Chroma at OLCF

A grab bag of Chroma related topics

Bálint Joó

Lattice QCD Workshop

Oak Ridge National Laboratory

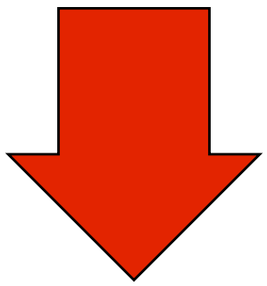
April 30, 2013

Gauge Generation

- Essential First Step of any Lattice Calculation

$$\int \mathcal{D}U \mathcal{O}(U)\ e^{-S(U)}$$

Monte Carlo Integration



Generate {U}={U_0, U_1, U_2, ... }

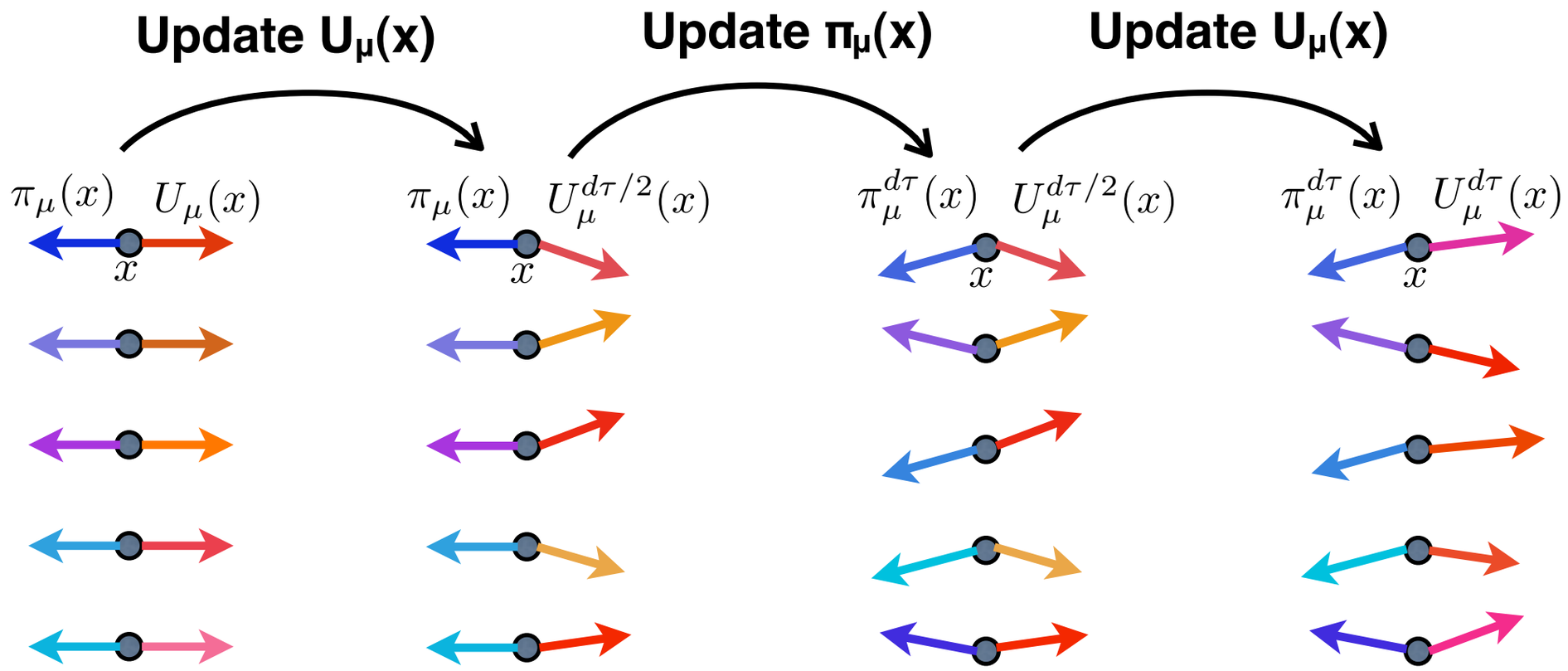
$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{O}(U_i)$$

$$P(U_i) = e^{-S(U_i)}$$

Importance Sampling

Hybrid Monte Carlo

- A.k.a Hybrid Molecular Dynamics Monte Carlo (MDMC)
- Update all links, treating them as coordinates of a Hamiltonian System
- Accept updates with Metropolis acceptance probability: P_{acc} = min(1, exp{ -(H'-H) })
- Advantage of MD: Update all links, H'-H small due to energy conservation
  - reasonable acceptance achievable, control <P_{acc}> with step-size dτ



$$H = H\left(\pi_\mu(x), U_\mu(x)\right)$$

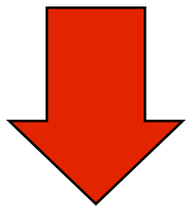
$$H' = H\left(\pi_\mu^{d\tau}(x), U_\mu^{d\tau}(x)\right)$$

