

Parallelization of molecular dynamics

Jaewoon Jung

(RIKEN Advanced Institute for Computational Science)

Overview of MD

Molecular Dynamics (MD)

1. Energy/forces are described by classical molecular mechanics force field.
2. Update state according to equations of motion

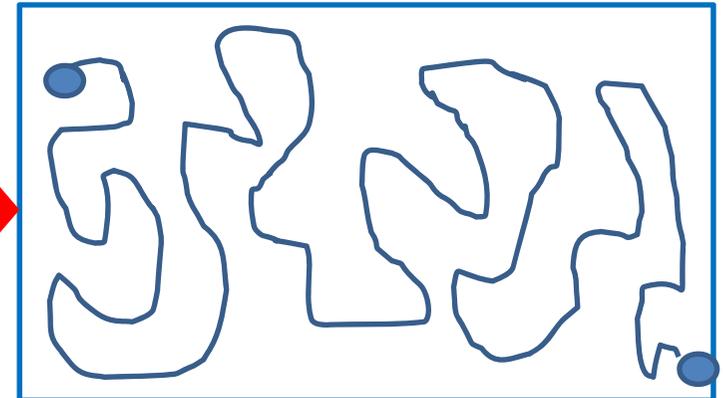
$$\frac{d\mathbf{r}_i}{dt} = \frac{\mathbf{p}_i}{m}$$

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{F}_i$$



$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \frac{\mathbf{p}_i}{m} \Delta t$$

$$\mathbf{p}_i(t + \Delta t) = \mathbf{p}_i(t) + \mathbf{F}_i \Delta t$$



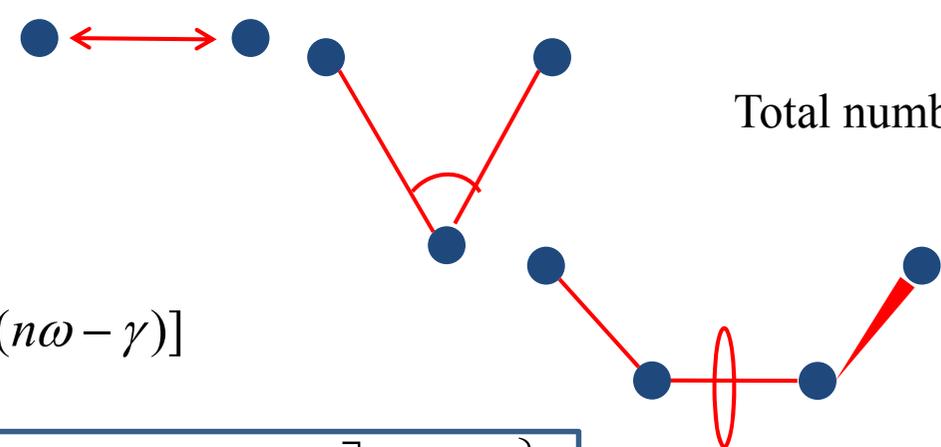
Equation of motion

Integration

Long time MD trajectory
=> Ensemble generation

Long time MD trajectories are important to obtain thermodynamic quantities of target systems.

Potential energy in MD

$$\begin{aligned}
 E_{\text{total}} = & \sum_{\text{bonds}} k_b (b - b_0)^2 && \text{Total number of particles } \mathbf{O(N)} \\
 & + \sum_{\text{angles}} k_a (\theta - \theta_0)^2 && \mathbf{O(N)} \\
 & + \sum_{\text{dihedrals}} V_n [1 + \cos(n\omega - \gamma)] && \mathbf{O(N)} \\
 & + \sum_{j=1}^{N-1} \sum_{i=j+1}^N \left\{ \varepsilon_{ij} \left[\left(\frac{r_{0ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{0ij}}{r_{ij}} \right)^6 \right] + \frac{q_i q_j}{r_{ij}} \right\} && \mathbf{O(N^2)} \\
 & && \text{Main bottleneck in MD}
 \end{aligned}$$


$$\sum_{|i-j| < R} \left\{ \varepsilon_{ij} \left[\left(\frac{r_{0ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{0ij}}{r_{ij}} \right)^6 \right] + \frac{q_i q_j \text{erfc}(\alpha r_{ij})}{r_{ij}} \right\} + \sum_{\mathbf{k} \neq 0} \frac{\exp(-\mathbf{k}^2 / 4\alpha^2)}{\mathbf{k}^2} \text{FFT}(Q(\mathbf{k}))$$

Real space, $O(CN)$

Reciprocal space, $O(N \log N)$

Non-bonded interaction

1. Non-bond energy calculation is reduced by introducing cutoff

$$\sum_{j=1}^{N-1} \sum_{i=j+1}^N \left\{ \epsilon_{ij} \left[\left(\frac{r_{0ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{0ij}}{r_{ij}} \right)^6 \right] + \frac{q_i q_j}{4\pi\epsilon r_{ij}} \right\} \quad O(N^2)$$

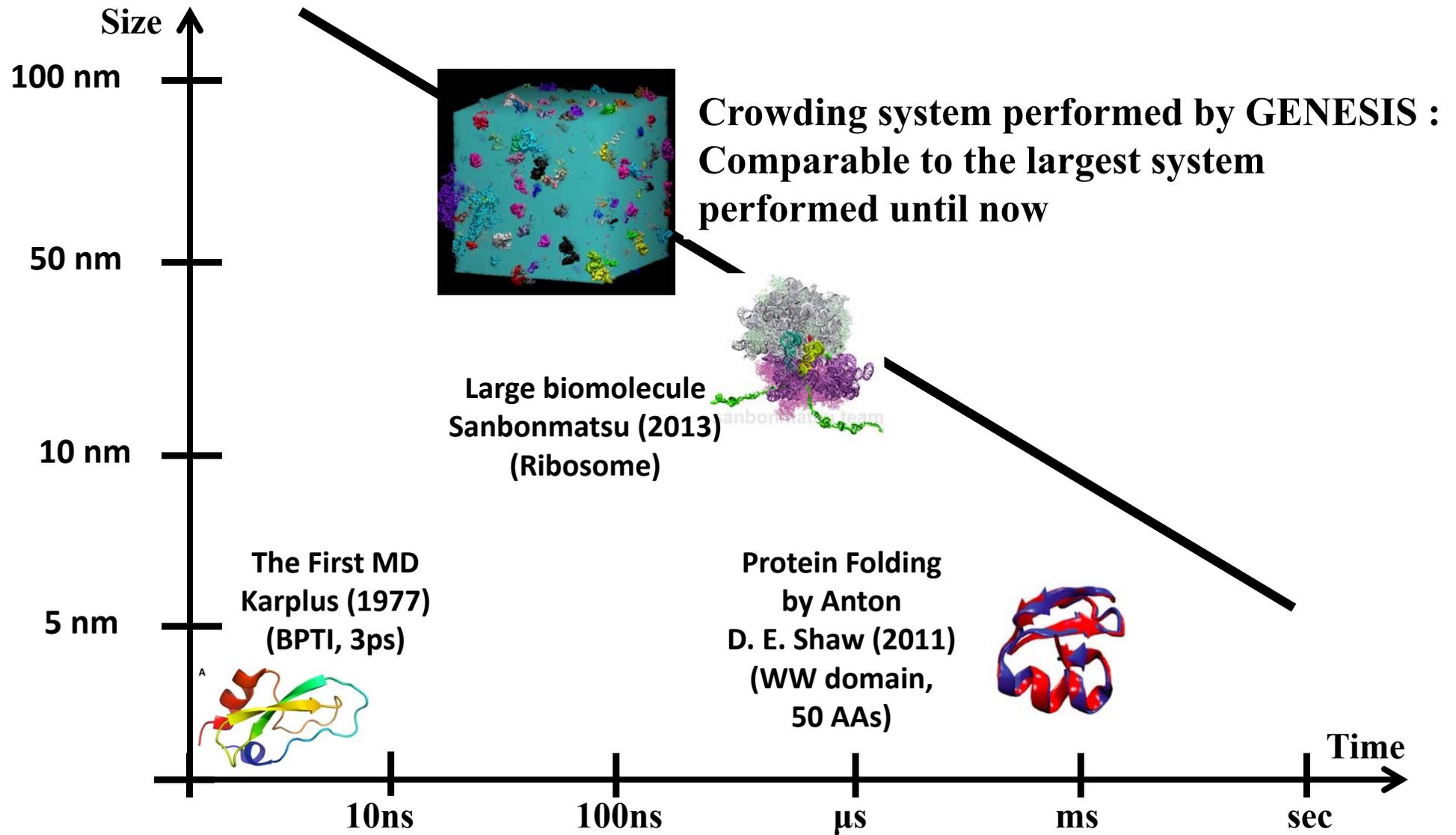
$$\sum_{|i-j| < R}^N \left\{ \epsilon_{ij} \left[\left(\frac{r_{0ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{0ij}}{r_{ij}} \right)^6 \right] \right\} + U_{elec} \quad O(N^1)$$

2. The electrostatic energy calculation beyond cutoff will be done in the reciprocal space with FFT

$$U_{elec} = \underbrace{\sum_{|i-j| < R} \frac{q_i q_j}{4\pi\epsilon_0} \frac{\text{erfc}(\alpha_{ij})}{r_{ij}}}_{\text{Real part}} + \underbrace{\frac{2\pi}{V} \sum_{\mathbf{G} \neq 0} \frac{\exp(-|\mathbf{G}|^2/4\alpha^2)}{|\mathbf{G}|^2} \sum_{ij} \frac{q_i q_j}{4\pi\epsilon_0} \cos(\mathbf{G} \cdot \mathbf{r}_{ij})}_{\text{Reciprocal part}} - \underbrace{\sum_i \frac{q_i q_i}{4\pi\epsilon_0} \frac{\alpha}{\sqrt{\pi}}}_{\text{Self energy}}$$

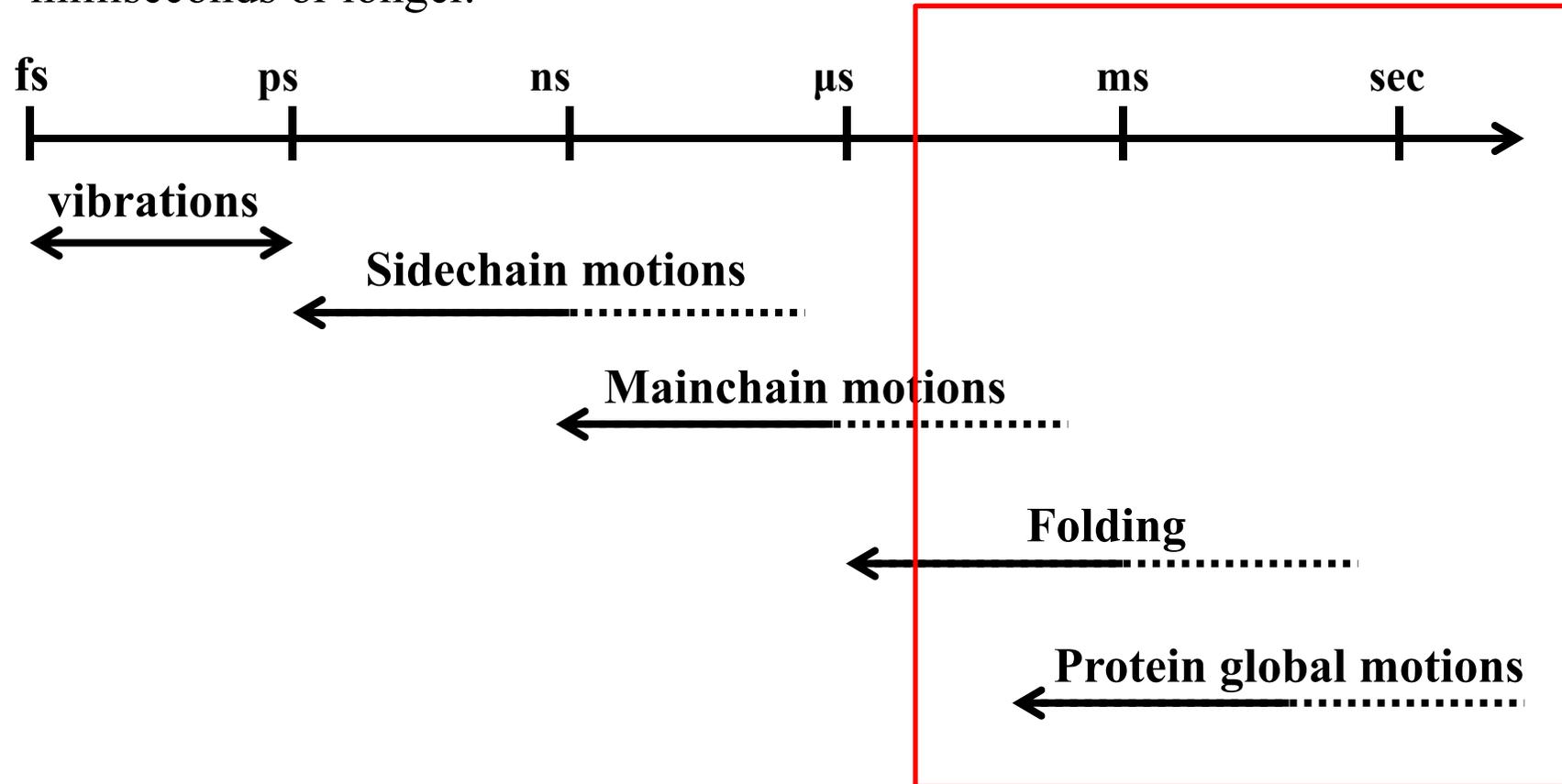
3. Further, it could be reduced by properly distributing over parallel processors, in particular good domain decomposition scheme.

Current MD simulations of biological systems



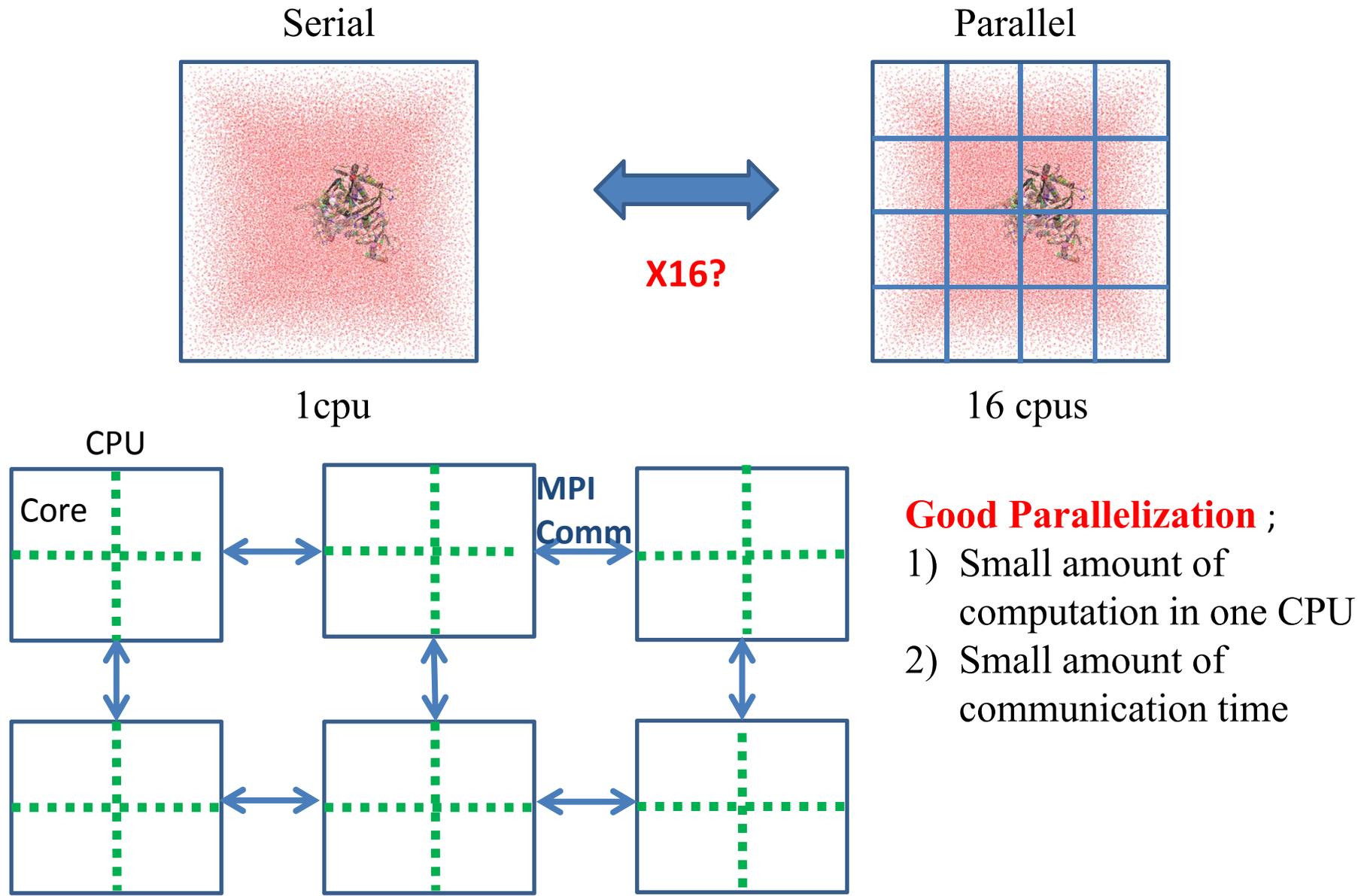
Difficulty to perform long time MD simulation

1. One time step length (Δt) is limited to 1-2 fs due to vibrations.
2. On the other hand, biologically meaningful events occur on the time scale of milliseconds or longer.



