More Lattice QCD on the Intel(R) Xeon Phi(tm) coprocessor

Bálint Joó,
Jefferson Lab, Newport News, VA, USA

(But I am just the presenter… see next page)

IXPUG 2014, TACC, July 8, 2014

Key people

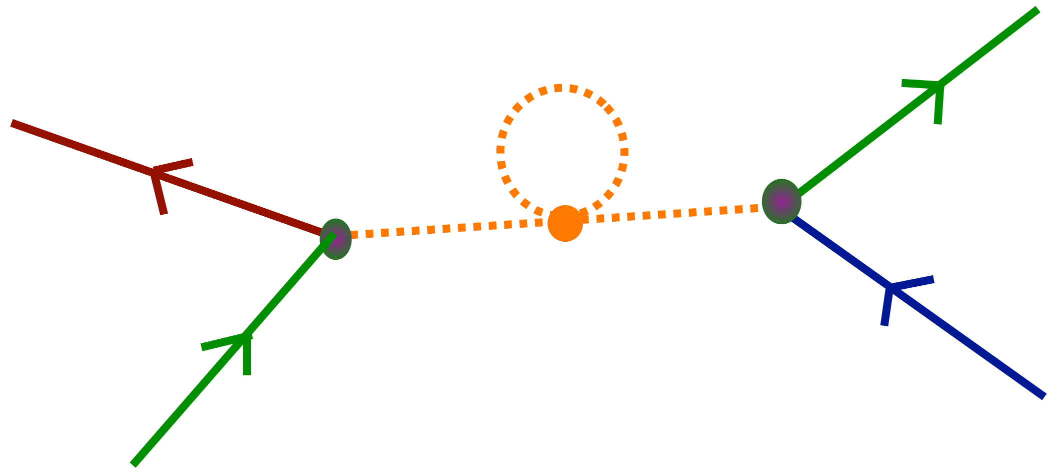
- This is more of a review than last time. The results here are the work of many people
  - Mikhail Smelyanskiy, Dhiraj D. Kalamkar, Karthikeyan Vaidyanathan from Intel Parallel Computing Labs in Santa Clara and Bangalore
  - Prof Steve Gottlieb and Ruizi Li from the Indiana University working as part of the BEACON project.
  - Simon Heybrock, Tilo Wettig and Jacques Bloch, University of Regensburg from the QPACE project
  - Robert Edwards, who using our efforts linked into the Chroma code very efectively consumed our Stampede allocation last year for Hadron Spectroscopy calculations

Outline

- LQCD calculations, and formulations
- Most recent results on Wilson Dslash for Single Node
- Communications
- Production Running on Stampede
- The future… (?)

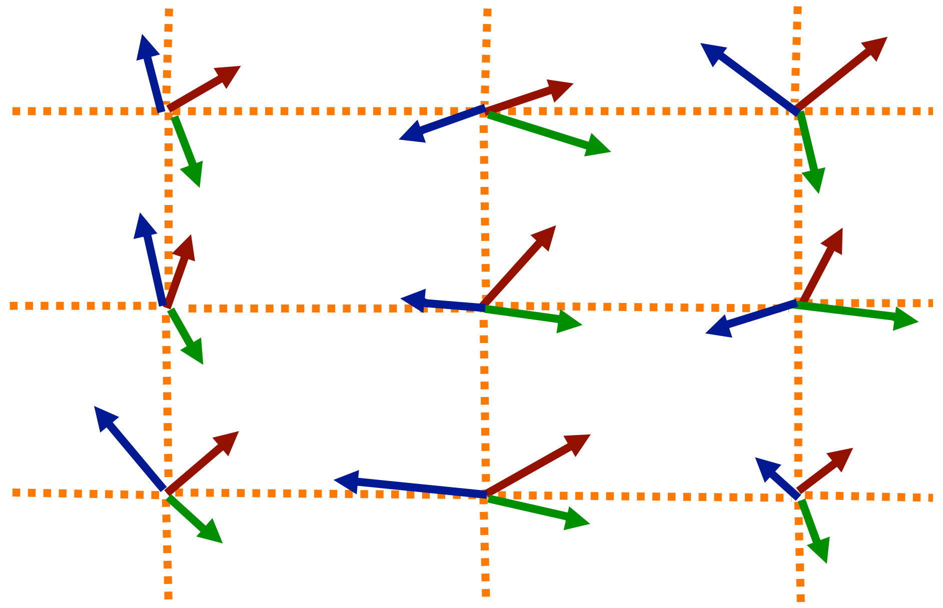
The Basic Lattice Method

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



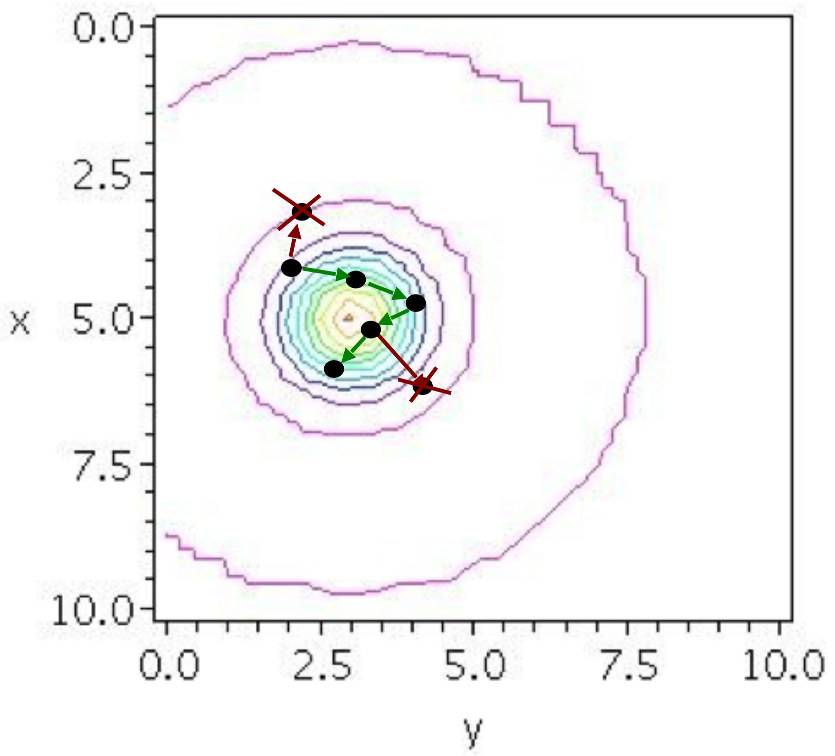
Continuum QCD Path Integral

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



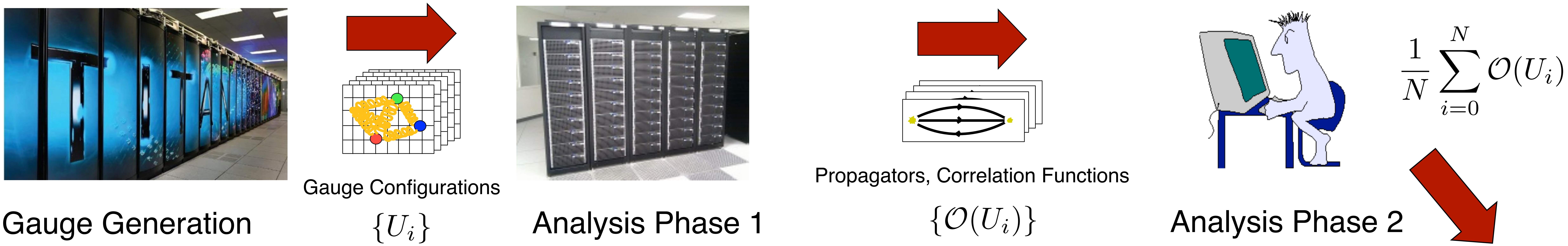
Lattice QCD Integral

$$\langle \mathcal{O} \rangle \;\approx\; \bar{\mathcal{O}} = \frac{1}{N} \sum_{i=0}^{N} \mathcal{O}(U_i)$$



Monte Carlo
Ensemble Average

Carrying out the method



Gauge Generation

Gauge Configurations

$\{U_i\}$

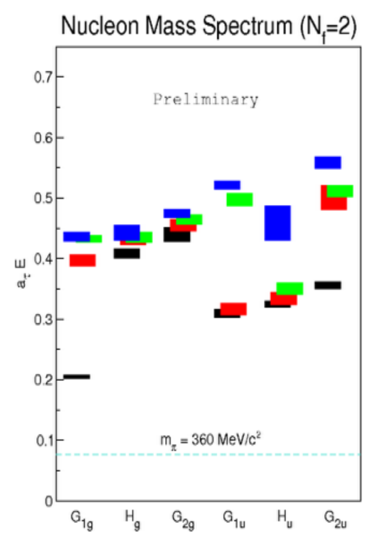
Analysis Phase 1

Propagators, Correlation Functions

$\{\mathcal{O}(U_i)\}$

Analysis Phase 2

$$\frac{1}{N}\sum_{i=0}^{N}\mathcal{O}(U_i)$$

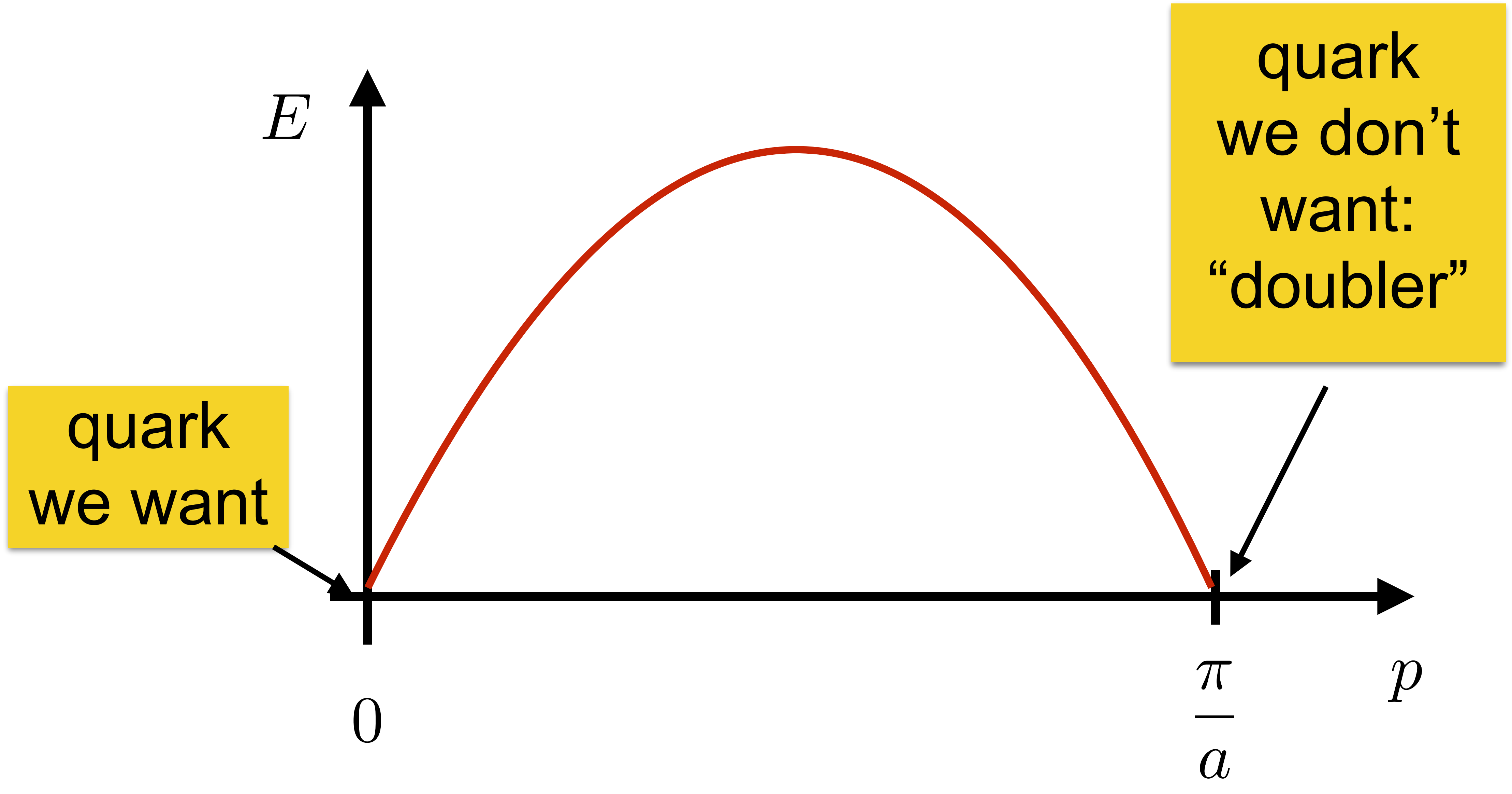


Physics Result

- Gauge Generation: Sequential Markov Chain Monte Carlo
  - Exploit data parallelism offered by lattice
  - Strong scaling challenge (fix global lattice volume, increase nodes)
- Analysis:
  - Exploit task parallelism over gauge configurations as well as data paralleism
  - Primarily a throughput problem (use ‘optimal’ local problem size/node, do many problems)

Fermions have issues…

- Naive discretization of lattice quarks leads to ***extra unwanted*** species of quarks
  - in 4D => 16 species of quarks… way too many.



