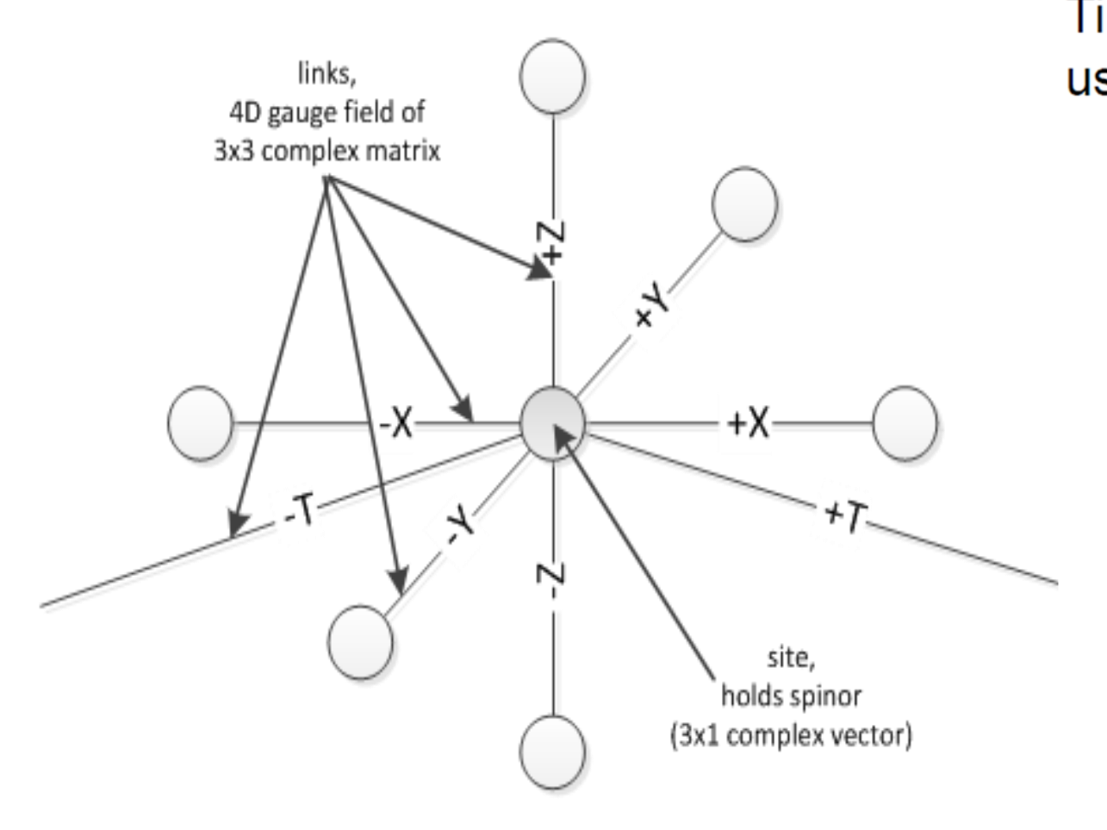
Multi-GPU Implementation of MILC using QUDA Framework

Guochun Shi[†], *Steven Gottlieb*[‡†], *Aaron Torok*[‡], *Volodymyr Kindratenko*[†]
[†]National Center for Supercomputing Applications, University of Illinois, Urbana, IL, USA
[‡]Department of Physics, Indiana University, Bloomington, IN, USA

MIMD Lattice Computation (MILC)

- The MIMD Lattice Computation (MILC) code, a Quantum Chromodynamics (QCD) application used to simulate four-dimensional SU(3) lattice gauge theory, is one of the largest compute cycle users at the national supercomputing centers. The code is scalable to thousands of processors, however its per-processor peak floating-point performance remains low [see the performances in the table below].
- We implemented the conjugate gradient (CG) solver, which is the most time-consuming part of MILC, to run on a GPU cluster using the QUDA framework[1][2][3] developed at Boston University.
- We also implemented other time-consuming parts of MILC to run on GPUs, such as the fermion force, gauge force and fat link computation.

