Linear Scaling Computation of the Fock Matrix. VI. Data Parallel Computation of the Exchange-Correlation Matrix.[§]

Chee Kwan Gan[†] and Matt Challacombe[‡]

Theoretical Division,
Group T-12, MS B268,
Los Alamos National Laboratory,
Los Alamos, New Mexico 87545

(Dated: Feb 20, 2003)

Recently, early onset linear scaling computation of the exchange-correlation matrix has been achieved using Hierarchical Cubature [J. Chem. Phys. **113**, 10037 (2000)]. Hierarchical Cubature differs from other methods in that the integration grid is adaptive and purely Cartesian, which allows for a straightforward domain decomposition in parallel computations; the volume enclosing the entire grid may be simply divided into a number of non-overlapping boxes. In our data parallel approach, each box requires only a fraction of the total density to perform the necessary numerical integrations due to the finite extent of Gaussian-Orbital basis sets. This inherent data locality may be exploited to reduce communications between processors as well as to avoid memory and copy overheads associated with data replication. Although the Hierarchical Cubature grid is Cartesian, naive boxing leads to irregular work loads due to strong spatial variations of the grid and the electron density. In this paper we describe **Equal Time partitioning**, which employs time measurement of the smallest sub-volumes (corresponding to the primitive cubature rule) to load balance grid-work for the next self-consistent-field iteration. After start up from a heuristic Center of Mass partitioning, Equal Time partitioning exploits smooth variation of the density and grid between iterations to achieve load balance. With the 3-21G basis set and a medium quality grid, Equal Time partitioning applied to taxol (62 heavy atoms) attained a speedup of 61 out of 64 processors, while for a 110 molecule water cluster at standard density it achieved a speedup of 113 out of 128. The efficiency of Equal Time partitioning applied to Hierarchical Cubature improves as the grid work per processor increases. With a fine grid and the 6-311G(df,p) basis set, calculations on the 26 atom molecule α-pinene achieved a parallel efficiency better than 99% with 64 processors. For more coarse grained calculations, superlinear speedups are found to result from reduced computational complexity associated with data parallelism.

Keywords: Density Functional Theory, linear scaling, numerical integration, Gaussian-Orbital, hierarchical methods, adaptive grid, load balance, parallel computation

I. INTRODUCTION

Density Functional Theory (DFT) and its variant, the hybrid Hartree-Fock/Density Functional Theory (HF/DFT) have proven to be accurate and computationally efficient. These model chemistries are routinely used by conventional Gaussian-Orbital Quantum Chemistry codes to excellent advantage. Recently, significant progress has been achieved in the algorithmic development of $O(N)$ alternatives to these conventional methods, where compute times scale linearly with system size N [1, 2]. These linear scaling methods overcome a number of bottlenecks of order $O(N^{2-3})$ associated with conventional methods, including computation of the exact Hartree-Fock exchange matrix [3–8], the Coulomb matrix [9–14], the exchange-correlation matrix [15–18], and density matrix alternatives to eigensolution of the Self-Consistent-Field (SCF) equations [19**?** –26, 28**?** , 29].

The existence of linear scaling methods has been argued from the concept of "nearsightedness" or quantum locality [31, 32], which implies that the density matrix $n(\mathbf{r}, \mathbf{r}')$ goes to zero as $|\mathbf{r} - \mathbf{r}'| \to \infty$. The fundamental reason for this is the loss of quantum phase coherence between points that are far apart. It is important to note that quantum locality also implies data locality, which is an essential property for scalable parallel algorithms. With parallel linear scaling methods, an n-fold increase in processors should lead approximately to an n-fold increase in simulation capability. In practice however, this is true only for scalable algorithms. While there has been continued efforts to parallelize conventional Quantum Chemistry codes [33–41], taking advantage of quantum locality to achieve parallel efficiency is a largely unexplored but important area of research in linear scaling SCF theory.

One of most computationally demanding parts of a modern Gaussian-Orbital SCF code is numerical inte-

[†]CKGan@LANL.Gov
[‡]MChalla@LANL.Gov
[§]Preprint LA-UR-02-6838.

gration of the exchange-correlation potential for DFT or HF/DFT calculations. Linear scaling approaches have been proposed to solve this problem [15–18]. Most works use Becke [43] nuclear weight functions that partition the integrand into a sum of single-center integrations, each with a nuclear origin. Standard numerical techniques may then be used to perform one-center numerical integration in spherical polar coordinates. It should be observed that this transformation overcomes the primary nuclear cusp in the electron density. However, there are additional complexities in the Becke approach. First, the radial Jacobian cannot suppress cusps at other nuclear centers. These secondary cusps can be damped to zero with a sharper partition, but this has the risk of introducing additional discontinuities in the integrand. Second, increasing density of the radial grid does not guarantee convergence to a more accurate result. Third, the overlapping nature of the Becke weight functions impedes clean spatial partitioning, which is important for achieving true linear scaling, simple domain decomposition and data parallelism.

A different approach, Hierarchical Cubature (HiCu) employs an adaptive telescoping Cartesian grid with primitive cubature rules at the finest level of resolution [18]. In HiCu, the hierarchical adaptive grid resolves strong variations and multiple length scales encountered in numerical integration of the exchange-correlation matrix $\mathbf{K}_{\rm xc}$. The k-d tree [44–46] data structure is used to represent the adaptive grid or CubeTree. The electron density of the system is also represented by the k-d tree data structure, called a RhoTree. Each node in a k-d tree contains a bounding box (BBox) that encloses the spatial data contained by it and its children (if the node is not a leaf node). Because of the hierarchical organization of data in a k-d tree, a search for all data within a given Euclidean distance (i.e., a range query) is an efficient $O(\log_2 N)$ operation.

