MuCoSim: Analysis of MiniMD

Jinwen Pan

Advisors: Jan Eitzinger and Dominik Ernst

July 18, 2022

Agenda

01	Project Description	Talk I
02	Scalability Comparison of OpenMP and MPI	
03	Whole Application Measurements	Talk II
04	Runtime Profile	
05	Hot Spots Measurements	

Project Description

Molecular Dynamics Simulation Mini-App

MiniMD is a molecular dynamics proxy application simulating a multiparticle system based on Newtonian Mechanics.

- Force Calculation: Lennard-Jones Potential and Embedded Atom Model Potential
- Cutoff Distance: Half/Full Neighbor Lists Approach and Cell Lists Approach

Testsystem Fritz

- CPU Name: Intel(R) Xeon(R) Platinum 8360Y CPU @ 2.40GHz
- CPU Type: Intel Icelake SP Processor
- 944 compute nodes with 2 sockets per node and 36 cores per socket
- 4 NUMA domains (2+2) per node: 18 cores per domain (Cluster On Die / SNC active)
- One thread per core: SMT is not active
- Each core has an individual L1 cache (48 KB) and L2 cache (1.25 MB)
- Each socket has an individual L3 cache (54 MB)

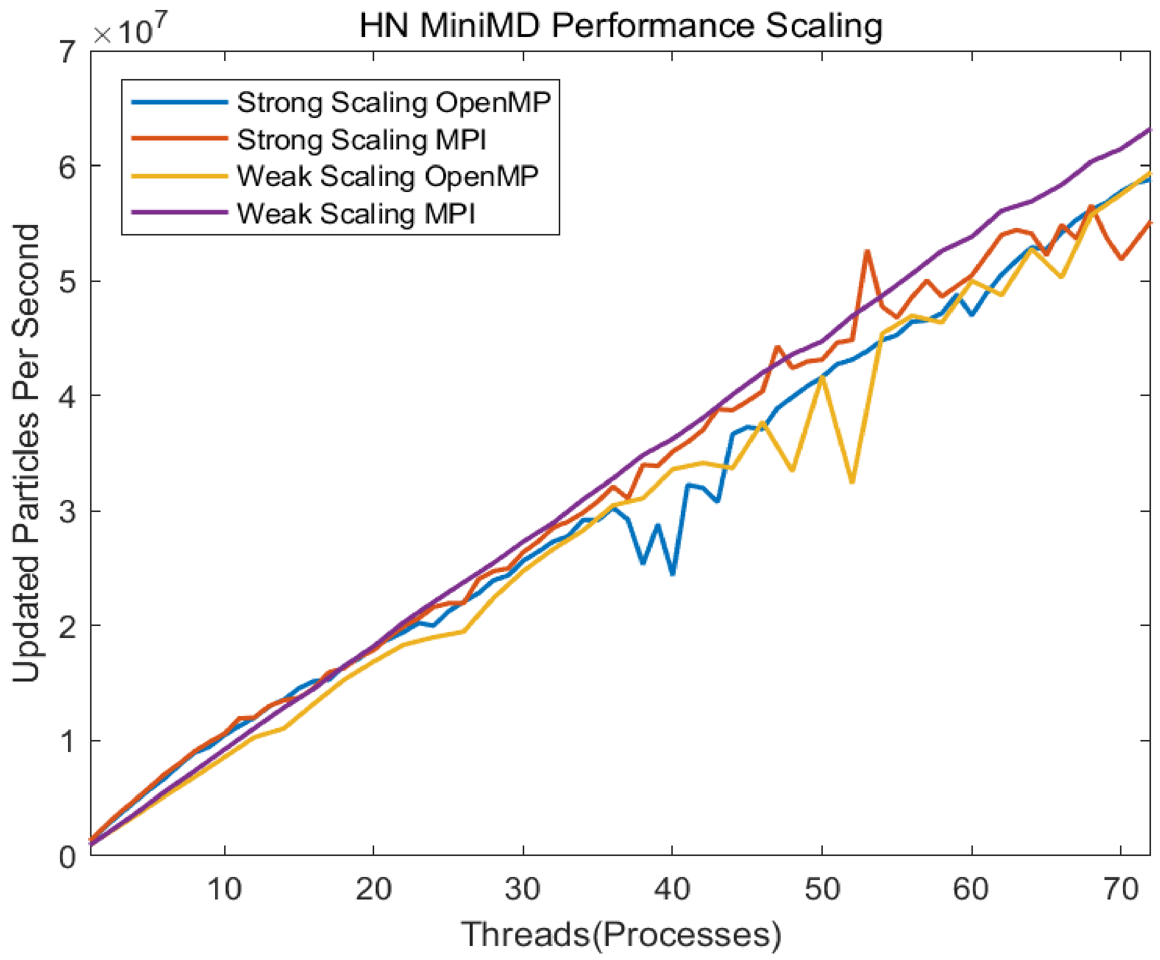
Modules Loaded

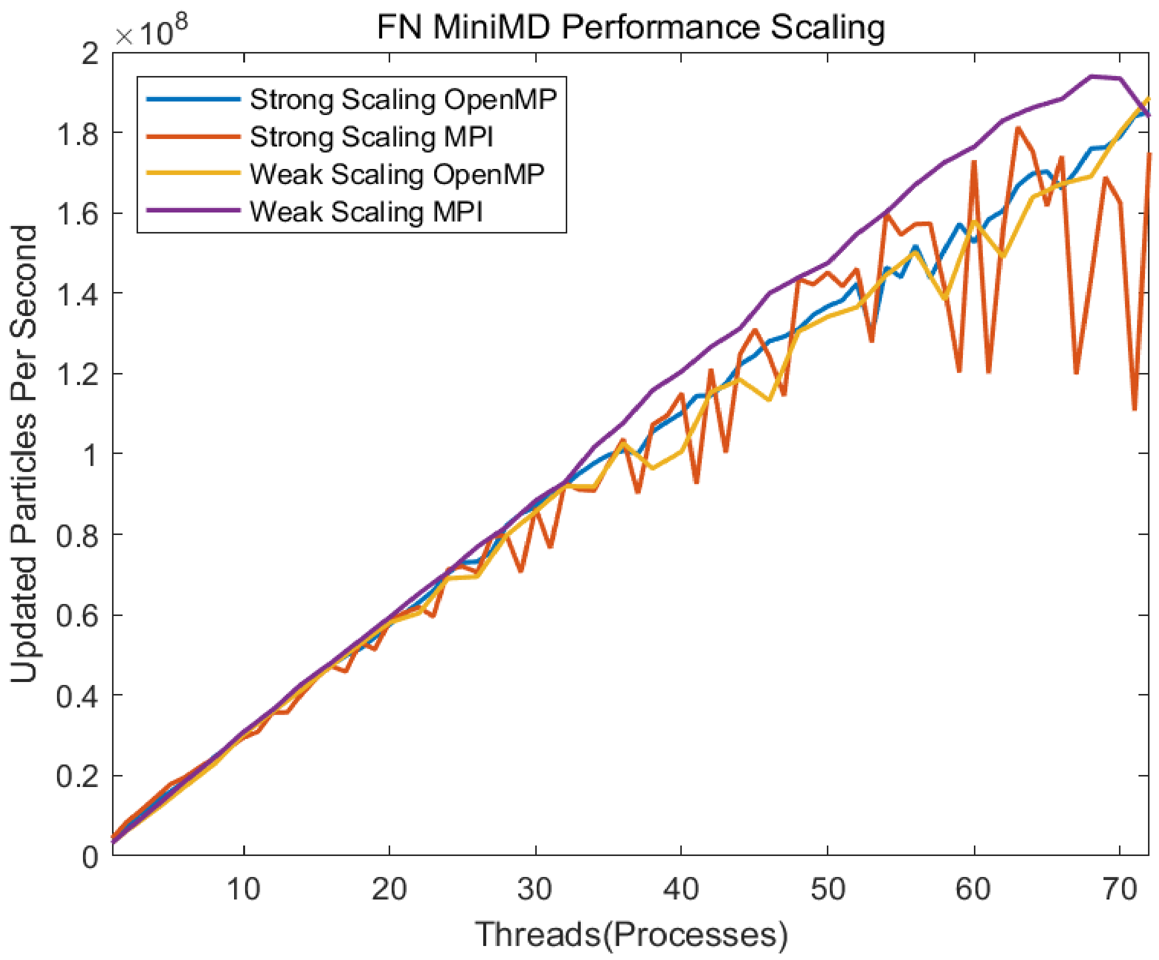
- Intel/2021.4.0
- Intelmpi/2021.4.0
- Likwid/5.2.1

Scalability Comparison of OpenMP and MPI

- MiniMD (HN/FN)
  - OpenMP
    - One Socket (1-36 threads)
      - Strong Scaling
      - Weak Scaling
    - One Node (1-72 threads)
      - Strong Scaling
      - Weak Scaling
  - MPI
    - One Node (1-72 processes)
      - Strong Scaling
      - Weak Scaling
    - Multi-Node (1-20 nodes)
      - Strong Scaling
      - Weak Scaling

Scalability Comparison of OpenMP and MPI





Whole Application Measurements

Single Instruction Multiple Data (SIMD) Vectorization

SIMD is a feature of a single core, but applies a same instruction to multiple operands in parallel with wide registers.

