

**Scalable computational chemistry:
New developments and applications**

by

Yuri Alexeev

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Physical Chemistry

Program of Study Committee:
Mark S. Gordon Major Professor
Robert J. Angelici
Bruce Harmon
Jacob W. Petrich
Xueyu Song

Iowa State University

Ames, Iowa

2002

Copyright © Yuri Alexeev, 2002. All rights reserved.

Graduate College
Iowa State University

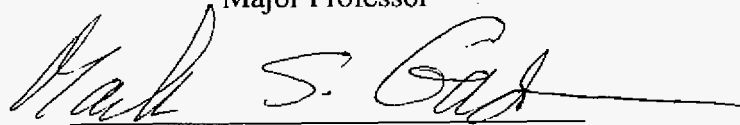
This is to certify that the doctoral dissertation of

Yuri Alexeev

has met the dissertation requirements of Iowa State University

A handwritten signature in black ink, reading "Mark S. Gaud". The signature is written in a cursive style with a long horizontal flourish extending to the right.

Major Professor

A second handwritten signature in black ink, identical to the one above, reading "Mark S. Gaud".

For the Major Program

iii

to Katia

TABLE OF CONTENTS

ACKNOWLEDGEMENTS		vi
ABSTRACT		vii
CHAPTER 1.	GENERAL INTRODUCTION	
	Introduction	1
	Dissertation Organization	2
	Literature Review	4
	References	8
CHAPTER 2.	A THEORETICAL STUDY OF THE BYS-SILYLATION OF ETHYLENE CATALYZED BY TITANIUM DICHLORIDE	
	Abstract	10
	Introduction	11
	Computational methods	13
	Results and discussion	14
	1. Oxidative addition	17
	2. Ethylene insertion	22
	3. Reductive elimination	25
	Conclusions	27
	Acknowledgement	28
	References	28
CHAPTER 3.	THE DISTRIBUTED DATA SCF	
	Abstract	32
	Keywords	32
	1. Introduction	33
	2. Tools and platforms	35
	3. Brief review of replicated data SCF parallel implementation	38
	4. Distributed Data SCF (DDSCF)	42
	5. Performance models	51
	6. Results	55
	7. Conclusions	58
	Acknowledgement	59
	References	60

CHAPTER 4.	A DISTRIBUTED DATA PARALLEL CPHF ALGORITHM FOR ANALYTIC HESSIANS	
	Abstract	63
	Keywords	64
	Introduction	64
	Tools and platforms	67
	Brief review of CPHF theory	68
	Distributed data CPHF algorithm	70
	1. A blocking technique	73
	2. A localized data access technique	73
	3. Self consistent work distribution	75
	4. Cache optimization technique	77
	Results	78
	Conclusions	80
	Acknowledgement	80
	References	81
CHAPTER 5.	A DISTRIBUTED DATA PARALLEL CASSCF ALGORITHM	
	Abstract	84
	Keywords	84
	Introduction	85
	Tools and platforms	86
	Review of CASSCF theory	87
	CASSCF parallelization strategy	92
	Conclusions	95
	Acknowledgement	95
	References	96
CHAPTER 6.	GENERAL CONCLUSIONS	98
APPENDIX.	SUPPLEMENTARY MATERIALS	100

ACKNOWLEDGMENTS

I would like to thank Professor Mark S. Gordon for his help and support during time I spent in graduate school. Mark's help made it possible for me to live up to expectations. The research work presented in this thesis would never been possible without Mark's support especially when I just began my research career in the United States. We argued and often disagreed on different issues, but we always were able to find a resolution which ultimately resulted in the original research presented in this thesis.

I would also like to thank the members of Mark's research group. In particular, I would like to thank Dr. Mike Schmidt for his help and fruitful discussions during my graduate work.

This work was performed at Ames Laboratory under Contract No. W-7405-Eng-82 with the U.S. Department of Energy. The United States government has assigned the DOE Report number IS-T 2013 to this thesis.

ABSTRACT

The computational part of the thesis is the investigation of titanium chloride (II) as a potential catalyst for the bis-silylation reaction of ethylene with hexachlorodisilane at different levels of theory. Bis-silylation is an important reaction for producing bis(silyl) compounds and new C-Si bonds, which can serve as monomers for silicon containing polymers and silicon carbides. *Ab initio* calculations on the steps involved in a proposed mechanism are presented. This choice of reactants allows us to study this reaction at reliable levels of theory without compromising accuracy. Our calculations indicate that this is a highly exothermic barrierless reaction. The TiCl_2 catalyst removes a 50 kcal/mol activation energy barrier required for the reaction without the catalyst. The first step is interaction of TiCl_2 with ethylene to form an intermediate that is 60 kcal/mol below the energy of the reactants. This is the driving force for the entire reaction. Dynamic correlation plays a significant role because RHF calculations indicate that the net barrier for the catalyzed reaction is 50 kcal/mol. We conclude that divalent Ti has the potential to become an important industrial catalyst for silylation reactions.

In the programming part of the thesis, parallelization of different quantum chemistry methods is presented. The parallelization of code is becoming important aspect of quantum chemistry code development. Two trends contribute to it: the overall desire to study large chemical systems and the desire to employ highly correlated methods which are usually computationally and memory expensive. In the presented distributed data algorithms computation is parallelized and the largest arrays are evenly distributed among CPUs. First,

the parallelization of the Hartree-Fock self-consistent field (SCF) method is considered. SCF method is the most common starting point for more accurate calculations. The Fock build (sub step of SCF) from AO integrals is also often used to avoid MO integral computation. The presented distributed data SCF increases the size of chemical systems that can be calculated by using RHF and DFT. The important *ab initio* method to study bond formation and breaking as well as excited molecules is CASSCF. The presented distributed data CASSCF algorithm can significantly decrease computational time and memory requirements per node. Therefore, large CASSCF computations can be performed. The most time consuming operation to study potential energy surfaces of reactions and chemical systems is Hessian calculations. The distributed data parallelization of CPHF will allow scientists carry out large analytic Hessian calculations.

CHAPTER 1: GENERAL INTRODUCTION

Introduction

Computational quantum chemistry is a useful tool for many areas of science such as biochemistry, material science, catalysis, material design and biology. *Ab initio* calculations can provide reliable predictions of structures and various properties of chemical compounds. Computational quantum chemistry describes interactions of nuclei and electrons which define physical and chemical properties of molecules. These interactions are described by the Schrodinger equation. One limitation of *ab initio* calculations is that solving the Schrodinger equation even for a small size system is a challenging problem. Therefore an important advance in the effort to expand the size of systems that can be studied by quantum chemistry is the development of new algorithms and parallel quantum chemistry software.

The ultimate purpose of computational quantum chemistry is to apply these new algorithms and methods to real chemical problems. Quantum chemistry can help to find novel effective catalysts for important industrial reactions, new rocket fuels, or drugs to cure people. There is an unlimited number of applications where quantum chemistry can help.

Dissertation Organization

In this dissertation two important aspects of computational quantum chemistry have been addressed. First, new parallel quantum chemistry algorithms were developed to expand the size of systems that can be studied by quantum chemical methods. The new algorithms employ non standard approaches to achieve good performance results. Another important aspect is applications of quantum chemistry methods to find new catalysts for the bis-silylation reaction.

In the second chapter, titanium chloride (II) is investigated as a potential catalyst for the bis-silylation reaction of ethylene with hexachlorodisilane at different levels of theory. Bis-silylation is an important reaction for producing bis(silyl) compounds and new C-Si bonds, which can serve as monomers for silicon containing polymers and silicon carbides. Many of these organosilicon materials have desirable chemical and physical properties, such as thermal stability and the ability to store and transfer optical and electrical information.

In the second part of this dissertation new parallel algorithms are described. Chapter 3 addresses parallelization of the self consistent field procedure. The Hartree-Fock self-consistent field (SCF) method is the most common starting point for more accurate calculations.

In chapter 4, a new distributed data parallel CPHF step for an analytic Hessian algorithm is described. Analytic Hessian calculations are a fast and efficient method to study potential energy surfaces. The analytical Hessians can be many times faster and more accurate than numerical Hessian calculations, but the programming and parallelization of these codes is often a challenge. Non traditional approaches were utilized to parallelize CPHF.

Chapters 2 through 5 are papers either published, submitted to journals, or in preparation for submission to refereed journals. In chapter 6 general conclusions are presented for the dissertation.

Literature Review

In this section, a brief theoretical background of quantum chemistry concepts and standard quantum chemical methods are reviewed. These methods are used throughout in the dissertation.

Quantum chemistry is based on solving the time-independent Schrodinger equation

$$H\Psi = E\Psi \quad (1)$$

E is the energy of the system, Ψ is a wave function of the system. H is called the Hamiltonian operator.

$$H = K_{nuclei} + K_{el} + V_{el-el} + V_{nuclei-el} + V_{nuclei-nuclei} \quad (2)$$

The Hamiltonian for a molecule with N nuclei and n electrons consists of the kinetic energy of the nuclei K_{nuclei} ; the kinetic energy of the electrons K_{el} , the potential energy due to repulsion between electrons V_{el-el} , the potential energy due to the attraction between the electrons and the nuclei $V_{nuclei-el}$, the potential energy due to the electrostatic repulsion between the nuclei $V_{nuclei-nuclei}$. The Hamiltonian depends both on electron (r) and nuclear coordinates (R). Therefore the wave function Ψ also depends on r and R . It makes solving equation (1) challenging.

One of the most important assumptions in quantum chemistry in solving the Schrodinger equation is the Born-Oppenheimer approximation [1]. Ordinarily, the nuclei are moving much more slowly than electrons because nuclei are much heavier than electrons

(mass of protons and neutrons are about 1800 times heavier than electrons). Thus to a good approximation the electron motion can be separated from the nuclear motion

$$\Psi(r, R) \approx \Psi_{el}(r; R) \Psi_{nuclei}(R) \quad (3)$$

Therefore equation (1) can be solved in two steps. First solve for the electronic part:

$$H_{el} \Psi_{el} = E_{el}(R) \Psi_{el} \quad (4)$$

Since the nuclei are fixed, at each R one can add the nuclear repulsion $V_{nuclei-nuclei}(r)$ to $E_{el}(R)$:

$$U(R) = E_{el}(R) + V_{nuclei-nuclei} \quad (5)$$

Then,

$$H = H_{el} + U(R) \quad (6)$$

Then, the nuclear part of Schrodinger equation:

$$H \Psi_{nuclei} = E \Psi_{nuclei} \quad (7)$$

The total energy of the system is

$$E \approx E_{el} + E_{nuclei} \quad (8)$$

In general, the Born-Oppenheimer approximation correctly describes molecules in their ground electronic states.

In a typical quantum chemistry computation equation (4) is solved by using various approximations for a particular set of fixed nuclear coordinates R. The resulting electronic energy is summed with the nuclear repulsion energy. The entire procedure is repeated for each set of fixed nuclear coordinates R. The total energy as a function of coordinates R, $E(R)$, describes the potential energy surface (PES) for the molecular system.

To study potential energy surfaces of reactions and chemical systems two operations are performed most often: single energy + gradient calculations and Hessian

calculations. At stationary points on potential energy surfaces the gradient of the energy with respect to nuclear coordinates is zero. The stationary points can be minima, first order saddle points (transition states) or higher order saddle points. To distinguish them Hessian calculations are performed. The Hessian is the second derivative matrix of the total energy with respect to nuclear coordinates. The diagonalized Hessian provides harmonic normal modes and corresponding vibrational frequencies. Minima and n th order saddle points correspond to zero and n negative eigenvalues of the Hessian, respectively. Since the harmonic frequencies are the square roots of the Hessian eigenvalues, a negative eigenvalue corresponds to an imaginary frequency. The parallelization of analytical Hessian is addressed in chapter 4.

The transition states are typically connected by two minima which correspond to reactants and products. The difference in energy between reactant and transition state is the barrier height. The transition states can be connected with minima using the intrinsic reaction coordinate (IRC) method [2]. In the IRC the minimum energy path or reaction path is obtained in mass weighted Cartesian coordinates.

To solve equation (4) a number of methods are utilized. All of these methods are approximate, since an exact solution can be found only for one electron molecules. The simplest method is the Hartree-Fock method, in which the n -electron Schrodinger equation is replaced by a series of one-electron equations, called the Hartree-Fock equations:

$$F_i \psi_i = \epsilon_i \psi_i \quad (9)$$

where F_i is the one electron Fock operator; ψ_i is the one electron molecular orbital; ϵ_i is the one electron molecular orbital energy; $i = 1, \dots, n$ where n is the number of one electron molecular orbitals.

The total wavefunction is an antisymmetrized product of molecular orbitals which can be represented by Slater determinants. Each molecular orbital is a product of a spatial function and a spin function.

The molecular orbitals can be expanded as linear combinations of atomic orbitals χ_k .

$$\psi_i = \sum_k C_{ik} \chi_k \quad (10)$$

After atomic orbitals are introduced the Hartree-Fock equations (1.6) can be rewritten as a set of algebraic Hartree-Fock-Roothaan equations [3]:

$$FC = SC\varepsilon \quad (11)$$

where C is the matrix of expansion coefficients introduced in equation (10); F is the Fock matrix; S is the overlap matrix and ε is the matrix of orbital energies. The equations must be solved iteratively because the Fock matrix is a function of the expansion coefficients. The procedure is commonly called the self consistent field (SCF) method [4]. The parallelization of the SCF method is addressed in chapter 3.

In the Hartree-Fock operator in equation (9) the electron – electron repulsion is substituted by an average potential experienced by the i th electron due to the presence of the other electrons. Therefore the Hartree-Fock method does not take into account the correlation of electron motions. Thus the Hartree-Fock method often produces incorrect energies especially for non equilibrium structures for which electron correlation is especially important.

The solution of the Hartree-Fock equations (9) provides a set of one electron molecular orbitals ψ_i and one electron molecular orbital energies ε_i . This is the variationally best approximation to the ground state, of the single determinant form. However, it is only one of many possible determinants that could be formed from M

electrons in N molecular orbitals. In the full configuration interaction (FCI) method [5] all possible determinants are considered and coefficients A_k are optimized variationally in equation:

$$\Psi_{FCI} = \sum_k A_k \Psi_k \quad (12)$$

The FCI wavefunction is the exact solution of the Shrodinger equation for a given atomic basis set. If the basis set is incomplete then the FCI energy E_{exact} is the exact energy of the system for the given basis. The difference in energy between E_{exact} and Hartree-Fock in the complete basis set limit is called the correlation energy

$$E_{\text{corr}} = E_{\text{exact}} - E_{\text{hf}} \quad (13)$$

Unfortunately, basis sets are finite and the FCI method is very expensive. Even for small systems the number of determinants in FCI can be extremely large. There are several quantum chemistry methods to recover the correlation energy at a fraction of the FCI cost with moderate basis sets. The correlation energy is often divided into two types. "Static" correlation ensures that a correct zeroth order wavefunction is employed. "Dynamic" correlation corrects the interactions between electrons in close proximity to each other. The HF method is most commonly used to provide the zeroth order wavefunction. When the HF method is not appropriate, as in the case of diradicals, the alternative is multiconfigurational SCF (MCSCF) method. In the MCSCF method [6], the wavefunction is a linear combination of a subset of FCI configurations. The complete active space SCF (CASSCF) method [7] is the most widely used version of MCSCF. In CASSCF the most chemically important orbitals and electrons are selected. All possible configurations are generated by distributing these selected electrons in the selected orbitals. The CASSCF wavefunction is obtained by optimizing both configuration

coefficients and orbital expansion coefficients. The parallelization of the CASSCF method is addressed in chapter 5.

The simplest method to recover dynamic correlation energy is perturbation theory. The most popular type is Moller-Plesset perturbation theory. The perturbation is defined as the difference between the exact Hamiltonian and the sum of one-electron Fock operators. The perturbation expansion is most often truncated at the second order term, referred to as MP2 [8].

Another widely used method is the coupled cluster method [9]. The excitation configurations are generated by using an exponential excitation operator to produce single (CCS), double (CCSD) and so on excitations.

References

1. J. C. Tully, "*Dynamics of Molecular Collisions*", edited by W. H. Miller, pages 217-267. Plenum, New York, 1976.
2. (a) C.Gonzales, H.B.Schlegel, *J.Chem.Phys.* 90, 2154-2161(1989); (b) K.K.Baldrige, M.S.Gordon, R.Steckler, D.G.Truhlar, *J.Phys.Chem.* 93, 5107-5119 (1989).
3. C.C. Roothaan, *J. Rev. Mod. Phys.* 1951, 23, 69.
4. J.Almlof, K.Faegri, K.Korsell, *J.Comput.Chem.* 3, 385-399 (1982).
5. J.Ivanic, K.Ruedenberg *Theoret.Chem.Acc.* 106, 339-351(2001).
6. M.W. Schmidt and M.S. Gordon, *Ann. Rev. Phys. Chem.* 49, 233 (1998).
7. B.O.Roos, in "*Advances in Chemical Physics*", vol.69, edited by K.P.Lawley, Wiley Interscience, New York, 1987, pp 339-445.

8. (a) J.A.Pople, J.S.Binkley, R.Seeger, *Int. J. Quantum Chem.* S10, 1-19(1976); (b) M.J.Frisch, M.Head-Gordon, J.A.Pople, *Chem.Phys.Lett.* 166, 275-280(1990).
9. P. Piecuch, S.A. Kucharski, K. Kowalski, and M. Musial, *Comput. Phys. Comm.*, in press (2002).

**CHAPTER 2: A THEORETICAL STUDY OF THE
BIS-SILYLATION OF ETHYLENE CATALYZED BY
TITANIUM DICHLORIDE**

A paper submitted for publication to
Journal of American Chemical Society

Yuri Alexeev and Mark S. Gordon

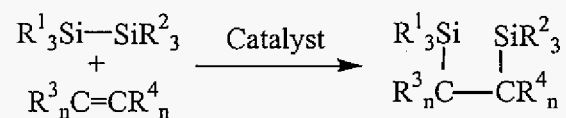
Abstract:

Titanium dichloride was investigated as a potential catalyst for the bis-silylation reaction of ethylene with hexachlorodisilane. *Ab initio* electronic structure calculations at the restricted Hartree-Fock (RHF), density functional theory (DFT), second order perturbation theory (MP2), and couple cluster (CCSD) levels of theory were used to find optimized structures, saddle points, and minimum energy paths that connect them. The reaction was found to have a net zero barrier at the DFT, MP2 and CCSD levels of theory. Dynamic correlation is found to be important for this reaction.

Introduction

The bis-silylation reaction [1] is an important process for producing bis(silyl) compounds and new C-Si bonds, which can serve as monomers for silicon containing polymers and silicon carbides. Many of these organosilicon materials have desirable chemical and physical properties, such as thermal stability and the ability to store and transfer optical and electrical information [2,3]. However, there is a lack of quantum chemical calculations for the study of the effect of catalysis on the bis-silylation reaction. Such calculations can potentially lead to the development of new catalysts. In this paper, TiCl_2 is proposed as an effective catalyst for the bis-silylation reaction. This is supported by a series of quantum chemical calculations at different levels of theory.

The bis-silylation reaction is a method to add an Si-Si bond across a C-C double or triple bond. The general reaction for Si-Si addition to a double bond can be written



Experimental and theoretical studies of this reaction in the absence of a catalyst suggest that the reaction has a high activation barrier. The predicted barrier height for the addition of disilane to ethylene is approximately 50 kcal/mol [4]. So some catalyst is needed to achieve high yields. A number of catalysts have been studied since 1972 when Okinoshima et al. carried out the first successful double silylation of 1,3-butadienes using Ni phosphine complexes as catalysts [5]. Later Okinoshima [6], Watanabe[7,8,9], and others discovered that Rh, Ni, Pt, and Pd phosphine complexes can be used to add substituted disilanes across various unsaturated acetylene and ethylene derivatives. It was found that complexes such as $\text{M}(\text{PPh}_3)_4$ and $\text{MCl}_2(\text{PPh}_3)_2$, where M is Pt or Pd are the

most efficient catalysts. Bottoni et al. [10] used density functional theory (DFT) with the B3LYP functional [11] to study the bis-silylation reaction of acetylene with disilane, $\text{H}_3\text{Si-SiH}_3$, in the presence of $\text{Pd}(\text{PH}_3)_2$. $\text{Pd}(\text{PH}_3)_2$ was used to emulate $\text{Pd}(\text{PPh}_3)_2$, $\text{Pd}(\text{PET}_3)_2$, and other catalysts often used in experimental studies. The reaction net barrier was found to be 18 kcal/mol.

A theoretical study of $\text{Pt}(\text{PH}_3)_2$ catalyzed bis-silylation and hydrosilylation (Chalk-Harrod and modified Chalk-Harrod mechanisms) of alkenes was recently performed by Sakaki et al. [12,13] These authors used second order perturbation theory (MP2) [12], fourth order perturbation theory (MP4SDQ) [13], and doubles coupled cluster theory (CCD) [14] to study the reactions. The net reaction barrier is predicted to be 19 kcal/mol in the bis-silylation reaction and 5 kcal/mol in the Chalk-Harrod mechanism of the hydrosilylation reaction.

It is well known that earlier transition metals, such as Ti and Zr, complexes exhibit catalytic properties. In the Ziegler-Natta polymerization reaction, the commonly used catalysts are $\text{MCl}_4\text{-AlR}_3$, MR_2 where M is Ti or Zr [15]. The first step in the currently accepted mechanism is formation of a metal-alkyl-olefin complex. The addition of $\text{Cl}_2\text{TiCH}_3^+$ to C_2H_4 was studied recently by Bernardi et al. [16] with DFT using the B3LYP functional. The reaction requires no net barrier. A π -complex intermediate is lower than reactants by 38 kcal/mol.

Titanocene (TiCp_2 where Cp=cyclopentadienyl) and zirconocene (ZrCp_2) were recently reported to be efficient catalysts for silylation by Terao et al. [17] and Harrod et al. In particular, silylation of isoprene with chlorotriethylsilane proceeds with 91% yield at 0°C in the presence of Cp_2TiCl_2 and BuMgCl in THF solution. The double silylation of p-chlorostyrene by Me_2PhSiCl in the presence of BuMgCl and Cp_2TiCl_2 in THF solution

under the same conditions gives a 72% yield.

Bode, Day, and Gordon [18] demonstrated that TiCl_2 is an efficient catalyst for the hydrosilation reaction. The reaction was studied using restricted Hartree-Fock (RHF), second order perturbation theory (MP2), and coupled cluster theory (CCSD(T)). All levels of theory predict that the reaction has no net barrier. The highest CCSD(T) energy is 31 kcal/mol below reactants.

