Multilevel parallelization in BigDFT

Huan Tran

Department of Physics, University of Basel

October 17, 2011

Density functional theory for many-body problems

- Many-body problem $\longrightarrow$ one-body problem
- Electron density $\rho(\vec{r})$ as the degree of freedom
- Universal functional $E[\rho]$, to be minimized with respect to ρ.
- Obtained $\rho(\vec{r})$ corresponds to the ground state of the system
- Ground state properties determined from the ground state electron density $\rho(\vec{r})$

Kohn-Sham equation

- Electron density ρ of a closed-shell gapped system of N electrons in terms of Kohn-Sham orbitals $\phi_i(\vec{r})$

$$\rho(\vec{r}) = 2\sum_{i=2}^{N/2} \phi_i^*(\vec{r})\phi_i(\vec{r}) \tag{1}$$

- Kohn-Sham Hamiltonian

$$\mathcal{H}^{\mathrm{KS}} = -\frac{1}{2}\nabla^2 + V_{\mathrm{H}}[\rho] + V_{\mathrm{xc}}[\rho] + V_{\mathrm{en}} \tag{2}$$

- Kohn-Sham equation by minimizing $E[\rho]$

$$\mathcal{H}^{\mathrm{KS}}\phi_i(\vec{r}) - \epsilon_i\phi_i(\vec{r}) = 0; \qquad \int \phi_j^*(\vec{r})\phi_i(\vec{r}) = \delta_{ij} \tag{3}$$

Solving Kohn-Sham equation

- Residue $g_i(\vec{r})$, needs to be small enough

$$g_i(\vec{r}) = \mathcal{H}^{\mathrm{KS}}\phi_i(\vec{r}) - \sum_{ij} \Lambda_{ij}\phi_j(\vec{r}) \tag{4}$$

- Lagrange multipliers

$$\Lambda_{ij} = \int d\vec{r}\phi_j^*(\vec{r})\mathcal{H}^{\mathrm{KS}}\phi_i(\vec{r}) \tag{5}$$

- Kohn-Sham equation 3 solved by direct minimization guided by preconditioning gradient $\tilde{g}_i(\vec{r})$ determined from $g_i(\vec{r})$ via

$$\left(\frac{1}{2}\nabla^2 - \epsilon_i\right)\tilde{g}_i(\vec{r}) = g_i(\vec{r}) \tag{6}$$

Numerical solving the DFT problem

1. Algorithm
   - Initiate charge density $\rho_{\mathrm{in}}(\vec{r})$
   - Calculate the Hamiltonian from $\rho_{\mathrm{in}}(\vec{r})$
   - Guess the initial orbitals $\phi_i(\vec{r})$
   - Improve the orbitals $\phi_i(\vec{r})$
   - Calculate the residue of $g_i(\vec{r})$
   - If δ is not small, go back to 1. If δ is small, jump to 1
   - Calculate ρ_{out} by Eq. (1). If ρ_{out} convergent, jump to 1; if not, go back to 1
   - From ρ_{out}, calculate $\rho_{\mathrm{in}}^{\mathrm{new}}$ and go back to 1
   - STOP
2. Most of the operations are performed on a given orbital $\rightarrow$ orbitals are distributed over MPI processes

Orbital distribution scheme

1. Orbitals are distributed over MPI processes
2. Each MPI process holds all the coefficients of all the orbitals on the basis set
3. Kohn-Sham equation is solved within a given MPI process
4. Lagrange multiplier Λ_{ij} and orthogonalization need both $\phi_i(\vec{r})$ and $\phi_j(\vec{r})$
   - Each MPI process calculated its contribution from the coefficients
   - A global reduction sum is used to collect all the contributions