The CubeTree is constructed through recursive bisection of a Cartesian domain (i.e. a box) enclosing the electron density, applying a fixed fully symmetric C_3 (cubature) rules [47] within each BBox. Given a local error threshold τ, the recursive bisection process stops when $\delta < \tau$, where

$$\delta = \Delta\rho + \frac{1}{3}\sqrt{(\Delta\rho_x)^2 + (\Delta\rho_y)^2 + (\Delta\rho_z)^2} \qquad (1)$$

Here $\Delta\rho$ is the magnitude of the difference between the exact electron charge in the BBox and the numerically integrated electron charge,

$$\Delta\rho = \left| \int_{\rm BBox} \rho(\mathbf{r}) d\mathbf{r} - \sum_{i=1}^{N_{\rm g}} w_i \rho(\mathbf{r}_i) \right| \qquad (2)$$

where w_i are the grid weights and $N_{\rm g}$ is the number of grid points in the cubature rule. The magnitude of the difference between the analytic and numerical integrations of the ith component (i.e., $i = x$, y, or z) of the density gradient $\nabla\rho$, denoted by $\Delta\rho_i$, is calculated in a fashion similar to $\Delta\rho$.

Unlike the first implementation of HiCu [18], the current implementation incorporates errors due to the density gradient, which leads to improved convergence properties for functionals that employ the Generalized Gradient Approximation (GGA). Following the spirit of Direct SCF methods [48, 49], the accuracy of HiCu can be systematically improved by decreasing a threshold (τ). We also note that no fitting functions are employed. The non-overlapping Cartesian approach combined with advanced data structures allows maximal exploitation of quantum locality, leading to an early onset of true linear scaling even for large basis sets and 3D systems. This non-overlapping approach to quantum locality may be leveraged to achieve both data locality and straightforward domain decompositions.

The purpose of this work is to describe our development of data parallel HiCu, with emphasis on new, efficient and generally applicable load balancing algorithms. For the first time, we propose measurement based load balancing (Equal Time partitioning) for domain decomposition in Quantum Chemistry. Equal Time partitioning exploits the smooth variation of the electron density between SCF cycles (so called temporal locality [50]), to predict an optimal domain decomposition for the next iteration.

The remainder of this paper is organized as follows. Section II discusses our strategies to efficiently parallelize the computation of $\mathbf{K}_{\rm xc}$ using HiCu. Section III discusses the issue of data locality to reduce communications between processors. Section IV describes a computational implementation of data parallel HiCu. Section V presents the results and discussions of the speedup tests performed on three systems. Section VI summarizes the main conclusions of the paper.

II. PARALLELIZATION OF HICU

In this section we describe parallelization of HiCu with the goal of obtaining good speedups even for fine-grained parallelism, which we consider to be one heavy atom per processor. This is difficult, as the spatial distribution of the grid-work, which depends on the electron density, is highly irregular for an inhomogeneous system. We note that grid points can experience very different work loads depending on their environment, as the HiCu method will always try to minimize the work associated with griding the density. To obtain a good speedup, we need to reduce both load imbalance and communications between processors. First we discuss the issue of load balancing. The issue of communication and data parallelism is discussed in Section III.

Since HiCu is purely Cartesian, spatial decomposition can be performed by simply sectioning the root BBox of the RhoTree into $N_{\rm b}$ non-overlapping subboxes. The root BBox encloses the entire electron density, and is constructed by an upward merge of sub BBoxes in the RhoTree. At the lowest level, each BBox of the RhoTree is determined by the Gaussian extent.

One approach to load balance is the master-slave approach, which has been used to calculate $\mathbf{K}_{\rm xc}$ in conventional Quantum Chemistry programs [40, 37, 39]. In the context of HiCu, a naive approach to master-slave load balancing involves making $N_{\rm b}$ larger than $N_{\rm p}$ (say, $N_{\rm b} = 4N_{\rm p}$), where $N_{\rm p}$ is the number of processors. The master program collects the results from a slave program which has finished its work on a particular subbox. The master program then instructs the slave program to work on a new subbox which has not been assigned to any of the slave programs before.

We have tested this simple dynamic master-slave load balancing scheme. Unfortunately, poor speedups are obtained, especially when both $N_{\rm b}$ and $N_{\rm p}$ are large. This is because as $N_{\rm b}$ increases, the time to deal with a subbox is generally small (since the volumes are smaller), so most slave programs will finish their work very quickly and they will spend significant time in contention for the master program, a well known problem with this approach [? ?]. In HiCu, this problem is exacerbated by the fact that the maximum load does not necessarily decrease linearly with $N_{\rm b}$ due to the irregular nature of the problem. It is interesting to note that Guerra *et al.* [34] preferred a static dynamic load balancing scheme over master-slave load-balancing because they found that repeated distribution of data requires much more communication.

An alternative is to employ a heuristic work estimate to guide the domain decomposition. We experimented with a number of heuristics, including splitting the density into equal-charge volumes and partitioning by Center of Mass. Of these heuristic approaches, the Center of Mass (CM) method, explained in the following, was found to have the greatest, but still unsatisfactory efficiency.

Another approach to load balance is measurement. Measurement approaches have several advantages: First they are easy to program. Second, they work well with smooth variation of work load in the time or iteration space (temporal locality), an exploitable feature in the SCF procedure. Third, they have been employed with great success in tree-code methods for solving the N-body problem [50–56] – HiCu has significant homology with the current implementation of the Quantum Chemical Tree-Code (QCTC) [11, 12, 14] for solving the Quantum Coulomb problem.

The remainder of this work describes our development of a measurement based method for load balancing data parallel HiCu. Briefly, the overall scheme is as follows: First the heuristic CM partitioning is used for parallel calculation of $\mathbf{K}_{\rm xc}$ in the first SCF cycle. This provides a first measurement for repartitioning the root BBox in the next SCF cycle. As long as the starting guess is reasonable and pathological charge sloshing does not occur each repartition should yield good load balance for the next iteration. In this way, temporal locality [50] of the problem is exploited to achieve a continuously improved load balance. Equal Time repartitioning will be further explained in Subsection ??.

