Parallel implementation of semiempirical quantum methods for the Intel platforms

Kim K. Baldridge

San Diego Supercomputer Center, P.O. Box 85608, La Jolla, CA 92138-9784, USA

Received 27 April 1995; revised 11 October 1995

The restructuring of quantum mechanical applications for use on message-passing, distributed memory multicomputers is found to be a challenge. A key computation in these large scale quantum chemistry packages is the determination of eigenvalues and eigenvectors of real symmetric matrices. These computations arise during geometry optimization and vibrational analysis, and typically consume at least half of the total computation time. This work illustrates the parallelization of both tasks within the semiempirical quantum chemistry code, MOPAC, on Intel parallel platforms. The application of this parallel code is demonstrated on novel organic systems.

0. Introduction

Utilization of computationally derived chemical and physical properties has vastly enhanced the success of experimental ventures into the creation of designer molecules of technological and medicinal importance. Rational drug design and novel nanomolecular materials would be complete fantasies if not for the atomic scale insight provided by computational chemistry. Because of the high demand for pharmaceuticals and composite materials to display a special uniqueness of action or efficiency in response, the tightness on specific structural tolerances and hence the degree of complexity in these molecular blueprints are increasing at a rate only manageable by advanced computing methods (e.g. massive parallelization, or ultrafast vectorization). Despite the extraordinary abilities of modern hardware technology and coding methods to manipulate the raw data, the rate limiting step in harmonizing the intricacy and precision required to push forward these chemical frontiers ultimately comes down to the optimization of the complex computational methodologies on state-of-the-art hardware platforms.

There are currently three commonly employed theoretical methods for the study of the properties of molecules: *Ab Initio*; molecular mechanics; semiempirical. It has been well-established that quantum mechanical methods based on Hartree–Fock (HF) theory provide a successful and thoroughly tested framework for molec-

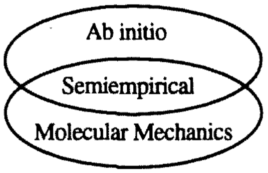


Fig. 1. Schematic showing the overlap of the three commonly employed theoretical methods.

ular calculations [1,2]. There are, however, major limitations in the size of molecular systems that can be reasonably calculated on the available hardware. Computational costs and complexity of solving the large iterative eigenvalue/eigenvector systems associated with the theoretical methods become quite demanding [3]. Even the fastest computers have limitations on the size of molecular systems that can be solved due to CPU speed, memory, and disk space requirements. At present, the upper limit is about 100 basis functions [1], which corresponds to less than 40 first row atoms at a modest level basis set, i.e. about a tetrapeptide. On the other hand, molecular mechanics and molecular dynamics techniques are extremely fast empirical methodologies which are able to handle very large molecular systems, such as entire enzymes with over 100 peptide residues. These methods sacrifice in generality and accuracy. In addition, they are not parameterized for other than ground state systems, and are unable to adequately represent geometries involved in bond-making/bond-breaking processes.

Intermediate to Hartree–Fock methods and empirical-based methods are semiempirical methods. Like *ab initio* methods, they are basically quantum mechanical in nature, the main difference being that the semiempirical methods involve more approximations based on experimental data, thus simplifying the calculations considerably. Semiempirical methods are right on the verge of becoming of routine use in polymer and biochemical applications. The major constraint, despite the numerous methodological advanced in past years [4,5], is that the size of chemical systems that can be analyzed, being largely a function of available single-processor computer power. Although this power continues to increase in magnitude, it cannot continue to improve at a rate that keeps pace with the desires and expectations of the scientific community. Parallel architectures promise to make calculations of this size more of a reality. However, only recently has it been realistic to turn towards the parallel computing environment for any of these types of calculations [6–9], primarily due to the fact that new distributed-memory algorithms that utilize the architectures of the parallel platforms must be developed.

This work focuses on the promises and concerns of applying parallel methods to semiempirical calculations for the solution of problems that are currently not

[1] Basis functions are mathematical functions which represent atomic orbitals as in descriptive organic chemistry. The number of basis functions used in a calculation of a particular molecule determines the level of accuracy of that calculation, and forms what is called a basis set.

possible with either *ab initio* or parallel *ab initio* methods [10–13], and with an accuracy greater than that achievable with the molecular mechanics and dynamics type procedures. We describe the conversion and performance evaluation of the semiempirical quantum chemistry code, MOPAC [14–16] on the Intel iPSC/860 and Paragon platforms currently housed at the San Diego Supercomputer Center.

1. Semiempirical quantum methods

The primary goal of quantum chemical codes is to solve the molecular Schrödinger equation [1,17].

$$H\Psi = E\Psi ,$$

where H is the molecular Hamiltonian operator,

$$H_{el} = -\frac{1}{2}\sum_i \nabla_i^2 - \sum_i \sum_\alpha \frac{Z_\alpha}{r_{i\alpha}} + \sum_i \sum_{j>i} \frac{1}{r_{ij}} .$$

The solution of this eigensystem provides the molecular wave function, from which a total description of the molecule, including all molecular properties such as equilibrium geometry, dipole moments, energetics, kinetics, and dynamics is obtained. The applications programs [18–20] for these theories are typically large and complex, and large real symmetric eigenproblems [12,21–23] arise in various options, notably self-consistent field (SCF) [24] computations and molecular vibration analysis.

Multiplying the Schrödinger Equation on the left by Ψ^* and integrating over all space gives an expression that, in principle, allows the calculation of the eigenvalue as

$$E = \frac{\int \Psi^* H \Psi \, d\tau}{\int \Psi^* \Psi \, d\tau} .$$

The ratio, $H\Psi^*/\Psi$, in this expression will vary with position in space, giving rise to what has been historically referred to as the local energy method. For most problems of chemical interest, this problem is not possible to solve. An approximate solution, however, which actually corresponds to the quantum state of lowest energy (the ground state), can be obtained by choosing a trial wave function, and adjusting parameters in that trial function to give a minimum energy. Thus, for an approximate wave function, Φ, with the correct boundary conditions, one can solve

$$E_{grd} = \frac{\int \Phi^* H \Phi \, d\tau}{\int \Phi^* \Phi \, d\tau}$$

such that $E_{grd} \geqslant E_{true}$; the variational principle [1]. To solve this, the linear variation method is used. In this method, a set of fixed basis functions are chosen, and the trial wave function is expressed as a linear combination of these functions (linear combination of atomic orbitals, LCAO, approach).

$$\Phi_i = \sum_n c_{ni}\chi_n$$

with the restriction c_{ni}, $\chi_n \in \Re$. In this method, the functions χ_i are not allowed to vary throughout the calculation, only the numerical coefficients, c_{ni}. The minimum energy satisfies a set of linear homogeneous equations in the n parameters. For this set of equations to have a nontrivial solution, the determinant of the coefficients must vanish:

$$\det\left[\int \Phi_m^* H' \Phi_n \, d\tau - E_j \int \Phi_m^* \Phi_n \, d\tau\right] = 0$$

or, more simply,

$$\det(H'_{mn} - S_{mn}E_j) = 0$$

with

$$H'_{mn} \equiv \int \Phi_m^* H' \Phi_n \, d\tau \,,$$

$$S_{mn} \equiv \int \Phi_m^* \Phi_n \, d\tau \,.$$

The H'_{ni} are integrals representing the interactions of electrons within orbitals, and S_{ni} is the overlap matrix. Expanded in the basis set representation then, this becomes

$$c_{ni}|F - \epsilon_j S|c_{ni} = 0 \,,$$

where Φ, the Fock matrix, represents the interaction of electrons in a 'field' of the other electrons. With ϵ_i determined as a root of the secular equation, the c_i can be found by solving the $n - 1$ equations and imposing the condition that the wave function be normalized. One often sees this written as

$$\mathrm{HC} = \mathrm{ESC} \,,$$

a generalized matrix eigenvalue equation. For each value of i, the energy ϵ_i gives an approximation to the ith state of the molecular system, and the column, C, comprises n coefficients which describe state i as a linear combination of the χ_k. The matrix eigenvalue problem is well known and well studied in science, engineering and many branches of pure and applied mathematics.

