Bridge++ for Fugaku

Hideo Matsufuru (KEK)
for Bridge++ project/FS2020 Codesign

葛飾北斎 冨嶽三十六景《深川万年橋下》
Hokusai, “Fukagawa Mannen-bashi shita” in Fugaku Thirty-six scenery

Fugaku QCD coding workshop

12-13 December 2019, RIKEN R-CCS, Kobe, Japan

Summary

- Bridge++ is preparing to Fugaku
- Basic strategy
  - Call QWS if available
  - Extend QWS if necessary
- In progress
  - Code with the same data layout and convention as QWS
  - Code for SIMD (Intel AVX-512) is also prepared for comparison
  - Benchmark code sets are ready
  - Optimization on RIKEN simulagtior is started (in addition to QWS)

Bridge++

Lattice QCD code Bridge++

Cf. posters by Y. Namekawa and I. Kanamori

- General purpose code set for simulations of lattice gauge theory
- C++, object-oriented design

- Development policy
  - Readable: for beginners
  - Extendable: for testing new ideas
  - Portable: works on many machines
  - Practically enough high performance
- Histroy
  - Project launched in October 2009, first public release in July 2012
  - Latest version: 1.5.3 (09 Dec 2019)
- Current active members

  Y. Akahoshi (Kyoto), S. Aoki (YITP), T. Aoyama (KEK),
  I. Kanamori (Riken R-CCS), K. Kanaya (Tsukuba), H. Matsufuru (KEK),
  Y. Namekawa (KEK), H. Nemura (RCNP), Y. Taniguchi (Tsukuba)

Bridge++

- Implementation
  - Parallelized by MPI (possible to replace with a low-level library)
  - Multi-threaded by OpenMP
  - Algorithms are generally implemented making use of polymorphism
  - Guided by Design Patterns
- Examples
  - Hybrid Monte Carlo with arbitrary nested integrators, RHMC
  - Fermion operators with link smearing (APE, HYP)
  - Many linear equation solvers, Implicitly restarted Lanczos eigensolver
  - Gradient flow
- Restriction
  - Fixed data layout
  - Double precision

  → requirement of optimization for each architecture

Extended Bridge++

Extended Bridge++ framework:

Core library + *extension ("alternative")*

- Bridge++ core library
  - Original Bridge++ code (while still actively developed)
  - Used as a firm framework and general purpose tool set
- Extension ("alternative" code)
  - Arbitrary data type and layout
  - Exploit C++ template keepingthe class structure of the core library
  - Machine-specific implementation is easily incorporated
  - So far not public, but provided on demand
- Class for field object
  - `Field → AField<REALTYPE, IMPL>` (IMPL is dfnied by enum)
    - e.g. `AField<double, SIMD>`

Extended Bridge++

Development of extended Bridge code

- GPUs, accelerators
  - OpenCL and OpenACC
    - S.Motoki et al., Proc. Comp. Sci. 29 (2014) 1701
    - H.Matsufuru et al., Proc. Comp. Sci. 51 (2015) 1313
  - Pezy-SC with OpenCL
    - T. Aoyama et al. Proc. Comp. Sci. 80 (2016) 1418
- SIMD architecture (Intel AVX-512 on Xeon Phi KNL and Xeon)
  - Optimization with AVX-512 intrinsics and manual prefetching
  - “Grid” and simple layouts compared – no large difference in performance
    - I.Kanamori and H.Matsufuru, LNCS 10962 (2018) 456-471
  - HMC is imlemented including gauge part
- Vector architecture (NEC SX-Aurora)
  - In progress