The purpose of this paper is to present *ab initio* calculations on the steps involved in a proposed mechanism. In the model reaction TiCl_2 is used as the catalyst for the bis-silylation reaction of ethylene with hexachlorodisilane. This choice of reactants allows us to study this reaction at reliable levels of theory without compromising accuracy. Although TiX_2 has not previously been proposed as a catalyst for bis-silylation, TiX_2 is a promising model catalyst based on previous theoretical and experimental studies.

Computational methods

All calculations performed for this paper were carried out using the GAMESS program [19], and figures were generated using the MacMolPlt program [20]. The basis set used in the calculations was the SBKJC effective core potential (ECP) basis [21] on Si, Cl, and Ti. One d-type polarization function was added on each Si and Cl atom [22]. The H and C atom basis used was 6-31G (d,p). This basis set was compared previously [23] with an all-electron triple- ζ plus polarization basis set. It was found that the difference in relative energies between these is less than 0.5 kcal/mol, consistent with the present work.

The RHF, DFT, MP2, and CCSD methods were used to study the bis-silylation reaction. Preliminary geometry calculations were carried out using RHF. These geometries were then used as starting geometries for the MP2 calculations. DFT and CCSD calculations were carried out at selected transition states with the highest energy barriers. These methods predict results that are similar to those from MP2. Only the RHF method predicts a nonzero net barrier.

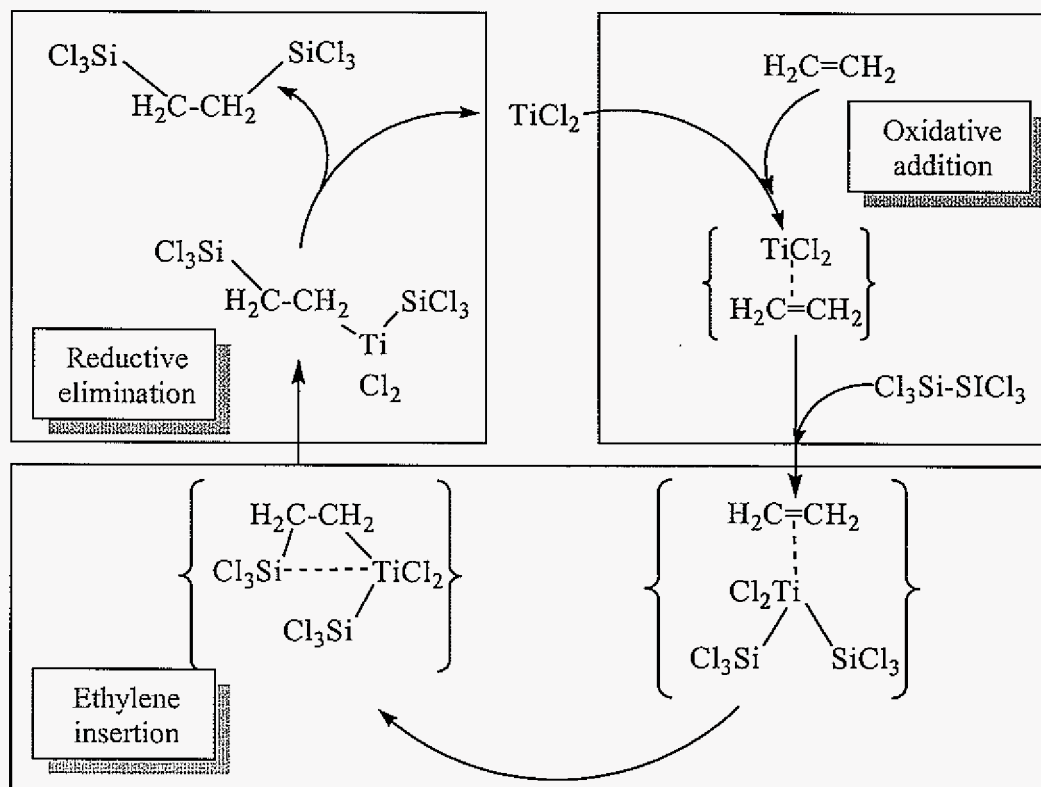
All geometries and energies for reactants, products, and all stationary points on the reaction path presented in this paper are at the MP2 level of theory. Each stationary point was confirmed by computing the Hessian (matrix of second derivatives of energy). The diagonalized Hessian provides harmonic normal modes and corresponding vibrational frequencies. Transition states and minima are indicated by one and no imaginary mode, respectively. The calculated frequencies were also used to compute zero point vibrational energies (ZPE).

Finally, each confirmed transition state has been connected to reactants and products using the Gonzales-Schlegel second-order intrinsic reaction coordinate (IRC) method with a step size of $0.3 \text{ amu}^{1/2} \cdot \text{bohr}$.

Results and discussion

The previous studies established that dynamic correlation is important for reactions in which Ti is a catalyst, so the MP2 and CCSD methods were employed. MP2 natural orbital occupation numbers (NOONs) were calculated and inspected at each stationary point. The largest observed deviation from the HF values of 2.0 and 0.0 for occupied and virtual orbitals respectively is 0.08, suggesting that there is little multiconfigurational

character in the wavefunction [24]. The MP2 geometries and energies are presented in all Figures and Tables.



Scheme 1. Catalytic cycle for double silylation of ethylene with hexachlorodisilane

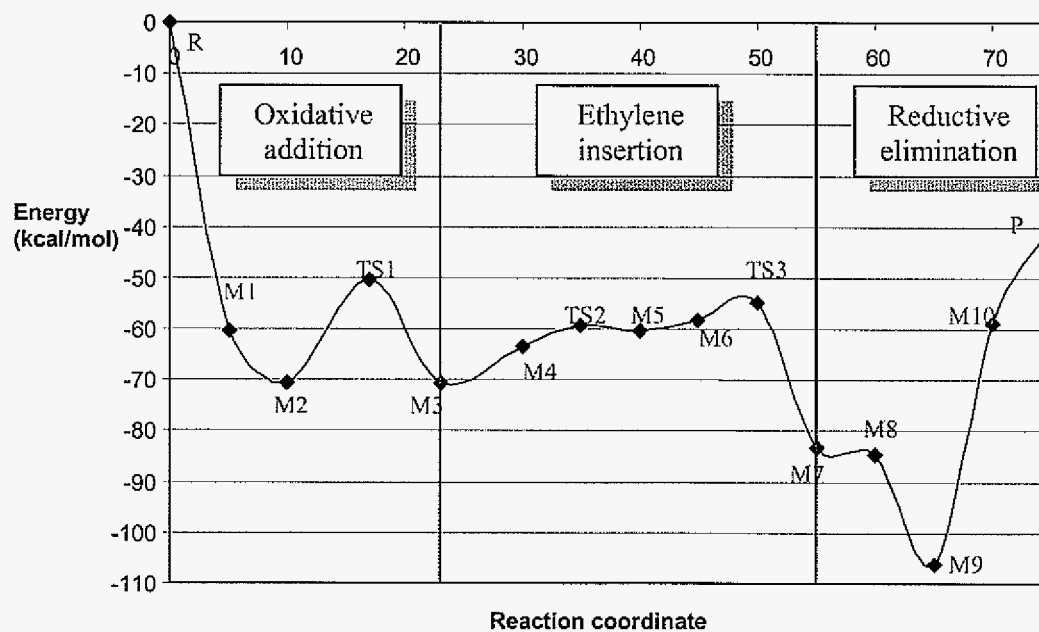


Figure 1. Minimum energy reaction path

Table 1. MP2 relative energies with ZPE corrections in kcal/mol.

Structure	MP2 Relative Energy	MP2 + MP2 ZPE Relative Energy
Oxidative addition		
R	0	0
M1	-60.5	-59.6
M2	-70.6	-68.9
TS1	-50.5	-48.2
M3	-70.5	-67.9
Ethylene insertion		
M4	-63.3	-60.9
TS2	-59.4	-57.1
M5	-60.1	-57.9
M6	-58.4	-55.9
TS3	-54.7	-52.3
M7	-83.3	-80.7
Reductive elimination		
M8	-84.6	-82.2
M9	-106.4	-102.9
M10	-59.0	-55.4
P	-39.1	-36.2

In the commonly accepted mechanism for bis-silylation of alkenes and alkynes [25, 26, 27] the first step is oxidative addition of the catalyst to the disilane; then the alkene or alkyne is inserted into the metal-silyl bond. The final stage is reductive elimination and the regeneration of the catalyst. In the experiments no intermediates have been detected in the oxidative addition of Pt(0) to disilane, presumably confirming the first step [28] in this mechanism. Based on the current calculation with TiCl_2 an alternative mechanism is presented in a Scheme 1, in which the first step is coordination of the catalyst to the ethylene not the disilane to form an initial complex. The complex interacts with disilane. Subsequently, ethylene is inserted into the Ti-Si bond to form product after reductive elimination of the TiCl_2 catalyst.

The overall reaction path energetics are demonstrated in Figure 1. The reactants are labeled as R, minima as MX (where X is an integer number), transition states as TSX, and products as P. Minima M3 and M4, M5 and M6, M7 and M8, M8 and M9, M9 and M10 were connected by using linear least motion paths and constrained optimization techniques. The highest point on a constrained optimization path is an upper bound to the energy barrier for that path. Each step will be discussed in detail in the following sections. The relative MP2 energies presented in Figure 1 do not include vibrational ZPE corrections. In the first column of Table 1 relative MP2 electronic energies corresponding to the data in Figure 1 are listed. ZPE corrected relative energies are presented in the last column of Table 1.

In the following sections we will discuss in detail the potential energy surface (PES) of the proposed mechanism: (1) oxidative addition, (2) ethylene insertion into the Ti-Si bond, and (3) reductive elimination. For each step a figure with detailed geometry information for the stationary points is presented, with bond lengths shown in angstroms. For transition states the magnitude of the imaginary frequency is included.

The MP2 total energies and MP2 total ZPE corrected energies for each stationary point in Figure 1 and Table 1 are available in supplementary material Table S1. The Cartesian coordinates of all geometries can be found in Table S2.

1. Oxidative addition

Two scenarios have been investigated. The catalyst TiCl_2 can attack the C-C bond first as shown in Figure 2 or TiCl_2 can initially attack the Si-Si bond as shown in Figure 3.

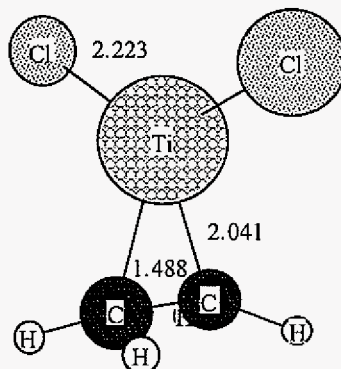
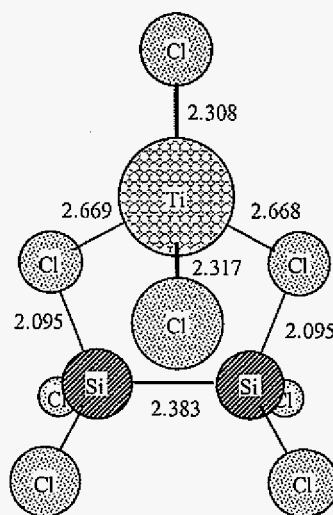
Figure 2. $\text{TiCl}_2\text{-C}_2\text{H}_4$ complexFigure 3. $\text{TiCl}_2\text{-Si}_2\text{Cl}_6$ complex

Table 2. MP2 relative energies with ZPE corrections in kcal/mol

Structure	MP2 Relative Energy	MP2 + MP2 ZPE Relative Energy
Reactants	0	0
$\text{TiCl}_2\text{-C}_2\text{H}_4 + \text{Si}_2\text{Cl}_6$	-60.5	-59.6
$\text{TiCl}_2\text{-Si}_2\text{Cl}_6 + \text{C}_2\text{H}_4$	-17.8	-17.1

The optimized structures in Figures 2 and 3 have a large difference in energy relative to the energy of the initial reactants: $\text{TiCl}_2\text{-C}_2\text{H}_4$ is lower in energy than $\text{TiCl}_2\text{-Si}_2\text{Cl}_6$ by 42.7 kcal/mol as can be seen in Table 2. In the first mechanism, the $\text{TiCl}_2\text{-C}_2\text{H}_4$ complex shown in Figure 2 reacts with Si_2Cl_6 in a series of steps ultimately leading to products. In the second mechanism, the $\text{TiCl}_2\text{-Si}_2\text{Cl}_6$ complex shown in Figure 3 reacts with C_2H_4 in a series of steps that converges to the minimum M2 in Figure 4. Therefore, the second mechanism leads to the same reaction path as the first mechanism. Since (a) the first mechanism leads to a much lower energy initial intermediate M1, (b) both mechanisms proceed through the M2 intermediate shown in Figure 4, and (c) the highest point on each path is lower in energy than the separated reactants, only the first mechanism is presented in detail here (see Scheme 1 and Figure 4).

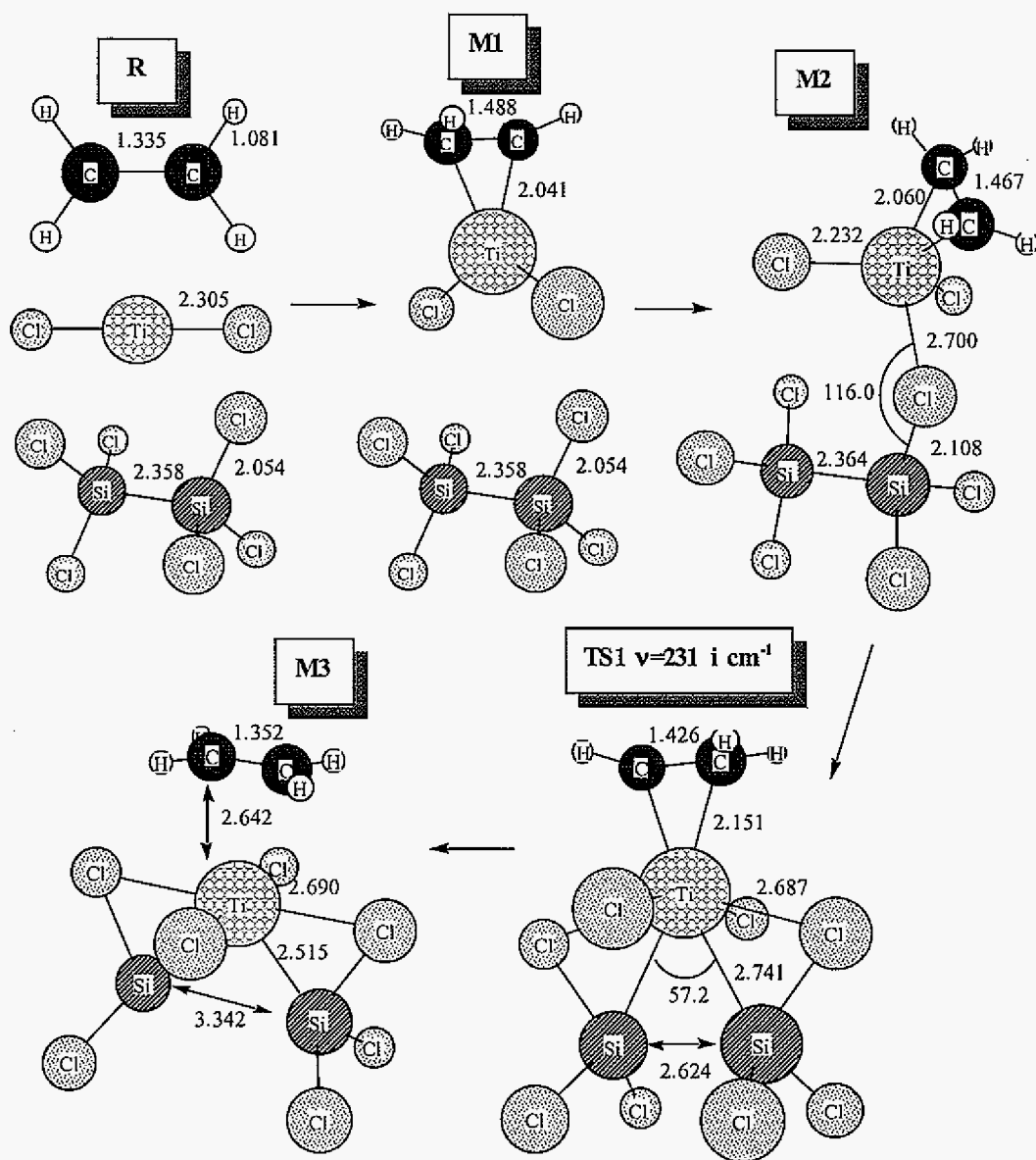


Figure 4. Oxidative addition

The MP2 structures for the oxidative addition step are presented in Figure 4. After TiCl_2 forms a complex with ethylene in M1, Ti in this complex interacts with a Cl from disilane, leading to a new intermediate M2. A transition state TS1 connects M2 with M3, in which Ti has broken the hexachlorodisilane Si-Si bond. TS1 and M3 both have C_{2v} symmetry, but M2 has C_1 symmetry. Therefore, a bifurcation [29] occurs along the

reaction path that connects TS1 with M2, since the valley-ridge inflection point does not coincide with transition state TS1. The valley-ridge inflection point was found by first performing a series of Hessian calculations. Then, the imaginary mode was offset by 5% and the IRC run was resumed to find the correct minimum M2. TS1 is the highest point on the minimum energy reaction path. Based on bond and valence analysis, Ti in TS1 forms 8 partial bonds with bond orders varying from 0.3 to 1.3 (Figure 5). Ti forms strong bonds with two chlorines perpendicular to the Si-Ti-Si plane and with the two carbon atoms.

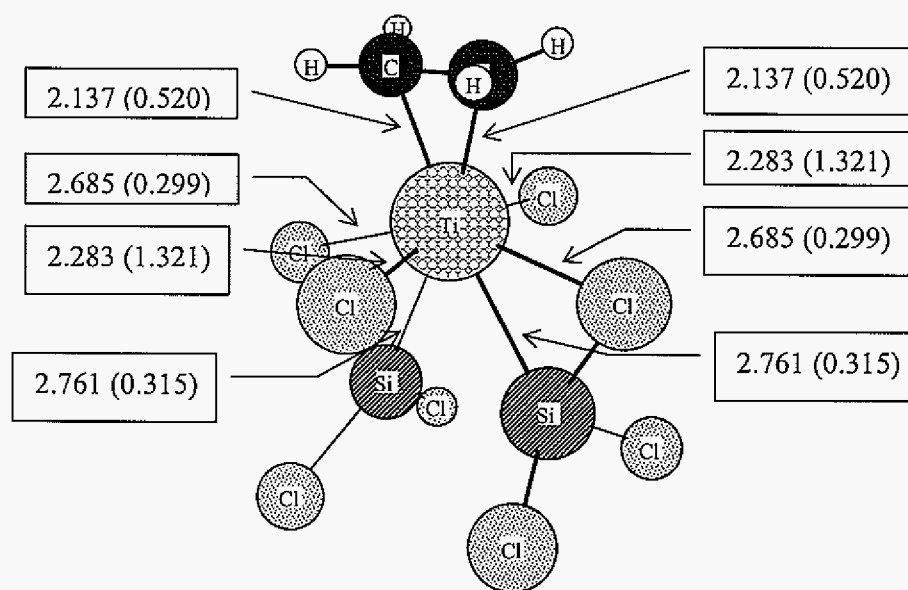


Figure 5. Transition state 1 (TS1). Bond distance in Å (bond order)

It is interesting to track the C-C bond length changes in the oxidative addition step. In M1 the C-C length has single bond character after ethylene has interacted with Ti. Later after Ti has inserted into the Si-Si bond in M3, the C-C distance has decreased back to a distance very close to that in ethylene.

The activation energy leading from M2 to TS1 to MS is 20 kcal/mol. This is the largest activation energy along the reaction path. The energy of M2 relative to M3 is 1 kcal/mol after the ZPE correction is applied. It was expected that the Ti insertion into the Si-Si bond would have one of the highest activation energies, but it is still 50 kcal/mol below the energy of the reactants.

2. Ethylene insertion

The first step in the ethylene insertion (see Scheme 1 and Figure 7) is to position the ethylene molecule just above the Ti-Si bond. The potential energy surface in this region is shallow, because the ethylene molecule can essentially undergo free rotation. Therefore, a linear least motion path [30] and constrained optimization techniques [31] were employed to connect the minimum M3 with M4, and M5 with M6 (Scheme 1).

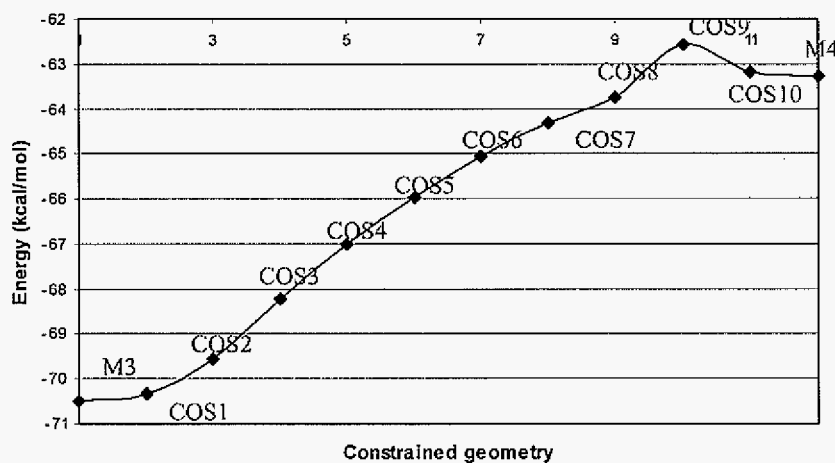


Figure 6. Constrained optimization path connecting M3 and M4, COSX are constrained optimized structures

An example of how the constrained optimization technique is employed is shown in Figure 6. The energies of 10 constrained optimized structures from COS1 to COS10 connecting the M3 and M4 minima are plotted. The estimated activation barrier for the reaction at COS9 is 7.9 kcal/mol excluding the ZPE correction. The difference in energy between M3 and M4 is 7.0 kcal/mol. The geometry and location of COS9 on the PES is consistent with the Hammond postulate: COS9 is structurally close to the minimum M4 which is 7.2 kcal/mol higher in energy than M3. The primary effect of the M3→M4 rearrangement is to move one ethylene C closer to its Si partner. The M4 and M5 minima in Figure 7 are connected via a transition state TS2 with an activation energy of 4 kcal/mol (3 kcal/mol with the ZPE correction). In M5 a rotation about the Ti-Si bond has occurred, in order to further facilitate the formation of the new C-Si bond. Minima M5 and M6 are connected in a manner similar to that used to connect minima M3 and M4, with an activation barrier of approximately 2 kcal/mol.

