Wilson matrix kernel for lattice QCD on A64FX architecture

Issaku Kanamori (RIKEN R-CCS), Keigo Nitadori (RIKEN R-CCS),
Hideo Matsufuru (KEK)

Feb. 27, 2023

International workshop on Arm-based HPC: Practice and Experience (IWAHPCE 2023)
The International Conference on High Performance Computing in the Asia-Pacific Region (HPC Asia) 2023

Outline

1. Introduction: Target application　Lattice QCD
2. Implementation　Stencil computation with Arm C-Language Extension (ACLE)
3. Performance　supercomputer Fugaku
4. Summary and Outlooks

Target Application: Lattice QCD

Problem as a physics

- Quantum ChromoDynamics (QCD): describes interaction of quarks
- the theory (eq. of motion) is known, but we cannot analytically solve in both classical/quantum senses
- can be formulated on lattice (*Lattice QCD*)

Target Application: Lattice QCD

Problem as a physics

- Quantum ChromoDynamics (QCD): describes interaction of quarks
- the theory (eq. of motion) is known, but we cannot analytically solve in both classical/quantum senses
- can be formulated on lattice (*Lattice QCD*)

Problem as an HPC application

- high dim. ($O(10^9)$ or more) integral:
  Monte Carlo method (cf. statistical mechanics)
- a typical example of large scale HPC
- integrand $\propto \det(D^\dagger D) \propto \int d\eta\, d\eta^* \exp(-|D^{-1}\eta|^2)$
  Bottleneck: solving linear equation $D\xi = \eta$ (CG, BiCGStab, etc.)

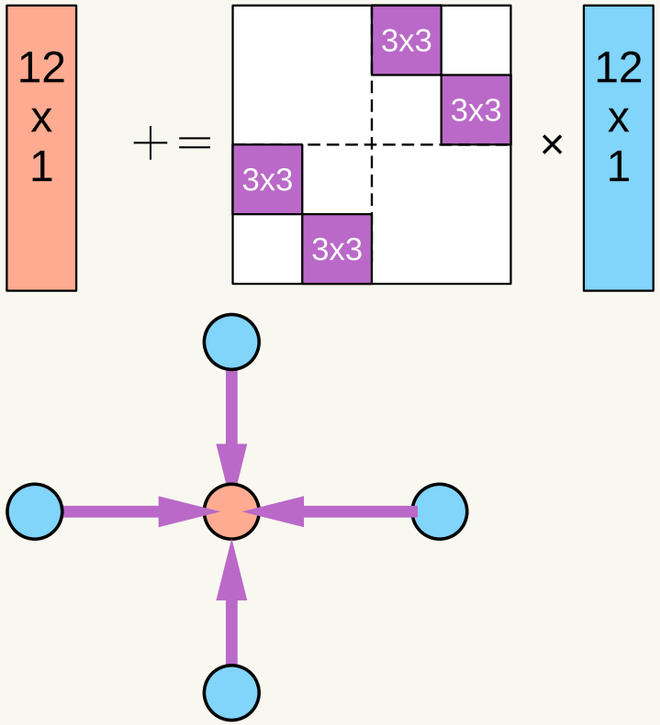
  For each Markov chain update, 10–100 solves are needed
- D: complex sparse matrix stencil computation on 4-dim structured lattice focus of this talk

More Details on the Application

want to solve: $D\psi = \phi$ for ψ (i.e., $\psi = D^{-1}\phi$, ϕ is given)

Several choices of D: symmetry, discretization error, computational cost,...

Wilson type D: 1368 flop/site, 1.12 byte/flop



9-point stencil on 4-dim. lattice $(\mu = x, y, z, t)$

$$D(x,y) = \delta_{x,y} - \kappa \sum_{\mu=1}^{4} \left[(1-\gamma_\mu) U_\mu(x) \delta_{x+\hat{\mu},y} + (1+\gamma_\mu) U_\mu^\dagger(x-\hat{\mu}) \delta_{x-\hat{\mu},y} \right]$$

$\psi_{i\alpha}(x)$: 12 cmpl. components on each site

$i = 1, 2, 3$ (color d.o.f.)

$\alpha = 1, \ldots, 4$ (spin and quark/anti-quark)

$U_\mu(x)_{ij}$: 3×3 cmpl. matrix $\in$ SU(3) [gluons]

$(\gamma_\mu)_{\alpha\beta}$: 4×4 cmpl. permutation matrix

κ: real number [related to the mass of quark]

Site Even-Odd preconditioned

- hopping: btw even-site and odd-site

$$D_{\mathrm{W}} \begin{pmatrix} \xi_{\mathrm{e}} \\ \xi_{\mathrm{o}} \end{pmatrix} = \begin{pmatrix} D_{\mathrm{ee}} & D_{\mathrm{eo}} \\ D_{\mathrm{oe}} & D_{\mathrm{oo}} \end{pmatrix} \begin{pmatrix} \xi_{\mathrm{e}} \\ \xi_{\mathrm{o}} \end{pmatrix} = \begin{pmatrix} 1 & D_{\mathrm{eo}} \\ D_{\mathrm{oe}} & 1 \end{pmatrix} \begin{pmatrix} \xi_{\mathrm{e}} \\ \xi_{\mathrm{o}} \end{pmatrix} = \begin{pmatrix} \eta_{\mathrm{e}} \\ \eta_{\mathrm{o}} \end{pmatrix}$$

