Parallel scaling of Teter's minimization for *Ab Initio* calculations

Torsten Hoefler and Wolfgang Rehm
Technical University of Chemnitz,
Dept. of Computer Science,
09107 Chemnitz, Germany
Email: {htor,rehm}@cs.tu-chemnitz.de

Rebecca Janisch
Technical University of Chemnitz,
Dept. of Electrical and Information Technology,
09107 Chemnitz, Germany
Email: rebecca.janisch@etit.tu-chemnitz.de

***Abstract*— We propose a parallelization scheme for the conjugate gradient method by Teter et. al. and report a detailed analysis of its scalability. We use MPI collective operations exclusively to take advantage of optimized collective implementations with possible hardware support. Our parallel conjugate gradient calculation can be applied in addition to the already implemented parallelism in the application ABINIT. We propose distribution schemes for the band vectors and the 3D-FFT, and provide both a detailed runtime and scalability analysis and a model for the used collective operations. We use this model of collective communication to predict the parallel scaling and to show that the scalability is mostly limited by the communication. Our codes scales up to 52 processors for a small 43 atom system and up to 120 processors for a larger 86 atom system for a single k-point on our test cluster. Our results suggest that non-blocking collective communication could be used to enhace the application running time especially for cluster computers.**

I. Introduction

Ab initio density functional theory (DFT, [28], [30]) has proven to be an invaluable tool in understanding modern materials. However, since this method is computationally demanding (O($natom^3$), where natom is the number of particles in the simulation box, in our case the number of atoms in the supercell), one is usually restricted to small model supercells. To tackle problems like extended defects or amorphous systems, code optimizations and parallelization are necessary.

There are many applications available which perfom *ab initio* calculations. However, many of them are only available as binary form or the source-code is licensed restrictively (cf. Related Work). We use the application ABINIT [20] to implement our parallelization scheme mainly because of its GPL license and the fact that it already implements a serial version of the Teter method. ABINIT solves the effective one-particle Schrödinger equation derived by means of the DFT. This equation represents an eigenvalue problem that has to be solved self-consistently. 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) minimization scheme [38], [40], 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) [13]. For further details please refer to [20].

A previous study with ABINIT [26] showed that the biggest share of the calculation time (about 87%) is used to minimize the bands with the conjugate gradient based method proposed by Teter et. al. [40].

The following Section presents related work in the field of quantum mechanical calculations and shows a rationale to choose ABINIT. We analyze the current serial calculation in Section I-C. A detailed scaling analysis is presented in Section II followed by a brief description of the actual implementation and measurements in Section III. Section IV draws conclusions and points out further work.

A. Related Work

Different approaches exist to determine the ground state atomic and electronic structure of a crystal.

Packages like the open source ABINIT and the commercial ones VASP [41] and CASTEP [7] perform a self-consistent field (SCF) calculation to determine the electronic ground state for a given arrangement of atoms, respectively electrons. The gradient of the ground state energy with respect to the ionic positions can then be used to shift the ions towards their equilibrium positions and repeat the electronic ground state calculation (static relaxation). In contrast, programs like CPMD [9] and PWSCF [8] are based on the *ab initio* molecular dynamics scheme of Car and Parrinello [6], where ions and electrons are treated as classical particles, coupled by pseudo-Newtonian equations of motion, and the positions of ions and electrons are varied at the same time. Here the computational effort needed to obtain a fully relaxed atomistic structure is smaller than in the SCF ground state - static relaxation scheme. However, during the calculation the electronic system is not in its adiabatic ground state, and the challenge is to prevent the algorithm from drifting too far away to converge. ABINIT is not subject of this problem because it uses the standard SCF calculation.

The Teter conjugate gradient has not been parallelized yet because most programs use the scheme proposed by Car and Parrinello [6] to solve the Schrödinger equation in parallel. However, the above-mentioned advantages and the fact that the Teter Method is very memory-efficient (needs only the

data for a single band to minimize it) make a parallelization useful.

B. Introduction to Electronic Structure Calculations

In solid state physics, bonding and electronic structure of a material can be investigated by solving the quantum mechanical (time-independent) Schrödinger equation,

$$\widehat{H}_{\rm tot}\Phi = E_{\rm tot}\Phi \quad , \tag{1}$$

in which the Hamilton operator $\widehat{H}_{\rm tot}$ describes all interactions within the system. The solution Φ, the wavefunction of the system, describes the state of all N electrons and M atomic nuclei, and $E_{\rm tot}$ is the total energy of this state.
Usually, the problem is split by separating the electronic from the ionic part by making use of the Born-Oppenheimer approximation [4]. Next we consider the electrons as independent particles, represented by one-electron wavefunctions ϕ_i. Density functional theory (DFT), based on the work of Hohenberg and Kohn [28] and Kohn and Sham [30], enables us to represent the total electronic energy of the system by a functional of the electron density $n(\vec{r})$:

$$\begin{aligned} n(\vec{r}) &= \sum_i |\phi_i|^2 & (2)\\ \rightarrow E &= E\left[n(r)\right] = F[n] + \int V_{\rm ext}(\vec{r})n(\vec{r})d\vec{r} \\ &= E_{\rm kin}[n] + E_{\rm H}[n] + E_{\rm xc}[n] + \\ &\quad \int V_{\rm ext}(\vec{r})n(\vec{r})d\vec{r} & (3)\end{aligned}$$

