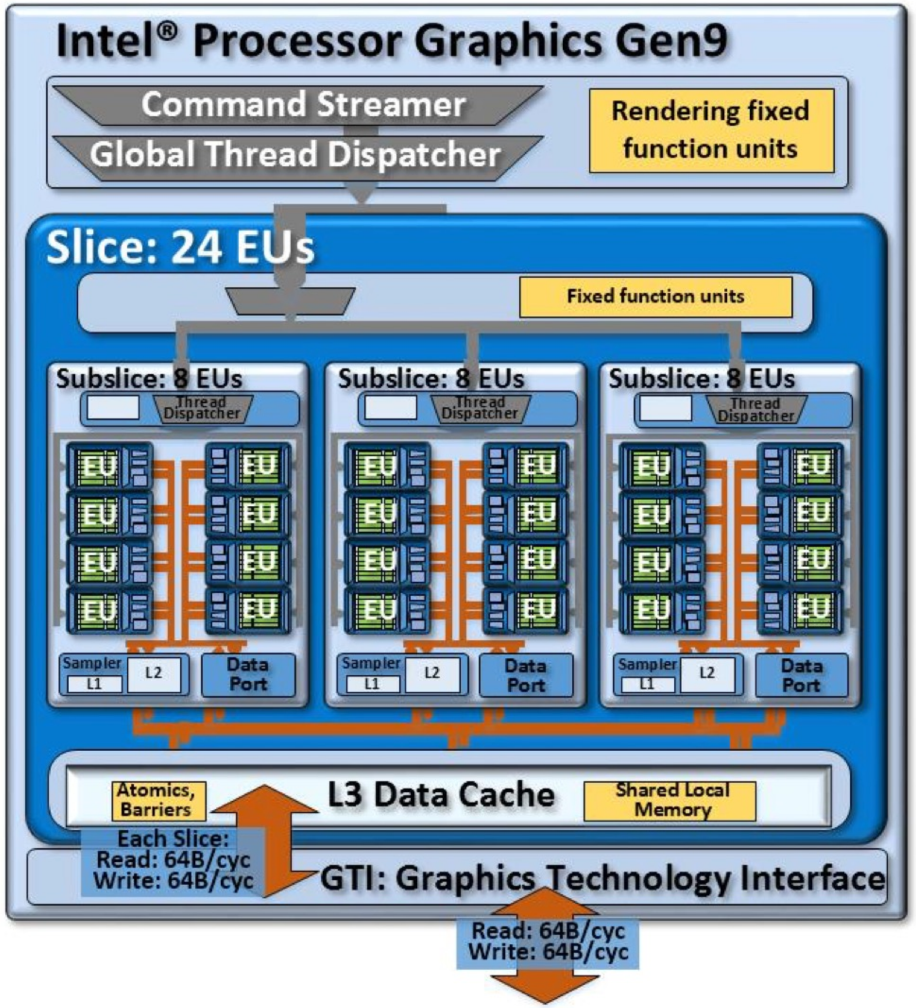
SPMD / SIMD on GPUs

Patrick Steinbrecher

Terminology

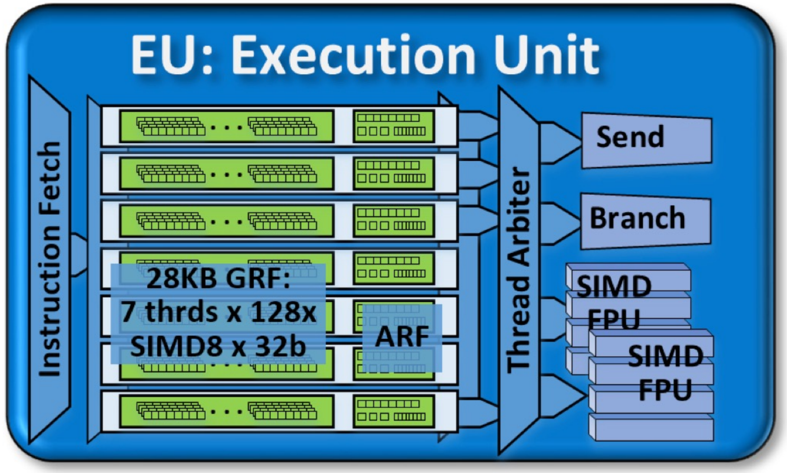
CPU	GPU
Core	EU Execution Unit
Hyper-Thread	Hardware-Thread
Vector Registers	GRF General Register File

- Hardware-thread executes SIMD
- EU has 7 hardware-threads
  - 4 kB of GRF per thread
    - e.g. 128 SIMD8 registers for 4 byte data type
- 8 EUs == 1 Sub-Slice (SS)
  - shared L1 cache per SS



Execution Unit

- HW-thread executes SIMD
  - Native SIMD length = 256 bit
- Flexible SIMD width
  - SIMD1, SIMD2, … , SIMD16, SIMD32
    - might be masked or broken down in hardware for native width
  - SIMD8 enough to reach SP-peak
  - SIMD4 = ½ SP-peak
- Code branches
  - HW-thread will execute both branches in serial and mask off SIMD lanes for each branch correspondingly

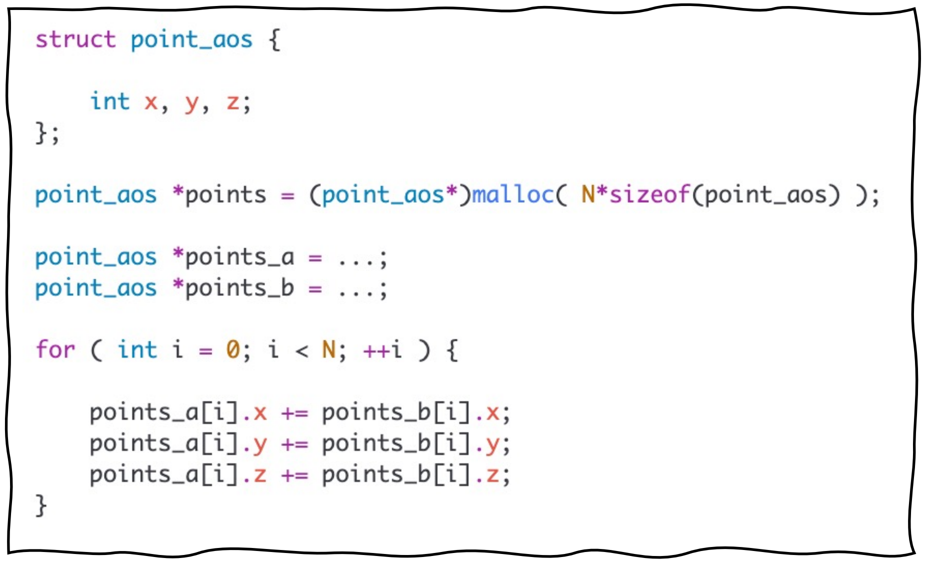


- Load/Store
  - Referred to as 'send' instruction
  - Scatter/Gather supported
    - Same performance as aligned block load if gather has addresses in same cache line
  - 'send' can permute data in load path

Programming Models

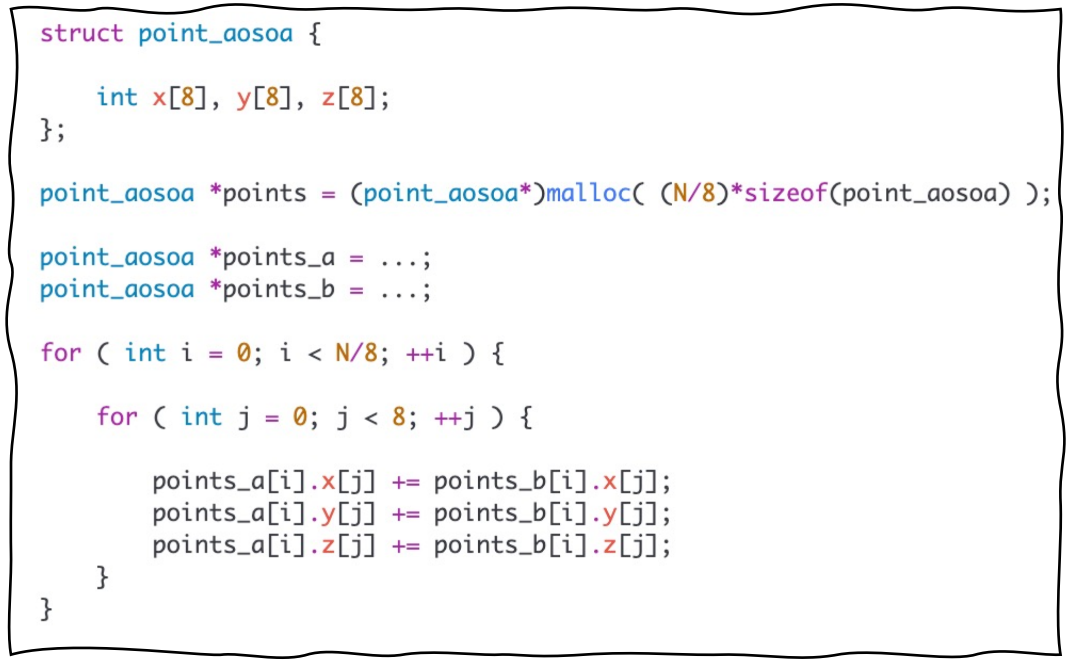
- How can parallelism be expressed?
  - SIMD: Single Instruction Multiple Data
  - SPMD: Single Program Multiple Data
- SIMD and SPMD have strong memory layout requirements
  - See next slide
- Support for SIMD and SPMD is not limited by hardware in general. It is dependent on availability of modern compilers.
- Intel GPUs and CPUs have compiler support for SPMD and SIMD!

