Numerical Lattice QCD Simulations on Titan

Bálint Joó

Jefferson Lab, Newport News, VA, USA

OLCF User Group Meeting 2015
Oak Ridge National Laboratory

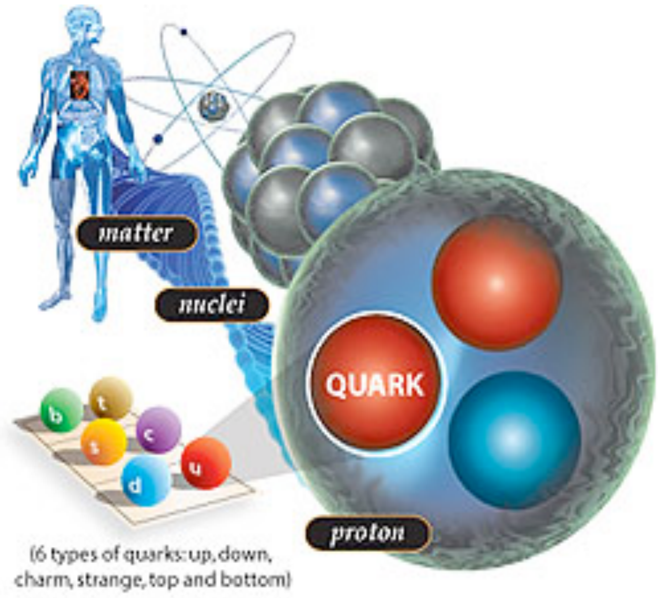
June 24, 2015

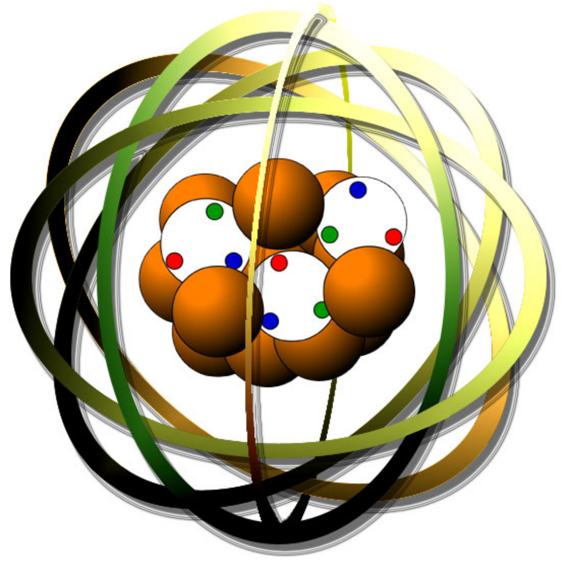
Contents:

- QCD On a Lattice
- Computational Workflow & High Level Algorithms
- Optimizing Solvers for GPUs on Titan
- Fighting Amdahl's Law: Moving all of the code to the GPU
- Future Perspectives

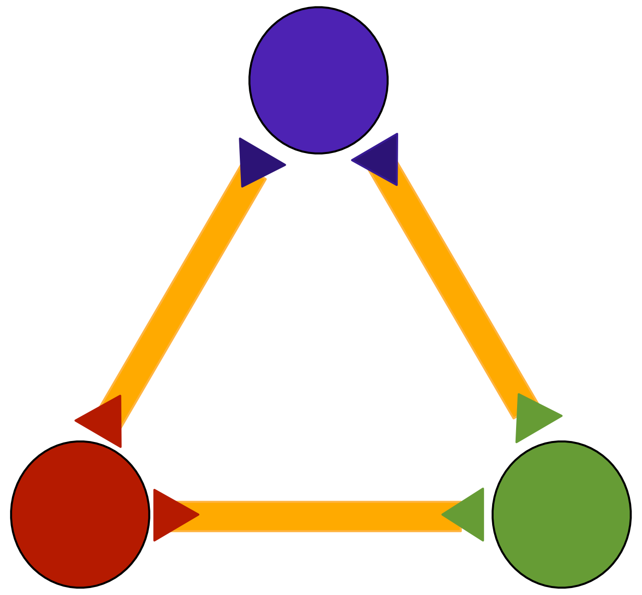
Quantum Chromodynamics (QCD)

- QCD is the theory of the strong nuclear force
  - matter is made of quarks, interacting by exchanging gluons
  - quarks and gluons carry color charges
  - we can only ever see ‘color neutral’ combinations
- Quarks make up protons, neutrons and mesons
- Residual strong force interactios hold together nuclei
- QCD is a quantum-field theory

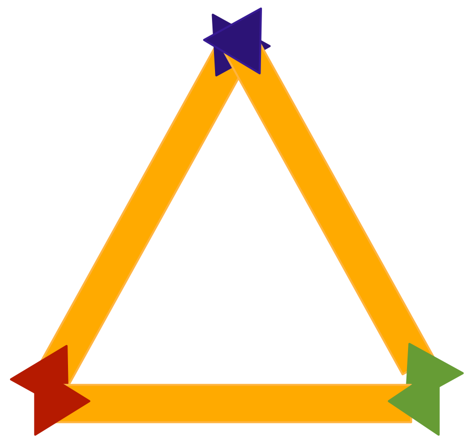




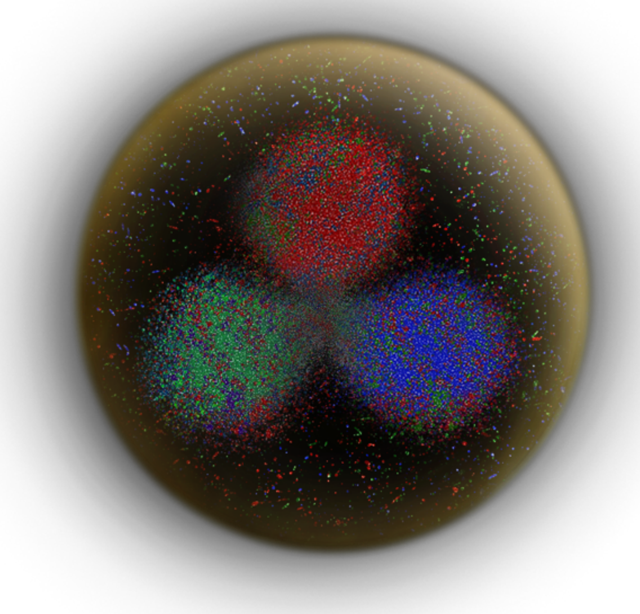
meson: quark-antiquark pair



baryon: 3 quarks



glueball: 0 quarks
only gluons

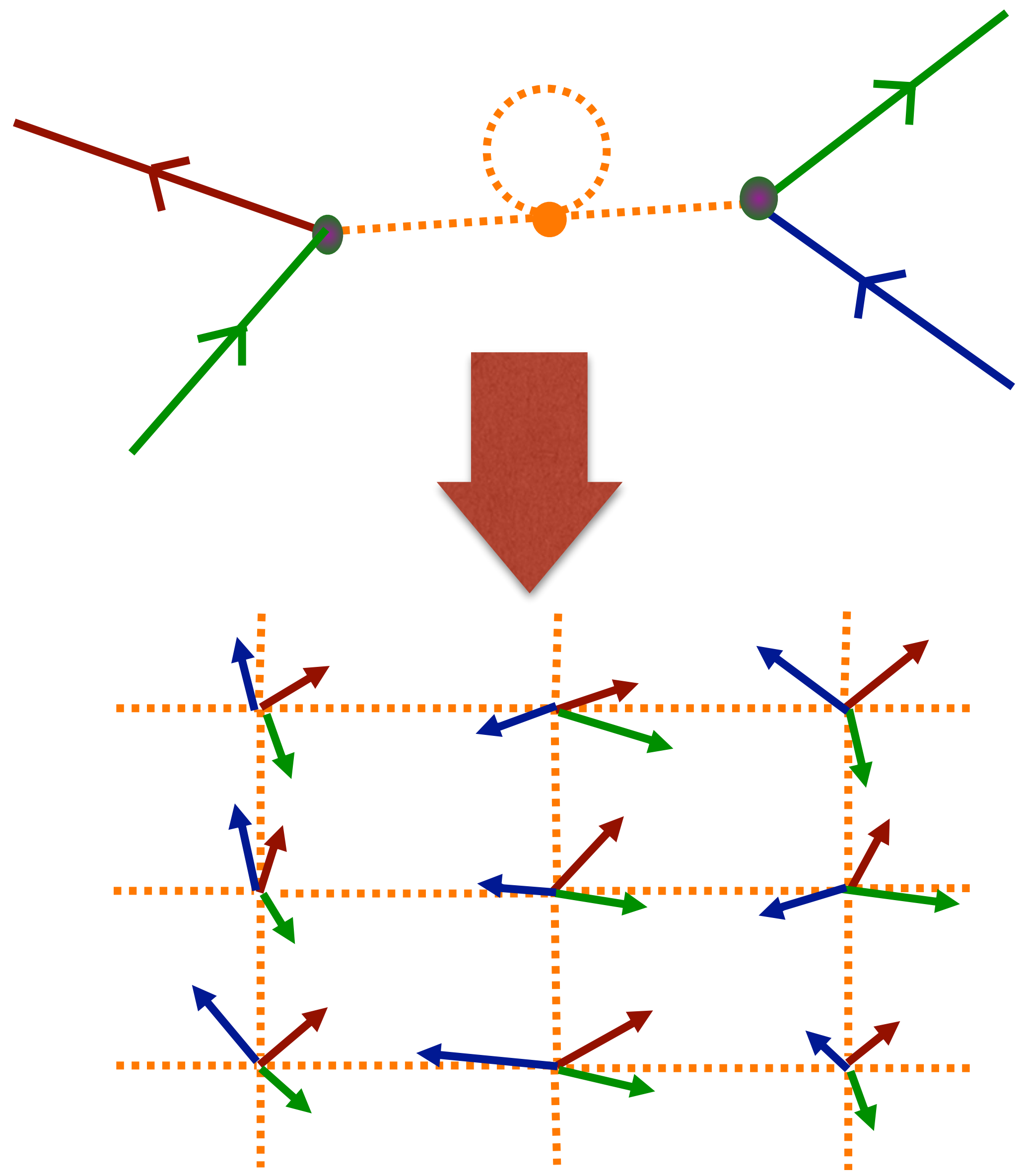


From Continuum to the Lattice

- Replace continuum space time by 4D Lattice

$$\langle \mathcal{O} \rangle = \frac{1}{\mathcal{Z}} \int \mathcal{D}A \, \mathcal{D}\bar{\psi} \, \mathcal{D}\psi \; \mathcal{O} \; e^{-S(A,\bar{\psi},\psi)}$$

- Discretize quark fields onto lattice *sites* and gluon fields onto lattice links
  - QCD local gauge symmetry: different color bases on each site
  - 3x3 matrices on links act as “parallel transporters” along links
  - rotate color basis at one site into that on another site.



- use finite differences for derivatives
- Rotate to: ’imaginary’ time (t ⇒ it)
- Functional integrals become ‘regular’ integrals

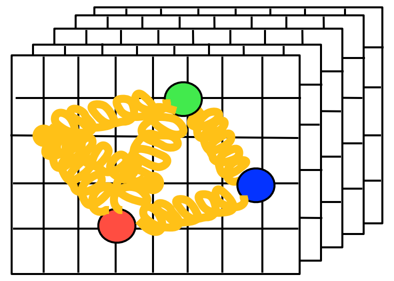
$$\langle \mathcal{O} \rangle = \frac{1}{\mathcal{Z}} \int \prod_{\text{all links}} dU \prod_{\text{all sites}} d[\bar{\psi}, \psi] \; \mathcal{O} \; e^{-S(U,\bar{\psi},\psi)}$$

- Evaluate integrals with importance sampling Monte Carlo method

$$\rightarrow \bar{\mathcal{O}} = \frac{1}{N} \sum_{N} \mathcal{O}(U)$$

LQCD Calculation Workflow

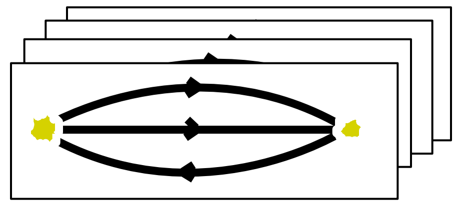
Gauge Configurations



Gauge Generation (Monte-Carlo)

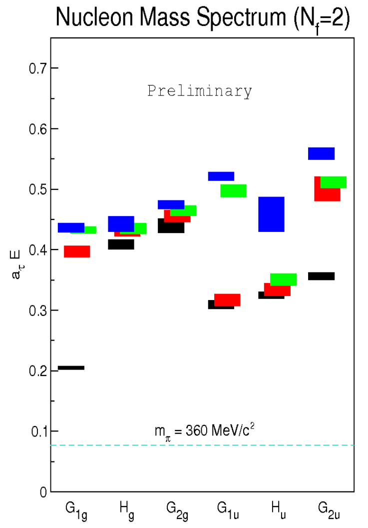
Measurement (Propagators & Contractions)

Propagators, Correlation Functions



Fitting and Statistics

Bulava *et al*, PRD 79, 034505 (2009)



Physics Result

Compute: {U}

Compute: $\{\mathcal{O}(U)\}$

Compute: $\bar{\mathcal{O}} = \frac{1}{N}\sum_{N}\mathcal{O}(U)$

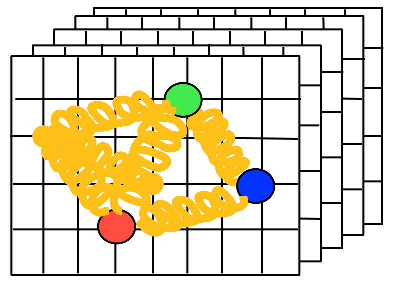
& Extract Physical Information

LQCD Calculation Workflow

Bulava *et al*, PRD 79, 034505 (2009)

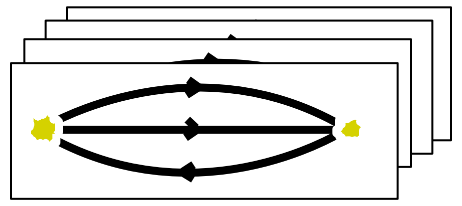
Gauge Generation
(Monte-Carlo)

Gauge Configurations

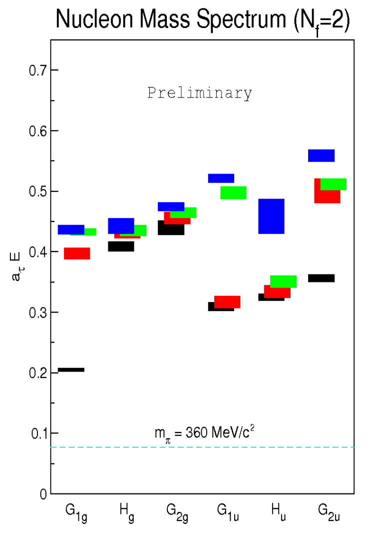


Measurement
(Propagators &
Contractions)

Propagators, Correlation Functions



Fitting and Statistics



Physics
Result

Compute: {U}

Compute: $\{\mathcal{O}(U)\}$

Compute: $\bar{\mathcal{O}} = \frac{1}{N}\sum_N \mathcal{O}(U)$

& Extract Physical Information

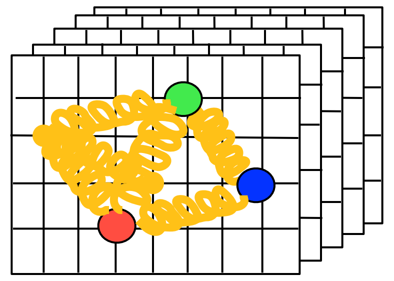
LQCD Calculation Workflow

Strong Scaling Challenge

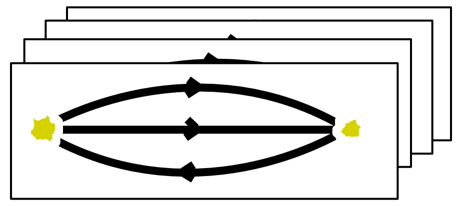
Throughput Challenge

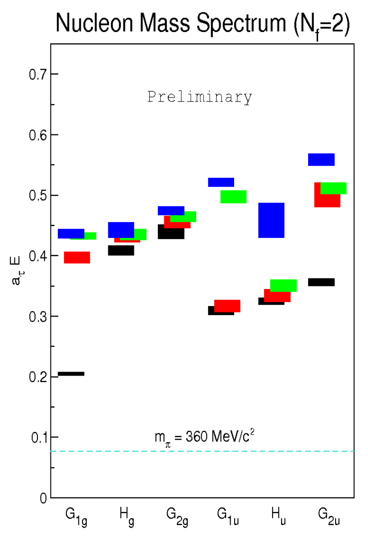
Bulava *et al*, PRD 79, 034505 (2009)

