
ELSEVIER

Parallel Computing 26 (2000) 887–911

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Computational chemistry on Fujitsu vector–parallel processors: Development and performance of applications software

Alistair P. Rendell [a,*], Andrey Bliznyuk [a], Thomas Huber [a], Ross H. Nobes [b], Elena V. Akhmatskaya [b], Herbert A. Früchtl [b], Paul W.-C. Kung [c], Victor Milman [d], Han Lung [e]

[a] *Supercomputer Facility, Australian National University, Canberra ACT 0200, Australia*
[b] *Fujitsu European Centre for Information Technology, 2 Longwalk Road, Stockley Park, Uxbridge UB11 1AB, UK*
[c] *Molecular Simulations, 9685 Scranton Road, San Diego, CA 92121-3752, USA*
[d] *Molecular Simulations, The Quorum, Barnwell Road, Cambridge CB5 8RE, UK*
[e] *Fujitsu America, 3055 Orchard Drive, San Jose, CA 95134-2022, USA*

Received 28 May 1999; accepted 20 July 1999

Abstract

In this and a preceding paper, we provide an introduction to the Fujitsu VPP range of vector–parallel supercomputers and to some of the computational chemistry software available for the VPP. Here, we consider the implementation and performance of seven popular chemistry application packages. The codes discussed range from classical molecular dynamics to semiempirical and ab initio quantum chemistry. All have evolved from sequential codes, and have typically been parallelised using a replicated data approach. As such they are well suited to the large-memory/fast-processor architecture of the VPP. For one code, CASTEP, a distributed-memory data-driven parallelisation scheme is presented. © 2000 Published by Elsevier Science B.V. All rights reserved.

Keywords: Fujitsu supercomputers; Molecular dynamics; Semiempirical quantum chemistry; Ab initio quantum chemistry; Parallelisation; Performance

* Corresponding author.
E-mail address: alistair.rendell@anu.edu.au (A.P. Rendell).

PII: S0167-8191(00)00017-X

1. Introduction

In the preceding paper [1], we introduced a range of high-performance vector–parallel processors from Fujitsu known as the VX/VPP300/VPP700 series (or VPP series for short). These distributed-memory machines feature proprietary vector processors connected via a high-throughput crossbar switch. As was discussed, programming the VPP series requires consideration of the vector unit efficiency in addition to the normal issues associated with the communication latency and bandwidth for a distributed-memory machine. Computational chemistry applications are normally coded using explicit message passing (such as MPI [2]) or by using a 'global-memory' model (such as Linda [3] or the Global Arrays [4]) which effectively hides the underlying message passing from the user.

In the present paper, we describe the development and the sequential and parallel performance of a variety of well-known computational chemistry packages on the VPP. Seven application packages are considered, taken from the areas of classical molecular dynamics, and semiempirical and ab initio quantum chemistry. The packages can be applied to problems involving solids, liquids and gases, and together their combined functionality covers a large fraction of what is loosely termed computational chemistry. Most of these application packages have been parallelised using MPI, although one uses Linda and another the Global Arrays.

Our discussion is broken down into the three application areas, and within these the different application codes are described. The application codes are the AMBER [5] and MASPHYC [6] molecular dynamics codes, the MOPAC2000 [7] semiempirical quantum chemistry code, and the *Gaussian® 98* [8], DMol^3 [9–12], MOLPRO [13] and CASTEP [14] ab initio quantum chemistry codes. Further details on all the benchmarks reported in this paper are available on request from the authors.

2. Classical molecular dynamics codes

In classical molecular dynamics (MD) [15], the behaviour of atoms and molecules is simulated using equations of motion from classical mechanics. Usually we have a penalty function which may, for example, represent the potential energy of a molecule as a function of conformation or be a measure of how well a conformation fits an experimentally measured property. By evaluating the derivative of this function with respect to the co-ordinates of the atoms in the system, it is possible to run a classical dynamics scheme and simulate the motion of the system over time. How accurately the results of such simulations reflect what is observed will not be discussed here, but obviously depends on a number of factors, not least of which is the underlying use of classical mechanics and the form of the penalty function.

In practice, the domain of applicability of MD simulations is intrinsically limited by the molecular nature of the systems under study. Specifically, to simulate the dynamics of a molecule by numerical integration of Newton's equations of motion, the time step required must be small enough to resolve the highest frequency mode in the system. For a molecular system this is typically a bond stretching motion and

requires a time step of the order of 1 fs (10^{-15} s) or less. Unfortunately, many interesting phenomena occur in the microsecond or longer timeframe, requiring over 10^9 integration steps. This is particularly true for biological systems where, for example, it takes on the order of microseconds for protein folding to occur. To compute such processes in a reasonable elapsed time (1–2 months) necessitates that each time step in an MD simulation takes about 3–4 milliseconds (ms); this is substantially faster than the 1–10 s that is typically required at the moment on a high-end workstation. For this reason alone it is obviously desirable that MD codes have both the highest possible performance on single processors, and are then able to utilise multiple processors in an efficient fashion.

The implementation of MD algorithms on vector and parallel computers has been investigated thoroughly by Plimpton [16] and is also covered in great depth in a recent review [17]. Three different approaches are commonly adopted in implementing MD on parallel computers:

- replicated data (or 'atom decomposition'), where each processor has a full copy of the co-ordinate and force arrays;
- force decomposition, often based on a systolic loop approach, where particle data is fully distributed over the processors and data packets must be passed between nodes until the force computation is complete. This approach is highly favourable for massively parallel computers due to its reduced communication requirement. A drawback, however, is that it does not take advantage of Newton's third law and each pairwise interaction is computed twice;
- spatial or domain decomposition, where the simulation cell is divided into equal regions with each region being allocated to a processor. Each processor is then responsible for calculating forces and updating positions and velocities of the particles within its assigned region. This approach is highly efficient for simulations of very large systems, where the cut-off used in evaluating potential interactions is very short in comparison with the overall dimensions of the simulation cell.

All commonly used MD packages, such as AMBER [5], CHARMM [18], DL_POLY [19], GROMOS96 [20] and MASPHYC [6], use the replicated-data parallelisation scheme. This is primarily due to the simplicity of the method and ease of implementation, although it does offer some other advantages [17,21]. A schematic outline of the replicated-data approach is shown in Fig. 1.

An initial set of co-ordinates and velocities is broadcast to all processors. If necessary, a pairlist describing which of the $N(N-1)/2$ two-body interactions (where N is the number of particles) are to be included is calculated. Each node evaluates the force contributions for the atoms assigned to that node, and then the partial force arrays on each node are summed via a global reduction operation. The equations of motion are integrated and any constraints are applied, and the updated co-ordinates and velocities are then broadcast once again to all processors in readiness for the next simulation time step.

The inherent weakness of the replicated-data approach is in the global communication requirements, specifically the global reduction operation needed to sum the force arrays. This will limit the scalability for large processor counts. However, for machines such as the Fujitsu VPP, with only a modest number of powerful

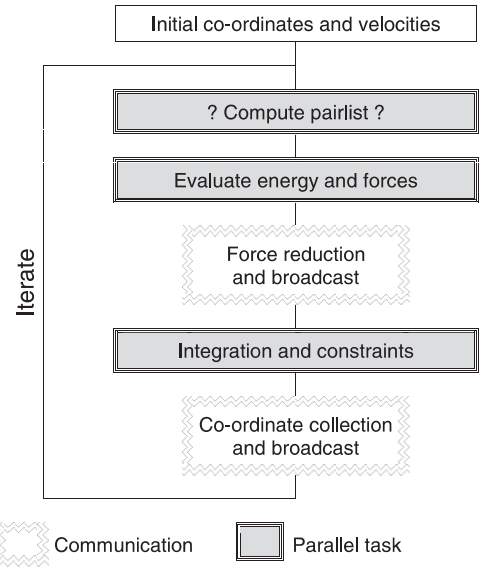

Fig. 1. General scheme for replicated-data parallel MD calculation.

processors, this communication requirement will not be a serious impediment to good performance.

2.1. Amber

At the Australian National University, we have been working to enhance the performance of the AMBER 5.0 molecular dynamics code [5] for single and multiple processors of the Fujitsu VPP range. The performance of the code has been assessed by using standard benchmarks available from the AMBER WWW site (http://www.amber.ucsf.edu/amber/amber.html).

