Improving the parallel scaling of ABINIT

Torsten Hoefler
Technical University of Chemnitz, Dept. of Computer Science, Chair of Computer Architecture, htor@informatik.tu-chemnitz.de
Rebecca Janisch
Technical University of Chemnitz, Dept. of Electrical and Information Technology, Chair of Opto- and Solid State Electronics, jreb@etit.tu-chemnitz.de
Wolfgang Rehm
Technical University of Chemnitz, Dept. of Computer Science, Chair of Computer Architecture, rehm@informatik.tu-chemnitz.de

Abstract

This work presents the results of a three week stay of T.H. at the High Performance Systems Department of the CINECA Institute in Bologna, Italy. The main goals of the short project - to understand and parallelize the core functions of the application ABINIT - have been achieved. A huge share of the time consuming part of the code has been parallelized efficiently and scales well on modern supercomputers. We demonstrate parallel scaling on the IBM SP5 in CINECA and a small Opteron PC cluster.

1 Introduction

The software package ABINIT [1] is used to perform *ab initio* electronic structure calculations based on the density functional theory (DFT) of Hohenberg and Kohn [2] and Kohn and Sham [3]. The code is the object of an ongoing open software project of the Université Catholique de Louvain, Corning Incorporated, and other contributors [4]. ABINIT mostly aims at solid state research. Periodic boundary conditions are applied and the majority of the integrals that have to be calculated are represented in reciprocal space (k-space). The interactions between valence electrons and ionic cores are modelled by pseudopotentials, and the electronic wavefunctions are expanded in a set of plane waves[1].

1.1 Related Work

Different approaches exist to determine the ground state atomic and electronic structure of a crystal.
Packages like ABINIT[4], VASP[5], and CASTEP[6] perform a self-consistent electronic ground state calculation for a given arrangement of atoms. The energy of the ionic cores in the self-consistent potential of the electrons and the

[1]ABINIT for a short time (since version 4.2.x) also features the projector-augmented wave method, but this is still under developement. In the following we refer to the planewave method.

gradient thereof (the Hellmann-Feynman forces) can then be used to shift the ions towards their equilibrium positions, e.g. by a conjugate gradient scheme. As an example, Fig. 1 shows the ground state electron density of a relaxed SiO_2 structure calculated with ABINIT. The challenge in this kind of approach is to perform a very accurate determination of the electronic ground state after each move of the atoms within affordable time.

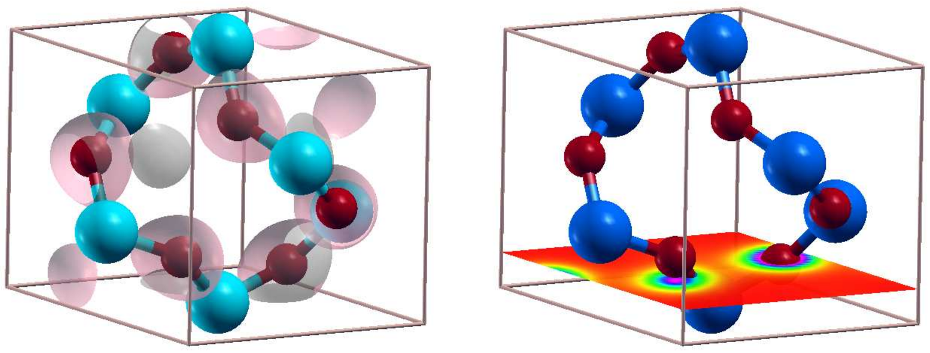

Figure 1: Electronic structure of a relaxed SiO_2 unit cell calculated with ABINIT. Left: isosurfaces of the electron density, illustrating the directional character of the bonds. Right: Planar cut through the electron density, showing that charge is accumulated at the oxygen atoms (red).

Programs like CPMD [7] and PWSCF [8, 9] are based on the *ab initio* molecular dynamics scheme of Car and Parrinello [10], i.e. the ions are treated as classical particles, coupled to the movement of the electrons by pseudo-Newtonian equations of motion. Since the positions of ions and electrons are varied at the same time, the computational effort is smaller than in the scf ground state calculation scheme. However, in this approach the system is not exacatly in its adiabatic ground state, and the challenge is to prevent it from drifting too far away.

2 Goals

Main goals of the project were to analyze the current implementation and to find a new parallelization scheme to improve the parallel scaling of ABINIT.