Gauge Configurations



Propagators, Correlation Functions





Gauge Generation (Monte-Carlo)

Measurement (Propagators & Contractions)

Fitting and Statistics

Physics Result

Compute: {U}

Compute: $\{\mathcal{O}(U)\}$

Compute: $\bar{\mathcal{O}} = \frac{1}{N}\sum_N \mathcal{O}(U)$

& Extract Physical Information

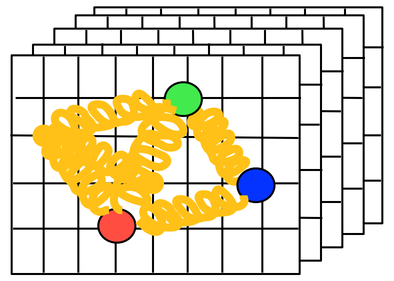
LQCD Calculation Workflow

Strong Scaling Challenge

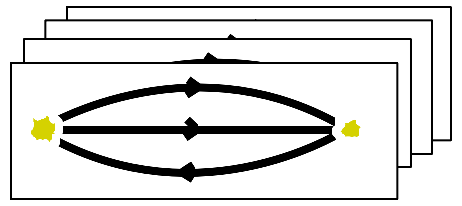
Throughput Challenge

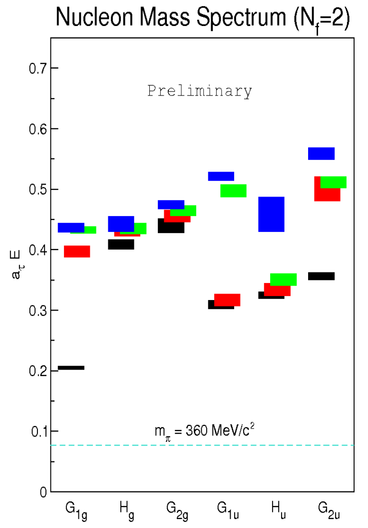
Bulava *et al*, PRD 79, 034505 (2009)

Gauge Configurations



Propagators, Correlation Functions





Gauge Generation (Monte-Carlo)

Measurement (Propagators & Contractions)

Fitting and Statistics

Physics Result

Compute: {U}

Compute: $\{\mathcal{O}(U)\}$

Compute: $\bar{\mathcal{O}} = \frac{1}{N}\sum_N \mathcal{O}(U)$

Community INCITE

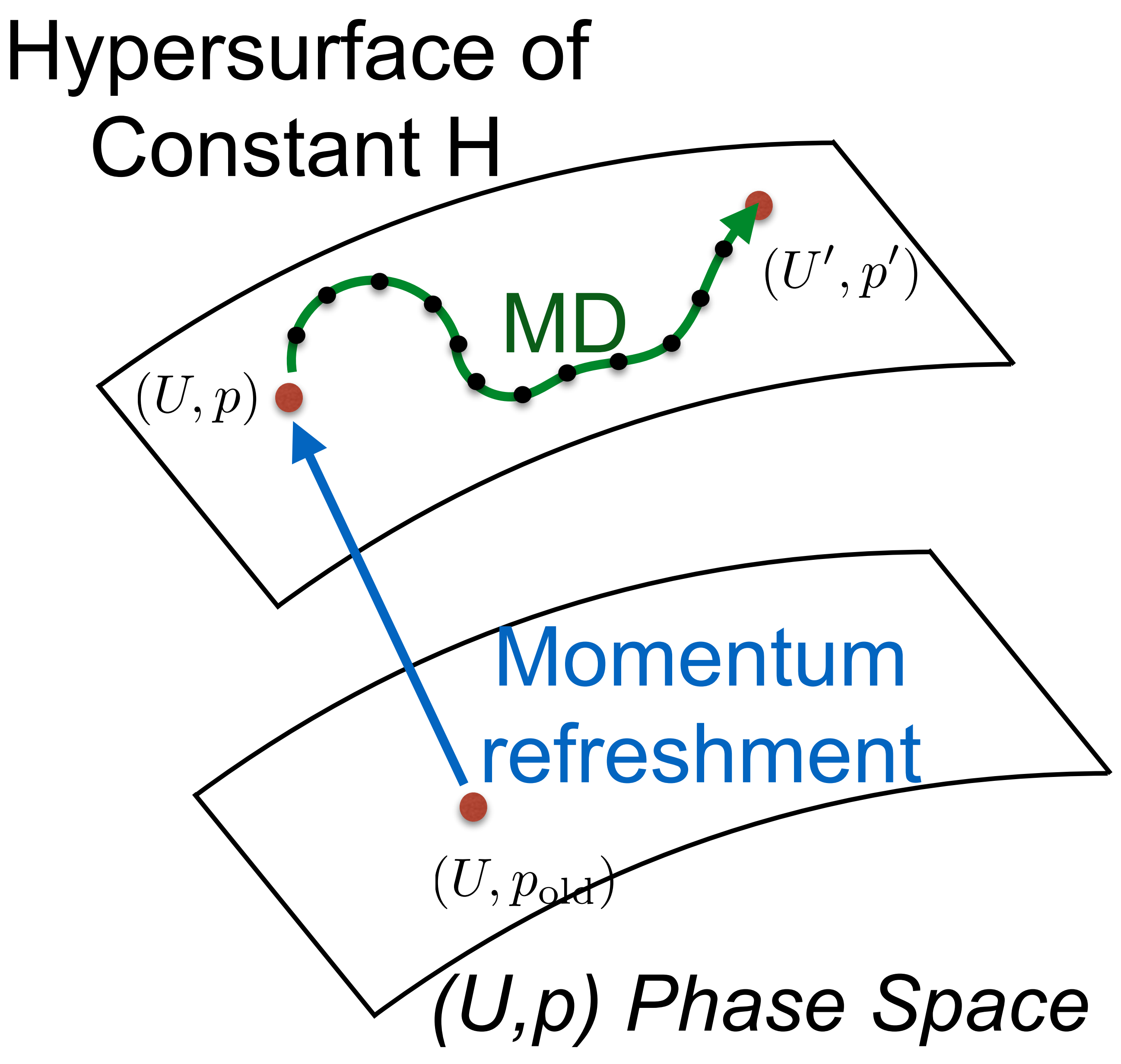
Individual ALCC,
Other Allocations (e.g. NERSC)
USQCD cluster resources

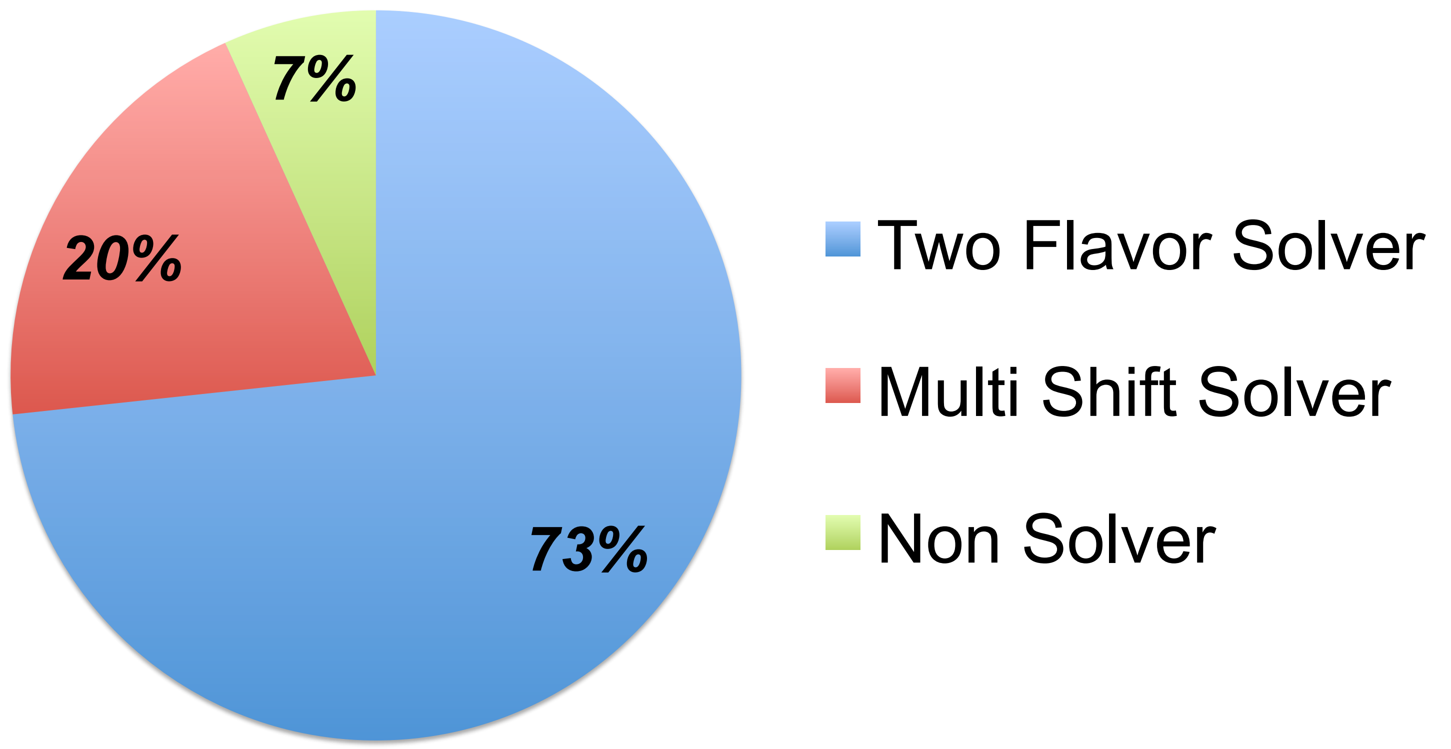
& Extract Physical Information

LQCD as a data driven science

Gauge Generation: Hybrid Monte Carlo

- Treat 'U' links as coordinates & define canonical momenta
- Extend Action 'S' to Hamiltonian 'H'
- Interleave:
  - momentum & pseudofermion refreshment
  - Hamiltonian Molecular Dynamics
  - Metropolis Accept/Reject
- Energies & MD Force:
  - Need to solve Dirac Equation: $M^{\dagger}M\, x = \phi$
  - Physical Mass run: ***93% of time in solvers***
    - …and this is after acceleration



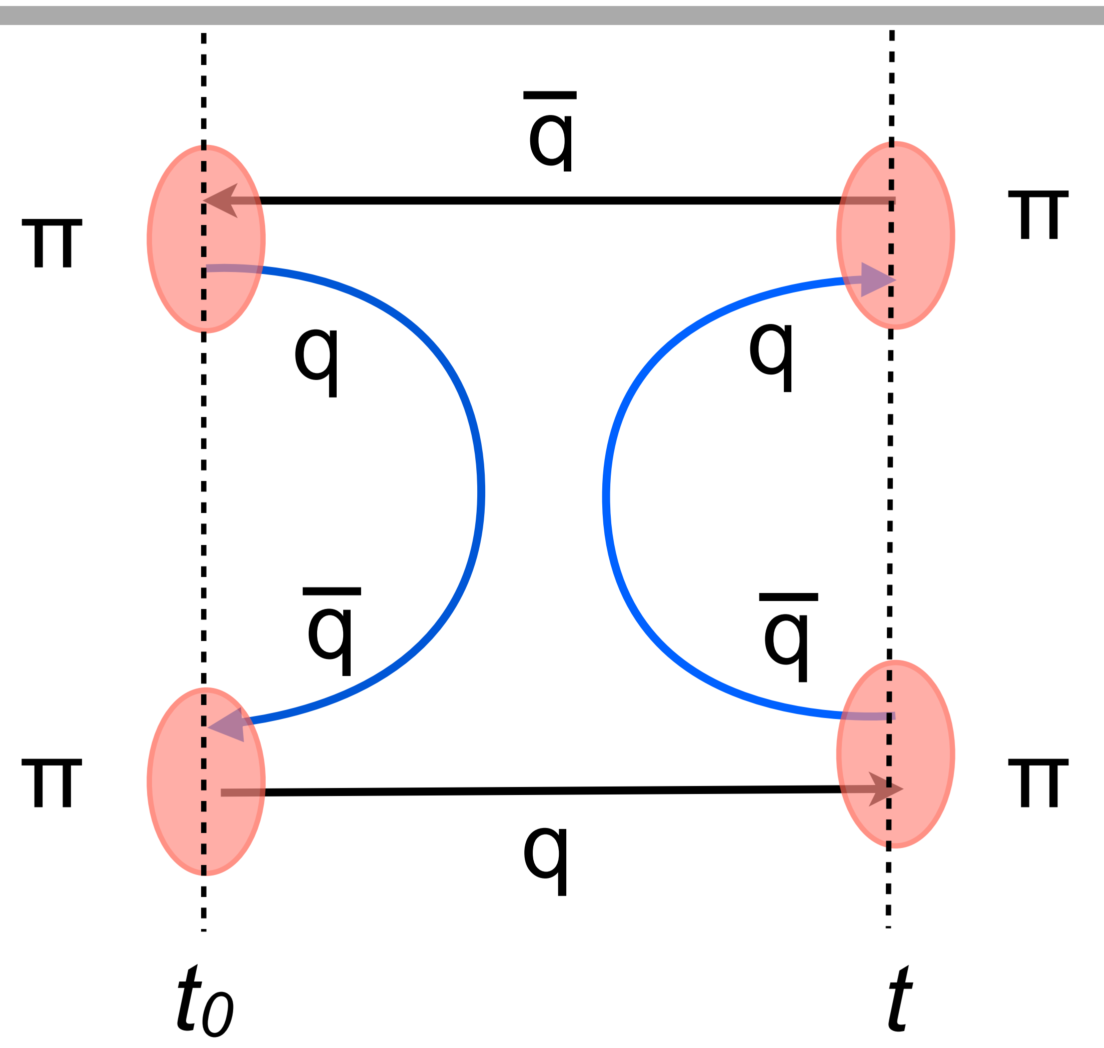


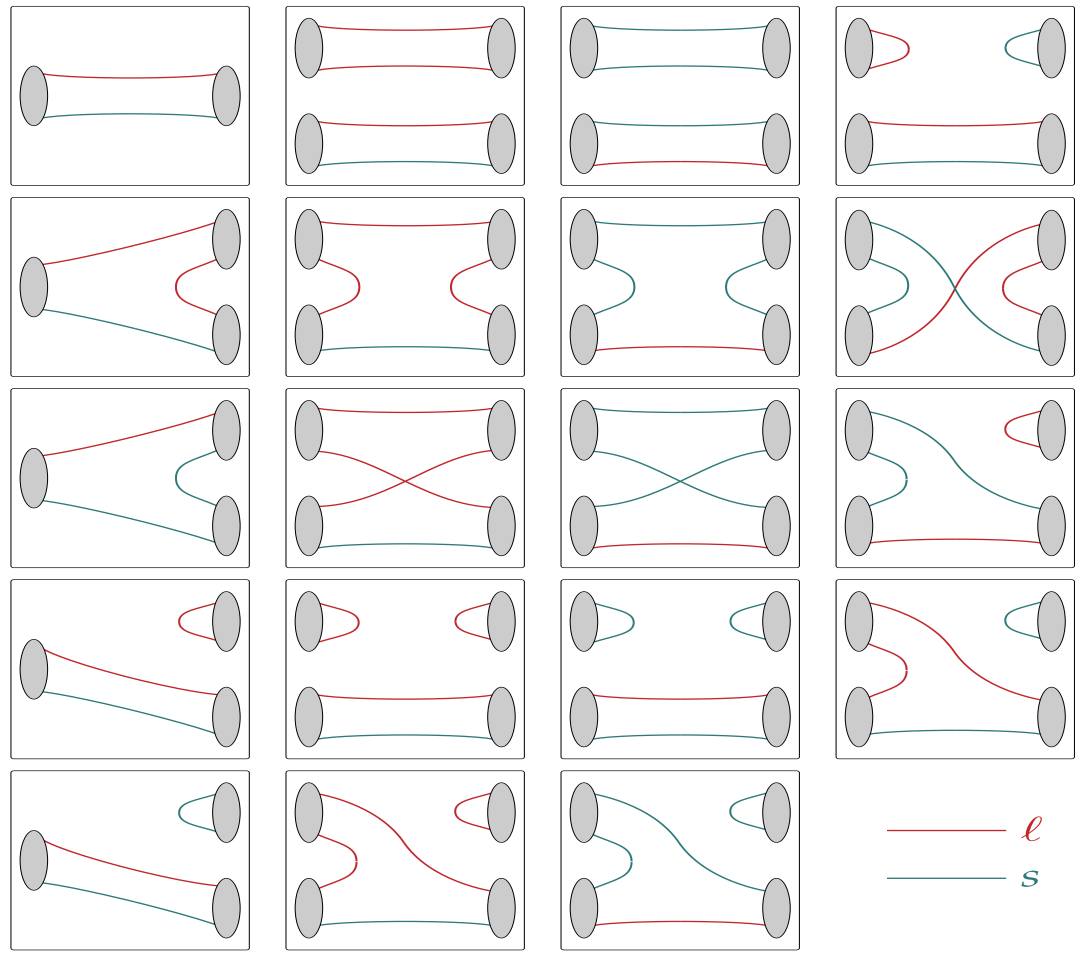
Propagators & Contractions

- Propagator G(**x**,t; **y**,t_0) from a 'source' S(**y**,t_0) is solution of the Dirac Equation:

$$M(\mathbf{y},t_0\,;\,\mathbf{x},t)\ \ G(\mathbf{x},t;\,\mathbf{y},t_0) = S(\mathbf{y},t_0)$$