For polyatomic molecules, the LCAO method involves one or more basis functions (atomic orbitals) centered on each atom,

$$\chi_i = \sum d_i x^l y^m z^n e^{-ax^2} \,.$$

The presence of more than two atoms causes difficulties in evaluating the needed integrals. For a triatomic molecule, three-centered as well as one- and two-center integrals arise. For a molecule with four or more atoms, four-center integrals also arise, but the number of centers occurring in any one integral does not exceed four, as now illustrated. The Hamiltonian expression contains only one-electron and two-electron terms. A typical two-electron integral, as expanded in the basis set is

$$c_i^* c_j d_k^* d_l \int\int \chi_i^*(1)\chi_j(1)\chi_k^*(2)\chi_l(2)\frac{1}{r_{12}}d\nu_1 d\nu_2 .$$

If the basis functions χ_i, χ_j, χ_k, χ_l, are each centered on a different nucleus, then the above is a four-centered integral.

In *ab initio* SCF computations, with n different choices for each basis function, the matrix element computations involve the evaluation of up to $O(n^4)$ floating point operations for the evaluation of Coulomb and Exchange (interaction) integrals, whereas the solution of a single eigensystem is $O(n^3)$ (i.e. evaluation of the integrals dominate the computational effort). In semiempirical techniques, an approximate Hamiltonian is used so that the number of calculated Fock matrix elements is greatly reduced. These methods are based on the assumption that only electrons on the same atoms have significant interaction energies; all others can be ignored. This reduces the calculation of integrals to $O(n^2)$ and thus, solution of the eigensystem becomes the primary computational effort. Even still, values of n on the order of a few hundred are easily reached in even moderate-sized systems with several heavy atoms in these types of calculations. MOPAC supports four semiempirical Hamiltonians: MNDO [25], MNDO/3 [26], AM1 [15], and PM3 [27]. These are used in the electronic part of the calculation to obtain molecular structures, molecular orbitals, heats of formation, and vibrational modes. The advantages of semiempirical over *ab initio* methods are that semiempirical methods are several orders of magnitude faster, and thus calculations for larger molecular systems are possible by using one of these semiempirical Hamiltonians. The reliability of these methods in predicting accurate geometries and heats of formation has been demonstrated in many applications [28,29].

Within the semiempirical approach then, the equations to be solved take the form

$$\sum_n (F_{mn} - E_i\delta_{mn})c_{ni} = 0 ,$$

where E_i is the eigenvalue of the molecular orbital Φ_i and δ_{mn} is the Kronecker δ. The elements F_{mn} of the Fock matrix are the sum of a one-electron part H_{mn} (core Hamiltonian) and a two-electron part, and the electronic energy E_{el} is given by the expression

$$E_{el} = \tfrac{1}{2}\sum_m \sum_n P_{mn}(H_{mn} + F_{mn}) ,$$

where P_{mn} is an element of the bond order matrix (made up of the coefficients of the atomic orbitals).

The SCF computation is iterative in nature, as the Fock operator depends on its own eigenfunctions, and the Fock matrix is usually constructed from the orbitals computed on the previous iteration. Thus, a sequence of eigensystems must be solved until convergence is attained. Moreover, the SCF iteration often is the inner iteration in a geometry optimization in which the nuclear coordinates are optimized with respect to energy. Thus, a single geometry optimization for a molecule with even a few heavy atoms (light atom refers to hydrogen; heavy atom refers to all other types) may require the solution of hundreds of large real symmetric eigensystems.

Geometry optimization proceeds by calculating the resultant forces of each atom in the system and then moving the atoms in the direction determined by these forces so as to lower the energy of the system. When the geometry is within a preset distance or energy of the local minimum, the optimization is stopped. The minimization routine in MOPAC can be chosen as a modified Broyden–Fletcher–Goldfarb–Shanno or BFGS method [30], or, an eigenvector following minimization routine [31,32]. The additional relevant mathematics in these procedures for parallel implementation is the evaluation of derivatives of the energy expression with respect to the atomic coordinates of the molecular system.

An additional computation in the SCF procedure is the vibrational analysis. This calculation characterizes the stationary point as found by the geometry optimization, and provides thermodynamic information for the molecular system. In vibrational analyses, the Hessian matrix of the energy with respect to the $3 * N - 6$ (N = number of atoms in the molecule) degrees of vibrational freedom in the nuclear coordinates, needs to be evaluated. The elements of the Hessian matrix are of the form,

$$A_{ij} = \left(\frac{\partial^2 E}{\partial q_i \partial q_j}\right),$$

where $q_i, \cdots, q_n$ are internal coordinates suitable for describing nuclear motion, and the energy expression is as developed above. Diagonalization of this matrix results in sets of eigenpairs (λ_i, x_i) which determine vibrational frequencies and corresponding normal modes for the molecule being calculated. The vibrational eigensystems are usually dimensionally somewhat smaller than in the SCF case, but again they may need to be solved repeatedly, for example, as part of a reaction path following computation.

2. Parallel MOPAC: Structure and task distribution

MOPAC is public-domain software available through QCPE [33]. Version 6.0 of MOPAC runs on VAX, CRAY and workstation platforms, and consists of approximately 50,000 lines of FORTRAN code, in 190 subroutines. Resident memory usage in MOPAC is governed entirely by parameter settings chosen at compile time. The amount of storage required by MOPAC depends on the number of heavy (non-hydrogen) and light (hydrogen) atoms that the code has been parametrized to handle at compile time, and whether configuration interaction capabilities are incorporated.

The majority of the computational time required to run MOPAC sequentially is divided among evaluating the electronic interaction integrals (Hartree–Fock matrix preparation), calculating first derivatives (geometry optimization procedure), calculating second derivatives (vibrational analysis) and solving the resulting eigensystem (diagonalization). The precise division of CPU time among the tasks for a geometry optimization procedure may vary with molecular composition (e.g., the diagonalization task can represent from 30–80% of the total computational load); however, the general procedures which dominate the work load for the total calculation will remain the same. Therefore, the SCF calculation, geometry optimization, and second derivative evaluation (vibrational analysis) for the available Hamiltonians were parallelized in this work.

The parallelized algorithms were implemented on a 64-processor Intel iPSC/860 hypercube, and subsequently on an Intel Paragon; both distributed-memory, message-passing parallel computers. In the Intel hypercube, each processing node contains an Intel i860 CPU and 8 Mbytes of RAM (16 Mbytes/node on the Paragon). The communication links are through Intel's Direct-Connect Communications (DCM) hardware with a 2.8 Mbyte/s maximum bandwidth for the iPSC/860 and 10–12 Mbytes/s on the Paragon[2].

[2] The band width on the Intel Paragon is widely variant depending on the system configuration; potentially, one could see a value as high as 4–5 times this.

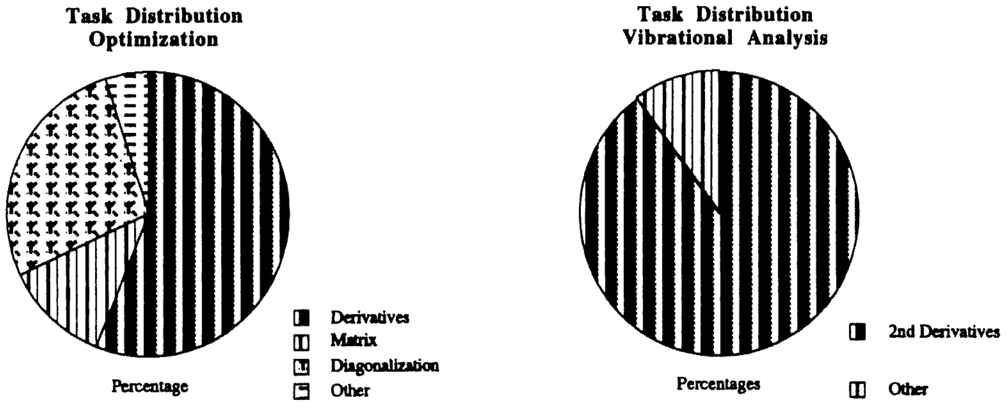


Fig. 2. Pie charts illustrating the distribution of tasks for a representative optimization and vibrational analysis calculation.