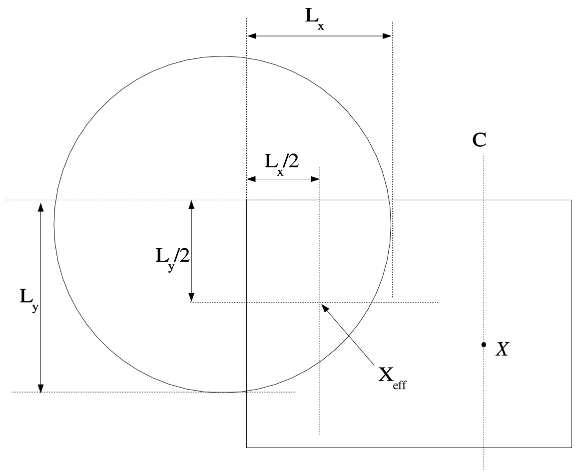

FIG. 1: A schematic diagram to explain the Center of Mass (CM) partitioning scheme. A sphere of Bragg-Slater [57] radius R_i centered on the atom i is drawn for each atom (for simplicity, only one atom is considered in this case). The volume of box-sphere intersection is approximated by $L_x L_y L_z$, where the L_x is the range of intersection between the sphere and the box in the x direction. L_y and L_z are calculated in a similar way. For simplicity we have assumed that the effective charge $Z_{{\rm eff},i}$ is centered at $\mathbf{X}_{\rm eff}$. Also shown is a cutting plane C passing through the center of mass $\mathbf{X}$.

A Center of Mass partitioning scheme

In the Center of Mass (CM) partitioning scheme, we make a plausible assumption that the computing time for a subbox is proportional to the total electron charge in the box. To partition a BBox (see Figure **??**), we first calculate the center of the electron charge, which we have loosely called the “center of mass”. We assume that the electron charge due to an atom i is smeared out evenly in a sphere of radius R_i, which we have taken to be the Bragg-Slater radius [57] for the atom. The Bragg-Slater radius for an atom is the empirical atomic radius, derived from the fact that two atoms forming a bond in a crystal or molecule gives an approximate value of the internuclear distance. It is observed that inter-atomic bond lengths in solids and molecules, whether ionic or covalent, are approximately equal to pairwise sums of unique atomic radii [57]. The center of mass (charge) $\mathbf{X}$ is given by

$$\mathbf{X} = \frac{\sum_i Z_{{\rm eff},i}\mathbf{X}_{{\rm eff},i}}{\sum_i Z_{{\rm eff},i}} \qquad (3)$$

where $Z_{{\rm eff},i} = Z_i(L_xL_yL_z/V_i)$, with $V_i = 4\pi R_i^3/3$. The symbols L_x, L_y, L_z, and $\mathbf{X}_{{\rm eff},i}$ are explained in Figure **??**.

The index i in Eq. (3) runs through all atoms which overlap (totally or partially) with the box to be partitioned. In practice, we only calculate one component of $\mathbf{X}$, which is along the largest dimension of the box, since only one cutting plane is passing through $\mathbf{X}$. Each of the two subboxes after a CM partitioning may be subjected to another CM partitioning. For simplicity, we always partition the root BBox in such a way that the number of the subboxes is a power of two, even though this restriction can be dropped easily (see Appendix A for a general Equal Time partitioning). In a parallel calculation, only one processor will perform the serial CM partition and broadcast the dimensions of subboxes to all other processors. It is important to emphasize that although the CM partitioning scheme might not give a good load balance, it does serve as a cheap and good starting partition for parallel HiCu. This is not a problem even for a large system because we can use a reasonably large local error threshold τ and a minimal basis set to start a calculation while the Equal Time partitioning scheme, to be explained in Subsection **??**, will improve the efficiency in subsequent SCF cycles. One can then switch to a better basis set or lower the threshold τ during the iterative calculations.

B Equal Time partitioning scheme

Next we discuss measurement approaches to domain decomposition, with the goal of repartitioning the root BBox based on workload information collected during CubeTree construction in the previous SCF cycle. By analogy with other parallel methods for computation of $\mathbf{K}_{\mathrm{xc}}$, we first considered distributing an equal share of grid points, naively reasoning that work load would be proportional to their number. If we evaluated the density using conventional methods, this would certainly be so. However, this is not a reliable tack because each leaf node interacts with the RhoTree in a different way, always attempting to minimize the total work load. Also, because each interaction of a leaf node with the RhoTree is so non-uniform, the simple interaction counting mechanisms employed by parallel N-Body methods [50–56] are not a viable alternative. This line of reasoning led us in the development of Equal Time (ET) partitioning, which uses the elapsed wall time associated with Cartesian sub-volumes (here the leaf nodes) to predict a good load balance in the next iteration.

In the ET partitioning of HiCu, the leaf-node time is stored along with other attributes in a variable of type CubeNode [18]. With all processors holding leaf-node times in their respective local CubeTrees, ET creates a new domain decomposition by recursively partitioning a box into two subboxes such that each subbox carries approximately the same total leaf-node time (hence "Equal Time"). At the end of the decomposition we are left with N_{p} ET BBoxes in which to construct a HiCu grid. The astute reader will by now realize this construction re-

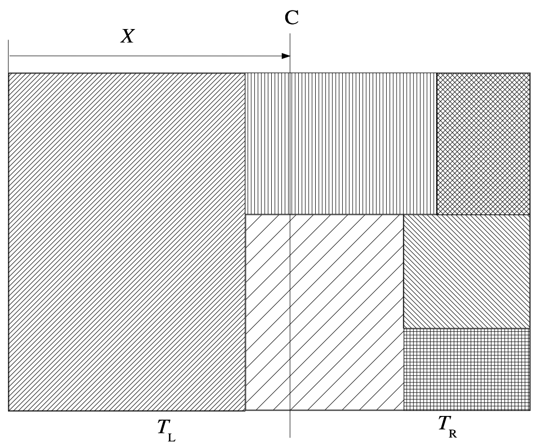

FIG. 2: A schematic diagram to illustrate the Equal Time (ET) partitioning scheme. For simplicity a two-dimensional analog is used. The rectangles represent the bounding "boxes" of the leaf nodes in a CubeTree. Different shades are used for different rectangles to indicate that the leaf-node times are different in general. The distance of the cutting line C from the left edge, x, is determined by the condition that the sum of the leaf-node times on both sides of the partition line is equal (i.e., $T_L = T_R$). Notice that we have chosen the direction for partitioning as the direction which gives the largest dimension.

quires N_{p} to be a power of two, and that errors made early in the decomposition will propagate to the lowest level. The inconvenience associated with the restriction of a power of two for the number of processors is addressed in Appendix A. We have used a robust bisection method [58] to find the plane which equally (to some threshold) divides the workload in half (see Figure 2). Notice that the cutting direction may be varied at each level of the recursion to avoid elongated ET BBoxes which could lead to poorly-balanced workloads. Our decomposition always choses a direction which cuts along the largest box dimension, since the cubature rules will work best for leaf nodes close to cubic [47].

