Theor Chim Acta (1993) 84: 325–334

Theoretica Chimica Acta

Parallel processing for *ab initio* total energy pseudopotential calculations

Lyndon J. Clarke

Edinburgh Parallel Computing Centre, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK

Received October 1, 1991/Accepted January 28, 1992

Summary. The total energy pseudopotential method is well suited to parallel processing. This paper discusses a procedure for calculating the valence electronic wavefunction in a given ionic configuration, and considers the exploitation of parallel processing using a data parallel approach. The implementation of this procedure on two message passing i860 based machines, containing up to 64 nodes, is described, and the prospects for massively parallel execution are examined.

Key words: Total energy pseudopotential – Valence electronic wavefunction

1. Introduction

The UK CP Grand Challenge Programme is a collaborative venture, involving groups at five universities. The objectives are the development and exploitation of parallel computing in Car–Parrinello [1] methods for the computer simulation of systems in chemistry, physics and materials science. Three important components of this project are (**a**) a range of scientific projects, (**b**) acquisition of parallel hardware, and (**c**) development of parallel software. We now briefly describe each of these components.

The projects are wide ranging, including studies of the behaviour of dissolved hydrogen in metals, surface structure and surface chemistry, simulation of molecular beam epitaxy growth, silicates at pressures and temperatures corresponding to the earth mantle, and fluids and clusters of metallic elements. Each of these projects are of current interest and demand exceptional computing power.

The programme has acquired a 25% share of a dedicated "Grand Challenge" machine, in collaboration with a similar Grand Challenge Programme and The University of Edinburgh, located at Edinburgh University. This is a message passing MIMD machine consisting of 64 nodes, each of which comprises an Intel i860 application processor, 16 Mbyte of application memory, and two Inmos T800 processors. The two transputers provide message routing between arbitrary pairs of nodes, and access to the host operating system. The peak performance of the i860 application process is 80 Mflops, so that the peak performance of this

machine is notionally 5 Gflops. In fact, as is widely known, it is not practical to obtain this kind of performance from the i860. Hand coded computational kernels which achieve ~ 30 Mflops are known, and it would be more conservative to rate the machine as having a performance of around 2 Gflops. Each of the transputers provides up to four serial Inmos links, which can be connected to the links of other nodes in almost any fashion, providing a notional peak communication bandwidth of 16 MBs at each node. Again, it is not practical to realise this performance, in this case due to the capabilities of the memory system and the fact that message routing is performed in software on the transputers.

The software effort is primarily directed at producing code which is able to efficiently exploit this machine, although we have also ported the parallel program to the Intel iPSC/860. Over the past few years, the various groups involved in the collaboration have independently developed three programs, for vector processors, which implement CP-type algorithms. These groups have been able to perform some major simulations, and some of this work has been published [2, 5, 7]. These programs differ, reflecting the differing scientific interests of the authors, but they have a common core which has been implemented in similar ways, and they each contain of order 10 000 lines of FORTRAN. A core program was derived from the program written by Mike Payne et al., to which individual groups may contribute procedures as required for specific projects. A key objective in the development of the first version of this parallel program was the retention of the overall structure of the original program, and the re-use of as much serial code as possible. In fact it turned out that in about 10 000 lines of serial Fortran, only about 600 lines of the application needed to be modified.

In this paper we shall concentrate on the core program, the concurrency therein, the parallelism which we were able to exploit on the machines described above, and the prospects for exploitation of massively parallel machines. Section 2 briefly describes the total energy pseudopotential method as implemented in this program, and in Sect. 3 we discuss the parallel computation.

2. Description of methods

The total energy pseudopotential method is well described in the literature, see for example [3, 6]. In this section we shall only briefly discuss the fundamental features. The core of the computation consists of calculating the valence wavefunction for a given ionic configuration, using a conjugate gradients technique. We shall describe this technique in more detail, since it consumes practically all of the run time in real calculations and is the basis for our discussion of parallelism in the next section.