- equivalent to 2 equations:

$$\begin{cases} (1 - D_{\mathrm{eo}} D_{\mathrm{oe}})\, \xi_{\mathrm{e}} = (\eta_{\mathrm{e}} - D_{\mathrm{eo}} \eta_{\mathrm{o}}) \\ \xi_{\mathrm{o}} = (\eta_{\mathrm{o}} - D_{\mathrm{oe}} \xi_{\mathrm{e}}) \end{cases}$$

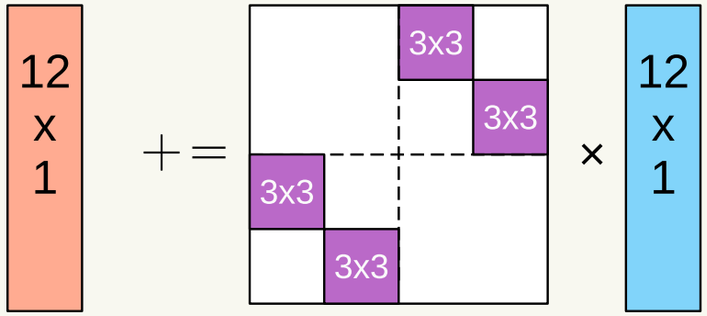
- target kernel: $(1 - D_{\mathrm{eo}} D_{\mathrm{oe}})\, \xi_{\mathrm{e}}$

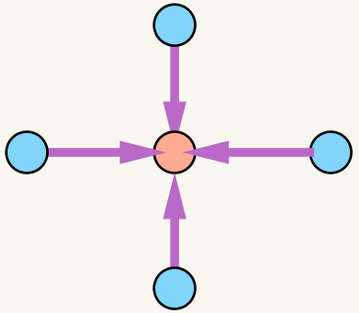
  better condition number, smaller memory footprint in iterative solvers

Implementation

Structure of the kernel

- **EO1**: pack the boundary data to the buffer
  multiplication of 3×3 matrices on the data to the forward directions
- start sending
- **Bulk**: stencil computation in the bulk main focus
- wait for the data arrival
- **EO2**: accumulate the arrived data
  multiplication of 3×3 matrices on the data from the forward directions





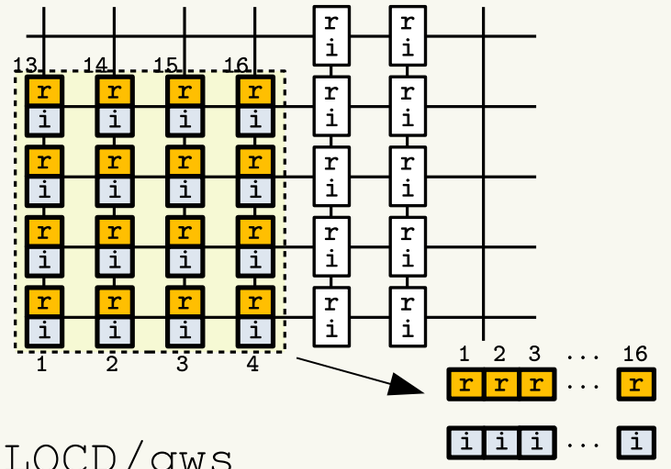
Implementation for A64FX: code set Bridge++

cf. Y .Akahoshi et al. (2022); `https://bridge.kek.jp/Lattice-code/`

- Data structure
  - re/im part of complex numbers to separated SIMD registers
  - 512-bit: 16 single prec. numbers, to the lattice sites in x- and y- directions

    `float spinor` $\underbrace{\texttt{[NT][NZ][NY/VLENY][NX/2/VLENX]}}_{\text{4-dim site/SIMD}}\underbrace{\texttt{[3][4][2]}}_{\text{on site}}\underbrace{\texttt{[16]}}_{\text{SIMD}}\texttt{;}$

    (similar for 3×3 gauge field matrix)
  - `VLENX` $\times$ `VLENY` $= 16$ (`VLENX` ≥ 2)
- Arm C-Language Extension (ACLE)



