Kernel

Halo communication

Halo update Kernel

MULTI-GPU PROFILE

overlapping comms and compute



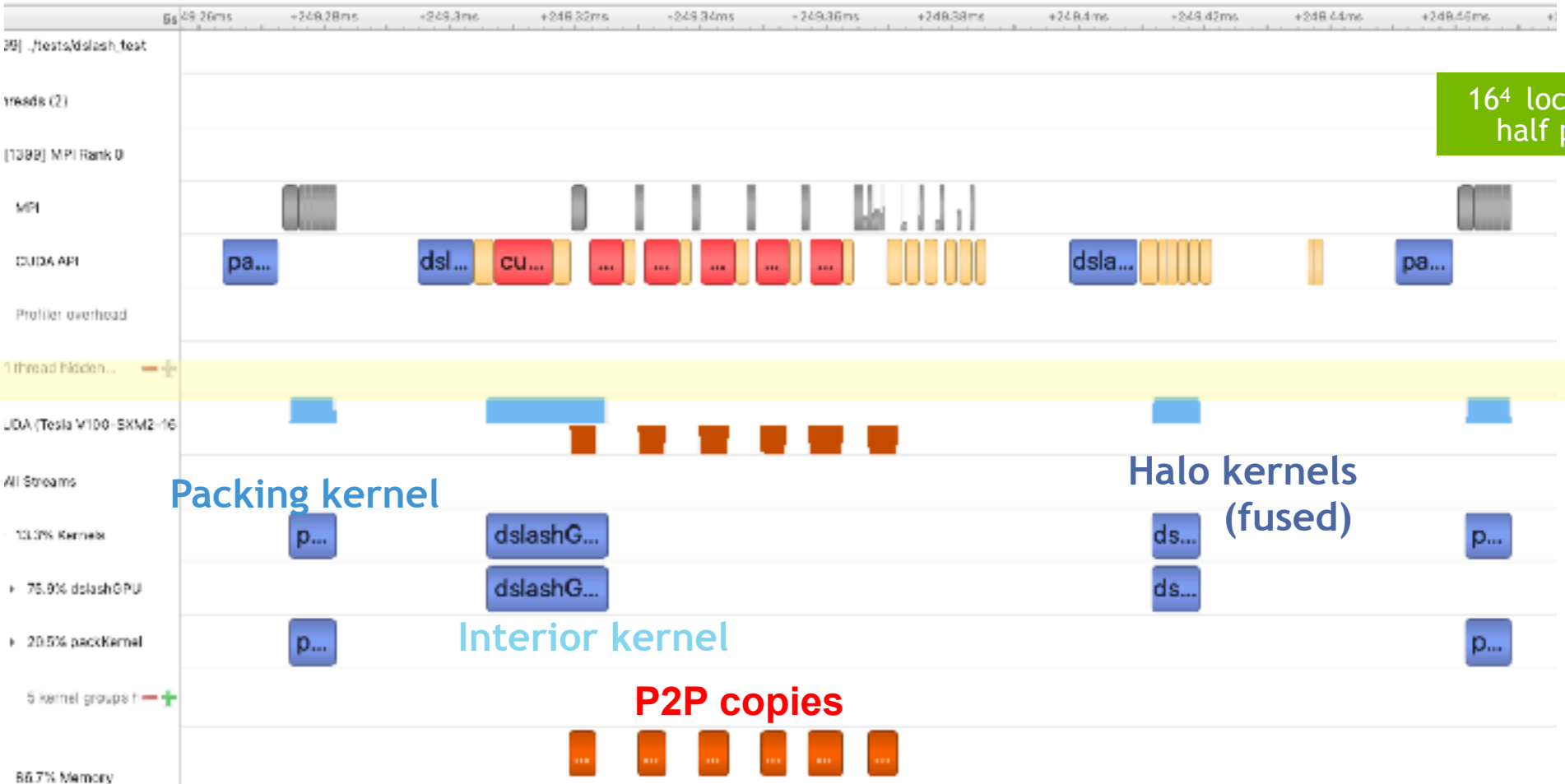
32^4 local volume, single precision

DGX-1, 1x2x2x2 partitioning

STRONG SCALING PROFILE

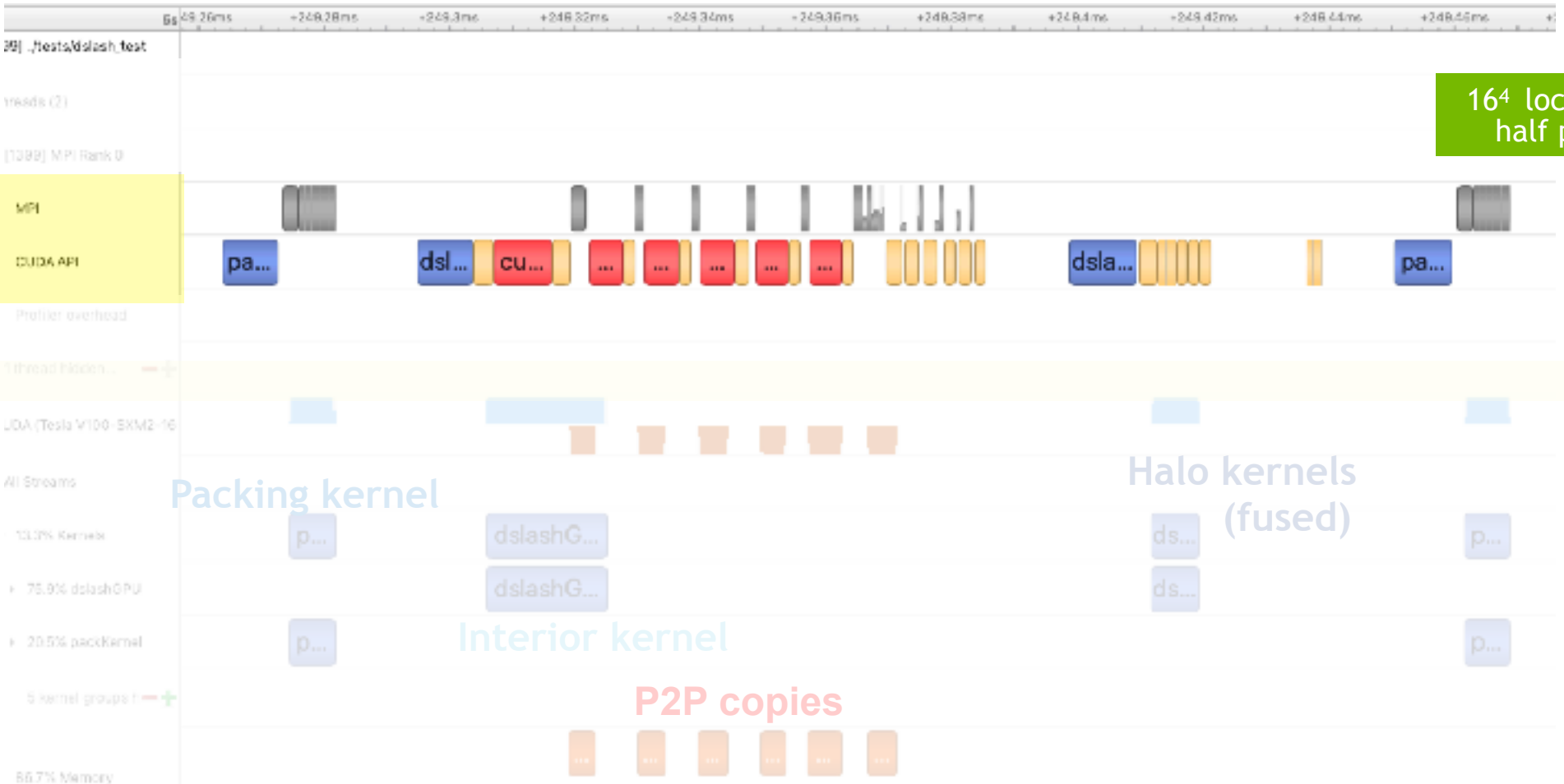