2.1.1. Single processor performance on the Fujitsu VPP

The most time-consuming part in a MD simulation is the calculation of the non-bonded forces, and achieving good performance in this part of the code is critical to achieving good overall performance. In this respect, several changes were made to the non-bonded part of the AMBER code for the VPP along the lines outlined in the preceding paper [1], i.e. increasing vector lengths and the arithmetic density in vector loops and avoiding memory bank conflicts. We note that MD codes are particularly prone to memory bank conflicts that occur as a result of the small number of atom types and repeated access to data relating to those atom types.

Brief details of the standard AMBER 5.0 benchmarks and the performance obtained on the VPP300 using our tuned AMBER 5.0 code are given in Table 1. The times are compared with those given on the AMBER WWW site for other machines. Overall, the performance of the VPP compares favourably with other vector

Table 1
Comparison of the non-setup times (ms) for one simulation step of the AMBER benchmarks

	Benchmark							
Standard number	5	1	2	3	6	4	7	8
System	DNA	DNA[a]	DNA[a]	DNA[a]	Plas[b]	DNA[a]	H_2O	dhfr
Cut-off (nm)	Vac	0.8	0.9/1.2	1.2	1.2	PME[c]	PME[c]	PME[c]
Number of atoms	4282	7682	7682	7682	11585	7682	12288	22930
Fujitsu VPP300 (7 ns)	206	327	386	464	679	715	1008	2570
Cray T90	170	270	390	510	720	760		
NEC SX4	220	330	440	540	810	1210		
SGI R10000	320	440	690	1400	2150	1220	2150	4890
DEC Alpha 5/625	240	290	450	950	1550	820	1630	3550
Sun UltraSPARC™ (167 MHz)	600	590	900	1720	2930	2650	4640	10740

[a] Solvated in water with counter-ions.
[b] Plastocyanin in water.
[c] Particle mesh Ewald.

platforms like the NEC SX4 and Cray T90. To give an indication of the absolute performance of the code, a sustained performance of 425 Mflop/s (which is 20% of the peak performance of 2.2 Gflop/s [1]) was measured when performing 1000 MD time steps for benchmark 6.

2.1.2. Parallel performance on the Fujitsu VPP

Fig. 2 shows parallel AMBER speedup curves on the VPP300 for the three larger benchmarks. Benchmark 6 employs a cut-off in the computation of long-range interactions, while benchmarks 7 and 8 use the particle mesh Ewald (PME) method.

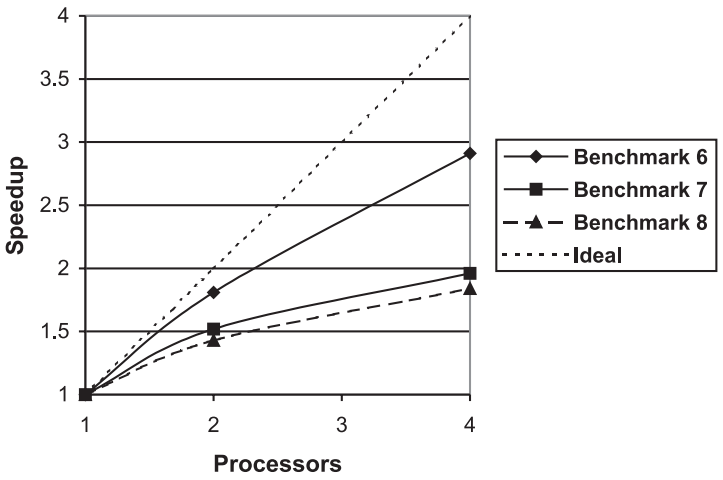

Fig. 2. Speedup curves for AMBER benchmark calculations.

As evident from Fig. 2, the performance of parallel AMBER on the VPP is mixed, with benchmarks 6 and 7 achieving a speedup of about 3 on four processors, while benchmark 8 performs slightly worse with a speedup of about 2.6 on four processors. The poorer scaling for benchmark 8 is largely due to its more complex nature and the fact that the time taken to satisfy bond constraints is significantly longer (18% of the total time on one processor, compared with 1.4% for benchmark 6 and even less for benchmark 7). Specifically, implementation of constraints in AMBER is carried out using the Shake algorithm and it is well known that this parallelises and vectorises relatively poorly [22]. Other limitations to the scalability are load imbalances in the computation of the non-bonded interactions, communication overheads associated with summing the forces and broadcasting the co-ordinates, and the input/output operations required to write trajectory files to disk.

In comparing the speedup results reported here with other parallel MD work, it is important to realise two things. First, AMBER is a general purpose molecular dynamics simulation code and the benchmarks reported here are 'real' biomolecular applications that include long-range Coulomb interactions. (In contrast much of the work of Plimpton [16] is based on a specialised MD code and considers only short-range Lennard–Jones interactions.) Second, the ratio of network bandwidth to peak performance on the VPP300 (0.26 bytes/flop) is much less than on machines like the Intel Paragon (1.9 bytes/flop), making speedup comparisons very difficult.

2.2. MASPHYC

MASPHYC [6] is a general-purpose molecular dynamics package developed independently by Fujitsu. It is designed as a tool to allow atomistic simulations to be used to guide computational materials design in fields such as advanced ceramics and functional materials. MASPHYC has a number of attractive features, including an advanced graphical user interface and an extensive library of potentials, enabling simulations on a wide variety of materials from organic crystals to metals and ceramics. The package consists of a workbench with a number of subsystems (database control, data entry, MD control, analysis and display). This workbench is used to set up and control MASPHYC/MD, the molecular dynamics simulator, which runs on a computational server such as the Fujitsu VPP series.

MASPHYC has been specially tuned for good performance on vector processors. In addition, the two-body interaction and Coulomb interaction calculations have been parallelised, using both message passing (via MPI) and through Fujitsu's proprietary data-parallel VPP Fortran. Typically, outer loops over the atoms in the system being simulated are distributed cyclically to processors during the force evaluation. In the case of the Ewald summation used in the evaluation of Coulomb interactions, a band partitioning of the reciprocal-space vectors is used.

As an example of the practical application of MASPHYC/MD, we consider a simulation of dimyristoylphosphatidylcholine (DMPC) in the liquid crystal phase. The study of DMPC comes from a joint project between Fujitsu and Japan's Institute for Physical and Chemical Research.

DMPC's lipid bilayer structure is the fundamental framework of cell membranes and has characteristics that reveal various important functions of organic membranes. To understand such characteristics, it is necessary to analyse the structure and motion of the lipid molecules and the surrounding water at the atomic level through molecular dynamics. Table 2 shows the simulation conditions and Fig. 3 shows a snapshot of the lipid bilayer structure 600 ps after the start of the calculation. The simulation clearly shows that the lipid bilayer structure prevents the passage of water molecules and shows random variations in the thickness of the two layers at some points.

Fig. 4 shows the performance of MASPHYC/MD for this simulation. The MPI version is used here; the data-parallel Fortran version is expected to show almost identical performance. On a single VPP processor, simulation of this 9612 atom

Table 2
Simulation conditions used for DMPC simulation with MASPHYC

Ensemble	NTP (constant temperature and pressure)
Temperature	300 K
Pressure	1 atm
Time step	0.2 fs
Basic cell	$50 \times 50 \times 82$ Å
Periodic boundary condition	Three-dimensional
Simulation system	54 lipid molecules = 6372 atoms
	1080 water molecules = 3240 atoms
	Total = 9612 atoms

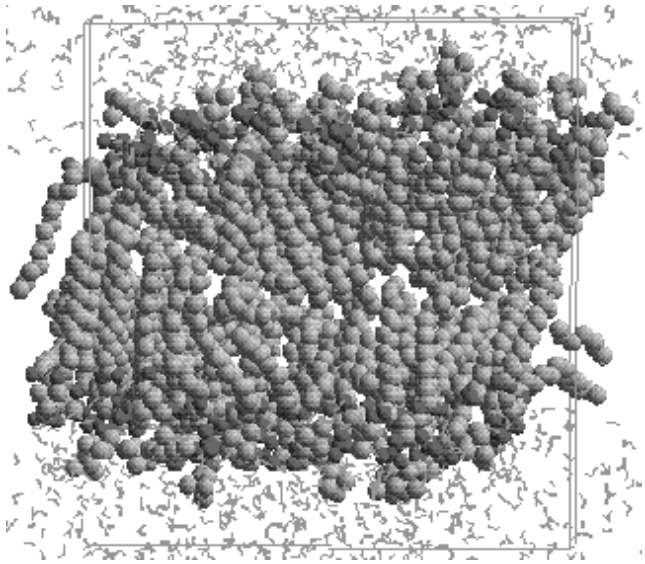

Fig. 3. Lipid bilayer structure 600 ps after start of simulation.

