ELSEVIER

4 August 1995

Chemical Physics Letters 241 (1995) 490–496

CHEMICAL PHYSICS LETTERS

Pseudospectral correlation methods on distributed memory parallel architectures

Todd J. Martinez, Emily A. Carter

Department of Chemistry and Biochemistry, University of California, Los Angeles, Los Angeles, CA 90095-1569, USA

Received 17 February 1995; in final form 23 May 1995

Abstract

We describe an efficient implementation of the pseudospectral multi-reference single- and double-excitation configuration interaction method on a distributed memory parallel architecture. Near-linear speedups are achieved up to 16 processors for a single-reference test case, demonstrating that pseudospectral methods are uniquely suited to parallel processing.

1. Introduction

Quantum chemical applications are widely known to consume large amounts of computer resources. Over the last decade, parallel computer architectures have been maturing and hold the promise of allowing computation of the electronic structure of large molecules [1]. Unfortunately, electronic structure theory places heavy demands on input/output as well as floating point computational facilities. Traditionally, parallel machines have been very poor at the former, primarily as a result of the inherently serial nature of random access memory and disk storage. Early attempts at parallelization of electronic structure programs were largely discouraging because of this [2]. More recent attempts have focused on clusters of networked workstations, a trend which was largely foreseen by the early work of Clementi et al. on the LCAP (loosely coupled array of processors) machines [3,4]. The advantage of such a setup is that each processor is likely to have generous amounts of core memory and local disk storage. Since the memory and disk are local to each processor, there is no problem with concurrent access. In general, this strategy entails slow communication between processors since standard networking facilities are subject to high traffic and limited bandwidth. So far, all efforts to parallelize quantum chemistry codes on such networked workstation clusters have focused on integral generation and/or solution of the Hartree–Fock equations [5,6]. Both of these can be formulated such that very little interprocessor communication is needed, and hence implementations have been quite successful. However, methods which include electron correlation require significant amounts of interprocessor communication, so one expects the networked workstation cluster will not be adequate. Fortunately, the SP architecture of IBM adds a high speed communications 'switch' to a homogeneous workstation network, yielding a paral-

SSDI 0009-2614(95)00654-0

lel machine which is well suited to the demands of electronic structure theory. Therefore, we have chosen such a machine (SP1) for the work described in this Letter.

The first distributed memory implementation of an algorithm for correlated electronic structure calculations beyond second-order perturbation theory was by Guest and co-workers [7]. However, they provided only a rough sketch of the algorithm and no timing information. More recently, Rendell, Lee and Lindh have described a distributed memory implementation of the single-reference coupled cluster method [8]. They implemented the algorithm on an Intel i860 hypercube, a machine which has only very limited (typically 8–16 Mb) core memory on each processor and no local disk storage. For their benchmark molecule, Rendell et al. found near-linear speedups up to four processors, at which point the speedup began to plateau, reaching a maximum of six at 64 processors (7% parallel efficiency). This was attributed to poor performance of input/output (I/O) on the machine. The method described in this Letter is a significant improvement, yielding a parallel efficiency of 80% for single-reference CI using 16 processors.

We are interested in multi-reference versions of correlated methods and therefore have designed and implemented a parallel version of pseudospectral multi-reference single-and double-excitation configuration interaction (MRSDCI). Note that multi-reference methods place even greater demands on input/output than single-reference methods because of the complicated interactions between the myriad of spin couplings which can be present. Thus, while the single-reference methods need only access to integrals and wavefunction coefficients, multi-reference methods additionally require access to precomputed interaction matrix elements between 'prototype' configurations [9]. Because of this, some form of local disk storage is almost a necessity. Schuler et al. have recently described a parallel implementation of conventional MRSDCI [10], and they have opted to recompute the interaction matrix elements every time they are needed. Most of the timings presented are for shared memory machines, but they also include results for a distributed memory Intel iPSC/860. Although the shared-memory performances are very pleasing, the results for the iPSC/860 were discouraging (less than 50% efficiency on 7 processors) and again this was attributed to poor I/O performance.