Memory Layouts

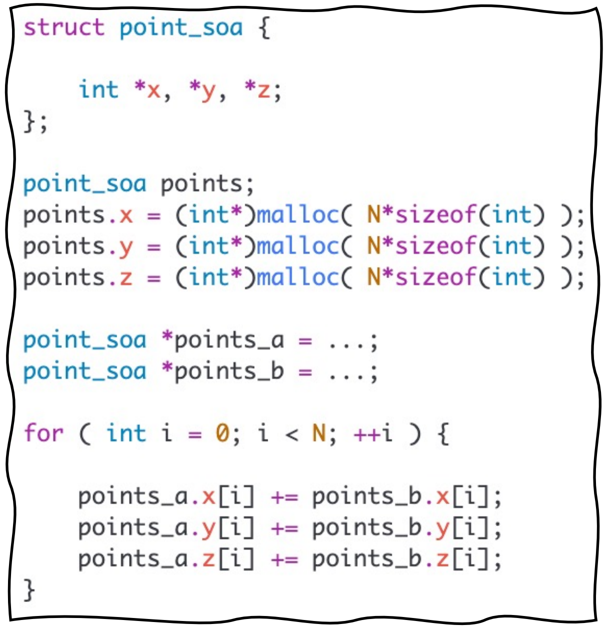


```
struct point_aos {

    int x, y, z;
};

point_aos *points = (point_aos*)malloc( N*sizeof(point_aos) );

point_aos *points_a = ...;
point_aos *points_b = ...;

for ( int i = 0; i < N; ++i ) {

    points_a[i].x += points_b[i].x;
    points_a[i].y += points_b[i].y;
    points_a[i].z += points_b[i].z;
}
```

- Array of Structs (AoS)
  - Very poor cache line utilization
- Struct of Arrays (SoA)
  - Close to 100% cache line utilization
  - In simple cases can have 100% cache line utilization
- Array of Structs of Arrays (AoSoA)
  - 100% cache line utilization
- AoSoA and SoA are required memory layouts for SPMD and SIMD programming model



```
struct point_aosoa {

    int x[8], y[8], z[8];
};

point_aosoa *points = (point_aosoa*)malloc( (N/8)*sizeof(point_aosoa) );

point_aosoa *points_a = ...;
point_aosoa *points_b = ...;

for ( int i = 0; i < N/8; ++i ) {

    for ( int j = 0; j < 8; ++j ) {

        points_a[i].x[j] += points_b[i].x[j];
        points_a[i].y[j] += points_b[i].y[j];
        points_a[i].z[j] += points_b[i].z[j];
    }
}
```



```
struct point_soa {

    int *x, *y, *z;
};

point_soa points;
points.x = (int*)malloc( N*sizeof(int) );
points.y = (int*)malloc( N*sizeof(int) );
points.z = (int*)malloc( N*sizeof(int) );

point_soa *points_a = ...;
point_soa *points_b = ...;

for ( int i = 0; i < N; ++i ) {

    points_a.x[i] += points_b.x[i];
    points_a.y[i] += points_b.y[i];
    points_a.z[i] += points_b.z[i];
}
```

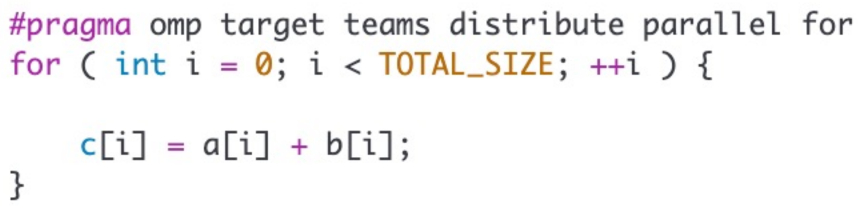
OpenMP compile-time options

- SPMD-mode
  - OpenMP thread = SIMD lane
    - Vectorization over multiple threads
  - Use GPU like a traditional GPU
    - Ideal for applications with prior OpenCL-like background

- SIMD-mode
  - OpenMP thread = EU thread
    - Vectorization within a thread
  - Use GPU like a CPU
    - Ideal for applications with prior CPU background
  - Has 1 additional level of hierarchical parallelism
    - #pragma omp simd

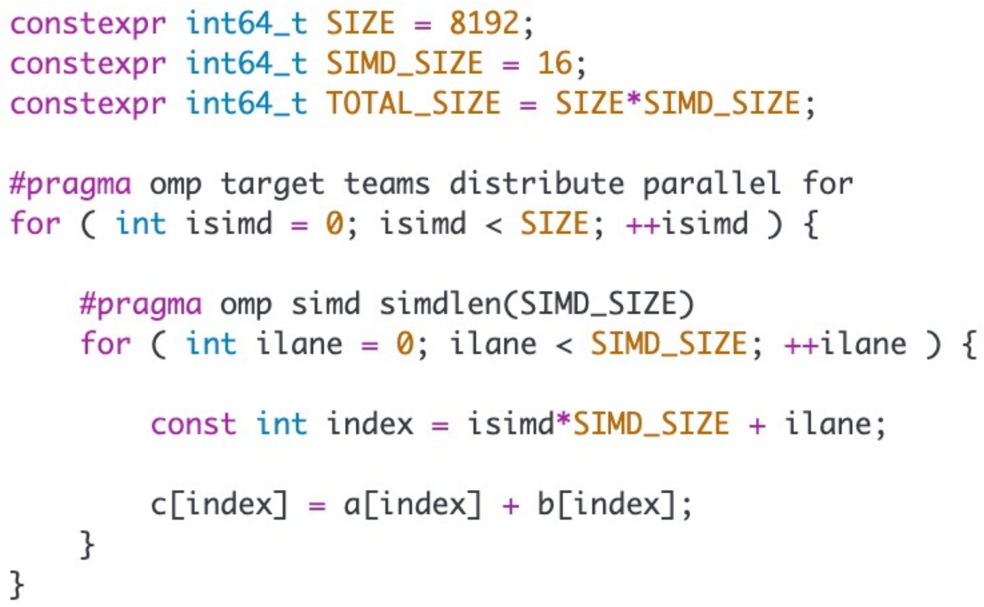
OpenMP compile-time options

- SPMD-mode



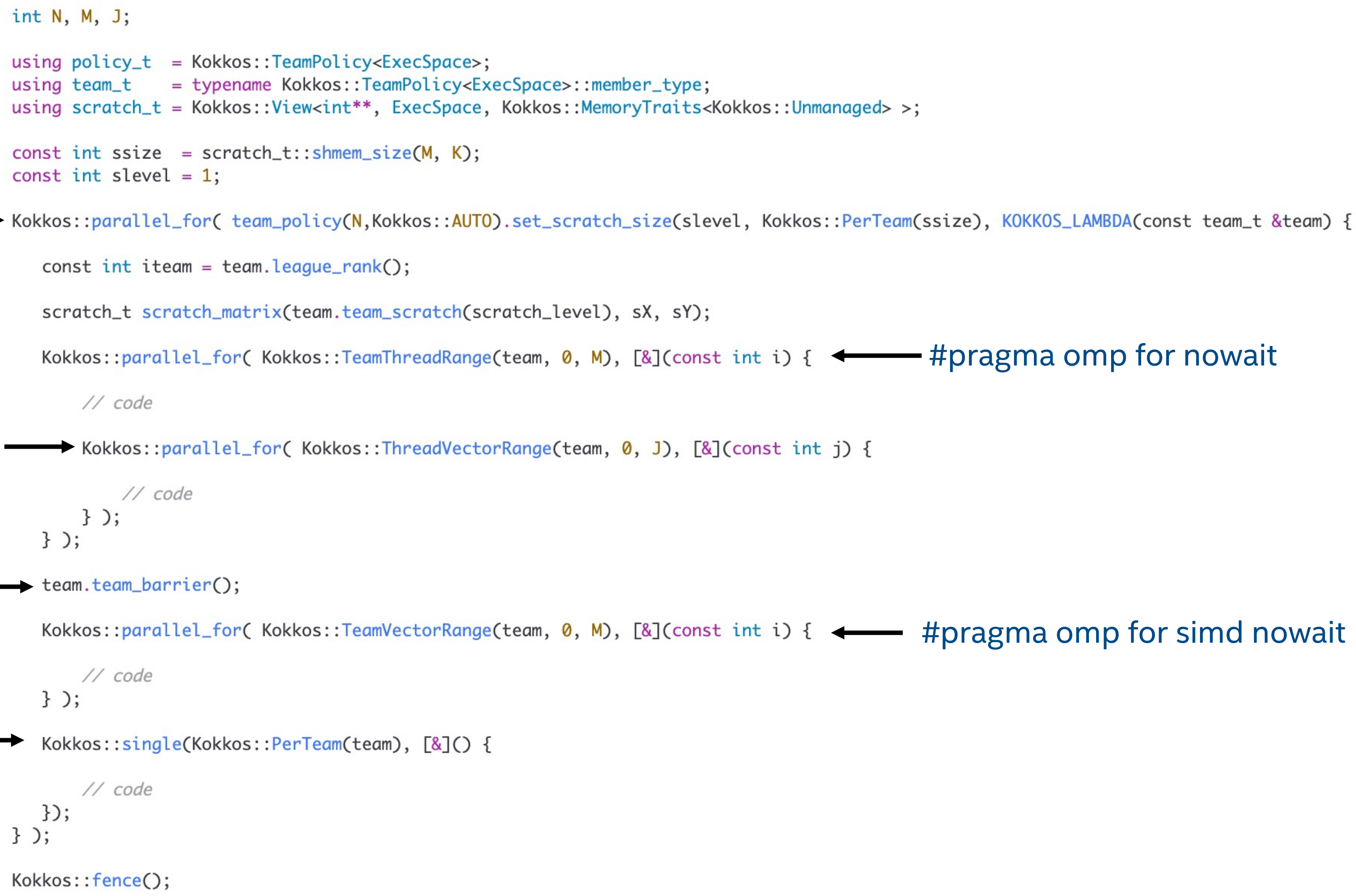
```
#pragma omp target teams distribute parallel for
for ( int i = 0; i < TOTAL_SIZE; ++i ) {

    c[i] = a[i] + b[i];
}
```

- SIMD-mode



```
constexpr int64_t SIZE = 8192;
constexpr int64_t SIMD_SIZE = 16;
constexpr int64_t TOTAL_SIZE = SIZE*SIMD_SIZE;

#pragma omp target teams distribute parallel for
for ( int isimd = 0; isimd < SIZE; ++isimd ) {

    #pragma omp simd simdlen(SIMD_SIZE)
    for ( int ilane = 0; ilane < SIMD_SIZE; ++ilane ) {

        const int index = isimd*SIMD_SIZE + ilane;

        c[index] = a[index] + b[index];
    }
}
```