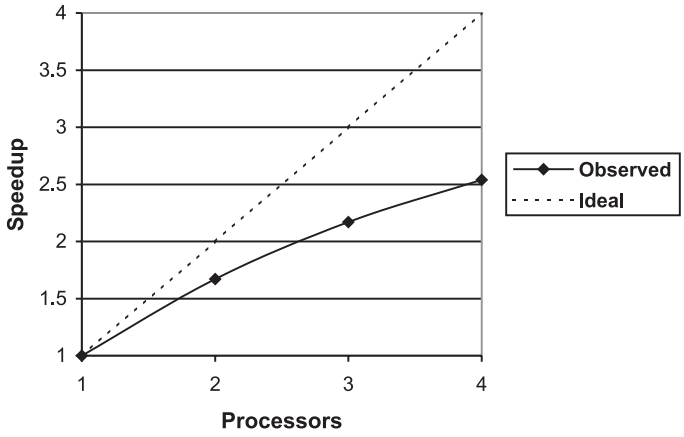

Fig. 4. Speedup curve for simulation of DMPC with MASPHYC.

system requires 1.87 s per time step and shows a performance of close to 300 Mflop/s. The scaling on up to four nodes is reasonable, and is limited by the fact that only the two-body and Coulomb interactions have been parallelised.

3. Semiempirical codes: MOPAC2000

Traditionally, semiempirical methods have been used for calculations of chemical reactions involving relatively large organic molecules, where accurate ab initio computations cannot be performed in a reasonable amount of time [23]. Comparison of computational times between various quantum chemical programs shows [24] that semiempirical computations are faster by at least three orders of magnitude than typical ab initio or density functional calculations.

MOPAC is a popular general-purpose semiempirical molecular orbital program designed for the study of molecular and solid-state structures and reactions, with MOPAC2000 being the latest release of the package. The central part of the program is the solution of self-consistent-field (SCF) equations, generated using a number of widely used semiempirical Hamiltonians. Solution of the SCF equations produces a density matrix, which can be used to compute the heat of formation. Gradients of the energy with respect to nuclear displacements are available and may be used to find stationary points (minima and transition states) on the molecular potential energy hypersurface. A wide range of molecular properties can be computed, such as solvation energies (COSMO [25], MST [26]), electrostatic potentials and point charges (ESP [27], PMEP [28]) and thermodynamic properties (zero-point vibrational energy, enthalpy, entropy, etc.). The SCF wavefunction may be improved by using the method of configuration interaction.

MOPAC2000 uses two quite different approaches to solve the SCF equations. The 'conventional' approach (see Fig. 5) uses dense matrix operations in computational

steps such as the transformation of the Fock matrix from the atomic orbital (AO) to the molecular orbital (MO) basis and formation of the density matrix. The computational scaling is of order N^3 and the memory scaling is of order N^2, where N is some measure of the size of the molecular system. This unfavourable scaling limits calculations to systems of less than about a thousand atoms. MOPAC2000 also includes a new computational scheme (Fig. 5) based on the use of localised molecular orbitals (LMOs) [29] which effectively reduces the computational effort to linear ($O(N)$) scaling. As a result, calculations on systems containing several thousands of atoms, such as proteins, have become practical.

3.1. Parallelisation of the self-consistent-field procedure

Parallel implementation of the ab initio direct SCF procedure has been studied widely by a number of research groups (for a review, see [30]). In ab initio programs, the SCF procedure is dominated by evaluation of blocks of two-electron integrals and this provides a convenient mechanism for a task-driven parallelisation of the code. The situation with semiempirical quantum chemistry programs is very different because integral generation scales (at most) as $O(N^2)$ and accounts for only a relatively minor part of the total computational time.

The two iterative SCF procedures available in MOPAC2000 are shown in Fig. 5. In conventional calculations, the computational time is spread mainly over tasks such as AO Fock matrix formation, eigensolution, pseudo-diagonalisation [31]

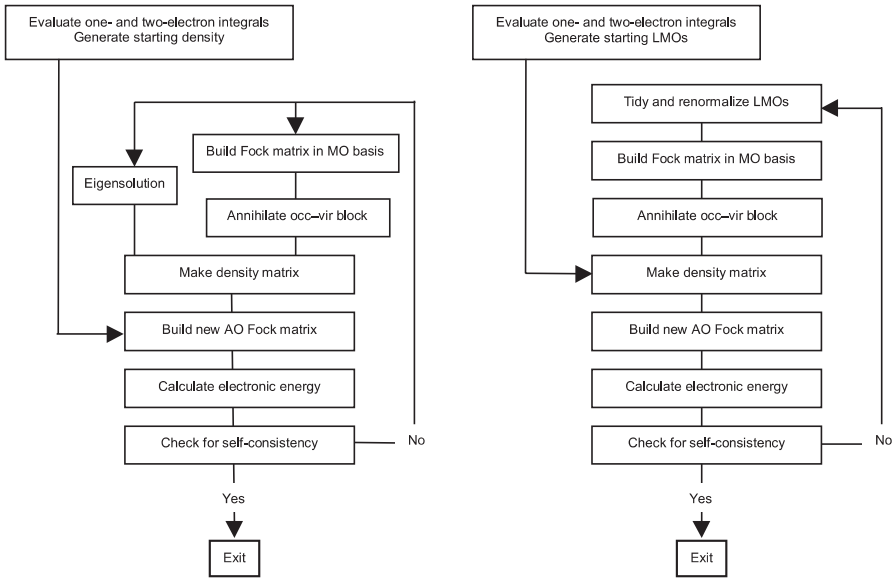

Fig. 5. Schemes for solution of the SCF equations in MOPAC2000. Conventional approach on left, linear-scaling approach on right.

(comprising a transformation of the Fock matrix to the molecular orbital (MO) basis followed by annihilation of the occupied-virtual block) and density matrix formation. During geometry optimisation the eigensolution step is performed very rarely, and the time becomes dominated by the remaining three steps.

The results obtained from a preliminary investigation into parallelisation of the conventional SCF procedure on the VPP revealed some benefit from using a small number of processors (probably four at most). The speedup achieved was approximately 2.1 on four processors for a medium-sized molecule (167 atoms). Similar results were noted previously [32]. The main bottleneck to scalability is the communication requirements for the parallel code. This is not surprising, because each step of the SCF procedure requires a relatively small amount of CPU time and communications times quickly become dominant. In addition, the AO Fock matrix build is relatively well vectorised and transformation of the Fock matrix and density formation include matrix multiplications that are well suited for the vector unit of the VPP, exacerbating the unfavourable communication-to-computation ratio. For larger systems somewhat better scaling will be obtained, but for such systems the new linear-scaling code is more appropriate.

The linear-scaling procedure (see Fig. 5) uses essentially the same steps as the traditional algorithm but the diagonalisation of the Fock matrix is done using pseudo-diagonalisation only. The method uses sparse matrices, and logical operations (e.g. determining which interactions to include) and manipulation of indirect addresses dominate time-consuming tasks. Inner loops are typically over the atomic orbitals on a particular atom, and so the vector length is usually just four or one. These effects make the communication-to-computation ratio somewhat more favourable than for the conventional approach. Accordingly, we have implemented parallelisation of the SCF procedure in MOPAC2000 for the linear-scaling code only.

MOPAC2000 is designed to be run on a wide range of platforms, from personal computers to parallel HPC platforms, and in the interests of code maintainability the parallelisation changes must be kept as localised as possible. To this end we have parallelised the code using MPI and adopted a replicated-data approach. An exception to this is in the computation and manipulation of two-electron integrals, where each processor holds only the integrals evaluated on that processor. (MOPAC2000 also has a direct SCF option, where integrals are evaluated as needed rather than being evaluated once and stored in memory.)

Fig. 6 shows the breakdown of wall time on a single PE of the VPP for a calculation on a typical small protein (barnase, a 109-residue ribonuclease from *Bacillus amyloliquefaciens*). Clearly, effective parallelisation of the five tasks (MO Fock matrix formation, annihilation, density formation, AO Fock matrix formation and integral evaluation) should lead to a code that scales well on a modest number of processors.

