

## Partially Direct SCF 法の開発と性能評価

稲富 雄一<sup>a,b,\*</sup>, 佐々木 徹<sup>c</sup>, 長嶋 雲兵<sup>a,b</sup>, 村上 和彰<sup>d</sup>

<sup>a</sup> 産業技術総合研究所グリッド研究センター, 〒 305-8568 つくば市梅園 1-1-1 中央第二

<sup>b</sup> 科学技術振興機構構造的創造研究推進事業, 〒 332-0012 埼玉県川口市本町 4-1-8 川口センタービル

<sup>c</sup> (株) アプリオリマイクロシステム, 〒 212-0054 川崎市幸区小倉 308-10 かわさき新産業創造センター 236

<sup>d</sup> 九州大学大学院システム情報科学研究院, 〒 812-8581 福岡市東区箱崎 6-10-1

\*e-mail: y.inadomi@aist.go.jp

(Received: July 4, 2005; Accepted for publication: August 12, 2005; Published on Web: November 18, 2005)

膨大な 2 電子積分の一部をメモリに保存して、2 回目の SCF サイクル以降再利用する計算手法 (partially direct SCF, PDSCF 法) を提案し、PC クラスタを用いて、その性能評価を行った。非常に簡単な改良であるにもかかわらず、PDSCF 法を用いた Fock 行列作成時の並列処理の性能を大きく向上させることが分かった。また、積分値 1 つ当たりの計算コストを考慮すると、軌道量子数の和が小さな (ss,ss) タイプの積分から保存したほうが有利であることも分かった。この計算手法を用いることで、我々が開発してきた 2 電子積分専用 LSI を多数搭載し、ディスクレス構造を採る Fock 行列作成のための超並列専用計算機でも、高い効率を保ったまま並列計算ができる。

キーワード: 分子軌道法, 2 電子積分, 専用計算機, 並列化, Partially Direct SCF 法

### 1 はじめに

Hartree-Fock (HF) 分子軌道 (MO) 法や密度汎関数 (DFT) 法の計算では、2 電子積分の計算、処理に多くの時間を費やす。そのため、MO 計算を高速に行うためには、2 電子積分の計算を高速に行う必要がある。高速な 2 電子積分計算には、高速な計算アルゴリズムの開発やソフトウェアによる並列処理など、いろいろな方法が考えられる。Embedded High Performance Computing (EHPC) プロジェクトでは、2 電子積分計算を行うための専用ロジック (ERIC チップ) を用いた超並列処理により、積分計算を高速に、かつ低消費電力で行う方法を検討してきた [1]。ERIC チップには、適用範囲が 2 電子積分計算のみと狭い代わりに、積分計算に対する処理能力が高く、かつ、消費電力も汎用プロセッサに比べて低く抑えられる、という特徴がある。したがって、数 100~ 数 1,000 の ERIC チップを用いることで、一般的な研究室の電源設備で駆動できる高速な分子軌道計算専用マシンを、小さな設置スペー

スに構築することが可能になる。

Figure 1 に ERIC チップを使用した 2 電子積分 (Fock 行列作成) 専用計算機システムの構成を示す。本プロジェクトで開発している Fock 行列作成専用計算機は、数個の ERIC チップを搭載した compact PCI (CPCI) 規格準拠のボードを CPCI ボード用のスロットを持つ汎用計算機に装着する構造を採っている。これは、ERIC チップの部分のみを他の専用ロジックと交換することにより、他の専用計算機へ変更することを容易にするためである。ここで、1 つの CPCI ボードに注目すると、ボード上には磁気ディスクなどの 2 次記憶装置がない (ディスクレス構造) が、ERIC チップが各自で専用アクセスできるメモリを持つ。このメモリの大きさは 256MB~2GB くらいである。ERIC チップが対象としている系の大きさが、計算精度の点から約 5,000 軌道まで、ということを考えてみると、Fock 行列や密度行列の行列要素などの必要データを格納しても、メモリに余剰領域が生じる。このメモリを有効に利用するために膨大な 2 電子積分の一部を保存して、MO 計算

