The 47th International Conference on Parallel Processing, ICPP 2018
August 13-16, University of Oregon, Eugene, Oregon, USA

Massively Scaling the Metal Microscopic Damage Simulation on Sunway TaihuLight Supercomputer

Shigang Li, Baodong Wu, Yunquan Zhang
SKL Computer Architectures
Institute of Computing Technology, Chinese Academy of Sciences

Xianmeng Wang, Jianjiang Li, Changjun Hu
University of Science and Technology Beijing

Jue Wang, Yangde Feng, Ningming Nie
Computer Network Information Center, Chinese Academy of Sciences

CONTENTS

Molecular Dynamics (MD)

Molecular Dynamics

A thermodynamic calculation method based on the theory of Newtonian mechanics.

Based on solving the equations of motion of all the particles in the system, thermodynamic quantities and other macroscopic properties of system can be obtained.

Application fields

MD simulation has been widely used in various fields of physical, chemical, biological, materials, medicine, etc.

We aim at the simulation of **Metal Microscopic Damage under the environment of irradiation**. We use **MD to simulate the defect generation caused by cascade collision.**

Simulation scales

Temporal Scale:
Since the time step of MD is very short (typically at femtosecond scale), the temporal scale of MD simulation is generally limited to **nanoseconds or less**.

Spatial Scale:
Up to **10^{12} atoms** as far as we know (Ls1-MarDyn).

There is still a big gap between simulation results and physical phenomena due to the limitation of simulation scales.

Kinetic Monte Carlo (KMC)

Kinetic Monte Carlo, KMC

A very popular algorithm to simulate the dynamics of stochastic process.

It is an efficient tool for exercising the combined actions of fundamental, stochastic, and physical mechanisms.

Application fields

KMC has been widely applied in the simulations of materials micro-damage, grain growth, and film-forming.

In our simulation, we use KMC to simulate the defect evolution and vacancies clustering.

Simulation scales

Temporal Scale:
Simulate in the unit of “event”, rather than “time step”. Thus, the temporal scale can go up to **days or more.**

Spatial Scale:
Commonly larger than the spatial scale of MD.

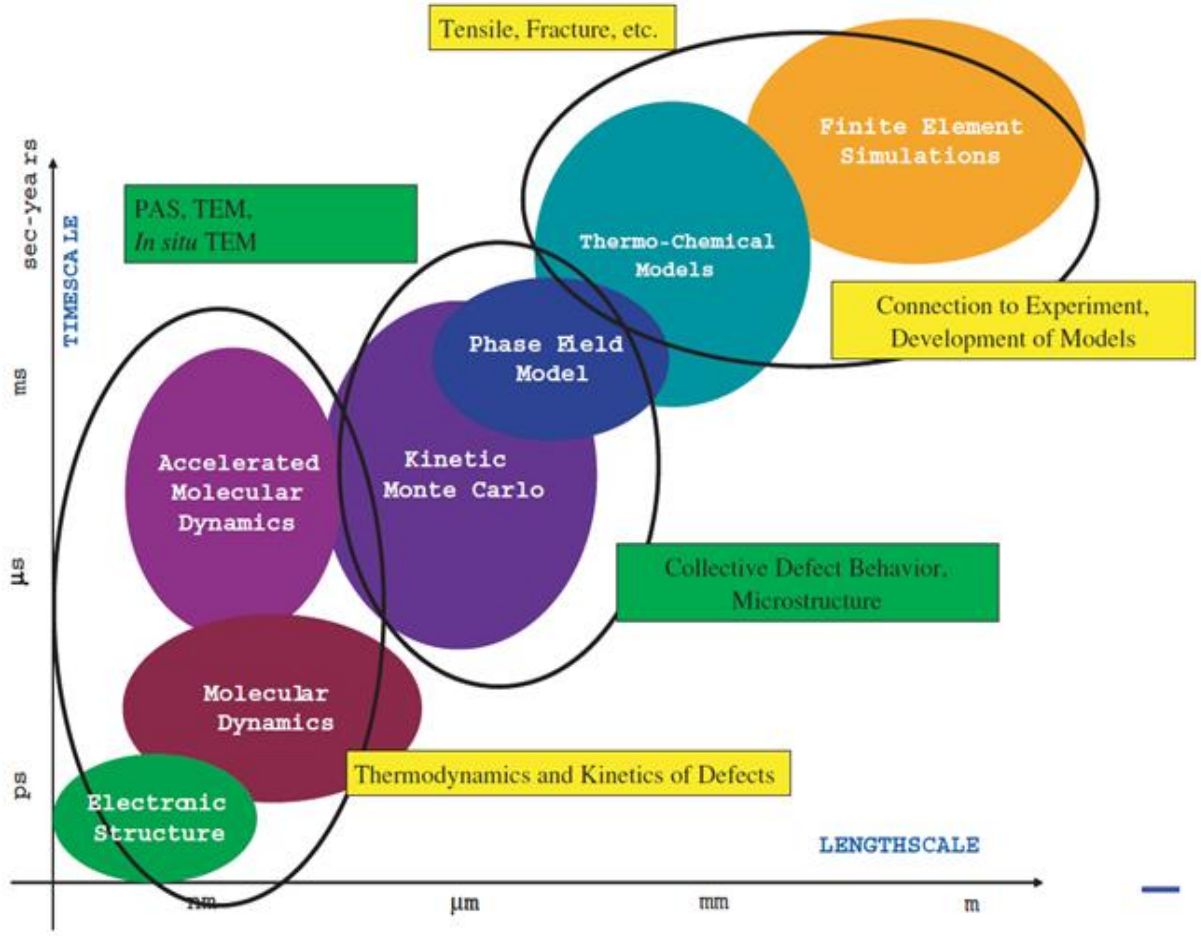
MD vs KMC

MD

- ✓ F=ma
- ✓ **Deterministic method**
- ✓ Real dynamics
- ✓ Many realistic processes are in accessible.
- ✓ **Time step is very short**, and thus very high computation complexity.

KMC

- ✓ Stochastic Process
- ✓ **Non-deterministic method**
- ✓ **Sampling method for large temporal scale**
- ✓ Need all diffusion barrier a priori.
- ✓ KMC steps in REAL TIME and ALL EVENTS accept.



A coupled MD-KMC approach

Issues

1. **A single model usually only performs well at a narrow temporal and spatial range.** For example, MD commonly simulates millions to billions of atoms at picoseconds to nanoseconds.

2. **Large-scale simulation has high computation complexity and requires a great amount of computing resources.** Although the supercomputers are becoming more powerful, the existing software lag behind because of the high effort of code porting and optimization on new machines.

Solutions

1. We use a **coupled MD-KMC approach** intending to achieve both high temporal and spatial scales.

2. **MD simulates the defect generation caused by collision cascades**, and outputs the coordinates of vacancy and the information of atoms. **KMC simulates the defect evolution and clustering.**

3. **We optimize our coupled MD-KMC code on Sunway TaihuLight supercomputer**, and resort to its rich computing resources to conduct large-scale microdamage simulation.

CONTENTS

Neighbor list establishment for MD

- The simulation is based on the interactions whose range is comparatively short, namely **short-range forces**. Thus, only the atoms within a specific **cutoff radius** interact with the central atom.

- **Two data structures** are commonly used to find the neighbor atoms for MD: **neighbor list** and **linked cell,** i.e. LAMMPS uses neighbor list and IMD uses linked cell structure.

Neighbor list establishment for MD

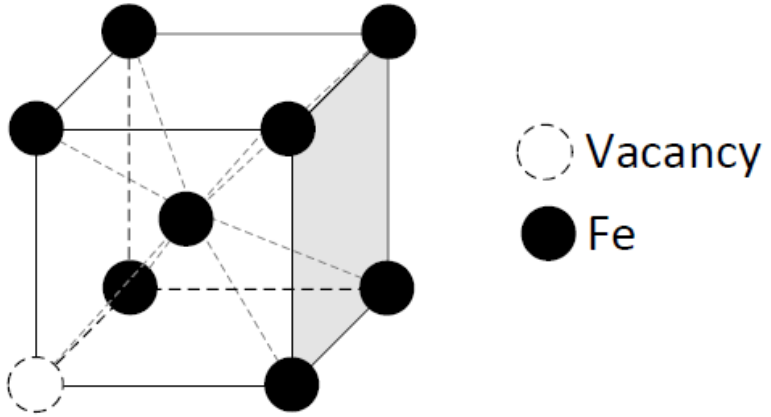
Neighbor list

1. Each atom maintains a list to store all the neighbor atoms within a distance which is equal to the **cutoff radius** plus a **skin distance.**
2. The **memory consumption** of neighbor list is costly.
3. **$O(N^2)$ complexity** for each iteration of update.
4. The neighbor atoms should **be updated after several time steps**.

Linked cell

1. Linked cell **divides the simulation box into cubic cells**, whose edge length is equal to the cutoff radius.
2. Linked cell ensures that all interaction partners for any given atom are located either within the cell of itself or the surrounding neighbor cells.
3. **$O(NN_c)$ complexity** for each iteration of update, where N_c is the average number of particles in each cell and **$N_c << N$.**
4. Linked cell should **update the atoms within each cell at each time step,** which leads to poor computation efficiency.