~~overlapping~~ comms and compute

DGX-1, 1x2x2x2 partitioning



STRONG SCALING PROFILE

Latencies ate my scaling



API Calls

16^4 local volume, half precision

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

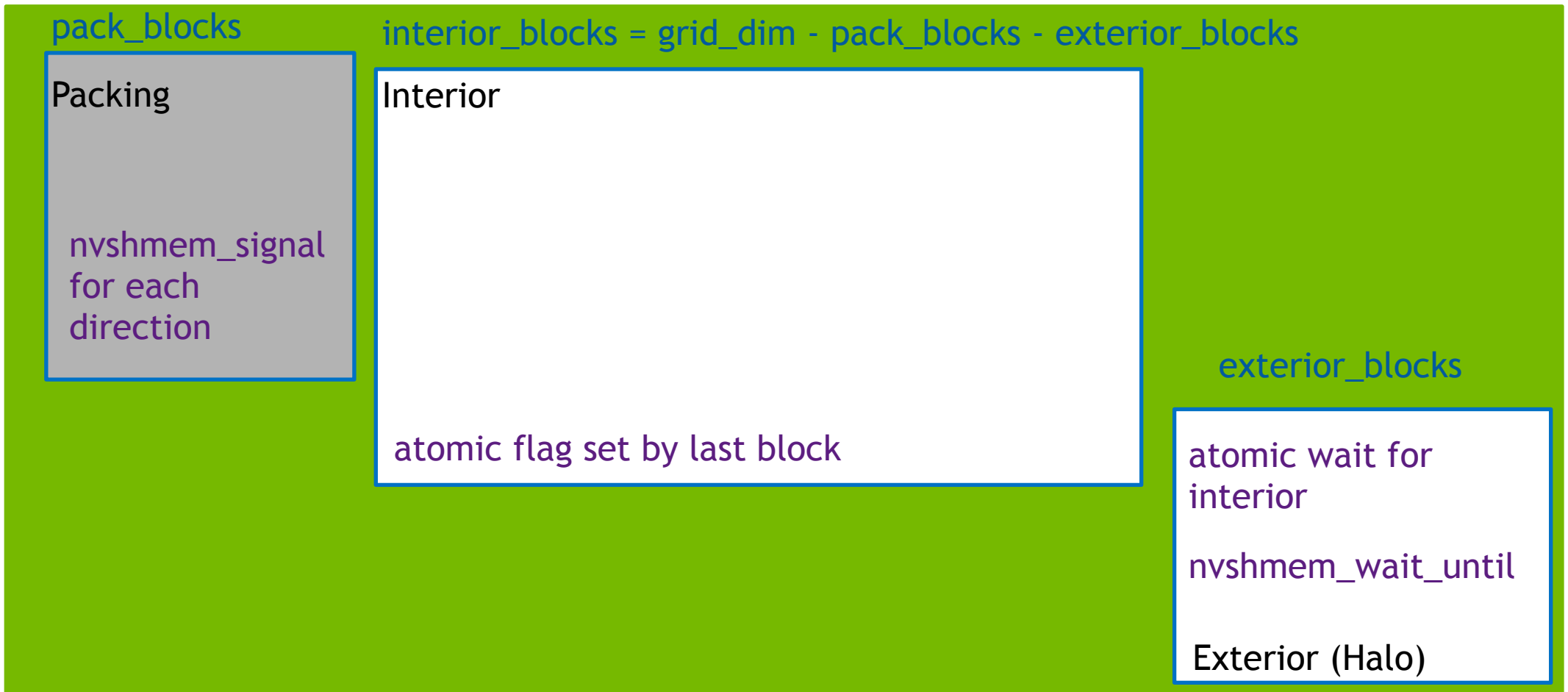
NVLink and PCIe support (intra-node)

