Current State of Performance Analysis for Molecular Dynamics

Rafael Ravedutti L. Machado, Jan Eitzinger
Erlangen National High Performance Computing Center (NHR@FAU)
June 8, 2021

Overview

- Separate contributions:
  - Core execution
  - Bandwidth
  - Latency
- Correlate core execution measurements with IACA and OSACA outputs
- "Randomness" of MD accesses
- Understand the generated assembly
- Explore and compare performance for different strategies

Progress so far

- MD-Bench
  - AoS and SoA layouts
  - Different atom types
  - Stubbed force calculation within cache sizes
- Gather Benchmark
  - Written in pure assembly
  - AoS and SoA variants for MD
- OSACA and IACA Analysis
- Analysis of Assembly
  - Three variants
  - "Prefetching" instructions
  - Software vs Hardware Gathers

MD-Bench

- `https://github.com/RRZE-HPC/MD-Bench`
- Sequential re-implementation of miniMD in C
- Aim: as simple, clear and understandable as possible
- **Features:**
  - Standard test case (Cu FCC lattice)
  - Lennard Jones potential
  - Full neighbor lists
- **Runtime Parameters:**
  - Number of timesteps
  - Number of unit cells in x, y and y dimensions
- 4 atoms per unit cell with about 64 neighbors per atom
- Improvements:
  - Stubbed force-calculation to run within L1, L2 and L3 caches
  - Choose data layouts for atoms during compile-time (AoS vs SoA)
  - Allow multiple atom types for simulation.

Simulation Loop

```
for(int n = 0; n < param.ntimes; n++) {
  initialIntegrate(&param, &atom);
  if((n + 1) % param.every) {
    updatePbc(&atom, &param);
  } else {
    reneighbour(&param, &atom, &neighbor);
  }
  computeForce(&param, &atom, &neighbor);
  finalIntegrate(&param, &atom);
  if(!((n + 1) % param.nstat) && (n+1) < param.ntimes) {
    computeThermo(n + 1, &param, &atom);
  }
}
```

- Computing the forces is normally the most expensive part!
- Building the neighbor lists can also take a considerable fraction of the simulation time, specially when:
  - Force calculation time gets smaller (simpler potential to compute, optimizations)
  - Rebuilding frequency gets smaller!
- **For now, we just focus on the force computation!**

Force Computation Loop

```
for(int i = 0; i < Nlocal; i++) { // 131072 for standard case
  neighs = &neighbor->neighbors[i * neighbor->maxneighs];
  int numneighs = neighbor->numneigh[i];
  MD_FLOAT xtmp = atom_x(i);
  MD_FLOAT ytmp = atom_y(i);
  MD_FLOAT ztmp = atom_z(i);
  MD_FLOAT fix = 0;
  MD_FLOAT fiy = 0;
  MD_FLOAT fiz = 0;

  for(int k = 0; k < numneighs; k++) { // average of 64 neighbors per atom
    int j = neighs[k];
    MD_FLOAT delx = xtmp - atom_x(j);
    MD_FLOAT dely = ytmp - atom_y(j);
    MD_FLOAT delz = ztmp - atom_z(j);
    MD_FLOAT rsq = delx * delx + dely * dely + delz * delz;
    if(rsq < cutforcesq) {
      MD_FLOAT sr2 = 1.0 / rsq;
      MD_FLOAT sr6 = sr2 * sr2 * sr2 * sigma6;
      MD_FLOAT force = 48.0 * sr6 * (sr6 - 0.5) * sr2 * epsilon;
      fix += delx * force;
      fiy += dely * force;
      fiz += delz * force;
    }
  }

  fx[i] += fix, fy[i] += fiy, fz[i] += fiz;
}
```

Neighbors Loop

```
for(int k = 0; k < numneighs; k++) {
  int j = neighs[k];
  MD_FLOAT delx = xtmp - atom_x(j);
  MD_FLOAT dely = ytmp - atom_y(j);
  MD_FLOAT delz = ztmp - atom_z(j);
  MD_FLOAT rsq = delx * delx + dely * dely + delz * delz;
  if(rsq < cutforcesq) {
    MD_FLOAT sr2 = 1.0 / rsq;
    MD_FLOAT sr6 = sr2 * sr2 * sr2 * sigma6;
    MD_FLOAT force = 48.0 * sr6 * (sr6 - 0.5) * sr2 * epsilon;
    fix += delx * force;
    fiy += dely * force;
    fiz += delz * force;
  }
}
```

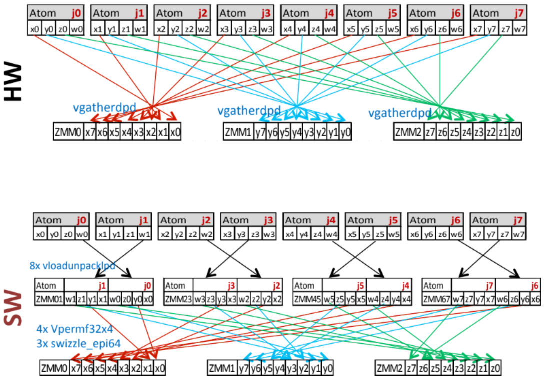
- To vectorize the code for `k`, gathering of the data is required
- Two possibilities: Hardware or Software gather

Hardware vs. Software gather

- **Hardware:** use asm `vgather` instructions
- **Software:** manually do gather operations with vloadunpack, permutations and swizzles
  - Take advantage of data locality on AoS
  - Probably better for out-of-order execution as well (several instructions vs. one gather)
- For MD-Bench, all generated kernels (both AoS and SoA) for AVX512 use hardware gathers