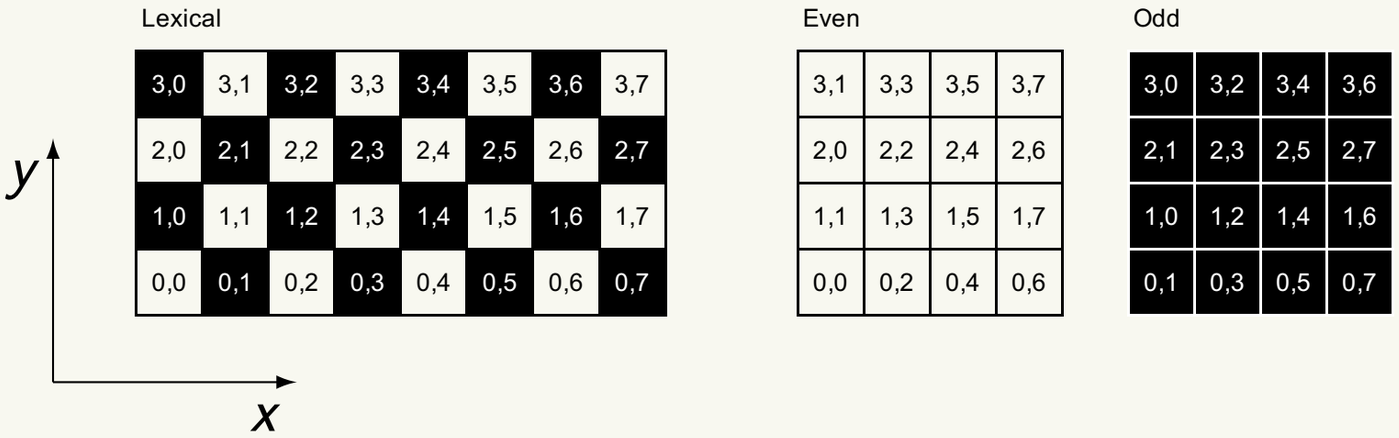
Other implementations for A64FX

- QCD Wide SIMD library (QWS) `https://github.com/RIKEN-LQCD/qws`
  co-design for Fugaku, solver for Clover type D, `VLENX` $\times$ `VLENY` $= 16 \times 1$ only K.-I. Ishikawa et al. (2023)
- GRID `https://github.com/paboyle/Grid`
  `riri...` for complex numbers, sub-grid for SIMD N.Meyer et al., Lattice 2021

Implementation: MPI and thread parallelization (skip)

- asynchronous communication, overlap btw the bulk kernel and communication
- Fujitsu extension for the MPI persistent communication:
  the assistant cores help the overlap btw computation and communications
  `FJMPI_Prequest_start(...)` etc.
- `MPI_THREAD_FUNNELED`: equal OMP division for the site loop

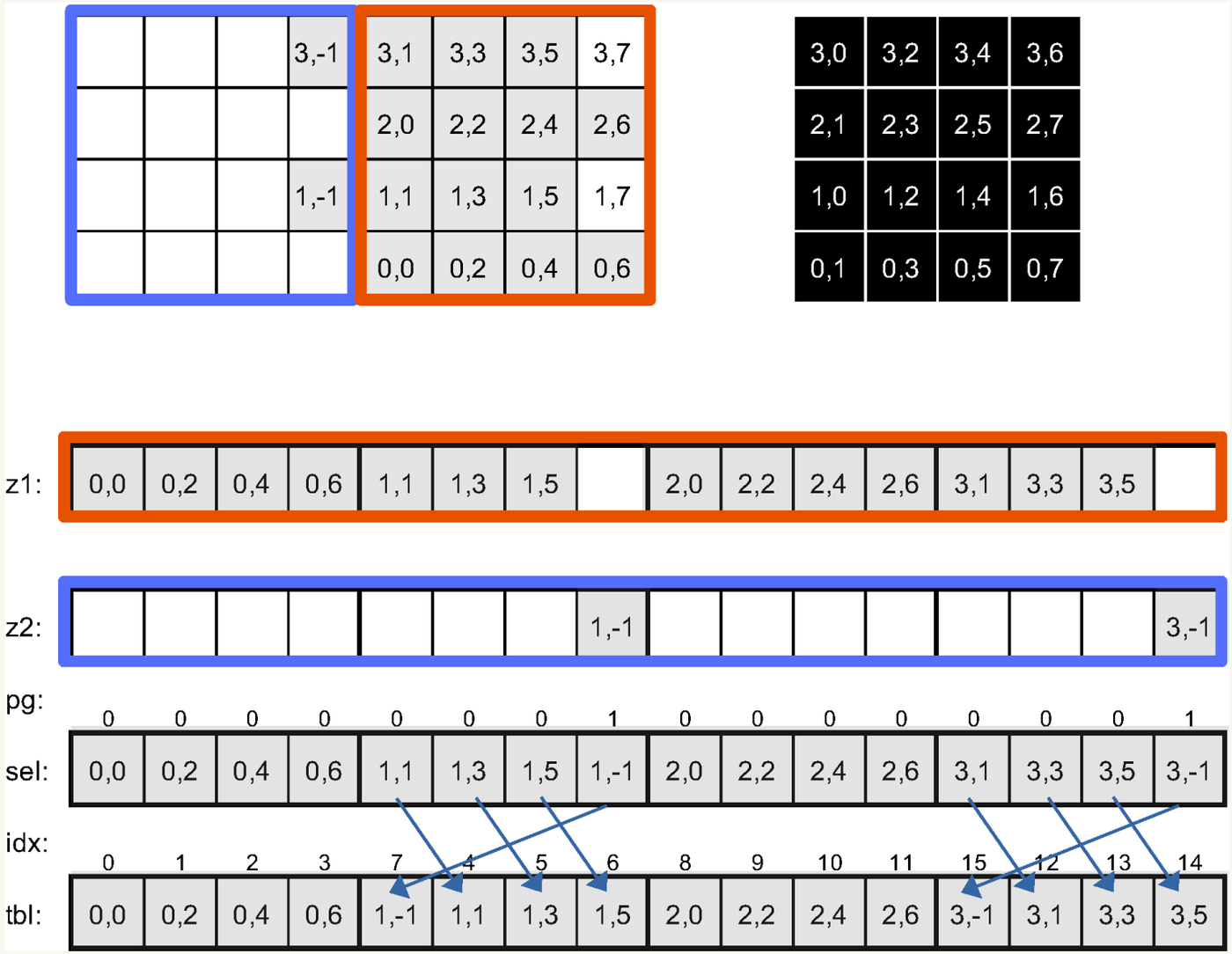
More about data layout: site even-odd decomposition



in (x, y)-plane
the labels are (y, x)