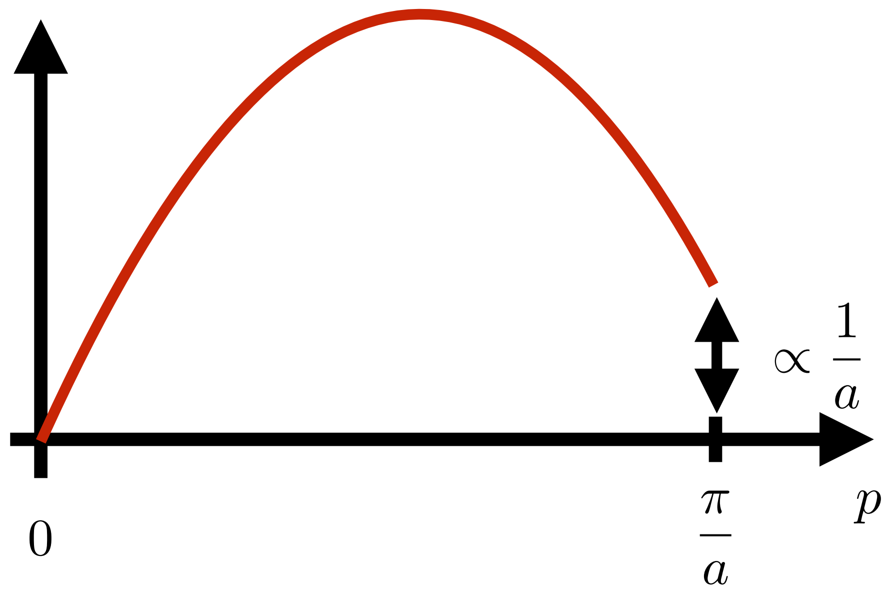
$$G(p) = \frac{i}{a}\sum_{\mu}\gamma_\mu \sin\left(p_\mu a\right)$$

- Nielsen-Ninomiya No Go Theorem:
  - one cannot simultaneously have all the following desireable properties:

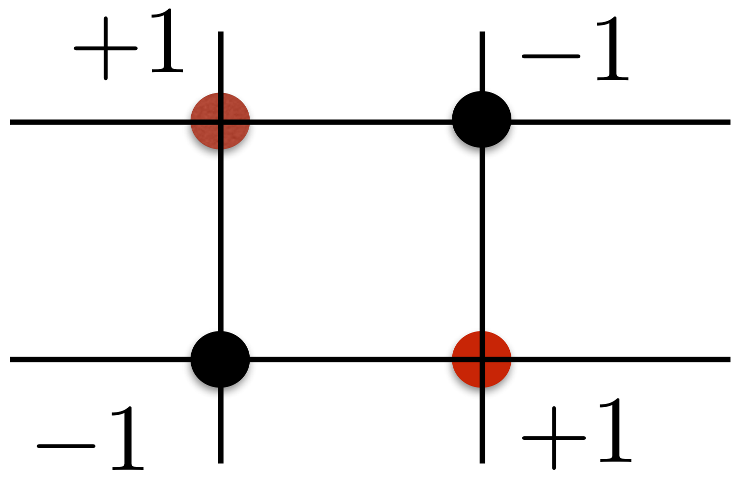
Ultra-locality
Chiral Symmetry
No Doublers
Still look like a fermion

A variety of quark formulations

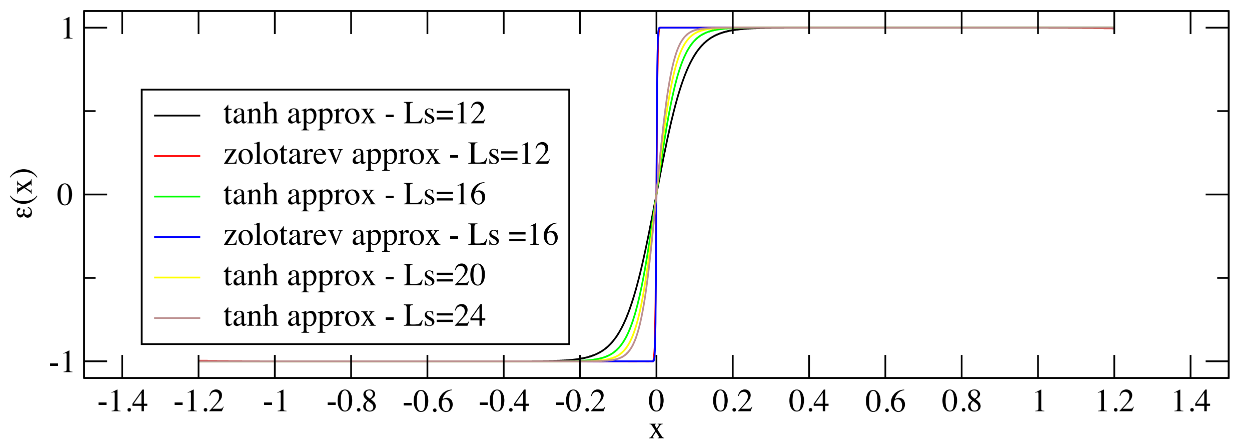
- Wilson-like Fermions: (My primary focus)
  - give doubler modes mass proportional to 1/a
  - explicitly break Chiral symmetry

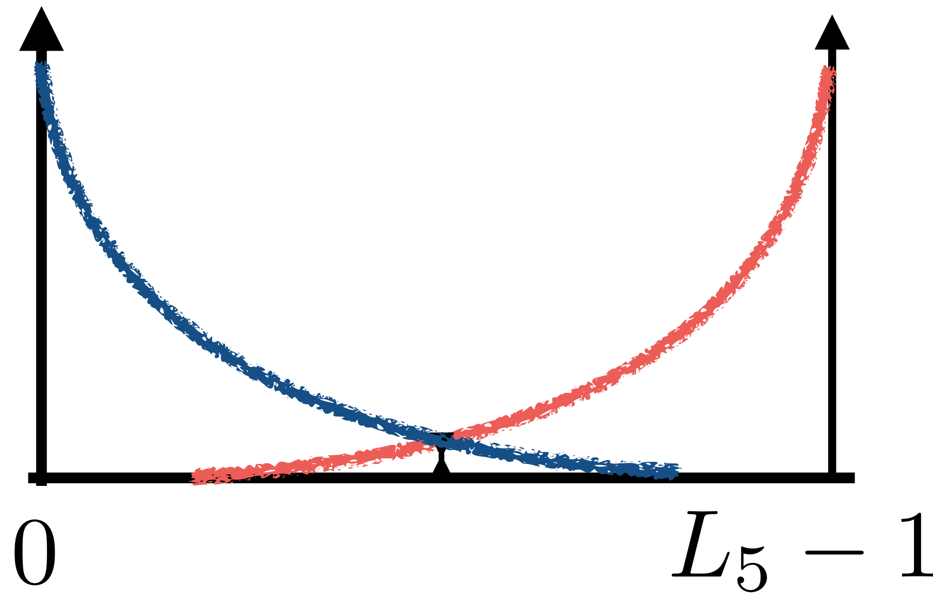


- Staggered Fermions: (e.g., MILC collaboration)
  - reinterpret some doublers 'as spins' of fewer species
  - taste symmetry breaking
  - remnant U(1) "Chiral Symmetry"



- 4D & 5D Chiral Fermions: (e.g., Chi-QCD collaboration)
  - get chiral-symmetry, but loose 'ultralocality' (still local tho)
  - 4D (Overlap): sign function of operator
  - 5D (DWF et. al): Length of 5th dimension





The Role of Linear Solvers

- A majority of work in LQCD is spent in linear solvers
- Gauge Generation:
  - Typically ~60-80% of time is spent in solvers for MD Forces
  - U fields change between solves
- Analysis:
  - Up to 95-96% of the time is spent in linear solvers
  - Many solves per configuration
- Optimize solvers & linear operators first:
  - while they are expensive, everything else is cheap
  - willing to go to ‘close to the metal’ optimization
  - when solvers are cheap, other code becomes expensive
    - DSL approach (QDP-JIT/PTX, QDP-JIT/LLVM) F. Winter
    - See paper about QDP-JIT/PTX at IPDPS’14 by Winter et. al.

```
($\phi_0 = \phi$ is an Initial Guess)
1. Compute $r_0 = \chi - M^\dagger M \phi_0,\ p_0 = r_0$
2. For $j = 0, 1, \ldots$ until convergence:
3. $\alpha_j = \frac{\langle r_j, r_j \rangle}{\langle M p_j, M p_j \rangle}$
4. $\phi_{j+1} = \phi_j + \alpha_j p_j$
5. $r_{j+1} = r_j - \alpha_j \left( M^\dagger M \right) p_j$
6. $\beta_j = \frac{\langle r_{j+1}, r_{j+1} \rangle}{\langle r_j, r_j \rangle}$
7. $p_{j+1} = r_{j+1} + \beta_j p_j$
8. End For
```