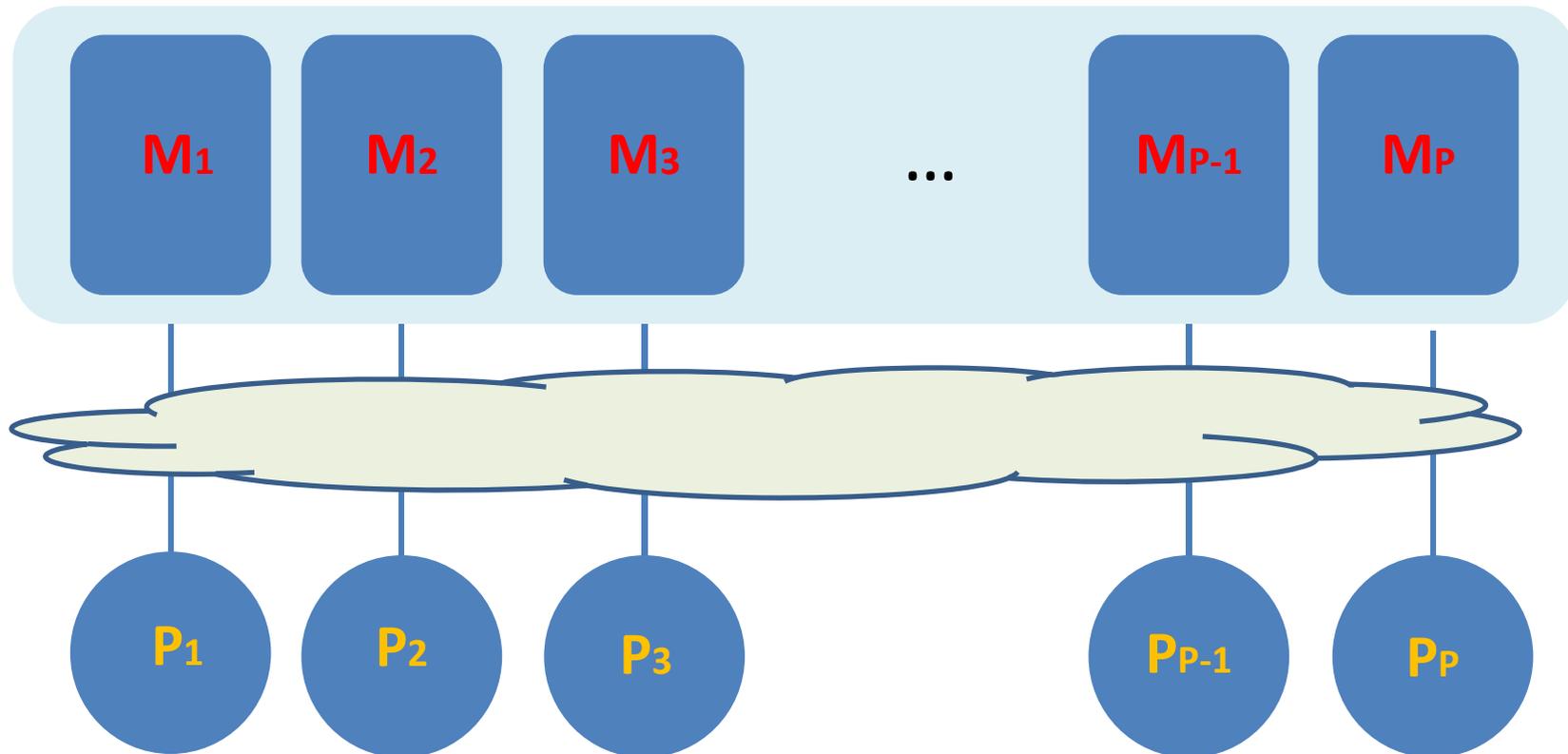
How to accelerate MD simulations?

=> Parallelization

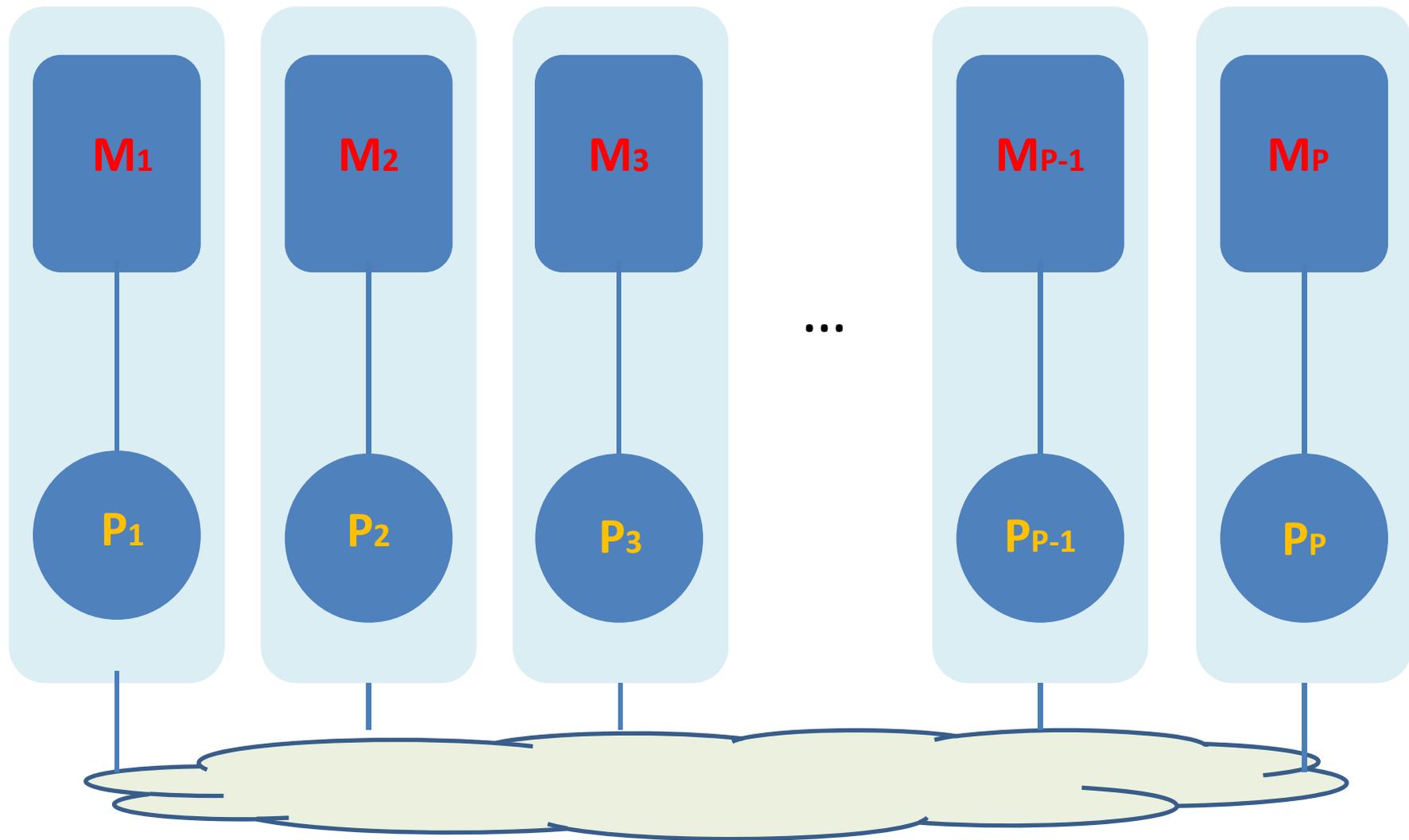


Overview of MPI and OpenMP parallelization

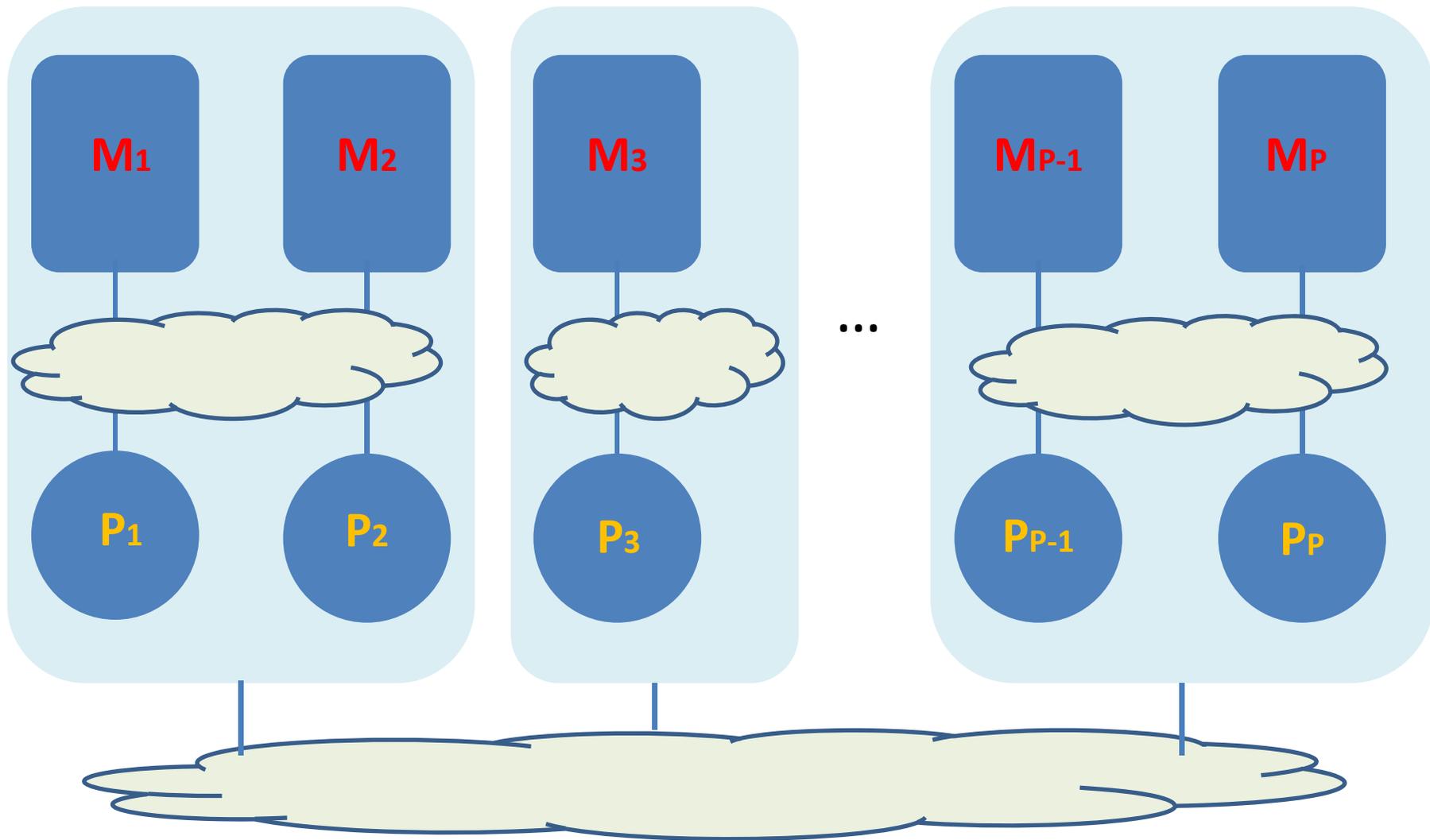
Shared memory parallelization (OpenMP)



Distributed memory parallelization (MPI)



Hybrid parallelization (MPI+OpenMP)



Parallelization of MD (real space)

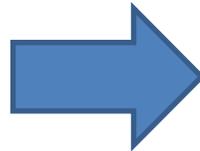
Parallelization scheme 1 : Replicated data approach

1. Each processor has a copy of all particle data.
2. Each processor works only part of the whole works by proper assign in do loops.

```
do i = 1, N
  do j = i+1, N

    energy(i,j)
    force(i,j)

  end do
end do
```



```
my_rank = MPI_Rank
proc = total MPI

do i = my_rank+1, N, proc
  do j = i+1, N

    energy(i,j)
    force(i,j)

  end do
end do

MPI reduction (energy,force)
```

Hybrid (MPI+OpenMP) parallelization of the Replicated data approach

1. Works are distributed over MPI and OpenMP threads.
2. Parallelization is increased by reducing the number of MPIs involved in communications.

```
my_rank = MPI_Rank
proc = total MPI

do i = my_rank+1, N, proc
  do j = i+1, N

    energy(i,j)
    force(i,j)

  end do
end do

MPI reduction
(energy,force)
```