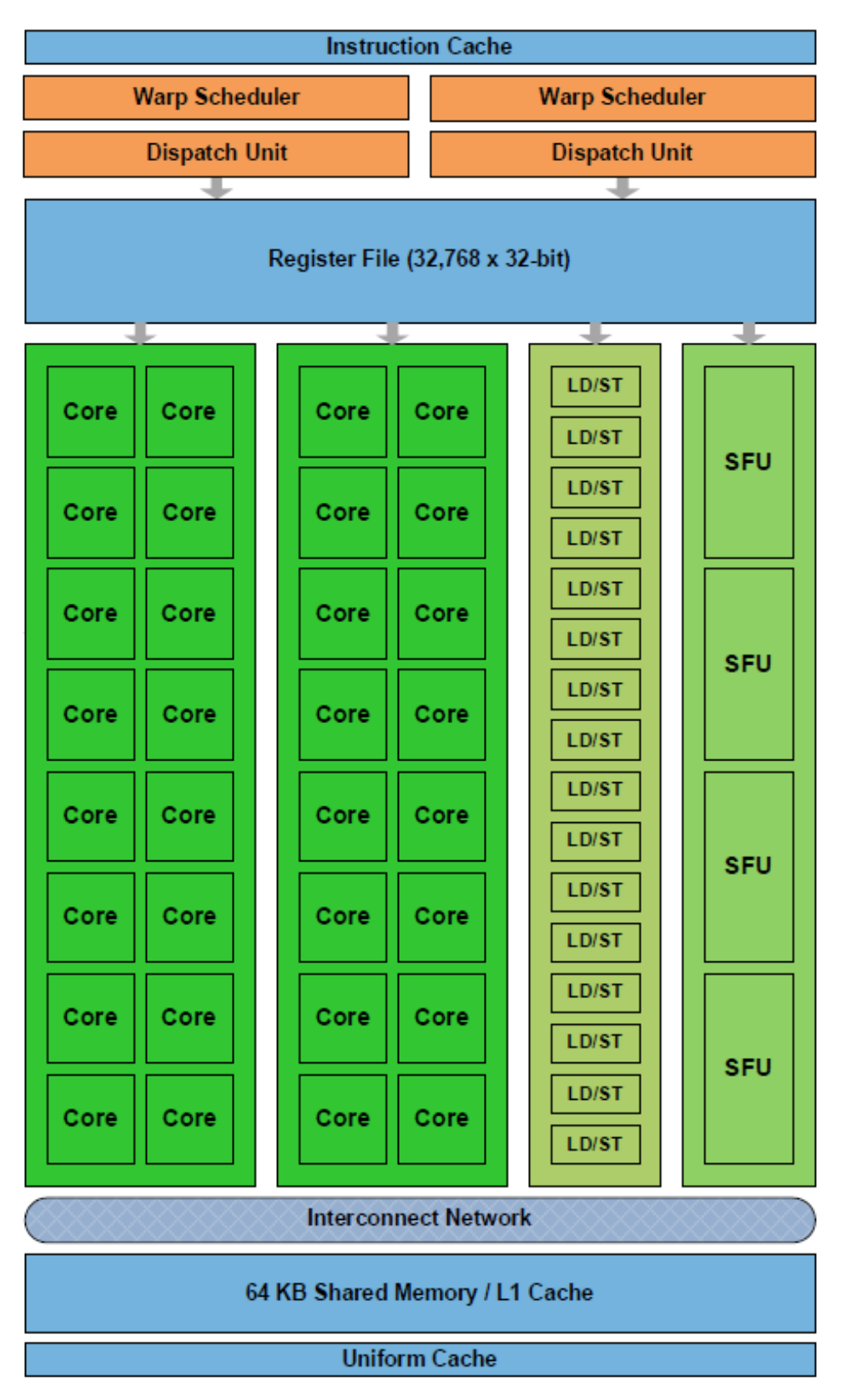


Time distribution for a run on 2048 XT3 (BigBen) cpus using a $40^3 \times 96$ grid ($5 \times 10^2 \times 6$ per cpu) with $m_l = 0.1m_s$:

