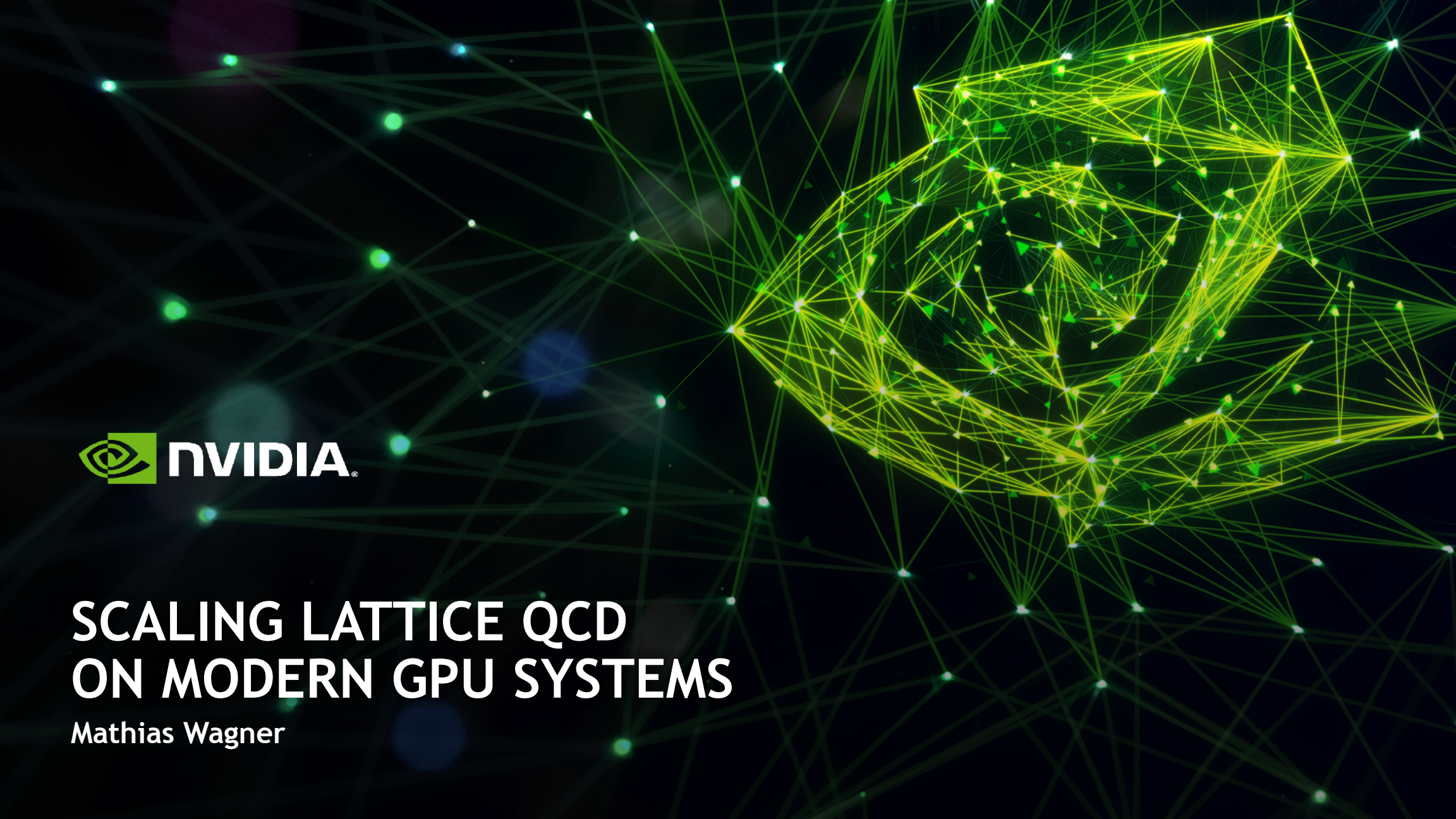


SCALING LATTICE QCD ON MODERN GPU SYSTEMS

Mathias Wagner

AGENDA

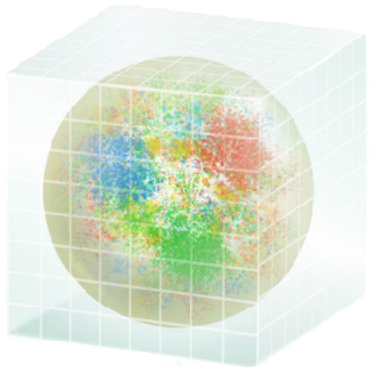
Lattice Quantum Chromodynamics

GPU refresh

- lower precision
- multigrid
- more nodes

Summary

LATTICE QUANTUM CHROMODYNAMICS



Theory is highly non-linear ⇒ cannot solve directly

Must resort to numerical methods to make predictions

Lattice QCD

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

Finite spacetime ⇒ periodic boundary conditions

PDEs ⇒ finite difference equations

Consumer of 10-20% of public supercomputer cycles

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

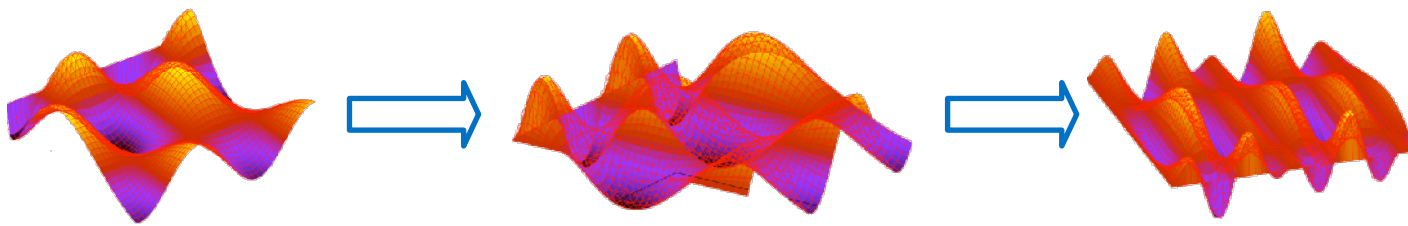
STEPS IN AN LQCD CALCULATION

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

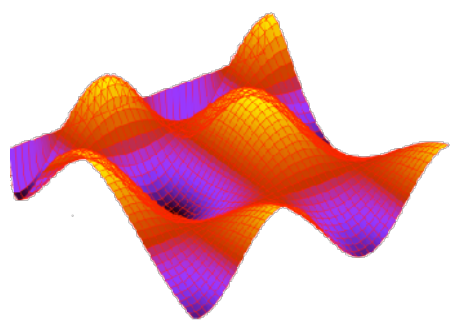
or Ax = b

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

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



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



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, tmLQCD, etc.
- Provides:
  - Various solvers for all major fermionic discretizations, with multi-GPU support
  - Additional performance-critical routines needed for gauge-field generation
- Maximize performance
  - Exploit physical symmetries to minimize memory traffic
  - Mixed-precision methods
  - Autotuning for high performance on all CUDA-capable architectures
  - Domain-decomposed (Schwarz) preconditioners for strong scaling
  - Eigenvector and deflated solvers (Lanczos, EigCG, GMRES-DR)
  - Multi-source solvers
  - Multigrid solvers for optimal convergence
- A research tool for how to reach the exascale

QUDA CONTRIBUTORS

10 years - lots of contributors

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

MAPPING THE DIRAC OPERATOR TO CUDA

Finite difference operator in LQCD is known as Dslash

Assign a single space-time point to each thread

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

Looping over direction each thread must

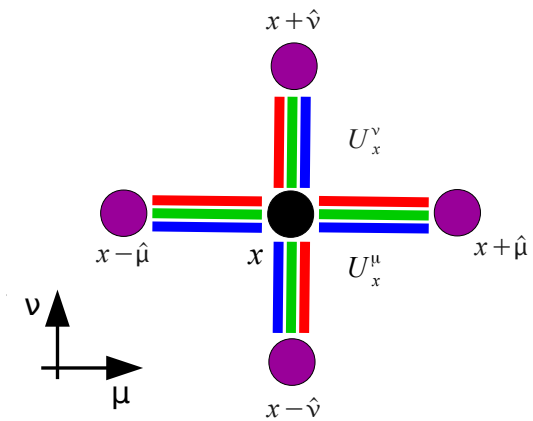
- Load the neighboring spinor (24 numbers x8)
- Load the color matrix connecting the sites (18 numbers x8)
- Do the computation
- Save the result (24 numbers)

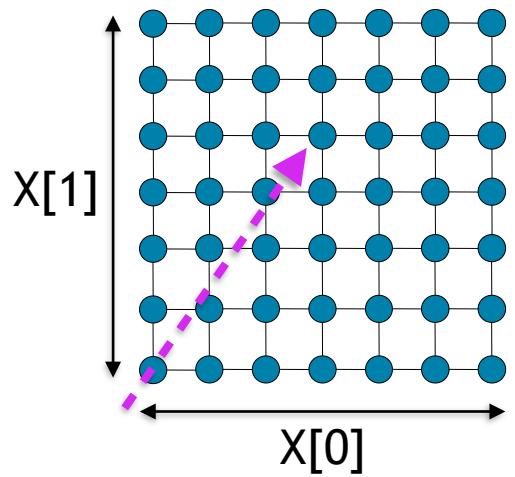
Each thread has (Wilson Dslash) 0.92 naive arithmetic intensity

QUDA reduces memory traffic

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

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



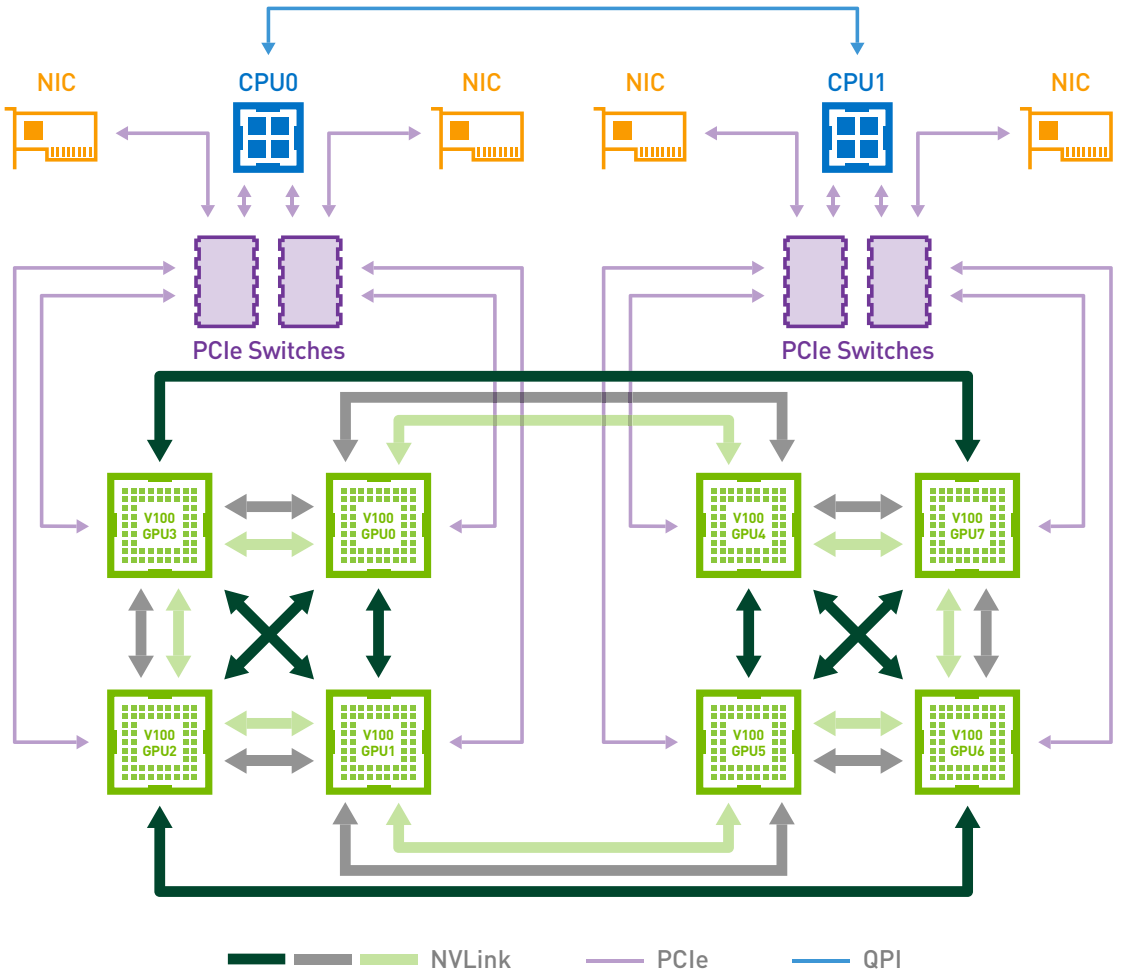


NVIDIA DGX-1

8 V100 GPUs (16/32 GB)

Hypercube-Mesh NVLink

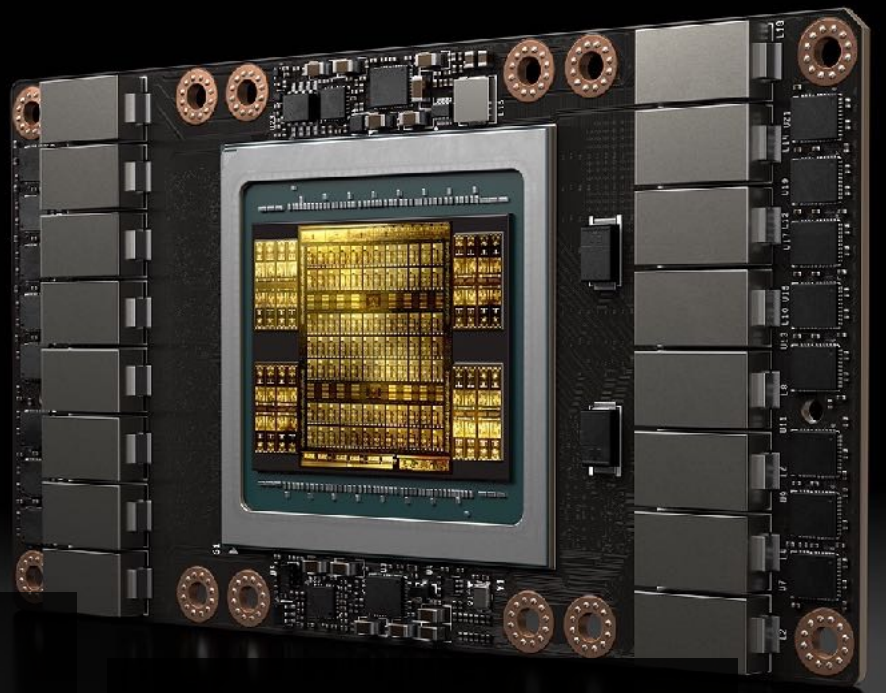
4 EDR IB



NVIDIA POWERS WORLD'S FASTEST SUPERCOMPUTER

Summit Becomes First System To Scale The 100 Petaflops Milestone

149 PF
HPC



27,648
Volta Tensor Core GPUs

TESLA V100 32GB

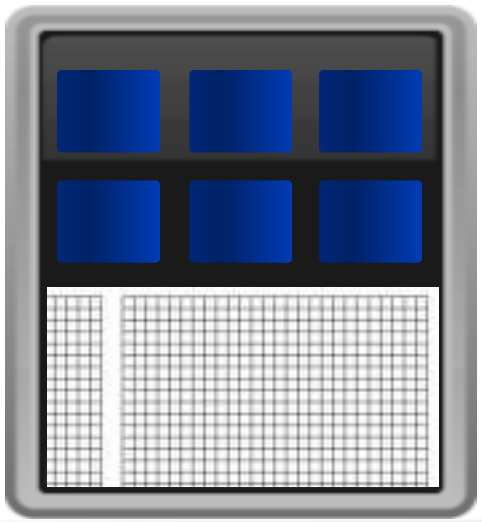
5,120 CUDA cores
640 NEW Tensor cores
7.8 FP64 TFLOPS | 15.7 FP32 TFLOPS | 125 Tensor TFLOPS
20MB SM RF | 16MB Cache
32GB HBM2 @ 900GB/s | 300GB/s NVLink