In the second part of the ethylene insertion reaction (Figure 7), the transition state TS3 connects minima M6 and M7. The barrier height of this reaction is 3.6 kcal/mol. TS3 is the second highest stationary point. The energies of M6 and M7 relative to reactants are -55.9 and -80.7 respectively.

In M6, the C-C bond is already slightly stretched from 1.335Å in ethylene to 1.359Å, the Ti-Si bond is stretched from 2.515Å in M3 to 2.753Å in M6. In transition state TS3, a four-membered ring is formed that consists of Ti, two carbons, and Si. The Ti-Si bond is stretched even further to 2.949Å in TS3. The four-membered ring is opened via breaking a Ti-Si bond to give the minimum M7. In M7 the Ti-Si bond is broken, since the distance is 3.680Å, and a C-Si bond is finally formed. Therefore, M7 is the first time the Si-Ti-C-C-Si chain is formed.

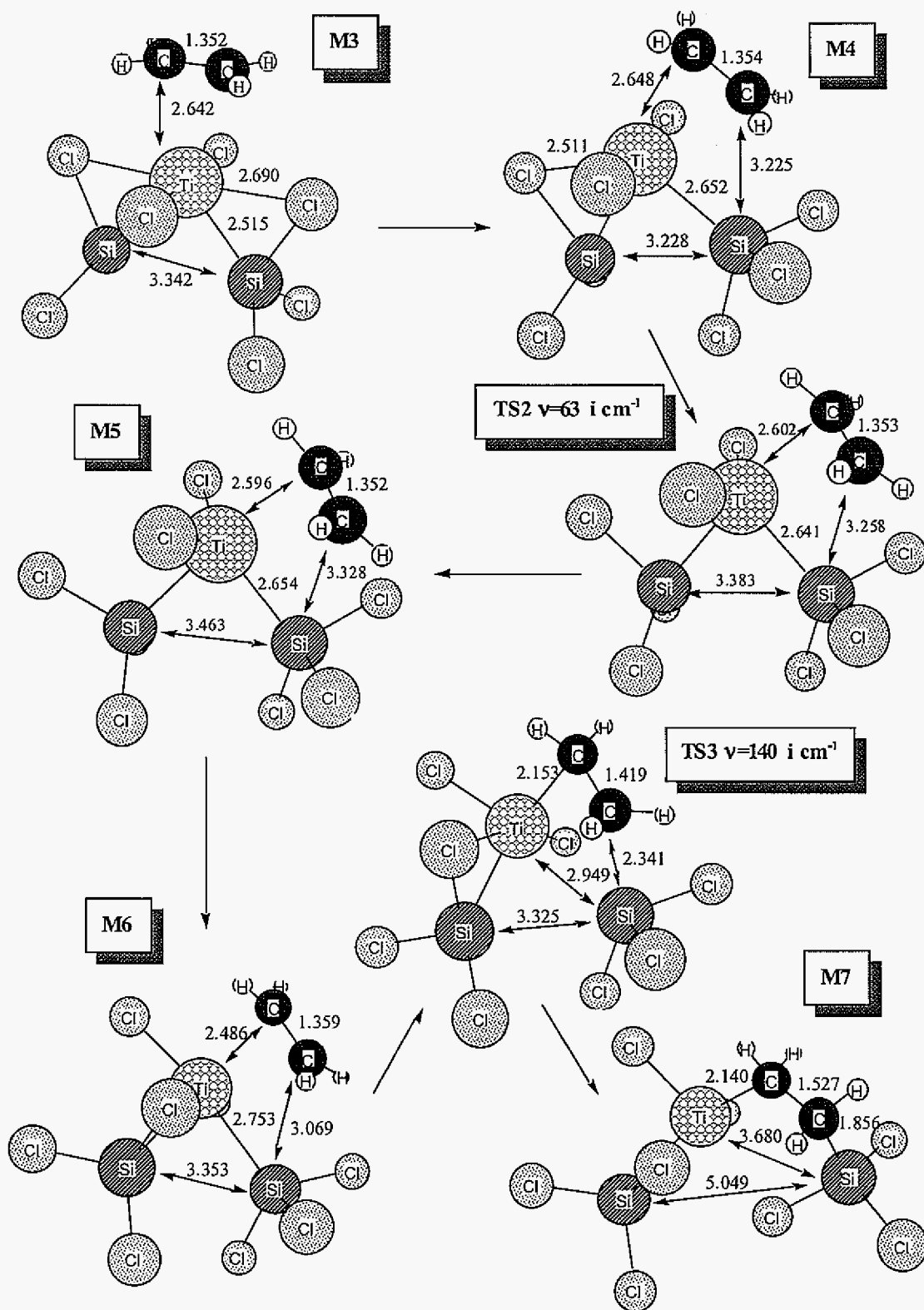


Figure 7. Ethylene insertion

3. Reductive elimination

The final phase in the overall mechanism is the regeneration of the catalyst and formation of the final product. The geometries of all stationary points in this step are shown in Figure 8. The reaction proceeds in four steps. In the first step, the SiCl_3 group attached to TiCl_2 moves into the staggered position (M8) relative to the Ti-C bond. Since internal rotations usually require little activation energy, the constrained optimization technique was utilized to connect M7 and M8. The estimated activation energy required to connect minima M7 and M8 is 3.4 kcal/mol. The energies of M7 and M8 relative to the reactants are -80.7 and -82.2, respectively, including the ZPE correction.

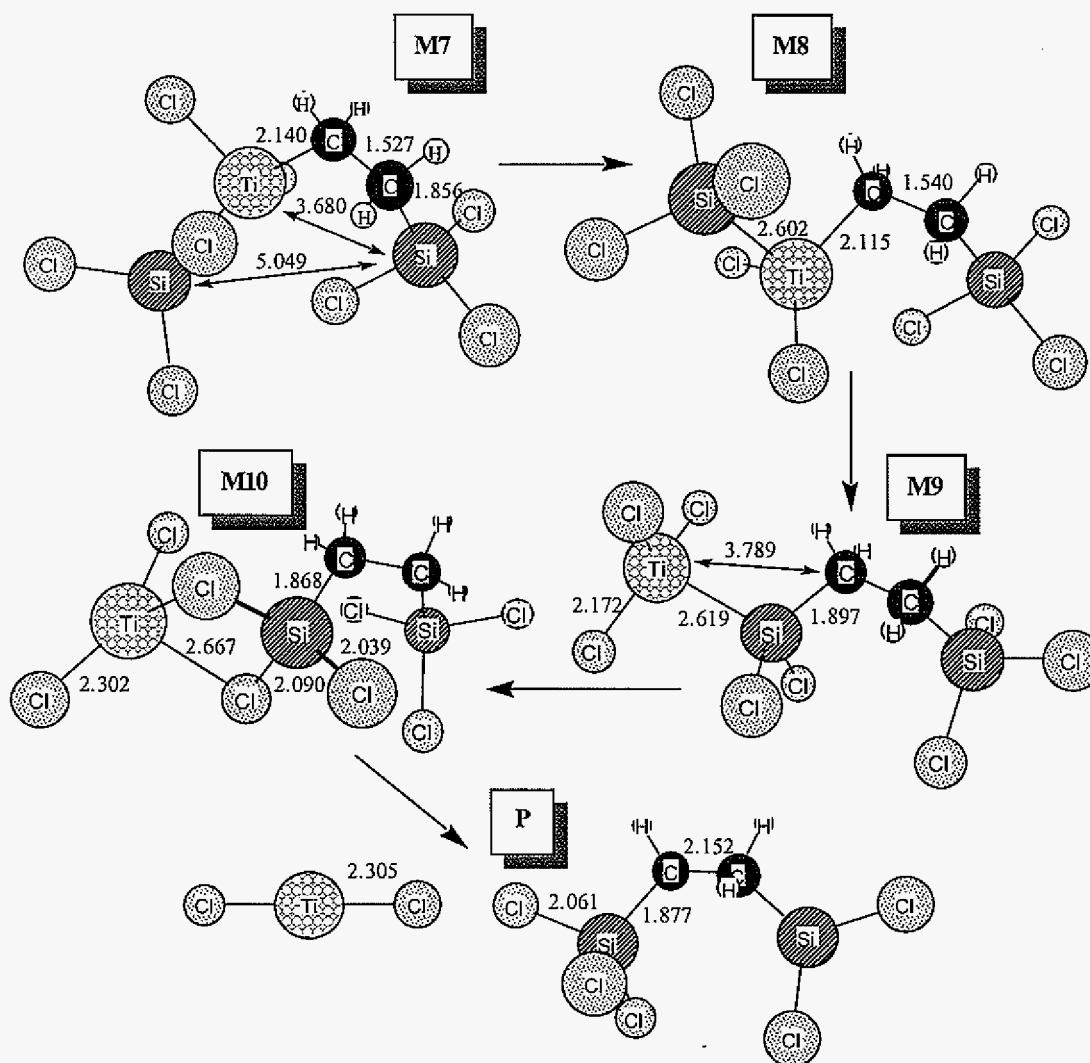


Figure 8. Reductive elimination

In the next step $M8 \rightarrow M9$, Ti and Si interchange their positions. M9 is the first species in which the Ti-Si-C-C-Si linkage appears. To connect minima M8 and M9 the constrained optimization technique was used. The estimated barrier height is 5.7 kcal/mol. The intermediate M9 is the global minimum on the reaction path. Including the ZPE correction, M9 is 102.9 kcal/mol below the energy of separated reactants.

In the last two steps the catalyst is regenerated. First, one Cl atom on the $TiCl_3$ group migrates to a Si atom. Then, a new minimum M10 is formed in which Ti, Si, and two

chlorines form a four-membered ring. Two chlorines are shared between Ti and Si. Note that Ti is not connected to either Si in M10. The reaction M9→M10 is endothermic, with the top of the constrained optimization path being 2.9 kcal/mol above M10. The last step removes the catalyst from the system. No net transition state is expected for this step. The reverse reaction for adding TiCl₂ to product proceeds readily without any barrier. The product is the cis conformer of 1,2-bis-chlorosilyl ethane. The energies of M10 and the product P relative to reactants are -55.4 and -36.2 kcal/mol respectively with the ZPE correction.

Conclusions

The overall reaction is a highly exothermic barrierless process. The products are 36.2 kcal/mol lower in energy than the combined energies of the separated reactants after the ZPE correction is applied to the MP2 energy. The TiCl₂ catalyst removes a 50 kcal/mol activation energy barrier required for the reaction without the catalyst. The first step is interaction of TiCl₂ with ethylene to form an intermediate that is 60 kcal/mol below the energy of the reactants. This is the driving force for the entire reaction. After that TiCl₂ easily cleaves the Si-Si bond with modest a 20 kcal/mol activation energy. The transition state for this step is the highest point except products for the entire reaction, and it is still 50 kcal/mol below the energy of the reactants. Dynamic correlation plays a significant role because RHF calculations indicate that the net barrier for the catalyzed reaction is 50 kcal/mol. There are also significant differences in relative energies and structures between RHF and MP2 levels of theory.

As discussed in several recent papers [32,33,34], Ti is an electron deficient atom, in much the same manner as B. This means Ti readily forms additional bonds beyond “usual” four. As noted before for the hydrosilation reaction this makes divalent Ti a particularly effective catalyst, since the initial steps in which Ti binds to one of the reactants is especially facile with a large energy decrease. This energy decrease is sufficient to ensure that the activation barriers for all subsequent steps are well below the energy of the reactants. Consequently, the reaction proceeds easily.

Acknowledgement

The calculations in this work were performed on an IBM workstation cluster made possible by grants from IBM in the form of a Shared University Research grant, the Department of Energy, and a DURIP grant from the Air Force Office of Scientific Research.

The research reported here was made possible by a grant from the Air Force Office of Scientific Research.

References

1. R.G. Jones, W. Ando, J. Chojnowski, *Silicon-Containing Polymers: The Science and Technology of Their Synthesis and Applications*; Publisher: Kluwer Academic Publishers, 2000.
2. (a) H. H. Lee, *Fundamentals of Microelectronic Processing*; McGraw-Hill: New York, 1990. (b) J. Mort, F. Jansen, Eds.; *Plasma Deposited Thin Films*; CRC Press: Boca Raton,

- FL, 1986. (c) T. Sueta, T. Okoshi, *Ultrafast and Ultra-Parallel Optoelectronics*; John Wiley & Sons: Tokyo, 1995. (d) P. N. Prasad, D. R. Ulrich, *Nonlinear Optical and Electroactive Polymers*; Plenum Press: New York, 1988.
3. (a) D. C. Bradley. *Chem. Rev.* 1989, 89, 1317. (b) L. L. Hench, J. K. West, *Chem. Rev.* 1990, 90, 33.
4. F. Raaii, M.S. Gordon; *J. Phys. Chem. A*, 1998, 102 (24), 4666 -4668.
5. H. Okinoshima, K. Yamamoto, M. Kumada, *J. Am. Chem. Soc.* 1972, vol. 94, 9263.
6. H. Okinoshima, K. Yamamoto, M. Kumada, *J. Organomet. Chem.* 1975, 86, C27.
7. H. Watanabe, M. Kobayashi, M. Saito, Y. Nagai, *J. Organom. Chem.* 1981, 149, vol. 216.
8. H. Watanabe, M. Saito, N. Sutou, K. Kishimoto, J. Inose, Y. Nagai, *J. Organom. Chem.* 1982, 225, vol. 343.
9. H. Watanabe, T. Kitahara, T. Motegi, Y. Nagai, *J. Organom. Chem.* 1977, 215, vol. 139.
10. A. Bottoni, A.P. Higuero, G.P. Miscione, *J. Am. Chem. Soc.*, 124 (19), 5506 - 5513, 2002.
11. W. Koch, M. Holthausen, "A Chemist's Guide to Density Functional Theory." Wiley-VCH Wiley & Sons. Weinheim, 2000.
12. C. Moler, M. S. Plesset, *Phys. Rev.* 1934, 46, 618.
13. C.J. Cramer, "Essentials of Computational Chemistry", 2002, John Wiley & Sons, LTD
14. T. D. Crawford, H. F. Schaefer III, , *Reviews in Computational Chemistry*; 1996, Vol. 14, pp. 33-136.
15. (a) O. Novaro, E. Blaisten-Barojas, E. Clementi, G. Giunchi, M. E. Ruiz-Vizcaya, *J. Chem. Phys.* 1978, 68, 2337.; (b) H. Kawamura-Kuribayashi, N. Koga,

- K. Morokuma, *J. Am. Chem. Soc.* 1992, 114, 2359; (c) R. J. Meier, G. H. J. van Doremaele, S. Iarlori, F. Buda, *J. Am. Chem. Soc.* 1994, 116, 7274.
16. F. Bernardi, A. Bottoni, G. P. Miscione, *Organometallics*, 1998, 17(1); 16-24.
17. (a) J. Terao and N. Kambe, *Journal of Synthetic Organic Chemistry Japan*, 2001, 59(11), 1044-1051. (b) J. Terao, N. Kambe, and N. Sonoda, *Tetrahedron Lett.* 1998, 39, 9697. (c) J. Terao, K. Torii, K. Saito, N. Kambe, A. Baba, and N. Sonoda; *Angew. Chem., Int. Ed. Engl.* 1998, 37, 2653.
18. B. M. Bode, P. N. Day, M. S. Gordon, *J. Am. Chem. Soc.*; 1998; 120(7); 1552-1555.
19. M.W. Schmidt, K.K. Baldrige, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.J. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L. Windus, M. Dupuis, J.A. Montgomery, *J. Comput. Chem.* 14, 1347-1363, 1993
20. B. M. Bode, M. S. Gordon, *J. Molec. Graphics.*, 16, 133 (1999).
21. (a) For Si, Cl: W.J. Stevens, H. Basch, M. Krauss, *J. Chem Phys* 81: 6026, 1984; (b) For Ti: W.J. Stevens, H. Basch, M. Krauss, P. Jasien, *Can. J. Chem* 70: 612, 1992
22. The exponents used are: Si $\zeta_d = 0.364$, Cl $\zeta_d = 0.566$
23. B. Bode, M.S. Gordon, Volume 102, Issue 1-6, pp 366-376, *Theoretical Chemistry Accounts*, 1998
24. M.S. Gordon, M.W. Schmidt, G.M. Chaban, K.R. Glaesemann, W.J. Stevens, C. Gonzalez, *J. Chem. Phys.*, 110, 4199 (1999).
25. T. Hayashi, T. Kobayashi, A. M. Kawamoto, H. Yamashita, M. Tanaka, *Organometallics*; 1990 9; 280
26. M. Murakami, T. Yoshida, Y. Ito, *Organometallics*; 1994 13; 2900
27. S. Sakaki, M. Ogawa, Y. Musashi, *J. Organom. Chem.* 1997, 25-28, vol. 535.

28. M. J. Michalczyk, J. C. Calabrese, C. A. Recatto, M J. Fink; *J. Am. Chem. Soc.*; 1992; 114(20); 7955-7957. R. H. Heyn, T. D. Tilley; *J. Am. Chem. Soc.*; 1992; 114(5); 1917-1919.
29. K. Ruedenberg, P. Valtazanos, *Theoretica Chimica Acta*, 69, 281-307 (1986).
30. P. Pechukas, *J. Chem. Phys.* 64, 1516-1521 (1976)
31. J. Baker, A. Kessi, B. Delley, *J. Chem. Phys.* 105, 192-212(1996)
32. S.P. Webb, M.S. Gordon, *J. Am. Chem. Soc.*, 121, 2552 (1999).
33. S.P. Webb, M.S. Gordon, *J. Am. Chem. Soc.*, 120, 3846 (1998).
34. S.P. Webb, M.S. Gordon, *J. Am. Chem. Soc.*, 117, 7195 (1995).

CHAPTER 3: THE DISTRIBUTED DATA SCF

A paper published in and reprinted with permission from

Computer Physics Communication 143, 69 (2002)

© Copyright 2002, Elsevier Science, All rights reserved

Yuri Alexeev, Ricky A. Kendall, Mark S. Gordon

Abstract:

This paper describes a distributed data parallel SCF algorithm. The distinguishing features of this algorithm are: (a) columns of density and Fock matrices are distributed evenly among processors, (b) pair-wise dynamic load balancing is developed to achieve excellent load balance, (c) network communication time is minimized via careful analysis of data flow in the SCF algorithm. The developed performance models and benchmarking results illustrate good performance of the distributed data SCF algorithm.

Keywords:

Quantum chemistry, Cluster Computing, Self Consistent Field, Distributed Data Interface, Nonuniform Memory Access, Dynamic Load Balancing

1. Introduction

Quantum chemistry is a useful tool for many areas of science. *Ab initio* calculations provide reliable energetics and molecular properties for chemical reactions applicable to areas such as biochemistry, material science, catalysis, material design and others. One limitation of *ab initio* calculations is that they require significant computational resources that increase rapidly with the size of the molecular system. An important advance in the effort to expand the size of systems that can be studied by such methods is the development of parallel quantum chemistry software.

Recently, the definition of supercomputing broadened significantly with the introduction of cluster computing [1]. This technology has become increasingly popular in recent years. There are several advantages of cluster computing:

1. It can provide the computational power of supercomputers for a small fraction of the price.
2. One can use commodity parts.
3. Scalability can be attained in principle.
4. Installation is relatively simple.
5. One has local direct control of computational resources.

There are also some serious disadvantages of cluster computing. For example, the low hardware cost of clusters is accompanied by slower communication between nodes. In addition, parallel programming software is less robust for clusters. In general, not all parallel libraries are available or optimized for the cluster environment [2,3]. A good example is an IBM cluster of IBM RS/6000 dual processor Power 3 running AIX version 4.3.3, using Gigabit Ethernet to connect the nodes [4]. The MPI bandwidth utilizes less

than half of the TCP performance, since the MPI library is not optimized to handle Jumbo Frames [5]. However, this is likely to change as researchers in several laboratories develop robust libraries that are optimized for cluster environments [6,7,8,9,10].

Replicated data models, in which the data is replicated on each node, are straightforward to code, but limit the size of systems that can be calculated by the memory on each individual node. In quantum chemistry codes, one must deal with both two dimensional and four dimensional arrays. The size of each dimension N (total number of basis functions) can range from a small number to a few thousand. Most scalable algorithms focus on distributing the four dimensional arrays and replicating the two dimensional arrays. However, this is still limiting if one wants to study very large systems where $N \gg 1,000$.

The quantum chemistry package GAMESS [11] is a multi-functional code that performs a broad variety of electronic structure calculations. Over the past 10 years, several parts of the quantum chemistry package GAMESS have been made parallel; however, most of the functionality was implemented using the replicated data model. Recently, the distributed data interface (DDI) [7] was implemented in GAMESS, in order to perform second order perturbation theory energy and gradient calculations. The DDI/MP2 code scales linearly to 512 nodes on the Cray T3E [12].

In this paper a direct algorithm is discussed. Direct methods do not save data on disk thereby avoiding large storage problems. This eliminates input/output (I/O) bottlenecks associated with lack of local disk space on nodes and possible poor I/O performance.

The purpose of this paper is to present a scalable distributed data direct Fock builder, as this is the most important computational kernel of a SCF program. The initial guess and diagonalization will not be addressed in this paper. The Hartree-Fock self-

consistent field (SCF) method is the most common starting point for more accurate calculations. The current bottleneck in replicated data SCF (RDSCF) that is implemented in GAMESS [11] is that the density (D) and Fock (F) matrices are replicated on each node. Because D and F are two dimensional rather than four dimensional arrays, this only becomes an issue when one is interested in very large systems. Then one needs a distributed SCF. There have been several other efforts to develop distributed data SCF (DDSCF) codes [13,14,15]. The large latency and low bandwidth of clusters that constitute a primary development platform, however, limit performance and scalability of these algorithms. Because communication overhead degrades performance, one desires a ratio of computation time to communication time that is at least a few orders of magnitude to achieve good performance.

A major problem with most SCF algorithms is “irregular” data access and update patterns to D and F arrays since we intend to distribute them among nodes. The careful analysis of loop structures at the beginning of Section 5 illustrates how access and update of corresponding D and F elements can be organized. The algorithm presented in Section 5 illustrates the distribution of D and F matrices, loop structure, and dynamic load balancing tasks. The performance analysis in Section 5 shows that the communication cost is much less than the computation cost. Hence, we expect that the algorithm will be efficient and scalable on most parallel machines. This is explicitly demonstrated by benchmark timings presented in Section 6.

2. Tools and platforms

PC clusters are popular since small research groups and departments usually don't have sufficient resources to purchase large computer systems. Currently, the most common types of clusters are PC clusters connected by a commodity network such as Fast Ethernet. Hence, the code described here has been developed and tested on a PC cluster [16] running Red Hat Linux 5.2 that consists of 16 Pentium II 400Mhz nodes connected by Fast Ethernet. Each node has 1 processor, 512 MB of memory, 8 GB local disk, and total aggregate distributed memory is about 8 GB.