A novel lattice neighbor list data structure



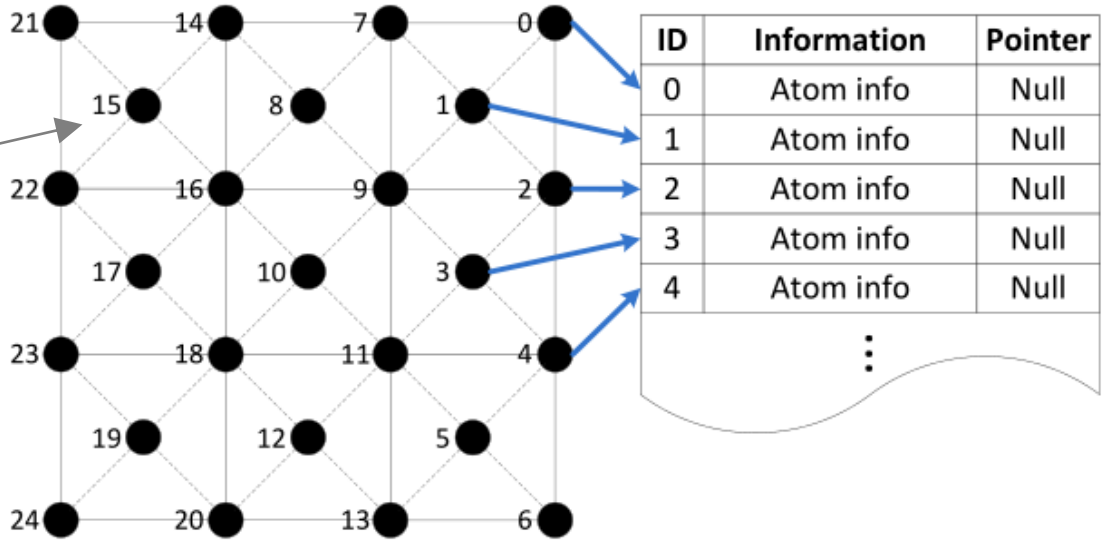
The structure of body-centered cubic.

In the simulation **only a few atoms break the constrain and run away from the lattice point**. According to this phenomena, we propose a new data structure, called **lattice neighbor list** to store the atoms.

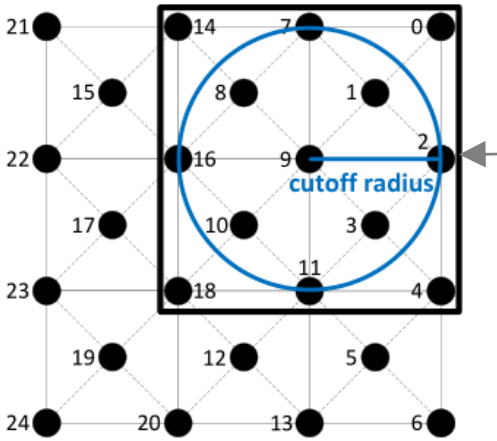
A novel lattice neighbor list data structure

- ✓ We rank the atoms in the order of their spatial distribution. The information of the atoms is **sequentially stored** in an array in the order of the atoms ranks.
- ✓ It's easy to calculate the indexes of the neighbor atoms in the array for a central atom.

1



ID	Information	Pointer
0	Atom info	Null
1	Atom info	Null
2	Atom info	Null
3	Atom info	Null
4	Atom info	Null
⋮		



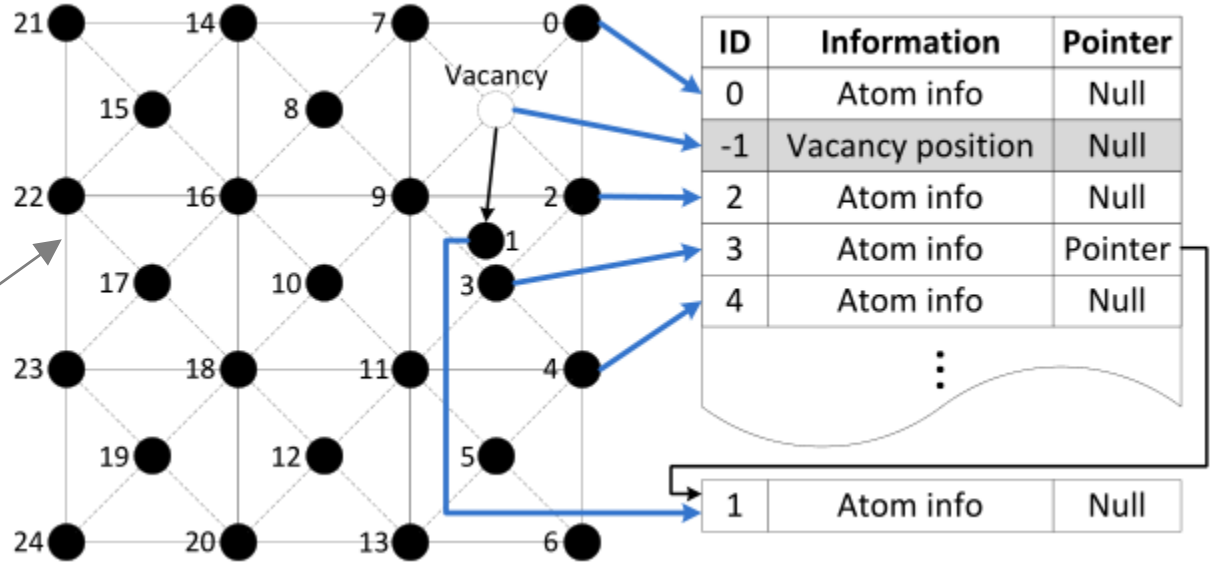
2

- ✓ All the possible neighbor atoms of atom 9 are within the black box.
- ✓ For each central atom, **the offsets of the neighbor atoms relative to the central atom are the same**.

A novel lattice neighbor list data structure

We use the linked list structure to store the run-away atoms.

- ✓ When an atom runs away from the lattice point, it generates a vacancy. (3)
- ✓ We allocate another memory space to store the information of the run-away atom, and leave the original entry in the array to record the coordinates of the vacancy.
- ✓ The run-away atom is **linked to its nearest neighbor atom**.

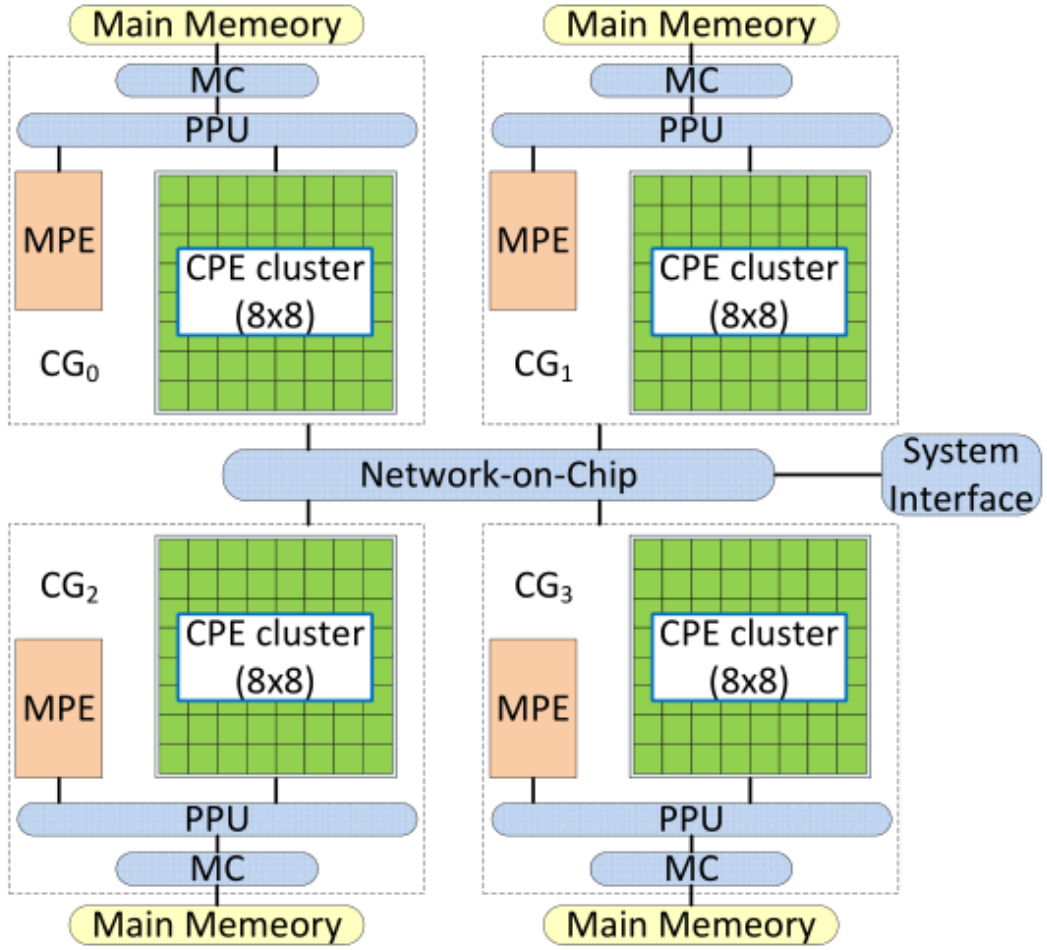


Overall, compared with the traditional neighbor list and linked cell structures, our lattice neighbor list structure significantly reduces the memory and computation cost, since it does not have to maintain the extra structures and finds most of the neighbor atoms by static indexes.

