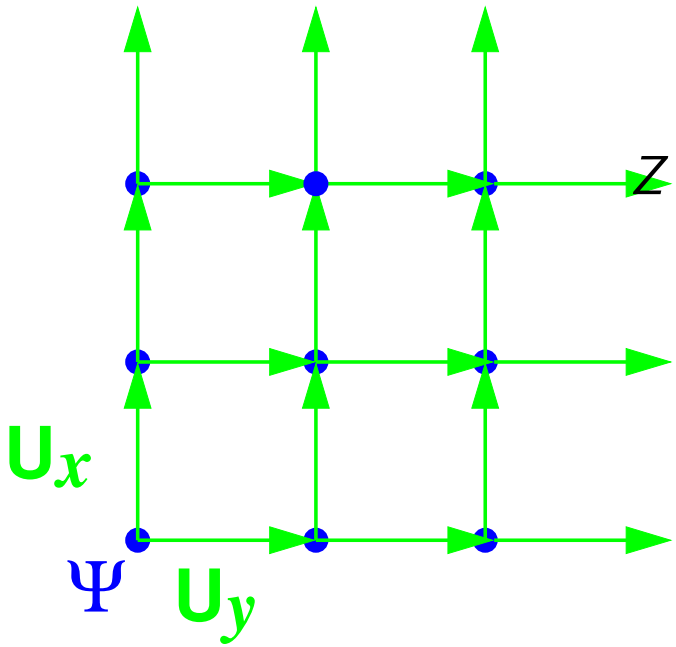
BG/Q Programming Challenges

Chulwoo Jung
Brookhaven National Laboratory

November 10, 2011
USQCD Algorithms and Computing Workshop
HPCIC, Livermore, CA

Introduction: lattice QCD

Quantum ChromoDynamics (QCD): Theory of strong interaction which governs interaction between quarks and gluons.
In contrast to Quantum Electrodynamics (QED), The effective coupling of QCD decreases in high energy, hence is calculable by hand, but not in low energy. $\rightarrow$ Nonperturbative techniques such as lattice QCD is needed for *ab initio* calculations. $(\psi(x), A_\mu(x)) \rightarrow (\psi(n), U_\mu(n) = \exp(-iA_\mu))$



$$Z = \int [dU] \det(\not{D} + m) \exp(-(S_g))$$

$$= \int [dU][d\bar{\psi}][d\psi] \ \exp(-(S_g + S_f))$$

$$S_f = \bar{\psi}(M^\dagger M)^{-1}\psi, \qquad S_{eff} = S_g + S_f$$

$$S_g = \beta \sum \left[(U_\mu(x) U_\nu(x+\hat{\mu}) U_\mu^\dagger(x+\hat{\nu}) U_\nu^\dagger(x)\right]$$

