Efficient ab-initio geometry optimization guided by force field

T. Deutsch, S. Goedecker

DRFMC, CEA Grenoble

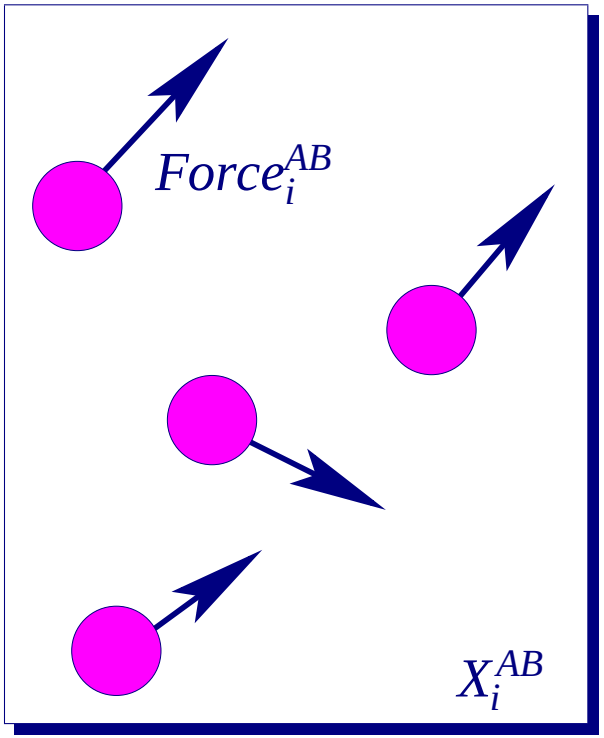
tdeutsch@cea.fr

Goal

- Ab-initio geometry optimization: 100–200 steps
- Reduce the number of ab-initio calculation using a force field:
  - force field iteration very low cost compared to an ab initio iteration

Algorithm

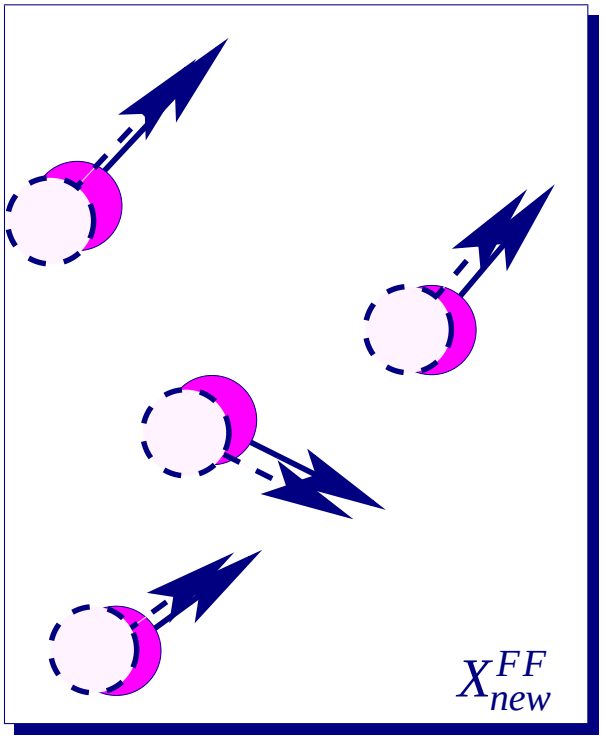
1. **Ab initio (AB) positions** X_i^{AB} **and** $Force_i^{AB}$



Algorithm

1. **Ab initio (AB) positions** X_i^{AB} **and** $Force_i^{AB}$

2. **Force Field (FF)**
   Looking for X_{new}^{FF} **so that**
   $Force^{FF}\left(X_{new}^{FF}\right) == Force_i^{AB}$
   using SD or CG.