Hardware vs. Software gather

Impl: HW vs. SW Gather



HW G/S ICC codegen for KNC:

```
-mGLOB_default_function_attrs=
"gather_scatter_loop_unroll=7;
use_gather_scatter_hint=on"
..L769:
    vgatherdpd (%r15,%zmm18,8), %zmm19{%k3}
    jkzd    ..L768, %k3
    vgatherdpd (%r15,%zmm18,8), %zmm19{%k3}
    vgatherdpd (%r15,%zmm18,8), %zmm19{%k3}
    vgatherdpd (%r15,%zmm18,8), %zmm19{%k3}
    vgatherdpd (%r15,%zmm18,8), %zmm19{%k3}
    vgatherdpd (%r15,%zmm18,8), %zmm19{%k3}
    vgatherdpd (%r15,%zmm18,8), %zmm19{%k3}
    vgatherdpd (%r15,%zmm18,8), %zmm19{%k3}
    jknzd   ..L769, %k3
..L768:
```

SW G/S – taking advantage of spatial locality of elements of an Atom

Similar opportunity in "neighbor_build" function as in "Force_Compute"

Full "function" implemented in Intrinsic KNC, HSW

Iapichino, L.; Karakasis, V.; Karmakar, A.; Hammer, N.; Petkova, M.; Dolag, K.: Quickly selecting nearest particles in the Friends-of-Friends component of Gadget. Frankfurt 2015

Hardware vs. Software gather

miniMD v1.2 Full Neighbor List, 864K atoms, 100 TimeSteps, DP FP All meas on cthor-knc2. Time in sec (**lower is better**)			
Measurements		**Gains: C-Code/Intrinsic**	
Xeon 2S IVB	**Xeon Phi KNC**	**Xeon 2S IVB**	**Xeon Phi KNC**
Baseline C code: HW G/S; autovectorized by ICC			
2.1	2.6	1.00	1.00
Intrinsic SW G/S: Force & Neigh func for KNC, Force for IVB			
1.76	2.12	1.19	1.23

Iapichino, L.; Karakasis, V.; Karmakar, A.; Hammer, N.; Petkova, M.; Dolag, K.: Quickly selecting nearest particles in the Friends-of-Friends component of Gadget. Frankfurt 2015

Assembly Analysis

- Intel compiler (ICC) v19.0.5.281 Build 20190815
- Flags: `-S -masm=intel -D_GNU_SOURCE -DAOS -DPRECISION=2 -DALIGNMENT=64 -restrict -Ofast -xCORE-AVX 512 -qopt-zmm-usage=high -o ICC/force.s`
- For AVX512, all kernels with zmm registers and gather instructions
- Three kernel variants are generated by the compiler (Consider `rmng_neighs = numneighs - k`):
  - `rmng_neighs < 8`: last iteration, vectors are not fulfilled
  - `rmng_neighs in ]8, 1200]`: with mov+lea instructions (prefetching?), L1 case? 1200 * 3 * 8 = 28.8kB, L1 cache size is 32kB on Cascade Lake
  - `rmng_neighs >= 1200`: no mov+lea instructions
- We will focus on the `rmng_neighs in ]8, 1200]` variant

Assembly Analysis

```
vmovdqu    ymm3, YMMWORD PTR [r13+rbx*4]          # ymm3 <- neighs[k]
vpaddd     ymm4, ymm3, ymm3                       # ymm4 <- neighs[k] * 2
vpaddd     ymm3, ymm3, ymm4                       # ymm3 <- neighs[k] * 3
# -------------------- mov+lea instructions (prefetching?) -----------------------
mov        r10d, DWORD PTR [r13+rbx*4]            # r10d <- neighs[k]
mov        r9d, DWORD PTR [4+r13+rbx*4]           # r9d  <- neighs[k + 1]
mov        r8d, DWORD PTR [8+r13+rbx*4]           # r8d  <- neighs[k + 2]
mov        esi, DWORD PTR [12+r13+rbx*4]          # esi  <- neighs[k + 3]
lea        r10d, DWORD PTR [r10+r10*2]            # r10d <- neighs[k] * 3
mov        ecx, DWORD PTR [16+r13+rbx*4]          # ecx  <- neighs[k + 4]
lea        r9d, DWORD PTR [r9+r9*2]               # r9d  <- neighs[k + 1] * 3
mov        edx, DWORD PTR [20+r13+rbx*4]          # edx  <- neighs[k + 5]
lea        r8d, DWORD PTR [r8+r8*2]               # r8d  <- neighs[k + 2] * 3
mov        eax, DWORD PTR [24+r13+rbx*4]          # edx  <- neighs[k + 6]
lea        esi, DWORD PTR [rsi+rsi*2]             # esi  <- neighs[k + 3] * 3
mov        r15d, DWORD PTR [28+r13+rbx*4]         # edx  <- neighs[k + 7]
lea        ecx, DWORD PTR [rcx+rcx*2]             # ecx  <- neighs[k + 4] * 3
lea        edx, DWORD PTR [rdx+rdx*2]             # edx  <- neighs[k + 5] * 3
lea        eax, DWORD PTR [rax+rax*2]             # eax  <- neighs[k + 6] * 3
lea        r15d, DWORD PTR [r15+r15*2]            # r15d <- neighs[k + 7] * 3
# -------------------- end of mov+lea instructions ------------------------------
vpcmpeqb   k1, xmm0, xmm0                         # k1    <- [true for all elements]
vpcmpeqb   k2, xmm0, xmm0                         # k2    <- [true for all elements]
vpcmpeqb   k3, xmm0, xmm0                         # k3    <- [true for all elements]
vpxord     zmm4, zmm4, zmm4                       # zmm4  <- 0.0
vpxord     zmm17, zmm17, zmm17                    # zmm17 <- 0.0
vpxord     zmm18, zmm18, zmm18                    # zmm18 <- 0.0
vgatherdpd zmm4{k1}, QWORD PTR [16+rdi+ymm3*8]    # zmm4  <- atom->x[j * 3 + 2]
vgatherdpd zmm17{k2}, QWORD PTR [8+rdi+ymm3*8]    # zmm17 <- atom->x[j * 3 + 1]
vgatherdpd zmm18{k3}, QWORD PTR [rdi+ymm3*8]      # zmm18 <- atom->x[j * 3]
```

