4P087

SIMD 演算向け原始基底 RHF 計算

○今村憲亮[1], 本田宏明[2,3], 稲富雄一[1,3], 南里豪志[2,3]
(九大シス情[1], 九大情基センター[2], JST-CREST[3])

RHF calculation based on primitive basis for SIMD operation

○ Noriaki Imamura[1], Hiroaki Honda[2,3], Yuichi Inadomi[1,3], Takeshi Nanri[2,3]
(ISEE, Kyushu Univ.[1], RIIT, Kyushu Univ.[2], JST-CREST[3])

【はじめに】

京をはじめとしたスーパーコンピュータの高い演算性能は，相互結合網内の多数の計算ノード，各計算ノード内の複数のプロセッサ，プロセッサ内メニーコア，各コア上の SIMD 演算器といった階層的並列環境によって実現されている．本研究では特に SIMD 演算器の有効利用に着目している．SIMD 演算を利用することで複数の同一演算を単一命令で処理することが可能となる．今後開発される高性能計算向けプロセッサでは，高性能低消費電力化の要請への対処の一つとして，16 ウェイや 32 ウェイの倍精度浮動小数点 SIMD 演算が可能になると予想されており，SIMD 演算の有効利用の重要性はより一層高まっている．しかしながら，現在の Hartree-Fock 法プログラムでは，その実行時間ボトルネックとなる二電子 Fock 行列 ($\boldsymbol{G}$ 行列) 計算において，直接には SIMD 演算の有効利用が難しい．この問題に対し，GPGPU を利用した高速化に関する研究 [1-4] や複数入力並列計算による SIMD 演算器の効率的利用に関する研究 [5] がある．しかしながら，通常の 1 分子計算における SIMD 演算の有効利用に関しての研究報告はなされていない．

そこで本研究では，$\boldsymbol{G}$ 行列計算における効率的な SIMD 演算の利用法を提案し，SIMD 演算率ならびに実行時間に関する性能評価を行うことを目的とする．

【SIMD 演算の有効利用に向けたコード変更】

一般的に，$\boldsymbol{G}$ 行列計算は多重ループ構造をとる．ループ箇所に SIMD 演算を適用する際には，1．多重ループにより記述されている場合，最内ループである，2．ループ内部にジャンプ命令や多数の分岐先を持つ条件分岐を含まない，3．ループに入る以前に反復回数が判明しており，ループ内部にデータ依存性のある出口条件を含まない，4．後方へのループ伝播の依存性がない，すなわち，計算結果がループの反復順序によらない，の条件を満たすことが必要である．

しかしながら，既存の $\boldsymbol{G}$ 行列計算コードは，主に，1．多重ループの最内ループはループ長が短く，SIMD 演算器のウェイ数に対して十分なデータ並列性が確保できない，2．ループ内部に条件分岐が存在する，といった問題から SIMD 演算を有効に利用できていないと考えられる．

これらの問題に対し本研究では，これまでの縮約基底ではなく原始基底に基づく方法を試み SIMD 演算の効率的利用を図る．この際，下位縮約ループの上位シェルループへの組み込みを行うことで，SIMD 演算器に対して十分なデータ並列性のある長いループ長をもつ 2 重ループのみにすることを可能とした．さらに，SIMD 演算箇所をコンパイラへ明確に指示するためループボディ部では配列表記を利用した実装を行った．また条件分岐に対しては，当該箇所を内包するループを新たに作成し逐次処理を施す実装を試みた．

【性能評価実験】

本研究で実施したコード変更について，RHF 法プログラムの性能に対する影響の調査のため，SIMD 演算率ならびに実行時間に関して性能評価実験を行った．今回は $\boldsymbol{G}$ 行列計算の中でも最も演算数の少ない $(ss|ss)$ 型計算ルーチンのみを対象とした．