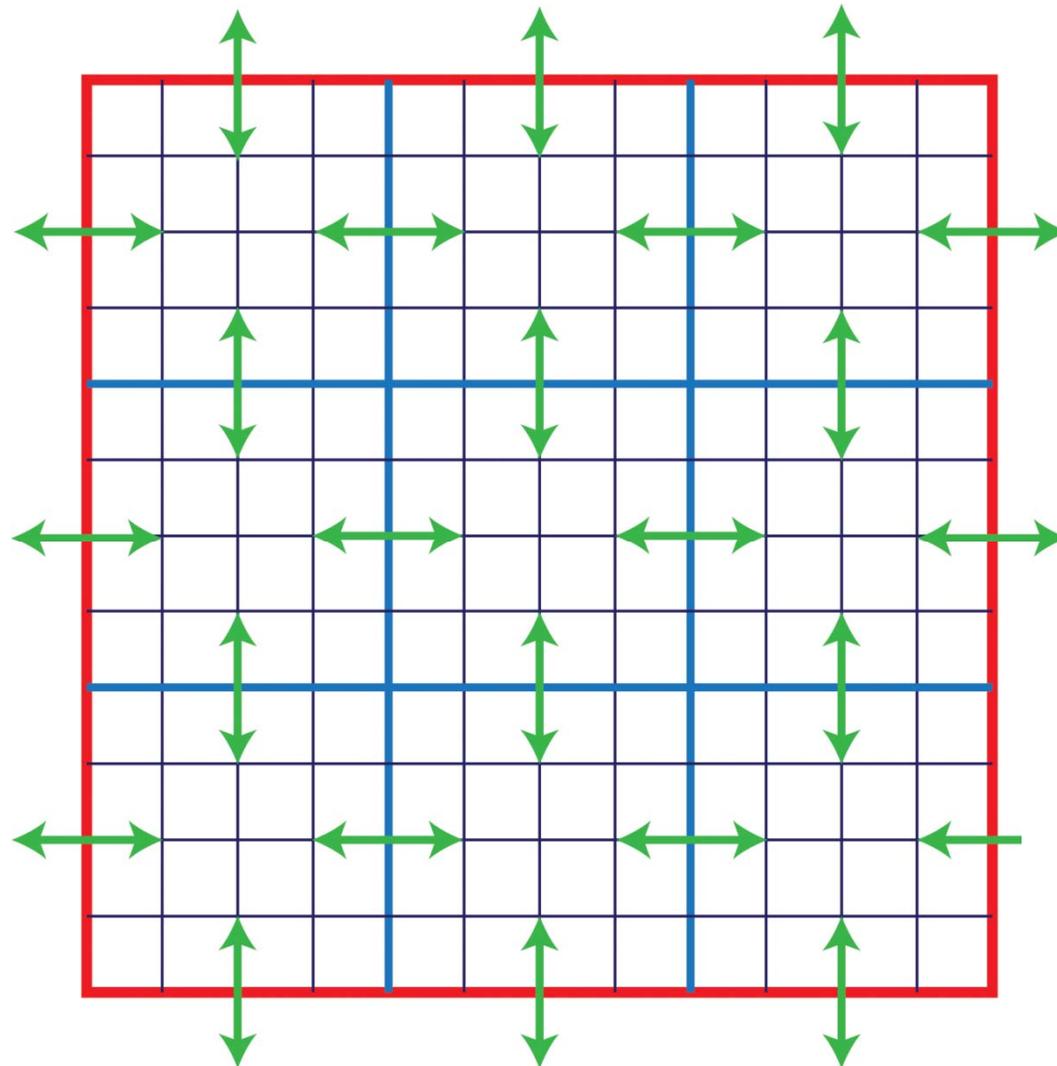
```
my_rank = MPI_Rank
proc = total MPI
nthread = total OMP thread
!$omp parallel
id = omp thread id
my_id = my_rank*nthread + id
do i = my_id+1,N,proc*nthread
  do j = i+1, N
    energy(i,j)
    force(i,j)
  end do
end do
Openmp reduciton
!$omp end parallel

MPI reduction (energy,force)
```

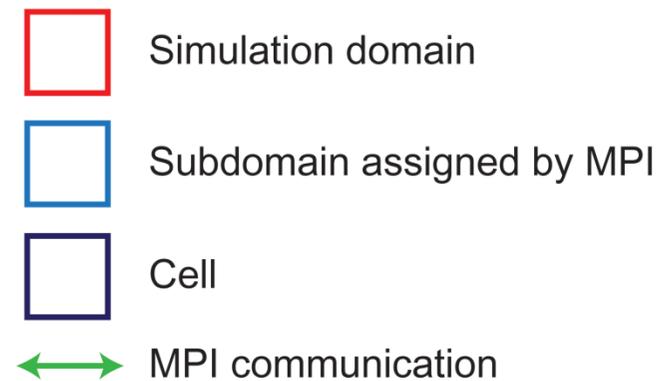
Advantage/Disadvantage of the Replicated data approach

- 1. Advantage** : easy to implement
- 2. Disadvantage**
 - 1) Parallel efficiency is not good
 - a) Load imbalance
 - b) Communication is not reduced by increasing the number of processors
 - 2) Needs a lot of memory
 - b) Memory usage is independent of the number of processors

Parallelization scheme 2 : Domain decomposition



1. The simulation space is divided into subdomains according to MPI
2. Domain size is usually equal to or greater than the cutoff value.



Advantage/Disadvantage of the domain decomposition approach

1. Advantage

- 1) Parallel efficiency is very good compared to the replicated data method due to small communicational cost
- 2) The amount of memory is reduced by increasing the number of processors

2. Disadvantage

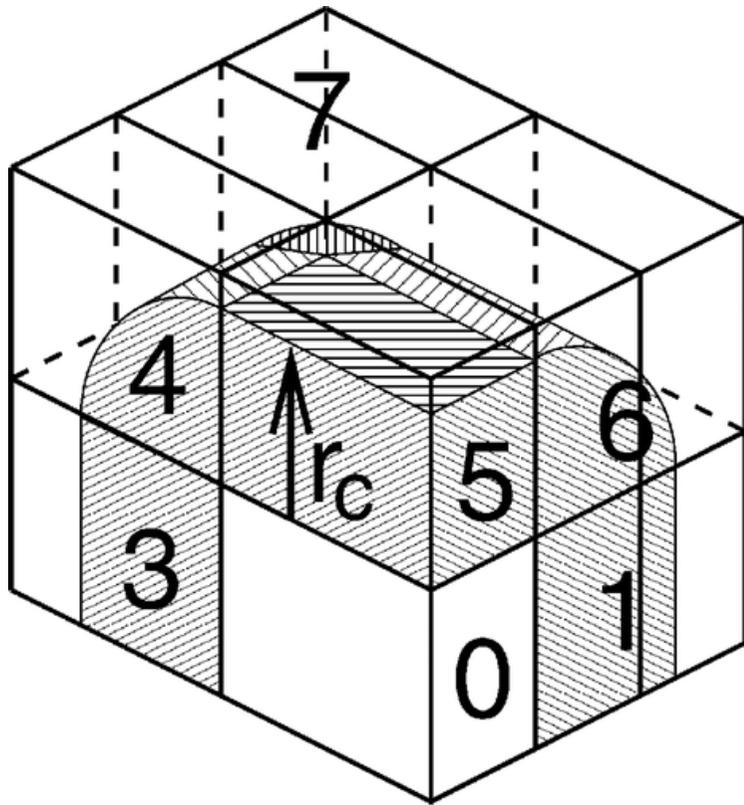
- 1) Difficult to implement

Comparison of two parallelization scheme

	Computation	Communication	Memory
Replicated data	$O(N/P)$	$O(N)$	$O(N)$
Domain decomposition	$O(N/P)$	$O((N/P)^{2/3})$	$O(N/P)$

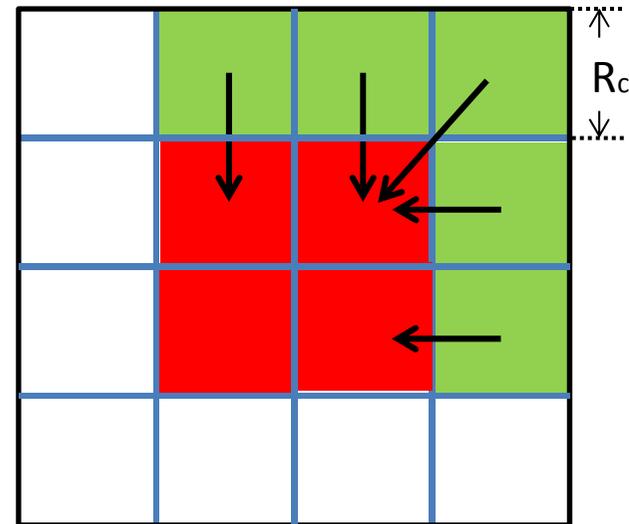
Domain decomposition of existing MD :

(1) Gromacs



Coordinates in zones 1 to 7 are communicated to the corner cell 0

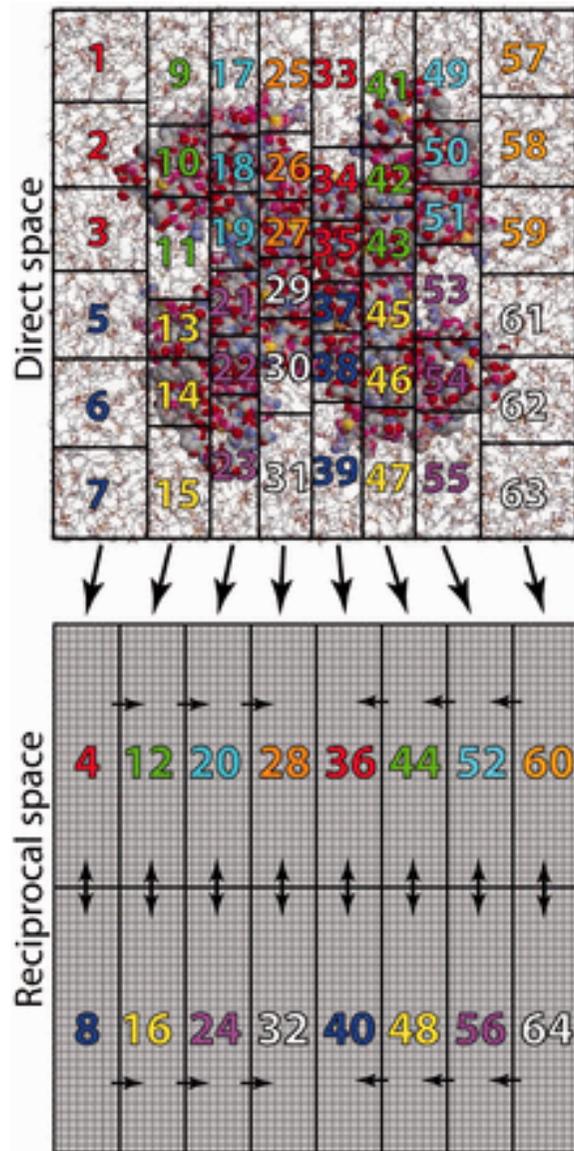
1. Gromacs makes use of the 8-th shell scheme as the domain decomposition scheme



8th shell scheme

Domain decomposition of existing MD :

(1) Gromacs



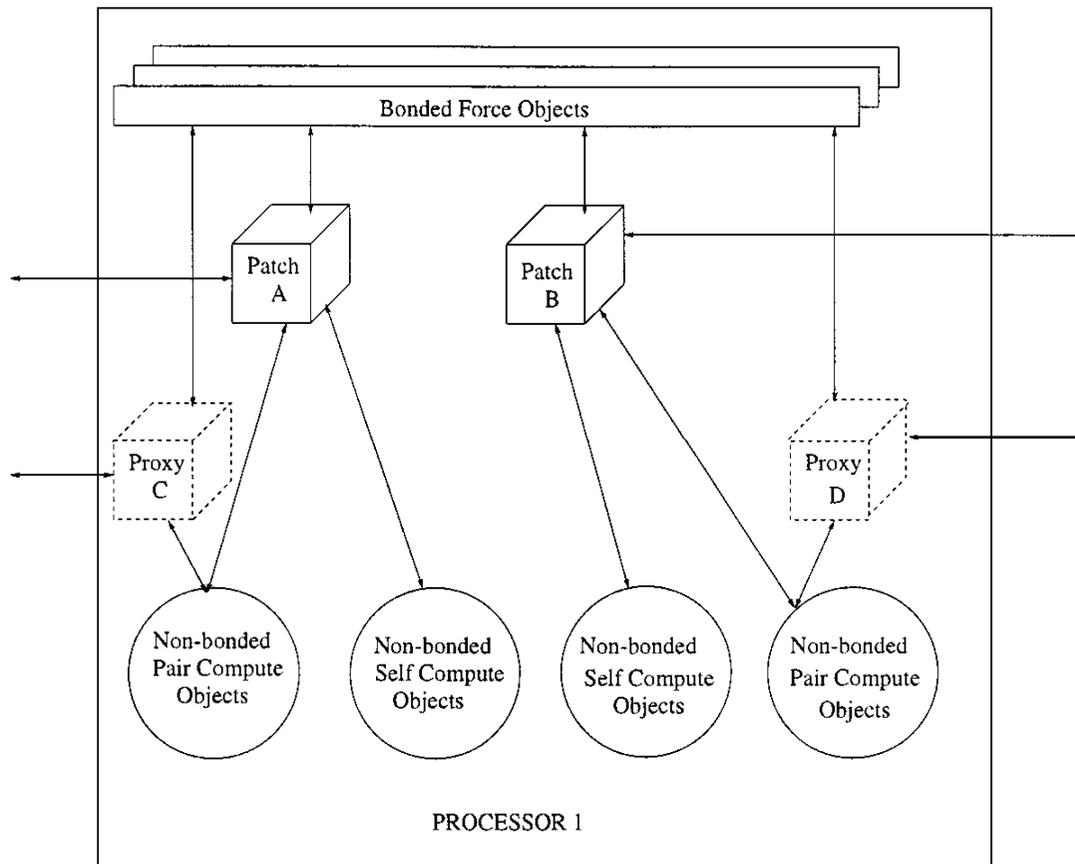
2. Multiple-Program, Multiple-Data PME parallelization

- 1) A subset of processors are assigned for reciprocal space PME
- 2) Other processors are assigned for real space and integration

Domain decomposition of existing MD :

(2) NAMD

1. NAMD is based on the Charmm++ parallel program system and runtime library.
2. Subdomains named patch are decided according to MPI
3. Forces are calculated by independent compute objects

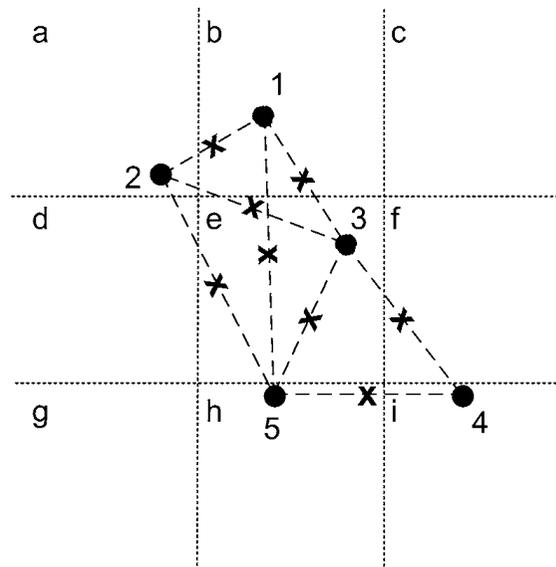


Ref : L. Kale et al. J. Comput. Chem. 151, 283 ((1999))

Domain decomposition of existing MD :

(3) Desmond – midpoint method

1. Two particles interact on a particular box if and only if the midpoint of the segment connecting them falls within the region of space associated with that box



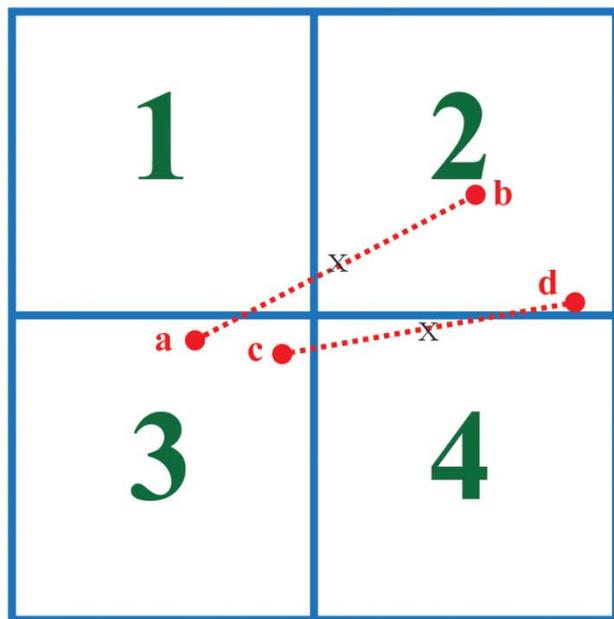
Each pair of particles separated by a distance less than R (cutoff distance) is connected by a dashed line segment, with “x” at its center lying in the box which will compute the interaction of that pair

2. This scheme applies not only for non-bonded but also bonded interactions.

Domain decomposition of existing MD :

(4) GENESIS – midpoint cell method

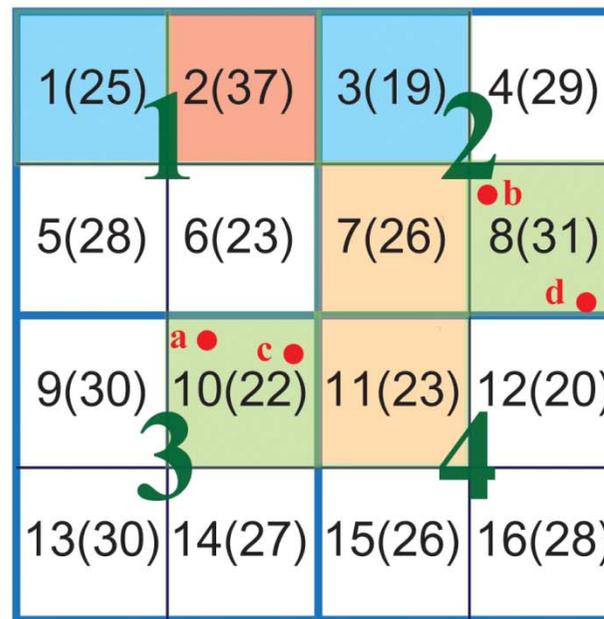
(a)



Midpoint method : interaction between two particles are decided from the midpoint position of them.

