BENCHMARKING PORTABLE STAGGERED FERMION KERNEL WRITTEN IN KOKKOS AND MPI[1]

KOKKOS USER GROUP MEETING 2023

12th December, 2023 | Simon Schlepphorst | Jülich Supercomputing Centre

[1] DOI: 10.1145/3624062.3624179

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Staggered fermions

staggered fermionic action

$$S_F = a^4 \sum_{n\in\Lambda} \bar{X}(n)\left(\sum_{\mu=1}^{4} \eta_\mu(n) \frac{U_\mu(n)X(n+\hat{\mu}) - U_\mu^\dagger(n-\hat{\mu})X(n-\hat{\mu})}{2a} + mX(n)\right)$$

- space-time coordinates $n = (n_1, n_2, n_3, n_4)$ separated by a lattice spacing a
- and $U_\mu \in \mathbf{C}^{3\times3}$, $X \in \mathbf{C}^3$
- with direction index μ
- required dimensions for naive layout [n_1][n_2][n_3][n_4][4][3][3][2]
- using $L = n_1 = n_2 = n_3$, $T = n_4$ and $L = T$ for convenience

Staggered fermions

staggered fermionic action

$$S_F = a^4 \sum_{n\in\Lambda} \bar{X}(n) \left(\sum_{\mu=1}^{4} \eta_\mu(n) \frac{U_\mu(n)X(n+\hat{\mu}) - U_\mu^\dagger(n-\hat{\mu})X(n-\hat{\mu})}{2a} + mX(n) \right)$$

arithmetic intensity (for 32-bit floating point numbers)

$$I = \frac{570\,\text{FLOP}}{792\,\text{B}} = 0.72\,\text{FLOP}/\text{B}.$$

Implementation in Kokkos and MPI

Data Layout

```
using complex_t = Kokkos::complex<float>;
using Vector_t = Kokkos::View<complex_t ****[3]>;
using Link_t = Kokkos::View<complex_t ****[4][3][3]>;
```

- use in place buffer for computation

$L^3 \cdot T = 32^4$

```
Vector_t Y(34,34,34,34);
Vector_t X(34,34,34,34);
Link_t U(33,33,33,33);
```

- and additional continuous buffer for communication with MPI
- apply local periodic boundary conditions by recalculating the site indices

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Implementation in Kokkos and MPI

- different execution spaces for bulk and halo

Execution Spaces

```
using BulkSpace_t = Kokkos::DefaultExecutionSpace;
using HaloSpace_t = Kokkos::DefaultExecutionSpace;

BulkSpace_t BulkExecSpace = BulkSpace_t();
HaloSpace_t HaloExecSpcae = HaloSpace_t();
```

- simple MDRange

Execution Policy

```
auto PB = Kokkos::MDRangePolicy<Kokkos::Rank<4>>(BulkSpace, {1,1,1,1}, {33,33,33,33});
auto PH = Kokkos::MDRangePolicy<Kokkos::Rank<4>>(HaloSpace, {1,1,1,1}, {2,33,33,33});
```

Implementation in Kokkos and MPI

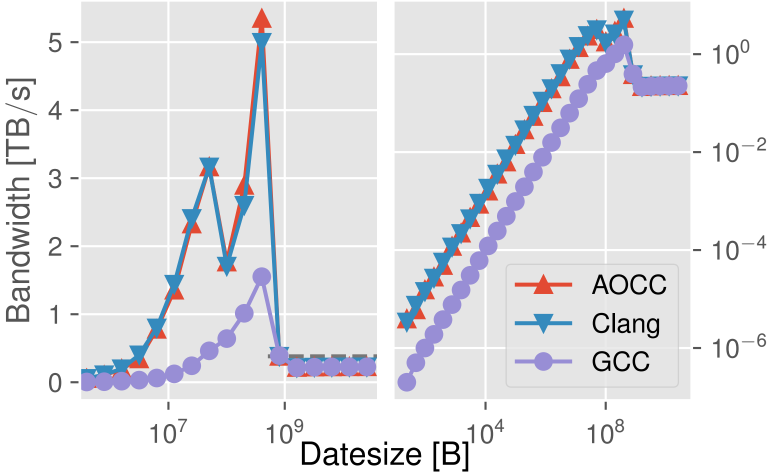
Kernel Algorithm (Input: U, X Output: Y)