ACCELERATED COMPUTING

10X PERFORMANCE & 5X ENERGY EFFICIENCY FOR HPC

CPU

Optimized for
Serial Tasks



CPU Strengths

- Very large main memory
- Very fast clock speeds
- Latency optimized via large caches
- Small number of threads can run very quickly

CPU Weaknesses

- Relatively low memory bandwidth
- Cache misses very costly
- Low performance/watt

ACCELERATED COMPUTING

10X PERFORMANCE & 5X ENERGY EFFICIENCY FOR HPC

GPU Strengths

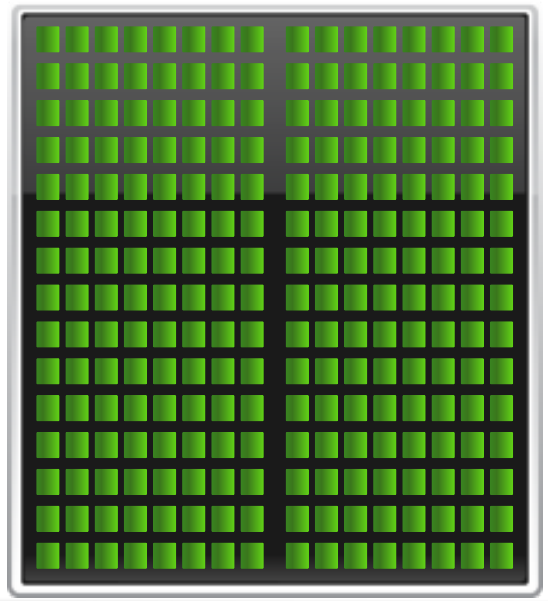
- High bandwidth main memory
- Significantly more compute resources
- Latency tolerant via parallelism
- High throughput
- High performance/watt

GPU Weaknesses

- Relatively low memory capacity
- Low per-thread performance

GPU Accelerator

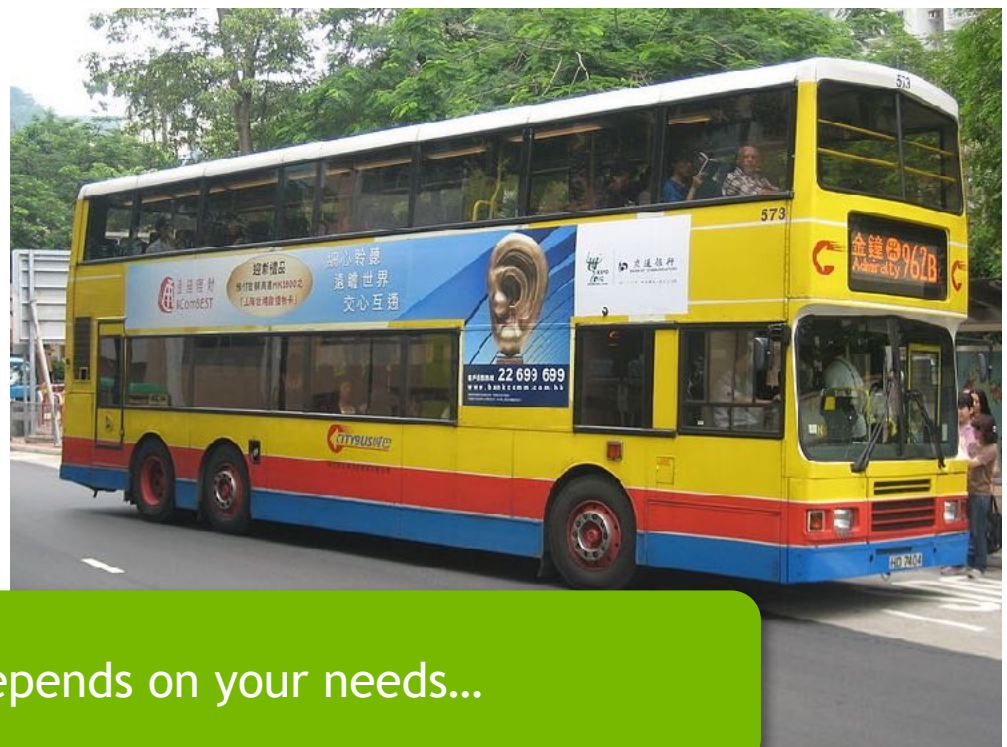
Optimized for Parallel Tasks



SPEED V. THROUGHPUT

Speed

Throughput

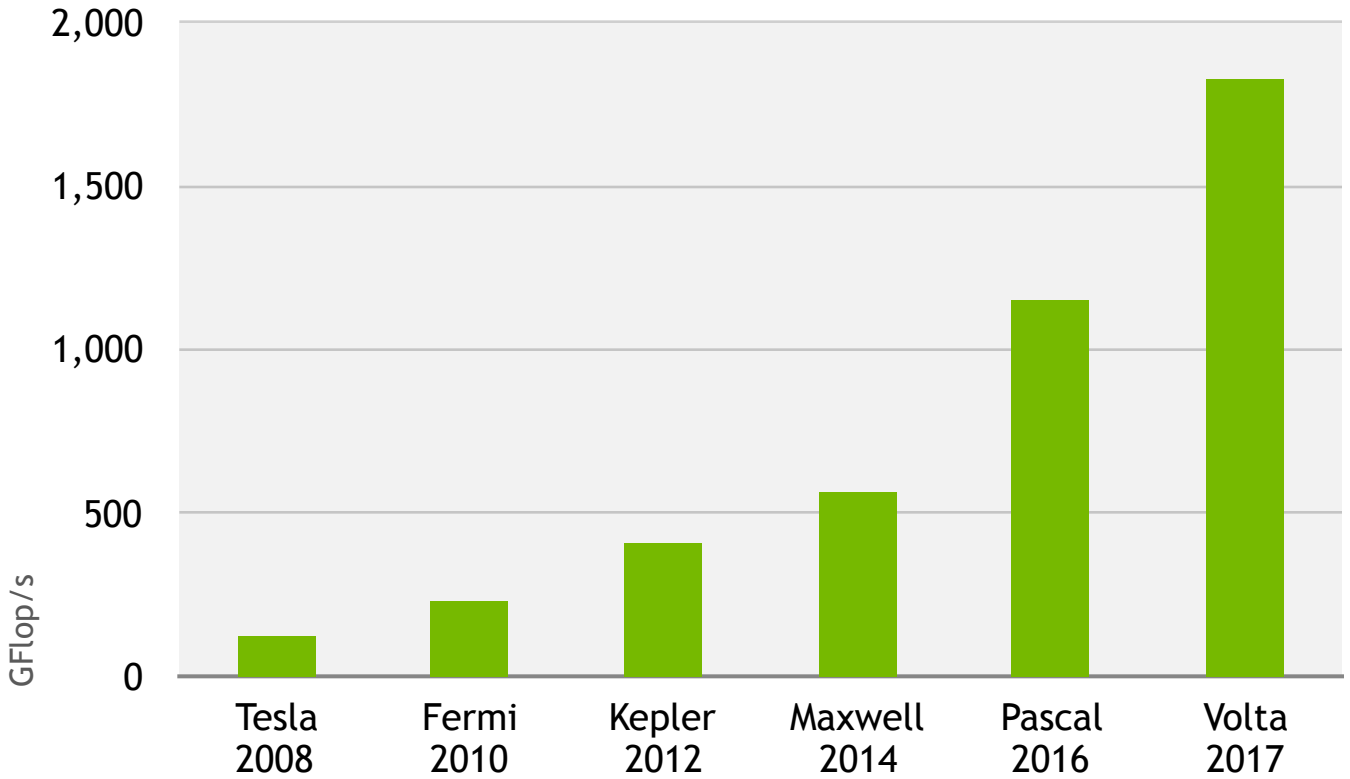


Which is better depends on your needs…

*Images from Wikimedia Commons via Creative Commons

QUDA STENCIL PERFORMANCE

GPUs now drive 10x faster than they used to ..



SCALING

A single GPU is not enough

Lattice QCD needs to be scaled - in one of many ways:

Multiple meanings

- Same problem size, more nodes, more GPUs
- Same problem, next generation GPUs
- Same problem, fewer bits to solve (mixed precision)
- Multigrid - strong scaling within the same run (not discussed here)

To tame strong scaling we have to understand the limiters

MIXED PRECISION

LINEAR SOLVERS

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

conjugate gradient

QUDA supports a wide range of linear solvers

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

Condition number inversely proportional to mass

- Light (realistic) masses are highly singular
- Naive Krylov solvers suffer from critical slowing down at decreasing mass

Entire solver algorithm must run on GPUs

- Time-critical kernel is the stencil application
- Also require BLAS level-1 type operations

MIXED-PRECISION CG

apply Dslash in sloppy precision (single, half)

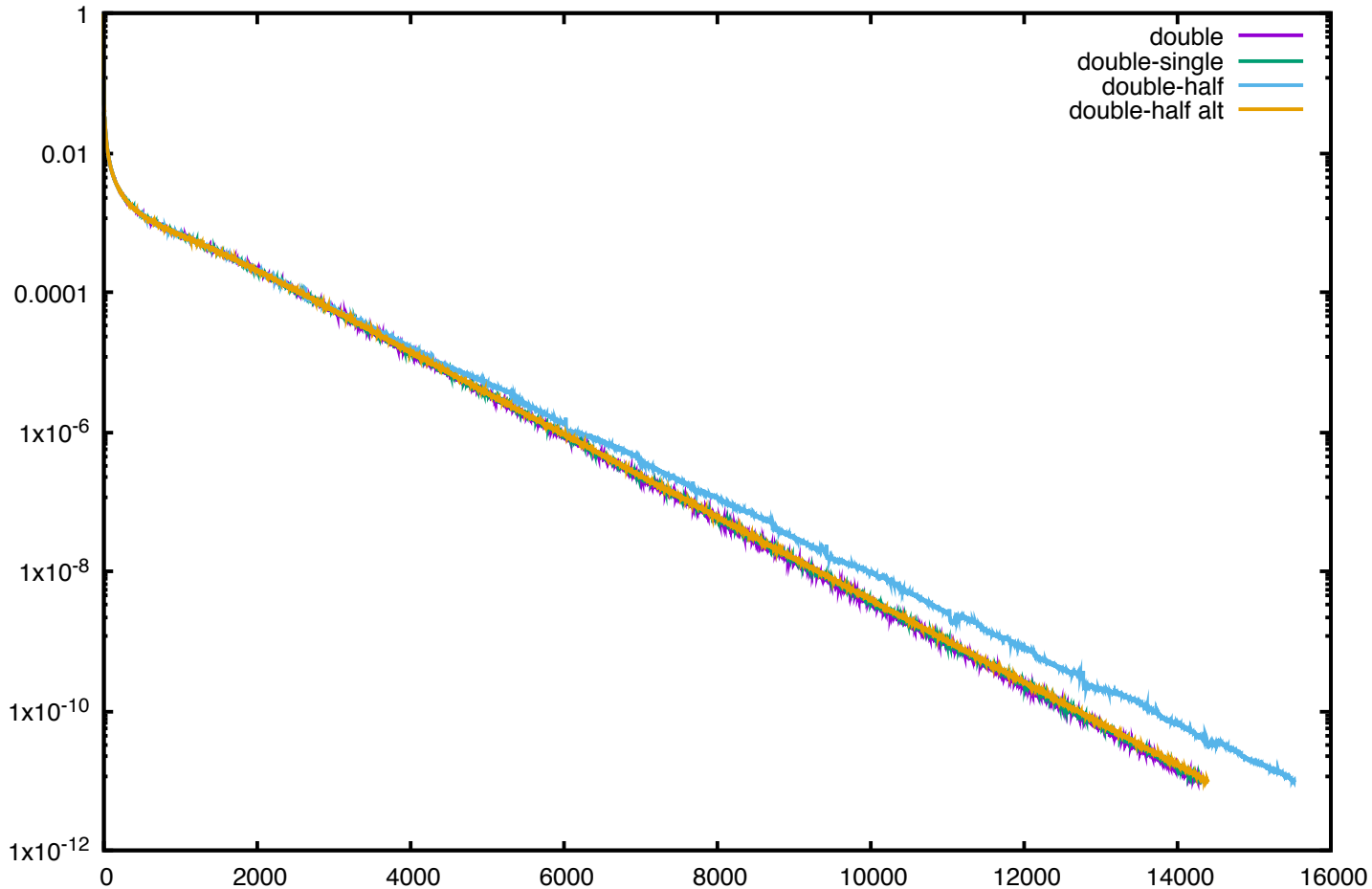
reliable residual replacement in high precision

ensures double-precision accuracy of final result

half precision **storage**:

```
Links: [-1,1] 16 bit fixed
Spinor:
16bit fixed (24 numbers)
float (exponent, 1)
```

use fp32 for actual arithmetics



MIXED-PRECISION CG

apply Dslash in sloppy precision (single, half)