Expensive Part: MD Forces

- Fermion Forces:

$$S_f = \phi^\dagger \left(M^\dagger M\right)^{-1} \phi$$

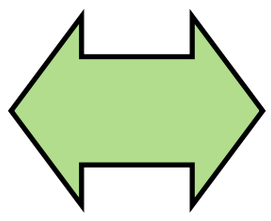
Fermionic Action



$$F = -X^\dagger \left[\dot{M}^\dagger M + M^\dagger \dot{M}\right] X$$

MD Force

$$X = \left(M^\dagger M\right)^{-1} \phi$$



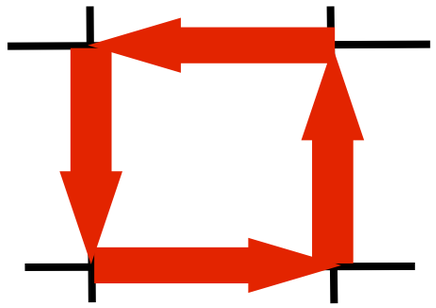
Need to Solve Dirac Equation

Computational Character

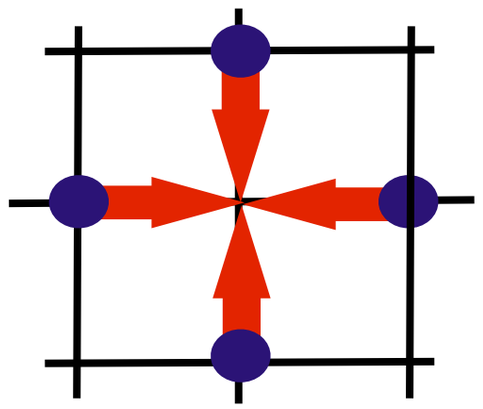
- In terms of Berkeley Dwarfs
  - Sparse Linear Algebra (Dwarf #2)
    - Dirac Equation in various forms: Large, Sparse, Complex, Linear Systems
    - Sparse Matrix is not explicitly assembled, custom SpMV directly using fields
    - Krylov methods:
      - Standard: Conjugate Gradients, BiCGStab, GCR, GMRES, + Shifted variants
      - Emerging: +DD preconditioner, +Deflation, Multi-Grid methods
  - Dense Linear Algebra (Dwarf #1)
    - 3x3 complex matrix-matrix, matrix-vector, trace, etc at each lattice site
  - Structured Grids (Dwarf #5)
    - New Multi-Grid methods add “uniform” grid-refinement (blocking)

Nearest Neighbors

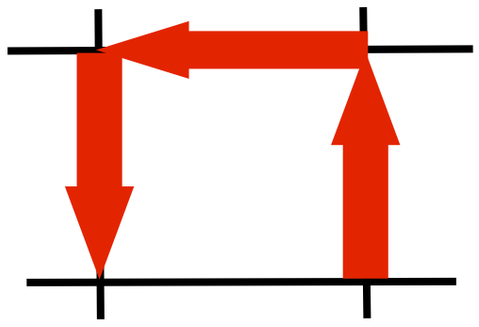
- Most communication is between nearest neighbors
- Gauge Action:
  - plaquette
- Fermion Matrix
  - Key Component: Wilson Dslash
    - AI: ~ 0.92 FLOP/byte in single prec.
    - AI: ~0.46 FLOP/byte in double prec.
    - Nearest Neighbor stencil



Plaquette



Derivative



Staple

Strong Scaling

1 for solver + 1/4 for reducing dτ to keep P_{acc} constant

$$\text{Cost} \propto V^{5/4} \left[k_1 + \frac{k_2}{(m_\pi a)^2} \right] \left(\frac{1}{a} \right)^5$$