Kokkos with SIMD-mode on Intel GPUs



```
int N, M, J;

using policy_t  = Kokkos::TeamPolicy<ExecSpace>;
using team_t    = typename Kokkos::TeamPolicy<ExecSpace>::member_type;
using scratch_t = Kokkos::View<int**, ExecSpace, Kokkos::MemoryTraits<Kokkos::Unmanaged> >;

const int ssize  = scratch_t::shmem_size(M, K);
const int slevel = 1;

Kokkos::parallel_for( team_policy(N,Kokkos::AUTO).set_scratch_size(slevel, Kokkos::PerTeam(ssize), KOKKOS_LAMBDA(const team_t &team) {

   const int iteam = team.league_rank();

   scratch_t scratch_matrix(team.team_scratch(scratch_level), sX, sY);

   Kokkos::parallel_for( Kokkos::TeamThreadRange(team, 0, M), [&](const int i) {

       // code

       Kokkos::parallel_for( Kokkos::ThreadVectorRange(team, 0, J), [&](const int j) {

           // code
       } );
   } );

   team.team_barrier();

   Kokkos::parallel_for( Kokkos::TeamVectorRange(team, 0, M), [&](const int i) {

       // code
   } );

   Kokkos::single(Kokkos::PerTeam(team), [&]() {

       // code
   });
} );

Kokkos::fence();
```

#pragma omp target
#pragma omp parallel

#pragma omp for nowait

#pragma omp simd

#pragma omp barrier

#pragma omp for simd nowait

if (omp_get_thread_num() == 0)

HotQCD implementation using OpenMP

- One code base that runs on both CPU and GPU
  - Without any changes in kernels between both architectures
  - No defines to change code path
  - No intrinsics
- Relies on *complex_simd<float_type,n>* class which:
  - Uses internal float_type array
  - Uses **#pragma omp simd** for SIMD-mode
  - Can load from AoSoA with scalar types for SPMD-mode

Stencil Operator

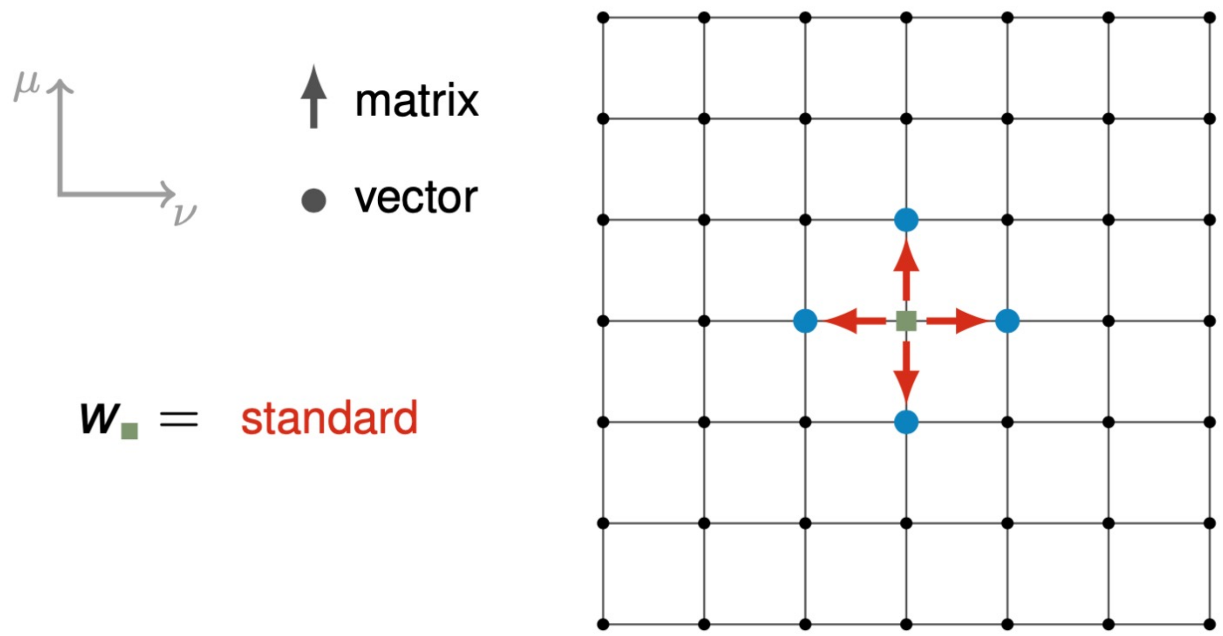
complex 3-dim vector

$$w_n = \sum_{\mu=0}^{4} \Big[\Big(U_{n,\mu} v_{n+\mu} - U^{\dagger}_{n-\mu,\mu} v_{n-\mu}\Big) + \Big(N_{n,\mu} v_{n+3\mu} - N^{\dagger}_{n-3\mu,\mu} v_{n-3\mu}\Big)\Big]$$

complex 3×3 matrix

$U(3)$ matrix

↪ reconstruct from 14 floats



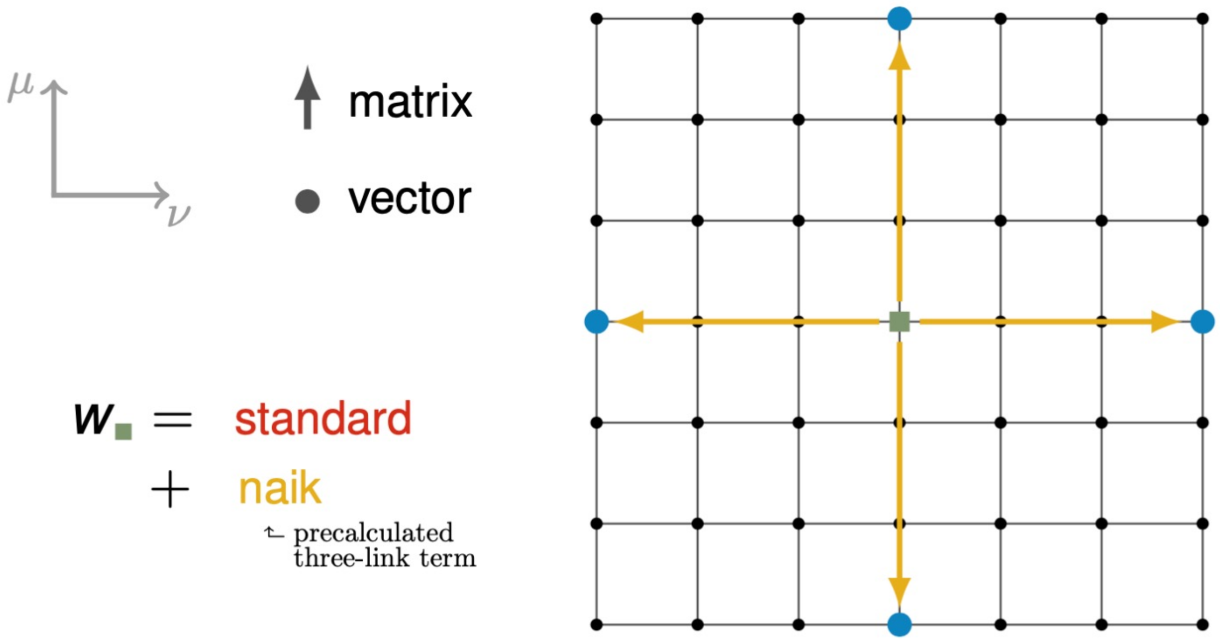
Stencil Operator

complex 3-dim vector

$$w_n = \sum_{\mu=0}^{4} \left[\left(U_{n,\mu} v_{n+\mu} - U^{\dagger}_{n-\mu,\mu} v_{n-\mu} \right) + \left(N_{n,\mu} v_{n+3\mu} - N^{\dagger}_{n-3\mu,\mu} v_{n-3\mu} \right) \right]$$

complex 3×3 matrix

$U(3)$ matrix
↪ reconstruct from 14 floats

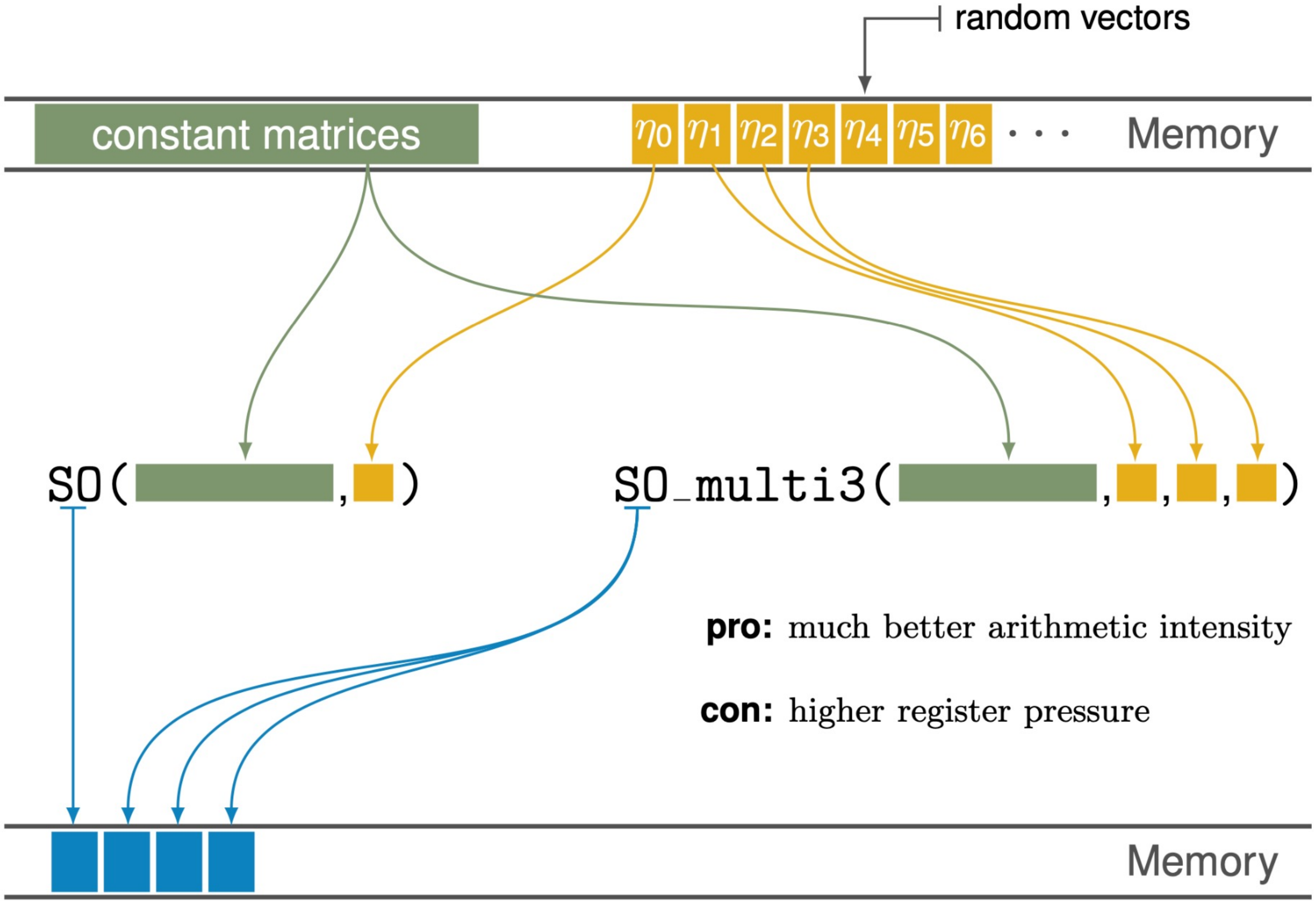


$w_{\blacksquare}$ = standard
+ naik
↰ precalculated three-link term

1146 Flop/site

0.8 Flop/byte
↪ single-precision

Multiple Right-hand Sides


random vectors
constant matrices
η_0 η_1 η_2 η_3 η_4 η_5 η_6 · · ·
Memory
SO(,)
SO_multi3(, , ,)
Memory