Sunway many-core processor

Sunway TaihuLight Supercomputer

Cores:	10,649,600
Linpack Performance (Rmax)	93,014.6 TFlop/s
Theoretical Peak (Rpeak)	125,436 TFlop/s
Nmax	12,288,000
Memory:	1,310,720 GB
Processor:	Sunway SW26010 1.45GHz
Many-core architecture	4 CGs(each CG 1MPE+8*8CPEs)
Slave core	**64KB local store**



Sunway SW26010 many-core architecture

Compacted interpolation table and online interpolation

The traditional interpolation table (size: 5000*7)

a) **The interpolation table is used to calculate the EAM potential.**

b) Each traditional interpolation table is a 5000*7 2D array in double precision.

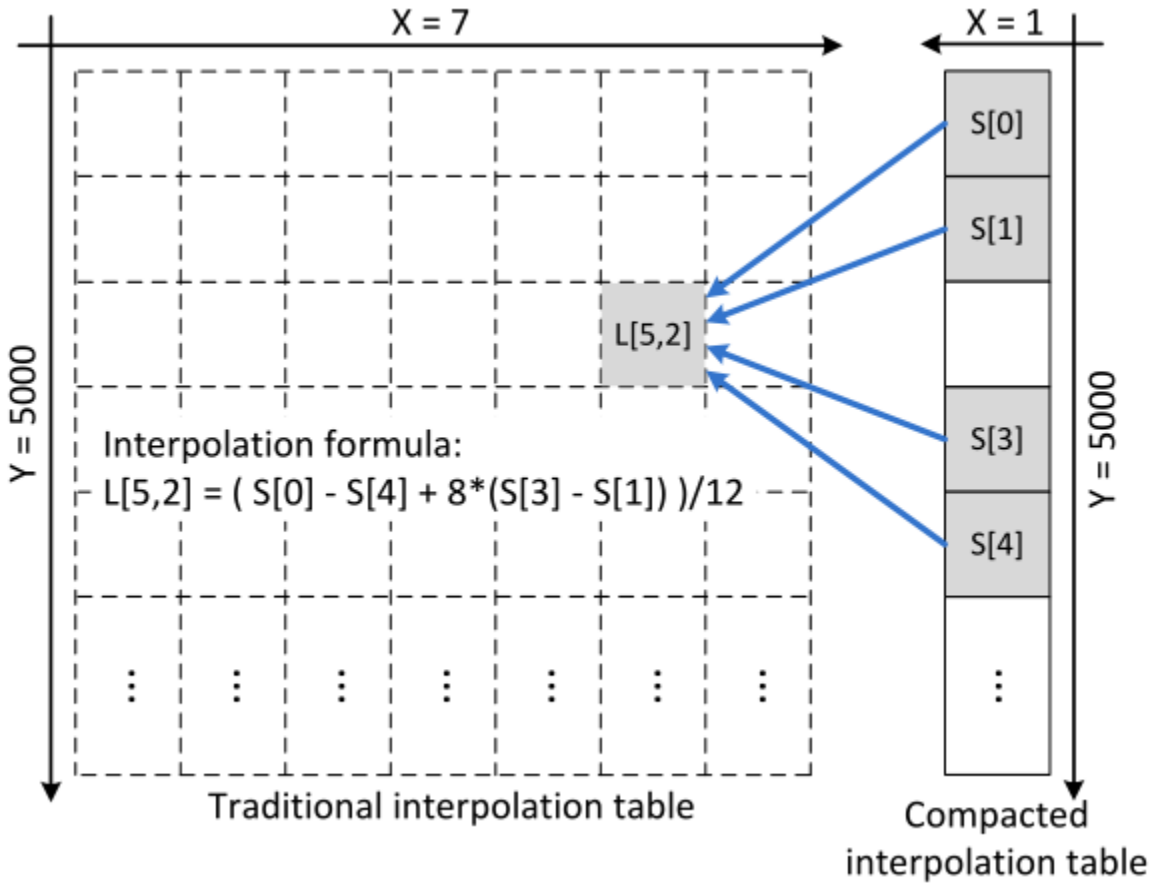
c) The table is about 273 KB, which exceeds the size of local store (64 KB) on each slave core. Thus, **frequent DMA operations** are necessary.

The compacted interpolation table (size: 5000*1)

A. **We only load the compacted table (39 KB) into the local store at one time.**

B. The final value is calculated online using a specific interpolation formula (cubic spline interpolation).

C. **The memory consumption is reduced by 7 folds, although higher computation complexity.**



Reduce the overhead of atom information transfer

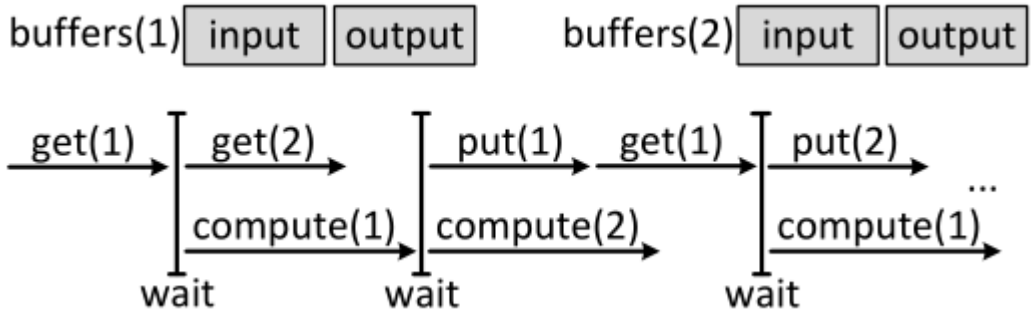
Since our simulation has large spacial scale**, the atoms information of one slab cannot be loaded into the local store at one time either**. **To reduce the overhead of atoms information transfer, we propose two methods:**

(1) Ghost data reuse

For the atoms information of one block, the data on the edge of the block is actually the ghost data for the next block. Thus, we keep the ghost data in the local store and reuse it in the next block, which reduces the overhead of data transfer to some extent.

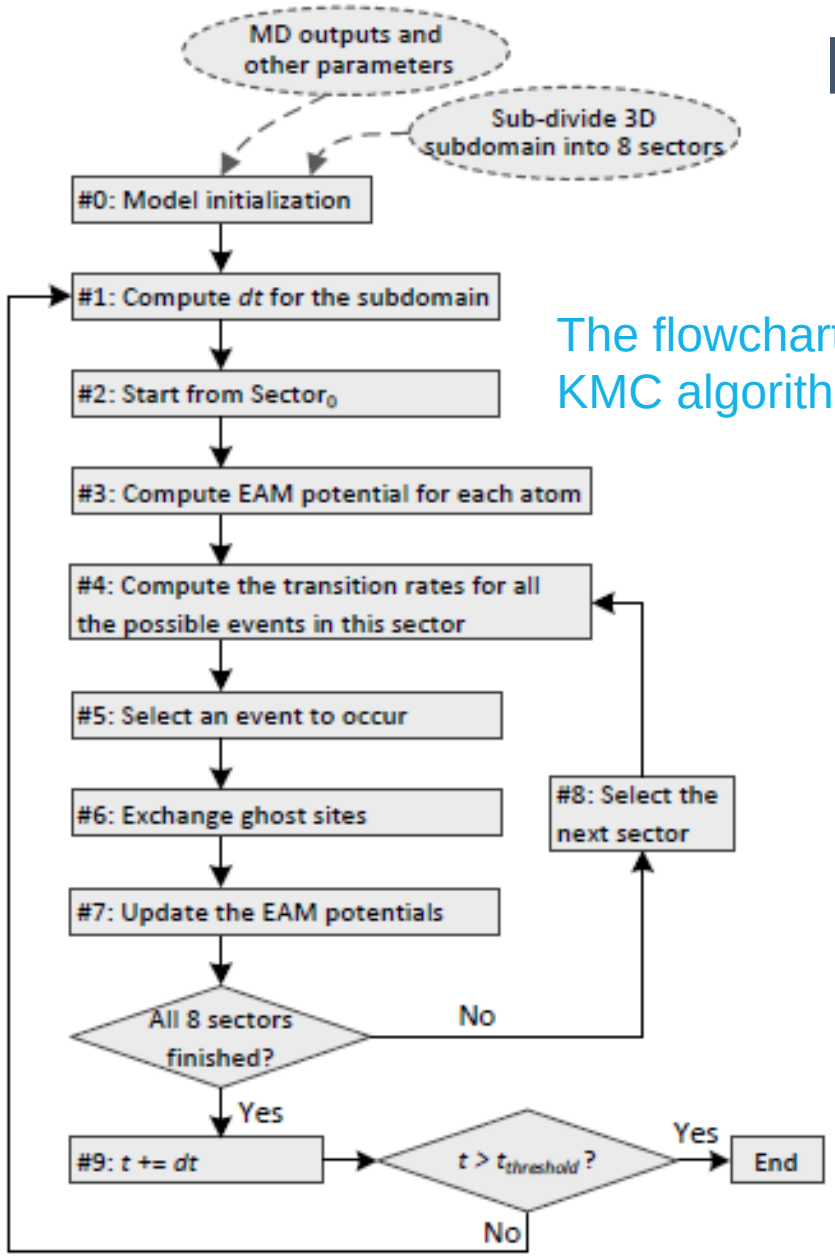
(2) Double buffer

Each slave core allocates two buffers on local store for the input and output data of each block. While carrying out DMA put or get on one buffer, it computes the potential or force on the other buffer, and vice versa. The data transfer and computation are overlapped.