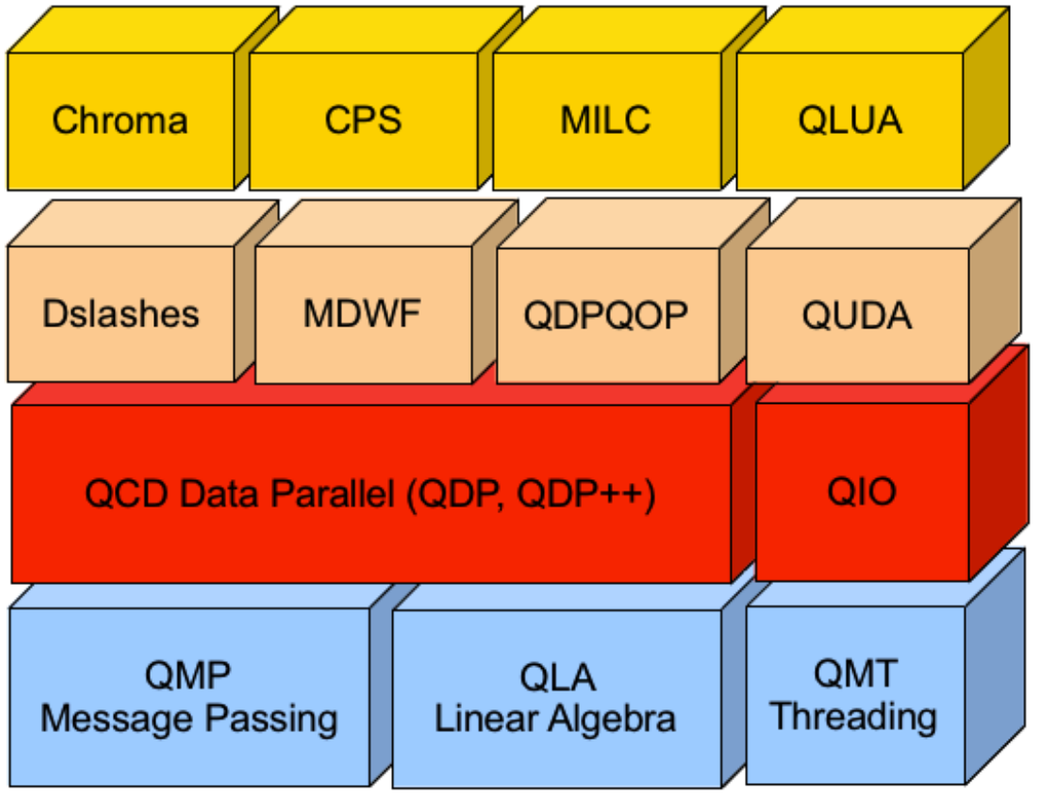
- Volume dependence of computational cost is mild
- Lattice spacing and quark mass dependence much harder
- Science dictates
  - ***finer*** (closer to continuum) lattices
  - ***physical*** quark masses
  - a necessary ***minimum*** volume
- Focus power on $\mathbf{m_\pi}$, **a** and statistics => **Strong Scaling**

Operational Characteristics

- Current CPU Running on Titan
  - Volume: 40^3x256 sites, $m_\pi \sim 230$MeV
  - 3 streams using 25600 ‘cores’ each: 76800 cores (bin #2)
  - 12-24 hour jobs, 2920 sec / traj on average
  - Typical: ~30 traj / 12 hour job
  - I/O: save 9 GB files, ~35 sec per file, ~263MB/sec
    - Writes to Lustre, with stripe-count of 10.
    - Occasionally tar these to HPSS (offsite transfer later)
  - Current INCITE Usage: 43M / 140M ~ 30%
    - We got this on the cheap tho, since charge factor is 16 ‘cores’ per node
    - Burn rate will increase when OLCF starts charging for GPUs too.