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

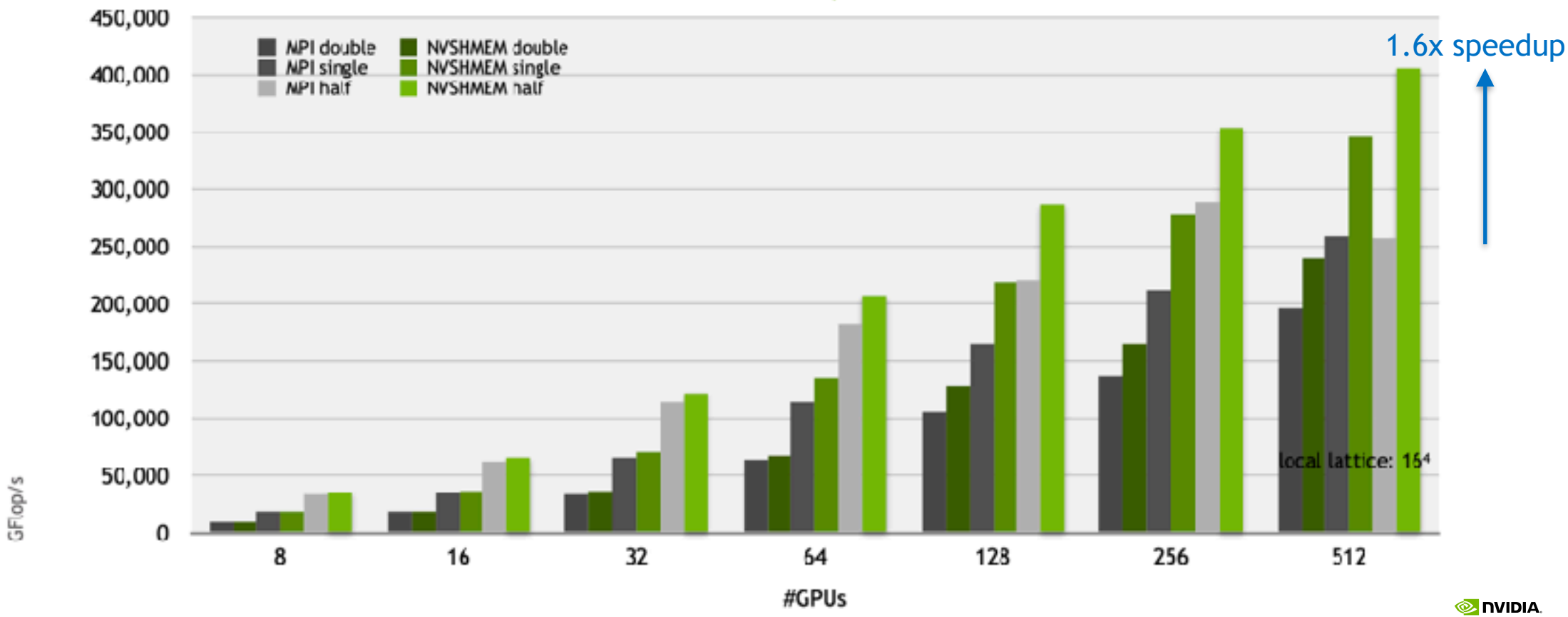
16^4 local volume, half precision

DGX-1, 1x2x2x2 partitioning

Mathias Wagner