Assembly Analysis

When removing mov+lea instructions, performance is better on Cascade Lake:

```
With lea+mov:
TOTAL 9.30s FORCE 4.81s NEIGH 4.25s REST 0.24s

Without lea+mov:
TOTAL 8.95s FORCE 4.43s NEIGH 4.28s REST 0.24s
```

Stubbed Force Calculation

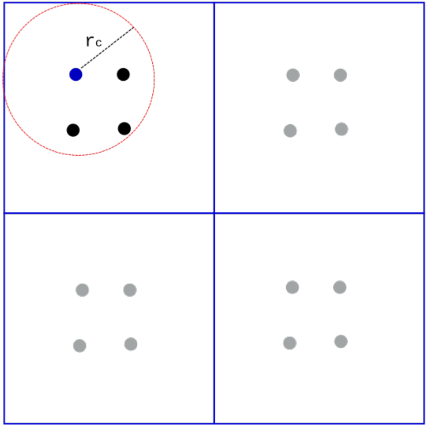
- Goal: core execution time contribution
- Execute most internal loop with data fitting into cache sizes
- Keep number of neighbors per atom fixed
- Compare cycles per iteration with OSACA and IACA outputs

Stubbed Force Calculation

Example:

atoms_per_unit_cell=4

nx=2, ny=2



Properties:

```
natoms = atoms_per_unit_cell * nx * ny
nneighs = atoms_per_unit_cell - 1
atoms_volume = natoms * 3 * sizeof(MD_FLOAT)
neighbors_volume = natoms * (nneighs + 2) * sizeof(int)
total_volume = natoms * 6 * sizeof(MD_FLOAT) +
neighbors_volume (considering writes)
```

OSACA Analysis

```
                                   Port pressure in cycles
 0  - 0DV  | 1    | 2   -  2D  | 3   -  3D  | 4    | 5    | 6    | 7    || CP  | LCD |
---------------------------------------------------------------------------------------
           |      |            |            |      |      |      |      ||     |     |  ..B1.25:
0.00       | 0.00 |            |            |      | 0.00 | 1.00 |      || 1.0 | 1.0 |  movq          %r8, %r13
           | 1.00 |            |            |      |      |      |      || 3.0 | 3.0 |  imulq         %rcx, %r13
           |      |            |            |      | 1.00 |      |      ||     |     |  vbroadcastsd  %xmm6, %zmm2
           |      |            |            |      | 1.00 |      |      ||     |     |  vbroadcastsd  %xmm7, %zmm1
           |      |            |            |      | 1.00 |      |      ||     |     |  vbroadcastsd  %xmm12, %zmm0
0.00       | 0.00 |            |            |      | 0.00 | 1.00 |      ||     |     |  movslq        %r12d, %rbx
0.00       | 0.00 |            |            |      | 0.00 | 1.00 |      || 1.0 | 1.0 |  addq          %r10, %r13
           |      | 0.00       | 0.00       | 1.00 |      |      | 1.00 ||     |     |  movq          %rax, -64(%rsp)
           |      | 0.00       | 0.00       | 1.00 |      |      | 1.00 ||     |     |  movq          %r8, -56(%rsp)
           |      | 0.00       | 0.00       | 1.00 |      |      | 1.00 ||     |     |  movq          %r10, -48(%rsp)
           |      | 0.00       | 0.00       | 1.00 |      |      | 1.00 ||     |     |  movq          %rsi, -40(%rsp)
           |      | 0.00       | 0.00       | 1.00 |      |      | 1.00 ||     |     |  movq          %rcx, -32(%rsp)
           |      | 0.00       | 0.00       | 1.00 |      |      | 1.00 ||     |     |  movq          %r9, -80(%rsp)
           |      | 0.00       | 0.00       | 1.00 |      |      | 1.00 ||     |     |  movq          %rdx, -72(%rsp)
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      || 4.0 |     |  vmovdqu       (%r13,%rbx,4), %ymm3
0.00       | 1.00 |            |            |      | 0.00 |      |      || 1.0 |     |  vpaddd        %ymm3, %ymm3, %ymm4
0.00       | 1.00 |            |            |      | 0.00 |      |      || 1.0 |     |  vpaddd        %ymm4, %ymm3, %ymm3
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          (%r13,%rbx,4), %r10d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          4(%r13,%rbx,4), %r9d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     | 4.0 |  movl          8(%r13,%rbx,4), %r8d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          12(%r13,%rbx,4), %esi
           | 1.00 |            |            |      | 0.00 |      |      ||     |     |  lea           (%r10,%r10,2), %r10d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          16(%r13,%rbx,4), %ecx
           | 1.00 |            |            |      | 0.00 |      |      ||     |     |  lea           (%r9,%r9,2), %r9d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     | 4.0 |  movl          8(%r13,%rbx,4), %r8d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          12(%r13,%rbx,4), %esi
           | 1.00 |            |            |      | 0.00 |      |      ||     |     |  lea           (%r10,%r10,2), %r10d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          16(%r13,%rbx,4), %ecx
           | 1.00 |            |            |      | 0.00 |      |      ||     |     |  lea           (%r9,%r9,2), %r9d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          20(%r13,%rbx,4), %edx
           | 1.00 |            |            |      | 0.00 |      |      ||     | 1.0 |  lea           (%r8,%r8,2), %r8d
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          24(%r13,%rbx,4), %eax
           | 1.00 |            |            |      | 0.00 |      |      ||     |     |  lea           (%rsi,%rsi,2), %esi
           |      | 0.50  0.50 | 0.50  0.50 |      |      |      |      ||     |     |  movl          28(%r13,%rbx,4), %r15d
```