QDP++ and Chroma

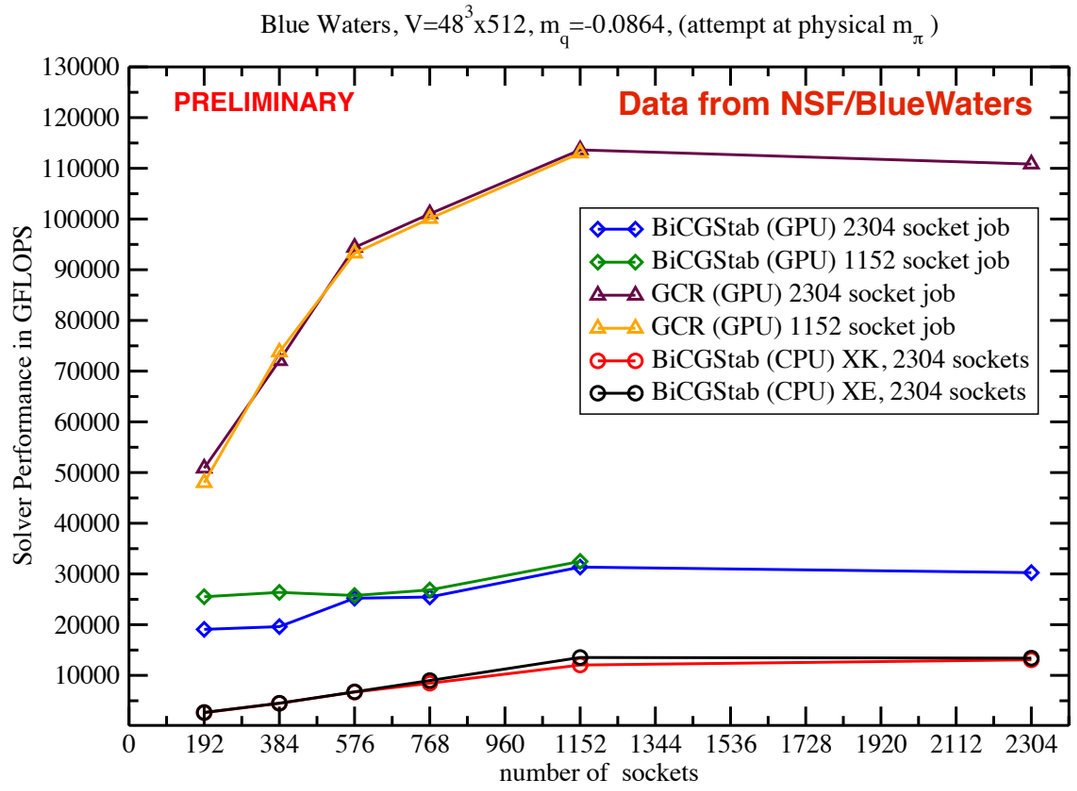
- QDP++ is a data parallel ‘layer’ in the USQCD Software stack
- QDP++ provides ‘matlab like’ expressions on QCD data-types, via “expression templates”
- Chroma is an application suite coded in terms of QDP++
- Additional acceleration from libraries:
  - Wilson Dslash operators
  - Solver libraries like QUDA