- Total number of solves for annihiliation (blue) lines:
  - \# t-slices x #spins x # of sources x 2 quark masses
  - **786,432 solves per configuration** for the 32^3x256 dataset
  - solves **are independent** of each other => throughput challenge
- Many Wick Contractions: O(10,000) depending process
  - Graphs are **independent of each other,** but can **share sub-graphs**
  - I/O challenge reading propagators for all contractions
  - Want to reduce redundant I/O and contractions: Robert's redstar code



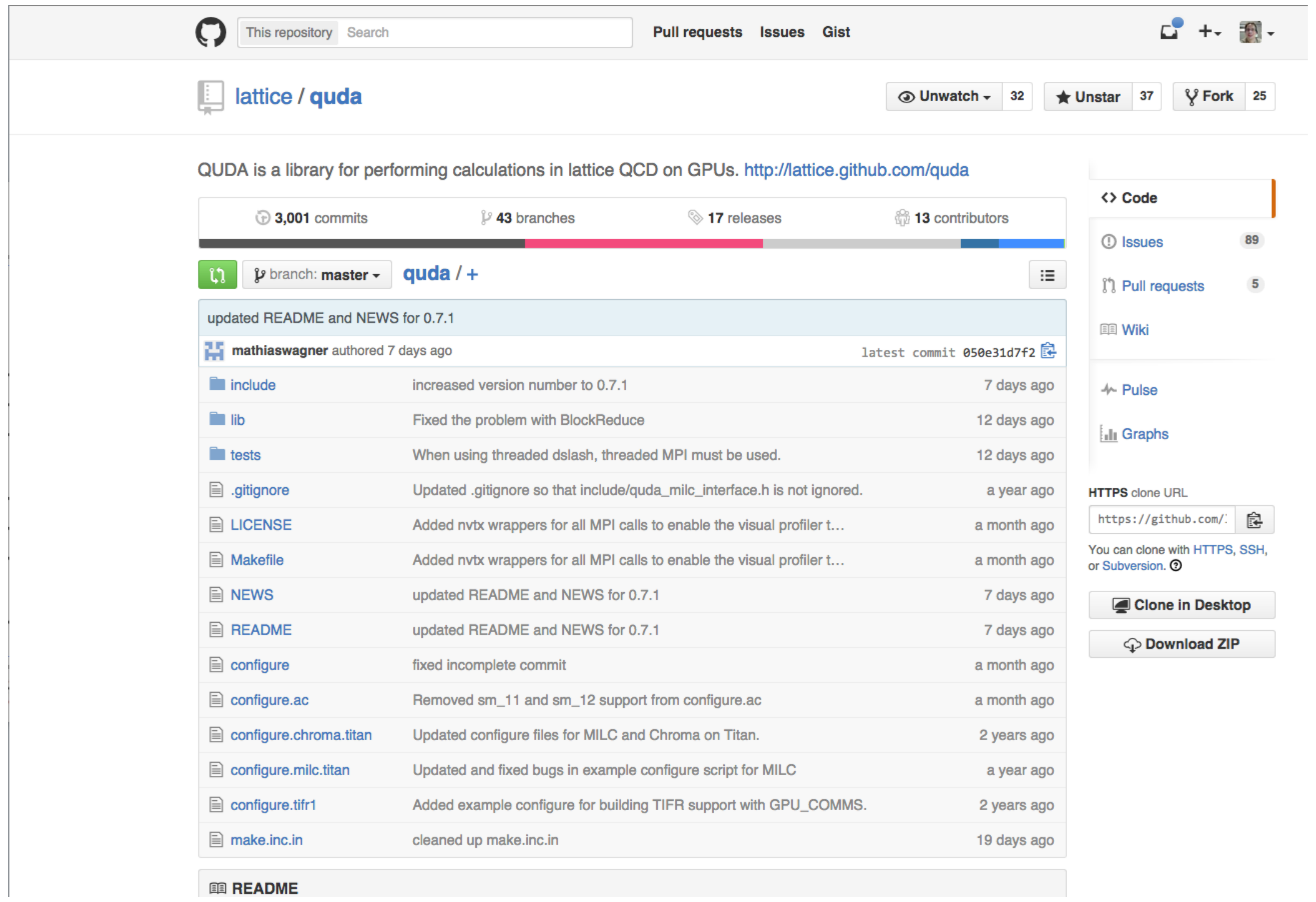


I=1/2 K*π arXiv:1406.4158

QUDA: Optimized QCD solvers

- QUDA is a library of optimized LQCD components (inc. solvers) for GPUs
- Community Library
  - started at Boston University
  - original developers have moved to NVIDIA
  - now QUDA is a community developed library, supported by NVIDIA
- Supports a variety of LQCD formulations & Codes
  - Wilson Clover
  - Improved Staggered (e.g. HiSQ for MILC)
  - Chiral formulations (Domain Wall & variants)
  - Various parts Interfaced to Chroma, MILC, CPS, BQCD, …
- Development 'playground' for GPU LQCD algoritms
  - Deflated solvers, Multi-Grid, Communications avoiding solvers etc.

http://github.com/lattice/quda.git



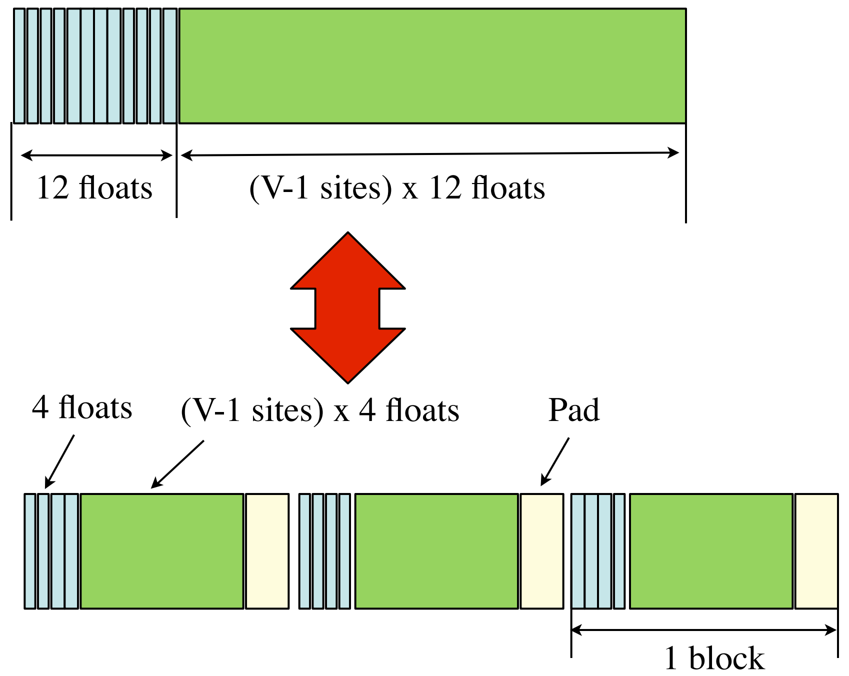
M.A.Clark, R. Babich, K. Barros, R.C. Brower, C.Rebbi
"Solving Lattice QCD Systems of Equations using mixed precision solvers on GPUs", Comput. Phys. Commun. 181.1517

R. Babich, M.A. Clark, B. Joo, SC'10 Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis

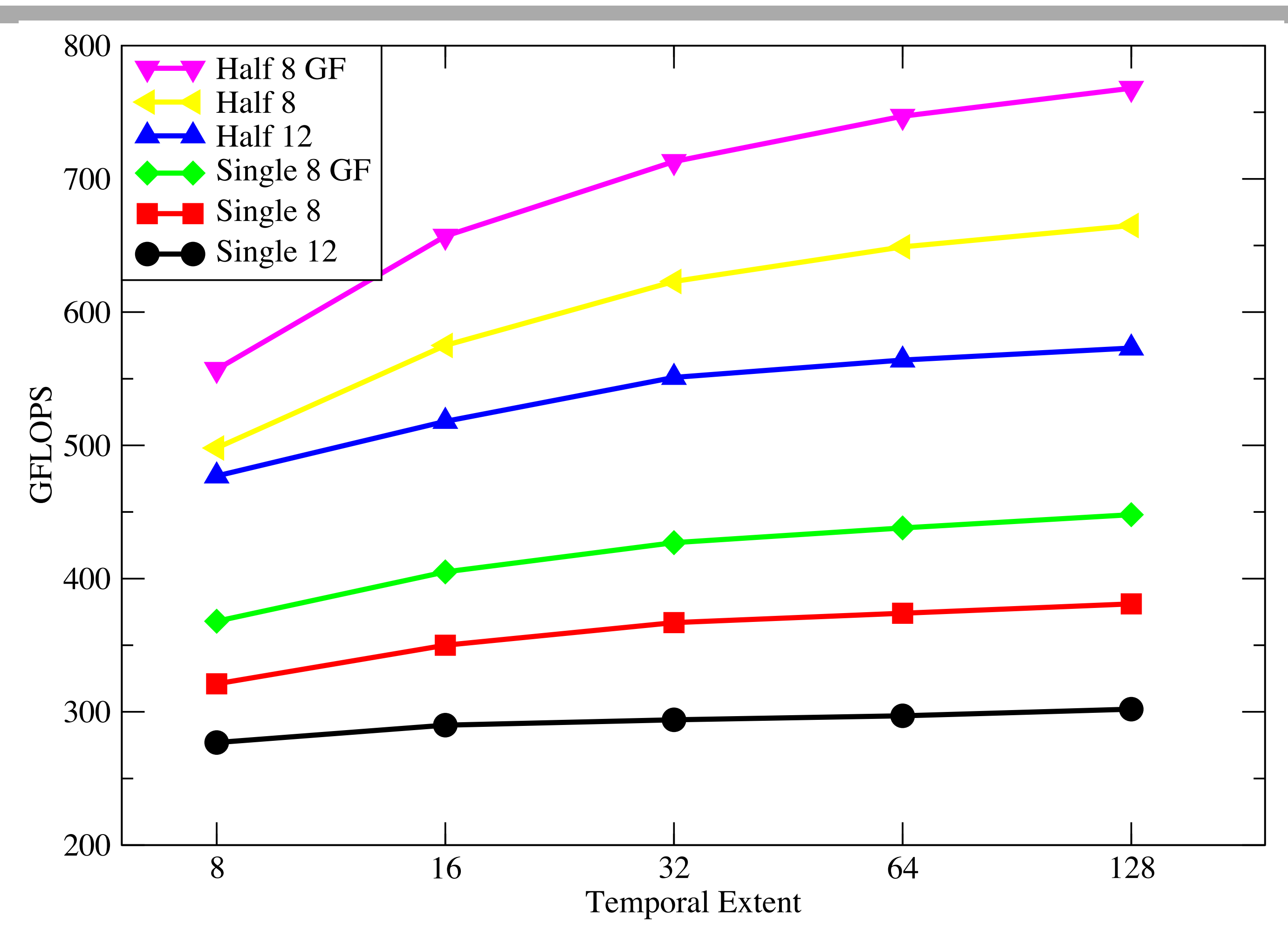
R. Babich, M.A. Clark, B. Joo, G.Shi, R.C. Brower, S. Gottlieb: SC'11 Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis

QUDA Optimizations

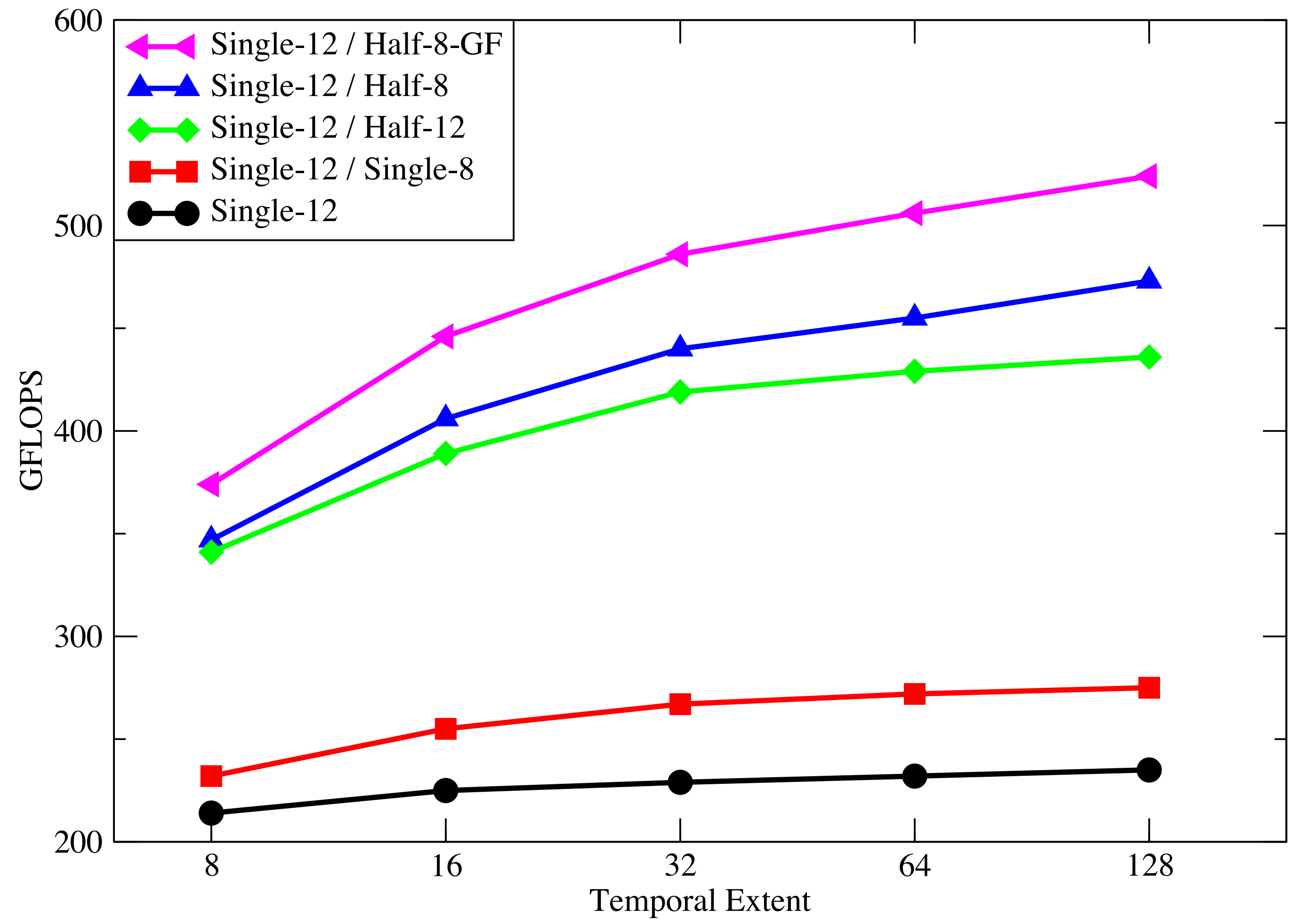
- The most predominantly used LQCD kernels are finite difference stencils and are memory bandwidth bound
- Memory Bandwidth Optimizations:
  - Improve memory performance: read/write coalescing friendly data layout
  - Aggressive use of reduced precision (inc. 16-bit precision)
    - Reliable Updates (BiCGStab) & Iterative Refinement
    - Reduced Precision Preconditioners
  - Domain Specific Optimizations:
    - “On the fly” compression: store SU(3) matrices as 8 or 12 real numbers (instead of full 18)



$$\begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ x & x & x \end{pmatrix} \xrightarrow{\begin{array}{l} \mathbf{a} = (a_1, a_2, a_3) \\ \mathbf{b} = (b_1, b_2, b_3) \\ \mathbf{c} = (\mathbf{a} \times \mathbf{b})^* \end{array}} \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix}$$