CONTENTS

Kinetic Monte Carlo



The flowchart of KMC algorithm.

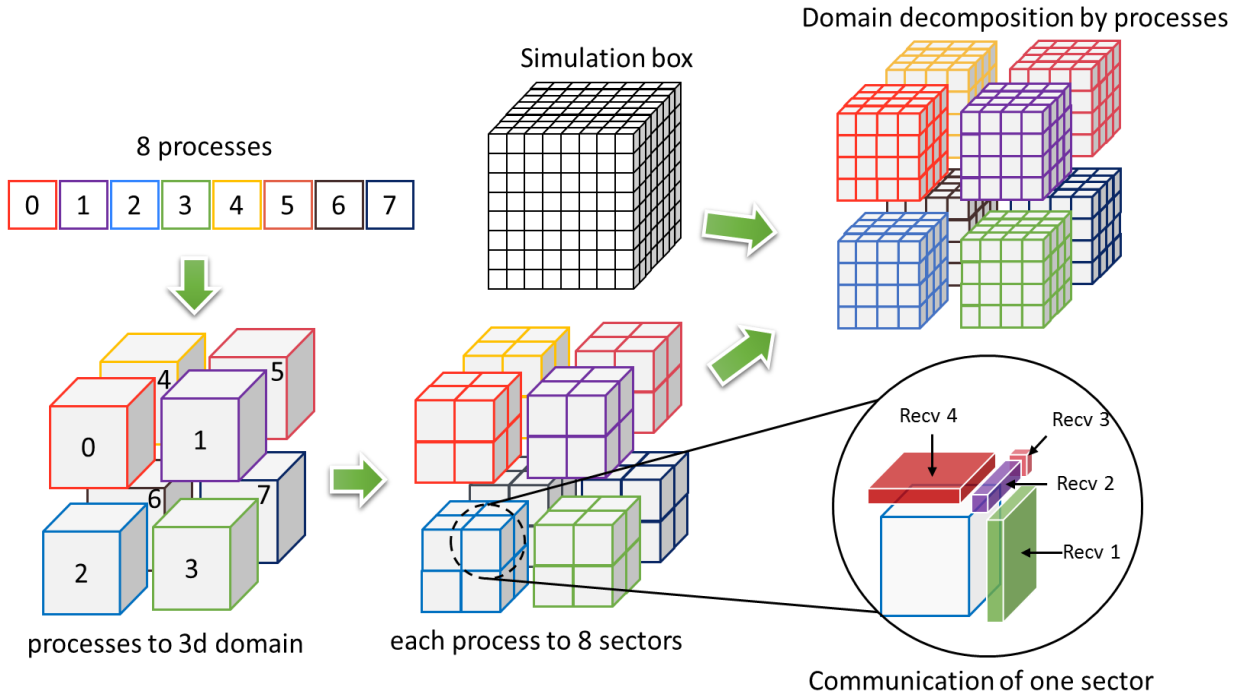
1. Initialize the simulation lattice, sites (atoms and vacancies) data.
2. Calculating the transition probability for the events, generate a random number to select an event to occur.
3. Perform the event, exchange sites data, update the transition probability。
4. Accumulate *dt* to *t*, and determine whether to end conditions (whether *t* reaches the threshold).

The parallel KMC algorithm

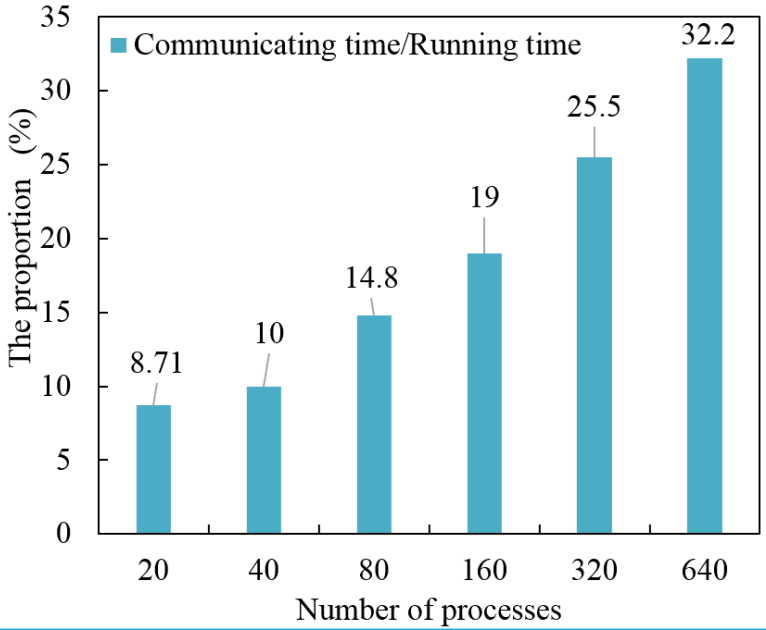
The algorithm is parallelized based on domain decomposition.

The simulation domain is a 3d box, consists of many lattice sites.

For 3D simulation box, processes are arranged to 3d topology to minimize surface area of each subdomain.



Communication optimization for parallel KMC



Communication overhead is the performance bottleneck

- ✓ The proportion of communication time raises rapidly.
- ✓ Communication congestion.
- ✓ Communication redundancy.

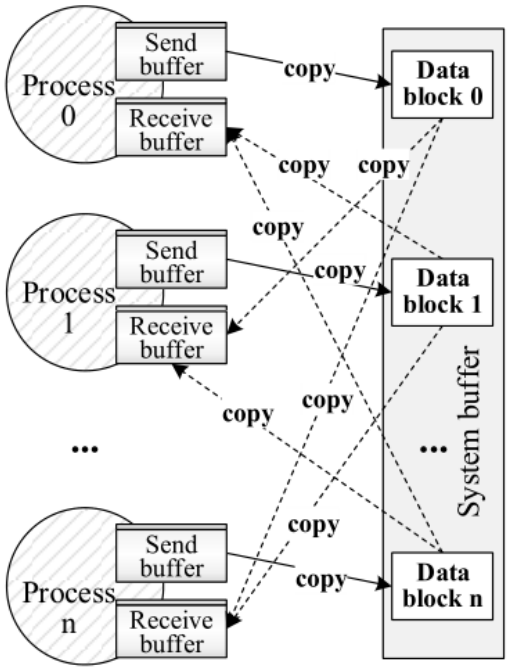
Optimization approach

- ✓ **Shared-memory communication**
- ✓ **On-demand communication strategy**

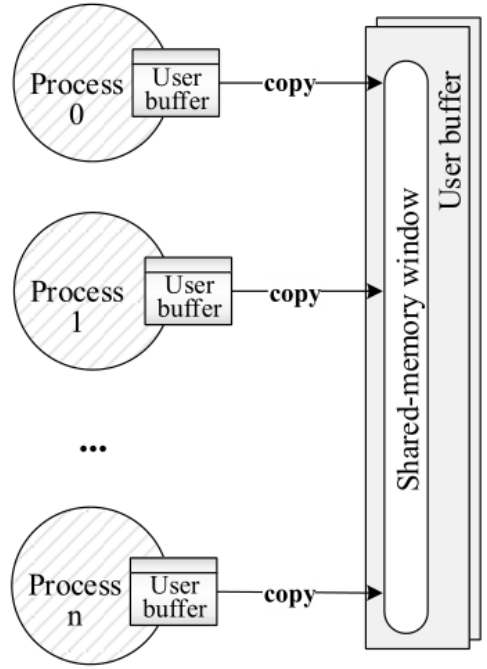
Shared-Memory optimization for intra-node communication

We use the latest MPI interfaces to create shared-memory regions for intra-node processes. The shared region can be directly accessed by the processes.

- ✓ The number of cores per intra-node has been increasing rapidly.
- ✓ **Shared-memory method has a less memory copies than the traditional intra-node point-to-point communication.**
- ✓ Reducing the memory copy overhead of the intra-node communication.
- ✓ We choose to use the ***MPI_Win_allocate_shared*** **and** ***MPI_Win_shared_query*** functions, which allow us to directly replace the send and receive operations with memory copies.