Currently, the message-passing model is widely used because it has been ported to a large variety of platforms. However, complex scientific applications often require a sophisticated distribution of data with irregular access patterns [17]. The complexity of programming within the limits of a message-passing model can be too high for such applications. An alternative is a model based on the shared memory paradigm. However, until recently only a few vendor specific libraries have been available for shared memory computers. These include SHMEM on Cray and SGI platforms, Fujitsu MPlib, and IBM LAPI on the IBM SP platform. The need for a portable shared-memory library resulted in the development of OpenMP [3] libraries and partially in the development of the MPI-2 [18] library. The uncertainty with regard to the wide availability and implementation of MPI-2 has resulted in the development of alternative portable libraries, such as GA/ARMCI [19], DDI (Distributed Data Interface) [7], and GP SHMEM [20].

It is important to understand that the global memory access model is based on a data-passing model. The purpose of DDI or GA [21,22] is not to replace MPI-1 but to add functionality that one hopes will ultimately be available in MPI-2. DDI has a set of point to point messages implemented by using a socket code. On platforms for which socket

code is not available, MPI is used. At present, DDI uses only simple distributed memory operations based on the point to point messages presented in Table 1.

Table 1. Distributed memory operations

DDI_CREATE	Create distributed matrix
DDI_DESTROY	Destroy distributed matrix
DDI_DISTRIB	Obtain distributed matrix distribution
DDI_GET	Get patch of distributed matrix
DDI_PUT	Put patch of distributed matrix
DDI_ACC	Accumulate patch of distributed matrix

The performance of two of these operations, DDI_GET and DDI_ACC used in DDSCF, is shown on Table 2. The data presented in Table 2 will be used to estimate communication time. This information is essential for the performance analysis carried out in Sections 4 and 5.

Table 2. Performance of remote GET and ACC on a Pentium II 400 Mhz cluster connected by Fast Ethernet

Operation	Latency (μ s)	Bandwidth (MB/sec)
GET	67	9.7
ACC	95	8.9

The RHF method expresses molecular orbitals in terms of atomic basis functions. However, when coding this method it is advantageous to group basis functions in shells, in order to exploit the shared arithmetic and symmetry of shells utilized by most integral packages. The loop structure in the following algorithms is done in terms of shells. All formulas in Section 4 devoted to distributed data SCF and Section 5 devoted to performance models are developed in terms of shells unless it is specified explicitly that

basis functions or atoms are used. We use N when we refer to the total number of basis functions and N_{sh} when we refer to the total number of shells.

A necessary step for any *ab initio* quantum chemistry calculations is evaluation of atomic two electron integral quartets $(ij|kl)$ where $i, j, k,$ and l represent shells. GAMESS uses two integral packages. For fast evaluation of integral quartets consisting of s and p atomic orbitals, the Pople-Hehre [23] integral package is used. For all higher angular momentum integrals, the Rys polynomial code [24] is used. The average performance of these two codes on a typical organic molecule is presented in Table 3. The data is given for three basis sets: a minimal, double zeta plus polarization, and triple zeta plus polarization basis sets.

Table 3. Total time to run Hehre-Pople and Rys integral packages for luciferin, divided by the total number of non-zero integral quartets. Performance measured on a 16-node Pentium II 400 Mhz PC cluster

Basis Set	sec/int	Total number of integrals
Mini	$2 \cdot 10^{-05}$	2,445,366
6-31G**	$6 \cdot 10^{-05}$	8,386,560
TZV**	$2 \cdot 10^{-05}$	202,015,050

The data presented in Table 3 will be used in the following sections to analyze computational time.

3. Brief review of replicated data SCF parallel implementation

The major bottleneck of an SCF procedure is calculation of the Fock matrix. It is therefore useful to review the steps involved in formation of Fock matrix elements and

analyze them. Additional information about SCF procedure can be found in Almolf's review [26]. Each Fock matrix element is the sum of a one-electron part.

$$F_{ij}^{(1)} = (i | h | j) \quad [1]$$

and a two-electron part [25],

$$F_{ij}^{(2)} = \sum_{lk} D_{lk} [(ij|kl) - \frac{1}{2}(il|kj)] \quad [2]$$

where i and j represent basis functions, h is the one electron Hamiltonian, D is the density matrix, and $(ij|kl)$ is a two electron integral quartet. So our final expression for the Fock matrix element is:

$$F_{ij} = F_{ij}^{(1)} + F_{ij}^{(2)} \quad [3]$$

Calculation of the one-electron part takes a relatively small amount of time, since it scales as N^2 , where N is number of basis functions. The two-electron part scales as a much faster growing function N^4 . The calculation of these integral quartets and entering them into F constitutes the major bottleneck of the SCF process, so it is not surprising that this is a part that needs parallelization. The actual number of integral quartets one must evaluate may be reduced by: (a) symmetry of integral quartets, (b) prescreening of integral quartets and (c) symmetry of the system.

Because of permutation symmetry, the following integral quartets are equivalent:

$$\begin{aligned} (ij | kl) &= (ij | lk) = (ji | kl) = (ji | lk) = \\ &= (kl | ij) = (kl | ji) = (lk | ij) = (lk | ji) \end{aligned}$$

Hence, the total number integral quartets to be computed is reduced to

$$\frac{\left(\frac{N*(N+1)}{2}\right) * \left(\frac{N*(N+1)}{2} + 1\right)}{2} = O(N^4) \quad [4]$$

The upper bound of the magnitude of an integral quartet can be estimated from the Schwartz inequality [26]:

$$|(ij|kl)| \leq \sqrt{(ij|ij)} * \sqrt{(kl|kl)} \quad [5]$$

Since the $O(N^2)$ $(ij|ij)$ and $(kl|kl)$ exchange integral quartets can be evaluated before calculation of $(ij|kl)$, Schwartz screening means not all $(ij|kl)$ integral quartets have to be evaluated. Screening typically reduces the number of integral quartets to approximately $O(N^3)$ for three dimensional systems containing about 50 atoms, and the asymptotic limit is $O(N^2)$ for linear systems.

Each integral quartet must be multiplied by a density matrix element to produce a Fock matrix element. Using permutational symmetry, each unique integral quartet multiplies up to six density matrix elements to update up to six Fock matrix elements, for example,

$$F_{kl} + D_{ij}(ij|kl) \quad \rightarrow \quad F_{kl}$$

Since an integral quartet must be multiplied by up to six different density matrix elements, this can potentially lead to a huge communication overhead to fetch these elements if they are located on remote nodes. In GAMESS [11], to eliminate communication overhead, density and Fock matrices are currently replicated on each node. The algorithm is presented in Figure 1


```

DO I=1, N
  DO J=1, I
    Dynamic Load Balancer

    DO K=1, I
      L_H=K
      IF ( K.EQ.I ) L_H=J
      DO L=1, L_H
        Screen (ij|kl)
        Compute (ij|kl)
        {Dij,Dik,Dil,Djk,Djl,Dkl} *(ij|kl) → {Fij,Fik,Fil,Fjk,Fjl,Fkl}
      END DO
    END DO
  END DO
END DO
SUM(F)

```

Figure 1. The direct replicated data SCF (RDSCF) algorithm. N_{shells} is the total number of shells. SUM(F) operation sums up partial contributions of Fock matrices

The do loops access only the unique list of integral quartets. The DLB (dynamic load balancer) in the J loop assigns the next block of integral quartets to be calculated. After looping over all integral quartets, each node will have a partial Fock matrix. These partial matrices are summed to obtain the complete Fock matrix. The algorithm is almost perfectly parallel. Load balancing and global summation of Fock matrices take a negligible amount of time. However, the largest problem that one can calculate is constrained by the local memory of each node. Since density and Fock matrices are replicated, each node has to allocate $2 \cdot 8 \cdot N^2$ bytes of memory. For large systems it is desirable instead to have a distributed data SCF code in which density and Fock matrices are distributed among all nodes. This effectively utilizes the aggregate memory of the parallel computer or cluster, which is often the most expensive component.

4. Distributed Data SCF (DDSCF)

The algorithm described in Figure 1 will not be useful when density and Fock matrices are distributed among nodes, due to the communication overhead involved in manipulating these matrix elements inside four loops. Consider a system that has 1000 shells, and assume that all elements of the distributed matrices are remote. This would be roughly true for large clusters of computers. The total number of integral quartets to be calculated is $N_{\text{int}} \approx 1000^4 = 10^{12}$. Since each integral quartet needs as many as six density matrix elements to update six Fock matrix elements, the worst case is that the total number of remote operations is $N_{\text{comm}} = 12 * 10^{12}$. Now,

$t_{\text{comp}} = 10^{-05}$ = average time to calculate one integral quartet (from Table 3).

$t_{\text{comm}} = t_{\text{latency}} = 1 * 10^{-2}$ = time to get/acc one remote matrix element (from Table 2).

T_{comp} = time to compute all integral quartets

T_{comm} = time to get or acc all necessary density or Fock elements

$$T_{\text{comp}} \approx N_{\text{int}} * t_{\text{comp}} = 10^{12} * 10^{-05} = 10^7 \quad [6]$$

$$T_{\text{comm}} \approx N_{\text{comm}} * t_{\text{comm}} = 12 * 10^{12} * 10^{-02} = 1.2 * 10^{11} \quad [7]$$

So, using a naive algorithm, communication would take four orders of magnitude more time than calculation of the integral quartets. A viable distributed code requires the communication time to be a small fraction of the computation time, $T_{\text{comp}}/T_{\text{comm}} \gg 1$.

So, the communication time must be minimized by using data efficiently. This goal may be achieved by analyzing the construction of the Fock matrix in detail.

First note that six density and six Fock matrix elements can each be grouped into three blocks, as shown in Figure 2.



Figure 2. The grouping of density and Fock matrix elements.

Indices for these elements obey $i \geq j, k \geq l, (ij) \geq (kl)$ to account for the permutation symmetry of the integral quartets. So, we address only the lower triangular parts of the density and Fock matrices. Access to density and Fock matrix elements can be moved to the corresponding outer loops. In Figure 3, the algorithm is presented with a complexity analysis of each critical step. Note that the Schwarz screening already mentioned is applied inside the innermost loop to the computation of integral quartets and effectively reduces the formal $O(N_{sh}^4)$ complexity to roughly $O(N_{sh}^3)$.

	Cost	Number of operations
DO I=1,N _{sh}		
1. GET D _i block	1. t _{comm}	1. N _{sh}
DO J=1,I		
2. GET D _j block	2. t _{comm}	2. $\frac{N_{sh}*(N_{sh}+1)}{2} = O(N_{sh}^2)$
DO K=1,I		
3. GET D _k block	3. t _{comm}	3. $\frac{N_{sh}*(N_{sh}+1)*(2N_{sh}+1)}{6} = O(N_{sh}^3)$
L _H =K		
IF (K.EQ.I) L _H =J		
DO L=1,L _H		
4. Compute (ij kl)	4. t _{comp}	4. $\frac{\left(\frac{N_{sh}*(N_{sh}+1)}{2}\right)*\left(\frac{N_{sh}*(N_{sh}+1)}{2}+1\right)}{2} = O(N_{sh}^4)$
END DO		
5. ACC F _k block	5. t _{comm}	5. $\frac{N_{sh}*(N_{sh}+1)*(2N_{sh}+1)}{6} = O(N_{sh}^3)$
END DO		
6. ACC F _j block	6. t _{comm}	6. $\frac{N_{sh}*(N_{sh}+1)}{2} = O(N_{sh}^2)$
END DO		
7. ACC F _i block	7. t _{comm}	7. N _{sh}
END DO		

Figure 3. DDSCF Pseudocode with performance analysis of time consuming steps. Note: this analysis is based on the number of operations ignoring of the cost of each operation. t_{comm} and t_{comp} are time(sec) per each operation defined in Section 5

The most expensive communication operations are in lines 3 and 5, where the order is $O(N_{sh}^3)$. So, the communication scales as $O(N_{sh}^3)$ while computation scales as $O(N_{sh}^4)$. This algorithm may perform well on supercomputers. However, on clusters where each communication operation is 10 times more expensive than computing one integral quartet, one will obtain poor performance due to the communication overhead.

There are many possible solutions. One is to work with large blocks of data. The extreme case occurs when the size of a block is equal to N_{sh}/N_p . So each time one shell is requested from a remote processor, all shells from the remote processor are copied to a local buffer on the compute server. Then, the latency is offset by moving large blocks of data and the “effective” bandwidth improves significantly. But in this case, six buffers: D_i, D_j, D_k, F_i, F_j, F_k have to be allocated on each processor, where the size of each buffer

is N_{sh}/N_p . This effectively increases local storage requirements by a factor of 6. This measure is justified only for large massively parallel processor platforms (MPPs).

A second approach is to calculate integral quartets only on the processor that has the k block on its local data server. Then, the communication cost scales as $O(N_{sh}^2)$ vs. computation $O(N_{sh}^4)$. The communication overhead drops significantly, but the computation of integral quartets is then based on a static distribution of density and Fock matrices. This algorithm has poor load balance, because:

- Time to compute integral quartets greatly depends on the angular momentum of the basis functions, on the integral package and on the basis set used.
- Distribution of integral quartets to be calculated is not constant among processors.

Figure 4 illustrates this load balancing problem for the simple case of $N_{sh}=20$ and $N_p=20$ from a simulation of our application. Then, N_{int} is the total number of $(ij|kl)$ integral quartets calculated on processor p . An integral quartet is calculated on processor p only if index $(ij|kl)$ integral quartet index $k = p$, so that the D_k block from Figure 3 is local. The data is obtained from simulation of code and holds for much larger values of N_{sh} and N_p .

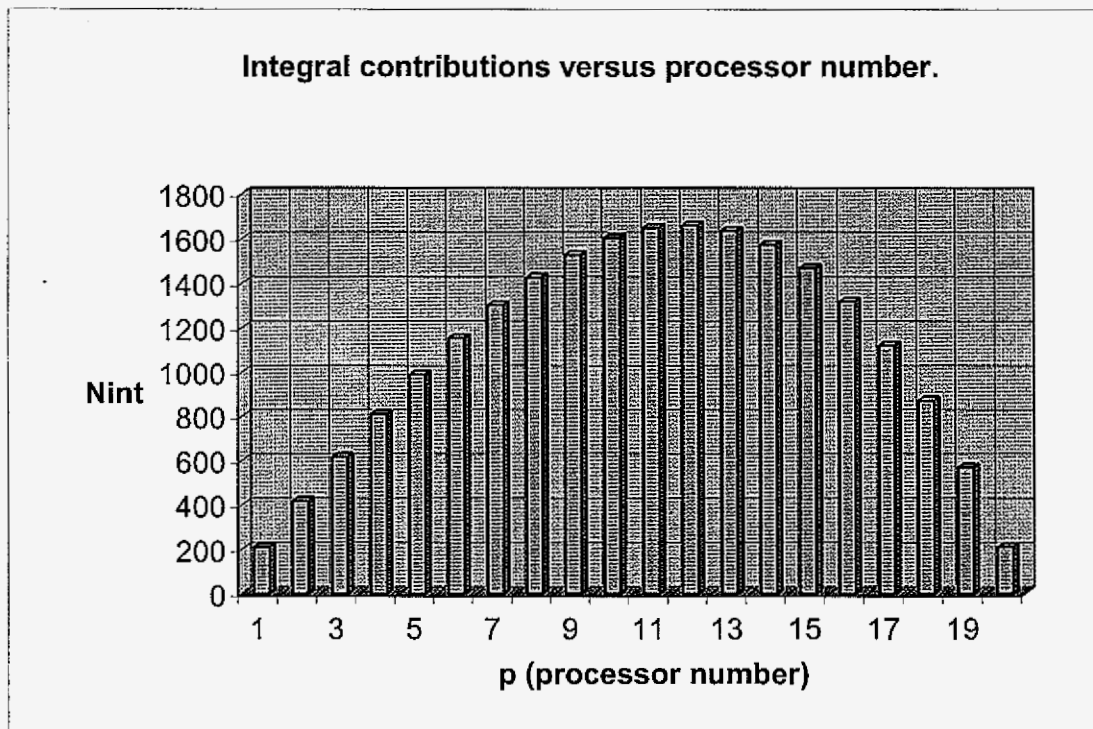


Figure 4. Distribution of integral quartets to be calculated over processors

Due to the reasons listed above, the wall clock time for the SCF step for the problem illustrated in Figure 4 averaged over all nodes scales perfectly, but there is a significant discrepancy in wall clock time among different nodes. The wall clock time on the last and first nodes tends to be much smaller than on other nodes because of the nature of the index k . It is possible to balance computation time by introducing shell and integral quartet weights, determined by predicting how much time each shell will require to calculate integrals. Once this information is available, the shells are distributed in such a way that each processor spends equal time. The weight for each shell may be calculated from Figure 5. To account for the diversity in integral calculations, the calculation time for each integral quartet can be estimated based on the type of shell. However, this type of prediction is unreliable, since it depends greatly on the specific integral package, and

leads to asymmetric distribution of both matrices. Further, the implementation is complicated.

A composite algorithm is needed to achieve reasonable performance and good scaling. Good performance should be achieved by preserving communication scaling as $O(N_{sh}^2)$ and good scaling via dynamic load balancing.

Since both density and Fock matrices are symmetric, in the original replicated data algorithm we addressed only the lower triangles. The key to achieving dynamic load balancing is to store the entire symmetric matrix. The kl element in the lower triangle is equal to the lk element in the upper triangle, but they belong to different nodes unless l and k shells are on the same processor. This is rare if one is using more than a few processors.

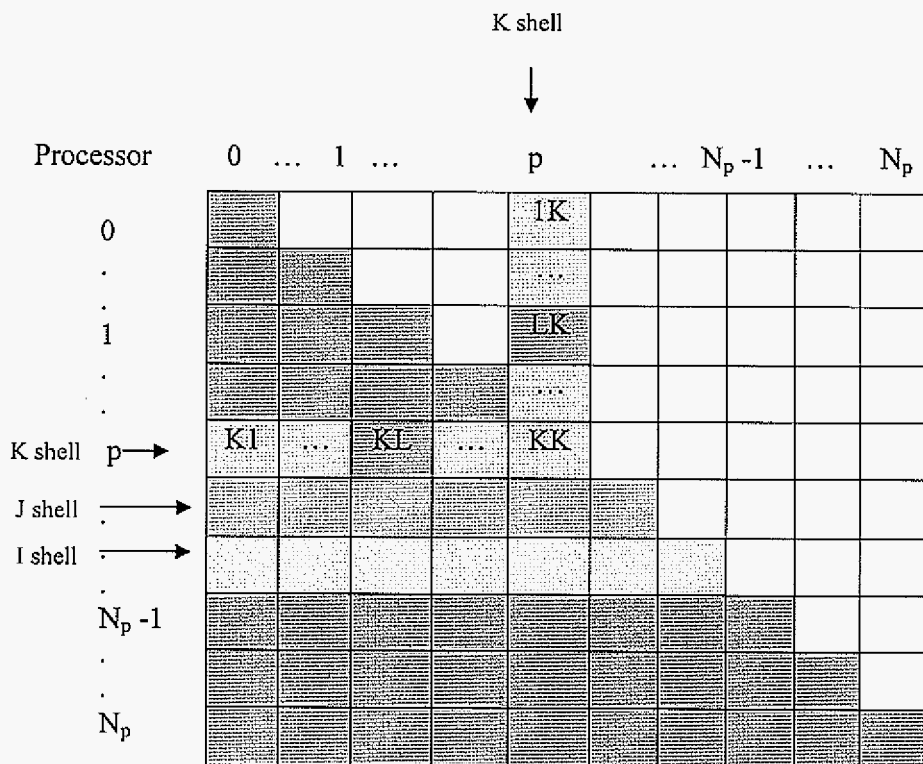


Figure 5. Distributed square matrix

The access pattern to I, J, and K shells is demonstrated in Fig. 5. Processor p has the whole K shell (KL block) that is located in the lower triangle. At the same time, processors with index $< p$ have part of the K shell (LK block) located in the upper triangle. This creates the basis for dynamic load balancing. The workload is balanced between a processor that has the KL block and a processor that has the LK block. This approach, which distributes the workload by taking advantage of the symmetric property of density and Fock matrices, is called pair-wise dynamic load balancing. The K and L indices are frequently changed inside the four loops, providing good average load balancing demonstrated in Figure 6.

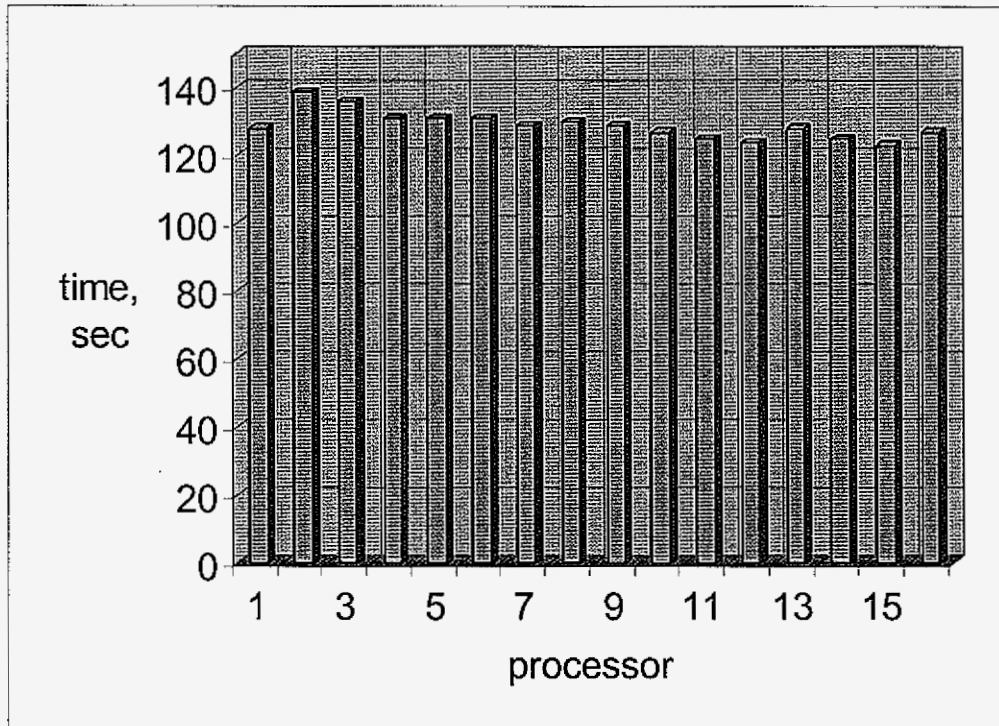


Figure 6. Time to complete one SCF iteration on 16 processors. The tested system is described in Section 6

In conventional dynamic load balancing a local counter on each processor is incremented for all task values and compared with a global counter; the global counter is increased via message passing. In the pair-wise dynamic load balancing scheme, only processors that have the local K or L shells compare the local counter with the global counter. Since the dynamic load balancer is located inside four loops, fine-grained load balancing is achieved. The price for such a load balancer is the communication overhead associated with incrementing the global counter, since this may affect the scalability of the algorithm on a system with a large number of processors.

```

DO I=1,Nsh
  DO J=1,I
    DO K=1,I
      L_H=K
      IF ( K.EQ.I ) L_H=J
      DO L=1,L_H
        IF K or L is local then
          Pair-wise Dynamic Load Balancer

          GET Di and ACC Fi blocks if I is changed
          GET Dj and ACC Fj blocks if J is changed
          Screen (ij|kl)
          Compute (ij|kl)
          {Dij,Dik,Dil,Djk,Djl,Dkl}*(ij|kl)={Fij,Fik,Fil,Fjk,Fjl,Fkl}

        END IF
      END DO
    END DO
  END DO
END DO

```

Figure 7. Pseudocode of final version of DDSCF

Figure 7 presents the final version of the DDSCF algorithm. Both density and Fock matrices are distributed evenly among the processors. The distribution is done over shells since integral quartets are evaluated over shells. All communication calls are placed inside the four loops to dynamically load balance them. DDI operations are called only when the outer loop counter has been incremented. This is also a scheme for enhancing the overlap of communication with computation since a node only does communication when it has received a task to compute that can be enhanced further if we would use non blocking GET and ACC operations.