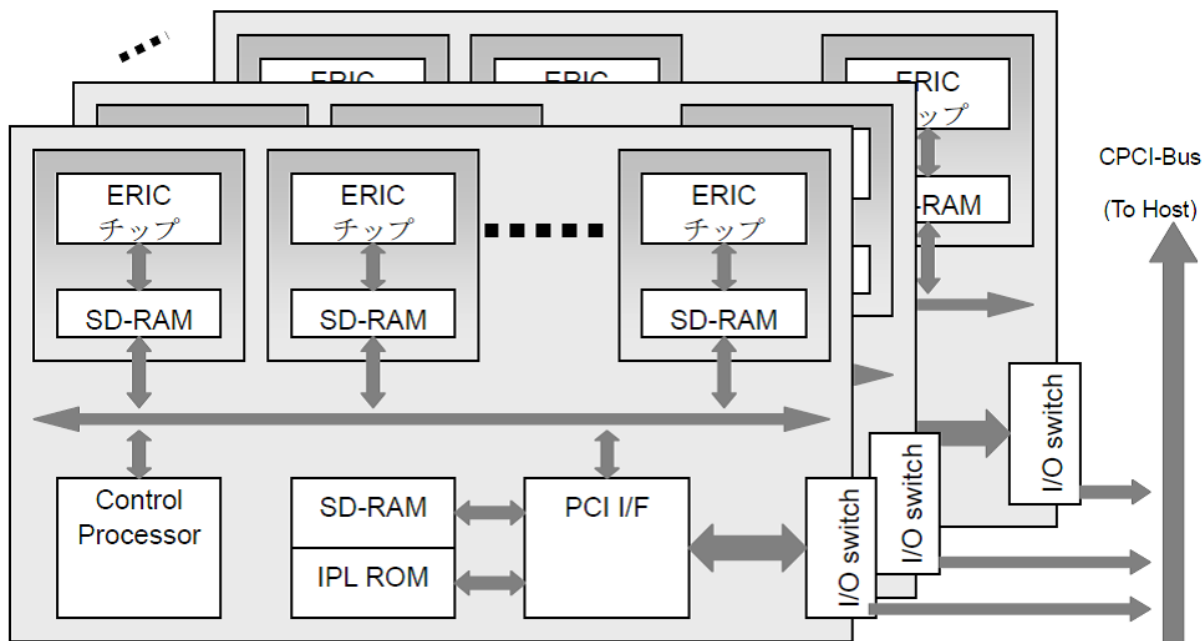


Figure 1. System configuration of the special purpose computer for the fast Fock matrix generation with special purpose processors for electron repulsion integral (ERI) evaluation

で行う繰り返し ( SCF ) 計算の際に再利用することを考える。そうすれば、2 回目以降の SCF サイクルでの 2 電子積分の計算量が減少して、見かけ上の積分計算が高速になるはずである。そこでこのような二電子積分の一部を再利用する SCF 計算法 ( 以降、partially direct SCF (PDSCF) 法、と呼ぶ ) のプログラムを作成して、開発中の分子軌道専用計算機システムを用いた超並列処理が、高い効率を保ったまま行えるかどうかの評価を行った。

## 2 計算方法

2 電子積分計算には、小原らが提案した Vertical Recurrence Relation ( VRR ) [2] と Head-Gordon らが提案した Horizontal Recurrence Relation ( HRR ) [3] を組み合わせた hybrid アルゴリズムを用いて筆者らが独自に開発したプログラム ( C 言語版、および F77 版 ) を使用した。

分子積分には積分に関わる軌道の軌道量子数により (ss,ss) や (ps,ss) などといった様々な積分タイプが存在し、その演算量も異なる。PDSCF 法では、どのタイプの積分からメモリに保存するかが性能に影響する。そこで、各タイプの 2 電子積分に必要な計算時間、

演算数などのプロセッサ資源を調べるために、Performance Counter Library ( PCL ) を用いて、積分タイプごとの性能を評価した。積分計算の並列化は Message Passing Interface ( MPI ) [5] を用いて行った。使用するプロセッサ数や積分保存領域 ( バッファ ) の大きさを変化させて、PDSCF 法の性能評価を行った。計算には産業技術総合研究所 ( AIST ) グリッド研究センターにある Gfarm Cluster を使用した。コンパイラには、インテルコンパイラ ( Linux 版、version8.0 ) [6, 7] を用いた。

