A CPU/GPU Linear Scaling Three Dimensional Fragment Method for Large Scale Electronic Structure Calculations on Titan Supercomputer

Weile Jia[1], Long Wang[1], Xuebin Chi[1], Lin-Wang Wang[2]

Supercomputing Center of Chinese Academy of Science

Lawrence Berkeley National Lab

2015-06-23

Outline

Motivation

LS3DF Algorithm

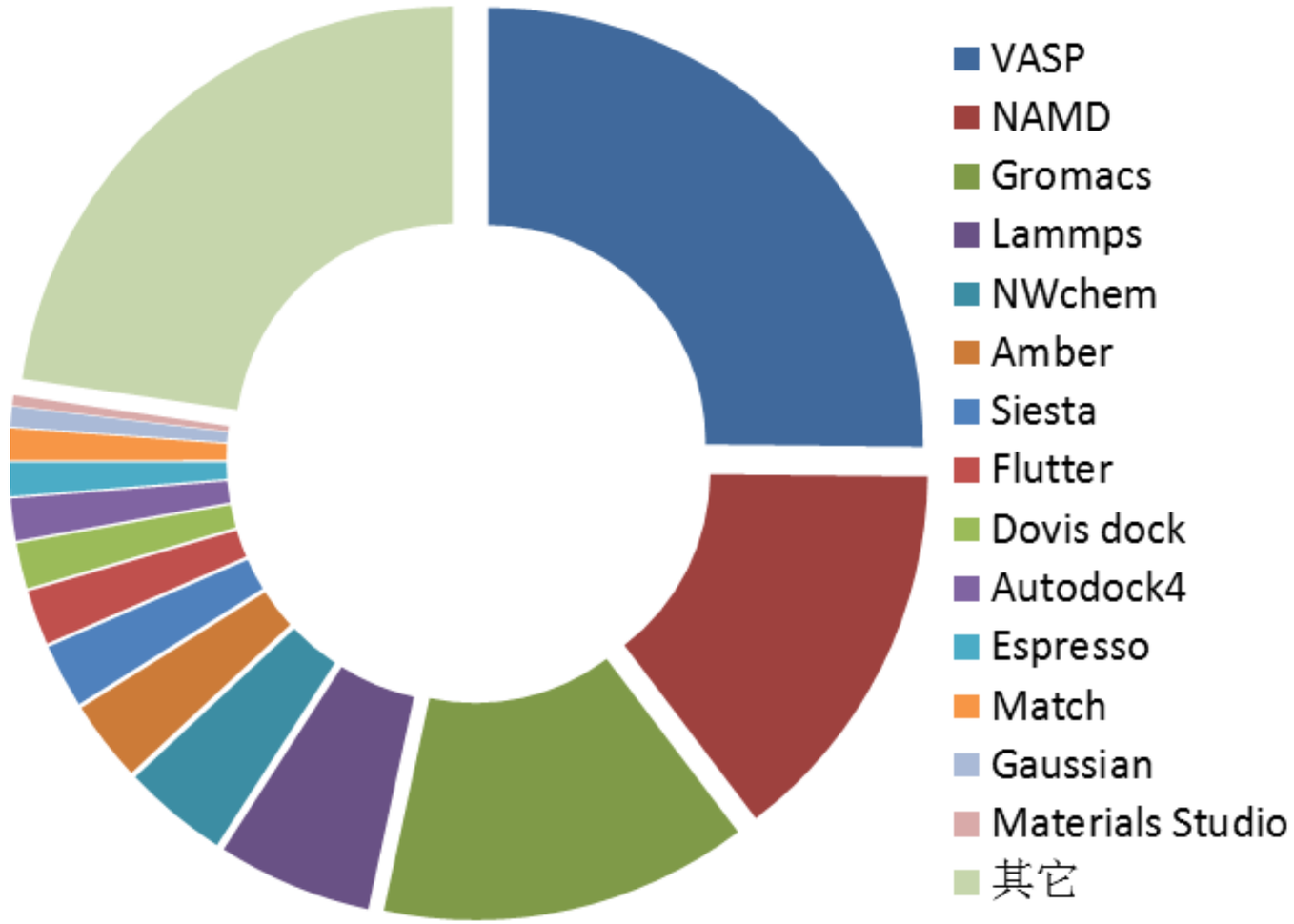
One Fragment on GPU

Testing results

Future work

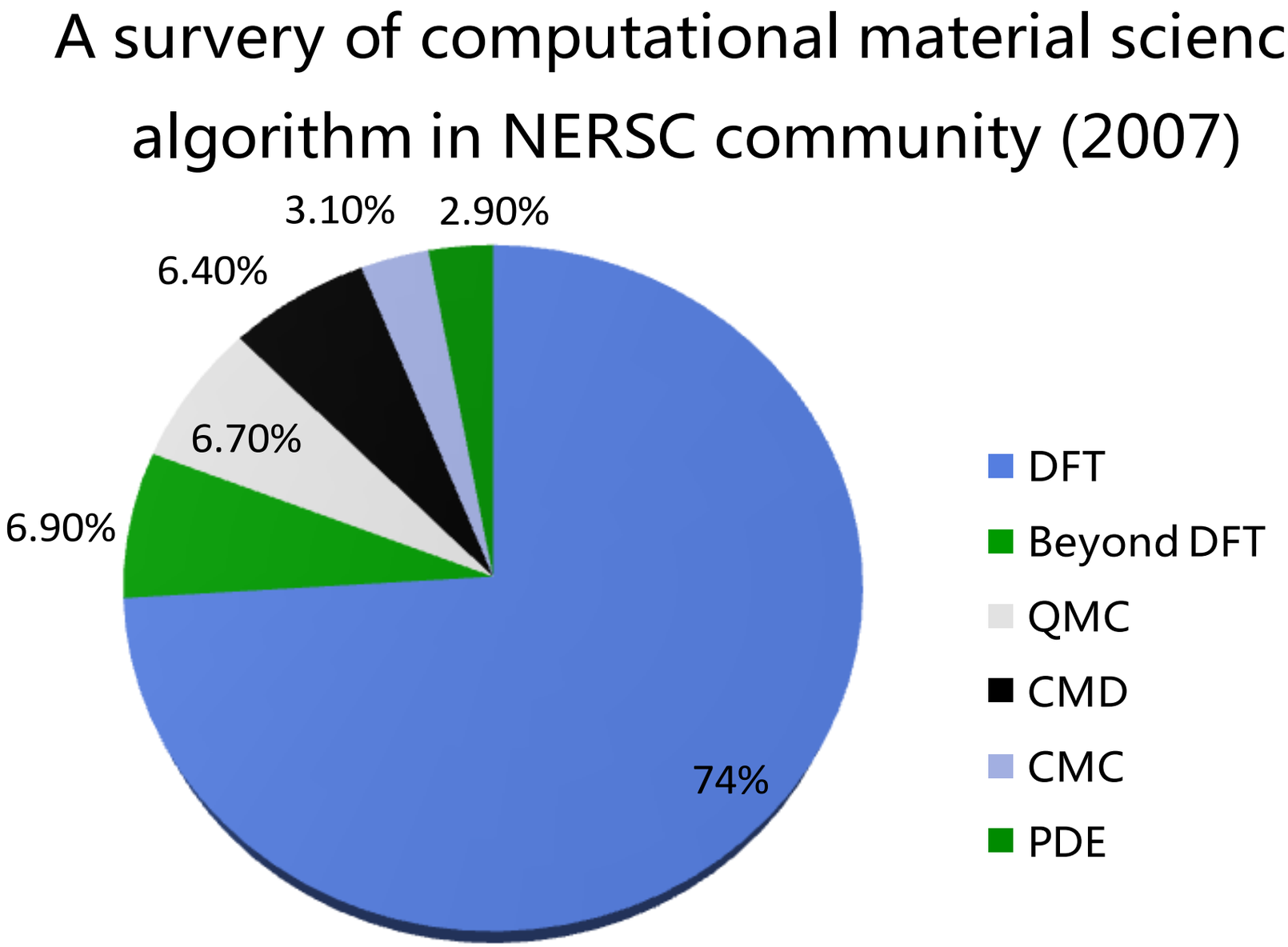
Software in Supercomputing center of CAS

Software	Fields	USER Num	Percentage
VASP	First principle, commercial	65	25.2%
NAMD	MD, open source	14	14.5%
Gromacs	MD, open source	36	13.7%
Lammps	MD, open source	63	5.8%
NWchem	Frist principle, open source	17	3.9%
Amber	MD， Commercial	18	3.0%
Siesta	Material simu, open source	6	2.4%
Flutter	Force simulation	1	2.0%
Dovis dock	Medicine	1	1.7%
Autodock4	Molecular simulation， open source	2	1.6%
Espresso	DFT， open source	12	1.2%
Match	User developed code	1	1.2%
Gaussian	First principle， commercial	59	0.7%
Materials Studio	First principle， commercial	15	0.4%



First principle: 31%

DFT in NERSC community



DFT on Titan

18,688 nodes

16-core AMD Opteron 6274 CPU
1 Nvidia Tesla K20X GPU -
1.31Tflops

Peak Performance 27 Pflops

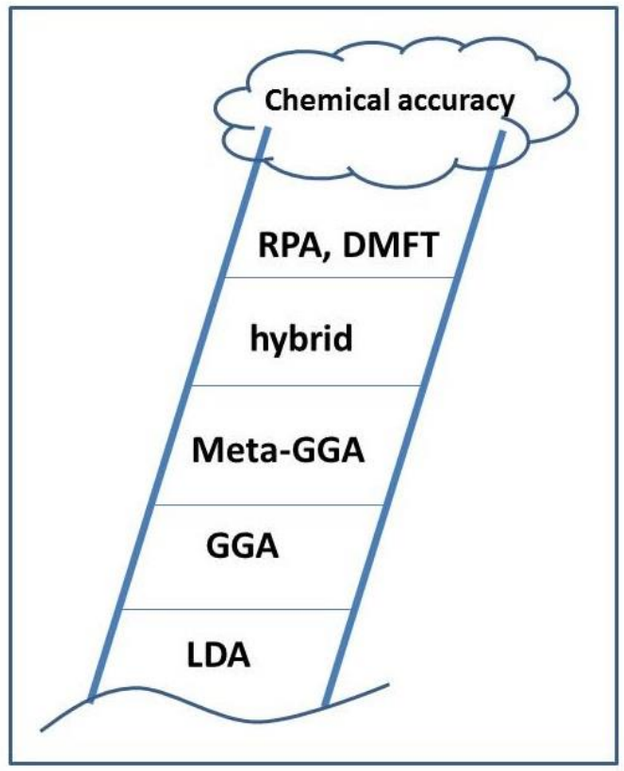
GPU contributes 24Pflop
s

Running CPU application only use 12% of the T
itan computing power.

Three main challenges for ab initio material simulations

(1) Accuracy

(climb Jacob's ladder)



(2) Temporal scale (from fs to seconds)

(new algorithms, like the accelerated MD)

(3) Size scale (mesoscale problems)

(Divide & Conquer methods)

All can be helped by exascale computing

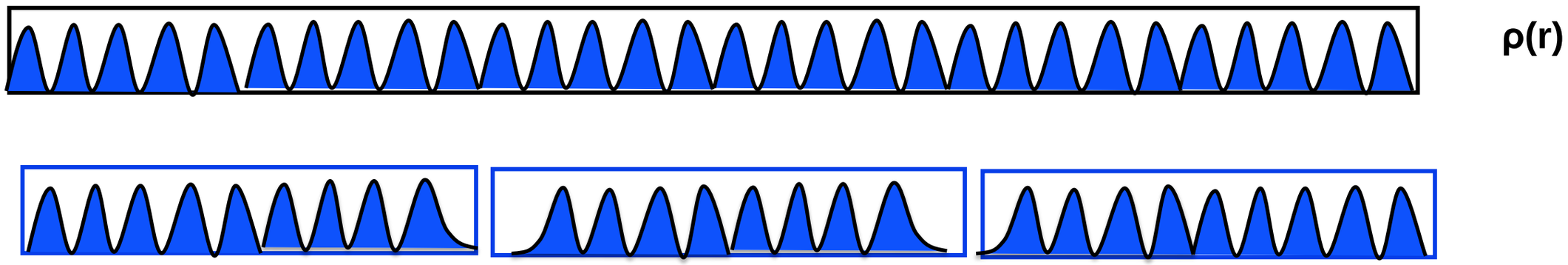
L.W. Wang, Divide and conquer quantum mechanical material Simulations with exascale supercomputers, Nat. Sci. Rev. 2014.