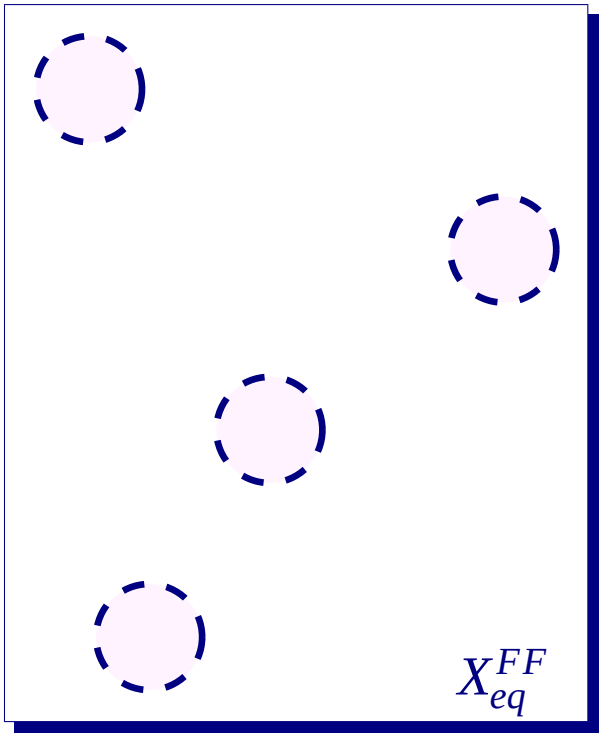


Algorithm

1. **Ab initio (AB) positions**
   X_i^{AB} **and** $Force_i^{AB}$

2. **Force Field (FF)**
   Looking for X_{new}^{FF} **so that**
   $Force^{FF}\left(X_{new}^{FF}\right) == Force_i^{AB}$
   using SD or CG.

3. **Force Field (FF)**
   Looking for X_{eq}^{FF} **with SD of CG**



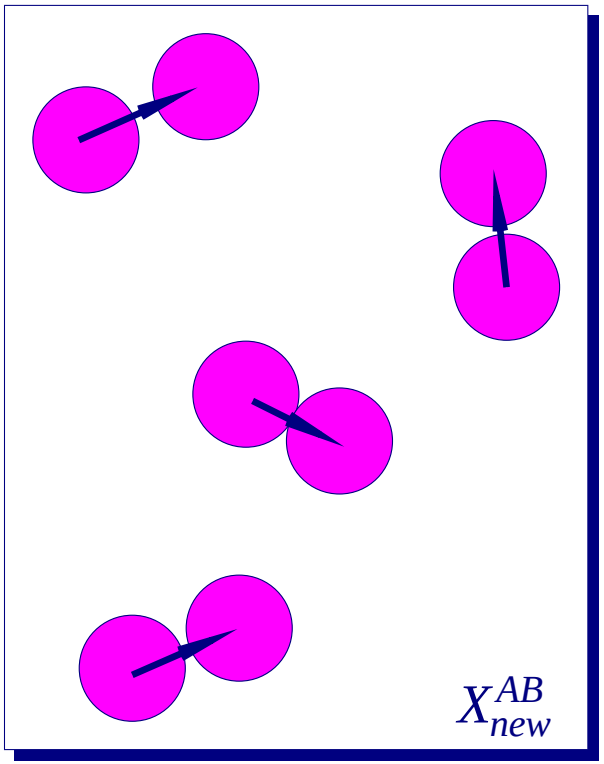
Algorithm

1. **Ab initio (AB) positions**
   X_i^{AB} **and** $Force_i^{AB}$

2. **Force Field (FF)**
   Looking for X_{new}^{FF} **so that**
   $Force^{FF}\left(X_{new}^{FF}\right) == Force_i^{AB}$
   using SD or CG.