The application of ET to HiCu centers around interaction of the CubeTree leaf nodes with the RhoTree. This implementation does not account for the time taken to build each matrix element by traversing the CubeTree. This is because of the close homology between basis function products and the electron density; they visit nearly identical portions of the CubeTree. However, if this homology did not exist, ET could associate primitive basis function products with Cartesian sub-volumes and wall times, using the combined information to load balance.

A working hypothesis of ET partitioning applied to HiCU is that the sum of leaf-node times for the root BBox is constant irrespective of sectioning. In practice this assumption is only approximately true, as the total

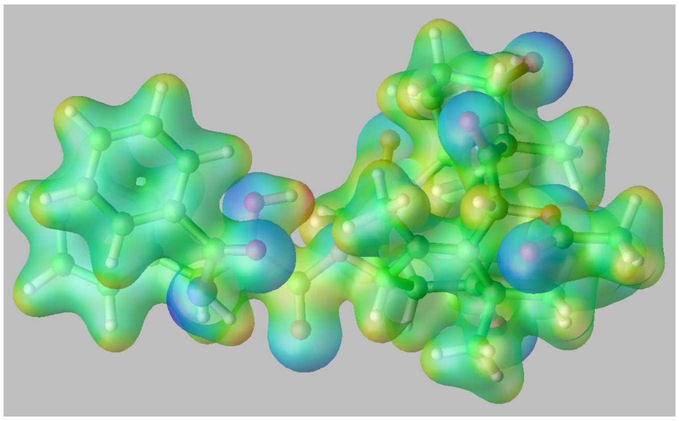

FIG. 3: The taxol electron density color-mapped with the electrostatic potential.

number of leaf nodes and their times may fluctuate if the cutting planes are shifted. This is because with each new partitioning the CubeTree will readjust, potentially using a different number of leaf nodes to satisfy $\delta < \tau$. Also, as the time to create each leaf node depends on its size and position, readjustments will lead to leaf-node times that are somewhat different. Finally, variation of the density and hence the grid will occur in all but the last few SCF cycles. All of these factors work against ET partitioning. The magnitude of these factors depends on the granularity of the partition, as will be seen in Section V.

III. DATA PARALLEL APPROACH

Data parallelism is essential for scalable algorithms, as data replication stands to exhaust local memory and throttle communications with increasing N and N_{p}. For parallel HiCu, each of the ET BBoxes requires only local information about the electron density. By building a *locally essential RhoTree*, which is *just* sufficient to create the grid in an ET BBox, redundant replication of data is avoided. Another advantage is that the smaller locally essential RhoTree is more efficient to traverse relative to the entire RhoTree that would be encountered in a replicated data scheme. This reduced complexity due to data parallelism can lead to superlinear speedups as shown in SectionV.

Data parallelism begins with a distributed atom-blocked compressed sparse row (DBCSR) [59] density matrix $\mathbf{P}^p$, which is used by program MakeRho to create an intermediate distributed Hermite-Gaussian (HG) [60, 14, 18] density ρ^p. The intermediate density consists of primitive distributions associated with only the rows spanned by $\mathbf{P}^p$. MakeRho writes ρ^p to local disk in parallel, avoiding IO and network bottlenecks. As an aside, this is the same density also used by program QCTC [11, 12, 14]. The intermediate density ρ^p is generally insufficient to build a CubeTree in an ET BBox. Communication involving all processors now occurs to construct the locally essential density ρ^p_{LE}. Ultimately, sophisticated ordering and scheduling techniques could be used to reduce complexity of this step. So far however, we have found no degradation of performance due to this redistribution. Construction of ρ^p_{LE} involves all processes going through their local primitive density distributions, performing box-box overlap tests as described in Ref. 18 to determine the exchange of data. After this redistribution of densities, construction of the locally essential RhoTree, local CubeTree, and local partially summed exchange-correlation matrix (denoted by $\mathbf{L}^p_{\mathrm{xc}}$) proceeds independently. The local, partially summed exchange-correlation matrix results from numerical integration of all primitive basis function products in the ET BBox, and is related to the full exchange-correlation matrix $\mathbf{K}_{\mathrm{xc}}$ by

$$\mathbf{K}_{\mathrm{xc}} = \sum_{p=1}^{N_{\mathrm{p}}} \mathbf{L}^p_{\mathrm{xc}} \ , \tag{4}$$

although the summation in Eq. (4) is not explicitly carried out. Rather, $\mathbf{L}^p_{\mathrm{xc}}$ is redistributed and resumed to yield a non-overlapping row-distributed $\mathbf{K}^p_{\mathrm{xc}}$.

As N_{p} becomes large, the ET BBox will become small and the corresponding local exchange-correlation matrix will become very sparse. During matrix construction there can be significant overheads associated with simple box-box testing of primitive basis function product BBox overlaps with the ET BBox. This problem is also encountered for periodic HiCu [**?**], where a double sum over lattice vectors leads to a substantial fraction of basis function products that are outside the root BBox. Our solution, described in detail elsewhere [**?**], is to perform a high level test between basis function products in an atom-atom pair and the ET BBox using a cylinder-box overlap test.

Another potential complexity encountered with fine grained parallelism involves the redistribution and resummation of $\mathbf{L}^p_{\mathrm{xc}}$. If one uses sparse-matrix data structures related to the standard CSR [61, 62] data structure such as the DBCSR or BCSR constructs [59], the resummation of very many small sparse matrices can lead to large copy overheads. We have developed the Fast Matrix [**?**] (FastMat) data structure to address this problem, in which all rows are connected by a linear linked list, while each row is represented by a binary tree. This new construct has a low overhead for random insertions and updates of atom-atom blocks. It also allows many operations such as matrix-matrix addition and matrix truncation to be done in-place, reducing both memory and copy overheads.