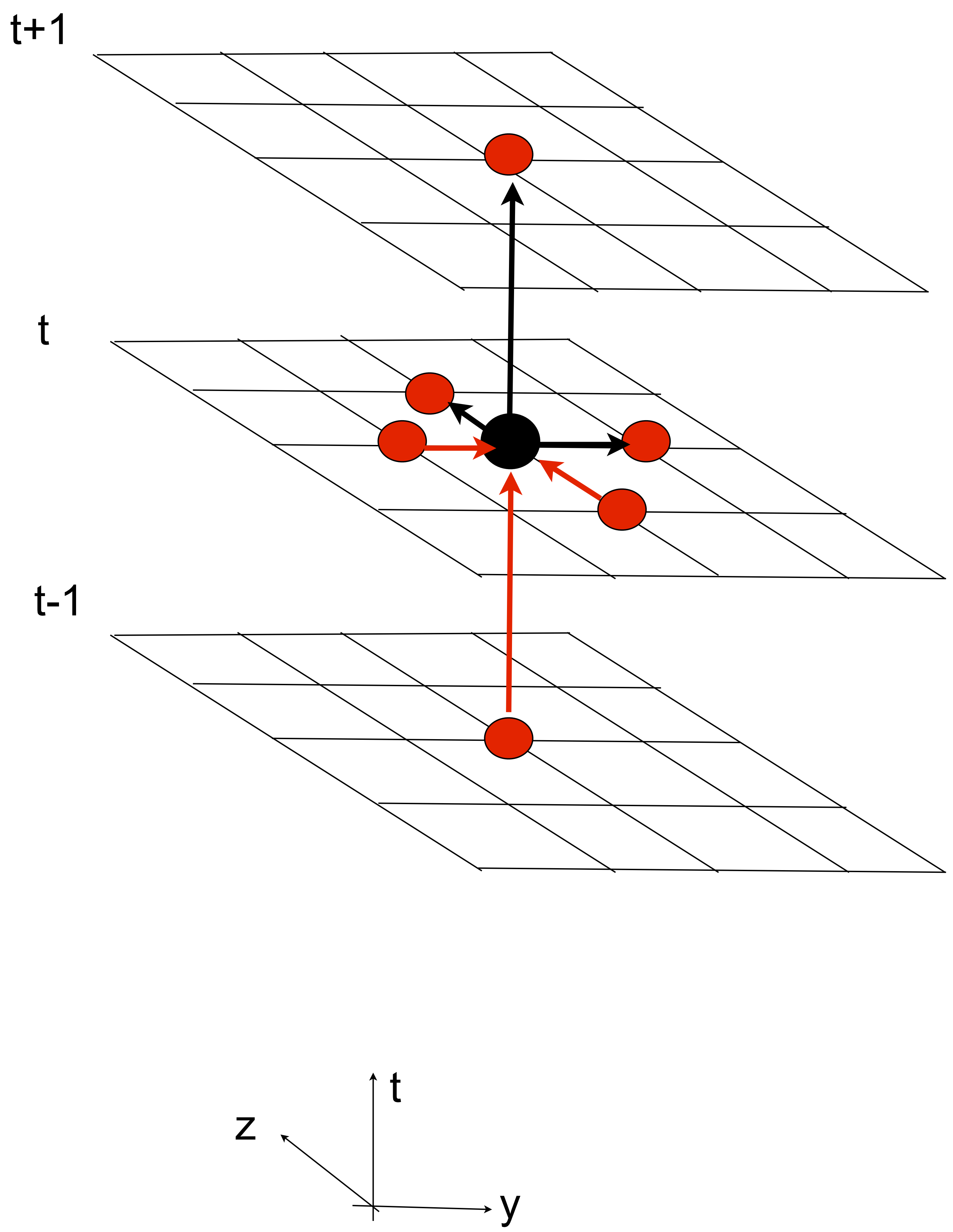
$$\tilde{M}_{oo} = A_{oo} - D_{oe} A_{ee}^{-1} D_{eo}$$

Dslash Term

Wilson Dslash Operator

$$D_{x,y} = \sum_{\mu} \left[(1-\gamma_\mu) U_{x,\mu} \delta_{x+\hat{\mu},y} + (1+\gamma_\mu) U^{\dagger}_{x-\hat{\mu},\mu} \delta_{x-\hat{\mu},y} \right]$$

- Key LQCD Kernel: Wilson Dslash Operator
  - U matrices on links. 3x3 Unitary Matrices (complex)
  - spinors on sites: 3x4 complex matrices
  - 9 point stencil: nearest neighbors in 4-dimensions
    - read 8 neighbors, write central value
  - Naive Intensity: 0.92 flop/byte (SP), 0.46 flop/byte (DP)
- Main Non-Local Operator in a solver
  - all other operations are local or are global reductions.



Basic Performance Bound for Dslash

- R = no of reused input spinors
- B_r = read bandwidth
- B_w = write bandwidth
- G = size of Gauge Link matrix (bytes)
- S = size of Spinor (bytes)
- Simplify: Assume $B_r = B_w = B$
- This model assumes nontemporal stores

$$FLOPS = \frac{1320}{8G/B_r + (8-R)S/B_r + S/B_w}$$

See Smelyankiy et. al. Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC11),

Reuse (R)	Compress	SP FLOPS/B
0	No	0.92
0	Yes	1.06
7	No	1.72
7	Yes	2.29

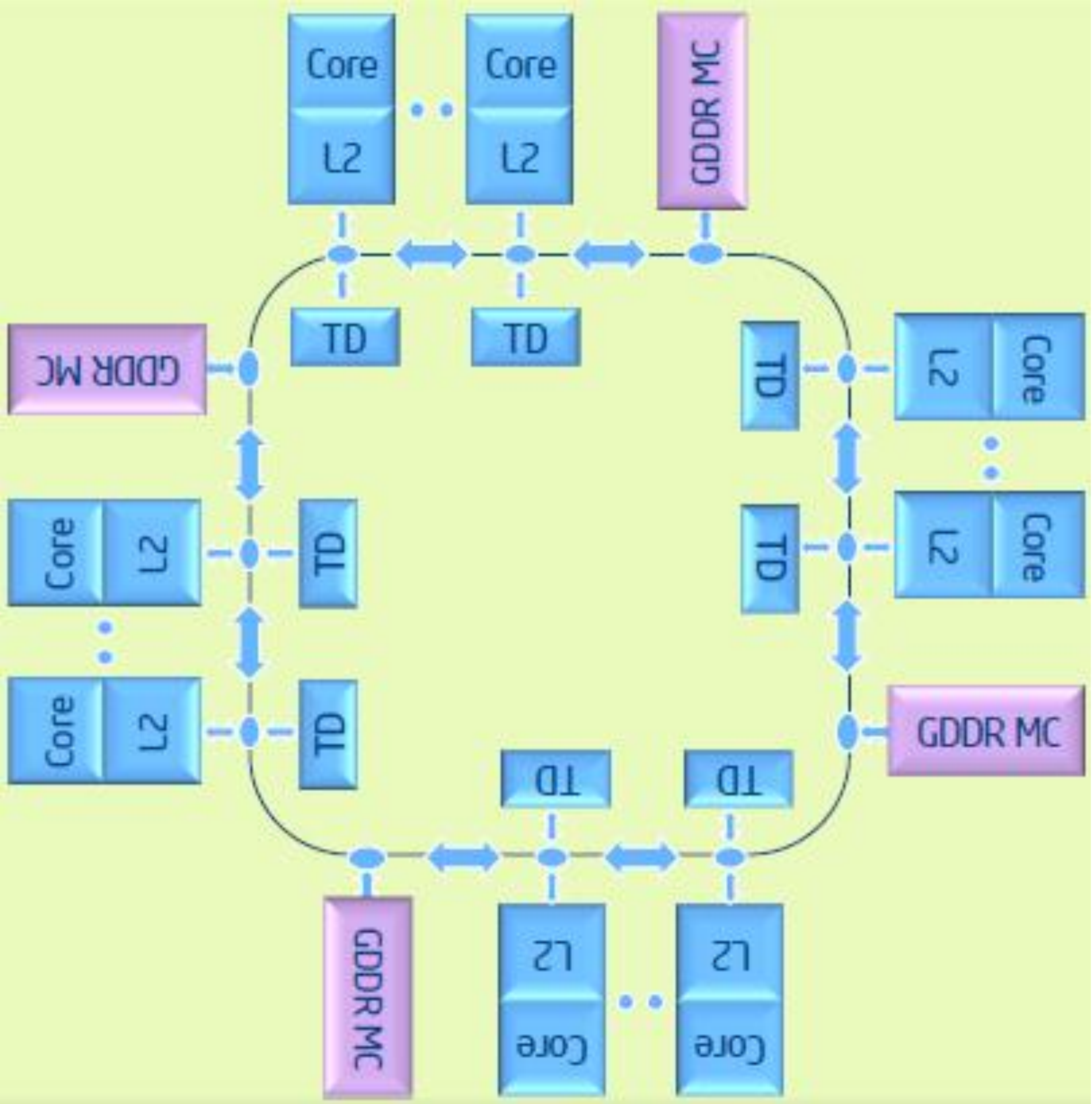
Compression: 2 row storage:

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

See M. Clark et. al. Comput. Phys. Commun. 181:1517-1528,2010

Xeon Phi, Knights Corner

- 60-61 Processor Cores
  - 5110P: 60 cores @ 1.053 GHz
  - 7110P: 61 cores @ 1.1 GHz
- 2 x vector operations per clock
  - 512 bit vectors = 16 floats = 8 doubles
- Core Private Caches
  - 32K L1 / core, 512K L2 / core
- STREAMS B/W to GDDR: 150-170 GB/sec
  - our own streaming kernels are similar
- 5110P FLOPS/byte
  - Mem B/W=150 GB/s: F/B ~13.5 (SP), ~6.74(DP)
  - Mem B/W=170 GB/s: F/B ~11.9(SP), ~5.94 (DP)

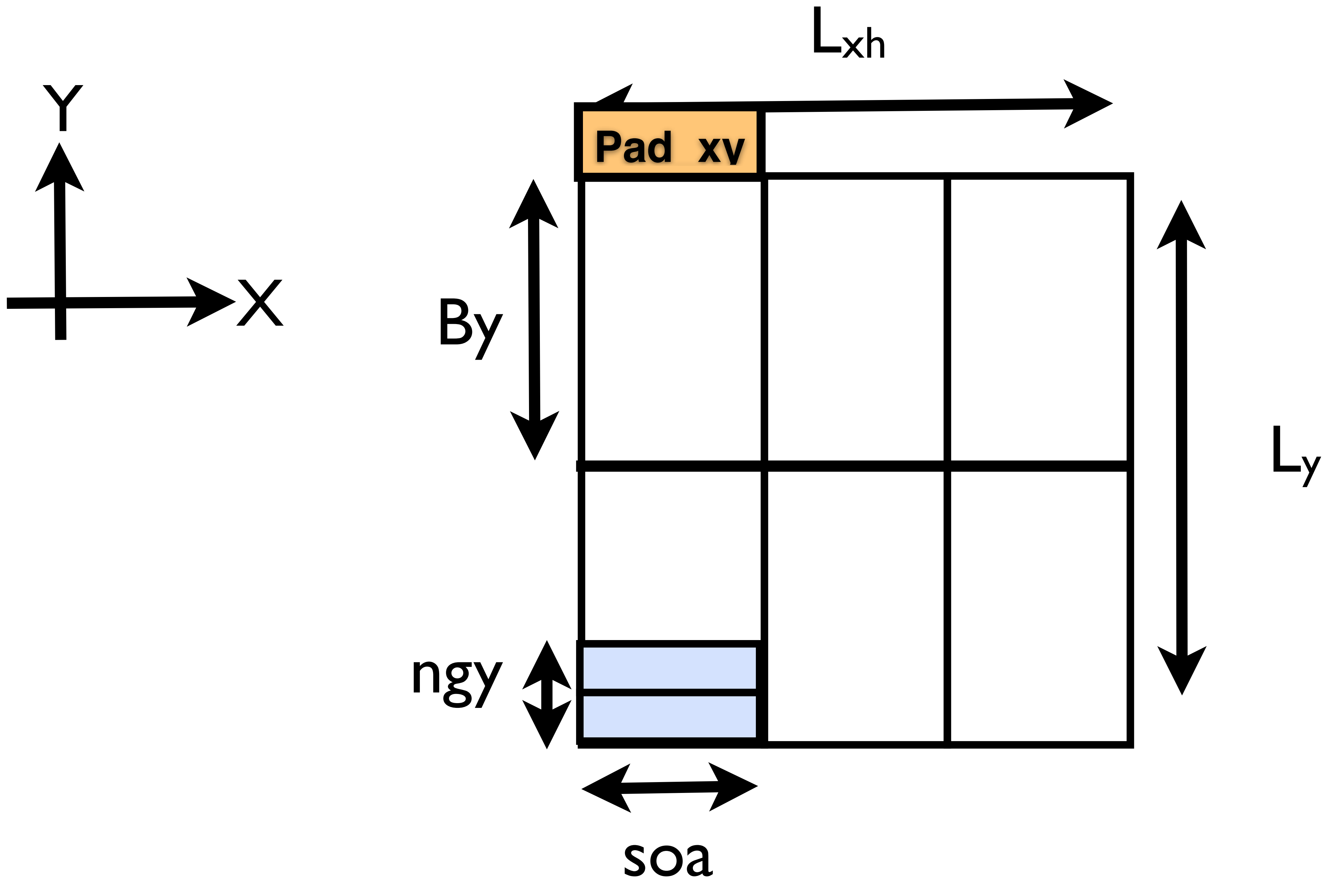


Data Layout For Vectorization

- Partial Structure of Arrays (SOA) layout
- Gather 'ngy' chunks of length 'soa' from 'ngy' different 'y' coordinates.
  - vec=hardware vector length (e.g. 16)
  - soa=SOA Len (e.g. 4,8,16)
  - ngy=vec/soa
- Code Generator
  - generate load-unpack/pack-store
  - generate software pre-fetching
  - allow switching between XeonPhi, AVX, SSE…
- Gauge fields constant:
  - can 'pre-gather' the 'ngy' chunks.
- Can add Padding (e.g., after XY plane)