2. Theory

The details of the pseudospectral method which we have adapted for parallel computation have been given previously [11]. Therefore, we focus here only on the points which are important to the parallelization. Configuration interaction, be it conventionally or pseudospectrally formulated, boils down to the computation of the vector $\boldsymbol{\sigma} = \mathbf{H}\boldsymbol{c}$, where $\boldsymbol{c}$ is a vector of coefficients describing the wavefunction and $\mathbf{H}$ is the Hamiltonian matrix. Iterative diagonalization schemes such as that of Davidson [12] use this vector to obtain an improved estimate of the wavefunction coefficients.

Pseudospectral methods use both a physical space grid and function space basis representation during computation [13]. As the operators of interest are typically local (i.e. diagonal) in the physical space representation, it is conceptually straightforward to divide the work in a way that is amenable to parallel architectures. Summations which are to be carried out over grid points are simply parcelled out to various processors. To some extent, the structure required to do this is already present in a pseudospectral code because memory constraints prohibit the storage of all physical space quantities concurrently. Therefore, summations over gridpoints are written as loops over 'gridblocks' which consist of a fixed number of gridpoints according to the amount of memory available [14]. Parallelization of these outer loops is simple compared to the effort required for conventional electronic structure theory codes.

At this point our parallel implementation assumes the ability to store the vectors $\boldsymbol{c}$ and $\boldsymbol{\sigma}$ in core memory on each processor. Schemes which do not impose this requirement become very complicated, especially in the multi-reference case. We do not presently see any reason to introduce this extra complexity since the architectures targeted typically have enough core memory per processor that computations with millions of coefficients are possible. We also assume some local disk storage exists on each processor. The general strategy will be to communi-

cate the $\boldsymbol{c}$ vector to every processor, have each processor compute some part of the $\boldsymbol{\sigma}$ vector, and globally sum all the contributions on one processor. This processor then computes the new estimate for $\boldsymbol{c}$, and determines if convergence is reached. If so, it tells all other processors to commit suicide. Otherwise, the new $\boldsymbol{c}$ vector is communicated to all processors and the cycle continues. Because of the form of the communications required (global broadcast and global sum), efficient binary tree structures can be used and the cost of communication [15] becomes $2C\tau_c \log_2 P$, where C is the number of wavefunction coefficients, P is the number of processors used and τ_c is the amount of time taken to communicate a double-precision number. For constant C the communication cost therefore rises very slowly with increasing number of processors, implying good scalability can be achieved. Some distributed memory machines may not support the use of binary tree structures for global broadcast and summation. In such a case, the communication cost could be as high as $2C\tau_c P$. This would have a substantial negative effect on the scalability of our algorithm, or indeed any algorithm using data replication with global broadcast and/or summation.

We now write some representative equations to detail the parallelization strategy. Our algorithm is a hybrid of conventional and pseudospectral methods. The dominant effort in the calculation is the interaction of doubly excited configurations with other doubly excited configurations, which is treated pseudospectrally. However, the other interactions which are treated conventionally (spectrally) must also be parallelized because they take up to 30% of the computation time. If we did not parallelize this part of the computation, the maximum speedup would be only a factor of three. Thus, a complete description of the algorithm must include the conventional interactions. Nevertheless, we stress that the primary novel aspect of this Letter is the demonstration that the pseudospectral method is uniquely suited to parallel processing. Therefore, we will spend more time discussing the pseudospectral parts than the conventional ones.

The pseudospectral exchange interactions are written as

$$\sigma_{Scd} = \sum_{gTb} BK_{ij}^{ST} Q_{jg} R_{gb} A_{ic}(g) c_{Tbd}. \qquad (1)$$

Here we have introduced S and T to denote occupation and spin coupling among the reference orbitals for configurations missing two electrons. The full specification of the configuration therefore requires the indices of the two external orbitals (not occupied in any of the references) which are occupied. In this and the following, the n internal orbitals (occupied in at least one of the reference configurations) are indexed by i, j, k, l and the N external orbitals are indexed by a, b, c, d. The coupling coefficients BK are matrix elements of excitation operators [9], and for our purposes can simply be regarded as numbers of arbitrary origin. The index g labels one of the M gridpoints, and $A_{ic}(g)$ is the Coulomb potential produced by the charge distribution $\varphi_i(r)\varphi_c(r)$ at the gridpoint g. Finally, the matrix $\mathbf{R}$ effects the transformation from the function space spanned by the orbitals φ to the physical space grid and $\mathbf{Q}$ performs the inverse transformation.