(a) Point-to-point communication

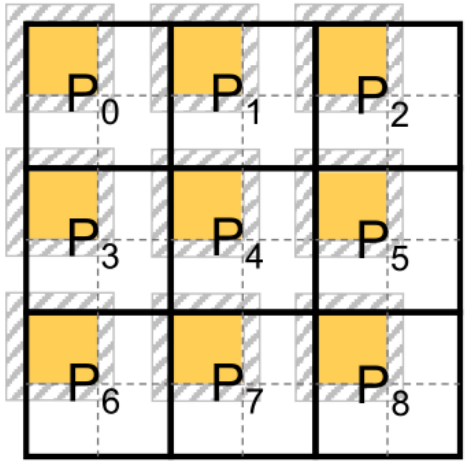


(b) Shared-memory communication

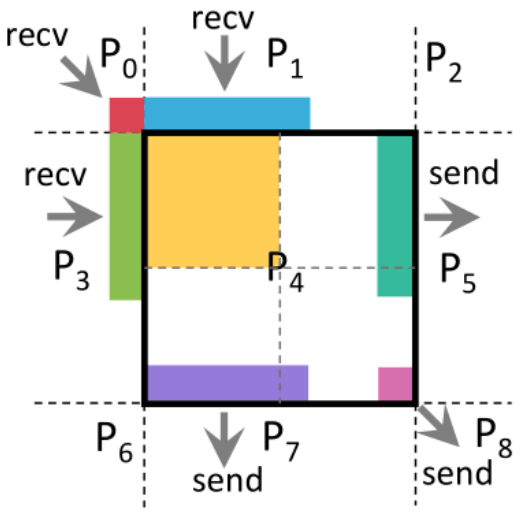
On-demand communication strategy for KMC

The on-demand communication strategy changes the traditional static communication pattern to a dynamic communication pattern, namely the data to be transferred and the communication object are determined at runtime according to the simulation system status. This can eliminate the communication redundancy.

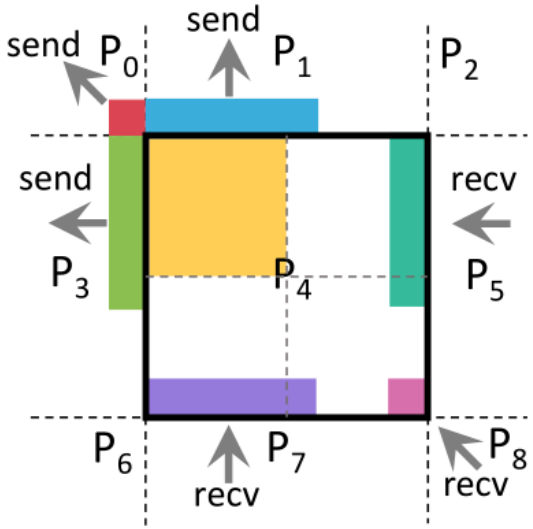
- ✓ The processor adds the new ghost sites and boundary sites to the send queue, and packages those sites data which have the same destination.
- ✓ Then sends the data of sending queue to the corresponding neighboring processors using MPI_isend().
- ✓ Lastly, using MPI Probe to query the message sites and the receiver can receive the actual message using MPI_recv().



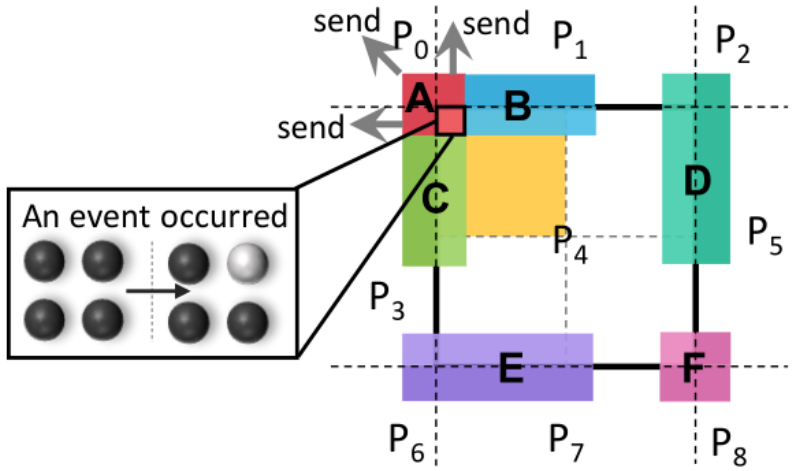
(a) 2d domain partition for 9 processors



(b) Get the latest ghost sites from neighbor processes



(c) Put the ghost sites back to the neighbor processes



(d) On-demand communication

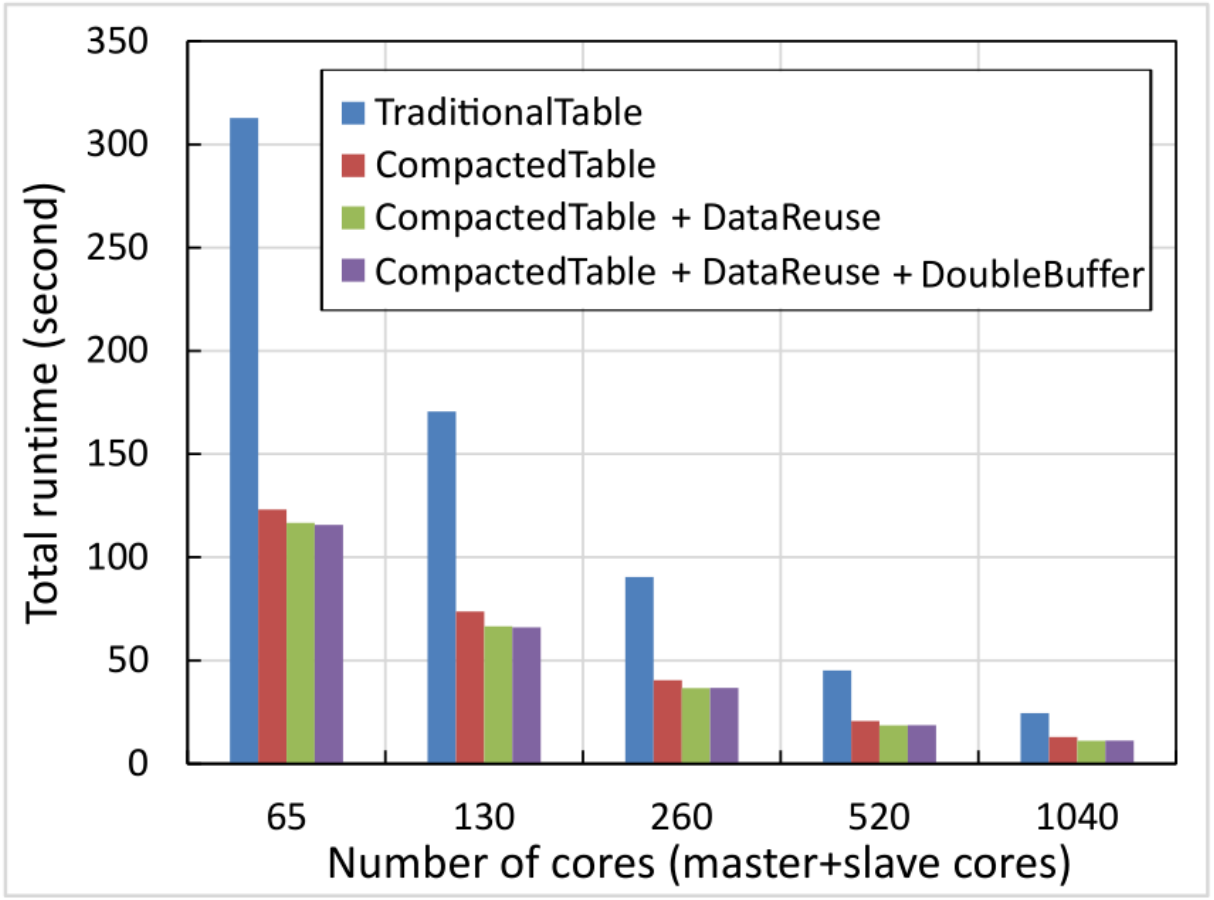
CONTENTS

Experimental environment

Machine name	Sunway TaihuLight supercomputer
Processers	Sunway SW26010 260C 1.45GHz
OS	customized 64-bit Linux kernel
MPI version	SWCC Compilers Version 5.421-sw-495
Application	**Fe metal material**
temperature	600K
Number of cores	Tens of cores to six million cores
Number of sites	2.0*10^7 to 1.024 * 10^12

