QUDA Programming For Staggered Quarks

Steven Gottlieb, Guochun Shi, Aaron Torok, Volodymyr Kindratenko

National Center for Supercomputing Applications
&
Indiana University

Outline

- Background
- New staggered code
- Benchmarks
- Production experience
- Where to get the code
- Future

Background

- NCSA's Innovative Systems Laboratory
  - Worked on a Cell B.E. port of MILC, arXiv: 0910.0262
- Boston University: QUDA for Wilson type quarks, arXiv:0810.5365, arXiv:0911.3191
- 8/2009: sabbatical at NCSA
- We are extending QUDA to staggered quarks

New Staggered Code

- First effort was staggered Dslash for single GPU
- Extended to CG and then multimass CG
- Fat link computation
- Gauge Force
- Fermion Force
- Wrappers allow call of GPU code from MILC
- Next step was to merge into BU QUDA code which had evolved from initial release

MultiGPU code

- MultiGPU inverter is now working
    - can overlap communication with computation by using interior and exterior kernels or use a single kernel that is launched after message from neighbor is transferred to GPU
    - however, this code is not in current (0.3) release

- Differences between Wilson and staggered code:
  - for improved staggered quarks need both fat links and long links (more memory)
  - fat links are not unitary, so reconstruction not used (more memory, more time)
    - Fat+Naik: (18+[18|12|8])*4 link operands/site
    - Wilson: 12*4 or 8*4 link operands/site
  - for multiGPU need to fetch from three planes of neighboring node

Benchmarks

- several systems available for benchmarking or production:
  - NCSA: GTX 280, Tesla S1070, Fermi GTX 480
  - Jlab: GTX285, Tesla C1060, S1070, Fermi GTX480
  - FNAL: Tesla S1070
  - NERSC: Tesla C1060, Fermi C2050

Hardware Comparison

Type	Cores	BW (GB/s)	SP (GF/s)	DP (GF/S)	RAM (GB)
GTX280	240	142	933	78	1.0
GTX285	240	159	1062	88	1-2
Tesla C1060	240	102	933	78	4.0
Tesla S1070	four copies of above				
Fermi GTX480	480	177	1345	168	1.5
Fermi C2050	448	148	1030	515	3.0

- Fermi C2050 supports ECC

	reconstruct	CG(GF/s)	multimass CG (GF/s)
DP	12	31	31
	8	15	16
	18	33	34
SP	12	98	92
	8	108	96
	18	83	80
HP	12	123	106
	8	128	113
	18	108	98

$24^3 \times 32$ on GTX280. Four masses for last column.

	STANDALONE	GOAL:ASSUMING 100GB/S	WITH PCIE OVERHEAD
CG	98	100	71
MM-CG	92	100	71
Fat link	178	168	62
Gauge Force	208	349	112
Fermion Force	111	128	94

All values in single precision GF/s. CG speeds measured for 500 iterations. 24^3×32 lattice. Use 12-reconstruct when possible. (GTX280)

Fermi Results

	RECONSTRUCTION	GTX 280	GTX 480	C2050 ECC	C2050 NO ECC
DP	12	29	31	20	24
	8	15	16	11	13
	18	32	50	30	41
SP	12	92	116	66	96
	8	99	126	72	100
	18	79	104	57	86
HP	12	77	154	97	122
	8	74	157	101	123
	18	76	131	84	104

CG speed: $24^3 \times 32$

MULTIGPU BENCHMARKS

- Two kernels used for multiGPU Dslash:
  - internal kernel includes contributions for all directions for t=3,4 ...T-3 and spatial contributions for other t values.
  - boundary kernel adds in the terms in the time direction that need on off-node spinors
- For 24^3 spatial size with single precision on GTX280:
  - pack (D2H, 0.29ms, 3.3GB/s)
  - MPI (0.16ms, 6.14GB/s)
  - unpack(H2D, 0.20ms, 4.8 GB/s)

CG Performance (gflops) per GPU in AC's compute nodes (GPUs: S1070)

	reconst ruct	1 GPU	2 GPUs	4 GPUs	8 GPUs	12 GPUs	16 GPUs	20 GPUs
DP	12	22	22	22	21	18	17	16
	8	13	13	13	12	11	11	10
	18	23	23	23	21	18	18	17
SP	12	58	56	43	40	32	31	31
	8	65	56	40	39	32	33	32
	18	50	50	43	41	35	34	31
HP	12	61	60	40	40	33	33	31
	8	60	59	41	39	36	31	31
	18	61	59	40	40	36	29	32

- Weak scaling, lattice size per GPU 24^3x32
- Run 3 times and get the best number

❑ Time breakdown when running with 1 GPU (SP, recon=8)
- Dlsash time : 3.47(ms)
- Exchange_walltime: 1.79(ms)
- internal_kernel 3.15(ms)
- boundary_kernel: 0.30(ms)
- Computation dominant, communicatin time hidden

❑ Time breakdown when running with 4 GPUs (SP, recon=8)
- Dlsash time : 4.34(ms)
- exchange_walltime: 3.94 (ms)
- internal_kernel: 3.13(ms)
- boundary_kernel: 0.32(ms)
- Communication dominant, worsen when we go offnode, probably due to slow CPU, slow Infiniband, PCIe sharing within 2 gpus

CG Performance (gflops) per GPU @ NERSC (GPUs: C2050, no ECC, one GPU/node)

	reconstruct	1 GPU	2 GPUs	4 GPUs	8 GPUs	12 GPUs	16 GPUs	20 GPUs
DP	12	24	23	23				
	8	13	12	12				
	18	41	41	41				
SP	12	96	94	93				
	8	100	100	100				
	18	86	83	83				
HP	12	122	120	116				
	8	123	120	119				
	18	104	101	101				

- Weak scaling, lattice size per GPU 24^3x32
- Run 3 times and get the best number

- When runnng with 4 GPUs (SP, recon=8)
  - Total dsash time = 2.10 (ms)
  - exchange_walltime = 1.19 (ms)
  - internal_kernel = 1.92 (ms)
  - boundary_kernel = 0.17 (ms)
  - Total dslash time is almost the as the sum of two kernels. Communication time is completely hidden

Production Experience

- Have been using GPUs for electromagnetic effects, i.e., SU(3) × U(1)
- So far only using ensembles that fit in single GPU
- Have analyzed about 4,000 configurations from 20^3×64 to 28^3×96. (Talk by A. Torok)
    - AC (NCSA), FNAL, Dirac (NERSC), JLab
- CPU: 6.04 node-hr=48.2 core-hr
- GPU: 1.49 node-hr=11.9 core-hr (only 1 core used)

Where to get the Code

- With release of QUDA 0.3, staggered code will be integrated with Wilson code (RSN)
- The code can be found at
  - **http://lattice.bu.edu/quda**
- Requires CUDA 3.0.14
- MultiGPU code to be released later
- Repository under construction

Future

- Study of heavy-light mesons might use GPUs. Need to combine both Clover and staggered inverters.
- Although we have asqtad code modules for gauge configuration generation, now generating HISQ configurations, so new code must be added.
- Investigate strong scaling as supercomputers now reaching for petaflop/s performance.
- Essential to decide what other parts of production running can be profitably shifted to GPUs