When speaking of "scaling" we distinguish between weak and strong scaling. Strong scaling means that at a constant problem size the parallel speedup increases linearly with the number of used processors. Whereas by weak scaling we mean that the time to solve a problem with increasing size can be held constant by enlarging the number of used processors. Strong scaling is usually limited by Amdahl's law [11] to a certain number of hosts that depends on the problem size. Weak scaling is easier to achieve for hundreds of nodes and is often merely limited by the resources (e.g. memory) per node. On the long run we want to achieve maximum weak scaling to enable the application of ABINIT to bigger problems while keeping the calculation times constant at the current level. For simplicity we focus in this report on the strong scaling behaviour of ABINIT, but the weak scaling behaviour can be extrapolated from that to a certain extend.

3 The Current Implementation of ABINIT

The effective one-particle Schrödinger equation derived by means of the DFT represents an eigenvalue problem that has to be solved selfconsistently. In ABINIT a self-consistency cycle is begun by constructing a starting density and deriving a starting potential. Then the electronic eigenvalues (bands) and eigenvectors are determined by a band-by-band conjugate gradient (CG) scheme [12, 13], during which the density (i.e. the potential) is kept fixed until the whole set of functions has been obtained. At the end of one CG loop the density is updated by the scheme of choice (e.g. simple mixing, or Anderson mixing). For a comparison of different schemes see e.g. [14]). For a more detailed description of the DFT implementation in ABINIT see [15].

Different levels of parallelization are implemented. The most efficient parallelization is the distribution of the k-points that are used to sample the reciprocal space on different processors. The required number of k-points depends on the system size and is determined by convergence tests. It usually decreases with increasing system size, so the scaling with the number of atoms is rather unfavourable. One can partially make up for this by distributing the work related to different bands within a given k-point. Since the number of bands increases with the system size, the overall scaling with number of atoms improves. A blocked version of the CG algorithm can be used to optimize the wavefunctions, which provides the possibility to parallelize over the bands within one block. Instead of a single eigenstate, as in the band-by-band scheme, *nbdblock* states are determined at the same time, where *nbdblock* is the number of bands in one block. This leads to a small increase in the time that is needed to orthogonalize the eigenvectors with respect to those obtained previously. Furthermore, to guarantee convergence, a too high value for *nbdblock* should not be chosen. A meaningful choice has to be determined for each system, but so far our experience is that $nbdblock \leq 4$ is a meaningful choice.

These parallelization methods, which are based on the underlying physics of the calculation, are useful only for a finite number of CPUs. Within the k-point parallelization the best speedup is achieved if the number of k-points is an integer multiple of the number of CPUs:

$$nkpt = n \times N_{\text{CPU}} \quad \text{with} \quad n \in \mathbb{N}. \tag{1}$$

Ideally, $n = 1$.
If the number of available CPUs is larger than the number of k-points required for the calculation, the speedup saturates. In this case, the additional parallelization over bands can improve the performance of ABINIT, if

$$N_{\text{CPU}} = nbdblock \times nkpt \quad . \tag{2}$$

In principle the parallelization scheme also works for $N_{\text{CPU}} = nbdblock$, which results in a parallelization over bands only.

4 Parallelization Techniques

We recently analyzed ABINIT for its performance at a specific problem size on a cluster system [16]. We found that about 97% of the running time of ABINIT is spent in the subroutine `vtowfk`, where 83% are spent in `cgwf`, the implementation of the band-by-band minimization scheme proposed by Teter et al. [12]. Thus, we decided to start our optimization project in this routine. The maximum strong scaling s for ABINIT with a parallel `cgwf` is given by Amdahls law [11]:

$$s = \frac{1}{1 - 0.83} = 5.88$$

A simplified pseudo code of the subroutine `cgwf` is given in Listing 1. The outer loop (Line 1) optimizes band-by-band, and the inner loop (Line 5) performs the conjugate gradient line minimization steps (by default 4 per band). One band `cwavef` is read out of the `cg`-array (the array holding all wavefunction coefficients, Line 2) and normalized. The operator of the non-local potential is applied (after a FFT) to the band in Line 4 (subroutine `nonlop`) and the result is stored in the $|ghc\rangle$ vector, which indicates the direction to proceed in the minimization (the gradient) of the band energy. The inner conjugate gradient loop proceeds as described in [13, 12] The `cg`, `ghc` and `gvnlc` arrays are updated at the end of each conjugate gradient step (Line 33-35).

