Enhancing Parallelizing Capabilities of ABINIT

3rd International ABINIT Developer Workshop 2007

Philipp Plänitz, Markus Franke and Nico Mittenzwey

Chemnitz University of Technology
Department of Opto- and Solid State Electronic
Prof. Dr. Ch. Radehaus

philipp@plaenitz.info

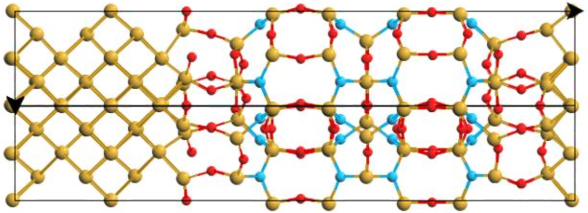
January 29, 2007

CHEMNITZ UNIVERSITY OF TECHNOLOGY

Outline

Motivation

- strong connection to semiconductor industries
- interest in physical properties of amorphous materials



Outline

Current Implementation

- Sequential process over atoms + calculation of each movement
- Calculation of total energy is done in parallel (K-Point/Band-by-Band Parallelizing)
- But: each perturbation is almost independent from the other ones
- Goal: parallelizing over Perturbations
- Establishment of a new level of parallelizing on top of K-Point parallelizing

Concept

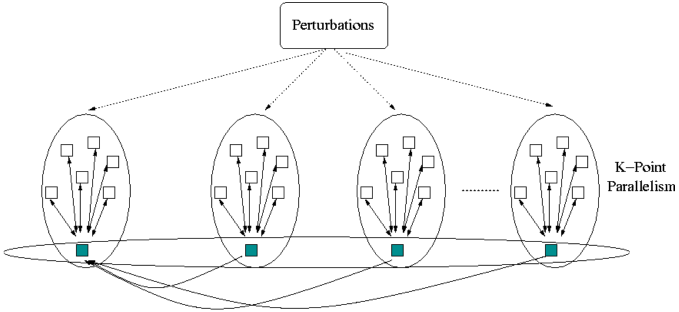


Figure 1: concept of parallelizing

Major Modifications I

- call some_mpi_function(...,mpi_enreg%me,...)

OBTAIN DYNAMICALLY MY RANK WITH RESPECT TO COMMUNICATOR

```
call xme_init(mpi_enreg,me)
call some_mpi_function(...,me,...)
```

Major Modifications II

- call some_mpi_function(. . . ,MPI_COMM_WORLD,. . .)

Replacement of global communicators

call xcomm_init(mpi_enreg,spaceComm)
call some_mpi_function(. . . ,spaceComm,. . .)

Major Modifications III

- modification of xcomm_world, xme_init, xcomm_init, xproc_init in <ABINIT_ROOT>/src/Src_1managempi/xdef_comm.F90

  return different communicators, number of processors and ranks depending on mpi_enreg%paral_compil_respfn

Major Modifications IV

- adding new member variables in datatype "MPI_type"

modifications in MPI_type

```
  ! parallelizing over perturbations activated?
  integer :: paral_compil_respfn
  ! number of my processor in my group of perturbations
  integer :: me_respfn
  ! number of processors in my group of perturbations
  integer :: nproc_respfn
  ! my group for calculating perturbations
  integer :: my_respfn_group

...
```

Major Modifications V

- introduced 2 input variables (VARPAR)
  - paral_rf, activate parallelizing over perturbations
  - ngroup_rf, number of parallelizing groups
- possibility to switch on/off paralrf for each dataset separately
- adaption of <ABINIT_ROOT>/src/Src_1managempi/distrb2.F90

  generate different mpi_enreg%proc_distrb with respect to parallelizing over perturbations
- initialisation of respfn-groups in <ABINIT_ROOT>/src/01managempi/initmpi_respfn.F90

Major Modifications VI

- masters of each respfn-group form another group

Group of masters

used in <ABINIT_ROOT>/src/08seqpar/respfn.F90 for gathering some arrays after calculation of first-order wavefunctions

```
call xsum_master(blkflg,0,mpi_enreg%respfn_master_comm,ierr)
call xsum_master_dp5d(d2lo,0,mpi_enreg%respfn_master_comm,ierr)
call xsum_master_dp5d(d2nl,0,mpi_enreg%respfn_master_comm,ierr)
call xsum_master_dp2d(vtrial,0,mpi_enreg%respfn_master_comm,ierr)
```

Major Modifications VII

- parallelizing of the big loop over perturbations in <ABINIT_ROOT>/src/08seqpar/loper3.F90

parallelized loop in loper3.F90

```
do icase=1,ipert_cnt
  ! calculate only private part of perturbations
  ! when current perturbation is not part of my
  ! group then skip this loop
#if defined MPI
  if(dtset%paral_rf==1) then
     if(mpi_enreg%respfn_group(modulo(icase,ngroup_respfn)+1) /=
        my_group) then
          cycle
     endif
  endif
#endif
```