2.1. GEOMETRY OPTIMIZATION COMPONENT

Detailed inspection of the algorithmic format of MOPAC indicates that most of the computational work in the semiempirical geometry optimization procedure is distributed over the following three tasks:

(1) Evaluation of one- and two-electron matrix elements.
(2) Formation of the Fock matrix and diagonalization.
(3) Evaluation of derivatives.

(1) *Evaluation of one- and two-electron elements.* Geometry optimization begins with a call to a controller for the specific optimization method. This routine makes several calls to subprograms to carry out the various aspects of full geometry optimization. Much of the calculation occurs in setting up the Hamiltonian matrix (Scheme I: Task 1, Loop over ATOMS). The resulting matrix elements are used to calculate the SCF heats of formation, the nuclear energy, and the one- and two-electron interaction integrals. MOPAC is based on a semiempirical approach,

```
Specific      Loop Sequence
 Task

              Loop according to SCF criteria (COMPFG)
  1           ==>Evaluation of Hamiltonian matrix elements (HCORE)
                    Loop over total number of ATOMS
                        *  fill 1 e- diagonal/off diagonal of same atom
                        *  fill atom-other atom 1 e- matrix (H1ELEC)
                        *  Calculate 2 e- integrals
                           Calculate e- - nuclear terms          (ROTATE)
                           Calculate nuclear-nuclear terms
                    Merge 1 electron contributions private to each CPU
  2           ==>Formation of Fock matrix and Diagonalization (ITER)
  2a                Loop over number of ATOMS
                        *  remaining 1 e- elements (FOCK1)
                        *  2 e-/2-center repulsion elements of Fock matrix (FOCK2)
  2b                Loop over number of ORBITALS
                        *  density matrix (DENSIT)
  2c                Loop over matrix BLOCKS
                        Diagonalization (DIAG)
                            *  Construct part of the secular determinant over MO's
                               which connects occupied & virtual sets.
                            *  Crude 2x2 rotation to "eliminate" significant elements.
                            *  Rotation of pseudo-eigenvectors.
                    Merge contributions private to each CPU
  3           ==>Evaluation of Derivatives (DERIV)
                    Loop over number of ATOMS
                        * derivatives of energy w.r.t. Cartesians (DCART)
                    Loop over number of VARIABLES
                        *  Jacobian :  d(Cart)/d(internals)   (JCARIN)
                    Merge derivatives private to each CPU
```

Scheme I. Schematic showing the parallel SCF procedure.

therefore, many of the integrals are ignored, others are calculated using experimental parameters stored in common blocks, and a few are calculated fully.

The computation of the one-electron and two-electron integrals has been distributed over nodes by partitioning the number of atoms over nodes and giving each node an independent number of integrals to calculate. In general, 100 integrals are calculated for each heavy atom–heavy atom interaction, 10 integrals for each heavy atom–light atom interaction, and 1 integral for each light atom–light atom interaction. Ideal load balancing can be achieved by splitting up the integrals in accord with the type of interaction so that each node receives approximately equal work to do, i.e. statistical decomposition.

Because each two-electron integral contributes to several Fock matrix elements, it is necessary to have the independent node results collected before the Fock matrix is created. A way around this is to have each processor work on its own partial Fock matrix, which is gathered once at the very end. The construction of MOPAC makes this more difficult, but is currently being investigated.

(2) *Formation of Fock matrix and diagonalization.* The formation of the Fock matrix involves computation of the remaining contributions to the one-center integrals, and the two-electron two-center repulsion terms. Each of these subtasks is split over nodes in accord with the number of atoms (Scheme I: Task 2a, Loop over ATOMS). Once this is done, the density matrix can be computed along with information about orbital occupancy. This task is distributed over nodes in accord with the number of orbitals (Scheme I: Task 2b, Loop over ORBITALS).

MOPAC employs a combination of techniques for complete diagonalization. A "fast" or pseudo-Jacobi diagonalization procedure is invoked in initial SCF iterations. The diagonalizations during the final SCF iterations are then taken over by a more rigorous QL algorithm [34–36].

Typically, a diagonalization method consists of a sequence of orthogonal similarity transformations. Each transformation is designed to annihilate one or more of the off-diagonal matrix elements. In the case of the Jacobi method, successive transformations then undo previously set zeros, but the off-diagonal elements continue to decrease until the matrix is diagonal to the precision of the machine. Accumulating the product of eigenvector transformations gives the matrix of eigenvectors, and the elements of the final diagonal matrix are the eigenvalues. In general, the QL (QR if the matrix is reversed graded) algorithms are much faster than the Jacobi methods, however, the Jacobi methods can be computationally time-favorable relative to QL if a good initial approximation is available, and only a single Jacobi-sweep is done.

MOPAC replaces the full QL eigensolution by a single Jacobi-like sweep of just the occupied-virtual block for intermediate SCF iterations often with considerable speed enhancements [37]. The algorithm is considered a pseudo- *diagonalization* technique because the vectors generated by it are more nearly able to block-diagonalize the Fock matrix over molecular orbitals than the starting vectors. It is considered *pseudo* for several reasons [3], the most important of which is that the procedure does not generate eigenvectors. In the chemical sense, the full orbital matrix representations is not diagonalized, only the occupied-virtual intersection is. All of the approximations used in this pseudo-diagonalization routine become valid at self-consistency, and further, the approach to self consistency is not slowed down [38].

Given the lower half triangle of the matrix to be diagonalized in packed form, the algorithm has three primary loop sequences that constitute the procedure (Scheme 1: Task 2c, Loop over matrix BLOCKS). The first two loops together perform the similarity transformation

$$V^t F V$$

that transforms from the atomic orbital to the molecular orbital representation (F represents the Fock matrix). This representation ensures that the resulting eigenvectors are orthogonal, spanning the N-atom dimensional space. This step is followed by rotation, which eliminates off-diagonal elements. The matrix is then block diagonalized only, because only Fock elements connecting occupied and virtual orbitals must be zero at convergence.

Two methods of parallel decomposition were investigated for the diagonalization procedure. The initial attempts distributed the work load over rows or columns of the matrix, i.e., control decomposition. This method resulted in timings that were actually significantly slower than the original unparallelized routine. This is due to large communication overhead from processing such small amounts of data. In addition, two utility routines were written to establish each node's starting work load position. Calls to these routines, along with additional global calls to gather and broadcasts to announce individual node data, resulted in extreme overhead costs.

To avoid some of the complications of the above, a domain decomposition was employed. In this method, large groups or blocks of the matrix are distributed over nodes. Parallelization in this manner eliminates the need for broadcasting intermediate results. Only the final computed vectors are gathered via a global routine. Broadcasting of intermediate results is no longer necessary and scratch arrays already available are used for parallel decomposition so that no additional memory is required for this parallel method.

(3) *Evaluation of derivatives.* Additional CPU-intensive subroutines involved in the geometry optimization include those that carry out derivative evaluation (Scheme I: Task 3, Loop over VARIABLES). The derivatives of the energy with respect to the internal coordinates is done via finite differences. The total work involves $3 * N$ variables that can be distributed equally over the number of nodes.

2.2. VIBRATIONAL ANALYSIS COMPONENT

Vibrational analysis (second-derivative evaluation) of molecular systems can be a formidable task. These calculations are, however, essential to characterize stationary points and to assess vibrational and thermodynamic properties of molecules. The vibrational analysis procedure involves construction of a $3 * N$ dimensional matrix of second derivatives of energy with respect to Cartesian coordinates (Scheme II). The calculation of each of these matrix elements represents an independent calculation, and the procedure holds the potential of being perfectly parallel. Following the calculation of matrix elements across nodes, the results are collected using a global routine and the full matrix diagonalized.

