MiniMD Benchmark on CPU with OpenMP and MPI (Part I)

Jinwen Pan

Advisors:
Jan Eitzinger
Dominik Ernst

04/07/2022

Introduction

- MiniMD is a molecular dynamics simulation mini-application in the Mantevo project at Sandia National Laboratories. The primary authors of miniMD are Steve Plimpton, Paul Crozier and Christian Trott.
- MiniMD is implemented in C++ and intended to be used on parallel supercomputers and new architectures for testing purposes.
- MiniMD consisting of less than 5000 lines of code is a miniature version of the molecular dynamics application LAMMPS. They are both based on the spatial decomposition algorithm.
- MiniMD enables users to specify problem size, atom density, temperature, timestep size, number of timesteps to perform, and particle interaction cutoff distance.

Basic Concepts

- Molecular dynamics is a computer simulation of the evolution of a multi-molecules/atoms system over time within the framework of classical **Newtonian mechanics** based on the initial positions and velocities of molecules/atoms.

- Interaction Forces and Potential Functions:
  - Lennard-Jones Potential:

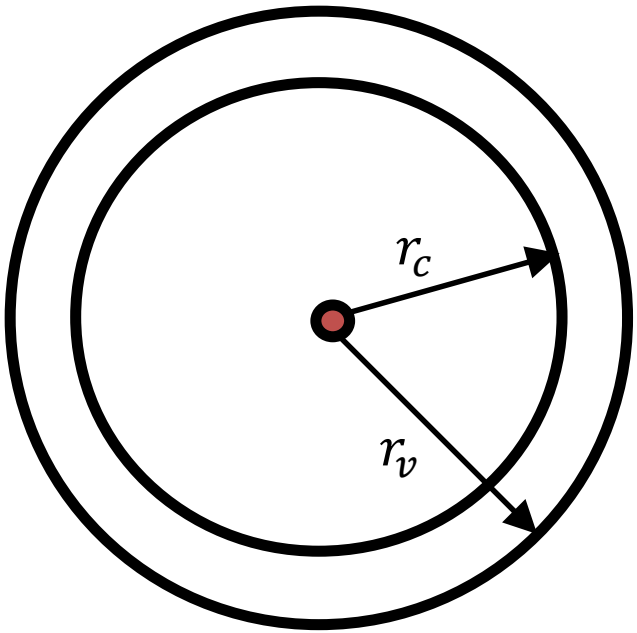
$$V(r) = 4\varepsilon[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}];$$

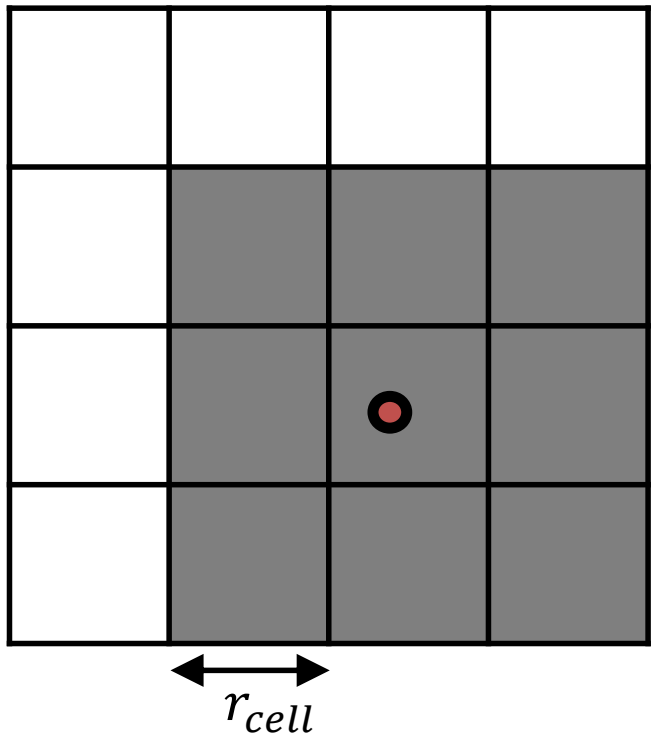
  - Embedded Atom Model Potential:

$$E_i = F_\alpha(\sum_{j \neq i} \rho_\beta(r_{ij})) + \frac{1}{2}\sum_{j \neq i} \emptyset_{\alpha\beta}(r_{ij}).$$

Basic Concepts

- Cutoff Distance:
  - Half and Full Neighbor Lists Approach;
  - Cell Lists Approach.