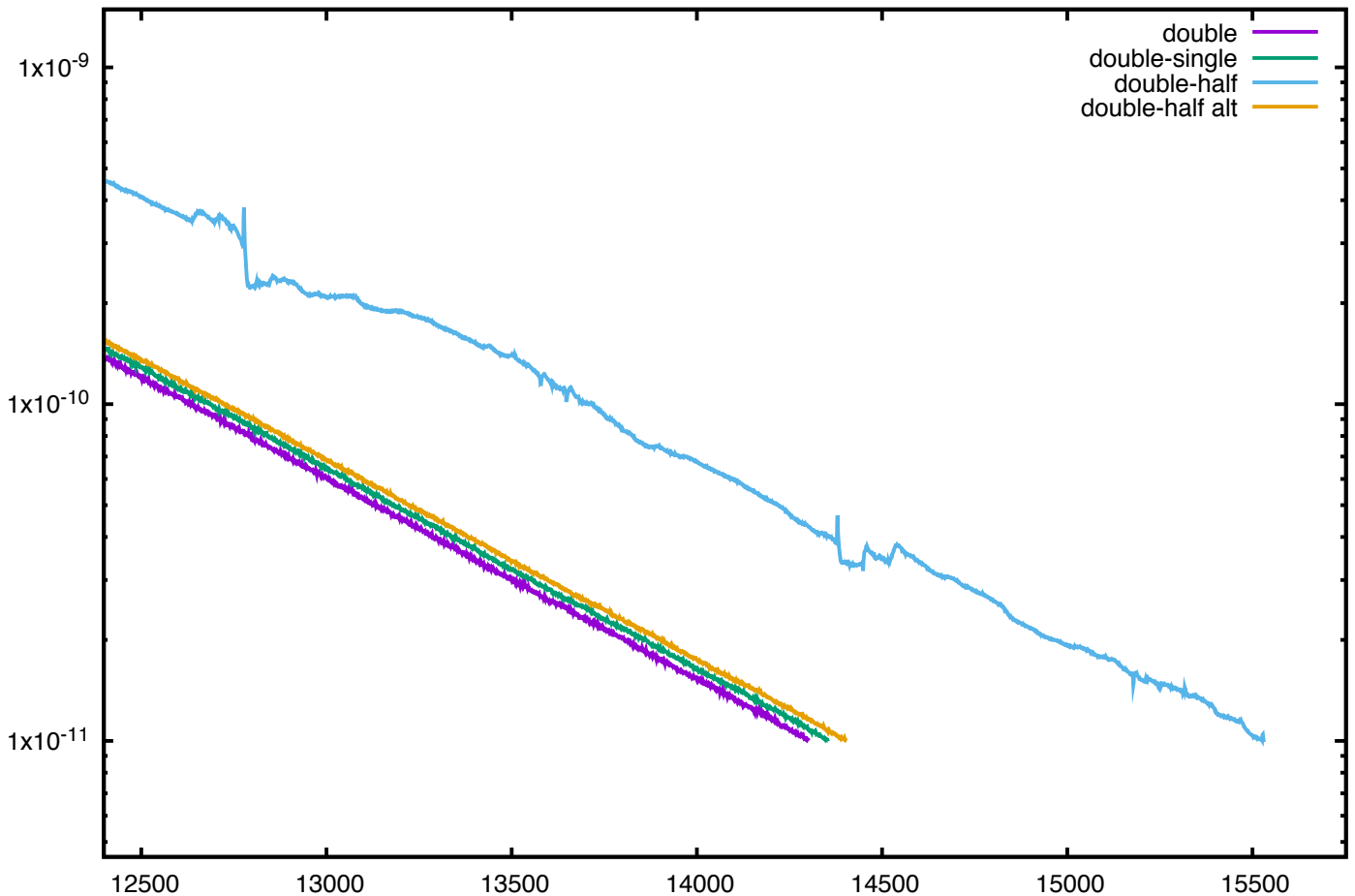
reliable residual replacement in high precision

ensures double-precision accuracy of final result

half precision **storage**:

```
Links: [-1,1] 16 bit fixed
Spinor:
16bit fixed (24 numbers)
float (exponent, 1)
```

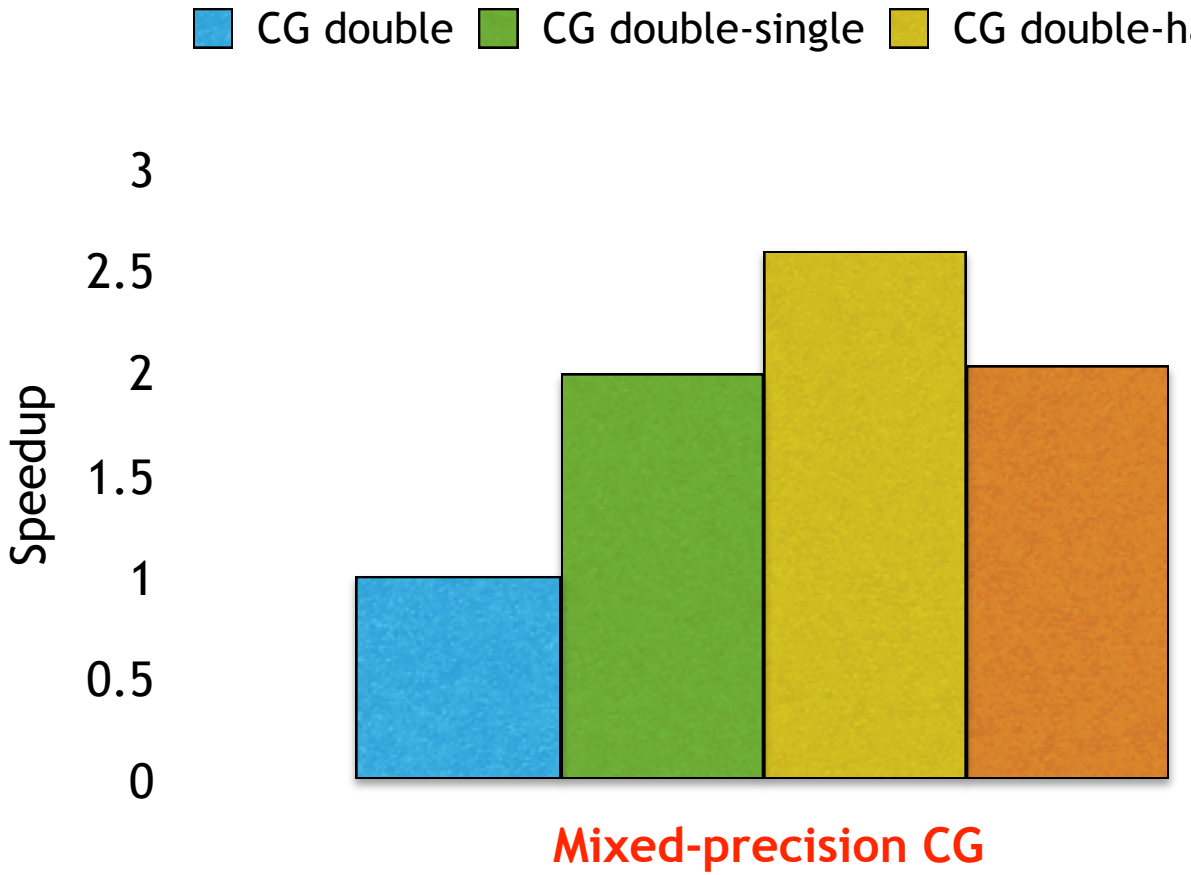
use fp32 for actual arithmetics



Configuration provided by
HotQCD collaboration (Mukherjee *et al*)

MIXED-PRECISION DEFLATION

V=48^3x12, HISQ operator, physical light quarks, tol 10^{-10}, 2xV100



EIGENSOLVERS

Multiple workflows require repeated solution with different RHS with the same matrix

Multigrid not amenable to all linear operators

Eigenvector deflation is a robust alternative applicable to all operators

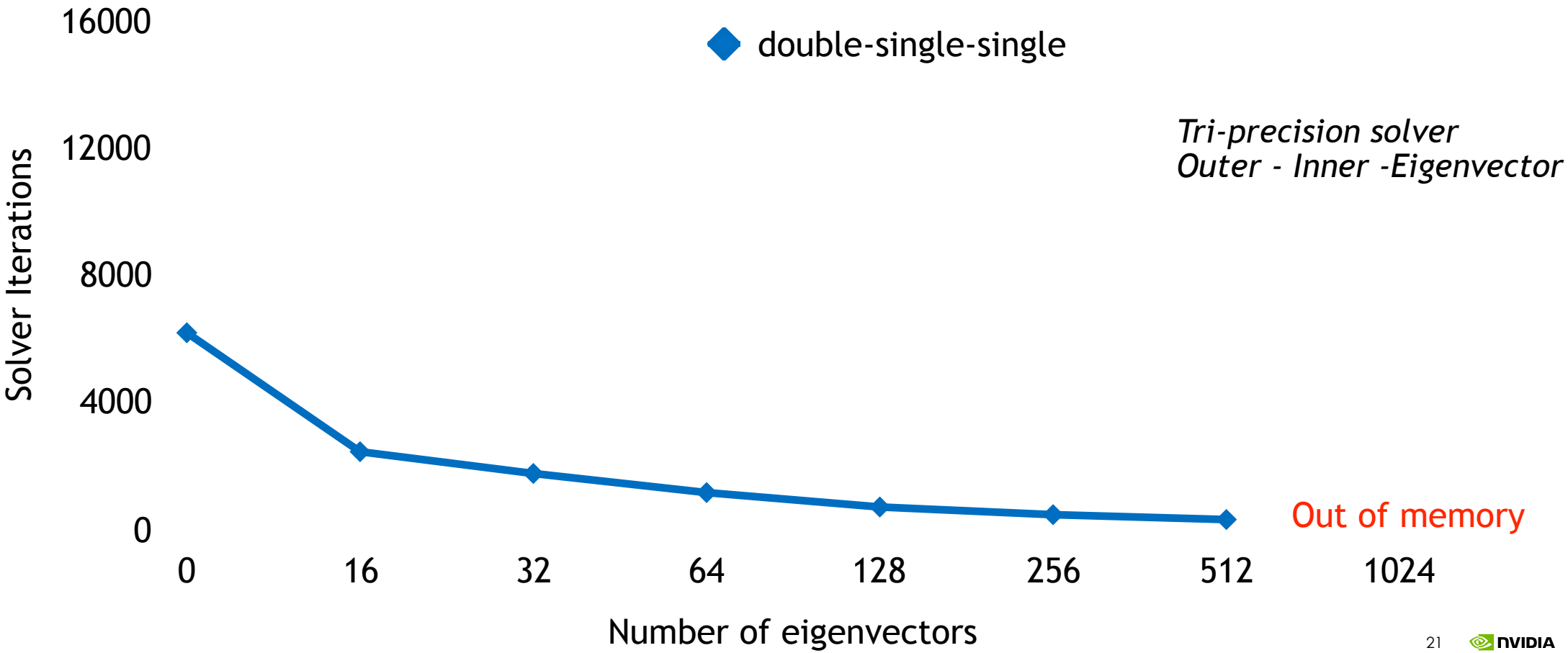
- Deflate out low modes from linear operator to accelerate the solver
- Cost of eigensolver is amortized if we solve enough RHS
- Aside: also use deflation to accelerate multigrid