Wilson Dslash (finite difference kernel) Single GPU (K20X) V=24³xT

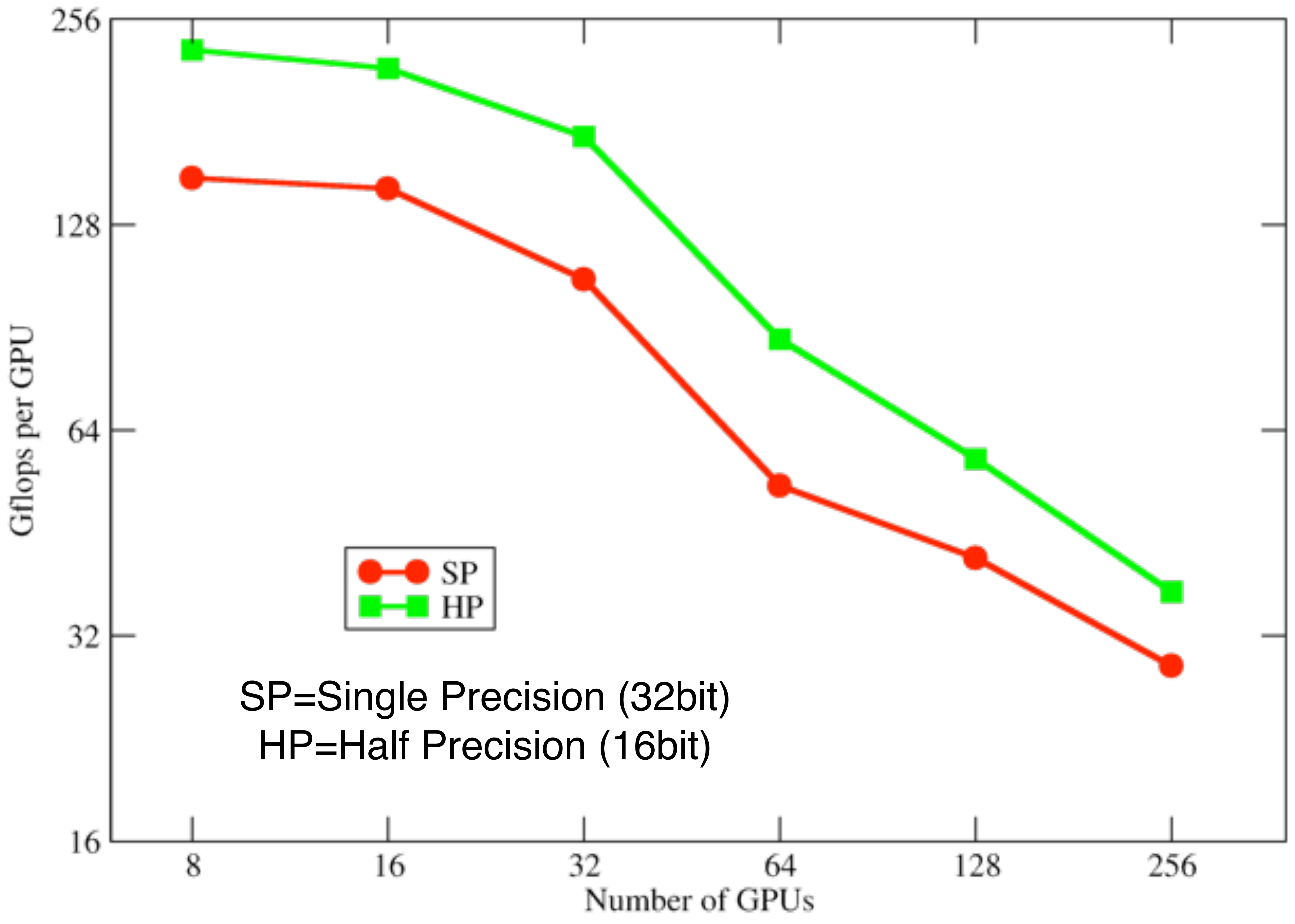


Wilson CG Solver Single GPU (K20X) V=24³xT

Plots from: “State of QUDA” presentation to USQCD, Oct, 2014, courtesy of K. Clark, NVIDIA

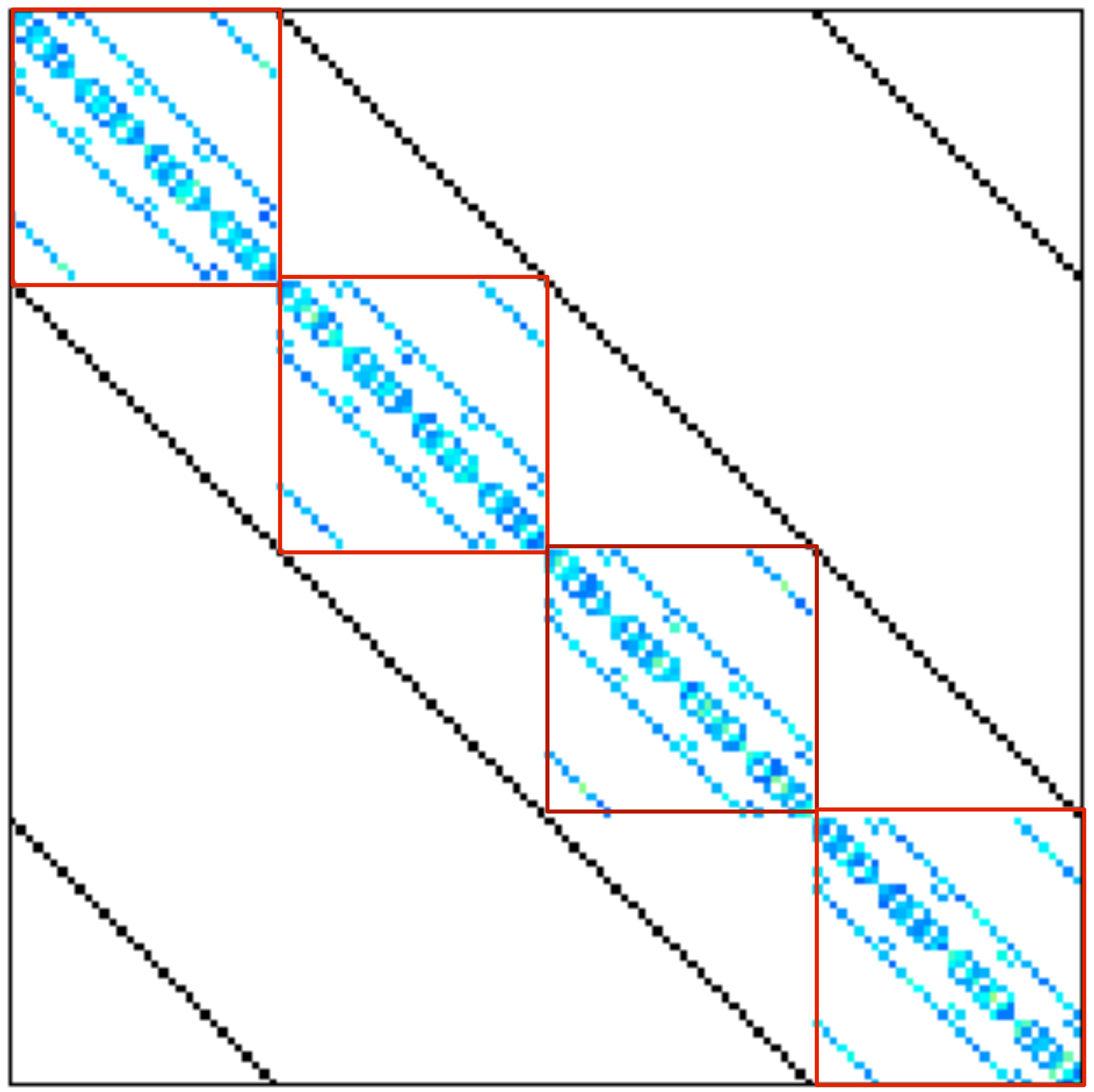
Scaling Bottleneck Example:

R.Babich, M. A. Clark, B. Joo, G. Shi, R. C. Brower, S. Gottlieb. "Scaling Lattice QCD Beyond 100 GPUs" Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11) page 70:1-70:11, New York, NY, USA, ACM (2011)



- One of the original findings was that strong scaling was difficult with accelerators
- Inter-device communications was considered to be the main bottleneck
- Mismatch of bandwidths
  - 8+8 GB on PCIe Gen2
  - ~150-170 GB/sec on device
- Spurred the development of Domain decomposed solvers…