- Scalar: register width = 64 bits (32 bits)

  -> 1 double (1 single) precision FP operand
- SSE: register width = 128 bits

  -> 2 double (4 single) precision FP operands
- AVX: register width = 256 bits

  -> 4 double (8 single) precision FP operands
- AVX512: register width = 512 bits

  -> 8 double (16 single) precision FP operands

Compilation Configuration

```
#OPTS   = -Ofast -no-vec
#OPTS   = -Ofast -xSSE4.2 -DUSE_SIMD
#OPTS   = -Ofast -xCORE-AVX2 -DUSE_SIMD
#OPTS   = -Ofast -xCORE-AVX512 -qopt-zmm-usage=high -DUSE_SIMD
```

FLOPS_DP Group

```
$ srun --constraint=hwperf likwid-perfctr -g FLOPS_DP -C S0:1 ../miniMD-AVX --half_neigh 1
```

- INSTR_RETIRED_ANY
- FP_ARITH_INST_RETIRED_SCALAR_DOUBLE……………….(1)
- FP_ARITH_INST_RETIRED_128B_PACKED_DOUBLE………..(2)
- FP_ARITH_INST_RETIRED_256B_PACKED_DOUBLE………..(3)
- FP_ARITH_INST_RETIRED_512B_PACKED_DOUBLE………..(4)
- CPI (Cycle Per Instruction)
- Vectorization Ratio = [(2) + (3) + (4)] / [(1) + (2) + (3) + (4)]

Whole Application Measurements

	HN Novec	HN SSE	HN AVX	HN AVX512	FN Novec	FN SSE	FN AVX	FN AVX512
Total Instructions	1.04419E+11	1.25319E+11	1.10110E+11	4.16818E+10	1.68247E+11	1.70410E+11	9.70627E+10	6.55391E+10
Arithmetic Instructions	4.27175E+10	2.76718E+10	2.83674E+10	1.55086E+10	7.47834E+10	4.82002E+10	3.29768E+10	2.64749E+10
Arithmetic Percentage [%]	40.91	22.08	25.76	37.21	44.45	28.28	33.97	40.40
Runtime (RDTSC) [s]	19.64	20.93	20.30	11.47	33.44	23.67	15.84	13.07
CPI	0.5787	0.5157	0.5665	0.7620	0.6474	0.4358	0.4871	0.5708
Clock Frequency [MHz]	3235.4168	3256.9114	3238.7995	3018.8747	3354.7131	3285.0358	3177.8148	3079.8612
Vectorization Ratio [%]	0	71.70	72.39	57.46	0	73.30	58.62	52.73
Instructions vs. HN [%]	-	-	-	-	161.13	135.98	88.15	157.24
Instructions vs. Novec [%]	-	120.02	105.45	39.92	-	101.29	57.69	38.95
Arithmetic Instructions vs. Novec [%]	-	64.78	66.41	36.31	-	64.45	44.10	35.40

Runtime Profile

HN MiniMD: Function Name	Time(%)	Self Seconds	Calls
ForceLJ::compute	77.47	14.40	102
Neighbor::build	17.21	3.20	6
Integrate::run	1.72	0.32	1
Atom::pack_comm	0.48	0.09	570
Atom::unpack_reverse	0.38	0.07	612

FN MiniMD: Function Name	Time(%)	Self Seconds	Calls
ForceLJ::compute	77.43	25.28	102
Neighbor::build	19.57	6.39	6
Integrate::run	1.26	0.41	1
create_atoms	0.25	0.08	1
Atom::pack_comm	0.18	0.06	570