Memory overheads can be limiting factor

- May require storage of 1000s of vectors, ideally in fast memory

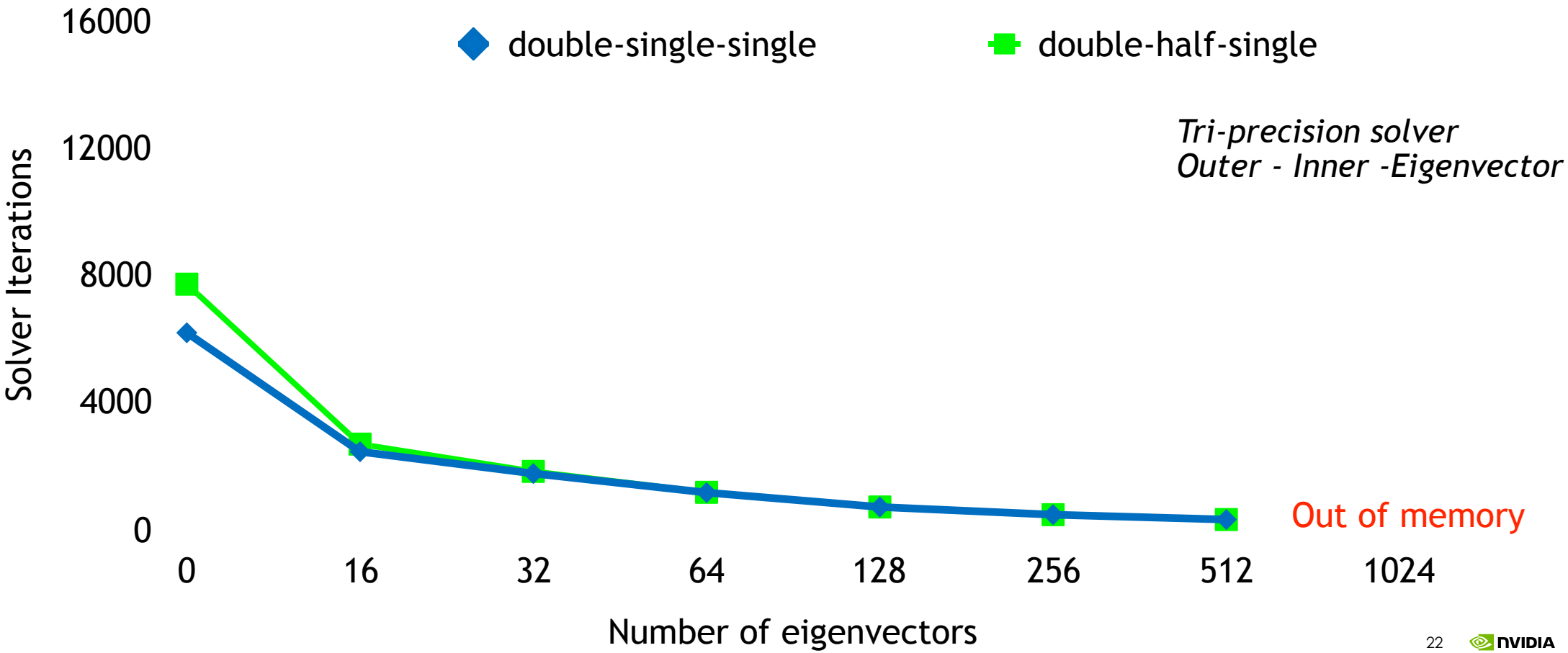
DEFLATION STABILIZES LOW PRECISION

V=48^3x12, HISQ operator, physical light quarks, tol 10^{-10}



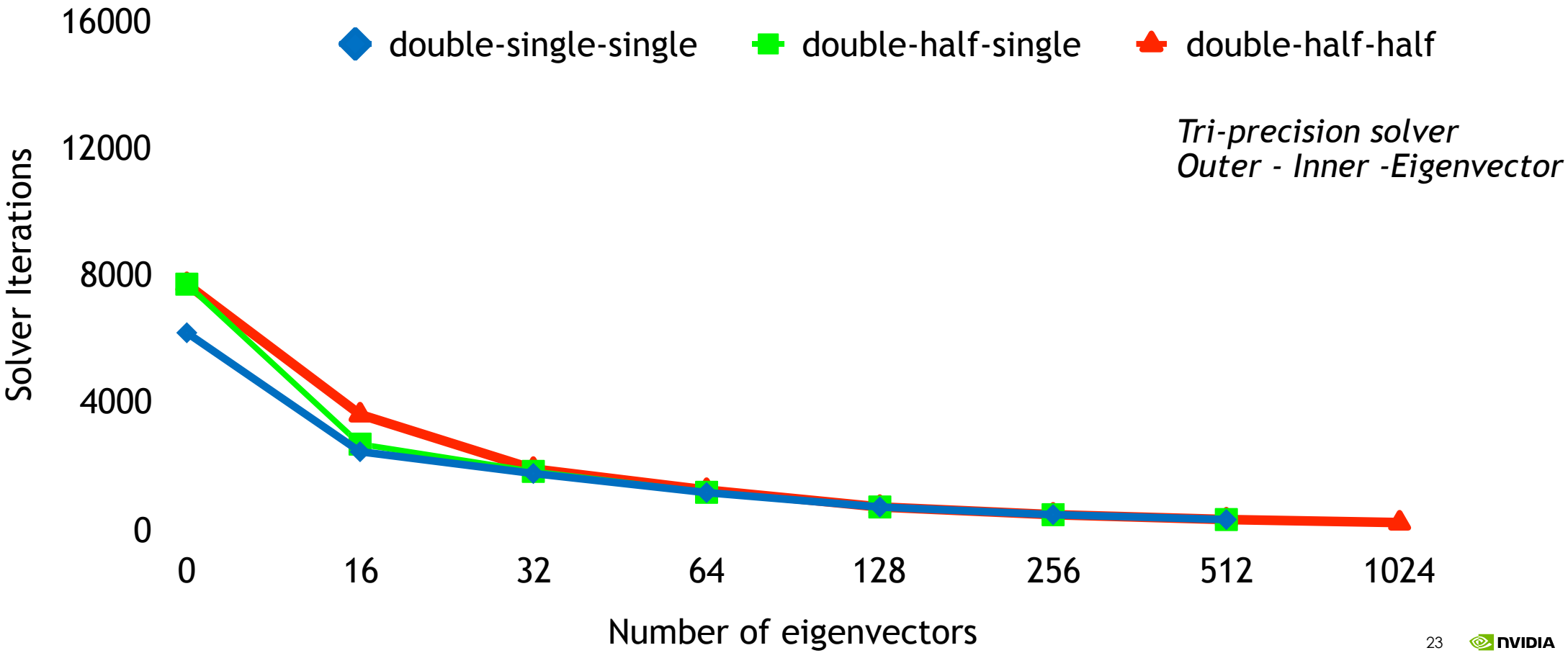
DEFLATION STABILIZES LOW PRECISION

V=48^3x12, HISQ operator, physical light quarks, tol 10^{-10}



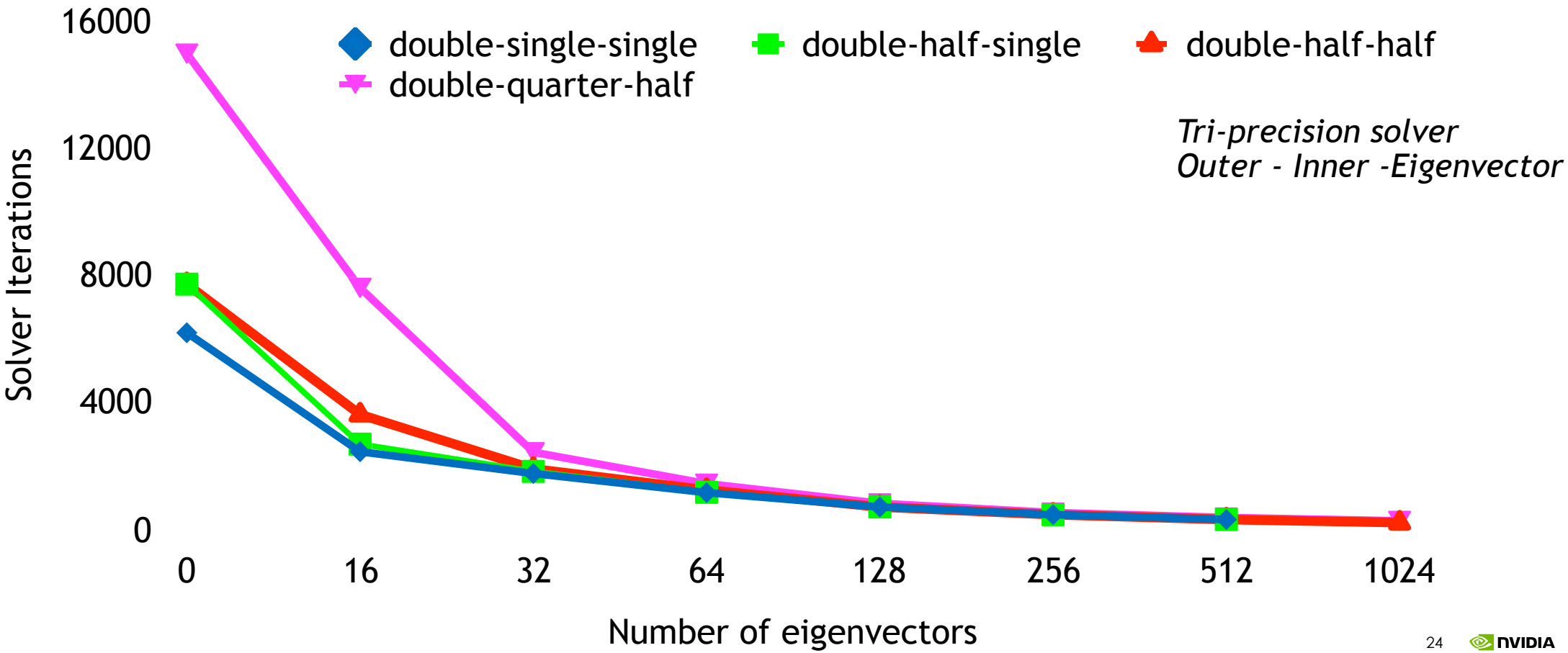
DEFLATION STABILIZES LOW PRECISION

V=48^3x12, HISQ operator, physical light quarks, tol 10^{-10}



DEFLATION STABILIZES LOW PRECISION

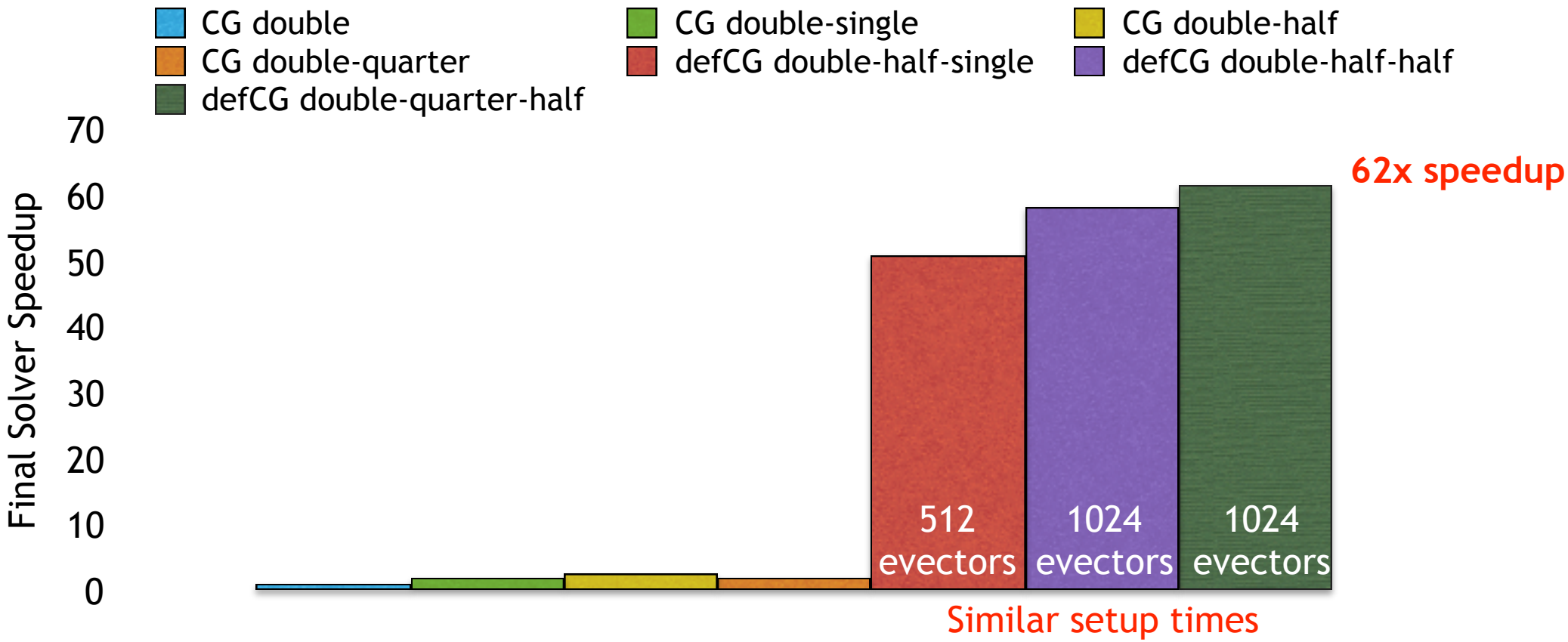
V=48^3x12, HISQ operator, physical light quarks, tol 10^{-10}



Configuration provided by
HotQCD collaboration (Mukherjee *et al*)

MIXED-PRECISION DEFLATION

V=48^3x12, HISQ operator, physical light quarks, tol 10^{-10}, 2xV100



DEFLATED ADAPTIVE MG AT THE PHYSICAL POINT

in collaboration with Dean Howarth (BU)

Multigrid shifts the lowest eigenvalues to the coarse grids

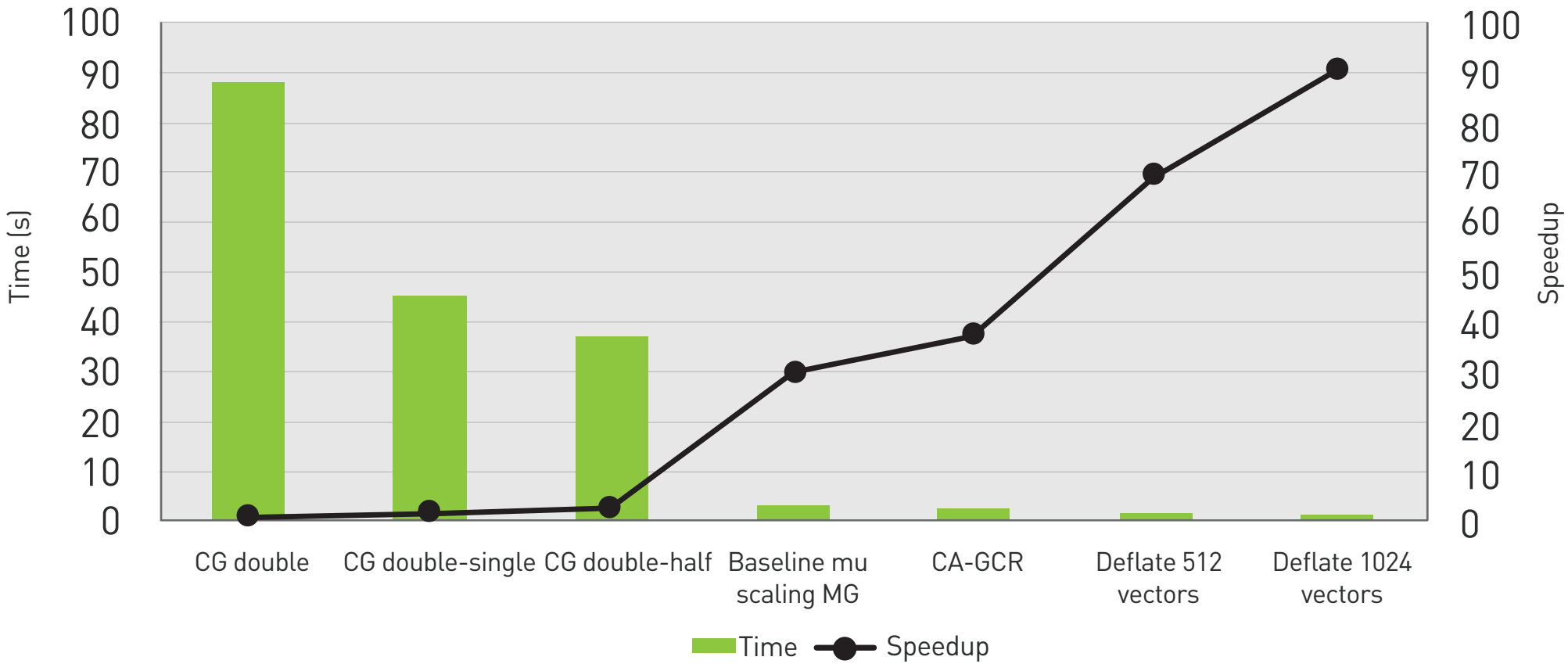
- some Dirac operators (staggered / twisted mass) end up with pathological coarse-grid spectrum