- field (i.e.,vector) is decomposed into “even” field and “odd” field
  4-dim coordinate (x, y, z, t): $x + y + z + t =$ even or odd
- stencil: even to odd (or odd to even) site field
  non-trivial re-arrangement inside SIMD vector is needed

Stencil in x-direction



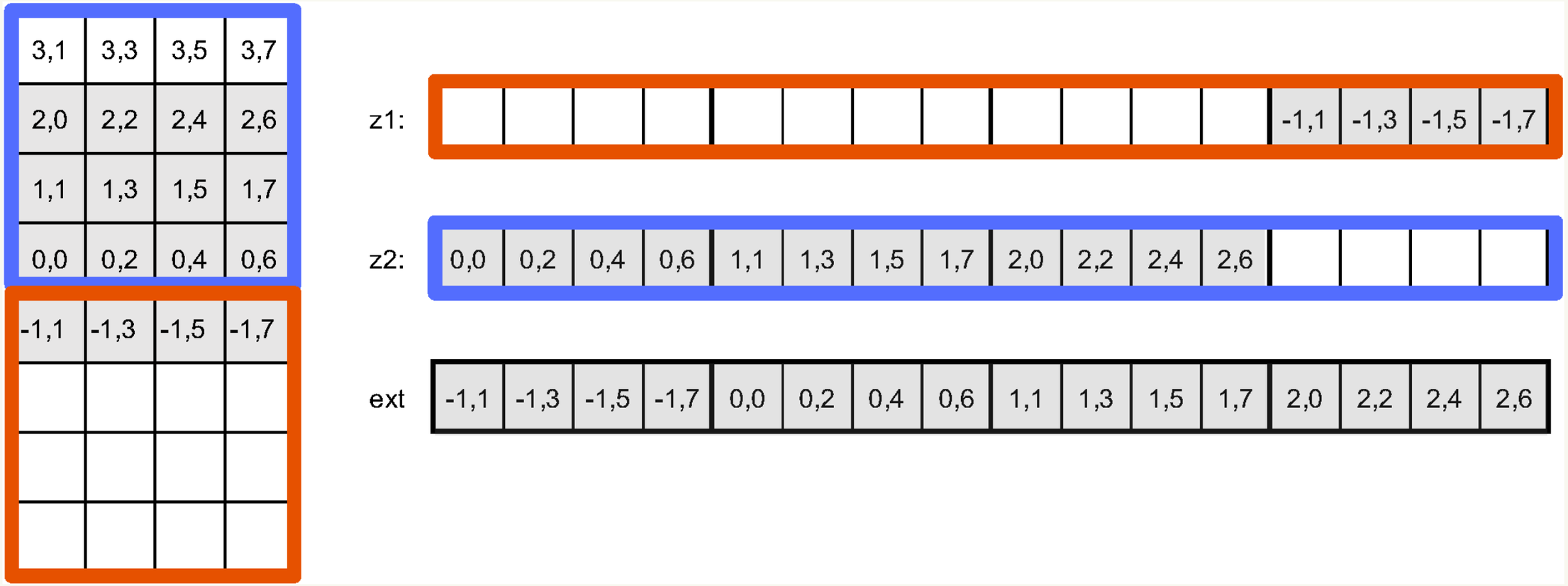
want to merge `z1` and `z2`

`sel` to merge `z1` and `z2`

`tbl` to reorder the elements

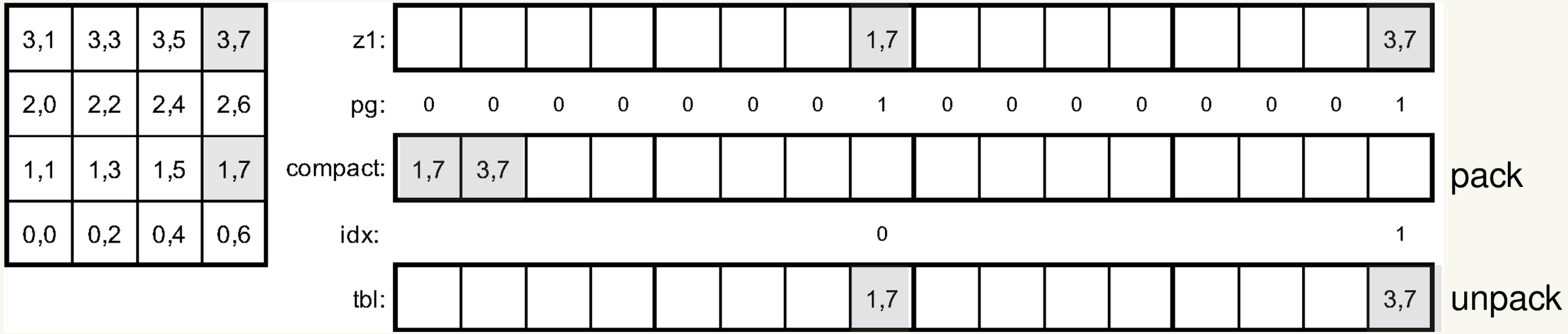
Stencil in y-direction

want to merge `z1` and `z2`



`ext` merges `z1` and `z2`

communication buffer: packing and unpacking (x-direction)