3. **Force Field (FF)**
   Looking for X_{eq}^{FF} **with SD of CG**

4. **New position (AB) using SD or DIIS**
   $\delta X^{AB} = X_{eq}^{FF} - X_{new}^{FF}$
   $X_{i+1}^{AB} = X_i^{AB} + \alpha \delta X^{AB}$



Advantages

- In contrast to BFGS, valid also outside of a quadratic region
- If Force Field identical to Ab initio: only 1 step
- Applicable also for the search of saddle points

Advantages

- In contrast to BFGS, valid also outside of a quadratic region
- If Force Field identical to Ab initio: only 1 step
- Applicable also for the search of saddle points

Drawbacks

- Need an accurate Force Field

(Preliminary) Results

Silicon HGH pp

ecut 7.5 Hartree

tolmxf 2.57e-6 Hartree/bohr (1.e-4 eV/angström)

Type	bfgs	lenosky	bazant
Cluster(5)	18	20	45
Si-64	31	11	13
Si-63	123	15	16
Surface			143

Comparison in parallel

ABINIT – CPMD

Methods

ABINIT or CPMD (diagonalisation scheme):

$$\frac{\partial}{\partial \Psi}\left(\frac{1}{2}\Psi^{T} H[\rho]\Psi\right) = H[\rho]\Psi$$

Si 64 atoms: **ABINIT 370 s, CPMD 415 s**

Methods

ABINIT or CPMD (diagonalisation scheme):

$$\frac{\partial}{\partial \Psi}\left(\frac{1}{2}\Psi^T H[\rho]\Psi\right) = H[\rho]\Psi$$

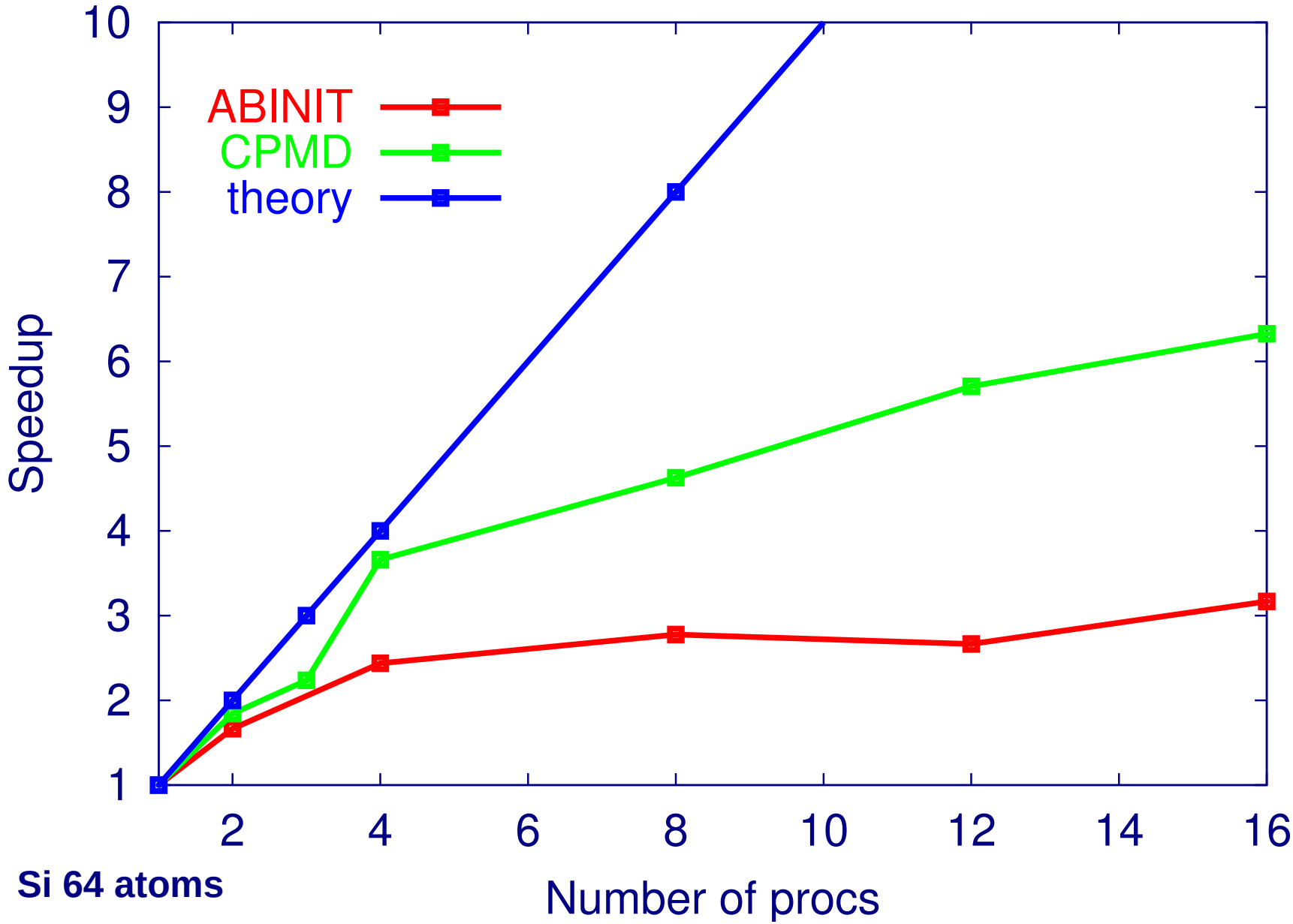
Si 64 atoms: **ABINIT 370 s, CPMD 415 s**