IV. IMPLEMENTATION

We have implemented the parallel HiCu algorithm in MondoSCF [**?**], a suite of programs for linear scaling

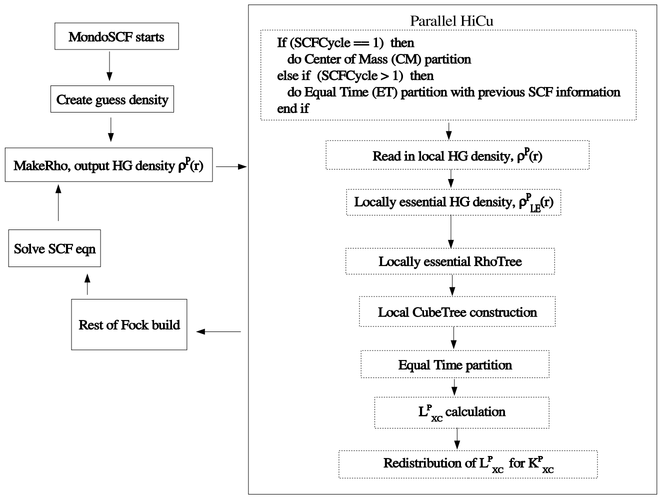

FIG. 4: The general flow of MondoSCF, with parallel HiCu subroutine expanded. Refer to Section IV for more details.

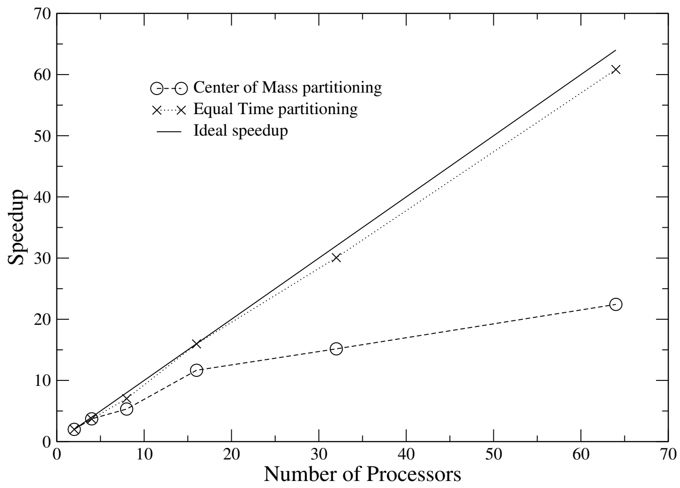

FIG. 5: Scaling of parallel HiCu on taxol ($C_{47}H_{51}NO_{14}$) RPBE/3-21G, with $\tau = 1.0 \times 10^{-6}$. Speedups are relative to a 2-processor calculation.

electronic structure theory and *ab initio* molecular dynamics. Figure 5 shows the general flow of an SCF cycle, with expanded detail of parallel HiCu. MondoSCF has been written in Fortran 90/95 with the message-passing library MPI [64]. Leaf-node times are calculated with the `MPI_WTIME` MPI function.

V. RESULTS

We have performed scaling tests on taxol, a cluster of 110 water molecules, and α-pinene with 2^n processors, where n typically ranges from 1 to 7. These systems are chosen because they are highly inhomogeneous, three-dimensional system posing a challenge to parallel HiCu. All runs on the first two systems were performed on a single SGI Origin 2000 with 128 (250 MHz MIPS R10000 MIPS) processors. For the calculations on α-pinene, we used a cluster of 256 4 CPU HP/Compaq Alphaserver ES45s with the Quadrics QsNet High Speed Interconnect.

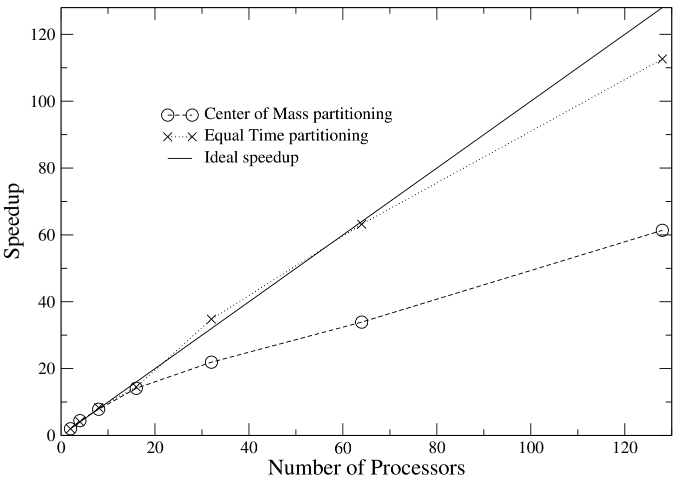

FIG. 6: Scaling of parallel HiCu on $(H_2O)_{110}$ RPBE/3-21G, with $\tau = 1.0 \times 10^{-6}$. Speedups are relative to a 2-processor calculation.

We start the calculations with the STO-3G basis set and a large threshold τ, switch to the 3-21G basis set using our mixed integral approach, and run for three SCF cycles. The density matrix $\mathbf{P}$ is saved to disk and scaling tests of parallel HiCu are performed. Timings for parallel HiCu are taken at the fourth cycle where the Center of Mass partition is used, and at the fifth cycle where the Equal Time partition is used. The procedure of switching the basis set and iterating for a few SCF cycles places the electron density in the middle of convergence. This is not really necessary, as performance is quite insensitive to SCF cycle.

The result of taxol scaling tests is shown in Figure 4, where the Center of Mass partitions perform rather poorly past 16 processors, with a speedup of only 22.4 obtained at 64 processors. However, the Equal Time partitions give good scaling results compared to the ideal speedup, obtaining a speedup of 60.8 with 64 processors. We note that our result of a speedup of 7.03 with 8 processors compares favorably with the speedup of about 6.0 of Sosa *et al.* [38], which is for an entire single-point energy calculation.