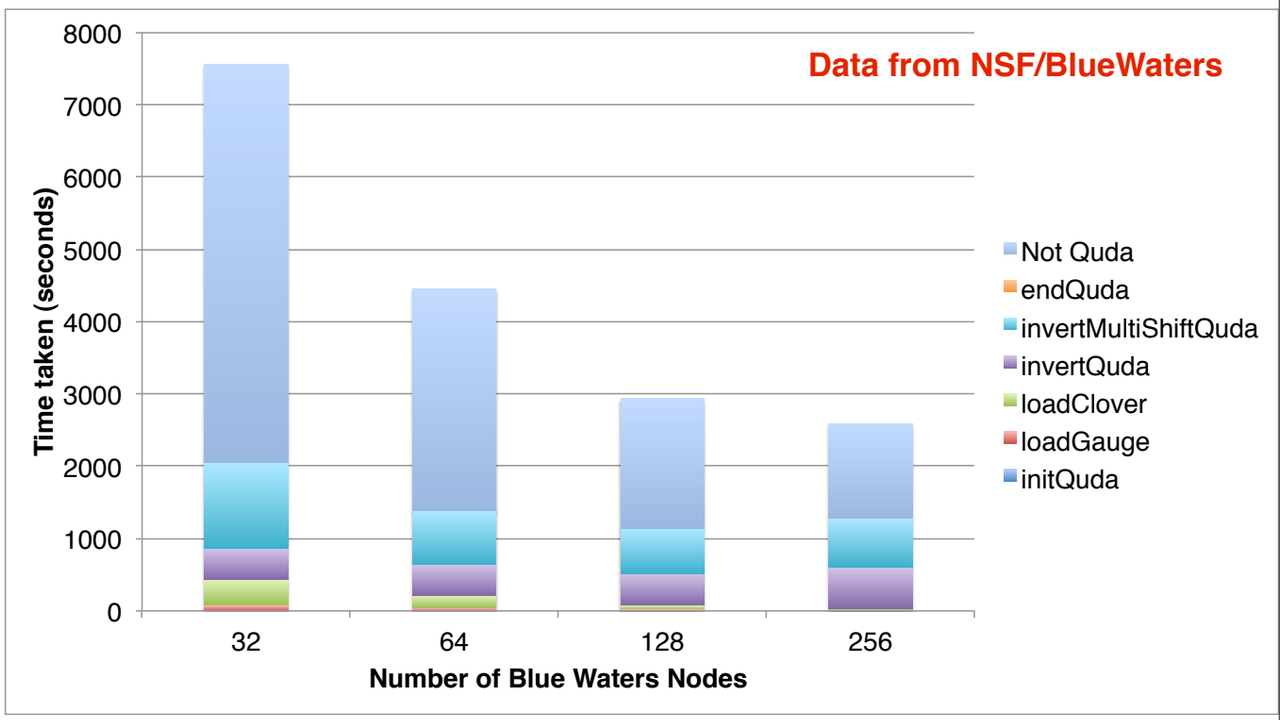


- C/C++ with OpenMP threads/pthreads
- Code Size (measured by sloccount on 4/27/13) :
  - **QDP++ ~ 137.6 KLOC** (including QIO)
  - **Chroma ~ 299.0 KLOC** (including bundled libraries)
- QMP built over MPI
- Library dependencies: libxml2 (parameter files)

Chroma On GPUs

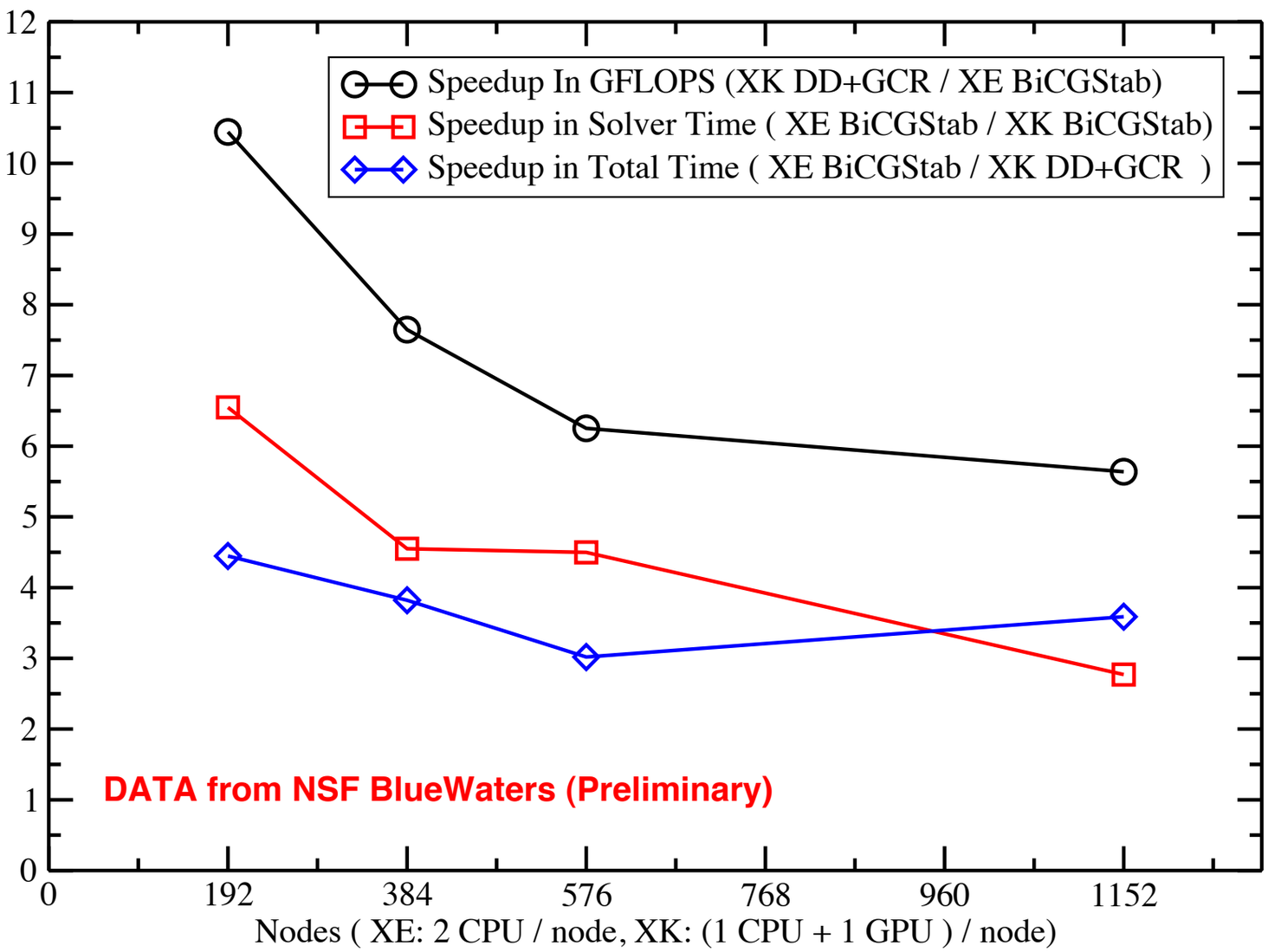
- Using GPUs since 2009 via QUDA library (Mike Clark's talk)
  - Accelerated solvers
- Recently: move all of QDP++ to the GPUs
  - QDP-JIT (F. Winter)
    - JIT/C is production ready
    - JIT/PTX is full featured
      - some interfacing with QUDA remains
      - work in progress (almost complete)
  - Titan porting testing via LGT006 discretionary project (Thank You!)
  - Friendly/Early use of TitanDev