The diagonalization of the matrix results in a set of eigenvectors, corresponding to the $3 * N - 6$ vibrational motions, and a corresponding set of eigenvalues, which represent the respective vibrational frequencies of these motions. The other 6 eigenvectors correspond to the rotational and translational matrix, with associated zero eigenvalues (disregarding numerical artifacts).

Scheme II shows the vibrational analysis procedure. The parallelization of the vibrational analysis component requires partitioning $3 * N$ variables over nodes to calculate a matrix of second derivative elements. Because this is a symmetric matrix, there are $3 * N * (3 * N + 1)/2$ unique elements to be computed. It is critical to maintain proper indices over the nodes as the results are calculated. A global routine is invoked to collect the matrix in preparation for diagonalization.

3. Results

The parallel procedures were first implemented on the 64 nodes Intel iPSC/860 at the San Diego Supercomputer Center. Since the first implementation of parallel MOPAC, we have replaced the iPSC/860 with the Paragon. In general, code performance is identical with the exception of a 25% faster clock in the Paragon, thus shifting the resulting curves by the appropriate amount. Code performance was demonstrated on a large group of molecules, varying in symmetry construction and heavy atom/light atom ratios (fig. 3). Computation timing results for geometry optimization (table 1) and vibrational analysis (table 2) calculations are reported.

Although a principal performance measure is the elapsed time necessary to solve the problem of interest, temporal performance shows more clearly the behavior of a parallel program as a function of the number of processors. Temporal performance is defined [39] as the inverse of the execution time, and has the units of [(fraction of total solution completed)/second] in this application (sol/s). The objective behind this metric is to judge the performance of a particular algorithm based on

```
Calculation of force constants and vibraional frequencies (FORCE)
    Loop over number of VARIABLES
    *   Calculate second-order of the energy with repect to
        the Cartesian coordinates (FMAT)
    Merge second derivative components private to each CPU
```

Scheme II. Schematic showing the parallel vibrational analysis procedure.

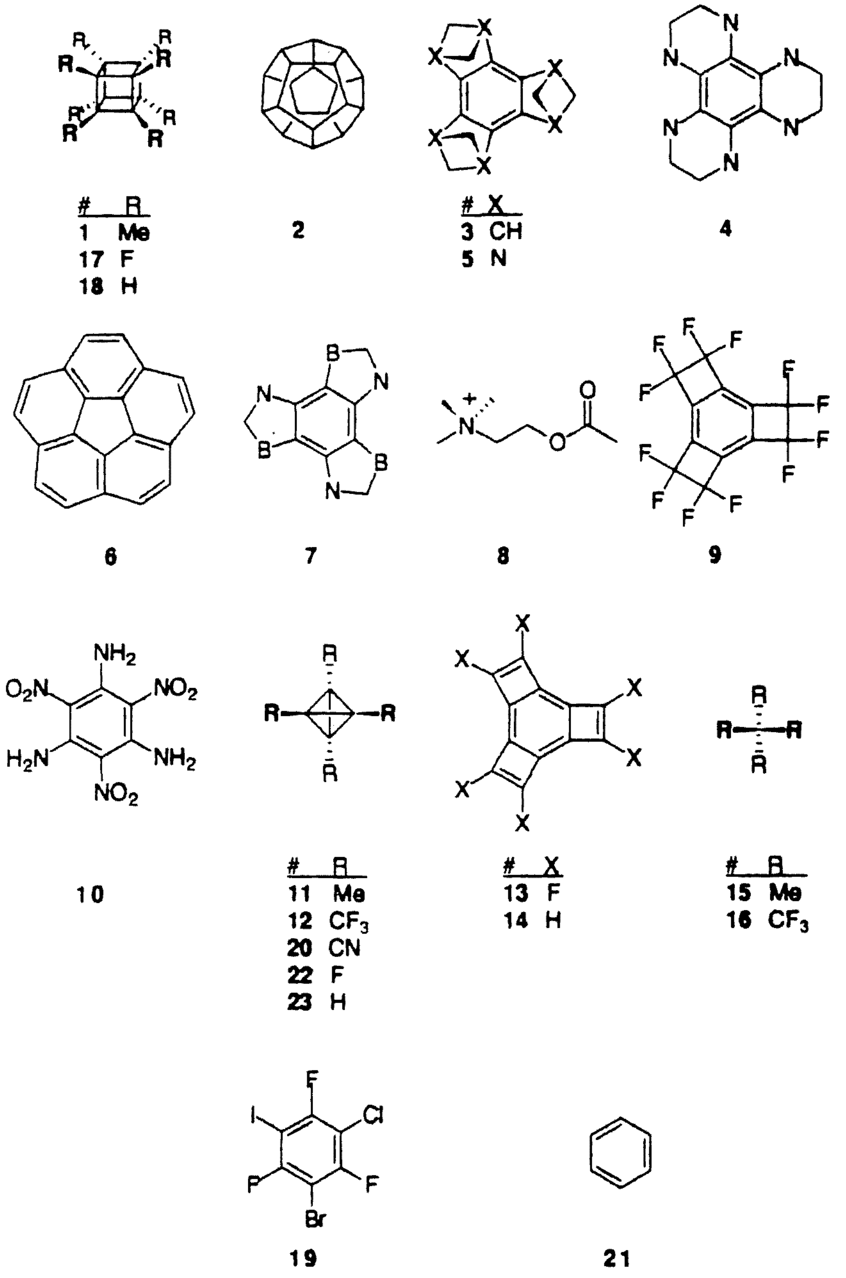


Fig. 3. Molecular structures.

the fastest execution time. Single processor timings are taken as the best serial algorithm. For MOPAC, timings for the serial and parallel algorithms on a single node are identical. This is due to the particular parallel implementation which is

Table 1
MOPAC optimization results.