Similar scaling tests have been performed on a cluster of 110 water molecules. Figure 6 shows that the Center of Mass partition gives a better speedup than in the case of taxol. Equal Time repartitioning is highly effective. The speedup is almost ideal up to 64 processors. We observe a superlinear speedup at 32 processors, which is a medium-grained calculation. We attribute the superlinear speedup to a good load balance and to reduced complexity associated with data parallelism, where the memory access time has been greatly reduced with a smaller RhoTree. The speedup with 128 processors de-

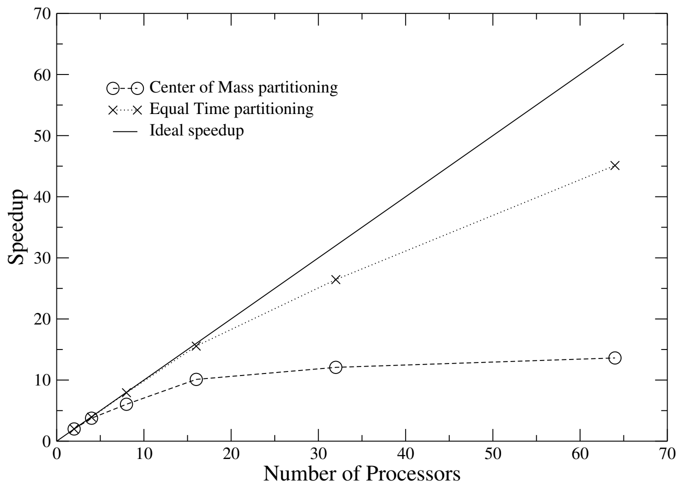

FIG. 7: Scaling of parallel HiCu on α-pinene RPBE/6-311G(df,p), with $\tau = 1.0 \times 10^{-6}$. Speedups are relative to a 2-processor calculation.

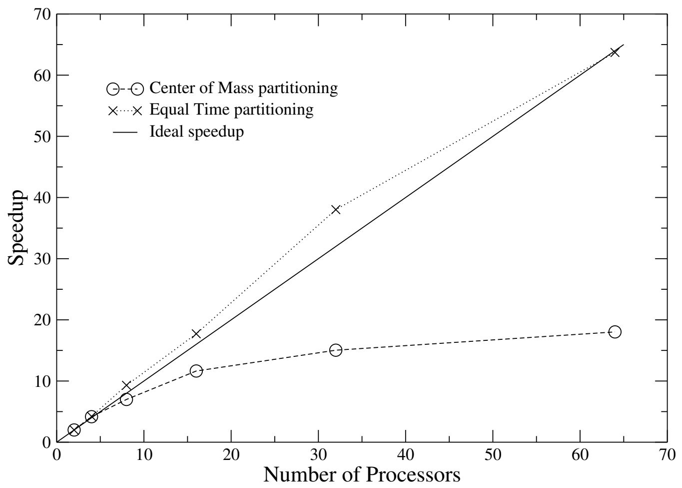

FIG. 8: Scaling of parallel HiCu on α-pinene RPBE/6-311G(df,p), with $\tau = 1.0 \times 10^{-8}$. Speedups are relative to a 2-processor calculation.

grades slightly to 112.64 (an efficiency of 88%). Still, this is an encouraging result as the 128-processor run corresponds to very fine-grained parallelism, with less than one heavy atom per processor. The reason for this degraded efficiency is not due to communication overhead but to intrinsic limitations in the ET scheme associated with fluctuations in the leaf-node count and small non-conservation of total times as explained in **??**. While there are almost certainly ways to overcome/compensate for these effects, we feel that the overall performance of ET for parallel HiCu is outstanding. To explore the behavior of the current implementation with $N_{\mathrm{p}} \sim 1024$ will require a fully parallel MondoSCF, which we are actively pursuing.

A well studied system in parallel Quantum Chemistry is 6-311G(df,p) α-pinene ($C_{10}H_{16}$) [37, 38]. To date the most competitive rate for this system is 37.3 out of 64 processors, achieved by Furlani *et al.* [37] over a complete SCF cycle. Scaling results for a medium accuracy grid corresponding to $\tau = 1.0 \times 10^{-6}$ are shown in Fig. **??**, with HiCu achieving a competitive rate of 45.11 (70% efficiency) out of 64 processors. More impressively, with a high accuracy grid ($\tau = 1.0 \times 10^{-8}$) Equal Time partitioning delivers an unrivaled 99.5% efficiency with $N_{\mathrm{p}} = 64$ as shown in Fig. **??**. These results demonstrate that the efficiency of ET improves when the grid work per processor is increased. This is not surprising as increasing the grid work decreases the effective granularity, reducing sensitivity to fluctuations in leaf-node times and leading to a more predictive Equal Time partition.

VI. CONCLUSIONS

We have developed an efficient and general load balancing technique, Equal Time (ET) partitioning, and used it in data parallel implementation of the linear scaling Hierarchical Cubature (HiCu) algorithm for numerical integration of the exchange-correlation matrix. In the ET approach, strong spatial irregularities in the work load are overcome by exploiting temporal locality between iterations. We expect ET to also exploit this effect between geometry steps in an optimization or molecular dynamics run. In this data parallel implementation, quantum locality has been used to reduce communications between processors, to lower memory usage and to reduce the computational complexity associated with traversal of the RhoTree. In some cases, this last effect can lead to superlinear speedups for medium grained calculations.

To the best of our knowledge, Equal Time partitioning is the first measurement based load balancing strategy applied to Quantum Chemistry. While ET partitioning shares some commonalities with partitioning schemes for parallel N-body tree-code solvers such as Orthogonal Recursive Bisection (ORB) [51] and Costzones [54], ET differs by (i) using exact load timing information rather than counting the number of interactions as in the ORB and Costzones methods and (ii) associating this load with rectilinear partitioning of Cartesian space rather than particle groupings (ORB) or nodes in a tree (Costzones). These are important distinctions for Quantum Chemistry. Often methods are much more complicated than the simple N-body problem. Measured loads encompass irregularities associated with different Gaussian extents, non-uniform access patterns, etc. that cannot be accounted for by counting interactions. Also, because we are often doing several complicated things at once, attaching work loads to a portion of 3D space is a convenient catch-all that can be associated with most objects and processes: basis function products and matrix construction, electron densities and grid construction, etc.

Our parallel approach to construction of $\mathbf{K}_{\mathrm{xc}}$ differs fundamentally from other all-electron approaches in that (i) we do not employ Becke weights and (ii) we employ measurement based load balancing. It is also different from many implementations in that (iii) it is data parallel. In addition to the numerical issues discussed in Section I, we believe the non-overlapping Cartesian subdivision used by HiCu enables a more straightforward approach to data locality and domain decomposition. Both the replicated data and the master-slave approach have well known shortcomings for fine grained massive parallelism, which have been avoided here.