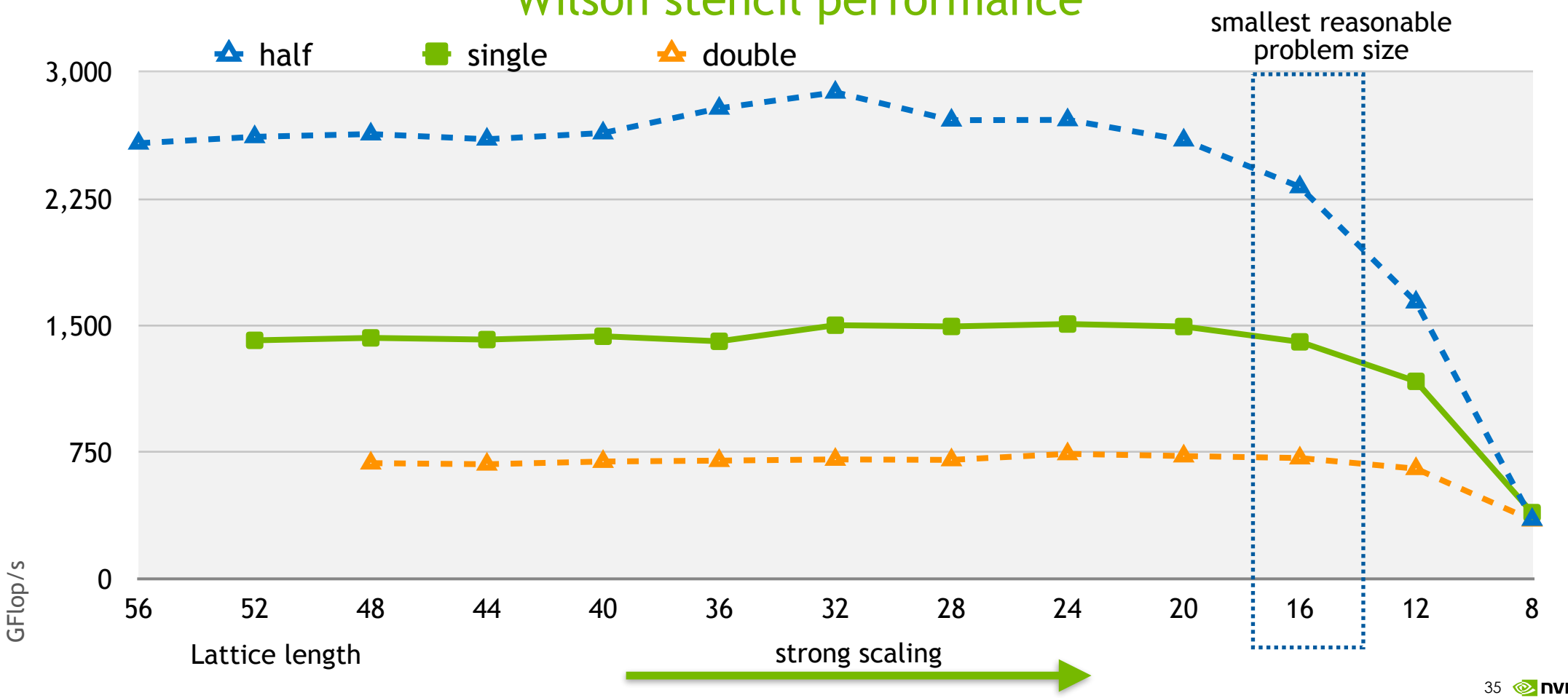
SELENE STRONG SCALING

Global volume 64^3x128

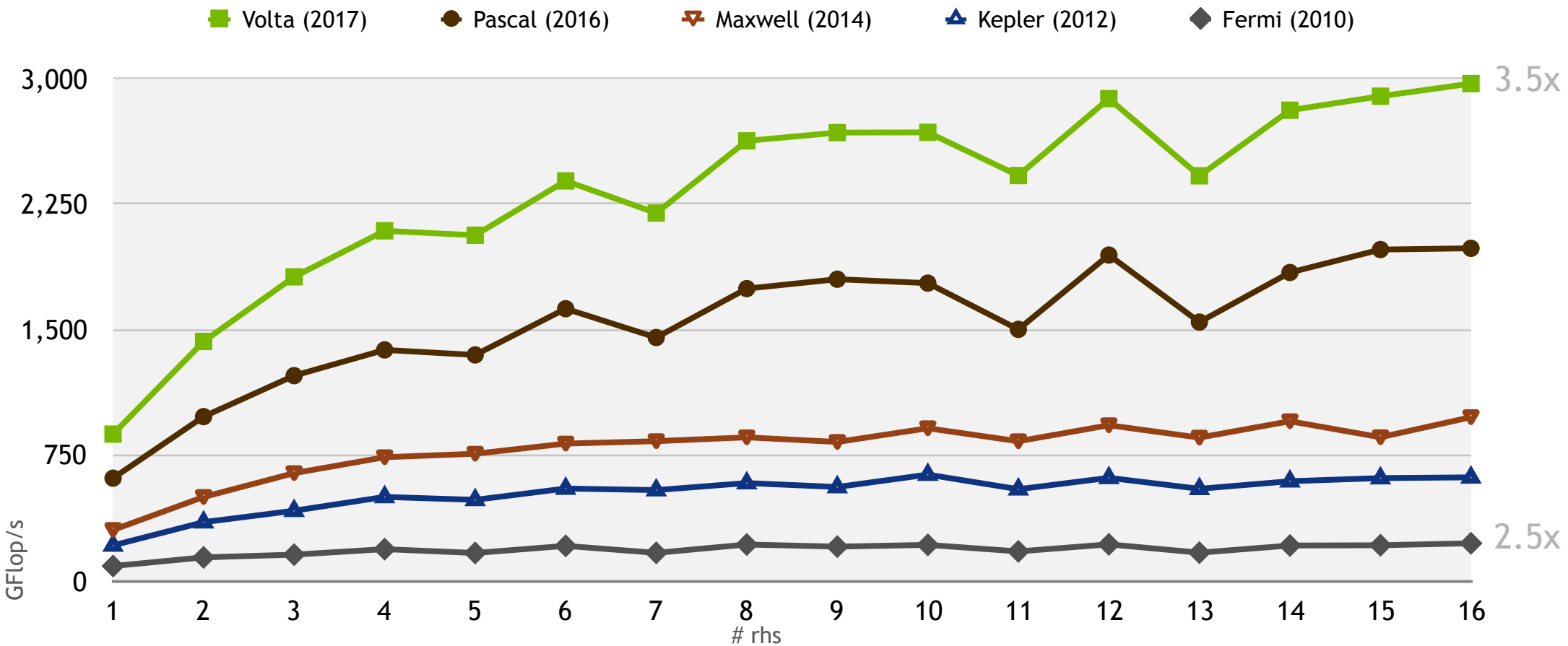


PARALLELISM ISN'T INFINITE...