## 3 結果

### 3.1 各積分タイプに必要な計算量の比較

Figure 2 は、F77 版の積分プログラムと PCL を用いて、各タイプの積分 1 個あたりの計算コストを示したもので、(dd,dd) タイプの結果を 1 とした場合の比が描かれている。この計算は Pentium4 ( 2.0GHz ) を搭載したデスクトップパソコンで計測したものである。対象分子として Thymine、基底関数として 6-31G\* ( 147 軌道 ) を用いた。この図で、“Cycles”は計算に要するプロセッサのクロックサイクル数の比を表しており、

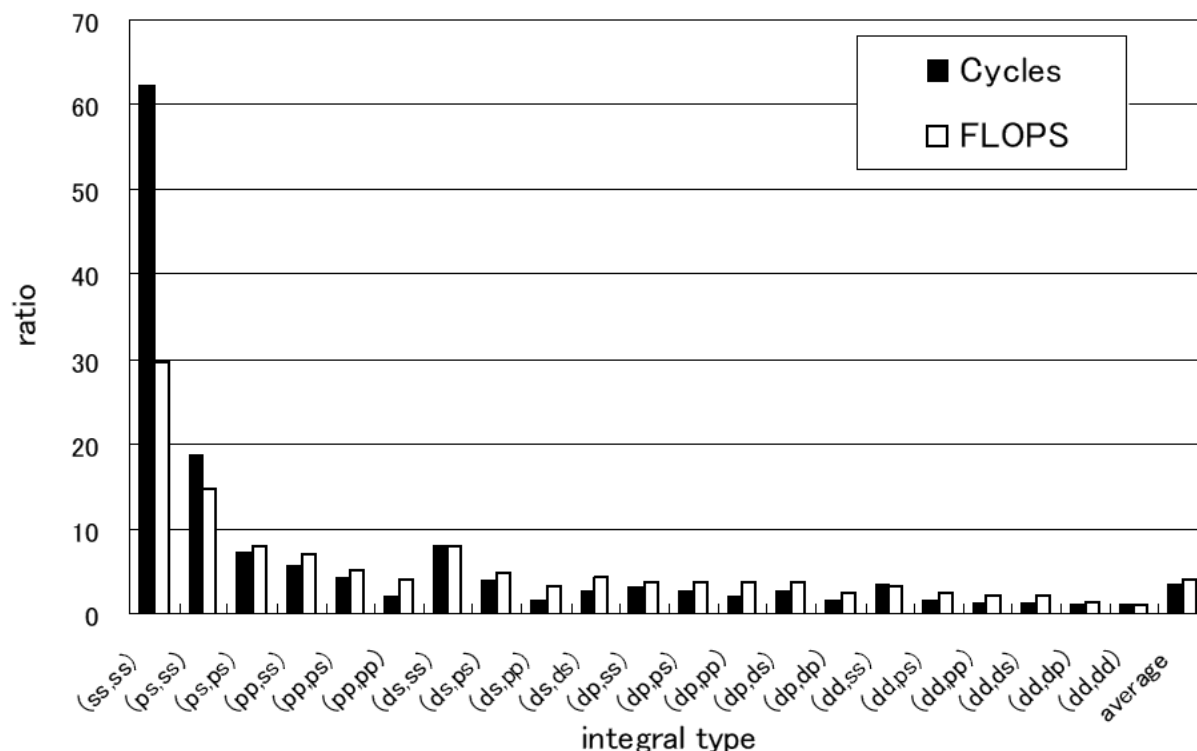


Figure 2. Ratios of cost to evaluate a two-electron integral value for every integral type to (dd,dd) results for Thymine molecule using 6-31G\* basis set (Cycles: the number of clock cycles, FLOPS: the number of floating point operations)