Small communication, efficient energy/force evaluations

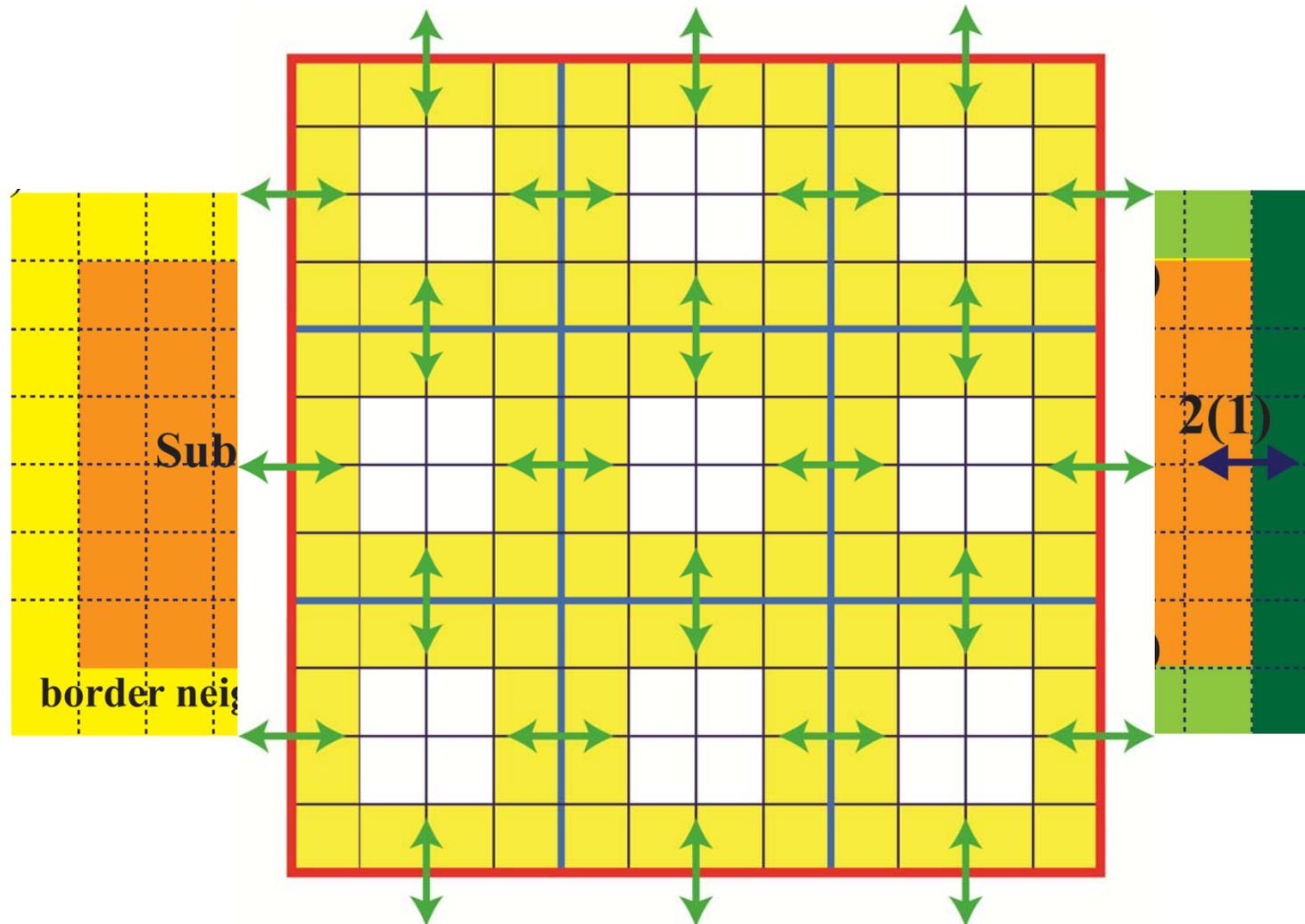
(b)



Midpoint cell method : interaction between two particles are decided from the midpoint cells where each particle resides.

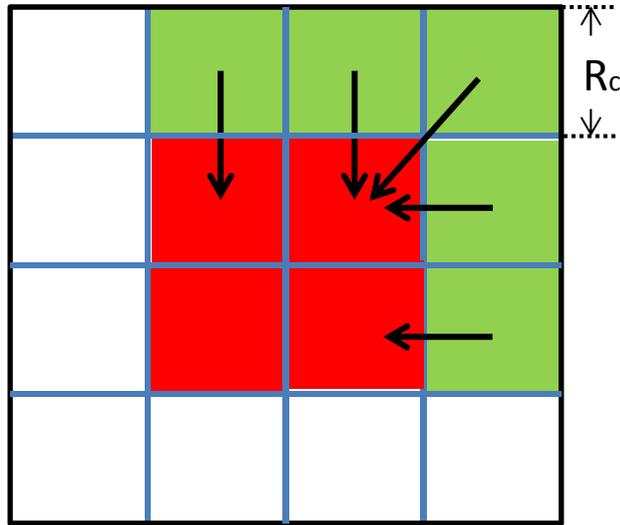
Ref : J. Jung et al, J. Comput. Chem. 35, 1064 (2014)

Communication space

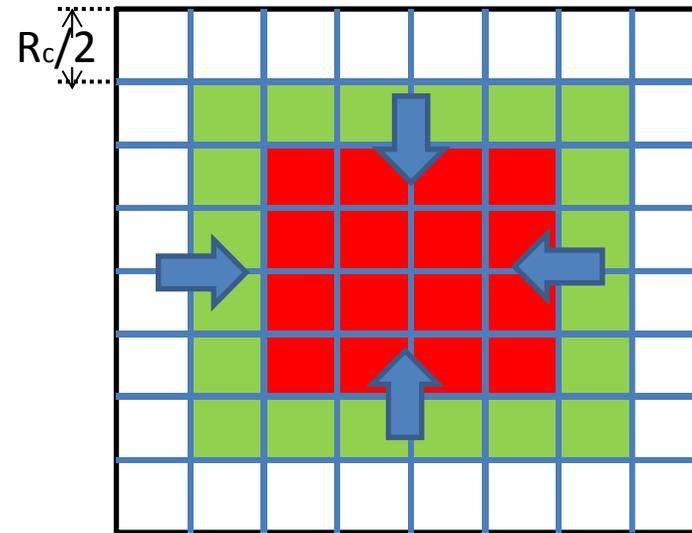


Communicate only with neighboring cells to minimize communication

Eighth shell v.s. Midpoint (cell)



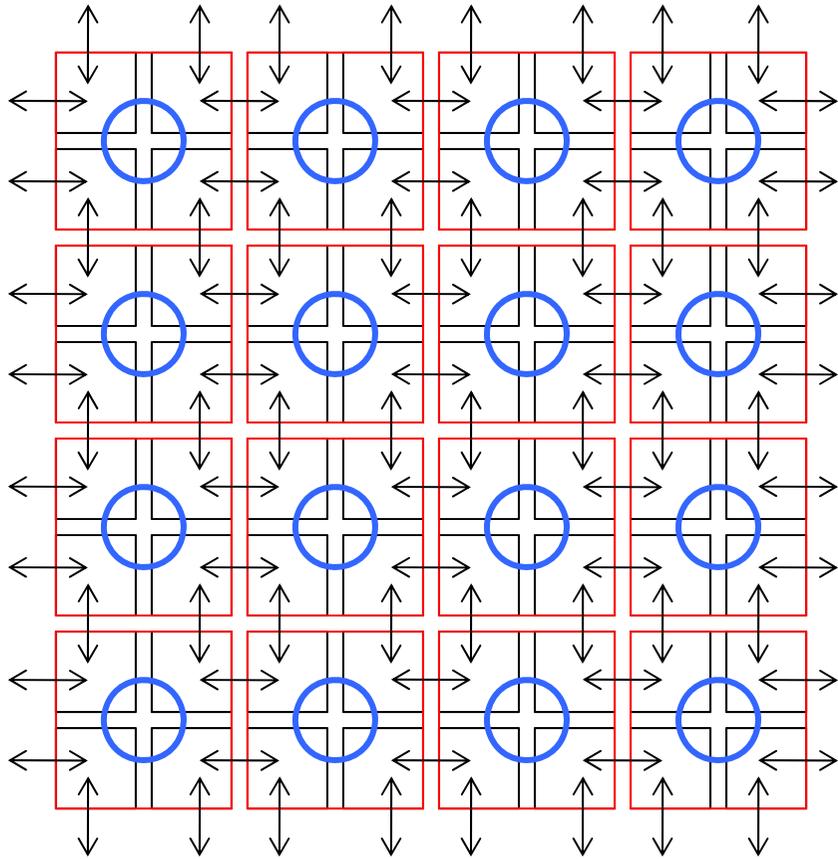
Eighth shell



Midpoint

1. Midpoint (cell) and eighth shell method have the same amount of communication (import volumes are identical to each other)
2. Midpoint (cell) method could be move better than the eighth shell for certain communication network topologies like the toroidal mesh.
3. Midpoint (cell) could be more advantageous considering cubic decomposition parallelization for FFT (will be explained after)

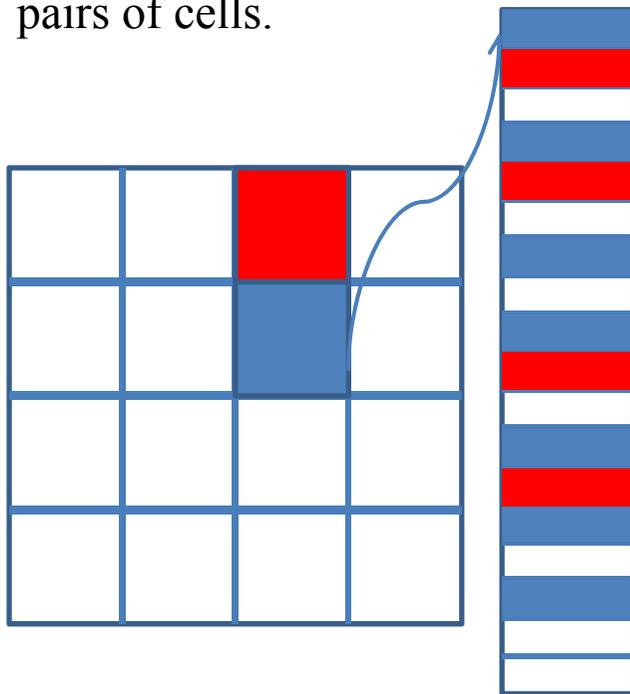
Hybrid parallelization in GENESIS



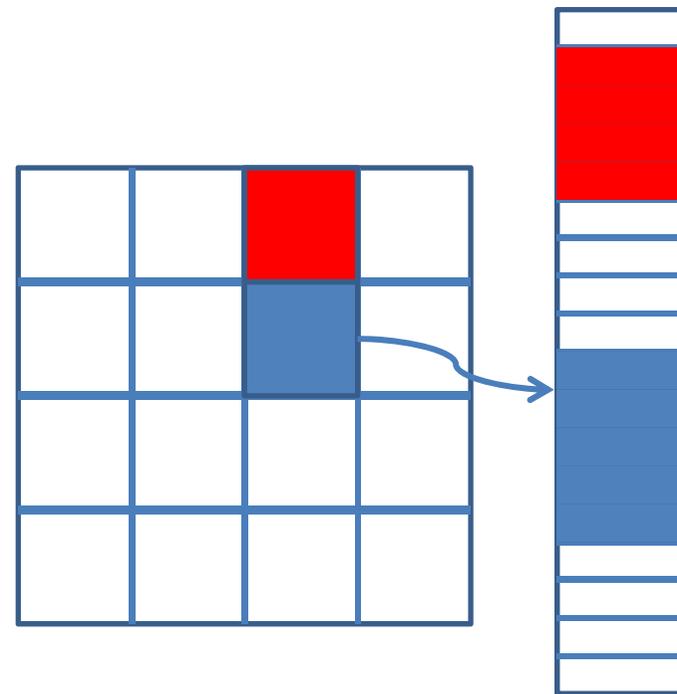
1. The basic is the Domain decomposition with the midpoint scheme
2. Each node consists of at least two cells in each direction.
3. Within nodes, thread calculation (OPEN MP) is used for parallelization and communication is not necessary
4. Only for different nodes, we allow point to point communication (MPI) for neighboring domains.

Efficient shared memory calculation =>Cell-wise particle data

1. Each cell contains an array with the data of the particles that reside within it
2. This improves the locality of the particle data for operations on individual cells or pairs of cells.



particle data in traditional cell lists



cell-wise arrays of particle data

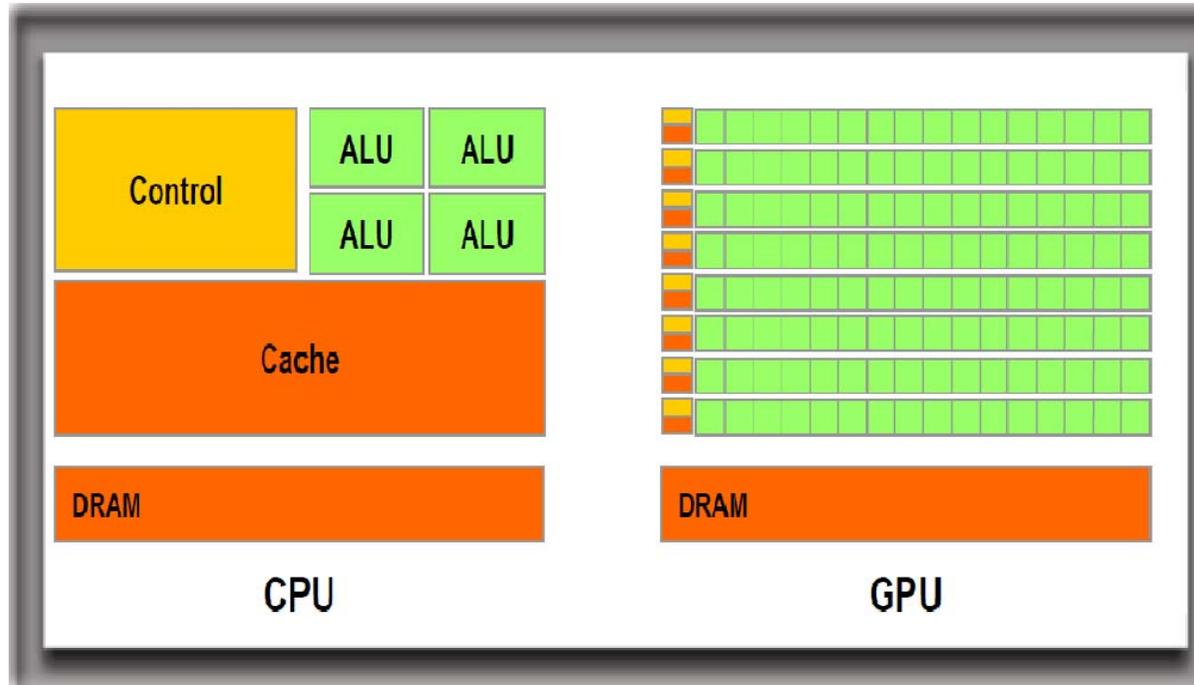
How to parallelize by OPENMP?