Thus the many-body problem is projected onto an effective one-particle problem, resulting in a reduction of the degrees of freedom from 3N to 3. The one-particle Hamiltonian $\hat{H}$ now describes electron i, moving in the effective potential $V_{\rm eff}$ of all other electrons and the nuclei.

$$\left.\begin{aligned} \hat{H}\ \phi_i &= \epsilon_i\ \phi_i \\ \left\{-\tfrac{\hbar^2\Delta}{2m} + V_{\rm eff}[n(\mathbf{r})]\right\}\phi_i(\vec{r}) &= \epsilon_i\ \phi_i(\vec{r}), \\ \text{where}\quad V_{\rm eff}[n(\vec{r})] &= V_{\rm eff}(\mathbf{r}) \\ V_{\rm eff}(\mathbf{r}) = V_{\rm H}(\vec{r}) + V_{\rm xc}(\vec{r}) + V_{\rm ext}(\vec{r}) \quad . & \end{aligned}\right\} \tag{4}$$

In (4), which are part of the so-called Kohn-Sham equations, $-\frac{\hbar^2\Delta}{2m}$ is the operator of the kinetic energy, $V_{\rm H}$ is the Hartree and $V_{\rm xc}$ the exchange-correlation potential. $V_{\rm ext}$ is the external potential, given by the lattice of atomic nuclei. For a more detailed explanation of the different terms see e.g. [34]. The self-consistent solution of the Kohn-Sham equations determines the set of wavefunctions ϕ_i that minimize the energy functional (3). In order to obtain it, a starting density $n_{\rm in}$ is chosen from which the initial potential is constructed. The eigenfunctions of this Hamiltonian are then calculated, and from these a new density $n_{\rm out}$ is obtained. The density for the next step is usually a combination of input and output density. This process is repeated until input and output agree within the limits of the specified convergence criteria.
There are different ways to represent the wavefunction and to model the electron-ion interaction. In this paper we focus on pseudopotential+planewave methods.
If the wavefunction is expanded in plane waves,

$$\phi_i = \sum_{\vec{G}} c_{i,\vec{k}+\vec{G}} e^{i(\vec{k}+\vec{G})\vec{r}} \tag{5}$$

the Kohn-Sham equations assume the form [38]

$$\sum_{\vec{G}'} H_{\vec{k}+\vec{G},\vec{k}+\vec{G}'} \times c_{i,\vec{k}+\vec{G}'} = \epsilon_{i,\vec{k}} c_{i,\vec{k}+\vec{G}} \quad , \tag{6}$$

with the matrix elements

$$\begin{aligned} H_{\vec{k}+\vec{G},\vec{k}+\vec{G}'} = \quad & \frac{\hbar^2}{2m}|\mathbf{k}+\mathbf{G}|^2\delta_{\mathbf{G}\mathbf{G}'} \\ & + V_{\rm H}(\mathbf{G}-\mathbf{G}') + \\ & V_{\rm xc}(\vec{G}-\vec{G}') + V_{\rm ext}(\vec{G}-\vec{G}') \quad . \end{aligned} \tag{7}$$

In this form the matrix of the kinetic energy is diagonal, and the different potentials can be described in terms of their Fourier transforms. Equation (6) can be solved independently for each k-point on the mesh that samples the first Brillouin zone. In principle this can be done by conventional matrix diagonalization techniques. However, the cost of these methods increases with the third power of the number of basis states, and the memory required to store the Hamiltonian matrix increases as the square of the same number. The number of plane waves in the basis is determined by the choice of the cutoff energy $E_{\rm cut} = \hbar^2/2m|\vec{k}+\vec{G}_{\rm cut}|^2$ and is typically of the order of 100 per atom, if norm-conserving pseudopotentials are used. Therefore alternative techniques have been developed to minimize the Kohn-Sham energy functional (3), e.g. by conjugate gradient (CG) methods (for an introduction to this method see e.g. [36]). In a band-by-band CG scheme one eigenvalue (band) $\epsilon_{i,\vec{k}}$ is obtained at a time, and the corresponding eigenvector is orthogonalized with respect to the previously obtained ones.