pro: much better arithmetic intensity

con: higher register pressure

HotQCD approach to SPMD and SIMD

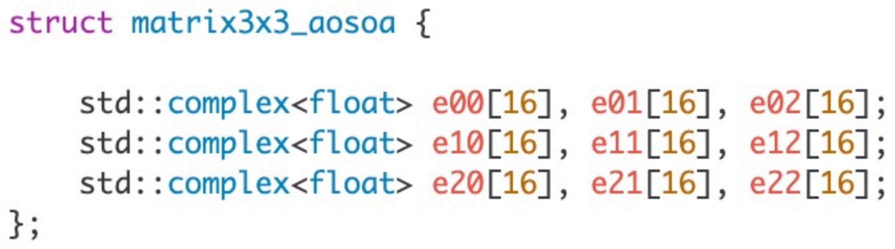
- Code is based on *complex_simd<float_type,n>* class

```
QCD<no_simd_memory_layout> stencil;
stencil.run<no_simd_kernel>();
```

- **no_simd_memory_layout:** defines AoSoA memory layout
- **no_simd_kernel:** defines if kernel uses scalar or SIMD operators

- SIMD-mode: *no_simd_memory_layout == no_simd_kernel*
- SPMD-mode: *no_simd_kernel=scalar < no_simd_memory_layout*
    - introduces outer loop over SIMD lanes
    - scalar types can read from AoSoA memory layout

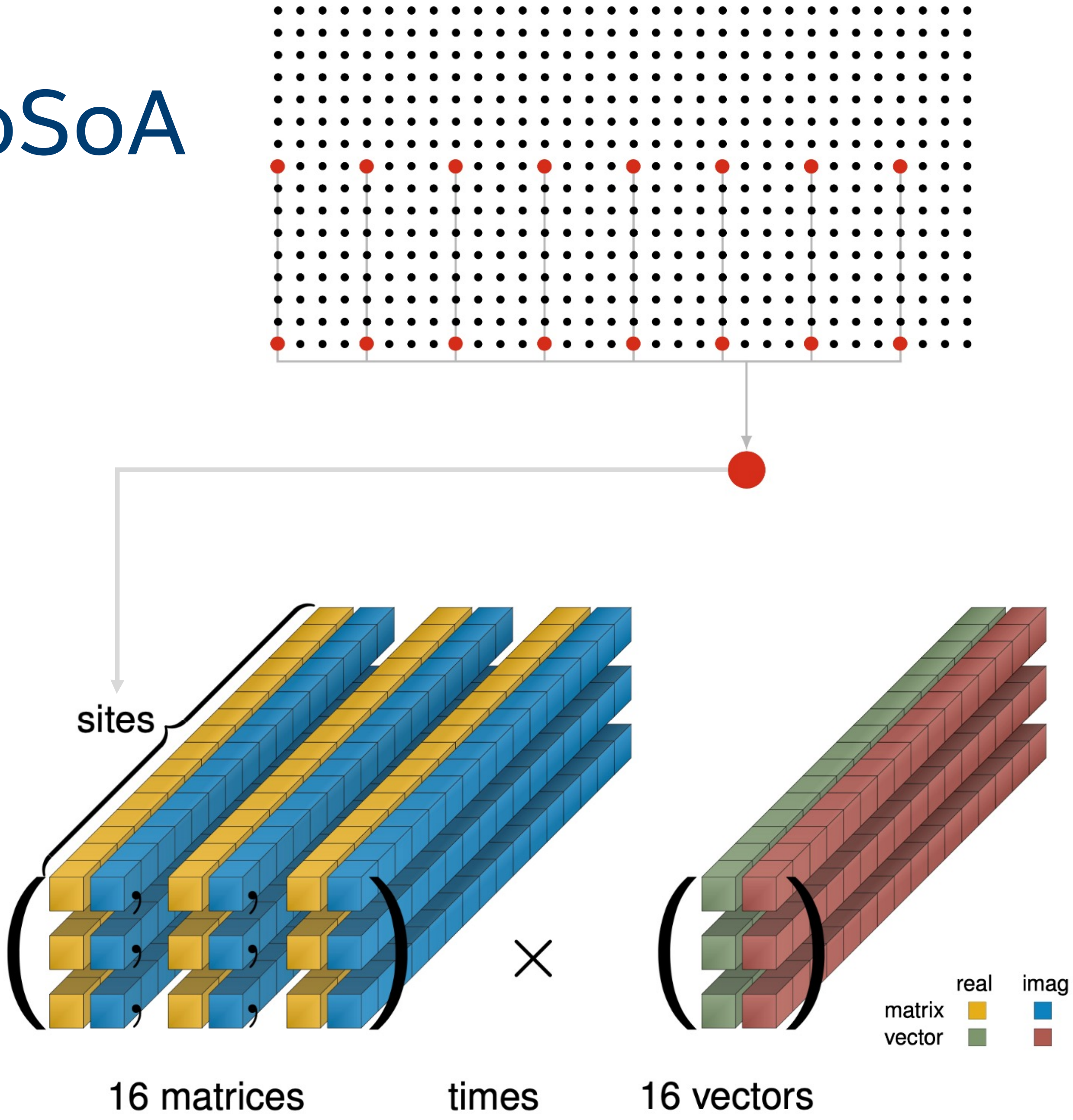
Vectorization using AoSoA



```
struct matrix3x3_aosoa {

    std::complex<float> e00[16], e01[16], e02[16];
    std::complex<float> e10[16], e11[16], e12[16];
    std::complex<float> e20[16], e21[16], e22[16];
};
```

- Store separated sites continuously in memory
  - Single Instruction = Stencil
  - Multiple Data = Sites

- Use vector registers like scalars to perform matrix times vector operations



Defines AoSoA layout

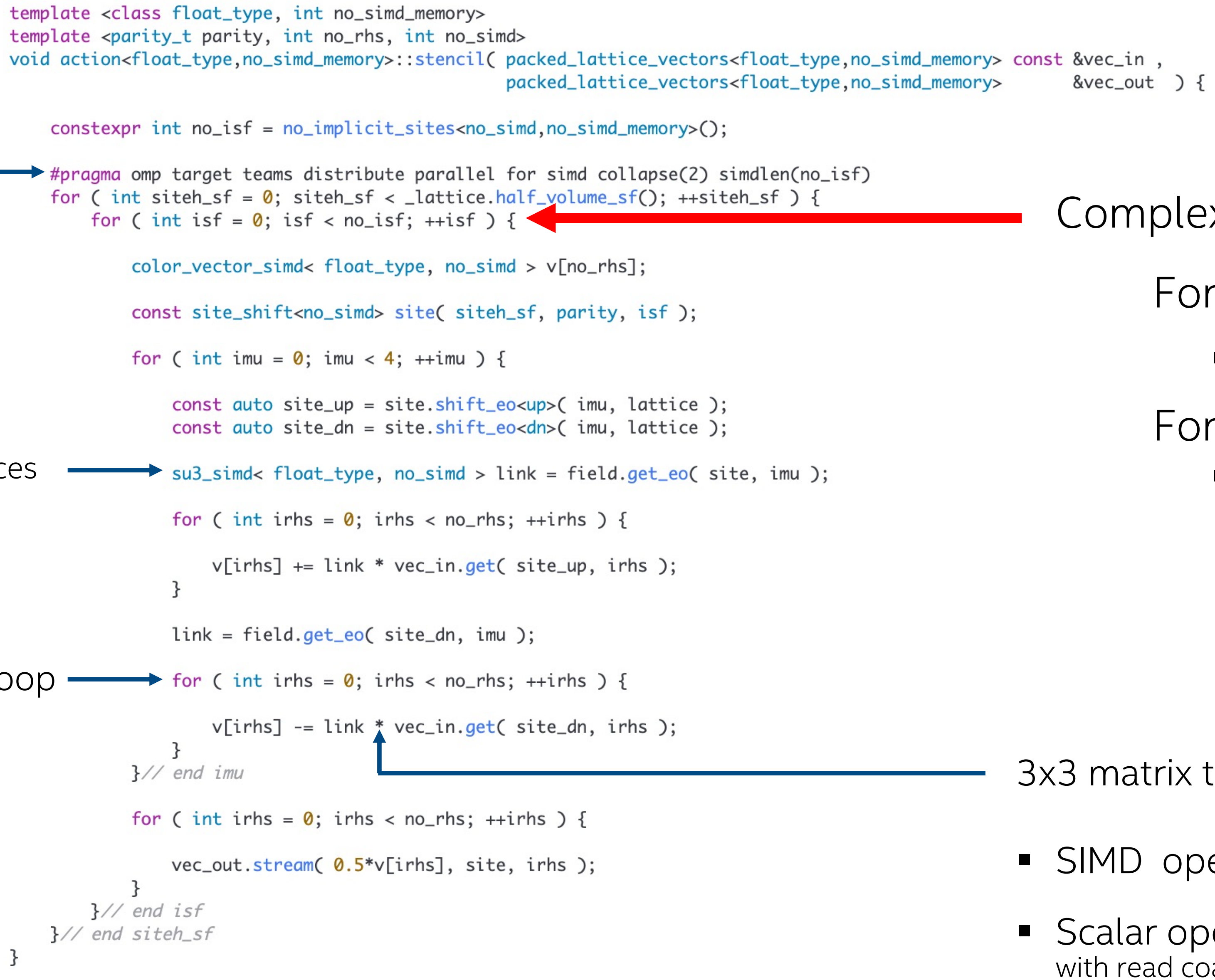
For SPMD mode

For SIMD mode:

- move simd clause inside vector class objects

no_simd 3x3 complex matrices

Right-hand side loop



```
template <class float_type, int no_simd_memory>
template <parity_t parity, int no_rhs, int no_simd>
void action<float_type,no_simd_memory>::stencil( packed_lattice_vectors<float_type,no_simd_memory> const &vec_in ,
                                                 packed_lattice_vectors<float_type,no_simd_memory>       &vec_out  ) {

    constexpr int no_isf = no_implicit_sites<no_simd,no_simd_memory>();

    #pragma omp target teams distribute parallel for simd collapse(2) simdlen(no_isf)
    for ( int siteh_sf = 0; siteh_sf < _lattice.half_volume_sf(); ++siteh_sf ) {
        for ( int isf = 0; isf < no_isf; ++isf ) {

            color_vector_simd< float_type, no_simd > v[no_rhs];

            const site_shift<no_simd> site( siteh_sf, parity, isf );

            for ( int imu = 0; imu < 4; ++imu ) {

                const auto site_up = site.shift_eo<up>( imu, lattice );
                const auto site_dn = site.shift_eo<dn>( imu, lattice );

                su3_simd< float_type, no_simd > link = field.get_eo( site, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] += link * vec_in.get( site_up, irhs );
                }

                link = field.get_eo( site_dn, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] -= link * vec_in.get( site_dn, irhs );
                }
            }// end imu

            for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                vec_out.stream( 0.5*v[irhs], site, irhs );
            }
        }// end isf
    }// end siteh_sf
}
```

Complex SIMD lane loop

For SIMD-mode:

- no_isf == 1

For SPMD-mode:

- e.g. no_isf == 8

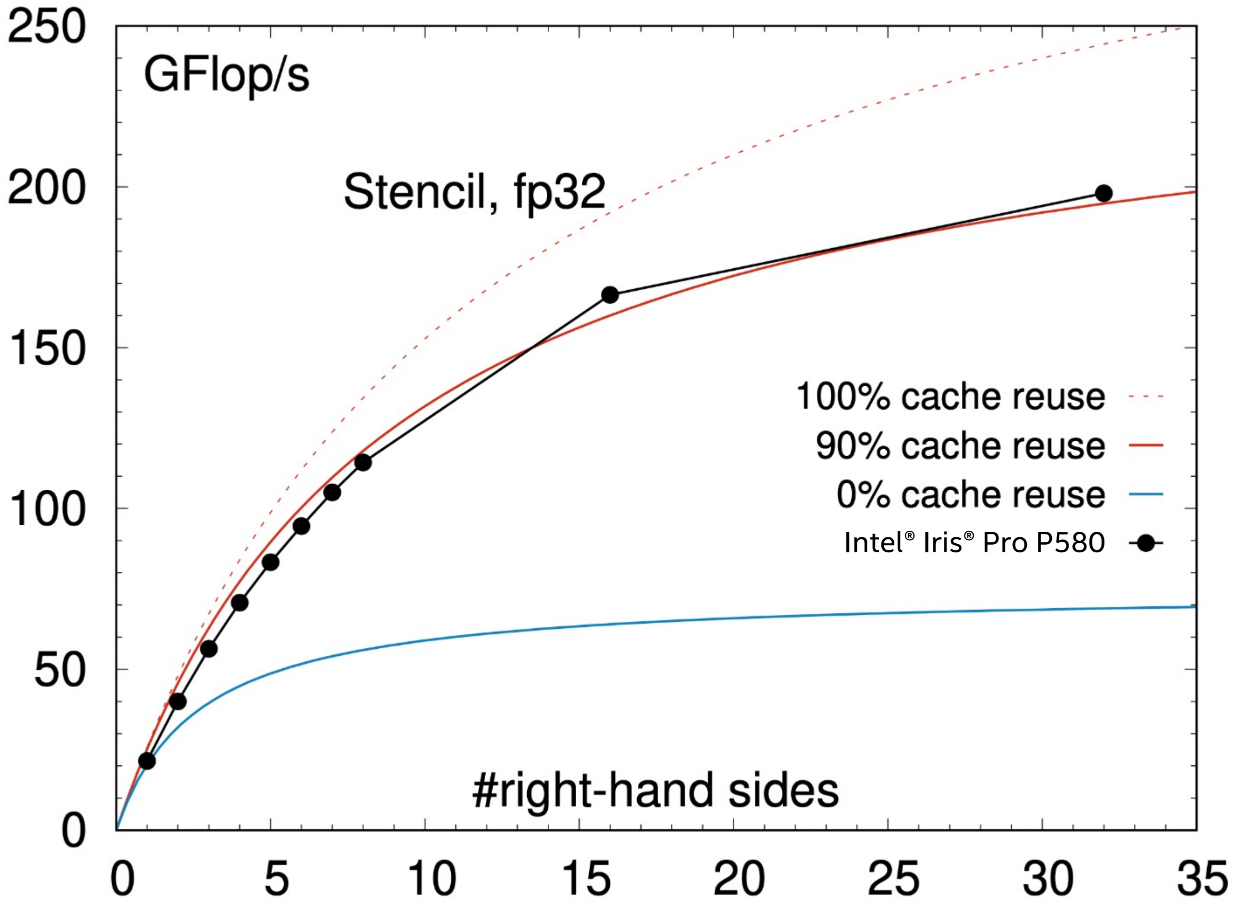
3x3 matrix times 3-dim vector

- SIMD operator for no_isf == 1
- Scalar operator for no_isf == 8
  with read coalescing

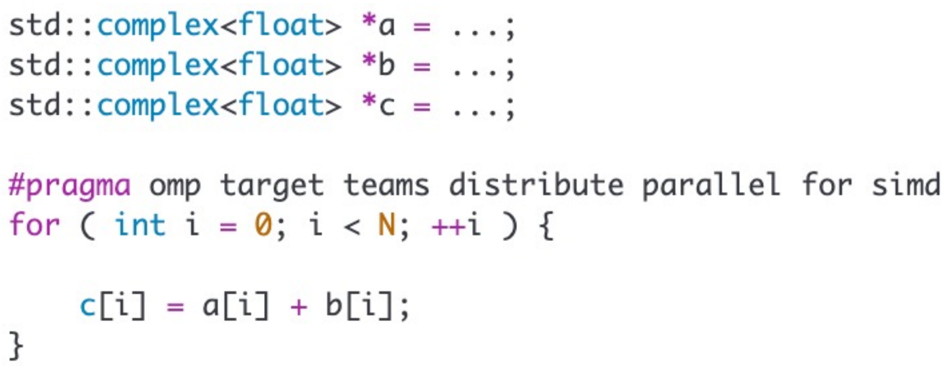
Performance on Integrated Graphics

- OpenMP compiler generates ideal code vectorization
  - See next slides

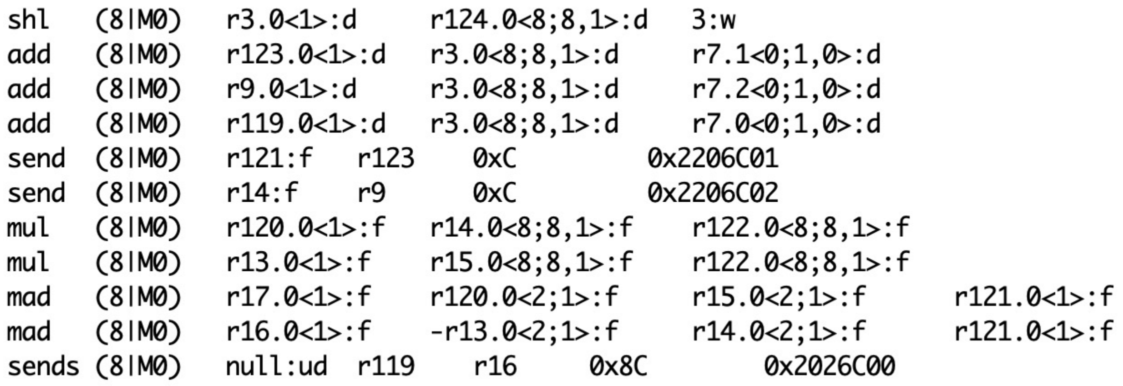
- Lattice QCD kernel runs at 90% caching efficiency
  - Scales with more right-hand sides
  - Matches theoretical expectation



Complex Multiplication



```
std::complex<float> *a = ...;
std::complex<float> *b = ...;
std::complex<float> *c = ...;

#pragma omp target teams distribute parallel for simd
for ( int i = 0; i < N; ++i ) {

    c[i] = a[i] + b[i];
}
```



```
shl   (8|M0)   r3.0<1>:d     r124.0<8;8,1>:d   3:w
add   (8|M0)   r123.0<1>:d   r3.0<8;8,1>:d     r7.1<0;1,0>:d
add   (8|M0)   r9.0<1>:d     r3.0<8;8,1>:d     r7.2<0;1,0>:d
add   (8|M0)   r119.0<1>:d   r3.0<8;8,1>:d     r7.0<0;1,0>:d
send  (8|M0)   r121:f   r123    0xC         0x2206C01
send  (8|M0)   r14:f    r9      0xC         0x2206C02
mul   (8|M0)   r120.0<1>:f   r14.0<8;8,1>:f    r122.0<8;8,1>:f
mul   (8|M0)   r13.0<1>:f    r15.0<8;8,1>:f    r122.0<8;8,1>:f
mad   (8|M0)   r17.0<1>:f    r120.0<2;1>:f     r15.0<2;1>:f      r121.0<1>:f
mad   (8|M0)   r16.0<1>:f    -r13.0<2;1>:f     r14.0<2;1>:f      r121.0<1>:f
sends (8|M0)   null:ud  r119    r16     0x8C        0x2026C00
```