Variational principle (CPMD):

$$\frac{\partial}{\partial \Psi} E_{tot}^{DFT}(\Psi_1, \Psi_2, \ldots, \Psi_N) = H[\rho]\Psi_i$$

Need a gap **CPMD 280s**

Speedup



Si 64 atoms

A 3-dim FFT

Stefan Goedecker

DRFMC, CEA Grenoble

SGoedecker@cea.fr

Basic problems

- **Low ratio between floating point operations and data (load/store's)**
  3-dim FFT:
  - N^3 **data points**
  - $15N^3 \log_2(N)$ **floating point operations**
- **Large data sets that do not fit into cache**
- **Highly nonlocal data access pattern**

Serial optimization discussed in:

S. Goedecker, A. Hoisie: "Performance Optimization of Numerically Intensive Codes", SIAM, 2001 (ISBN 0-89871-484-2)

Rotation technique for 3-dim FFT

Convention:

i1, i2, i3 untransformed dimensions
I1, I2, I3 transformed dimensions
i3 = (j3, jp3)
I3 = (J3, Jp3)

Input: i1, i2, j3, jp3
where i1 = 1, n1
i2 = 1,n2
j3 = 1, n3/nproc
jp3 = 1, nproc

Rotation technique for 3-dim FFT

multiple 1-dim FFT: i1, I2, j3, jp3

Rotation and removal: I2, i1, j3, jp3

multiple 1-dim FFT: I2, I1, j3, jp3

Rotation and removal: I1, I2, j3, jp3

Previous data set reformatted: I1, J2, Jp2, j3, jp3

Copy: I1, J2, j3, Jp2, jp3

MPI_ALLTOALL: I1, J2, j3, jp3, Jp2

Previous data set reformatted: I1, J2, i3, Jp2

FFT: I1, J2, I3, Jp2

Copy: I1, I3, J2, Jp2

Convention:

i1, i2, i3 untransf. dim.