Activity	time(s)	MF/cpu	per cent
CG	2987	530	58.5
FF	1125	579	22.0
GF	489	469	9.5
Fat	442	627	8.7
Long	24	340	<1
Input config.	41		<1
total above	5108		
unaccounted	104		1.9
wallclock	5212		

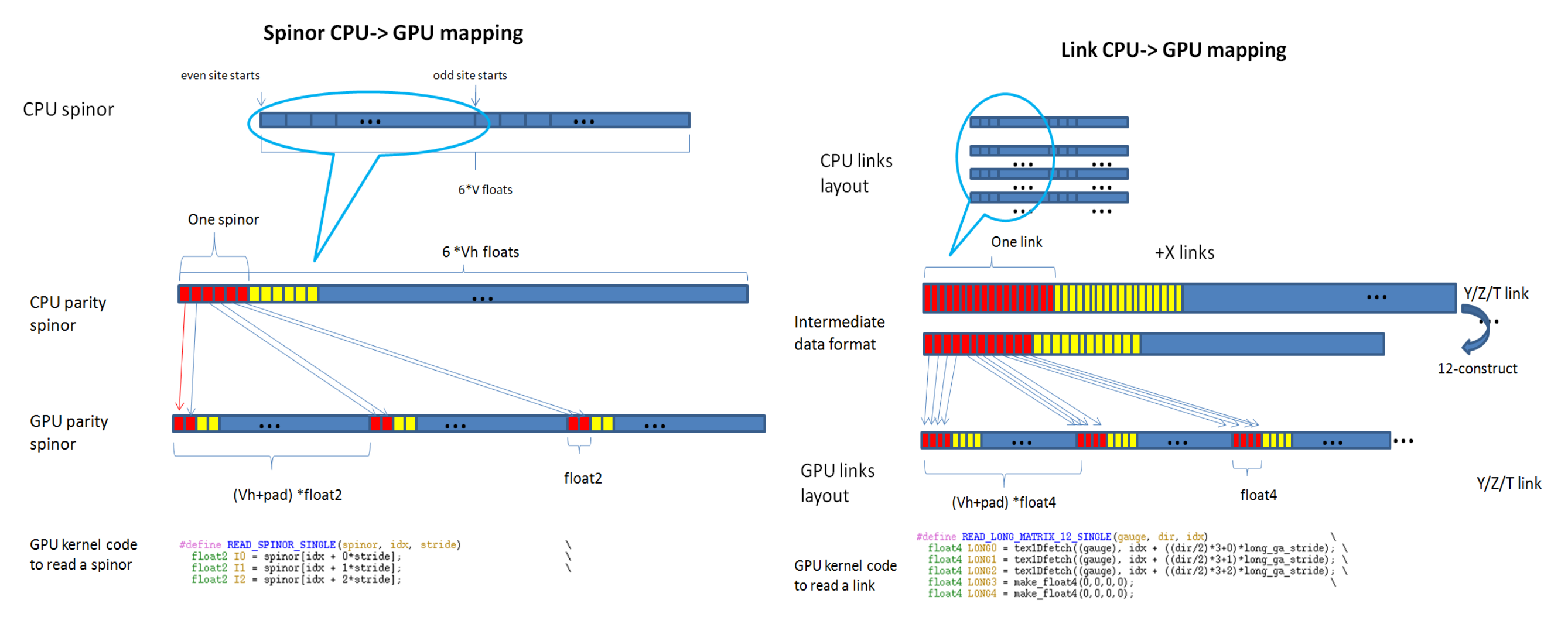
NVIDIA Fermi architecture

- Fermi architecture (C2050)
  - 14 Streaming Multiprocessors, each containing 32 Streaming Processors
  - At 1.15 GHz this provides
    - 1,030 GFLOPS (SP)
    - 515 GFLOPS (DP)
  - 384-bit interface to off-chip GDDR5 memory
    - 3.0 GB without ECC
    - 2.6 GB with ECC
    - 144 GB/s bandwidth
  - PCIe gen 2 interface to the host
    - 8 GB/s bandwidth
- CUDA programming model
  - Programs are composed of host code and GPU kernels
    - The compute-intensive part (GPU kernel) is executed on the GPU
    - The rest of code is executed on the CPU
  - GPU kernel is executed by many light-weight threads organized in thread blocks launched on a grid
    - Each thread has a unique thread ID within the thread block
    - Each thread block has a unique block ID within the grid
  - Individual GPUs can be controlled by different pthreads or MPI processes with each process controlling one GPU. We use the latter approach.