MPI parallelization: performance

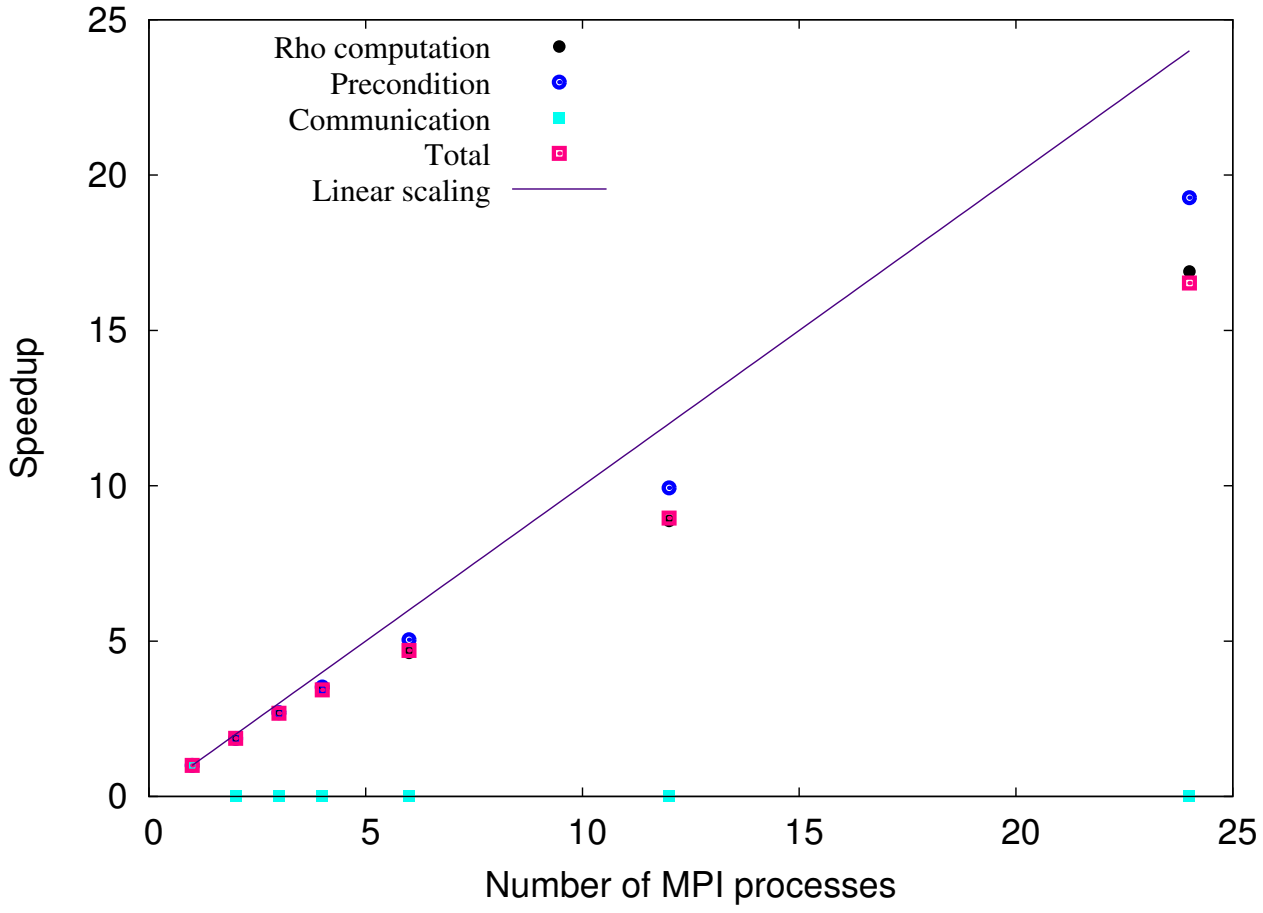
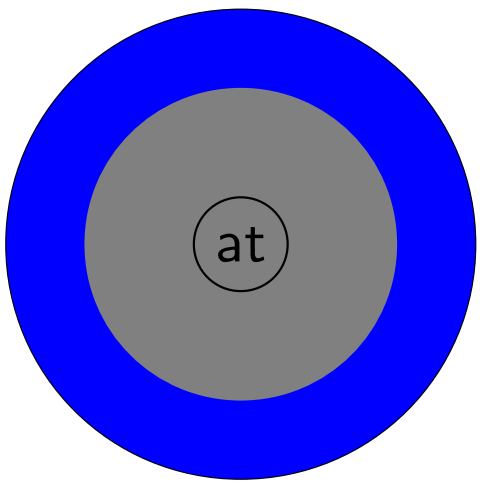


Figure: MPI parallelization performance of BigDFT on a Boron 80 cluster, Cray XE6 machine.