Each send loads 16 floats

SIMD8 complex mul

```
template <class float_type, int no_simd_memory>
template <parity_t parity, int no_rhs, int no_simd>
void action<float_type,no_simd_memory>::stencil( packed_lattice_vectors<float_type,no_simd_memory> const &vec_in ,
                                                 packed_lattice_vectors<float_type,no_simd_memory>       &vec_out  ) {

    constexpr int no_isf = no_implicit_sites<no_simd,no_simd_memory>();

    #pragma omp target teams distribute parallel for simd collapse(2) simdlen(no_isf)
    for ( int siteh_sf = 0; siteh_sf < _lattice.half_volume_sf(); ++siteh_sf ) {
        for ( int isf = 0; isf < no_isf; ++isf ) {

            color_vector_simd< float_type, no_simd > v[no_rhs];

            const site_shift<no_simd> site( siteh_sf, parity, isf );

            for ( int imu = 0; imu < 4; ++imu ) {

                const auto site_up = site.shift_eo<up>( imu, lattice );
                const auto site_dn = site.shift_eo<dn>( imu, lattice );

                su3_simd< float_type, no_simd > link = field.get_eo( site, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] += link * vec_in.get( site_up, irhs );
                }

                link = field.get_eo( site_dn, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] -= link * vec_in.get( site_dn, irhs );
                }
            }// end imu

            for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                vec_out.stream( 0.5*v[irhs], site, irhs );
            }
        }// end isf
    }// end siteh_sf
}
```

no_isf 3x3 complex matrices

- no_isf == 8
- Requires 9 SIMD16 loads

```
L2536:
        shl (8|M0)                r2.0<1>:d    r8.0<8;8,1>:d      5:w
(W)     add (1|M0)                r9.1<1>:q    r9.1<0;1,0>:q      1:w
        add (8|M0)                r8.0<1>:d    r2.0<8;8,1>:d      r83.0<8;8,1>:d
        add (8|M0)                r2.0<1>:d    r61.0<8;8,1>:d     r9.4<0;1,0>:d
(W)     cmp (8|M0)    (eq)f0.1    null<1>:q    r9.1<0;1,0>:q      4:w
        mul (8|M0)                r3.0<1>:d    r8.0<8;8,1>:d      192:w
        mul (8|M0)                r2.0<1>:d    r2.0<8;8,1>:d      576:w
        add (8|M0)                r3.0<1>:d    r102.0<8;8,1>:d    r3.0<8;8,1>:d
        add (8|M0)                r2.0<1>:d    r58.0<8;8,1>:d     r2.0<8;8,1>:d
        add (8|M0)                r11.0<1>:d   r3.0<8;8,1>:d      64:w
        add (8|M0)                r4.0<1>:d    r2.0<8;8,1>:d      64:w
        add (8|M0)                r5.0<1>:d    r2.0<8;8,1>:d      192:w
        add (8|M0)                r10.0<1>:d   r2.0<8;8,1>:d      448:w
        add (8|M0)                r6.0<1>:d    r2.0<8;8,1>:d      256:w
        add (8|M0)                r7.0<1>:d    r2.0<8;8,1>:d      384:w
        send (8|M0)               r80:f    r2     0xC             0x02206C01
        send (8|M0)               r78:f    r4     0xC             0x02206C01
        send (8|M0)               r74:f    r5     0xC             0x02206C01
        send (8|M0)               r66:f    r10    0xC             0x02206C01
        send (8|M0)               r4:f     r11    0xC             0x02206C02
        send (8|M0)               r72:f    r6     0xC             0x02206C01
        add (8|M0)                r10.0<1>:d   r2.0<8;8,1>:d      128:w
        send (8|M0)               r68:f    r7     0xC             0x02206C01
        add (8|M0)                r11.0<1>:d   r2.0<8;8,1>:d      320:w
        send (8|M0)               r6:f     r3     0xC             0x02206C02
        add (8|M0)                r2.0<1>:d    r2.0<8;8,1>:d      512:w
        add (8|M0)                r3.0<1>:d    r3.0<8;8,1>:d      128:w
        send (8|M0)               r76:f    r10    0xC             0x02206C01
        send (8|M0)               r70:f    r11    0xC             0x02206C01
        send (8|M0)               r64:f    r2     0xC             0x02206C01
        send (8|M0)               r2:f     r3     0xC             0x02206C02
        mul (8|M0)                r10.0<1>:f   r79.0<8;8,1>:f     r5.0<8;8,1>:f
        mul (8|M0)                r12.0<1>:f   r73.0<8;8,1>:f     r5.0<8;8,1>:f
        mul (8|M0)                r15.0<1>:f   r67.0<8;8,1>:f     r5.0<8;8,1>:f
        mul (8|M0)                r11.0<1>:f   r79.0<8;8,1>:f     r4.0<8;8,1>:f
        mul (8|M0)                r14.0<1>:f   r67.0<8;8,1>:f     r4.0<8;8,1>:f
        mul (8|M0)                r29.0<1>:f   r81.0<8;8,1>:f     r7.0<8;8,1>:f
        mul (8|M0)                r19.0<1>:f   r75.0<8;8,1>:f     r7.0<8;8,1>:f
        mul (8|M0)                r17.0<1>:f   r69.0<8;8,1>:f     r7.0<8;8,1>:f
        mul (8|M0)                r28.0<1>:f   r81.0<8;8,1>:f     r6.0<8;8,1>:f
        mul (8|M0)                r18.0<1>:f   r75.0<8;8,1>:f     r6.0<8;8,1>:f
        mul (8|M0)                r16.0<1>:f   r69.0<8;8,1>:f     r6.0<8;8,1>:f
        mad (8|M0)                r10.0<1>:f   -r10.0<2;1>:f      r78.0<2;1>:f      r4.0<1>:f
        mad (8|M0)                r29.0<1>:f   -r29.0<2;1>:f      r80.0<2;1>:f      r6.0<1>:f
        mad (8|M0)                r12.0<1>:f   -r12.0<2;1>:f      r72.0<2;1>:f      r4.0<1>:f
```

```
template <class float_type, int no_simd_memory>
template <parity_t parity, int no_rhs, int no_simd>
void action<float_type,no_simd_memory>::stencil( packed_lattice_vectors<float_type,no_simd_memory> const &vec_in ,
                                                 packed_lattice_vectors<float_type,no_simd_memory>       &vec_out  ) {

    constexpr int no_isf = no_implicit_sites<no_simd,no_simd_memory>();

    #pragma omp target teams distribute parallel for simd collapse(2) simdlen(no_isf)
    for ( int siteh_sf = 0; siteh_sf < _lattice.half_volume_sf(); ++siteh_sf ) {
        for ( int isf = 0; isf < no_isf; ++isf ) {

            color_vector_simd< float_type, no_simd > v[no_rhs];

            const site_shift<no_simd> site( siteh_sf, parity, isf );

            for ( int imu = 0; imu < 4; ++imu ) {

                const auto site_up = site.shift_eo<up>( imu, lattice );
                const auto site_dn = site.shift_eo<dn>( imu, lattice );

                su3_simd< float_type, no_simd > link = field.get_eo( site, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] += link * vec_in.get( site_up, irhs );
                }

                link = field.get_eo( site_dn, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] -= link * vec_in.get( site_dn, irhs );
                }
            }// end imu

            for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                vec_out.stream( 0.5*v[irhs], site, irhs );
            }
        }// end isf
    }// end siteh_sf
}
```

```
L2536:
     shl (8|M0)              r2.0<1>:d     r8.0<8;8,1>:d     5:w
(W)  add (1|M0)              r9.1<1>:q     r9.1<0;1,0>:q     1:w
     add (8|M0)              r8.0<1>:d     r2.0<8;8,1>:d     r83.0<8;8,1>:d
     add (8|M0)              r2.0<1>:d     r61.0<8;8,1>:d    r9.4<0;1,0>:d
(W)  cmp (8|M0)   (eq)f0.1   null<1>:q     r9.1<0;1,0>:q     4:w
     mul (8|M0)              r3.0<1>:d     r8.0<8;8,1>:d     192:w
     mul (8|M0)              r2.0<1>:d     r2.0<8;8,1>:d     576:w
     add (8|M0)              r3.0<1>:d     r102.0<8;8,1>:d   r3.0<8;8,1>:d
     add (8|M0)              r2.0<1>:d     r58.0<8;8,1>:d    r2.0<8;8,1>:d
     add (8|M0)              r11.0<1>:d    r3.0<8;8,1>:d     64:w
     add (8|M0)              r4.0<1>:d     r2.0<8;8,1>:d     64:w
     add (8|M0)              r5.0<1>:d     r2.0<8;8,1>:d     192:w
     add (8|M0)              r10.0<1>:d    r2.0<8;8,1>:d     448:w
     add (8|M0)              r6.0<1>:d     r2.0<8;8,1>:d     256:w
     add (8|M0)              r7.0<1>:d     r2.0<8;8,1>:d     384:w
     send (8|M0)             r80:f    r2      0xC            0x02206C01
     send (8|M0)             r78:f    r4      0xC            0x02206C01
     send (8|M0)             r74:f    r5      0xC            0x02206C01
     send (8|M0)             r66:f    r10     0xC            0x02206C01
     send (8|M0)             r4:f     r11     0xC            0x02206C02
     send (8|M0)             r72:f    r6      0xC            0x02206C01
     add (8|M0)              r10.0<1>:d    r2.0<8;8,1>:d     128:w
     send (8|M0)             r68:f    r7      0xC            0x02206C01
     add (8|M0)              r11.0<1>:d    r2.0<8;8,1>:d     320:w
     send (8|M0)             r6:f     r3      0xC            0x02206C02
     add (8|M0)              r2.0<1>:d     r2.0<8;8,1>:d     512:w
     add (8|M0)              r3.0<1>:d     r3.0<8;8,1>:d     128:w
     send (8|M0)             r76:f    r10     0xC            0x02206C01
     send (8|M0)             r70:f    r11     0xC            0x02206C01
     send (8|M0)             r64:f    r2      0xC            0x02206C01
     send (8|M0)             r2:f     r3      0xC            0x02206C02
     mul (8|M0)              r10.0<1>:f    r79.0<8;8,1>:f    r5.0<8;8,1>:f
     mul (8|M0)              r12.0<1>:f    r73.0<8;8,1>:f    r5.0<8;8,1>:f
     mul (8|M0)              r15.0<1>:f    r67.0<8;8,1>:f    r5.0<8;8,1>:f
     mul (8|M0)              r11.0<1>:f    r79.0<8;8,1>:f    r4.0<8;8,1>:f
     mul (8|M0)              r14.0<1>:f    r67.0<8;8,1>:f    r4.0<8;8,1>:f
     mul (8|M0)              r29.0<1>:f    r81.0<8;8,1>:f    r7.0<8;8,1>:f
     mul (8|M0)              r19.0<1>:f    r75.0<8;8,1>:f    r7.0<8;8,1>:f
     mul (8|M0)              r17.0<1>:f    r69.0<8;8,1>:f    r7.0<8;8,1>:f
     mul (8|M0)              r28.0<1>:f    r81.0<8;8,1>:f    r6.0<8;8,1>:f
     mul (8|M0)              r18.0<1>:f    r75.0<8;8,1>:f    r6.0<8;8,1>:f
     mul (8|M0)              r16.0<1>:f    r69.0<8;8,1>:f    r6.0<8;8,1>:f
     mad (8|M0)              r10.0<1>:f    -r10.0<2;1>:f     r78.0<2;1>:f     r4.0<1>:f
     mad (8|M0)              r29.0<1>:f    -r29.0<2;1>:f     r80.0<2;1>:f     r6.0<1>:f
     mad (8|M0)              r12.0<1>:f    -r12.0<2;1>:f     r72.0<2;1>:f     r4.0<1>:f
```