1. In the case of integrator, every cell indices are divided according to the thread id.
2. As for the non-bond interaction, cell-pair lists are first identified and cell-pair lists are distributed to each thread

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(1,2)	← thread1
(1,5)	← thread2
(1,6)	← thread3
(2,3)	← thread4
(2,5)	← thread1
(2,6)	← thread2
(2,7)	← thread3
(3,4)	← thread4
(3,6)	...
(3,7)	
(3,8)	
(4,7)	
(4,8)	
(5,6)	
(5,9)	
(5,10)	
(6,7)	
(6,10)	
...	

Recent trend of MD : CPU => GPU



[Image is from informaticamente.over-blog.it](http://informaticamente.over-blog.it)

1. A CPU core can execute **4 or 8** 32-bit instructions per clock, whereas a GPU can execute **much more (>3200)** 32-bit instructions per clock.
 2. Unlike CPUs, GPUs have a parallel throughput architecture that emphasizes executing many concurrent threads slowly, rather than executing a single thread very quickly.
- => **Parallelization on GPU is also important.**

Parallelization of MD (reciprocal space)

Smooth particle mesh Ewald method

$$E(\mathbf{r}) = \underbrace{\frac{1}{2} \sum_{\mathbf{n}'} \sum_{\substack{i,j=1 \\ |\mathbf{r}_i - \mathbf{r}_j + \mathbf{n}'| < r_c}}^N \frac{q_i q_j \operatorname{erfc}(\alpha |\mathbf{r}_i - \mathbf{r}_j + \mathbf{n}'|)}{|\mathbf{r}_i - \mathbf{r}_j + \mathbf{n}'|}}_{\text{Real part}} + \underbrace{\frac{1}{2\pi V} \sum_{\mathbf{k} \neq 0} \frac{\exp(-\pi^2 \mathbf{k}^2 / \alpha^2)}{\mathbf{k}^2} |S(\mathbf{k})|^2}_{\text{Reciprocal part}} - \underbrace{\frac{\alpha}{\sqrt{\pi}} \sum_{i=1}^N q_i^2}_{\text{Self energy}}$$

The structure factor in the reciprocal part is approximated as

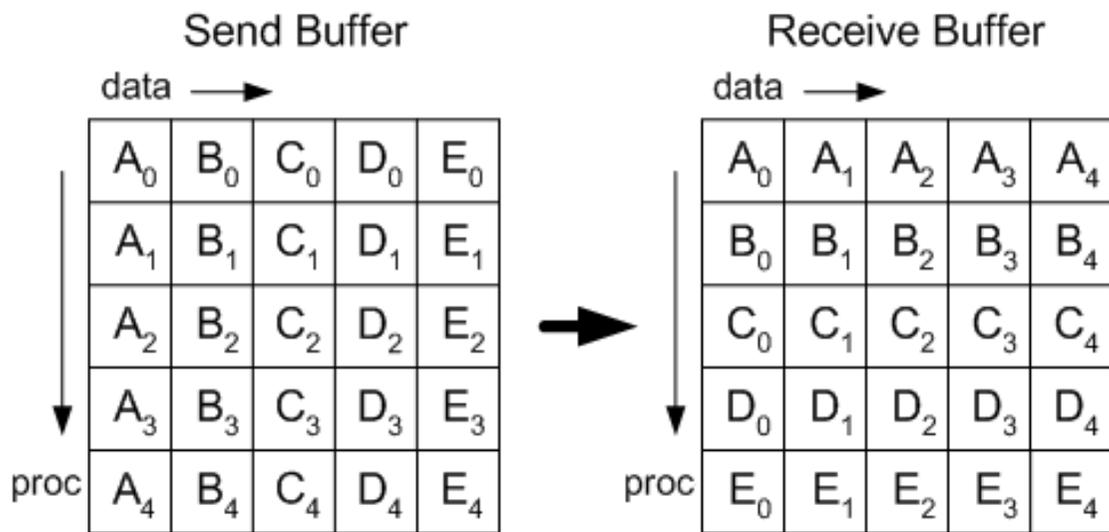
$$S(k_1, k_2, k_3) = b_1(k_1) b_2(k_2) b_3(k_3) \boxed{F(Q)}(k_1, k_2, k_3)$$

Using Cardinal B-splines of order n

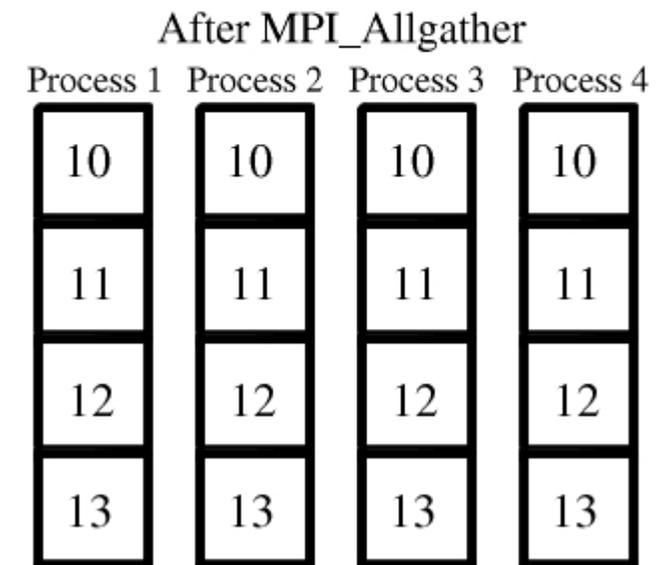
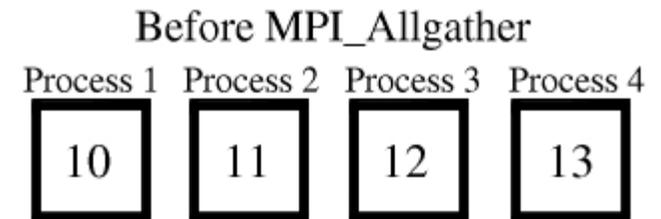
Fourier Transform of charge

It is important to parallelize the Fast Fourier transform efficiently in PME!!

Simple note of MPI_alltoall and MPI_allgather communications

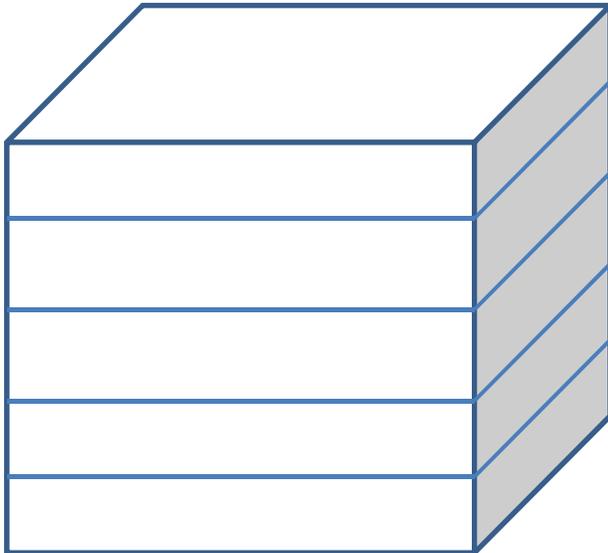


MPI_alltoall



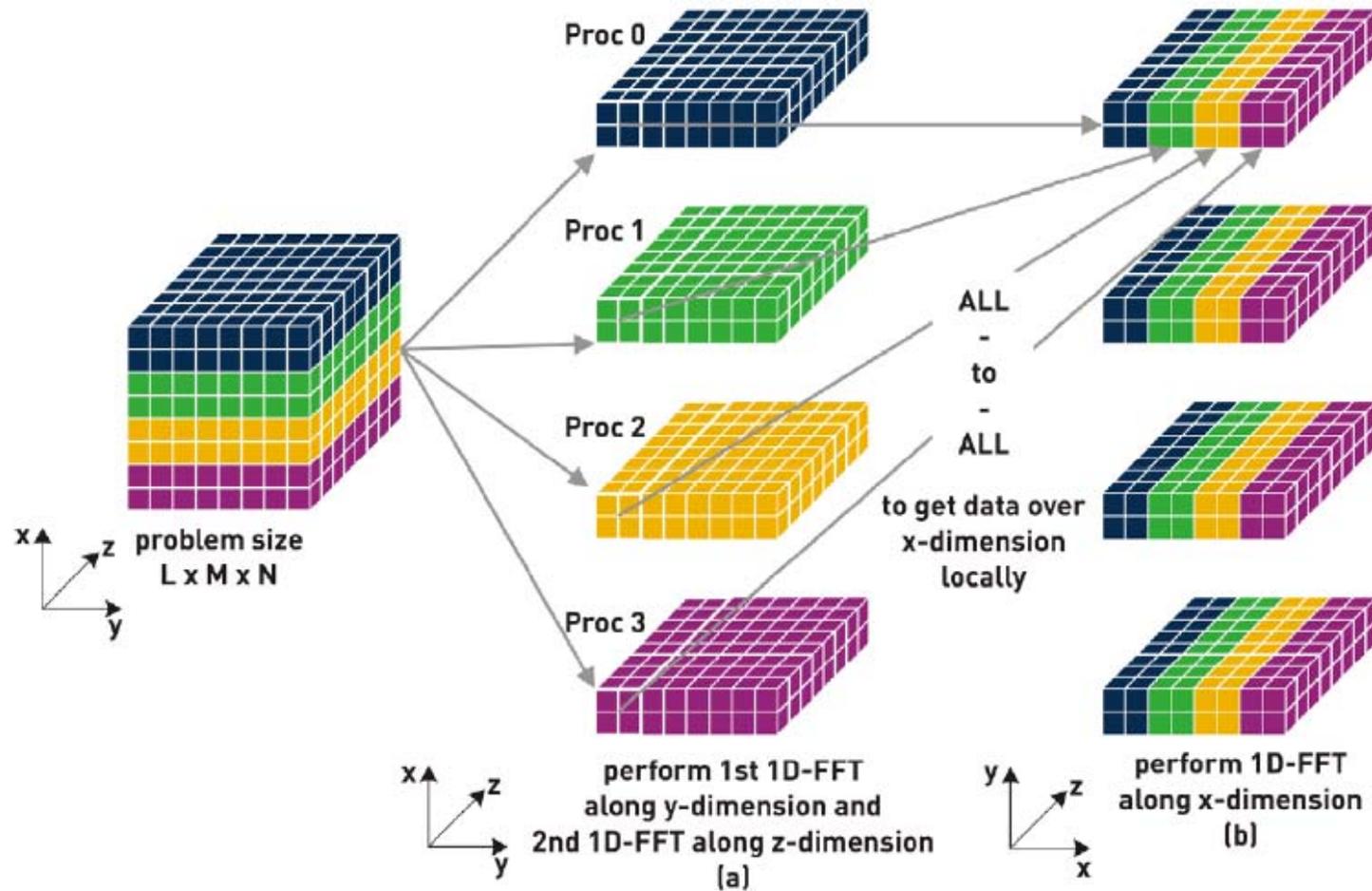
MPI_allgather

Parallel 3D FFT – slab(1D) decomposition



1. Each processor is assigned a slab of size $N \times N \times N/P$ for computing an $N \times N \times N$ FFT on P processors.
2. The parallel scheme of FFTW
3. Even though it is easy to implement, the scalability is limited by N , the extent of the data along a single axis
4. In the case of FFTW, N should be divisible by P

1D decomposition of 3D FFT

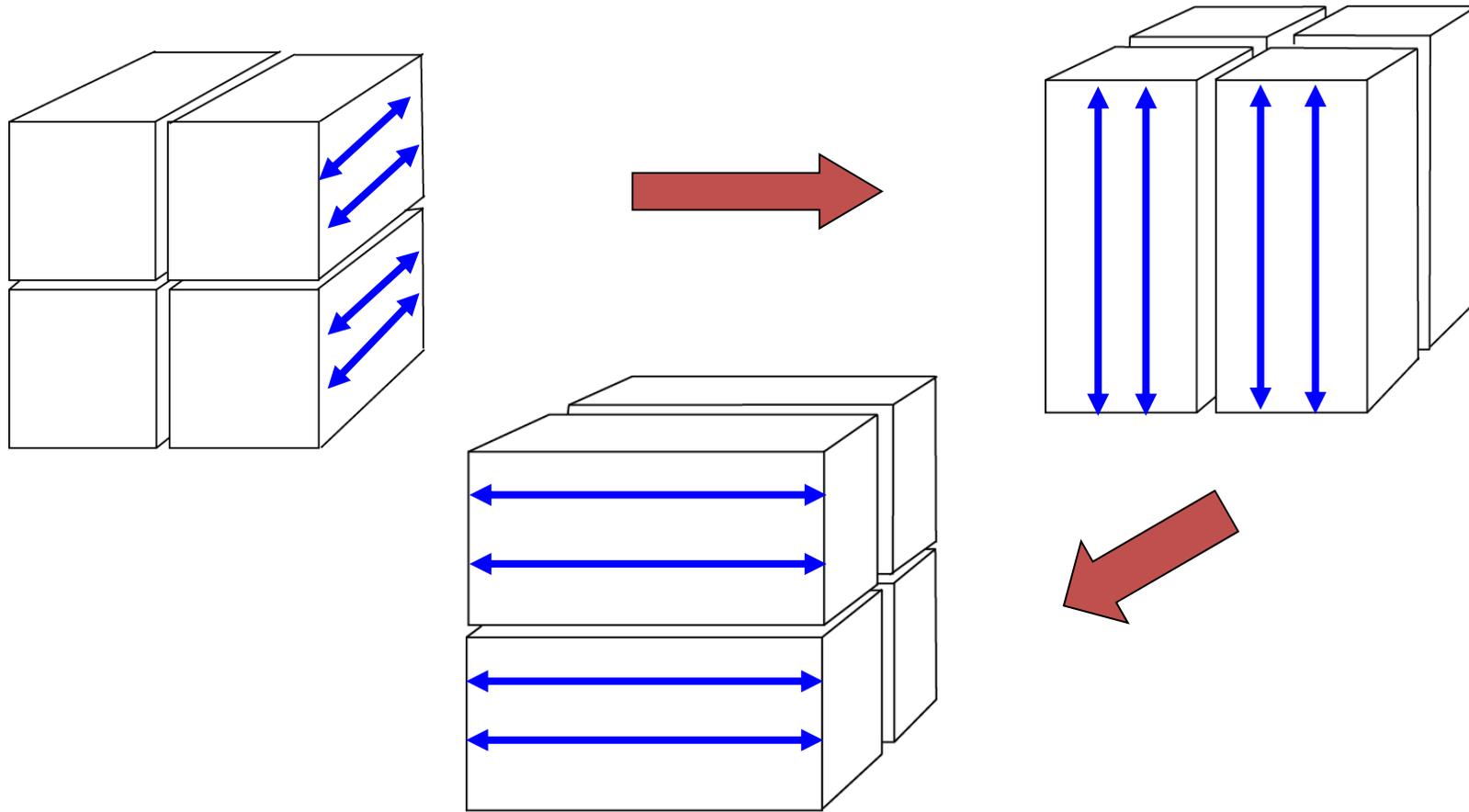


Reference: H. Jagode. Master's thesis, The University of Edinburgh, 2005

1D decomposition of 3D FFT (continued)

