計算科学技術特論A

Parallelization of molecular dynamics

2017年 7月 13日

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

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

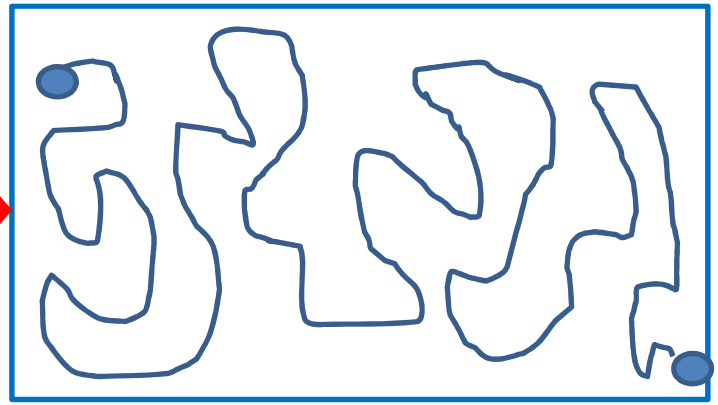
$$\frac{d\mathbf{p}_i}{dt} = \mathbf{F}_i$$

Equation of motion

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

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

Integration



Long time MD trajectory
=> Ensemble generation

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

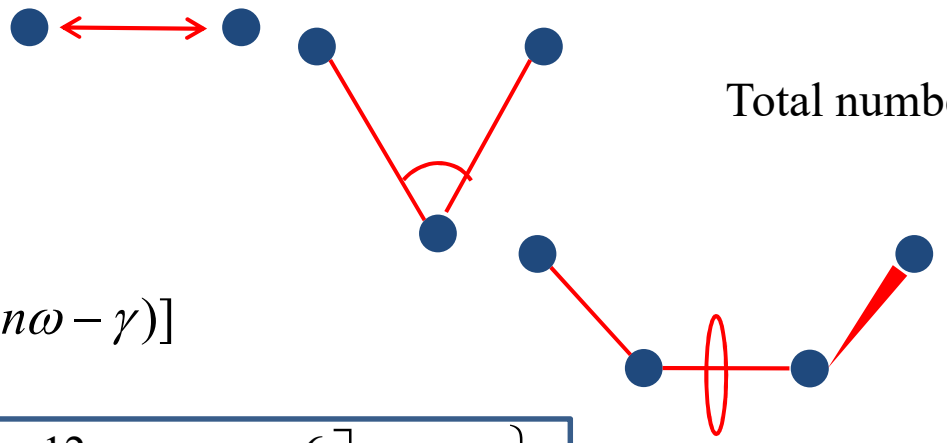
Potential energy in MD

$$E_{\text{total}} = \sum_{\text{bonds}} k_b (b - b_0)^2$$

$$+ \sum_{\text{angles}} k_a (\theta - \theta_0)^2$$

$$+ \sum_{\text{dihedrals}} V_n [1 + \cos(n\omega - \gamma)]$$

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



O(*N*)

Total number of particles

O(*N*)

O(*N*)

O(N^2)

Main bottleneck in MD

$$\sum_{|i-j|<R} \left\{ \varepsilon_{ij} \left[\left(\frac{r_{0ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{0ij}}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j \operatorname{erfc}(\alpha r_{ij})}{r_{ij}} \right\}$$

Real space, O(C*N*)

$$+ \sum_{\mathbf{k} \neq 0} \frac{\exp(-\mathbf{k}^2 / 4\alpha^2)}{\mathbf{k}^2} \operatorname{FFT}(Q(\mathbf{k}))$$

Reciprocal space, O(*N*log*N*)

Non-bonded interaction

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

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

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

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

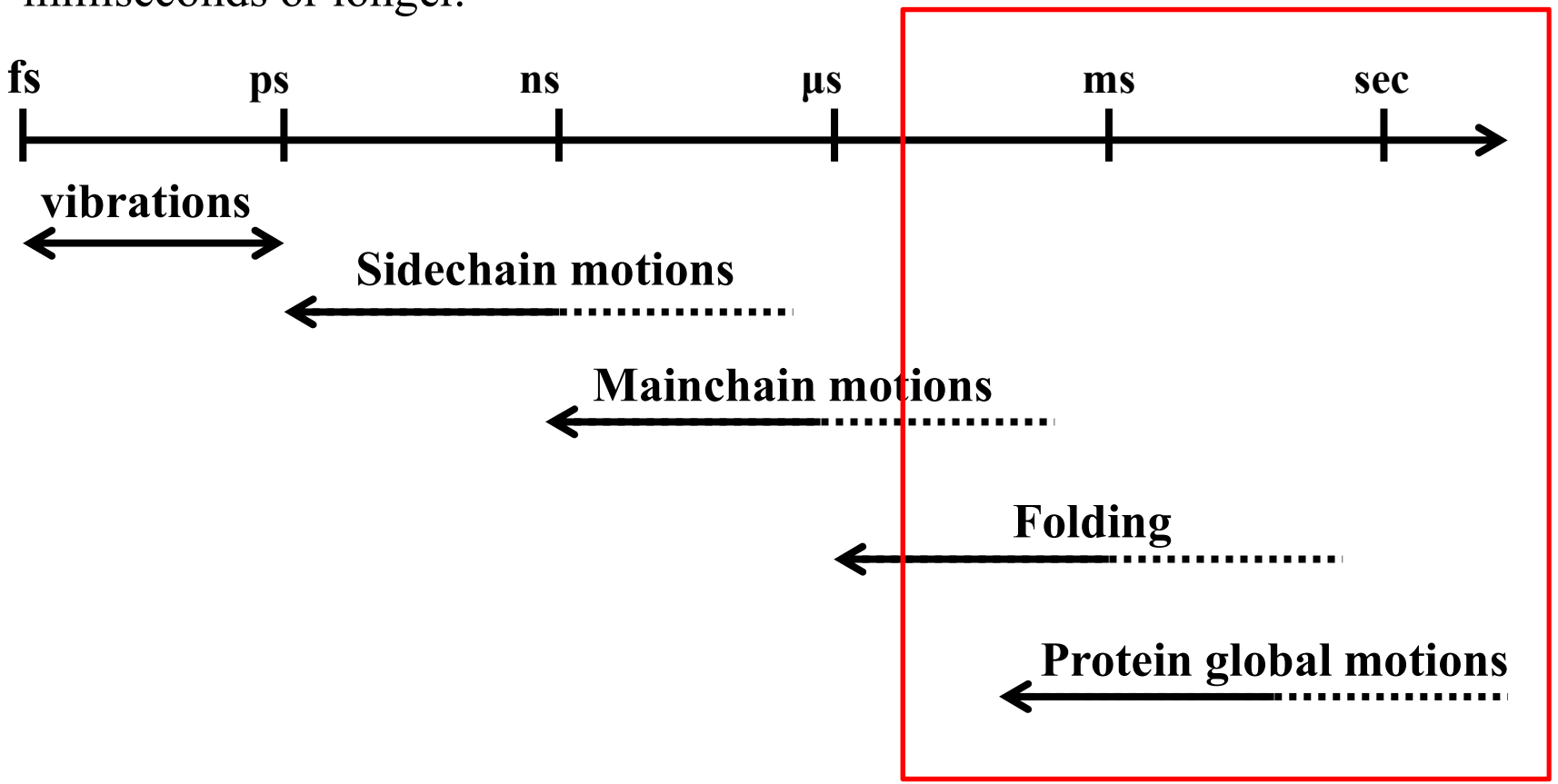
$$\mathrm{U}_{elec}=\underbrace{\sum_{|i-j|<R}\frac{q_i q_j}{4\pi\varepsilon_0}\frac{\mathrm{erfc}(\alpha_{ij})}{r_{ij}}}_{\text{Real part}}+\underbrace{\frac{2\pi}{V}\sum_{\mathbf{G}\neq 0}\frac{\exp(-|\mathbf{G}|^2/4\alpha^2)}{|\mathbf{G}|^2}\sum_{ij}\frac{q_i q_j}{4\pi\varepsilon_0}\cos(\mathbf{G}\cdot r_{ij})}_{\text{Reciprocal part}}-\underbrace{\sum_{i}\frac{q_i q_i}{4\pi\varepsilon_0}\frac{\alpha}{\sqrt{\pi}}}_{\text{Self energy}}$$

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

Difficulty to perform long time MD simulation

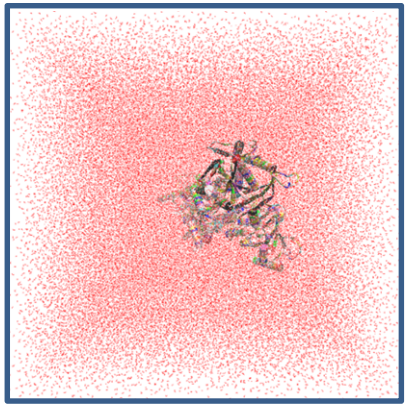
1. One time step length (Δt) is limited to 1-2 fs due to vibrations.

2. On the other hand, biologically meaningful events occur on the time scale of milliseconds or longer.



How to accelerate MD simulations?
=> Parallelization

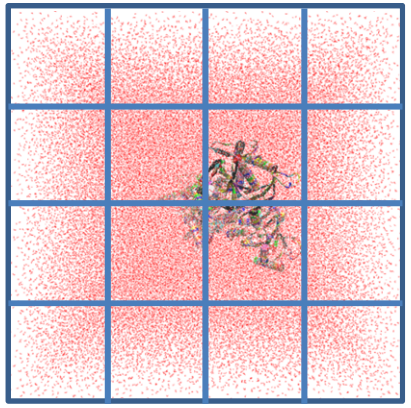
Serial



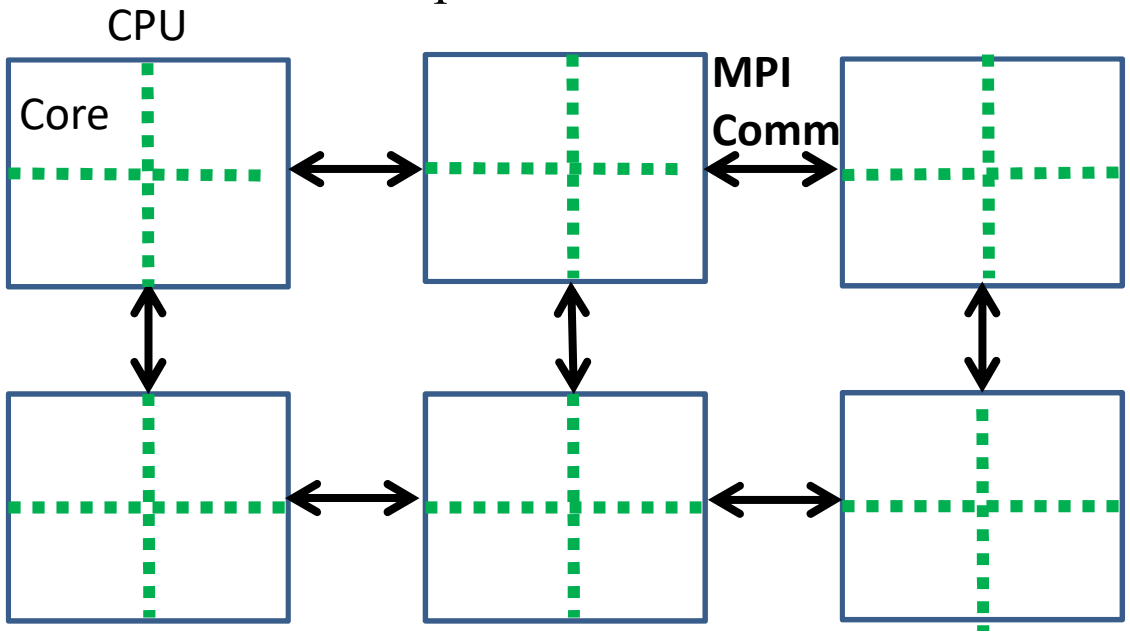
1cpu

X16?