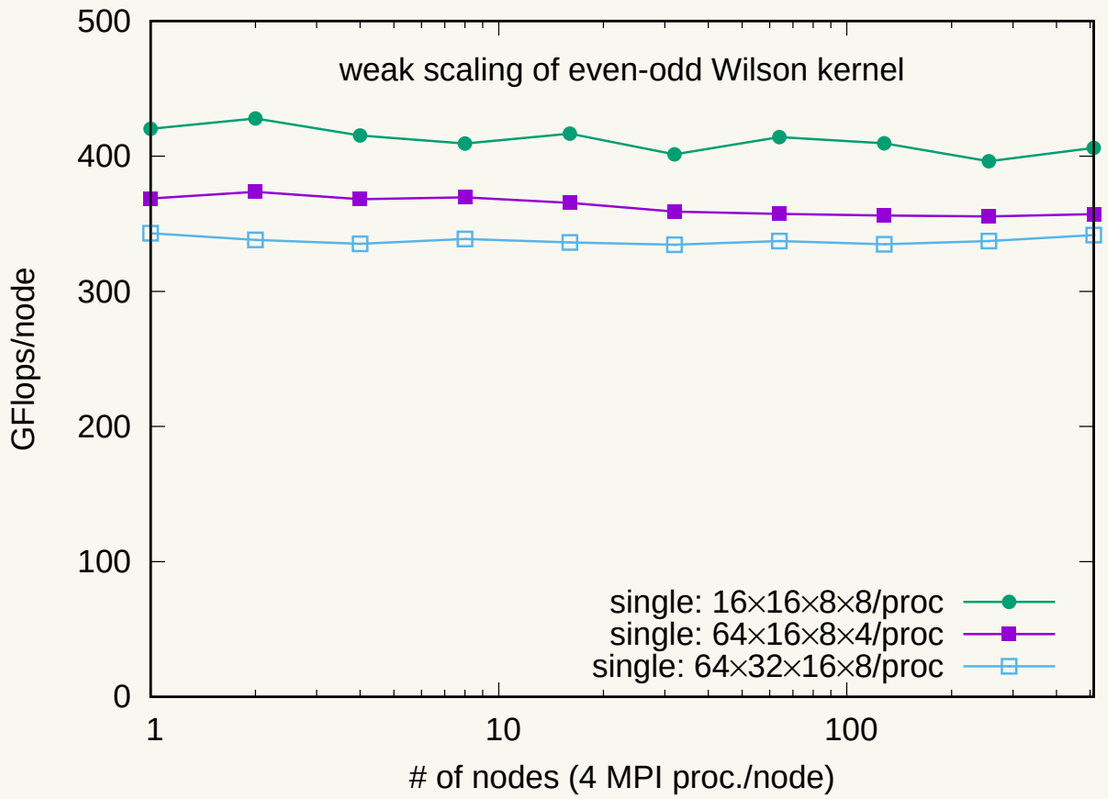


- `compact` to pack two elements to the buffer
- `tbl` to unpack to the original layout

Performance

- supercomputer Fugaku, RIKEN R-CCS
- environment: Fujitsu C/C++ compiler in its “clang mode” (version 4.8.1 tcsds-1.2.36)
- option: `-Kfast -Rpass-missed=inline -mllvm -inline-threshold=1000`
- `FLIB_BARRIER =HARD`