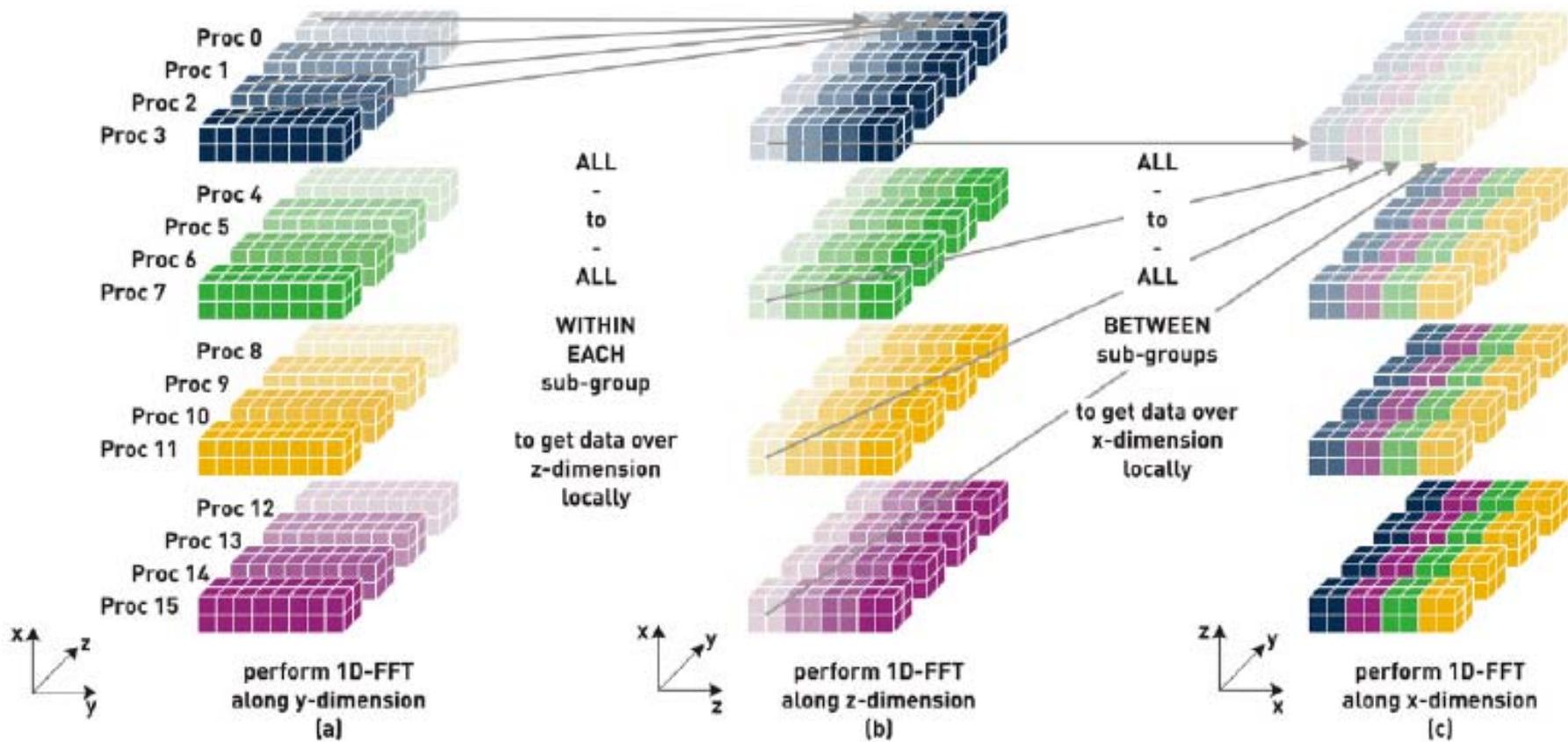
1. Slab decomposition of 3D FFT has three steps
 - 2D FFT (or two 1D FFT) along the two local dimension
 - Global transpose
 - 1D FFT along third dimension
2. Advantage : The fastest on limited number of processors because it only needs one global transpose
3. Disadvantage : Maximum parallelization is limited to the length of the largest axis of the 3D data (The maximum parallelization can be increased by using a hybrid method combining 1D decomposition with a thread based parallelization)

Parallel 3D FFT –2D decomposition



1. Each processor is assigned a slob of size $N \times N/P \times N/Q$ for computing an $N \times N \times N$ FFT on $P \times Q$ processors.
2. Current GENESIS adopt this scheme with 1D FFTW

2D decomposition of 3D FFT



Reference: H. Jagode. Master's thesis, The University of Edinburgh, 2005

2D decomposition of 3D FFT (continued)

1. 2D decomposition of 3D FFT has five steps
 - 1D FFT along the local dimension
 - Global transpose
 - 1D FFT along the second dimension
 - Global transpose
 - 1D FFT along the third dimension
 - Global transpose
2. The global transpose requires communication only between subgroups of all nodes
3. Disadvantage : Slower than 1D decomposition for a number of processors possible with 1D decomposition
4. Advantage : Maximum parallelization is increased
5. Program with this scheme
 - Parallel FFT package by Steve Plimpton (Using MPI_Send and MPI_Irecv)[1]
 - FFTE by Daisuke Takahashi (Using MPI_AlltoAll)[2]
 - P3DFFT by Dmitry Pekurovsky (Using MPI_Alltoallv)[3]

[1] <http://www.sandia.gov/~sjplimp/docs/fft/README.html>

[2] <http://www.ffte.jp>

[3] <http://www.sdsc.edu/us/resources/p3dfft.php>

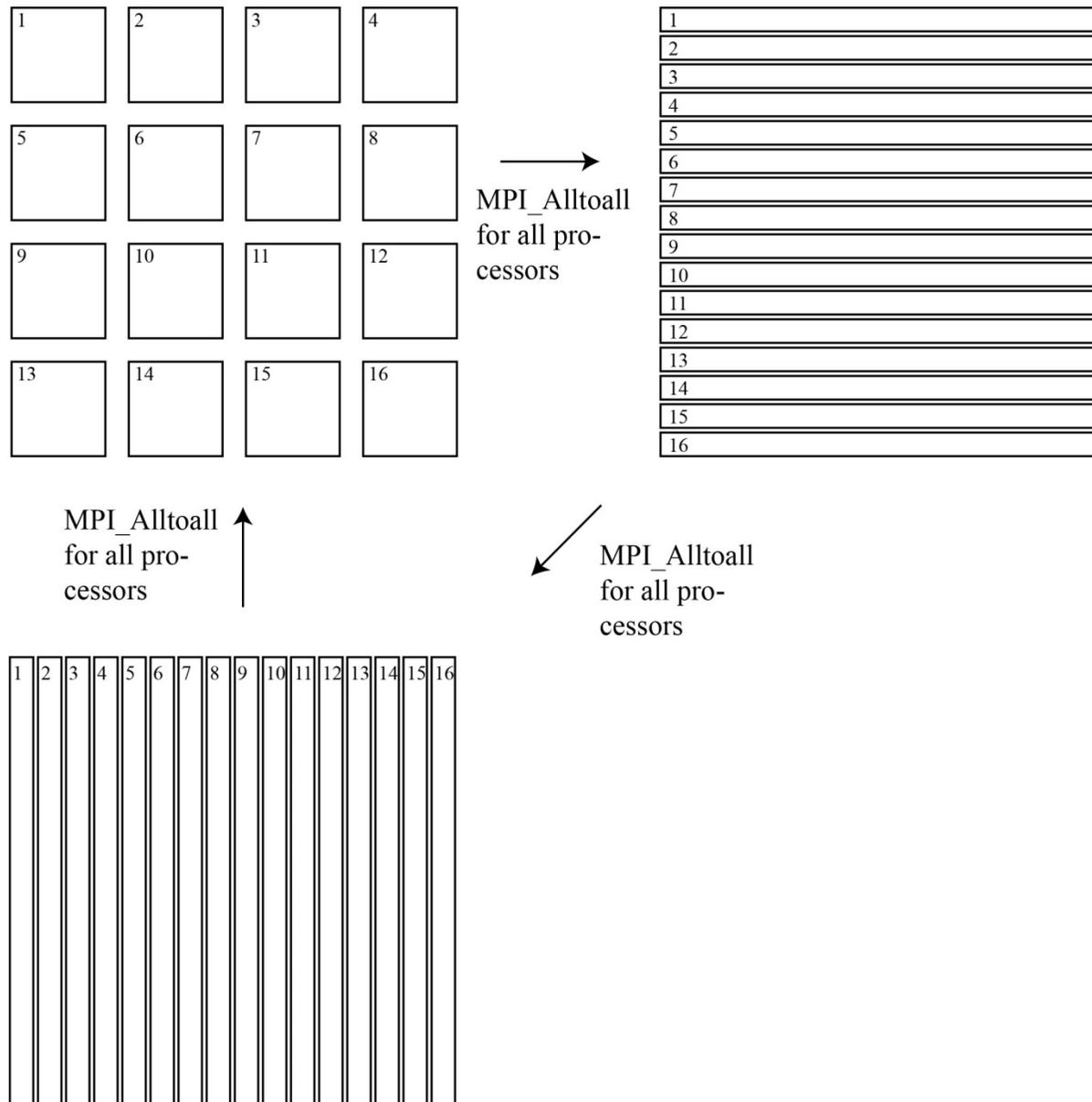
2D decomposition of 3D FFT (pseudo-code)

```
! compute Q factor
do i = 1, natom/P
  compute Q_orig
end do
call mpi_alltoall(Q_orig, Q_new, ...)
accumulate Q from Q_new
!FFT : F(Q)
do iz = 1, zgrid(local)
  do iy = 1, ygrid(local)
    work_local = Q(my_rank)
    call fftw(work_local)
    Q(my_rank) = work_local
  end do
end do
call mpi_alltoall(Q, Q_new,...)
do iz = 1, zgrid(local)
  do ix = 1, xgrid(local)
    work_local = Q_new(my_rank)
    call fftw(work_local)
    Q(my_rank) = work_local
  end do
end do
call mpi_alltoall(Q,Q_new,..)
```

```
do iy = 1, ygrid(local)
  do ix = 1, xgrid(local)
    work_local = Q(my_rank)
    call fftw(work_local)
    Q(my_rank) = work_local
  end do
end do
! compute energy and virial
do iz = 1, zgrid
  do iy = 1, ygrid(local)
    do ix = 1, xgrid(local)
      energy = energy + sum(Th*Q)
      virial = viral + ..
    end do
  end do
end do
! X=F_1(Th)*F_1(Q)
! FFT (F(X))
```

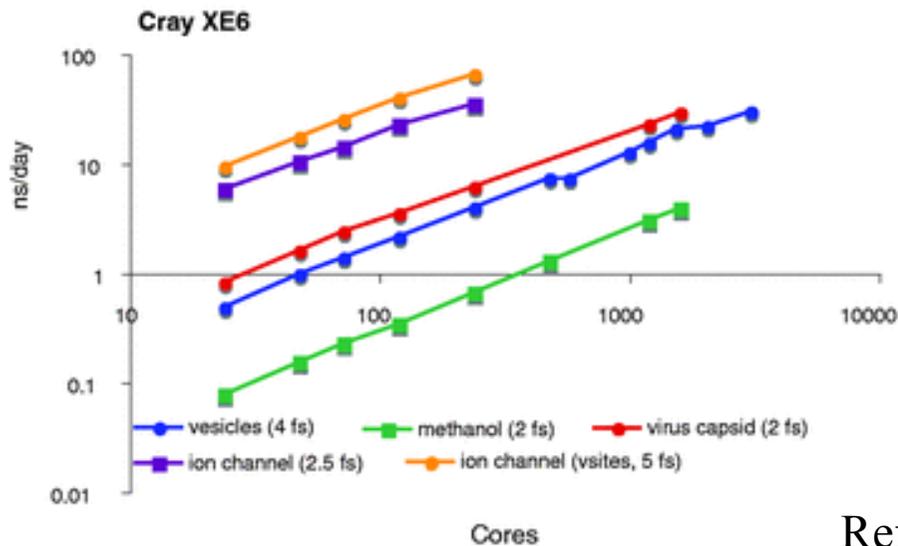
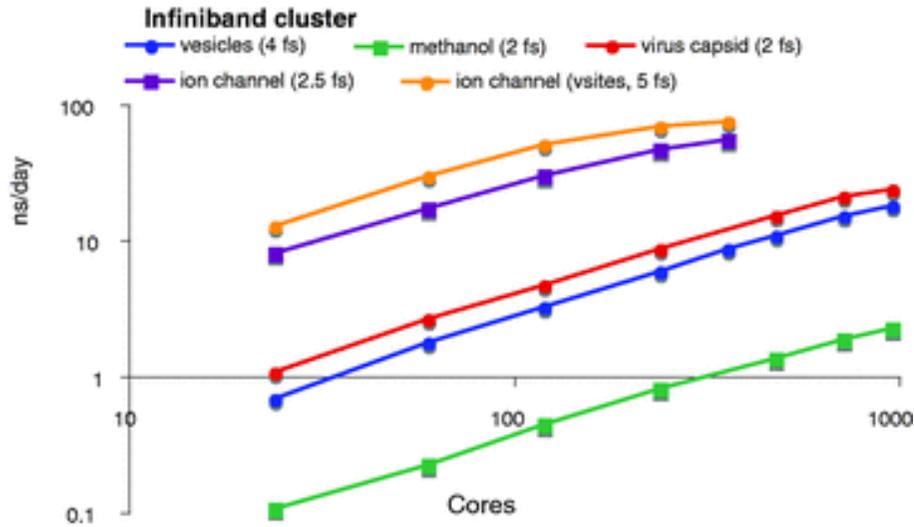
```
do iy = 1, ygrid(local)
  do ix = 1, xgrid(local)
    work_local = Q(my_rank)
    call fftw(work_local)
    Q(my_rank) = work_local
  end do
end do
call mpi_alltoall(Q,Q_new,..)
do iz = 1, zgrid(local)
  do ix = 1, xgrid(local)
    work_local = Q(my_rank)
    call fftw(work_local)
    Q(my_rank) = work_local
  end do
end do
call mpi_alltoall(Q,Q_new)
do iz = 1, zgrid(local)
  do iy = 1, ygrid(local)
    work_local = Q(my_rank)
    call fftw(work_local)
    Q(my_rank) = work_local
  end do
end do
compute force
```