C. Conjugate Gradient

The conjugate gradient method itself is an enhancement of the steepest descent method [17]. Both methods can be used to minimize the value of a function $f(\vec{x})$, where $\vec{x}$ is a n-dimensional vector. The search direction in the n-dimensional space for each iteration is the direction of the steepest descent:

$$\vec{d^i} = -\frac{\partial f}{\partial \vec{x}^i} = -G\vec{x}^i \quad , \tag{8}$$

where G is the gradient operator. The search continues in the direction of $\vec{d^i}$. The size of the step depends on the function $f(\vec{x})$. The length can be found by calculating the point on the line $\vec{x} + a\vec{d}$ where the gradient is orthogonal to the search direction. This can be done by solving the equation $\vec{d}\cdot G(\vec{x}+a\vec{d}) = 0$. This technique reduces the value of the function along a specific line in the n-dimensional space and introduces slight errors at each iteration (moves in several dimensions away from the minimum). The conjugate gradient technique minimizes this error by taking the direction vector of the previous iteration into account (see [17] for a detailed description). The conjugate gradient vector $\vec{c}$ in iteration i can

be derived from $\vec{c}^{i-1}$ and the steepest descent vector $\vec{d}$ as follows:

$$\vec{c}^{i} = \vec{d}^{i} + \gamma^{i}\vec{c}^{i-1}, \quad (9)$$

$$\text{with} \quad \gamma^{i} = \frac{\vec{d}^{i} \cdot \vec{d}^{i}}{\vec{d}^{i-1} \cdot \vec{d}^{i-1}} \quad . \quad (10)$$

This scheme leads to the exact solution of a n-dimensional problem in n steps. Teter et al. adapted this method to the minimization of the bands in *ab initio* calculations. The considered function $f(\vec{x})$ is the Kohn-Sham energy functional [40] E and the vector $\vec{x}$ with the wave function ψ_e of the actual band (electron). The elements of ψ_e are the wave function coefficients relative to the plane wave basis $e^{i\cdot\vec{k}\vec{r}}$. The Hamilton operator H takes the place of the gradient operator G. The steepest descent search direction at a given position ψ_e is given by (see equation (8))

$$\chi_e^i = -(H - \lambda_e^i)\psi_e^i, \quad (11)$$

$$\lambda_e^i = \langle\psi_e^i|H|\psi_e^i\rangle. \quad (12)$$

We have to preserve the orthogonality constraint for total energy band structure calculations. The easiest way to do this is to orthogonalize the conjugate gradient vector to all bands. This can be done by a simple projection of ψ on each band,

$$\chi_e^{'i} = \chi_e^i - \sum_{j\neq e} \langle\psi_j|\chi_e^i\rangle\psi_j \quad . \quad (13)$$

A preconditioning scheme is used to accelerate the convergence. This scheme is described in [40] and uses a diagonal polynomial matrix $p(\vec{f})$ with $f_{ipw} = \frac{E_{kin}(ipw)}{E_{kin}^e}$. $E_{kin}(ipw)$ denotes the kinetic energy of the plane wave ipw. The preconditioning is applied to the direction vector:

$$\chi_e^{''i} = p(\vec{f})\chi_e^{'i}. \quad (14)$$

The new direction vector $\chi_e^{''i}$ is not orthogonal to all bands (due to preconditioning) and requires a new orthogonalization step identical to equation (13). Now the preconditioned steepest descent direction $\chi_e^{''i}$ is calculated and used to aquire the conjugate gradient direction. According to equation (9) and (10), the conjugate gradient vector τ_e^i in iteration i equals to (note that γ_e^i differs from (10) due to preconditioning)

$$\tau_e^i = \chi_e^{''i} + \gamma_e^i\tau_e^{i-1} \quad , \quad (15)$$

$$\text{with} \quad \gamma_e^i = \frac{\langle\chi_e^{''i}|\chi_e^{'i}\rangle}{\langle\chi_e^{''i-1}|\chi_e^{'i-1}\rangle} \quad . \quad (16)$$

A next step includes the orthogonalization of τ_e^i to the current band and the normalization of τ_e^i,

$$\tau_e^{'i} = \tau_e^i - \langle\psi_e^i|\tau_e^i\rangle\psi_e^i \quad , \quad (17)$$

$$\text{and} \quad \tau_e^{''i} = \frac{\tau_e^{'i}}{\sqrt{\langle\tau_e^{'i}|\tau_e^{'i}\rangle}} \quad . \quad (18)$$

The remaining steps to aquire the length of the conjugate gradient vector are not discussed in detail because no new classes of computational operations are added. A single application of the Hamilton operator equivalent to equation (12), two dot products (xDOT), and scaled vector additions (xAXPY) are added in the last part of the computation. The reader may refer to [38], [40] for a detailed description. Different linear algebra operations are identified by their BLAS [31] abbreviations (e.g. xDOT for dot product) in the following.

II. Parallelization

The band-by-band minimization as described in Section I can be performed with standard linear algebra operations and the application of the Hamilton operator. We come upon four necessary basic operations, dot products (xDOT - equations (11,13,16,17,18)), vector scaling (xSCAL - equations (13,14,15)), vector addition (xAXPY - equations (13,15,17)) and the application of the Hamilton operator. The Hamilton operator can be split into the kinetic energy term and the local and non-local operators,

$$H = E_{kin}^e + V_{loc}^e + V_{nl}^e \quad . \quad (19)$$

The operators of E_{kin}^e and V_{loc}^e are applied in reciprocal space (k-space), while the operator V_{nl}^e is applied in real space. The transition from reciprocal to real space is done with a 3D-FFT which has to be performed on both directions (transform to real space, apply potential operator, transform back to reciprocal space).