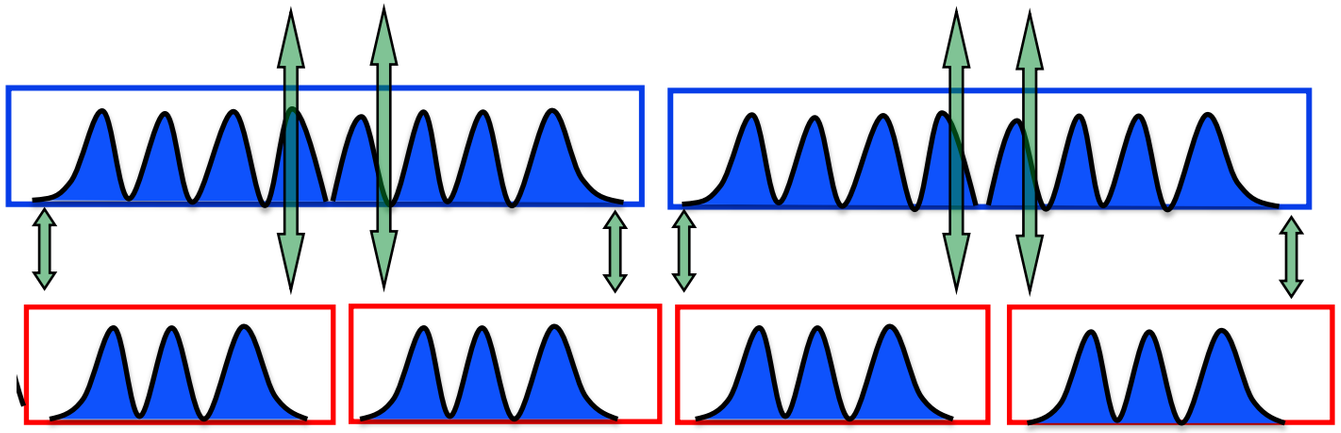
What is LS3DF?

divide-and-conquer method → ***O(N)* scaling**
Massively parallelizable

- A novel divide and conquer scheme with a new approach for patching the fragments together
- No spatial partition functions needed
- Uses overlapping positive and negative fragments
- New approach minimizes artificial boundary effects

LS3DF: 1D

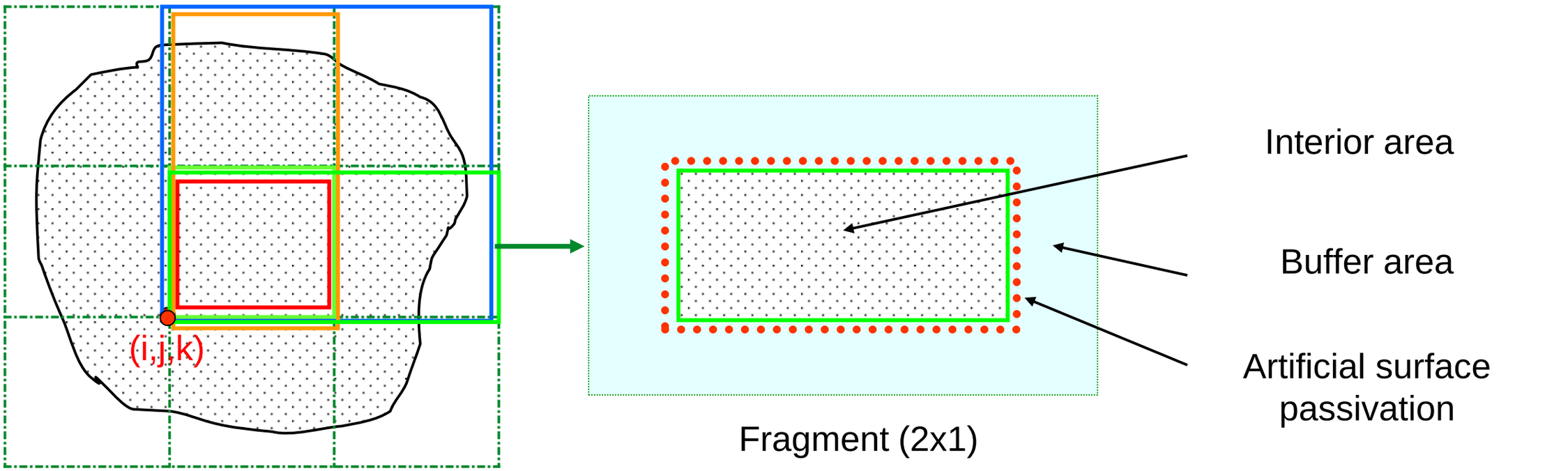




Total = Σ_F { F — F }

Phys. Rev. B 77, 165113 (2008); J. Phys: Cond. Matt. 20, 294203 (2008)

LS3DF in 2D and 3D

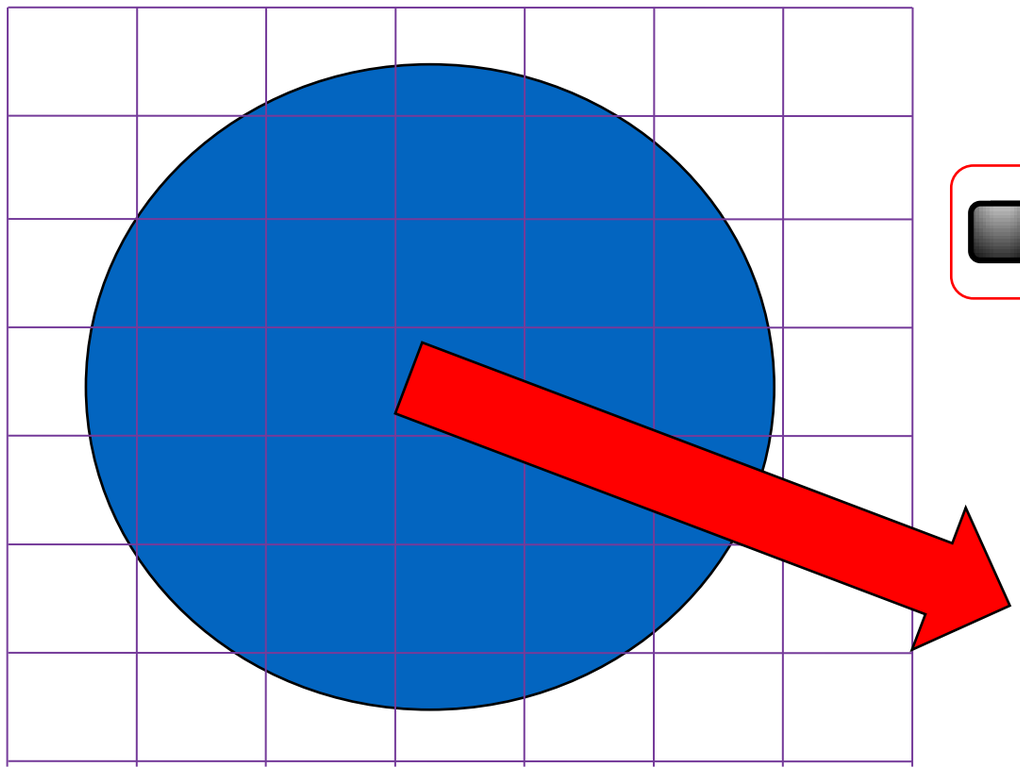


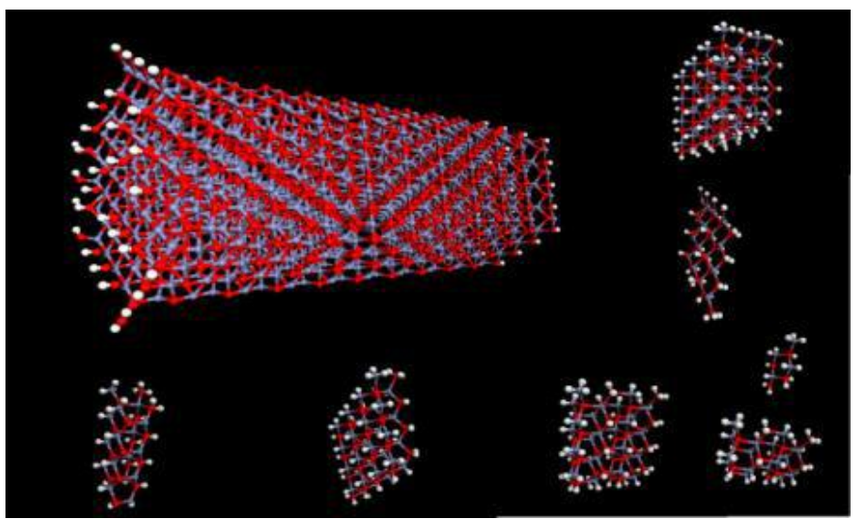
Total = Σ_F { F − F − F + F **}**

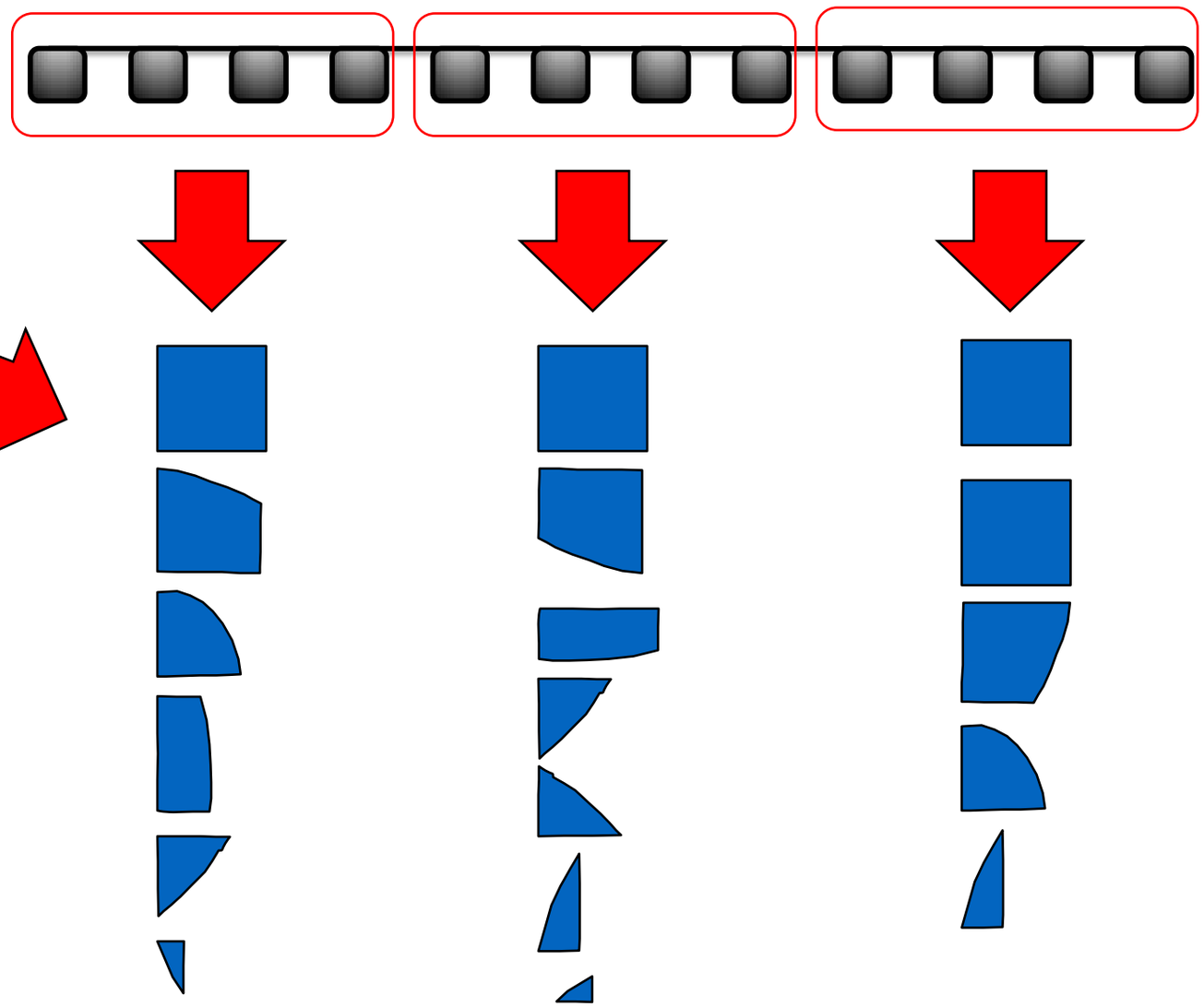
Boundary effects are (nearly) cancelled out between the fragments

$$System = \sum_{i,j,k} \{F_{222} + F_{211} + F_{121} + F_{112} - F_{221} - F_{212} - F_{122} - F_{111}\}$$