Parallel



16 cpus

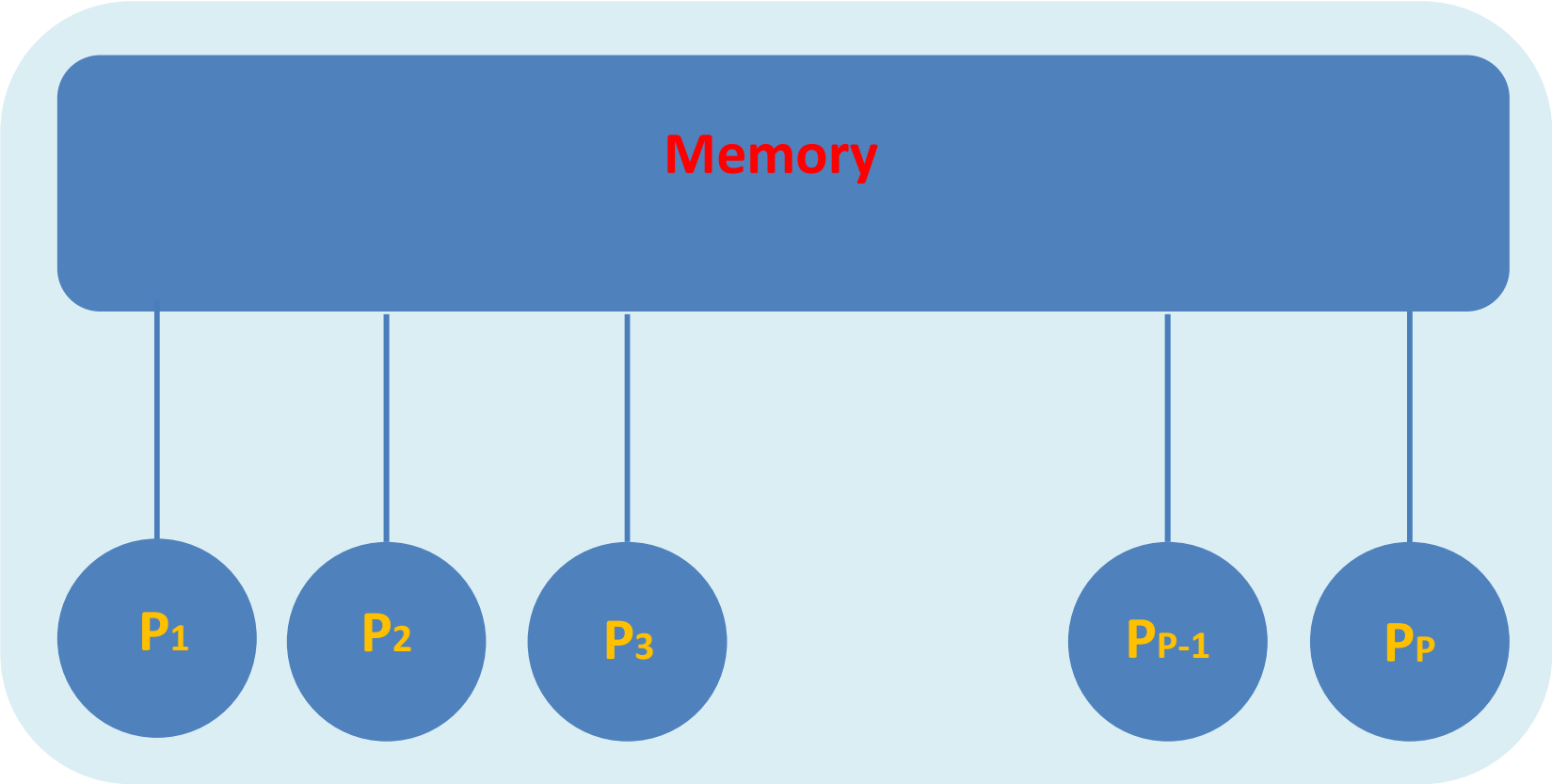


Good Parallelization ;

1) Small amount of computation in one CPU
2) Small amount of communication time

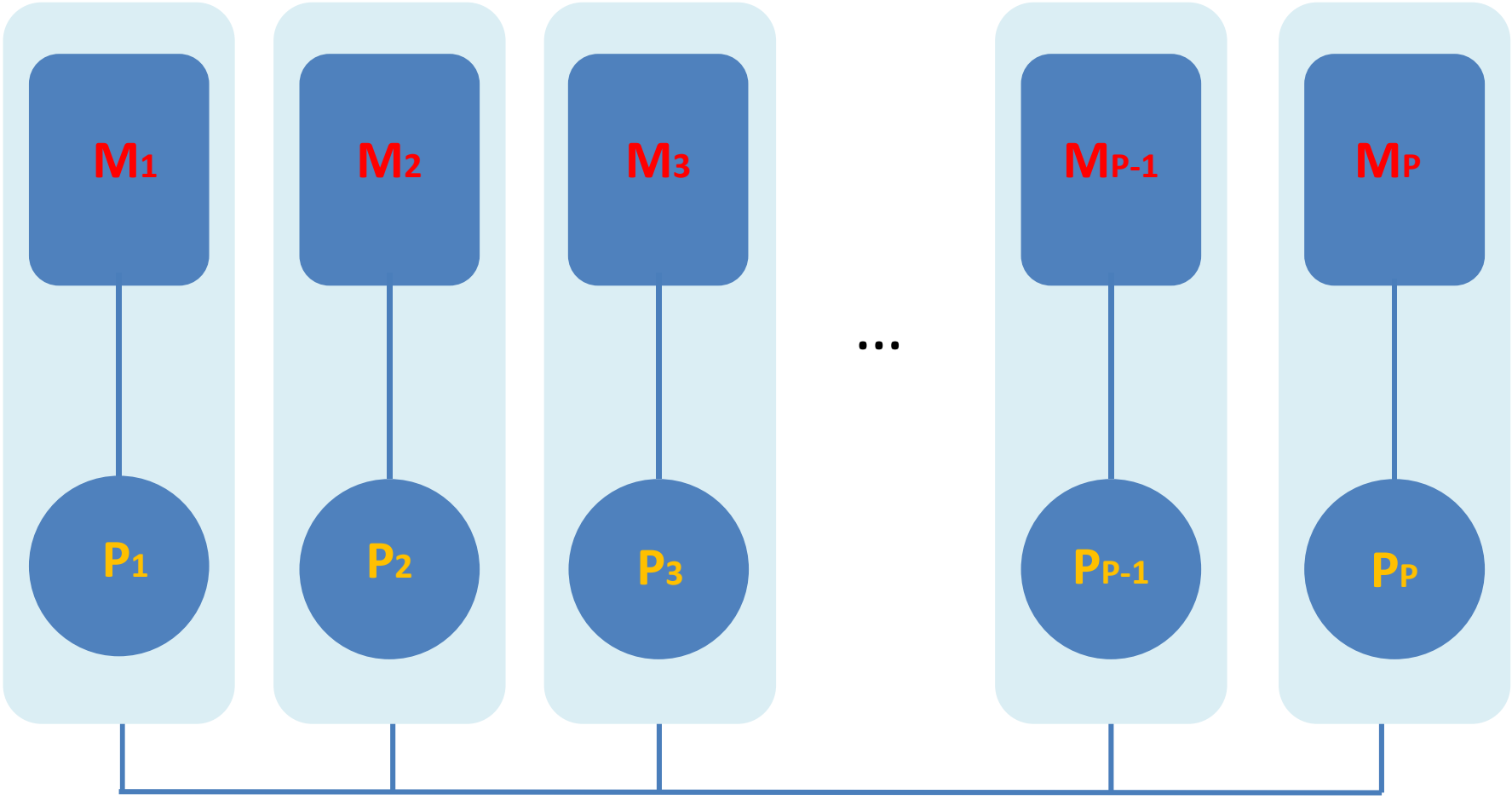
Parallelization

Shared memory parallelization (OpenMP)



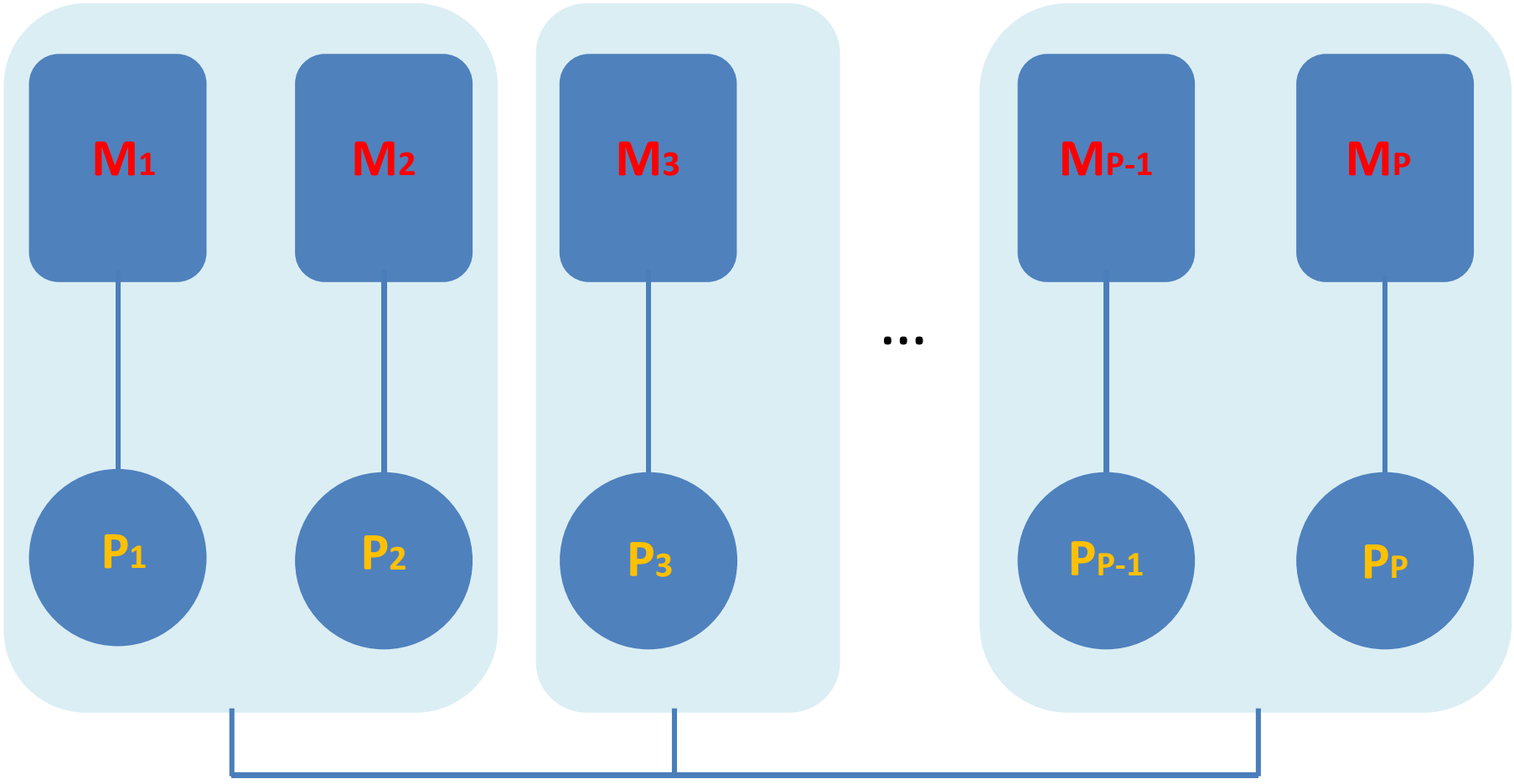
- All processors share data in memory
- For efficient parallelization, processors should not access the same memory address.
- It is only available for multi-processors in a physical node

Distributed memory parallelization (MPI)



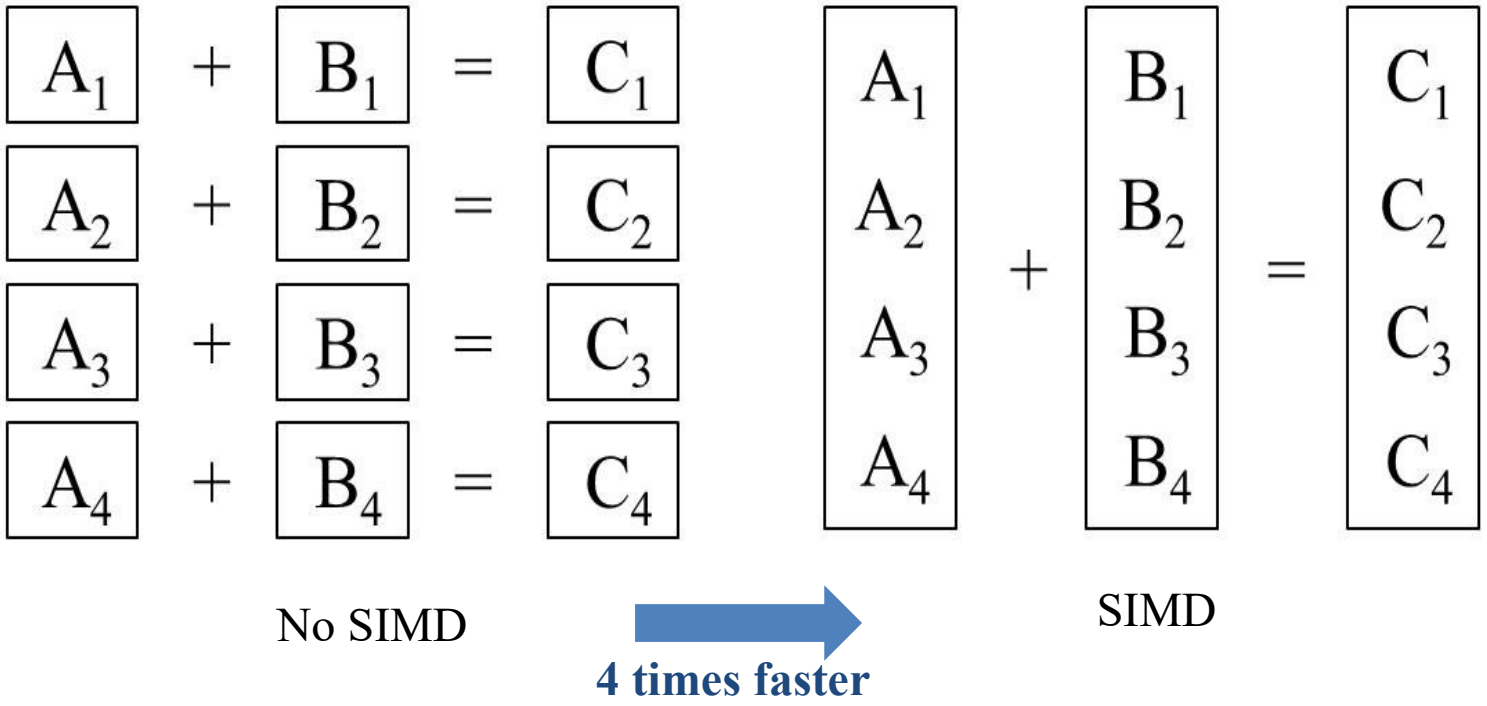
- Processors do not share data in memory
- We need to send/receive data via communications
- For efficient parallelization, the amount of communication data should be minimized

Hybrid parallelization (MPI+OpenMP)



- Combination of shared memory and distributed memory parallelization.
- It is useful for minimizing communicational cost with very large number of processors

SIMD (Single instruction, multiple data)



- Same operation on multiple data points simultaneously
- Usually applicable to common tasks like adjusting graphic image or volume
- In most MD programs, SIMD becomes the one of the important topics to increase the performance

SIMT (Single instruction, multiple threads)