Measurements

Test-Job

natom	9 / $\alpha - SiO_2$
nband	36
ngkpt	3 3 3
nsym	1 / chkprim 0
ecut	30.0

Measurements II

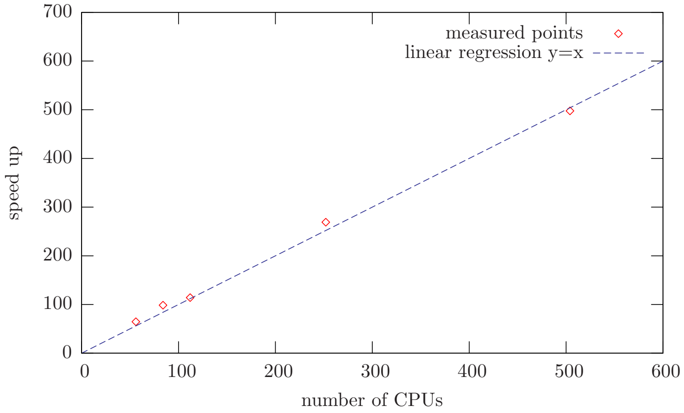


Figure 2: Measurement of speed up

State of the Art

- sucessfully tested on different clusters (including IBM BlueGene)
- ETOT's calculated on each processor are written in its LOG-File rather than in a compound OUT-File
- doesn't work together with "parareel"-parallelizing
- still some problems exist when activating band-by-band-parallelizing
- works for most combinations of #cpus and #groups <ABINIT_ROOT>/src/08seqpar/loper3.F90
  - usage of a wrong mkmem
  - should be recalculated based on values for the reduced brillouin-zone

Outline

Overview

- located inside the "outscfcv" subroutine
- no side effects on other abinit parts
- "partial_dos_fraction" returns the partial_dos array
- outscfcv calls "tetrahedron"
- "tetrahedron" calculates LDOS with the help of "partial_dos" array

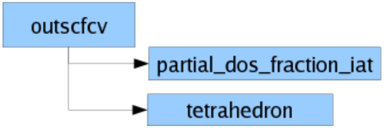


Figure 3: Current implementation scheme

Problems / Goals

- memory usage is directly depending on number of atoms
- large systems can exceed available memory
- parallelizing and scalability on large clusters can be improved

BENCHMARKS ORIGINAL CODE

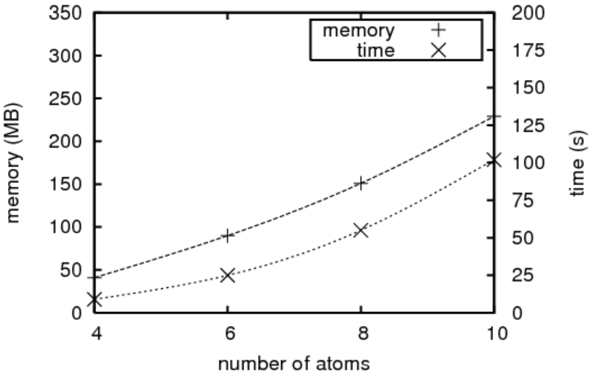


FIGURE 4: time and memory usage depending on number of atoms

Changes first step I

To reduce the memory usage we rearanged the code so that the LDOS of every atom gets calculated seperatly after one after another. This made the memory usage nearly (beside of the increase of the number of bands) independend on the number of atoms.

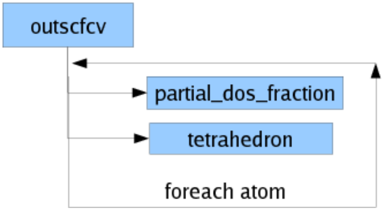
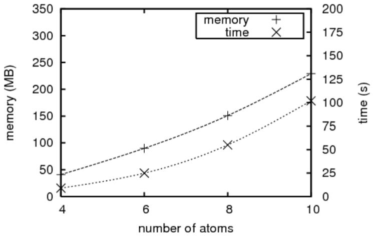


Figure 5: Overview

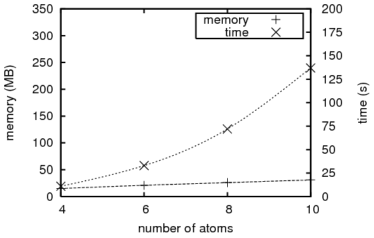
Changes first step II

- outscfcv
  - new loop over atoms around "‘partial_dos_fraction"’ and "‘tetrehedron"’
  - allocation of dos_fractions array withn respect to mbesslang (was ndosfraction)
- all other LDOS subroutines
  - array sizes where reduced to fit only one atom

Benchmarks first step I



(a) original version



(b) modified version

Figure 6: time and memory usage depending on number of atoms

Changes in second step I

To solve the time problem we searched for calculations which only needs to be calculated once for all atoms and placed them in front of the loop over all atoms in the outscfcv subroutine.

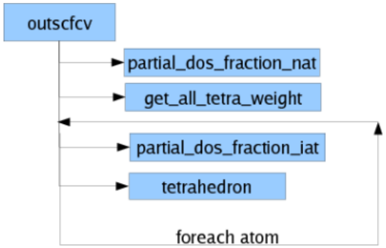
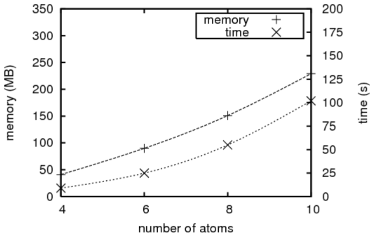


Figure 7: Overview

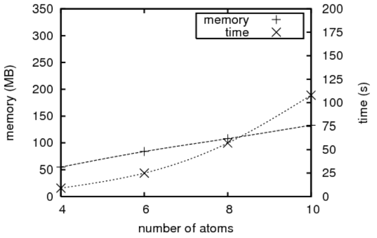
Changes in second step II

- splitting of function "partial_dos_fractions" in two new functions
- new function "get_all_tetra_weight" - calculates the arrays "all_dtweight" and "all_tweight" under use of old function "get_tetra_weight"

Benchmarks second step I



(a) original version



(b) modified version

Figure 8: time and memory usage depending on number of atoms

Next Steps

- parallelizing over atoms
- extensive debugging and testing on different architectures and test cases
- develop test inputs and documentation for next abinit version

Many thanks - for your attention !!!