2.1. Fundamentals

Bloch's theorem states that in a periodic system, each electronic wavefunction can be written as the product of a cell periodic part and a wavelike part. The cell periodic part can be expanded using a basis set consisting of discrete plane waves whose wave vectors are reciprocal lattice vectors. Therefore, each electronic wavefunction can be written as a sum of plane waves

$$\phi_{ik}(r) = \sum_G c_{i,k+G} e^{i(k+G)\cdot r}$$

We have changed the problem of calculating an infinite number of electrons to one of calculating a finite number of electronic wavefunctions at an infinite number of k-points. Methods have been devised for calculating very accurate approximations to the electronic potential from a filled band by considering the electronic states at special, small, sets of k-points in the Brillouin zone.

The plane wave basis set required to expand the electronic wavefunctions is also infinite. The plane wave basis set in total energy pseudopotential calculations is truncated by including only those waves whose kinetic energy is less than some chosen cut-off energy. Of course, this truncation leads to an error in the computed total energy. We can, in principle, control this error by performing calculations at increasing cut-off energy, until the calculated total energy has converged.

The wavefunctions of nuclei have negligible overlap in almost all materials. One notable exception is helium, which is a quantum liquid at low temperature. In general, however, we can say that the nuclei are indistinguishable and classical mechanics can be applied to them. We therefore treat the nuclei classically. The electrons, conversely, do have significant overlap, and we must treat them quantum mechanically. The potential field of the nuclei is handled as an external field acting on the electronic system.

It is extremely expensive to perform all electron calculations. Expanding the wavefunctions in a plane wave basis set, an extremely large number of plane waves are required in order to describe the core orbitals and follow the rapid oscillations of the valence electrons in the core region. It is well known that most physical properties of solids are dependent on the valence electrons to a far greater degree than the core electrons. The frozen core approximation treats the nucleus and core electrons as a single entity, replacing the nuclear potential by a pseudopotential, and the valence electron wavefunctions by pseudo-wavefunctions.

The pseudopotential is chosen such that the pseudo-wavefunctions and the actual valence electron wavefunctions are identical outside some core region, and within the core region the pseudo-wavefunctions contain no radial nodes. This vastly reduces the number of plane waves needed to describe the pseudo-wavefunctions compared to the actual valence wavefunctions. The form of the pseudopotential is not unique, and the core program represents the non-local, angular momentum dependent, pseudopotential in the form due to Kleinman and Bylander [4].

The Hohenberg–Kohn–Sham theorem states that (**a**) the ground state energy of a many particle system in an external field is a unique functional of the density, and (**b**) the ground state electron density is the one which minimises the total energy. The total ground state energy of a system can be written in terms of the ground state wavefunctions ϕ:

$$E_{tot} = \langle \phi | H_{tot} | \phi \rangle = T + U_{i\text{-}i} + U_{e\text{-}i} + U_H + U_{XC}$$

where T is the kinetic energy, $U_{i\text{-}i}$ is the ion-ion coulomb energy, $U_{e\text{-}i}$ is the electron-ion energy through the pseudopotential, U_H is the Hartree energy and U_{XC} is the exchange-correlation energy.

Due to the conservation of charge and the stationary property of the total energy, one obtains the Kohn–Sham equations for a single particle:

$$H | \phi_{n,k} \rangle = E | \phi_{n,k} \rangle$$

where H is the Kohn–Sham hamiltonian:

$$H = \nabla^2 + V_{ion} + V_H + V_{XC}$$

V_{ion} is the potential acting on the electron due to the presence of the ions, V_H is the Hartree potential and V_{XC} is the exchange-correlation potential.

The exchange-correlation potential is unknown. The simplest method of describing this term is to utilise the local density approximation, and this is almost universally employed in total energy pseudopotential calculations. This approximation constructs V_{XC} at some point by assuming that it is the same as that in a uniform electron gas of the same density. This is the only uncontrolled approximation in the *ab initio* total energy pseudopotential method.

2.2. Solution of the Kohn–Sham equations

Self-consistent solutions to the Kohn–Sham equations can be found using conventional matrix diagonalisation techniques, however this is a computationally expensive procedure and severely limits the sizes of systems which can be studied.

Car and Parrinello had the insight to consider treating the plane wave coefficients in the expansion of the wavefunction as dynamical variables in a classical molecular dynamics approach. In the usual partially constrained equations of motion the wavefunction is subjected to an acceleration $(H - \lambda_i)\phi_i$, where $\lambda_i = \langle \phi_i | H | \phi_i \rangle$.