- SIMD hardware with MIMD programming model

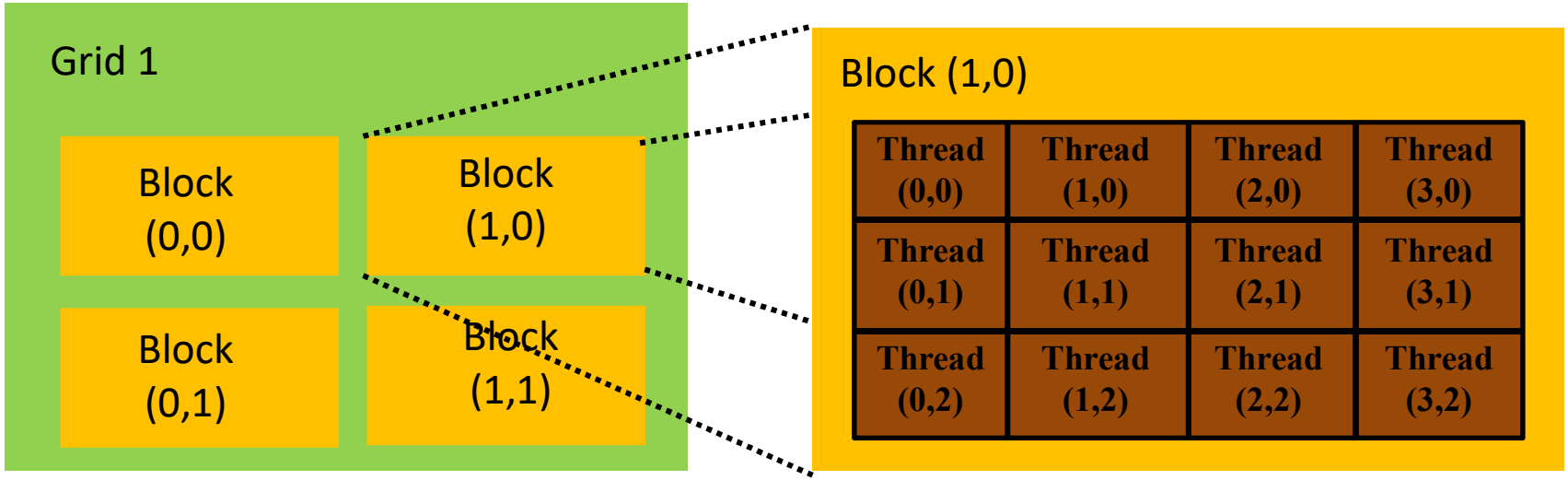
  Ex) if then .. end if => SIMD (×)

  SIMT (○)

- SIMT execution model is usually implement on GPUs and related with GPGPU

  (General Purpose computing on Graphics Processing Units)

- Currently, CUDA allows 32 threads for SIMT (warp size =32)

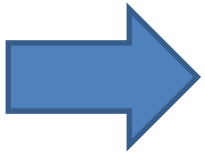


Parallelization of MD (real space)

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)
```

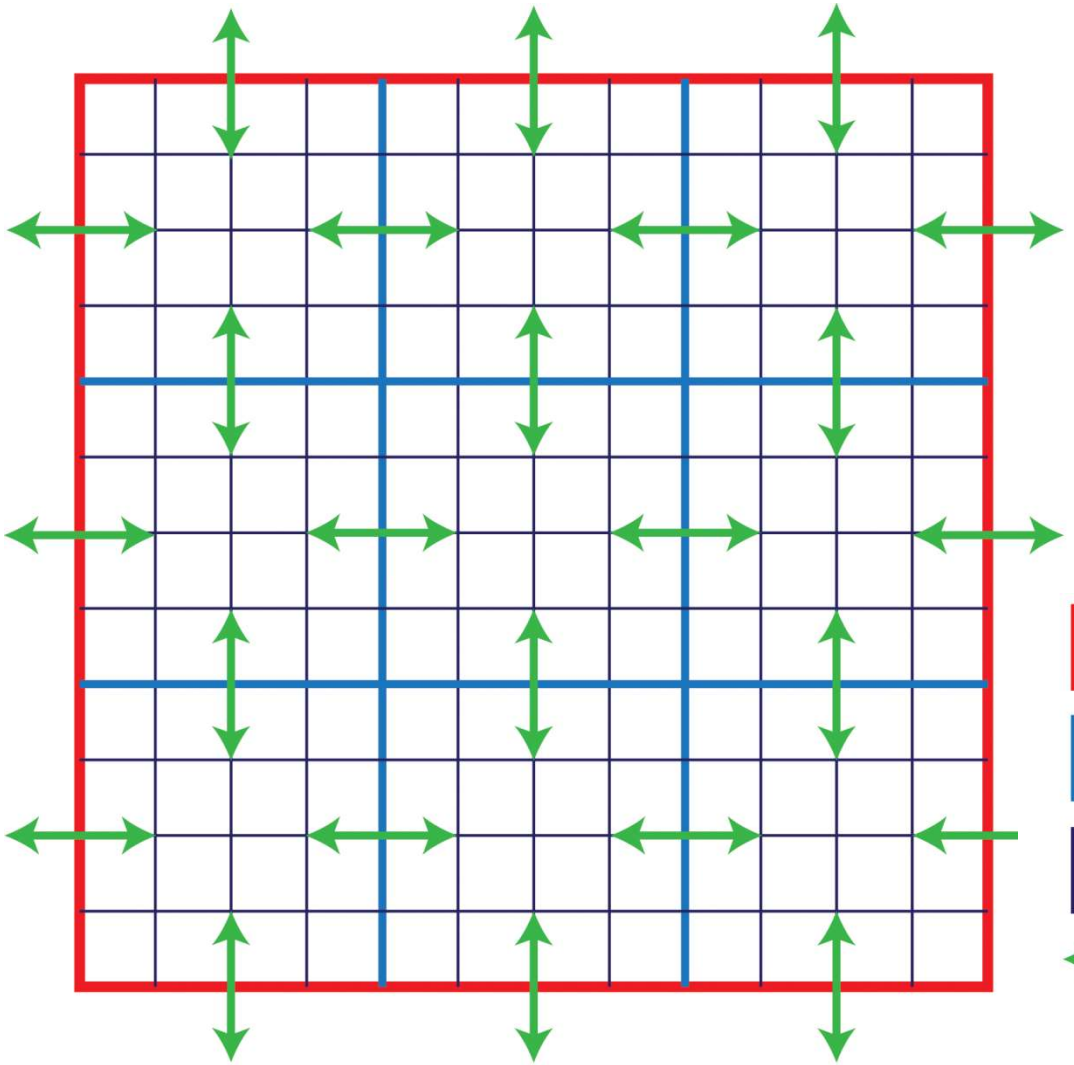
Advantage/Disadvantage of the Replicated data approach

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.

Simulation domain

Subdomain assigned by MPI

Cell

MPI communication

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	$O(N/P)$	$O((N/P)^{2/3})$	$O(N/P)$

Parallelization of MD (reciprocal space)

Smooth particle mesh Ewald method

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

Real part | Reciprocal part | Self energy

The structure factor in the reciprocal part is approximated as

$$S(k_1,k_2,k_3) = b_1(k_1)b_2(k_2)b_3(k_3)F(Q)(k_1,k_2,k_3)$$

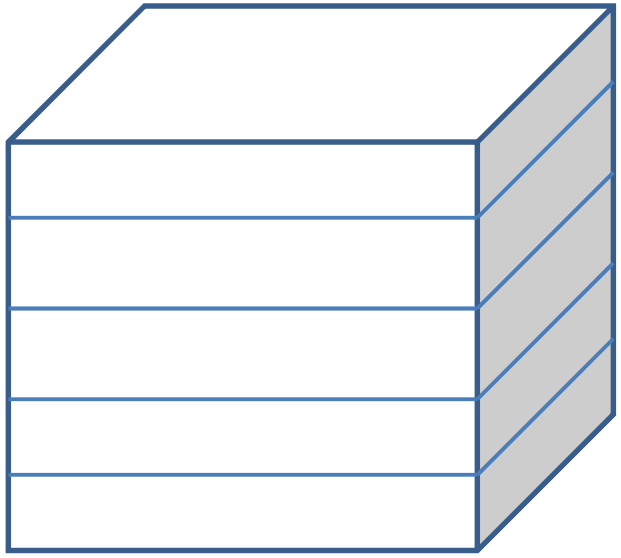
Using Cardinal B-splines of order n

Fourier Transform of charge

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

Ref : U. Essmann et al, J. Chem. Phys. 103, 8577 (1995)

Parallel 3D FFT – slab(1D) decomposition



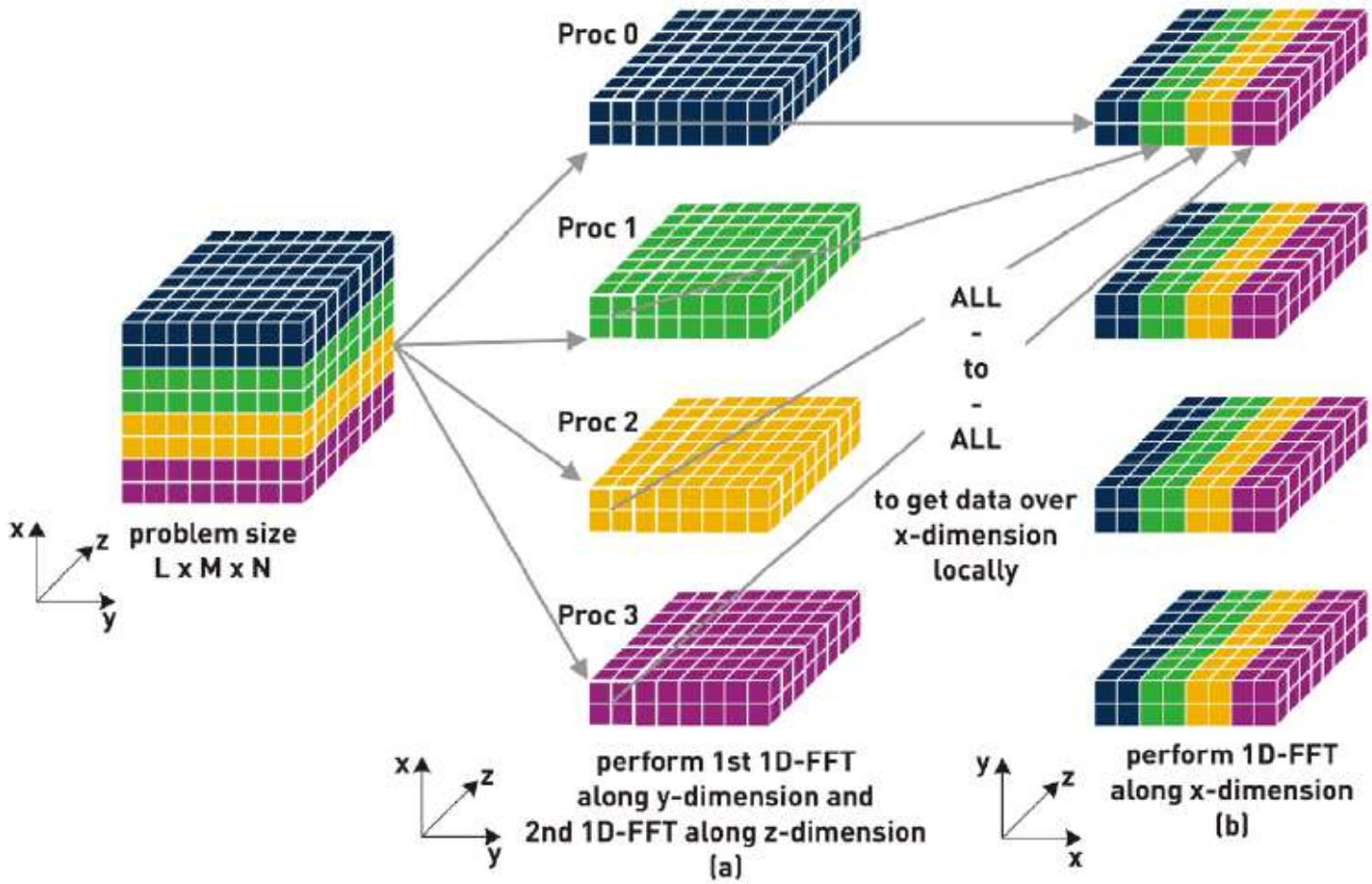
1. Each processor is assigned a slob of size N × N × N/P for computing an N × N × N FFT on P processors.

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

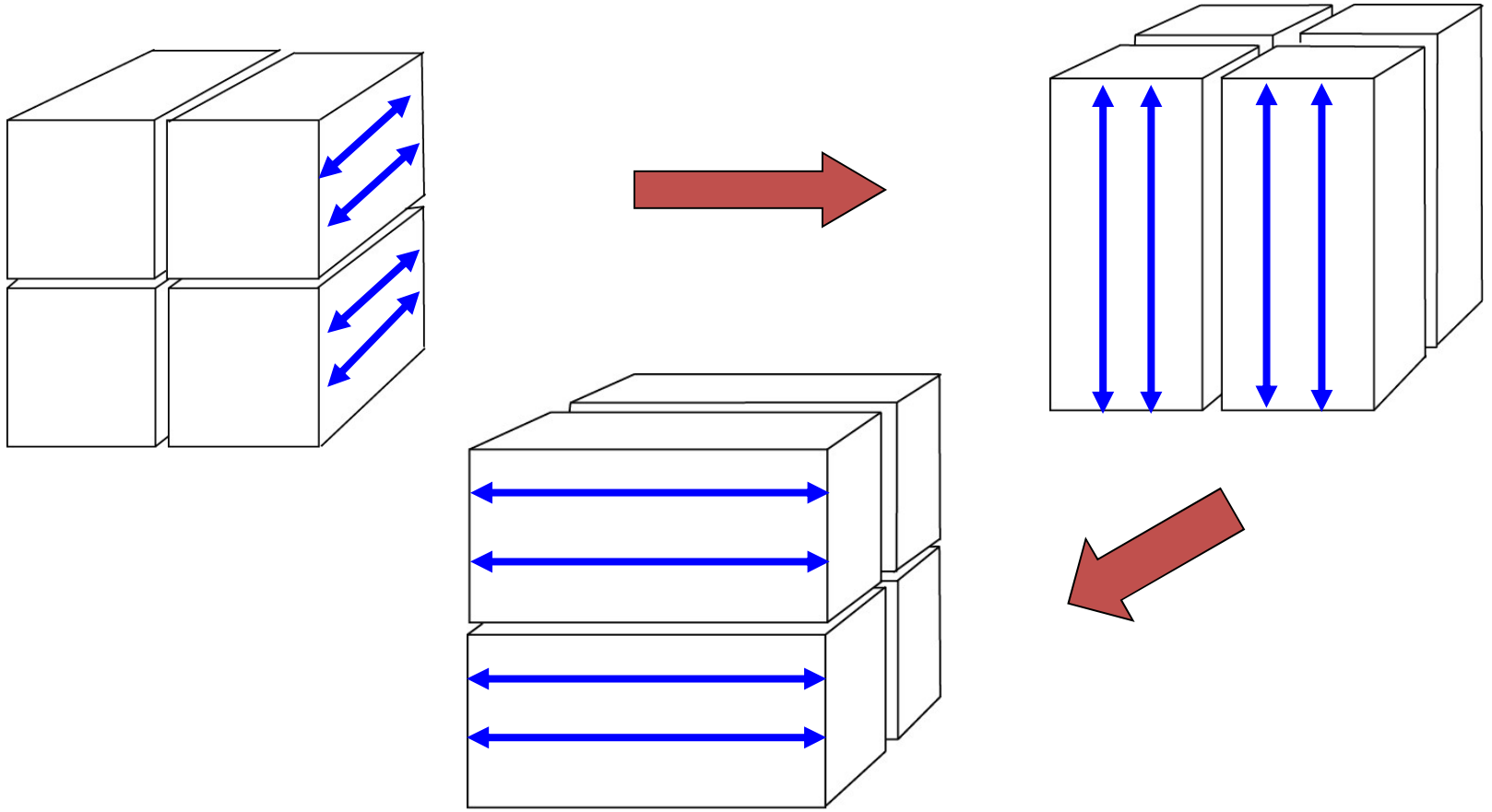
1D decomposition of 3D FFT (continued)

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

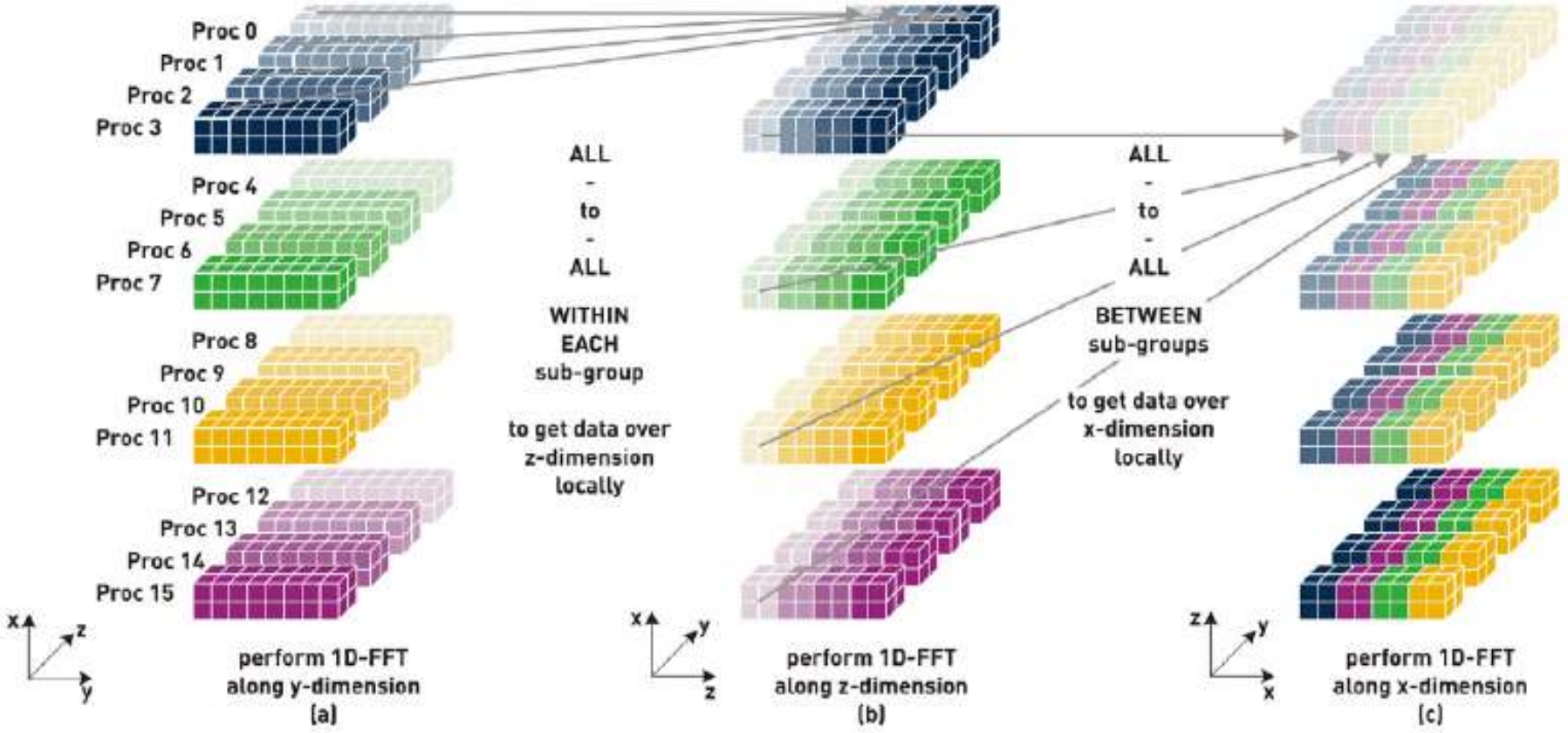
3. Disadvantage : Maximum parallelization is limited to the length of the largest axis of the 3D data (The maximum parallelization can be increased by using a hybrid method combining 1D decomposition with a thread based parallelization)

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 ×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 Plimpoton (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,..)
```