In order to parallelize Eq. (1), each processor must have access to the physical space quantities $\mathbf{Q}$, $\mathbf{R}$, and $A_{ic}(g)$ for the gridpoints it will process, as well as private copies of the vectors $\boldsymbol{c}$ and $\boldsymbol{\sigma}$. As mentioned above, the vector $\boldsymbol{c}$ is communicated to all processors at the beginning of each iteration and the vector $\boldsymbol{\sigma}$ is globally summed from all processors at the end of each iteration. The coupling coefficients B_{ij}^{ST} are initially communicated to all processors and stored on local disk. Thus, each processor has its own copy of this file. In principle this is somewhat wasteful, but in practice the penalty is minor since these coefficients are only those that couple doubly excited configurations to other doubly excited configurations. An approximate upper bound to the number of coefficients in this file is $N_R n^4$, where N_R is the number of reference configurations. For most cases of interest this is entirely negligible (less than 20 megabytes), and most importantly it is independent of the number of external orbitals.

The partitioning of gridpoints among processors is determined once and for all at the beginning of the program. The existing serial implementation already assumes that the grid is divided into N_B gridblocks with a fixed maximum number of gridpoints in each gridblock. To minimize the changes to the existing programs, we simply assign gridblocks to processors in round-robin fashion. Since the number of gridblocks has been determined by the serial codes which generate and transform $\mathbf{Q}$, $\mathbf{R}$, and $A_{ic}(g)$, this strat-

egy can lead to poor load-balancing. For example, if there were 15 gridblocks, running on 16 processors would leave one completely idle. The remedy is to combine the codes which generate and transform the pseudospectral quantities with the CI code. This will be done in the future, but we first wanted to verify the good scalability of the dominant portion of the effort. With the exception of the MRSDCI test case on 16 processors, we avoid specifying timings for particularly poor combinations of number of gridblocks and processors.

The current version of the CI code assumes that the $A_{ic}(g)$ integrals have been computed and transformed. Thus, after determining the gridblock assignment, the code communicates the necessary $A_{ic}(g)$ to each processor, which then stores them on local disk. This is a significant amount of communication, but it is only performed once. Furthermore, it can be completely eliminated by either computing and transforming the $A_{ic}(g)$ directly each iteration or by integrating the transformation and CI codes as suggested above. The direct strategy is particularly attractive as the $A_{ic}(g)$ consume much of the needed disk space. The **Q** and **R** matrices are communicated in each iteration in the current implementation, but they could be treated just as the $A_{ic}(g)$. Since their number is so small, the time to communicate them is completely negligible.

The specification of the parallelization of the pseudospectral part of the algorithm is thus complete. The disk storage requirement (summed over all processors) is only slightly higher than for the serial implementation. On the other hand, the replication in core memory on each processor of the two basic vectors means that the memory requirement (again summed over all processors) grows linearly with the number of processors used. As mentioned above, we do not see this as a disadvantage in practice for most calculations of interest. The amount of communication per iteration is essentially limited to the two basic vectors.

Most importantly, there is no need for all processors to have access to the (ab/cd) and (ai/bc) integrals, since these are never used as four-index quantities. On the SP1, there is no option to use a shared filesystem, so efficient conventional treatment of these integrals would require either regenerating them as needed or communication to and local storage on each node. Since these integrals represent the major storage requirement in conventional MRSDCI, communicating or duplicating them would be prohibitive. The first option is attractive and work on this is in progress in the Lischka group [16]. As mentioned above, such a 'doubly direct' strategy would be advantageous for our method as well, since it would both decrease the disk storage requirement of the $A_{ic}(g)$ (which is already much less than that required conventionally) and increase the parallelism. The latter point relies on the observation that the formation of $A_{ic}(g)$ is most efficiently done by blocks of gridpoints even in the serial case [14]. In contrast, parallel generation of spectral two-electron integrals typically suffers from poor load-balancing particularly when using symmetry or high angular-momentum basis functions [17].