As described in section 3 the current version of ABINIT is able to operate in parallel on a set of bands. In practice the scalability is limited to four processors due to the interdependency of these bands (each band has to be orthogonal to all others).

To go beyond the current scaling, we distribute the wave function coefficients (the elements of the vectors `cwavef`) among different processors. The only necessary points of communication are the calculations of dot products (they are marked with an asterisk (`*`) in Listing 1). The nonlop routine was also parallelized to enhance the parallelism further. A global reduction (MPI_ALLREDUCE) was used to accumulate the `gxafac` array on every node.

```
 |do iband=1, nband
 | cwavef(npw) = cg(:,npw)
*| call normalize(|cwavef>)
+| call nonlop(in |cwavef>, in |gvnlc>, inout |ghc>)
 | do iline=1,nline
*|  chc = lambda = <cwavef|ghc>
 |  |vresid> = -chc * |cwavef> + |ghc>
*|  resid(iband) = <vresid|vresid>
 |  |direc> = |ghc>
 |  ! orthogonalize
 |  do iiband=1,nband
*|   cgdirec = <cg(iiband)|direc>
 |   |direc> = -cgdirec*|cg(iiband)> + |direc>
 |  end do
*|  call precon(in cg(), out pcon(), inout direc())
 |  ! orthogonalize
 |  do iiband=1,nband
*|   cgdirec = <cg(iiband)|direc>
 |   |direc> = -cgdirec*|cg(iiband)> + |direc>
 |  end do
*|  dotgg = <direc|conjgr>
 |  |conjgr> = dotgg/dotgp * |conjgr> + |direc>
 |  ! orthogonalize |conjgr> to |iband>
*|  zdotures = <conjgr|cwavef>
 |  |direc> = -zdotures * |cwavef> + |conjgr>
 |  ! normalize direc
*|  dotr = <direc|direc>
 |  |direc> = 1/sqrt(dotr)*|direc>
+|  call nonlop(in |cwavef>, in |gvnlc>, inout |gh_direc>)
*|  dhc = <direc|ghc>
*|  dhd = <direc|gh_direc>
 |  ! calculate sinth, costh here
 |  cg(iband) = costh*|cwavef> + sinth*|direc>
 |  ghc() = costh*|ghc> + sinth*|gh_direc>
 |  gvnlc() = costh*|gvnlc> + sinth*|gvnl_direc>
 | end do ! iline=1,nline
 |end do ! iband=1, nband
 |! mimic old behavior
~|call allgather(cg())
```

Listing 1: Pseudocode for cgwf

Another approach has to be taken to perform the FFT in parallel (marked with a plus (+) in Listing 1). The 3D-FFT is parallelized in real space. First, a FFT is performed along all z-planes in parallel, second a FFT is done for all xy-lines in parallel. The wave function coefficients have to be redistributed at the beginning of the FFT and the real-space grid has to be swapped between Phase 1 (z-planes) and Phase 2 (xy-lines) of the FFT. The collective MPI routine MPI_ALLTOALL was used to perform that task.

5 Results

We conducted several benchmarks to test the scalability of our code on two different systems. A small PC cluster, called "Botanix", with dual core CPUs and the IBM SP5 [17] from CINECA were used to calculate a fixed size system. As already mentioned in section 2, we demonstrate only strong scaling which is limited due to the fixed system size. However, this should not influence the results for a small number of CPUs and gives us the possibility to extrapolate that a good weak scaling can be easily achieved for a huge number of processors.

The two nodes of Botanix come with 4 dual core Opteron 865 (1.8 GHz) CPUs per node and are interconnected with Gigabit Ethernet. The IBM SP5 comes with 64 nodes, each with 8 Power5 (1.9 GHz) Processors and is connected with an IBM High Performance Switch (Federation).

nproc (P)	Runtime (s)	Speedup
1	228.91	1.00
2	108.24	2.11
3	69.32	3.30
4	49.24	4.65
8	25.53	8.97
16	17.31	13.22

Table 1: Consumed Runtime and achieved Speedup of the parallel subroutine `cgwf` on the SP5.