H/L	# atoms	molecule	Nodes															
			1		2		4		8		16		32		64		CRAY	
			T[a]	MF[b]	S[c]	MF	S	MF	S	MF	S	MF	S	MF	S	MF	S	MF
16/24	40	1	404.3	2.9	1.9	5.1	2.4	7.0	3.3	9.5	4.1	11.7	4.6	13.2	5.2	**15.1**	8.4	22.2
20/20	40	2	749.7	2.8	1.6	4.4	2.0	5.8	2.6	8.1	3.2	10.0	3.6	11.4	4.9	**13.2**	8.9	23.7
18/18	36	3	953.8	2.8	1.6	4.8	2.6	7.3	3.1	9.2	4.0	11.7	4.7	13.8	4.8	**14.0**	9.0	25.2
18/18	36	4	4095.0	2.9	1.6	4.6	1.9	5.7	2.4	7.2	2.9	8.6	3.2	9.5	3.6	**10.7**	7.7	22.7
18/12	30	5	723.0	3.5	1.6	5.7	2.2	7.8	3.0	10.5	3.4	12.0	4.1	14.2	4.1	**14.3**	7.4	25.9
20/10	30	6	1506.9	2.8	1.5	4.4	2.1	6.0	2.9	8.2	3.3	9.3	3.8	10.8	3.8	**10.9**	8.3	23.0
15/12	27	7	512.0	2.9	1.7	4.9	2.2	6.6	3.1	9.2	2.4	10.2	4.1	12.0	4.0	**11.9**	8.1	23.9
10/16	26	8	421.2	2.7	1.3	3.4	1.6	4.3	2.1	5.8	2.6	6.8	2.9	**7.6**	2.8	7.5	6.8	19.3
18/6	24	9	467.8	3.0	1.5	4.7	2.4	7.3	3.2	9.8	3.6	10.9	4.4	**13.4**	3.9	12.0	7.8	23.2
18/6	24	10	930.0	2.8	1.4	4.0	1.7	4.9	2.4	6.8	2.8	8.0	3.4	**9.6**	3.4	9.4	8.4	23.7
8/12	20	11	49.4	2.5	1.7	4.3	2.3	5.8	3.2	8.0	3.7	9.1	3.7	**9.4**	3.7	9.2	7.8	20.0
20/0	20	12	136.9	2.7	1.3	3.5	1.9	5.1	2.4	6.5	2.8	7.5	3.3	**8.9**	3.3	8.7	9.1	24.6
18/0	18	13	272.2	2.6	1.3	3.4	1.8	4.7	2.3	6.0	2.7	7.1	3.3	**8.7**	3.2	8.5	9.0	23.8
12/6	18	14	163.4	2.5	1.5	3.9	2.1	5.3	2.6	6.7	3.0	7.6	3.3	**8.3**	3.2	8.1	8.5	21.4
5/12	17	15	12.3	2.1	1.5	3.3	2.1	4.6	2.7	6.0	3.4	7.1	3.7	**7.8**	3.4	7.5	8.2	16.4
17/0	17	16	54.6	2.4	1.3	3.0	1.7	4.1	2.2	5.3	2.6	6.2	3.1	**7.4**	3.0	7.3	9.1	22.2
16/0	16	17	178.0	2.8	1.3	3.7	1.9	5.2	2.5	7.0	3.1	**8.7**	3.1	8.6	3.1	8.5	9.2	24.5
8/8	16	18	83.4	2.7	1.7	4.4	2.0	5.4	2.5	7.0	3.1	**8.5**	3.1	8.4	3.1	8.1	8.0	21.8
12/0	12	19	231.6	3.7	0.85	3.1	1.4	5.2	1.8	7.4	2.4	**8.9**	2.4	8.8	2.3	8.7	5.1	18.7
12/0	12	20	43.3	2.6	1.3	3.4	1.8	4.8	2.2	5.8	2.7	**7.1**	2.7	7.0	2.6	6.8	9.0	23.3
6/6	12	21	21.7	2.4	1.6	3.8	1.9	4.6	2.3	5.4	2.8	**6.8**	2.7	6.5	2.6	6.2	8.0	19.1
8/0	8	22	14.1	2.6	1.2	3.2	1.7	4.3	2.2	**5.6**	2.2	5.4	2.1	5.4	2.0	5.1	10.0	20.9
4/4	8	23	5.1	2.2	1.7	3.5	1.7	3.9	2.5	**4.7**	2.5	4.5	2.0	4.2	1.7	4.0	7.1	7.1

[a] CPU time in seconds.
[b] Mflops as defined in text.
[c] Speedup as defined in text. For nodes 2–64, Time = Time (1 node) / Speedup.

Table 2
MOPAC vibrational analysis results.

H/L	# atoms	molecule	Nodes															
			1		2		4		8		16		32		64		CRAY	
			T[a]	MF[b]	S[c]	MF	S	MF	S	MF	S	MF	S	MF	S	MF	S	MF
16.24	40	1	3135.5	2.6	1.9	5.0	3.8	9.8	7.3	19.0	13.0	33.9	23.4	61.0	38.6	100.5	8.1	21.0
20/20	40	2	4075.5	2.6	2.0	5.1	3.5	9.1	7.5	19.4	13.4	34.7	24.3	62.8	40.8	105.2	8.7	22.7
18/18	36	3	2827.7	2.8	1.9	5.5	3.7	10.5	7.1	20.2	12.7	36.1	20.3	57.5	32.9	93.7	7.9	22.5
18/18	36	4	2949.2	3.0	1.9	5.7	3.6	10.7	6.8	20.5	12.1	36.4	19.5	58.6	31.5	94.6	7.8	23.5
18/12	30	5	1967.8	2.9	1.9	5.5	3.8	10.8	7.0	20.0	12.6	36.1	21.9	62.7	28.7	82.1	8.1	23.2
20/10	30	6	2520.1	3.2	1.9	6.1	3.7	11.7	6.8	21.7	12.6	40.4	22.3	71.6	29.6	95.0	7.9	25.3
15/12	27	7	1347.0	2.9	1.9	5.6	3.7	10.7	7.0	20.4	12.3	36.0	19.3	56.3	25.3	74.0	7.8	22.9
10/16	26	8	750.4	2.5	2.0	5.0	3.7	9.5	7.0	17.8	12.7	32.4	19.2	48.4	24.4	61.9	7.6	19.4
18/6	24	9	1278.0	2.8	1.9	5.5	3.7	10.4	7.0	19.9	11.9	33.7	18.1	51.2	22.7	64.3	8.4	23.8
18/6	24	10	1330.8	2.9	1.9	5.6	3.7	10.7	7.1	20.4	12.0	34.6	17.9	51.6	23.5	67.7	8.5	24.4
8/12	20	11	357.5	2.6	1.9	5.0	3.7	9.6	6.7	17.2	12.0	30.7	19.9	50.7	27.7	70.8	7.6	19.4
20/0	20	12	1048.2	2.6	2.0	5.1	3.8	9.8	6.9	17.7	12.4	31.9	21.0	54.0	31.3	80.2	9.4	24.1
18/0	18.	13	804.0	2.8	2.0	5.5	3.7	10.3	7.0	19.5	11.3	31.8	19.3	54.1	28.4	79.5	8.8	24.6
12/6	18	14	459.3	2.9	1.9	5.6	3.6	10.4	6.9	19.8	11.1	32.0	18.7	53.6	27.7	79.4	7.9	22.7
5/12	17	15	168.1	2.3	1.9	4.5	3.7	8.6	6.5	15.0	9.7	22.6	16.3	38.0	21.5	50.1	7.2	16.8
17/0	17	16	615.7	2.4	1.9	4.7	3.7	9.2	6.6	16.1	10.6	26.1	17.9	43.9	26.3	64.6	9.4	23.1
16/0	16	17	543.9	2.5	2.0	5.0	3.8	9.6	7.2	18.2	12.7	32.3	17.1	43.5	25.4	64.5	9.4	23.9
8/8	16	18	226.8	2.6	1.9	5.1	3.8	9.8	7.4	18.4	12.0	31.3	16.3	42.7	22.9	59.9	7.8	20.4
12/0	12	19	321.0	3.5	1.9	6.8	3.7	13.0	6.3	22.3	9.6	34.2	12.8	45.5	17.7	60.5	5.2	18.5
12/0	12	20	230.9	2.7	1.9	5.2	3.7	9.9	6.2	16.6	9.3	24.9	12.4	33.2	17.4	46.4	8.8	23.7
6/6	12	21	92.3	3.0	1.9	5.6	3.6	10.7	6.1	18.2	9.3	27.7	12.3	36.6	14.2	42.2	6.9	20.4
8/0	8	22	62.8	2.5	1.9	4.7	3.6	8.9	6.1	15.1	8.3	20.5	12.1	29.9	10.3	25.5	8.8	21.9
4/4	8	23	26.5	2.5	1.9	4.8	3.5	9.0	6.0	15.3	7.4	18.7	9.8	25.0	7.6	19.3	7.4	18.8

[a] CPU time in seconds.
[b] Mflops as defined in text.
[c] Speedup as defined in text. For nodes 2–64, Time = Time (1 node)/Speedup.

basically a shared memory version of the code with global libraries used to synchronize local copies of the data structure.