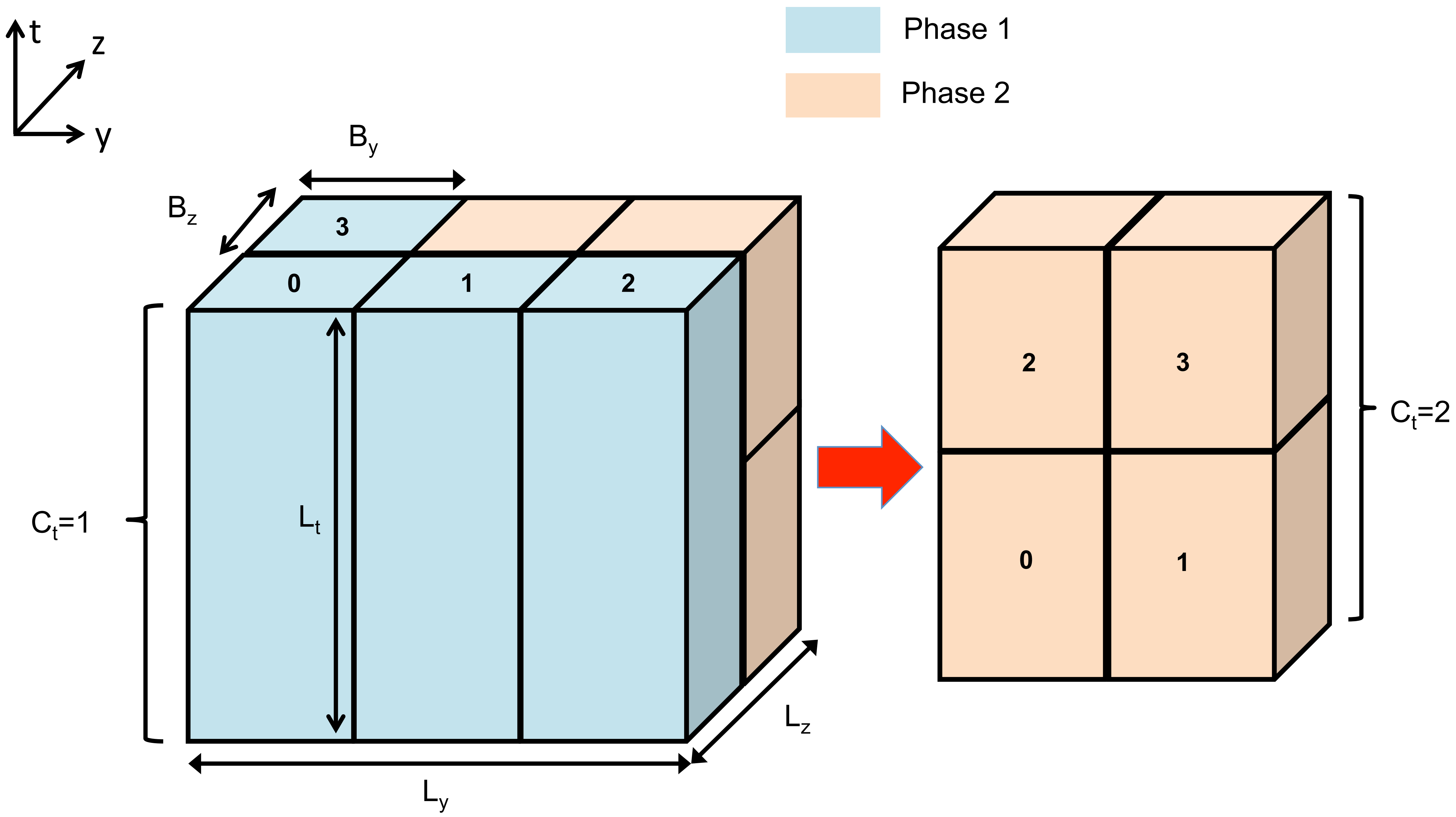
```
typedef float SU3MatrixBlock[8][3][3][2][vec];
typedef float FourSpinorBlock[3][4][2][soa];

// Vh is number of sites in checkerboard
SU3MatrixBlock  gauge[Vh/vec];  // Gauge field
FourSpinorBlock spinor[Vh/soa]; // Spinor field
```



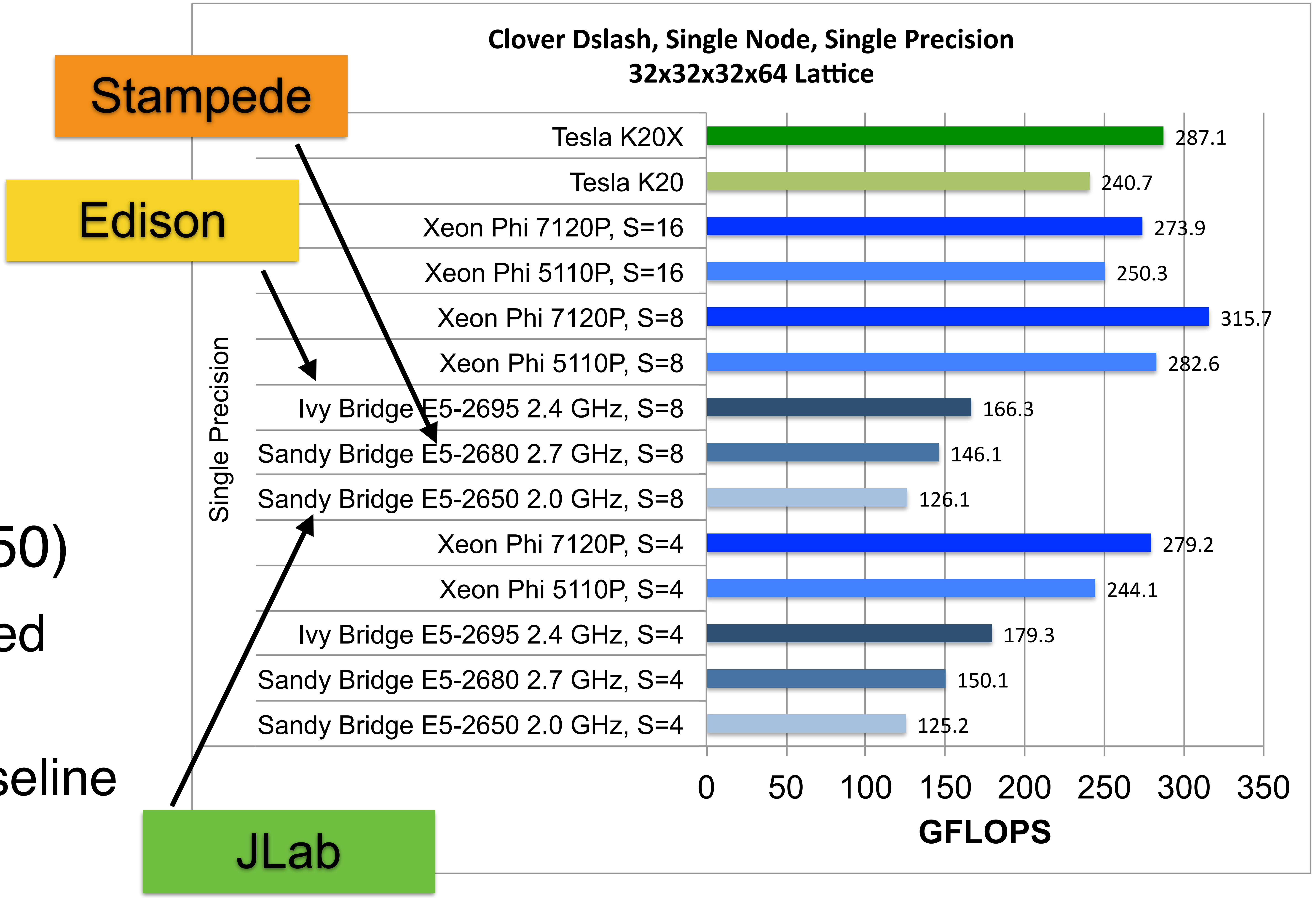
Blocking & Block Mapping

- 3.5D blocking
  - block in Y and Z dims, stream through T
- Challenge:
  - How to assign blocks to cores?
  - Maintaining Node Balance & maximizing number of cores
- Solution: multi-phase block allocation
  - No. of blocks more than cores => allocate round robin to all cores
  - When number of blocks less than cores,
    - either split in T: make more blocks than cores
    - just allocate remaining blocks and finish
    - heuristic to terminate process: when T gets small



Recent Numbers: Single Precision

- Slightly surprising that SOA=8 is better than SOA=16 for Xeon Phi
- Similarly SOA=4 seems better than SOA=8 for Xeon E5 series
- At best speed
  - Xeon Phi 7120 = 2.1x E5-2680 (SNB)
  - Xeon Phi 7120 = 1.76x E5-2695 (IVB)
- Chroma Baseline: 35.8 GF (E5-2650)
  - 3.5x gain from Xeon Phi optimization fed back to E5-2650
  - Xeon Phi 7120: 8.8x speedup over baseline
  - Quite competitive/similar to GPUs