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 ADAPTIVE MG AT THE PHYSICAL POINT

in collaboration with
Dean Howarth



SCALING

MULTI-GPU SCALING

HPC nodes continue to get denser

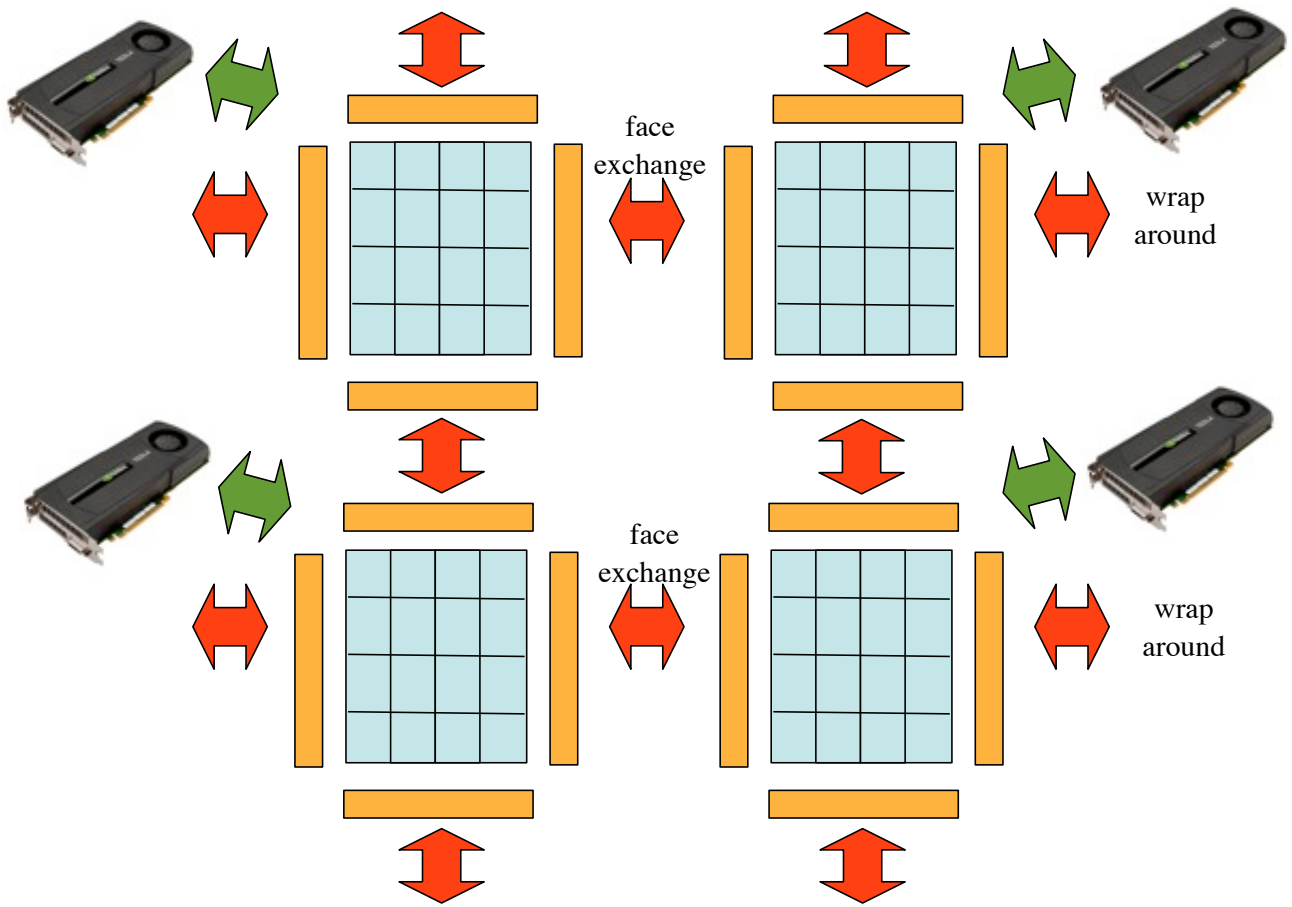
Much effort to improve strong scaling in QUDA

Key technologies employed

- Peer-to-peer communication - skip MPI and directly utilize DMA copies
- GPU Direct RDMA (GDR) - GPU <-> NIC <-> GPU without traversing CPU memory
- Node topology awareness
- Autotuning for Dslash communication policy

Framework for Exascale communication models - NVSHMEM (OpenSHMEM for GPUs)

MULTI-GPU BUILDING BLOCKS



Halo packing Kernel

Interior Kernel

Halo communication

Halo update Kernel

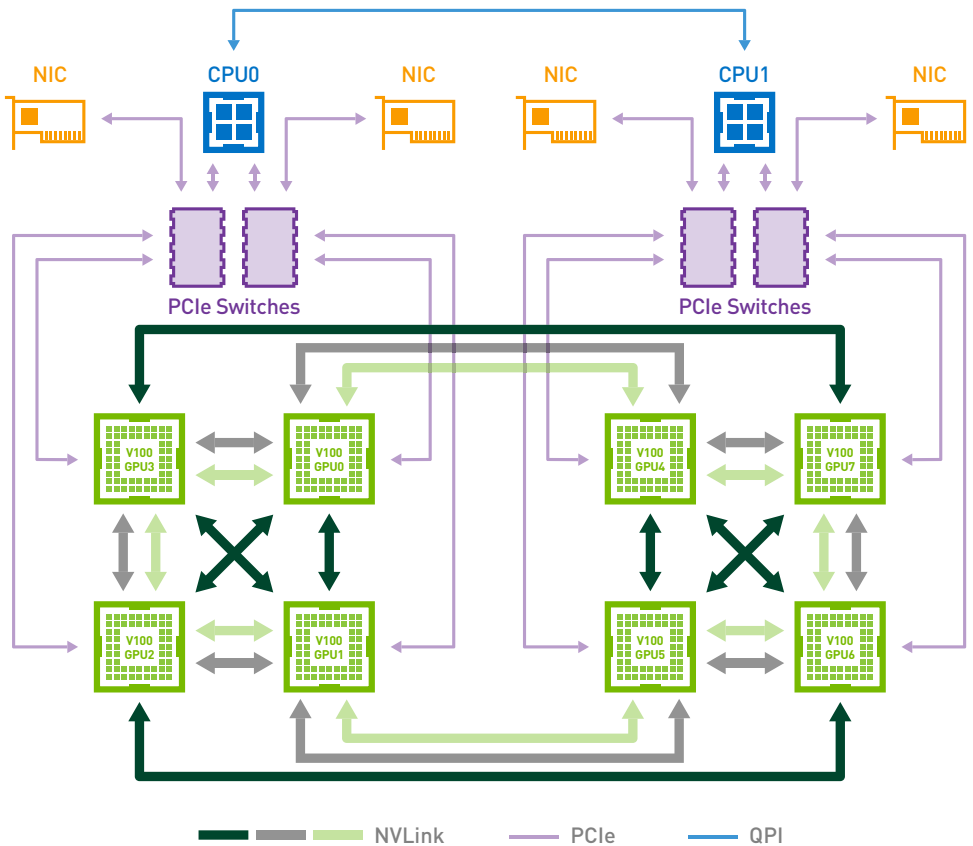
BENCHMARKING TESTBED

NVIDIA Prometheus Cluster

DGX-1 nodes

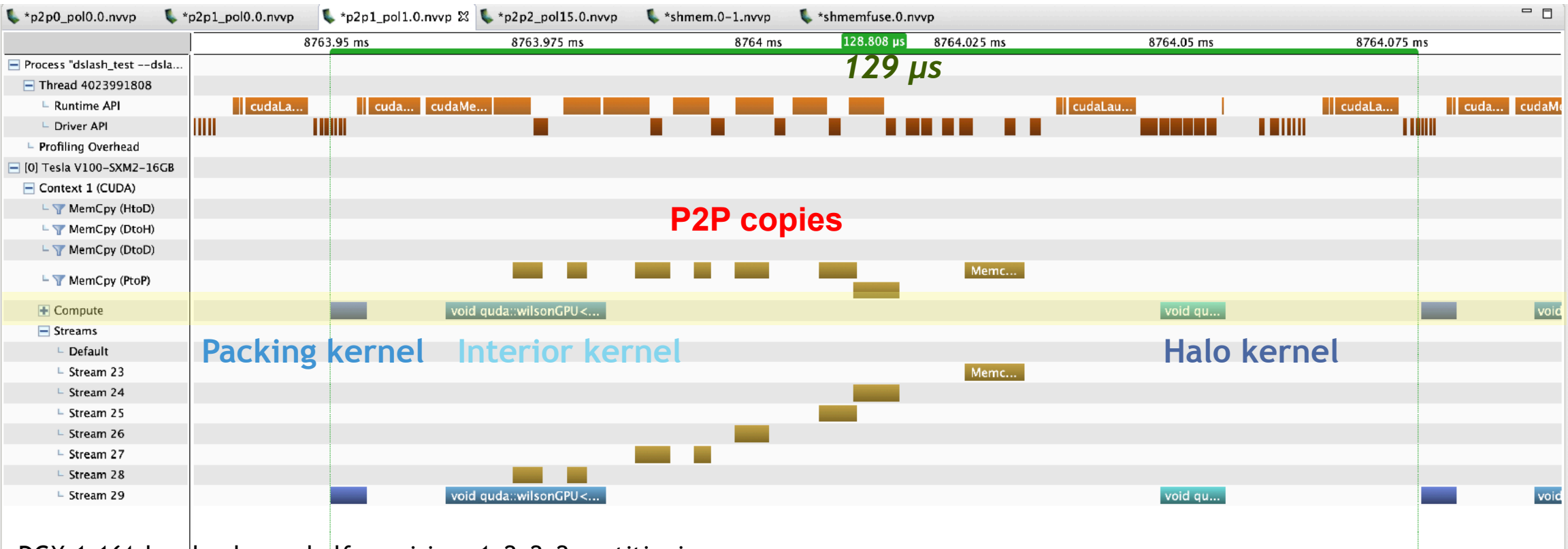
- 8x V100 GPUs connected through NVLink
- 4x EDR for inter-node communication
- Optimal placement of GPUs and NIC

Balanced GPU / IB configuration



WHAT IS LIMITING STRONG SCALING

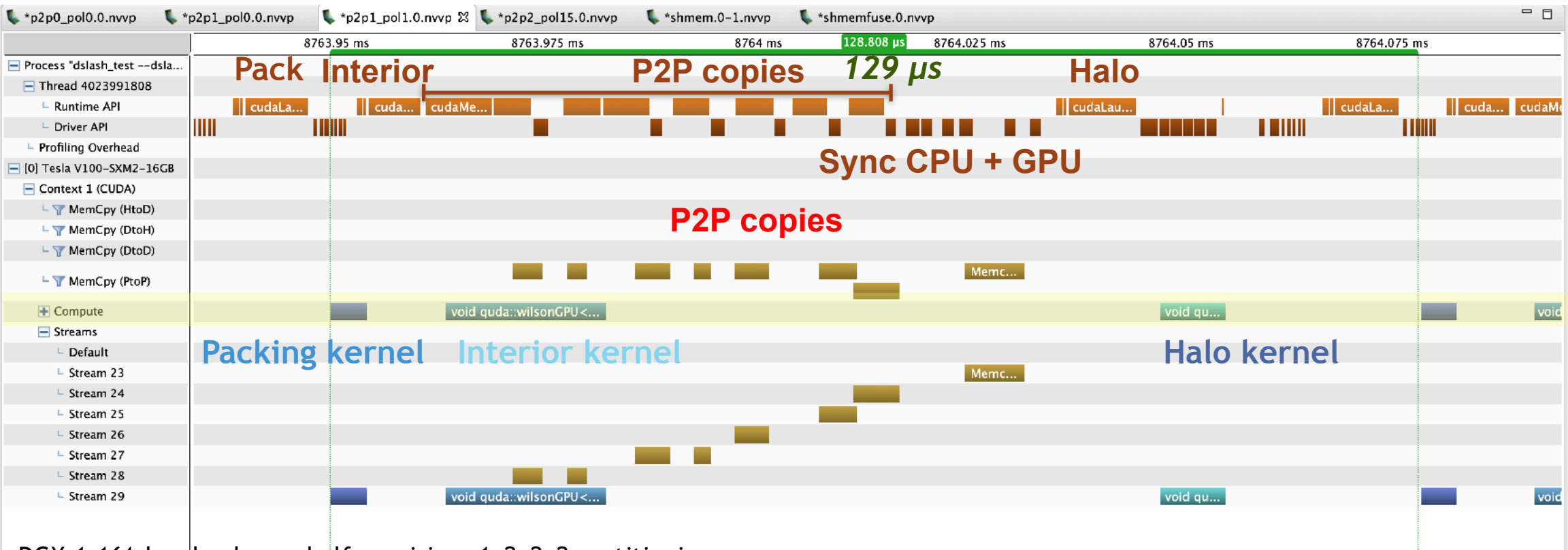
current state of the art using CUDA IPC (P2P) over NVLink



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

WHAT IS LIMITING STRONG SCALING

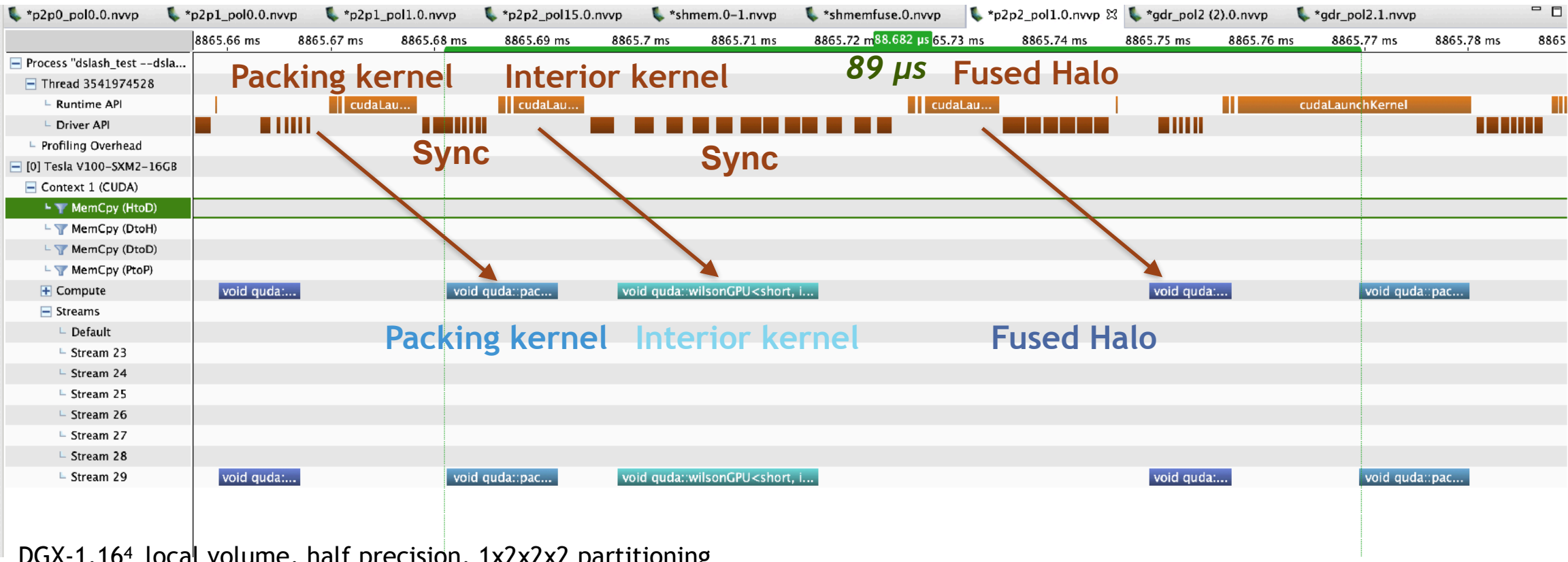
Significant API overhead



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

REDUCING API OVERHEADS

Packing kernel writes to remote GPU using CUDA IPC



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

NVSHMEM

GPU centric communication

Implementation of OpenSHMEM[1], a Partitioned Global Address Space (PGAS) library