Our parallelization scheme is distinct from already implemented schemes as k-point parallelization or band parallelization. We calculate the linear algebra and the 3D-FFT in parallel by distributing the plane wave coefficients (vector elements) and real space grid points among the different processors. This scheme allows us to incorporate both available established schemes (band, k-point) in a later stage of the development. The parallelization strategy and scalability prediction is presented in the following sections.

A. Linear Algebra

We use a simple and well understood distribution scheme for the parallel conjugate gradient. The plane wave coefficients are distributed among different processors. The maximum number of plane waves is called npw and a single plane wave is addressed as ipw. As we concentrate on an ideal load balancing to avoid process skew during calculation, we distribute the vectors equally. The number of processors is P and each processor owns npw/P plane wave coefficients (a processor may have less, if $npw/P \notin \mathbb{N}$). The analysis of the different vector operations (xDOT,xSCAL,xAXPY) shows that only xDOT requires communication. We used the collective operation MPI_ALLREDUCE on an isolated communicator to perform this task. The parallel running time t_{lin} can be described as:

$$t_{lin} = t_{calc,lin} + t_{red}(P,1)$$

$$\text{with} \quad t_{calc,lin} = \frac{npw}{P} \cdot n_{lin} \cdot t_{npw}. \quad (20)$$

$t_{red}(P, count)$ represents the time to perform MPI_ALLREDUCE for $count$ `double complex` values on a communicator with P nodes, n_{lin} the number of vector

operations between two communication steps and t_{npw} the time to fetch, multiply and store two `double complex` values. The vector distribution scheme is shown in Figure 1.

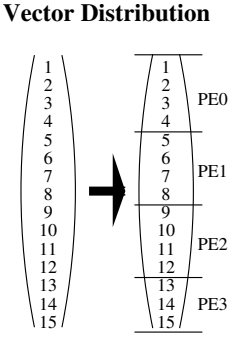

Fig. 1. Plane wave coefficient vector distribution scheme for $npw = 15$ and $P = 4$. Each Processing Element holds a group of coefficients.

B. 3D-FFT

The 3D-FFT implementation is based on Stefan Goedecker's serial implementation for Abinit [18] which leverages the special properties of the *ab initio* FFT and avoids the transformation of zeros (see Figure 2), thus saving a huge amount (around 50%) of the computational time. This prevents the usage of standard optimized FFT libraries as FFTW [14] or vendor optimized FFT libraries because they can not take advantage of the special properties of the FFT space. We parallelized the serial FFT to work in two steps. The first transformation is done over all z-planes of the FFT box in parallel and the second one over all xy-lines in parallel. If N_x, N_y, N_z denote the number of grid points in each direction, every processor has N_z/P z-planes and $N_y \cdot N_x/P$ xy-lines. We need two global communication operations to implement this. Each processor has to gather all needed plane wave coefficients before transforming its z-planes. This communication operation may be unbalanced because each processor needs a different amount of coefficients (see Figure 2), but experimental results show that the effect of this communication unbalance is negligible. Another global exchange has to be performed between z-plane transformation and xy-line transformation. We use MPI_ALLTOALL with a seperate communicator for this step to enable the MPI implementation to optimize communications. The time of this computation step can be assessed as

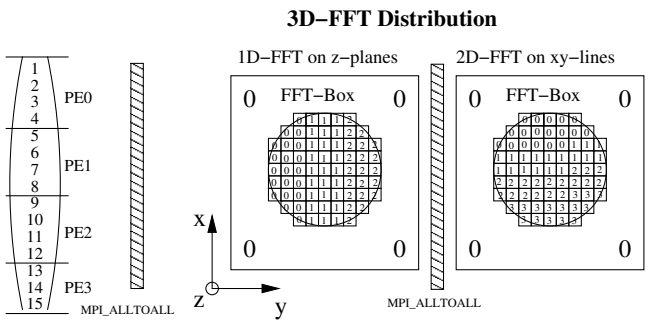

Fig. 2. Example of a 3D-FFT with $N_x = N_y = N_z = 9$ and 4 Processing Elements

$$\begin{aligned} t_{ges,fft} &= t_{calc,fft} + t_{comm,fft} \\ \text{with} \quad t_{calc,fft} &= \frac{N_z}{P} \cdot t_{fft,z} + \frac{N_x \cdot N_y}{P} \cdot t_{fft,xy}. \end{aligned} \quad (21)$$

The communication time for the irregular distribution is hard to assess, we present a worst-case calculation with

$$t_{comm,fft} = t_{a2a}(P, npw/P) + t_{a2a}(P, N_x \cdot N_y/P) \quad ,$$

where $t_{a2a}(P, count)$ is the MPI_ALLTOALL communication time of *count* `double complex` values between P partners.