The replicated-data approach makes implementation of parallelisation changes straightforward in most subroutines. Tasks such as integral formation, AO Fock matrix construction and density matrix formation have outer DO loops which provide a natural task-driven decomposition, with good load balance for typical

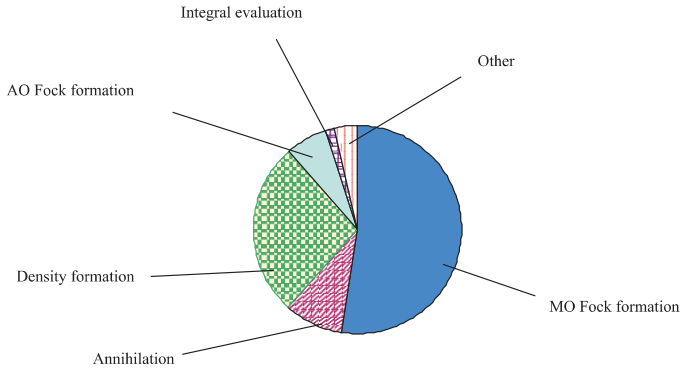

Fig. 6. Breakdown of single-processor CPU times for a linear-scaling SCF calculation on barnase.

molecular systems. Communication in these cases is simply a global reduction to sum and distribute the final results at the end of the subroutine. The MO Fock matrix construction can also be parallelised straightforwardly, with elements collected together on each processor by using MPI_ALLGATHERV calls. The pseudo-diagonalisation is somewhat more complex. Here, the occupied orbitals are divided into blocks and these must be passed in a cyclic fashion amongst the processors during the pseudo-diagonalisation procedure. Both point-to-point and global communications are required. Further details of the parallelisation of the linear-scaling SCF procedure in MOPAC2000 may be found in Ref. [33].

Fig. 7 shows speedups obtained for several different proteins. The times typically reduce by a factor of three on four processors. The major overhead leading to this modest scaling is the need to perform global reduction operations on very large arrays at various places in the code.

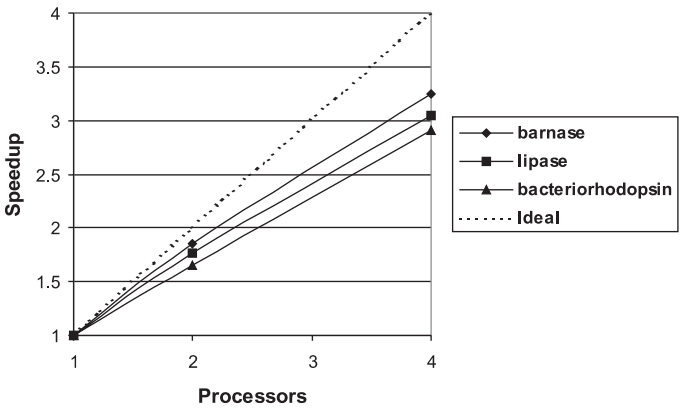

Fig. 7. Parallel speedup for linear-scaling SCF energy calculations on three proteins.

These new linear-scaling methods open up enormous opportunities for the computational investigation of important problems in biochemistry, such as the mechanisms of enzyme-catalysed reactions. Having the ability to run time-critical applications in parallel mode on machines such as the VPP (with large local memory on each node) will be crucial in furthering our understanding of such reaction mechanisms.

3.2. Gradient evaluation

It has previously been shown that parallelisation of the gradient evaluation for semiempirical methods is straightforward and of benefit on scalar machines [32]. On the VPP with powerful vector processors and for medium sized systems the situation is less clear. This is particularly true when using our new gradient evaluation algorithm [34], which through a combination of better vectorisation and reduced operation count is an order of magnitude faster than the original code on the VPP. Thus currently in MOPAC2000, gradients are not parallelised.

3.3. Parallel frequency calculations

Vibrational frequency calculation in MOPAC2000 is done using a numerical finite-difference approach. For each degree of freedom X, the energy and gradient are evaluated at the points $X + \Delta$ and $X - \Delta$, where Δ has some small value. This method allows parallelisation using a task-driven replicated-data approach with geometric displacements being statically assigned to different processors. Communication requirements are limited to a global reduction operation to sum the partial force constant matrices produced on each processor, and the diagonalisation to produce vibrational frequencies and normal modes is performed sequentially on each node. As can be seen from Fig. 8, such an approach is very effective and allows close to ideal speedup.

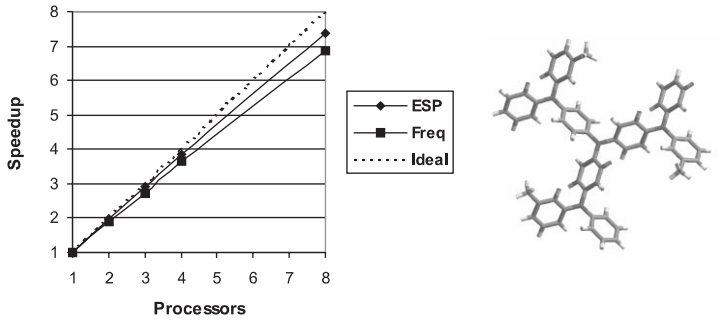

Fig. 8. Parallel speedup for evaluation of vibrational frequencies and electrostatic-potential-derived atomic charges for the 'star-burst amine' molecule shown on right.

It should be noted that calculations for closed-shell molecules are much faster than calculations on an equivalent open-shell system. Traditionally, open-shell calculations using semiempirical methods have used the so-called 'half-electron' approach. In this approach, the expression for the total energy contains a term depending on the molecular wavefunction. In the numerical finite-difference calculation, these functions need to be redefined at each displaced co-ordinate (i.e. at each $X \pm \Delta$), which in turn requires the self-consistent process to be repeated. In comparison, the energy expression for a closed-shell molecule contains only terms depending on the density matrix, thus making the recalculation of the molecular orbitals at each displaced geometry unnecessary. This consideration makes communication time less important for open-shell calculations compared with closed-shell cases.

3.4. Parallel electrostatic potential calculations

Electrostatic potential (ESP) computation is a commonly used feature in MOPAC. It can be used, for example, to assign point charges to atomic centres. This is done by computing the molecular electrostatic potential at many points on a set of molecular surfaces. These potential data are then used for a least-squares fit of point charges [27] which requires solution of a system of linear equations.

Computation of the electrostatic potential is parallelised in MOPAC2000 by a static distribution of surface points to processors. Once again a replicated-data approach is used and the only communication is a global reduction to sum arrays once all points have been determined. The solution of the system of linear equations is done sequentially on all processors. Scaling is excellent, as can be seen from Fig. 8.

In conclusion, MOPAC2000 contains parallel functionality for linear-scaling energy evaluation and conventional vibrational frequency and electrostatic potential computation. Good scaling, especially for frequency and ESP calculation, means that the parallel code offers significant advantages in reducing the time for solution for time-critical applications.

4. Ab initio quantum chemical codes

4.1. Gaussian® 98

The *Gaussian*® program suite [8] is arguably the most widely used computational chemistry package. The current release, *Gaussian® 98*, appeared in the latter half of 1998 and represents a significant enhancement over the previous release (*Gaussian® 94*). The new program includes a wide range of functionality, from simple molecular mechanics through semiempirical quantum chemical methods to advanced ab initio techniques (both density-functional-theory (DFT) and Hartree–Fock (HF) based). For most methods, analytic first and second derivatives of the energy with respect to nuclear displacements are available, and these can be used with a variety of algorithms for exploring potential-energy surfaces. The new version of *Gaussian*® also

includes the so-called hybrid methods that use different levels of theory for different parts of the system, and linear-scaling semiempirical and DFT methods.

Running a *Gaussian*® calculation involves running a series of separate executables or 'links'. In general, which links are run is determined automatically based on the type of calculation to be performed. Before terminating, and if required, each link initiates the next link in the series. Communication between links is performed by storing data to and retrieving data from disk files.

Gaussian® *98* is very large with over 700 000 lines of code producing more than 70 links. Not surprisingly, this represents a significant investment of manpower over many years. The latest code supports two parallel paradigms, a shared-memory parallel model and a distributed-memory model. On shared-memory machines, coarse-grained parallelism is implemented using the standard UNIX *fork* facility to generate multiple parallel processes that communicate by using shared memory segments. On distributed-memory machines a very similar procedure is followed, but using the Linda *eval* function to spawn the multiple tasks and data tuples to communicate between them. We note that the shared-memory parallel code also contains some fine-grained parallelism implemented at the loop level and may also exploit multi-threaded vendor-supplied matrix multiplication libraries. The Fujitsu machines discussed here are, however, all distributed-memory machines and so we will only consider the Linda parallel version of *Gaussian*®.