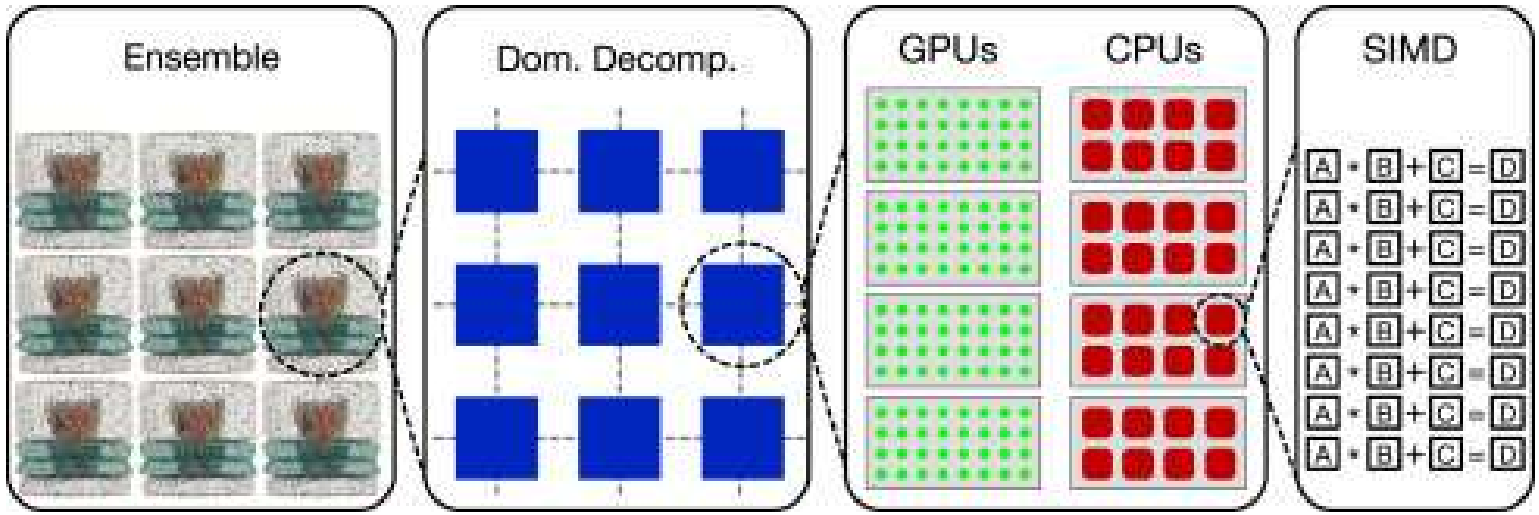
```
do iy = 1, ygrid(local)
  do ix = 1, xgrid(local)
     work_local = Q(my_rank)
     call fftw(work_local)
     Q(my_rank) = work_local
   end do
end do

! 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
```

Parallelization scheme of existing MD programs

Gromacs

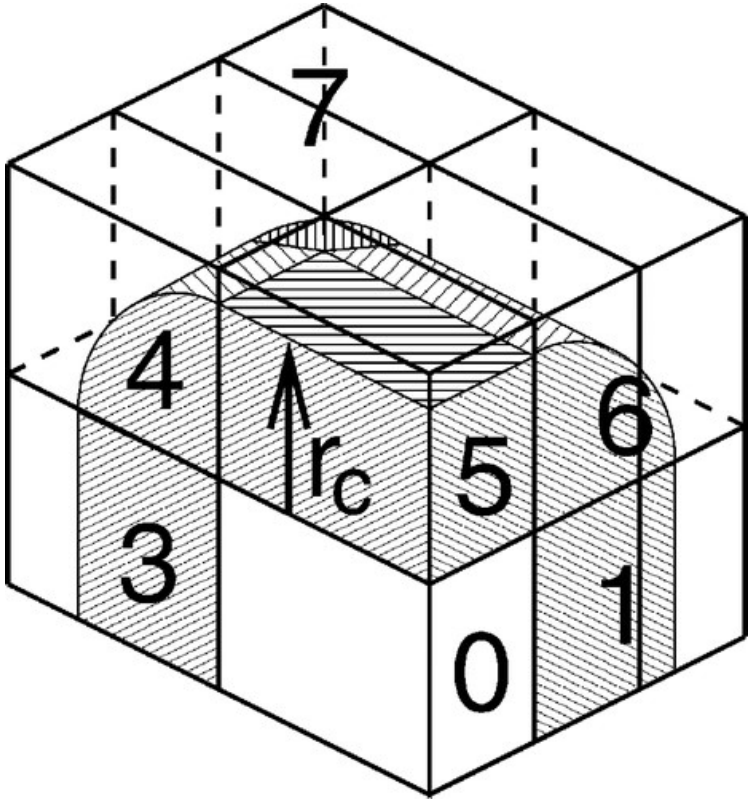
1. Gromacs makes use of multiple parallelism scheme



- SIMD register : cluster interaction kernel or bonded interaction
- OpenMP multithreading : inside domain decomposition
- Nonbed interactions by GPU if GPU cards exist
- MPI for each single domain

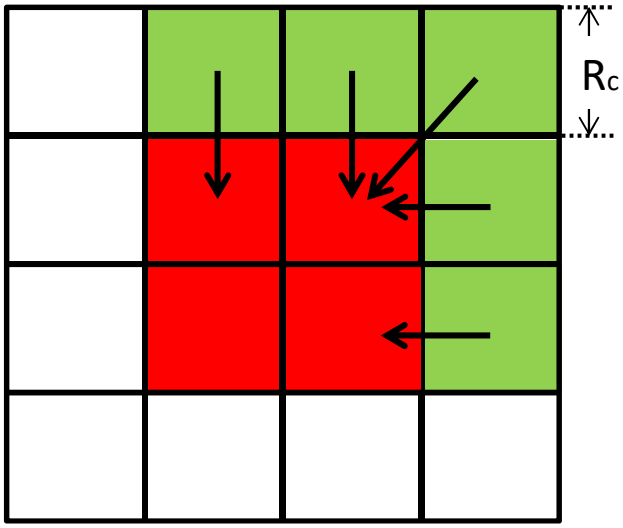
Ref : M. J. Abraham et al, Software X, 1-2, 19-25 (2015)

Gromacs



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

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

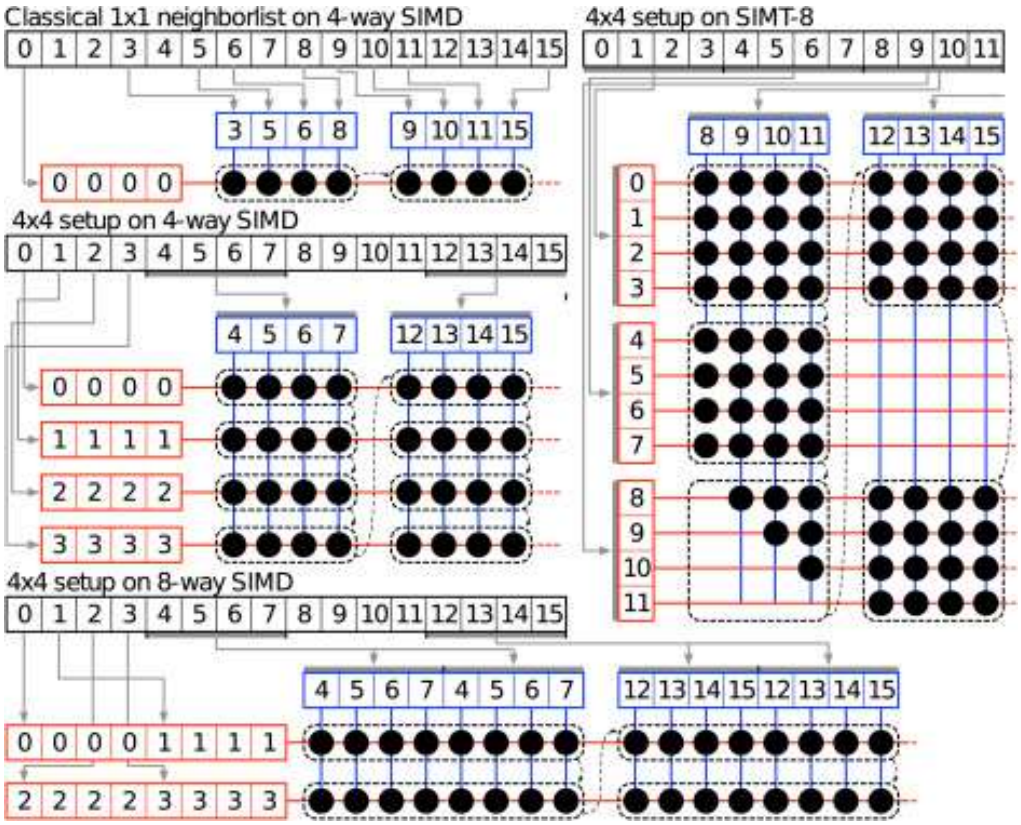


8th shell scheme

Ref : B. Hess et al. J. Chem. Theor. Comput. 4, 435 (2008)