C. Collective Communication

The running times of the collective communication operations can be modelled with a simple network model. Previous studies [25] showed that the LogP model (consisting of the parameters **L**atency, **o**verhead, **g**ap and Number of **P**rocessors, see [10]) is well suited to assess the running time of collective communication for small messages. Additional studies [39] show that the extended LogGP model (which adds an extra **G**ap parameter to model the gap per byte for bulk transfers, [2]) is well suited to model larger messages. The used Open MPI [15] with the basic collective component performs MPI_ALLREDUCE as MPI_REDUCE to node 0 followed by MPI_BCAST to all nodes. MPI_REDUCE and the MPI_BCAST are implemented as a binomial tree ($\forall P > 8$) and MPI_ALLTOALL is implemented linearly. We use the LogP model and assume a simple linear scaling with the message size to simplify the model equations. This simplification is valid because all communication operations used in the linear algebra part (Section II-A) communicate only a single `double complex` value. The number of communicated values in the 3D-FFT is indirectly proportional to the number of processors P. Thus, we can assume small message sizes for a large number of processors. The running times of the allreduce operation can be assessed as (see [25] Section 4.2 or Fig. 7)

$$t_{red}(P, size) = 2 \cdot size \cdot (2o + L + (\lceil log_2 P \rceil - 1) \cdot max\{g, 2o + L\}) \quad . \quad (22)$$

The alltoall communication time can be estimated by

$$t_{a2a}(P, size) = size \cdot (P \cdot (2o + L) + (P - 1) \cdot g) \quad (23)$$

III. Implementation and Measurements

We used Kielmann's `logp-mpi` benchmark [29] to assess the LogP parameters for the used InfiniBand™ network and a data size of 16 bytes (a single `double complex` value). The benchmark returned $L = 9.78\mu s$, $o = 0.05\mu s$ and $g = 0.01\mu s$. As a test system we used a typical supercell which is employed to model electronic properties of the silicon (Si) - silicon oxide (SiO2) interface in microelectronic devices.

The parametrized functions are shown together with collective benchmark results returned by the Pallas Microbenchmark [37] in the upper part of Figure 3. We see that our

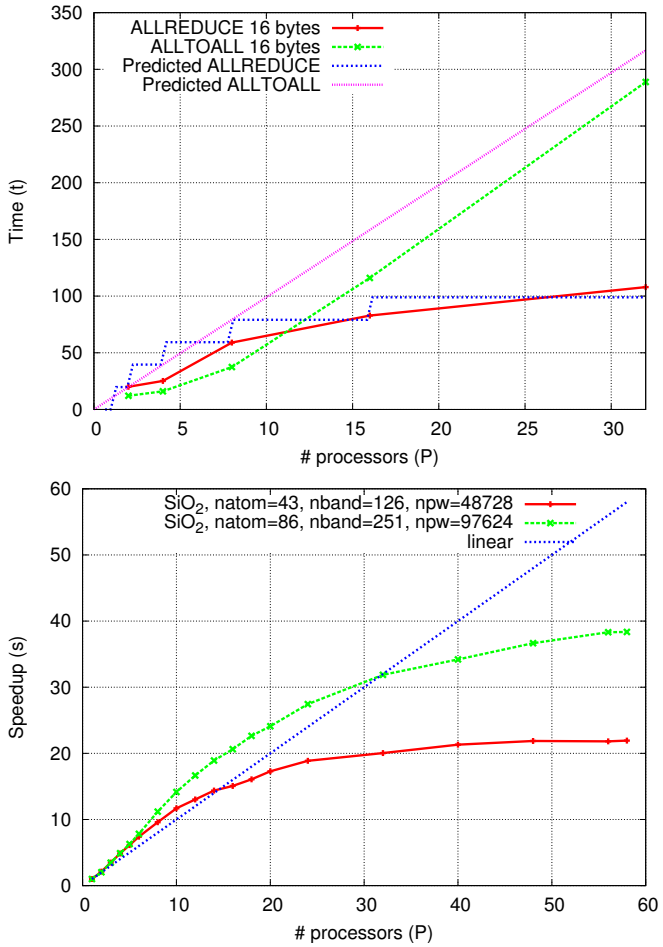

Fig. 3. Benchmarked and predicted collective Function Latencies (upper) and Parallel Scaling of the parallelized `cgwf` Routine for a single k-point (lower)