The DDSCF algorithm can be improved further if communication between local computation and data-server processes becomes a bottleneck. A copy of the local portions of F and D matrices, held simultaneously by the data-server process, is placed in the

innermost loop shown above. This additional improvement provides true scalability of both data and work loads on commodity hardware such as PC clusters at the expense of a four-fold redundancy in data storage.

Note that the access pattern to density elements is different from the access pattern to Fock elements. Each time, when a new integral quartet is calculated, the whole set of Fock elements is updated:

$$F_i \text{ block} \leftarrow F_i \text{ block} + (ij|kl) * \{D_{jk}, D_{jl}, D_{kl} \text{ or } D_{lk}\}$$

$$F_j \text{ block} \leftarrow F_j \text{ block} + (ij|kl) * \{D_{ik}, D_{il}\}$$

$$F_{kl} \leftarrow F_{kl} + (ij|kl) * D_{ij}$$

Or

$$F_{lk} \leftarrow F_{lk} + (ij|kl) * D_{ij}$$

The partial contributions in the lower and upper triangles of the Fock matrix are summed on each processor at the end of each iteration.

The next two sections consider the performance model and results that mathematically and experimentally illustrate that DDSCF is an efficient and scalable algorithm.

5. Performance models

The quality of the DDSCF parallel algorithm shown schematically in Figure 7, is estimated in this section by two important performance models [27,28,29,30]. The first is the ratio of computation time to communication time. The second model addresses

communication efficiency. These two models use estimates of computation and communication times that are calculated below.

The total number of integral quartets is given by

$$N_{\text{int}} = \frac{\left[\frac{N_{sh} * (N_{sh} + 1)}{2} \right] * \left[\frac{N_{sh} * (N_{sh} + 1)}{2} + 1 \right]}{2} \quad [8]$$

The actual number is usually less than the theoretical number, since factors such as the molecular symmetry, basis set, and integral screening can significantly reduce N_{int} . Some of these factors are included in a “scale factor” α that ranges from 0 to 1. So the computation time on N_p processors is :

$$T_{\text{comp}} = \alpha * t_{\text{comp}} * \frac{N_{\text{int}}}{N_p} \quad [9]$$

where t_{comp} is the average time required to calculate integral quartet.

From Figure 3, the total number of communication operations N_{comm} is the sum of Get ($N_{\text{comm_get}}$) and Acc ($N_{\text{comm_acc}}$) operations from I and J loops:

$$N_{\text{comm}} = N_{\text{comm_get}} + N_{\text{comm_acc}} = 2 * N_{sh} + 2 * \frac{N_{sh} * (N_{sh} + 1)}{2} \quad [10]$$

For simplicity we assume that bandwidth (BW) and latency (t_{latency}) of Get and Acc operations are same. The time required to perform one communication operation, t_{comm} , is the sum of the latency time and the transmission time (t_{transm}):

$$t_{\text{comm}} = t_{\text{latency}} + t_{\text{transm}} \quad [11]$$

The transmission time is calculated from the bandwidth obtained from the formula:

$$t_{\text{transm}} = \frac{N_{sh} * N_{\text{nbfpsh}} * 8}{BW * 1024 * 1024} \quad [12]$$

where N_{nbfpsh} is number of basis functions per shell.

Thus, the total communication time on N_p processors is:

$$T_{comm} = \alpha * t_{comm} * \frac{N_{comm}}{N_p} \quad [13]$$

The total execution time on N_p processors is:

$$T_{exec} = T_{comp} + T_{comm} \quad [14]$$

The first performance model measures the ratio of computation time to communication time (T_{ratio}):

$$T_{ratio} = \frac{T_{computational}}{T_{communication}} = \frac{\alpha * t_{comp} * N_{int} * N_p}{\alpha * t_{comm} * N_{comm} * N_p} = \frac{t_{comp} * N_{int}}{t_{comm} * N_{comm}} \quad [15]$$

After substitution of appropriate values taken from Tables 2 and 3, T_{ratio} is in the range

$$T_{ratio} = \frac{T_{computational}}{T_{communication}} = \frac{t_{comp} * N_{int}}{t_{comm} * N_{comm}} = 2210 - 22116 \quad [16]$$

where

N_{sh} ranges from 1000 to 10000

$N_{bfsh}=4$

$t_{comp}=6*10^{-05}$ sec (6-31G** basis set)

$t_{latency}=1*10^{-4}$ sec

BW = 9 Mb/sec

The second performance model addresses the communication efficiency (Ef_{comm}) of DDSCF; this is the ratio of computational time (eq. 9) to the total execution time (eq. 19) :

$$Ef_{comm} = \frac{T_{comp}}{T_{exec}} = \frac{T_{comp}}{T_{comp} + T_{comm}} = \frac{t_{comp} * N_{int}}{t_{comp} * N_{int} + t_{comm} * N_{comm}} \quad [17]$$

After substitution of same constants as in T_{ratio} :

$$Ef_{comm} = \frac{t_{comp} * N_{int}}{t_{comp} * N_{int} + t_{comm} * N_{comm}} = 99.9548-99.9955 \% \quad [18]$$

Such high T_{ratio} and Ef_{comm} suggest that in the algorithm presented in Figure 7, communication time is negligible compared to computation time. This means the CPU utilization will be high on each processor, resulting in good performance of DDSCF. Since these two models don't include the sparsity factor that arises from the molecular geometry and basis set, there can be significant variation between results of performance analysis and actual calculations. It is clear that the DDSCF algorithm will perform best for dense molecules and large basis sets because of improved T_{comp} over T_{comm} .

These two performance models demonstrate that for a sensibly chosen problem, the DDSCF algorithm is very efficient. One can expect that DDSCF will have good scalability on a large number of processors, as well as good performance relative to a replicated data SCF algorithm. In the next section, the results of test calculations are shown that are consistent with results of the performance analysis.

6. Results

To test the scalability and overall performance of the DDSCF algorithm two molecules are chosen. The first molecule is calphostin [31], a potential anti-cancer drug [32,33]. The optimized structure is illustrated in Figure 8 (a). The second molecule is luciferin [34], whose equilibrium structure is shown in Figure 8 (b). Luciferin's fluorescent properties are responsible, for example, for the light emitted by fireflies. The basis sets used for calphostin and luciferin are the split-valence basis sets [35] 6-31G (d) and 6-31G (p,d,f), respectively. The choice of molecules and basis sets was determined to design computationally demanding calculations that produce reliable benchmarking results, such that all arrays fit in the memory of a single node of a PC cluster [16].

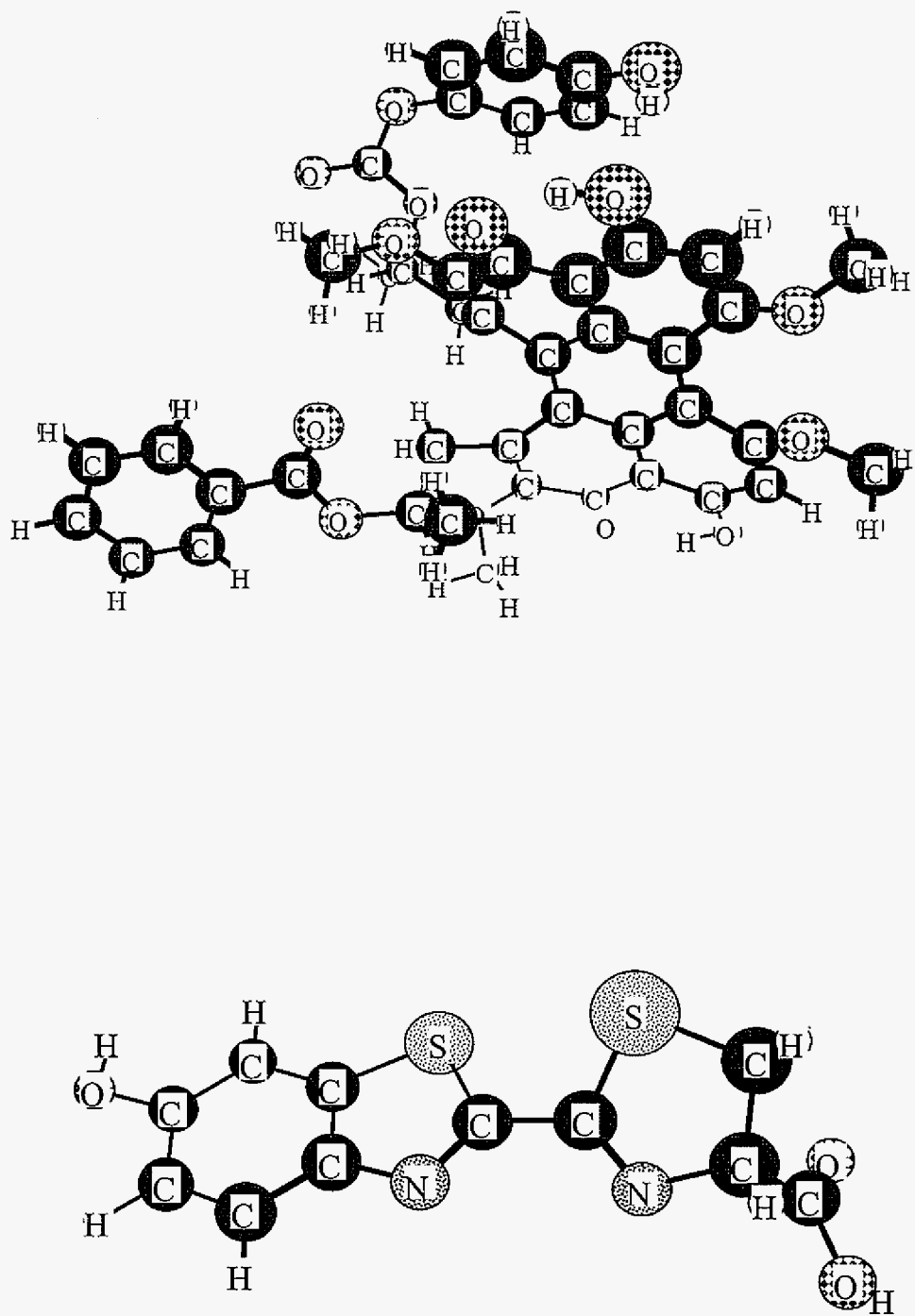


Figure 8. Calphostin (946 basis functions) on the top (a) and Luciferin (498 basis functions) on the bottom (b)

Nodes	Calphostin		Luciferin	
	Wall	Speedup	Wall	Speedup
1	14756	1.0	2485	1.0
4	3764	3.9	618	4.0
8	1923	7.7	308	8.0
16	1010	14.6	160	15.5

Table 4. Speedup and wall clock time (sec) of the Fock builder in the first RDSCF iteration on a 16 node PC cluster

Nodes	Calphostin				Luciferin			
	Wall	Speedup	Comm	Comm%	Wall	Speedup	Comm	Comm %
1	14756	1.0	0.0	0.0	2485	1.0	0.0	0.0
4	3804	3.9	78.8	2.1	621	4.0	5.4	0.9
8	2024	7.3	101.2	5.0	318	7.8	5.4	1.7
16	1117	13.2	106	9.4	162	15.3	4.9	3.0

Table 5. Speedup, wall clock time, Comm time, and Comm% of Fock builder in first DDSCF iteration on a 16 node PC cluster; times in seconds

In Tables 4 and 5 the wall clock time is measured on the compute server, while communication (Comm) time is measured on the data server. Since all communication operations are synchronous, the communication time on the data server provides a good estimate of the ratio of communication time to computation time on the compute server. The communication time for calphostin is bigger than that predicted by our performance model. This can be explained by DLB overhead or inefficient work of the data server.

First, compare the performance of the RDSCF and DDSCF algorithms. In Tables 4 and 5, the wall clock times for calphostin and luciferin are presented for different numbers of nodes [16]. The wall clock time is almost identical for RDSCF and DDSCF

algorithms. As expected, even though the density and Fock matrices are distributed in the DDSCF algorithm, communication time does not contribute significantly to the overall timing.

The relative speedup data of DDSCF is demonstrated in Table 5. Calphostin and luciferin scale very well, but there is a small loss in scalability on 16 processors. The loss in scalability is mainly due to communication operations between computational and data servers. In Table 5, the Comm columns present the time spent to perform message passing by each data server. The percentage (Comm%) wall clock time on computational server to Comm clock time on data server is demonstrated in Table 5. Up to 10 % of the Comm time is consumed by communication operations such as GET, ACC, and message passing operations to update the global counter for the dynamic load balancer.

The overall performance of DDSCF is good as expected from the performance model analysis. For much bigger molecules, the speedup data is likely to improve, since the number of computational operations grows much faster than the number of communication operations in the DDSCF algorithm.

7. Conclusions

We have successfully developed an efficient, scalable distributed data algorithm that solves a major bottleneck in SCF calculations on distributed memory platforms. A new dynamic load balancing technique for symmetric matrices is developed. This technique allows us to significantly improve the ratio of computation time to communication time, and to achieve excellent load balancing. The performance models and results demonstrate the high efficiency and scalability of the DDSCF algorithm.

The cluster solution for supercomputing is an attractive alternative to large massively parallel processor platforms due to a good performance/price ratio. The slow speed interconnection between nodes on clusters is the source of poor performance and scalability of algorithms developed for MPPs. Typically, in distributed data quantum chemistry algorithms, bulk data is moved between processors. This means that to achieve good performance and scalability high bandwidth is required. Bulk data movement is designed to hide the need for low latency network.

Acknowledgement

Authors are thankful to Mike Schmidt and Graham Fletcher for useful discussions and support. We also want to thank Robert Harrison for his advice on earlier stages of work.

All calculations were performed on a PC cluster that was funded by the Fundamental Interactions program of the Ames Laboratory, supported by the US-DOE Basic Energy Sciences Division.

References

1. <http://www.beowulf.org> (Data accessed: November 26, 2002)
2. W. Gropp, S. Huss-Liderman, A. Lumsdaine, E. Lusk, B. Nitzburg, W. Saphir, M. Shir, MIT Press, 1998
3. R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, R. Menon, Morgan Kaufmann Publishers, 2000
4. <http://www.scl.ameslab.gov/Projects/IBMCluster> (Data accessed: November 26, 2002)
5. <http://www.scl.ameslab.gov/Projects/IBMCluster/Benchmarks.html> (Data accessed: November 26, 2002)
6. http://cmp.ameslab.gov/MP_Lite (Data accessed: November 26, 2002)
7. G. D. Fletcher, M. W. Schmidt, B. M. Bode, and M. S. Gordon, Computational Physics Communications, 1999
8. <http://www.nersc.gov/research/ftg/via> (Data accessed: November 26, 2002)
9. <http://www.nersc.gov/research/ftg/mvich> (Data accessed: November 26, 2002)
10. <http://oss.sgi.com/projects/stp> (Data accessed: November 26, 2002)
11. M.W. Schmidt, K.K. Baldrige, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L. Windus, M. Dupuis, J.A. Montgomery, *J. Computational Chemistry*, 14 (1993) 1347-1363.
12. G.D. Fletcher, M.W. Schmidt, M.S. Gordon, *Advances in Chemical Physics*, Vol. 110, 1999 John Wiley & Sons, Inc.
13. Thomas R. Furlani and Hary F. King, Spencer, France, 1996

14. R.J. Harrison, R.A. Kendall, R.J. Littlefield, I.T. Foster, J.L. Tilson, A. F. Wagner, R.L. Shepard, *Journal of Computational Chemistry*, Vol. 17, No. 1, 109-123, 1996
15. R.J. Harrison, M.F. Guest, R.A. Kendall, D.E. Bernholdt, A.T. Wong, M. Stave, J.L. Anchell, A.C. Hess, R.J. Littlefield, G.L. Fann, J. Nieplocha, G.S. Thomas, D. Elwood, J.L. Tilson, R.L. Shepard, A.F. Wagner, I.T. Foster, E. Lusk, R. Stevens, *Journal of Computational Chemistry*, Vol. 17, No. 1, 124-132, 1996
16. <http://www.msg.ameslab.gov> (Data accessed: November 26, 2002)
17. H. Dachsel, J. Nieplocha, R. Harrison, in the Proceedings of SC 1998
18. W. Gropp, E. Lusk, R. Thakur, MIT Press, 2000
19. J. Nieplocha, B. Carpenter, Proc. 3rd Workshop on Runtime Systems for Parallel Programming (RTSPP) of International Parallel Processing Symposium IPPS/SPDP '99, San Juan, Puerto Rico, April 1999, in (1) J. Rolim et al. (eds.) *Parallel and Distributed Processing*, Springer Verlag LNCS 1586, and (2) IPPS/SDP'99 CDROM, 1999.
20. K. Parzyszek, J. Nieplocha, R.A. Kendall, Las Vegas, Nevada, November 2000, pp. 401-406. IASTED, Calgary (2000)
21. J. Nieplocha, R.J. Harrison, and R.J. Littlefield, Proc. Supercomputing'94, pages 340-349, 1994.
22. J. Nieplocha, R.J. Harrison, and R.J. Littlefield, *The Journal of Supercomputing*, 10:197-220, 1996.
23. J.A. Pople, W.J. Hehre, *J. Comput. Phys.* 27, 161-168(1978)
24. H.F. King, M. Dupuis, *J. Comput. Phys.* 21,144(1976)
25. A. Szabo, N.S. Ostlund, Dover Publications, Inc., 1996
26. J. Almlof, European Summer School in Quantum Chemistry, pages 1-90, Springer-Verlag Berlin Heidelberg, 1994

27. I. Foster., Addison-Wesley, 1995
- [28] Adolfo Hoisie, Harvey Wasserman, Tutorial Notes, SC 1999
29. D.A. Bader et al., Tutorial Notes, SC 2000
30. Thomas H. Cormen and others, The MIT press, McGraw-Hill, 1999
31. A. Datta, P. Bandyopadhyay, J. Wen, J. Petrich, M.S. Gordon, The Journal of Phys. Chem. A, Vol 105, Number 6, pages 1057-1060, 2001
32. N. Duran, P.S. Song, Photochem. Photobiol. 1986, 43, 677-680
33. G.A. Kraus, W. Zhang, M.J. Fehr, J.W. Petrich, Y. Wannemuehler, S. Carpenter, Chem. Rev. 1996, 96, 523-535
34. E.H. White, F. Capra, W.D. McElroy, J. Am. Chem. Soc. 83, 2402-3(1961)
35. M.S., Gordon, J.S. Binkley, J.A. Pople, W.J. Pietro, W.J. Hehre, J. Am. Chem. Soc., 1982, 104, 2797-2803.

CHAPTER 4: A DISTRIBUTED DATA PARALLEL CPHF ALGORITHM FOR ANALYTIC HESSIANS

A paper to be submitted for publication to

Journal of Computational Chemistry

Yuri Alexeev, Michael W. Schmidt,

Theresa Windus, Mark S. Gordon

Abstract:

One of the most commonly used operations to study potential energy surfaces of reactions and chemical systems is the Hessian calculation. The most accurate analytic Hessian is computationally and memory demanding. A new highly scalable, efficient, distributed data analytical Hessian algorithm is presented. Features of the distributed data parallel CPHF are (a) columns of density-like and Fock-like matrices are distributed among processors, (b) an efficient static load balancer scheme was developed to achieve good work load distribution among processors, (c) network communication time is minimized, (d) numerous performance improvements in analytic Hessian steps. As result, the new code has excellent performance. The performance of the code is demonstrated via calculations on large biological systems.

Keywords:

Quantum chemistry, Cluster Computing, Couple Perturbed Hartree Fock, CPHF, Distributed Data Interface, DDI, Nonuniform Memory Access, Dynamic Load Balancing, Static Load Balancing.

Introduction

An important advance in theoretical chemistry is the development of new methods and algorithms for calculating large chemical systems. The need for new parallel algorithms arises because many important quantum chemistry methods such as second order perturbation theory (MP2) [1], coupled cluster theory (CCSD(T)) [2], full configuration interaction (FCI) [3], complete active space self-consistent field (CASSCF) [4], and analytical Hessian calculations [5] are bounded not only by CPU power but by the amount of the memory and disk on a computer as well. Thus, the sizes of systems that can be studied by such methods often limit their usefulness to the scientific community. An important direction in quantum chemistry is developing algorithms so that both computation and memory are scalable.

To study potential energy surfaces of reactions and chemical systems two operations are performed most often: energy + gradient calculations and Hessian calculations. The Hessian is the second derivative matrix of the total energy with respect to geometric coordinates. The diagonalized Hessian provides harmonic normal modes and corresponding vibrational frequencies. Transition states and minima are indicated by one and no imaginary mode, respectively. So, a Hessian calculation is useful both for providing vibrational frequencies and as a diagnostic for the nature of a stationary point.

There are two types of Hessian calculations: numerical using a finite difference method [6] and analytic [7]. The computational and memory requirements are rather different for these two methods. The analytic approach is usually preferable due to the accuracy of calculated vibrational frequencies and considerable time savings.