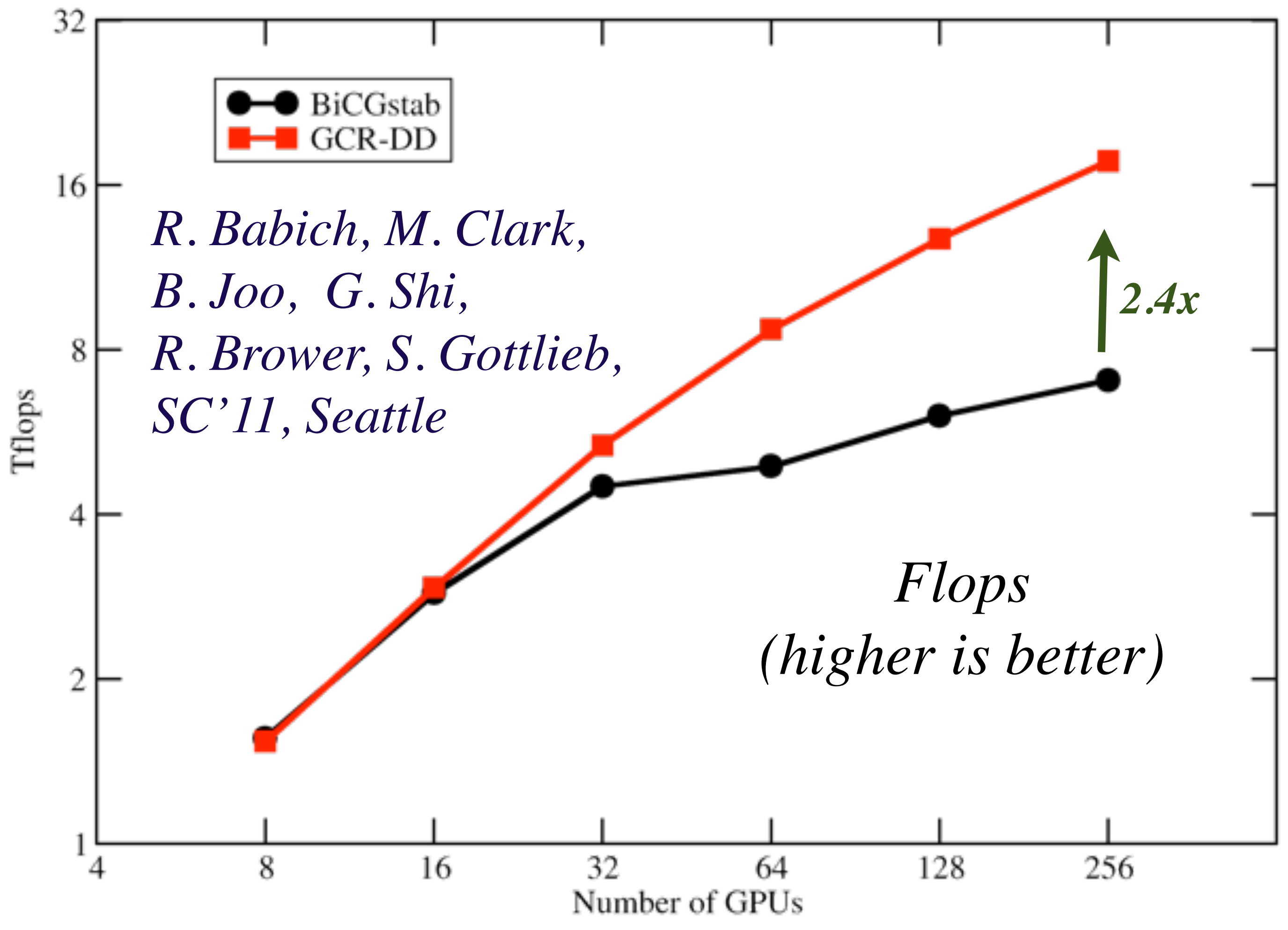
Architecture Awareness

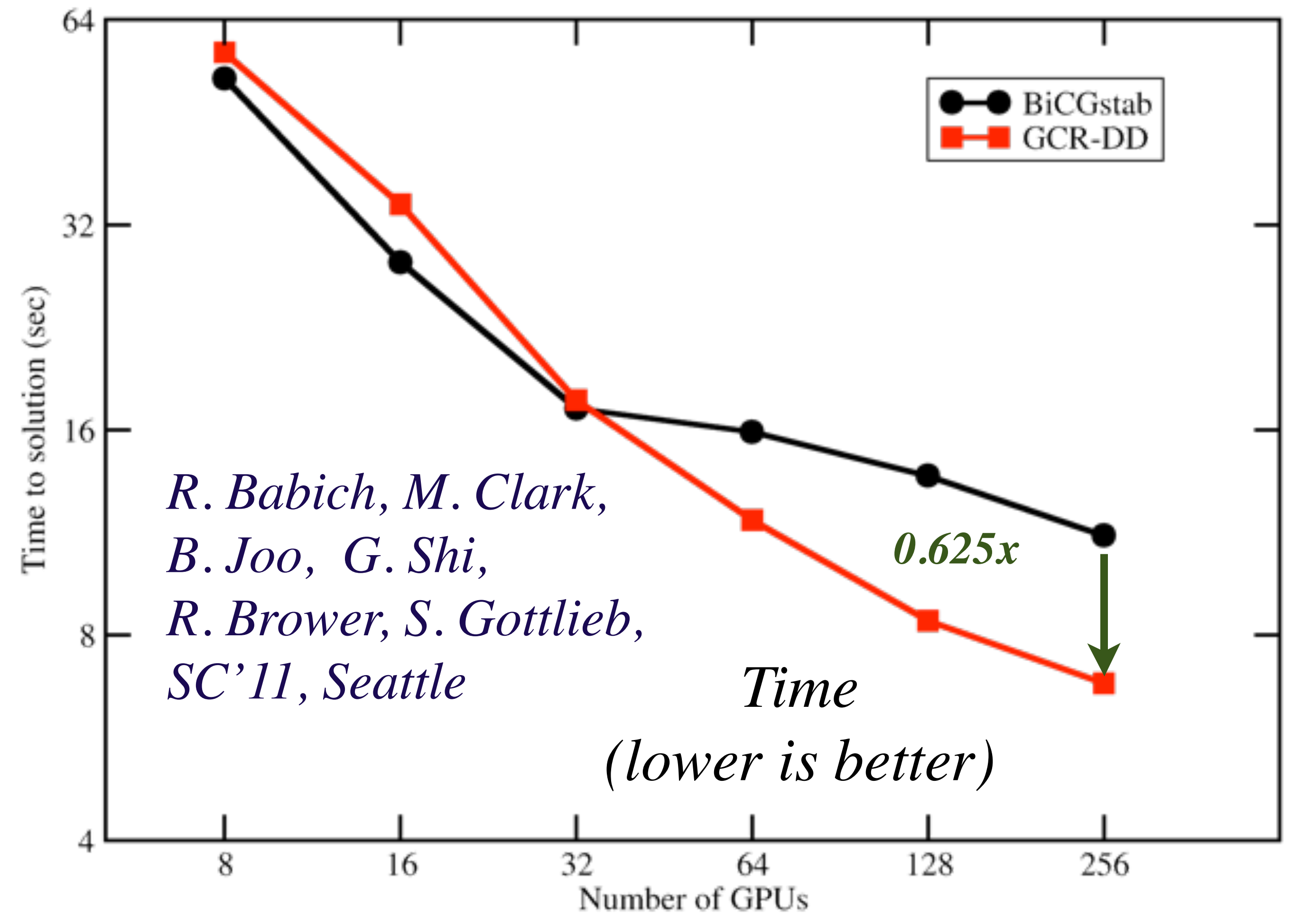


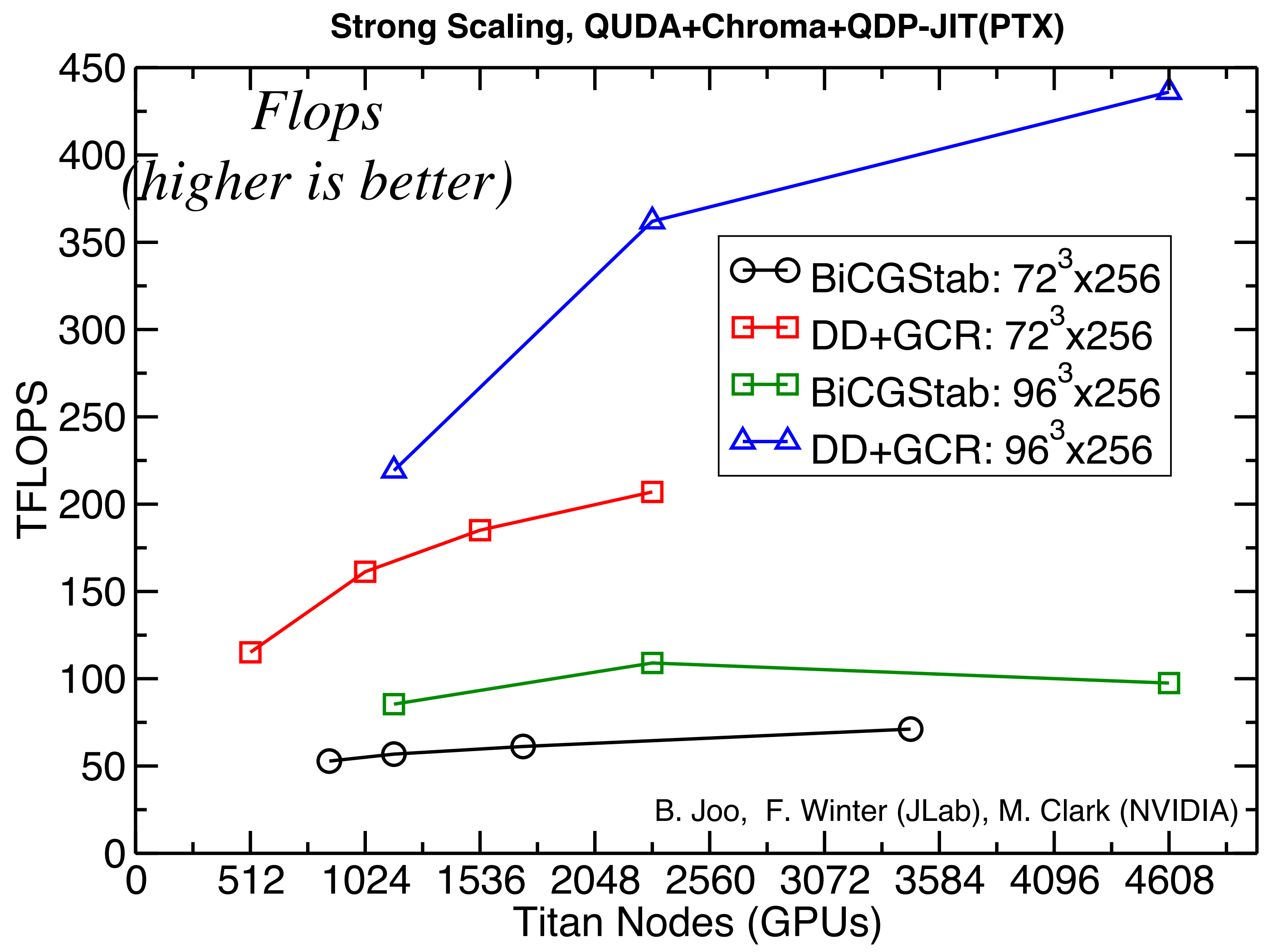
- Attempt to deal with communications bottleneck:
  - don’t communiate at all
- Use a block-diagonal operator as a ‘preconditioner’ in the solver
  - Inner-Outer Scheme: Approx. Invert Preconditioner with inner solver
  - Outer Scheme must tolerate variable preconditioner: GCR / FGMRES
  - GPUs do not need to communicate to apply operator
  - Inner solve could terminate on fixed iterations rather than residuum
- Arrange to spend most time in the preconditioner.
- But be aware:
  - block diagonal operator is a ‘wavelength filter’
  - outer scheme still needs to deal with long wavelength modes
- Example of interplay of architecture, algorithm, applied maths and physics.

Solver Performance

- DD-Solver started giving improved performance at around 32 GPUs (SC'11, using LLNL Edge Cluster)
  - this is problem size dependent
  - lots of FLOPs in DD-GCR algorithm, important to look at wallclock time gain also
- Solver performance on Titan
  - Large problems (72^3x256, 96^3x256)
  - DD-GCR can be scaled over 20% of Titan on the largest problem

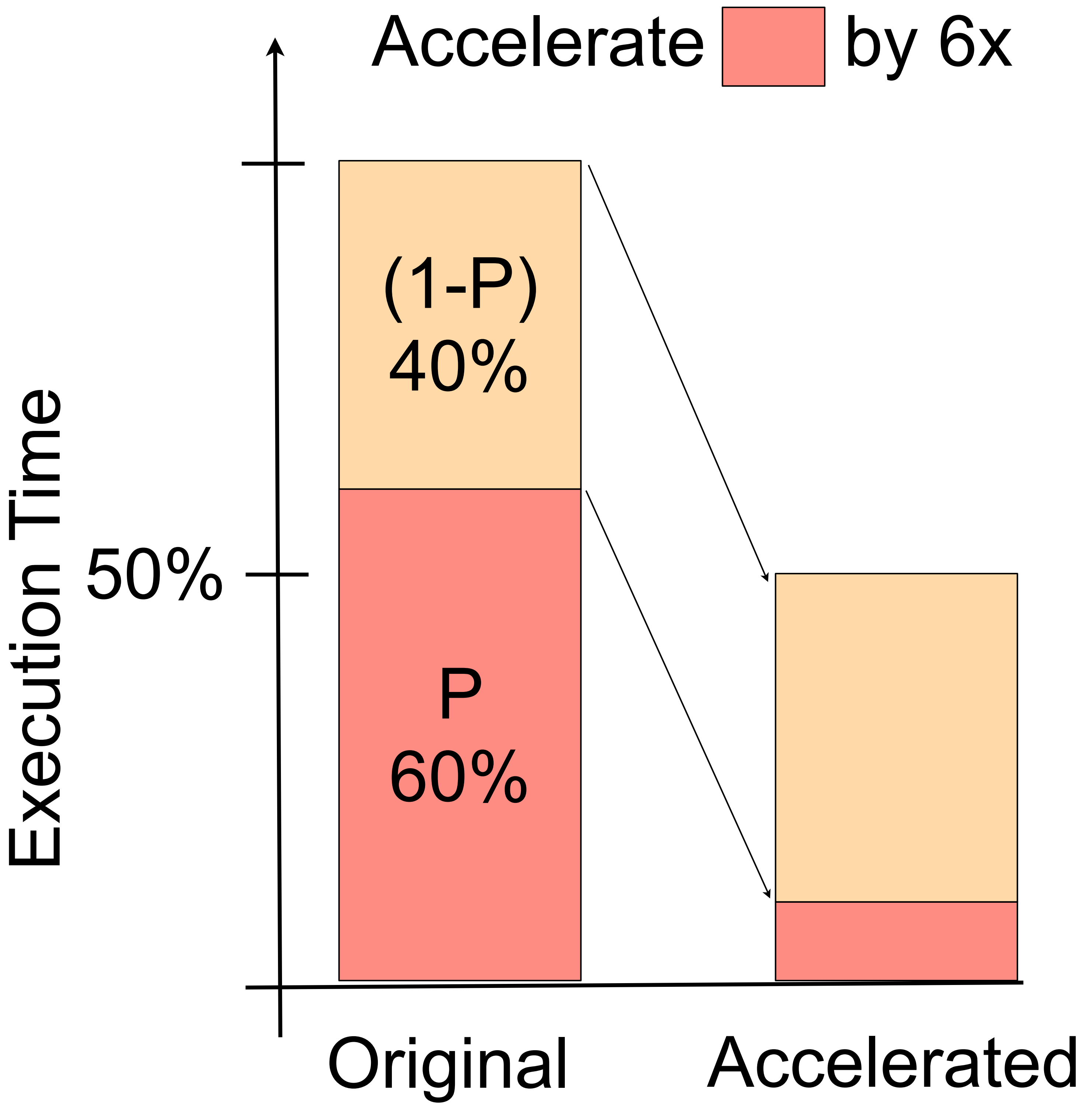




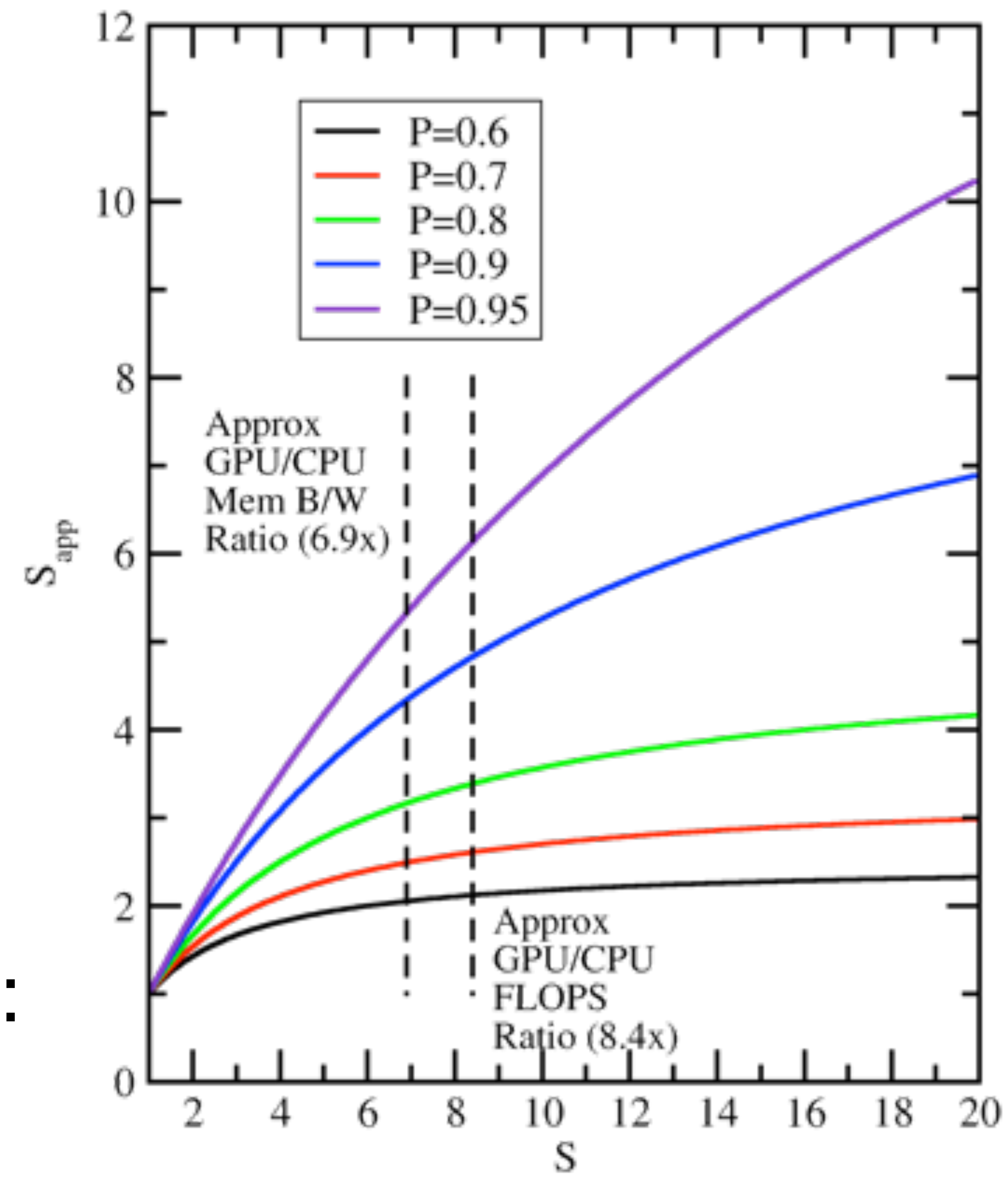


Non-Solver Performance: Amdahl's Law

$$S_{\mathrm{app}} = \frac{1}{(1-P) + \frac{P}{S}}$$



- Amdahl's Law
  - if you speed up portion P of your code, overall speedup limited by the1-P portion
  - E.g. speed up portion P by 6.9x
    - P=72% => S=2.6x
    - P=95% => S=5.3x
- Want to move as much code to GPU as possible
- Limitation on code in libraries:
  - the part of your code not in the library can become your limiter

