Bridge++
- URL: http://bridge.kek.jp/Lattice-code/
- Programming language: C++
- Object oriented/Design patterns
- Covers wide range of architectures
- MPI, OpenMP/ptthread, 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

Multi-thread programing
  Popular and easy for beginner.
- pthread: API-based thread library.
  Can control threads directly.

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-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

<table>
<thead>
<tr>
<th>Topology</th>
<th>GFlops</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagator</td>
<td>32MPI x 2threads</td>
<td>53 5.4</td>
</tr>
<tr>
<td>Propagator</td>
<td>16MPI x 4threads</td>
<td>42.72 4.4</td>
</tr>
<tr>
<td>Propagator</td>
<td>1MPI x 32threads</td>
<td>14.00 1.5</td>
</tr>
</tbody>
</table>

IBM BG/Q 32nodes @ KEK (204.8GFlops/node)
Domain wall op. @ $16^3 \times 32 \times 8$ lattice

<table>
<thead>
<tr>
<th>Libraries</th>
<th>Topology</th>
<th>GFlops/nodes %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagator</td>
<td>MPI + OpenMP</td>
<td>5.70 2.8</td>
</tr>
<tr>
<td>Propagator</td>
<td>BGNET + OpenMP</td>
<td>7.51 3.7</td>
</tr>
<tr>
<td>Propagator</td>
<td>BGNET + OpenMP</td>
<td>7.63 3.7</td>
</tr>
<tr>
<td>(Mult) Wilson library(IBM)</td>
<td>16MPI x 4threads</td>
<td>18.0 8.8</td>
</tr>
<tr>
<td>(Solver) Wilson library(IBM)</td>
<td>16MPI x 4threads</td>
<td>13.9 6.8</td>
</tr>
</tbody>
</table>

Many core device
- NVIDIA/AMD GPGPUs
- Intel Xeon Phi
- FPGA

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

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.

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

Accelerator Specifications:

<table>
<thead>
<tr>
<th>Device name</th>
<th>Vendor</th>
<th>Architecture</th>
<th>Chip</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeForce GTX Titan</td>
<td>NVIDIA</td>
<td>Kepler</td>
<td>1053</td>
</tr>
<tr>
<td>Xeon Phi</td>
<td>Intel</td>
<td>Mic</td>
<td>1018</td>
</tr>
</tbody>
</table>

| Core clock [MHz] | 925 | 876 |
| Peak LP [GFlops] | 847 | 1570 |
| Peak SP [GFlops] | 13789 | 4709 |
| Global memory:   | 14072 | 2022 |

Size [Gbytes]: 3 8 6 8

B/W [Gbyte/s]: 254 258 320

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

Reference
- S. Ueda et al, PoS(LATTICE 2013)412
- S. Motoki et al, Procedia Computer Science(2014) 1701