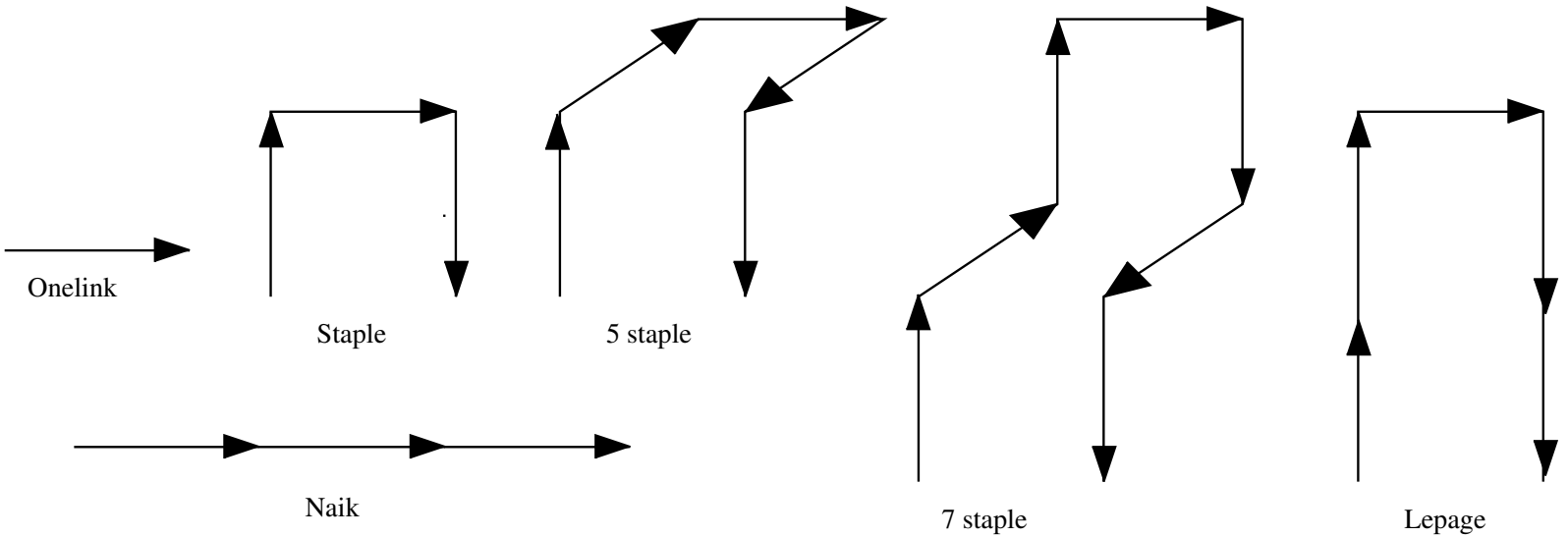
Various improvements

- Replace simple link and /or plaquette operator with more complicated operator to control various finite lattice spacing effect(Asqtad,HISQ,Iwasaki,DBW2)

Fermion action $S_f = \sum \bar{\psi}(M^\dagger M)^{-1}\psi$

Fermion force $\frac{\partial S_f}{\partial U_\mu} = M^{-1}\frac{\partial M}{\partial U_\mu}(M^\dagger M)^{-1}\left(\sum \psi(x+\mu)\bar{\psi}(x)\right)\cdots$

$M(x,y) =$



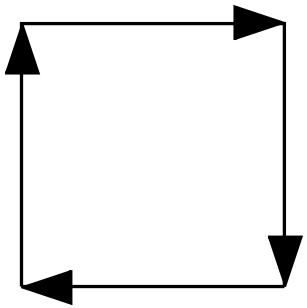
$$S_g = \beta \sum \left[(U_\mu(x) U_\nu(x+\hat{\mu}) U_\mu^\dagger(x+\hat{\nu}) U_\nu^\dagger(x) \right]$$

$$+\beta' \sum \left[(U_\mu(x) U_\mu(x+\hat{\mu}) U_\nu(x+2\hat{\mu}) U_\mu^\dagger(x+\hat{\mu}+\hat{\nu}) U_\mu^\dagger(x+\hat{\nu}) U_\nu^\dagger(x) \right]$$

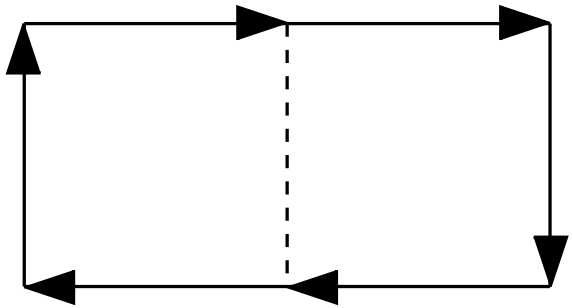
$$+\beta'' \sum \left[(U_\mu(x) U_\mu(x+\hat{\nu}) U_\rho(x+\hat{\mu}+\hat{\nu}) U_\mu^\dagger(x+\hat{\mu}+\hat{\rho}) U_\nu^\dagger(x+\hat{\rho}) U_\rho^\dagger(x) \right] \cdots$$

$$\frac{\partial S_g}{\partial U_\mu} = \beta \sum_\nu \left[U_\nu(x+\hat{\mu}) U_\mu^\dagger(x+\hat{\nu}) U_\nu^\dagger(x) + \right.$$

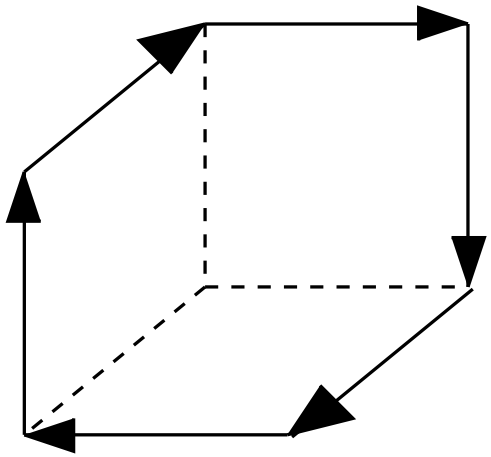
$$\left. U_\nu^\dagger(x+\hat{\mu}-\hat{\nu}) U_\mu^\dagger(x-\hat{\nu}) U_\nu(x-\hat{\nu}) \right] \cdots$$