Gromacs

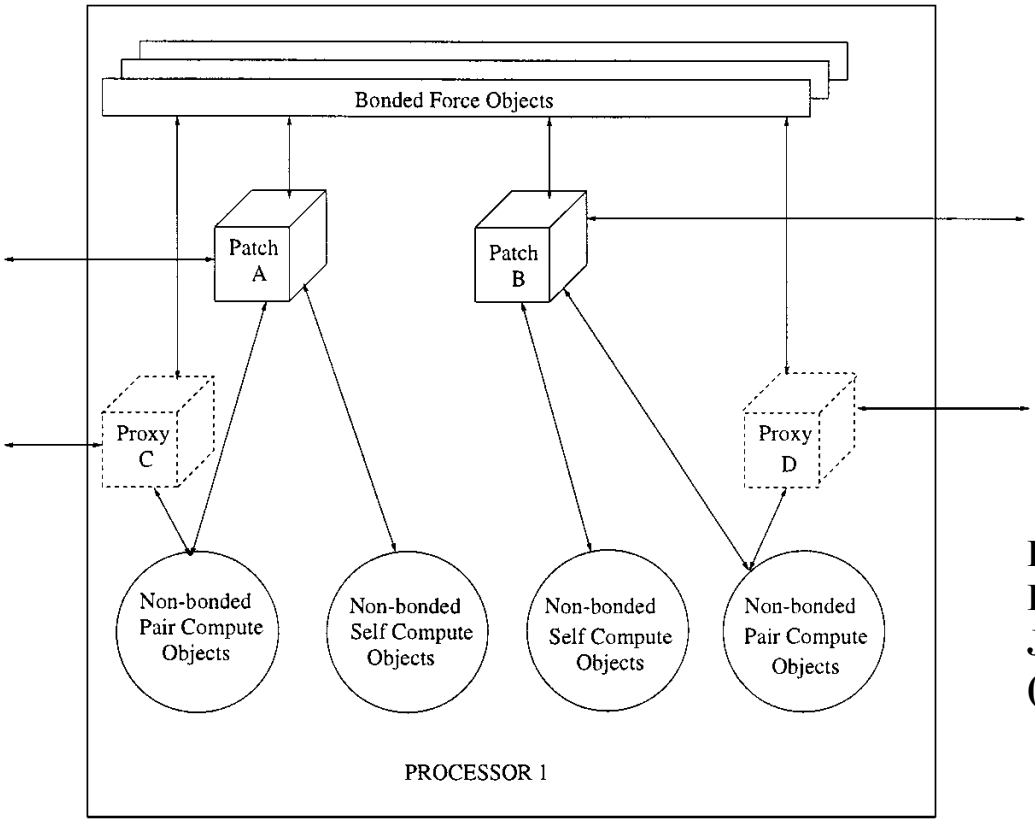
3. Gromacs assigns kernels according to available SIMD / SIMT widths



Ref : M. J. Abraham et al, Software X, 1-2, 19-25 (2015)

NAMD

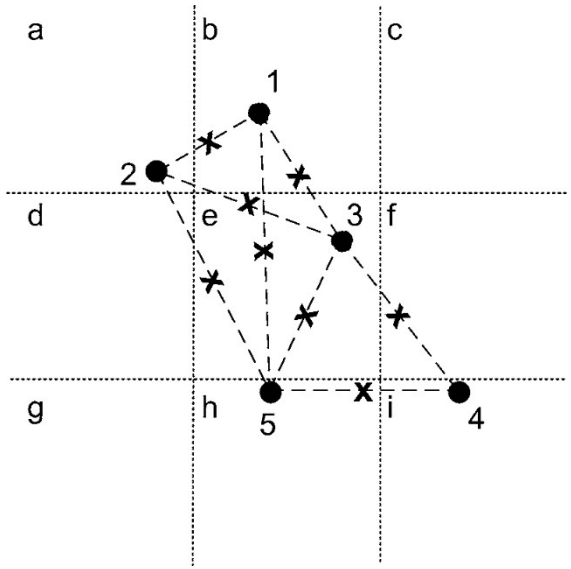
1. NAMD is based on the Charmm++ parallel program system and runtime library.
2. Subdomains named patch are decided according to MPI
3. Forces are calculated by independent compute objects



Ref :
L. Kale et al. J. Comput. Phys. 151, 283 ((1999)
J. C. Phillips et al. J. Comput. Chem. 26, 1781 (2005)

Desmond

1. Midpoint method
   - 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



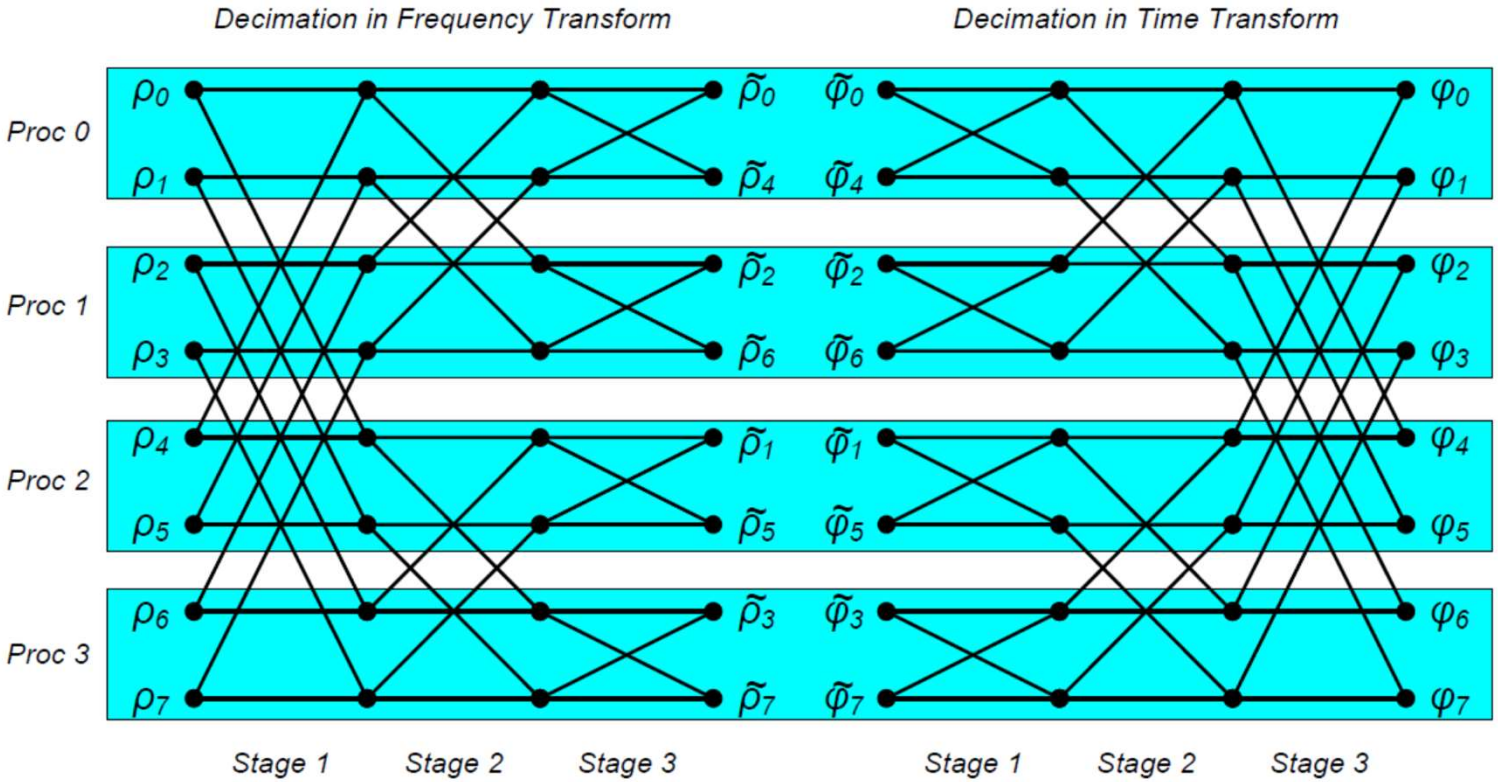
Each pair of particles separated by a distance less than R (cutoff distance) is connected by a dashed line segment, with "x" at its center lying in the box which will compute the interaction of that pair

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

Ref : KJ. Bowers et al, J. Chem. Phys. 124, 184109 (2006)

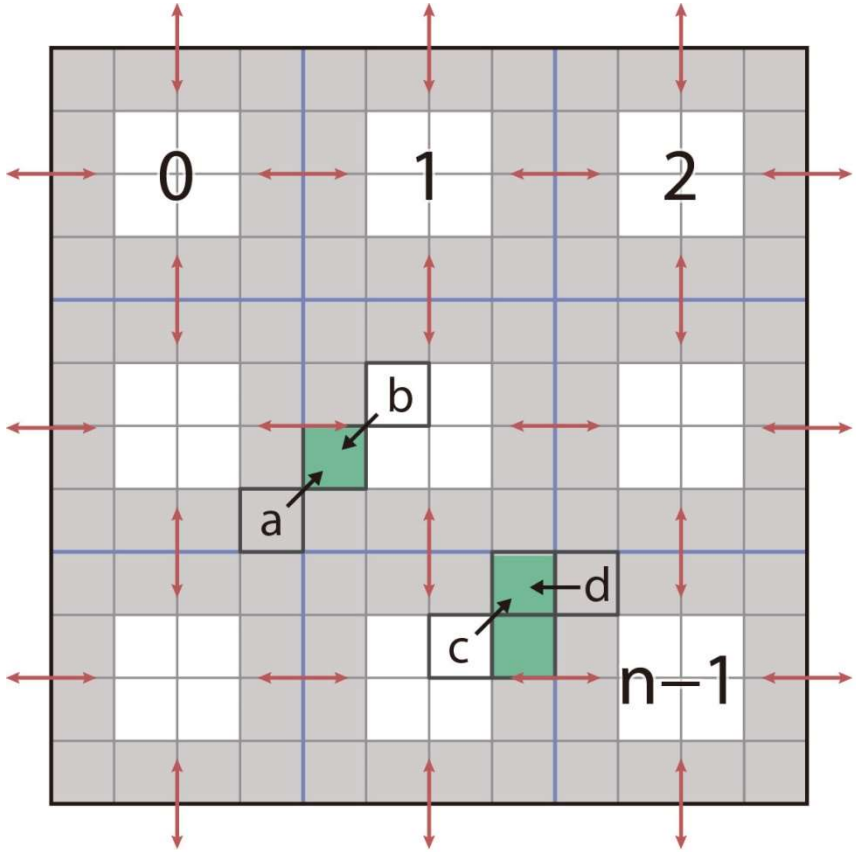
Desmond

2. FFT scheme
   - Not distribute the data (not to use all-to-all communications)
   - Parallelization of one dimensional FFT by performing directly the butterfly communications



Ref : KJ. Bowers et al, Proceedings of the 2006 ACM/IEEE conference on Supercomputing. ACM, 2006

GENESIS



Subdomain assigned by MPI

(Computation is assigned to one CPU)

