Lattice QCD code Bridge++ on multi-thread and many core accelerators

Satoru UEDA, S. Aoki, T. Aoyama, K. Kanaya,H. Matsufuru, S. Motoki,
Y. Namekawa, H. Nemura, Y. Taniguchi and N. Ukita
High Energy Accelerator Research Organization (KEK)

Bridge++

- URL: http://bridge.kek.jp/Lattice-code/
- Programming language: C++
- Object oriented/Design patterns
- Covers wide range of architectures
  MPI, OpenMP/pthread, OpenCL for arithmetic accelerators
- General purpose: wide range of actions, algorithms, measurements.
- Rich documents and Test modules

Aiming at the followings features

- Readability: easy to read and use
- Portability: form laptop PC to HPC
- Extensibility: easy to test new ideas
- High-performance: enough for productive runs

Trends of resent supercomputers

- Massively parallel: → Hybrid parallel programing with multi-node and multi-thread.
- Many core arithmetic accelerators: → CUDA, OpenCL, etc.

We need elaborated programing technique to use these system.

Multi-thread programing

- OpenMP: Directive-based program for multi-thread.
  Popular and easy for beginner.
- pthread: API-based thread library.
  Can control threads directly.

Many core device

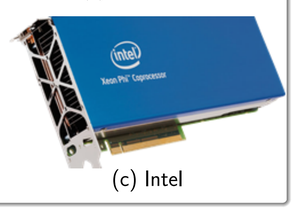
- NVIDIA/AMD GPGPUs
- Intel Xeon Phi
- FPGA

(c) AMD

(c) NVIDIA

Program language for accelerators

- CUDA: API-based library for NVIDIA GPGPUs
- OpenCL: API-based library for general devices
- OpenACC: Directive-based program for general devices



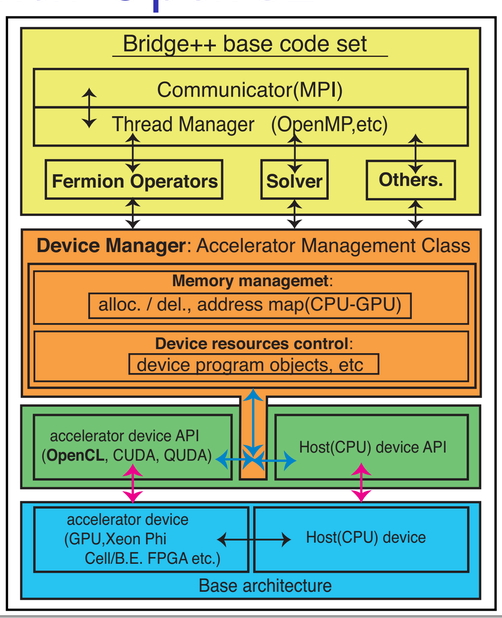
(c) Intel

Implementation Strategy with OpenCL

Seek for a framework independent of devices and other library.

Device manager

- Mediate between device APIs and Host code.
- Abstract and integrate device APIs, exchange devices easily.
- Main code controls devices through this manager.



Multi-threading with OpenMP

Where should start a parallel region?

- Small scope: Frequent thread creation may cause large overheads
- Large scope: Creation shared objects is difficult.

Policy for Implementation with OpenMP

- All shared objects are created at the beginning.
- Thread safe member functions are used in a parallel region.
- Divide a for-loop by thread ID.
- Control threading through a thread manager class.

Present Performance

Hitachi SR16000 1 node @ KEK (980.48GFlops/node)

Domain wall op. @ $8^3 \times 16 \times 8$ lattice

	Topology	GFlops	%
Propagator	32MPI x 2threads*	53	5.4
Propagator	16MPI x 4threads*	42.72	4.4
Propagator	1MPI x 32threads*	14.00	1.5

IBM BG/Q 32nodes @ KEK (204.8GFlops/node)

Domain wall op. @ $16^3 \times 32 \times 8$ lattice

	Libraries	Topology	GFlops/nodes	%
Propagator	MPI + OpenMP	4MPI x 16threads*	5.70	2.8
Propagator	BGNET+OpenMP	4MPI x 16threads†	7.51	3.7
Propagator	BGNET+OpenMP	16MPI x 4threads†	7.63	3.7
(Mult	BG Wilson library(IBM)	16MPI x 4threads†	18.0	8.8)
(Solver	BG Wilson library(IBM)	16MPI x 4threads†	13.9	6.8)

* MPI_THREAD_FUNNELED
† MPI_THREAD_MULTIPLE

Present Performance

- Same code works on 3 kinds of devices.
- Wilson mult @ $16^3 \times 32$ lattice

Accelerator Specifications:

Device name	Radeon HD 7970	GeForce GTX Titan	Xeon Phi 5110P
Vendor	AMD	NVIDIA	Intel
Architecture	Southern Islands	Kepler	MIC
Chip:			
Core clock [MHz]	925	876	1053
Peak DP [GFlops]	947	1570	1011
Peak SP [GFlops]	3789	4709	2022
Global memory:			
Size [Gbytes	3	6	8
B/W [Gbyte/s]	254	258	320
Results:			
GFlops	22	16	25
Performance (%)	0.5	0.3	1

Performance to be improved!!

- Change data layout.
- Reduce data transfers.
- Optimize block/thread parameters.
- Use of libraries, clBLAS, cuBLAS, QUDA etc.

Other Update

- General Nc fermions (in fundamental repr.)
- New measurements, topological change etc.
- To be released soon after this conference.

Reference

- S. Ueda et al, 2014 J. Phys.: Conf. Ser. 523 012046
- S. Ueda et al, PoS(LATTICE 2013)412
- S. Motoki et al, Procedia Computer Science(2014) 1701