A second point to be made in comparing the parallelism of the spectral and pseudospectral MRSDCI methods concerns the computation to communication ratio. Formally, the pseudospectral approximation reduces the amount of computation [11] from $O(n^2N^4)$ to $O(n^2MN^2)$ and the amount of communication is reduced from $O(N^4)$ integrals to the $O(n^2N^2)$ elements of the CI vector. Thus, the ratio of computation to communication goes from $O(n^2)$ for conventional MRSDCI to $O(M)$ using the pseudospectral approximation. Although this suggests decreased inherent parallelism in the pseudospectral method, one should be aware of the large prefactors involved in comparing these two ratios. The number of gridpoints, M, is generally $20N$, and for moderate basis sets, N is typically $5n$. Thus, the pseudospectral method remains more suited to parallelization up to $n = 100$, which covers essentially all MRSDCI calculations for the foreseeable future. On the other hand, the use of sparser grids will adversely affect the parallelism. This is expected since the grid is the primary parallelization dimension.

Our parallel treatment of the conventional parts of the algorithm is similar to that of Guest et al. and is essentially based on partitioning the integrals among processors according to basis function index pairs. That is to say, all integrals (ij/kl) with ij in some range are treated by a given processor. Here we are using the common 'chemist's notation' for the electron repulsion integrals [18], where the indices i and j refer to the same electron. The conventional treat-

```
c   ProcID() should return the processor number starting from 0
    iproc=ProcID()
    if (iproc .eq. 0) then
        Global communication of c vector to all nodes
    endif
c   ngblocks is the number of gridblocks (batches of gridpoints)
    do i=1,ngblocks
        if (mod(iproc,i) .eq. 0) then
            Compute contribution to σ vector from gridpoints in this gridblock
        endif
    enddo
c   nfb is the number of unique pairs of internal orbitals
    do ij=1,nfb
        if(mod(iproc,ij) .eq. 0) then
            Compute contribution to σ vector from (ij|ka), (ij|kl), (ij|ab) and (ia|jb)
            for all k,l,a,b
        endif
    enddo
    if (iproc .eq. 0) then
        Global sum of σ vectors from each node
    endif
```

Fig. 1. Pseudocode for parallel algorithm to form σ vectors.

ment divides the integrals into classes according to the number of indices which are internal. The integrals with three or more indices in the external space were treated pseudospectrally, so they need not be considered here. The integrals with one external index appear in summations typified as

$$\sigma_{sa} = \sum_{ijkb} B_{ijk}^{Ss}(ij|kb)c_{Sba}.$$

Similar to the index S defined above, the index s describes the internal orbital occupation and spin coupling for a configuration missing one electron in the internal space. This is parallelized by communicating all integrals and coupling coefficients for a range of ij to the appropriate processor which then stores them on local disk. This communication is done at the beginning of the program and is negligible compared to the communication of the CI vector which is done every iteration. Given all the requisite integrals and coupling coefficients on disk, execution proceeds as in the serial case, except only terms in the summation with ij in the range assigned to a given processor are computed. Exactly the same strategy is used for the (ij/kl), (ij/ab) and (ia/jb) integrals, which appear in the interactions which were not treated pseudospectrally. In Fig. 1, we present a pseudocode implementation of the algorithm which shows the basic structure of the parallelization.

3. Results and discussion

Given the specification of the algorithm, it is of interest to examine its scalability. We have benchmarked the implementation for two test cases. The first is a single-reference SDCI calculation for ethylene oxide using a 6-311G++** basis set [19] [1], and the second is a multi-reference SDCI calculation for ethylene with nine reference configurations with a 6-31G** basis set [20] [2]. The grids used are the 'fine' grids of PSGVB v1.00 [21] which give microhartree agreement with conventional (fully spectral) MRSDCI results. The number of gridblocks for the two cases is 31 and 49, respectively, and a maximum of 32 gridpoints in each gridblock. The benchmarks

[1] The 6-311G++** basis sets are of valence triple-zeta quality with both polarization and diffuse functions on all atoms.

