

2 電子積分計算ルーチンの性能評価

稲富 雄一^{a,b}, 小原 繁^c, 長嶋 雲兵^{a,b}

^a 産業技術総合研究所計算科学研究部門, 〒 305-8568 つくば市梅園 1-1-1 中央第二

^b 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

^c 北海道教育大学釧路校, 〒 085-8580 北海道釧路市城山 1-15-55

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

(Received: July 28, 2005; Accepted for publication: November 7, 2005; Published on Web: December 15, 2005)

本論文では、分子軌道計算で最も時間のかかる 2 電子反発積分の計算を新小原法 [6, 7] と Vertical Recurrence Relation (VRR)[5] と Horizontal Recurrence Relation (HRR)[8] の hybrid アルゴリズムをもちいた 2 種類のプログラムで行い、Pentium4 (3.6GHz, EM64T, 1GB L2 cache) 上で浮動小数演算数や必要クロックサイクル数などのプロセッサイベントを計測することにより、その性能を評価した。

新小原法プログラムの浮動小数点演算数は、hybrid 法のそれに比べ 20%ほど少ないが、性能は hybrid 法プログラムの方が高かった。これは、新小原法プログラムのメモリアクセス回数が hybrid 法プログラムに比べ 3 倍ほど多く、キャッシュミスは 25 倍も多いことに起因する。このように浮動小数演算数が少なくても、メモリアクセスが多くなることで、計算速度が大きく低下することが分かった。したがって、2 電子積分を高速に行うためには、演算数だけでなく、中間積分などを保存、利用するためのメモリアクセス数を減少させる工夫が必要である。

キーワード: 2 電子積分, 性能カウンタ, メモリアクセス

1 はじめに

分子軌道 (MO) 法は、さまざまな化学現象を理論的に解明するための重要な手段である。Gaussian[1] や GAMESS[2]、あるいは MOPAC[3] などの様々なパッケージが開発されたことで、今では理論計算の専門家ではない化学、生物化学分野の研究者でも MO 計算を手軽に行えるまでになっている。しかし、経験的パラメータを一切含んでいない非経験的分子軌道計算では、対象とする系の大きさの 4 乗に比例する計算量 ($O(N^4)$) を持つ 2 電子積分の計算、およびその操作が計算時間の 95% 以上の割合を占めているため、計算時間の短縮や、より大規模系へ MO 法を適用するためには 2 電子積分を高速に計算する必要がある [4]。そのため、2 電子積分計算のための効率的なアルゴリズムの開発や、専用計算機の開発などが考えられる。これまでに積分の高速計算のために多くのアルゴリズムが提案されているが、Obara らが提案した一連のアルゴリズム [5–7] は、積和の形を持つ簡単な漸化式で表されており、専用計算機への実装も容易である。

ところで、著者らが関わっている専用計算機開発のプロジェクトで 2 電子積分専用プロセッサ (ERIC チップ) の開発を目指しているが、その設計を行う際には積分実行に必要なプロセッサ資源情報の収集、分析が必須である。そこで、今回 Obara ら [5–7]、あるいは Head-Gordon ら [8] が提案した 2 種類のアルゴリズムに基づいて作成したルーチンを、最近では科学技術計算にも広く用いられている Pentium4TM[9] プロセッサを用いて実行し、プロセッサが持つ performance counter を利用した積分ルーチンの性能評価を行った。一般には、必要演算数が少ないアルゴリズムほど、より高速に積分計算を行える、と思われがちであるが、今回の結果からは、演算数だけでなく、計算途中で必要となるメモリアクセス数も、積分計算の性能に大きく影響を及ぼすことが分かった。

2 評価方法

使用したプログラムは Kikuchi らによって開発された非経験的分子軌道計算プログラム ABINIT[10] の 2 電子積分計算部分を、1) Vertical Recurrence Relation (VRR)[5] と Horizontal Recurrence Relation (HRR)[8] の hybrid アルゴリズムを用いたルーチン (hybrid 法)、2) Obara らによる改良ルーチン (new Obara 法)[6, 7]、の 2 つでそれぞれ置き換えたものを用いた。各ルーチンの性能評価は、命令数やキャッシュミスなどを、Linux Performance-Monitoring Counters Driver (Perfctr)[11] を組み合わせた Performance Application Programming Interface (PAPI)[12] を用いて測定することで行った。実際の測定は、2 電子積分を行うルーチン呼び出す直前にカウンタをリセットし、計算終了直後にカウンタの値を取り出す、という手順で行った。それぞれのプログラムを 5 回実行したが、実行時間のばらつきは 0.1% 未満であった。ほかのプロセスの影響が最も小さいと考えられるため、実行時間が最も短かった場合における測定値を、測定結果として後ほど報告する。計算の対象分子は Glycine ($C_2H_5NO_2$) で、用いた基底関数は 6-31G* であり、基底関数の数は 85 である。プロセッサ、コンパイラなどの使用した計算機環境の詳細は Table 1 に示した。