Fermi Streaming Multiprocessor (SM)

Data layout, Staggered Dslash Operator, and CG Solver



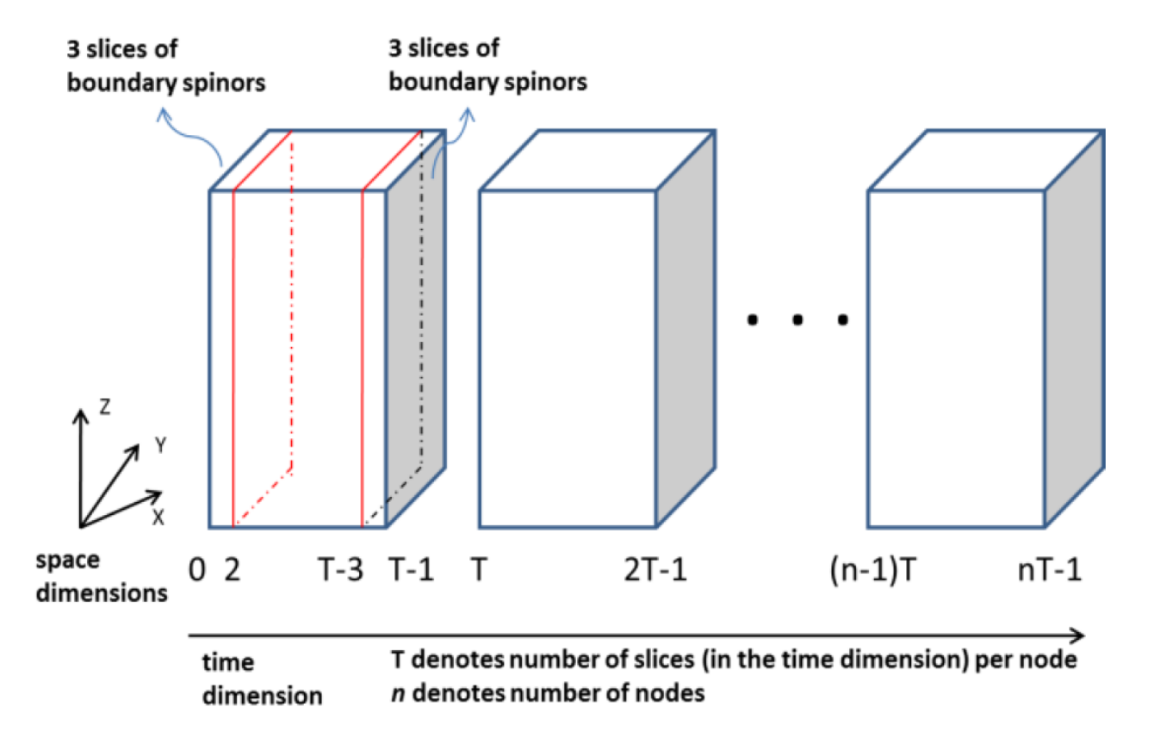
```
#define READ_LONG_MATRIX_12_SINGLE(gauge, dir, idx)                    \
  float4 LONG0 = tex1Dfetch((gauge), idx + ((dir/2)*3+0)*long_ga_stride); \
  float4 LONG1 = tex1Dfetch((gauge), idx + ((dir/2)*3+1)*long_ga_stride); \
  float4 LONG2 = tex1Dfetch((gauge), idx + ((dir/2)*3+2)*long_ga_stride); \
  float4 LONG3 = make_float4(0,0,0,0);                                 \
  float4 LONG4 = make_float4(0,0,0,0);
```