After calculation of these accelerations, the equations of motion are integrated for some timestep Δt, using a finite difference method such as the Verlet algorithm. The wavefunctions are then no longer orthogonal, and must be orthogonalised to one another. The electronic potentials, Hartree and exchange, are then recalculated, and the Kohn–Sham hamiltonian is updated.

Two factors lead to the molecular dynamics method being computationally more efficient than matrix diagonalisation. In the first place, it is possible to divide the calculation of the acceleration into two parts, one of which is diagonal in real space and the other diagonal in reciprocal space. Thus the time taken to compute the acceleration is dominated by the need to perform fourier transforms between real and reciprocal space, and vice versa. In the second place, the processes of calculating the eigenstates of the Kohn–Sham hamiltonian and obtaining self-consistency are performed simultaneously in the molecular dynamics method.

2.3. Minimisation of the Kohn–Sham functional

One problem with the molecular dynamics method is that the size of the timestep Δt must be decreased with the size of the system being studied, in order to ensure stable evolution of the Kohn–Sham hamiltonian. This problem can be circumvented by utilising a direct search for the minimum of the Kohn–Sham energy functional.

Teter, Payne, Allan and Joannopoulos [6] have described a conjugate gradients method for directly locating the minimum of the Kohn–Sham functional, which allows extremely fast total energy pseudopotential calculations to be performed.

In this method the constrained conjugate gradient iteration described below is applied to single bonds. One can apply this procedure to the first band until some convergence criteria are met, then to the second, third etc. The whole process is repeated with increasingly stringent convergence criteria until satisfactory overall convergence is obtained.

In the first instance the steepest descent vector α_i^m for iteration m of band i is calculated. This is essentially the same as the calculation of the acceleration in the molecular dynamics method briefly discussed above. The constraint of orthogonality of bands must be observed, and the steepest descent vector is orthogonalised to all other bands:

$$\alpha_i'^m = \alpha_i^m - \sum_{i \neq j} \langle \phi_j | \alpha_i^m \rangle \phi_j$$

The Kohn–Sham hamiltonian has a broad eigenvalue spectrum, and the convergence of the conjugate gradients algorithm is improved by use of a diagonal preconditioning matrix, K, which is essentially the inverse of the kinetic energy operator.

The preconditioned steepest descent vector β_i^m is then obtained:

$$\beta_i^m = K\alpha_i'^m$$

and is again orthogonalised to all bands:

$$\beta_i'^m = \beta_i^m - \langle \phi_i^m | \beta_i^m \rangle \phi_i^m - \sum_{i \neq j} \langle \phi_j | \beta_i^m \rangle \phi_j$$

The conjugate direction γ_i^m, in the presence of preconditioning, for the mth iteration can now be calculated by:

$$\gamma_i^m = \beta_i'^m + \eta_i^m \gamma_i^{m-1}$$

where

$$\eta_i^m = \frac{\langle \beta_i'^m | \alpha_i'^m \rangle}{\langle \beta_i'^{m-1} | \alpha_i'^{m-1} \rangle}$$

and

$$\gamma_i^1 = 0$$

A further orthogonalisation to the present band is now formed and a normalised conjugate direction $\gamma_i'^m$ calculated. This is then used in order to determine the energy minimum and update the wavefunction. The combination:

$$\phi_i^{m+1} = \phi_i^m \cos\theta + \gamma_i'^m \sin\theta$$

is a normal vector which is orthogonal to all bands $j \neq i$. Any vector of this form satisfies the constraints of orthonomality for the electronic wavefunctions.

We require the value of θ which minimises the Kohn–Sham functional. This is estimated by considering a first order expansion of the energy in θ:

$$E(\theta) \approx E_0 + E_1^c \cos 2\theta + E_1^s \sin 2\theta$$

We can evaluate the unknowns in this expression with three pieces of information. $E(0)$ is already known, and since $H|\phi_i^m\rangle$ has been calculated to determine the steepest descents vector, we can calculate $\partial E/\partial\theta$ at $\theta = 0$ quite easily. Finally, we can calculate the energy at some other value of θ, close to zero, in order to obtain the three unknowns and the "best" value of θ to use in computing the new wavefunction.