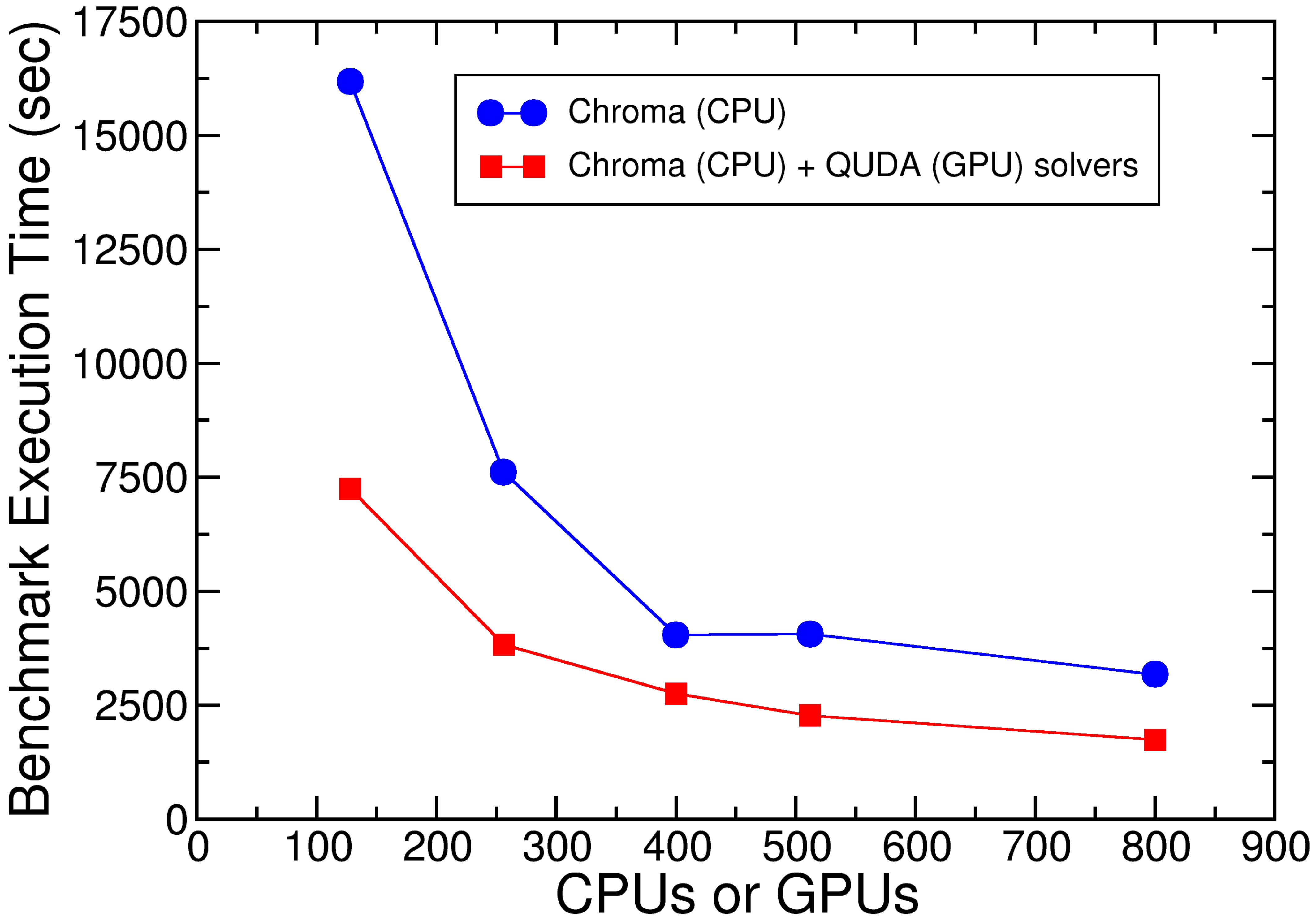


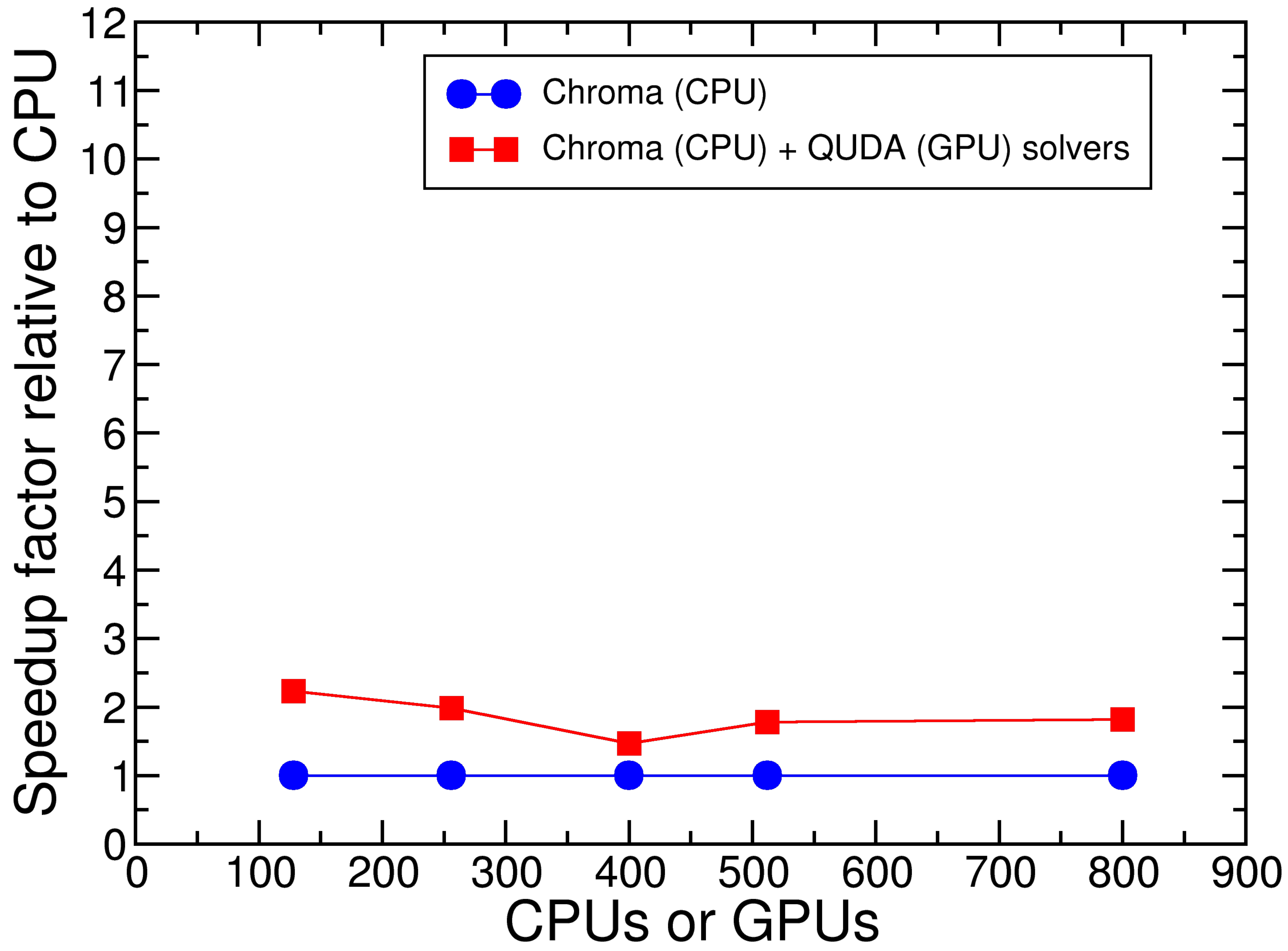
Non-Solver Performance

Wallclock Time
(lower is better)

Benchmarks from NCSA BlueWaters

Speedup
(higher is better)





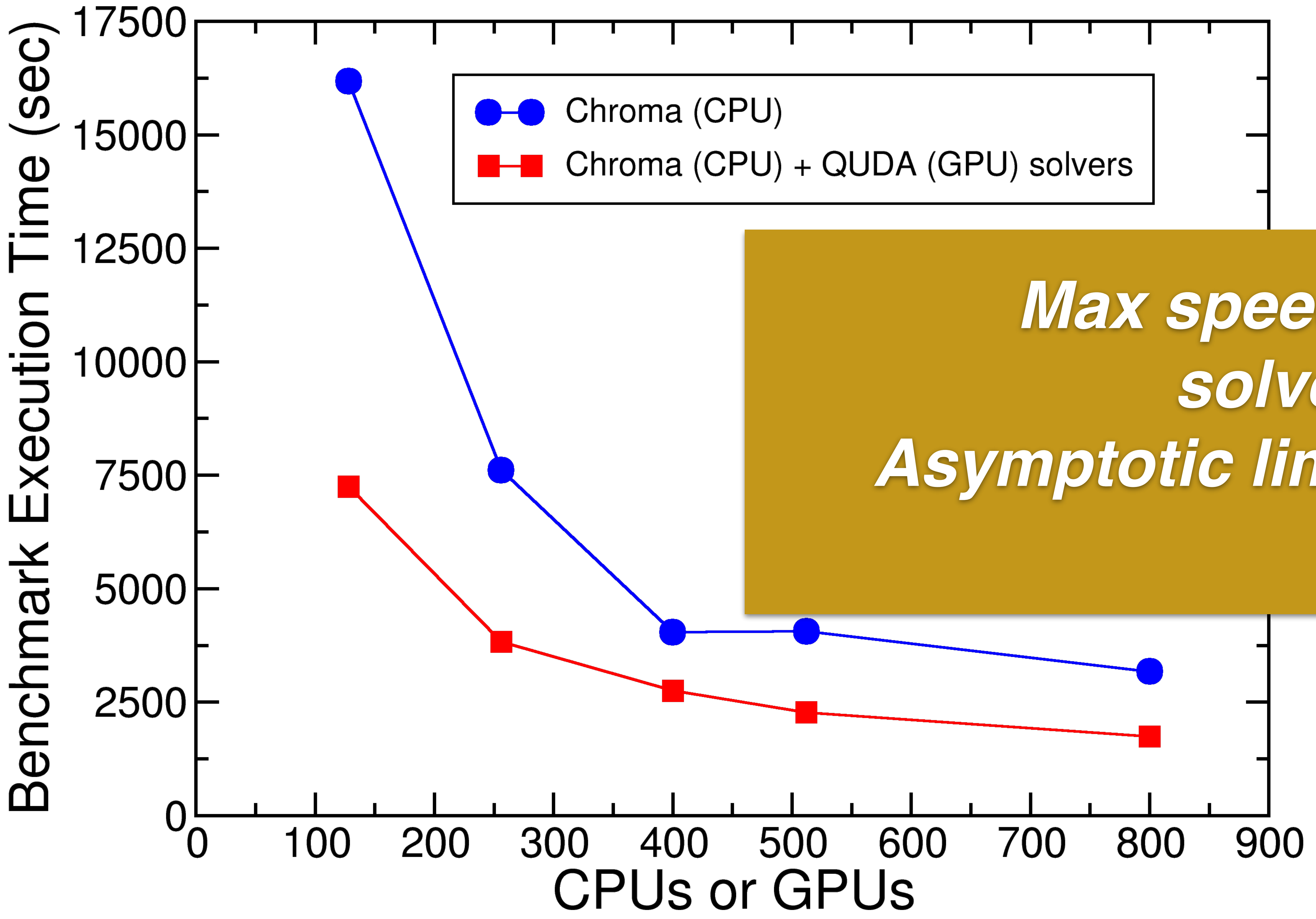
Data replotted from F. Winter, et. al. IPDPS'14

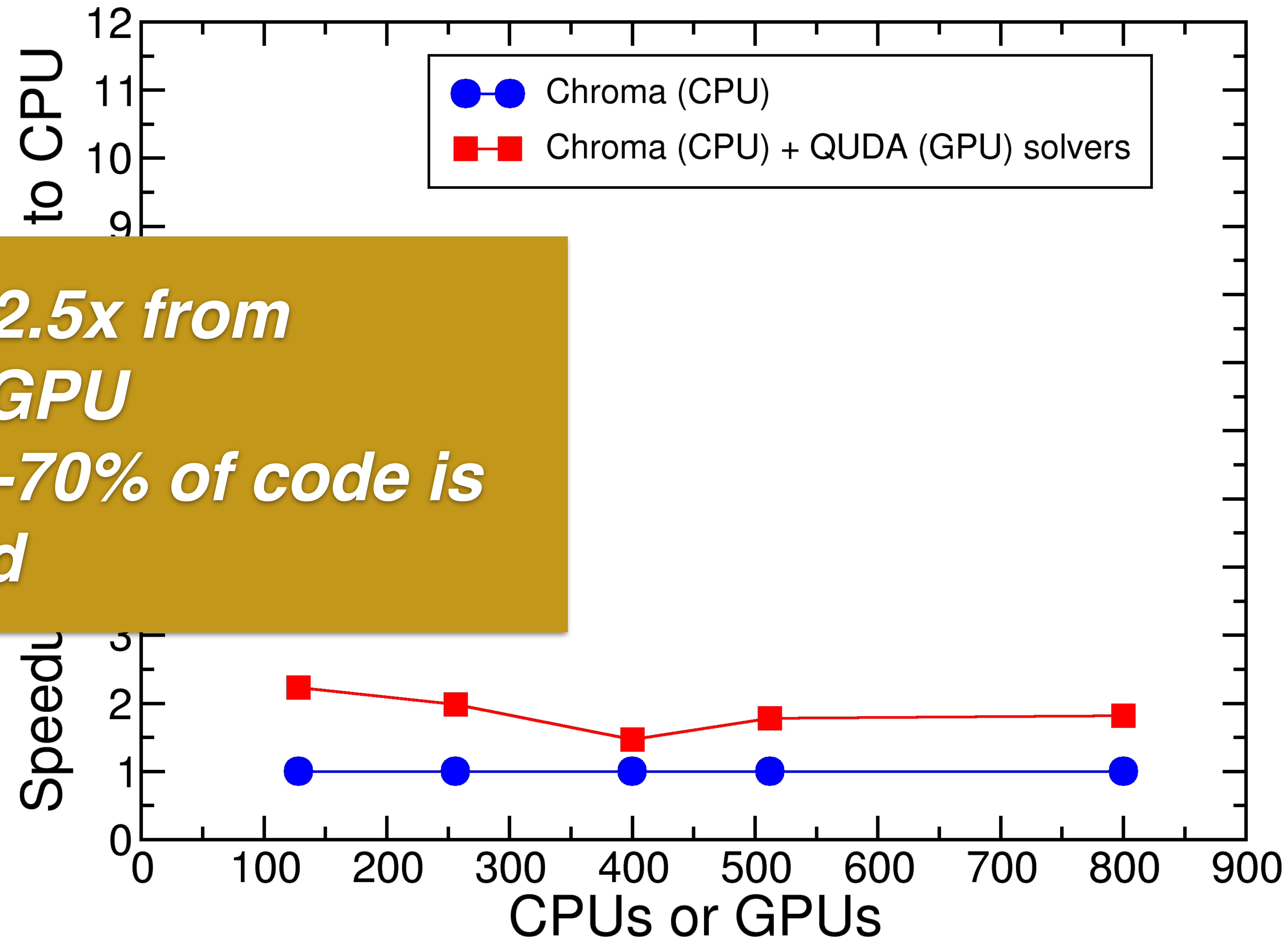
Non-Solver Performance

Wallclock Time
(lower is better)

Benchmarks from NCSA BlueWaters

Speedup
(higher is better)