Clover Dslash, Single Node, Single Precision
32x32x32x64 Lattice

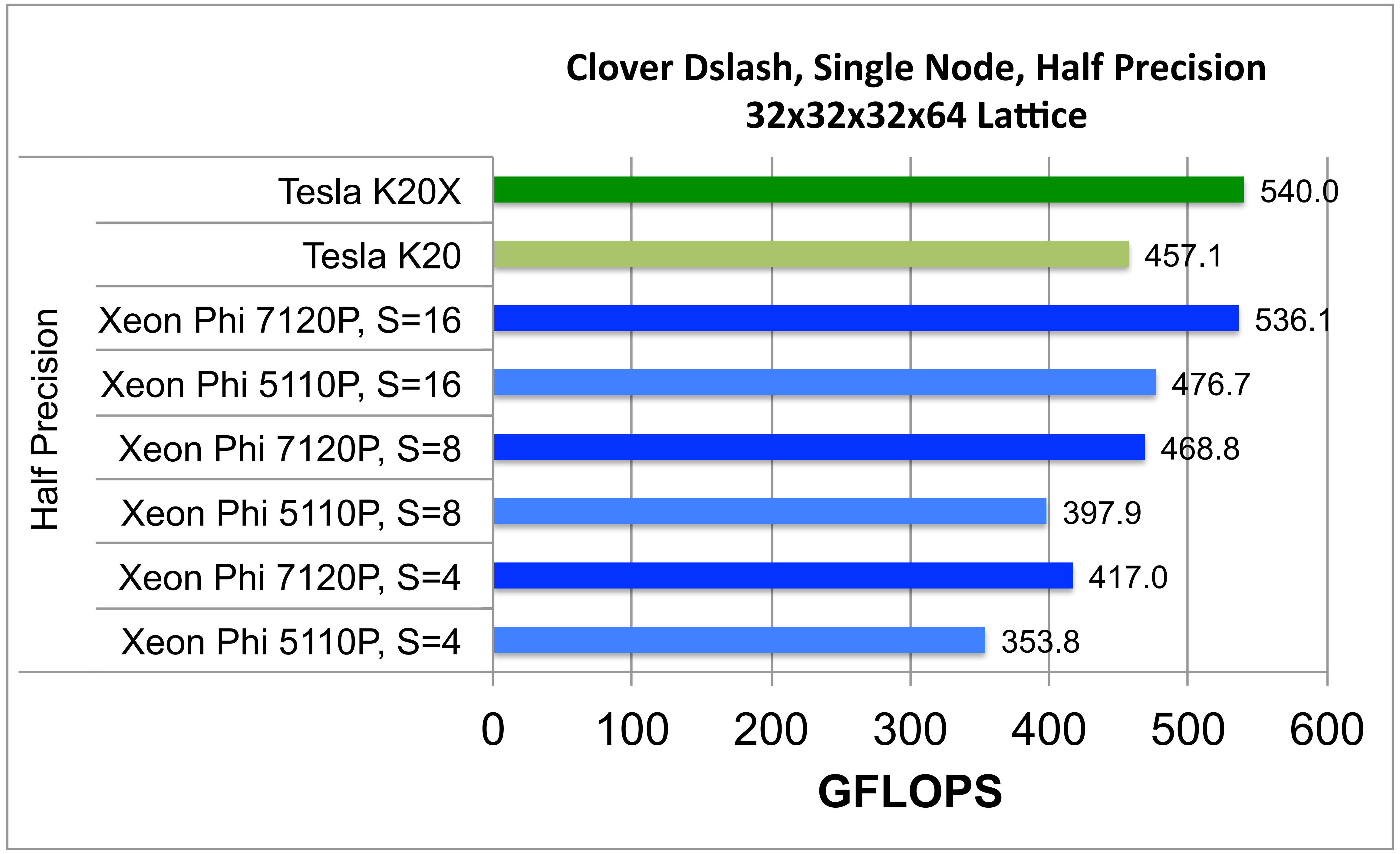
Stampede

Edison

JLab

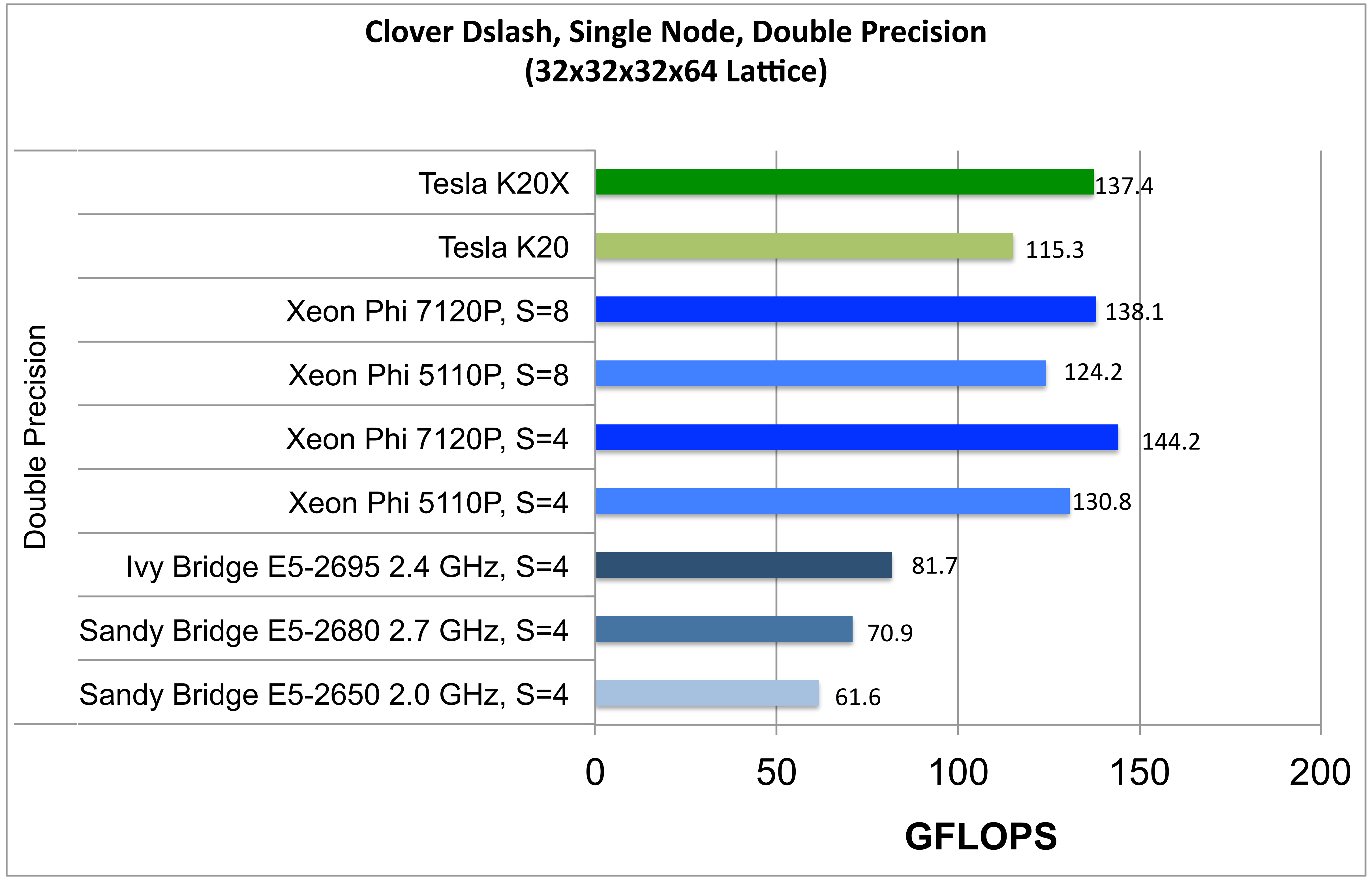
Recent Results: Half Precision

- On Xeon Phi Arithmetic is 32 bit
  - But can up/downconvert to 16 bit on load/store
- As SOA is decreased performance on Xeon Phi drops
  - is this an instruction issue question?
- The best performances are competitive with 16-bit benchmarks on NVIDIA GPUs



Recent Numbers: Double Precision

- Surprising: S=4 better than S=8
  - parallels SP result (S=8 better than S=16)
- Best DP performance ~ 2x SP perf.
  - 7120: 144 GF (DP) vs 315 GF (SP)
  - 5110: 130 GF (DP) vs 282 GF (SP)
- Again similar to 2 dual socket Xeons
  - Xeon Phi 7120 = ~ 2x Xeon E5-2680 (SNB)
  - Xeon Phi 7120 = ~1.76x Xeon E5-2695 (IVB)



Staggered Fermions

- MILC Projects use Staggered Fermions
- Chief computational difference between Wilson and Staggered is the number of Spin components.
  - Wilson has 4 spin componentsbe.
  - For Sttaggered, each site has only 1 spin component.
- Practical implications:
  - Half the number of FLOPS from 3x3 matrix by 3 vec per flop compared to Wilson
  - 4 times less data from Spinors than for Wilson
  - Un-reused Gauge Fields form larger fraction of working set for Staggered than for Wilson
  - Working set smaller than for Wilson :)
- Optimized Implementation by Ruizi Li and Steve Gottlieb, BEACON project

Staggered Dslash Perf. Bounds

- R, B_r, B_w as before
- assume maximum R=7
- G = size of Gauge Link matrix (bytes)
  - compression: 6 complex numbers
  - uncompressed: 9 complex numbers
- S = size of Spinor (bytes)
  - no spin: 3 complex numbers
  - 1/4 size of Wilson/Clover
- c = 0 or 1 for NTA store on or off
- Arithmetic intensity less than Wilson

$$FLOPS = \frac{570}{8G/B_r + (8 - R + c)S/B_r + S/B_w}$$

Compress	NTA store	Staggered FLOPS/B (SP)	Wilson FLOPS/B (SP)
No	No	0.88	-
No	Yes	0.91	1.72
Yes	No	1.25	-
Yes	Yes	1.32	2.29

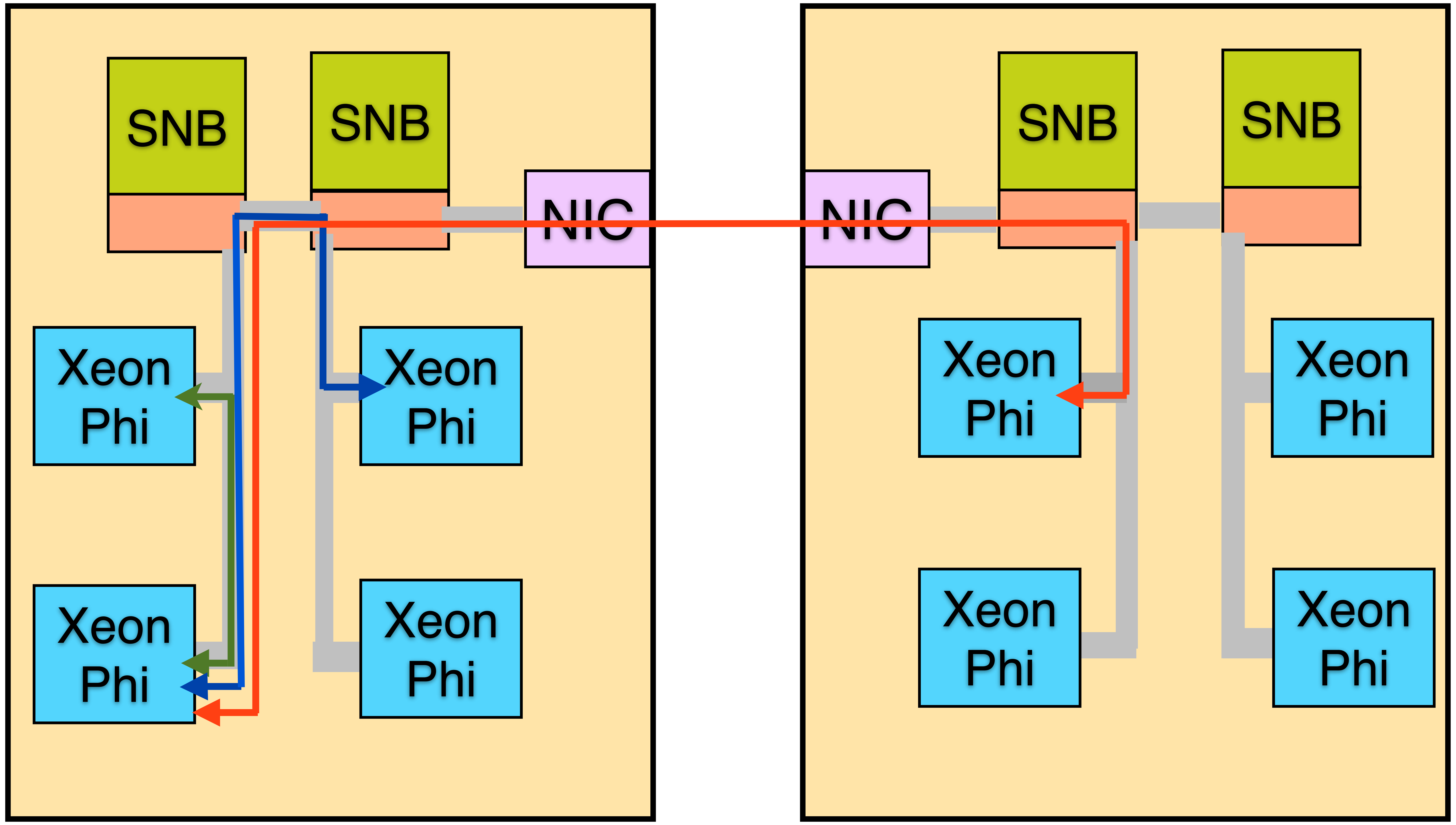
Single Node Staggered Dslash

Compress		NTA store	Model Maximum at B/W=160 GB/s	Measured by Ruizi & Gottlieb
No	No	0.88	140 GF	139 GF
No	Yes	0.91	145 GF	140 GF
Yes	No	1.25	200 GF	184-189 GF
Yes	Yes	1.32	211 GF	187-190 GF