Update Strategy of MiniMD

- Initialize:

  For each molecule/atom:
  - Position
  - Velocity
  - Force
  - Neighbor List

- Update:

  For each time step:

  For each molecule/atom:
  - Update velocity by force and Newton's Second Law (1/2 time step)
  - Update position (1 time step)
  - Update neighbor list
  - Update force
  - Update velocity by updated force (1/2 time step)

Test System Fritz

- 944 compute nodes with 2 sockets per node and 36 cores per socket
- CPU name: Intel(R) Xeon(R) Platinum 8360Y CPU @ 2.40GHz
- CPU type: Intel Icelake SP processor
- 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)
- 4 NUMA domains (2+2): 18 cores per domain (Cluster On Die / SNC active)

Build and Run MiniMD

- Build MiniMD:
  - $ module load intel/2021.4.0
  - $ module load intelmpi/2021.4.0
  - $ module load likwid/5.2.1
  - $ make (can be compiled by ICC, GCC, or CLANG)

- Run MiniMD:
  - $ cd ./data (LJ and EAM inputs are available and can be modified)
  - $ sbatch job.sh (an MPI starting script is always needed)

```
#!/bin/bash -l

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=72
#SBATCH --time=01:00:00
#SBATCH --export=NONE
#SBATCH --cpu-freq=2400000
#SBATCH --array=1-72
unset SLURM_EXPORT_ENV

module load intel/2021.4.0
module load intelmpi/2021.4.0
module load likwid/5.2.1
```

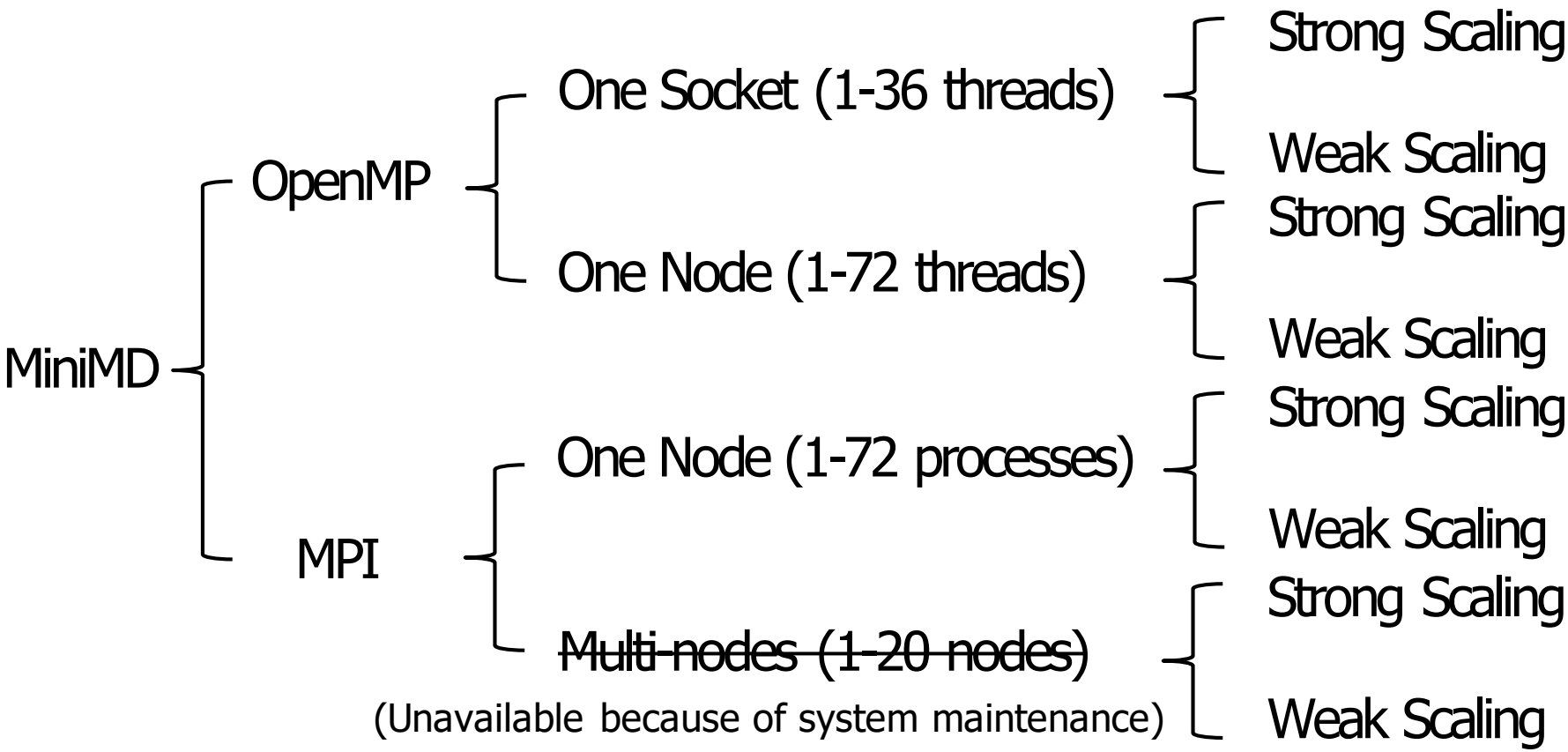
N=4*nx*ny*nz=4*s^3

```
likwid-mpirun -np 1 -mpi intelmpi -omp intel -pin N:0-71 ../miniMD-ICC --num_threads ${SLURM_ARRAY_TASK_ID} --half_neigh 0 -nx 48 -ny 48 -nz 48
```