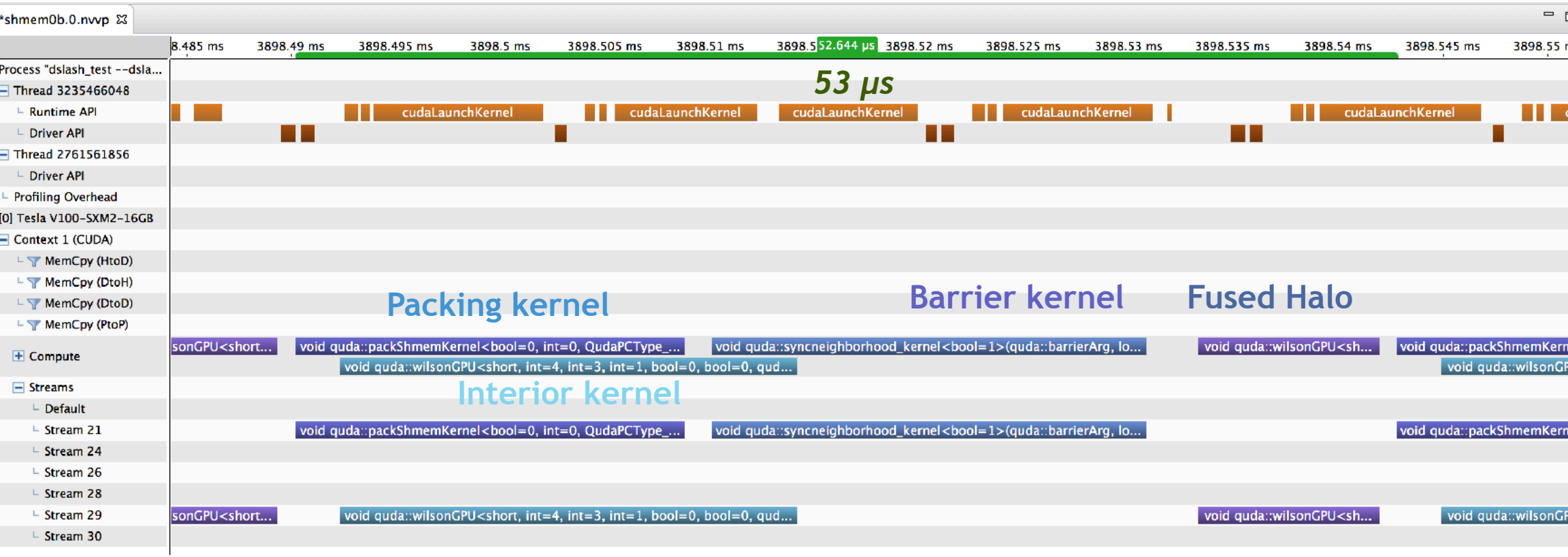
NVSHMEM features

- 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 (intranode)
- InfiniBand support (internode)
- Interoperability with MPI and OpenSHMEM libraries

currently in early access

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

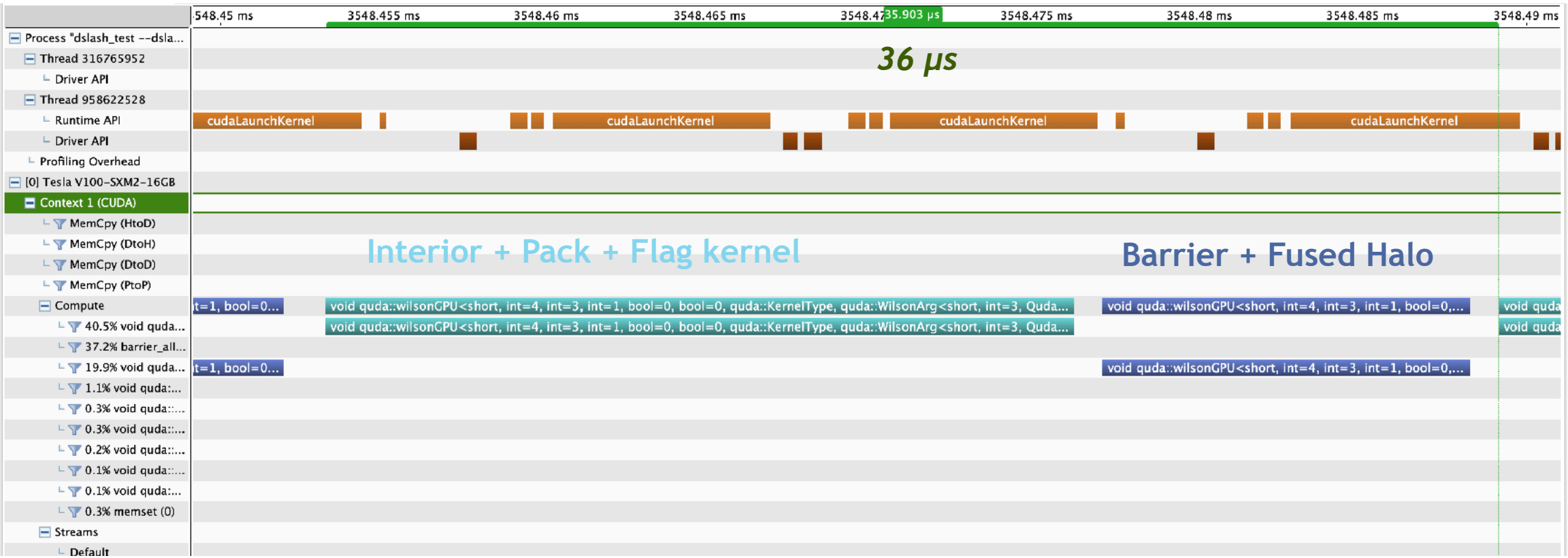
NVSHMEM DSLASH



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

NVSHMEM + FUSING KERNELS

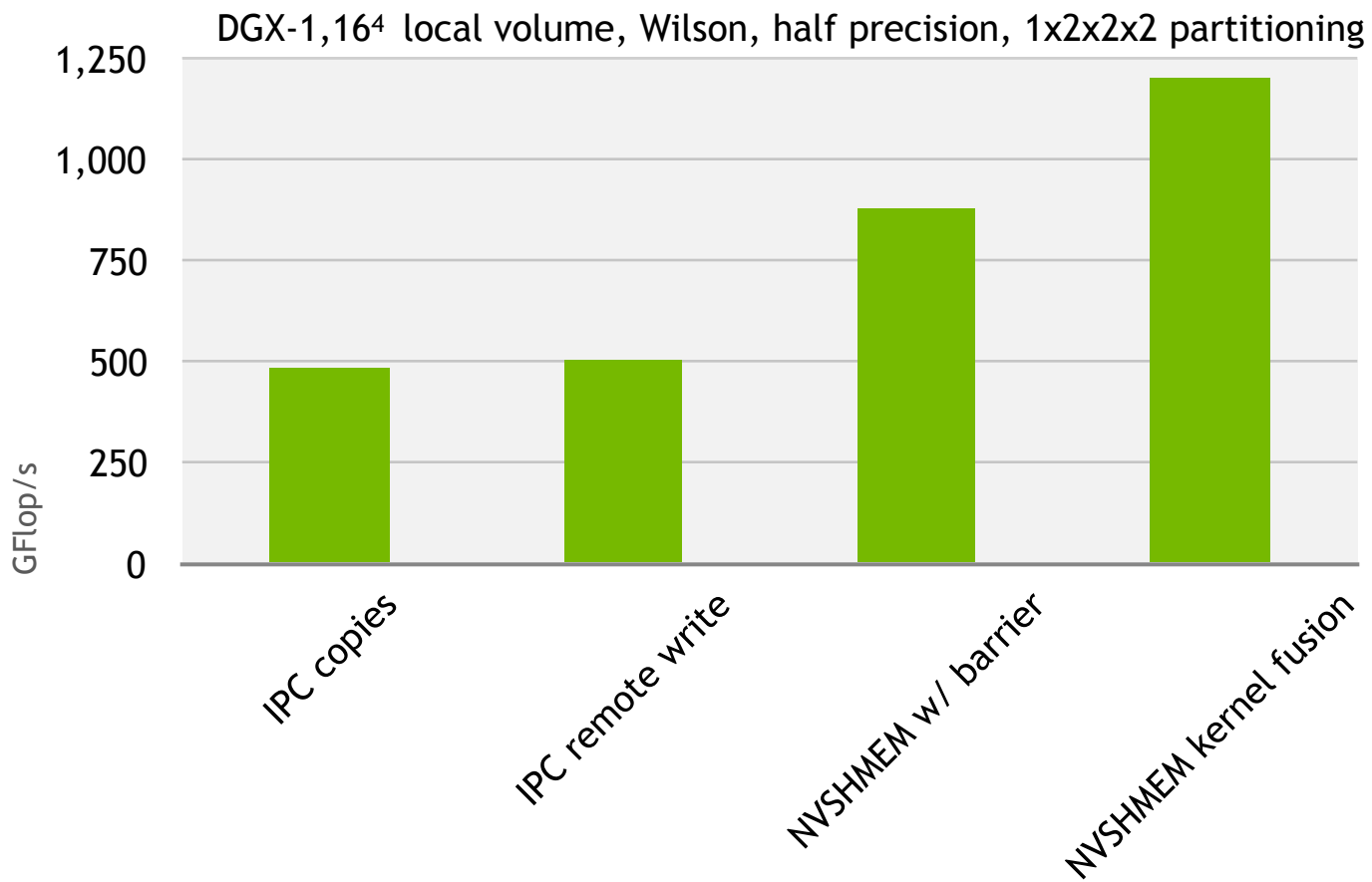
no extra packing and barrier kernels needed