- Numbers courtesy of Ruizi Li, from BEACON: shown at Lattice 2014, New York, NY
  - run on BEACON Xeon Phi 5110P, using 59 or 60 cores in single precision
- Single node optimized numbers in single prec close to perf bound @ 160 GB/sec
- Unoptimized MILC code is slower (42-45 GFLOPS in regular CG)

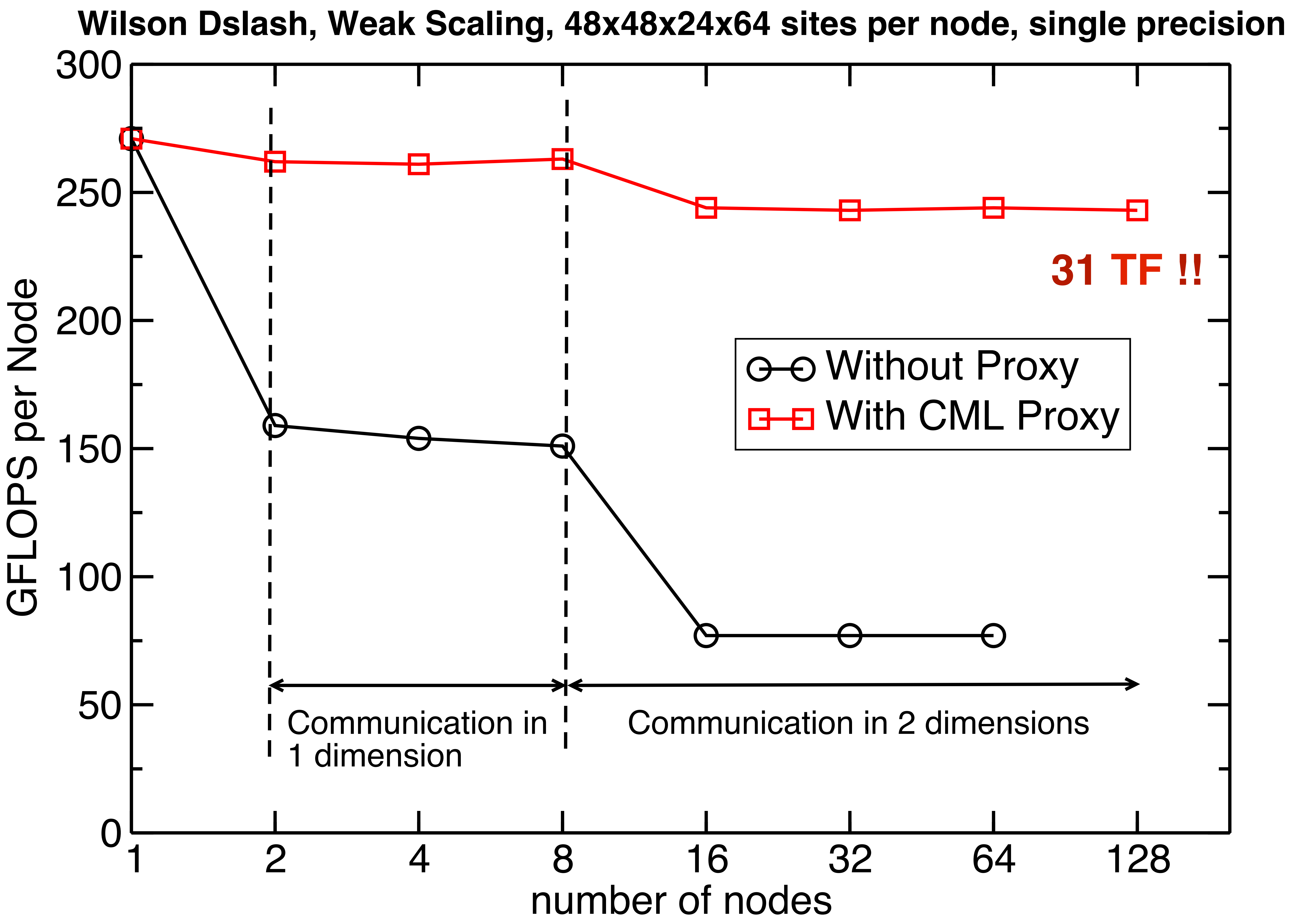
Multi-Node Challenges

- Multiple paths in a Multi-Xeon Phi system with different speeds
  - Xeon Phi - SNB
  - Xeon Phi - Xeon Phi same PCie
  - Xeon Phi - Xeon Phi different PCIe
  - Xeon Phi - Xeon Phi different nodes
  - etc...
- Best path is not always the obvious one
- MPI Standard doesn't guarantee/require asynchronous progress
- Solution: wrote an MPI proxy which finds best path (CML proxy - Vaidyanathan)
  - similar proxy in Intel MPI (CCL)
  - similar in MVAPICH 2 - MIC



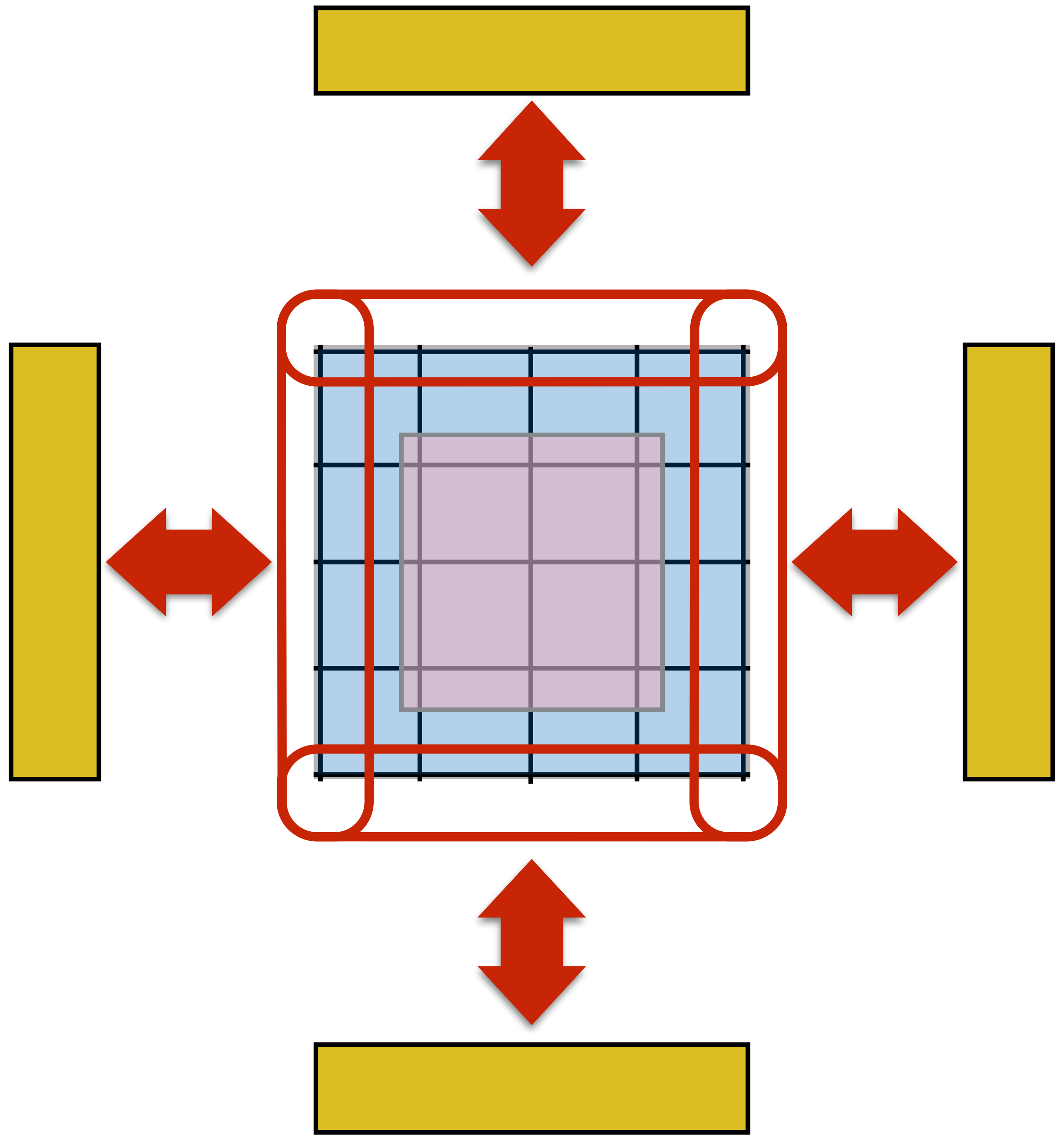
Stampede Weak Scaling Last Summer

- 1 Xeon Phi per node
- Without proxy
  - drop in performance when going to multiple nodes
  - performance halves when introducing second comms direction
  - suggests issue is with async progress rather than attainable bandwidth or latency
- With proxy
  - small drop in performance from 1 to 2D comms. More likely due to B/W constraints...