Variety of Speedups

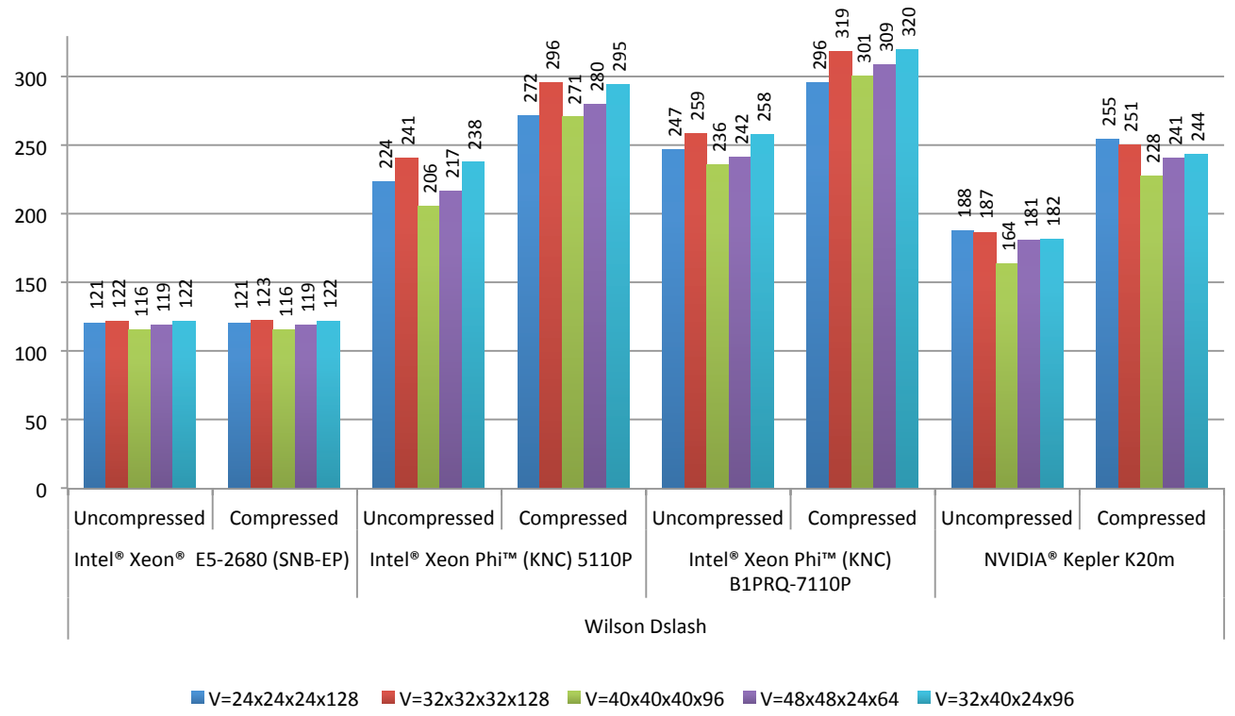
- Quantify Speedup in either **GFLOPS** or **Wallclock** time
- Speedups tend to decrease as jobs get larger:
  - Strong Scaling effects (S/V)
  - Algorithmic improvement from Domain Decomposition is **INCLUDED** here
- Whole app speedup different from solver speedup. Suspect:
  - Amdahl's law effects
  - performance variations
- Summary: ~3-4x at scale in wallclock time