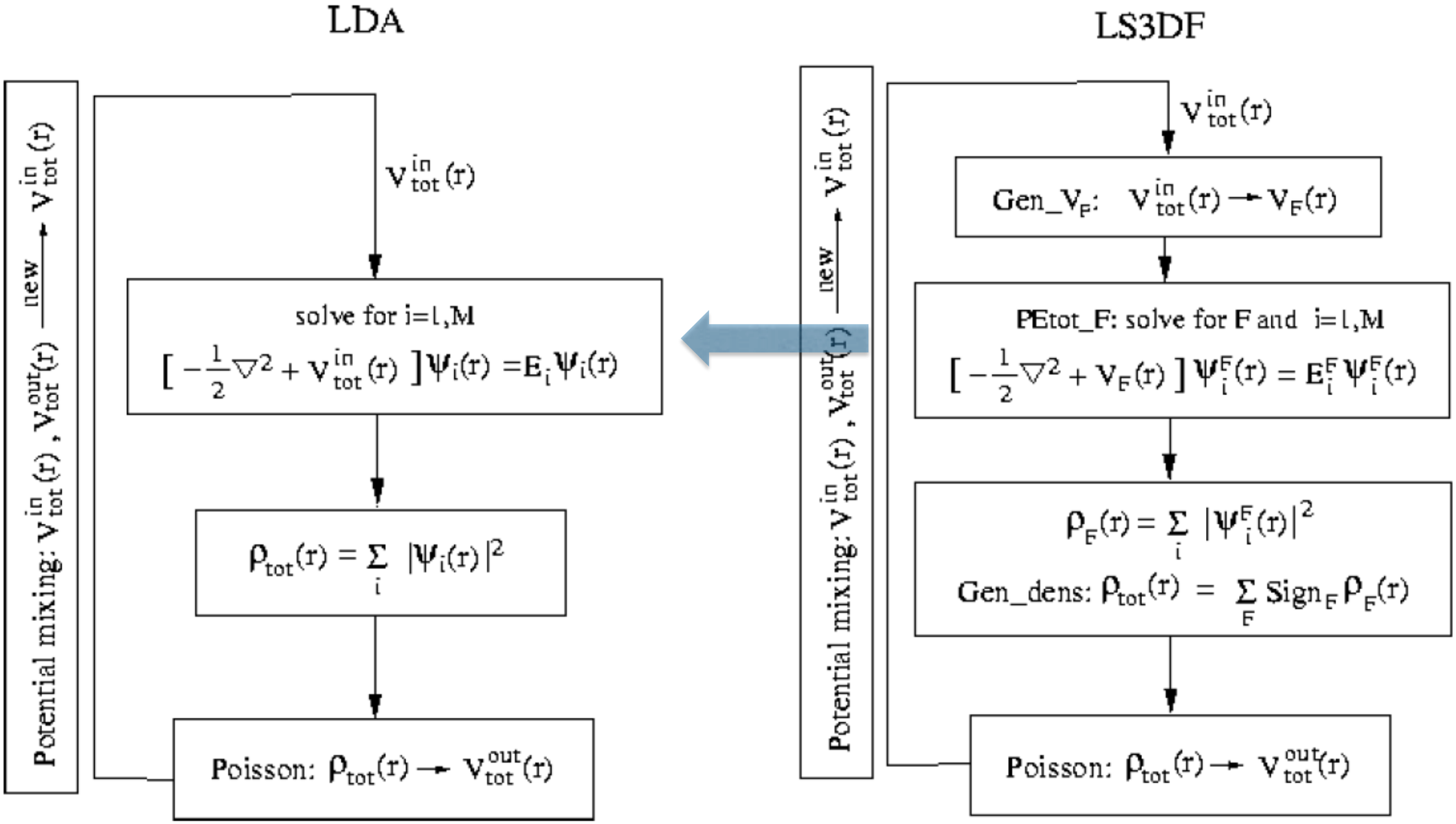
Schematics for LS3DF calculation





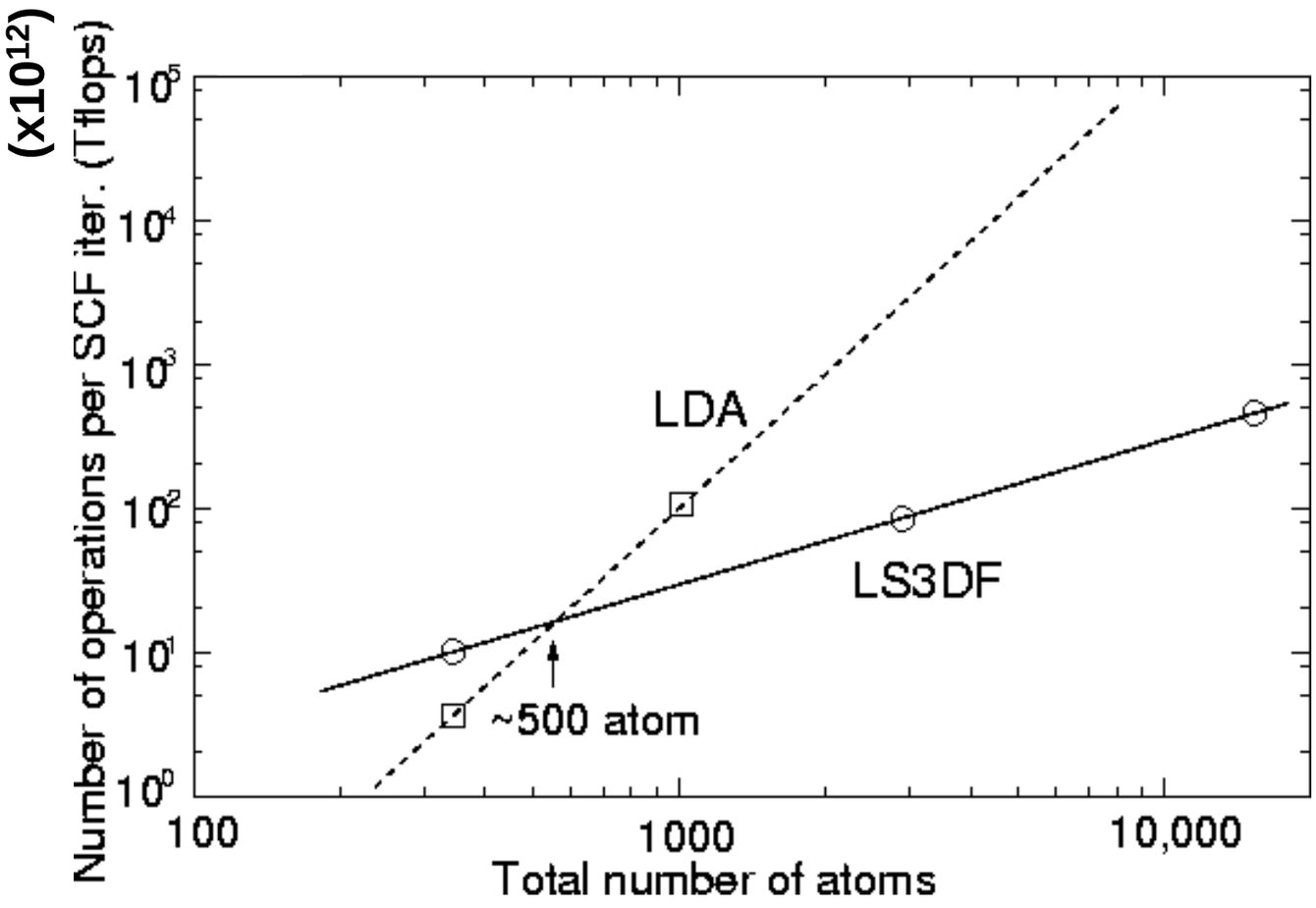


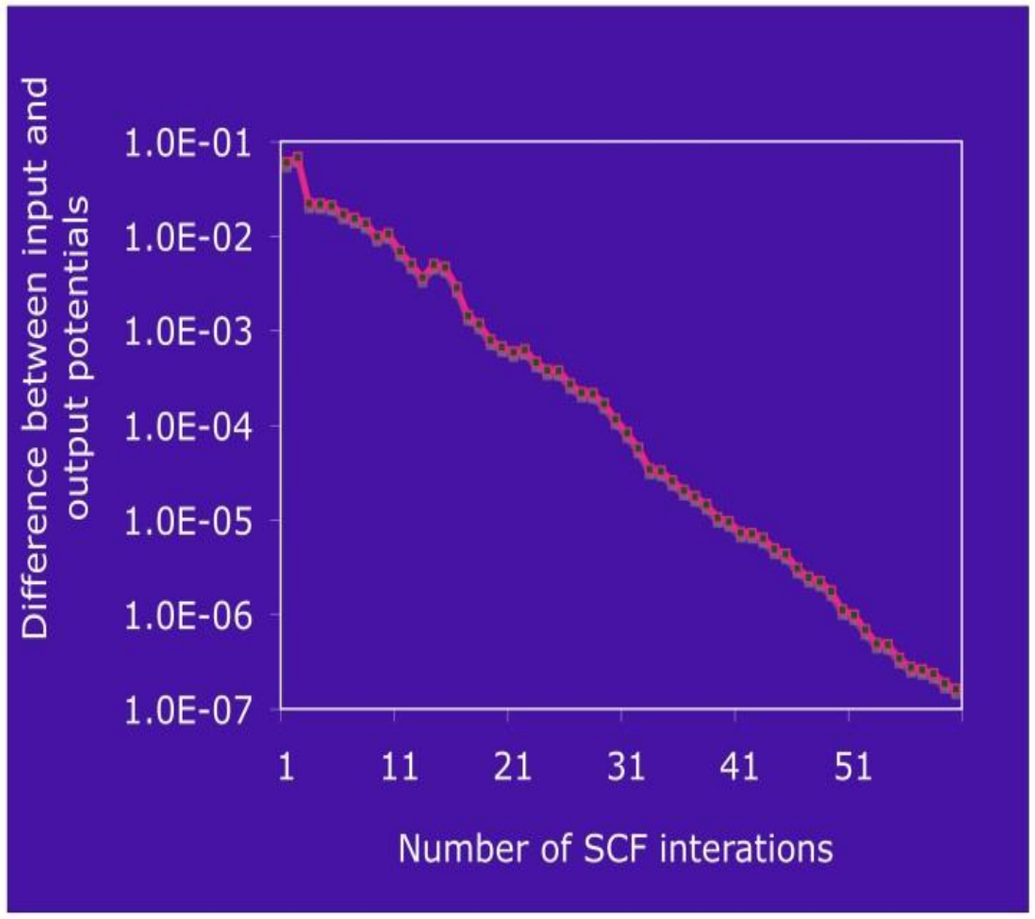
Flow chart of LS3DF



Based on the plane wave PEtot code: http://hpcrd.lbl.gov/~linwang/PEtot/PEtot.html