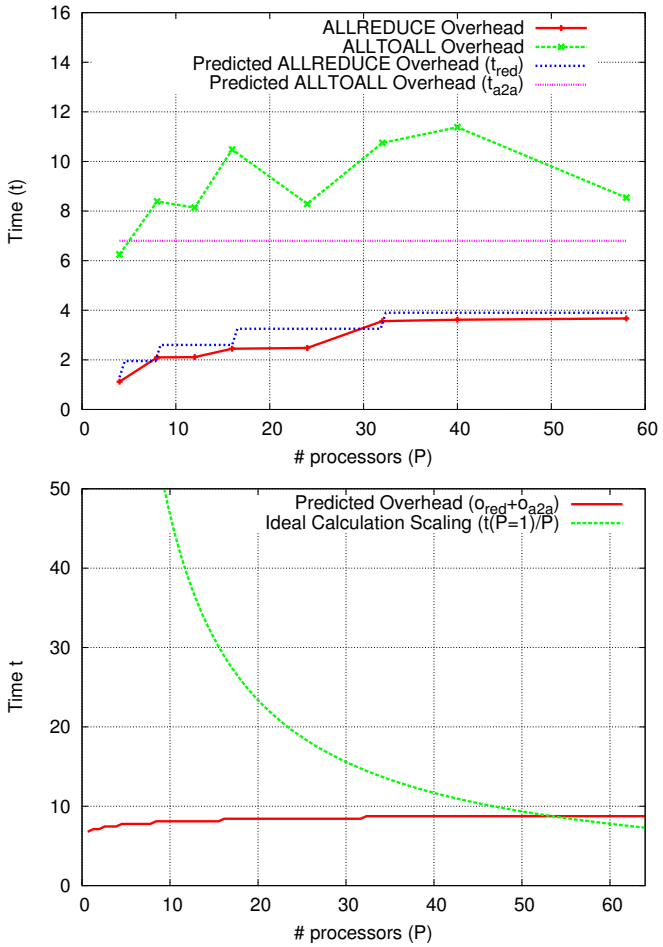

Fig. 4. The benchmarked Overhead of a system with 43 atoms (upper). The predicted overhead and ideal calculation scaling for this system (lower).

LogP model for the collective operations (equations (22) and (23)) overestimates the real results slightly. This is due to a special property of the used InfiniBand™ interconnect. We investigated this behavior and proposed a more accurate model for InfiniBand™ in [24], [27] but we accept this small overestimation for this study to ensure simplicity.

We implemented our proposed scheme in ABINIT 4.6.5 by modifying the routine `cgwf`. Only collective communication operations are used to exchange data. This enhances optimization possibilities and readability (cf. [21]) of the code. The strong scaling results for two small $Si-SiO_2$ systems with 43 atoms, 126 bands, 48728 plane waves and a 61x61x256 FFT grid as well as 86 atoms, 251 bands, 97624 and a 81x81x256 FFT grid are shown in the lower part of Figure 3. The diagram shows that the parallel speedup saturates around P = 22 for 43 atoms and around P = 40 for 86 atoms. We used MPE2 and Jumpshot [42] to measure the communication overhead of the calculations with 43 atoms. The upper part of Figure 4 shows the predicted and measured overhead. The predicted functions for t_{red} and t_{a2a} were deduced from (22) and (23). Our code issues 9 reduction operations with a single element per band (`double complex`, see (11,16,17,18)). Two times $nband$ reduction operations are issued to orthogonalize the direction vector to all other bands (see (13)). Alltogether all mentioned operations have to be performed for each band per SCF cycle. Thus, $nband \cdot (9 + 2 \cdot nband)$ reduction operations have to be performed:

$$o_{red}(P) \quad = \quad nband \cdot (9 + 2 \cdot nband) \cdot t_{red}(P, 1) \quad .(24)$$

For 126 bands and with (22) combined with our measured LogP parameters we get

$$\begin{aligned} o_{red}(P) &= 126 \cdot (9 + 2 \cdot 126) \cdot 2 \cdot (\lceil log_2 P \rceil \cdot 9.88) \\ &= 65772 \cdot (\lceil log_2 P \rceil \cdot 9.88) \end{aligned} \quad (25)$$

The overhead prediction for $o_{a2a}(P)$ can be assessed similarly. The prediction underestimates the real overhead because it does not take the process skew and additional synchronization overheads into account. The measured overhead shows the maximal overhead for all processes. We see that the additional synchronization overhead is not neglegible and slows down the collective communication. However, the scaling predictions of the communication overhead are correct (MPI_ALLTOALL

remains constant and MPI_ALLREDUCE grows logarithmically). The lower part of Figure 4 shows the accumulated predicted overhead ($o_{red}(P) + o_{a2a}(P)$) and the ideal scaling curve of `cgwf` ($t(P = 1)/P$). The overhead crosses the ideal scaling at $P \approx 52$. At this point saturation is reached and additional parallelization can not lead to any speedup (t increases due to logarithmic scaling of $a_{red}(P)$). We see the same saturation effect in the lower part of Figure 3. We have to solve the following equation to deduce the exact crossing point:

$$\begin{aligned} t_{idealscal} &= o_{ges} \\ \frac{t(P=1)}{P} &= \frac{t_{a2a}(P, count) + t_{red}(P, count)}{t(P=1)} \end{aligned} \quad (26)$$

In the 86 atom example (omitted due to space restrictions) the saturation occurs around $P \approx 120$. However, since at this point the scaling is already far from ideal, using P = 120 would mean a waste of resources. Usually one has to define a overhead which is still meaningful (e.g. 50%) and solve (26) for $o_{ges}^{'} = 50\% \cdot o_{ges}$. Doing so for our 43 atom example results in $P_{50\%} \approx 28$ which is a reasonable number of processors for this problem size.