MPI data communication: less for more

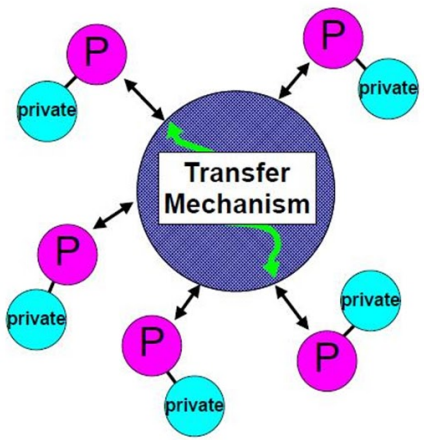
1. Global sum on a large volume of charge density (`MPI_ALLREDUCE`)
2. Less volume of data for more speed

- Double precision ρ for gray region, single precision ρ for blue region, zero otherwise
- Size of the regions are adjustable to get desired accuracy
- For a reasonable accuracy, data volume reduced a factor of 5, communication time a factor of 3



Distributed/shared memory

1. Programming models on distributed memory: Sockets, PVM(Parallel Virtual Machine), MPI(Message Passing Interface)
2. Programming models on shared memory: OpenMP, posix thread, Automatic parallelization



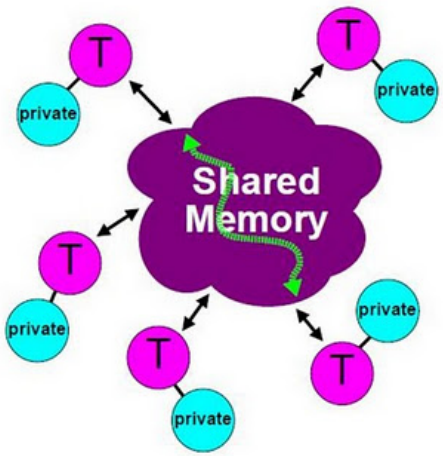
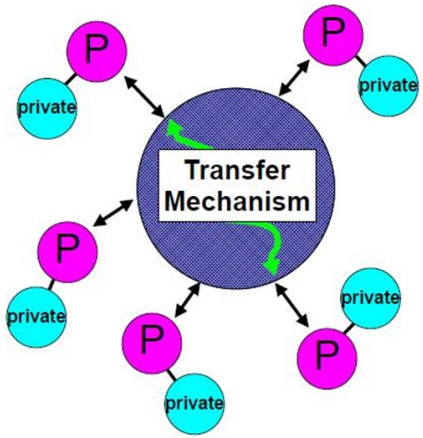
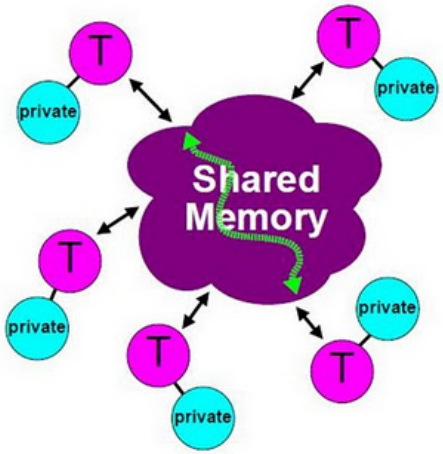


Figure: Stolen from http://openmp.blogspot.com/

Distributed vs shared memory



- MPI is an example
- Private memory
- Private data, shared by exchanging buffers
- Data transfer programmed explicitly



- OpenMP is an example
- Shared or private data
- Globally shared memory for all threads $\rightarrow$ data transfer
- Private data accessible only to its owner

OpenMP in BigDFT: introduction

1. Solving KS equation is performed within given MPI process
2. Most time-consuming operations easily parallelized by OpenMP
3. An example of OpenMP parallelization

```
!$omp parallel default(private) &
!$omp shared(n1,n2,n3,nfl1,nfu1,nfl2)
!$omp do schedule(static,1)
do i3=0,n3
      ...
enddo
!$omp end parallel
```