2 dimensional view of conventional method



Benchmark result of existing MD programs

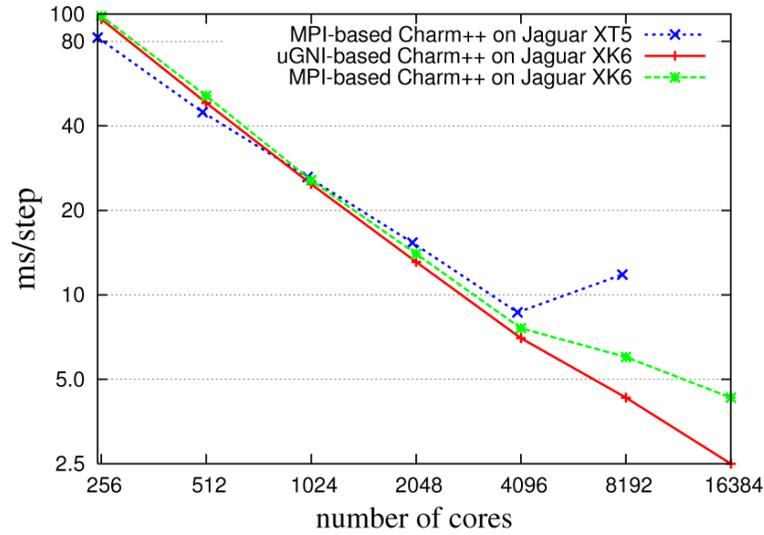
Gromacs



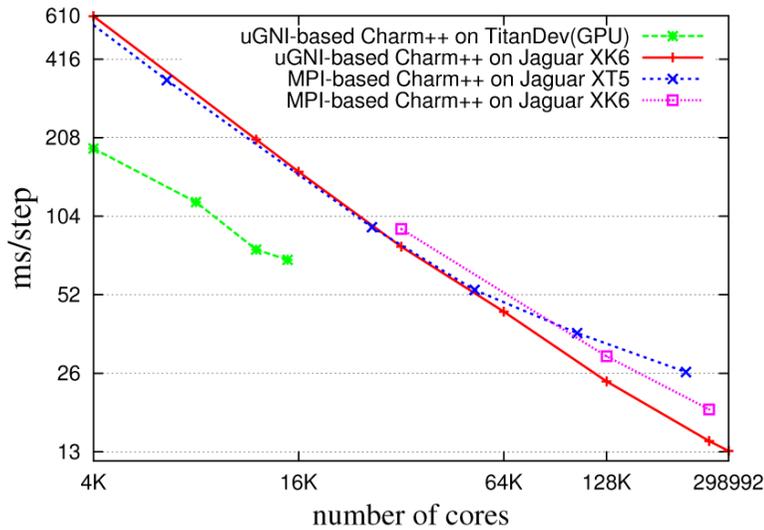
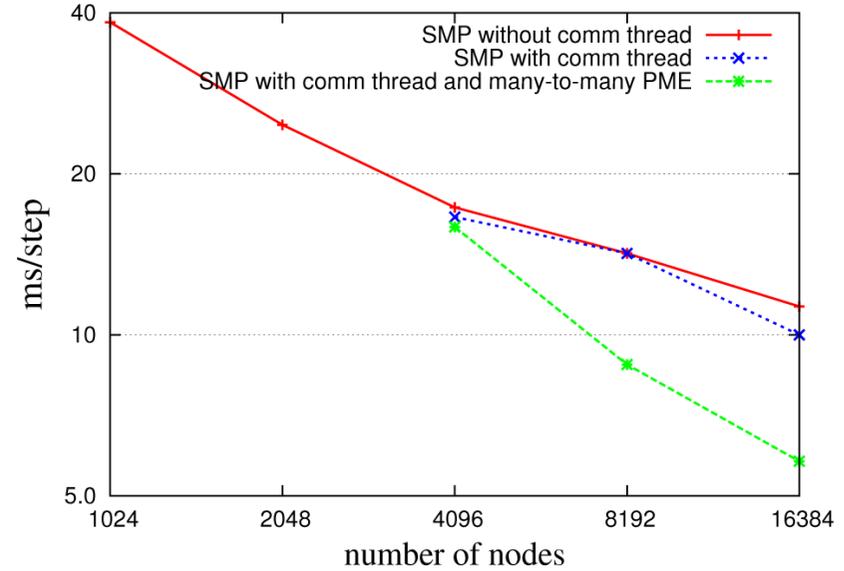
Ion channel /w virtual sites : 129,692
 Ion channel /w.o virtual site : 141,677
 Virus capsid : 1,091,164
 Vesicle fusion : 2,511,403
 Methanol : 7,680,000

Ref : S. Pronk et al. Bioinformatics, btt055 (2013)

NAMD



(a) 1M STMV



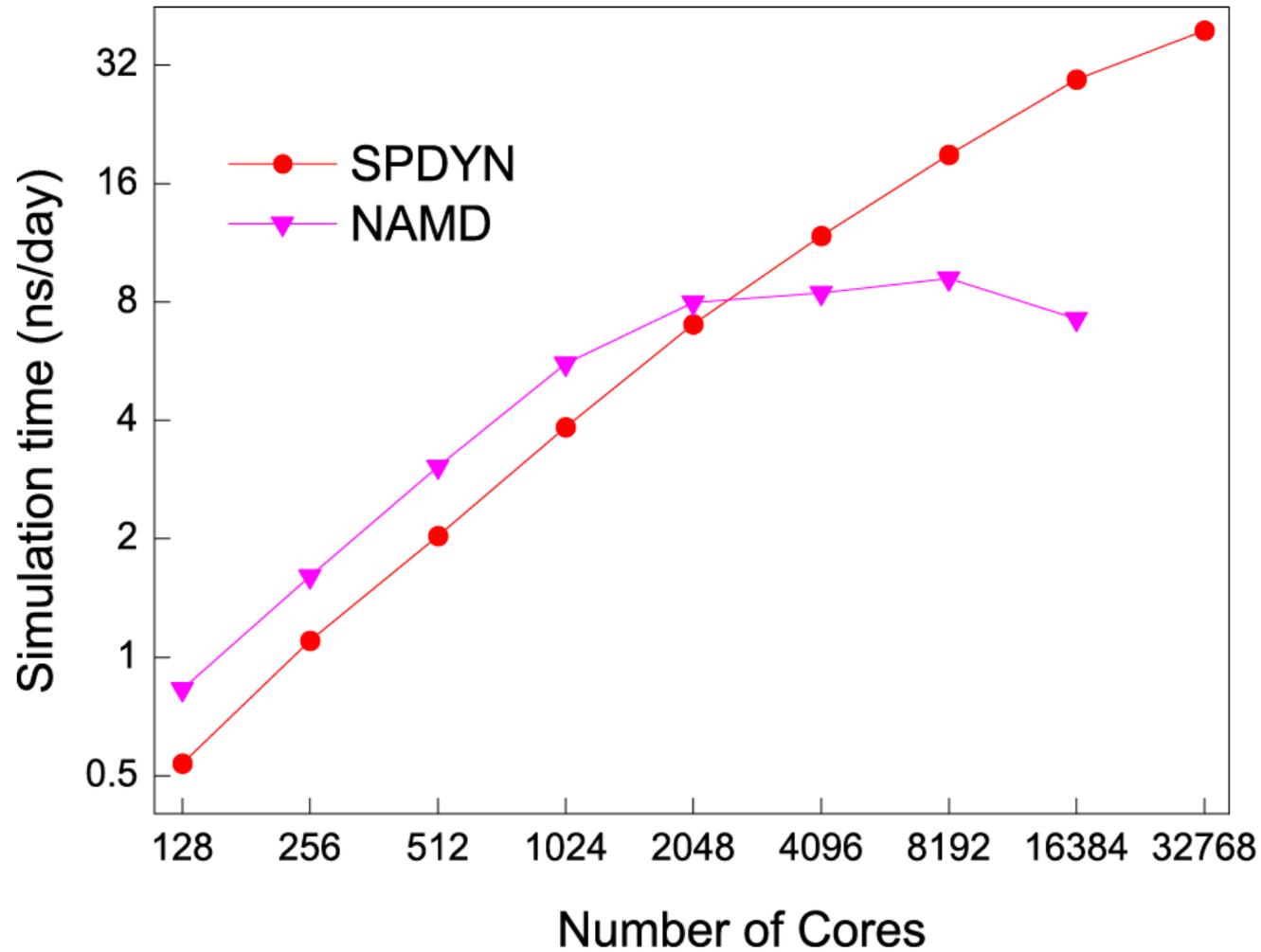
(b) 100M STMV

Nodes	Cores	Timestep(ms)
2048	32768	98.8
4096	65536	55.4
8192	131072	30.3
16384	262144	17.9

Titan (Ref : Y. Sun et al. SC12)

20M and 100M system on Blue Gene/Q
(Ref : K. Sundhakar et al. IPDPS, 2013 IEEE)

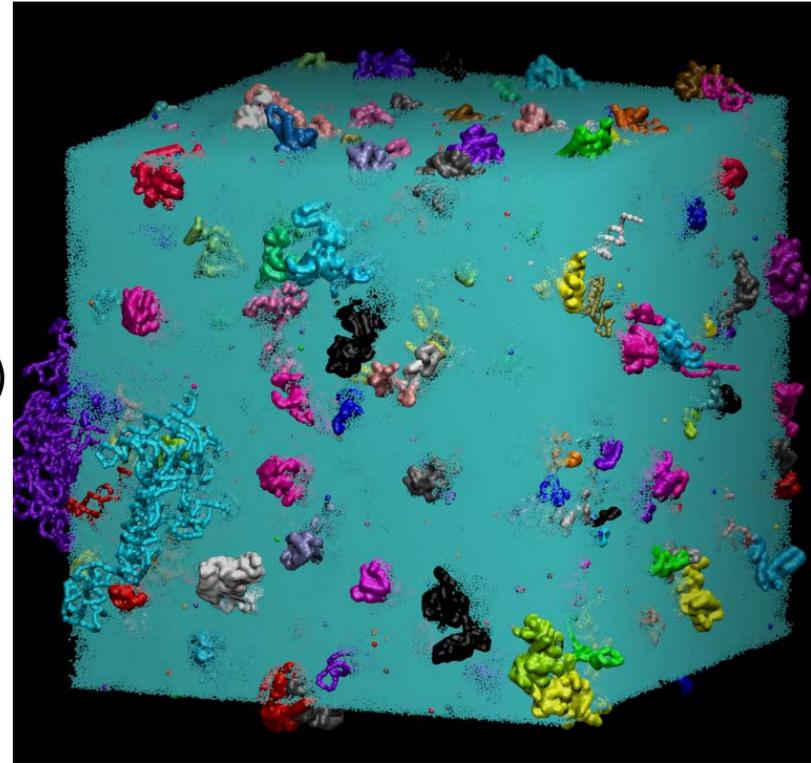
GENESIS on K (1M atom system)



Ref) J. Jung et al. WIREs CMS, 5, 310-323 (2015)

■ 12M System (Provided by Dr. Isseki Yu)

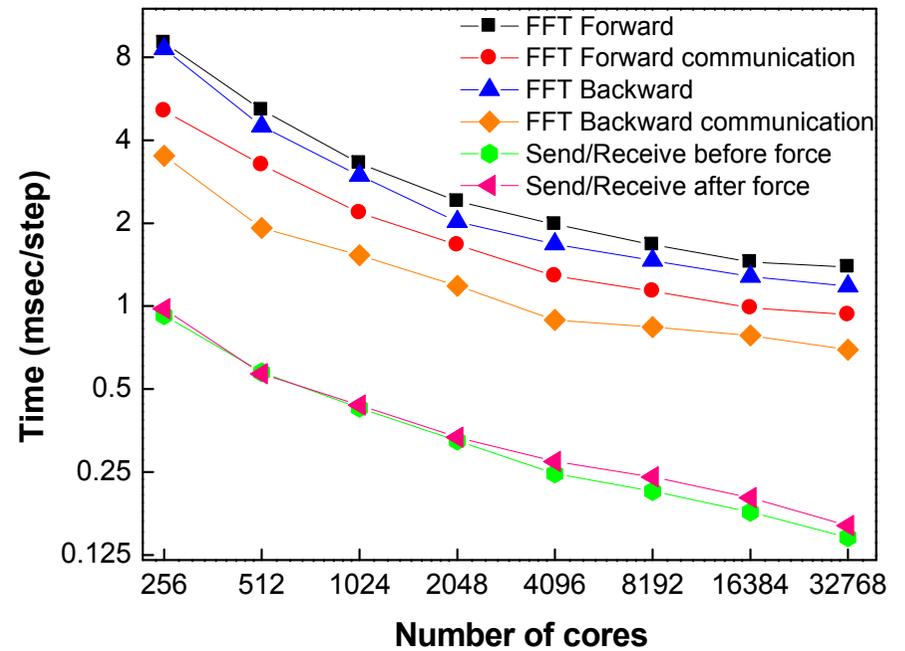
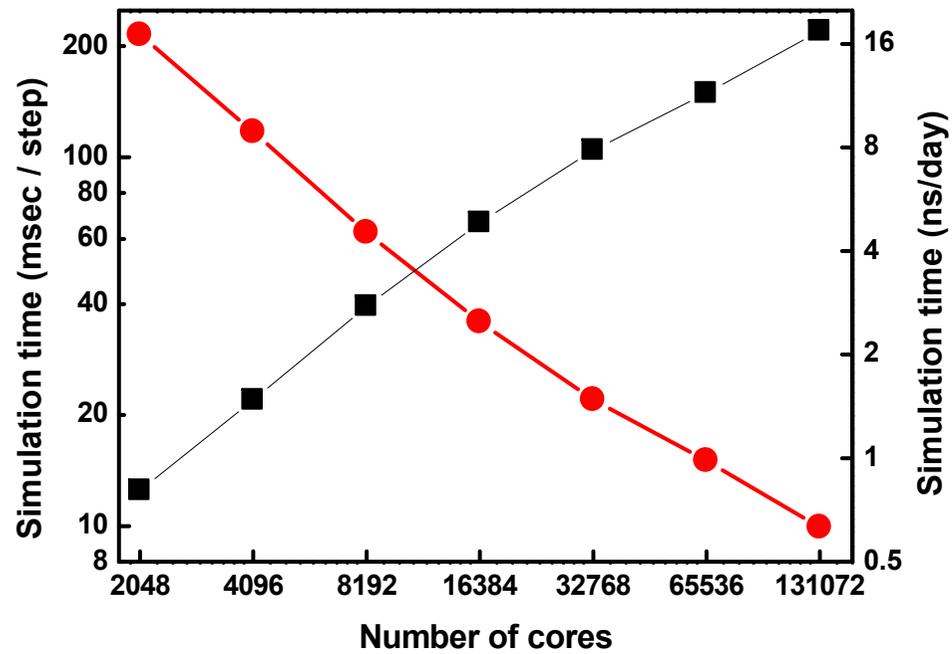
- atom size: 11737298
- macro molecule size: 216 (43 type)
- metabolites size: 4212 (76 type)
- ion size: 23049
(Na⁺, Cl⁻, Mg²⁺, K⁺)
- water size: 2944143
- PBC size: 480 x 480 x 480(Å³)
- Volume fraction of water 75%