Operation counts and convergence

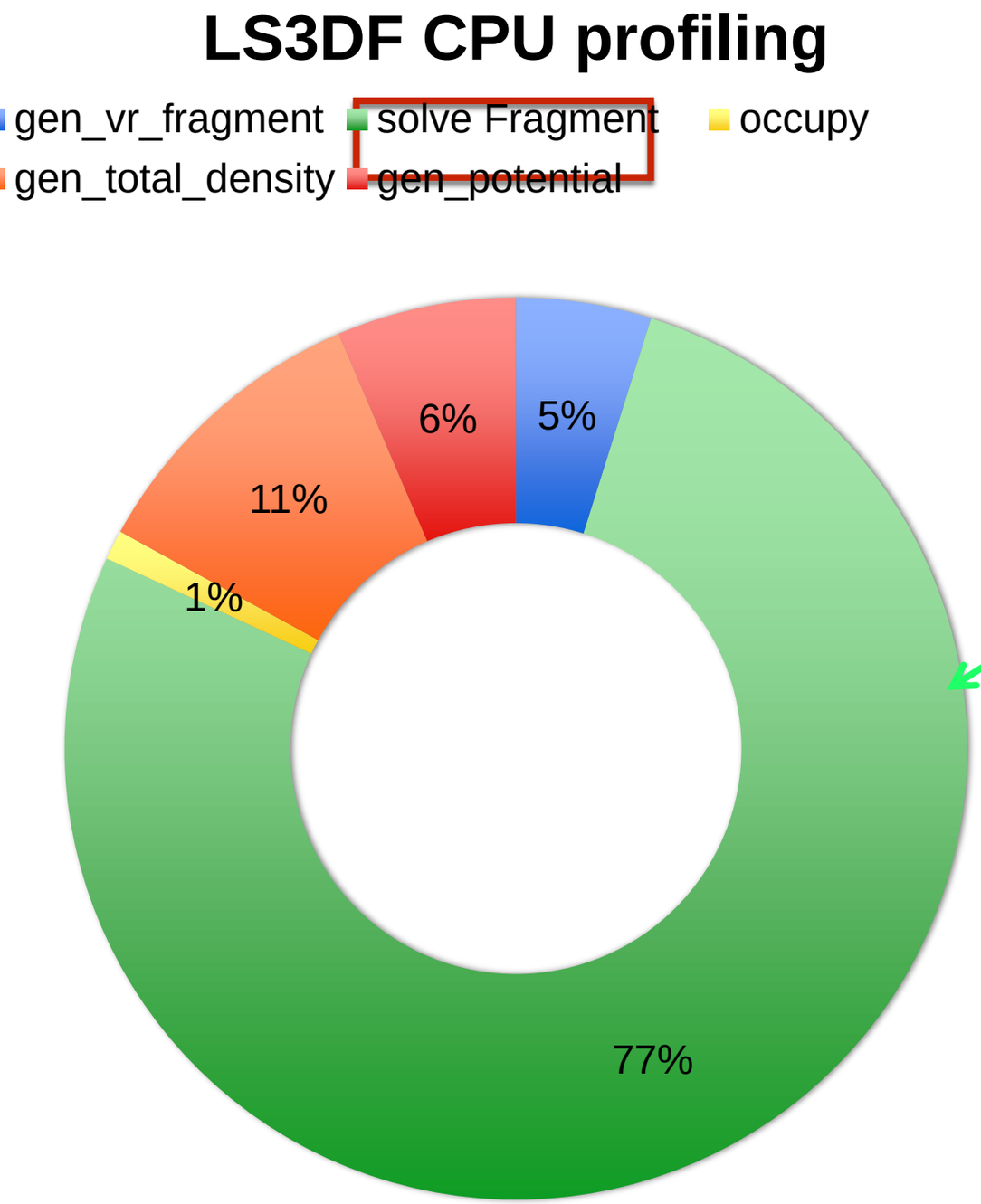


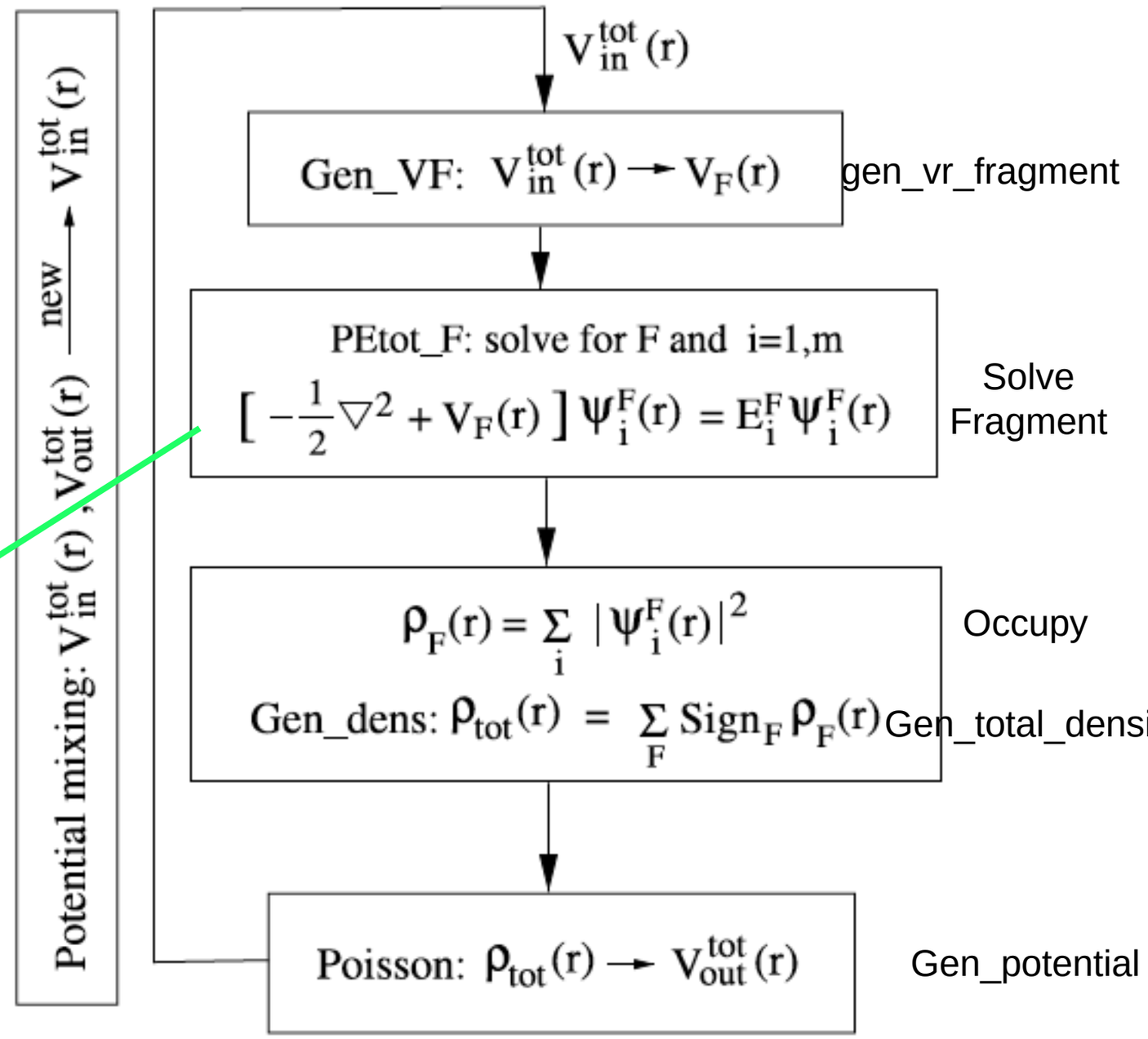


- **Cross over with direct LDA method [PEtot] is 500 atoms.**
- **Similar to other O(N) methods.**