これは単純に積分計算に必要な計算時間の比と考えてよいものである。また、“FLOPS”の値は、積分計算に必要な四則演算や開平計算などの浮動小数演算数の比を示している。ただし、各演算の演算コストは考慮せずに、演算数を単純に比較したものである。まず、“Cycles”の値を見ると、(ss,ss)や(ps,ss)といった、2電子積分1個を構成する4つの基底関数の軌道量子数の和が小さな積分の方が、(dd,dd)などの軌道量子数の和が大きなものに比べて大きな値をもつことが分かる。2電子積分の値1つを計算するコストは(dd,dd)タイプに比べて、たとえば、(ss,ss)タイプで約62倍、(ps,ss)タイプで約19倍大きい。したがって、PDSCF法においては、積分1つ当たりの計算コストの大きな(ss,ss)タイプの積分の値を優先してバッファに保存したほうが性能向上の点で有利である。次に、積分1つあたりの演算数を示している“FLOPS”の値を見ると、先ほどの“Cycles”と同様の傾向を示している。ただし“Cycles”の場合ほどの大きな開きがない。(dd,dd)の結果と比べた場合、たとえば、(ss,ss)タイプでは、演算

回数は30倍弱であるのに対して、実際の計算時間は前述の通り60倍を越えており、演算数と計算時間の比に約2倍の開きがある。これは、すべての積分タイプで共通に必要な初期積分計算には、汎用プロセッサが加減乗算に比べて不得意としている除算や開平計算が多く含まれていること、さらに、(ss,ss)や(ps,ss)などの計算では、(pp,pp)、(dd,dd)などの積分に比べて、1回に計算できる積分数が少ないため、積分値1つの計算における初期積分計算の割合が大きいこと、などが原因である。

### 3.2 保存順序による性能の違い

積分タイプによる計算コストの差から、PDSCF法においては、軌道量子数の和が小さな積分(ss,ss)からバッファに保存して再利用したほうが高い性能が得られることがわかった。そこで、実際にFock行列作成を行って、積分を保存する順序によるPDSCF法の性能の違いを調べた。対象分子として(Gly)<sub>15</sub>(原子数

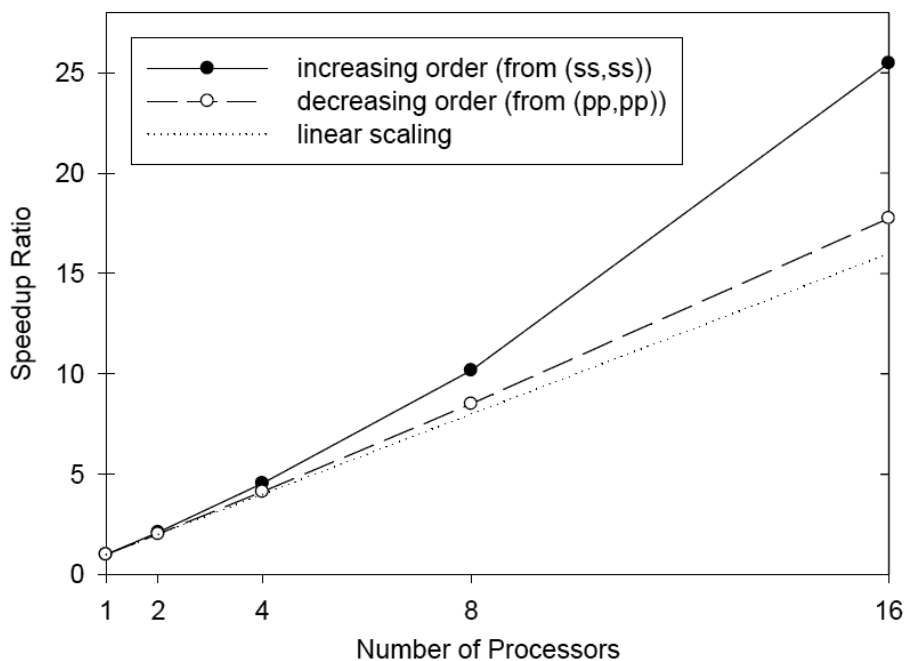


Figure 3. Effect of the storage order of ERI in the PDSCF (Speedup ratios in Fock matrix generation at Second SCF cycle) (buffer size = 64MB/processor, Target molecule= (Gly)<sub>15</sub>, basis set = STO-3G (108 atoms, 352 basis functions))