Fugaku

Fugaku will be available in 2021

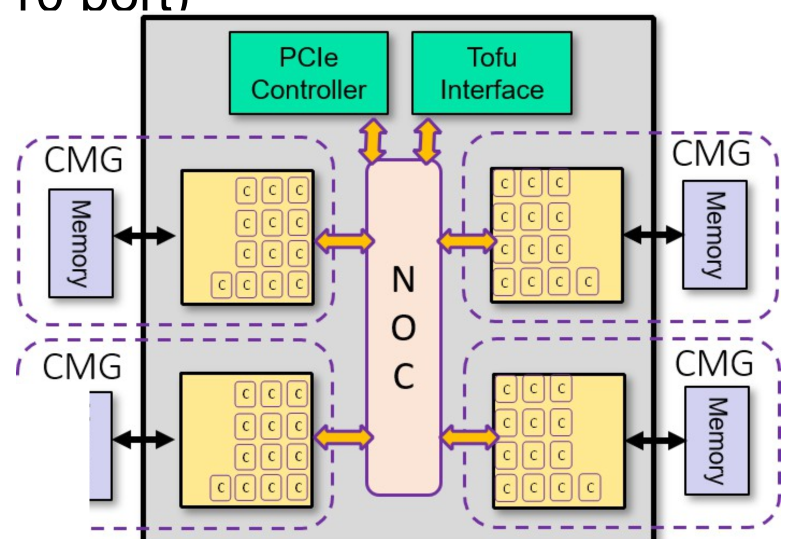
- ARM based processor by Fujitsu with Scalable Vector Extension
  - Architecture: Armv8.2-A SVE 512bit
  - Core: 4 CMGs (NUMA nodes), 48 cores for compute

    DP: 2.7+ TF, SP: 5.4+ TF, HP: 10.8 TF
  - Cache: L1D/core: 64 KiB, L2/CMG: 8 MiB, 16way
  - Memory: HBM2 32 GiB, 1024 GB/s
  - Tofu Interconnect D (28 Gbps x 2 lane x 10 port)

(From Fujitsu site)

- Prototype was aworded top ranking of Green500 in Nov. 2019
- RIKEN Fugaku processor simulator is available for application development

DP 3.072 TF/core
3.379 TF (boost)
(From Kodama-san's talk)



(From RIKEN R-CCS site)