Wilson stencil performance

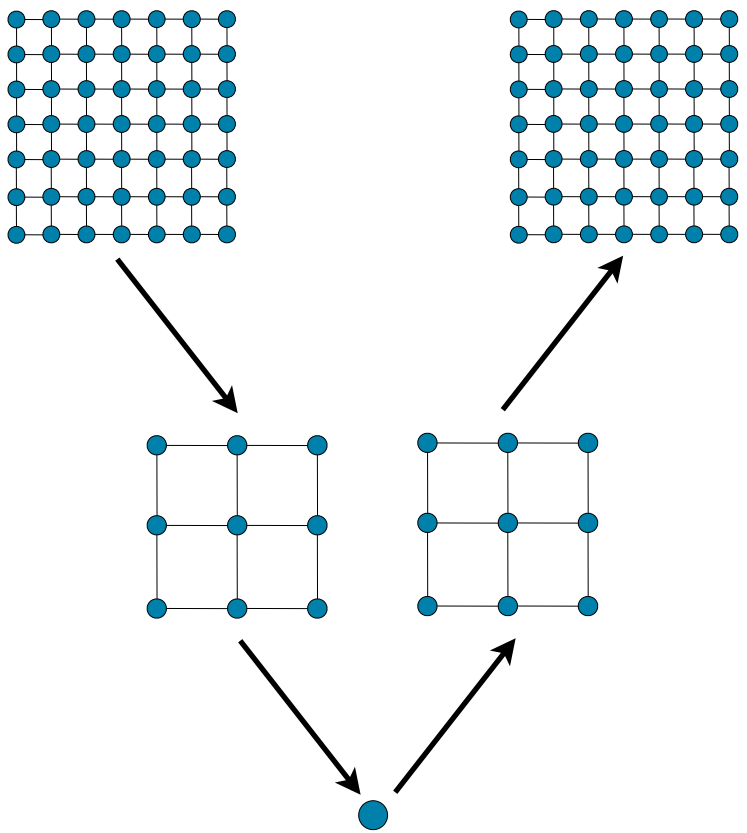


…UNLESS WE MAKE IT



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

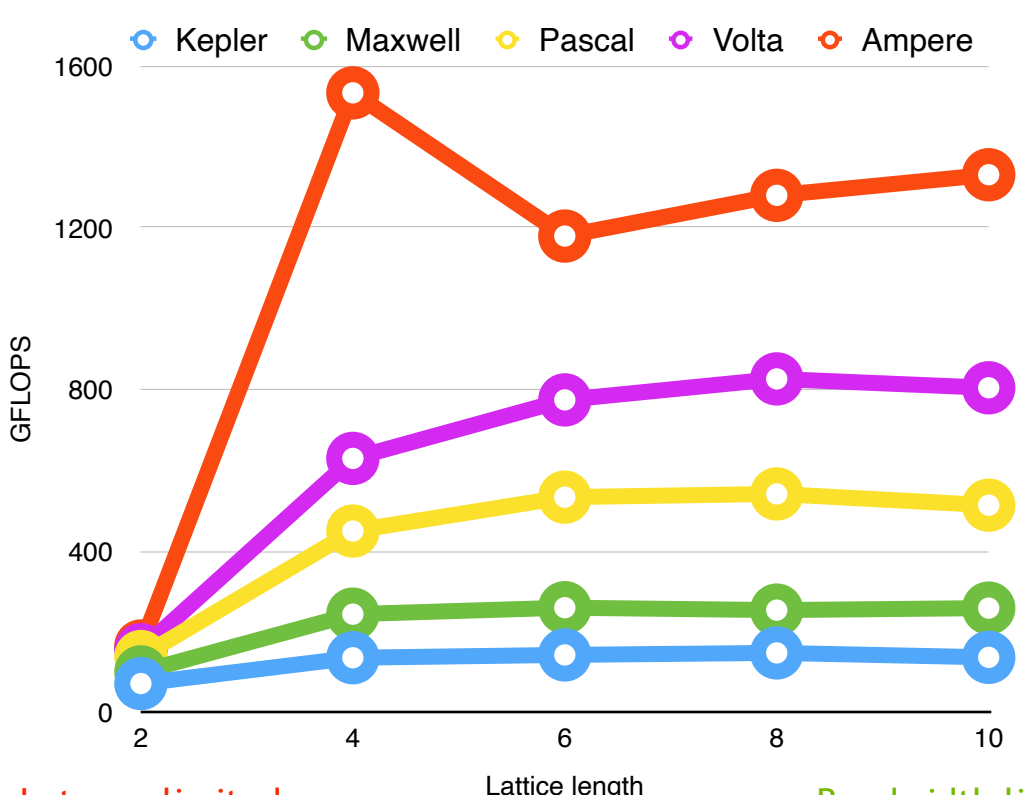
- High parallel throughput problem

Coarse grids are worst possible scenario

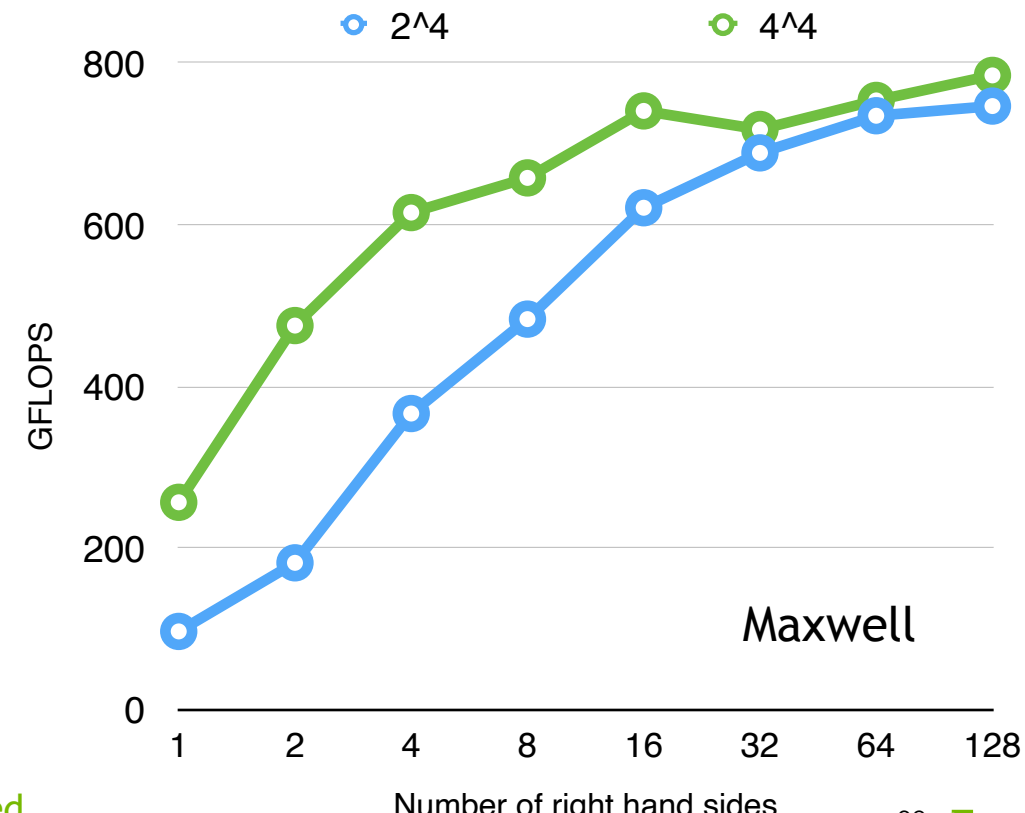
- More cores than degrees of freedom
- Increasingly serial and latency bound
- Little's law (bytes = bandwidth * latency)
- Amdahl's law limiter