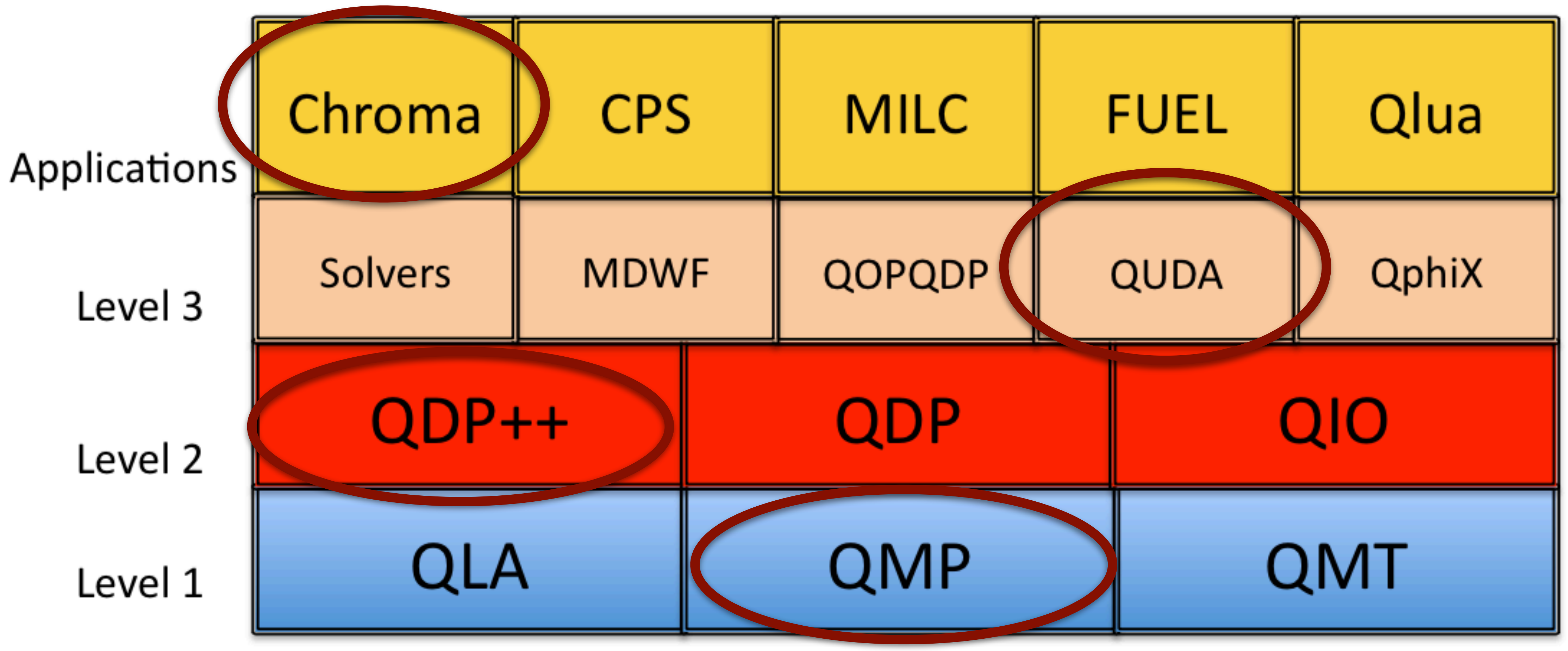


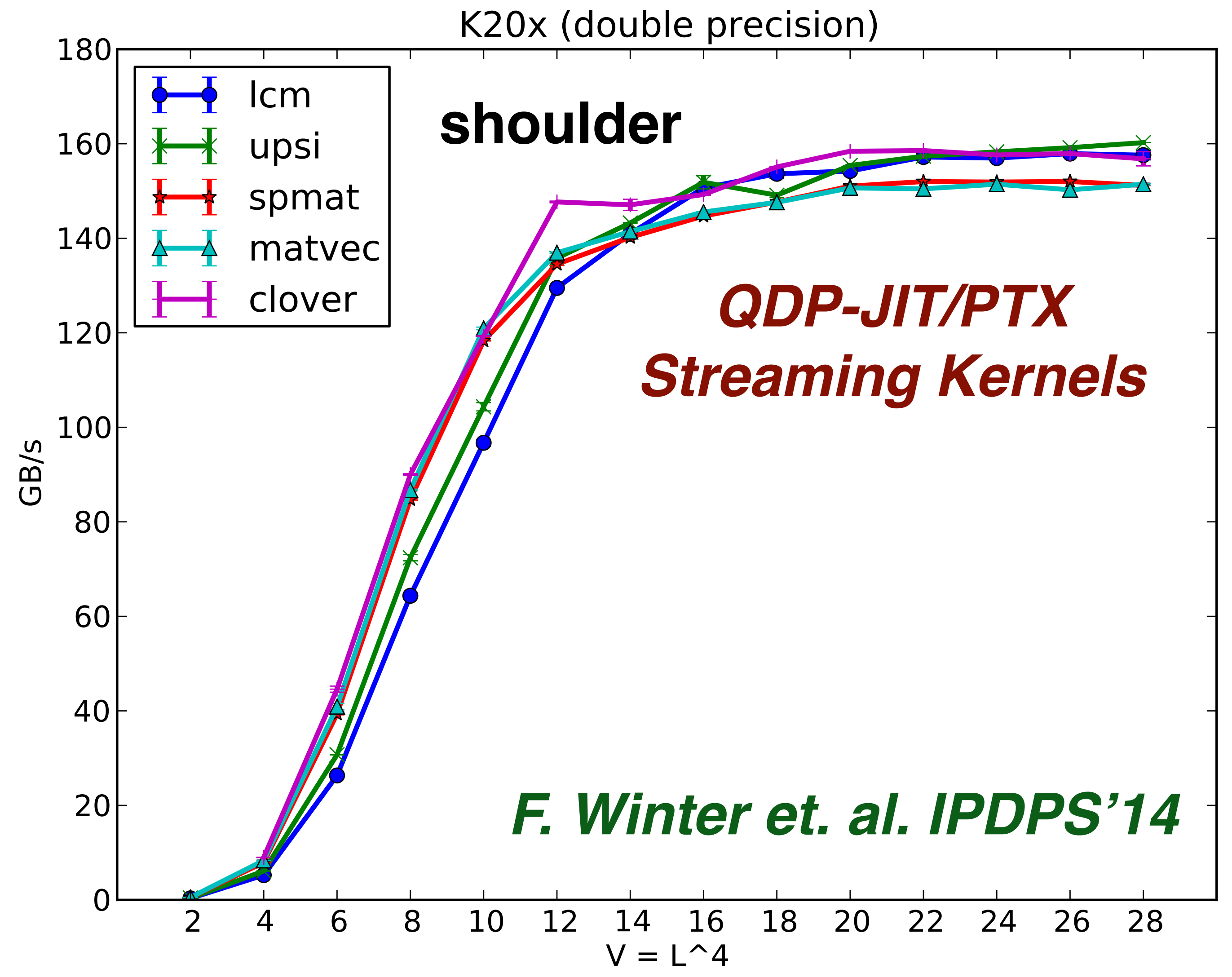
Max speedup of ~2-2.5x from solvers on the GPU
Asymptotic limit, if ~60%-70% of code is accelerated

Data replotted from F. Winter, et. al. IPDPS'14

Accelerating Non Solver Code

- Chroma code is based on a data parallel framework: QDP++
- GPU Challenges:
  - generateing GPU kernels from expression templates (ETs) of QDP++
  - coalesced data layout, host/GPU memory spaces
- Solution: QDP-JIT (F. Winter et. al., IPDPS'14)
  - QDP++ ETs generate code generators
  - Generate PTX kernels at runtime
  - Kernels are cached — only generated once
  - Data cache manages which data stays on GPU
  - Data layout changed appropriately when data is moved between host and GPU
  - All Chroma computations are done on GPU



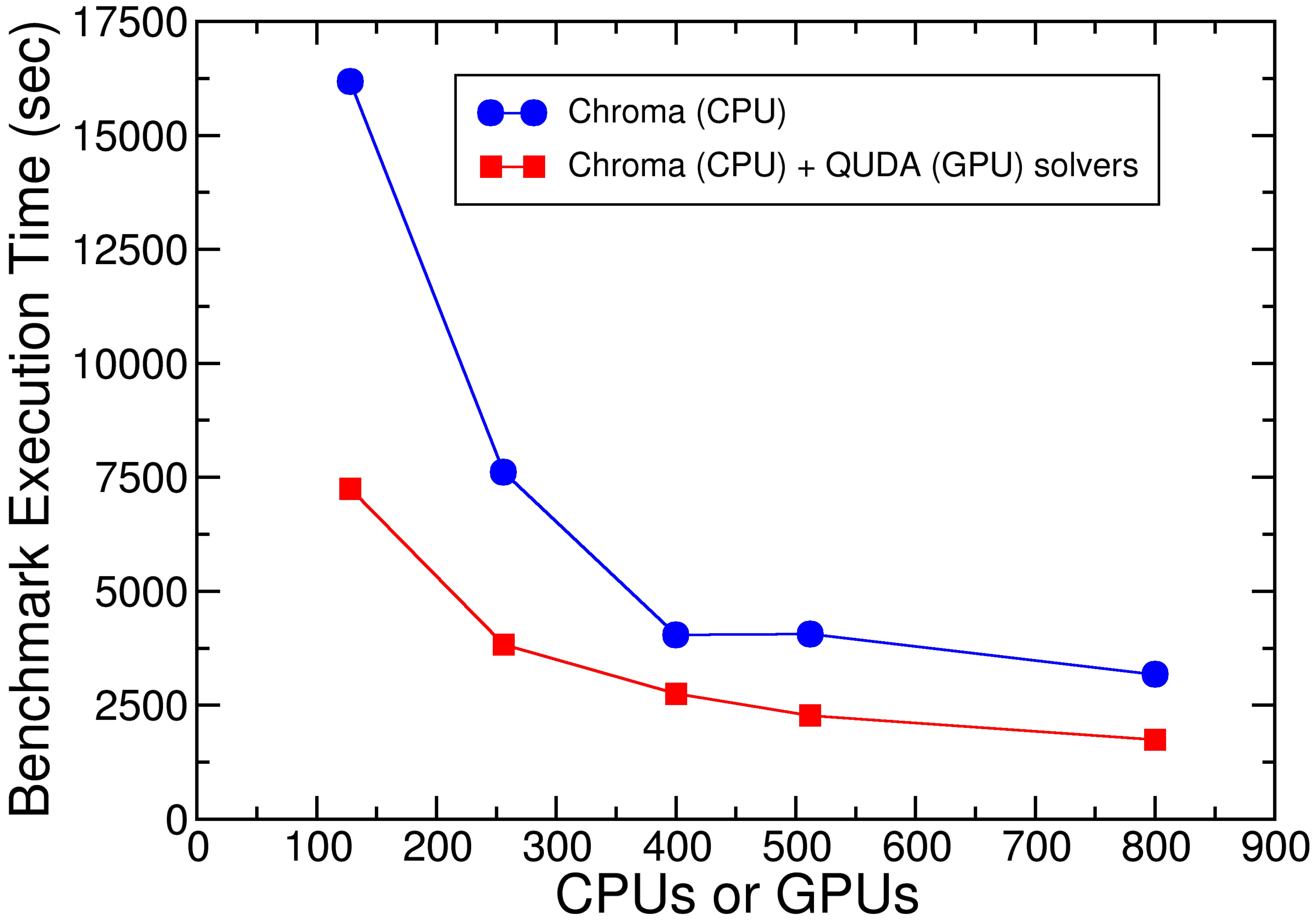


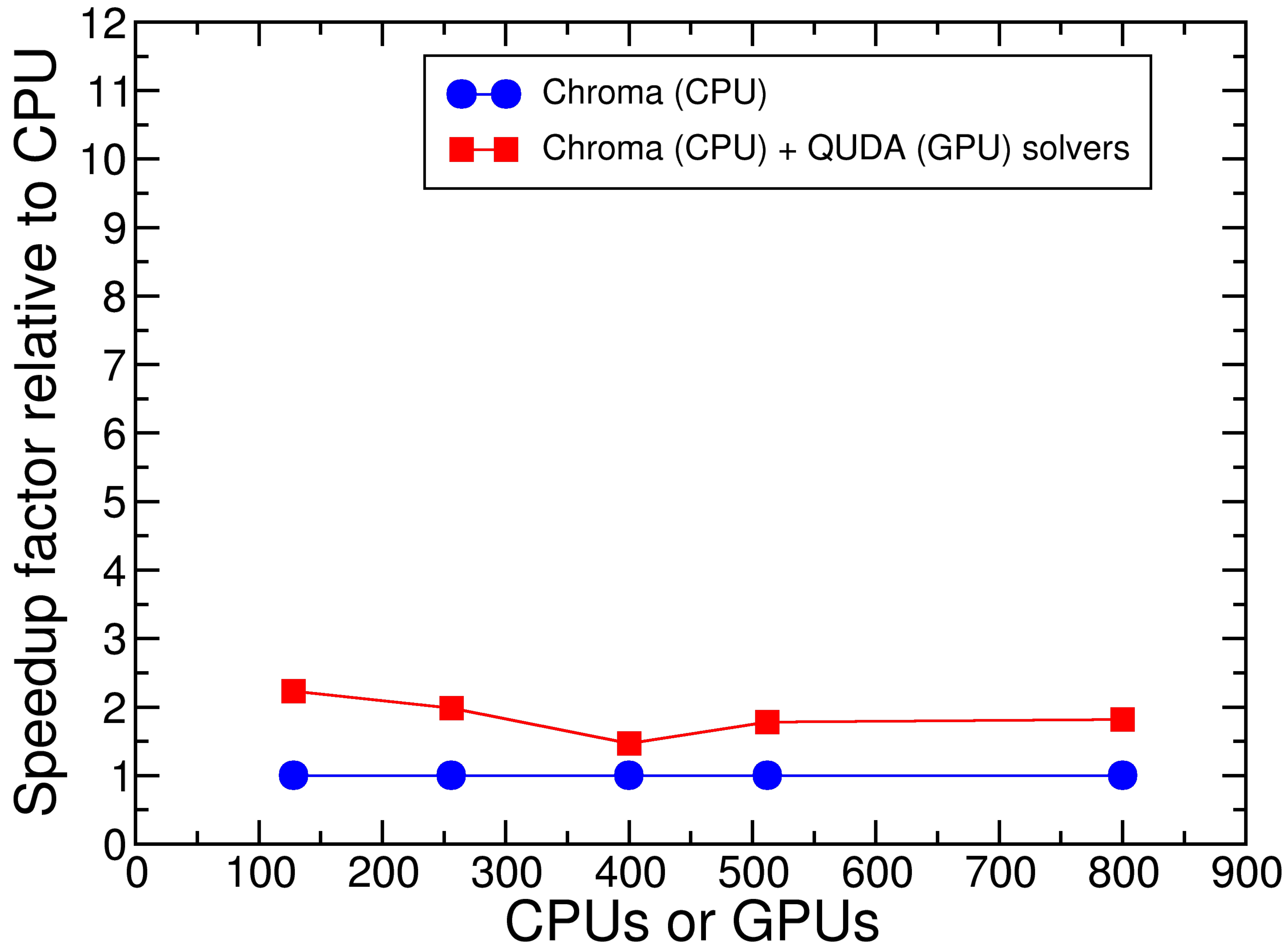
Non-Solver Performance

Wallclock Time
(lower is better)

Benchmarks from NCSA BlueWaters

Speedup
(higher is better)





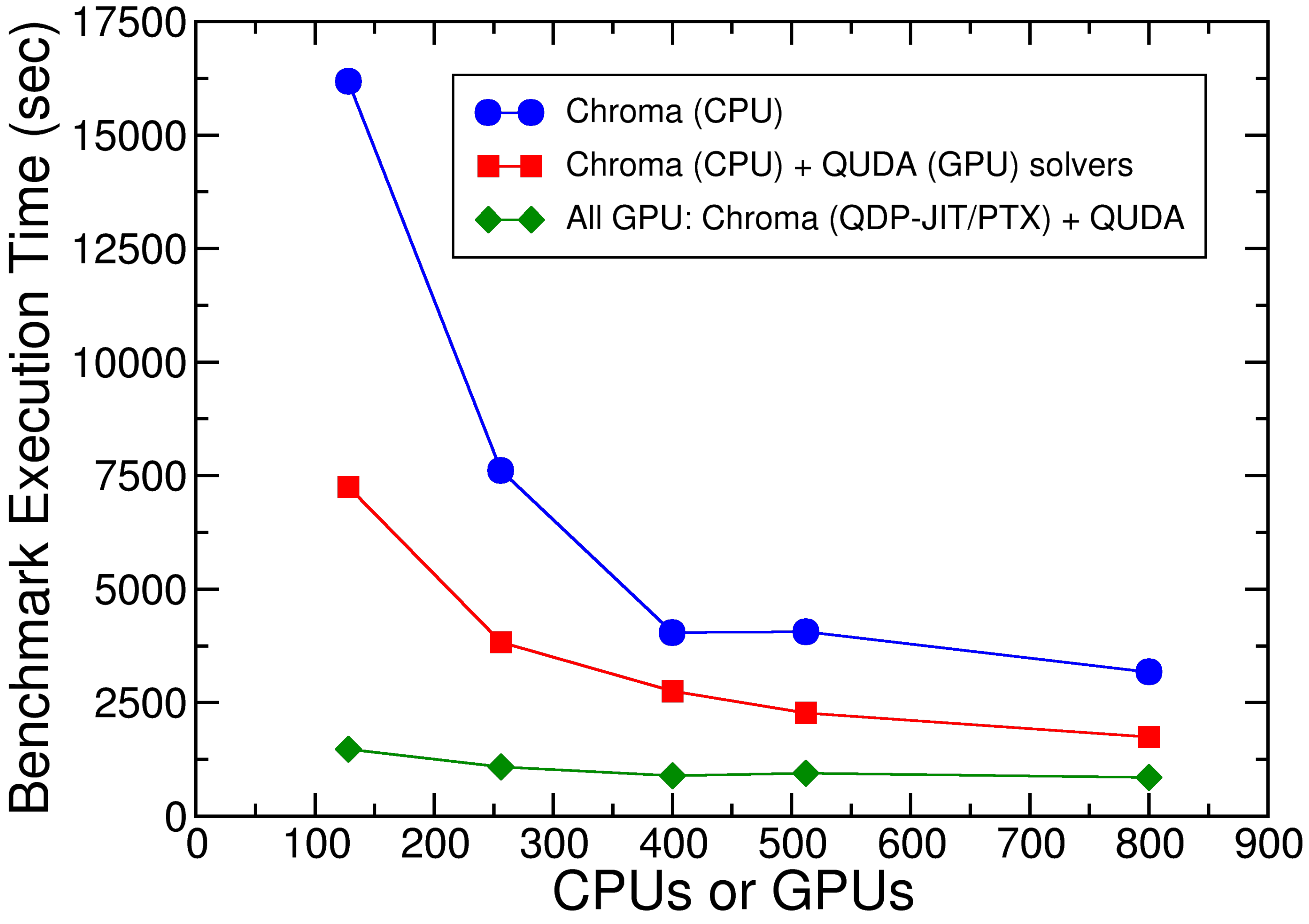
Data replotted from F. Winter, et. al. IPDPS'14