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

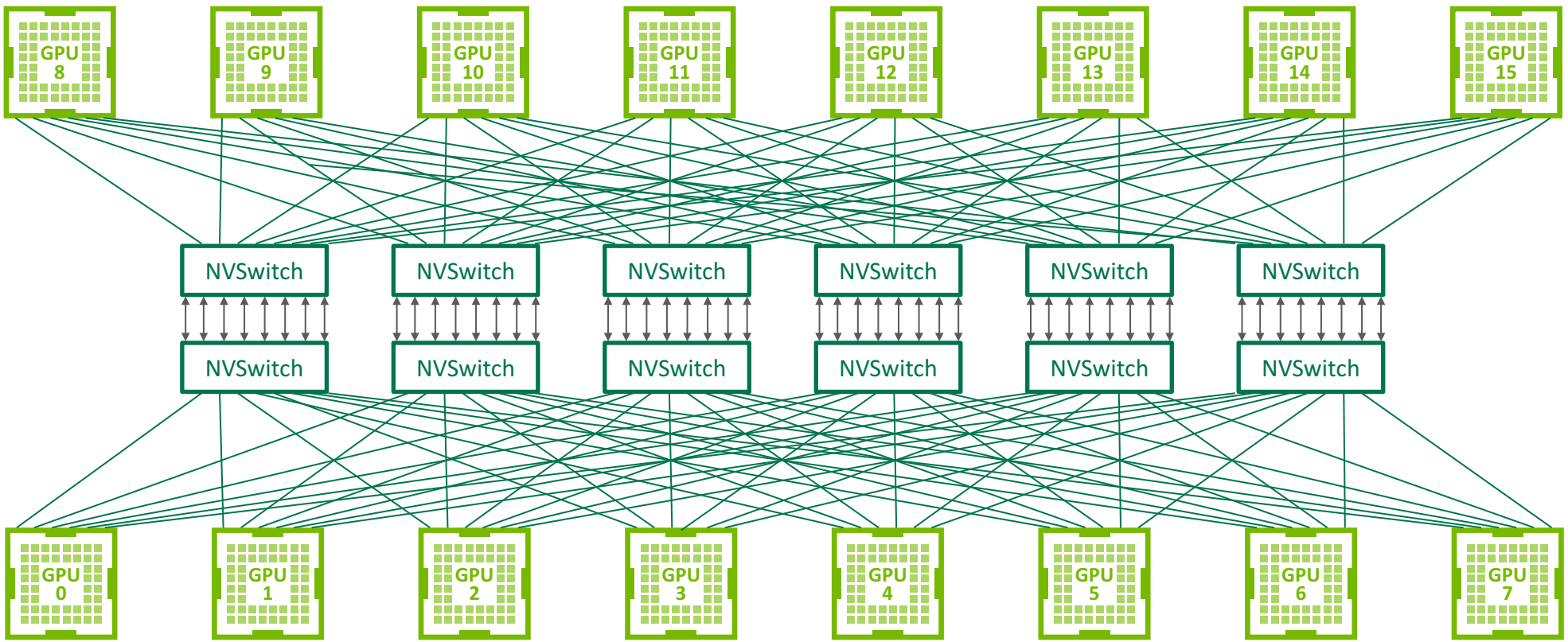
LATENCY OPTIMIZATIONS

Different strategies implemented



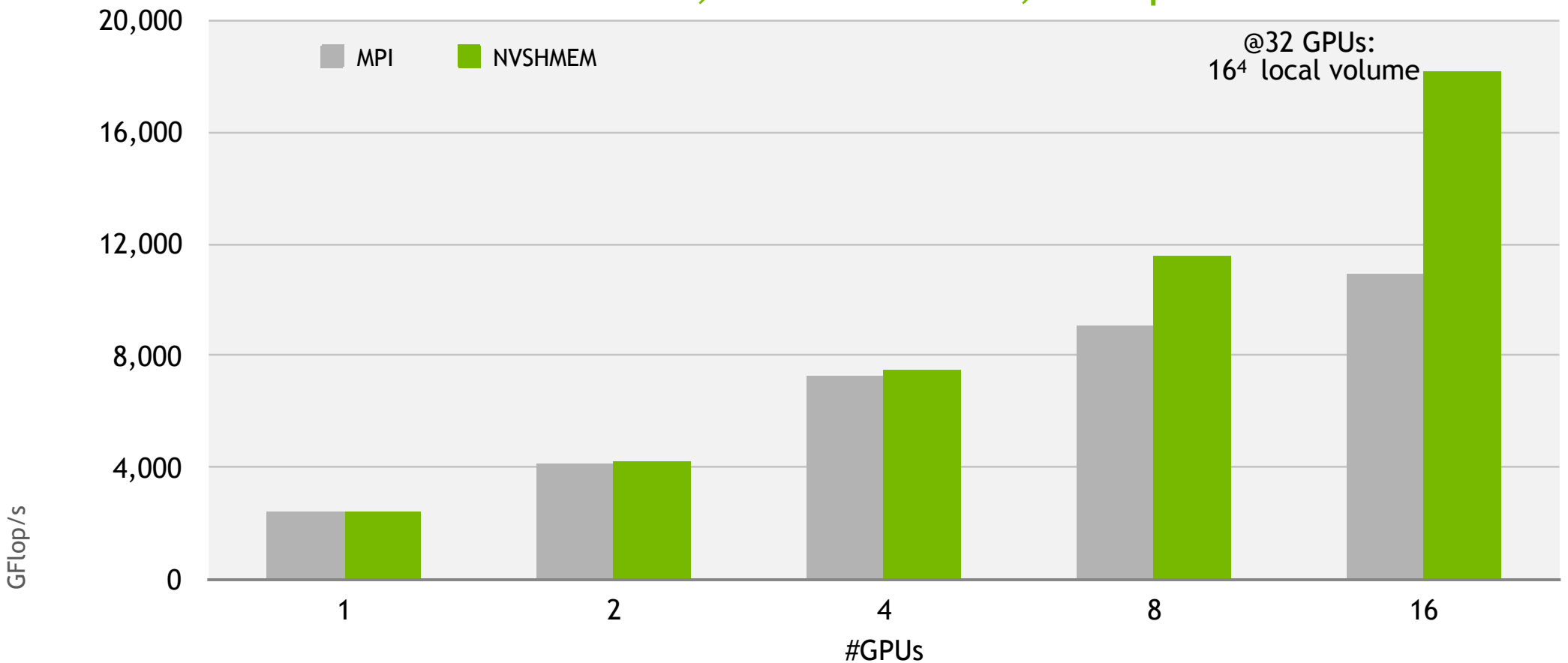
DGX-2: FULL NON-BLOCKING BANDWIDTH

2.4 TB/s bisection bandwidth



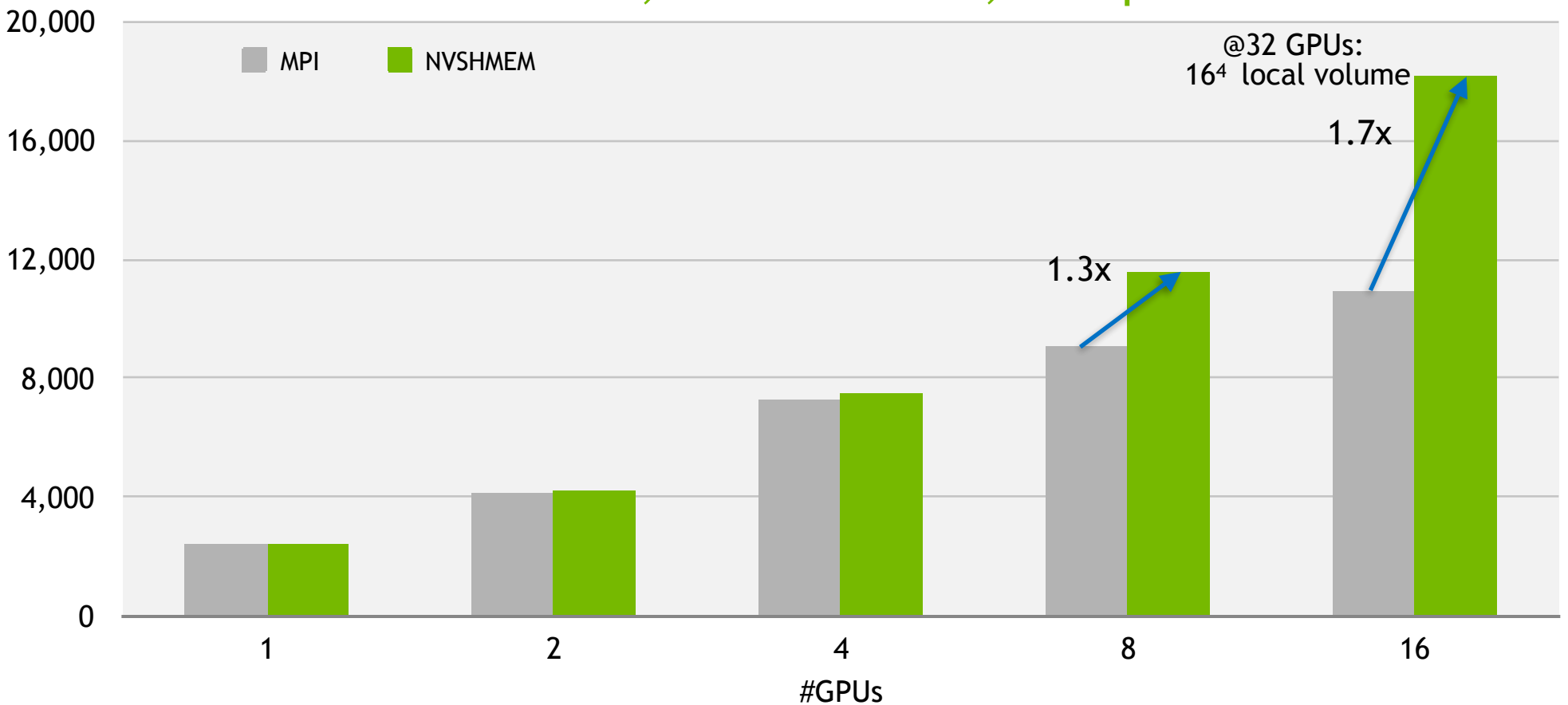
DGX-2 STRONG SCALING

Global Volume 32^4, Wilson-Dslash, half precision



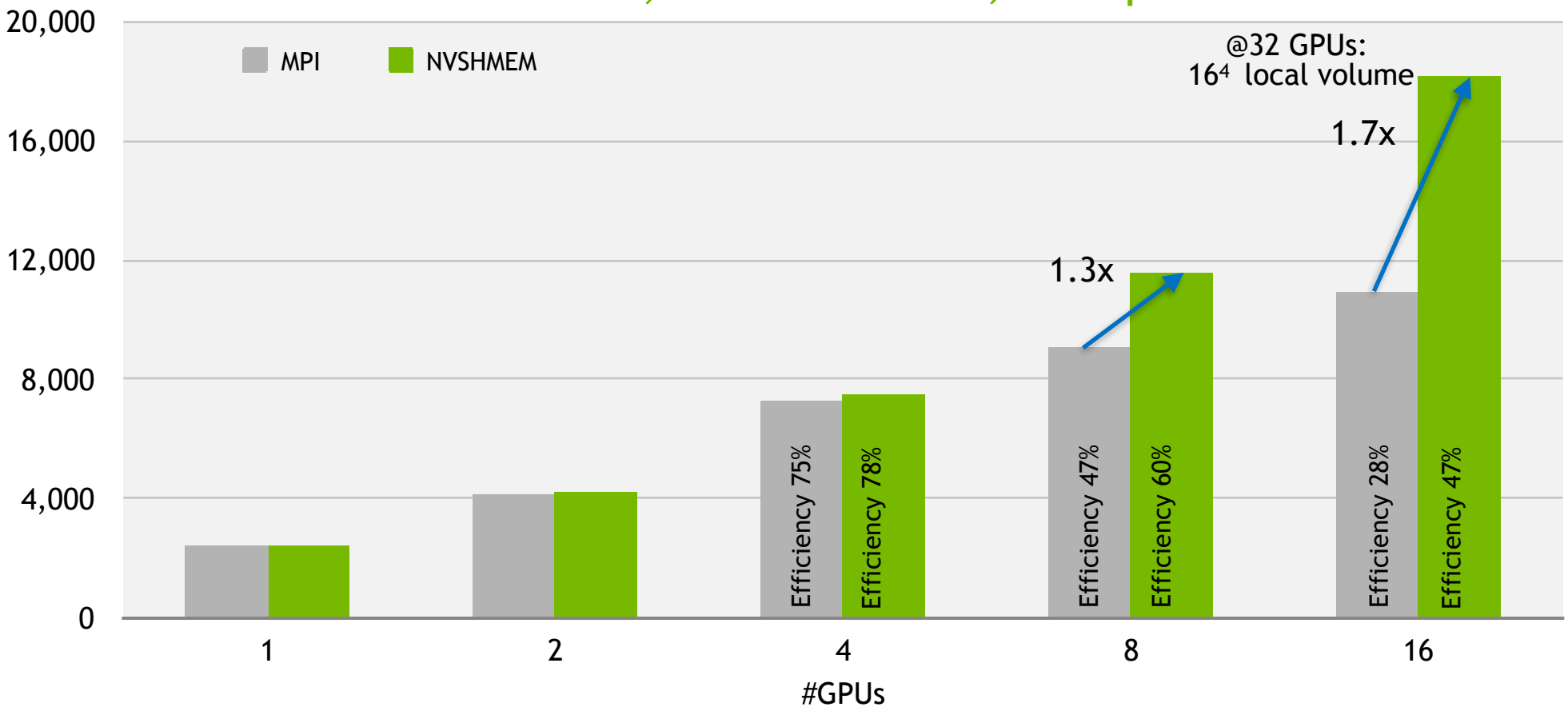
DGX-2 STRONG SCALING

Global Volume 32^4, Wilson-Dslash, half precision



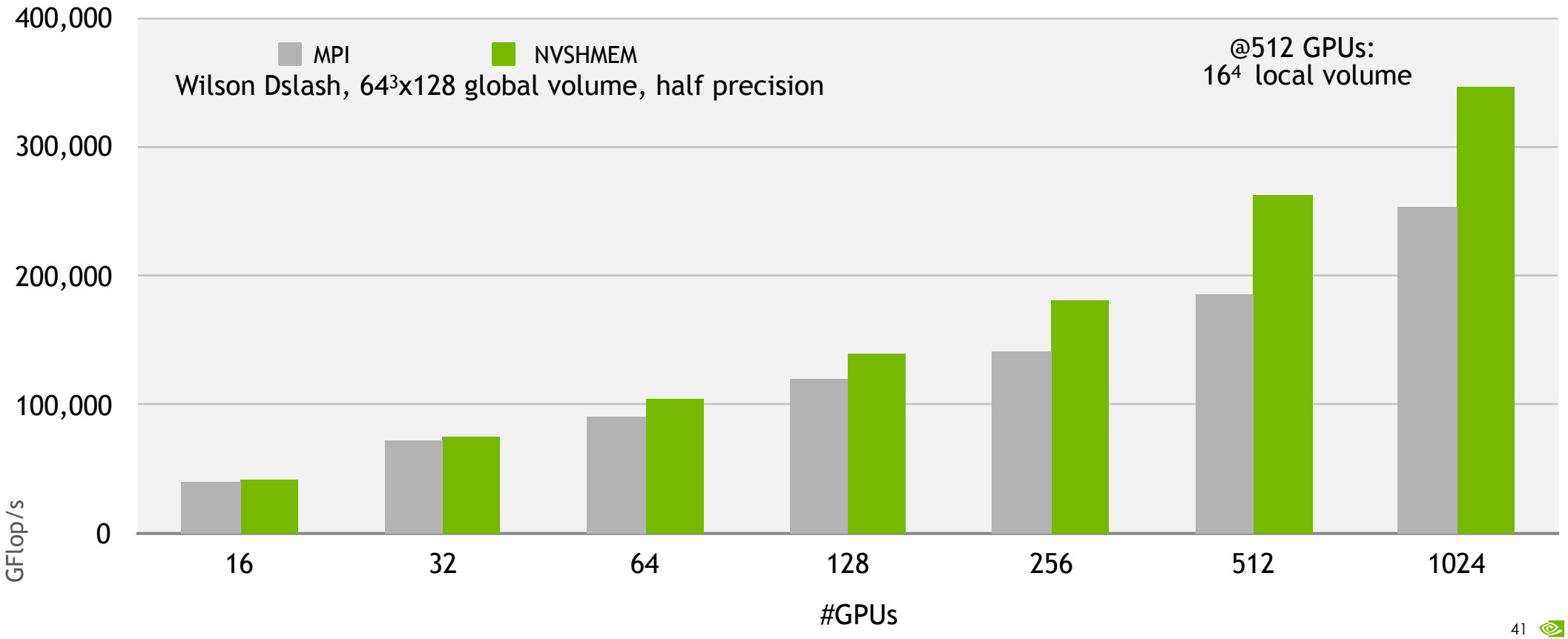
DGX-2 STRONG SCALING

Global Volume 32^4, Wilson-Dslash, half precision



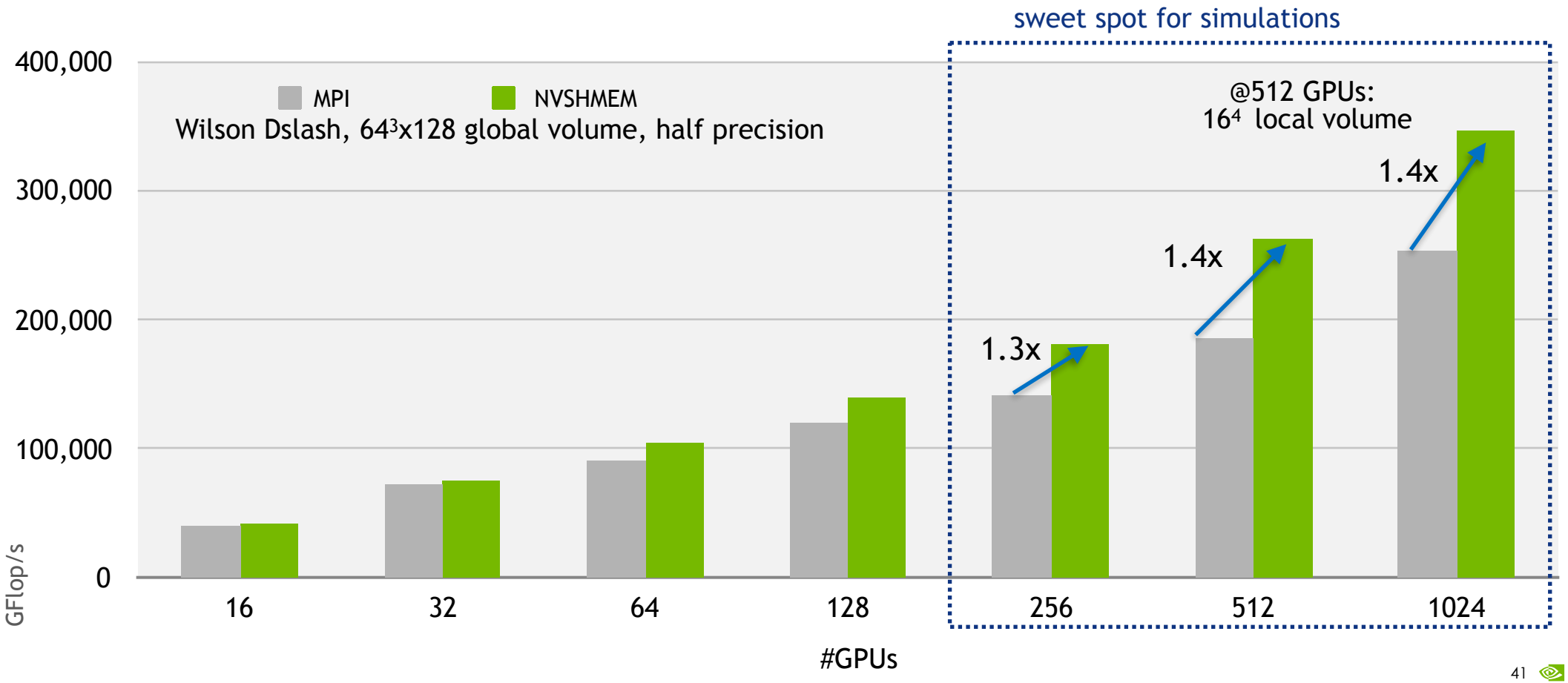
MULTI-NODE SCALING

DGX SuperPOD (DGX2-H: 16 V100 (32GB), 8 EDR IB)