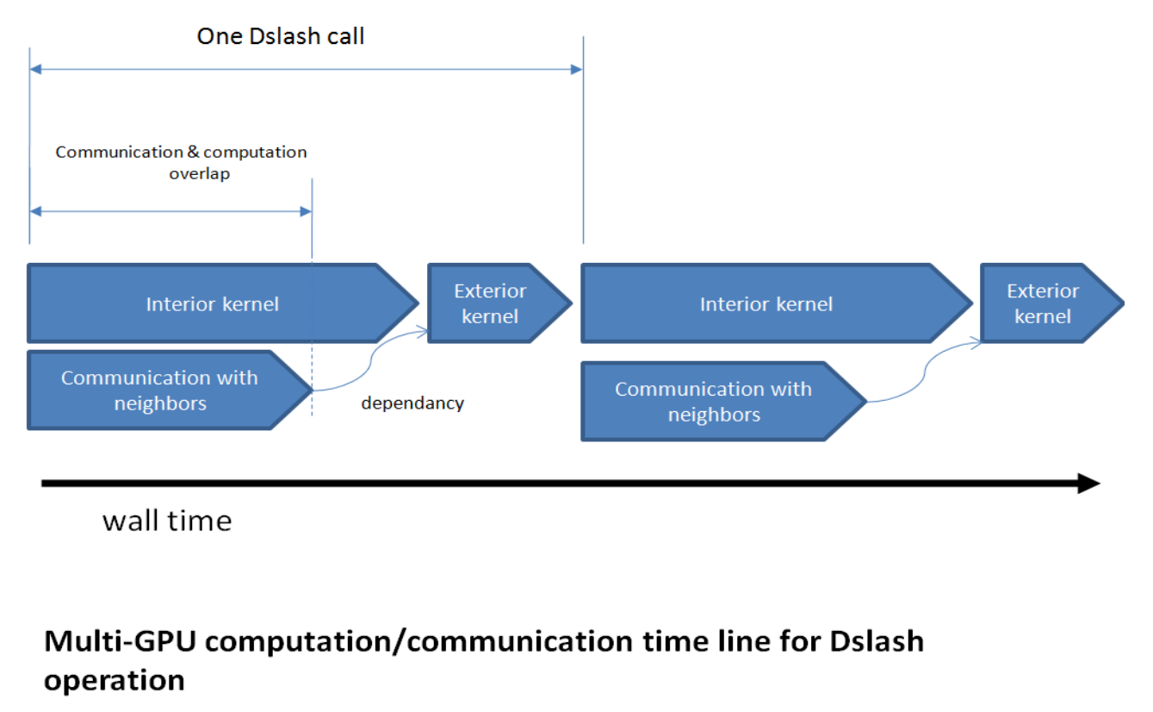
- The most expensive operation in the CG solver is the dslash operation
- Data for spinors and gauge links is rearranged on the host before it is copied to the device memory to enable coalesced memory access
- We have implemented the following flavors of the CG solver [4][5]
  - Single Precision/Double precision/Half precision/Mixed precision
  - 8/12/18 reconstruction method for the long gauge link
  - Multi Shift (multi mass) solver

```
Input:
 D: dslash operator
 A_h: D^+D + 4ma^2 in high precision
 A_l: D^+D + 4ma^2 in low precision
 b: source vector
 x: guess solution vector
Output:
 y: solution vector

y ← 0
i ← 0
r ← b - A_h x                                 D+K7
d ← r
δnew ← r^T r                                  K1
δ_0 ← b^T b                                   K1
while i < i_max and δ > δ_0 ε^2 do
  q ← A_l d                                   D
  α ← δnew / (d^T q)                          K2
  δ_old ← δ_new
  r ← r - α q                                 K3
  δ_new ← r^T r
  β ← δ_new / δ_old
  if (high-precision update is needed)
    x ← x + α d                               K4
    y ← x + y                                 K5
    r ← b - A_h y                             D+K7
    δ_new ← r^T r                             K1
    x ← 0
    β ← δ_new / δ_old
    d ← r + β d                               K8
  else
    x ← x + α d                               K6
    d ← r + β d
  i ← i + 1
```