no_isf 3-dim complex vectors

- no_isf == 8
- Requires 3 SIMD16 loads

```
template <class float_type, int no_simd_memory>
template <parity_t parity, int no_rhs, int no_simd>
void action<float_type,no_simd_memory>::stencil( packed_lattice_vectors<float_type,no_simd_memory> const &vec_in ,
                                                 packed_lattice_vectors<float_type,no_simd_memory>       &vec_out  ) {

    constexpr int no_isf = no_implicit_sites<no_simd,no_simd_memory>();

    #pragma omp target teams distribute parallel for simd collapse(2) simdlen(no_isf)
    for ( int siteh_sf = 0; siteh_sf < _lattice.half_volume_sf(); ++siteh_sf ) {
        for ( int isf = 0; isf < no_isf; ++isf ) {

            color_vector_simd< float_type, no_simd > v[no_rhs];

            const site_shift<no_simd> site( siteh_sf, parity, isf );

            for ( int imu = 0; imu < 4; ++imu ) {

                const auto site_up = site.shift_eo<up>( imu, lattice );
                const auto site_dn = site.shift_eo<dn>( imu, lattice );

                su3_simd< float_type, no_simd > link = field.get_eo( site, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] += link * vec_in.get( site_up, irhs );
                }

                link = field.get_eo( site_dn, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] -= link * vec_in.get( site_dn, irhs );
                }
            }// end imu

            for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                vec_out.stream( 0.5*v[irhs], site, irhs );
            }
        }// end isf
    }// end siteh_sf
}
```

no_isf matrix times vector

```
mul (8|M0)          r12.0<1>:f    r73.0<8;8,1>:f    r5.0<8;8,1>:f
mul (8|M0)          r15.0<1>:f    r67.0<8;8,1>:f    r5.0<8;8,1>:f
mul (8|M0)          r11.0<1>:f    r79.0<8;8,1>:f    r4.0<8;8,1>:f
mul (8|M0)          r14.0<1>:f    r67.0<8;8,1>:f    r4.0<8;8,1>:f
mul (8|M0)          r29.0<1>:f    r81.0<8;8,1>:f    r7.0<8;8,1>:f
mul (8|M0)          r19.0<1>:f    r75.0<8;8,1>:f    r7.0<8;8,1>:f
mul (8|M0)          r17.0<1>:f    r69.0<8;8,1>:f    r7.0<8;8,1>:f
mul (8|M0)          r28.0<1>:f    r81.0<8;8,1>:f    r6.0<8;8,1>:f
mul (8|M0)          r18.0<1>:f    r75.0<8;8,1>:f    r6.0<8;8,1>:f
mul (8|M0)          r16.0<1>:f    r69.0<8;8,1>:f    r6.0<8;8,1>:f
mad (8|M0)          r10.0<1>:f    -r10.0<2;1>:f     r78.0<2;1>:f      r4.0<1>:f
mad (8|M0)          r29.0<1>:f    -r29.0<2;1>:f     r80.0<2;1>:f      r6.0<1>:f
mad (8|M0)          r12.0<1>:f    -r12.0<2;1>:f     r72.0<2;1>:f      r4.0<1>:f
mul (8|M0)          r27.0<1>:f    r77.0<8;8,1>:f    r3.0<8;8,1>:f
mad (8|M0)          r19.0<1>:f    -r19.0<2;1>:f     r74.0<2;1>:f      r6.0<1>:f
mul (8|M0)          r31.0<1>:f    r71.0<8;8,1>:f    r3.0<8;8,1>:f
mad (8|M0)          r17.0<1>:f    -r17.0<2;1>:f     r68.0<2;1>:f      r6.0<1>:f
mad (8|M0)          r15.0<1>:f    -r15.0<2;1>:f     r66.0<2;1>:f      r4.0<1>:f
mul (8|M0)          r6.0<1>:f     r65.0<8;8,1>:f    r3.0<8;8,1>:f
mul (8|M0)          r13.0<1>:f    r73.0<8;8,1>:f    r4.0<8;8,1>:f
mad (8|M0)          r28.0<1>:f    r28.0<2;1>:f      r80.0<2;1>:f      r7.0<1>:f
mad (8|M0)          r18.0<1>:f    r18.0<2;1>:f      r74.0<2;1>:f      r7.0<1>:f
mad (8|M0)          r16.0<1>:f    r16.0<2;1>:f      r68.0<2;1>:f      r7.0<1>:f
mul (8|M0)          r30.0<1>:f    r77.0<8;8,1>:f    r2.0<8;8,1>:f
mul (8|M0)          r32.0<1>:f    r71.0<8;8,1>:f    r2.0<8;8,1>:f
add (8|M0)          r10.0<1>:f    r10.0<8;8,1>:f    r29.0<8;8,1>:f
mad (8|M0)          r27.0<1>:f    -r27.0<2;1>:f     r76.0<2;1>:f      r2.0<1>:f
mul (8|M0)          r7.0<1>:f     r65.0<8;8,1>:f    r2.0<8;8,1>:f
add (8|M0)          r12.0<1>:f    r12.0<8;8,1>:f    r19.0<8;8,1>:f
mad (8|M0)          r31.0<1>:f    -r31.0<2;1>:f     r70.0<2;1>:f      r2.0<1>:f
add (8|M0)          r15.0<1>:f    r15.0<8;8,1>:f    r17.0<8;8,1>:f
mad (8|M0)          r6.0<1>:f     -r6.0<2;1>:f      r64.0<2;1>:f      r2.0<1>:f
mad (8|M0)          r11.0<1>:f    r11.0<2;1>:f      r78.0<2;1>:f      r5.0<1>:f
mad (8|M0)          r14.0<1>:f    r14.0<2;1>:f      r66.0<2;1>:f      r5.0<1>:f
mad (8|M0)          r13.0<1>:f    r13.0<2;1>:f      r72.0<2;1>:f      r5.0<1>:f
mad (8|M0)          r30.0<1>:f    r30.0<2;1>:f      r76.0<2;1>:f      r3.0<1>:f
mad (8|M0)          r32.0<1>:f    r32.0<2;1>:f      r70.0<2;1>:f      r3.0<1>:f
mad (8|M0)          r7.0<1>:f     r7.0<2;1>:f       r64.0<2;1>:f      r3.0<1>:f
add (8|M0)          r5.0<1>:f     r27.0<8;8,1>:f    r10.0<8;8,1>:f
add (8|M0)          r3.0<1>:f     r31.0<8;8,1>:f    r12.0<8;8,1>:f
add (8|M0)          r6.0<1>:f     r6.0<8;8,1>:f     r15.0<8;8,1>:f
add (8|M0)          r11.0<1>:f    r11.0<8;8,1>:f    r28.0<8;8,1>:f
add (8|M0)          r13.0<1>:f    r13.0<8;8,1>:f    r18.0<8;8,1>:f
add (8|M0)          r14.0<1>:f    r14.0<8;8,1>:f    r16.0<8;8,1>:f
add (8|M0)          r57.0<1>:f    r5.0<8;8,1>:f     r103.0<8;8,1>:f
add (8|M0)          r5.0<1>:f     r3.0<8;8,1>:f     r111.0<8;8,1>:f
add (8|M0)          r3.0<1>:f     r113.0<8;8,1>:f   r6.0<8;8,1>:f
```

```
template <class float_type, int no_simd_memory>
template <parity_t parity, int no_rhs, int no_simd>
void action<float_type,no_simd_memory>::stencil( packed_lattice_vectors<float_type,no_simd_memory> const &vec_in ,
                                                 packed_lattice_vectors<float_type,no_simd_memory>       &vec_out   ) {

    constexpr int no_isf = no_implicit_sites<no_simd,no_simd_memory>();

    #pragma omp target teams distribute parallel for simd collapse(2) simdlen(no_isf)
    for ( int siteh_sf = 0; siteh_sf < _lattice.half_volume_sf(); ++siteh_sf ) {
        for ( int isf = 0; isf < no_isf; ++isf ) {

            color_vector_simd< float_type, no_simd > v[no_rhs];

            const site_shift<no_simd> site( siteh_sf, parity, isf );

            for ( int imu = 0; imu < 4; ++imu ) {

                const auto site_up = site.shift_eo<up>( imu, lattice );
                const auto site_dn = site.shift_eo<dn>( imu, lattice );

                su3_simd< float_type, no_simd > link = field.get_eo( site, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] += link * vec_in.get( site_up, irhs );
                }

                link = field.get_eo( site_dn, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] -= link * vec_in.get( site_dn, irhs );
                }
            }// end imu

            for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                vec_out.stream( 0.5*v[irhs], site, irhs );
            }
        }// end isf
    }// end siteh_sf
}
```