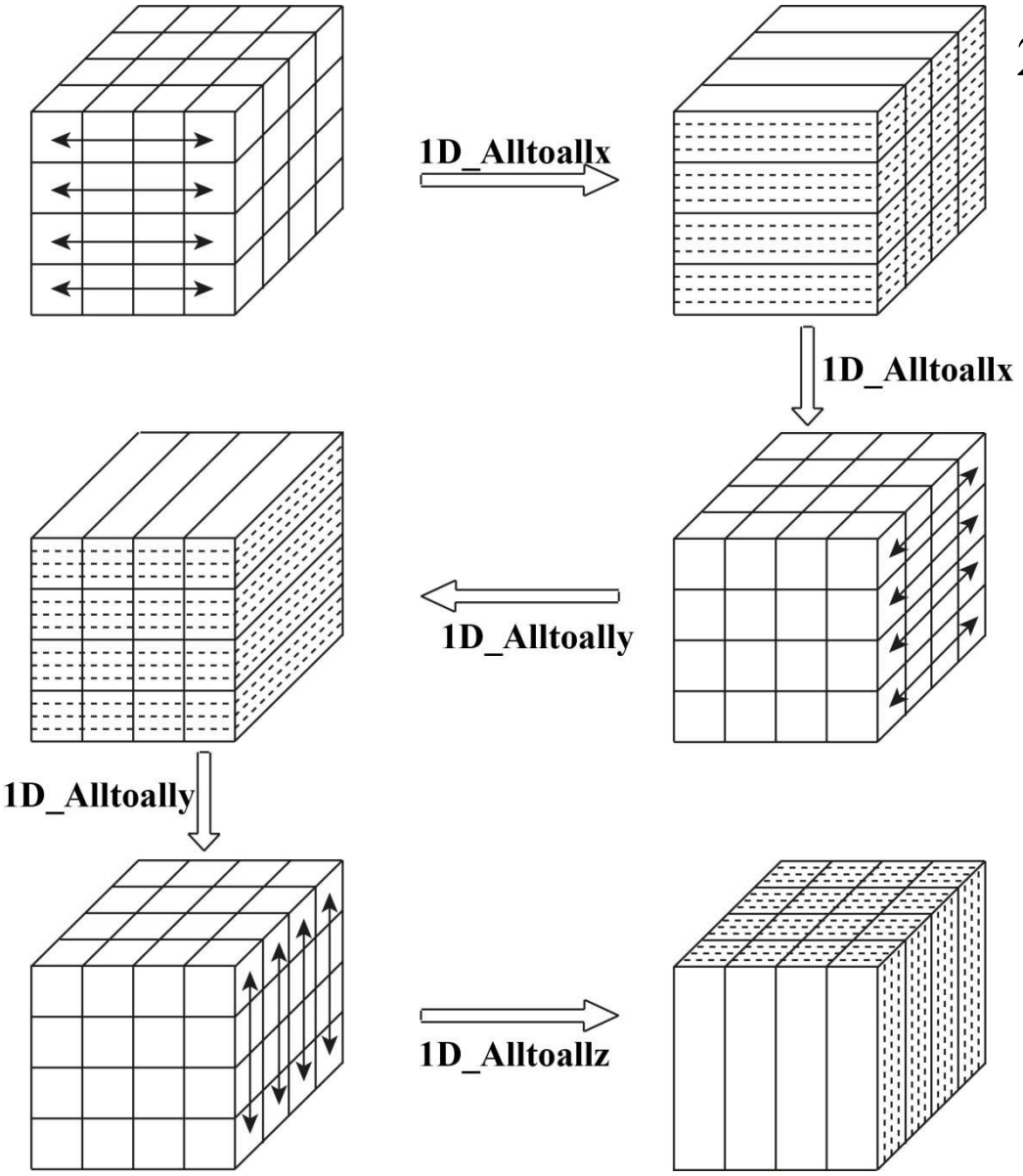
Cell

Boundary cell

1. Midpoint cell method

- Partitioning space into fixed size boxes, with dimension larger than the cutoff distance.
- We need only information of neighbor space(domain) for computation of energies.
- Communication is reduced by increasing process number .
- Efficient for good parallelization and suitable for large system with massively parallel supercomputers.

GENESIS

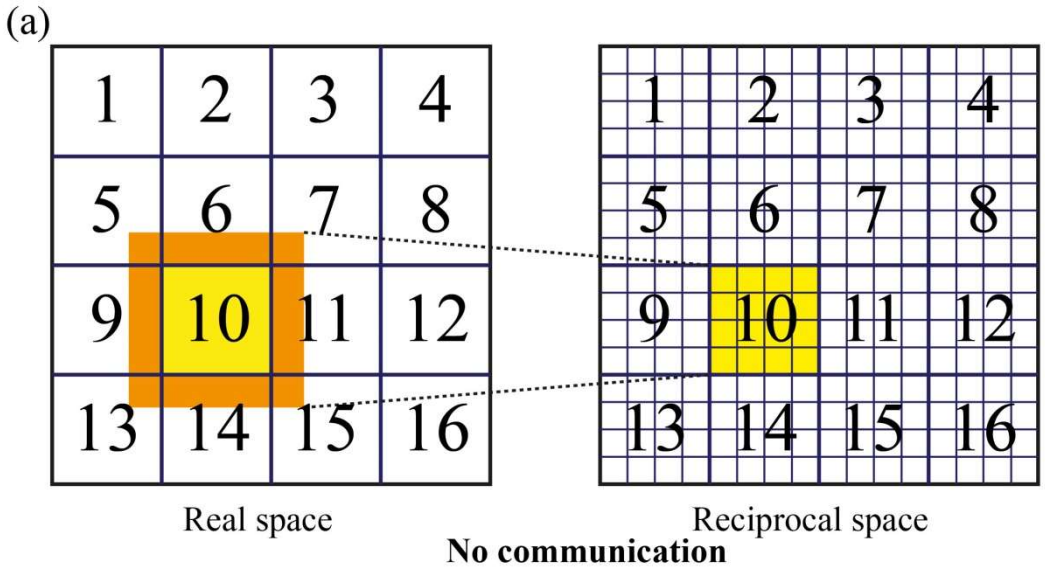


2. Volumetric decomposition FFT

- More communications than existing FFT
- MPI_Alltoall communications only in one dimensional space
- Reduce communicational cost for large number of processors

Ref) J. Jung et al. Comput. Phys. Comm. 200, 57-65 (2016)

FFT in GENESIS (2 dimensional view)

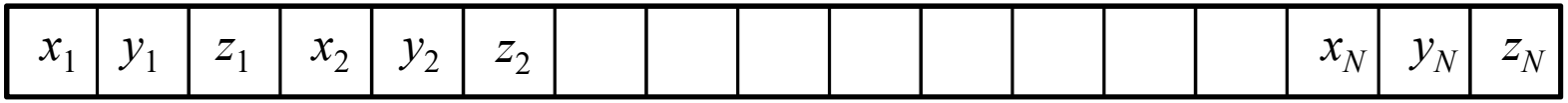


GENESIS
(Identical domain decomposition between two space)

NAMD, Gromacs
(Different domain decomposition)

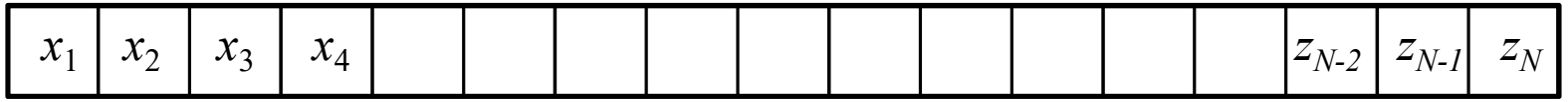
SIMD in GENESIS (developer version)

- Array of Structure (AoS)



In GENESIS, it is expressed as `coord_pbc(1:3,1:natom,1:ncell)`

- Structure of Array (SoA)



In updated GENESIS source code for KNL, it will be expressed as
`coord_pbc(1:natom,1:3,1:ncell)`

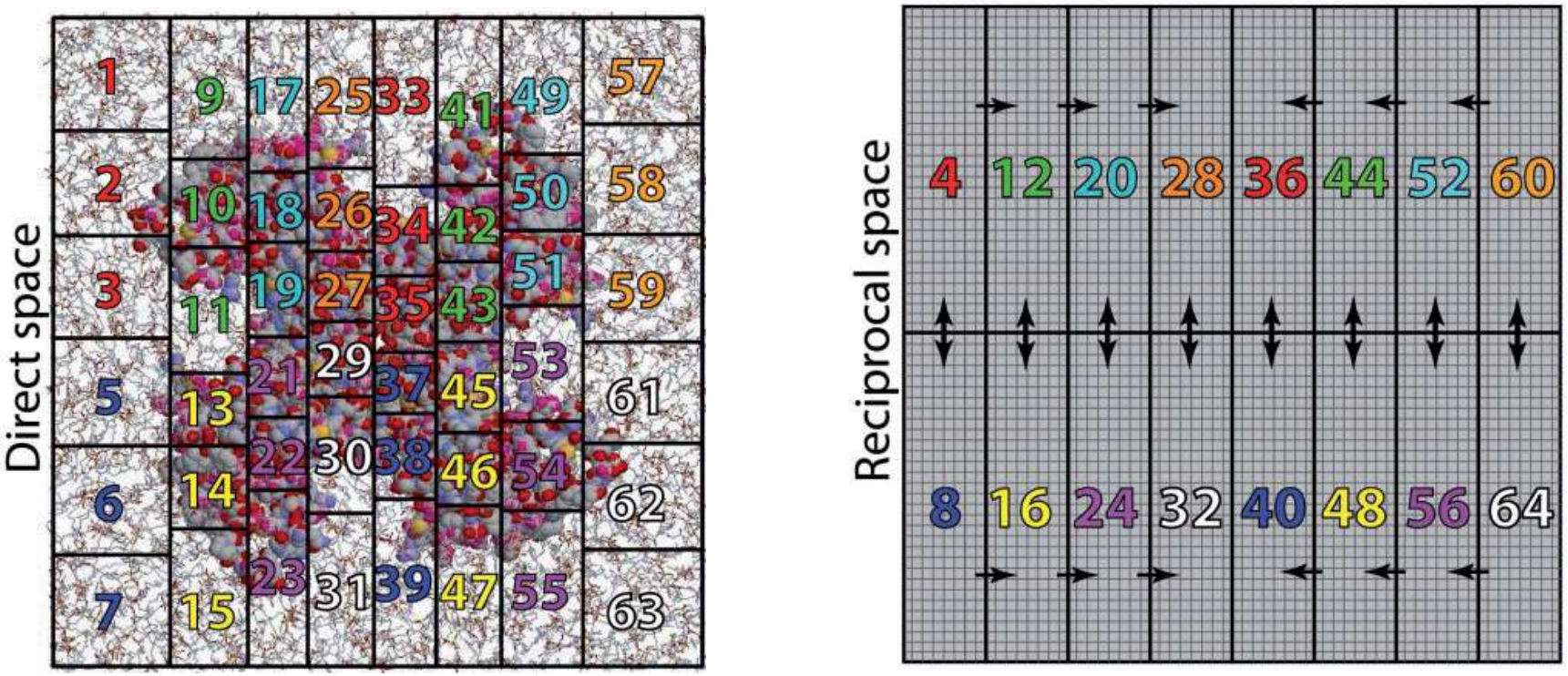
- For Haswell/Broadwell/KNL machines, SoA shows better performance than AoS due to efficient vectorization (SIMD)

Further improvements in Parallelization of MD :

MPMD (Multiple Program and Multiple Data)

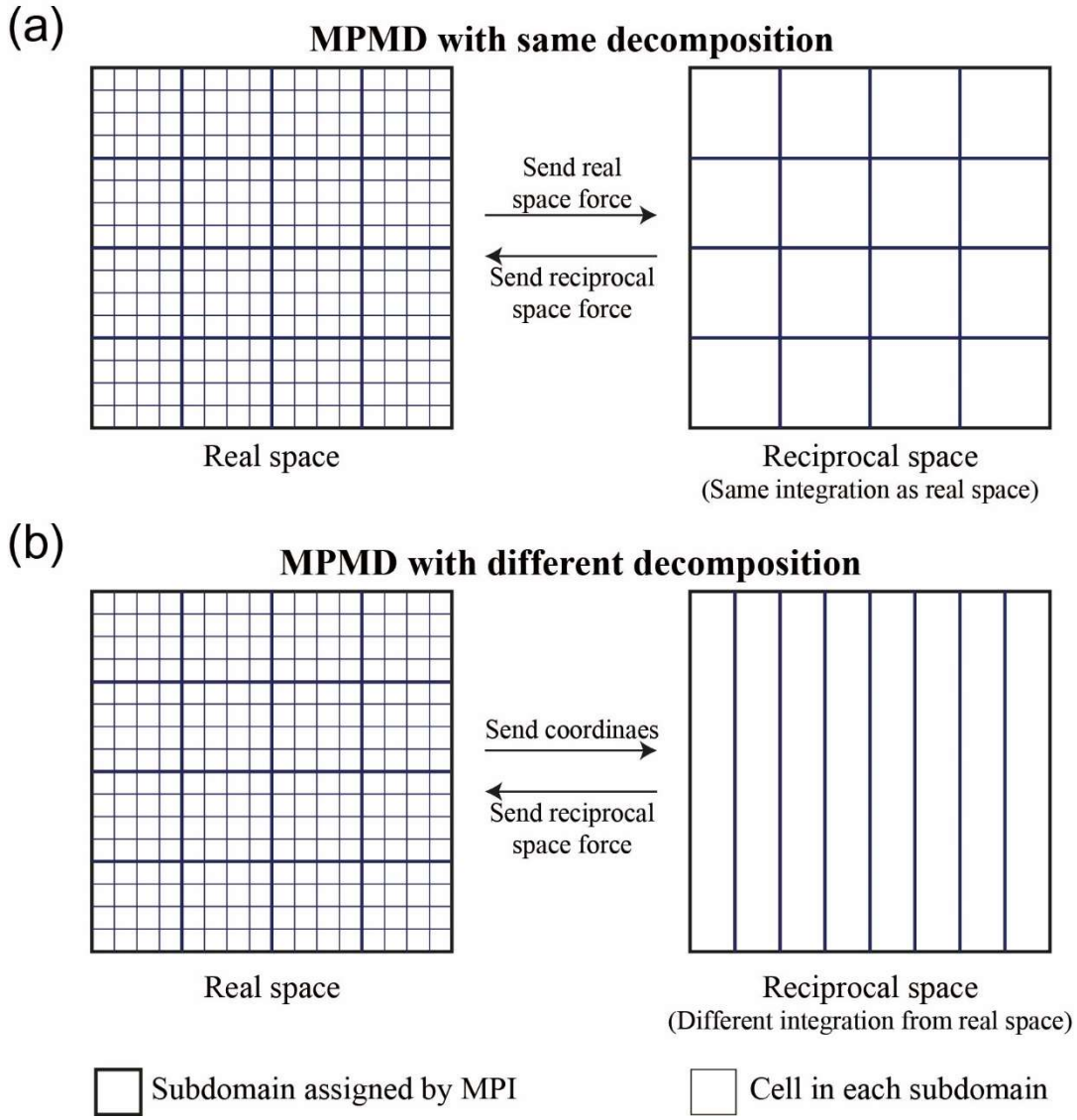
Multiple program/multiple data (MPMD) scheme

1. A subset of nodes are dedicated to real space and another set of nodes are dedicated to reciprocal space interactions.
2. Representative program : Gromacs, CHARMM.
3. Usually different domain decomposition of real space is different from that of reciprocal space => reciprocal space nodes is used only for calculation (not integration).