The coupled perturbed Hartree-Fock (CPHF) approach for computing analytic second derivatives was developed by many authors including Pople et al. [8], since 1968 when it was introduced by Gerratt and Mills [7]. The first implementations used a molecular orbital (MO) based approach in which the orbital Hessian used in the CPHF was computed via two electron MO integrals. This approach requires a four index transformation of atomic orbital (AO) integrals and therefore storage of the huge intermediate orbital Hessian matrix. Osamura et al. [9] suggested an AO based approach to solve the CPHF equations. Later Head-Gordon et al. [10] developed a “direct CPHF” method to avoid the four index integral transformation and storing the orbital Hessian matrix. This approach is used in the present paper.

The CPHF step is one of the most computationally demanding steps in the analytic Hessian code. Further, the memory requirements of the CPHF step are the largest in analytic Hessian calculations. This limits the size of chemical systems that can be studied using this method. The current bottleneck in the replicated data CPHF algorithm that is implemented in GAMESS [11] is that the density-like (D) and Fock-like (F) matrices which scale as $O(N^3)$, where N is the size of the atomic basis set, are replicated on each node. Both matrices are three dimensional matrices, so CPHF calculations are limited to relatively small chemical systems if all matrices are replicated.

There have been several other efforts to parallelize the CPHF step in Hessian calculations. Windus et al. [12] developed a “small scale parallel algorithm” for first and second derivatives of the integrals. In their implementation, the orbital Hessian used in

the CPHF step was computed via MO integrals. Both the four index transformation of AO integrals and the CPHF step were carried out sequentially. Marquez et al. [13] implemented parallel second derivatives of the integrals, four index transformation of the AO integrals, and the CPHF step in HONDO [14]. However, in their implementation the MO integrals were saved to the disk. The product, orbital Hessian with response matrix, is directly computed from saved MO integrals. As result, on 8 CPUs a relative speedup of only four is achieved. Sosa et al. [15,16] parallelized a "direct" implementation of CPHF. The relative speedup is 13 on 16 CPUs, using is α - Pinene and the 6-31G(d) basis set on a Cray T3E 600. The main disadvantage of all these algorithms is that all arrays are duplicated. So the largest system that can be calculated is limited by the maximum memory on a single CPU. Recently Prakashan et al. [17] reported a simple "direct" parallel implementation of CPHF [18], in which both computation and memory are distributed among all CPUs. The parallelization is done via responses that are evenly divided among the CPUs. Subsequently, each CPU runs a sequential CPHF within a given subset of responses. Thus, memory and computation are distributed. This is a very appealing approach because of simplicity of implementation. There are however, a few drawbacks to this approach: (a) the computation of integrals is replicated on each CPU, (b) the work load balancing is not addressed: some of the responses usually converge earlier than others (the average CPU time reported may differ from the real time), (c) possibly slower convergence depending on the converger: next trial responses are obtained within a subset of responses on each CPU [19]. The replication of the integral computation is a fundamental limitation due to Amdahl's law scalability of this algorithm. The relative speedup of $C_{24}H_{50}$, using the 6-31G* basis set on an IBM SP is 12 on 16 CPUs.

The purpose of the present paper is to present a distributed data coupled perturbed Hartree-Fock (DDCPHF) algorithm. The algorithm uses recently developed algorithms and libraries to achieve good scalability of both computation and memory among available CPUs. The “direct” approach pioneered by Head-Gordon [10] is used to avoid storage of the orbital Hessian. To solve multiple sets of linear CPHF equations a fast preconditioned conjugate gradient procedure is used [20]. The memory is distributed via the distributed data interface (DDI) [21]. The novel DDCPHF algorithm presented here uses an approach originally developed for parallelization of the self consistent field SCF method [22]. Hence, the algorithm is efficient and scalable. This is demonstrated by benchmark timings presented in the Results section. The code is developed for closed shell wavefunction, but can be easily extended to UHF and ROHF wavefunctions.

Tools and platforms

Beowulf-class clusters [23] are popular since small research groups and departments usually don't have sufficient resources to purchase large computer systems. Currently, the most common types of clusters are workstations connected by a commodity network such as Fast Ethernet or Gigabit Ethernet. The DDI codes in GAMESS have been developed and tested on an IBM pSeries cluster [24]. This cluster consists of 32 IBM RS/6000 pSeries p640 servers connected by dual Fast Ethernet and dual Gigabit Ethernet. Each p640 has 4 Power3II processors running at 375MHz, with 16 GB of memory and 73 GB local disk. The aggregate system has 512 GB of RAM and is capable of 192 GFLOPs peak performance.

Complex scientific applications, such as many quantum chemistry methods including CPHF, often require a sophisticated distribution of data with irregular access patterns. The complexity of programming within the limits of a classical message-passing model [25] can be too high for such applications. An alternative is a model based on the global memory access model. The global memory access model is implemented in GAMESS in the DDI library. At present, DDI uses only simple distributed memory operations based on the point to point messages presented in Table 1.

Table 1. Distributed memory operations

DDI_CREATE	Create distributed matrix
DDI_DESTROY	Destroy distributed matrix
DDI_DISTRIB	Obtain distributed matrix distribution
DDI_GET	Get patch of distributed matrix
DDI_PUT	Put patch of distributed matrix
DDI_ACC	Accumulate patch of distributed matrix

Brief review of CPHF theory

There are two methods for computing Hessians: analytic and numerical. A numerical Hessian calculation has a modest N^2 (N is the total number of basis functions) memory requirement, but is computationally ineffective because it scales as $N_{xyz} * N^4 * N_{iter}$ (where N_{xyz} is the total number of atoms in the system multiplied by three and N_{iter} is the number of iterations) and frequently does not predict accurate vibrational frequencies.

The analytic Hessian method presented in this paper has $N^2 * N_{xyz}$ memory requirements and $N_{iter} * N^4$ computational requirements.

One of the most computational and memory consuming steps of analytic Hessian calculations is the CPHF step. The CPHF equation is:

$$\begin{aligned} \sum_k \sum_l \{ (4[ij|kl] - [ik|jl] - [il|jk]) - (\epsilon_k - \epsilon_l) \delta_{kl} \delta_{ij} \} U_{kl}^a = \\ = \sum_k \sum_l (S_{kl}^a \{ 2[ij|kl] - [ik|jl] \} + S_{ij}^a \epsilon_j - F_{ij}^a) \end{aligned} \quad (1)$$

where i, k represent virtual MO orbitals, and j, l represent occupied MO orbitals. Therefore $[ij|kl]$, etc. are two electron integrals over MO, ϵ_k are the orbital energies, U is the response matrix, S^a is the derivative of the overlap matrix, F^a is the active Fock matrix. One can rewrite equation [1] in the more compact form

$$\sum_k \sum_l A_{ij,kl} U_{kl}^a = B_{ij}^a \quad (2)$$

where A is called the orbital Hessian and B is referred to as the inhomogeneity. All matrices are formed in the molecular orbital (MO) space. To avoid calculations of the large A matrix, $A_{ij,kl} U_{kl}^a$ and B_{ij}^a are calculated directly from AO integrals. GAMESS uses two integral packages. For fast evaluation of integral quartets consisting of s and p atomic orbitals, the Pople-Hehre [26] integral package is used. For all higher angular momentum integrals, the Rys polynomial code [27] is used. Since the integrals are evaluated in shell integral quartets this explains the loop structure shown in Figure 4 below. The computation is converted to a series of Fock-like builds from density-like matrices with dimension $N^2 * N_{xyz}$. The AO integrals required by the CPHF are calculated in exactly the same way as in the Fock builder. The only difference is that each unique

integral quartet multiplies up to $6*N_{xyz}$ density - like matrix elements to update up to $6*N_{xyz}$ Fock - like matrices:

$$F_{kl}^{ixyz} + D_{ij}^{ixyz} (ij|kl) \rightarrow F_{kl}^{ixyz}$$

$$F_{jl}^{ixyz} + D_{ik}^{ixyz} (ij|kl) \rightarrow F_{jl}^{ixyz}$$

...

As a result, the computation to communication ratio is even smaller for CPHF than for the Fock builder because communication requirements have grown more than computation requirements.

After $A_{ij,kl}^a$ and B_{ij}^a are formed, the multiple set of linear equations

$$\sum_k \sum_l A_{ij,kl}^a U_{kl}^a - B_{ij}^a = 0 \quad (3)$$

is solved using a preconditioned conjugate gradient procedure.

Distributed data CPHF algorithm

To explain clearly the strategy used to parallelize the CPHF algorithm and how it is implemented, this chapter is divided into subsections that clearly explain each step.

The goals of the DDCPHF parallelization scheme are

1. Distribute evenly the computation and largest Fock – like and density – like matrices among available processors. The code should have both computational and data scalability.
2. Minimize communication and other overheads. The code is supposed to have comparable performance to replicated data CPHF. It is often possible to design a code that has good scalability but poor performance.

A few techniques can be applied to parallelize the CPHF code and satisfy the goals listed above. These techniques are discussed in detail in the next four sections.

1. A blocking technique

In the commonly used blocking technique small computational tasks are grouped into larger tasks. If the computational tasks are grouped then the necessary data needed to get or accumulate is grouped too. This will increase the network bandwidth for communication operations if the size of the data block before grouping is less than ~3 MB (the actual number depends on network type, operating system, etc). A typical bandwidth graph is shown in Figure 1. The graph will look different for different MPI implementations and other factors [28]. However, the typical change is decreasing bandwidth and shift of the curve to the right which will make blocking of data even more justified.

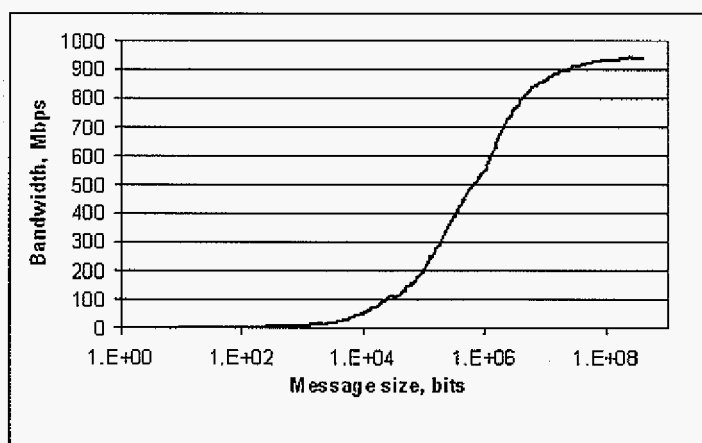


Figure 1. TCP/IP send/receive performance on the IBM RS/6000

In GAMESS, AO integrals are computed in shell blocks. Therefore every $D(I,J)$ element is accessed as a block of data $D(1:N_{bfps}I,1:N_{bfps}J)$ where $N_{bfps}I$ and $N_{bfps}J$ are

number of basis functions per corresponding I and J shell. This approach can be extended further to compute integrals over atom blocks [29,30]. In the DDCPHF algorithm a more effective strategy was employed to block data. Note that the six Fock-like update operations can each be divided into three groups, as shown in Figure 2.

$F_{kl}^{ixyz} + D_{ij}^{ixyz} (ij kl) \rightarrow F_{kl}^{ixyz}$ $F_{jl}^{ixyz} + D_{ik}^{ixyz} (ij kl) \rightarrow F_{jl}^{ixyz}$ $F_{jk}^{ixyz} + D_{il}^{ixyz} (ij kl) \rightarrow F_{jk}^{ixyz}$ D_i^{ixyz} block contains: $D_{ij}^{ixyz}, D_{ik}^{ixyz}, D_{il}^{ixyz}$ F_i^{ixyz} block contains: $F_{ij}^{ixyz}, F_{ik}^{ixyz}, F_{il}^{ixyz}$	$F_{il}^{ixyz} + D_{jk}^{ixyz} (ij kl) \rightarrow F_{il}^{ixyz}$ $F_{ik}^{ixyz} + D_{jl}^{ixyz} (ij kl) \rightarrow F_{ik}^{ixyz}$ D_j^{ixyz} block contains: $D_{jk}^{ixyz}, D_{jl}^{ixyz}$ F_j^{ixyz} block contains: $F_{jk}^{ixyz}, F_{jl}^{ixyz}$	$F_{ij}^{ixyz} + D_{kl}^{ixyz} (ij kl) \rightarrow F_{ij}^{ixyz}$ D_k^{ixyz} block contains: D_{kl}^{ixyz} F_k^{ixyz} block contains: F_{kl}^{ixyz}
---	---	---

Figure 2. Grouped update operations and corresponding block structures

The idea is to access, for example, a line D_i^{ixyz} in a 3 dimensional density – like or Fock – like matrix every time $D_{ij}^{ixyz}, D_{ik}^{ixyz}, D_{il}^{ixyz}$ are needed. The density – like matrix is shown in Figure 3. The access to the Fock – like matrix is performed in the same way.

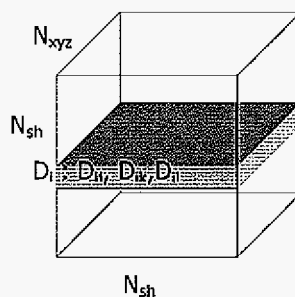


Figure 3. Density like matrix D; Di block is a blue box

For large target systems the size of each block is estimated to be ~3-10 MB which is beyond the 3MB threshold. Therefore, operations GET and ACC in the DDCPHF algorithm operate on the optimal size of the data.

2. A localized data access technique

Dynamic load balancing (DLB) is often used when an even distribution of computations is desirable. The disadvantages of DLB are that there is communication overhead associated with distributing jobs on the master CPU and there is irregular access to the data. The irregular access often implies that if data is not local then it must be fetched from remote CPUs. The cost of fetching data can easily surpass the time required to compute a task, especially if small tasks are utilized. Therefore, DLB is best suited for replicated data/computationally driven algorithms. The alternative approach is to utilize a static work distribution. In this data distribution driven paradigm, the computation of a task is performed only if the data is local. This is the approach used in the DDCPHF algorithm. In the proposed DDCPHF algorithm in Figure 4, an integral is computed only if shell J or K is local.

	Cost	Number of operations
DO I=1,N _{sh}		
1. GET D(I) shell	1. t _{comm}	1. N _{sh}
DO J=1,I		
DO K=1,J		
DO L=1,K		
IF J or K is Local		
2. Compute (IJ KL)	2. t _{comp1}	2. $\frac{\left(\frac{N_{sh} * (N_{sh} + 1)}{2}\right) * \left(\frac{N_{sh} * (N_{sh} + 1)}{2} + 1\right)}{2} = O(N_{sh}^4)$
DO Ixyz=1,Nxyz		
3. Update Fock elements	3. t _{comp2}	3. O(N _{sh} ⁵)
END DO		
END DO		
END DO		

```

      END DO
4.   ACC F(I) shell      4.  $t_{comm}$       4.  $N_{sh}$ 
      END DO

```

Figure 4. DDCPHF algorithm with performance analysis of time consuming steps;
 t_{comm} is the time to get/acc one shell, t_{comp1} is the time to calculate two electron integral block,
 t_{comp2} is the time to update up to six Fock elements

Up to six elements of the Fock matrix are updated from up to six elements of the density matrix. All CPUs execute the GET D(I) shell so that all CPUs have D_{ij} , D_{ik} , D_{il} elements. After the integrals are computed the F(I) shell is accumulated to the distributed matrix F so that the F_{ij} , F_{ik} , F_{il} elements are updated. Other elements are fetched and updated based on whether shell J or shell K is local. If shell J is local D_{jk} and D_{jl} elements are available and F_{jk} and F_{jl} elements can be updated in array F(I) which will be accumulated at the end of the cycle:

Update array F(I):

$$F(I,K) = F(I,K) - D(J,L) * (IJ|KL)$$

$$F(I,L) = F(I,L) - D(J,K) * (IJ|KL)$$

Update local F(J):

$$F(J,K) = F(J,K) - D(I,L) * (IJ|KL)$$

$$F(J,L) = F(J,L) - D(I,K) * (IJ|KL)$$

If K shell is local D_{kl} element is available and F_{kl} element can be updated:

Update array F(I):

$$F(I,J) = F(I,J) - D(K,L) * (IJ|KL)$$

Update local F(K):

$$F(K,L) = F(K,L) - D(I,J) * (IJ|KL)$$

3. Self consistent work distribution

The advantage of using a static work distribution is clear from Figure 4. This algorithm has a large computation cost to communication cost ratio:

$$\text{Ratio} = \frac{\text{Computation}}{\text{Communication}} = \frac{N_{sh}^5}{N_{sh}} = N_{sh}^4 \quad (4)$$

The reason that DDCPHF is such a communication conservative algorithm is that $t_{comm} \gg t_{comp}$ where t_{comp} is defined in Figure 4.

In the new DDCPHF algorithm the computation of integral quartets is then based on a static distribution of density and Fock matrices. The work load per CPU is affected by the following factors:

1. The time required to compute integral quartets greatly depends on the angular momentum of the basis functions, on the integral package and on the basis set used.
2. The distribution of integral quartets to be calculated is not constant among processors (This is explained in detail in [22]).
3. The time required to compute integral quartets depends on distances between atoms in a chemical system or how many integrals are screened out.

To solve the workload imbalance note that CPHF has an iterative nature as do many quantum chemistry algorithms. The responses are unknown, and it takes 10-15 iterations to converge. Initially, shells are distributed evenly among CPUs for the D and F matrices. After each iteration, the time to compute the integrals associated with a given shell is estimated based on the wall clock time required to finish the CPHF step on one CPU divided by the number of shells per CPU. The distribution of shells is adjusted, based on this information on the next iteration. Typically, the self consistent work

distribution converges within 3-5 iterations if the convergence criterion is 10% imbalance. After the self consistent work distribution converges the shell distribution is locked and CPHF proceeds further until responses are found. The work distribution is self consistent because redistribution of the shells affects the time associated with the computation per shell. Since wall clock timings are used, at convergence, all factors such as molecule properties, network performance, heterogeneity of a cluster are taken into account to achieve good work load distribution.

To demonstrate the performance of the self consistent work distribution scheme the cAMP molecule [31] was chosen. cAMP is the second messenger that carries signals from a cell surface to proteins within the cell, and is found widely in eukaryotes. cAMP also frequently acts to stimulate protein kinases. The basis set used for cAMP calculations is 6-31G (d,p). On the first iteration 122 shells are distributed evenly on 8 CPUs as shown in Figure 5. This results in an uneven work load distribution which is a direct result of using the localized data access technique discussed in the previous section. The weight or workload associated with a shell is estimated by using wall clock timings. This information is used to redistribute shells in the D and F matrices. After four iterations good workload is achieved. The green line is an optimal workload. The final workload imbalance is an acceptable 8%.

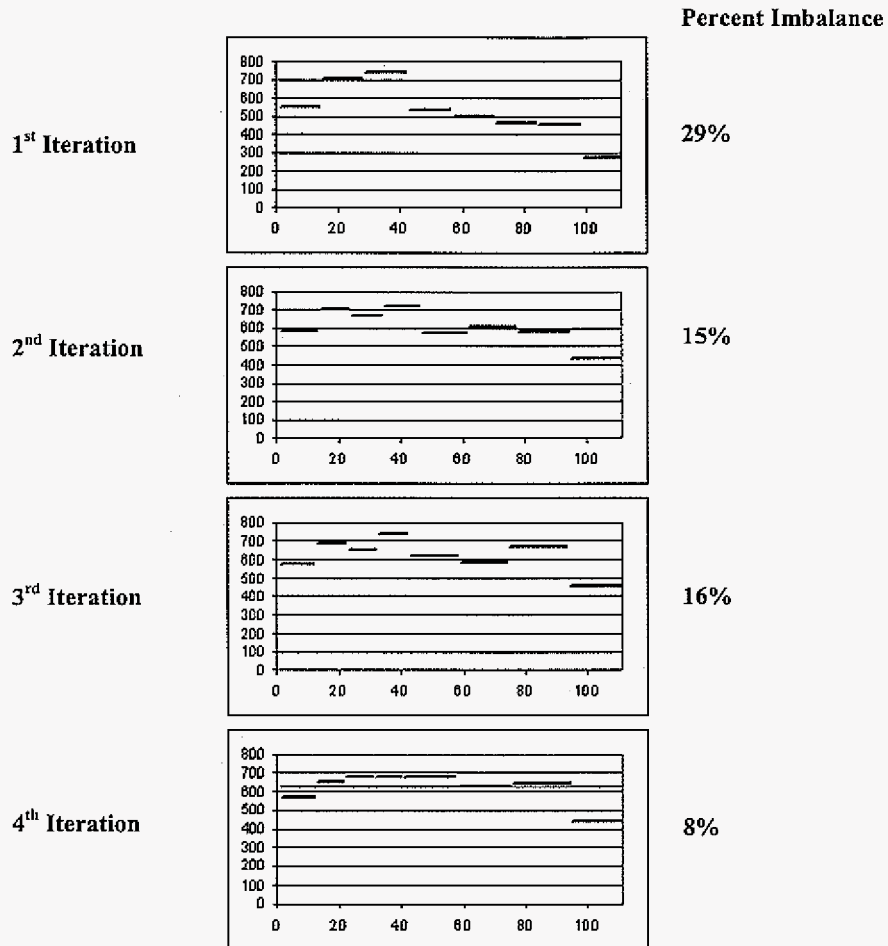


Figure 5. Workload and shells distribution of cAMP in the first four CPHF iterations; Shell number is the x axis and wall clock time is y axis

4. Cache optimization technique

Cache optimization is often underestimated or not taken into account in parallelization of code. A code running on a cluster of computers can potentially utilize cache on each CPU, vs. a sequential code for which only one cache is available. As the number of CPUs grows the data is partitioned so that it fits completely into the cache on

each CPU. In such a case a super linear speedup may be observed. This was achieved in the algorithm, developed by Korambath et al. [17].

In the DDCPHF code it is hard to achieve super linear speedup due to the manner in which parallelization is organized. Parallelization is implemented in the same way as SCF for which no cache optimization approach exists to achieve super linear scalability.

There is currently underway an effort to cache optimize the code for a replicated data parallel CPHF code. This will improve the performance of sequential CPHF and DDCPHF codes as well. The “copy method” [32] will be utilized to insure conflict misses are minimized. The density, Fock blocks, and integrals will be grouped in a unified block to increase data reuse and reduce possibility of cache conflicts. This method will be implemented via reorganizing the internal loop in Figure 4.

Results

The molecule calphostin C was chosen to test the scalability and overall performance of the DDCPHF algorithms. Calphostin C is a widely-used inhibitor of protein kinase C [33]. The basis set used for the calphostin C calculations is 6-31G (d,p). The choice of molecule and basis set was determined to design computationally demanding calculations that produce reliable benchmarking results.