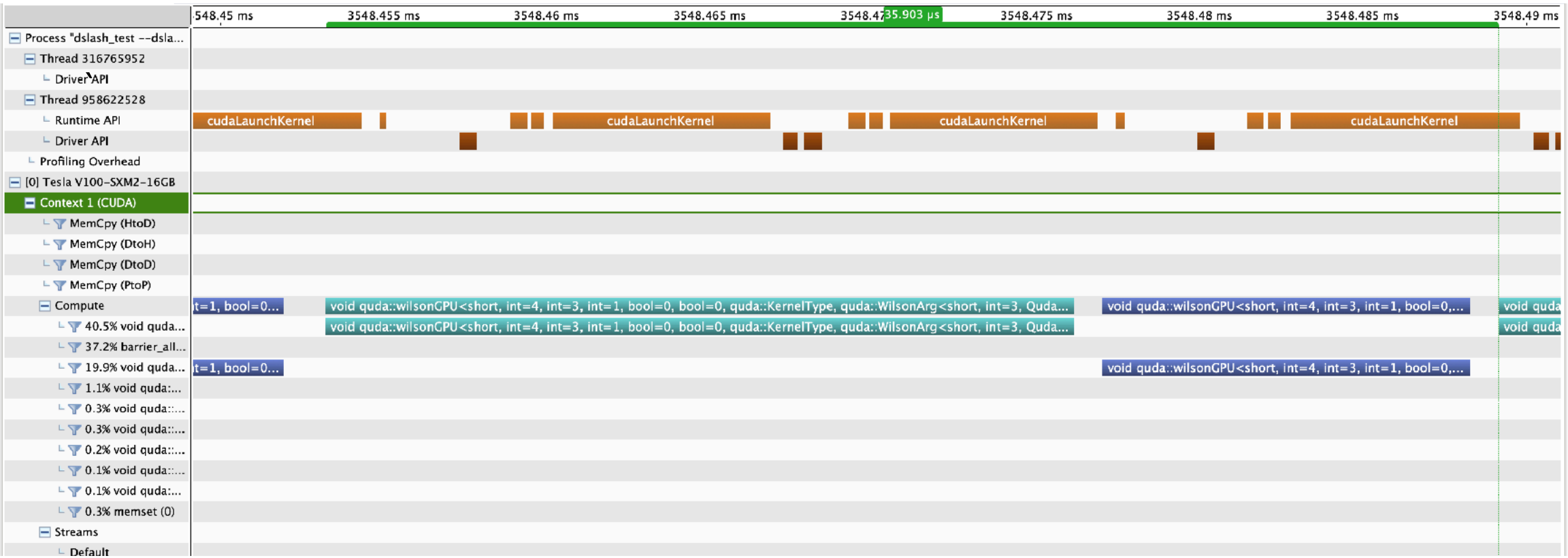


MULTI-NODE SCALING

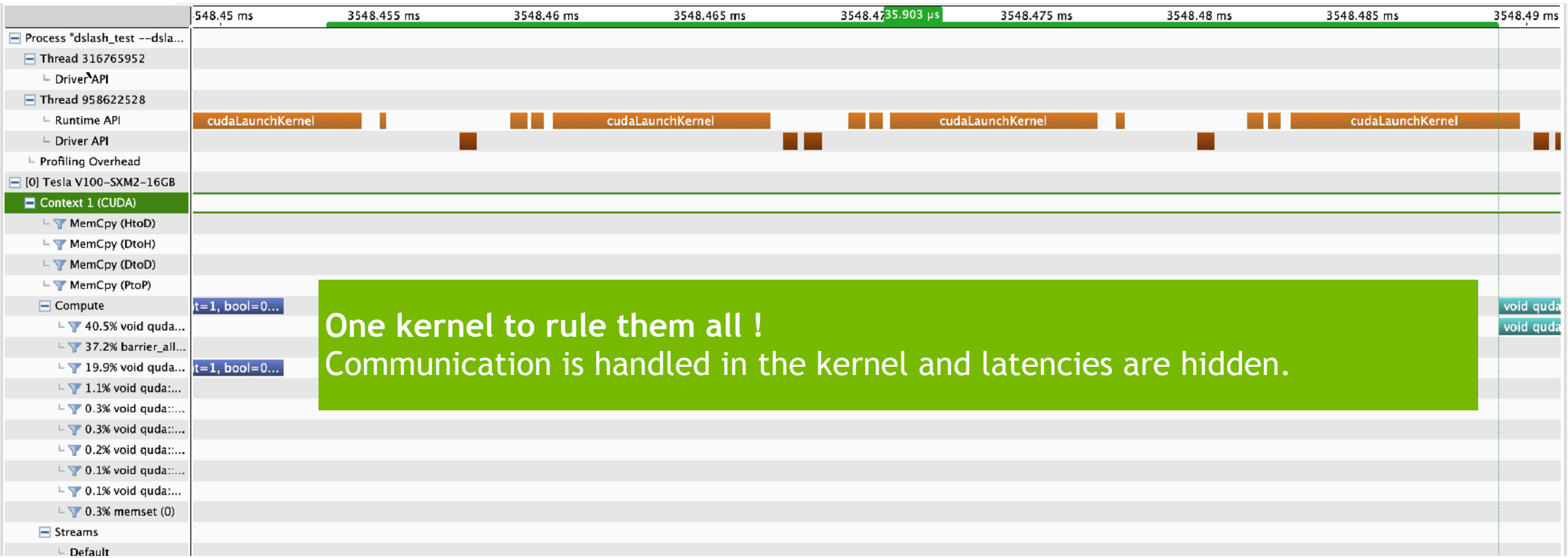
DGX SuperPOD (DGX2-H: 16 V100 (32GB), 8 EDR IB)



NVSHMEM OUTLOOK



NVSHMEM OUTLOOK



IN THE WILD

CHROMA HMC MULTIGRID

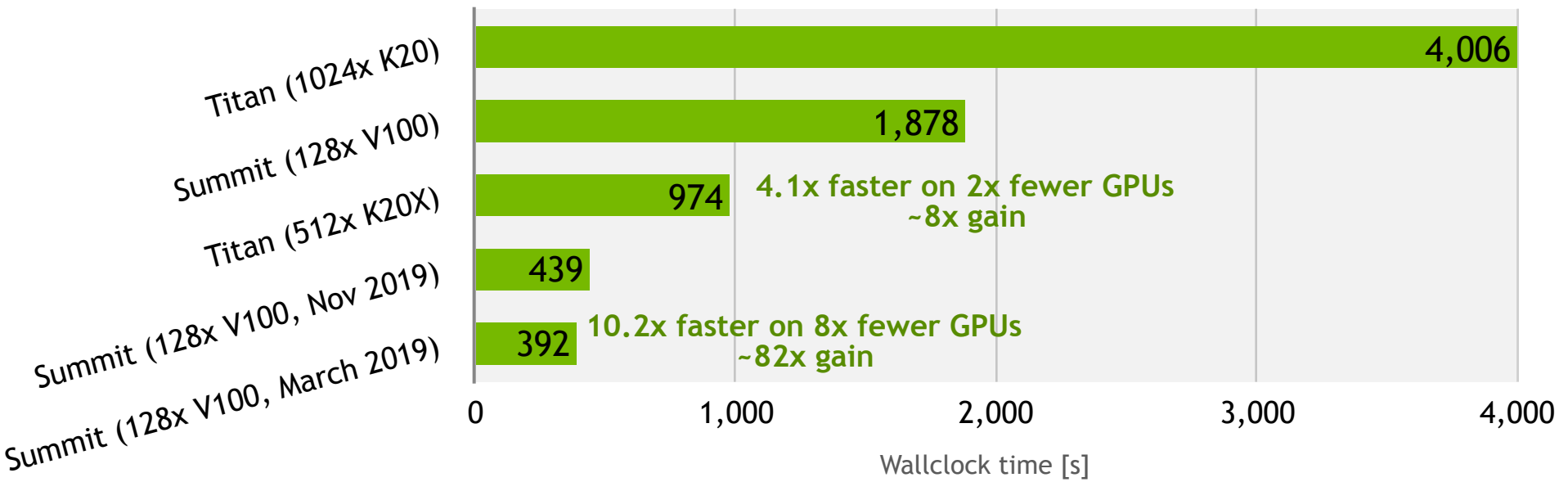
HMC typically dominated by solving the Dirac equation, but

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

Multigrid setup must run at speed of light

- Reuse and evolve multigrid setup where possible
- Use the same null space for all masses (setup run on lightest mass)
- Evolve null space vectors as the gauge field evolves (Lüscher 2007)
- Update null space when the preconditioner degrades too much on lightest mass

CHROMA HMC-MG ON SUMMIT



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

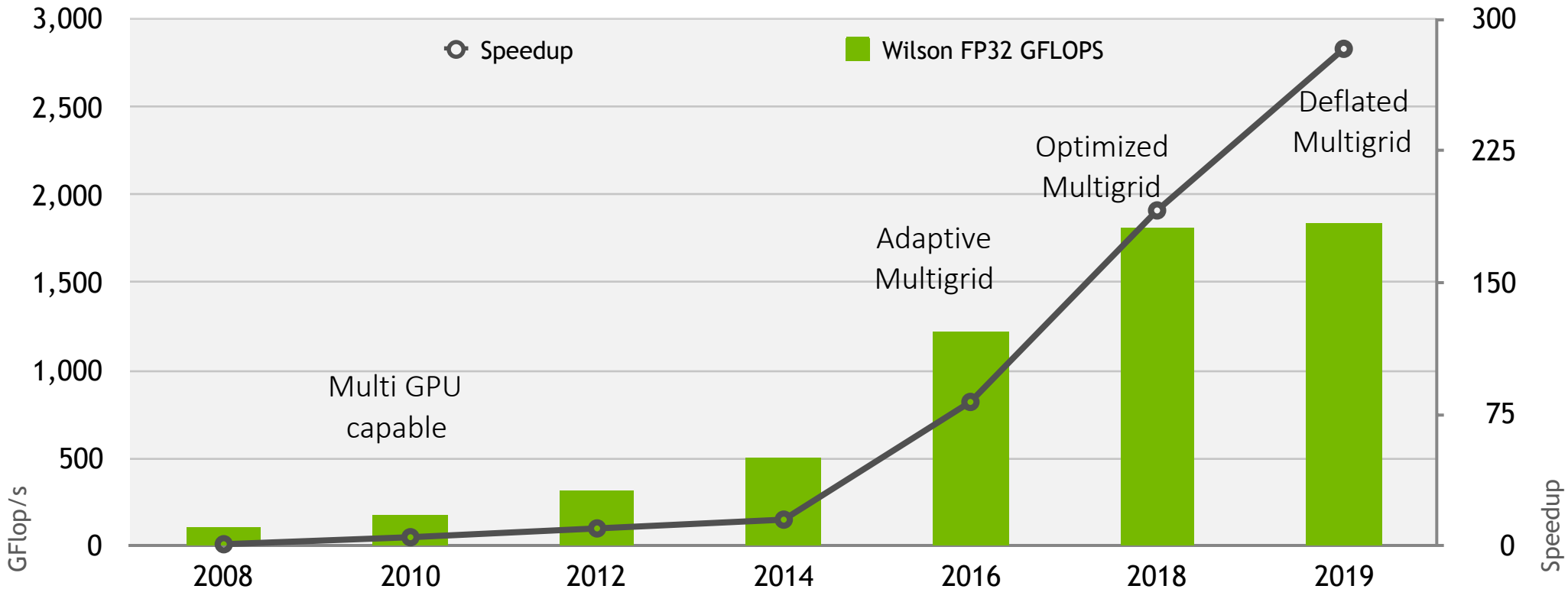
From Titan running 2016 code to Summit running 2019 code we see >82x speedup in HMC throughput

Multiplicative speedup coming from machine and algorithm innovation

Highly optimized multigrid for gauge field evolution

NODE PERFORMANCE OVER TIME

Multiplicative speedup through software and hardware



Time to solution is measured time to solution for solving the Wilson operator against a random source on a 24x24x24x64 lattice, β=5.5, M_π= 416 MeV. One node is defined to be 3 GPUs

QUDA - LATTICE QCD ON GPUS

Widely used for Lattice QCD applications on GPUs

State of the art solvers using mixed precision

- Multigrid
- Deflation
- Block-Krylov solver

All components for gauge field evolution

Portable high-performance kernels through auto-tuning and careful optimization
Tuned Multi-GPU scaling

- GPU centric communication with NVSHMEM takes CPU limitations out

Multiplicative speedup from hardware and software: more science


NVIDIA