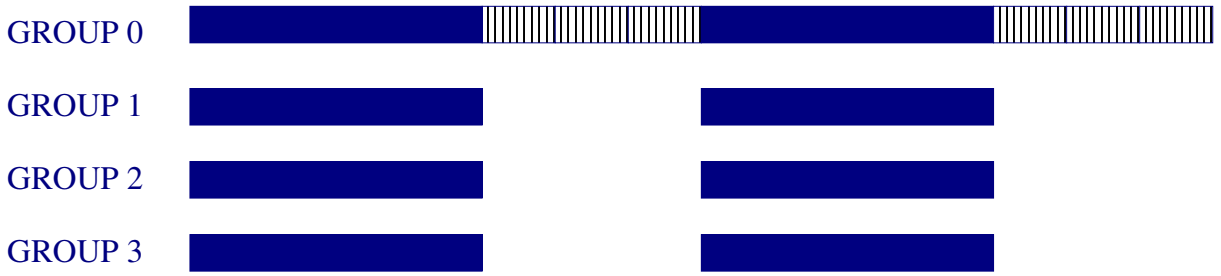
I1, I2, I3 transf. dim.

i3 = (j3, jp3)

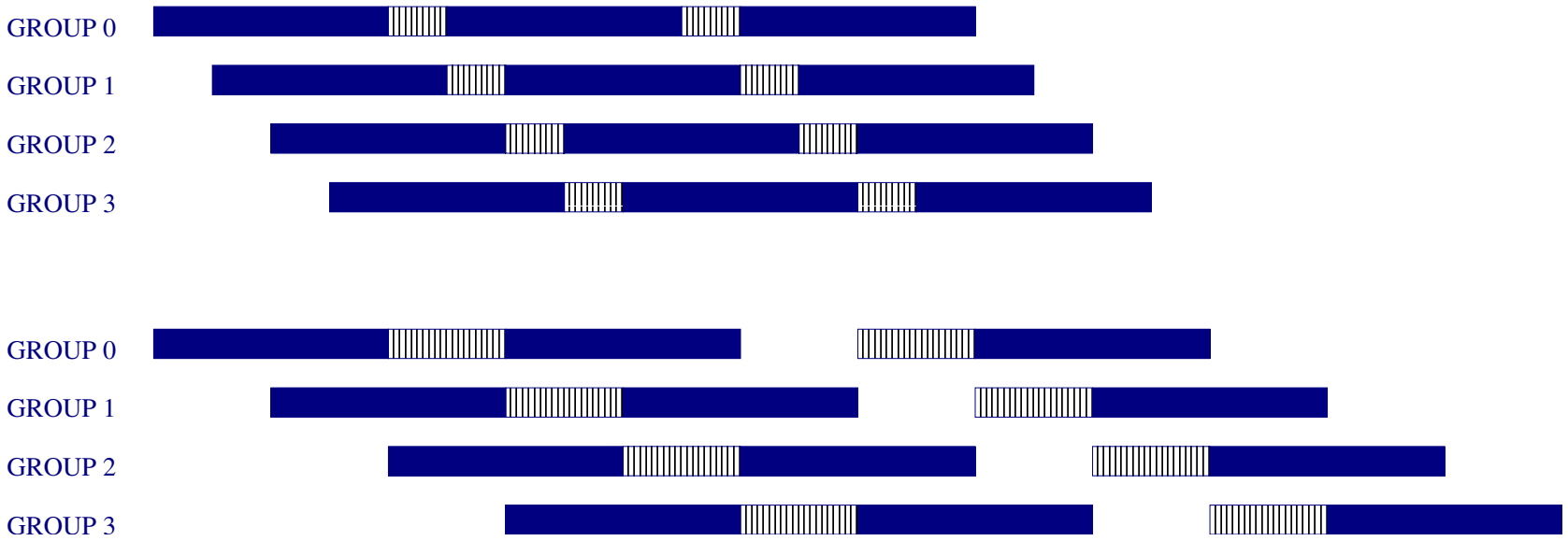
I3 = (J3, Jp3)

Communication traffic patterns

In a conventional MPI/OpenMP implementation



In this MPI/OpenMP implementation



Results obtained so far

OpenMP/MPI approach gives 2 times higher speed than pure MPI approach

60 Gflops for a 128^3 FFT on 64 nodes (256 processors)
Speedup of more than 100 compared to serial FFT

Speedup of 3.3 for the pure OpenMP version on the 4 processors of one node