Input Parameters

– Sequential and no vectorization

– Lennard-Jones potential

– Cutoff distance: 2.5

– Neighbor skin width: 0.3

– Number of atoms: 48 * 48 * 48 * 4 = 442368

– Timesteps: 100

– Timestep size: 0.005

Hot Spots Measurements

```
LIKWID_MARKER_START ("HalfNeigh");
for(int i = 0; i < nlocal; i++) {
        neighs = &neighbor.neighbors[i * neighbor.maxneighs];
        const int numneighs = neighbor.numneigh[i];
        const MMD_float xtmp = x[i * PAD + 0];
        const MMD_float ytmp = x[i * PAD + 1];
        const MMD_float ztmp = x[i * PAD + 2];

        MMD_float fix = 0.0;
        MMD_float fiy = 0.0;
        MMD_float fiz = 0.0;

#ifdef USE_SIMD
#pragma simd reduction (+: fix,fiy,fiz)
#endif
        for(int k = 0; k < numneighs; k++) {
          const int j = neighs[k];
          const MMD_float delx = xtmp - x[j * PAD + 0];
          const MMD_float dely = ytmp - x[j * PAD + 1];
          const MMD_float delz = ztmp - x[j * PAD + 2];
          const MMD_float rsq = delx * delx + dely * dely + delz * delz;
```

```
 if(rsq < cutforcesq) {
          const MMD_float sr2 = 1.0 / rsq;
          const MMD_float sr6 = sr2 * sr2 * sr2 * sigma6;
          const MMD_float force = 48.0 * sr6 * (sr6 - 0.5) * sr2 * epsilon;

          fix += delx * force;
          fiy += dely * force;
          fiz += delz * force;

          if(j < nlocal) {
            f[j * PAD + 0] -= delx * force;
            f[j * PAD + 1] -= dely * force;
            f[j * PAD + 2] -= delz * force;
          }
        }
      }
      f[i * PAD + 0] += fix;
      f[i * PAD + 1] += fiy;
      f[i * PAD + 2] += fiz;
    }
LIKWID_MARKER_STOP("HalfNeigh");
```

Hot Spots Measurements

```
LIKWID_MARKER_START ("FullNeigh");
for(int i = 0; i < nlocal; i++) {
    neighs = &neighbor.neighbors[i * neighbor.maxneighs];
    const int numneighs = neighbor.numneigh[i];
    const MMD_float xtmp = x[i * PAD + 0];
    const MMD_float ytmp = x[i * PAD + 1];
    const MMD_float ztmp = x[i * PAD + 2];

    MMD_float fix = 0;
    MMD_float fiy = 0;
    MMD_float fiz = 0;

    for(int k = 0; k < numneighs; k++) {
      const int j = neighs[k];
      const MMD_float delx = xtmp - x[j * PAD + 0];
      const MMD_float dely = ytmp - x[j * PAD + 1];
      const MMD_float delz = ztmp - x[j * PAD + 2];
      const MMD_float rsq = delx * delx + dely * dely + delz * delz;
      if(rsq < cutforcesq) {
            const MMD_float sr2 = 1.0 / rsq;
            const MMD_float sr6 = sr2 * sr2 * sr2 * sigma6;
            const MMD_float force = 48.0 * sr6 * (sr6 - 0.5) * sr2 * epsilon;

            fix += delx * force;
            fiy += dely * force;
            fiz += delz * force;
        }
    }

    f[i * PAD + 0] += fix;
    f[i * PAD + 1] += fiy;
    f[i * PAD + 2] += fiz;
  }
LIKWID_MARKER_STOP("FullNeigh");
```

Hot Spots Measurements

	HN Novec	HN SSE	HN AVX	HN AVX512	FN Novec	FN SSE	FN AVX	FN AVX512
Total Instructions	7.59497E+10	9.70560E+10	8.58887E+10	1.75340E+10	1.16419E+11	1.18790E+11	5.31449E+10	2.16958E+10
Arithmetic Instructions	3.59216E+10	2.10575E+10	2.18203E+10	8.99739E+9	6.21597E+10	3.57582E+10	2.06019E+10	1.41359E+10
Arithmetic Percentage [%]	47.30	21.70	25.41	51.31	53.39	30.10	38.77	65.16
Runtime (RDTSC) [s]	14.71	15.78	16.12	6.32	25.10	15.39	7.82	4.88
CPI	0.6242	0.5270	0.6083	1.0790	0.7230	0.4263	0.4710	0.6952
Clock Frequency [MHz]	3234.4792	3252.9333	3252.1095	3006.7474	3362.6897	3300.8917	3213.3516	3104.8627
Vectorization Ratio [%]	0	93.50	93.72	98.49	0	98.38	93.41	98.42
Instructions vs. HN [%]	-	-	-	-	153.28	122.39	61.88	123.74
Instructions vs. Novec [%]	-	127.79	113.09	23.09	-	102.04	45.65	18.64
Arithmetic Instructions vs. Novec [%]	-	58.62	60.74	25.05	-	57.53	33.14	22.74

Hot Spots Measurements

Analysis of SIMD Vectorization Effectiveness

- When no SIMD vectorization is applied, the numbers of arithmetic instructions and total instructions of FN MiniMD are both larger than those numbers of HN MiniMD.
- For both HN and FN MiniMD, whichever SIMD vectorization is applied, the number of arithmetic instructions will always reduce. Especially for AVX512, that will reduce by about 75%.
- When no SIMD vectorization is applied, FN MiniMD costs more time than HN MiniMD. However, SIMD vectorization is much more effective for FN MiniMD, so that the runtime of FN MiniMD will reduce by about 80% at most with SIMD vectorization, while the runtime of HN MiniMD will only reduce by about 50% at most with SIMD vectorization.

Friedrich-Alexander-Universität
Technische Fakultät

Thank You!