Multi-GPU CG Implementation

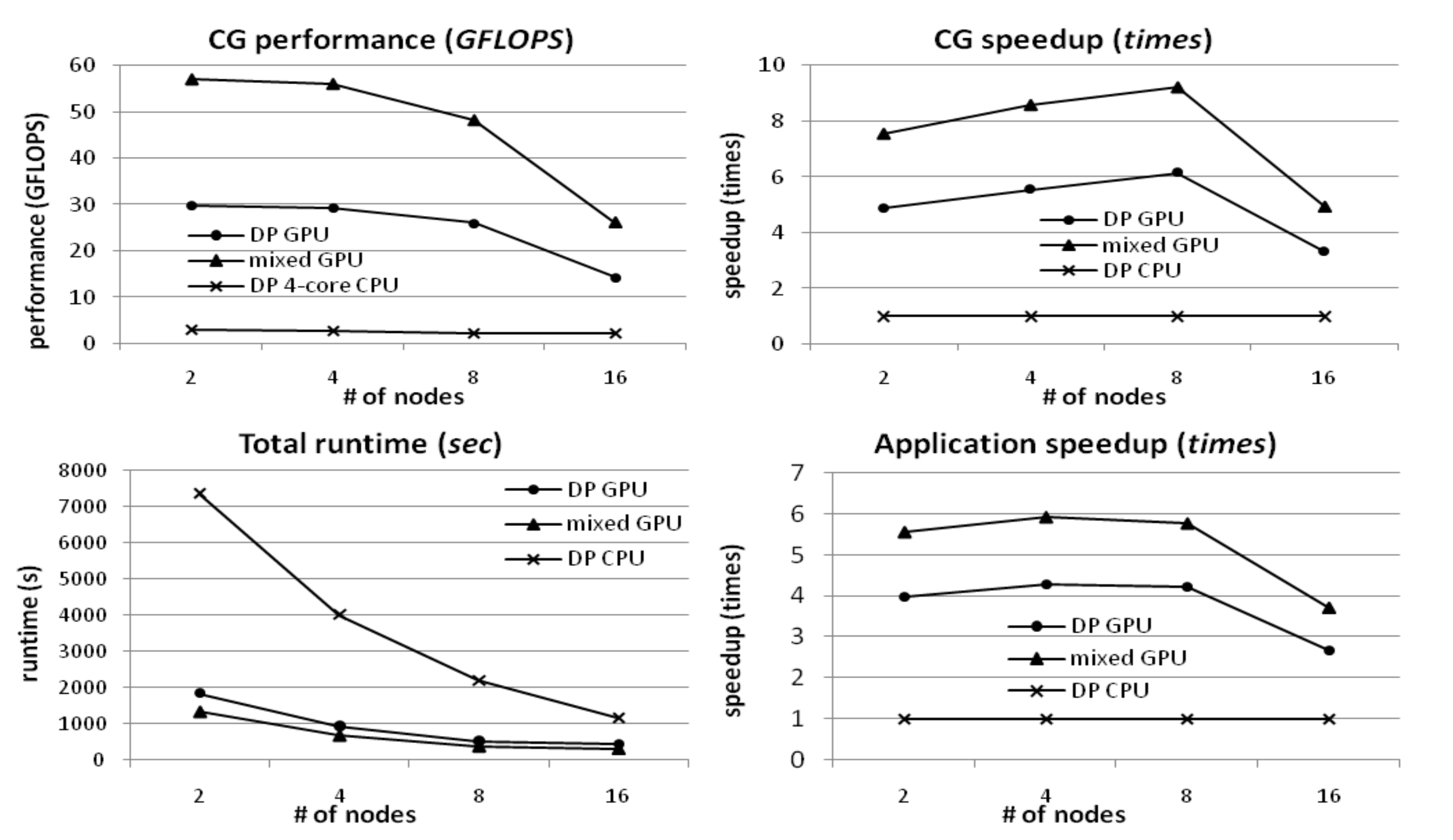

3 slices of boundary spinors
3 slices of boundary spinors
z
y
x
t
space dimensions
0 2
T-3 T-1 T
2T-1
(n-1)T
nT-1
time dimension
T denotes number of slices (in the time dimension) per node
n denotes number of nodes