The initial Linda parallel *Gaussian*® implementation was on a network of workstations and is described in Ref. [35]. Essentially, only a subset of the many links have been parallelised, typically those that consume the most time. For the links that are parallel, the parallelism is associated with the generation or processing or different blocks of integrals on different processors. Thus, we are considering direct or semi-direct calculations. For example, in a direct self-consistent-field (SCF) computation the Fock matrix formation is parallel, but diagonalisation and other associated operations are sequential. In a direct second-order Møller–Plesset (MP2) energy calculation, each processor forms a subset of the transformed integrals. In all cases a replicated data approach is used.

The Linda parallel version of *Gaussian*® is, by design, only intended for large calculations which are dominated by integral generation/processing. To demonstrate the performance of the code, we consider three different types of test calculations: HF and DFT gradients and frequencies, and MP2 energy calculations. Details of the benchmarks used are given in Table 3 together with the CPU time taken to run each job on one processor of the VPP300.

The scalability of *Gaussian*® *98* Release A.6 on one to four processors of the VPP is shown in Fig. 9. Speedup is calculated as the CPU time measured on one processor divided by the wall clock time obtained on multiple processors. The results are calculated based on the total job times, not just the times recorded for the parallel links. The results show that for these sorts of calculations the user can typically expect a speedup of between 3 and 4 when using four processors.

It is pertinent to note here that Sosa et al. [36] recently reported some impressive results for a version of Linda parallel *Gaussian* running on various Cray T3E computers. In comparison to the VPP, their results show much better speedups,

Table 3
Details of *Gaussian® 98* benchmarks

Name	Theory	Calculation	Atoms/symmetry	Basis/functions	CPU time (s) 1PE VPP300
Star-burst amine	HF	Gradient	109/C_1	3-21G**/789	4429
	BLYP				9453
α-pinene	HF	Frequency	26/C_1	6-31G**/182	8495
	BLYP				11271
18-crown-6-ether	MP2	Energy	42/C_1	6-31G**/390	9159

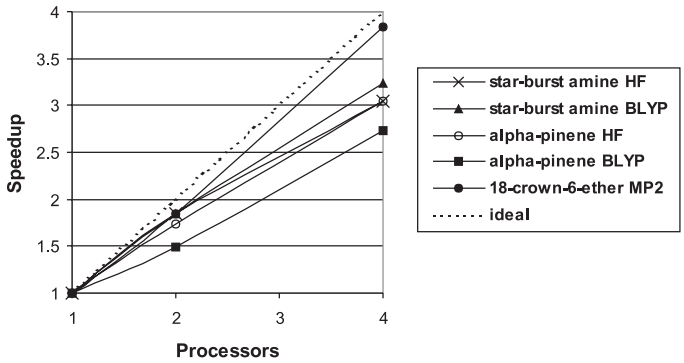

Fig. 9. Parallel performance of *Gaussian® 98* on the VPP for benchmarks shown in Table 3.

typically between 3.7 and 4 on four processors. Detailed comparison between the two machines is beyond the scope of this paper, but it is likely that this is a reflection of the much slower speed of the individual processors on the T3E. For example, the 6-311G(df, p) α-pinene Hartree–Fock energy calculation reported by Sosa et al. in Table 2 of Ref. [36] takes 7438 s on one processor of the 600 MHz T3E, compared with just 929 s on one processor of the VPP300.

4.2. *DMol*^{3}

DMol^3 [9–12] is a density functional theory (DFT) code with the ability to handle either molecular clusters or periodic systems. DFT [37–40] has gained popularity amongst computational chemists recently. This is because for many situations it provides molecular structures and energies at accuracy comparable to more expensive electronic structure methods (such as MP2) but at a significantly lower computational cost. DFT has the advantage over semiempirical methods in that it can be applied to a wide range of compounds, including metal clusters, biological compounds, organometallics and organic compounds.

The DMol^3 program calculates variational self-consistent solutions to the DFT equations using a basis set of numerical atomic orbitals. This is a unique feature that

leads to accurate electrostatic moments and polarisabilities. The use of numerical atomic orbitals means that a molecule can be dissociated exactly to its constituent atoms and that basis set superposition effects are minimised.

The excellent computational performance of DMol^3 is achieved by projecting the electron density into multipolar components on each atomic centre followed by calculation of the Coulomb potential via Poisson's equation. Thus calculation of the molecular potential, in principle a costly, N^3 step, is replaced by fast evaluation of the potential on each centre, an effort which scales as N.

The DFT matrix elements are calculated by using a sophisticated numerical integration algorithm which scales as N^3 (or even as N if sparsity is taken into account). This is the most computationally intensive part of DMol^3 and is very well optimised and vectorised. The algorithm used in DMol^3 allows for very efficient parallelisation of the numerical integration procedure [41]. Each processor performs the numerical integration for a batch of grid points, with the overlap and Hamiltonian matrices replicated across all processors. After the task is completed, the contributions from various processors are summed up and the final DFT equations are solved. All parallelisation is achieved by using message passing and MPI.

DMol^3 is well parallelised on the Fujitsu VPP in the self-consistent-field and gradient sections of the code [42]. The speedup observed for these tasks is more than 3.7 on four processors. Solution of the eigenvalue equations is done in sequential mode but amounts to only a few per cent of the total execution time for large molecules such as the aluminium isopropoxide tetramer discussed below. (This will of course limit the asymptotic speedup to a factor of 30 or so as a consequence of Amdahl's law.) The start-up calculations, such as generation of the atomic functions and numerical grid, are also done sequentially. Therefore, the total speed-up in practice is typically about 3.3 for four processors of the Fujitsu VPP300.

In Fig. 10, we present speedup data for an organometallic solid of 107 atoms in the unit cell with 500 electrons and 1129 orbitals [42]. This benchmark involved one

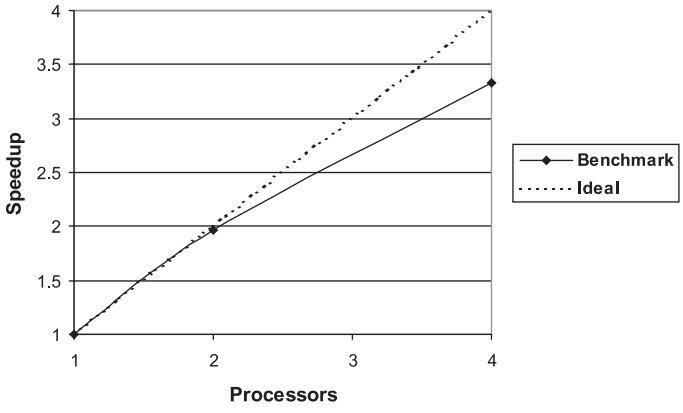

Fig. 10. Speedup data for a bis(phosphite) ruthenium carbonyl solid.

SCF run, with 12 iterations, followed by gradient calculation. The calculation was done using a local spin density Hamiltonian [43].

For larger molecular systems and longer runs involving several self-consistent-field and gradient calculations, the contributions from the start-up work become negligibly small. After parts of the program were modified to improve the vector performance on the Fujitsu machine, calculations such as those performed in our study of aluminium alkoxides run at a sustained rate of about 5.5 Gflop/s [44] on four processors of the Fujitsu VPP300.

An example of the power of DMol^3 on the vector–parallel machine has emerged from a recent study of aluminium alkoxides, a class of industrially relevant catalysts used in a number of chemical reactions. Recently some of us presented brief reports of calculations on both the structure of aluminium hydroxide aggregates and aluminium isopropoxide tetramer [44] and the transition state of the aluminium hydroxide catalysed ring-opening reaction of ε-caprolactone [45].

In this work on the structure of aluminium hydroxide aggregates and aluminium isopropoxide tetramer, we used a local spin density Hamiltonian (LSD) [43] for the hydroxides and isopropoxides, and a non-local spin density Hamiltonian (NLSD) [46–48] for the hydroxides. This has shown that the use of a 'model' compound such as the aluminium hydroxide tetramer (see Fig. 11) is a poor choice if one wishes to describe accurately the properties of a 'real' compound such as the aluminium isopropoxide tetramer. For example, further optimisation work has shown that a discrepancy of 6% in one particular angle, the O(terminal)–Al(4-coord)–O(terminal) angle, in the hydroxide tetramer when compared with a suitable experimental reference value is reduced to just 2% when the full isopropoxide tetramer is used (see Fig. 11).