OSACA Analysis

```
                                  Port pressure in cycles
 0   - 0DV  |  1   |  2    -  2D  |  3    -  3D  |  4   |  5   |  6   |  7   ||  CP  | LCD  |
-----------------------------------------------------------------------------------------
            | 1.00 |              |              |      | 0.00 |      |      ||      |      |   lea        (%rcx,%rcx,2), %ecx
            | 1.00 |              |              |      | 0.00 |      |      ||      |      |   lea        (%rdx,%rdx,2), %edx
            | 1.00 |              |              |      | 0.00 |      |      ||      |      |   lea        (%rax,%rax,2), %eax
            | 1.00 |              |              |      | 0.00 |      |      ||      |      |   lea        (%r15,%r15,2), %r15d
            |      |              |              |      |      |      |      ||      |      |   ..B1.29:
            |      |              |              |      |      |      |      ||      |      | X vpcmpeqb   %xmm0, %xmm0, %k1
            |      |              |              |      |      |      |      ||      |      | X vpcmpeqb   %xmm0, %xmm0, %k2
            |      |              |              |      |      |      |      ||      |      | X vpcmpeqb   %xmm0, %xmm0, %k3
0.50        |      |              |              |      | 0.50 |      |      ||      |      |   vpxord     %zmm4, %zmm4, %zmm4
0.50        |      |              |              |      | 0.50 |      |      ||      |      |   vpxord     %zmm17, %zmm17, %zmm17
0.50        |      |              |              |      | 0.50 |      |      ||      |      |   vpxord     %zmm18, %zmm18, %zmm18
1.50        | 0.17 | 4.00    0.50 | 4.00    0.50 |      | 0.50 | 0.83 |      ||      |      |   vgatherdpd 16(%rdi,%ymm3,8), %zmm4{%k1}
1.50        | 0.00 | 4.00    0.50 | 4.00    0.50 |      | 0.50 | 1.00 |      || 4.0  |      |   vgatherdpd 8(%rdi,%ymm3,8), %zmm17{%k2}
1.50        | 0.00 | 4.00    0.50 | 4.00    0.50 |      | 0.50 | 1.00 |      ||      |      |   vgatherdpd (%rdi,%ymm3,8), %zmm18{%k3}
            |      |              |              |      |      |      |      ||      |      |   ..B1.30:
0.00        | 0.00 |              |              |      | 0.00 | 1.00 |      ||      |      |   addl       $8, %r12d
0.00        | 0.00 |              |              |      | 0.00 | 1.00 |      ||      |      |   addq       $8, %rbx
0.50        |      |              |              |      | 0.50 |      |      ||      |      |   vsubpd     %zmm4, %zmm0, %zmm26
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vsubpd     %zmm17, %zmm1, %zmm24
0.50        |      |              |              |      | 0.50 |      |      ||      |      |   vsubpd     %zmm18, %zmm2, %zmm23
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vmulpd     %zmm24, %zmm24, %zmm3
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vfmadd231pd %zmm23, %zmm23, %zmm3
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vfmadd231pd %zmm26, %zmm26, %zmm3
2.50        |      |              |              |      | 0.50 |      |      || 8.0  |      |   vrcp14pd   %zmm3, %zmm22
            |      |              |              |      | 1.00 |      |      ||      |      |   vcmppd     $1, %zmm14, %zmm3, %k2
            |      |              |              |      | 1.00 |      |      ||      |      |   vfpclasspd $30, %zmm22, %k0
0.50        |      | 0.50    0.50 | 0.50    0.50 |      | 0.50 |      |      || 4.0  |      |   vfnmadd213pd .L_2il0floatpacket.9(%rip){1t
1.00        |      |              |              |      |      |      |      ||      |      |   knotw      %k0, %k1
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vmulpd     %zmm3, %zmm3, %zmm4
0.50        |      |              |              |      | 0.50 |      |      ||      |      |   vfmadd213pd %zmm22, %zmm3, %zmm22{%k1}
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vfmadd213pd %zmm22, %zmm4, %zmm22{%k1}
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vmulpd     %zmm13, %zmm22, %zmm17
0.50        |      |              |              |      | 0.50 |      |      ||      |      |   vmulpd     %zmm10, %zmm22, %zmm19
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vmulpd     %zmm17, %zmm22, %zmm20
0.50        |      |              |              |      | 0.50 |      |      || 4.0  |      |   vmulpd     %zmm20, %zmm22, %zmm18
```