A. Scaling with the System Size

The three basic parameters $nband$, npw and $nfft = N_x \cdot N_y \cdot N_z$ scale linearly with the number of atoms $natom$. If we assume this linear dependency, the calculation time

$$t_{calc,ges} \sim nband \cdot (t_{projbd} + t_{calc,lin} + t_{calc,fft}) \quad (27)$$

(see (13,20,21)) with $t_{projbd} = nband \cdot t_{calc,lin}$ scales with $O(natom^3/P)$. The communication overhead

$$\begin{aligned} o_{comm,ges} \sim\ & nband \cdot \\ & (nband \cdot t_{red}(P, 1) + t_{a2a}(P, 1)) \end{aligned} \quad (28)$$

scales with $O(log_2 P \cdot natom^2)$ in P and $natom$. We see that, for a fixed number of processors, P, the communication complexity ($O(natom^2)$) is much lower than the computation complexity ($O(natom^3)$). Thus we should be able to reach a very high reasonable speedup (e.g. 50% communication overhead) by slightly scaling the system. Typical systems we would like to calculate have about 500 atoms and should not be limited in theoretical scaling with today's supercomputers.

B. Overlap of Computation and Communication

This section analyzes if and how our proposed parallelization scheme can benefit from the overlapping of communication and computation. This is meant as an outlook to possible future work to reduce the overhead caused by the collective operations. Overlapping of computation and communication is a common tool to increase the scalability of parallel applications by lowering their communication overhead (cf. [1], [3], [5], [11], [12], [33]). However, this scheme does only apply to point-to-point communication operations because non-blocking collective operations are not defined in the defacto standards of parallel computing (MPI [35]/PVM [16]). We are currently developing an extension to the MPI standard to support non-blocking collective operations [23]. First results (cf. [22], [32]) indicate a reasonable potential for overlapping.

To assess the potential of overlapping for our parallelization of the Teter scheme, we can distiguish between the linear algebra parallelization of the dot-products that uses MPI_ALLREDUCE to perform this operation and the 3D-FFT that uses MPI_ALLTOALL.

The MPI_ALLREDUCE can not be overlapped for the calculation for a single band because the result is needed immediately after the operation. There is no calculation which can be performed without having the reduced value. However, ABINIT is able to calculate more than a single band in parallel. This could be used, together with a double-buffering technique which calculates more than a single band on one process in "parallel", to enable overlapping of communication and computation.

The MPI_ALLTOALL can be overlapped with a pipelined technique of the parallel 3D-FFT which uses the available network bandwidth more efficiently (cf. [19]).

The potential to increase the efficiency of overlapping communication and computation while retaining all the advantages of collective communication is given and can be applied to our parallelization. However, the implementation requires new programming concepts which have to be tested with smaller examples before it can be applied to a complex real-world application like ABINIT.

IV. Summary and Conclusions

We show a parallel implementation of Teter's band-by-band conjugate gradient scheme which uses only collective communication (cf. [21]). The analysis of the parallel scaling and the communication overhead and the provided model for the used collective communications shows that our implementation is scalable. Our model is also able to predict the number of processors to run specific calculations with a reasonable overhead. Comparisons of the prediction with the real scaling show the accuracy. We show that the simple LogP model is able to predict communication latencies quite accurately and that the parallel scaling of our implementation is mainly limited by the communication overhead. A new concept of non-blocking collective communication would help to overlap communication with computation and lead to lower overheads and better scaling. That could benefit especially cluster computers with restricted communication performance. In the future we are going to evaluate the suitability of non-blocking collective communication and provide a proof of concept code. So far, our code scales up to 28 processors for a small 43 atom system and up to ≈ 50 processors for a larger 86 atom system.

Acknowledgements: Major parts of this work have been 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. Thanks to Dr. Lumsdaine from

the Open Systems Lab at Indiana University who supported the authors to finish the final version of this paper.

REFERENCES

[1] Tarek S. Abdelrahman and Gary Liu. Overlap of computation and communication on shared-memory networks-of-workstations. pages 35–45, 2001.

[2] Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman. LogGP: Incorporating Long Messages into the LogP Model. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1995.

[3] Francoise Baude, Denis Caromel, Nathalie Furmento, and David Sagnol. Optimizing metacomputing with communication-computation overlap. In *PaCT '01: Proceedings of the 6th International Conference on Parallel Computing Technologies*, pages 190–204, London, UK, 2001. Springer-Verlag.

[4] M. Born and R. Oppenheimer. Zur quantentheorie der molekln. *Ann. Physik*, page 457, 1927.

[5] Ron Brightwell and Keith D. Underwood. An analysis of the impact of mpi overlap and independent progress. In *ICS '04: Proceedings of the 18th annual international conference on Supercomputing*, pages 298–305, New York, NY, USA, 2004. ACM Press.

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

[7] CASTEP. http://www.tcm.phy.cam.ac.uk/castep, 2005.

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

[9] CPMD. http://www.cpmd.org, 2005.

[10] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. LogP: towards a realistic model of parallel computation. In *Principles Practice of Parallel Programming*, pages 1–12, 1993.