3. Exploitation of parallel computing

When we consider extracting parallelism from existing algorithms, and codes, then it is often useful to examine the complexity of the problem. We therefore begin by introducing a little notation with which to describe these calculations.

Let A be the number of atoms in the system under question. The number of electronic bands B, is roughly equivalent to A, depending on the valency of the particular atoms, and the number of plane wave coefficients C used to describe a particular wavefunction is usually about $100B$.

In order to calculate the steepest descents vector, and the step length θ we must transform quantities, such as the charge density, between real and reciprocal spaces. This is most quickly achieved using a three dimensional fast fourier transform algorithm. In order to use this algorithm we must enclose the plane wave energy cut-off surface, in reciprocal space, in a rectangular box.

The FFT box must be large enough that the cut-off surface is surrounded by surfficient vacuum to prevent any significant interaction between the identical copies of the system introduced by the fourier transform. The number of mesh points within this box, D, is generally about $10C$.

3.1. Complexity

The storage requirement of the total energy pseudopotential method described above is dominated by the electronic wavefunction, and increases as BC. In other words, the storage increases as A^2. The FFT mesh requires storage proportional to D, in turn proportional to A, and should not be ignored as the prefactor is rather large. The computational cost of performing a single conjugate gradients iteration on a single band is dominated by the cost of fourier transforms and orthogonalisations. We shall ignore the cost of all other operations which scale no worse than B or C.

The cost of calculating the steepest descent vector is dominated by the requirement to perform two fourier transforms, each of which takes $\mathcal{O}(D \log D)$ operations. In order to calculate the Kohn–Sham energy at the trial value of θ, the trial wavefunction must be transformed into real space so that the charge density can be computed, and then the charge density must be transformed into reciprocal space in order to calculate the Hartree energy. Calculation on the energy after updating the wavefunction requires a further two fourier transforms. Thus six fourier transforms are required in all.

The steepest descent vector is orthogonalised to all bands, at a cost of $\mathcal{O}(BC)$ operations. After preconditioning, the steepest descent vector is again orthogonalised to all other bands. The orthogonalisation of the conjugate gradients direction is with the present band only, requiring C operations, and is not important.

Given that there are B bands, we can see that the cost of performing a single conjugate gradients iteration on every band is $\mathcal{O}(B^2C)$ for orthogonalisation and $\mathcal{O}(BD \log D)$ for fourier transforms.

3.2. Parallelism

We currently have access to parallel machines which deliver performance at the Gigaflop level, and anticipate machines delivering Teraflops before the end of

this decade. It would seem that a Teraflop machine must contain many, thousands of, nodes. Whereas such machines will probably be MIMD, the sheer number of nodes indicates that applications mounted on them will be data parallel. We therefore seek to exploit data parallelism in the above calculations.

Special k-points. The easiest way to parallelise these computations would be to assign each k-point to a different node in the machine, since the calculations at different k-points are quite independent. Unfortunately, the number of special k-points in the total energy pseudopotential calculation is very small, often one, and independent of the size of system under study. We have not exploited this concurrency in our parallel program.

Electronic bands. The next conceptual level of data parallelism is at the electronic band level. In this approach one divides the bands into groups and assigns each group to a different node within the machine, thus utilising $\mathcal{O}(B)$ nodes. The calculation of the steepest descents vector for the band is then trivially parallel since each node can proceed independently.

In order to maintain the orthogonality of the new wavefunctions, after a single conjugate gradient iteration is applied to every band, it would be necessary for the vectors $\gamma_i'^m$ to be orthogonal, which in turn implies that the vectors $\beta_i'^m$ be orthogonal to one another. This could be performed by a method such as that utilised by Car and Parrinello, in their molecular dynamics approach, in which the following algorithm would be repeatedly applied to the vectors $\beta_i'^m$ in parallel:

$$\beta_i''^m = \beta_i'^m - \tfrac{1}{2} \sum_{j \neq i} \langle \beta_j'^m | \beta_i'^m \rangle \beta_j'^m$$

In this problem every node needs to access the same data stored at every other node. The communication pattern required for this is all-to-all global, in which every node sends the identical message to every other node. Notice the similarity with the classical n-body problem. Given N nodes, this communication takes $\mathcal{O}(BC/N)$ time in terms of inter-node bandwidth, and $\mathcal{O}(N)$ time in terms of communication setup.