IACA Analysis

```
                           Port pressure in cycles
 0    - 0DV |  1   |  2    - 2D  |  3    - 3D  |  4   |  5   |  6   |  7   ||  CP  | LCD  |
-----------------------------------------------------------------------------------------------
0.50        |      |             |             |      | 0.50 |      |      ||      |      |   vfmsub213pd %zmm5, %zmm20, %zmm22
0.50        |      |             |             |      | 0.50 |      |      ||  4.0 |      |   vmulpd     %zmm19, %zmm18, %zmm21
0.50        |      |             |             |      | 0.50 |      |      ||  4.0 |      |   vmulpd     %zmm22, %zmm21, %zmm25
1.00        |      |             |             |      | 0.00 |      |      ||      |      |   vfmadd231pd %zmm23, %zmm25, %zmm9{%k2}
1.00        |      |             |             |      | 0.00 |      |      ||      |      |   vfmadd231pd %zmm24, %zmm25, %zmm8{%k2}
1.00        |      |             |             |      | 0.00 |      |      ||  4.0 |      |   vfmadd231pd %zmm26, %zmm25, %zmm11{%k2}
0.00        | 0.00 |             |             |      | 0.00 | 1.00 |      ||      |      |   cmpl       %r14d, %r12d
            |      |             |             |      |      |      |      ||      |      | * jb         ..B1.26        # Prob 82%

21.0          11.2   17.0   6.50   17.0   6.50   7.00   17.0   8.83   7.00    75.0   10.0

Loop-Carried Dependencies Analysis Report
-----------------------------------------
 6.0 | lea        (%r10,%r10,2), %r10d                    | [269, 283, 287]
10.0 | lea        (%r8,%r8,2), %r8d                       | [263, 264, 269, 285, 291]
 9.0 | lea        (%rcx,%rcx,2), %ecx                     | [264, 269, 288, 295]
 1.0 | addl       $8, %r12d                               | [314]
 4.0 | vfmadd231pd %zmm26, %zmm25, %zmm11{%k2}            | [339]
 4.0 | vfmadd231pd %zmm24, %zmm25, %zmm8{%k2}             | [338]
 4.0 | vfmadd231pd %zmm23, %zmm25, %zmm9{%k2}             | [337]
```

IACA Analysis

```
Throughput Analysis Report
--------------------------
Block Throughput: 36.70 Cycles       Throughput Bottleneck: Backend
Loop Count:  23
Port Binding In Cycles Per Iteration:
------------------------------------------------------------------------------------------
|  Port  |   0   -  DV   |   1   |   2   -   D   |   3   -   D   |   4   |   5   |   6   |   7   |
------------------------------------------------------------------------------------------
| Cycles | 17.5     0.0  | 11.0  | 20.5     17.0 | 20.5     17.0 |  7.0  | 20.5  |  7.0  |  0.0  |
------------------------------------------------------------------------------------------
...
| Num Of   |                    Ports pressure in cycles                         |      |
|  Uops    |  0  - DV    |  1   |  2  -  D    |  3  -  D    |  4   |  5   |  6   |  7   |
-----------------------------------------------------------------------------------------
|   1*     |             |      |             |             |      |      |      |      | mov r13, r8
|   1      |             | 1.0  |             |             |      |      |      |      | imul r13, rcx
|   1      |             |      |             |             |      | 1.0  |      |      | vbroadcastsd zmm2, xmm6
|   1      |             |      |             |             |      | 1.0  |      |      | vbroadcastsd zmm1, xmm7
|   1      |             |      |             |             |      | 1.0  |      |      | vbroadcastsd zmm0, xmm12
|   1      |             |      |             |             |      |      | 1.0  |      | movsxd rbx, r12d
|   1      |             |      |             |             |      |      | 1.0  |      | add r13, r10
|   2^     |             |      | 0.5         | 0.5         | 1.0  |      |      |      | mov qword ptr [rsp-0x40], rax
|   2^     |             |      | 0.5         | 0.5         | 1.0  |      |      |      | mov qword ptr [rsp-0x38], r8
|   2^     |             |      | 0.5         | 0.5         | 1.0  |      |      |      | mov qword ptr [rsp-0x30], r10
|   2^     |             |      | 0.5         | 0.5         | 1.0  |      |      |      | mov qword ptr [rsp-0x28], rsi
|   2^     |             |      | 0.5         | 0.5         | 1.0  |      |      |      | mov qword ptr [rsp-0x20], rcx
|   2^     |             |      | 0.5         | 0.5         | 1.0  |      |      |      | mov qword ptr [rsp-0x50], r9
|   2^     |             |      | 0.5         | 0.5         | 1.0  |      |      |      | mov qword ptr [rsp-0x48], rdx
|   1      |             |      | 0.5    0.5  | 0.5    0.5  |      |      |      |      | vmovdqu ymm3, ymmword ptr [r13+rbx*4]
|   1      |             | 1.0  |             |             |      |      |      |      | vpaddd ymm4, ymm3, ymm3
|   1      |             | 1.0  |             |             |      |      |      |      | vpaddd ymm3, ymm3, ymm4
|   1      |             |      | 0.5    0.5  | 0.5    0.5  |      |      |      |      | mov r10d, dword ptr [r13+rbx*4]
|   1      |             |      | 0.5    0.5  | 0.5    0.5  |      |      |      |      | mov r9d, dword ptr [r13+rbx*4+0x4]
|   1      |             |      | 0.5    0.5  | 0.5    0.5  |      |      |      |      | mov r8d, dword ptr [r13+rbx*4+0x8]
|   1      |             |      | 0.5    0.5  | 0.5    0.5  |      |      |      |      | mov esi, dword ptr [r13+rbx*4+0xc]
|   1      |             | 1.0  |             |             |      |      |      |      | lea r10d, ptr [r10+r10*2]
|   1      |             |      | 0.5    0.5  | 0.5    0.5  |      |      |      |      | mov ecx, dword ptr [r13+rbx*4+0x10]
|   1      |             | 1.0  |             |             |      |      |      |      | lea r9d, ptr [r9+r9*2]
```