本実験ではアミノ酸の 1 種であるトリプトファン分子を入力とし，九大の稲富らによって開発されている OpenFMO[6] にて利用されている RHF 法プログラムを用いた．また，Intel Xeon E5-2620 を 2CPU 搭載した計算機を用い，12 スレッドで並列計算を行った．プログラムのコンパイルには Intel Compiler 14.0.0 を利用している．

実験結果を図-1, 2 に示す．ここで，Contracted は既存プログラム，Primitive は原始基底計算

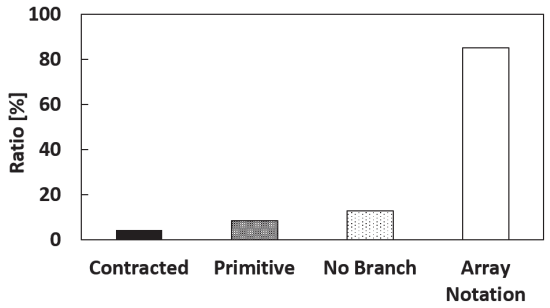

図 1: 各コード変更における $(ss|ss)$ 型 $\boldsymbol{G}$ 行列計算の SIMD 演算率比較

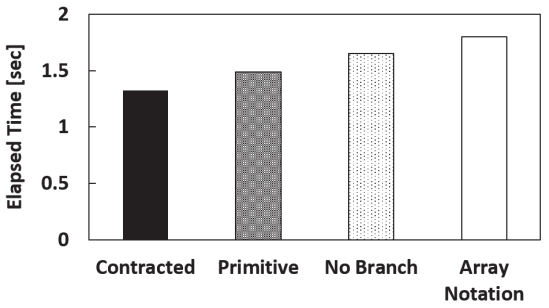

図 2: 各コード変更における $(ss|ss)$ 型 $\boldsymbol{G}$ 行列計算の所要時間比較

プログラム，No Branch は Primitive プログラムの条件分岐箇所のコード変更を施したプログラム，Array Notation は No Branch プログラムに対し配列表記を利用したループボディ部変更プログラムをそれぞれ示している．なお，SIMD 演算率の算出にはハードウェアカウンタ取得ライブラリ PAPI[7] を利用しており，各プログラムの実行時間は全 12 スレッドの処理時間の総和である．

図 1 から，$(ss|ss)$ 型 $\boldsymbol{G}$ 行列計算における SIMD 演算率は，原始基底に基づく計算への変更によって向上した．特に，条件分岐ならびにループボディ部への配列表記によるコード変更を加えることで SIMD 演算率は既存プログラムの 4.4% から 85.2% まで向上しており，SIMD 演算の有効利用を狙ったコード変更の効果が顕著に表れていると言える．

しかしながら，図 2 から，実行時間性能としては既存プログラムが最も優れており，原始基底プログラムは 1.12 倍，条件分岐除去プログラムは 1.25 倍，配列表記利用プログラムは 1.36 倍，実行時間が増加する結果となった．これに対しては，今回取得した演算数に関わるハードウェアカウンタ値に加え，データのロード・ストアのインストラクション数やキャッシュヒット・ミスなどに関わるデータを取得することにより性能低下の要因の解析が可能になると考えられる．

当日は，本実験結果に対する詳細な実行時間解析結果を報告する予定である．

【参考文献】

[1] K.Yasuda, J. Comp. Chem., Vol.29, pp.334-342, 2008.
[2] I.Ufimtsev, and T.Martnez, J. Chem. Theo. Comp., Vol.4, pp222-231, 2008.
[3] Y.Miao and K.M. Merz, J. Chem. Theo. Comp., Vol.9, pp.965-976, 2013.
[4] 梅田宏明 他, 情報処理学会論文誌 コンピューティングシステム, Vol.6, pp.26-37 (2013).
[5] 本田宏明 他, 情報処理学会論文誌 コンピューティングシステム, Vol.7, pp.15-24 (2014).
[6] OpenFMO Project (online) available from ⟨http://www.openfmo.org/⟩.
[7] PAPI: (online) available from ⟨http://icl.cs.utk.edu/papi/⟩.