The number of iterations which would be required in order to mutually orthogonalise the η_i increases with the number of bands. For this reason, and the fact that the single channel bandwidth in the target machine(s) is rather small, we have not attempted to exploit this concurrency in our parallel program.

Wavefunction coefficients. The finest conceptual level of data parallelism is at the wavefunction coefficient level. In this approach one divides the space of the wavefunction into a number of regions and assigns each region to a different node. In fact, one must assign portions of both the wavefunctions and the FFT mesh to nodes. One easy way to think about this is to imagine the FFT mesh as a cube, with a sphere in the middle – the wavefunction exists on mesh points within the sphere. This is, of course, a simplification since the FFT mesh is not usually cubic, and there may be more than one k-point within the mesh.

There is a great deal of flexibility in the manner in which the regions of the FFT mesh are defined and distributed over nodes, and a good division for one machine may well be far from optimal for another machine. Consider a hypothetical, although not unrealistic system, containing $A \approx 250$ atoms, from which $C \approx 25\,000$, and $D \approx 64^3$. Given that the target machine contains 64 nodes, the portion of a wavefunction stored at any node in this scenario is

$C/N = 25\,000/64 \approx 4000$ coefficients. With data stored again as FORTRAN **COMPLEX*16**, this vector occupies about 64 kbytes. This has the consequence that the orthogonalisation cannot make good use of the cache.

The vector length for the component 1D FFTs, which make up the 3D FFT, are only about 64. With data stored in FORTRAN **COMPLEX*16** data, this vector occupies about 1 kbyte. Therefore there is plenty of room for the vector, and another work vector if necessary, to be held in the cache during the 1D FFT calculation. In order to exploit this observation we have divided reciprocal space up into lines of points parallel to one of the axes (say the x-axis, since this was the inner loop throughout the program), and assigned distinct groups of lines to each node. The criteria for grouping lines were

1. The number of lines in each group should be equal – the cube from our simple picture should be evenly distributed over nodes.

2. The number of points in the FFT mesh corresponding to wavefunction coefficients within each group should be as nearly equal as possible – the sphere from our simple picture should be evenly distributed over nodes.

In actual fact it would be very difficult to satisfy the second of these criteria optimally. We have used a completely naïve algorithm which we find works in a satisfactory manner. We firstly divide these lines into those which do contain wavefunction plane waves, the *hard* lines, and those which do not contain wavefunction plane waves, the *easy* lines. We then just hand out the lines in turn to the nodes, starting with the hard lines and continuing with the easy lines until all the lines have been assigned. In our hypothetical example system, we have thousands of points in any plane, and there is very little load imbalance.

This completes the description of the manner in which the reciprocal space FFT mesh and the wavefunctions were distributed over nodes. The communication pattern required in the orthogonalisation is simply global summation of scalar values across all nodes. In a binary hypercube, this takes a time $\mathcal{O}(\log N)$ in terms of bandwidth and communication setup.

Now we shall turn to the distribution of the real space FFT mesh. Given that we want an equal amount of real space to be stored at each node, there seemed to be an obvious way to perform the FFT from reciprocal space to real space, and thus arrive at the distribution of real space. We could perform the 1D FFTs along x directly. We then had to rotate the data set such that lines parallel to the y-axis are assigned to nodes, perform the FFTs along y, rotate again so that lines parallel to the z-axis were assigned to nodes and perform the FFTs along the z direction. There are two rotations of the data set in this scheme.

The communication pattern for this problem is all-to-all personalised, in which every node sends a different message to every other node. Note the similarity with the matrix transpose problem. Given N nodes, in a binary hypercube, this communication takes $\mathcal{O}(D/N)$ in terms of bandwidth, and $\mathcal{O}(N)$ in terms of communication setup.

We can eliminate one of the communications in this 3D FFT. If we were to divide up the real space FFT mesh by assigning planes parallel to yz to nodes, then there is only one communication since the transposition of y and z dimensions can be performed locally. Of course, this introduces a load imbalance when the number of points along the x direction is not a multiple of N. It just so happens that the time wasted due to this load imbalance is less than the time taken to move the data around once, so we are actually using the latter

distribution of the real space. On a machine with better communication channels, we would choose the former, finer grained distribution.