With $N_{\rm p} \leq 128$, the predominant cause of deviation from ideal scaling is due to basic non-linearities in ET partitioning of the HiCu grid, rather than overheads associated with inefficient data structures or communication. These deviations increase as the processor to grid work ratio increases, *i.e.* with finer granularity. The main factors that determine grid work are the number of atoms and τ, the threshold controlling grid accuracy. For example, calculations on $(H_2O)_{110}$ with a medium accuracy grid and ~ 1 heavy atom per processor are fine grained, achieving a parallel efficiency of 88% with $N_{\rm p} = 128$. Even with $N_{\rm p}/N_{\rm Atoms} > 2$ it is possible achieve excellent scaling by lowering τ. This was demonstrated by 6-311G(df,p) calculations on α-pinene ($C_{10}H_{16}$) with $N_{\rm p} = 64$, achieving a parallel efficiency of 99.5%. Apparently, this is the first time this effect has been demonstrated in the context of parallel Quantum Chemistry. More importantly, because HiCu continues to achieve early onset linear scaling even with tight numerical thresholds [18], these results suggest the potential for very large, high quality calculations through a combination of massive parallelism and scalable $\mathcal{O}(N)$ algorithms.

ACKNOWLEDGMENTS

This work has been carried out under the auspices of the US Department of Energy under contract W-7405-ENG-36 and the ASCI project. The Advanced Computing Laboratory of Los Alamos National Laboratory, Los Alamos, NM 87545 is acknowledged. This work was performed on computing resources located at this facility. We thank the reviewer for valuable comments. A fruitful discussion on the general Equal Time partitioning scheme with C. J. Tymczak is appreciated.

REFERENCES

[1] S. Goedecker, Rev. Mod. Phys. **71**, 1085 (1999).
[2] S. Y. Wu and C. S. Jayanthi, Phys. Rev. Lett. **358**, 1 (2002).
[3] E. Schwegler and M. Challacombe, J. Chem. Phys. **105**, 2726 (1996).
[4] E. Schwelger, M. Challacombe, and M. Head-Gordon, J. Chem. Phys. **106**(23), 9708 (1997).
[5] E. Schwegler, M. Challacombe, and M. Head-Gordon, J. Chem. Phys. **109**(20), 8764 (1998).
[6] E. Schwegler, *Thesis: Linear Scaling Computation of the Hartree-Fock Exchange Matrix*, Ph.D. thesis, University of Minnessota (1998), at `http://www.t12.lanl.gov/home/mchalla/`.
[7] E. Schwelger and M. Challacombe, J. Chem. Phys. **111**(14), 6223 (1999).
[8] E. Schwegler and M. Challacombe, Theo. Chem. Acc. **104**(5), 344 (2000).
[9] C. A. White, B. Johnson, P. Gill, and M. Head-Gordon, Chem. Phys. Letters **230**(1-2), 8 (1994).
[10] C. A. White, B. G. Johnson, P. M. W. Gill, and M. Head-Gordon, Chem. Phys. Lett. **253**(3–4), 268 (1996).
[11] M. Challacombe, E. Schwegler, and J. Almlöf, *Computational Chemistry: Review of Current Trends* (World Scientific, Singapore, 1996), pp. 53–107.
[12] M. Challacombe, E. Schwegler, and J. Almlöf, J. Chem. Phys. **104**, 4685 (1996).
[13] M. C. Strain, G. E. Scuseria, and M. J. Frisch, Science **271**(5245), 51 (1996).
[14] M. Challacombe and E. Schwegler, J. Chem. Phys. **106**, 5526 (1997).
[15] J. M. Pérez-Jordá and W. Yang, Chem. Phys. Lett. **241**, 469 (1995).
[16] R. E. Stratmann, G. E. Scuseria, and M. J. Frisch, Chem. Phys. Let. **257**(3-4), 213 (1996).
[17] C. F. Guerra, J. G. Snijders, G. teVelde, and E. J. Baerends, Theo. Chem. Acc. **99**(6), 391 (1998).
[18] M. Challacombe, J. Chem. Phys. **113**, 10037 (2000).
[19] X. P. Li, R. W. Nunes, and D. Vanderbilt, Phys. Rev. B **47**, 10891 (1993).
[20] M. S. Daw, Phys. Rev. B **47**(16), 10895 (1993).
[21] S. Y. Qiu, C. Z. Wang, K. M. Ho, and C. T. Chan, J. Phys. Cond. Matt. **6**(43), 9153 (1994).
[22] E. Hernández and M. J. Gillan, Phys. Rev. B **51**, 10157 (1995).
[23] E. Hernández, M. J. Gillan, and C. Goringe, Phys. Rev. B **53**, 7147 (1996).
[24] C. M. Goringe, E. Hernández, M. J. Gillan, and I. J. Bush, Comp. Phys. Comm. **94**, 89 (1996).
[25] A. D. Daniels, J. M. Millam, and G. E. Scuseria, J. Chem. Phys. **107**(2), 425 (1997).
[26] D. R. Bowler and M. J. Gillan, Comp. Phys. Comm. **120**(2–3), 95 (1999).
[27] A. H. R. Palser and D. E. Manolopoulos, Phys. Rev. B **58**(19), 12704 (1999).
[28] M. Challacombe, J. Chem. Phys. **110**, 2332 (1999).
[29] A. M. N. Niklasson, Phys. Rev. B (2002), http://arXiv.org/abs/cond-mat/0209502, accepted.
[30] A. M. N. Niklasson, C. J. Tymczak, and M. Challacombe, J. Chem. Phys. (2002), submitted.
[31] W. Kohn, Int. J. Quant. Chem. **56**, 229 (1995).
[32] W. Kohn, Phys. Rev. Lett. **76**(17), 3168 (1996).
[33] R. J. Harrison and R. Shepard, Annu. Rev. Phys. Chem. **45**, 623 (1994).
[34] C. F. Guerra, O. Visser, J. G. Snijders, G. te Velde, and E. J. Baerends, in *Methods and techniques for*