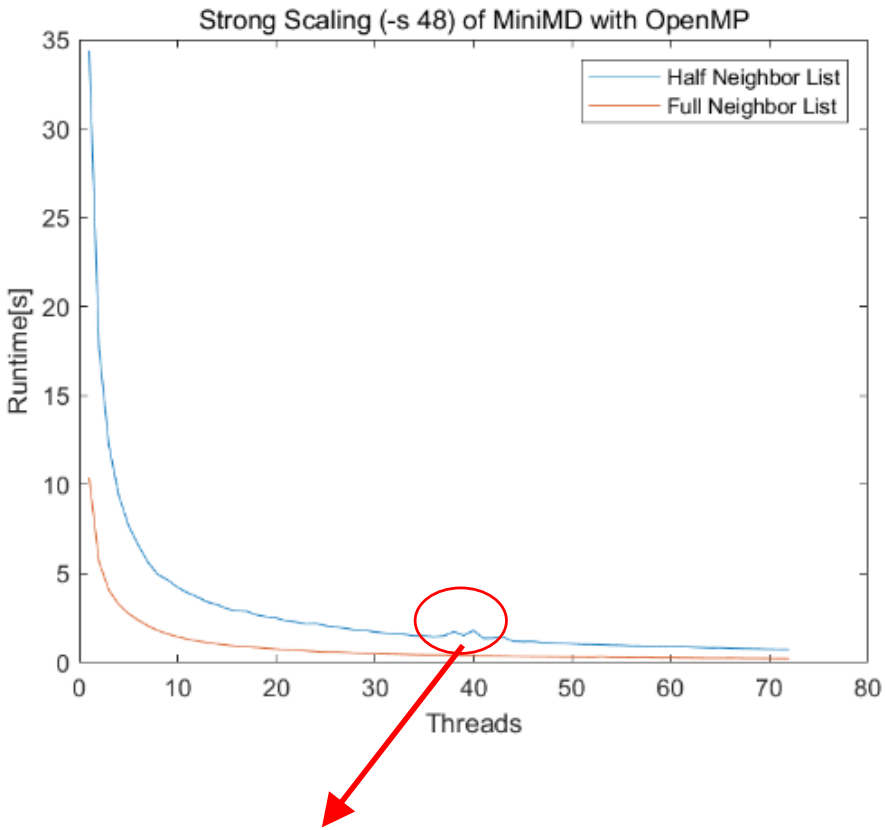
(-s 48)