This shows that the steric interactions among the isopropoxide groups play an important role in determining the structure of the Al_4O_{12} framework. Viewed

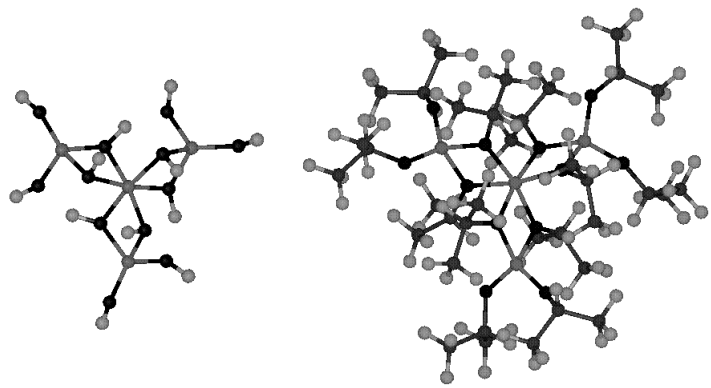

Fig. 11. The aluminium hydroxide and the aluminium isopropoxide tetramer as obtained from DMol LSD [54] calculations.

another way, for the purpose of obtaining an optimised structure, the hydroxide groups are a poor approximation to the isopropoxide groups. This points out the potential danger of the practice common amongst computational chemists where bulky substituents are replaced with simpler ones in order to save computational time. Calculations on realistic systems such as the isopropoxide tetramer are really only feasible on fast parallel machines such as the Fujitsu VPP series. Hence parallel computers will continue to play an important role in computational chemistry for years to come!

4.3. MOLPRO

As an example of the use of the Global Arrays in computational chemistry, we present results for the molecular quantum chemistry code MOLPRO [13]. MOLPRO is a general ab initio code capable of performing calculations based on Hartree–Fock, density functional theory and multi-configurational SCF starting points. However, its greatest strength is in its ability to perform highly accurate (and very costly) multireference configuration interaction (MRCI) calculations to obtain definitive information on reaction energies and barriers and to map out potential energy surfaces for use in dynamical studies.

MRCI calculations are very demanding in terms of CPU performance, memory requirements and disk storage. Within MOLPRO, such calculations utilise the internally contracted MRCI (ICMRCI) approach [49] to improve performance dramatically over conventional approaches at very little cost in terms of reduced accuracy. The inner part of the algorithm is formulated as a sequence of vector–vector, matrix–vector or matrix–matrix operations and so should be well suited to the vector unit of the VPP.

Very recently, Dobbyn, Knowles and Harrison [50] have reported a distributed-data parallel implementation of the ICMRCI method in MOLPRO. Their stated aim was to produce a portable and scalable parallel program but, at the same time, to minimise the number of differences between the sequential and parallel codes to aid in maintainability. To ensure portability, the parallelisation was based on use of the Global Arrays [4].

Details of the parallelisation procedure may be found in Ref. [50]. In essence, an existing disk-based algorithm designed for small-memory machines was modified to produce the parallel code. As the initial step, input/output to scratch files present in the sequential code was replaced with reading and writing to global memory. This required very localised changes in the subroutines responsible for implementing the I/O. The work involved in evaluating the interactions arising in the ICMRCI procedure was partitioned over processors by using static load balancing or, if this led to unacceptably poor balance between parallel tasks, a dynamic load balancing scheme based on the TCGMSG *nxtval* shared counter [51]. The resulting code contains a mixture of coarse- and fine-grained parallel tasks and a mixture of fully distributed and replicated data structures.

The performance of this code on the Fujitsu VPP is demonstrated by the results shown in Fig. 12. Speedup curves are presented for a single iteration of the ICMRCI

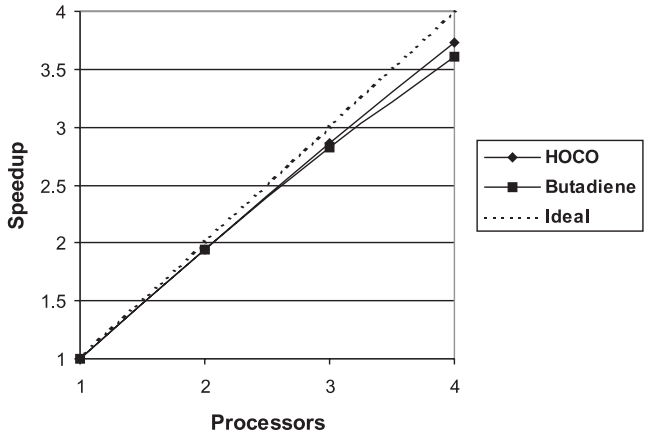

Fig. 12. Speedup curves for one iteration of ICMRCI program in MOLPRO.

program for two test cases: the butadiene benchmark 'D' from Ref. [50] (122 basis functions, 142 reference configurations, 47 million uncontracted configuration state functions) and the larger HOCO benchmark 'H' from Ref. [50] (161 basis functions, 3048 reference configurations, 1.24 billion uncontracted configuration state functions).

Scaling is extremely good, especially for the larger HOCO case. Clearly, the pragmatic approach adopted in Ref. [50] of developing a scalable parallel code by making a minimum number of changes to the existing sequential code has been very successful.

4.4. CASTEP

The density functional theory (DFT) method discussed earlier in the context of DMol^3 can also be formulated in terms of the total-energy pseudopotential plane-wave (PP–PW) approach (for a recent review, see [52]). The DFT method solves quantum mechanical equations for the electronic structure of systems containing arbitrary arrangements of atoms. This gives the ground-state energy and charge density of the system and enables physical properties such as lattice constants, elastic constants, geometric structure and binding energies to be determined. The PP–PW DFT method has become an essential part of computational materials science and is becoming indispensable in the search and design of novel materials and processes.

One very successful commercial software package employing the total-energy PP–PW approach is CASTEP [14], a code arising from a suite of programs developed in the Theory of Condensed Matter Group at the University of Cambridge. CASTEP has been used to simulate the properties of solids, interfaces and surfaces for a wide range of materials including ceramics, semiconductors and metals. It can be used to explore the properties of crystalline materials, properties of surfaces and surface reconstructions, chemical reactions at surfaces, band structures and densities of

states, optical properties of crystals, properties of point defects, grain boundaries and dislocations, and so on.

Two recent developments, the use of ultrasoft pseudopotentials [53] and an efficient density-mixing scheme [54] to solve the DFT equations, first appeared in the VASP code [54,55] and have now been incorporated into CASTEP. These advances offer dramatic performance improvements compared with earlier approaches and promise to extend the applicability of PP–PW calculations to an even broader range of systems.

The central quantities in a total energy pseudopotential calculation are the Kohn–Sham orbitals, which can be represented in a plane-wave basis [56] as

$$\psi_i(r) = \psi_{n,k}(r) = \frac{1}{\sqrt{\Omega}} \sum_G C_G^{n,k} e^{i(k+G)r},$$

n is the band index, k the k-point used to sample the Brillouin zone, Ω the unit cell volume and $\{G\}$ the reciprocal lattice vectors. This represents a Fourier transformation from a momentum-space representation to a real-space representation of the orbitals. The use of fast Fourier transformation allows one to operate with the Hamiltonian on the electronic wavefunctions very efficiently. The computational requirements of the PP–PW method are dominated by fast Fourier transforms together with complex vector and matrix operations.

The total wavefunction can be considered as a complex array $\Psi(M, N_b, N_k)$, where M represents the number of plane waves, N_b denotes the number of electronic bands and N_k represents the number of k-points. This suggests several data-driven approaches to parallelising the calculation [57]:

A. Division by k-point, where the relatively independent calculations at each k-point are done on separate processors. This has the disadvantage that the number of k-points is often very small. On the other hand, this approach is the most promising one for relatively small unit cells of metallic systems where the number of k-points should be quite large for accurate calculations.

B. Division by band, where the computations performed with respect to different electronic bands (at the same k-point) are assigned to different nodes. The major drawback with this method is that the FFT mesh must be replicated across all nodes, leading to a large memory requirement.

C. Divide real and reciprocal space, where the G-vectors are spread across the nodes. This approach requires much less memory, but has the disadvantage of further communication during the FFT.

The first two of these strategies are useful on modestly parallel machines, but on massively parallel computers only strategy C is effective. The widely used plane-wave code CETEP [56,57] uses such a strategy.