108) を、基底関数として STO-3G (軌道数 352) をそれぞれ用いた。使用した計算機は Gfarm cluster (Dual Xeon (2.8GHz) × 90 nodes) で、そのうち 1~16 プロセッサを使って性能評価を行った。プログラムは C 言語版の積分プログラムを用い、軌道量子数の和の小さな (ss,ss) タイプの積分から保存するもの (increasing order) と、軌道量子数の和が大きな (pp,pp) タイプの積分から保存するもの (decreasing order)、という 2 つの保存方法で PDSCF 法を実装して、そのテストを行った。

Figure 3 に、PDSCF 法の 2 回目の SCF サイクルでの Fock 行列作成時間における台数効果を示す (バッファサイズはプロセッサあたり 64MB) PDSCF 法は、積分の保存順序に関わらず、理想的な線形速度向上率 (図中で “linear scaling” と描かれている直線が表している) よりも高い並列化性能 (スーパーリニアスピードアップ) を持っていることが分かる。これは、台数が増えるに従って、実質的にバッファメモリ量が増大するためである。

また、優先して保存する積分のタイプが、性能に大きく影響していることがわかる。たとえば、16 プロセッサ使用時に、(pp,pp) から保存した場合には 17.7 倍の速度向上比であるのに対して、(ss,ss) から保存し

た場合では 25.5 倍と、大きく性能に差がある。したがって、積分タイプにおける計算コストの違いから予測されたとおり、PDSCF 法では、軌道量子数の和の小さい積分から保存、再利用するほうが、よい性能を得られることが明らかとなった。

### 3.3 バッファサイズによる性能の違い

PDSCF 法のやや大規模な計算における性能を調べるために、対象分子として Crambin (46 アミノ酸残基、642 原子) に対する Fock 行列作成における並列化の効率を測定した。基底関数としては STO-3G を用いた (軌道数 1,974) また、プロセッサあたりのバッファのサイズを従来の direct SCF 法 [8] と等価な 0MB (0MB/proc) から 128MB (128MB/proc) まで変化させて、バッファサイズが PDSCF 法に及ぼす影響を調べた。使用した計算機は Gfarm Cluster で、そのうち 1~64 ノード (1~128 プロセッサ) を用いて計算を行った。

Figure 4 に、この計算の 2 回目の SCF サイクルにおける Fock 行列作成における並列化効率を示す。ここで、並列化効率は、次式を用いて計算している。