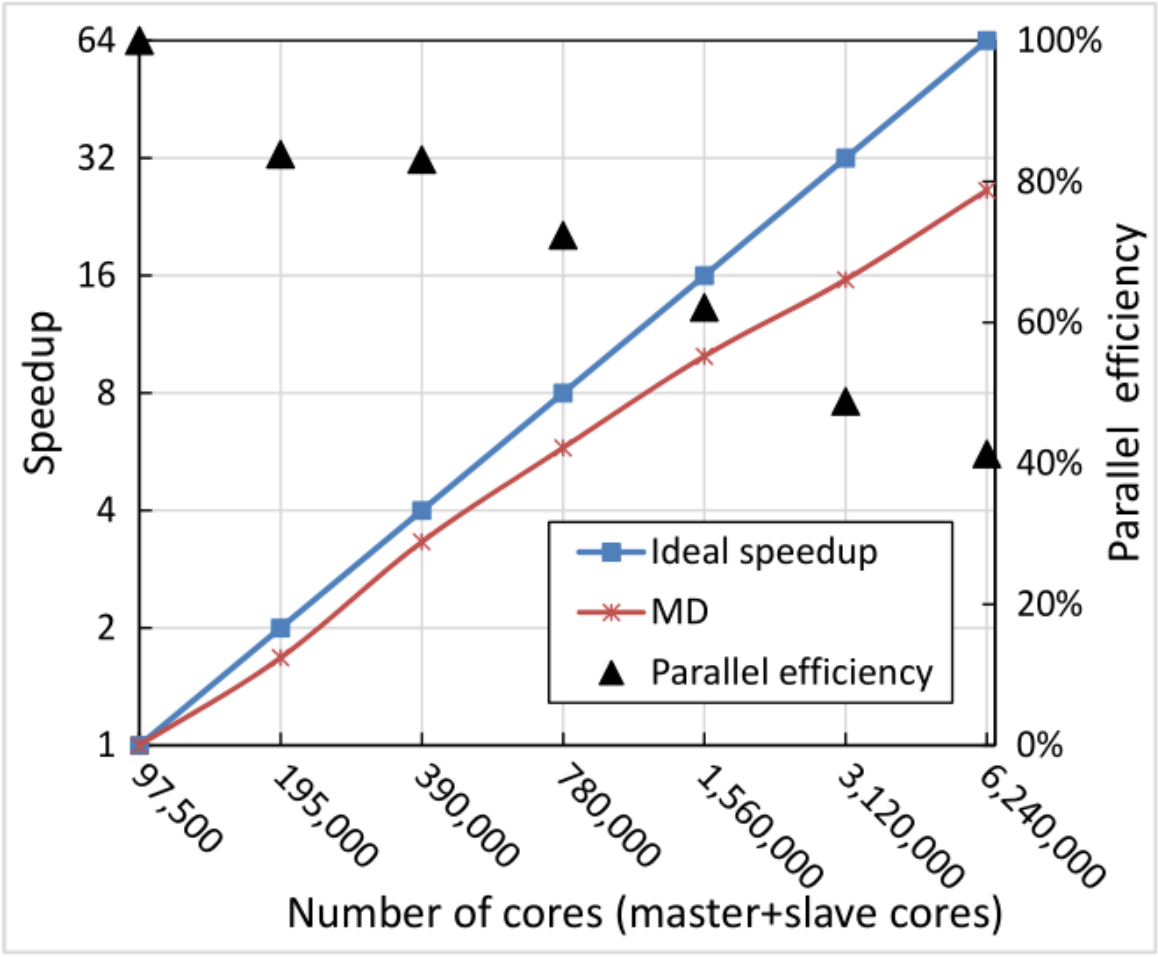
Examine the different optimization methods for MD

- ✓ Firstly, we compare the strong scaling for MD simulation using each optimization method with 2∗10^7 atoms.
- ✓ **The Compacted Table exhibits much better performance than traditional table due to less DMA operations.**
- ✓ **Ghost data reuse** further improves the performance by 4% on average.
- ✓ **Double buffer** does not bring obvious performance improvement, since there is not enough computation to overlap the data transfer.

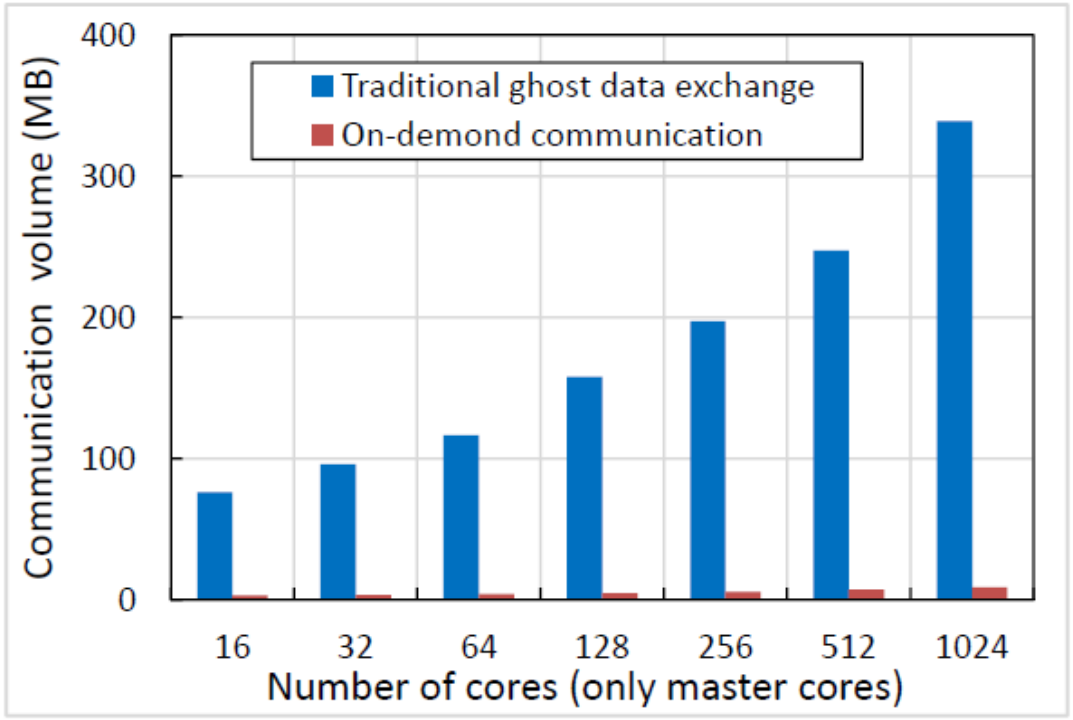


Strong scaling of the optimized MD

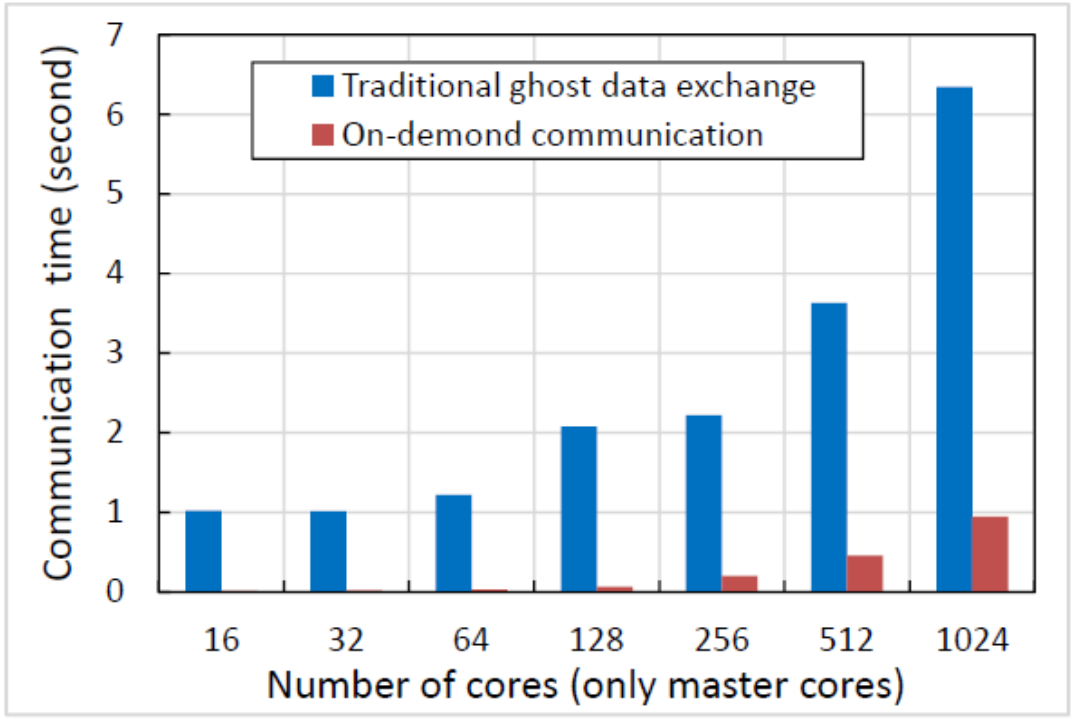
- ✓ We use both master and slave cores to show strong scaling of our optimized MD simulation method.
- ✓ Most communication operations are responsible for the master cores and the main computations are performed by the slave cores.
- ✓ Running with 3.2∗10^10 and scaling from 97,500 cores to 6,240,000 cores, we achieve 26.4-fold speedup or nearly 41.3% parallel efficiency.



On-demond communication for KMC



Communication volume comparison.