$n \in \Lambda$
for $i \leftarrow 1, 2, 3$ **do**
  $t \leftarrow 0$
  for $j \leftarrow 1, 2, 3$ **do**
    for $\mu \leftarrow 1, 2, 3, 4$ **do**
      $t \leftarrow t + U_\mu(n)_{ij} * X(p(n + \hat{\mu}))_j$
      $t \leftarrow t - U^*_\mu(p(n - \hat{\mu}))_{ji} * X(p(n - \hat{\mu}))_j$
    end for
  end for
  $Y_i \leftarrow t$
end for

- $p()$ calculates the correct n according to periodic boundaries

AMD Ryzen 7742 (x86 CPU, Dual Socket)

Stream Triad

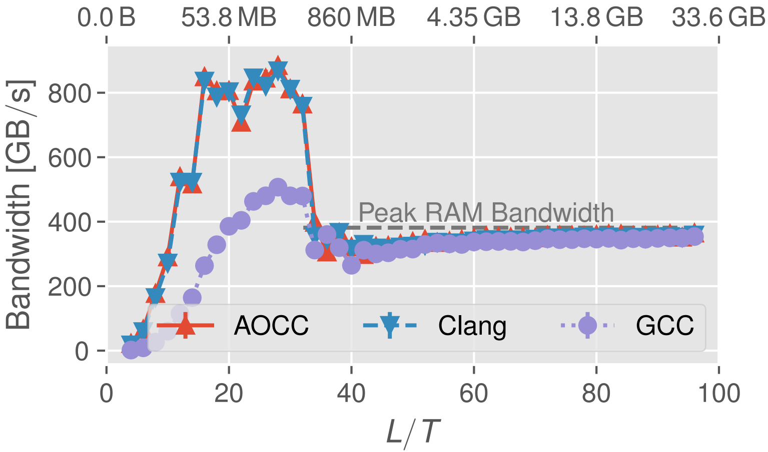


JURECA @ JSC, Kokkos 3.6, AOCC 3.2, Clang 13.0, GCC 11.2

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

AMD Ryzen 7742 (x86 CPU, Dual Socket)

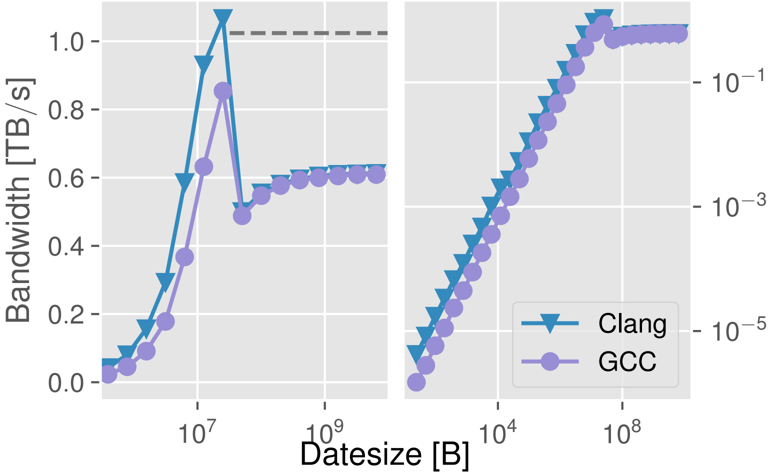
Staggered



JURECA @ JSC, Kokkos 3.6, AOCC 3.2, Clang 13.0, GCC 11.2

Fujitsu A64FX (ARM CPU)

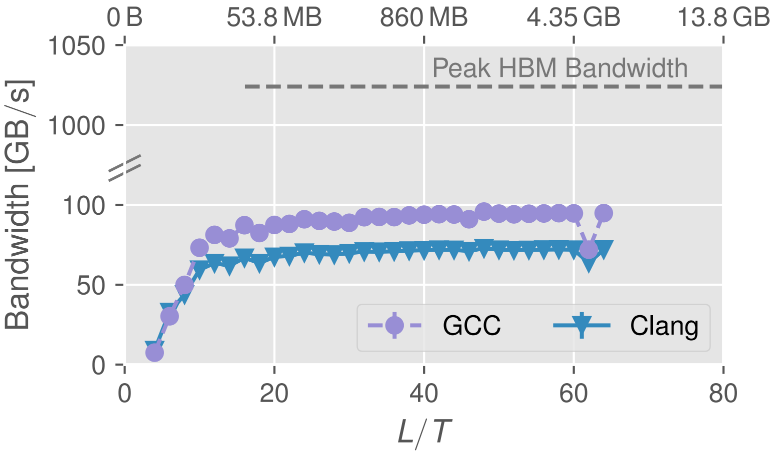
Stream Triad



CTE-ARM @ BSC, Kokkos 3.6, GCC 11.1, Clang 14.0

Fujitsu A64FX (ARM CPU)