Improving Strong Scaling

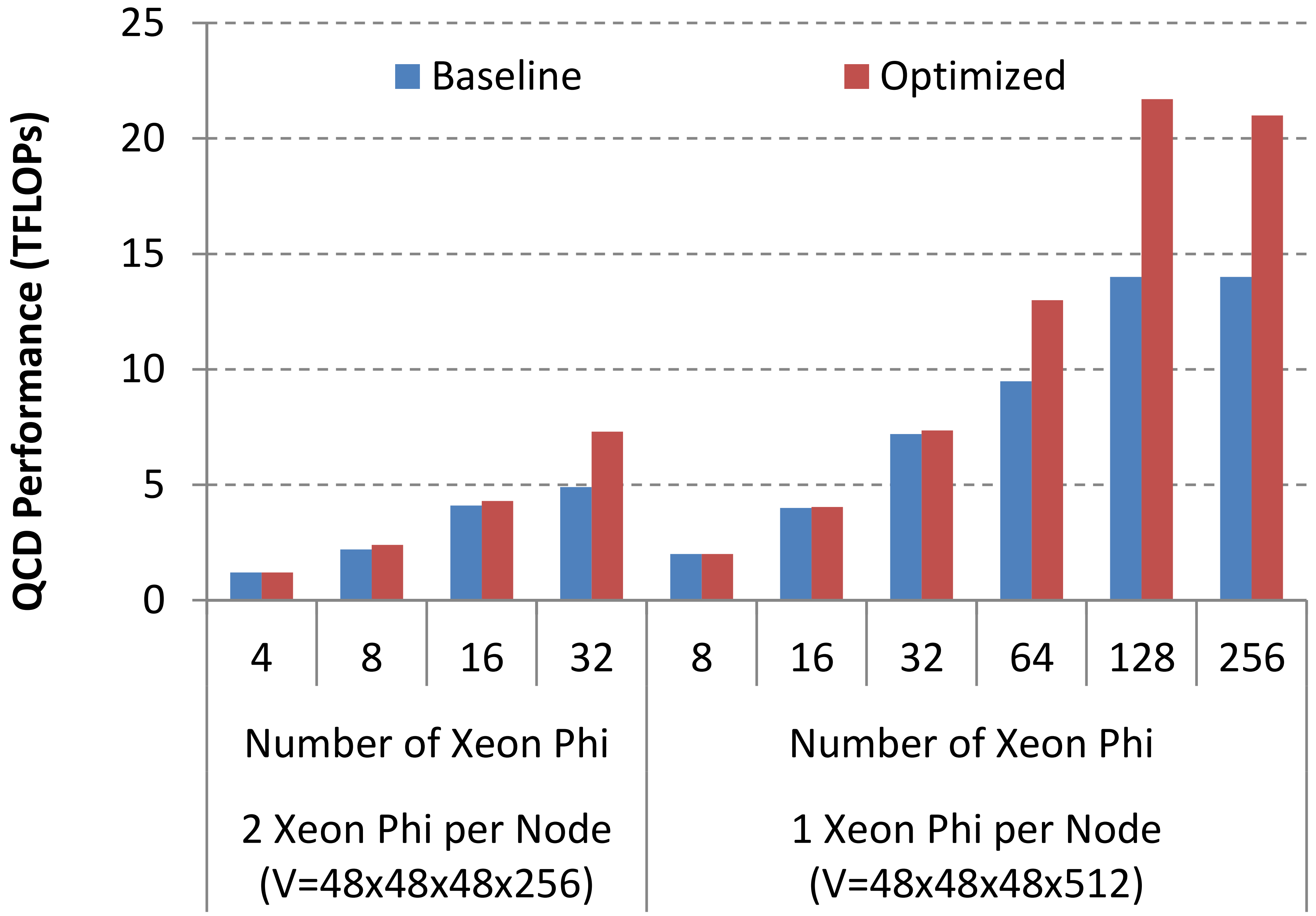
- Strong Scaling Limit: small local volume e.g.:
  - 32^4 sites: 1 face = 768 KB (SP)
  - 8^4 sites: 1 face = 12 KB (SP)
- Reduced opportunity to overlap comms/compute
- Small messages tend to be more bound by latency than by bandwidth
- Latency effects in code can become important
  - OpenMP thread joins in ‘master thread communicates’ model
  - Waiting for messages to arrive in the order they were sent



This discussion and results come from K. Vaidyanathan, et. al. “Improving Communication Performance and Scalability of Native Applications on Intel(R) XeonPhi(TM) Coprocessor Clusters”, IPDPS’14

Improving Strong Scaling

- Techniques to improve strong scaling:
  - Divide threads into groups,
    - one group per face to send
    - process faces concurently
  - All threads 'send' via lightweight API
    - reduce synchronization costs
  - Prepost receives way in advance
  - Poll on receives rather than block
    - faces can arrive in any order



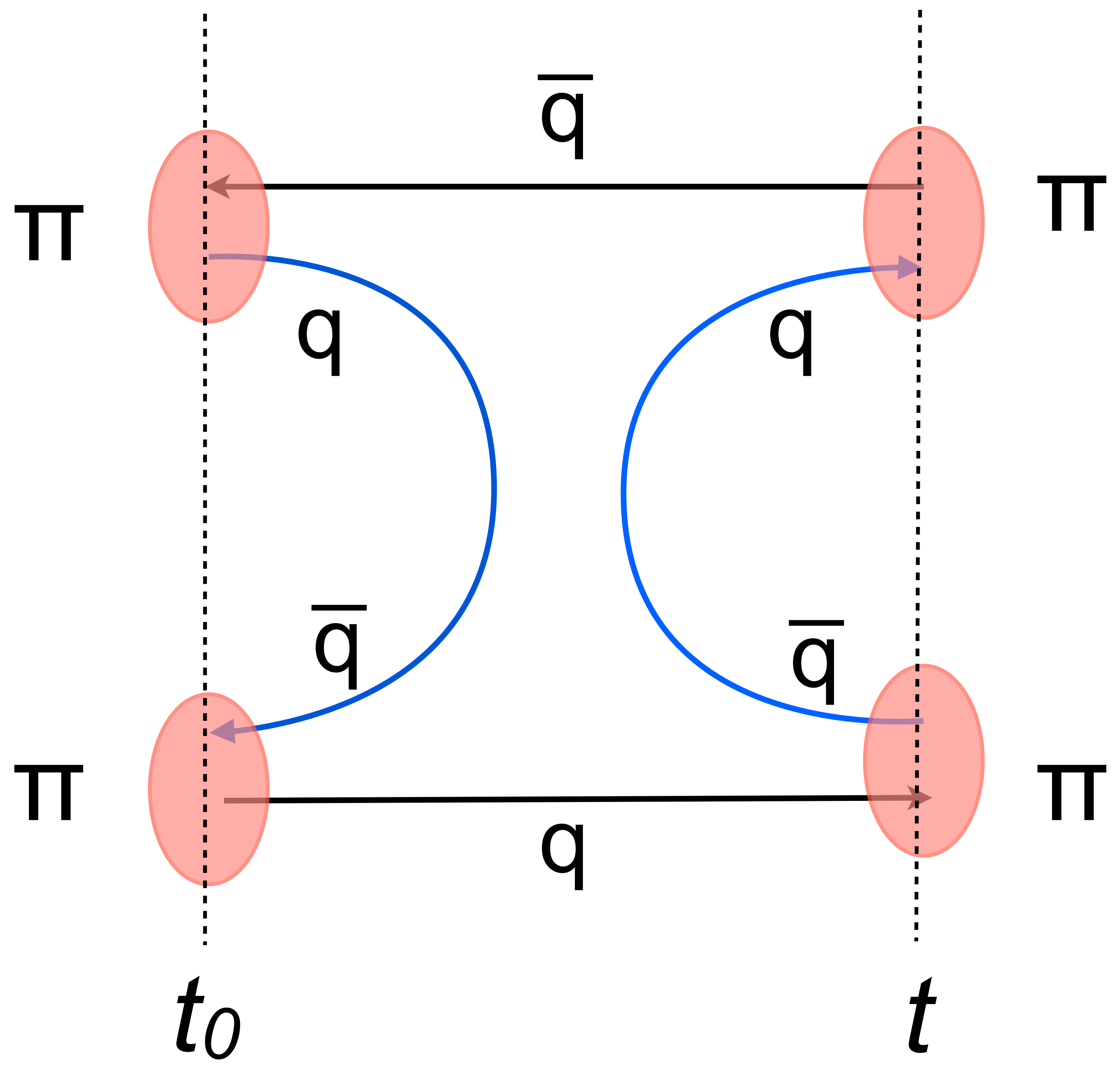
This discussion and results come from K. Vaidyanathan, et. al. "Improving Communication Performance and Scalability of Native Applications on Intel(R) XeonPhi(TM) Coprocessor Clusters", IPDPS'14

Domain Decomposition

- Split the lattice into domains
- Perform solves inside the domains
- Use this as a preconditioner for an ‘outer’ solver
- Implemented for Xeon Phi by Simon Heybrock (University of Regensburg) in collaboration with Intel & Jefferson Lab
- SC’14 paper in publication
  - unfortunately I cannot show results, until the paper is presented at SC.
- Notable differences between previous QUDA GPU implementation
  - DD method: Xeon Phi uses Multiplicative Schwarz DD, QUDA can do both Additive and Multiplicative
  - Outer solver: Xeon Phi uses Flexible GMRES with deflated restrats, QUDA uses GCR
  - Domain size: Xeon Phi uses small (cache resident) domains, QUDA uses local volume as domain

Production Running on Stampede

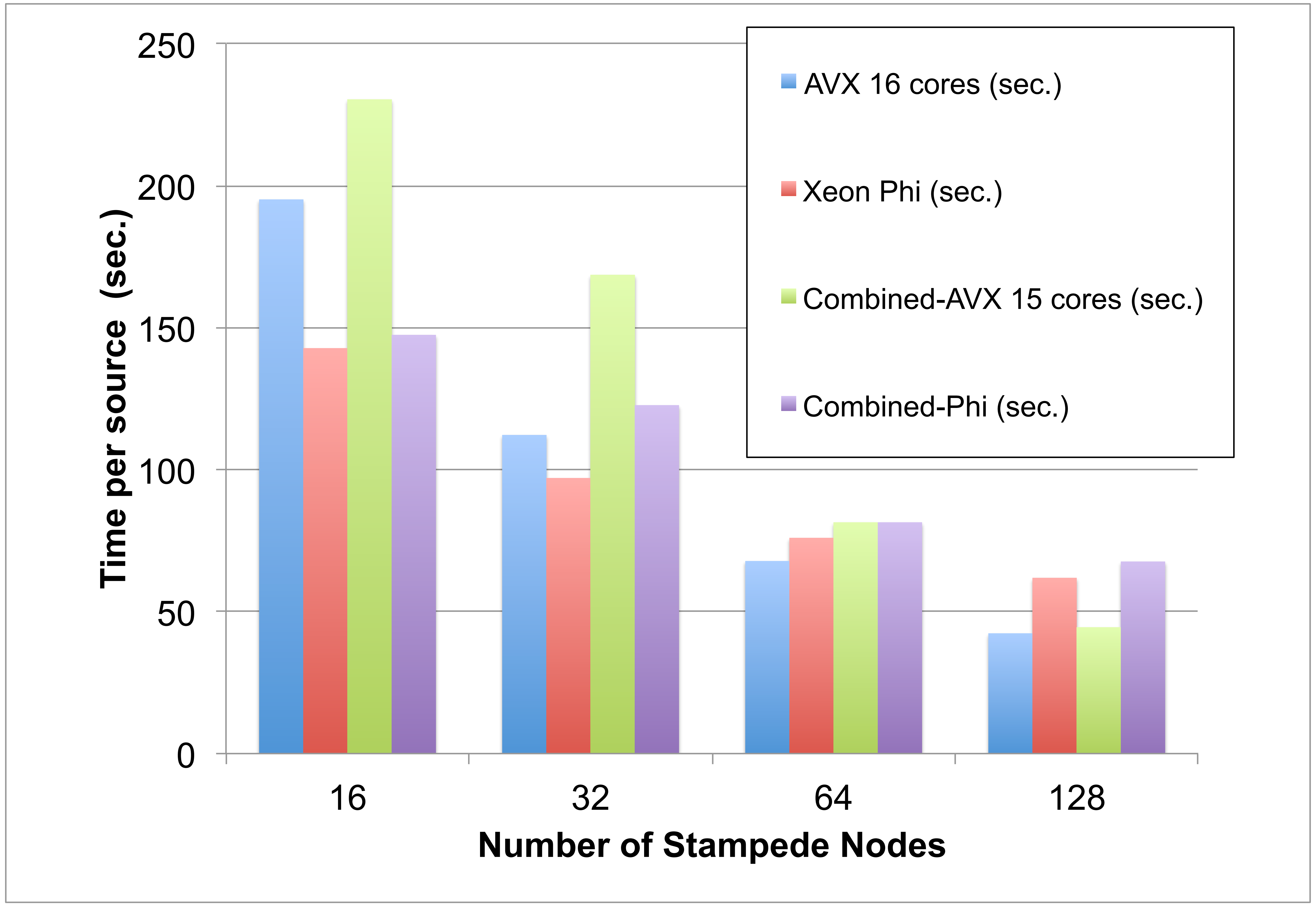
- Compute Energy Spectrum of 2-meson system
- Quark propagation from t to same t (blue) is the dominant cost.
- For every one of 220 field configurations:
  - 256 values of t
  - x 386 sources
  - x 4 values of spin
  - x 2 (light and strange quarks)
  - **= 790,528 individual solves per configuration**
- Single precision is good enough