Convergence of the LS3DF code –el ectronic structure SCF

CPU profiling





One Fragment on GPU

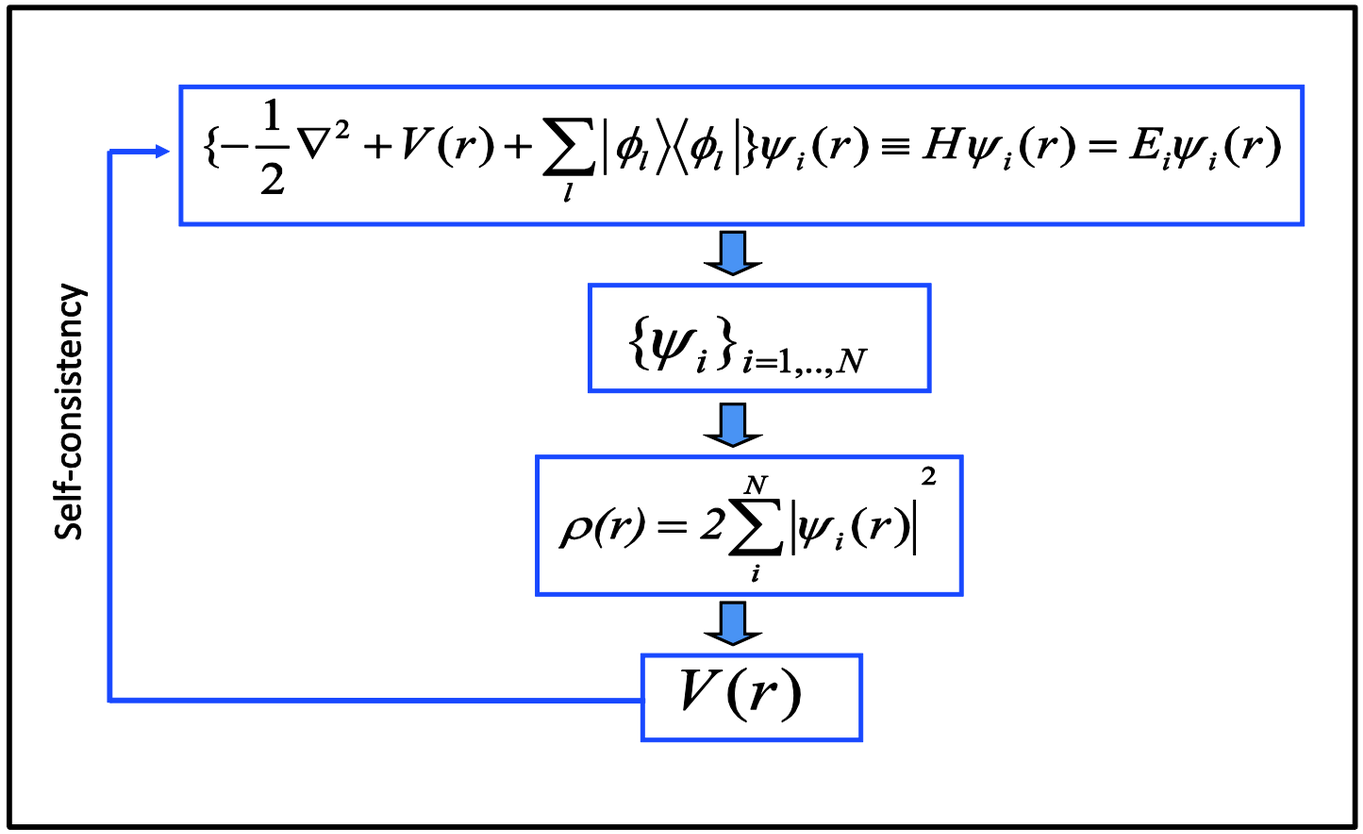
DFT algorithm

$$[-\frac{1}{2}\nabla^2 + V_{tot}(r)]\psi_i(r) = \varepsilon_i\psi_i(r)$$

- If the size of the system is N:
- N coefficients to describe one wavefunction

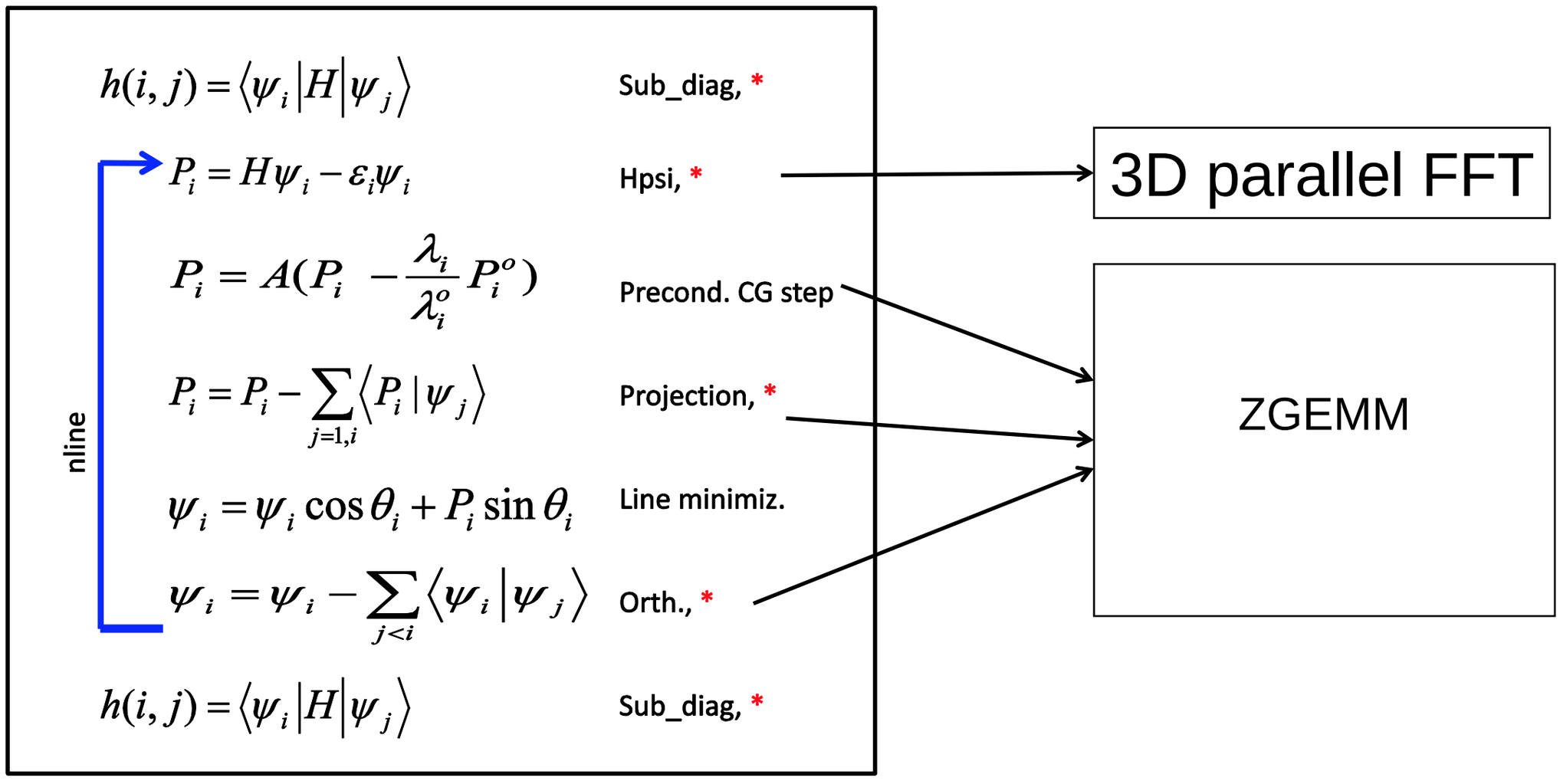
$$H\psi_i = \varepsilon_i\psi_i$$

SCF : Self-consist field



The flow chart of a DFT calculation. The DFT formula (e.g., local density approximation) is used to calculate V(r) from P(r). There are N electron wave functions $\Psi_{i,}$ where 2N is the number of total valence electrons in the system.

DFT algorithm

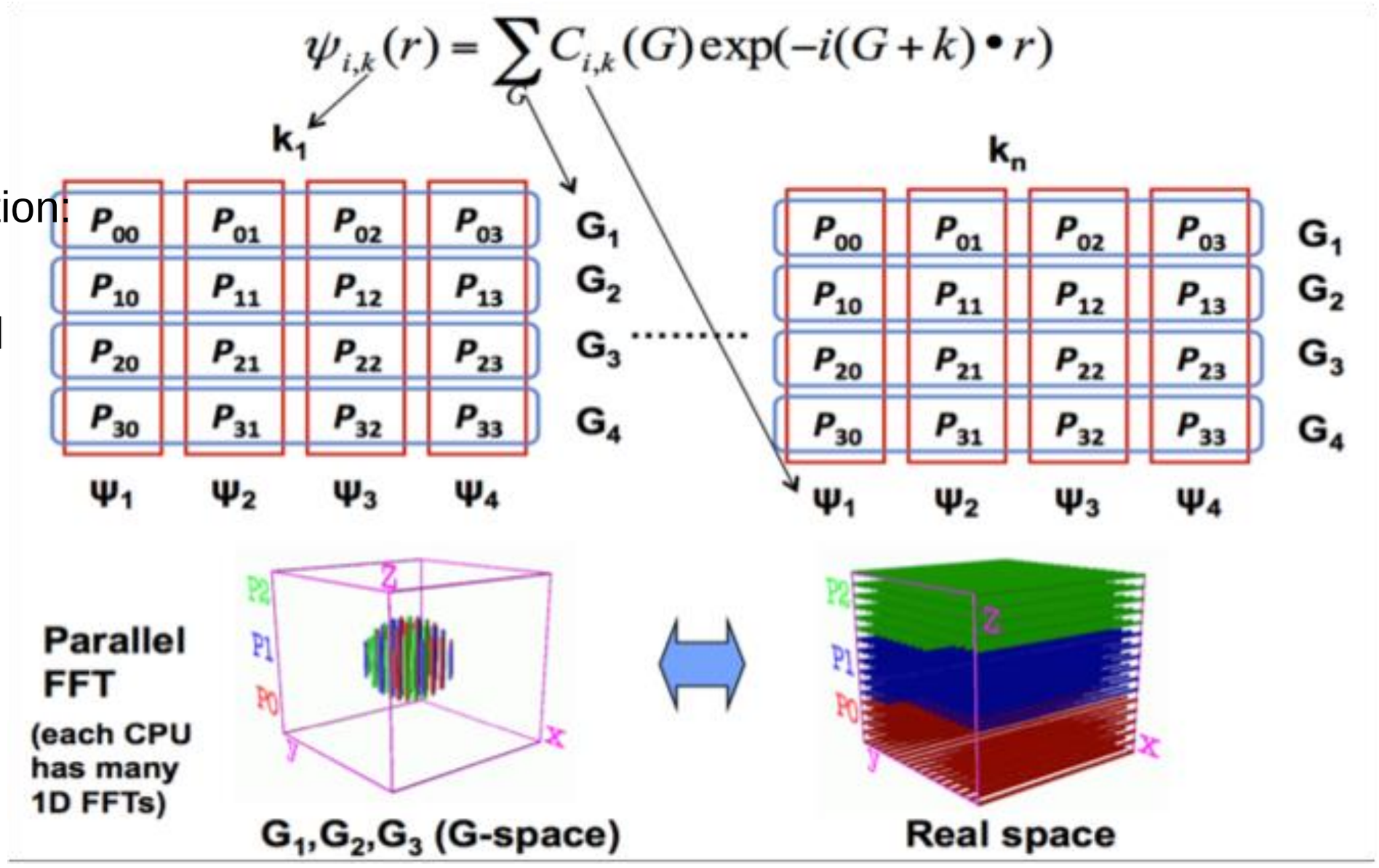


The all-band CG (AB-CG) method for $H\Psi_i=\varepsilon_i\Psi_i$. The time consuming steps are indicated by the asterisk sign. The other parts will be called collectively as the Fortran-do-loops.

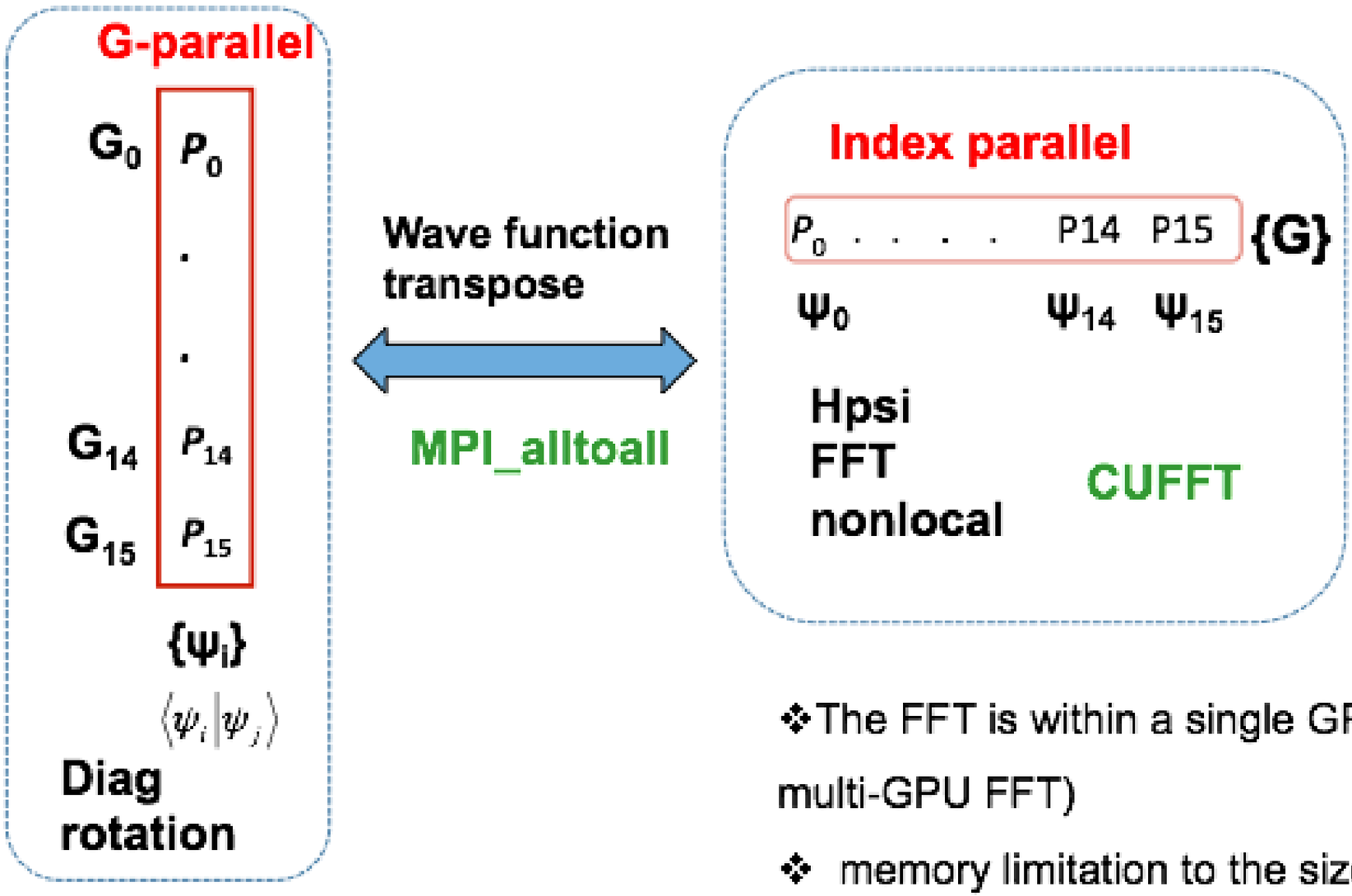
CPU parallelization

3-levels of parallelization:

- K-point parallel
- Band-index parallel
- G-space parallel



Hybrid GPU parallelization



❖The FFT is within a single GPU (no multi-GPU FFT)

❖ memory limitation to the size: a few thousand atoms

For GPU:

Original paralllelization are too fragmented for FFT

Too much communication

Memory copy between CPU-GPU

$$h(i,j) = \langle \psi_i | H | \psi_j \rangle$$ Sub_diag, *

$$P_i = H\psi_i - \varepsilon_i \psi_i$$ Hpsi, *

$$P_i = A(P_i - \frac{\lambda_i}{\lambda_i^o} P_i^o)$$ Precond. CG step

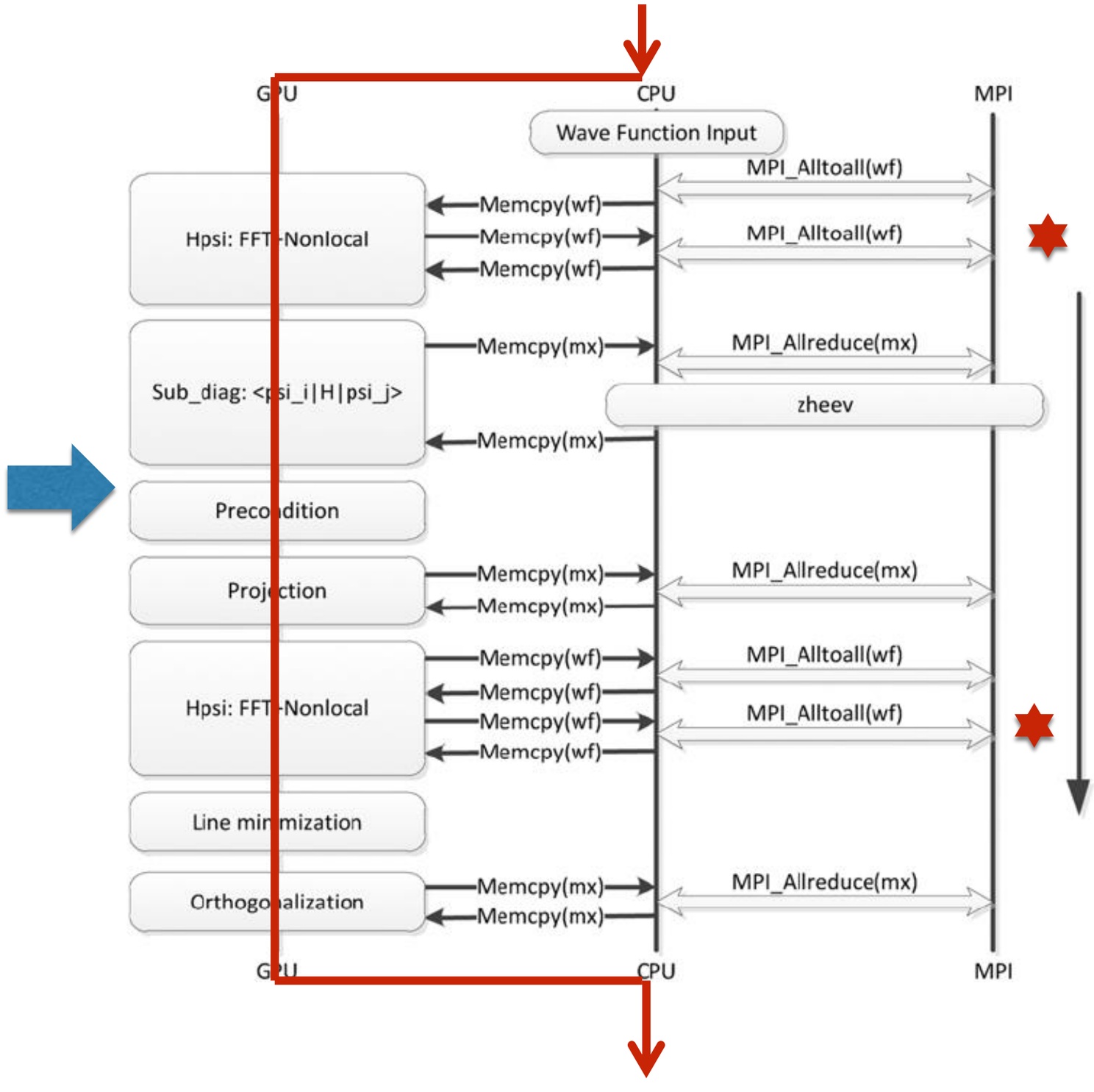
$$P_i = P_i - \sum_{j=1,i} \langle P_i | \psi_j \rangle$$ Projection, *

$$\psi_i = \psi_i \cos\theta_i + P_i \sin\theta_i$$ Line minimiz.

$$\psi_i = \psi_i - \sum_{j<i} \langle \psi_i | \psi_j \rangle$$ Orth., *

nline

$$h(i,j) = \langle \psi_i | H | \psi_j \rangle$$ Sub_diag, *



Data compression- reduce MPI Alltoall

nline

$h(i,j) = \langle \psi_i | H | \psi_j \rangle$ Sub_diag, *

$P_i = H\psi_i - \varepsilon_i \psi_i$ Hpsi, *

$P_i = A(P_i - \frac{\lambda_i}{\lambda_i^o} P_i^o)$ Precond. CG step

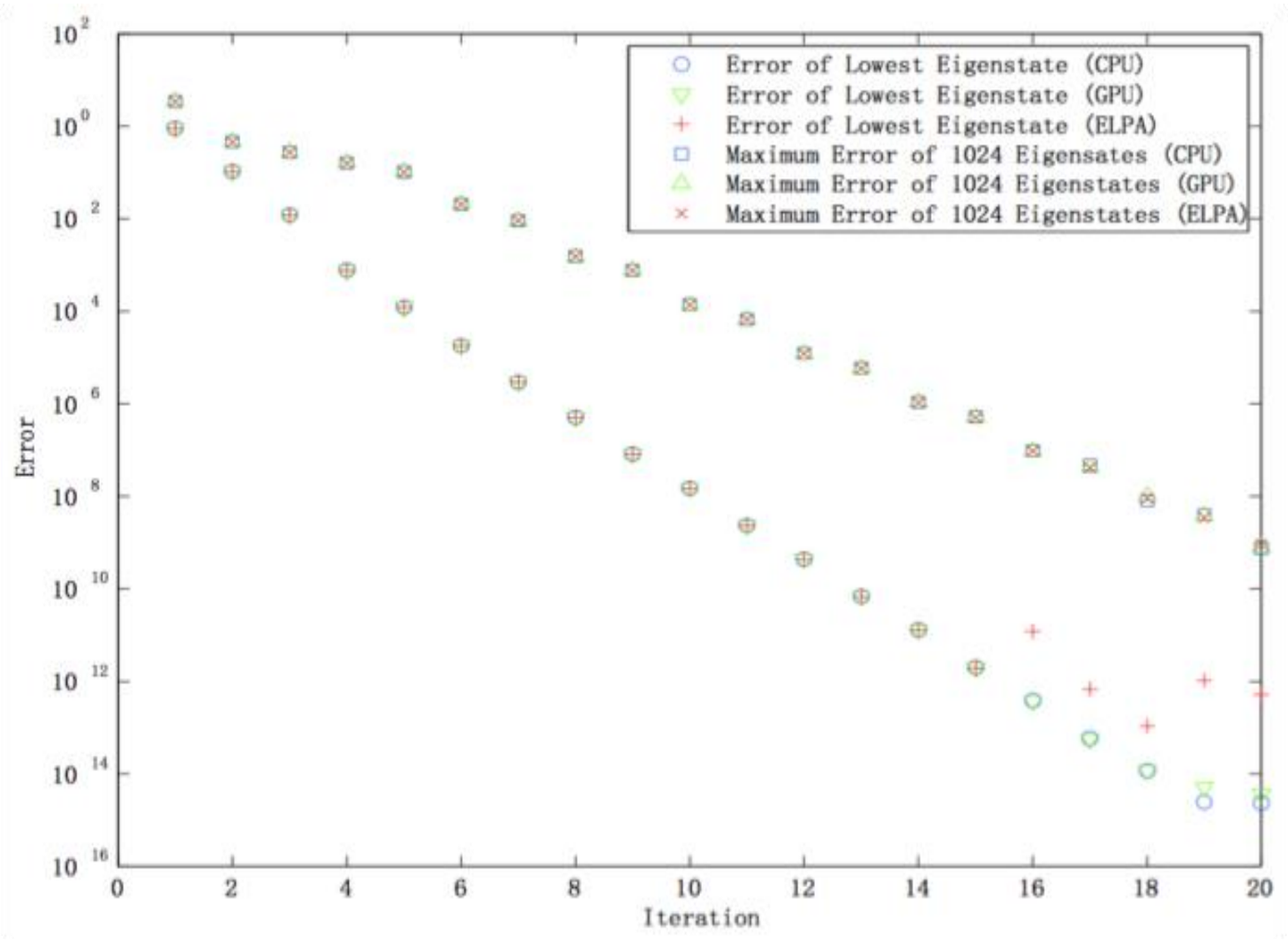
$P_i = P_i - \sum_{j=1,i} \langle P_i | \psi_j \rangle$ Projection, *

$\psi_i = \psi_i \cos\theta_i + P_i \sin\theta_i$ Line minimiz.

$\psi_i = \psi_i - \sum_{j<i} \langle \psi_i | \psi_j \rangle$ Orth., *

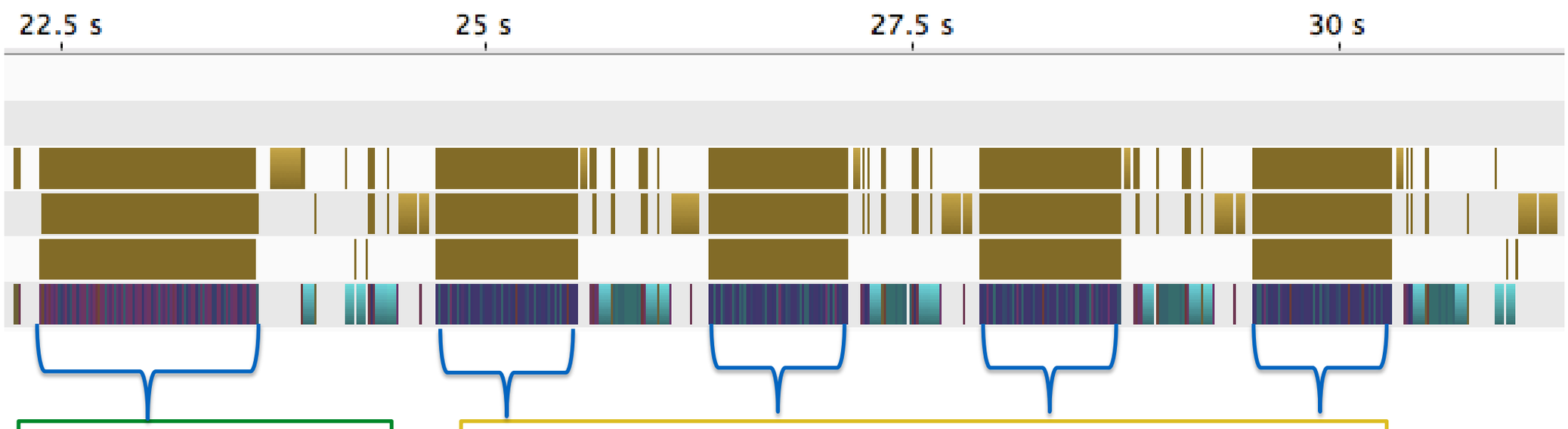
$h(i,j) = \langle \psi_i | H | \psi_j \rangle$ Sub_diag, *

Mix-precision for the wa ve function residual Pi



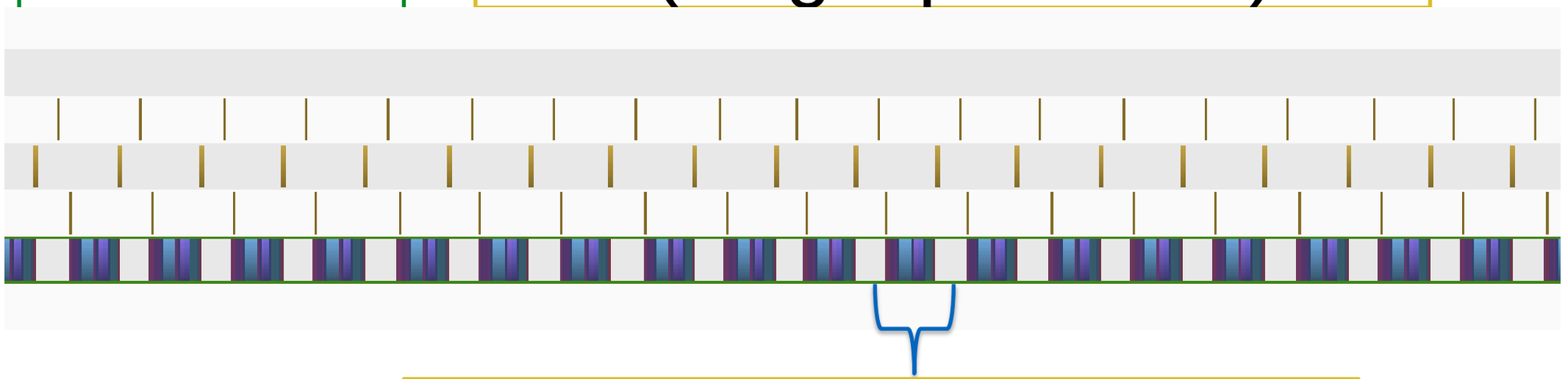
Convergence of the PEtot code after utili zing the Pi mix-precision calculation