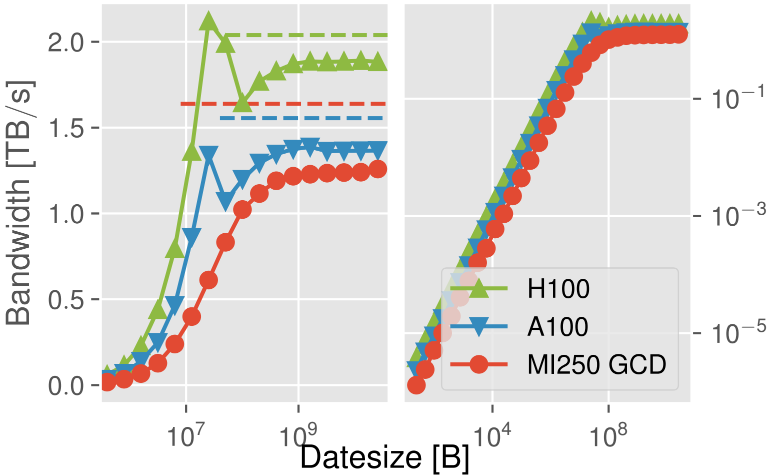
Staggered



CTE-ARM @ BSC, Kokkos 3.6, GCC 11.1, Clang 14.0

Nvidia A100, H100 and AMD MI250 (GPU)

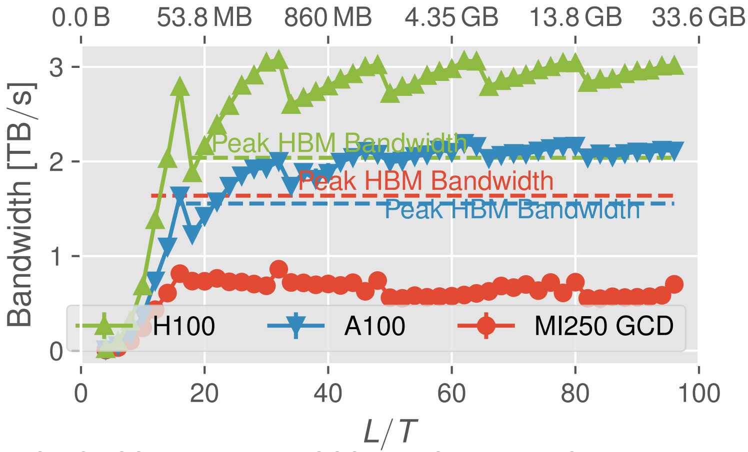
Stream Triad



A100: JURECA, H100 & MI250: JURECA Evaluation Platform

Nvidia A100, H100 and AMD MI250 (GPU)

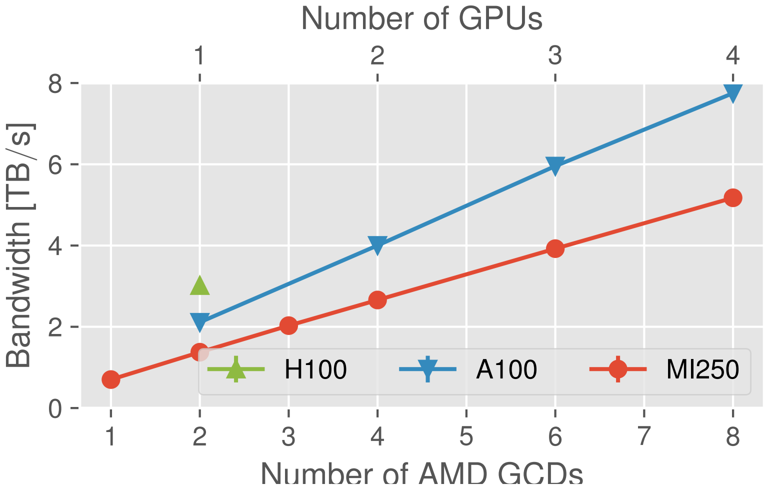
Staggered



A100: JURECA @ JSC, Kokkos 4.0.1, GCC 11.3, CUDA 11.7, OpenMPI 4.1.4, LB 384,1
H100: JURECA Evaluation Platform @ JSC, Kokkos 4.0.1, GCC 11.3, CUDA 12.0, OpenMPI 4.1.4, LB 384,1
MI250: JURECA Evaluation Platform @ JSC, Kokkos 4.0.1, Clang 15, ROCm 5.4, OpenMPI 4.1.4

Nvidia A100, H100 and AMD MI250 (GPU)

Staggered



A100: JURECA, H100 & MI250: JURECA Evaluation Platform

END

Thank you for your attention!