Preparation to Fugaku

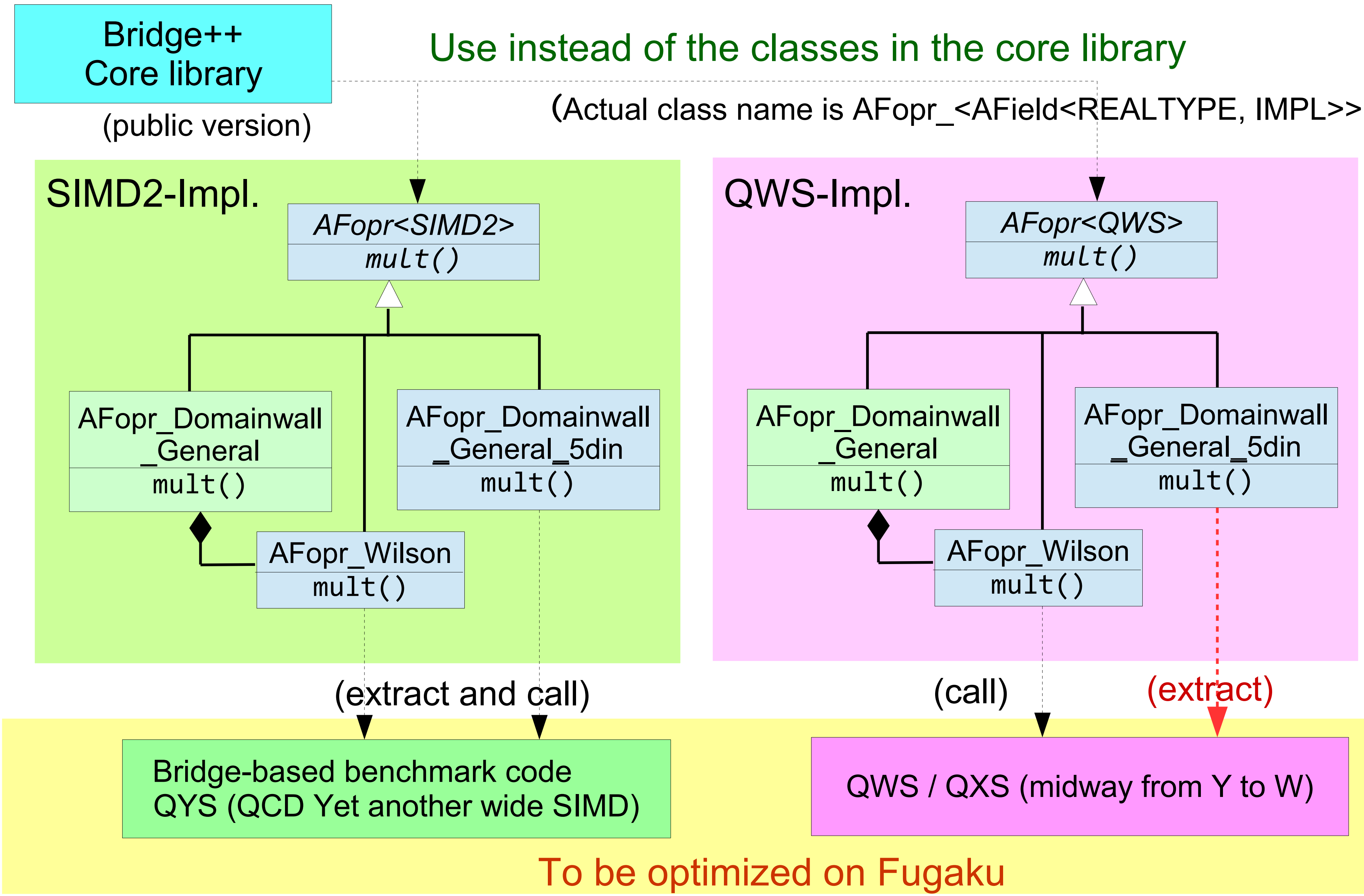
Basic strategy

- *Call QWS if available*
  - "QCD Wide SIMD" library (Cf. Nakamura-san's talk)
  - An optimized library developed mainly by Y. Nakamura in FS2020
  - Available: (domain-decomposed) even-odd clover
- *Extend QWS if necessary*
  - Fermion operators: Domainwall, Staggered, etc.
  - Apply optimization techiniques developed in QWS

Prparation

- "QWS" implementation
  - Code branch with same data layout and convention as QWS
  - Extract kernel codes and adjust them to QWS format
- "SIMD2" implementation
  - Developed for SIMD (for AVX-512) architecture
  - Another implentation that may work efficiently on Fugaku

Code structure



Convention

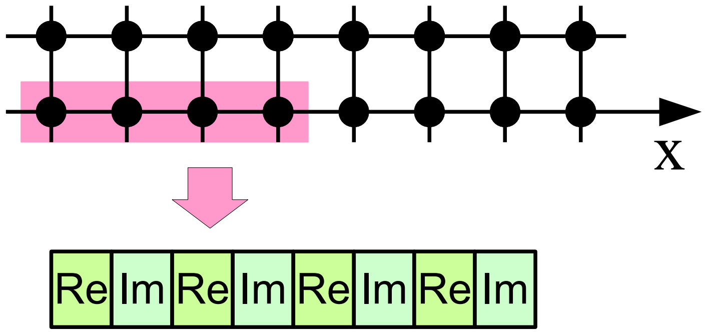
- Conversion is needed before and after callin QWS
  - Implemented in classes of QWS-impl.
  - Multiply γ_4 to spinor
  - Order of color/spinor/complex indices is changed

	QWS	Bridge++
VLEN	site(x)	complex * site(x)
spinor (even-odd)	v[Neo][Nt][Nz][Ny][Nxd] [Nc][Nd][Nri][VLEN]	v[Neo][Nt][Nz][Ny][Nxd] [Nd][Nc][VLEN]
Gauge (even-odd) U_{ab}	g[Neo][Nt][Nz][Ny][Nxd] [Ncb][Nca][Nri][VLEN]	g[Neo][Nt][Nz][Ny][Nxd] [Nca][Ncb][VLEN]
Nxd	Nx/VLEN	Nx/VLEN2 (VLEN2=VLEN/2)
gamma matrix (Dirac)	$\gamma_i^{(\mathrm{QWS})} = -\gamma_i^{(\mathrm{Bridge})}, \quad \gamma_4^{(\mathrm{QWS})} = \gamma_4^{(\mathrm{Bridge})}, \quad \gamma_5^{(\mathrm{QWS})} = \gamma_5^{(\mathrm{Bridge})}$	
Lattice size	Macro (#define)	Given at run-time