MULTIGRID

Gets harder with every generation



Latency limited

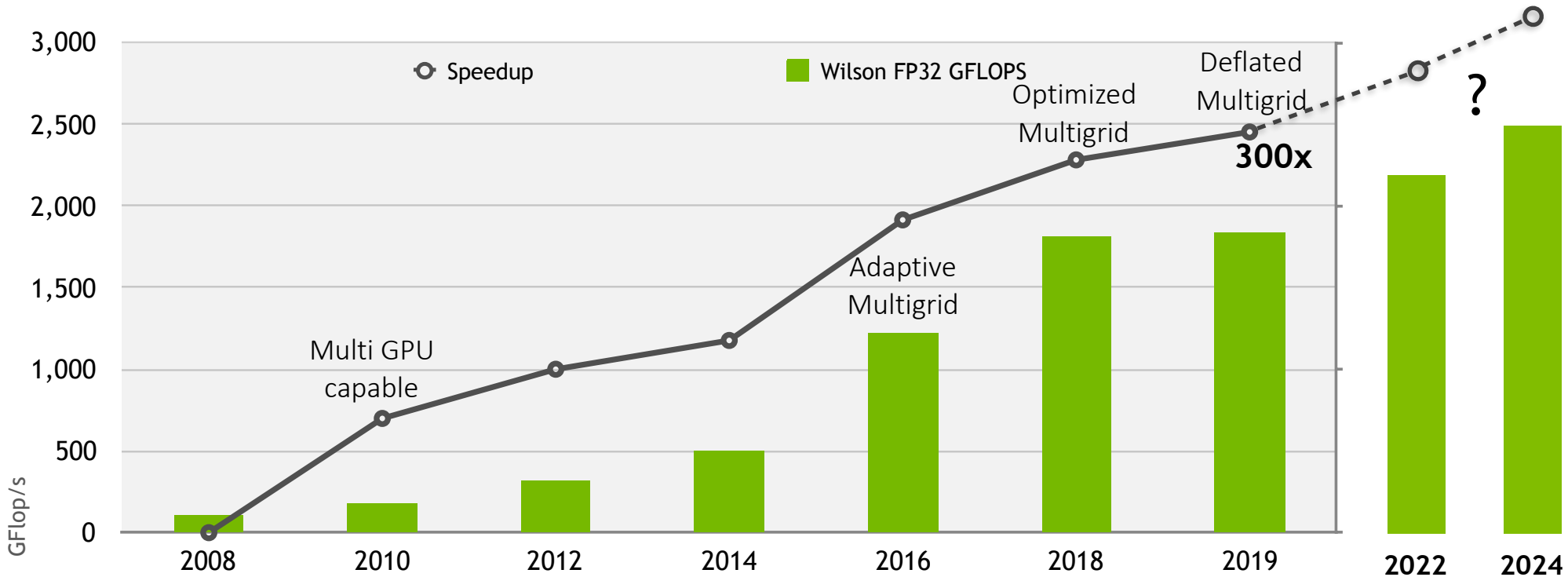


Bandwidth limited


FUTURE CHALLENGES

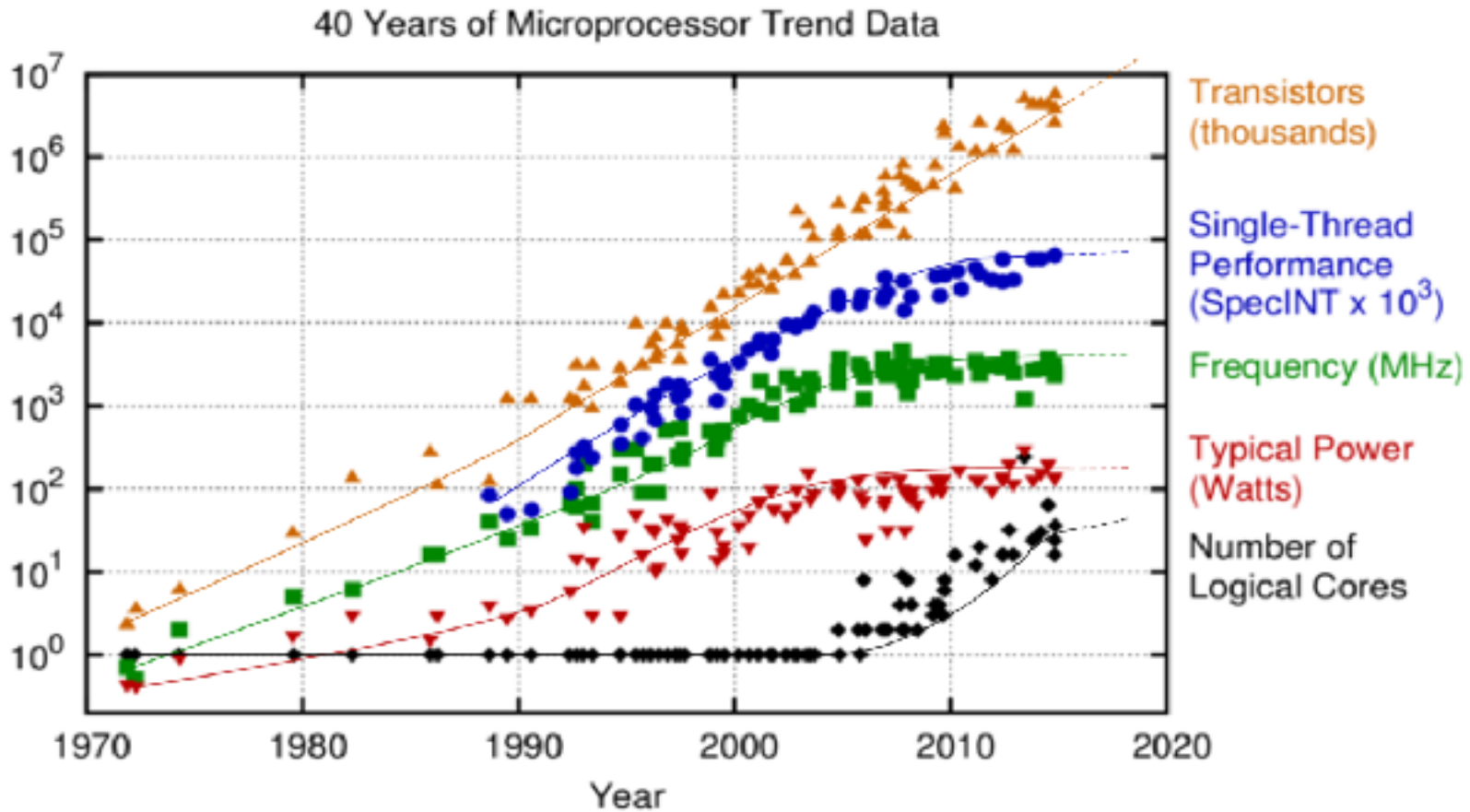
QUDA NODE PERFORMANCE OVER TIME

Multiplicative speedup through software and hardware