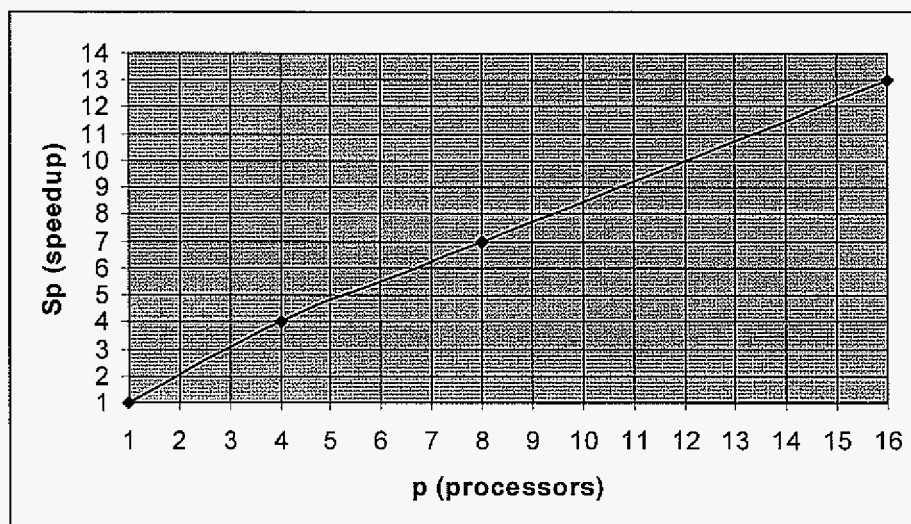


Figure 6. DDCPHF speedup; Tested system is calphostin C (1060 basis functions, 96 atoms)

Table 2. Time to complete different types of Hessian calculations

	Analytical Hessian, hrs	Numerical Hessian, hrs
Calphostin C	50	411

The speedup in Figure 6 and performance data in Table 2 are calculated based on wall clock time measured on the compute server. The longest wall clock time among all processors is used to compute each point in Figure 6. The DDCPHF code shows good scalability. The overall scalability of the Hessian is certainly not perfect. It may be possible to develop more sophisticated techniques to distribute the workload by redistributing shells in D and F. An advantage of the new algorithm does not have a built in scalability limit due to the network overhead. This is usually a common factor defining the scalability of an algorithm.

The advantage of using the new distributed data analytic Hessian code rather than a conventional numerical Hessian code is clear from Table 2. In the DDCPHF algorithm the ratio of computation time to communication time (4) is approximately four, the result of careful analysis of data flow. The algorithms were designed to minimize

communication overhead that often prevents scalability. The difference between performance of analytic and numerical Hessians should become even bigger as the size of system is increased.

Conclusions

An efficient and scalable distributed data algorithm has been developed that solves a major bottleneck in analytic Hessian calculations on distributed memory platforms. This novel algorithm has an excellent ratio of computation time to communication time and achieves good load balancing. The performance results demonstrate the good efficiency and scalability of the DDCPHF algorithm. Calculations on big chemical systems are now possible not only on large supercomputers but on Beowulf-class clusters where communication overhead is often a bottleneck.

The cluster solution for supercomputing is an attractive alternative to large massively parallel processor platforms due to a good performance/price ratio. The slow speed interconnection between nodes on clusters can be a source of poor performance and scalability of algorithms developed for MPPs. Typically, in distributed data quantum chemistry algorithms, bulk data is moved between processors. This means that to achieve good performance and scalability high bandwidth is required. It also means that new algorithms like that presented in this paper are required to efficiently utilize a cluster's potential.

Acknowledgement

The calculations in this work were performed on an IBM workstation cluster made possible by grants from IBM in the form of a Shared University Research grant, the Department of Energy, and a DURIP grant from the Air Force Office of Scientific Research.

The research reported here was made possible by a grant CHSSI from the Air Force Office of Scientific Research.

Theresa L. Windus was funded by the Office of Biological and Environmental Research in the U.S. Department of Energy. PNNL is operated by Battelle for the U.S. Department of Energy under contract DE-AC06-76RLO 1830.

References

1. C. Moler, M.S. Plesset, *Phys. Rev.* 1934, 46, 618.
2. T.D. Crawford, H.F. Schaefer III, 1996, Vol. 14, pp. 33-136.
3. J. Ivanic, K. Ruedenberg, *Theor. Chem. Accounts*, 106, 339-351 (2001)
4. M.T.B. Lam, S. T. Elbert, K. Ruedenberg, *Intern. J. Quantum Chem.* 31, 489-505 (1987).
5. M.J. Frisch, M. Head-Gordon, J.A. Pople, 1990, *Chem. Phys.*, 141, 189.
6. P. Pulay, 1969, *Molec. Phys.*, 17, 197.
7. J. Gerratt, I.M. Mills, 1968, *J. chem. Phys.*, 49, 1719.
8. J. A. Pople, R. Krishnan, H.B. Schelegel, J.S. Binkley, 1979, *Intl J. Quantum. Chem. Symp.*, 13, 225.
9. Y. Osamura, Y. Yamaguchi, H.F. Schaefer III, 1982, *J. chem. Phys.*, 77, 383.
10. M.J. Frisch, M. Head-Gordon, J.A. Pople, 1990, *Chem. Phys.*, 141, 189.

11. M.W. Schmidt, K.K. Baldrige, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L Windus, M. Dupuis, J.A. Montgomery, *J. Computational Chemistry*, 14 (1993) 1347-1363.
12. T.L. Windus, M.W. Schmidt, M.S. Gordon, *Chem. Phys. Lett.*, 216, 375 (1993).
13. A. M. Mórquez, J. Oviedo, J. F. Sanz, M. Dupuis, *J. Computational Chemistry*, Volume 18, Issue 2, (1997) 159-168.
14. M. Dupuis, J. Watts, H. Villar, G.Hurst, 1988, HONDO: Version 7.0 Documentation, IBM Kingston Technical Report KGN-169, and Quantum Chemistry Program Exchange Bulletin 8:2.
15. C.P. Sosa, J. Ochterski, J. Carpenter, M.J. Frisch, 1998. *J. Comput. Chem.*, 19, 1053.
16. Gaussian 98 (Revision A.1x), M.J. Frisch, et al., Gaussian, Inc., Pittsburgh PA, 2001.
17. P.P. Korambath, J. Kong , T.R. Furlani, M. Head-Gordon, *J. Molecular Physics*, Volume 100, Number 11/June 10, (2002) 1755 – 1761.
18. J. Kong, et al., *J. Comput. Chem.* (2000) 21 , 1532-1548.
19. J.A. Pople et. al *Quantum Chemistry Symposium* 13, 225-241 (1979).
20. P.E.S. Wormer, F. Visser, J. Paldus, *J. Comput. Phys.* 48, 23-44(1982).
21. M.W. Schmidt, G.D. Fletcher, B.M. Bode and M.S. Gordon, *Computer Physics Communications*, 128, 190 (2000).
22. Y. Alexeev, R.A. Kendall, M.S. Gordon, *Computer Physics Commun.*, 143, 69 (2002).
23. <http://www.beowulf.org> (Data accessed: November 26, 2002)
24. <http://www.scl.ameslab.gov/Projects/pCluster/> (Data accessed: November 26, 2002)

25. W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzburg, W. Saphir, M. Shir, MIT Press, 1998
26. J.A. Pople, W.J. Hehre, *J. Comput. Phys.* 27, 161-168 (1978)
27. H.F. King, M. Dupuis, *J. Comput. Phys.* 21, 144 (1976)
28. <http://www.scl.ameslab.gov/Projects/IBMCluster/Benchmarks.html> (Data accessed: November 26, 2002)
29. R. J. Harrison, R.A. Kendall, R.J. Littlefield, I.T. Foster, J. L. Tilson, A. F. Wagner, R. L. Shepard, *Journal of Computational Chemistry*, Vol. 17, No. 1, 109-123, 1996
30. R.J. Harrison, M. F. Guest, R.A. Kendall, D.E. Bernholdt, A.T. Wong, M. Stave, J.L. Anchell, A.C. Hess, R.J. Littlefield, G.L. Fann, J. Neiplocha, G.S. Thomas, D. Elwood, J.L. Tilson, R.L. Shepard, A.F. Wagner, I.T. Foster, E. Lusk, R. Stevens, *Journal of Computational Chemistry*, Vol. 17, No. 1, 124-132, 1996
31. A. Kolb, S. Busby, H. Buc, S. Garges, S. Adhya, *Annu. Rev. Biochem.* 62, 749-795, (1993)
32. (a) S. Chatterjee, S. Sen, *Proceedings of the 6th International Symposium on High-Performance Computer Architecture*, Toulouse, France, January 2000, pages 195-205.;
(b) S. Sen S. Chatterjee, *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, January 2000, pages 829-838.
33. M.Y. Berridge, A.S. Tan, *C. Biochemical and Biophysical Research Communications* 185: 806-11 (1992).

CHAPTER 5: A DISTRIBUTED DATA PARALLEL CASSCF ALGORITHM

A paper to be submitted for publication to

Journal of Computational Chemistry

Yuri Alexeev, Zhenting Gan, Michael W. Schmidt, Mark S. Gordon

Abstract:

A new parallelization scheme for the complete active space self consistent field (CASSCF) method is developed. The purpose is to have a highly scalable, distributed data CASSCF to perform computations on large chemical systems. The approach uses recently developed algorithms to achieve excellent scalability. The following codes are utilized: a highly scalable two-electron integral transformation code, a parallel full configuration interaction (FCI) code, and a second order self consistent field (SOSCF) code. The preliminary results are demonstrated with calculations on a moderate size system.

Keywords:

Quantum chemistry, complete active space self consistent field, CASSCF, full configuration interaction, FCI, second order self consistent field, SOSCF, DDI, Nonuniform Memory Access, Dynamic Load Balancing, Static Load Balancing.

Introduction

Computational quantum chemistry is a useful tool for providing reliable predictions of the structures and various properties of chemical compounds. A particularly important case is the study of chemical reactions in which bonds are forming and breaking. Another interesting case is the electronic excitation of an atom or molecule. In such cases, the wavefunction often has multiconfigurational character which requires multiconfigurational *ab initio* methods such as full configuration interaction (FCI) [1] or multiconfiguration self consistent field (MCSCF) [2] for a correct description of the electronic structure. Even for small systems, the FCI can be extremely expensive to calculate. MCSCF is usually a more computationally feasible method. The most commonly used version of MCSCF is the complete active space SCF (CASSCF) method [3]. In CASSCF the most chemically important orbitals and electrons are selected. They are usually referred as active orbitals and active electrons correspondingly. All possible configurations are generated by distributing active electrons in the active orbitals. CASSCF calculations consist of multiple steps: AO integral transformation, FCI computation within the active space, orbital optimization step. Each of these steps can become a computational and/or memory bottleneck.

There have been several other efforts to parallelize MCSCF. Dupuis et al. implemented a parallel first order MCSCF. The AO integral transformation was

parallelized, but the FCI computation and orbital optimization step were not parallelized. Therefore, this code has very limited scalability. Windus et al. developed a parallel algorithm for the integral transformations and GUGA based FCI in GAMESS [4]. The code has limited scalability on a small number of CPUs; this will be demonstrated in Section 4.

The purpose of this paper is to present a distributed data CASSCF (DDCASSCF) algorithm. To achieve excellent scalability and efficiency, recently developed state of the art algorithms and libraries were employed. The goal of DDCASSCF is to perform computations on large chemical systems. The code has been implemented in the *ab initio* quantum chemistry program GAMESS [5].

Tools and platforms

Beowulf-class clusters are an attractive option for small research groups and departments that usually have insufficient resources to purchase large computer systems. Currently, the most common types of clusters are workstations connected by a commodity network such as Fast Ethernet or Gigabit Ethernet. The DDI codes in GAMESS have been developed and tested on an IBM pSeries cluster [6]. This cluster consists of 32 IBM RS/6000 pSeries p640 servers connected by dual Fast Ethernet and dual Gigabit Ethernet. Each p640 has 4 Power3II processors running at 375MHz, 16 GB of memory, 73 GB local disk. The aggregate system has 512 GB of RAM and is capable of 192 GFLOPs peak performance.

A model based on the global memory access model is a convenient model for algorithms that require a sophisticated distribution of data with irregular access patterns.

The global memory access model is implemented in GAMESS in the DDI library [7]. At present, DDI uses only simple distributed memory operations based on point to point messages. In the DDCPHF algorithm the largest arrays are distributed and accessed via the DDI operations presented in Table 1.

Table 1. Distributed memory operations

DDI_CREATE	Create distributed matrix
DDI_DESTROY	Destroy distributed matrix
DDI_DISTRIB	Obtain distributed matrix distribution
DDI_GET	Get patch of distributed matrix
DDI_PUT	Put patch of distributed matrix
DDI_ACC	Accumulate patch of distributed matrix

Review of CASSCF theory

The complete active space SCF (CASSCF) method is the most widely used version of MCSCF. In CASSCF the most chemically important orbitals and electrons are selected. These are called active orbitals and electrons, respectively. All possible determinants are generated by distributing the active electrons in the active orbitals in all possible ways. Thus, one of the CASSCF steps is the FCI method.

The total CASSCF wavefunction Ψ_{casscf} is an antisymmetrized product of molecular orbitals (MO) ϕ_i which can be represented by Slater determinants Ψ_k :

$$\Psi_{\text{casscf}} = \sum_k A_k \Psi_k \quad (1)$$

$$\Psi_k = \hat{A} \left\{ \prod_{i < k} \phi_i \right\} \quad (2)$$

where A_k are the CI coefficients.

Each molecular orbitals ϕ_i is expanded in an atomic basis set χ_μ :

$$\phi_i = \sum_{\mu} C_{\mu i} \chi_{\mu} \quad (3)$$

where $C_{\mu i}$ are the MO expansion coefficients. Both A_k and $C_{\mu i}$ coefficients are variationally optimized. The orbitals ϕ_i can be divided into several subsets: core, active, and virtual MO orbitals. The following notation will be used to label different types of orbitals:

i, j, k, l – core MO orbitals
 a, b, c, d – virtual MO orbitals
 t, u, v, w – active MO orbitals
 p, q, r, s – general MO orbitals
 $\mu, \nu, \lambda, \sigma$ – AO orbitals

The energy expression for a CASSCF wavefunction is

$$E = \sum_{pq} \gamma_{pq} h_{pq} + \frac{1}{2} \sum_{pqrs} \Gamma_{pqrs} (pq | rs) \quad (4)$$

where h_{pq} and $(pq | rs)$ are one and two electron integrals respectively; γ_{pq} and Γ_{pqrs} are one and two body density matrices.

To minimize the energy (4) with respect to both A_k and $C_{\mu i}$ coefficients, the frequently used “unfolded, two step” approach is employed in GAMESS. In the first step, the Hamiltonian is constructed and energy minimization $\frac{\partial E}{\partial A_k} = 0$ leads to CI coefficients A_k . On the second step given A_k , the orbital improvement scheme generates $C_{\mu i}$ coefficients with fixed coefficients A_k . The procedure is repeated until convergence is achieved.

One of the most advantageous orbital improvement schemes is the approximate second order Newton-Raphson method:

$$x = -B^{-1}g \quad (5)$$

where x is a vector of independent rotational parameters, B is the orbital Hessian of the energy matrix, g is the orbital gradient of the energy vector. The independent rotational parameters x were utilized since not all MO expansion coefficients $C_{\mu i}$ are independent variables. It is desirable to preserve orthogonality of the starting MO orbitals without introducing Lagrange multipliers. So the orthogonal matrix U is constructed from vector x elements. The orthogonal matrix U can be written as the exponential of an antisymmetric matrix T :

$$U = \exp(T) = I + T + \frac{1}{2}T^2 + \dots \quad (6)$$

where T is

$$T = \begin{bmatrix} 0 & x \\ -x & 0 \end{bmatrix} \quad (7)$$

In the current implementation expansion $\exp(T)$ (6) is truncated at the first order.

The improved MO orbital coefficient matrix C_{new} is obtained iteratively from C_{old} and U :

$$C_{new} = C_{old}U \quad (8)$$

The orbital Hessian matrix B is a large matrix. In the approximate second order Newton-Raphson method only the diagonal elements are calculated [8]. The inverse orbital Hessian is updated using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula [9].

The gradient formula needed to compute x is derived from the Lagrangian formula (generalized Fock operator):

$$L_{pq} = \sum_i \gamma_{pi} h_{pi} + \sum_{i,j,k} \Gamma_{pijk} [pi | jk] \quad (9)$$

The Lagrangian formula can be derived from the minimization of the energy expression for a CASSCF wavefunction (4) subject to orthonormalization constraints.

The Lagrangian is the matrix which has the following arrangement:

	Core	Active	Virtual
Core	0	$L_{\text{core-active}}$	$L_{\text{core-virtual}}$
Active	$L_{\text{active-core}}$	$L_{\text{active-active}}$	$L_{\text{active-virtual}}$
Virtual	0	0	0

At convergence $L_{\text{active-active}}$, $L_{\text{core-virtual}}$, $L_{\text{active-virtual}}$, and $L_{\text{active-core}}$ are zero since L should be symmetric at convergence. Eliminating $L_{\text{core-virtual}}$ and $L_{\text{active-virtual}}$ at convergence means that the single excitation element is zero. This is simply the Brillouin theorem [10,11,12].

There is also another Brillouin-Levy-Berthier theorem that states that at convergence

$$g_{\text{core-active}} = L_{\text{core-active}} - L_{\text{active-core}} = 0$$

$$g_{\text{core-virtual}} = L_{\text{core-virtual}} - L_{\text{virtual-core}} = 0$$

$$g_{\text{active-virtual}} = L_{\text{active-virtual}} - L_{\text{virtual-active}} = 0$$

The detailed formulas for the gradients are

$$\text{Core-virtual: } g_{ia} = 4(F_{ai}^{\text{core}} + F_{ai}^{\text{act}}) \quad (10)$$

$$\text{Active-virtual: } g_{ta} = 2 \sum_u \gamma_{tu} F_{au}^{\text{core}} + 2 \sum_{uvw} \Gamma_{uvw} (au | vw) \quad (11)$$

$$\text{Core-active: } g_{it} = 4(F_{it}^{\text{core}} + F_{it}^{\text{act}}) - 2 \left[\sum_u \gamma_{tu} F_{iu}^{\text{core}} + \sum_{uvw} \Gamma_{uvw} (iu | vw) \right] \quad (12)$$

where F^{core} and F^{act} are defined as

$$F_{pq}^{\text{core}} = h_{pq} + \sum_k^{\text{core}} [2(pq | kk) - (pk | qk)] \quad (13)$$

$$F_{pq}^{act} = \sum_{uv} \gamma_{uv} \left[(pq | uv) - \frac{1}{2} (pu | qv) \right] \quad (14)$$

The gradient is always computed exactly because this is the convergence criteria of MCSCF. Note that to calculate the gradient vector g the following classes of MO integrals are needed:

(aa|aa)

(ca|aa)

(va|aa)

where “a” is an active orbital, “c” is core orbital, “v” is a virtual orbital.

However, the orbital Hessian B [13] can be approximated. Chaban et al. [14] suggested to approximate the diagonal of matrix B to avoid computation of the MO integrals other than those needed for gradient computation. The approximate diagonal Hessian elements are

$$\text{Core-virtual: } B_{ia,ia} = 4(F_{aa}^{core} + F_{aa}^{act}) - 4(F_{ii}^{core} + F_{ii}^{act}) \quad (15)$$

$$\text{Active-virtual: } B_{ia,ia} = 2\gamma_u F_{au}^{core} - 2\sum_u \gamma_u F_{tu}^{core} - 2\sum_{uvw} \Gamma_{uvw}(tu | vw) + 2\gamma_u F_{aa}^{act} \quad (16)$$

$$\begin{aligned} \text{Core-active: } B_{ii,ii} &= 4(F_u^{core} + F_u^{act}) - 4(F_{ii}^{core} + F_{ii}^{act}) + 2\gamma_u F_{ii}^{core} - \\ &2\sum_u \gamma_u F_{tu}^{core} - 2\sum_{uvw} \Gamma_{uvw}(tu | vw) + 2\gamma_u F_{ii}^{act} \end{aligned} \quad (17)$$

Note that to calculate both g and B , only F^{core} , F^{act} , and $(pu | vw)$ are needed. F^{core} and F^{act} can be calculated from AO integrals and via a similarity transformation which transforms F^{core} and F^{act} into appropriate MO space. The product of two body density and MO integrals $\sum_{u,v,w} \Gamma_{uvw}(pu | vw)$ needed in (11) and (12) can be calculated from the half transformed integrals $(\mu\nu | vw)$:

$$\sum_{u,v,w} \Gamma_{uvw}(pu | vw) = \sum_{u,v,w} \Gamma_{uvw} \sum_{\mu,\nu} C_{\mu p} C_{\nu u} (\mu\nu | vw) = \sum_{\nu} \sum_{\mu,\nu} C_{\mu p} C_{\nu u} \sum_{vw} \Gamma_{uvw} (\mu\nu | vw) \quad (18)$$

Therefore, only two types of integrals are needed for the whole CASSCF macroiteration:

All active MO integrals: $(tu | uv)$

Half transformed integrals: $(\mu\nu | uv)$

CASSCF parallelization strategy

Currently, CASSCF is partially parallelized. Only the integral transformation step is parallelized by Windus et al. which is based on HONDO's code [15]. The current FCI determinant code is not parallel. The molecule stilbene was chosen to test the performance of the current code on one macro iteration. The input file parameters are basis set: 6-311(d,p) (338 bf), group=C₁, number of core orbitals ncore=41, number of active orbitals nact=14, number of active electrons nels=14. The performance and scalability of the code is demonstrated in Table 2.

Table 2. CASSCF wall clock timings for one macroiteration on 1,4, and 16 CPUs; CPU time in parenthesis, DM is computation of CI density matrix, SOSCF is orbital improvement scheme described in the previous section

CPU	1	4	16
2-e AO integrals	366(363)	662(407)	895(373)
Integral transf.	348(344)	625(250)	1667(126)
FCI	8574(8573)	10001(8654)	8953(8933)
DM	1186(1185)	1193(1191)	1403(1192)
SOSCF	78(77)	534(99)	2102(99)
Total time	10559(10550)	11828(10608)	15026(10728)

The overall performance of the code is not satisfactory. There are a few reasons that contribute to poor performance. The computation of 2-e AO integrals in the first step is replicated because all AO integrals on each CPU are needed for integral transformation in the current code. AO integrals, MO integrals, CI density matrix, and half transformed

integrals are stored on hard disk to reduce RAM memory requirements per CPU. This strategy is hardly applicable to clusters of SMP workstations because the I/O performance is often not addressed there. On the IBM workstation cluster where the code was tested, each box consists of four CPUs, two Gigabit Ethernet cards, and one Ultra SCSI I/O controller. This particular Ultra SCSI I/O controller (one of the best available) can sustain bandwidth 80 MB/sec or 20 MB/sec per CPU. The network bandwidth is approximately 40 MB/sec per CPU. For comparison, the internal memory bandwidth for IBM Power3 series is 275 MB/sec per CPU. The failure of this approach can be clearly seen in integral transformation step. CPU timing has descent scaling, whereas wall clock time increased five times because of I/O on 16 CPUs compared to performance on one CPU. The distributed data algorithms including those used in parallelization of CASSCF have minimum HD I/O and network bandwidth requirements. The data in the large arrays is recomputed, if it is too expensive to recompute then data is stored in distributed data arrays.