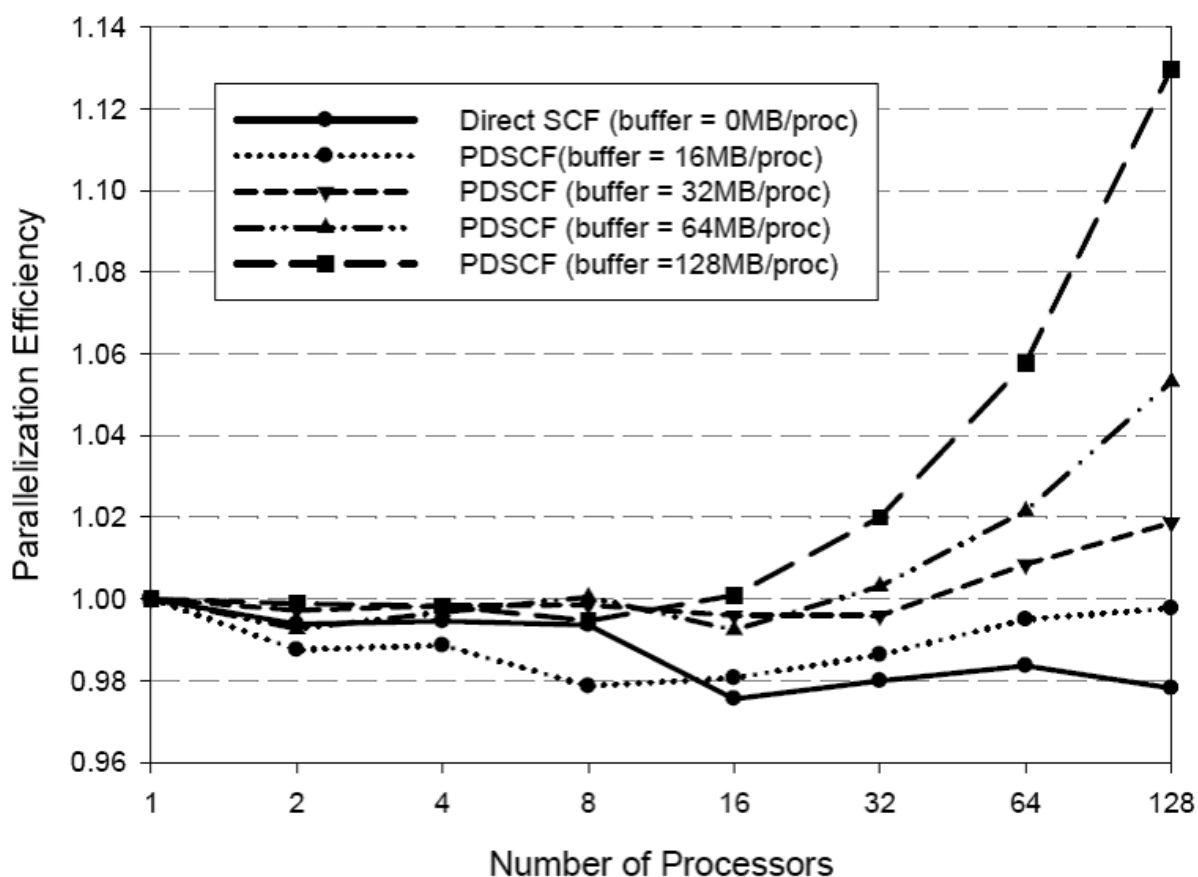


Figure 4. Parallelization efficiencies of the PDSCF and the conventional direct SCF methods for large systems at the second SCF cycle. (Target = Crambin, basis set = STO-3G (642 atoms, 1,974 basis functions))

$$\text{並列化効率} = \frac{\text{実際の速度向上率}}{\text{理想的な速度向上率 (= 用いたプロセッサ数)}}$$

この定義から分かるように、この値が1未満であれば、実際の並列性能が理想値を下回っていることを意味し、逆に、1を越えていれば、理想的な並列性能を上回る処理速度が得られていることになる。

まず、バッファサイズが 0MB/proc の場合、つまり、通常の direct SCF 法の結果を見ると、2 電子積分計算が本来持っている高い並列性のため、ほぼ理想的な並列化効率を示していることがわかる。たとえば、64 プロセッサ使用時に 0.980、128 プロセッサ使用時には 0.978 と、ほぼ 1 に近い並列化効率が得られていることがわかる。しかし、プロセッサ数が増えるにしたがって、並列化効率が徐々に低下している。これは、通信などの並列処理を行うためのオーバーヘッドや、並列化できない部分の存在がプロセッサ数の増大により大きな割合を示すようになること、によるもので、並列処

理を行う場合に生じる一般的な現象である。数 100 以上のプロセッサを用いた並列処理や、ERIC チップを用いた超並列処理を行う場合に、思ったような速度向上が得られないことが明らかである。

次にバッファサイズが 0MB/proc ではない PDSCF 法の結果についての考察を行う。Figure 4 からまず言えるのは、プロセッサ当たりのバッファサイズが大きいほど、見かけの並列化効率が向上していることである。これは、バッファサイズが大きくなれば 2 回目の SCF サイクル以降、再計算の必要がない積分量が増えることによるものである。具体的な数値で見ると、128 プロセッサ使用時に見た場合、32MB/proc、64MB/proc、128MB/proc と増えると、並列化効率はそれぞれ、1.020、1.058、1.130 とよくなっている。

さらに、2~16 プロセッサを用いた小規模な並列処理を行った場合には低下する傾向にあった並列化効率が、プロセッサ数が 32、64、128 と増えるにつれて、上昇していることが分かる。たとえば、バッファサイズ