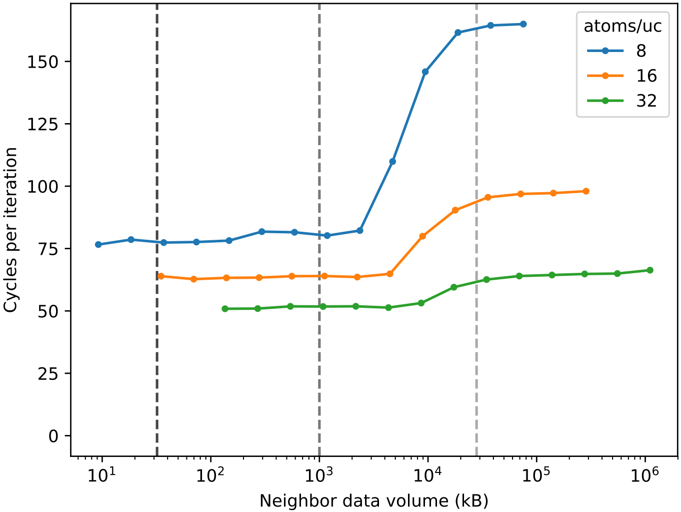
IACA Analysis

```
|  1    |         |      | 0.5    0.5 | 0.5    0.5 |     |     |     |     | mov edx, dword ptr [r13+rbx*4+0x14]
|  1    |         | 1.0  |            |            |     |     |     |     | lea r8d, ptr [r8+r8*2]
|  1    |         |      | 0.5    0.5 | 0.5    0.5 |     |     |     |     | mov eax, dword ptr [r13+rbx*4+0x18]
|  1    |         | 1.0  |            |            |     |     |     |     | lea esi, ptr [rsi+rsi*2]
|  1    |         |      | 0.5    0.5 | 0.5    0.5 |     |     |     |     | mov r15d, dword ptr [r13+rbx*4+0x1c]
|  1    |         | 1.0  |            |            |     |     |     |     | lea ecx, ptr [rcx+rcx*2]
|  1    |         | 1.0  |            |            |     |     |     |     | lea edx, ptr [rdx+rdx*2]
|  1    |         | 1.0  |            |            |     |     |     |     | lea eax, ptr [rax+rax*2]
|  1    |         | 1.0  |            |            |     |     |     |     | lea r15d, ptr [r15+r15*2]
|  1    |         |      |            |            |     | 1.0 |     |     | vpcmpeqb k1, xmm0, xmm0
|  1    |         |      |            |            |     | 1.0 |     |     | vpcmpeqb k2, xmm0, xmm0
|  1    |         |      |            |            |     | 1.0 |     |     | vpcmpeqb k3, xmm0, xmm0
|  1*   |         |      |            |            |     |     |     |     | vpxord zmm4, zmm4, zmm4
|  1*   |         |      |            |            |     |     |     |     | vpxord zmm17, zmm17, zmm17
|  1*   |         |      |            |            |     |     |     |     | vpxord zmm18, zmm18, zmm18
|  5^   | 2.0     |      | 4.0    4.0 | 4.0    4.0 |     |     | 1.0 |     | vgatherdpd zmm4, k1, zmmword ptr [rdi+ymm3*8+0
|  5^   | 1.5     |      | 4.0    4.0 | 4.0    4.0 |     | 0.5 | 1.0 |     | vgatherdpd zmm17, k2, zmmword ptr [rdi+ymm3*8+
|  5^   | 1.0     |      | 4.0    4.0 | 4.0    4.0 |     | 1.0 | 1.0 |     | vgatherdpd zmm18, k3, zmmword ptr [rdi+ymm3*8]
|  1    |         |      |            |            |     |     | 1.0 |     | add r12d, 0x8
|  1    |         |      |            |            |     |     | 1.0 |     | add rbx, 0x8
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vsubpd zmm26, zmm0, zmm4
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vsubpd zmm24, zmm1, zmm17
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vsubpd zmm23, zmm2, zmm18
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vmulpd zmm3, zmm24, zmm24
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vfmadd231pd zmm3, zmm23, zmm23
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vfmadd231pd zmm3, zmm26, zmm26
|  3    | 2.0     |      |            |            |     | 1.0 |     |     | vrcp14pd zmm22, zmm3
|  1    |         |      |            |            |     | 1.0 |     |     | vcmppd k2, zmm3, zmm14, 0x1
|  1    |         |      |            |            |     | 1.0 |     |     | vfpclasspd k0, zmm22, 0x1e
|  2^   | 1.0     |      | 0.5    0.5 | 0.5    0.5 |     |     |     |     | vfnmadd213pd zmm3, zmm22, qword ptr [rip]{1to8
|  1    | 1.0     |      |            |            |     |     |     |     | knotw k1, k0
|  1    |         |      |            |            |     | 1.0 |     |     | vmulpd zmm4, zmm3, zmm3
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vfmadd213pd zmm22{k1}, zmm3, zmm22
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vfmadd213pd zmm22{k1}, zmm4, zmm22
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vmulpd zmm17, zmm22, zmm13
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vmulpd zmm19, zmm22, zmm10
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vmulpd zmm20, zmm22, zmm17
|  1    | 0.5     |      |            |            |     | 0.5 |     |     | vmulpd zmm18, zmm22, zmm20
```