In the DDCASSCF algorithm, new recently developed algorithms and new parallelization schemes are employed to achieve good scalability and performance. The proposed DDCASSCF algorithm is reviewed step by step below.

1. Generate the following MO integrals distributed among nodes using the integral transformation adopted from the DDMP2 code written by Fletcher et al.:

$(tu|vw)$: for the FCI step and orbital Hessian Equations (16), (17)

$(\mu\nu|vw)$: the gradients in Equations (11), (12) are computed from half transformed integrals $(\mu\nu|vw)$

2. Compute F^{core} from AO integrals. F^{core} is needed not only in calculations of g and B but in the FCI step too

3. Generate CI coefficients A_k using the DDFCI code [16] where the CI vectors are distributed among nodes
4. Generate γ_{pq} and Γ_{pqrs} , respectively one and two body CI density matrices from CI coefficients A_k
5. Parallel orbital optimization step SOSCF generates C_{μ} coefficients

The most time consuming operations by far in SOSCF are computation of F^{core} and F^{act} from AO integrals. They are essentially Fock builds with different density matrices which have to be transformed into appropriate MO space via a similarity transformation. F^{core} matrix is needed for FCI step so it is calculated in the step 2. The AO integrals are computed and written to disk. F^{act} is computed directly from stored AO integrals. Both steps are scaling as good as SCF which is almost perfect up to 32 CPUs.

The estimated performance and scalability of code is demonstrated in Table 3 on the same molecule stilbene with the same input parameters as at the beginning of this section.

Table 3. DDCASSCF wall clock timings for one macroiteration on 1,4, and 16 CPUs; CPU time in parenthesis

CPU	1	4	16
Integral transf.	2635(2619)	674(665)	203(181)
F^{core}	366(363)	92(91)	23(22)
FCI	4460(4460)	1199(1115)	363(338)
DM	1186(1185)	297(296)	89(89)
SOSCF	78(77)	20(20)	6(6)
Total time	8725(8704)	2282(2187)	684(636)

The integral transformation step in Table 3 is the time to compute the distributed data MO integrals (oo|oo). The code is adopted from the distributed data MP2 code. Since the code was originally developed for MP2 the unnecessary MO integrals (cc|cc) are

computed too which is responsible for the difference in time to transform integrals on one CPU in Table 2 and Table 3. The code will be modified further to generate only the necessary MO integrals (aa|aa) and the half-transformed integrals. Thus, it will decrease the time required for the integral transformation step significantly. Computation of F^{core} matrix in Table 3 is the time to build the Fock matrix from AO integrals which are saved to disk. Timings for distributed data parallel FCI and DM steps are provided by Z. Gan. The code will be further improved in the computation of CI density matrix (DM step) because the time to compute the CI density matrix is too large. The most expensive operation in SOSCF is computation of F^{act} which is computed directly from stored AO integrals. This step has same scaling as computation of a Fock matrix from stored AO integrals.

Conclusions

A new efficient and scalable distributed data CASSCF algorithm is under development using recently developed highly scalable codes and parallelization schemes. The largest arrays such as CI vectors and half transformed integrals are distributed among nodes. Therefore, DDCASSCF will be able to perform calculations on big chemical systems. This is the major problem of the currently employed CASSCF method. DDCASSCF will have excellent performance. Further improvements of code performance are under way.

Acknowledgement

The calculations in this work were performed on an IBM workstation cluster made possible by grants from IBM in the form of a Shared University Research grant, the Department of Energy, and a DURIP grant from the Air Force Office of Scientific Research.

The research reported here was made possible by a SciDAC grant to the Ames Laboratory from the Department of Energy.

References

1. J.Ivanic, K.Ruedenberg *Theoret.Chem.Acc.* 106, 339-351(2001).
2. M.W. Schmidt and M.S. Gordon, *Ann. Rev. Phys. Chem.* **49**, 233 (1998).
3. B.O.Roos, in "Advances in Chemical Physics", vol.69, edited by K.P.Lawley, Wiley Interscience, New York, 1987, pp 339-445.
4. T.L. Windus, M.W. Schmidt, and M.S. Gordon, *Theor. Chim. Acta*, **89**, 77 (1994).
5. 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, M. Dupuis, and J.A. Montgomery, Jr., *J. Comp. Chem.*, **14**, 1347 (1993).
6. <http://www.scl.ameslab.gov/Projects/pCluster/> (Data accessed: November 26, 2002)
7. M.W. Schmidt, G.D. Fletcher, B.M. Bode and M.S. Gordon, *Computer Physics Communications*, **128**, 190 (2000).
8. T.H. Fischer, J.E. Almlof (1992) *J Phys Chem* **96**: 9768
9. R. Fletcher, "Practical methods of optimization", vol. 1. Wiley, New York (1980)
10. B. Levy, G. Berthier, *Int. J. Quantum Chem.* **2**: 307 (1969); *Int. J. Quantum Chem.* **3**: 247 (1969)

11. K. Ruedenberg, L.M. Cheung, S.T. Elbert *Int. J. Quantum Chem.* 16: 1069 (1979)
12. B.O. Roos. *Int. J. Quantum Chem.* S14: 175 (1980)
13. P. Siegbahn, A. Heiberg, B. Roos, B. Levy *Physica Scripta* 21: 323 (1980)
14. G. Chaban, M.W. Schmidt, and M.S. Gordon, *Theor. Chem. Accts.*, 97, 88 (1997).
15. M. Dupuis, S. Chin, A. Marquez (1994) In: G.L. Malli (ed) M. Dupuis, S. Chin, A. Marquez (1994) In: G.L. Malli (ed). Plenum Press, New York, p 315
16. Z. Gan, Y. Alexeev, R. Kendall, J. Ivanic, M.W. Schmidt, M.S. Gordon, in preparation

CHAPTER 6: GENERAL CONCLUSIONS

All papers presented in the thesis have conclusions. Therefore, only general conclusions will be presented in chapter 6.

In the second chapter, titanium chloride (II) is investigated as a potential catalyst for the bis-silylation reaction. This paper falls into line with other papers recently published in the Gordon group that demonstrates the effectiveness of titanium as a catalyst. Ti is an electron deficient atom, so Ti readily forms additional bonds beyond the “usual” four. In the studied reactions, Ti in the initial steps binds with a π donor reactant which results in large energy decrease. This large energy decrease ensures that the activation barriers for all subsequent steps are below the energy of the reactants. In the transition states of subsequent steps Ti typically forms more than four bonds which result in relatively small activation barriers compared to the energy of the reactants. We conclude that divalent Ti has the potential to become an important industrial catalyst for silylation reactions. The mechanism of bis-silylation reactions was studied in detail. The investigation of Pt as an effective catalyst for bis-silylation reactions is underway.

In this thesis, parallelization of different quantum chemistry methods is presented. The parallelization of code is becoming important aspect of quantum chemistry code development. Two trends contribute to it: the overall desire to study large chemical systems and the desire to employ highly correlated methods which are usually computationally and memory expensive. *Ab initio* methods can provide reliable energetics and molecular properties for chemical reactions. In recent years because of that, quantum chemistry is becoming a valuable tool to study important biological reactions that result in better

understanding about how nature works. It may ultimately lead to finding better drugs to cure people, prolong people's lives and other applications. The main obstacle is the typical size of biologically important molecules. The quantum chemistry methods that can be applied are usually very simple methods such as RHF and DFT. The highly accurate methods such as CASSCF, MP2 and others are not applicable. This problem is addressed in the thesis. The presented distributed data SCF increases the size of chemical systems that can be calculated by using RHF and DFT. The important *ab initio* method to study bond formation and breaking as well as excited molecules is CASSCF. The presented distributed data CASSCF algorithm can significantly decrease computational time and memory requirements per node. Therefore, large CASSCF computations can be performed. The most time consuming operation to study potential energy surfaces of reactions and chemical systems is Hessian calculations. The distributed data parallelization of CPHF will allow scientists carry out large analytic Hessian calculations. The parallelization of other quantum chemistry methods is underway.

APPENDIX: SUPPLEMENTARY MATERIALS

Table S1. MP2 total energies with ZPE corrections in Hartrees

Structure	MP2 Energy	MP2 + MP2 ZPE Energy
Oxidative addition		
R	-262.50226	-262.43524
M1	-262.59864	-262.53023
M2	-262.61475	-262.54507
TS1	-262.5828	-262.51201
M3	-262.6146	-262.54348
Ethylene insertion		
M4	-262.6031	-262.53234
TS2	-262.5969	-262.52621
M5	-262.5981	-262.52744
M6	-262.59536	-262.52428
TS3	-262.5895	-262.51855
M7	-262.63499	-262.56384
Reductive elimination		
M8	-262.63703	-262.56623
M9	-262.67181	-262.59928
M10	-262.59626	-262.52355
P	-262.56461	-262.4929

Table S2. The Cartesian coordinates of each stationary point in Å

Structure		Cartesian coordinates		
R				
C ₂ H ₂				
C	6	2.510704	0.708912	3.268617
H	1	2.3028	1.769408	3.266477
H	1	2.00058	0.112671	4.011646
C	6	3.358135	0.161058	2.394191
H	1	3.868259	0.757346	1.651159
H	1	3.56607	-0.89943	2.396284
Si ₂ Cl ₆				
SI	14	0.094767	-0.10908	-1.16864
SI	14	-0.10036	0.121383	1.169473
CL	17	1.640207	0.909694	1.924104
CL	17	-0.44212	-1.72113	2.01135
CL	17	-1.66785	1.37633	1.603596
CL	17	-1.64751	-0.89064	-1.92621
CL	17	1.658365	-1.36917	-1.60202
CL	17	0.443124	1.732963	-2.00884
TiCl ₂				
CL	17	-0.84389	0	-2.14549
TI	22	0.047493	0	-0.01989
CL	17	0.938856	0	2.105719
M1				
TiCl ₂ -C ₂ H ₂				
TI	22	-0.42276	2.117191	-0.49764
CL	17	-2.37642	1.453937	0.329696
CL	17	0.877375	1.168644	-2.03146
C	6	-0.11568	4.126405	-0.3163
C	6	0.59122	3.371178	0.752935
H	1	0.201957	3.450499	1.764437
H	1	1.675436	3.321452	0.699286
H	1	-0.98123	4.715883	-0.026
H	1	0.492072	4.584973	-1.09175
M2				
SI	14	0.45664	0.25742	-0.76333
CL	17	1.6126	-0.85709	0.51024
CL	17	1.34414	0.40249	-2.59577
CL	17	0.39562	2.18085	0.09714
TI	22	-1.44832	3.87206	-0.91826
CL	17	-3.1472	3.04965	0.27311
CL	17	-0.93347	3.4885	-3.05804
CL	17	-2.56331	-0.71885	0.92946
SI	14	-1.69146	-0.71678	-0.92058
CL	17	-1.39186	-2.64838	-1.55245
CL	17	-2.84409	0.27672	-2.28401
C	6	-1.4574	5.91327	-0.63962
C	6	-0.31454	5.30889	0.05362
H	1	-0.33542	5.26692	1.13974

H	1	0.67847	5.46965	-0.35873
H	1	-2.2882	6.2786	-0.0433
H	1	-1.26869	6.4876	-1.54206
H	1	-1.26869	6.4876	-1.54206
TS1				
SI	14	0.71669	-1.26645	0.223617
CL	17	0.755848	-1.08996	2.322236
CL	17	2.659266	-1.42655	-0.38865
CL	17	-0.26255	-3.00832	-0.20243
CL	17	-1.671	-0.70744	-2.50848
SI	14	-0.31584	0.426045	-1.48235
CL	17	1.256577	0.861445	-2.71219
CL	17	-1.26552	2.253735	-1.04123
CL	17	1.62113	1.888691	0.808931
TI	22	-0.45683	1.030207	1.207624
CL	17	-2.346	-0.24569	1.077524
C	6	-0.71767	1.771691	3.194591
H	1	0.244714	1.984217	3.646886
H	1	-1.36875	1.114546	3.760439
C	6	-1.28046	2.694442	2.264976
H	1	-2.35574	2.732517	2.129906
H	1	-0.74253	3.602738	2.016801
M3				
Si	14	0.85033	-1.48356	0.48456
Cl	17	0.72846	-0.98751	2.61177
Cl	17	2.84197	-1.74052	0.02993
Cl	17	-0.06714	-3.30901	0.22901
Cl	17	-1.89897	-0.29654	-2.80437
Si	14	-0.46608	0.68131	-1.69532
Cl	17	1.01013	1.27199	-3.00347
Cl	17	-1.3899	2.49616	-0.89608
Cl	17	1.64414	1.78692	0.69072
Ti	22	-0.31529	0.72023	0.81467
Cl	17	-2.28215	-0.33002	0.95942
C	6	-0.73873	1.80658	3.18595
H	1	0.28619	1.90226	3.51855
H	1	-1.3407	1.02509	3.62988
C	6	-1.27138	2.68246	2.30397
H	1	-2.31334	2.62459	2.01927
H	1	-0.68647	3.50177	1.90793
H	1	-0.68647	3.50177	1.90793
M4				
SI	14	0.944844	-1.49742	0.155723
CL	17	0.772916	-1.85959	2.36296
CL	17	2.95636	-1.57006	-0.23595
CL	17	0.037576	-3.12196	-0.70615
CL	17	-2.27424	1.61768	-1.61311
SI	14	-0.32327	1.261865	-0.93982
CL	17	0.503219	0.072752	-2.47358
CL	17	0.684631	3.084875	-1.14684
CL	17	1.647236	1.377278	2.048599

TI	22	-0.21845	0.289189	1.524645
CL	17	-2.22762	-0.65134	1.402166
C	6	-1.49821	2.709662	1.691436
H	1	-2.35803	2.546777	1.056668
H	1	-0.7263	3.376848	1.333471
C	6	-1.43618	2.172304	2.932175
H	1	-0.61651	2.405068	3.599397
H	1	-2.24992	1.574103	3.320365
TS2				
Si	14	1.00439	-1.52631	0.36736
Cl	17	1.12943	-2.27424	2.34209
Cl	17	2.92545	-1.22964	-0.3219
Cl	17	0.12465	-3.02712	-0.73956
Cl	17	-2.28176	1.88835	-1.54695
Si	14	-0.34613	1.21686	-1.07954
Cl	17	0.13467	-0.03901	-2.6598
Cl	17	0.95067	2.84526	-1.28105
Cl	17	1.36947	1.20427	2.58237
Ti	22	-0.33817	0.42026	1.4389
Cl	17	-2.1237	-0.8087	1.71724
C	6	-1.05901	3.16514	1.43198
H	1	-1.39679	3.51736	0.46999
H	1	-0.09574	3.50976	1.78036
C	6	-1.85717	2.38996	2.20114
H	1	-1.57534	2.12493	3.21506
H	1	-2.84745	2.11022	1.86821
H	1	-2.84745	2.11022	1.86821
M5				
Si	14	1.01796	-1.53281	0.5029
Cl	17	1.46944	-2.52844	2.28389
Cl	17	2.82084	-1.01629	-0.37869
Cl	17	0.07306	-2.94414	-0.68152
Cl	17	-2.22208	2.00402	-1.60347
Si	14	-0.33088	1.19112	-1.15617
Cl	17	0.04551	-0.0873	-2.73553
Cl	17	1.07039	2.73407	-1.33985
Cl	17	1.15568	1.06387	2.7943
Ti	22	-0.42618	0.48479	1.40049
Cl	17	-2.08753	-0.8269	1.88014
C	6	-0.98387	3.27456	1.35539
H	1	-1.13305	3.63882	0.35129
H	1	-0.08326	3.58134	1.8699
C	6	-1.92656	2.52517	1.96991
H	1	-1.82867	2.24806	3.01405
H	1	-2.85259	2.27441	1.46795
M6				
C	6	3.33648	0.17176	-0.32291
H	1	3.98836	-0.12372	0.49127
H	1	3.33982	-0.44745	-1.21097
Ti	22	1.18232	-0.67268	0.58644
Cl	17	1.29937	-0.74209	2.7523

Cl	17	1.85136	-2.71223	-0.00627
Si	14	-1.17257	-1.03862	-0.35977
Cl	17	-1.74357	-2.97416	-0.79316
Cl	17	-2.84437	0.10887	-0.67016
Cl	17	0.26405	-0.40911	-1.89053
Si	14	-0.03202	1.7349	1.14085
Cl	17	-1.72504	1.50896	2.34383
Cl	17	1.21327	2.91626	2.37086
Cl	17	-0.56003	3.01417	-0.42905
C	6	2.65826	1.34838	-0.28531
H	1	2.76781	2.01903	0.55361
H	1	2.09544	1.70118	-1.13878
H	1	2.09544	1.70118	-1.13878
TS3				
C	6	2.89277	0.44928	-0.18593
H	1	3.66973	0.47904	0.5692
H	1	3.12383	-0.10079	-1.09564
Ti	22	1.2607	-0.79751	0.45937
Cl	17	1.34868	-0.68766	2.63933
Cl	17	1.99817	-2.84269	0.07729
Si	14	-1.13945	-0.88843	-0.39786
Cl	17	-1.90975	-2.80303	-0.35907
Cl	17	-2.6732	0.32447	-1.03137
Cl	17	0.33292	-0.79312	-2.02514
Si	14	0.06528	1.81767	1.11277
Cl	17	-1.59994	1.22631	2.22974
Cl	17	1.16408	2.86385	2.55025
Cl	17	-0.66525	3.22649	-0.22851
C	6	1.93488	1.49324	-0.25833
H	1	2.15911	2.39534	0.29782
H	1	1.43364	1.65676	-1.20413
H	1	1.43364	1.65676	-1.20413
M7				
C	6	2.227296	0.677983	0.113923
H	1	2.87831	0.729726	0.995277
H	1	2.777445	0.159947	-0.67863
Ti	22	0.887433	-0.88247	0.703131
Cl	17	0.908181	-1.01148	2.899019
Cl	17	2.006123	-2.67082	0.097217
Si	14	-1.28559	-1.70028	-0.42572
Cl	17	-1.50198	-3.57499	-1.26392
Cl	17	-3.17325	-0.84973	-0.42279
Cl	17	-0.08063	-0.52782	-1.82561
Si	14	0.625798	2.782787	0.895208
Cl	17	-0.84097	1.282815	1.116224
Cl	17	1.495592	3.056486	2.728675
Cl	17	-0.33822	4.495149	0.317625
C	6	1.79719	2.066055	-0.35354
H	1	2.645753	2.742324	-0.49727
H	1	1.256274	2.010709	-1.30071
M8				

C	6	0.129437	0.958676	-0.60775
H	1	1.102431	1.412524	-0.84204
H	1	-0.28026	0.587862	-1.54812
TI	22	0.601307	-0.61118	0.729214
CL	17	-0.94995	-1.13635	2.196706
CL	17	2.626604	-1.43604	0.601378
SI	14	-0.41547	-2.16748	-1.09104
CL	17	-0.39452	-4.17097	-0.54921
CL	17	-2.40046	-1.64209	-1.46379
CL	17	0.585985	-2.00704	-2.91022
SI	14	-0.0384	2.914255	1.418972
CL	17	-1.31397	3.50142	2.903879
CL	17	1.367684	1.585978	2.272842
CL	17	1.038726	4.516203	0.734318
C	6	-0.84732	1.963718	0.030909
H	1	-1.19417	2.699026	-0.70296
H	1	-1.73852	1.465707	0.42005
M9				
C	6	1.362261	0.346123	0.526789
H	1	1.458953	0.425735	1.612463
H	1	2.130025	-0.36177	0.199207
TI	22	-0.14975	-3.1271	0.625278
CL	17	-2.18645	-3.71883	0.157881
CL	17	0.461423	-3.06025	2.712448
SI	14	-0.2787	-0.54651	0.197403
CL	17	1.417407	-3.62234	-0.80296
CL	17	-1.77563	0.197548	1.437253
CL	17	-0.82239	-0.23543	-1.79039
SI	14	0.706324	3.207079	0.472954
CL	17	0.614954	3.178601	2.530139
CL	17	1.760131	4.880649	-0.11501
CL	17	-1.19277	3.360955	-0.29157
C	6	1.626612	1.697712	-0.15722
H	1	2.683552	1.951616	-0.03627
H	1	1.443692	1.636327	-1.23224
M10				
CL	17	1.381678	0.986477	2.291924
SI	14	0.855777	2.070077	0.633065
CL	17	-1.19488	2.254611	0.568529
CL	17	1.675289	3.951172	0.73579
C	6	1.490905	1.226711	-0.9206
H	1	2.552781	1.480119	-0.96996
H	1	1.016308	1.717828	-1.77301
C	6	1.358035	-0.30249	-1.00713
H	1	1.785861	-0.80131	-0.13395
H	1	1.925077	-0.66322	-1.87089
CL	17	-1.49683	-0.01921	-2.59023
CL	17	-0.09224	-3.01564	-1.91737
SI	14	-0.33923	-1.04154	-1.25831
CL	17	-1.3778	-1.35093	0.528673
TI	22	-0.41368	-3.83588	0.609791

CL	17	1.720284	-3.31196	1.359398
CL	17	-2.28349	-5.17626	0.535566
P				
SiCl ₃ C ₂ H ₄ SiCl ₃				
CL	17	0.83278	1.14018	2.48915
SI	14	0.66578	2.40854	0.56363
CL	17	-1.24961	3.00187	0.15594
CL	17	1.80143	4.0788	0.8928
C	6	1.46938	1.46481	-0.85001
H	1	2.53324	1.69746	-0.75569
H	1	1.12814	1.95096	-1.76757
C	6	1.32183	-0.06155	-0.94898
H	1	1.7344	-0.56796	-0.07417
H	1	1.95711	-0.41615	-1.76822
CL	17	-1.15735	0.3068	-2.98894
CL	17	-0.25873	-2.74408	-1.92651
SI	14	-0.38635	-0.77414	-1.42503
CL	17	-1.71823	-0.55551	0.16483