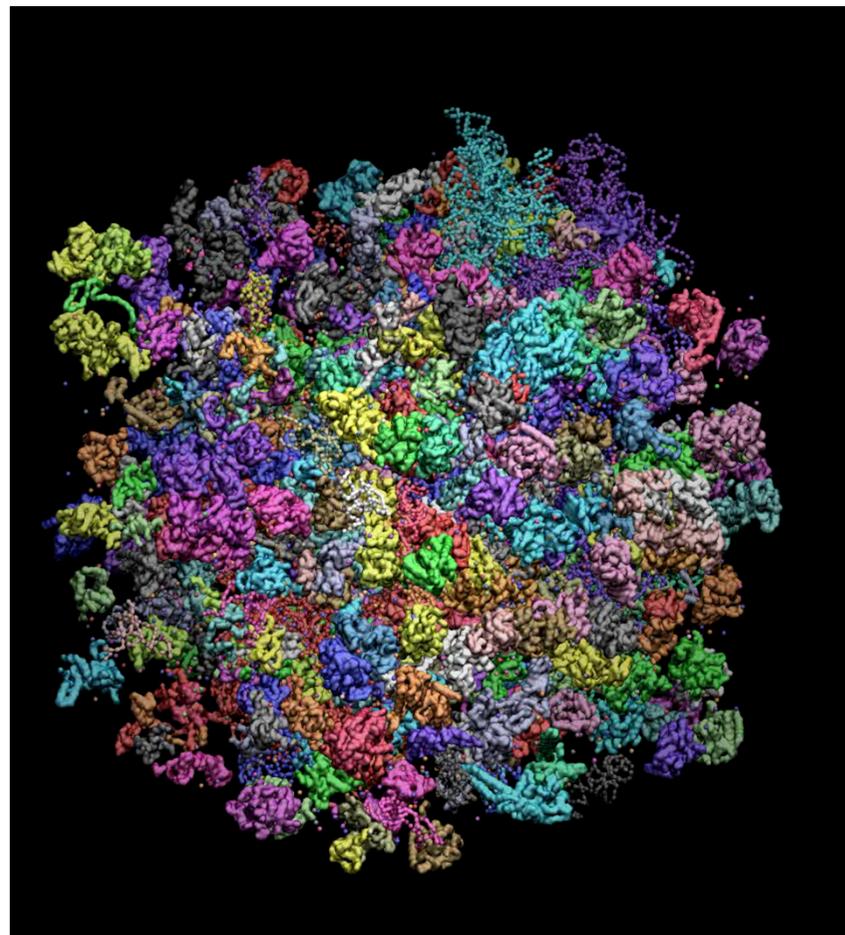
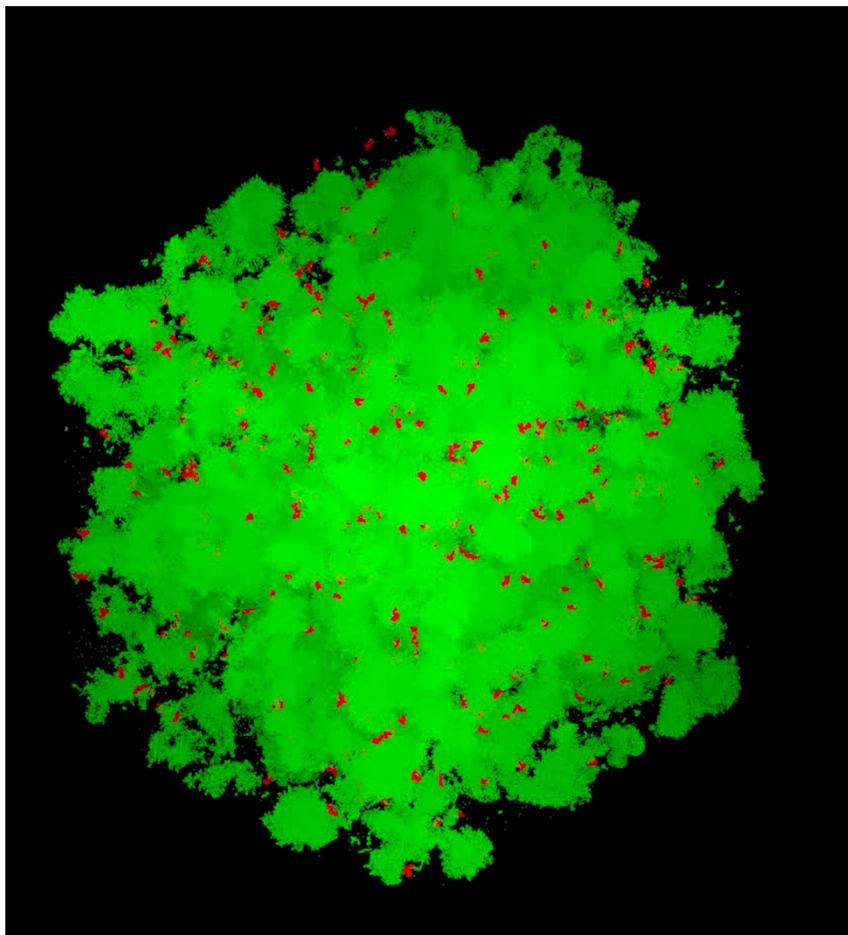
■ Simulation

- MD program: GENESIS
- Type of Model: All atom
- Computer: K (8192 (16x32x16) nodes)
- Parameter: CHARMM
- Performance: 12 ns/day, (without waiting time)

GENESIS on K (12 M system)

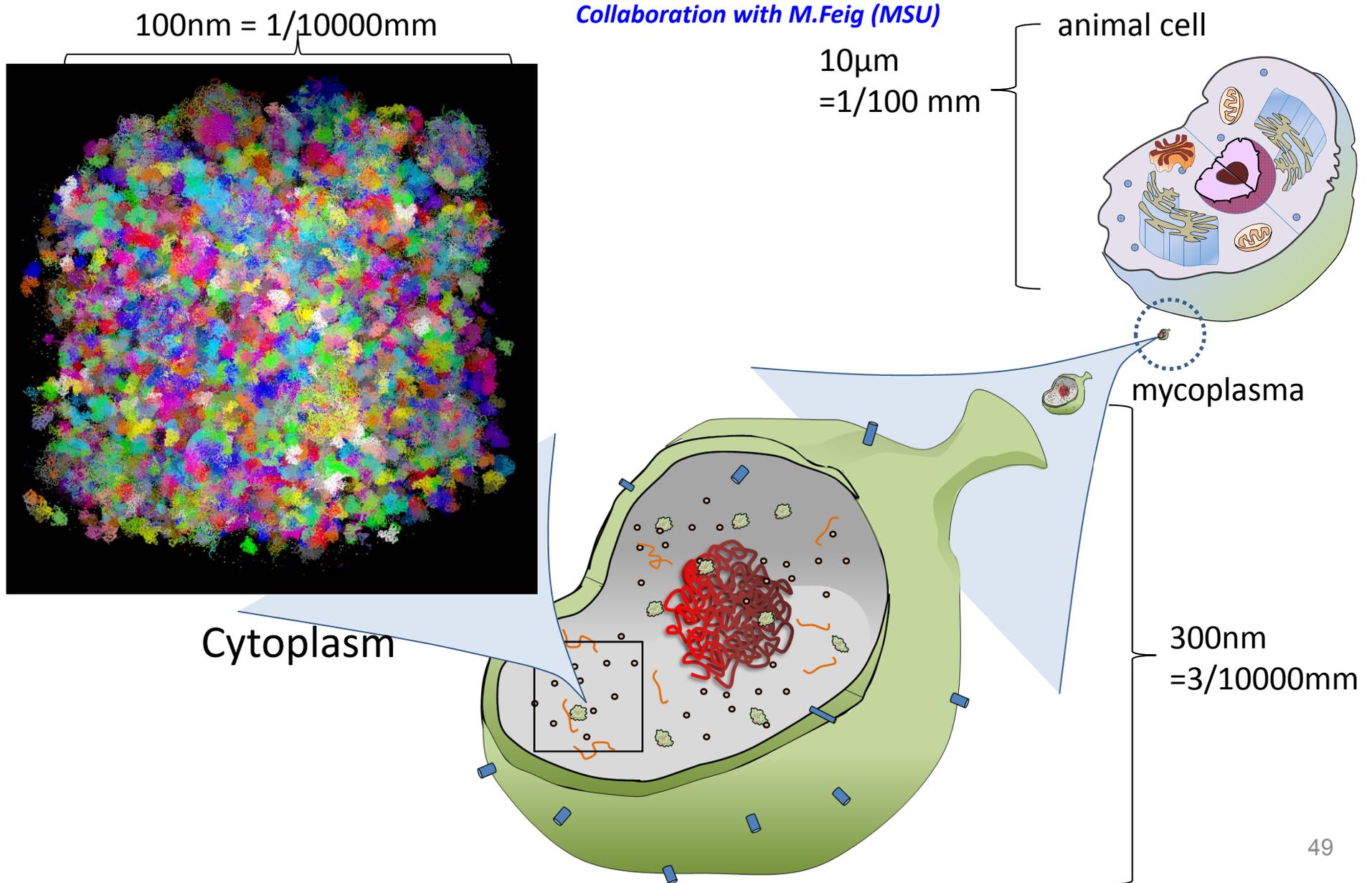


Simulation of 12 M atoms system (provided by Dr. Isseki Yu)

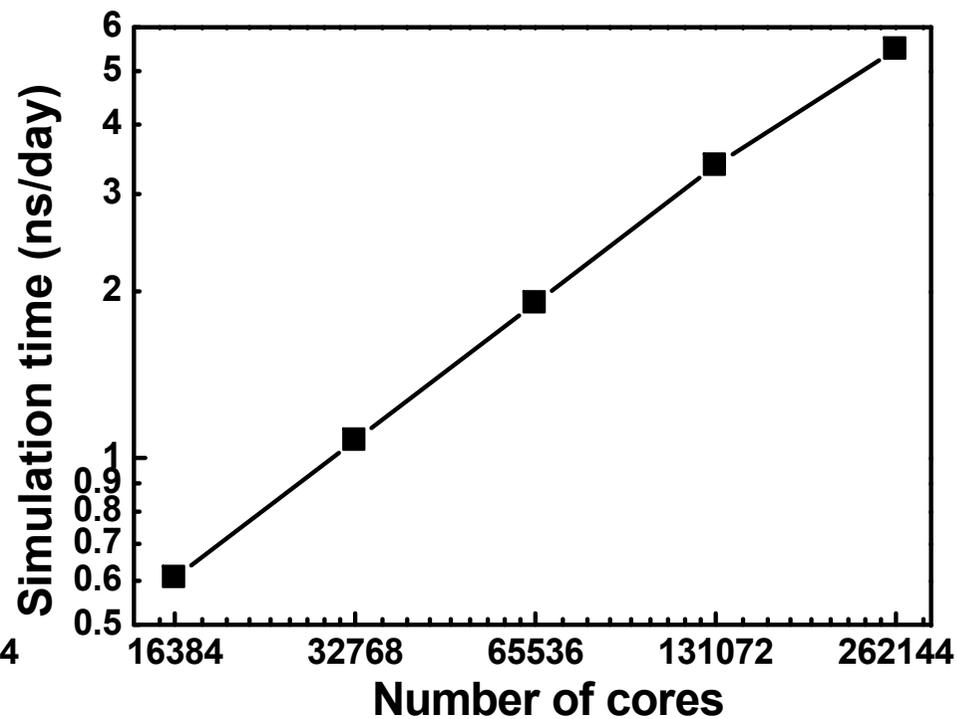
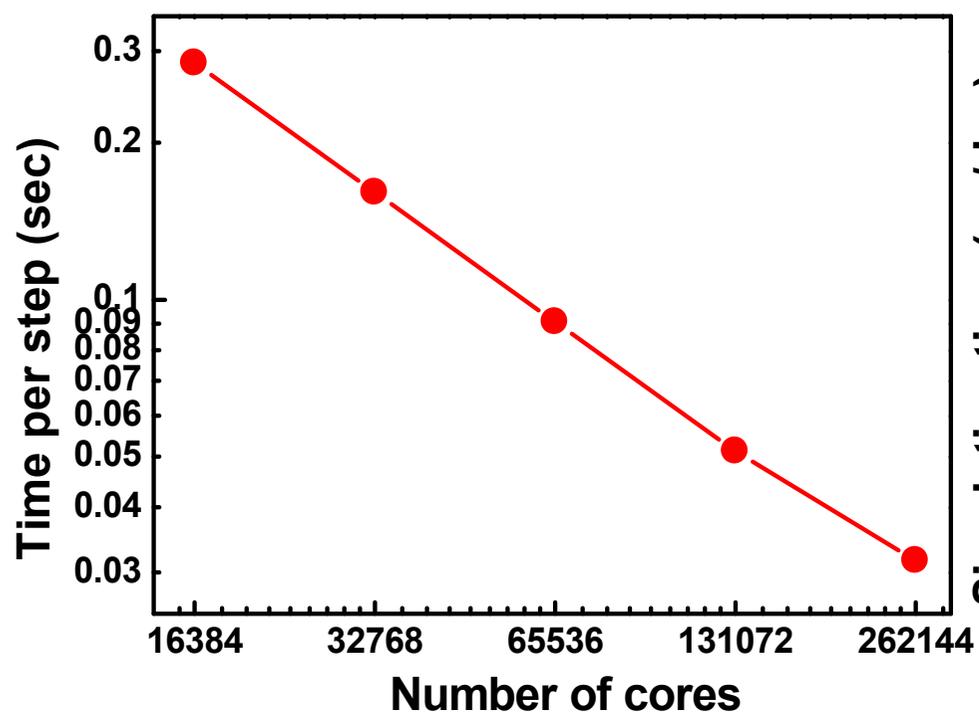


100 M System (provided by Dr. Isseki Yu)

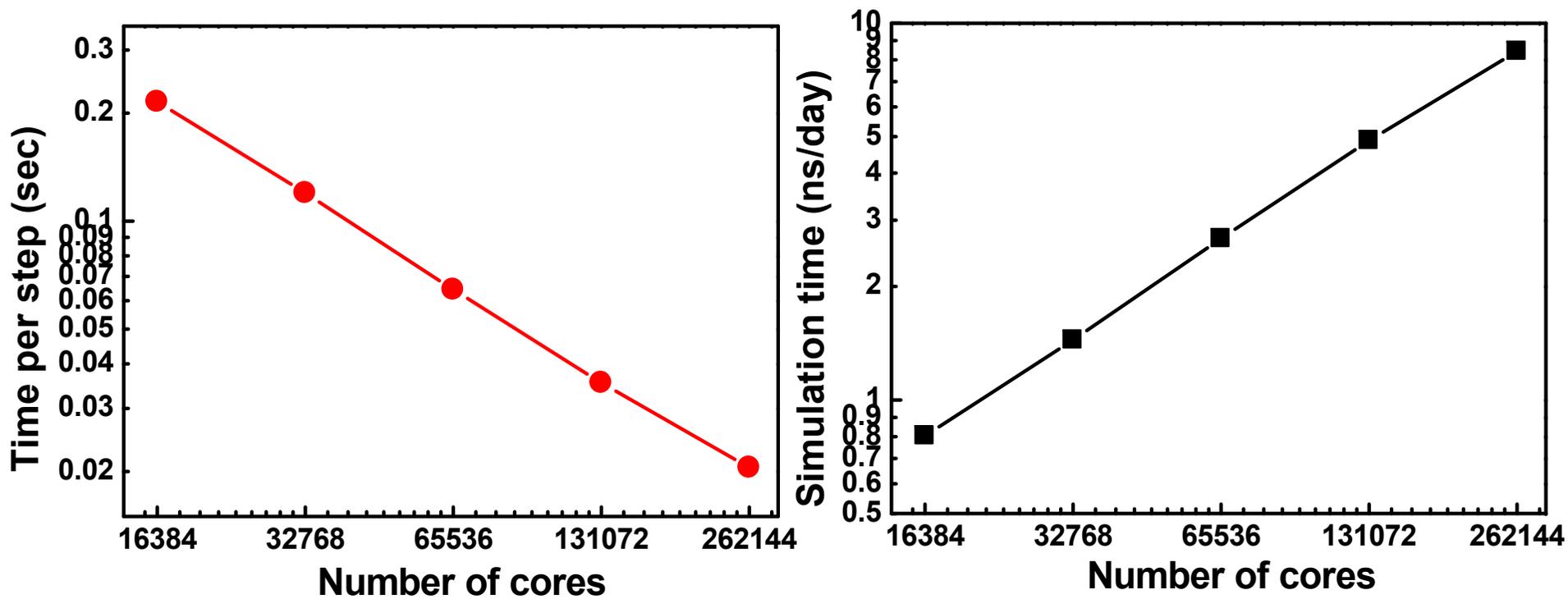
confidential data



GENESIS on K (100 M atoms)



GENESIS on K (100 M atoms, RESPA)



Summary

1. Parallelization of MD is important to obtain the long time MD trajectory.
2. Domain decomposition scheme is used for minimal communicational and computational cost. Energy/forces are described by classical molecular mechanics force field.
3. Hybrid parallelization (MPI+OpenMP) is very helpful to increase the parallel efficiency.
4. Parallelization of FFT is very important using PME method in MD.
5. There are several decomposition schemes of parallel FFT, and we suggest that the cubic decomposition scheme with the midpoint cell is a good solution for efficient parallelization