Computational Chemistry, edited by E. Clementi and G. Corongiu (STEF, Cagliari, 1995), p. 305.
[35] C. P. Sosa, J. Ochterski, J. Carpenter, and M. J. Frisch, J. Comp. Chem. **19**, 1053 (1998).
[36] F. Stephan and W. Wenzel, J. Chem. Phys. **108**, 1015 (1998).
[37] T. R. Furlani, J. Kong, and P. M. W. Gill, Comp. Phys. Commun. **128**, 170 (2000).
[38] C. P. Sosa, G. Scalmani, R. Gomperts, and M. J. Frisch, Parallel Comp. **26**, 843 (2000).
[39] T. Yoshihiro, F. Sato, and H. Kashiwagi, Chem. Phys. Lett. **346**, 313 (2001).
[40] J. Baker and P. Pulay, J. Comput. Chem. **23**, 1150 (2002).
[41] H. Takashima, S. Yamada, S. Obara, K. Kitamura, S. Inabata, N. Miyakawa, K. T. K, and U. Nagashima, J. Comp. Chem. **23**(14), 1337 (2002).
[42] G. Scuseria, J. Phys. Chem. **103**, 4782 (1999).
[43] A. D. Becke, J. Chem. Phys. **88**, 2547 (1988).
[44] J. L. Bentley and J. H. Friedman, Comp. Surv. **11**, 397 (1979).
[45] J. L. Bentley, Comm. ACM **23**, 214 (1980).
[46] V. Gaede and O. Günther, ACM Comp. Surv. **30**, 170 (1998).
[47] A. H. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall series in automatic computation (Prentice-Hall, New Jersey, 1971).
[48] J. Almlöf, K. Faegri, and K. Korsell, J. Comp. Chem. **3**(3), 385 (1982).
[49] M. Häser and R. Ahlrichs, J. Comp. Chem. **10**(1), 104 (1989).
[50] J. R. Pilkington and S. B. Baden, IEEE Trans. Parallel and Distributed Systems **7**, 288 (1996).
[51] M. S. Warren and J. K. Salmon, Proc. Supercomputing '92 p. 570 (1992).
[52] A. Y. Grama, V. Kumar, and A. Sameh, Proc. Supercomputing '94 p. 439 (1994).
[53] M. S. Warren and J. K. Salmon, Comp. Phys. Commun. **87**, 266 (1995).
[54] J. P. Singh, C. Holt, J. L. Hennessy, and A. Gupta, Proc. Supercomputing '93 p. 54 (1993).
[55] J. P. Singh, C. Holt, T. Totsuka, A. Gupta, and J. Hennessy, J. Parallel Distributed Comput. **27**, 118 (1995).
[56] A. Grama, V. Kumar, and A. Sameh, Parallel Comput. **24**, 797 (1998).
[57] J. C. Slater, J. Chem. Phys. **41**, 3199 (1964).
[58] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipies in FORTRAN* (Cambridge University Press, Port Chester, New York, 1992).
[59] M. Challacombe, Comp. Phys. Comm. (2000), to appear.
[60] G. R. Ahmadi and J. Almlöf, Chem. Phys. Lett. **246**, 364 (1995).
[61] F. G. Gustavson, ACM Trans. Math. Soft. **4**(3), 250 (1978).
[62] S. Pissanetzky, *Sparse Matrix Technology* (Academic Press, London, 1984).
[63] M. Challacombe, E. Schwegler, C. Tymczak, C. K. Gan, K. Nemeth, and A. M. N. Niklasson, MondoSCF *v1.0 $\alpha 4$, A program suite for massively parallel, linear scaling SCF theory and ab initio molecular dynamics*, `http://www.t12.lanl.gov/home/mchalla/` (2001).
[64] *Message Passing Interface Forum. MPI: A message-passing interface standard (version 2.0)* (1998), `http://www.mpi-forum.org`.

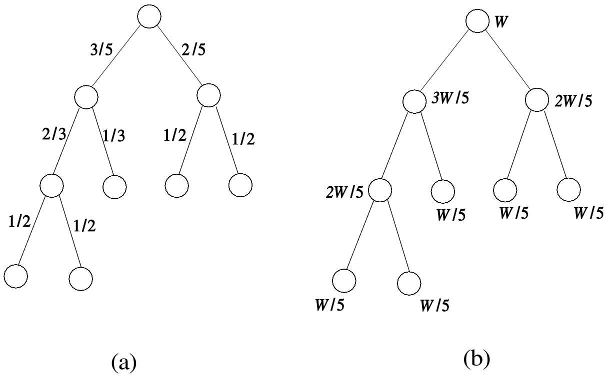

FIG. 9: An illustrative example to show the general Equal Time partitioning scheme for the case of $m = 5$.

APPENDIX A: A GENERAL EQUAL TIME PARTITIONING SCHEME

In this appendix a general Equal Time partitioning scheme is described where the number of subboxes, denoted by m, is not restricted to a power of two. To facilitate this general partitioning, first create a binary tree with m leaf nodes, with the condition that the depth between any two leaf nodes differs by no more than one. Next associate the edges connecting a parent node to its left and right child nodes with edge weights equal to $n_{\mathrm{L}}/(n_{\mathrm{L}} + n_{\mathrm{R}})$ and $n_{\mathrm{R}}/(n_{\mathrm{L}} + n_{\mathrm{R}})$, respectively. Here n_{L} (n_{R}) is the number of left (right) leaf nodes of the parent node. Figure **??**(a) shows a binary tree and all the edge weights for the case of $m = 5$. With these edge weights, it is a fairly simple matter to partition a total workload W into m equal workloads. Starting from the root node, recursively divide the workload in a parent node according to the ratio of the edge weights, i.e. $n_{\mathrm{L}} : n_{\mathrm{R}}$. At the end of this procedure, all leaf nodes will have a workload of W/m. Figure **??**(b) shows the detailed workload information in all nodes for the case of $m = 5$. In a domain decomposition of a root BBox, a bisection search may be used to partition a box into two subboxes with a workload ratio equal to $n_{\mathrm{L}} : n_{\mathrm{R}}$, which can be easily read off from the edge weights of the binary tree. It should be noticed that the "power-of-two" Equal Time partitioning scheme explained in section **??** is a special case of this general partitioning scheme.