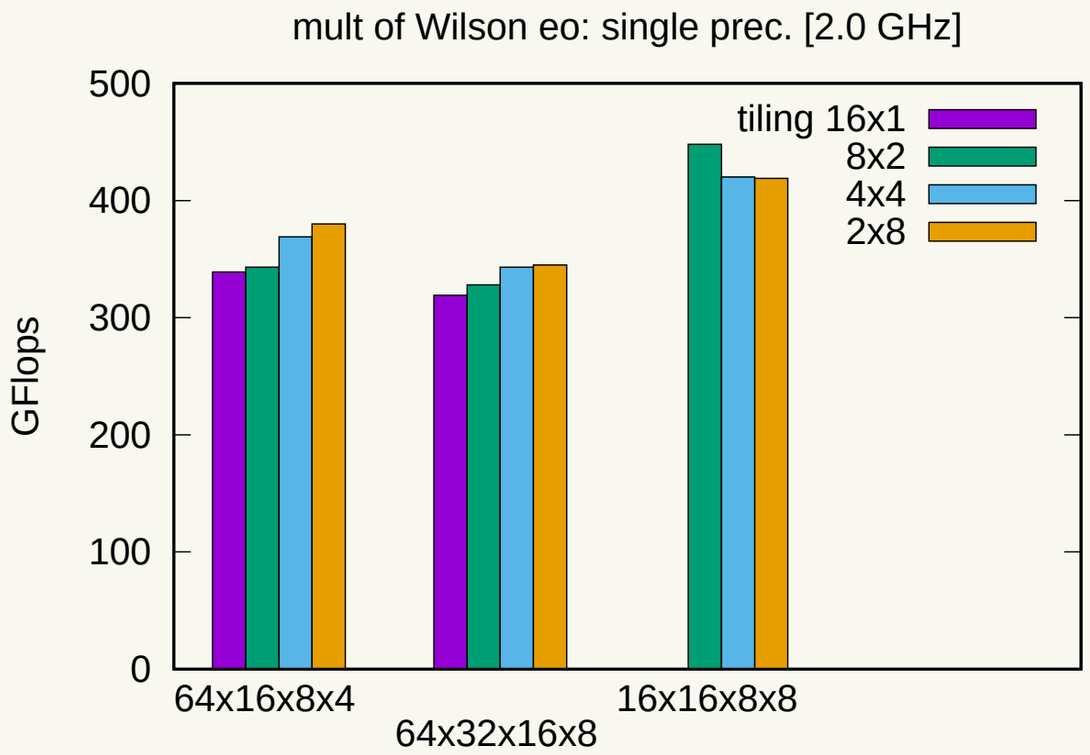
Weak scaling: very good



$$\mathtt{VLENX} \times \mathtt{VLENY} = 4 \times 4$$

- 1000 times of operations
- based on theoretical counting
  1368 FLOP/site
- 6–7 % of the theoretical peak
  A64FX 2.0 GHz: 6144 GFlops/node for single prec.
- always enforced MPI communication, even the number of process is 1 in that direction
- proper rankmaps: neighboring processes are always on the same node or physical neighbor in the 6-dim mesh-torus network
- (strong scaling is also good: backup slides)

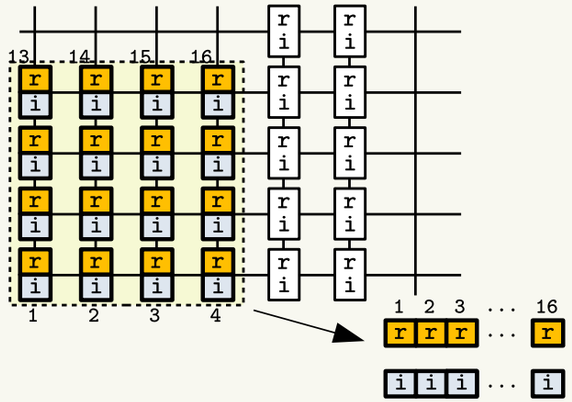
Dependence of 2-dim SIMD tiling: mild



- `VLENX` $\times$ `VLENY`:
  16×1, 8×2, 4×4 and 2×8
  16×1 does not fit to $16 \times 16 \times 8 \times 8$ latt.

- only a mild a dependence, no clear tendency
  $\Rightarrow$allows flexible lattice volume
  (cf. QWS is 16×1)

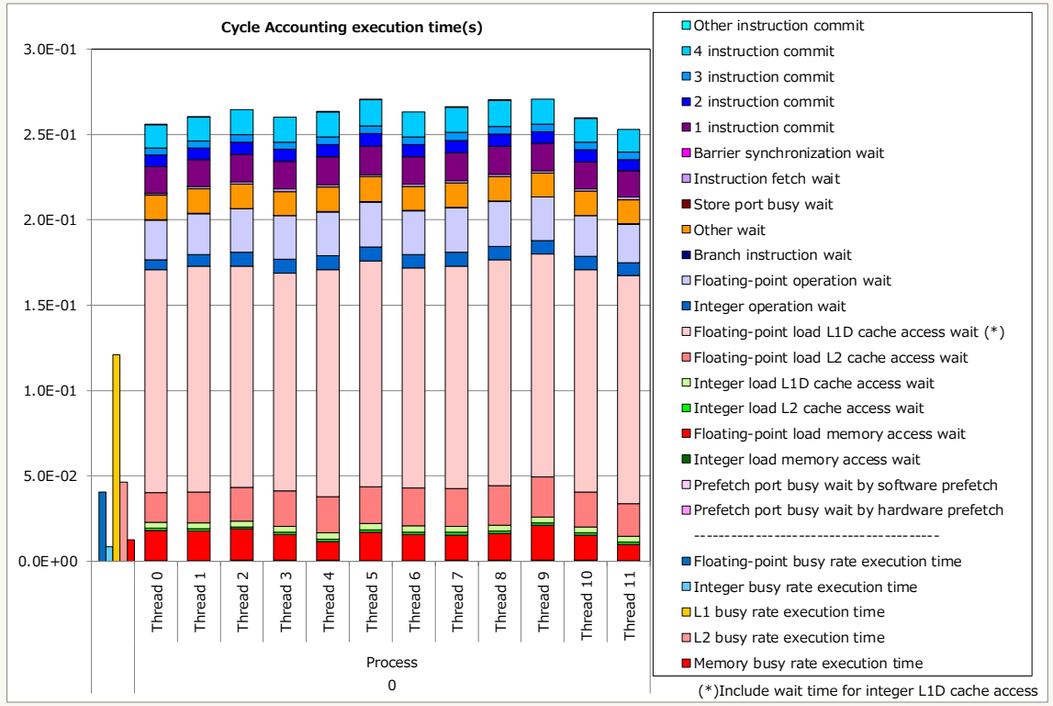
4×4:



1 node with $1 \times 1 \times 2 \times 2$ MPI proc.
force boundary communication in every directions
(emulating weak scaling)

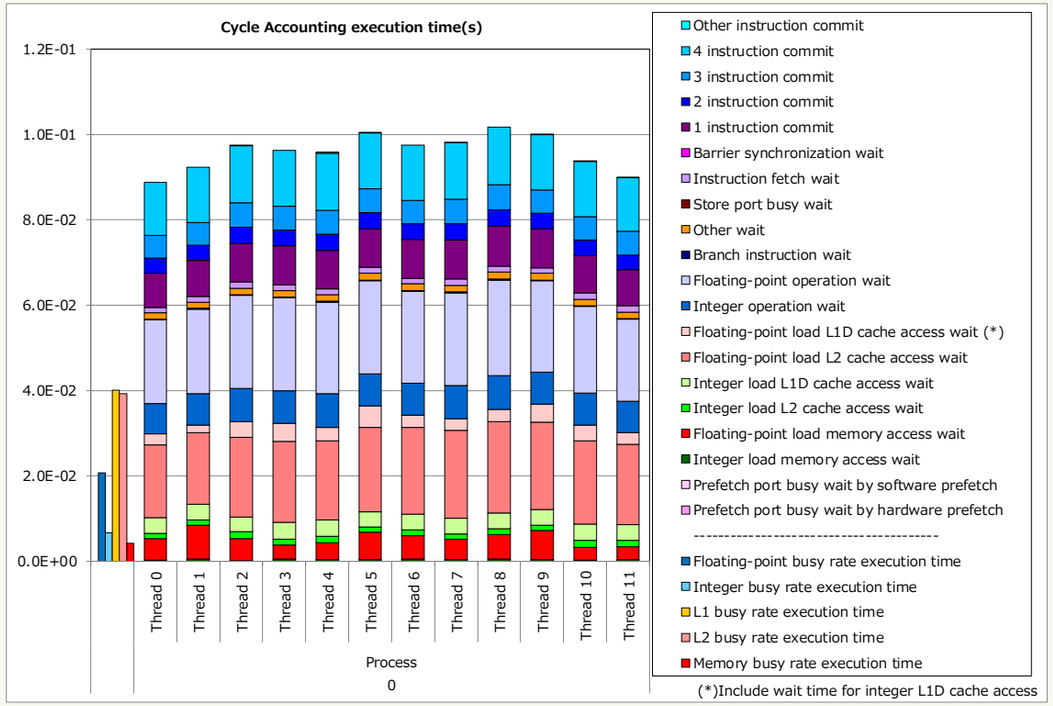
Profiling: Bulk part

1 node (2.2 GHz), $16 \times 16 \times 16 \times 16$ lattice, $1 \times 1 \times 2 \times 2$ MPI proc.



$\Longrightarrow$

×2.5 faster



busy L1 cache: loop over `struct{ float elem[16];} vsimd` caused gather load/scatter⇒rewrote with ACLE

```
void axpy(vsimd *out, float a, vsimd *in) { // slow (left)
  for(int j=0; j<N; j++){
    for(i=0; i<16; ++i){  out[x].elem[i] -= a*in[x].elem[i]; }
  }}
```

Summary and Outlooks

Summary and Outlooks

Lattice QCD: stencil computation on 4-dim structured lattice

- data layout: 2-dim lattice site for SIMD vector
- implementation with ACLE
  stencil computation with `sel`, `tbl`, `ext`, `compact`
- Profiler helps to find unintended gather load/scatters

Outlooks

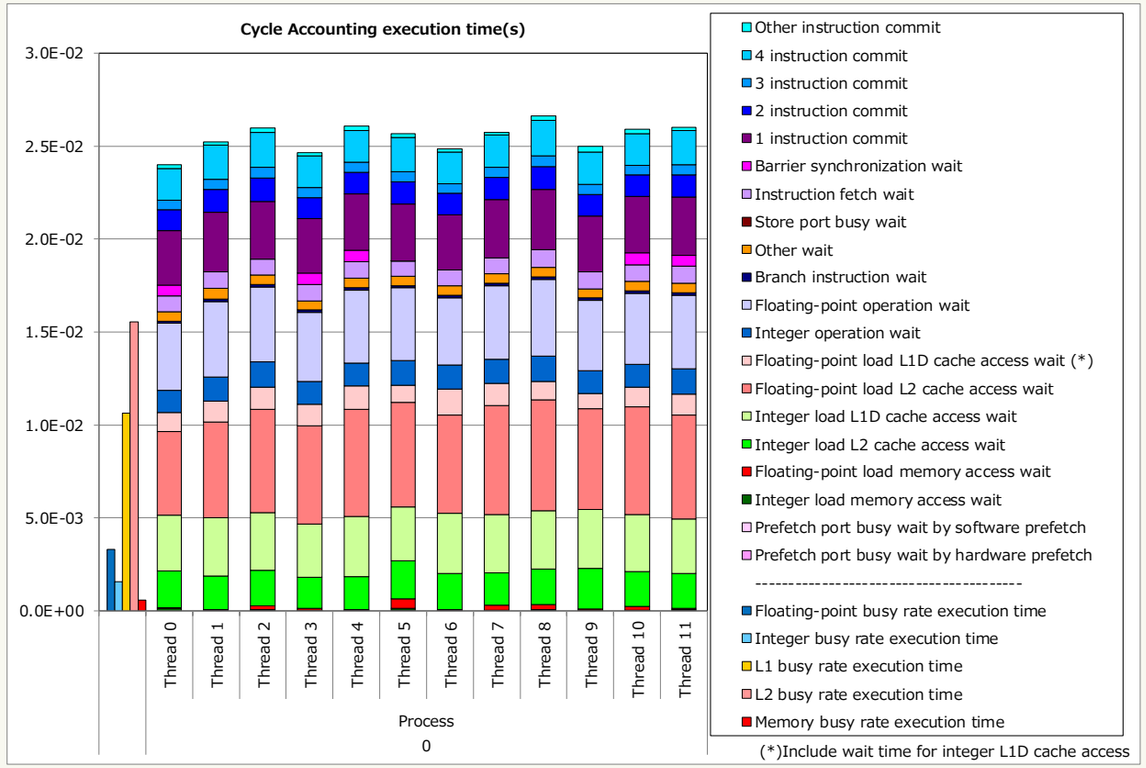
- room for some more tuning: load imbalance btw. threads
- Implementation in Bridge++ will be public soon 