MPMD scheme of Gromacs : 3D domain decomposition in real space combined with 2D pencil decomposition in reciprocal space (ref : Bioinformatics, btt055 (2013)

What is the problem of existing MPMD scheme?



1. Different domain decomposition between real and reciprocal space => Large amount of communication

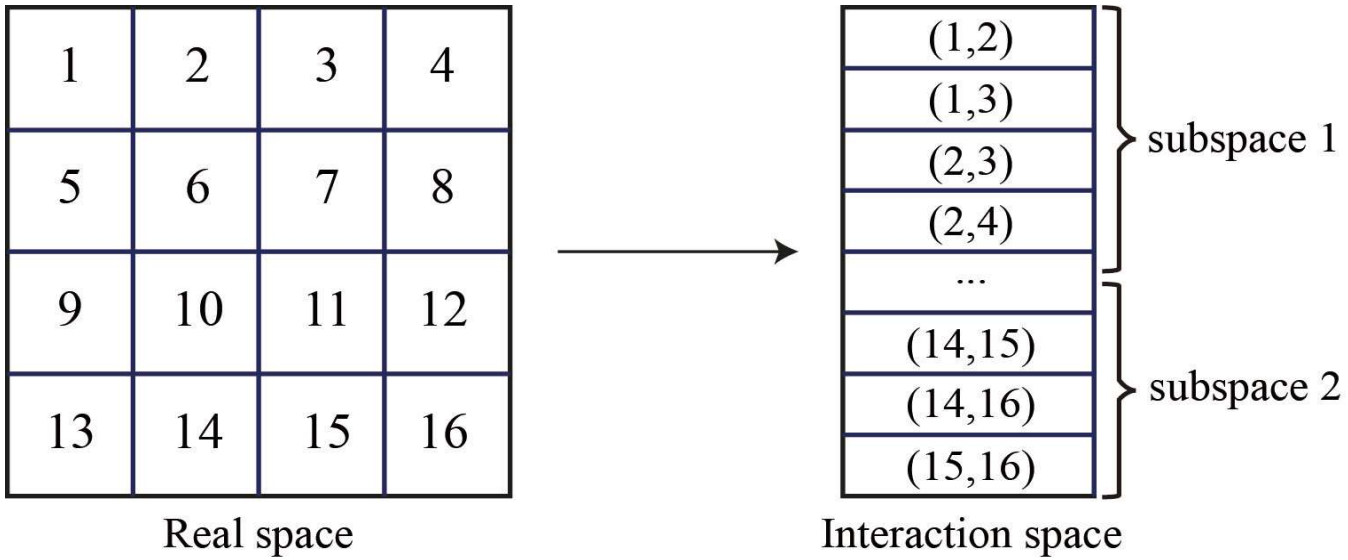
2. Integration of reciprocal space is not possible => RESPA cannot be implemented.

3. Communication pattern between real and reciprocal nodes is dependent on the number of processors

⇩

To overcome the problem, we suggest a new MPMD scheme where real and reciprocal space have same domain decomposition.

MPMD scheme with multiple time step integrator

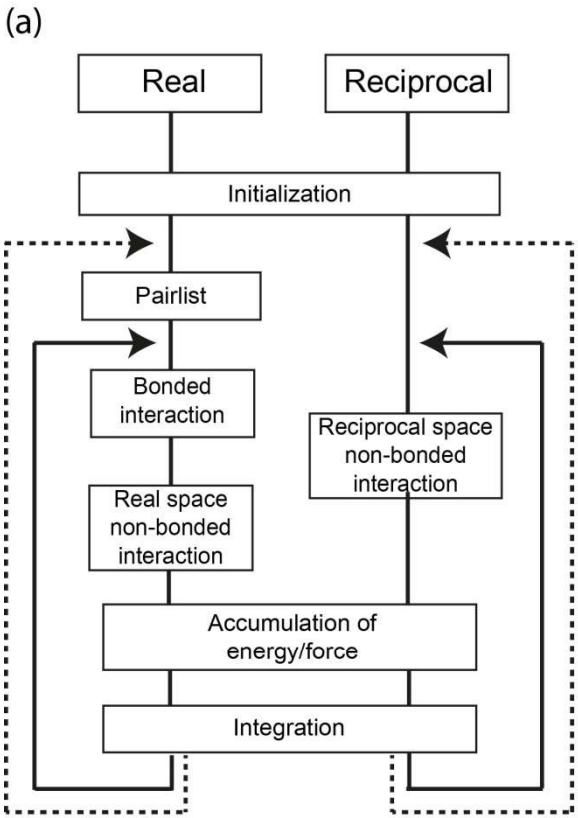


1. Both real and reciprocal space nodes are involved in integrator (it is possible by same domain decomposition between two spaces).

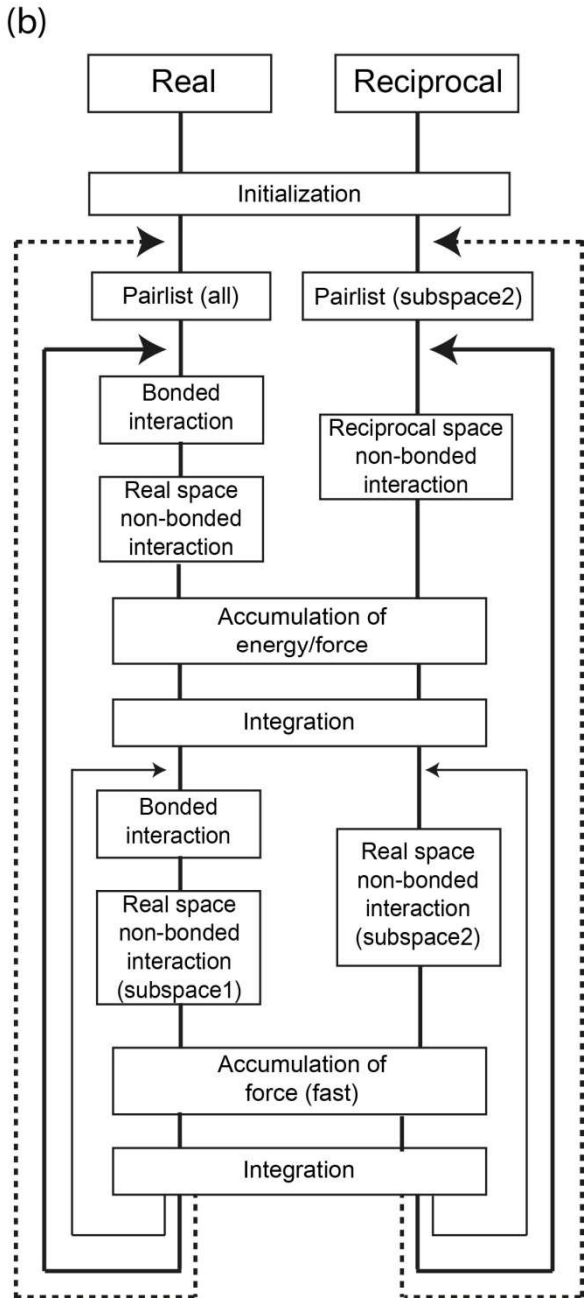
2. Real space non-bonded interactions are divided into two subspaces.

3. When reciprocal space interaction is not necessary, real and reciprocal space nodes are assigned to the evaluation of subspace 1 and subspace 2, respectively.

Flowchart of MPMD scheme



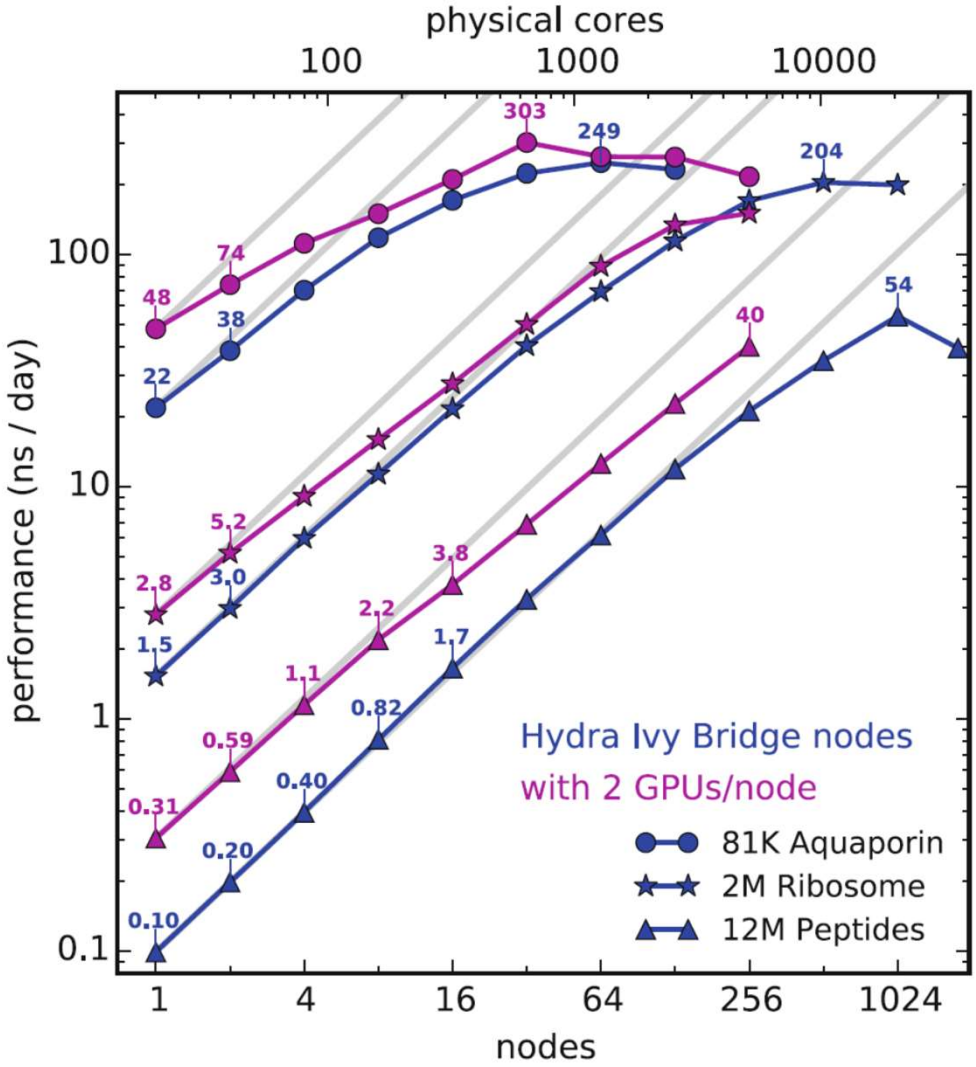
Conventional integrator



Multiple time step integrator

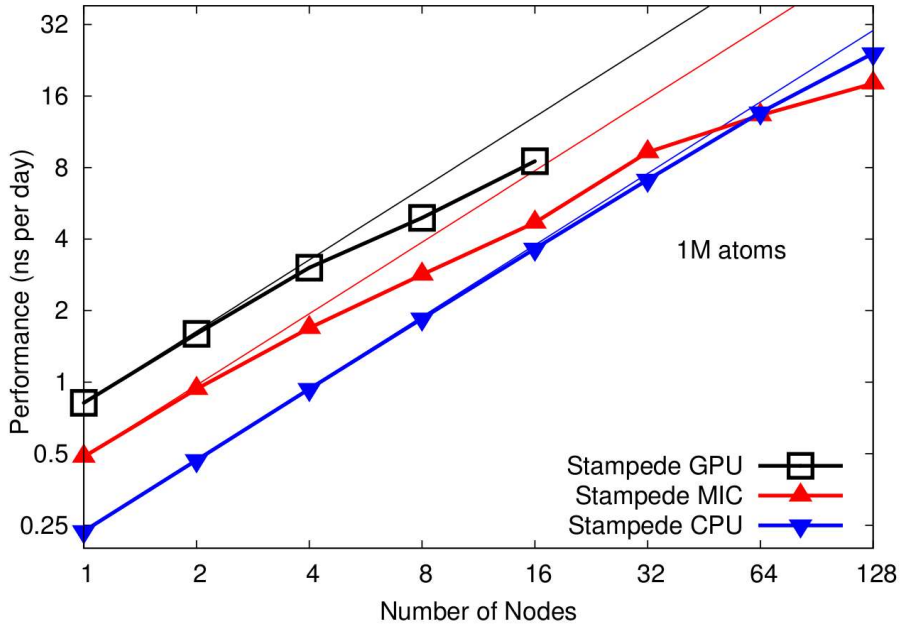
Benchmark performance

Gromacs

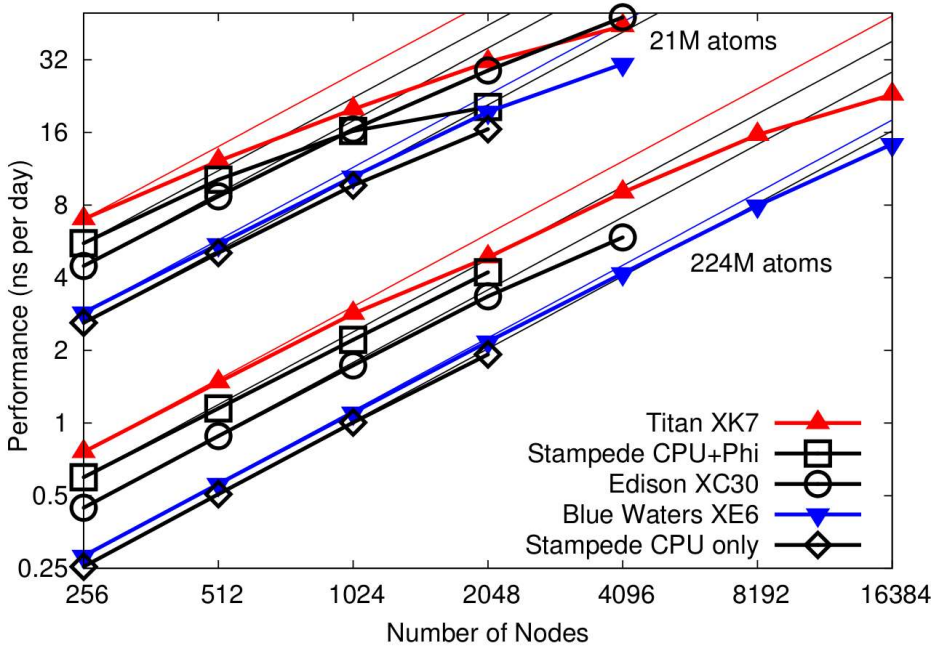


Ref : S. Pall et al., International Conference on Exascale Applications and Software, EASC 2014