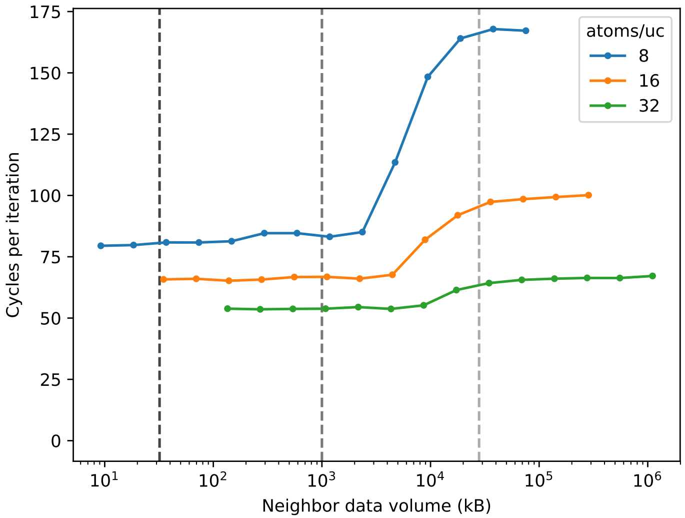
IACA Analysis

```
|    1     | 0.5         |       |             |             |       | 0.5   |       |       | vfmsub213pd zmm22, zmm20, zmm5
|    1     | 0.5         |       |             |             |       | 0.5   |       |       | vmulpd zmm21, zmm18, zmm19
|    1     | 0.5         |       |             |             |       | 0.5   |       |       | vmulpd zmm25, zmm21, zmm22
|    1     | 0.5         |       |             |             |       | 0.5   |       |       | vfmadd231pd zmm9{k2}, zmm25, zmm23
|    1     | 0.5         |       |             |             |       | 0.5   |       |       | vfmadd231pd zmm8{k2}, zmm25, zmm24
|    1     | 0.5         |       |             |             |       | 0.5   |       |       | vfmadd231pd zmm11{k2}, zmm25, zmm26
|    1*    |             |       |             |             |       |       |       |       | cmp r12d, r14d
|    0*F   |             |       |             |             |       |       |       |       | jb 0xffffffffffffffed3
Total Num Of Uops: 91
Analysis Notes:
Backend allocation was stalled due to unavailable allocation resources.
There were bubbles in the frontend.
```

Cycles per cache line - AVX512 MD-stub AoS force



Cycles per cache line - AVX512 MD-stub SoA force



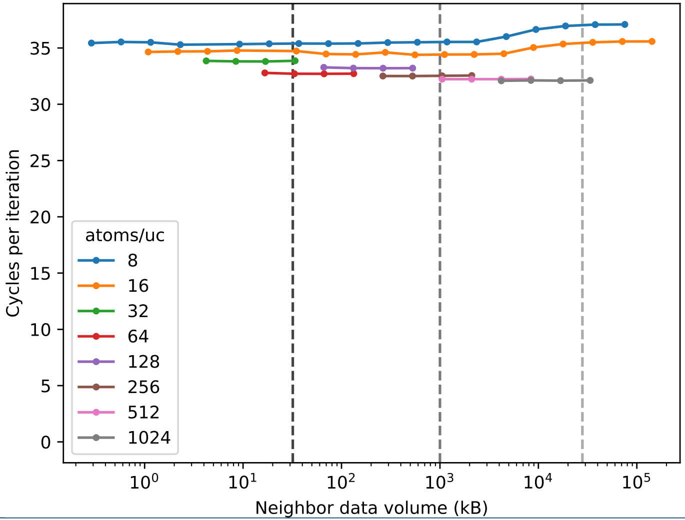
- **Issue: internal loop is not executed enough times for each atom!**
- Overhead to fill pipeline with instructions has a considerable effect on the cycles per iteration!
- Cycles measured for some cases are considerably higher than OSACA/IACA predictions!
- **Solution: repeat the most internal loop**

Neighbors Loop

```
...
for(int n = 0; n < ntimes; n++) {
  for(int k = 0; k < numneighs; k++) {
    int j = neighs[k];
    MD_FLOAT delx = xtmp - atom_x(j);
    MD_FLOAT dely = ytmp - atom_y(j);
    MD_FLOAT delz = ztmp - atom_z(j);
    MD_FLOAT rsq = delx * delx + dely * dely + delz * delz;

    if(rsq < cutforcesq) {
      MD_FLOAT sr2 = 1.0 / rsq;
      MD_FLOAT sr6 = sr2 * sr2 * sr2 * sigma6;
      MD_FLOAT force = 48.0 * sr6 * (sr6 - 0.5) * sr2 * epsilon;
      fix += delx * force;
      fiy += dely * force;
      fiz += delz * force;
    }
  }
}
...
```

Cycles per cache line - AVX512 MD-stub SoA force (100 repeats)



Stubbed Force Calculation

- Require internal loop repetition to provide more accurate results
- Increasing the atoms per unit cells (# neighbors) reduces cycles per iteration
  - Until which point?
- **Is there a better way to measure core execution contribution?**
- What about data traffic contributions?

gather-bench

- Benchmark for gathering data into registers
- Measure impact of gathers with several data volumes with respect to cache lines touched
- Input: stride of elements to be gathered
- Support for AVX2 and AVX512 (currently)
- Based on previous benchmark, some changes:
  - Gather kernels explicitly written in Assembly
  - MD variants: AoS and SoA
  - Tests for correctness
- Next steps to implement: SW gathers, evaluate with lea+mov instructions