Plaquette

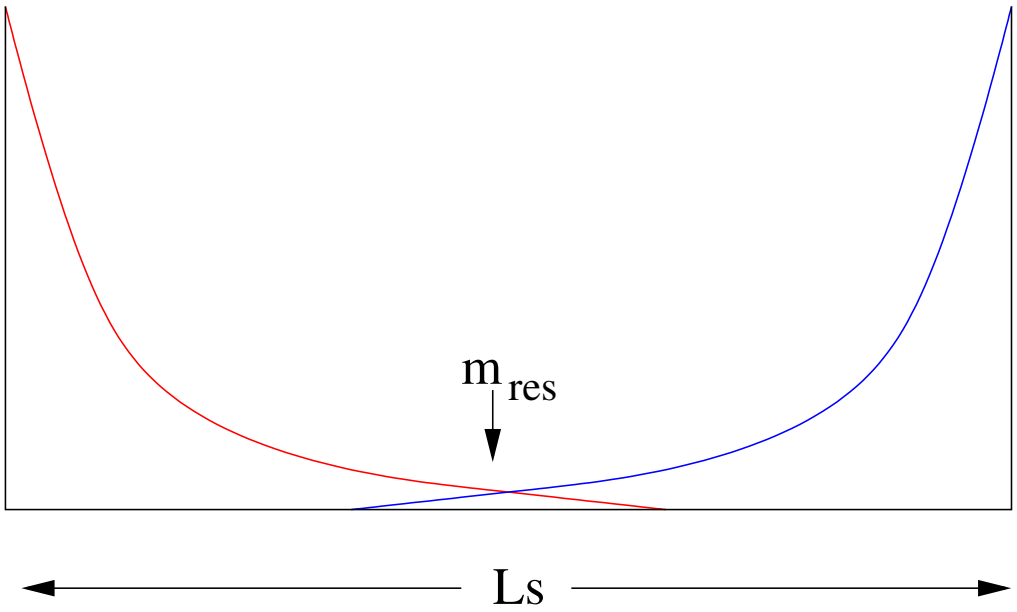


Rectangle



Chair

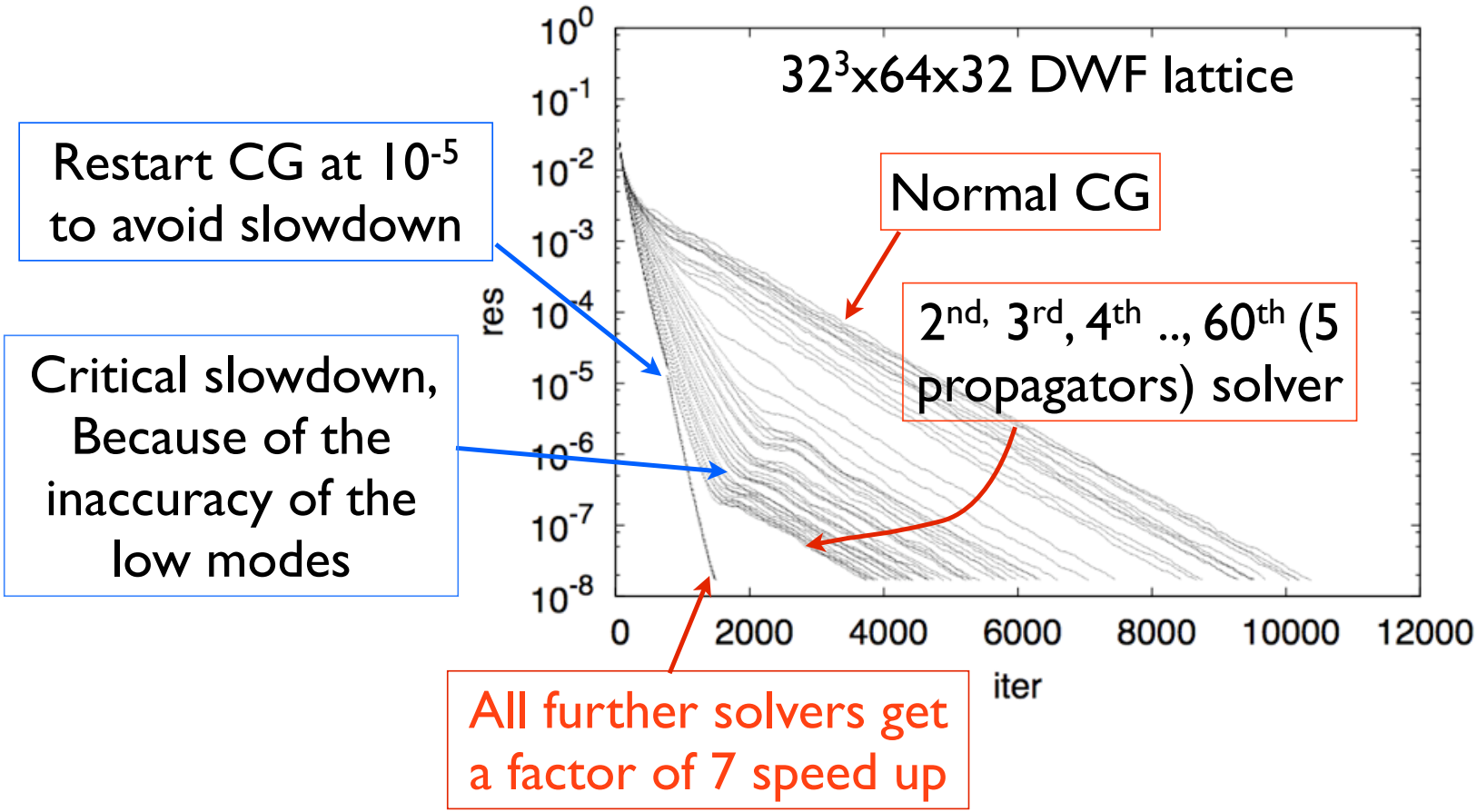
- Domain Wall Fermion: Improved chiral symmetry by introducing 5th dimension. Cancel propagating modes by additional (Pauli-Villars) fields with antiperiodic boundary condition in 5th dim. Left with exponentially decaying modes.