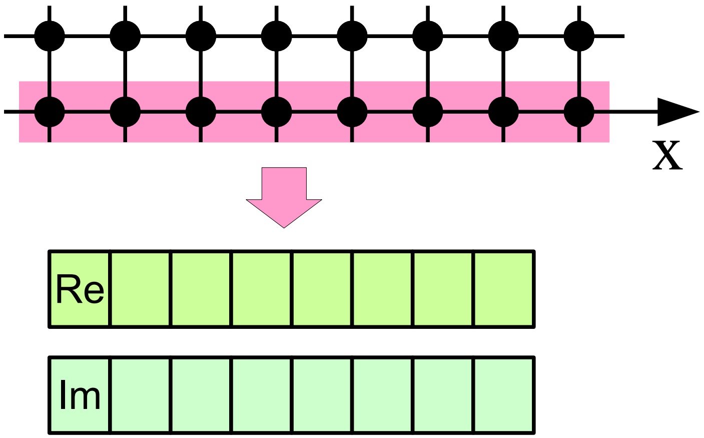
Data layout

- Length of SIMD vector
  - `VLEND=8 (double), VLENS=16 (float)`
  - Restriction of local lattice size in x-direction
- SIMD2 impl.
  - VLEN/2 sites in x-direction are packed in a SIMD vector
- QWS impl.
  - VLEN sites in x-direction are packed in a SIMD vector

SIMD2 (lexical site ordering)



QWS (lexical site ordering)



(double precision)

Domainwall fermions

$$D_{DW} = \begin{pmatrix} D_+^{(1)} & D_-^{(1)}P_- & 0 & \cdots & 0 & -mD_-^{(1)}P_+ \\ D_-^{(2)}P_+ & D_+^{(2)} & D_-^{(2)}P_- & & & 0 \\ 0 & D_-^{(3)}P_+ & D_+^{(3)} & D_-^{(3)}P_- & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & D_-^{(N-1)}P_+ & D_+^{(N-1)} & D_-^{(N-1)}P_- \\ -mD_-^{(N)}P_- & 0 & \cdots & 0 & D_-^{(N)}P_+ & D_+^{(N)} \end{pmatrix}$$

$$D_+^{(i)} = b_i D_W + 1, \qquad D_-^{(i)} = c_i D_W - 1.$$

- Special care is necessary for Domainwall fermions
  - In addition to multiplying γ_4
  - $\gamma_4\gamma_5\gamma_4 = -\gamma_5$ must be used instead of γ_5

Two types of implementation

- General template class: uses Wilson fermion object
  - Change of gamma5 is: specialization of mult_gm5() member function
- 5-th direction is treated as an on-site d.o.f.
  - Implementation specific to each architecture
  - Change of gamma5 is realized by changing inline functions

Benchmark

- SIMD2 implementation

→ QYS: "QCD Yet another wide SIMD" library
  - Extraction of kernel code from SIMD2 impl.
  - Parformance comparison to QWS-type impl.

- QWS implementation

→ QXS (midway from Y to W)
  - Works in the same way as QWS
  - But familiar implementation for us

→ Extension of QWS library
  - Apply the optimization techniques developed in FS2020

Status

- Fermion operators for each architecture
  - "QXS" implies QWS implementaion in Bridge++
  - Link smearing is planned
  - Gauge part for HMC force ? (ready in SIMD2)

	OpenACC	SIMD2	QXS	QWS
Wilson (lex)	○	○	○	
(eo)	○	○	○	◎
Clover (lex)	○	○		
(eo)	○	○		◎
Domainwall(5din) (lex)		○	○	
(eo)		○	○	
Staggered (lex)	○			
(eo)	○			

Summary

- Bridge++ is preparing to Fugaku
- Basic strategy
  - Call QWS if available
  - Extend QWS if necessary
- In progress
  - Code with the same data layout and convention as QWS
  - Code for SIMD (Intel AVX-512) is also prepared for comparison
  - Benchmark code sets are ready
  - Optimization on RIKEN simulagtior is started (in addition to QWS)