Load next vector iteration

- irhs loop is unrolled

```
add (8|M0)           r5.0<1>:f     r3.0<8;8,1>:f      r111.0<8;8,1>:f
add (8|M0)           r3.0<1>:f     r113.0<8;8,1>:f    r6.0<8;8,1>:f
or (8|M0)            r6.0<1>:d     r8.0<8;8,1>:d      1:w
add (8|M0)           r4.0<1>:f     r30.0<8;8,1>:f     r11.0<8;8,1>:f
add (8|M0)           r2.0<1>:f     r32.0<8;8,1>:f     r13.0<8;8,1>:f
add (8|M0)           r7.0<1>:f     r7.0<8;8,1>:f      r14.0<8;8,1>:f
mul (8|M0)           r6.0<1>:d     r6.0<8;8,1>:d      192:w
add (8|M0)           r50.0<1>:f    r4.0<8;8,1>:f      r104.0<8;8,1>:f
add (8|M0)           r4.0<1>:f     r2.0<8;8,1>:f      r112.0<8;8,1>:f
add (8|M0)           r2.0<1>:f     r114.0<8;8,1>:f    r7.0<8;8,1>:f
add (8|M0)           r6.0<1>:d     r102.0<8;8,1>:d    r6.0<8;8,1>:d
add (8|M0)           r7.0<1>:d     r6.0<8;8,1>:d      64:w
send (8|M0)          r12:f    r6     0xC              0x02206C02
add (8|M0)           r6.0<1>:d     r6.0<8;8,1>:d      128:w
send (8|M0)          r10:f    r7     0xC              0x02206C02
send (8|M0)          r6:f     r6     0xC              0x02206C02
mul (8|M0)           r30.0<1>:f    r75.0<8;8,1>:f     r13.0<8;8,1>:f
mul (8|M0)           r28.0<1>:f    r69.0<8;8,1>:f     r13.0<8;8,1>:f
mul (8|M0)           r29.0<1>:f    r75.0<8;8,1>:f     r12.0<8;8,1>:f
mul (8|M0)           r27.0<1>:f    r69.0<8;8,1>:f     r12.0<8;8,1>:f
mul (8|M0)           r33.0<1>:f    r81.0<8;8,1>:f     r13.0<8;8,1>:f
mul (8|M0)           r32.0<1>:f    r81.0<8;8,1>:f     r12.0<8;8,1>:f
mul (8|M0)           r14.0<1>:f    r79.0<8;8,1>:f     r11.0<8;8,1>:f
mul (8|M0)           r16.0<1>:f    r73.0<8;8,1>:f     r11.0<8;8,1>:f
mul (8|M0)           r19.0<1>:f    r67.0<8;8,1>:f     r11.0<8;8,1>:f
mul (8|M0)           r15.0<1>:f    r79.0<8;8,1>:f     r10.0<8;8,1>:f
mul (8|M0)           r17.0<1>:f    r73.0<8;8,1>:f     r10.0<8;8,1>:f
mul (8|M0)           r18.0<1>:f    r67.0<8;8,1>:f     r10.0<8;8,1>:f
mad (8|M0)           r30.0<1>:f    -r30.0<2;1>:f      r74.0<2;1>:f       r12.0<1>:f
mul (8|M0)           r35.0<1>:f    r71.0<8;8,1>:f     r7.0<8;8,1>:f
mad (8|M0)           r14.0<1>:f    -r14.0<2;1>:f      r78.0<2;1>:f       r10.0<1>:f
mad (8|M0)           r16.0<1>:f    -r16.0<2;1>:f      r72.0<2;1>:f       r10.0<1>:f
mad (8|M0)           r19.0<1>:f    -r19.0<2;1>:f      r66.0<2;1>:f       r10.0<1>:f
mad (8|M0)           r28.0<1>:f    -r28.0<2;1>:f      r68.0<2;1>:f       r12.0<1>:f
mul (8|M0)           r10.0<1>:f    r65.0<8;8,1>:f     r7.0<8;8,1>:f
mad (8|M0)           r15.0<1>:f    r15.0<2;1>:f       r78.0<2;1>:f       r11.0<1>:f
mad (8|M0)           r17.0<1>:f    r17.0<2;1>:f       r72.0<2;1>:f       r11.0<1>:f
mad (8|M0)           r18.0<1>:f    r18.0<2;1>:f       r66.0<2;1>:f       r11.0<1>:f
mul (8|M0)           r34.0<1>:f    r77.0<8;8,1>:f     r6.0<8;8,1>:f
mul (8|M0)           r36.0<1>:f    r71.0<8;8,1>:f     r6.0<8;8,1>:f
mul (8|M0)           r11.0<1>:f    r65.0<8;8,1>:f     r6.0<8;8,1>:f
mad (8|M0)           r35.0<1>:f    -r35.0<2;1>:f      r70.0<2;1>:f       r6.0<1>:f
add (8|M0)           r16.0<1>:f    r16.0<8;8,1>:f     r30.0<8;8,1>:f
add (8|M0)           r19.0<1>:f    r19.0<8;8,1>:f     r28.0<8;8,1>:f
mad (8|M0)           r10.0<1>:f    -r10.0<2;1>:f      r64.0<2;1>:f       r6.0<1>:f
mul (8|M0)           r31.0<1>:f    r77.0<8;8,1>:f     r7.0<8;8,1>:f
mad (8|M0)           r34.0<1>:f    r34.0<2;1>:f       r76.0<2;1>:f       r7.0<1>:f
```

```
template <class float_type, int no_simd_memory>
template <parity_t parity, int no_rhs, int no_simd>
void action<float_type,no_simd_memory>::stencil( packed_lattice_vectors<float_type,no_simd_memory> const &vec_in ,
                                                 packed_lattice_vectors<float_type,no_simd_memory>       &vec_out  ) {

    constexpr int no_isf = no_implicit_sites<no_simd,no_simd_memory>();

    #pragma omp target teams distribute parallel for simd collapse(2) simdlen(no_isf)
    for ( int siteh_sf = 0; siteh_sf < _lattice.half_volume_sf(); ++siteh_sf ) {
        for ( int isf = 0; isf < no_isf; ++isf ) {

            color_vector_simd< float_type, no_simd > v[no_rhs];

            const site_shift<no_simd> site( siteh_sf, parity, isf );

            for ( int imu = 0; imu < 4; ++imu ) {

                const auto site_up = site.shift_eo<up>( imu, lattice );
                const auto site_dn = site.shift_eo<dn>( imu, lattice );

                su3_simd< float_type, no_simd > link = field.get_eo( site, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] += link * vec_in.get( site_up, irhs );
                }

                link = field.get_eo( site_dn, imu );

                for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                    v[irhs] -= link * vec_in.get( site_dn, irhs );
                }
            }// end imu

            for ( int irhs = 0; irhs < no_rhs; ++irhs ) {

                vec_out.stream( 0.5*v[irhs], site, irhs );
            }
        }// end isf
    }// end siteh_sf
}
```

```
L12056:
      mul (8|M0)             r4.0<1>:d     r59.0<8;8,1>:d    48:w
      add (8|M0)             r2.0<1>:d     r62.0<8;8,1>:d    r9.0<0;1,0>:d
      mul (8|M0)             r10.0<1>:f    r104.0<8;8,1>:f   0.5:f
      mul (8|M0)             r9.0<1>:f     r103.0<8;8,1>:f   0.5:f
      mul (8|M0)             r7.0<1>:f     r111.0<8;8,1>:f   0.5:f
      mul (8|M0)             r8.0<1>:f     r112.0<8;8,1>:f   0.5:f
      mul (8|M0)             r5.0<1>:f     r113.0<8;8,1>:f   0.5:f
      shl (8|M0)             r4.0<1>:d     r4.0<8;8,1>:d     2:w
      mul (8|M0)             r6.0<1>:f     r114.0<8;8,1>:f   0.5:f
      mul (8|M0)             r11.0<1>:f    r84.0<8;8,1>:f    0.5:f
      mul (8|M0)             r12.0<1>:f    r85.0<8;8,1>:f    0.5:f
      add (8|M0)             r4.0<1>:d     r2.0<8;8,1>:d     r4.0<8;8,1>:d
      add (8|M0)             r3.0<1>:d     r4.0<8;8,1>:d     64:w
      sends (8|M0)           null:ud  r4      r9       0x8C            0x02026C03
      mul (8|M0)             r9.0<1>:f     r86.0<8;8,1>:f    0.5:f
      mul (8|M0)             r10.0<1>:f    r87.0<8;8,1>:f    0.5:f
      sends (8|M0)           null:ud  r3      r7       0x8C            0x02026C03
      add (8|M0)             r3.0<1>:d     r4.0<8;8,1>:d     128:w
      mul (8|M0)             r7.0<1>:f     r88.0<8;8,1>:f    0.5:f
      mul (8|M0)             r8.0<1>:f     r89.0<8;8,1>:f    0.5:f
      sends (8|M0)           null:ud  r3      r5       0x8C            0x02026C03
      or (8|M0)              r5.0<1>:q     r59.0<8;8,1>:d    1:w
      mov (8|M0)             r3.0<1>:ud    r5.0<2;1,0>:ud
      mul (8|M0)             r5.0<2>:ud    r3.0<8;8,1>:ud    0x30:uw
      mul (8|M0)             r3.0<1>:q     r3.0<8;8,1>:ud    0x30:ud
      mov (8|M0)             r4.0<1>:d     r3.1<2;1,0>:d
      mul (8|M0)             r3.0<1>:d     r5.1<2;1,0>:d     0x30:uw
      add (8|M0)             r5.1<2>:d     r4.0<8;8,1>:d     r3.0<8;8,1>:d
      shl (8|M0)             r4.0<1>:d     r5.0<2;1,0>:d     2:w
      or (8|M0)              r5.0<1>:q     r59.0<8;8,1>:d    2:w
      add (8|M0)             r4.0<1>:d     r2.0<8;8,1>:d     r4.0<8;8,1>:d
      add (8|M0)             r3.0<1>:d     r4.0<8;8,1>:d     64:w
      sends (8|M0)           null:ud  r4      r11      0x8C            0x02026C03
      mul (8|M0)             r11.0<1>:f    r90.0<8;8,1>:f    0.5:f
      mul (8|M0)             r12.0<1>:f    r91.0<8;8,1>:f    0.5:f
      sends (8|M0)           null:ud  r3      r9       0x8C            0x02026C03
      add (8|M0)             r3.0<1>:d     r4.0<8;8,1>:d     128:w
      mul (8|M0)             r9.0<1>:f     r92.0<8;8,1>:f    0.5:f
      mul (8|M0)             r10.0<1>:f    r93.0<8;8,1>:f    0.5:f
      sends (8|M0)           null:ud  r3      r7       0x8C            0x02026C03
      mov (8|M0)             r3.0<1>:ud    r5.0<2;1,0>:ud
      mul (8|M0)             r7.0<1>:f     r94.0<8;8,1>:f    0.5:f
      mul (8|M0)             r8.0<1>:f     r95.0<8;8,1>:f    0.5:f
      mul (8|M0)             r5.0<2>:ud    r3.0<8;8,1>:ud    0x30:uw
      mul (8|M0)             r3.0<1>:q     r3.0<8;8,1>:ud    0x30:ud
      mov (8|M0)             r4.0<1>:d     r3.1<2;1,0>:d
```

no_isf 3-dim complex vectors

- no_isf == 8
- Requires 3 SIMD16 stores

Summary

- The GPU programming style is not determined by hardware

- Intel GPU allows to program in the style that fits your application background. Both OpenMP and DPC++ support:
  - SPMD and SIMD programming model

- #pragma omp simd allows to use Intel GPU like a CPU

Notices & Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Refer to http://software.intel.com/en-us/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products

Configurations details: Performance shown for HotQCD stencil kernel is using 32^3x128 lattice volume on Intel® Xeon® Processor E3-1585 v5 with Intel® Iris® Pro Graphics P580 using Intel® oneAPI Compiler beta09.