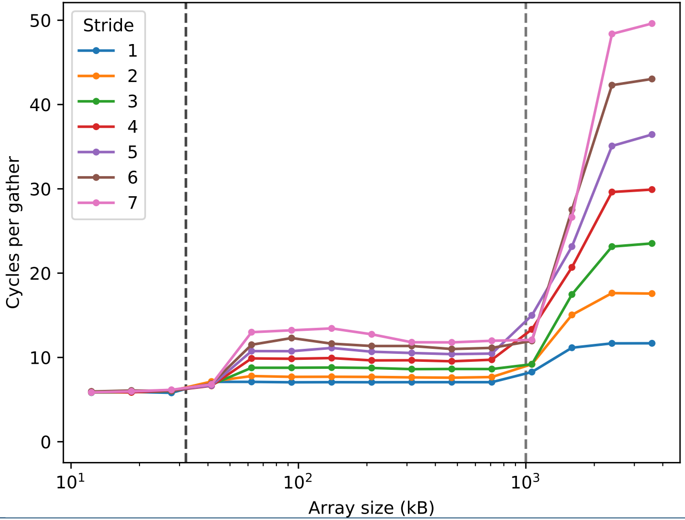
AVX512 Gather (simple array)

```
...
xor    rax, rax
.align 16
1:
vpcmpeqb k1, xmm0, xmm0
vpcmpeqb k2, xmm0, xmm0
vpcmpeqb k3, xmm0, xmm0
vpcmpeqb k4, xmm0, xmm0
vmovdqu ymm0, [rsi + rax*4]
vmovdqu ymm1, [rsi + rax*4+32]
vmovdqu ymm2, [rsi + rax*4+64]
vmovdqu ymm3, [rsi + rax*4+96]
vpxord zmm4, zmm4, zmm4
vpxord zmm5, zmm5, zmm5
vpxord zmm6, zmm6, zmm6
vpxord zmm7, zmm7, zmm7
vgatherdpd zmm4{k1}, [rdi + ymm0*8]
vgatherdpd zmm5{k2}, [rdi + ymm1*8]
vgatherdpd zmm6{k3}, [rdi + ymm2*8]
vgatherdpd zmm7{k4}, [rdi + ymm3*8]

# Required for test
#vmovapd [rcx + rax*8],     zmm4
#vmovapd [rcx + rax*8+64],  zmm5
#vmovapd [rcx + rax*8+128], zmm6
#vmovapd [rcx + rax*8+192], zmm7

addq rax, 32
cmpq rax, rdx
jl 1b
...
```

Cycles per cache line - AVX512 Gather with simple array on Cascade Lake



AVX512 Gather (AoS)

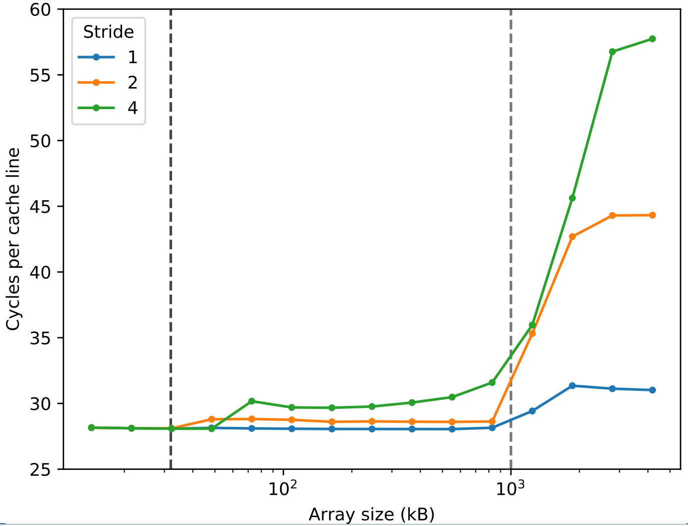
```
...
xor    rax, rax
.align 16
1:

vmovdqu ymm3, YMMWORD PTR [rsi + rax * 4]
vpaddd ymm4, ymm3, ymm3
vpaddd ymm3, ymm3, ymm4
vpcmpeqb k1, xmm5, xmm5
vpcmpeqb k2, xmm5, xmm5
vpcmpeqb k3, xmm5, xmm5
vpxord zmm0, zmm0, zmm0
vpxord zmm1, zmm1, zmm1
vpxord zmm2, zmm2, zmm2
vgatherdpd zmm0{k1}, [rdi + ymm3*8]
vgatherdpd zmm1{k2}, [8 + rdi + ymm3*8]
vgatherdpd zmm2{k3}, [16 + rdi + ymm3*8]

# Required for test
#vmovupd [rcx + rax*8], zmm0
#lea rbx, [rcx + rdx*8]
#vmovupd [rbx + rax*8], zmm1
#lea r9, [rbx + rdx*8]
#vmovupd [r9 + rax*8], zmm2

addq rax, 8
cmpq rax, rdx
jl 1b
...
```

Cycles per cache line - AVX512 MD-Gather with AoS layout on Skylake



Conclusion

- Measure contributions:
  - Core execution (stubbed force)
  - Latency and Bandwidth (gather-bench)
- Correlate "randomness" of memory access in MD to memory contributions
  - Irregular accesses on gather
  - Cache simulator
  - Reuse distance
- Investigate performance for different strategies/choices:
  - AoS vs SoA
  - Compiler
  - HW vs. SW gather
  - Different potentials (EAM)