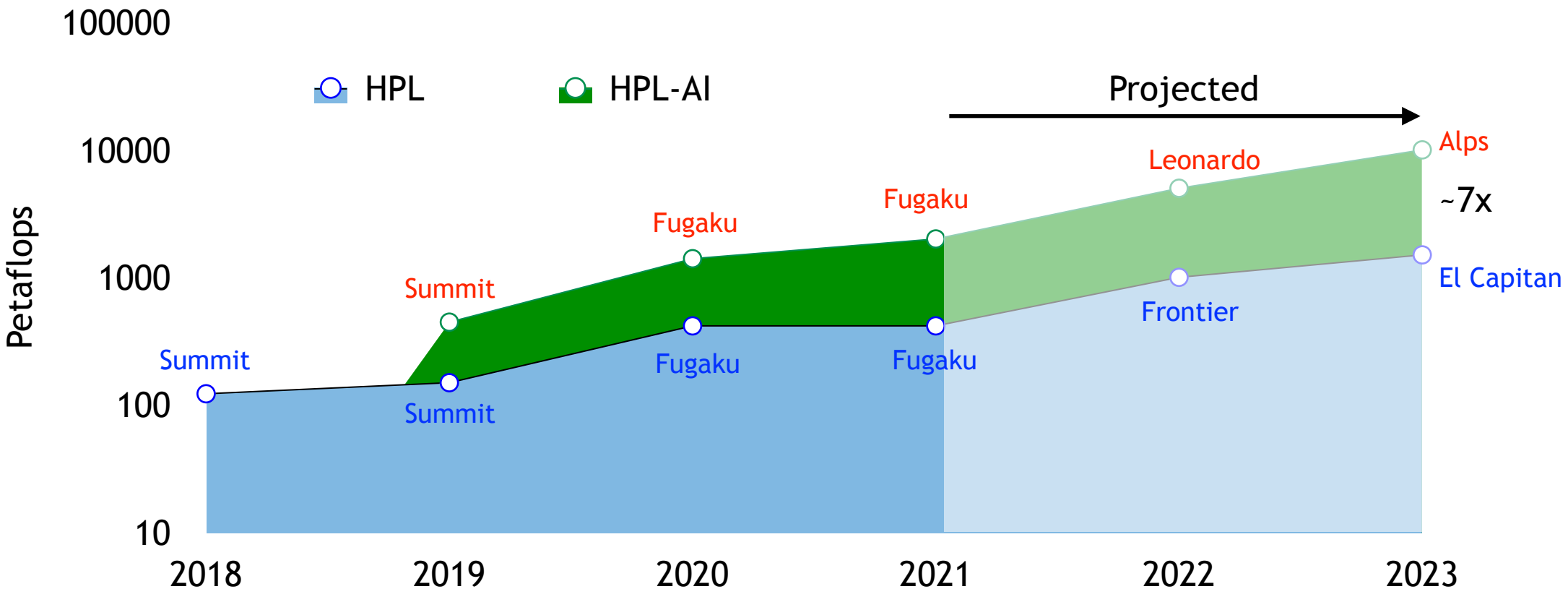
Speedup determined by measured time to solution for solving the Wilson operator against a random source on a $V=24^3 64$ lattice, $\beta=5.5$, $M_\pi = 416$ MeV. One node is defined to be 3 GPUs

EXASCALE IS ALMOST HERE
BUT WILL WE GET TO 10 EXAFLOPS?