Linking together two of the above-mentioned parallelisation strategies can be beneficial in minimising the need for global communications involving all participating processors. For example, the code FINGER [58] employs a combination of k-point and G-vector parallelisation. Similarly, Molecular Simulations, and the Fujitsu European Centre for Information Technology have been collaborating to parallelise CASTEP using a combination of k-point and G-vector distributions.

We first divide processors into blocks and then assign a set of k-points to each block. The computation in each block is to a great extent independent of computation in other blocks, and in principle the only communication requirement is a global summation of the total electron density over the blocks. Inside each block, G-vector parallelisation is used.

The proposed parallel strategy allows in fact three different types of distribution: k-point distribution, G-vector distribution, or combined G-vector and k-point distribution. The number of k-points and the number of processors to be used dictate the optimum choice of distribution for the specific system. Logic is built into the code so that this choice of distribution is performed automatically. A minor restriction is that the k-point distribution is available only for the all-bands minimisation scheme [59] used in CASTEP. It is not available for the band-by-band scheme [60] because the existing band-by-band algorithm is not consistent with a k-point distribution strategy.

All parallelisation is carried out using message passing and MPI.

In the G-vector distribution approach, the three-dimensional Fourier transform is replaced by a sequence of one-dimensional transforms. The first set of one-dimensional Fourier transforms is performed along the x-direction. This is followed by a global exchange of data between processors so that the data are now arranged by columns along the y-direction. The one-dimensional transforms along the y-direction are then carried out. A second global exchange of data is then performed so that the data are finally arranged by columns along the z-direction. The final set of one-dimensional transforms along the z-direction is then carried out. To transform wavefunctions to real space, a weighted Fourier transform was developed [57]. It is based on the idea of exchanging just non-zero elements occupying a sphere in reciprocal space rather than the whole cube.

In order to test the performance of CASTEP, we have performed calculations for a system of 64 silicon atoms. The simulation is for a single energy calculation using the all-bands scheme, a $48 \times 48 \times 48$ FFT grid and four k-points. Scaling on up to four nodes of the VPP is excellent, as shown in Fig. 13.

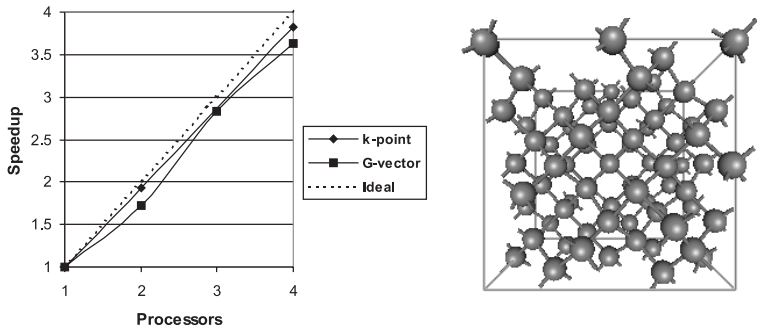

Fig. 13. Observed scaling for silicon test system (shown on right).

In this case, the *k*-point distribution is somewhat more efficient than the *G*-vector approach. The two strategies have similar communications requirements for global summation operations, but the *G*-vector approach includes an additional overhead arising from the global exchange of data during the fast Fourier transform steps.

CASTEP is an ideal code for vector–parallel computers. The single-node vector performance is good, with the silicon test case showing a performance of 650 Mflop/s on the Fujitsu VPP. Other cases also show single-node performance in the range of 400–700 Mflop/s. With its good single-node vector performance and excellent scaling on multiprocessor systems, CASTEP is arguably one of the best application codes available for the Fujitsu VPP series.

5. Conclusions

Chemists have always been quick to realise the potential of new computer architectures to assist in the interpretation of experiment and for predicting the properties of new molecules. As a consequence of this, many of the programs that are now in common use can trace their history back over several decades and have been adapted and re-adapted as each new architecture came along. This is both an advantage and disadvantage, enabling for example the developer of new methods to draw on the wealth of code that already exists, but also perhaps hindering these codes from exploiting efficiently new hardware as it became available.

Parallel computing is a good example of this. Many codes have had parallel functionality added as an extension to existing sequential algorithms. In general, this means task-driven replicated-data approaches and parallelisation of only a subset of the time-consuming tasks. As a result, such codes favour modestly parallel, large memory, high-performance machines such as the VPP.

Performance is in general limited by remaining sequential work and by communications requirements, and many codes can only achieve a speedup of 3 or so on four nodes. Even so, some impressive performance levels can be achieved, such as the 5.5 Gflop/s observed for the code DMol^3 on four processors of a VPP300.

The materials code CASTEP is a good example of an application that has been explicitly optimised for the Fujitsu VPP. Single-node vector performance is good, and the distributed-memory parallelisation exploits the aggregate memory of the VPP very efficiently and with excellent parallel scaling.

Fujitsu has recently announced the next generation VPP system, the VPP5000. As well as providing a four-fold increase in the peak performance, this new range of vector–parallel system promises to address many of the shortcomings of the current series, in particular the performance of the scalar processor. If this proves to be the case it is likely to be well suited to many computational chemistry applications.

Acknowledgements

We gratefully acknowledge valuable discussions on MOPAC2000 with J.J.P. Stewart, on MOLPRO with P. Knowles, and on MASPHYC with N. Tahara,

M. Takeuchi and T. Mitsui. PK and HL would like to thank Jan Andzelm and Noppawan Tanpipat for their help in some of the computations presented here and for useful discussions. The Australian National University acknowledges the support of Fujitsu Japan.

References

[1] R.H. Nobes, A.P. Rendell, J. Nieplocha, Computational chemistry on Fujitsu vector–parallel processors: Hardware and programming environment, Parallel Computing 26 (2000), this issue.
[2] See http://www.mpi-forum.org/.
[3] See http://www.cs.yale.edu/HTML/YALE/CS/Linda/linda.html.
[4] J. Nieplocha, R.J. Harrison, R.J. Littlefield, Global Arrays: A nonuniform memory access programming model for high-performance computers, J. Supercomput. 10 (1996) 197–220.
[5] S.J. Weiner, P.A. Kollman, D.A. Case, U.S. Singh, C. Ghio, G. Alagona, S.Jr. Profeta, P. Weiner, A force field for molecular mechanics simulations of nucleic acids and proteins, J. Chem. Soc. 104 (1984) 765–784.
[6] M. Takeuchi, T. Ishigai, K. Ishibashi, Development and application of MASPHYC computational material design system Application package in HPC, Fujitsu Sci. Technol. J. 33 (1997) 52–61.
[7] MOPAC2000, © Fujitsu Limited, 1999.
[8] M.J. Frisch, G.W. Trucks, H.B. Schlegel, G.E. Scuseria, M.A. Robb, J.R. Cheeseman, V.G. Zakrzewski, J.A. Montgomery Jr., R.E. Stratmann, J.C. Burant, S. Dapprich, J.M. Millam, A.D. Daniels, K.N. Kudin, M.C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G.A. Petersson, P.Y. Ayala, Q. Cui, K. Morokuma, D.K. Malick, A.D. Rabuck, K. Raghavachari, J.B. Foresman, J. Cioslowski, J.V. Ortiz, B.B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R.L. Martin, D.J. Fox, T. Keith, M.A. Al-Laham, C.Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P.M.W. Gill, B. Johnson, W. Chen, M.W. Wong, J.L. Andres, C. Gonzalez, M. Head-Gordon, E.S. Replogle, J.A. Pople, Gaussian, Pittsburgh, PA, 1998.
[9] DMol^3, Molecular Simulations, 1997.
[10] B. Delley, An all-electron numerical method for solving the local density functional for polyatomic molecules, J. Chem. Phys. 92 (1990) 508–517.
[11] B. Delley, Analytic energy derivatives in the numerical local-density-functional approach, J. Chem. Phys. 94 (1991) 7245–7250.
[12] B. Delley, DMol, a standard tool for density functional calculations: review and advances, in: J. Seminario, P. Politzer (Eds.), Density Functional Theory: A Tool for Chemistry, Elsevier, Amsterdam, 1995.
[13] MOLPRO is a package of ab initio programs written by H.-J. Werner, P. J. Knowles, with contributions from R.D. Amos, A. Berning, D.L. Cooper, M.J.O. Deegan, A.J. Dobbyn, F. Eckert, C. Hampel, T. Leininger, R. Lindh, A.W. Lloyd, W. Meyer, M.E. Mura, A. Nicklaß, P. Palmieri, K. Peterson, R. Pitzer, P. Pulay, G. Rauhut, M. Schütz, H. Stoll, A.J. Stone, T. Thorsteinsson.
[14] The CASTEP program is developed and distributed by Molecular Simulations.
[15] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Clarendon Press, Oxford, 1987.
[16] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1995) 1–19.
[17] R.A. Kendall, R.J. Harrison, R.J. Littlefield, M.F. Guest, High performance computing in computational chemistry: Methods and machines, in: K.B. Lipkowitz, D.B. Boyd (Eds.), Reviews in Computational Chemistry, vol. 6, VCH Publishers, New York, 1995, pp. 209–316.
[18] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, M. Karplus, CHARMM: A program for macromolecular energy, minimisation, and dynamics calculations, J. Comput. Chem. 4 (1983) 187–217.
[19] See http://wserv1.dl.ac.uk:801/TCSC/Software/DL_POLY/main.html.