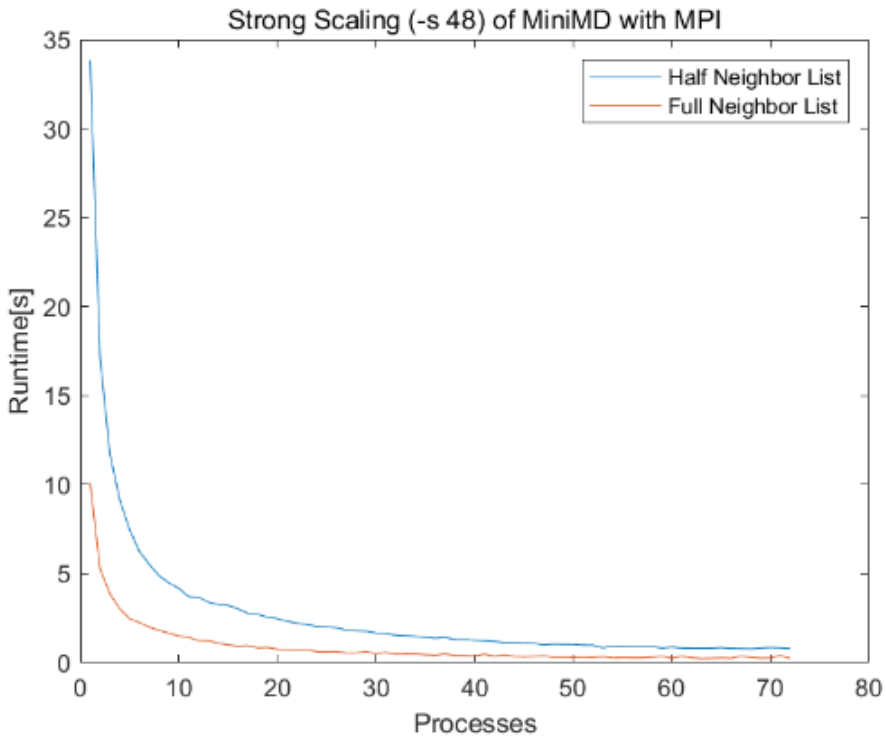
Scaling Runs Overview

For full and half neighbor list variants:



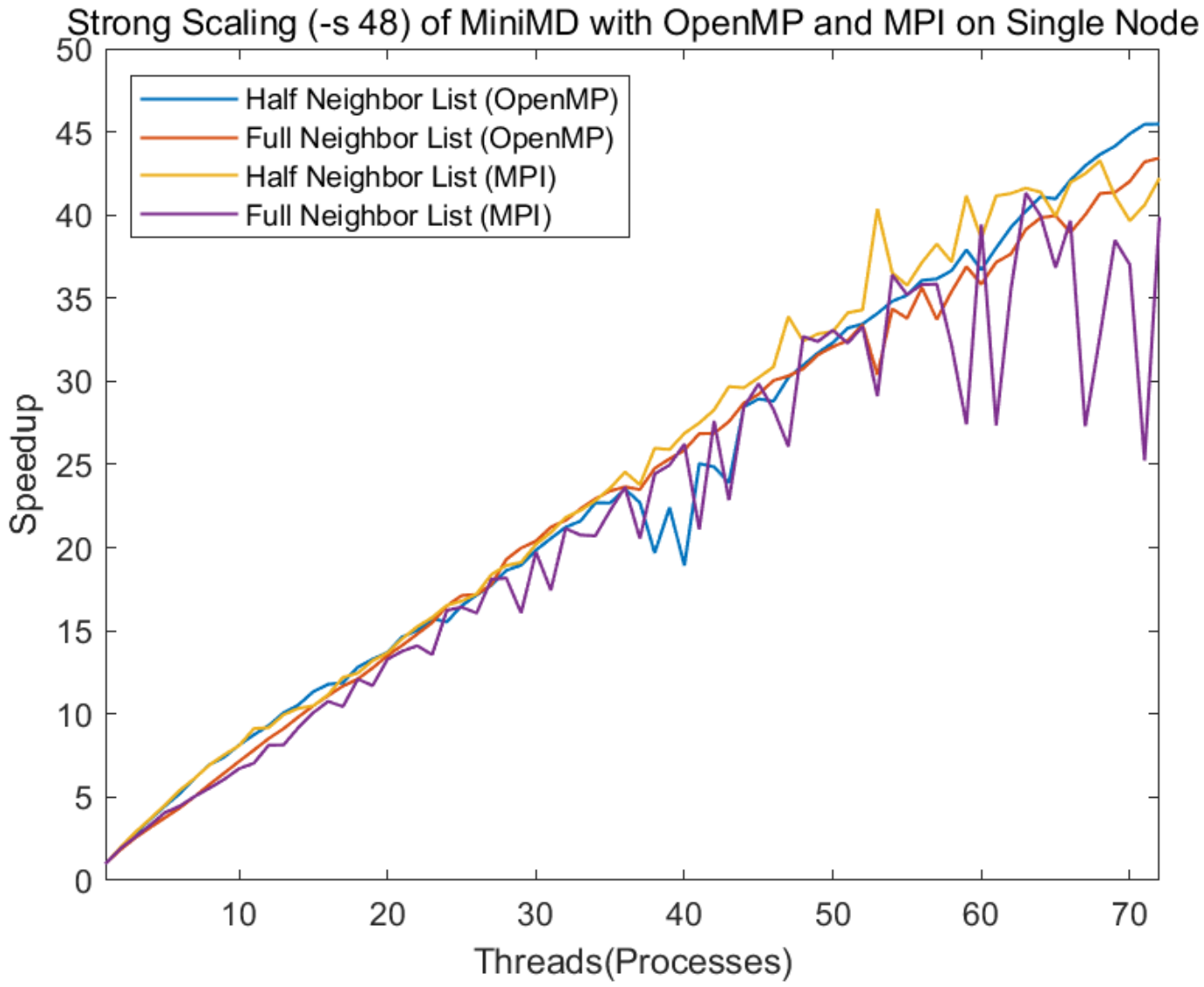
Strong Scaling – Runtime on Single Node (100 time steps)