OpenMP in BigDFT: performance

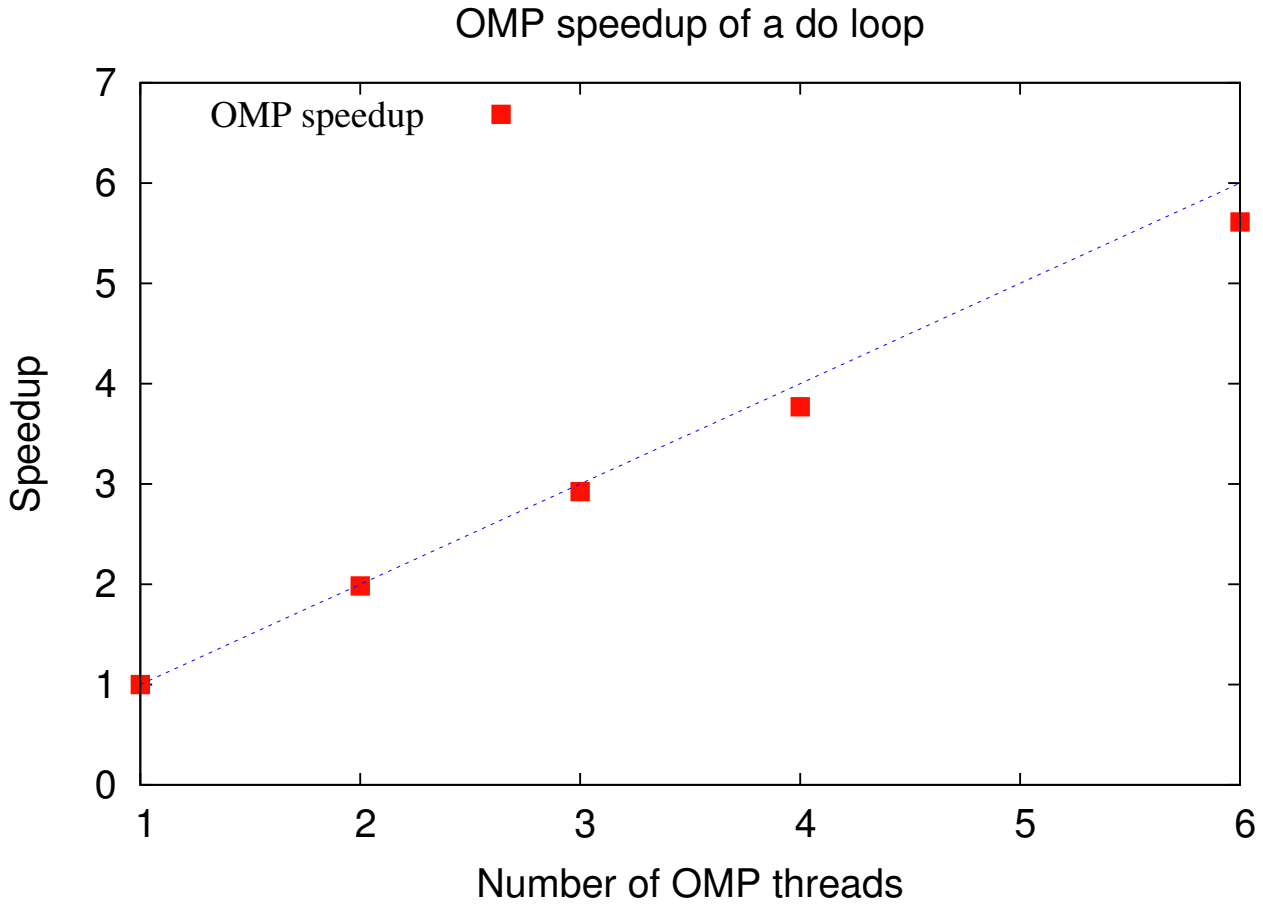


Figure: OpenMP parallelization performance on a single do loop in BigDFT, Cray XE6 machine.