The benchmark results for a small system with 43 atoms, 48728 plane waves, 126 bands and 921600 FFT grid points, which needs 21 iterations to reach self-consistency, are presented in the following. Figure 2 shows the parallel execution time and scaling of the parallelized `cgwf` routine. All benchmarked times and speedups on the SP5 are given in Table 1. Nearly all test cases on the SP5 show superlinear speedup which is due to the better cache utilization of the parallel variant. Both systems perform best if all calculations are done on a single node and the processors communicate via shared memory (up to 8 processors). The communication latency and bandwidth gets worse if the processes span more than 8 processors (one node). The result is a

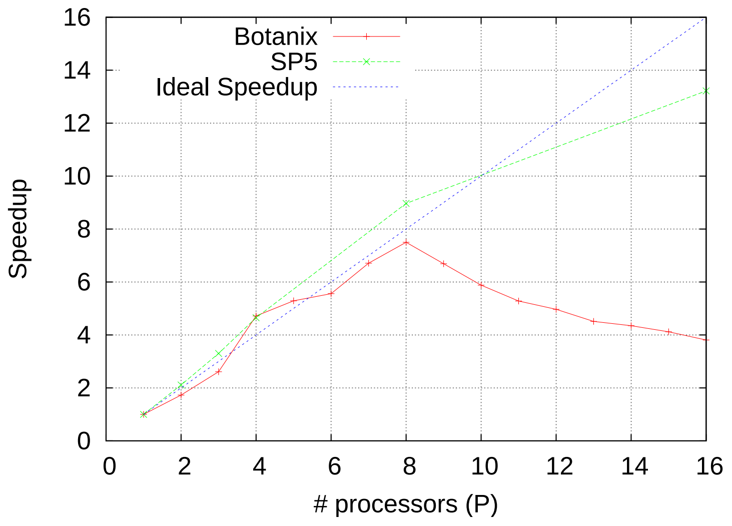

Figure 2: Speedup of the parallel `cgwf` routine on different systems.

catastrophic negative scaling on Botanix due to the very slow interconnection network (Gigabit Ethernet) and a slight decrease in the scaling on the SP5. We assume that this effect could be decreased if the collective communication would be topology aware and would adapt to this specific hierarchical interconnect layout.

The execution time of a fully converged calculation with ABINIT is examined in the following. Table 2 shows benchmark results for the SP5 and Figure 3 shows the according graphs for both systems. We see that the parallel scaling is very well up to 8 processors and gets slightly worse if the processes span more than one node (have to use a slower interconnect). The parallel overhead of the collective communication latency begins to increase for a bigger number of processors and a slower interconnect.

nproc (P)	Runtime (s)	Speedup
1	5724.9	1.00
2	3380.2	1.69
3	2269.7	2.52
4	1764.7	3.24
8	1242.4	4.61
16	1020.6	5.61

Table 2: Runtime consumed by a full calculation with ABINIT on the SP5.

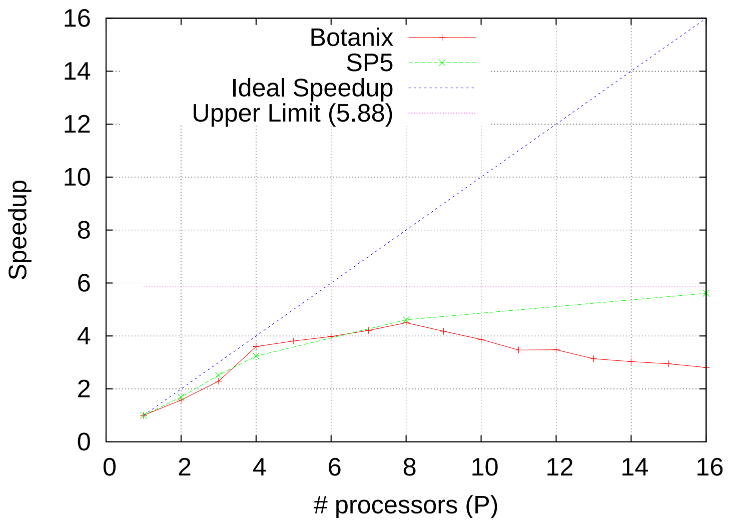

Figure 3: This Figure shows the parallel scaling of a full ABINIT calculation.

We know that the serial part of ABINIT uses about 17% of the application running time (cp. [16]. This means that the sequential (not parallelized) execution time for our specific example equals to

$$t = 0.17 \cdot 5724.9s = 973.23s$$

which clearly dominates the overall execution time on 16 processors ($> 95\%$).

6 Conclusion and Future Work

We can conclude that the `cgwf` routine has been parallelized efficiently. Strong scaling up to 16 processors with a small system is possible and weak scaling can be achieved with bigger input systems for much larger computers. The speedup of the whole calculation is asymptotically limited to 5.88 and is nearly reached with a fast interconnection network (shared memory). One needs to parallelize also the `vtowfk` and maybe other parts of the code to achieve a better scaling on large parallel computers.