Multi-GPU computation/communication time line for Dslash operation

- The 4D lattice is partitioned in the time dimension, each node computes T slices with total n*T slices
- Three slices in both forward and backward directions are needed by the neighbors in order to compute new spinors
- The dslash kernel is split into the interior kernel, which computes the internal slices (2<t<T-3) of sub-lattice and the space contribution of the boundary sub-lattices, and the exterior kernel, which computes the time dimension contribution for the boundary sub-lattice. The exterior kernel depends on the data from the neighbors.
- The interior kernel and the communication of boundary data can be overlapped using CUDA streams
- Best performance is achieved when the interior kernel runs longer than the communication, i.e., the sub-lattice per GPU is large enough

Multi-GPU CG Performance



- Quantum Electrodynamics (QED) application.
  - CG is the dominating part. Requires DP accuracy, which can be achieved using mixed-precision in GPU implementation.
- Calculations are done on the DIRAC GPU cluster at NERSC
  - Cluster node consists of 8 Intel Nehalem 2.4 Ghz CPU cores and one Nvidia C2050 GPU
- Mixed precision GPU performance is up to 57 GFLOPS
- Compared with an 8-core CPU node, the GPU speedup is up to 9x
- Multi-node speedup is close to 6x, compared with the same number of nodes without GPUs, using GPU and only one out of 8 CPU cores/node.

Fat Link/Gauge Force/Fermion Force GPU Performance

- Fat link, gauge force and fermion force are also ported to single GPU. Some preliminary results are shown in the table

	Performance (Gflops)
Fat link	178
Gauge force	208
Fermion force	111

GPU performance in single precision in GTX280

- Multi-GPU version of fat link computing is also done. The achieved speedup is 9x compared to a quad-core CPU.

# of nodes	2	4	8	16
DP CPU (Gflops/CPU)	2.2	2.2	2.2	2.1
DP GPU (Gflops/GPU)	19.9	18.8	17	11.4
Speedup	9.0	8.5	7.7	5.4

Fatlink computation in multi-GPU and CPU in double precision

Future work

- Partitioning over the space dimension to enable computation using lattices with a spatial dimension greater than 48
- Multi-GPU version of fermion force/gauge force computing
- Porting HISQ action force terms to single and multiple GPUs

References

[1] M. A. Clark, R. Babich, K. Barros, R. Brower, and C. Rebbi, "Solving Lattice QCD systems of equations using mixed precision solvers on GPUs" (2009), arXiv:0911.3191 [hep-lat]
[2] R.Babich, M.A.Clark, B.Joó, "Parallelizing the QUDA Library for Multi-GPU Calculations in Lattice Quantum Chromodynamics", Proceedings of the 22nd Annual International Conference for High Performance Computing, Networking, Storage, and Analysis 2010 (SC10)
[3] http://lattice.bu.edu/quda/
[4] G. Shi, S. Gottlieb, A. Totok, V. Kindratenko, "Accelerating Quantum Chromodynamics Calculations with GPUs", 2010 Symposium on Application Accelerators in High-Performance Computing (SAAHPC10)
[5] S.Gottlieb, G.Shi, A.Torok, V.Kindratenko, "Quda programming for staggered quarks", In Proc. XXVIII International Symposium on Lattice Field Theory (Lattice 2010), Villasimius, Sardinia, June 2010

Acknowledgements

This work is sponsored by the Institute for Advanced Computing Applications and Technologies (IACAT) and utilized the AC cluster at the National Center for Supercomputing Applications at the University of Illinois. This work is also supported by grants PHY-0555234 (NSF) DE-FC02-06ER41443 & FG02-91ER 40661 (DOE). This work in part is based on the MILC collaboration's public lattice gauge theory code. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231