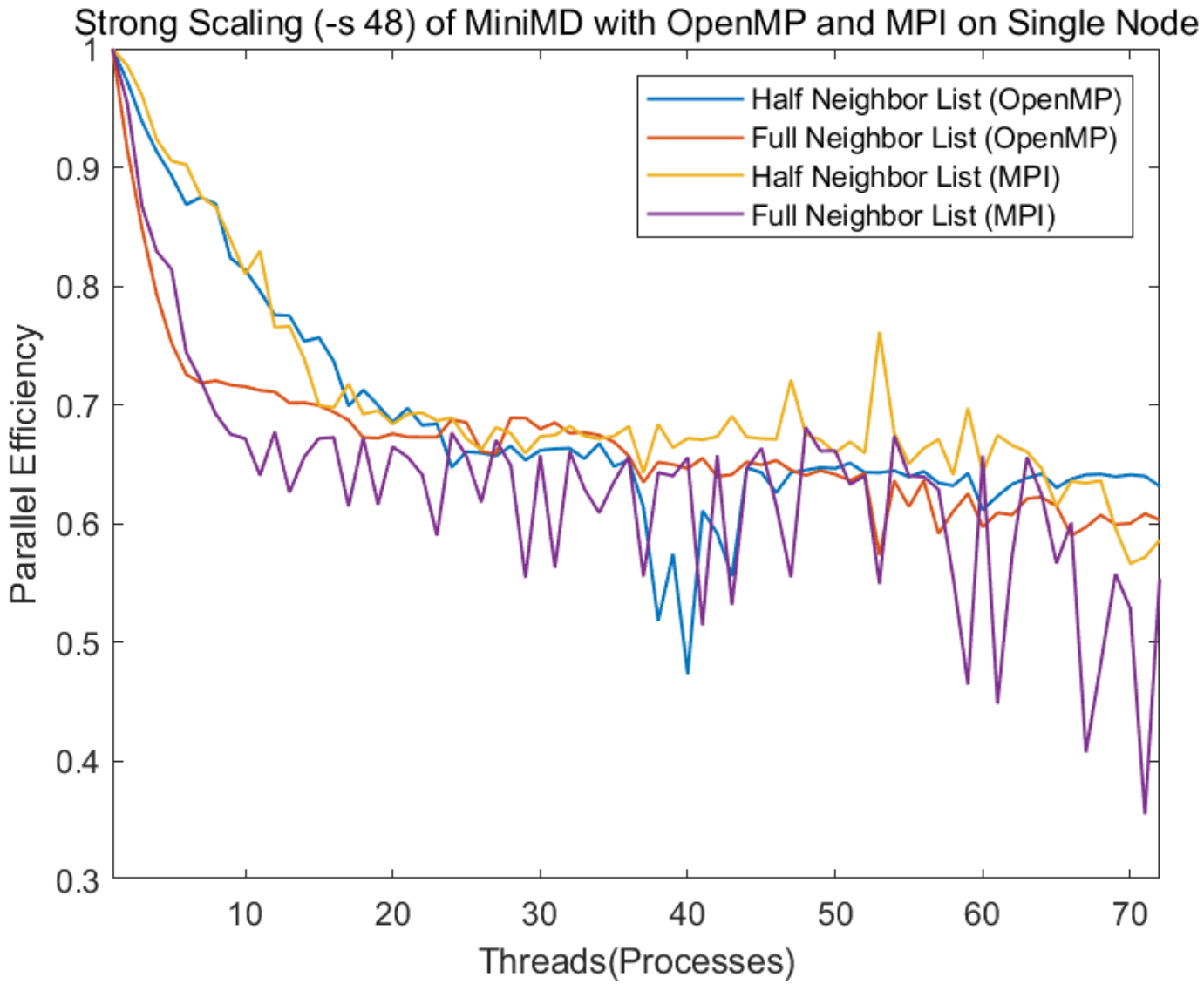


Different sockets (36+36)