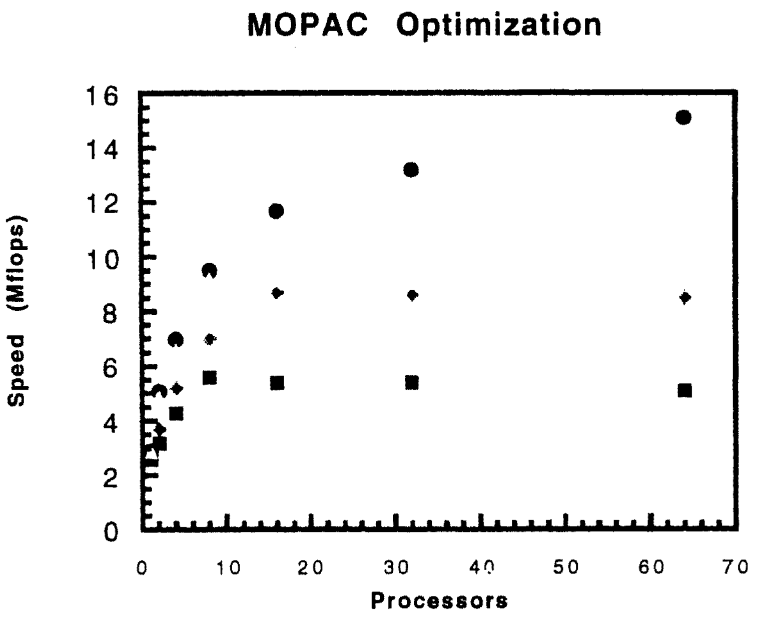
As mentioned previously, detailed timings demonstrate that the majority of the computational time spent in MOPAC is split over evaluation of Fock matrix elements, evaluation of derivatives, and diagonalization. Semiempirical methods process N^2 integrals instead of N^4 as with complete HF methods, therefore, the computational bottleneck lies at the diagonalization routine. Timings indicate that diagonalization can represent from 30–80% of the total computational load, depending on the size and makeup of molecular system being considered. This makes an overall N^3 time dependence.

The results from parallelization of the SCF and geometry optimization components of MOPAC are shown in table 1. Calculations were performed on 1, 2, 4, 8, 16, 32 and 64 nodes. Optimization level 3 was invoked during code compilation. This level incorporates global optimization and software pipelining. In general, there is a large variation in absolute values of timings across the series of molecules, a phenomenon that is dependent on much more than the total number of atoms. In particular, factors such as (a) the number of heavy versus light atoms, (b) the symmetry of the molecule, and (c) the initial guess structure, dictate the complexity of the computation. Factors (a) and (b) will govern the number and complexity of integrals that will need to be computed as discussed earlier, and (c) will govern how many iterations necessary for convergence of the geometry optimization.

Table 1 gives data for the geometry optimization calculations for the set of molecules arranged in order of decreasing number of atoms. Speedups approaching 5 with respect to single processor times are observed for individual computations. A definite compartmentalization of the data is noticeable on the basis of the number of atoms in the molecular system. The peak performance across node combinations is in bold face for each molecule. Scanning the entire data base of molecules shows the general rule of thumb:

Number of atoms, n	Number of nodes at optimal performance
> 29	64
$15 < n < 30$	32
$9 < n < 16$	16
< 10	8

Fig. (4a) shows a plot of temporal performance for one molecule within each of the first three groupings (Solutions of molecules that are < 10 atoms are so fast that they are off scale from plots including other categories). One can pick out the peak in these curves as the optimal node range, as the falloff in temporal performance is much faster for smaller and smaller molecules. Running the calculation on more than this optimal number of nodes will be inefficient due to either a very



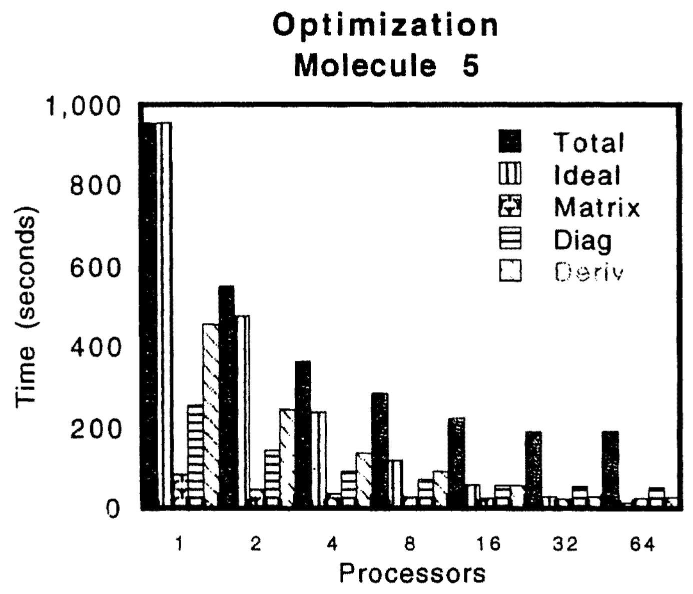


Fig. 4(a). Speedup curve illustrating the variation in Mflop rate with number of processors for an optimization calculation. Note that the optimal number of nodes to run the calculation results is dependent on the size of the molecular system. One molecule from each category of table 1 is plotted: dots: molecule 1; triangles: molecule 7; diamonds: molecule 17; boxes: molecule 22.

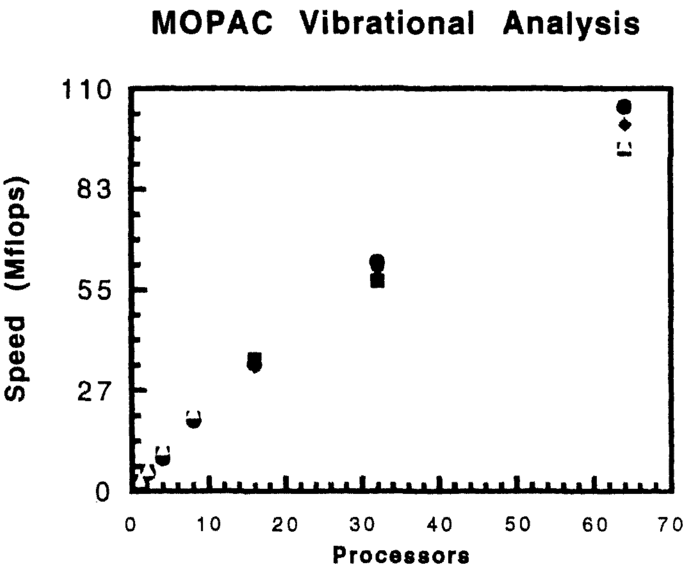
Fig. 4(b). Bar graph showing the effects of parallelization of the various tasks involved in the optimization calculation for molecule 5.

small distribution of work across nodes, or some nodes being left completely idle. In these cases, performance is degraded from communications costs dominating due to many nodes transferring very small amounts of data.

Fig. 4(b) shows a plot of temporal performance for the four largest molecules in table 1. From this plot, one can see the effects of the various factors contributing to the solution time for different molecular constructions. Although the molecules are all roughly similar in size, solution time is widely variant due to the differences in molecular construction. Molecule 1 has a larger number of light atoms to compute integrals for (16/24) than molecule 2 (20/20), thus reducing the overall arithmetic involved. Molecules 3 and 4, although identical in heavy/light atom (18/18) construction, are very different in the number of iterations that are necessary for convergence. Molecule 4 is a very floppy molecule with a flat molecular potential energy surface, and so convergence is very difficult, as seen with the relatively enormous computation time and number of iterations required for convergence in comparison to molecule 3. Molecules 2 and 3, on the other hand, are similar in size (40 versus 36 atoms), have equal number of heavy versus light atoms, and are similar in the number of iterations necessary for convergence of the geometry. All of these factors are reflected in the plot of temporal performance for the grouping. Molecule 1 shows the fastest overall solution time, molecules 2 and 3 are comparable in solution time, and molecule 4 is highly inefficient in solution time.

A breakdown of the overall computation into individual tasks illustrates more clearly the effects of the parallelization. Fig. 4(b) shows a breakdown over the three tasks parallelized for molecule 5 in table 1 as a representative example. The calculation of derivatives for this molecule represents the greatest component of work, and so the parallelization of the derivative routine is the most impressive, approaching linearity up to 32 nodes before leveling off. Diagonalization and formation of matrix elements comprise a much smaller portion of the overall computation time, and level off at a much faster rate (optimization levels off at 16 nodes, formation of matrix elements at 4 nodes). Geometry optimizations that involve larger matrices (i.e., have more integrals to compute) would show a greater level of parallelization in these latter two tasks.