[11] Anthony Danalis, Ki-Yong Kim, Lori Pollock, and Martin Swany. Transformations to parallel codes for communication-computation overlap. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 58, Washington, DC, USA, 2005. IEEE Computer Society.

[12] R. Dimitrov. *Overlapping of Communication and Computation and Early Binding: Fundamental Mechanisms for Improving Parallel Performance on Clusters of Workstations*. PhD thesis, Mississippi State University, 2001.

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

[14] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. special issue on "Program Generation, Optimization, and Platform Adaptation".

[15] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004.

[16] G. A. Geist and V. S. Sunderam. Network-based concurrent computing on the pvm system. *Concurrency: Pract. Exper.*, 4(4):293–311, 1992.

[17] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic, London, 1981.

[18] S. Goedecker. Fast radix 2, 3, 4, and 5 kernels for fast fourier transformations on computers with overlapping multiply–add instructions. *SIAM J. Sci. Comput.*, 18(6):1605–1611, 1997.

[19] S. Goedecker, M. Boulet, and T. Deutsch. An efficient 3-dim FFT for plane wave electronic structure calculations on massively parallel machines composed of multiprocessor nodes. *Computer Physics Communications*, 154:105–110, August 2003.

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

[21] Sergei Gorlatch. Send-receive considered harmful: Myths and realities of message passing. *ACM Trans. Program. Lang. Syst.*, 26(1):47–56, 2004.

[22] T. Hoefler and A. Lumsdaine. Design, Implementation, and Usage of LibNBC. 08 2006.

[23] T. Hoefler, J. Squyres, G. Bosilca, G. Fagg, A. Lumsdaine, and W. Rehm. Non-Blocking Collective Operations for MPI-2. 08 2006.

[24] T. Hoefler, C. Viertel, T. Mehlan, F. Mietke, and W. Rehm. Assessing Single-Message and Multi-Node Communication Performance of InfiniBand. 9 2006. Accepted for publication at the 5-th International Symposium on Parallel Computing in Electrical Engineering.

[25] Torsten Hoefler, Lavinio Cerquetti, Torsten Mehlan, Frank Mietke, and Wolfgang Rehm. A practical Approach to the Rating of Barrier Algorithms using the LogP Model and Open MPI. In *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPP'05)*, pages 562–569, June 2005.

[26] Torsten Hoefler, Rebecca Janisch, and Wolfgang Rehm. A performance analysis of abinit on a cluster system. In Karl Heinz Hoffmann and Arnd Meyer, editors, *Parallel Algorithms and Cluster Computing*. Lecture Notes in Computational Science and Engineering, 2006. accepted to be published.

[27] Torsten Hoefler, Torsten Mehlan, Frank Mietke, and Wolfgang Rehm. LogfP - A Model for small Messages in InfiniBand. In *Proceedings, 20th International Parallel and Distributed Processing Symposium IPDPS 2006 (PMEO-PDS 06)*, April 2006.

[28] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864, 1964.

[29] Thilo Kielmann, Henri E. Bal, and Kees Verstoep. Fast measurement of logp parameters for message passing platforms. In *IPDPS '00: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, pages 1176–1183, London, UK, 2000. Springer-Verlag.

[30] W. Kohn and L.J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev*, 140:A1133, 1965.

[31] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic Linear Algebra Subprograms for FORTRAN usage. In *In ACM Trans. Math. Soft., 5 (1979), pp. 308-323*, 1979.

[32] LibNBC. http://www.unixer.de/NBC, 2006.

[33] G. Liu and T.S. Abdelrahman. Computation-communication overlap on network-of-workstation multiprocessors. In *Proc. of the Int'l Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1635–1642, July 1998.

[34] R. M. Martin. *Electronic Structure - Basic Theory and Practical Methods*. Cambridge University Press, Cambridge, 2004.

[35] Message Passing Interface Forum. MPI-2: Extensions to the Message-Passing Interface. Technical Report, University of Tennessee, Knoxville, 1997.

[36] J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Num.*, page 199, 1991.

[37] Pallas GmbH. Pallas MPI Benchmarks - PMB, Part MPI-1. Technical report, Pallas GmbH, 2000.

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

[39] Jelena Pjesivac-Grbovic, Thara Angskun, George Bosilca, Graham E. Fagg, Edgar Gabriel, and Jack J. Dongarra. Performance Analysis of MPI Collective Operations. In *Proceedings of the 19th International Parallel and Distributed Processing Symposium, 4th International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 05)*, Denver, CO, April 2005.

[40] Michael P. Teter, Micheal C. Payne, and Douglas C. Allan. Solution of Schroedinger's equation for large systems. *Physical Review B*, pages 12255–12263, 1989.

[41] VASP. http://cms.mpi.univie.ac.at/vasp, 2005.

[42] Omer Zaki, Ewing Lusk, William Gropp, and Deborah Swider. Toward scalable performance visualization with Jumpshot. *The International Journal of High Performance Computing Applications*, 13(3):277–288, Fall 1999.