EigCG (Stathopoulos & Orginos, SIAM J. Sci. Comput. 32, p439(2010))
Multigrid in the 5the directions works (Möbius acceleration)

EigCG

EigCG accumulates low modes during the regular CG iterations, and uses these low modes to speed up later solvers.



Thursday, November 3, 2011

IBM BG/Q

	BG/L	BG/P	BG/Q (Approx.)
Core/chip	2	4	16+1(+1)
SIMD/core	2	2	4
Clock speed(Mhz)	700	850	1600
Threads/chip	2	4	64
Peak/Rack(TFlops)	5.6	13.9	209
L1/core(KB)	32	32	16
L3/chip(MB)	4	8	32 (L2)
Memory(GB)/node	1	2	16
DDR Bandwidth(GB/s)	5.6	13.6	40
Torus Dimension	3	3	5
Bandwidth (GB/s)	2.1	5.1	40(Nominal)
Watt/Rack (KW)	∼20	∼40	∼80
Linpack GF/W	0.23	0.37	1.68

Scheduled deployments

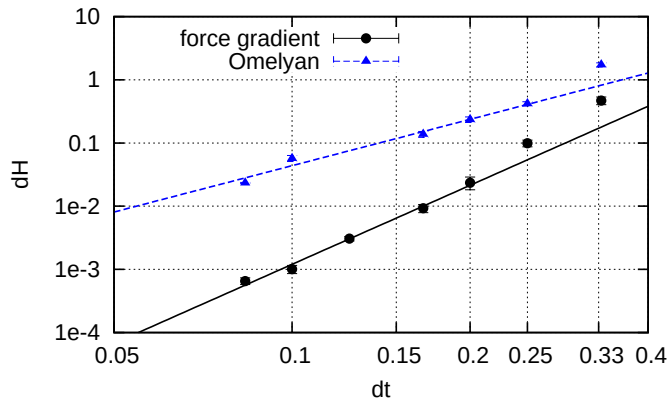
- 1 Midplane (512 node), #109 in Top 500 (June 2011) @ IBM Yorktown
- Planned
  - 4 rack @ EPCC in 2011.
  - 2 rack(DD1) @ RBRC(BNL) in 2011.
  - 1 rack @ BNL in 2011
  - MIRA (48? Rack, $\sim$ 10PTFlops) at ANL in 2012.
  - Sequoia (96 Rack, $\sim$ 20PTFlops) at LLNL in 2011?
- ? Racks for USQCD @ BNL in 2012,3? (Benchmarks critical in decision process)

LQCD on BG/Q

Integrated to Columbia Physics System(CPS)
64 Threads available per nodes can be utilzed with various combinations of MPI and openMP/pthread. IBM is commited to openMP. So far tested with 1(MPI) $\times$ 64(thread) only.