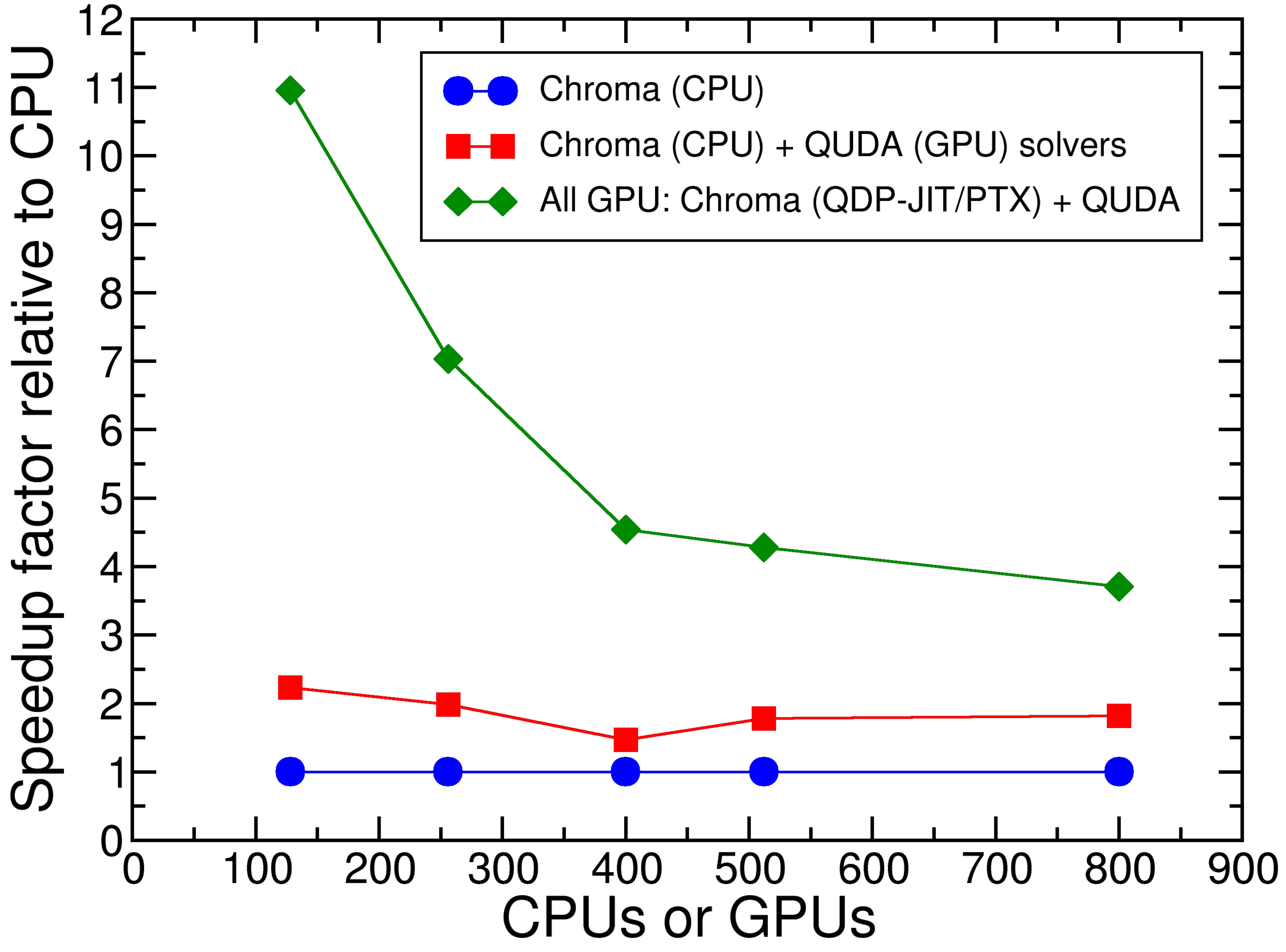
Non-Solver Performance

Wallclock Time
(lower is better)

Benchmarks from NCSA BlueWaters

Speedup
(higher is better)

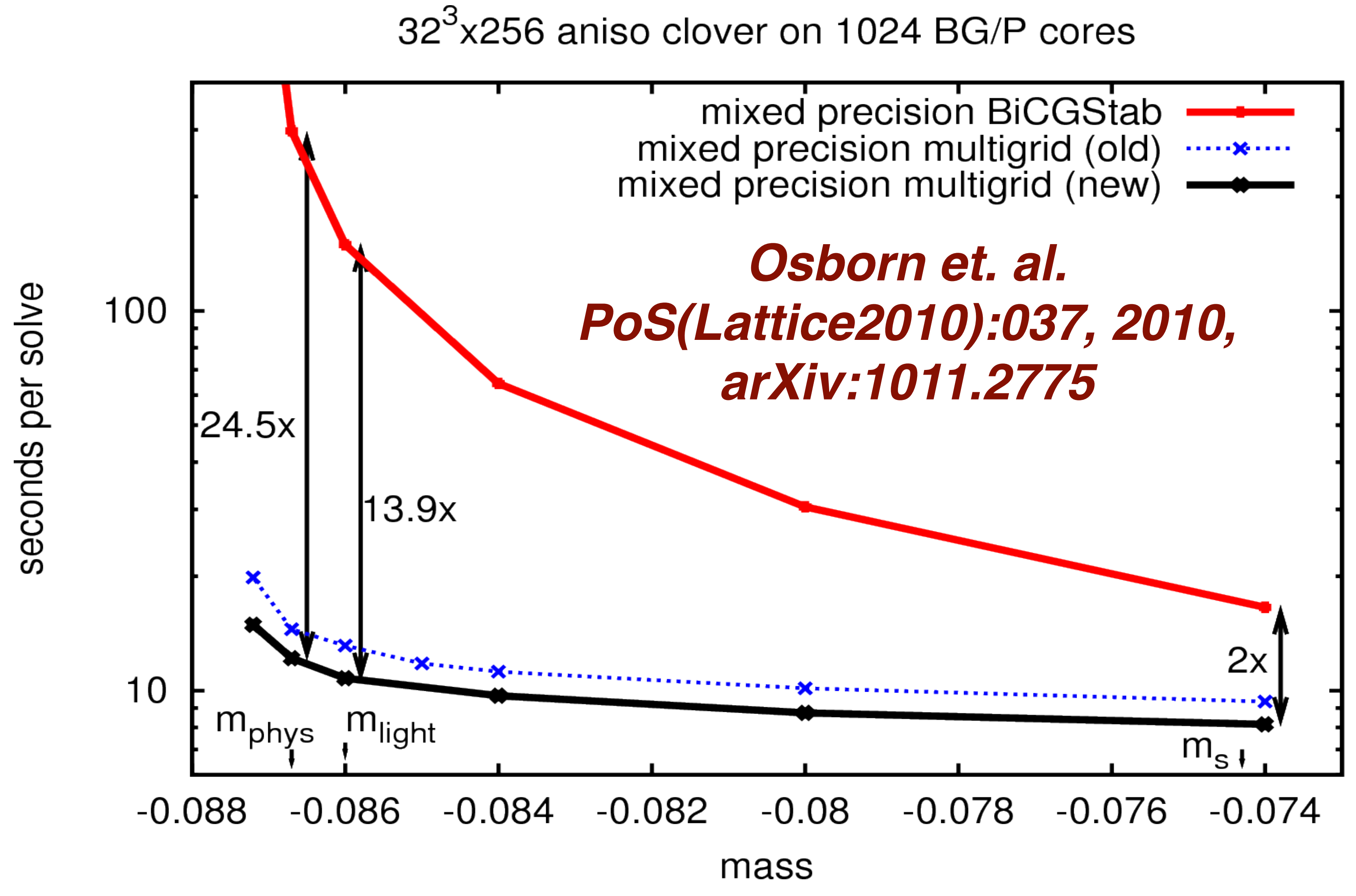


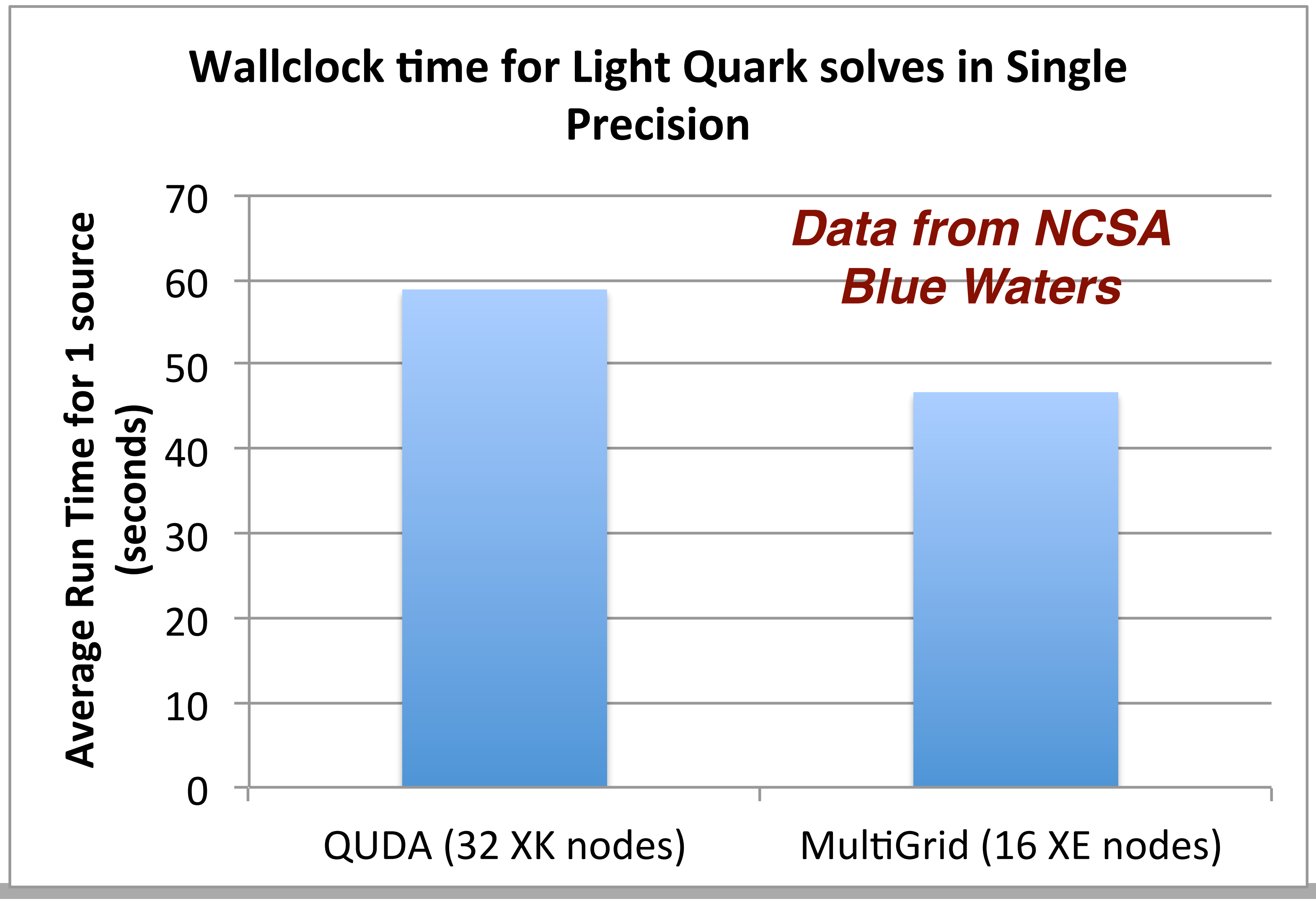


Data replotted from F. Winter, et. al. IPDPS'14

Future Perspectives

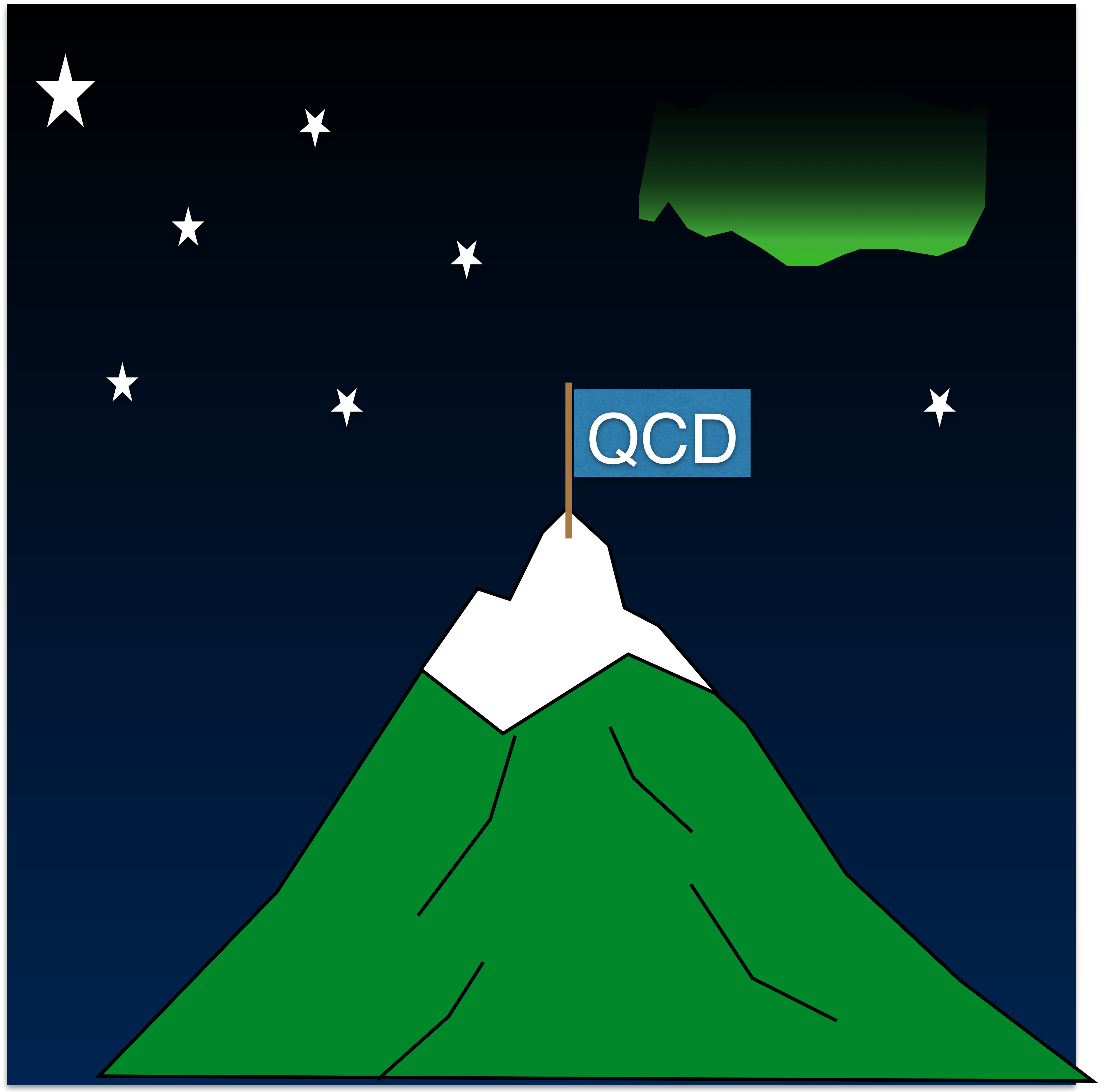
- The Rise of Multi Grid (in QCD)
  - recently developed Algebraic Multi Grid method promises over 10x speed improvement over conventional Krylov methods at light quark masses (Babich et. al. PRL 105:201602, 2010)
  - CPU implementation competitive with QUDA GPU Krylov solvers
  - Tends to be more stable than Krylov methods
- Need efficient GPU accelerated implementation
  - Combine algorithmic and architectural benefits
  - development is underway in QUDA library
- Need to incorporate MG into Gauge Generation
  - capability already exists for the CPU code, using QOPQDP library
  - need it in the GPU based production at physical quark masses
  - can expect between 2x-3x improvement (Amdahl's law for P=72%)





Gazing at Summit (& Cori, Theta, Aurora)

- Diverse Architectures on the horizon:
  - Summit: GPUs, Power CPUs, EDR IB
  - Cori & Theta: Xeon Phi, Knight's Landing, Aries network
  - Aurora: Xeon Phi, Knight's Hill
- Science Productivity Requires
  - portability & efficiency
- High Performance Libraries: QUDA, QPhiX, etc.
  - incorporate most-current algorithms, search for new ones
  - equivalent functionality on different architectures
- Domain Specific Productivity Layer: QDP-JIT/LLVM
  - allow porting of non-solver code: overcome Amdahl's law



Thanks and Acknowledgements

- This work has been funded by the U.S. Department of Energy, Office of Science, Offices of Nuclear Physics, High Energy Physics, and Advanced Scientific Computing Research.
- We gratefully acknowledge computer time for our Scientfic Program,
    - provided by OLCF under INCITE, ALCC and DD awards,
    - provided by ALCF through INCITE awards,
    - provided by NERSC through ERCAP awards,
    - provided by NCSA BlueWaters through PRAC awards,
    - provided by TACC under NSF XSEDE allocation
    - from cluster resources at Jefferson Lab operated under the US. National Facility for Lattice Gauge Theory funded by the U.S DOE, Office of Science, Offices of Nuclear and High Energy Physics