In fact, for 3 of the 6 3D FFTs we have to do, we know *a priori* that most of these x-lines are zero, transforming from reciprocal space to real space, or will be ignored after the reverse transform. This is because these points correspond to plane waves which lie outside the cut-off energy – they are not inside the sphere. We can save most of the communications bandwidth by not bothering sending the data for these lines, since the receiver knows that it will just be zero. Exploiting this knowledge, the distribution of real space by planes rather than lines has reduced the communication requirement by about a factor of five.

3.3. Prospects for massive parallelism

We finally wish to consider the prospects for exploitation of massive parallelism in this problem.

The program we have written for the current i860 based machines is only able to exploit $N = \mathcal{O}(A^{1/3})$ nodes. It follows that the node memory must increase as $\mathcal{O}(N^5)$, so this program will soon run out of memory if we try to solve really big problems on really big computers. Given somewhat faster communications, the program could easily use $N = \mathcal{O}(A^{2/3})$ nodes, and a node memory increasing as $\mathcal{O}(N^4)$, which is not a great deal better.

We would really like to be able to fix the node memory, since we do not expect to be able to build really big machines with vast amounts of memory at each node. To do this, we need to use $\mathcal{O}(A^2)$ nodes. We can use this many nodes, provided we combine the distribution of the fourier transform mesh, and wavefunction plane waves, with the distribution of electronic bands.

In this case we would assign fixed size groups of band to identical groups of nodes. The number of nodes in each of these groups would increase as A. Each node within a group would then be assigned a fixed size portion of the FFT mesh, which can only be done if we forget about dividing the mesh up in two dimensions and actually perform the transform in parallel. If we can arrange for these node groups to be connected together in one big ring, such that nodes are connected to the corresponding nodes in neighbouring groups, then the bandwidth between node groups increases as A, which means that the all-to-all global communication can be performed in time $\mathcal{O}(BC/A) = \mathcal{O}(A)$. This is acceptable, since $\mathcal{O}(A)$ was the run time of a conjugate gradient iteration on all bands anyway. Unfortunately, we do not actually get $\mathcal{O}(A)$ run time, since the time spent mutually orthogonalising the steepest descent vectors would increase faster than A.

Once we consider actual Teraflop machines which might appear later this decade, the kinds of calculations which we wish to perform, and some of the missing constants factors in this analysis, we might well find a slowly growing node memory acceptable. In this case a large part of our program could be used as the node group engine in this scenario. We are then able to utilise $N = \mathcal{O}(A^{5/3})$ nodes, and the node memory grows as $\mathcal{O}(N^{1/3})$.

4. Conclusion

The total energy pseudopotential method is highly concurrent. In this paper we have briefly described the utilisation of a conjugate gradient method for minimi-

sation of the Kohn–Sham energy functional, and discussed the execution of this method on message passing MIMD machines.

A serial FORTRAN program has been adapted to run in parallel on the i860 based Meiko Computing Surface and the Intel iPSC/860 computers, using a data parallel approach. We found that an effective distribution of the key data structures, i.e. the fourier transform mesh and the electronic wavefunctions, depends on the performance characteristics of the target machine, which highlights the requirement for performance modelling and machine characterisation.

Finally we discussed the prospects for exploitation of massive parallelism in these algorithms. This was certainly found to be possible, and perhaps without major modifications of existing codes. However, global communication patterns are an important part of the parallel program. In order to proceed in this direction, it will be necessary for prospective massively parallel machines to support very high throughput in global communications.

References

1. Car R, Parrinello M (1985) Phys Rev Lett 55:2471
2. Gillan MJ (1989) J Phys Condens Matter 1:689
3. Zunger A, Ihm J, Cohen MJ (1979) J Phys C, Solid State Physics 12
4. Kleinman L, Bylander DM (1982) Phys Rev Lett 48:1425
5. Bristowe PD, Payne MC, Joannopolous JD (1987) Phys Rev Lett 58:1348
6. Teter MP, Payne MC, Allan DC, Joanopoulos JD (1990) Molecular dynamics method for *ab-initio* total energy calculations
7. Remler D, Madden P (1990) Molec Phys 70:919