Strong Scaling – Speedup on Single Node

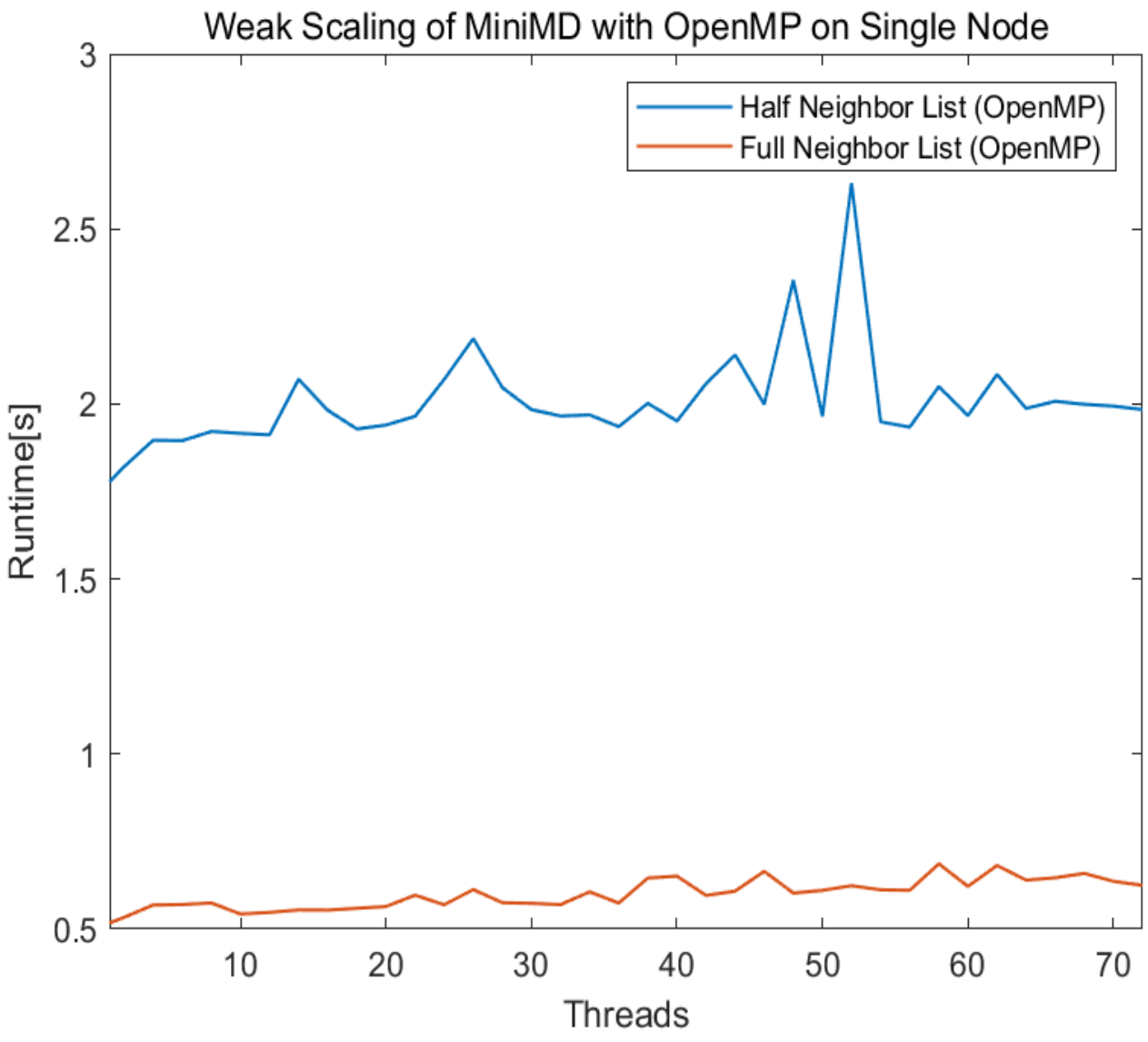


Strong Scaling – Parallel Efficiency on Single Node



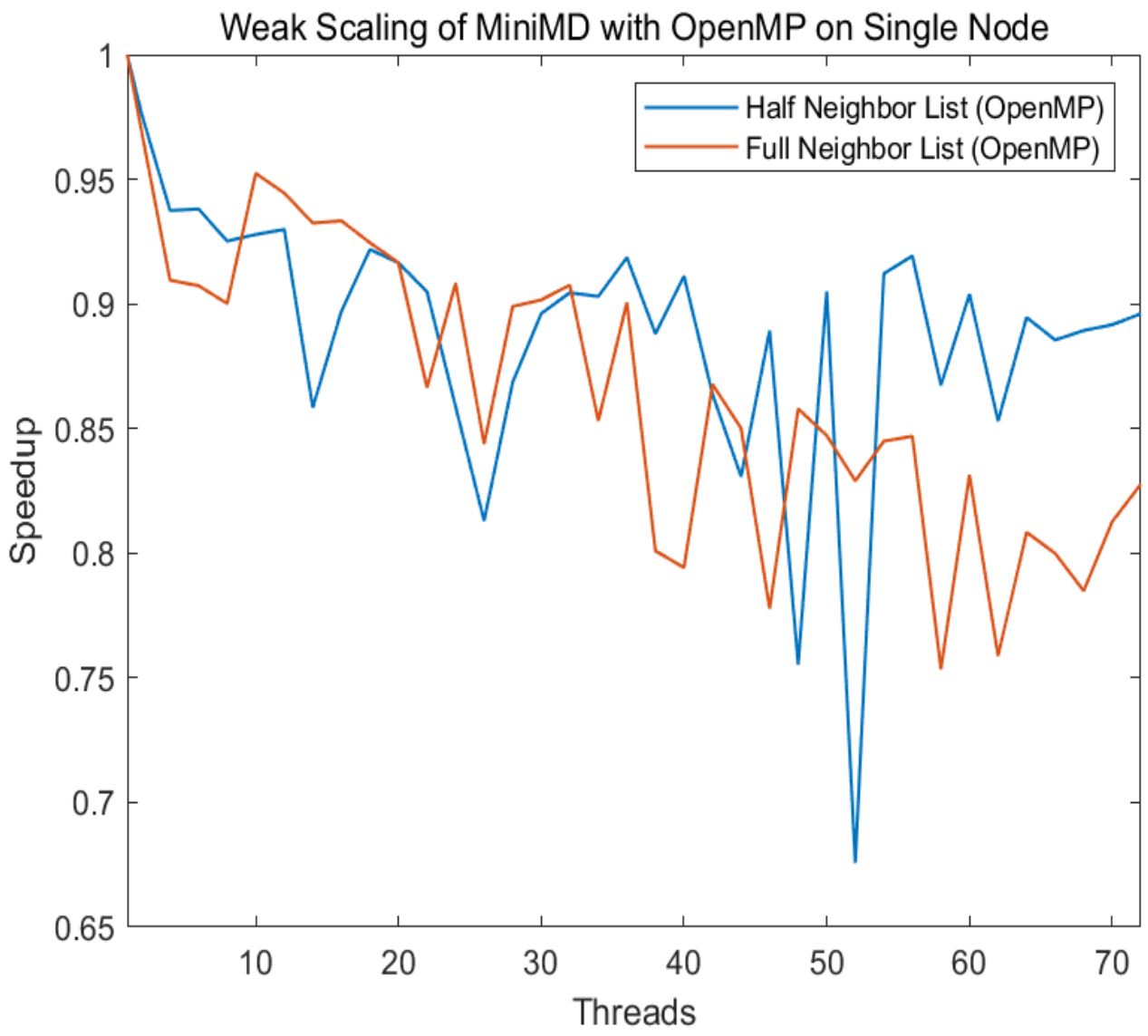
Weak Scaling – Runtime on Single Node (100 time steps)

Threads	nx	ny	nz
1	16	16	16
2	32	16	16
4	32	32	16
6	48	32	16
8	32	32	32
10	80	32	16
12	48	32	32
14	112	32	16
16	64	32	32
18	48	48	32
20	80	32	32
22	176	32	16
24	64	48	32
26	208	32	16
28	112	32	32
30	80	48	32
32	64	64	32
34	272	32	16
36	96	48	32
38	304	32	16
40	80	64	32
42	112	48	32
44	176	32	32
46	368	32	16
48	96	64	32
50	80	80	32
52	208	32	32
54	96	48	48
56	112	64	32
58	464	32	16
60	96	80	32
62	496	32	16
64	128	64	32
66	176	48	32
68	272	32	32
70	112	80	32
72	128	48	48