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

WE WILL WITH AI



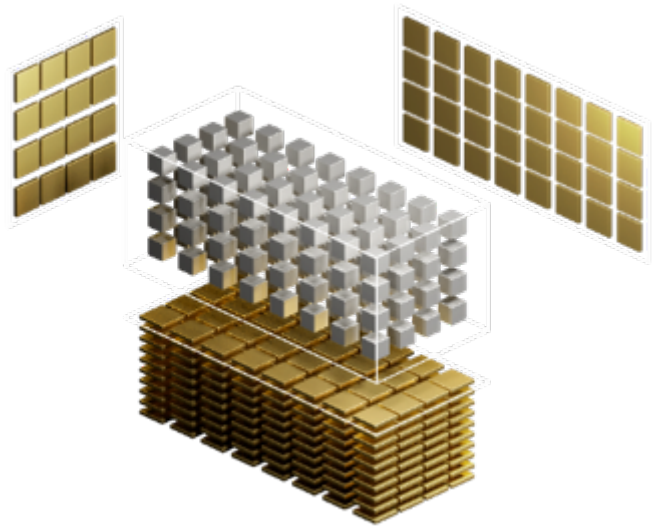
FUTURE MACHINES WILL BE CHALLENGING

Matrix and tensor operations required to saturate the machine

Low precision will go much faster

Extreme parallelism required

Hierarchy and Locality must be considered



Rework the pipeline to expose algorithms in matrix-matrix form

Maximize locality, parallelism for optimal mapping onto the GPU hierarchy

TENSOR CORES FOR LQCD

Initial first steps

Multigrid is a preconditioner

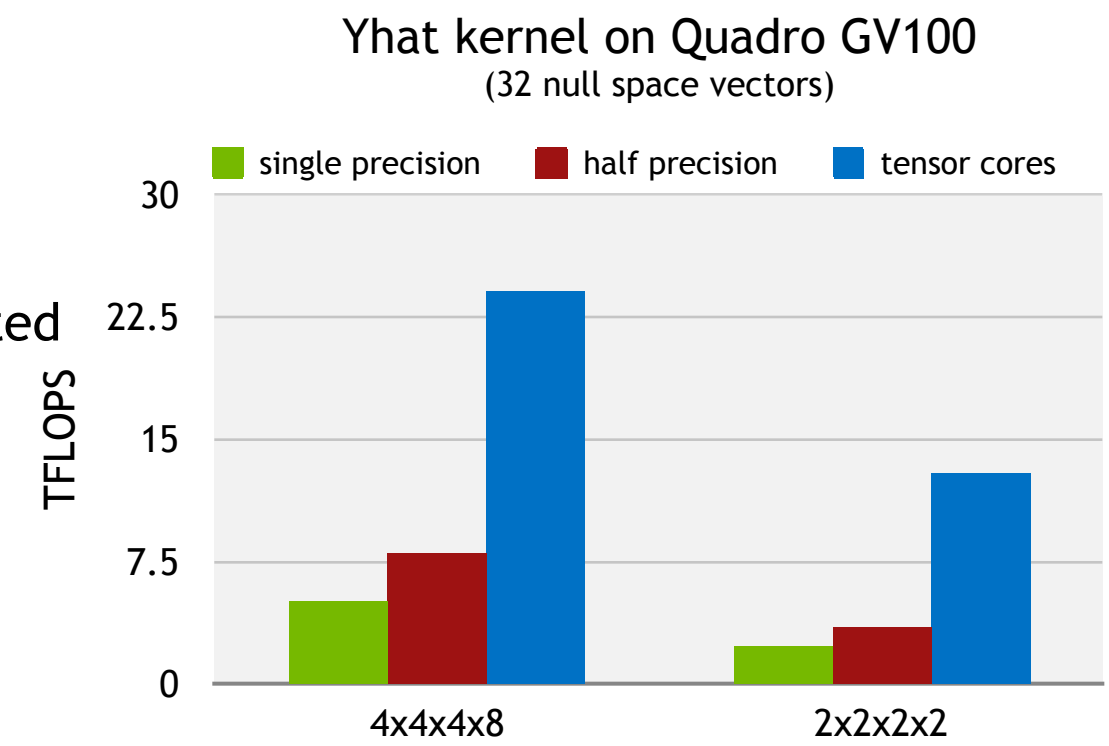
- 16-bit precision is perfectly adequate
- No impact on convergence rate

Majority of MG setup kernels now implemented

- 1.5x-10x kernel speedups observed

Next steps

- multi-RHS solver
- Eigen-vector orthogonalization



AI ~ flops / bytes

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 ~ $(N_\varphi \times N_\psi)/(N_\varphi + N_\psi)$
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

LQCD OUTLOOK

Super Linear Scaling

Reformulate vector-vector and matrix-vector problems as matrix-matrix

- Multi-RHS, Communication-Avoiding, block solvers
- Deploy using tensor cores where possible

Keep data on chip and minimize hierarchy level jumping

- Use NVSHMEM to fuse across communication boundaries

Follow trends towards future architectures and aim for *super-linear* scaling

WHAT COULD LQCD DO WITH 100X MORE?

Getting nowhere even faster?

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

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

Or solve it with an evolved approach (sMD, Fourier acceleration, etc.)

Or do we need a *completely different* approach…

That is a fundamental revolution in solving Lattice Field Theory?

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


NVIDIA