[2] The 6-31G** basis sets are of valence double-zeta quality with polarization functions on all atoms.

Table 1
Performance (s) of the parallel pseudospectral configuration interaction implementation

Molecule	No. SEFs [a]	Wall clock time versus No. of processors used [b]						
		1	2	5	7	8	10	16
ethylene oxide [c]	521731	658	329(2.0)	n/c [d]	n/c	99 (6.6)	n/c	52(12.7)
ethylene [e]	803633	1566	816(1.9)	391(4.9)	280(5.6)	n/c	217(7.21)	151(10.4)

[a] Number of spin eigenfunctions in wavefunction expansion.
[b] Wall clock time in s for one iteration. Numbers in parentheses are speedups relative to single-processor time.
[c] Single-reference SDCI with 6-311G** + + basis set.
[d] n/c = not computed. See text for details.
[e] RCI(2/4)*SDCI with 6-31G** basis set, where RCI(2/4) denotes the reference space of nine configurations.

were run on an IBM SP1 at UCLA with 24 nodes each having 128 Mb memory and 512 Mb of disk space. The IBM MPL message passing library [15] has been used for interprocessor communication in conjunction with the Chameleon interface [22] which makes parallel programming as machine independent as possible for supported architectures. Timings are reported in Table 1 and were obtained in dedicated mode, i.e. the benchmark job was the only one running on the assigned processors. As is usual for parallel programs, the times reported are 'wall clock' times, that is they include the time required for disk access, etc. It is evident that the implementation performs at between 65% and 100% of the theoretical maximum efficiency, with efficiency of course degrading as the number of processors increases. However, it is important to note that the degradation is not severe. This is clearly shown in Fig. 2, where we plot the speedup (time per iteration using only one processor divided by the time using n processors) for the two benchmark cases as well as the theoretical maximum.

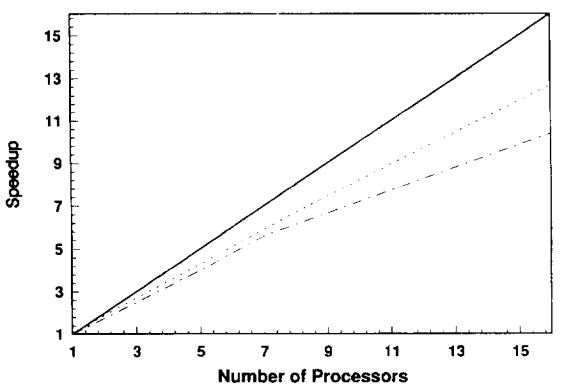

Fig. 2. Graphical depiction of scalability of the parallel algorithm described in the text. Full line is theoretical best possible performance. Dashed line denotes speedup obtained for the single-reference test case (ethylene oxide) and dot-dashed line represents the multi-reference test case (ethylene).

The multi-reference case is more adversely affected as the number of processors increases and there are two causes for this. First, when using 16 processors, the maximum speedup in the pseudospectral part is only 12.25 (instead of 16) because of the problem mentioned earlier with using a predetermined size for gridblocks. Secondly, the load-balancing issue is more severe in the treatment of the spectral integrals. Those orbitals which are active (of variable occupancy in the reference configurations) are involved in more interactions than orbitals which remain doubly occupied in all references. Since we divide the work among processors for spectral integrals evenly according to the basis function pair index ij, the load is not evenly balanced. This is easily remedied by counting the number of coupling coefficients for a given ij pair and weighting the number of pairs assigned to a given processor accordingly.

Even though these timings can be considerably improved, we feel that they already show the utility of parallel processing for correlated electronic structure calculations. The success of the implementation stems from two factors – the IBM SP1 architecture, which combines workstation class processors with high-speed communication, and the unique suitability of pseudospectral methods for distributed processing. Since the individual processors of the SP1 are respectably performing workstations, the speedups shown are a true measure of the advantage

of parallel processing. In our experience, the individual processors perform at about half the speed of a Cray YMP. Therefore, on the basis of real time required, it is already preferable to use the parallel algorithm with more than two processors over a Cray YMP.