が 64MB/proc の場合を見ると、16 並列時に 0.992 まで低下した並列化効率が、32 並列時には 1.003、64 並列時には 1.022 と改善していき、128 プロセッサ使用時には 1.058 にまで向上している。この傾向は、バッファサイズが小さな場合にも見られ、16MB/proc の場合には、8 並列時に 0.979 まで低下した並列化効率が、その後上昇に転じ、128 並列時には 0.998 まで回復している。

Fock 行列作成の並列化では、各プロセッサで Fock 行列の部分和を生成したあと、最後に、それらの総和を求めるために通信を行う必要があり、この処理の存在が、従来の direct SCF 法、PDSCF 法の両方において、並列化効率低下を引き起こす主な原因である。計算に使用するプロセッサ数を  $N$  とした場合、この通信にかかるコストは  $\log_2 N$  に比例して増加する。したがって、従来の direct SCF 法では、並列 Fock 行列生成全体での並列化効率は、プロセッサ数が増加するに従って、徐々に低下していく。一方で、PDSCF 法で 2 回目の SCF サイクル以降に計算しなくて済む 2 電子積分数は、用いるプロセッサ数に比例して増加するため、2 電子積分の計算コストは、プロセッサ数に比例して減少することになる。PDSCF 法における、通信コストと積分計算コストを比較すると、並列処理に用いるプロセッサ数を増やした場合、通信コストの増加よりも、2 電子積分計算コストの減少の程度が大きく、全体として、見かけの並列化効率がよくなることが予想される。今回の Crambin に対する PDSCF 法の計算結果は、そのことを実証した形になっている。

ここで、並列化効率が 1 を上回るために必要となるバッファの総量について考える。Figure 4 で並列化効率 1 付近にある点を見ると、バッファ総量が 2GB を越えたあたりで、並列化効率 1 にかなり近い値をとる、あるいは 1 を超えることが分かる。それは、( バッファサイズ, プロセッサ数 ) の組で、( 128MB/proc, 16 ) ( 64MB/proc, 32 ) ( 32MB/proc, 64 ) あるいは ( 16MB/proc, 128 ) の場合に相当し、その並列化効率は、それぞれ、1.001、1.003、1.008、および 0.998 を示している。このバッファには、今回の系で有意な値 ( 絶対値で  $10^{-15}$  以上 ) を持つ 2 電子積分のわずか 0.43% 程度しか保存されていない。しかしながら、このように非常に少数の 2 電子積分を保存するだけで、PDSCF 法が super linear な性能向上を示すことがわかる。先に述べたとおり、ERIC チップを用いた 2 電子積分専用計算機では 5,000 軌道くらいまでの系を対象としているが、その大きさは、今回計算した系 ( Crambin ) で

の軌道数の約 2.5 倍である。Crambin の場合ではバッファ総量が 2GB ( 1,024 プロセッサ換算時でのバッファサイズ = 2MB/proc ) で見かけの並列化効率が 1 を超えたこと、2 電子積分数は軌道数の 4 乗に比例すること、などを考慮すると、たとえば、5,000 軌道の計算を 1,024 並列で行う場合に、高々  $78 (= 2.5^4 \times 2)$  MB/proc のバッファを用いれば、ほぼ linear な性能向上が得られると予想される。

## 4 まとめ

メモリに 2 電子積分の一部を保存し、再利用する PDSCF 法を提案し、その性能評価を行った。積分タイプによる計算コストの比から、軌道量子数の和が小さな 2 電子積分から保存、再利用したほうが PDSCF 法の性能がよいことが明らかになった。また、プロセッサ当たりのバッファサイズが比較的小さい場合でも、使用するプロセッサ数を増やすと、PDSCF 法の見かけ上の並列化効率が大きくなることも明らかとなった。今回の計算は PC クラスタを用いて行ったが、この方法は、ディスクレスの並列計算機に一般的な並列 Fock 作成の高速化技法である。したがって、EHPC プロジェクトで開発を行ってきた、ERIC チップを多数搭載した超並列計算機への適用した場合には、使用するチップ数に応じたスケラブルな性能向上をもたらすと考えられる。