Timing results for the parallelized vibrational analysis procedure are given in table 2. The absolute timings are more uniform for molecules of similar size and molecular construction than was true for the geometry optimization. This is because the primary task in any particular molecule is the calculation of $3N - 6$ ($N = \#$atoms) second derivatives of energy with respect to coordinates, a function of the number of atoms only. Fig. 5(a) illustrates graphically the temporal performance for a representative sampling of the total group of molecules from table 2 (molecules 1, 2, 6, and 22). The vibrational data show initial linear parallelism for all sizes of molecules. As the work is distributed over more and more nodes, efficiency is lost, as a function of molecular size and temporal performance levels off. One factor contributing to this leveling off is that there is less and less work to distribute over nodes. This is why the larger molecules show better performance than



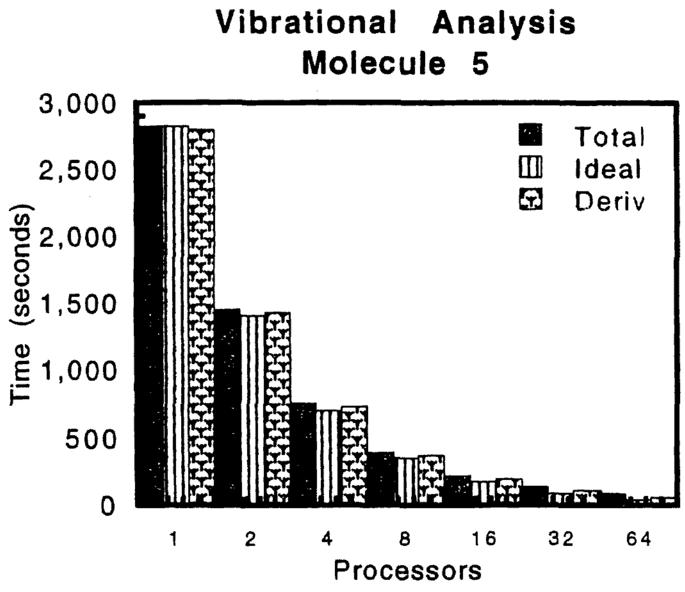


Fig. 5(a). Speedup curve illustrating the variation in Mflop rate with number of processors for a vibrational analysis calculation. Four molecules were chosen for illustration. dots: molecule 1; triangles: molecule 2; diamonds: molecule 3; boxes: molecule 6.

Fig. 5(b). Bar graph showing the effects of parallelization of the derivative task for vibrational analysis of molecule 5.

the smaller molecules. In addition, a latency effect could be contributing to a decrease in efficiency due to more nodes being involved in the global calls, an effect that is uniform over all sized molecules.

As mentioned previously, the calculation of the second derivatives comprise about 90–95% of the total work load in a vibrational analysis calculation. Other minor tasks include the initial SCF to obtain the energetics and first derivatives. Fig. 5(b) shows the effects of parallelization of the second derivative component with respect to both the overall computation time and a model based on Amdahl's law for molecule 5.

An important issue here is the range of problem sizes for which the performance is acceptable. Because keeping the number of processors fixed and increasing the problem size increases the amount of local computation each node does, performance is expected to improve for larger molecular systems. This is illustrated in figs. 6 and 7 for 64 node results over the entire range of molecules for geometry optimization and vibrational analysis, respectively. Similar curves are obtained for the other node combinations.

5. Discussion

A major limitation on performance, especially for the geometry optimization calculations, is the code memory requirement. Even for the largest molecules calcu-

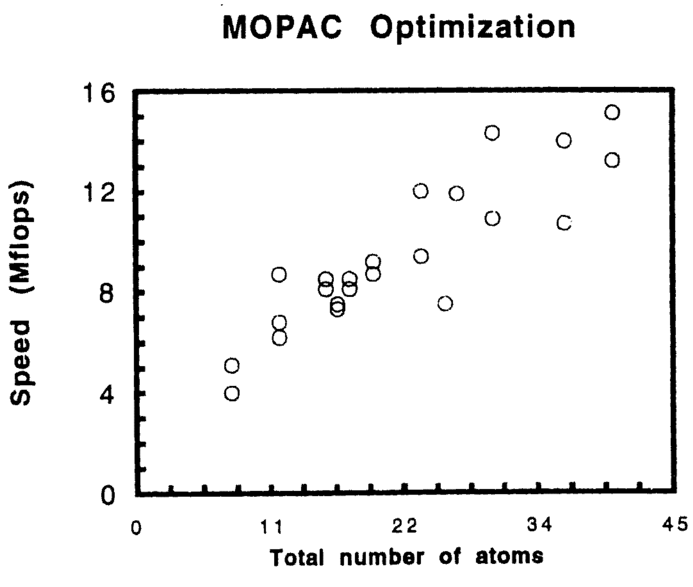


Fig. 6. Plot showing the increase in Mflop rate with increase in size of molecular system for an optimization calculation. The molecules range in size from 8 atoms to 40 atoms.

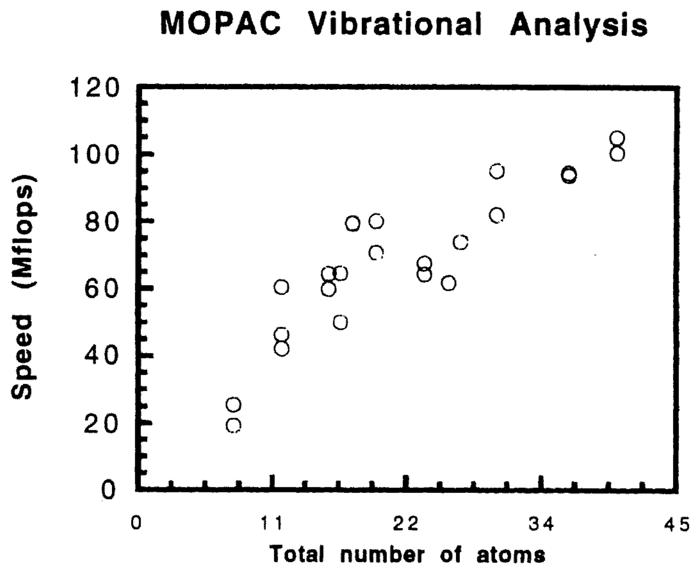


Fig. 7. Plot showing the increase in Mflop rate with increase in size of vibrational analysis calculation. The molecules range in size from 8 atoms to 40 atoms.

lated, there is a noticeable asymptote in the performance curve as the number of nodes increases. This is primarily because the molecular systems are relatively small in comparison to the number of nodes being allocated to do work, a restriction resulting from the memory constraints. The main problem in MOPAC stems from the rather poor structure of the code in terms of memory utilization. The use of replicated data parallel decomposition requires that sufficient memory be held on each processor for the entire symmetric Hamiltonian and Fock matrix. As a result, all internal communications throughout parallelized MOPAC are carried out with fast-library global routines and not via sending/receiving packets of information. This method of parallelization was chosen in order to minimize the communication overhead and latency costs, which were observed to be extremely high, especially with the first levels of operating systems on the MIMD machines [39]. There will still be startup time for these global routines that will contribute to the overall time costs, however, this will be much less due to better algorithmic construction with global routines, and the fact that the global routines are faster than the send/receive routines.

Unfortunately, the goal of being able to calculate larger molecules that can be currently calculated with *ab initio* techniques has not been met, due to these severe memory constraints. The declining cost of semiconductor memory makes it reasonable to assume that large-scale parallel computers will provide sufficient memory per node to accommodate much larger molecules with the existing software. In addition, memory need not be all semiconductor memory; one could think of employing common file system disk storage to accommodate large intermediate information such as integrals. Still, one will inevitably reach the limits of the increased memory capabilities. Therefore, we are currently giving attention to algorithmic modifications and distributed memory capabilities.