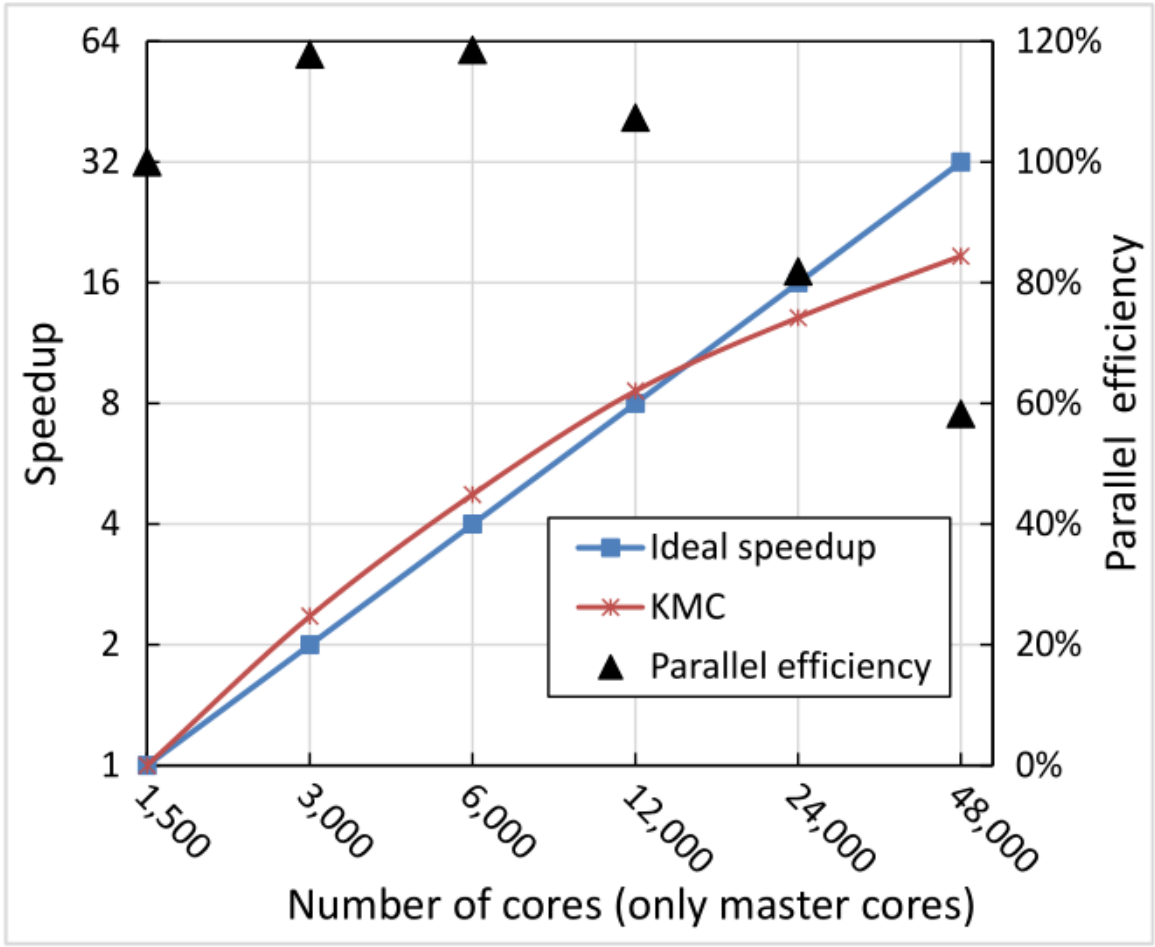


Communication time comparison.

Compared with the traditional method, the on-demand communication strategy obtains **21x speedup** on average in terms of communication time.

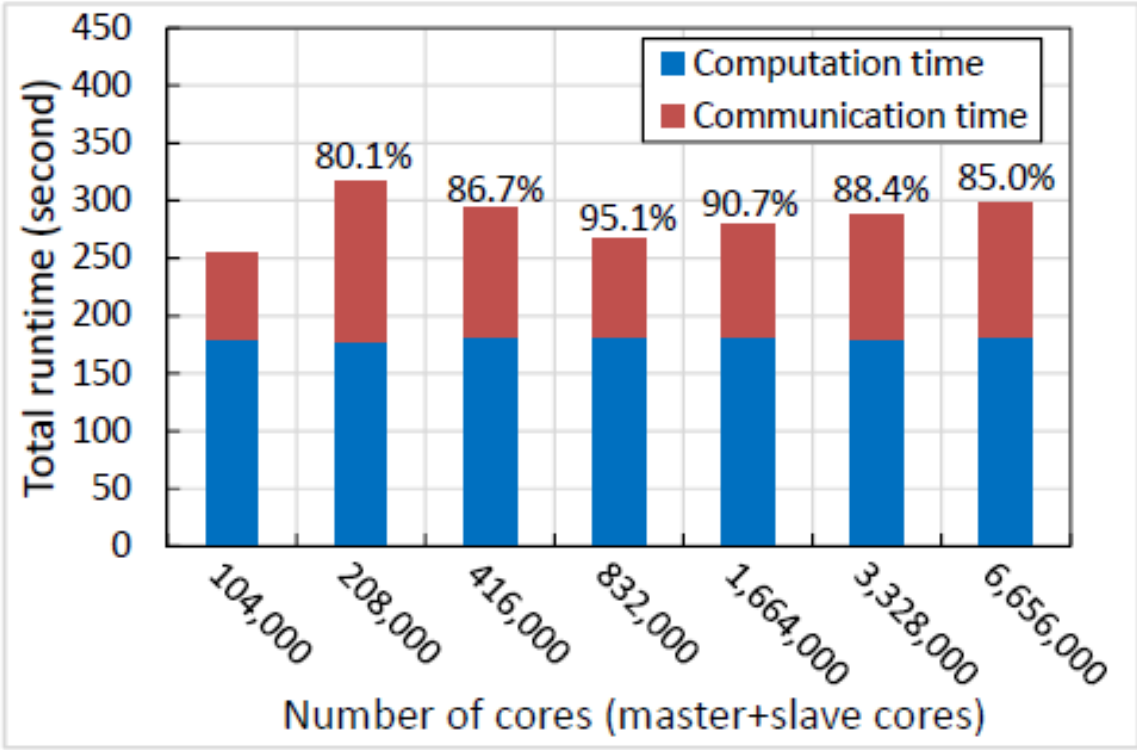
Strong scaling of the optimized KMC

- ✓ Next we tests the parallel performance and scalability of KMC simulations from strong scaling.
- ✓ The baseline run used 1,500 cores with 3.2∗10^10 sites, our algorithm exhibits 18.5-fold speedup on 48,000 cores, indicating 58.2% parallel efficiency in strong scaling.



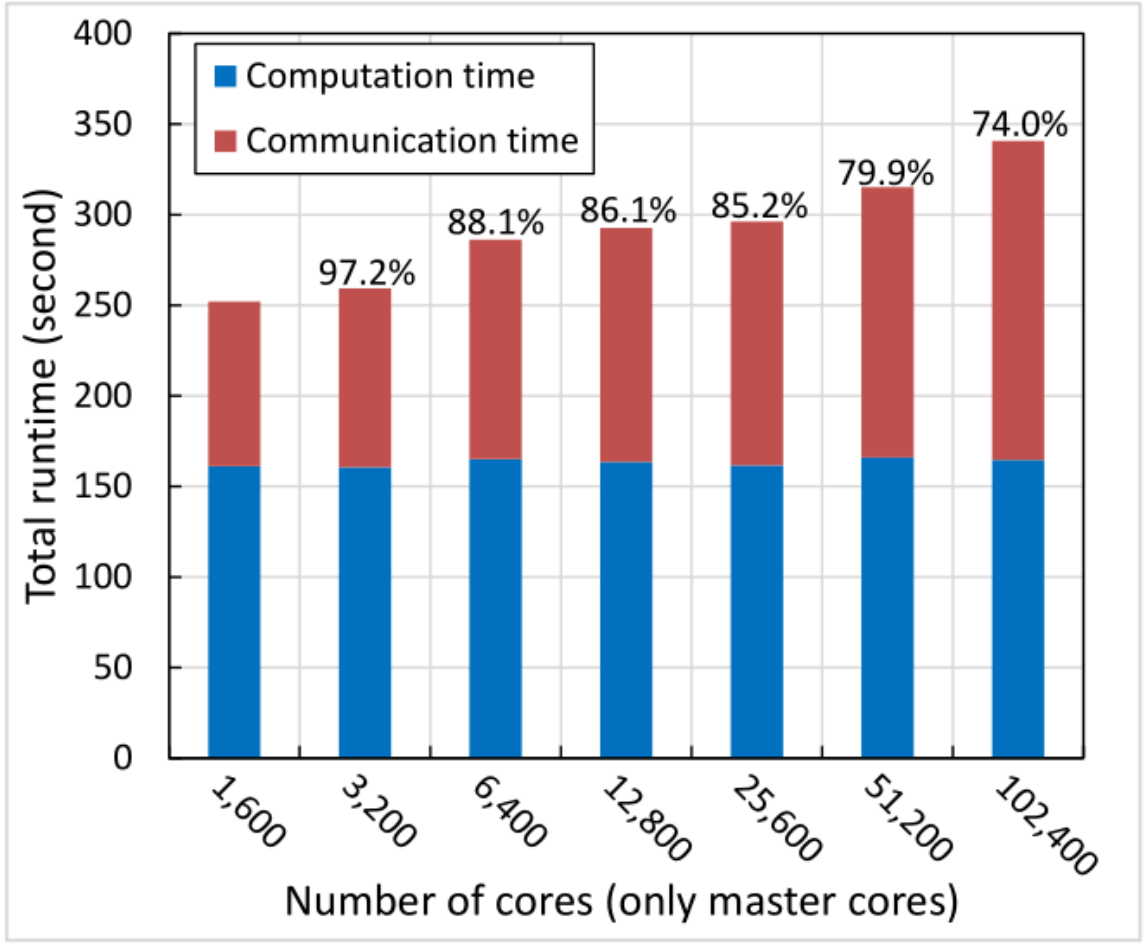
Weak scaling of the optimized MD

- ✓ We perform weak scaling experiments of MD simulation.
- ✓ As we increase the number of cores from 104,000 (1,600 master cores+1,024,000 slave cores) to 6,656,000 (102,400 master cores+6,553,600 slave cores), the problem size increases from 6.25∗10^10 to 4.0∗10^12.
- ✓ The experiment result indicates the MD code scaled up to 6 million cores by a 85% parallel efficiency.