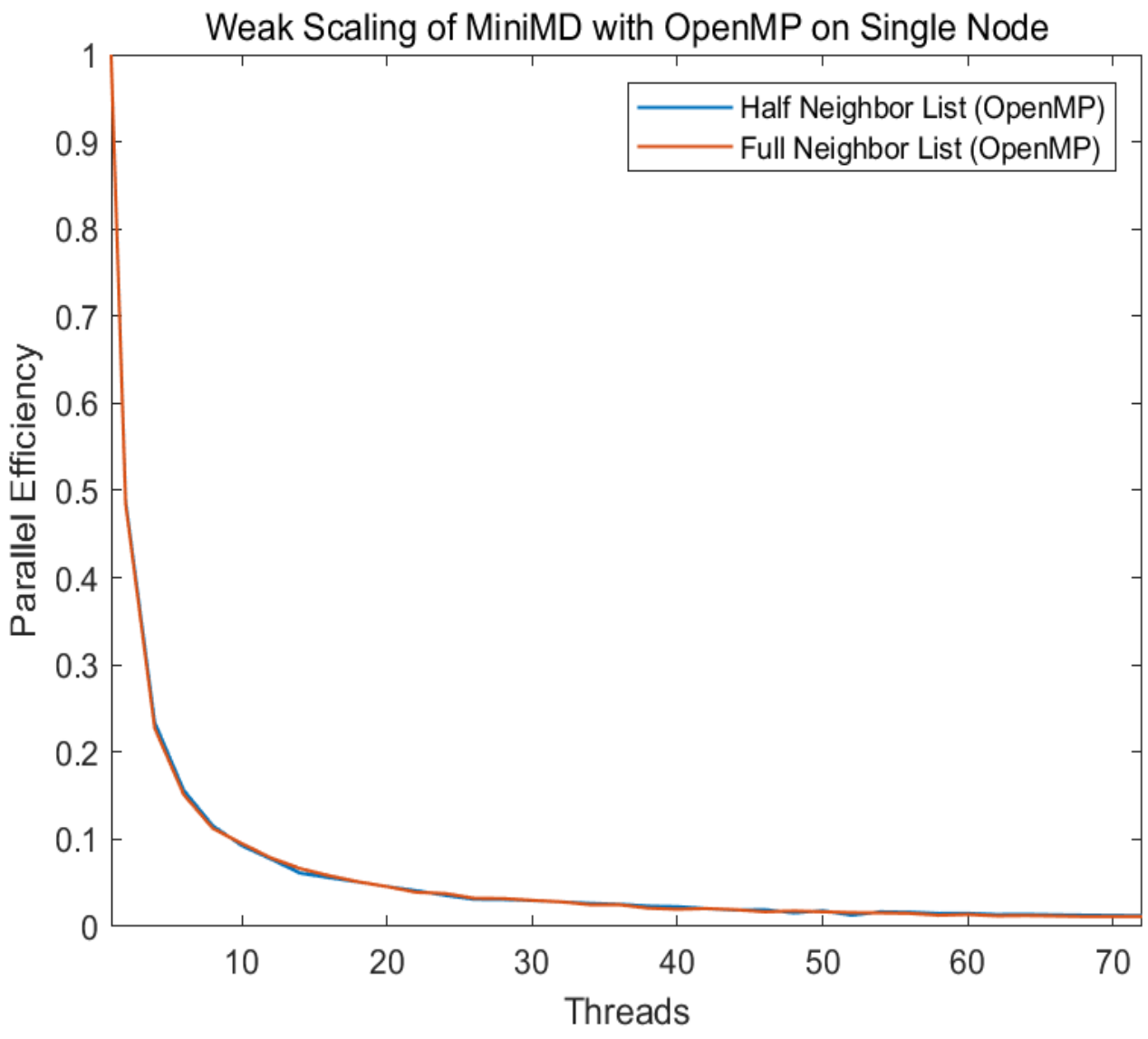
Weak Scaling – Speedup on Single Node

Threads	nx	ny	nz
1	16	16	16
2	32	16	16
4	32	32	16
6	48	32	16
8	32	32	32
10	80	32	16
12	48	32	32
14	112	32	16
16	64	32	32
18	48	48	32
20	80	32	32
22	176	32	16
24	64	48	32
26	208	32	16
28	112	32	32
30	80	48	32
32	64	64	32
34	272	32	16
36	96	48	32
38	304	32	16
40	80	64	32
42	112	48	32
44	176	32	32
46	368	32	16
48	96	64	32
50	80	80	32
52	208	32	32
54	96	48	48
56	112	64	32
58	464	32	16
60	96	80	32
62	496	32	16
64	128	64	32
66	176	48	32
68	272	32	32
70	112	80	32
72	128	48	48



Weak Scaling – Parallel Efficiency on Single Node

Threads	nx	ny	nz
1	16	16	16
2	32	16	16
4	32	32	16
6	48	32	16
8	32	32	32
10	80	32	16
12	48	32	32
14	112	32	16
16	64	32	32
18	48	48	32
20	80	32	32
22	176	32	16
24	64	48	32
26	208	32	16
28	112	32	32
30	80	48	32
32	64	64	32
34	272	32	16
36	96	48	32
38	304	32	16
40	80	64	32
42	112	48	32
44	176	32	32
46	368	32	16
48	96	64	32
50	80	80	32
52	208	32	32
54	96	48	48
56	112	64	32
58	464	32	16
60	96	80	32
62	496	32	16
64	128	64	32
66	176	48	32
68	272	32	32
70	112	80	32
72	128	48	48



Further Work

- Perform additional scaling runs on multi-nodes with MPI ;
- Instrument the benchmark with LIKWID markers ;
- Perform an instruction decomposition analysis .

THANKS