CG AllBand profiling



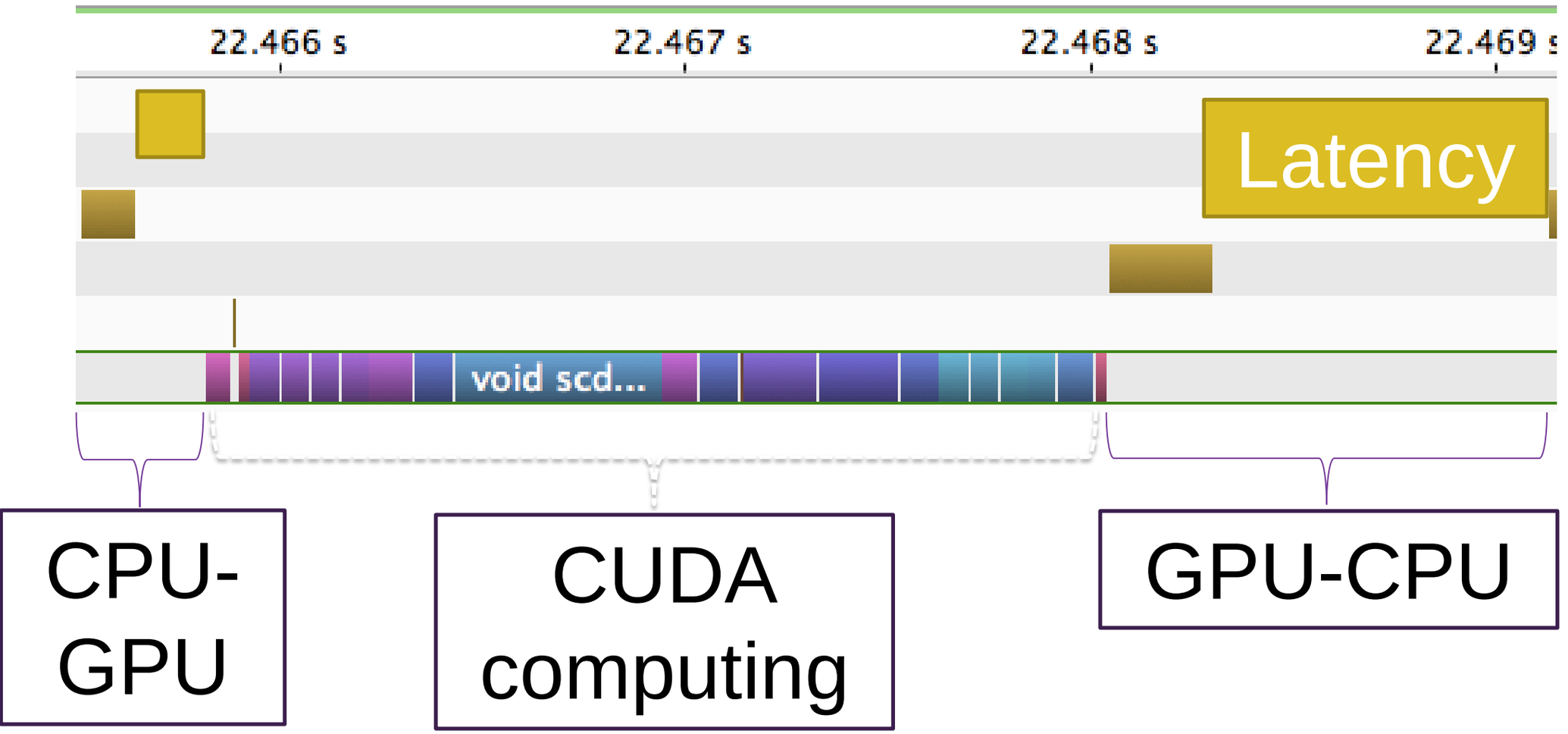
Hpsi calculatio

Hpsi calculation (single precision)



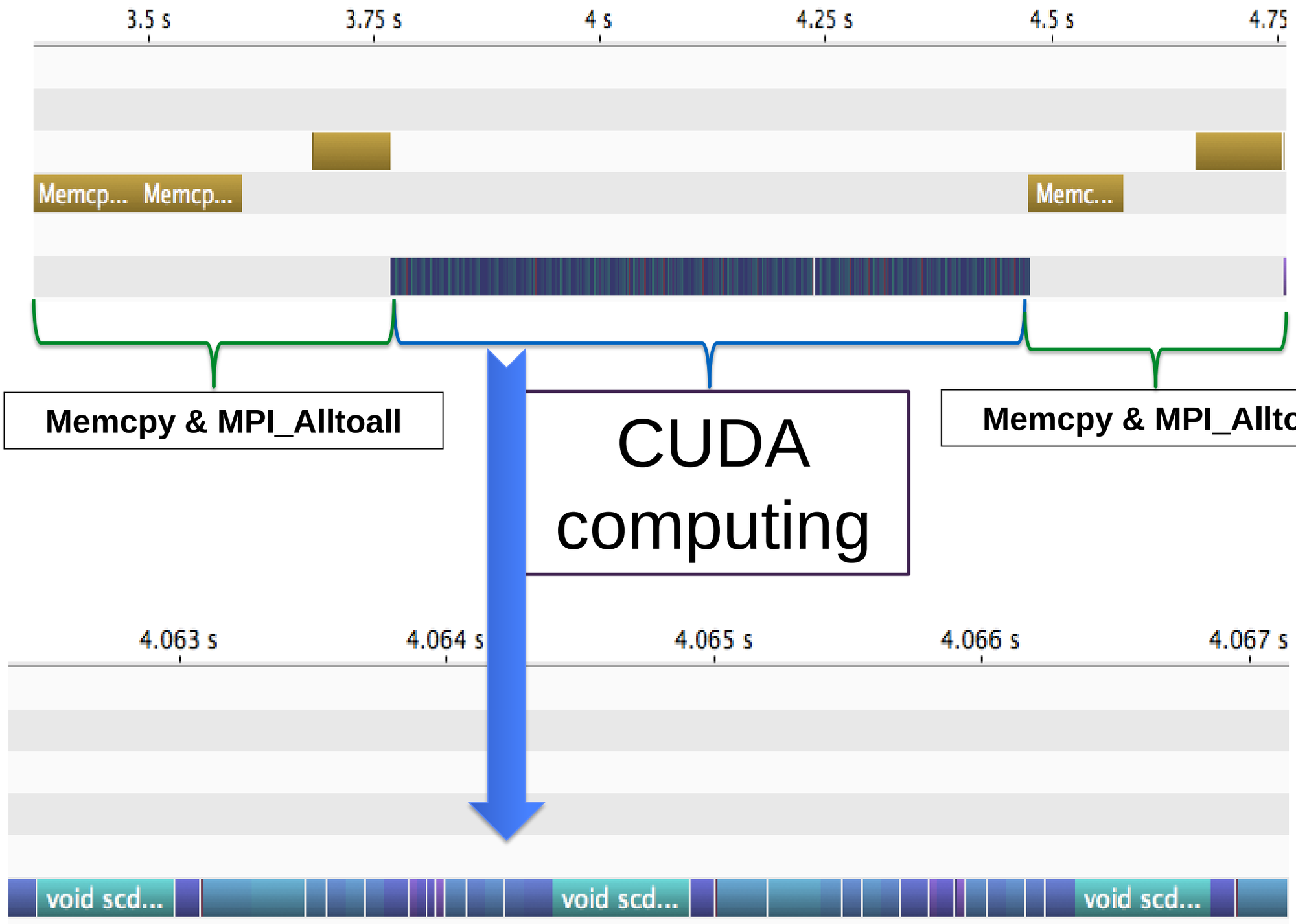
Hpsi calculation for one wave function.

Hψ calculation for a single wave function.

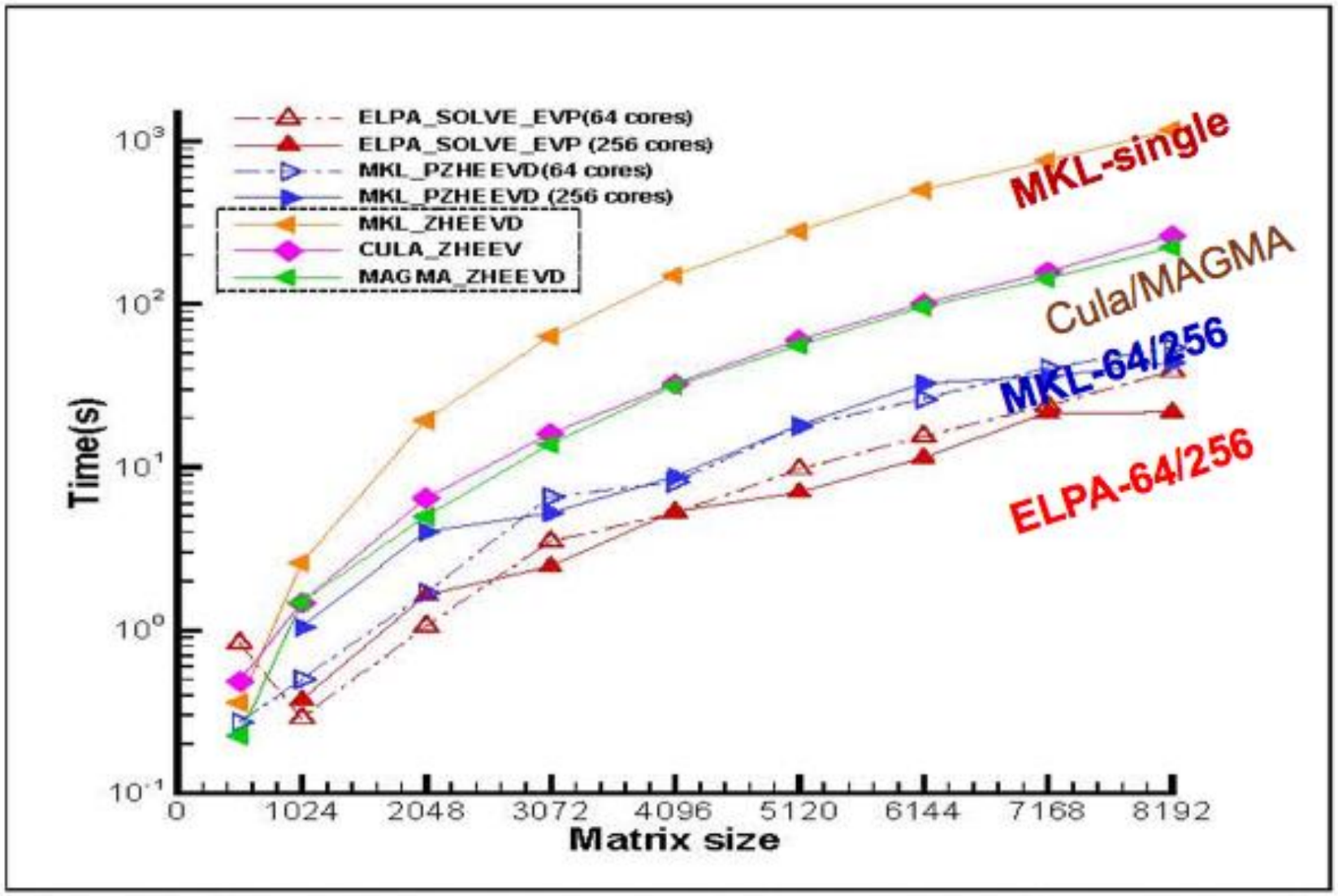


Percentage of one Hpsi calculation			
CPU-GPU	Computing	GPU-CPU	Latency
3.5%	62%	6.5%	28%

Convert latency problem to bandwidth problem.