There are two directions that one can think about pursuing for this problem. One involves usage of a shared memory model construct on top of a distributed memory platform utilizing the T3D parallel platform. This model supports several styles of programming: message passing, data parallel, global address (shared data), and work sharing, all of which may be combined in the same program. The model includes features that allow a user to define a program in terms of the behavior of the system as a whole (task parallel), and/or, in terms of the behaviors of individual tasks (data parallel). The second direction, which is in the same spirit as the first, involves the implementation of specialized global array tools [40] that allow a distributed data parallel strategy to be implemented within the iterative SCF procedure. Preliminary results in both of these areas show much more promise towards the calculation of molecules of the size of hundreds of atoms.

6. Conclusions

With (MIMD) computers clearing the way for record-breaking computation speeds, scientific programmers of the 90s are being pushed to the world of parallel programming. Massively parallel processors achieve their high speed by working on many parts of the problem in parallel. While it is difficult in many cases to structure a problem for efficient highly parallel solution, for those problems for which the technique is applicable, these computers are an increasingly important computational tool, especially for large and difficult chemistry problems. Thus, it is clear that implementing chemistry applications in parallel environments is a milestone for computational chemistry.

In this work, we have demonstrated the promise as well as the difficulties involved in the implementation of semiempirical quantum chemistry applications on the Intel hypercube platform. As the first level of implementation of these methods, a replicated data parallelism strategy has been employed. In this strategy, even though tasks are distributed over nodes, results of all distributed tasks are collected together on each node (replicated) at various points within the Hartree–Fock procedure, thereby causing limitations, especially for the geometry optimization calculation, due to the amounts of memory necessary to hold these quantities on each individual node. This severely limits the size molecular system that can be calculated and forces an unacceptably low ratio of processors to memory. With less than 32 Mbytes/node, the size of molecular systems that can be modeled is limited to less than 60 atoms, and the performance saturates at 16 to 32 nodes.

The vibrational analysis component shows more promise within the replicated data parallel implementation. Results show a near linear relationship between the number of nodes and the performance of a series of widely varying molecular constructs. In addition, this work has shown the potential for parallelization of the vibrational component within *ab initio* codes [13].

Significant attention by this author as well as others [40], is now being given towards the implementation of these methods using a distributed data parallel strategy, which clearly shows to be superior in light of the known memory problems associated with these methods [39]. In the distributed data parallel strategy, individual node tasks are not collected together on each node at any time during the calculation. Thus, performing a quantum mechanical calculation on a molecule of size N atoms, can be distributed over p processors such that only N/p amount of memory is ever needed on any individual node. This will allow our goals involving calculation of molecules with hundreds of atoms, and study of reaction paths and solvent effects of large systems to be a reality.

Acknowledgements

The author would like to thank Scott Kohn and Jay Siegel for several beneficial discussions. Support was provided by the National Science Foundation (Grant No. ASC-8902827 and Grant No. ASC-8902827), Intel Corporation, and the San Diego Supercomputer Center.

References

[1] W.J. Hehre, L. Radom, P. v. R. Schleyer and J.A. Pople, *Ab Initio Molecular Orbital Theory* (Wiley, New York, 1986).
[2] H.F. Schaefer, Science 231 (1986) 1100.
[3] J.J.P. Stewart, P. Csaszar and P. Pulay, J. Comp. Chem. 3 (1982) 227.
[4] P. Pulay, in: *Modern Theoretical Chemistry*, ed. H.F. Schaefer (Plenum Press, New York, 1977) p. 153.
[5] J.A. Pople, R.A. Krishnan and H.B. Schlegel, In. J. Quantum Chem. Symp. 13 (1979) 225.
[6] H.P. Luthi, J.E. Mertz, M.W. Feyereisen and J.E. Almlof, J. Comp. Chem. 13 (1992) 160.
[7] R.A. Whiteside, J.S. Binkley, M.E. Colvin and H.F. Schaefer III, J. Chem. Phys. 86 (1987) 2185.
[8] J.E. Hertz and J.W. Andzelm, CRAY Channels 10 (1991).
[9] H.P. Luthi and J.E. Mertz, Supercomputing Review (1992).
[10] M.F. Guest, R.J. Harrison, J.H. van Lenthe, L.C.H. van Corier, Theor. Chim. Acta. 71 (1987) 117.
[11] M. Dupuis and J.D. Watts, Theoret. Chim. Acta 7 (1987) 91.
[12] R. Carbó, L. Molino and B. Calabuig, J. Comp. Chem. 13 (1992) 155.
[13] M.W. Schmidt, K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L. Windus and S.T. Elbert, J. Comp. Chem. 14 (1993) 1347.
[14] J.J.P. Stewart, Quantum Chem. Exchange Bull. 5 (1985) 126.
[15] M.J.S. Dewar, E.G. Zoebisch, E.F. Healy and J.J.P. Steward, J. Am. Chem. Soc. 107 (1985) 3902.

[16] J.J.P. Stewart, (1988).
[17] A. Szabo and N. Ostlund, *Modern Quantum Chemistry* (MacMillan, New York, 1982).
[18] M.J. Frisch, M. Head-Gordon, G.W. Trucks, J.B. Foresman, B. Schlegel, K. Raghavachari, M. Robb, J.S. Binkley, C. Gonzalez, D.J. Defrees, D.J. Fox, R.A. Whiteside, R. Seeger, C.F. Melius, J. Baker, R.L. Martin, L.R. Kahn, J.J.P. Steward, S. Topial and J.A. Pople, Gaussian Inc., Chicago (1992).
[19] M.W. Schmidt, K.K. Baldridge, J.A. Boatz, J.H. Jensen, S. Koseki, M.S. Gordon, K.A. Nguyen, T.L. Windus and S.T. Albert, Quantum Chem. Program Exchange Bull. 10 (1990).
[20] R.D. Amos and J.E. Rice, *The Cambridge Analytical Derivatives Package*, Cambridge (1987).
[21] T.Z. Kalamboukis, Parallel Comput. 18 (1992) 207.
[22] J. Dongarra and D. Sorensin, SIAM J. Sci. Stat. Comput. 8 (1987) s139.
[23] C.F. Ipsen and E.R. Jessup, SIAM J. Sci. Stat. Comput. 12 (1991) 469.
[24] A. Szabo and N.S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory* (MacMillan, New York, London, 1982).
[25] M.J.S. Dewar and W. Thiel, J. Am. Chem. Soc. 99 (1977) 4899.
[26] R.C. Bingham, M.J.S. Dewar and D.H. Lo, J. Am. Chem. soc. 97 (1975) 1285.
[27] J.J.P. Stewart, J. Comp. Chem. 10 (1989) 209.
[28] J.J.P. Stewart, J. Comp. Aided Mol. Design 4 (1990) 45.
[29] J. Jensen, K.K. Baldridge and M.S. Gordon, J. Phys. Chem. 96 (1992) 8340.
[30] C.G. Broyden, J. Inst. Math. Appl. 6 (1970) 222.
[31] J. Baker, J. Comp. Chem. 7 (1986) 385.
[32] J. Baker, J. Comp. Chem. 8 (1987) 563.
[33] *QCPE: Quantum Chemistry Program Exchange*, Creative Arts Building 181, Indiana University, Bloomington, IN 47505, USA.
[34] B.N. Partlett, *The Symmetric Eigenvalue Problem* (Prentice-Hall, Englewood Cliffs, 1980).
[35] W.H. Press, *Numerical Recipes* (Cambridge University Press, New York, 1986).
[36] J. Butcher, Math. Comput. 19 (1965) 408.
[37] J. Demmel, Veselic University of Tennessee, CS-89-88 (1989).
[38] B. Yoshitake, Computers and Chem. 6 (1982).
[39] R. Hockney, *Computer Benchmarks* (Elsevier, Amsterdam, 1993).
[40] M.F. Guest, E. Apra, D.E. Bernholdt, H.A. Fruchtl, R.J. Harrison, R.A. Kendall, R.A. Kutteh, J.B. Nicholas, J.A. Nichols, M.S. Stave and A.T. Wong, unpublished results (1994).