3 結果と考察

PAPI を用いて測定したプロセッサイベントの数を Table 2 に示す。new Obara 法のカッコ内の値は、Hybrid 法の対応する数値に対する比を表している。まず、浮動小数演算数 (Floating-Point Operations) の値について考える。これは、倍精度実数に対する四則演算などの数に、整数 (固定小数) から浮動小数への変換などの操作を加えた値であり、積分アルゴリズムの良し悪しを決める指標として一般に用いられる「演算数」に対応する。この値を見ると、hybrid 法では 1.7×10^7

回であるのに対して、new Obara 法では 1.4×10^7 回と、hybrid 法に比べて 2 割ほど少ない演算回数で積分計算を実行できていることが分かり、演算数の点から見ると、new Obara 法が hybrid 法よりも性能が高いことが分かる。しかし、実際の計算時間と直接対応している値である必要クロックサイクル数 (Total Cycles) は、hybrid 法に比べると new Obara 法は 3 割以上大きい。すなわち、new Obara 法のほうが hybrid 法に比べて演算数では 2 割ほど少ないにもかかわらず、計算時間は 3 割以上長い、という結果になっている。この原因は、new Obara 法のメモリアクセスの多さにある。Table 2 中の “Load/Store Instructions” は、メモリアクセス数に相当するが、これを見ると、new Obara 法は hybrid 法に比べて 3.4 倍のメモリアクセスを行っていることがわかる。メモリアクセス数の増加は、当然ながらキャッシュミスの増加も招いており、レベル 1 データキャッシュミス (Level 1 Data Cache Miss) の値を比べると、new Obara 法では hybrid 法の 25 倍近い頻度でキャッシュミスを起こしていることがわかる。キャッシュミスが起こると、レベル 2 キャッシュ、あるいはアクセス速度がプロセッサに比べて格段に遅いメインメモリにまでアクセスする必要が出てくる。データがレジスタにロードされるまではプロセッサの演算の実行が中断するため、キャッシュミスはプログラムの実効性能を大きく低下させる要因になる。また、メモリアクセスはアドレス計算を伴うことが多いため、メモリアクセス命令だけでなく、整数演算も増加し、総命令数の増加にもつながる。実際に、積分計算に実行に要したプロセッサの総命令数 (Total Instructions、浮動小数演算数、メモリアクセス数なども含む) の値を見ると、new Obara 法は hybrid 法の 2 倍もあることが分かる。

したがって、高速に 2 電子積分計算を行うためには、演算数を減少させるだけではなく、計算途中で保存しておくべき中間積分などのデータ量を減らす工夫も必要である。

Table 1. Computational environment used in calculation

processor	Pentium4 (3.6GHz, EM64T, 1GB L2 cache)
OS	Fedora Core 3 (kernel 2.6.11 with Perfctr patches)
PAPI version	3.0.8.1
Perfctr version	2.6.13
C compiler	Intel C++ compiler (version 8.0, option = "-O3 -static -xP") [13]
Fortran compiler	Intel Fortran compiler (version 8.0, option = "-O3 -static -xP") [14]

Table 2. Number of counter value for each processor event in calculating two-electron integrals ^a using hybrid ^b and new Obara ^c algorithms

Counter Content	Hybrid	New Obara	(ratio ^d)
Floating-Point Operations	17,477,083	14,483,404	(0.83)
Total Cycles	181,076,389	245,536,511	(1.36)
Load/Store Instructions	18,941,281	64,709,252	(3.42)
Level 1 Data Cache Miss	34,909	870,671	(24.9)
Total Instructions	58,087,253	121,364,431	(2.09)

a; Target molecule is Glycine using 6-31G* basis set (including 10 atoms and 85 basis functions)

b; Hybrid algorithm between the vertical recurrence relation (VRR) [5] and horizontal recurrence relation (HRR) [8]

c; See references [6, 7]

d; Ratios vs. corresponding data in hybrid algorithm

4 まとめ

今回、分子軌道計算において計算量が最も多く、高速化が求められている2電子積分計算に対して、2種類の計算ルーチンを用いて、プロセッサ資源を指標とした性能評価を行った。その結果、積分の高速計算には、浮動小数演算数を減少させるだけでなく、積分計算の途中で出てくる保存すべき中間積分などの各種パラメータの数を減らし、メモリアクセス回数を減らすことが非常に重要であることが分かった。