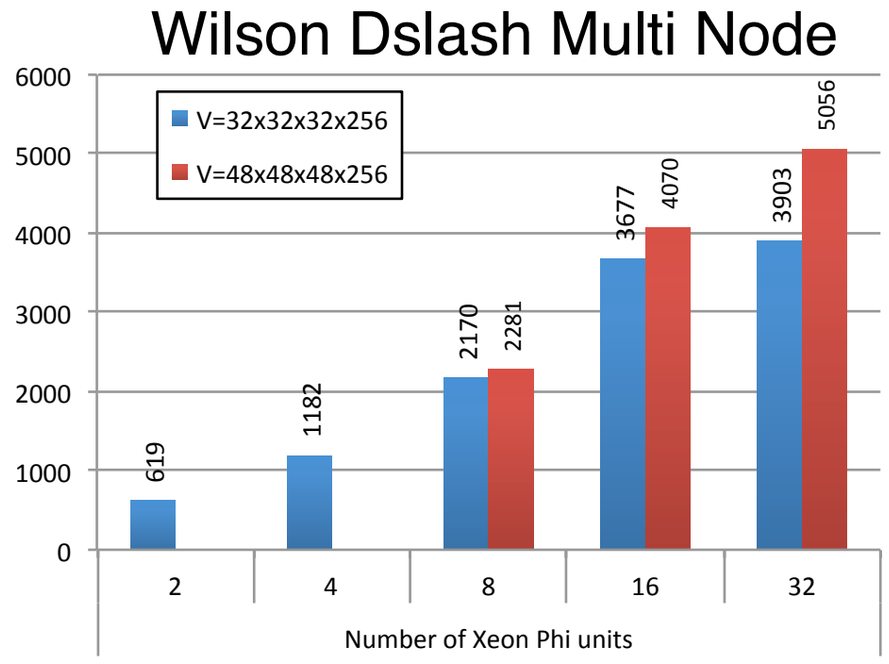


Preliminary: error bars needed

Xeon Phi Experiences

- In collaboration with Intel Parallel Labs
  - M. Smelyanskiy, D. G. Kalamkar, K. Vaidyanathan
- Achieving High Performance needed:
  - vectorization tricks
  - cache blocking
  - block-to-core mapping
  - L2 prefetching in software
- Performance portability to AVX via 'code generator'
- Ninja code: 1 Xeon Phi ~ 4 x SNB sockets
- Non-Ninja code: 1 Xeon Phi ~ 2 sockets

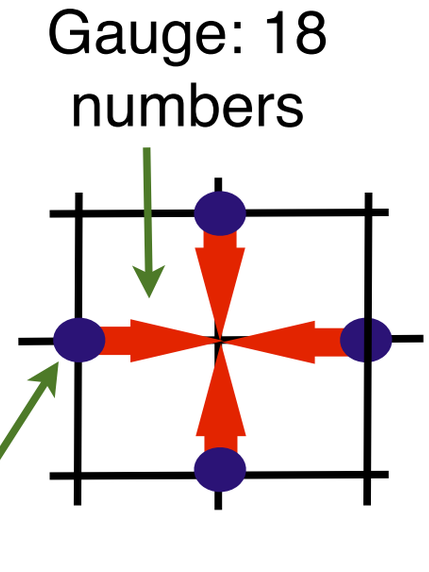




From: B. Joo, D. D. Kalamkar, K. Vaidyanathan, M. Smelyanskiy, K. Pamnani, V. W. Lee, P. Dubey, W. Watson III, "Lattice QCD on Intel(R) Xeon Phi(tm) Coprocessors", Proceedings of ISC'13 (Leipzig) Lecture Notes in Computer Science Vol 7905 (to appear),

Future Architectures

- Our primary desire from a future architecture is probably a good **balance** between memory and internode bandwidth.
  - Would also like stability & predictable performance (talk by DeTar)
- Simple model: Scaling of Wilson-Dslash Operator
  - Nearest neighbor ‘stencil’ in 4 Dimensions
- Assume:
  - $2L^4$ sites on a node, L^4 after checkerboarding
  - No reuse of gauge links
  - Maximum spinor reuse (load 1 new spinor for every lattice site)
  - Compute for body can be overlapped with the memory traffic, mem B/W is B_m
  - All faces communicated concurrently with B/W: B_n per face
  - Total network bandwidth $B_N=16B_n$ ((send+receive) x (forward+back) x 4 dims))
  - latencies are negligible