GPU Library

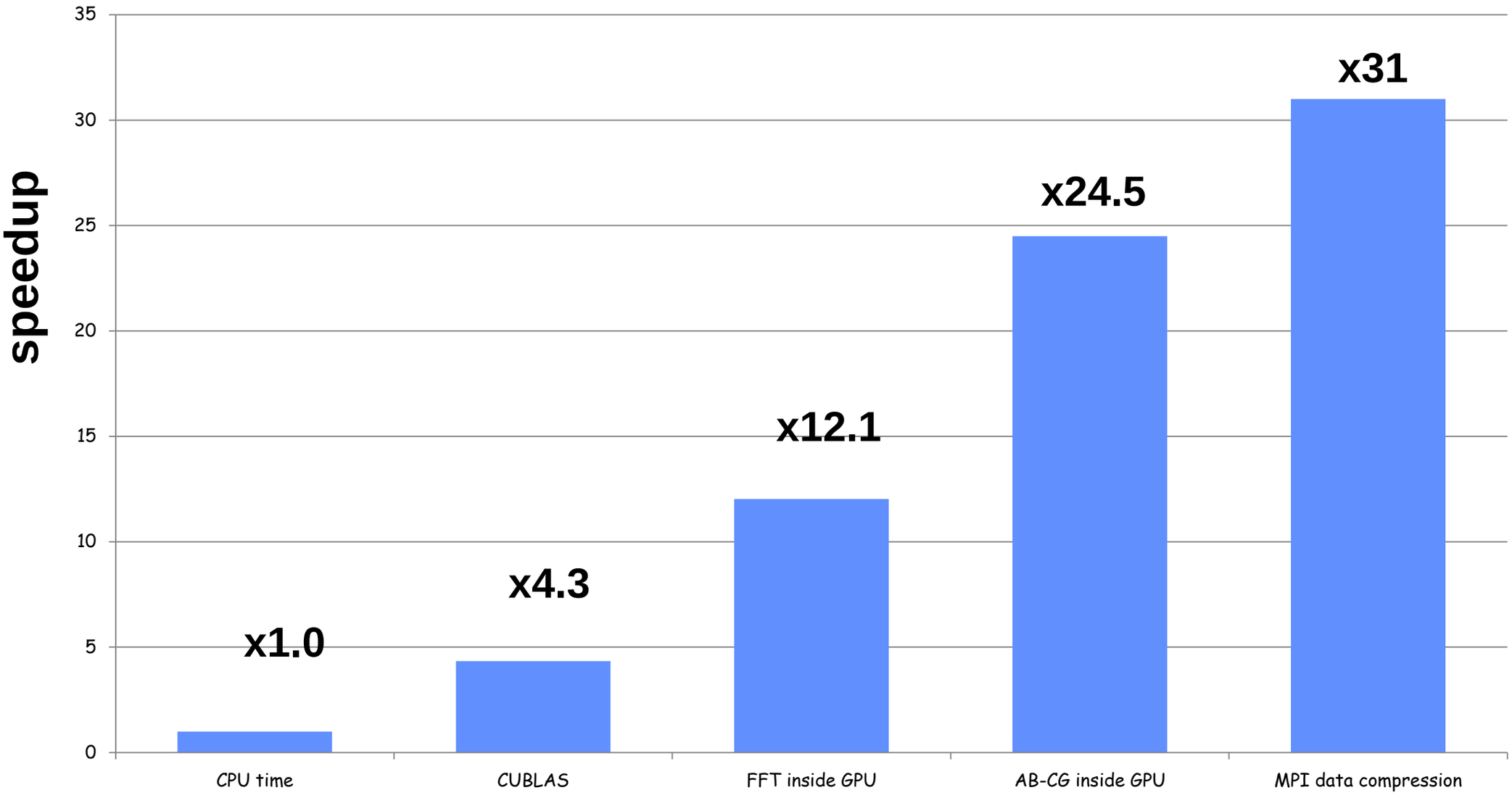


ELPA: A consortium Lead by Fritz-Haber-Inst. Max-Planck-Inst

CULA: Single GPU

MAGMA: Single GPU

Speedup of the CG_AllBand algorithm

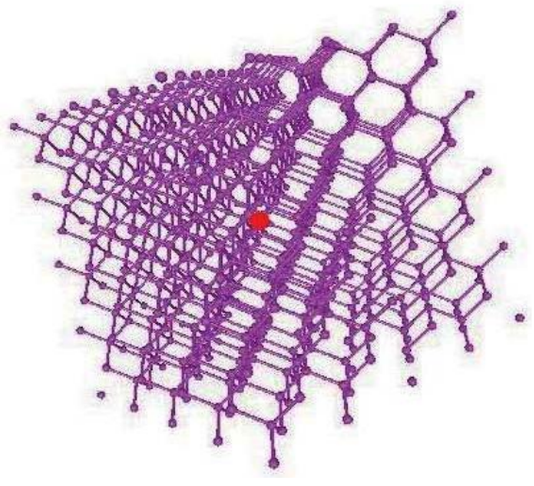


The speedup of GPU CG_AllBand over CPU PEtot code on Titan.

Testing results

PEtot testing result – one fragment

512 atom GaAs system Gamma Point				
GPU No.	32	64	128	256
PEtot GPU	31.6	20.8	13.2	**11.4**
CPU No.	32x16	64x16	128x16	256x16
PEtot CPU	277	223	203	216

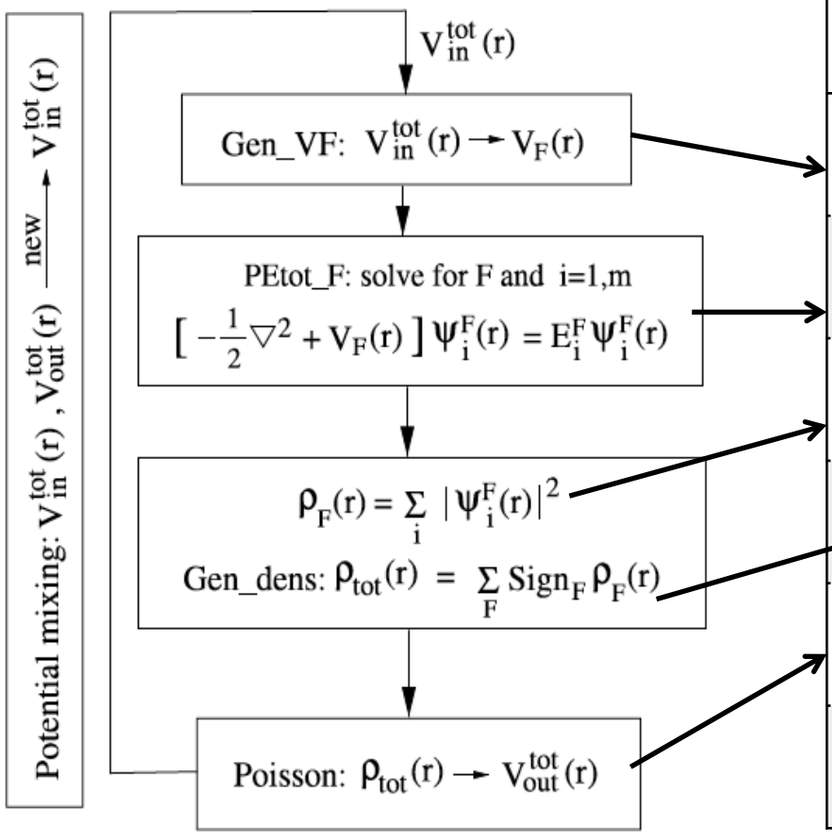


First phase of Titan:

GPU: Tesla C2090

CPU: AMD16core

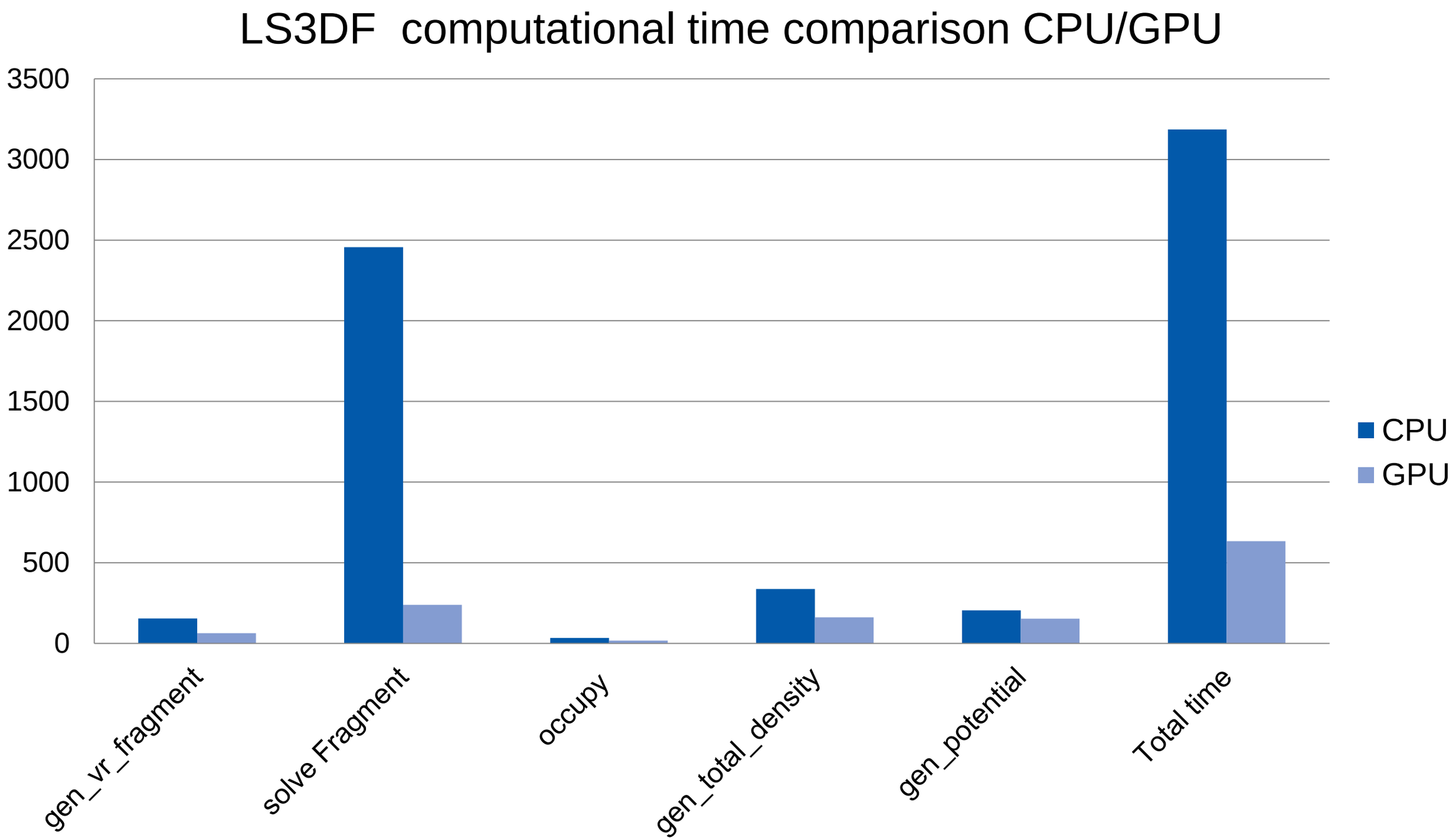
LS3DF Testing result



	CPU	GPU
gen_vr_fragment	154	63
solve Fragment	2455	239
occupy	34	17
gen_total_density	337	161
gen_potential	205	153
Total time	3186	634

Testing system is a 3877 atom system. LS3DF runs 18 step of S CF.

LS3DF Testing result



Conclusion

- We implemented a GPU LS3DF on hybrid CPU/GPU super computer Titan, currently it has 4x speedup compared with CPU code.

- For the single fragment calculation, Petot code, we have 10x-20x times speedup.

- Our results show that the data locality and local MPI commu nication makes divide-and-conquer algorithms ideal in utilizi ng the heterogeneous architecture computing power.

Future work

- Optimize charge gathering/patching from fragment charge to global charge (MPI/CPU)

- We estimate that 10x times of overall speedup could be achieved by using GPU

Acknowledgement

- INCITE program
- China Scholarship Council
- US Department Of Energy, BES, Office of Science

Thanks!