[20] W.F. van Gunsteren, S.R. Billeter, A.A. Eising, P.H. Hünenberger, P. Krüger, A.E. Mark, W.R.P. Scott, I.G. Tironi, Biomolecular simulation: The GROMOS96 manual and user guide, Zurich and Groningen: vdf Hochschulverlag AG an der ETH Zurich and BIOMOS b.v., 1996.
[21] W. Smith, Molecular dynamics on distributed memory (MIMD) parallel computers, Theor. Chim. Acta 84 (1993) 385–398.
[22] B. Hess, H. Bekker, H.J.C. Berendsen, J.E.E.M. Fraaije, LINCS: A linear constraint solver for molecular simulations, J. Comput. Chem. 18 (1997) 1463–1472.
[23] C.H. Reynolds, Semiempirical MO methods: The middle ground in molecular modelling, J. Mol. Struct. Theochem 401 (1997) 267–277.
[24] W. Thiel, Perspectives on semiempirical molecular orbital theory, Adv. Chem. Phys. 93 (1996) 703–757.
[25] A. Klamt, G. Schüürmann, COSMO: A new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradient, J. Chem. Soc. Perkin Trans. 2 (1993) 799–805.
[26] M. Orozco, M. Bash, F. Luque, Development of optimized MST/SCRF methods for semiempirical calculations. The MNDO and PM3 hamiltonians, J. Comput. Chem. 16 (1995) 629–635.
[27] B.H. Besler, K.M. Merz Jr., P.A. Kollman, Atomic charges derived from semiempirical methods, J. Comput. Chem. 11 (1990) 431–439.
[28] G.P. Ford, B. Wang, New approach to the rapid semiempirical calculation of molecular electrostatic potential based on the AM1 wave function: Comparison with ab initio 6-31G* results, J. Comput. Chem. 14 (1993) 1101–1111.
[29] J.J.P. Stewart, Application of localized molecular orbitals to the solution of semiempirical self-consistent field equations, Int. J. Quant. Chem. 58 (1996) 133–146.
[30] R.J. Harrison, R. Shepard, Ab initio molecular electronic structure on parallel computers, Ann. Rev. Phys. Chem. 45 (1994) 623–658.
[31] J.J.P. Stewart, P. Császár, P. Pulay, Fast semiempirical calculations, J. Comput. Chem. 3 (1982) 227–228.
[32] K.K. Baldridge, Parallel implementation of semiempirical quantum methods for Intel platforms, J. Math. Chem. 19 (1996) 87–97.
[33] R.H. Nobes, H.A. Früchtl, E.V. Akhmatskaya, Parallel MOZYME, in: K.Y. Lam, B.C. Khoo, K. Kumar (Eds.), Proceedings of HPC Asia '98, vol. 1, Institute of High Performance Computing, Singapore, 1998 pp. 753–758.
[34] A.A. Bliznyuk, A.P. Rendell, Faster gradients in semiempirical methods, J. Comput. Chem. 20 (1999) 629–635.
[35] D.P. Turner, G.W. Trucks, M.J. Frisch, In Parallel computing in computational chemistry, ACS Series 592 (1995) 62–74.
[36] C.P. Sosa, J. Ochterski, J. Carpenter, M.J. Frisch, Ab initio quantum chemistry on the Cray T3E massively parallel supercomputer: II, J. Comput. Chem. 19 (1998) 1053–1063.
[37] R.G. Parr, W. Yang, Density Functional Theory of Atoms and Molecules, Oxford University Press, New York, 1989.
[38] T. Ziegler, Approximate density functional theory as a practical tool in molecular energetics and dynamics, Chem. Rev. 91 (1991) 651–667.
[39] J. Labanowski, J. Andzelm (Eds.), Density Functional Methods in Chemistry, Springer, New York, 1991.
[40] B. Laird, R. Ross, T. Ziegler (Eds.), Chemical Applications of Density Functional Theory, ACS Symposium 629, American Chemical Society, Washington, 1996.
[41] Y.S. Li, M.C. Wrinn, J.M. Newsam, M.P. Sears, Parallel implementation of a mesh-based density functional electronic structure code, J. Comput. Chem. 16 (1995) 226–234.
[42] H. Lung, J. Andzelm, personal communication.
[43] S.H. Vosko, L. Wilk, M. Nusair, Accurate spin-dependent electron liquid correlation energies for local spin density calculations: A critical analysis, Can. J. Phys. 58 (1980) 1200–1211.
[44] P.W.-C. Kung, J.W. Andzelm, H. Lung, A density functional study of the structure of initiators in the catalytic ring-opening polymerization of ε-caprolactone, in: K.Y. Lam, B.C. Khoo, K. Kumar (Eds.),

Proceedings of HPC Asia '98, vol. 1, Institute of High Performance Computing, Singapore, 1998, pp. 303–311.
[45] N. Tanpipat, P.W.-C. Kung, J.W. Andzelm, A DFT investigation – the catalytic ring-opening polymerization of ε-caprolactone, in: Proceedings of ACS Division of PMSE, 78, American Chemical Society, 1998, pp. 285–286.
[46] A.D. Becke, Phys. Rev. A 38 (1988) 3098–3100.
[47] A.D. Becke, Density functional theories in quantum chemistry, in: D.R. Salahub, M.C. Zerner (Eds.), The Challenge of d and f Electrons, ACS, Washington, DC, 1989.
[48] Y. Wang, J.P. Perdew, Spin scaling of the electron-gas correlation energy in the high-density limit, Phys. Rev. B 43 (1991) 8911–8916.
[49] H.-J. Werner, P.J. Knowles, An efficient internally contracted multiconfiguration reference CI method, J. Chem. Phys. 89 (1988) 5803–5814.
[50] A.J. Dobbyn, P.J. Knowles, R.J. Harrison, Parallel internally contracted multireference configuration interaction, J. Comput. Chem. 19 (1998) 1215–1228.
[51] R.J. Harrison, Portable tools and applications for parallel computers, Int. J. Quantum Chem. 40 (1991) 847–863.
[52] V. Milman, B. Winkler, J.A. White, C.J. Pickard, M.C. Payne, E.V. Akhmatskaya, R.H. Nobes, Electronic structure, properties and phase stability of inorganic crystals: A pseudopotential plane-wave study, Int. J. Quantum Chem. 77 (2000) 895–910.
[53] D. Vanderbilt, Soft self-consistent pseudopotentials in a generalized eigenvalue formalism, Phys. Rev. B 41 (1990) 7892–7895.
[54] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab-initio total energy calculations using a plane-wave basis set, Phys. Rev. B 54 (1996) 11169–11186.
[55] G. Kresse, J. Furthmüller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, Comput. Mater. Sci. 6 (1996) 15–50.
[56] M.C. Payne, L.J. Clarke, I. Stich, Role of parallel architectures in periodic boundary conditions, Philos. Trans. Roy. Soc. London A 341 (1992) 211–220.
[57] L.J. Clarke, I. Stich, M.C. Payne, Large-scale ab initio total energy calculations on parallel computers, Comput. Phys. Comm. 72 (1992) 14–28.
[58] S. Pöykkö, Ab initio electronic structure methods in parallel computers, in: Proceedings of PARA'98, Lecture Notes in Computer Science, vol. 1541, Springer, Berlin, 1998, pp. 452–459.
[59] M.J. Gillan, Calculation of the vacancy formation energy in aluminium, J. Phys. 1 (1989) 689–711.
[60] M.C. Payne, M.P. Teter, D.C. Allen, T.A. Arias, J.D. Joannopolous, Iterative minimization techniques for ab initio total-energy calculations: Molecular dynamics and conjugate gradients, Rev. Mod. Phys. 64 (1992) 1045–1097.