PDSCF 法は一種のデータのキャッシュ方法であるが、従来の計算機アーキテクチャサイドから提案されてきたキャッシュ機構とは異なる機構に基づいている。これは、すべての二電子積分値が一回の SCF サイクル中では、それぞれ一回しか出現しないため、メモリアクセスに対する時間や空間のローカリティを活用することはできないためである。そこで、キャッシングの優先度として二電子積分値 1 個あたりの計算負荷を指標に用いた。また、今回の実装では、メモリアドレスではなく、もっと抽象度の高い四重ループ IJKL のインデックスをキャッシュした二電子積分値のタグとしている。従って、データアクセスのローカリティから予測される出現頻度に基づいて機械論的にキャッシングの優先度を決定するキャッシュ機構と比べて、よりアプリケーションに近い所から発想されたアプリケーション指向の高速化技術であると言える。これは EHPC プロジェクトのような計算化学と計算機のそれぞれの研究者が緊密に連携した結果であり、今

後の計算機開発手法のひとつの方向を示すものと考えている。

本研究の一部は、科学技術振興費の総合研究「科学技術計算専用ロジック組み込み型プラットフォーム・アーキテクチャに関する研究」の支援による。また、すべての並列計算は Gfarm クラスタを用いて、産総研 (AIST) のグリッド研究センター (GTRC) で行った。

## 参考文献

- [1] Embedded High Performance Computing (EHPC) プロジェクト;  
URL: <http://www.ehpc.jp>
- [2] S. Obara and A. Saika, *J. Chem. Phys.*, **84**, 3963 (1986).

- [3] M. Head-Gordon and J. A. Pople, *J. Chem. Phys.*, **89**, 5777 (1988).
- [4] PCL - The Performance Counter Library Version 2.2, January 2003,  
<http://www.fz-juelich.de/zam/PCL/>
- [5] MPI: A Message-Passing Interface Standard, MPI forum, 1995.
- [6] Intel Fortran compiler 8.0 for Linux, Copyright © 1998-2003 Intel Corporation.
- [7] Intel C++ compiler 8.0 for Linux, Copyright © 2001-2003 Intel Corporation.
- [8] J. Almlöf, K. Faegri Jr. and K. Korsell, *J. Comput. Chem.*, **3**, 385 (1982).

# Performance Evaluation of Partially Direct SCF Method

Yuichi INADOMI<sup>a,b\*</sup>, Tohru SASAKI<sup>c</sup>, Umpei NAGASHIMA<sup>a,b</sup> and Kazuaki MURAKAMI<sup>d</sup>

<sup>a</sup>National Institute of Advanced Science and Technology, Grid Technology Research Center  
Central 2, 1-1-1 Umezono, Tsukuba, 305-8568, JAPAN

<sup>b</sup> Core Research for Evolutional Science and Technology, Japan Science and Technology Agency  
Kawaguchi Center building, 4-1-8 Honcho, Kawaguchi, 332-0012, JAPAN

<sup>c</sup>A priori Microsystems, Inc. Keio University  
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, JAPAN

<sup>d</sup>Graduate School of Information Science and Electrical Engineering  
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, JAPAN

*\*e-mail: y.inadomi@aist.go.jp*

The partially direct SCF-MO method was developed to improve the parallelization efficiency in the Fock matrix generation using a PC cluster without secondary storage on each processor. Some of the electron repulsion integrals are stored in buffer (unused memory) with their four indices at the first SCF cycle, and they are reused at the later SCF cycles. This simple method achieved super-linear scalability, for example, the parallelization efficiency became ca. 1.13 in the Fock matrix generation of the Crambin molecule (1974 basis functions), equipped by the 128 Xeon processors (2.8GHz) with 16GB buffer area (See Figure 4). This algorithm is suitable for the special purpose computers for fast evaluation of the electron repulsion integrals because the recent special purpose processor has usually no secondary storage and has a relatively large main memory.

**Keywords:** Molecular orbital method, Two-electron integral, Special purpose machine, Parallelization, Partially direct SCF method