7 Biography

Torsten Hoefler graduated 2005 in computer science in the special field of parallel computing. He was awarded with the Best Students Award of the Technical University of Chemnitz 2005 and the PARS Junior Research Price 2005 of the German Computer Socienty (GI). He is currently working as a member of the Open MPI [18] team to optimize collective communications and to prove the useability and efficiency of non blocking collectives. One of his projects deals with the efficient implementation of a quantum mechanical calculation (Abinit) with the help of non blocking collectives. This work represents the first steps in this direction.

8 Publications

- T. Hoefler, R. Janisch, and W. Rehm: *Analyzing the parallel scaling of Teter's conjugate gradient based minimization for Ab Initio calculations - to be submitted*

9 Acknowledgement

This work was carried out under the HPC-EUROPA project (RII3-CT-2003-506079), with the support of the European Community - Research Infrastructure Action under the FP6 "Structuring the European Research Area" Programme. T.H. especially thanks his scientific host Carlo Cavazzoni for very helpful dicsussions and comments.

References

[1] Gonze, X., Rignanese, G.M., Verstraete, M., Beuken, J.M., Pouillon, Y., Caracas, R., Jollet, F., Torrent, M., Zerah, G., Mikami, M., Ghosez, P., Veithen, M., Raty, J.Y., Olevano, V., Bruneval, F., Reining, L., Godby, R., Onida, G., Hamann, D., Allan, D.: A brief introduction to the ABINIT software package. Z. Kristallogr. **220** (2005) 558

[2] Hohenberg, P., Kohn, W.: Inhomogeneous electron gas. Phys. Rev. **136** (1964) B864

[3] Kohn, W., Sham, L.: Self-consisten equations including exchange and correlation effects. Phys. Rev **140** (1965) A1133

[4] ABINIT: http://www.abinit.org/ (2005)

[5] VASP: http://cms.mpi.univie.ac.at/vasp (2005)

[6] CASTEP: http://www.tcm.phy.cam.ac.uk/castep (2005)

[7] CPMD: http://www.cpmd.org (2005)

[8] PWSCF: http://www.pwscf.org (2005)

[9] Cavazzoni, C., Chiarotti, G.L.: A parallel and modular deformable cell Car-Parrinello code. Computer Physics Communications **123** (1999) 56–76

[10] Car, R., Parinello, M.: Unified approach for molecular dynamics and density-functional theory. Phys. Rev. Lett. **55** (1985) 2471

[11] Amdahl, G.M.: Validity of the single processor approach to achieving large scale computing capabilities. Readings in computer architecture (2000) 79–81

[12] Teter, M.P., Payne, M.C., Allan, D.C.: Solution of Schroedinger's equation for large systems. Physical Review B (1989) 12255–12263

[13] Payne, M.C., Teter, M.P., Allan, D.C., Arias, T.A., Joannopoulos, J.: Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients. Reviews of Modern Physics **64**(4) (1992) 1045–1097

[14] Eyert, V.: A comparative study on methods for convergence acceleration of iterative vector sequences. J.Comp.Phys. **124** (1995) 271

[15] Gonze, X., Beuken, J.M., Caracas, R., Detraux, F., Fuchs, M., Rignanese, G.M., Sindic, L., Verstraete, M., Zerah, G., Jollet, F., Torrent, M., Roy, A., Mikami, M., Ghosez, P., Raty, J.Y., Allan, D.: First-principles computation of material properties : the ABINIT software project. Computational Materials Science 25, 478-492 (2002)

[16] Hoefler, T., Janisch, R., Rehm, W.: A performance analysis of abinit on a cluster system. In Hoffmann, K.H., Meyer, A., eds.: Parallel Algorithms and Cluster Computing. Lecture Notes in Computational Science and Engineering (2006) accepted to be published.

[17] Ballabio, G., Boschi, S., Calonaci, C., Cavazzoni, C., Emerson, A., Gheller, C., Gori, R., Tarsi, A.: High Performance Systems User Guide. Volume 1.2. CINECA, Supercomputing Group, CINECA Consorzio Interuniversitario, Caseleccio Di Reno (2005)

[18] Gabriel, E., Fagg, G.E., Bosilca, G., Angskun, T., Dongarra, J.J., Squyres, J.M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R.H., Daniel, D.J., Graham, R.L., Woodall, T.S.: Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In: Proceedings, 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary (2004)