4. Conclusions

In summary, we have presented a distributed algorithm for multi-reference configuration interaction which takes advantage of the strengths of the pseudospectral method and the IBM SP1 architecture. High parallel efficiencies are maintained up to 16 processors. We use a very modest amount of redundant disk storage, but this is completely independent of the quality of the basis set and in most cases of practical interest will be completely negligible. We have pointed out a few improvements which could increase efficiency further. Since the pseudospectral multi-reference configuration interaction method is already faster than conventional approaches on serial machines [11] and further benefits from parallelization, we feel that this method has a bright future.

Acknowledgements

We are grateful to the Office of Naval Research for primary support of this work. EAC thanks the National Science Foundation, the Camille and Henry Dreyfus Foundation, and the Alfred P. Sloan Foundation for Presidential Young Investigator, Teacher–Scholar, and Research Fellow Awards, respectively. TJM thanks the National Science Foundation for a research assistantship. We are grateful to R.A. Friesner for giving us access to a pseudospectral Hartree–Fock program.

References

[1] R.J. Harrison and R. Shepard, Ann. Rev. Phys. Chem. 45 (1994) 623.
[2] M.E. Colvin, R.A. Whiteside and H.F. Schaefer III, in: Methods in computational chemistry, Vol. 3, ed. S. Wilson (Plenum Press, New York, 1987).
[3] E. Clementi, G. Corongiu, J. Detrich, S. Chin and L. Domingo, Intern. J. Quantum Chem. Symp. 18 (1984) 601.
[4] M. Dupuis and J.D. Watts, Theoret. Chim. Acta 71 (1987) 91.
[5] S. Brode, H. Horn, M. Ehrig, D. Moldrup, J.E. Rice and R. Ahlrichs, J. Comput. Chem. 14 (1993) 1142.
[6] M.W. Feyereisen, R.A. Kendall, J. Nichols, D. Dame and J.T. Golab, J. Comput. Chem. 14 (1993) 818.
[7] M.F. Guest, R.J. Harrison, J.H. van Lenthe and L.C.H. van Corler, Theoret. Chim. Acta 71 (1987) 91.
[8] A.P. Rendell, T.J. Lee and R. Lindh, Chem. Phys. Letters 194 (1992) 84.
[9] P.E.M. Siegbahn, J. Chem. Phys. 72 (1980) 1647.
[10] M. Schuler, T. Kovar, H. Lischka, R. Shepard and R.J. Harrison, Theoret. Chim. Acta 84 (1993) 489.
[11] T.J. Martinez and E.A. Carter, J. Chem. Phys., in press.
[12] E.R. Davidson, J. Comput. Phys. 17 (1975) 87.
[13] S.A. Orszag, Stud. Appl. Math. 51 (1972) 253.
[14] M.N. Ringnalda, M. Belhadj and R.A. Friesner, J. Chem. Phys. 93 (1990) 3397.
[15] IBM AIX Parallel Environment Parallel Programming Subroutine Reference Release 2.0 SH26-7228.
[16] T. Kovar and H. Lischka, private communication.
[17] R.J. Harrison and R.A. Kendall, Theoret. Chim. Acta 79 (1991) 337.
[18] A. Szabo and N.S. Ostlund, Modern quantum chemistry (McGraw-Hill, New York, 1989).
[19] T. Clark, J. Chandrasekhar, G.W. Spitznagel and P. von R. Schleyer, J. Comput. Chem. 4 (1983) 294.
[20] W.J. Hehre, R. Ditchfield and J.A. Pople, J. Chem. Phys. 56 (1972) 2257.
[21] M.N. Ringnalda, J.M. Langlois, B.H. Greeley, T.V. Russo, R.P. Muller, B. Marten, Y. Won, R.E. Donnelly Jr., W.T. Pollard, G.H. Miller, W.A. Goddard III and R.A. Friesner, PS-GVB v1.00, Schroedinger, Inc. (1993).
[22] W. Gropp and B. Smith, Users Manual for the Chameleon Parallel Programming Tools. Argonne National Laboratory Technical Document ANL-93/23.