- Peter Boyle's DWF inverter has been run since early stages of BG/Q development.
  - Both 5D, 4D preconditioning available.
  - Utilizes 64 threads.
  - 5D version integrated into CPS. Regression testing done.
- 4D precondition/Möbius integration progressing.
- Rational HMC, Omelyan, Force Gradient integrator,....
- Gauge force, Fermion force,... for RHMC Mostly written with multi-directional, 1-hop, parallel tranport
  - Currently implementing openMP. Plans to add SIMD aseembly/instrinsics. performance by openMP-ing individual routines.

Scaling Behavior of the Force Gradient Integrator (FGI)



- The fitting suggests the following scaling behavior of the integrators:

$$\begin{array}{cc} \text{FGI} & \mathcal{O}(\tau^{4.16\pm0.21}) \\ \text{Omelyan} & \mathcal{O}(\tau^{2.44\pm0.21}) \end{array}$$

- FGI allows the top level step size to be increased to $1/3$ compared with $1/4$ in the Omelyan integrator.
- With the first and last updaing step combined, FGI requires solving 3 Dirac equations while Omelyan requires 2 each step.
- Due to the extra cost, FGI does not show much improvement over the Omeylan integrator on small lattices.

- CPS already well instrumented to profile routines at coarser level. Combining this with GNU (flat) profile proved useful.
  - Currently DWF/Twisted wilson inverters is consuming $\sim 40\%$ of total time on 128 nodes BG/Q, in constrast to $\sim 90\%$ on BG/P.

Time breakdown on BG/P

$32^3 \times 64 \times 32$ DWF+ID, $m_s = 0.045$						
2048×4 core BG/P			4096×4 core BG/P		8192×4 core BG/P	
m_l	0.0042		0.001		0.001	
Local volume	$4^4 \times 32$		$4^4 \times 16$		$4^3 \times 2 \times 16$	
Routines	time(sec)	MFlops/s	time	MFlops/s	time	MFlops/s
MInv	4073	488	854	473	499	405
CG	3438	538	2917	510	1693	444
DSDR	199	503	254	376	218	220
GF	137	344	78	156	45	137
RF	643	71	122	68	65	64
HF	12	27	166	6	87	6
Total time(s)	8888		4564		2716	

Considerations for optimizing for BG/Q

- SIMD obviously important.
- Factor of 64 is big! trivial routines like field import/export can be significant. Profiling is useful.
- typical operations needed for evolution (SU(3) Matrix multiply): 108 (m+a) operations / 2*8*18 Bytes $\sim$ 6 flops/8 bytes. Compared to 209(GFlops/s)/40(GB/s) $\rightarrow$ DDR bandwidth is bottleneck, reusing cache effectively is important. Combining local (no shift) operations likely very important. Inverter is likely to be more efficient for relatively small volume.
- Thread creation/destruction overhead CG/MInv $\sim$ 10000 times, Optimized DWF/Wilson keeps threads across different iterations. GF/FF/smearing: order of 1 $\sim$ 100
- Multi directional communication hardware: decrease overhead by doing comms multiple directions simnulateously

Optimization strategy for gauge evolution?

- Optimization strategy could be quite different for different local volumes & architectures.
- Inverters: Keep threads as much as you can, combine operations as much as you can.
- GF/FF/Smearing: Great if we can use level 2 routines to get close to hand-optimized routines. Should support multidimensional parallel-transport, which allows for amortizing overheads from communication and threading, maximize communication bandwidth.

pt($X'[], X[], U[], \mu[], N$,.....) {
  int i;
  for($i = 0; i < N$;i++){
    $X'[\mu[i]] = U_{\mu[i]} X[\mu[i]]$
  .....
}

```
const int N = 4;
Matrix *Units[4];
for(int i = 0;i<N;i++) Units[i] = Unit;
int mu,nu;
{
  int dirs_p[] = {0,2,4,6,0,2,4};
  int dirs_m[] = {1,3,5,7,1,3,5};
  ParTransGauge pt(*this);


    for(nu = 1;nu<4;nu++){
      pt.run(N,tmp1,Units,dirs_m+nu);
      pt.run(N,result,tmp1,dirs_m);
      pt.run(N,tmp1,result,dirs_p+nu);
      for(int i = 0; i<N;i++)
      vaxpy3_m(tmp2[i],&tmp,tmp1[i],tmp2[i],vol*3);
      pt.run(N,tmp1,Units,dirs_p+nu);
      pt.run(N,result,tmp1,dirs_m);
      pt.run(N,tmp1,result,dirs_m+nu);
      for(int i = 0; i<N;i++)
      vaxpy3_m(tmp2[i],&tmp,tmp1[i],tmp2[i],vol*3);
      ForceFlops +=vol*12*N;
    }
    pt.run(N,result,tmp2,dirs_p);
}
```