Combined MPI and OpenMP in BigDFT: performance

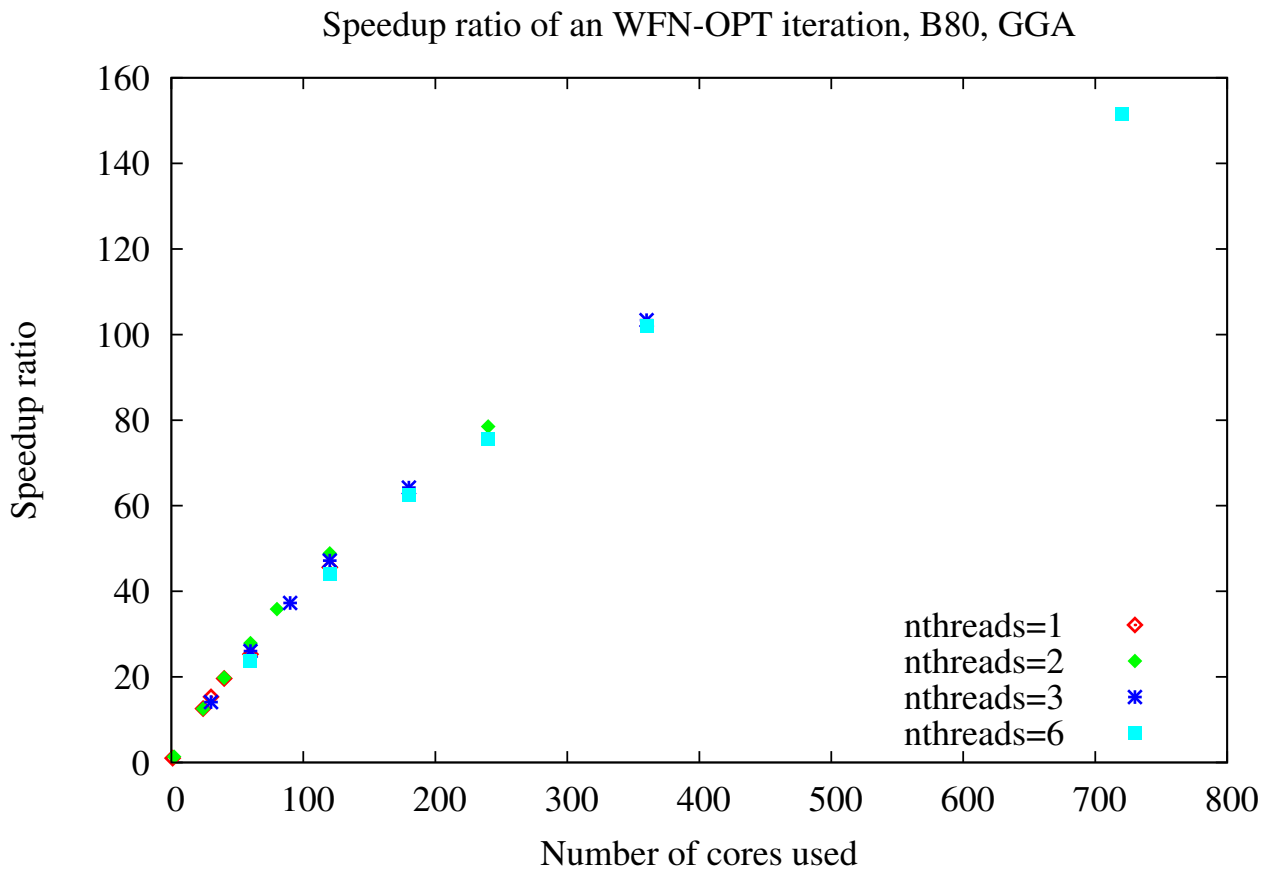


Figure: Combined MPI-OpenMP parallelization of BiGDFT on a B80 cluster, Cray XE6 machine.

Discussions

1. Efficiencies of MPI and OMP parallelization in BigDFT are comparable
2. MPI communication is a bottleneck and can be partly solved by reducing the data volume communicated