先にも述べたとおり、積分アルゴリズムの性能を評価する際の指標の1つとして、よく演算数が用いられている。しかし、ユーザから見た場合は、「計算時間が短い」、すなわち積分に要するクロックサイクル数が少ない、ことだけが重要である。積分プログラムの性能は、アルゴリズムの良し悪しだけでなく、キャッシュの構成や大きさ、演算器構成などのプロセッサアーキテクチャや、プログラムのコーディング手法に大きく左右される。ただ、「memory wall」という言葉が使われるほど、現在では、プロセッサ速度とメモリアクセス速度との間に大きな隔たりが生じており、その差は、今後、さらに広がっていくものと思われる。したがって、積分アルゴリズム開発、ならびにプログラム作成を行う際には、メモリアクセス数の減少も念頭に置くべきであり、2電子積分専用プロセッサ(ERIC)開発の際にも、積分計算中に頻りにアクセスする必要のある中間積分データが十分に収まるようなキャッシュ容量を考える、などの対策が必要である。

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(EHPCプロジェクト)」の支援による。

参考文献

- [1] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, Jr., R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, A. G. Baboul, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P. M. W. Gill, B. G. Johnson, W. Chen, M. W. Wong, J. L. Andres, M. Head-Gordon, E. S. Replogle and J. A. Pople, *Gaussian 98*, Gaussian, Inc., Pittsburgh PA (1998).
- [2] 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. J. Su, T. L. Windus, M. Dupuis, J. A. Montgomery, *GAMESS, J. Comput. Chem.*, **14**, 1347-1363 (1993).

- [3] *MOPAC2000 ver.1.0*, Fujitsu Ltd, Tokyo, Japan (1999).
- [4] 中野達也、高島一、長嶋雲兵, *シミュレーション*, **19(4)**, 271-281 (2000).
- [5] S. Obara and A. Saika, *J. Chem. Phys.*, **84**, 3963 (1986).
- [6] M. Honda, K. Sato and S. Obara, *J. Chem. Phys.*, **94**, 3790 (1991).
- [7] H. Honda, T. Yamaki and S. Obara, *J. Chem. Phys.*, **117**, 1457 (2002).
- [8] M. Head-Gordon and J. A. Pople, *J. Chem. Phys.*, **89**, 5777 (1988).
- [9] Pentium 4, Trade mark of Intel Corporation
- [10] O. Kikuchi, K. Morihashi, T. Nakano and Y. Inadomi, ABINIT, Ab initio program package, University of Tsukuba,(1999)
- [11] The Linux performance monitoring counters kernel extension (Perfctr) (version 2.6.13), Mikael Pettersson
<http://user.it.uu.se/~mikpe/linux/perfctr/>
- [12] PAPI, Performance Application Programming Interface (version 3.0.8.1)
<http://icl.cs.utk.edu/papi/>
- [13] Intel C++ compiler 8.0 for Linux, Copyright© 2001-2003 Intel Corporation
- [14] Intel Fortran compiler 8.0 for Linux, Copyright© 1998-2003 Intel Corporation

Performance Evaluation of the Calculation Programs of Electron Repulsion Integrals

Yuichi INADOMI^{a,b}, Shigeru OBARA^c and Umpei NAGASHIMA^{a,b}

^aNational Institute of Advanced Industrial Science and Technology, Research Institute for Computational Sciences
Central 2, 1-1-1 Umezono, Tsukuba, 305-8568, JAPAN

^bCore Research for Evolutional Science and Technology, Japan Science and Technology Agency
Kawaguchi Center building, 4-1-8 Honcho, Kawaguchi, 332-0012, JAPAN

^cHokkaido University of Education, Kushiro Campus
1-15-55 Shiroyama, Kushiro, Hokkaido, 085-8580, JAPAN

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

Performance evaluation of two programs for electron repulsion integral calculation was done using the performance counter on Intel Pentium 4 processor (3.6GHz, EM64T, 1GB L2 cache). The programs were of the new Obara method [6, 7], and of the hybrid method of Vertical Recurrence Relation method (VRR)[5] and Horizontal Recurrence Relation method (HRR)[8]. Though the floating point operations of the new Obara method are almost 20% smaller than those of the hybrid method, the total clock cycle, corresponding to the wall clock time, was more than 30% larger than that for the hybrid. The performance decrease is mainly due to memory access because load/store instructions of the new Obara method are almost 3 times larger, and the level 1 data cache difference is 25 times larger than that of the hybrid.

It becomes clear that the reduction of the memory accesses is very important to improve the performance of integral calculation as well as the reduction of the number of floating point operations.

Keywords: Two-electron integral, Performance counter, Memory access