Production Running on Stampede

- Can run on
  - either the Xeon (AVX 16 cores)
  - or on Xeon Phi
  - Combined
    - Use 15 cores for Xeon solve
    - Xeon Phi + 1 Xeon core for Proxy
- Xeon Phi performance relatively insensitive to Xeon also running
- Xeon performance degrades as
  - only 15 cores for AVX job
  - load imbalance: 8 + 7 cores

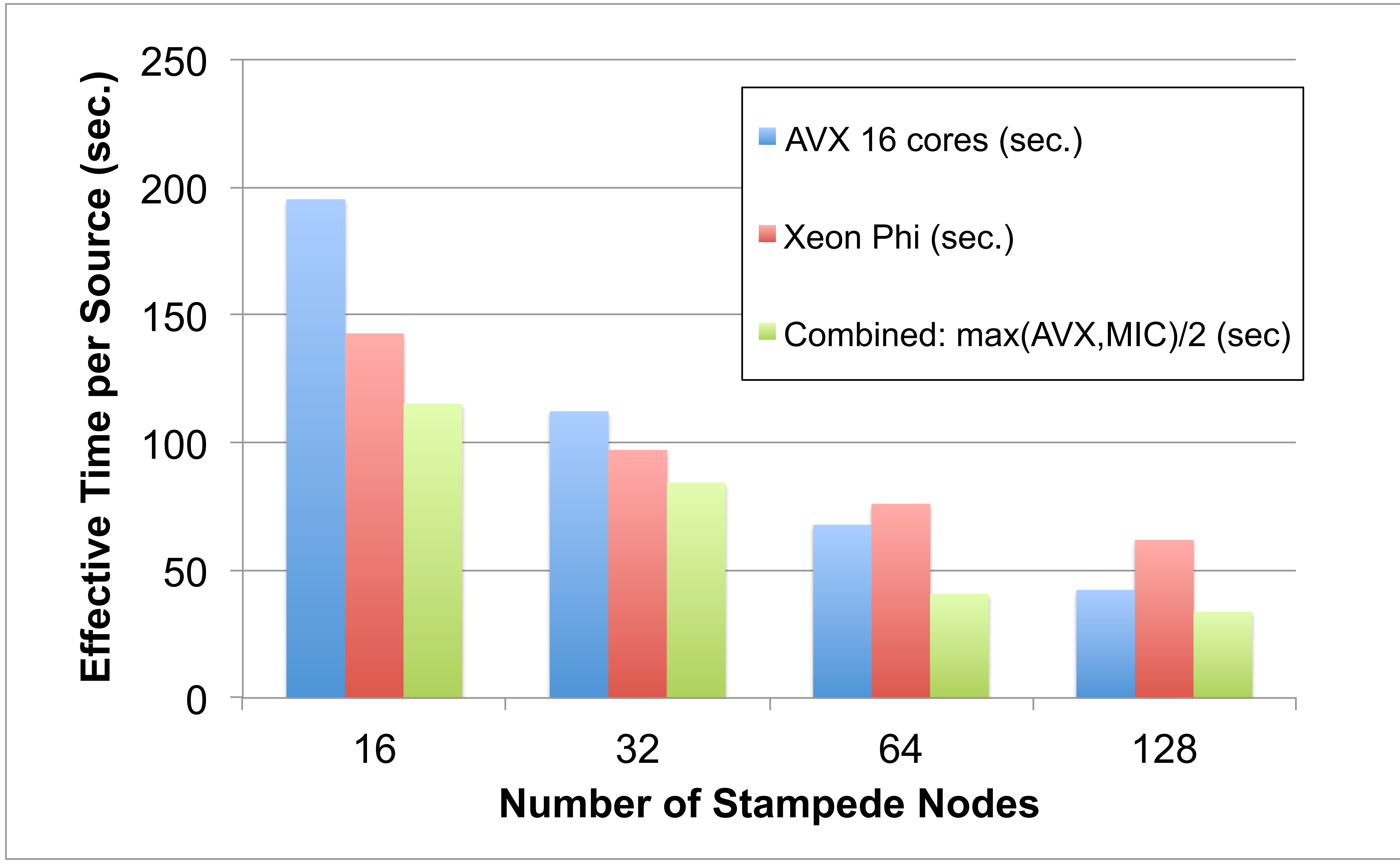
32^3x256 lattice sites, m_π~230 MeV



Production Running on Stampede

- Can run on
  - either the Xeon (AVX 16 cores)
  - or on Xeon Phi
  - Combined
    - Use 15 cores for SNB solve
    - Xeon Phi + 1 core for CML Proxy
- Consider overall throughput
  - Combined = max(AVX,MIC)/2
- Combining is better than not combining
- 2 of 256 t-s, 384 sources, light quark
  - ~20 hours on 32 nodes

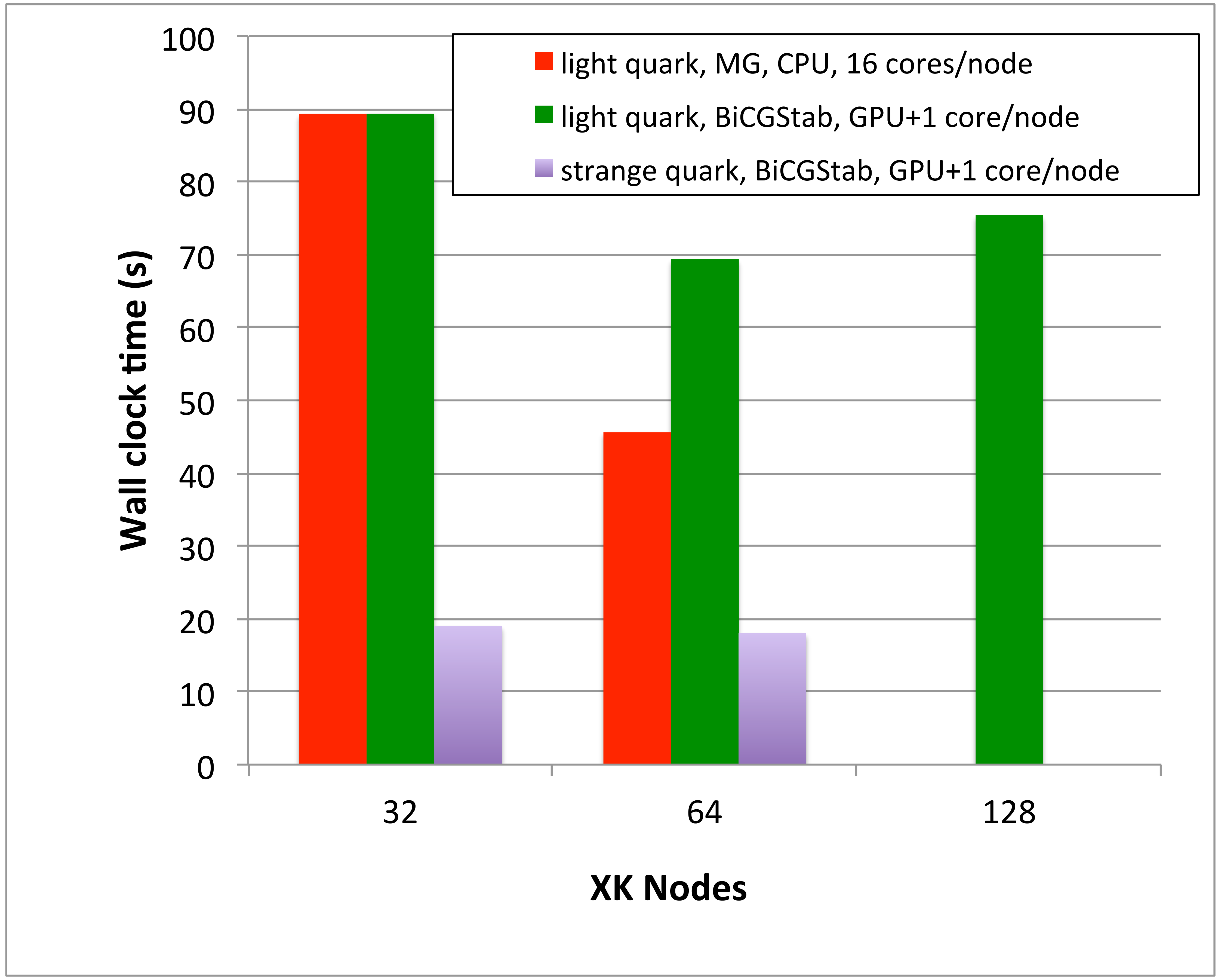
32^3x256 lattice sites, m_π~230 MeV



The Future?

- Algorithmic advances: ***Algebraic Multi-Grid*** to QCD
  - Brannick, Brower, Clark, Osborn, Rebbi, Phys. Rev. Lett. 100:041601,2008
  - Babich. et. al. Phys. Rev. Lett 105:201602,2010
  - Frommer, Kahl, Krieg, Leder, Rottmann, arXiv:1303.1377
- Multi-Grid implementation from QOPQDP+QDP/C
  - J. Osborn, PoS Lattice 2010: 037,2010, arXiv:1011.2775
  - Chroma Integration by S. Cohen, B. Joo
  - Basic build, no BLAS acceleration etc.
- On par with BiCGStab on GPUs at 32 Nodes.
  - surpass BiCGStab on GPUs at 64 Nodes
- Clearly, we want this going forward…
- Setup cost is currently expensive: careful considerations for use in gauge generation.

Timings on NCSA BlueWaters



V=40^3x256 sites, m_π~230MeV

Summary

- We have shown performance potential of Xeon Phi for QCD last year
- This year we made incremental advances: half & double prec, 4D comms, etc
- We have made successful use of the Xeon Phi-s on Stampede
  - Utilized node fully, by running separately on both Sandy Bridge and Xeon Phi parts
  - 18 M (?) SUs in 1.5 months
- Inter Xeon-Phi communication are still challenging - use proxy
- Progress also for Staggered Fermions (MILC)
- Look out for the paper on Optimizing a Domain Decomposed Solver at SC'14
- Future work
  - more and improved solvers (Multi-Grid), whole application optimization
  - consider applying Xeon Phi to non-solver parts of the analysis phase of calculations

Thanks and Acknowledgements

- We thankfully acknowledge time on
  - The Intel Endeavor Cluster
  - The Jefferson Lab 12m Cluster
  - The TACC Stampede Cluster
  - Beacon Cluster and NICS
- Computer time on Stampede through XSEDE Allocation TG-PHY130005 “Dynamical Anisotropic-Clover Lattice Production for Hadronic and Nuclear Physics”, Robert Edwards PI.
- This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.
- Funding through U.S. DOE Office of Science, Offices of Nuclear Physics, High Energy Physics and Advanced Scientific Computing Research through the SciDAC Program, and funding through the American Recovery and Reinvestment Act (ARRA)