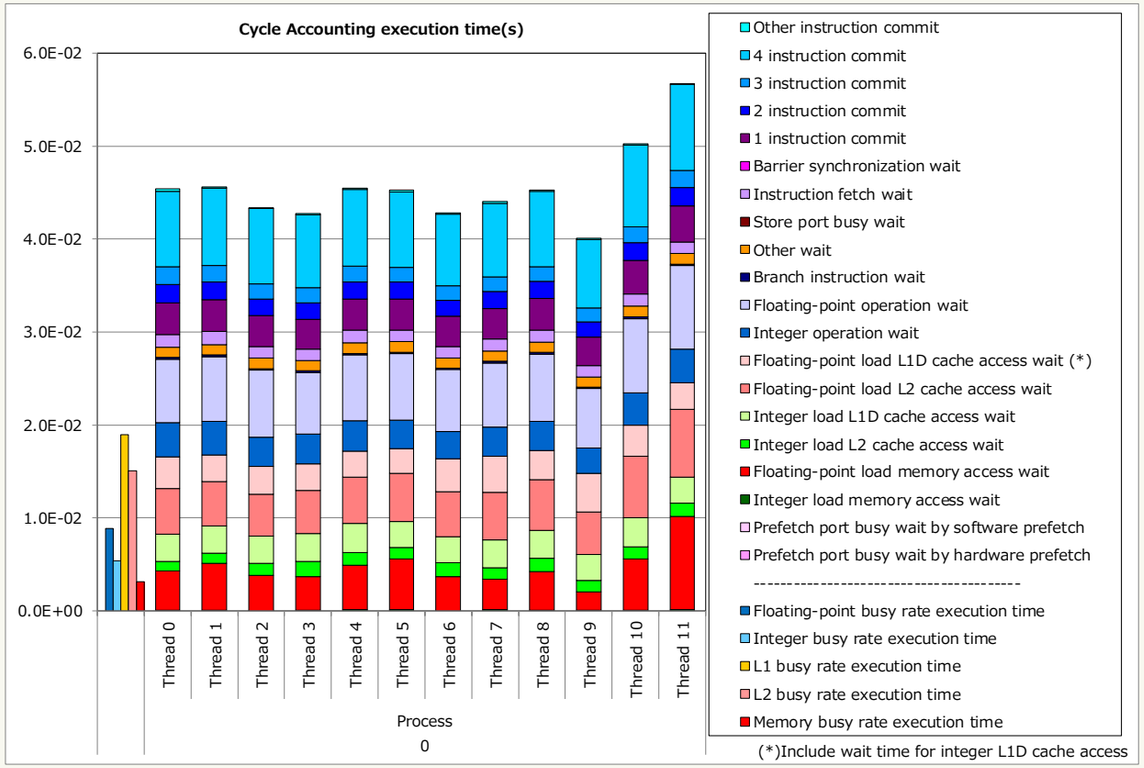
Acknowledgments

- Grants: JSPS KAKENHI (JP20K03961, JP22H01224), the MEXT as “Program for Promoting Researches on the Supercomputer Fugaku” (Simulation for basic science: from fundamental laws of particles to creation of nuclei, JPMXP1020200105) and “Priority Issue 9 to be Tackled by Using the Post-K Computer” (Elucidation of The Fundamental Laws and Evolution of the Universe), and Joint Institute for Computational Fundamental Science (JICFuS).
- Resources: supercomputer Fugaku (ra000001), RIKEN R-CCS, Flow at Information Technology Center, Nagoya U., and Wisteria/BDEC-01 Odyssey U. of Tokyo (through Multidisciplinary Cooperative Research Program in Center for Computational Sciences, U. of Tsukuba.

Backup Slides

Profiling: EO1 and EO2





- EO2: load imbalance (boundary sites)
- prefetch may help (especially EO2)

Scaling w/ and w/o Fujitsu extension of MPI: weak and strong scalings