Scaling Example

- **Face Size: 12 L^3 sizeof(F) , Body Size: 192 $(L-2)^4$ sizeof(F)**
- **Face Comms Time: 192 L^3 sizeof(F) / B_N**
- **Body Compute Time: 192 $(L-2)^4$ sizeof(F) / B**
- **Face Time /Compute Time = L^3 B_m/ $(L-2)^4 B_N$**
- To overlap compute with comms need: **$B_N/B_m \lesssim L^3/(L-2)^4 \sim 1/L$**

	B_m (GB/s)	B_N (GB/s)	L_{min}	B_N/B_m	$L^3/(L-2)^4$	V_L sites	Nodes for 96^3x256 lattice
Accelerator like	***180***	***16 (PCIe2)***	***16***	***0.09***	***0.11***	***$32x16^3$***	***1728***
"CPU" like	***45***	***16 (PCIe2)***	***8***	***0.36***	***0.39***	***$16x8^3$***	***27,648***

- Caveats:
  - not the whole story: reduced communications algorithms (e.g. DD+GCR) help
  - Hardware improvements: e.g. as in this presentation (move fabric onto chip, like BG/Q)

Scaling Example

- **Face Size: $12 L^3$ sizeof(F) Body Size: $192 (L-2)^4$ sizeof(F)**
- **Face**
- **Body**
- **Face**
- To ove

- Moral of this example:
  - Improve B_N/B_m by **4x** (by reducing B_m)
  - Lose **4x** in body compute
  - Gain **16x** in scalability
  - Overall **4x** speedup
- More Ideal Scenario: Keep high B_m & improve B_N by **4x**
  - would lead to overall 16x speedup
  - but this simple example doesn't consider power cost for network

							odes for $96^3 x256$ lattice
Accelerator like							***1728***
"CPU" like	***45***	***16 (PCIe2)***	***8***	***0.36***	***0.39***	***$16x8^3$***	***27,648***

- Caveats:
  - not the whole story: reduced communications algorithms (e.g. DD+GCR) help
  - Hardware improvements: e.g. as in this presentation (move fabric onto chip, like BG/Q)

Porting/Future Architectures

- Rough Effort estimates for Porting
  - On GPUs
    - QUDA developing since 2008/2009(?): **4-5 calendar years**
    - QDP-JIT: Since Dec 2009: **2 and 1/3rd FTE year, just over 3 calendar years**
  - On Xeon Phi
    - Chroma compiled ‘out of box’ but needs development for higher efficiency:
      - ‘parscalarvec’ work by Jie for example: vector friendly layout, more pervasive threading
    - Dslash work with Intel took off about Mid April 2012
      - So far at most 1 FTE year between self, Jie and Intel colleagues
    - Lots left to do: double precision, optimized clover, more work in QDP++, etc
    - In total **2-2.5 FTE years estimate seems not unreasonable**.

Porting/Future Architectures

- Does it take 2-4 years to stand up to a new architecture?
- It took about 2-3 years for Chroma to initially stabilize... (started 2002)
  - but architectures were then ‘stable’ for about 7-8 years
    - MPP with MPI/QMP: QCDOC, BG/L, BG/P, Cray XT, Xeon/AMD IB Clusters
- 4 years is about the lifetime of a leadership computer...
- Lessons:
  - Vendor Partnerships **really** critical: e.g. IBM, Intel, NVIDIA
  - Partnerships/Communication with LCFs is **really** critical.
    - help us prepare, make important decisions re. software, advocate our needs to vendors
  - rewrite needs payoff guarantee to be worth it
    - e.g. if one needs to take on a radically different programming model
  - otherwise porting is preferable - preserve investments

Conclusions

- BJ: Happy & Friendly USQCD user of OLCF **since 2007**
  - also of NICS (mostly Kraken and development on Keeneland)
- Chroma + QDP JIT/PTX + QUDA is well poised to use Titan
- Going forward
  - We need closer relationships to the LCFs
    - Especially if Hardware/Sfw environment is heavily site specific (e.g. interconnect) and is not available on the general commodity computing market
    - We need to be involved/informed very early to have time to stand up production ready code.
  - We need to continue our excellent relations with Vendors
    - We are willing to work under NDAs if needed.
- Looking forward to working with all stakeholders to keep LQCD viable and vibrant on future platforms on the road to Exascale, and to advance the USQCD Science program