NAMD

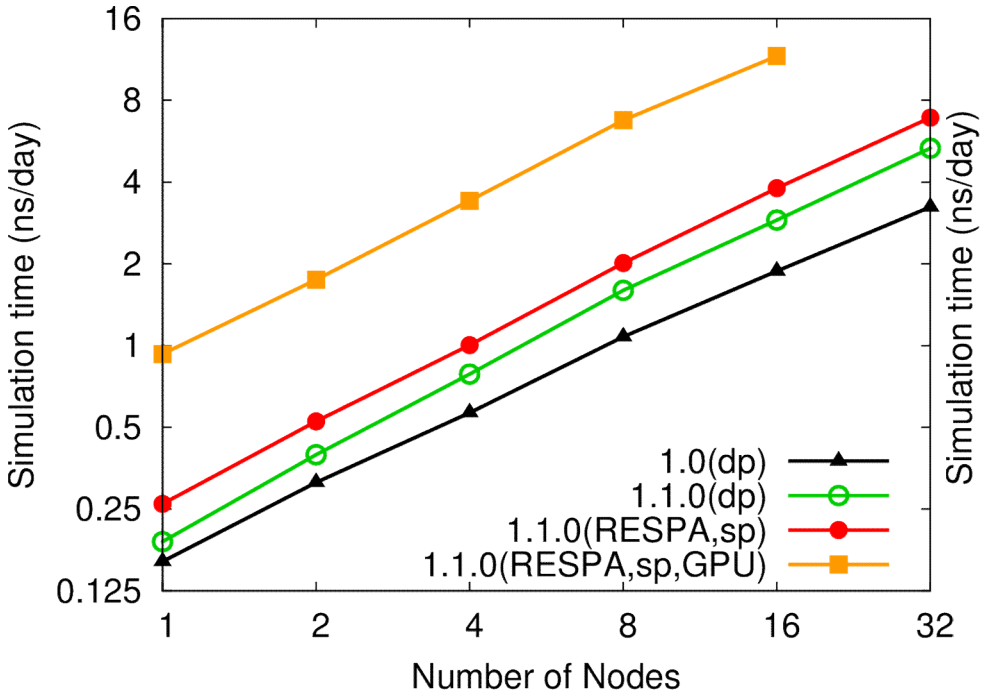


NAMD 2.11 on TACC Stampede
(PME every 3 steps)
(From NAMD webpage)

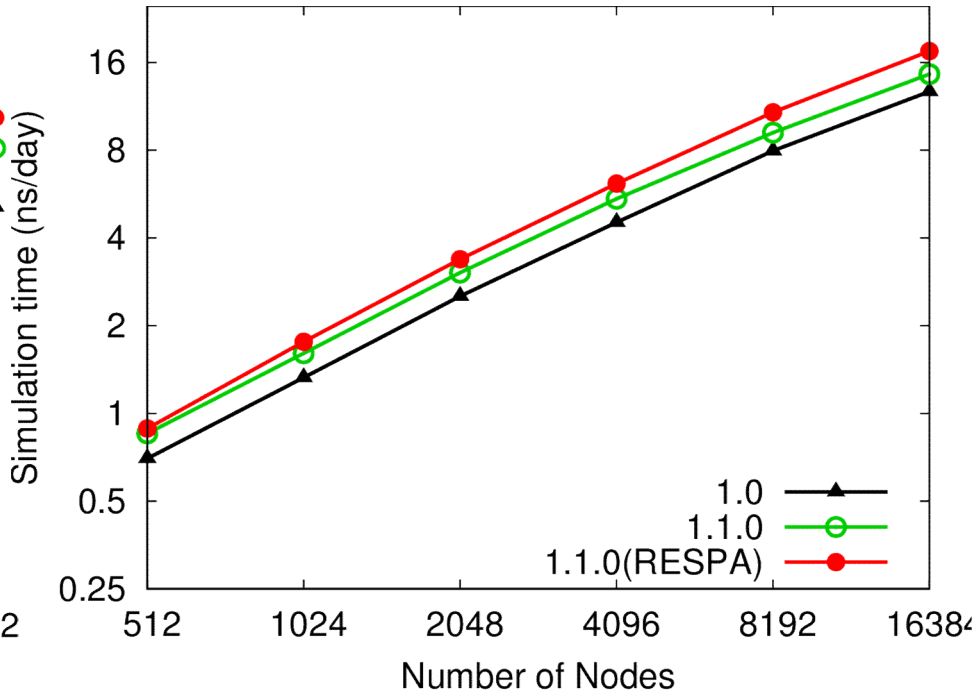


NAMD 2.10
(PME every 3 steps)
(In proceedings of the 2014 International Conference for High Performance Computing, Networking, Storage, and Analysis (SC14))

GENESIS 1.1

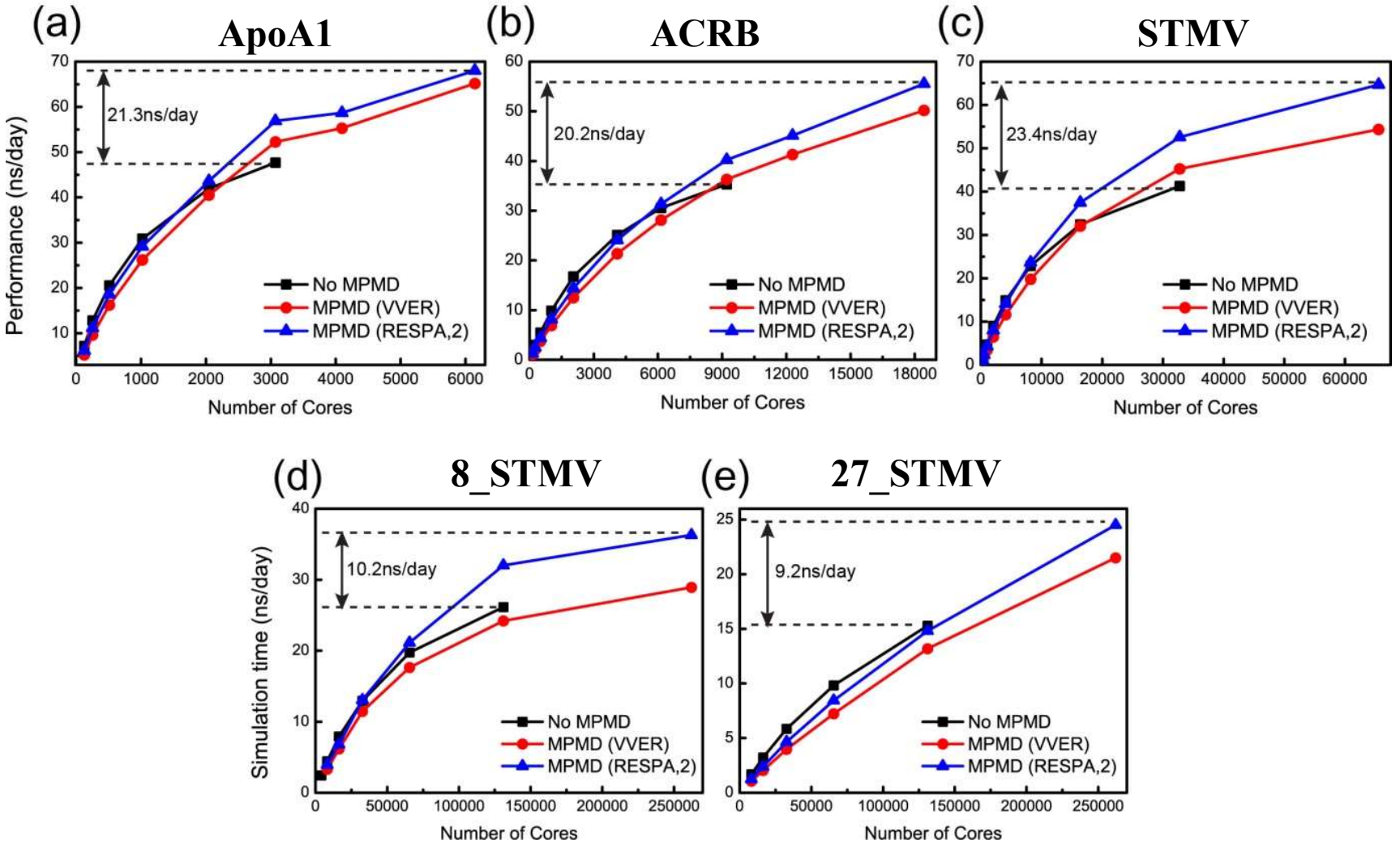


GENESIS 1.1 on PC clusters
(1M system)



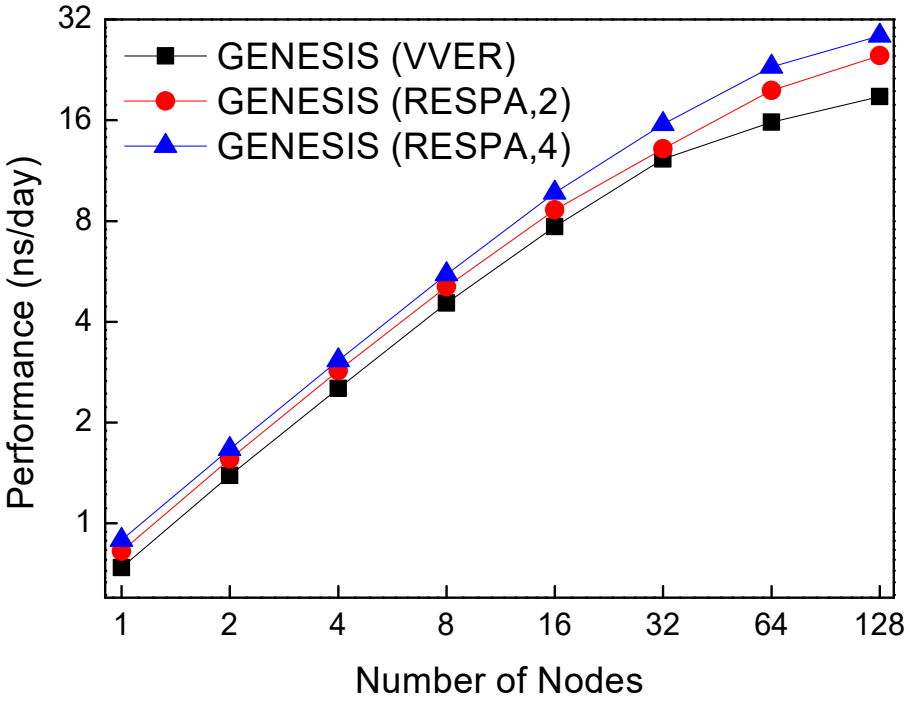
GENESIS 1.1 on K
(28M system)

GENESIS MPMD performance

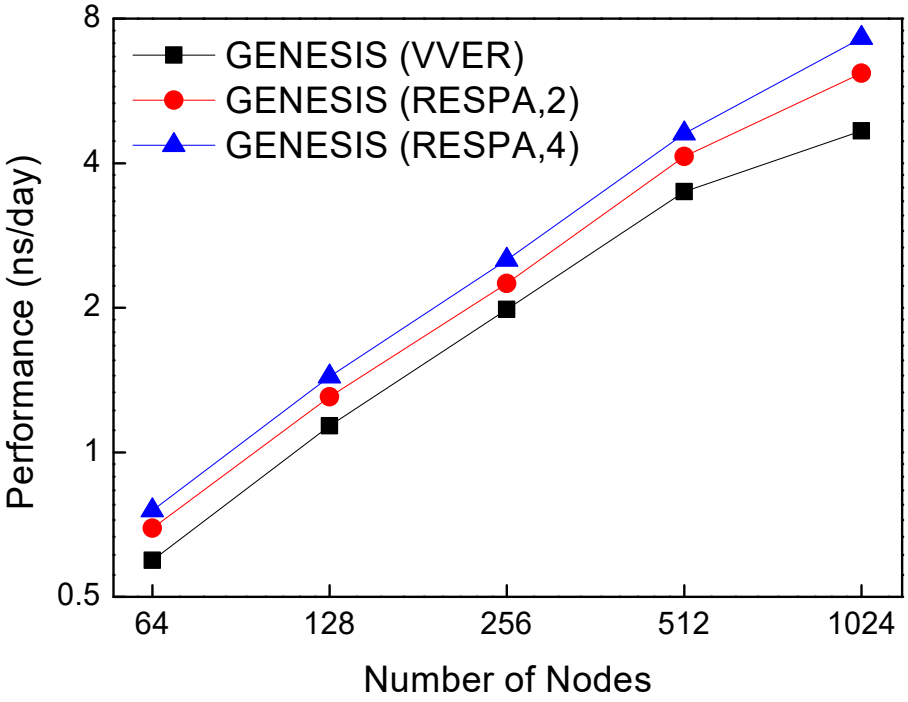


1. MPMD is not suitable for small number of processors.
2. MPMD could be a good solution for very large number of processors.
3. MPMD even increases the available number of processors.

GENESIS performance on KNL



1 STMV (1M)



64 STMV (66M)

Summary

1. Parallelization : Distributed memory (MPI), shared memory (OpenMP), hybrid (MPI+OpenMP), SIMD, SIMT, etc.

2. Real space parallelization => Mainly domain decomposition scheme

3. Reciprocal space parallelization => Parallelization of FFT

4. Further improvements : MPMD, GPU, and so on.