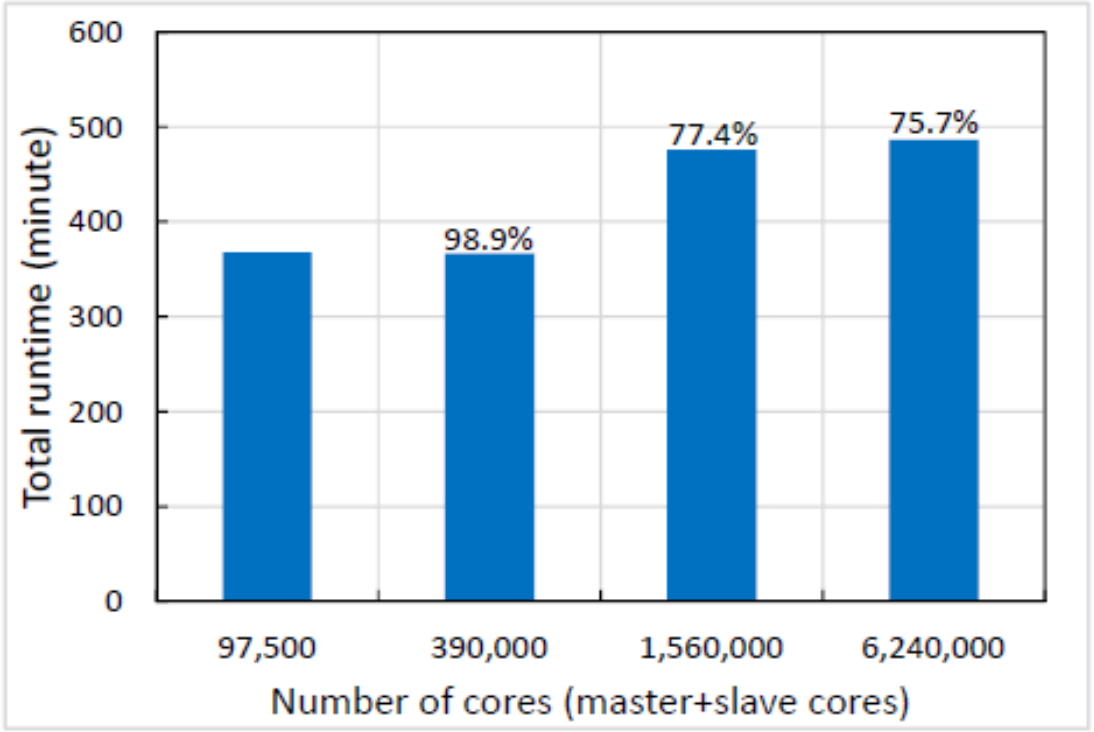
Weak scaling of the optimized KMC

- ✓ The Figure shows the weak scalability results ranging from 1,600 cores up to 102,400 cores.
- ✓ We observe that the change of computation time exhibits a nearly linear, and the communication time has increased somewhat.
- ✓ We also see that our optimized KMC code achieves 74% parallel efficiency over a 64-fold increase in sites size, from 1,600 cores to 102,400 cores.



Weak scaling of the coupled MD-KMC

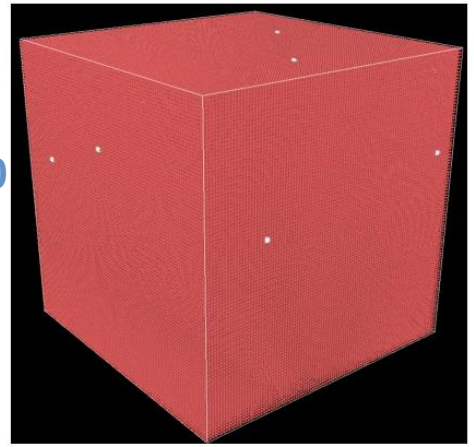
- ✓ Lastly, we study the weak scaling of the coupled MD-KMC with 3.3∗10^5 atoms per core group, and set the number of simulation atoms to 5.0∗10^8, 2.9∗10^9, 8.0∗10^9 and 3.2∗10^10 running on 97,500, 390,000, 1,560,000 and 6,240,000 cores.
- ✓ The coupled MD-KMC method achieves very good scalability, attaining 75.7% parallel efficiency.



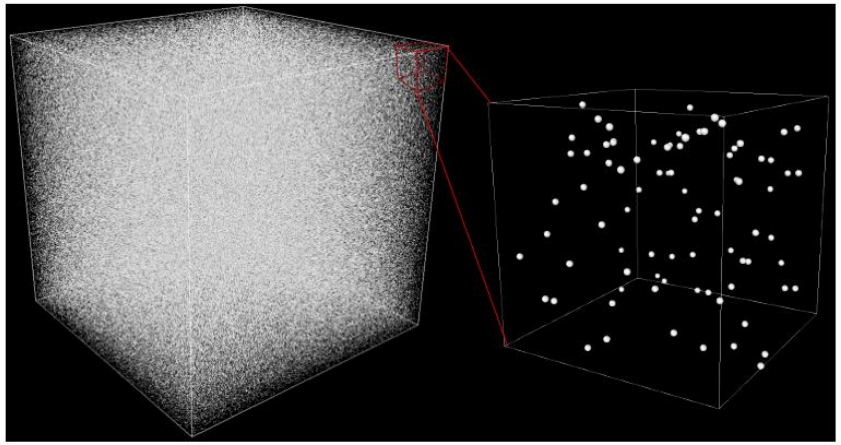
The simulation results

Furthermore, in order to demonstrate the ability of the coupled MD-KMC approach to perform large spatial scale and time scale on-lattice applications.

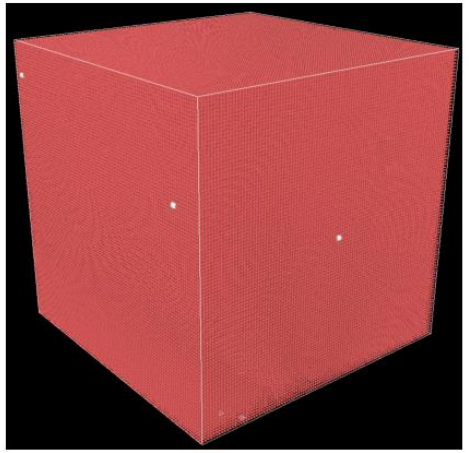
- ✓ **We conducted a simulation of microdamage evolution in Fe metal material with 3.2∗10^10 atoms on 6,240,000 cores in 19.2 days temporary scale.**
- ✓ The figure shows the distribution of Fe atoms (red) and vacancies (white) before and after KMC simulation.
- ✓ The evolution of the system after MD simulation is shown in Figure (a) and Figure (b).
- ✓ **We can see that after the KMC phase, the vacancies are relatively more aggregative and vacancy clusters are forming.**



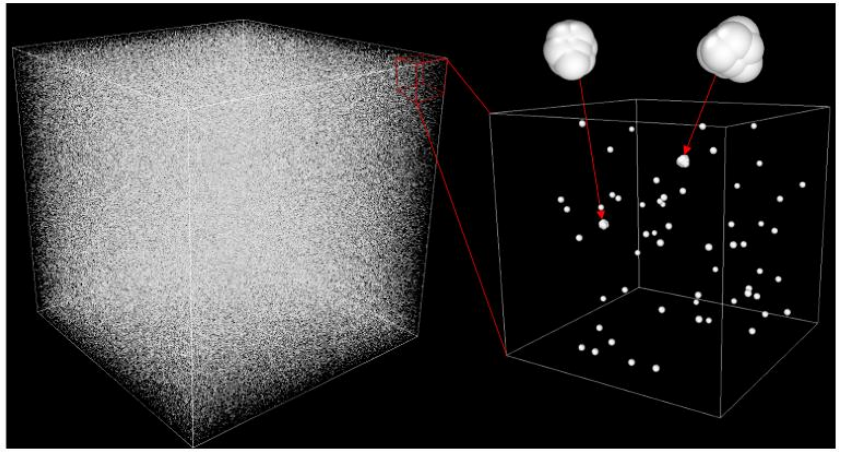
(a) The distribution of Fe atoms and vacancies after MD



(b) The distribution of vacancies after MD



(c) The distribution of Fe atoms and vacancies after KMC



(d) The partial enlarged view of vacancies after KMC

CONTENTS

Conclusion and Future work

1 Conclusion

- ✓ **We implement a scalable MD-KMC algorithm to simulate the microscopic damage of metal material (Fe) on Sunway TaihuLight supercomputer.**
- ✓ **We achieve excellent parallel scalability and high efficiency for both algorithms.**

2 Future work

- ✓ **To support multiple types of atoms, such as Fe, Cu, Mn, Ni.**
- ✓ **To extend our approach to future Exascale Supercomputers, and simulate in larger spatial-temporal scale for the coupled model.**

Thank You