1P128

RT 並列法によるメニーコアプロセッサ向け分子軌道法計算

(九大情基センター [a], 九州先端研 [b]) 本田宏明[a], 稲富雄一 [a], 真木淳 [b]

Molecular orbital calculation on many-core processor using RT parallelization method

(RIIT, Kyushu Univ.[a], ISIT[b]) Hiroaki Honda[a], Yuichi Inadomi[a], Jun Maki[b]

【はじめに】

現在、ノートブックからサーバコンピュータに至る多くのコンピュータで複数個の計算コアを持つマルチコアプロセッサが使用されている。現状では 16 コア以下が大半であるが、今後もコア数は増加してゆき数年以内には非常に多くのコア (> 100 コア) を持つメニーコアプロセッサも利用可能となると予想されている。また高性能低消費電力化の要請から、今後使用されるハイエンドプロセッサは、主要なものでも *A*: 100 コア以上のメニーコア構成、*B*: プロセッサあたりの主記憶量の低減ならびに主記憶に対するコアあたりの提供メモリバンド幅の低下、*C*: 16 ウェイ以上の SIMD 演算器、*D*: 命令処理におけるインオーダー実行、*E*: バス方式からパケットスイッチネットワーク通信型へのコア間通信方式の変更、といった特徴を持つと予想される。

そこで本研究では 2 電子フォック行列 (G 行列) 計算を対象とし、上記のうち *A*、*B*、*C* の特徴を取り上げ、*A* と *B* については RT 並列化法の適用を、*C* についてはデータ並列計算手法の適用を試み、その有効性の検証を行なう事を目的とした。

【RT 並列化法】

高島らによって開発された G 行列計算に対する RT 並列化法 [1] は、元々は各計算プロセッシングエレメントに 2.5MB 程度の非常に小さなメモリのみ搭載した、分散メモリアーキテクチャのアクセラレーター向けに開発され、図 1 に示すように添字駆動型と積分駆動型の G 行列計算方法の混ざりあった計算アルゴリズムとなっている。その利点として、各並列計算において必要な G 行列や密度行列 (D 行列) 量は各 4 列ずつとなっており、データ転送量に関わるコストが著しく少ない点が挙げられる。また、各並列計算部分ではこの転送された小さなサイズの G,D 行列片による計算が可能であるため、5000 次元の計算を行う場合でも 0.6MB 程度のメモリ量にて計算が可能となる。

```
Loop I  = 1, N
  Loop K = 1, I
    D_IJ(1<=J<=I), D_KJ(1<=J<=I), D_IL(1<=L<=I), D_IL(1<=L<=I)     データ転送
    Loop J = 1, I
      Loop L= 1, I
       (IJ,KL) 積分計算
       if ( J<= I && K<=I && L<=I ) then
                                        G_IJ +=D_KL * (IJ,KL)
        if ( L<=K && K < I ) then
           if ( J < I )                 G_KL +=D_IJ * (IJ,KL) * 2
           else                         G_KL +=D_IJ * (IJ,KL)
        endif
                                        G_IL −=D_KJ * (IJ,KL)
        if ( K <= J && J < I )          G_JK −=D_IL * (IJ,KL)
        if ( J <= K && K < I && L < I ) G_KJ −=D_IL * (IJ,KL)
       endif
    End Loop J, L
    G_IJ(1<=J<=I),  G_KJ(1<=J<=I), G_IL(1<=L<=I),  G_IL(1<=L<=I)     データ転送
End Loop I, K
```

図 1: RT 並列化アルゴリズム

更には 1000 次元程度の場合には 120KB 程度となるため、キャッシュメモリのみにて十分な計算が可能と考えられる。その一方、通常の積分駆動型計算に比較し、2 倍程度積分を重複して計算してしまう欠点を持つ。その結果、コア数の少ない場合では積分駆動型並列計算に比較し実行時間の面で不利だが、数百コア以上の並列計算を行う場合では有利となると予想されている。これらの特性により、数個のメニーコアプロセッサによる並列計算を行う場合では、メニーコアの特徴である多数のコアの有効利用を可能とし、主記憶バンド幅の低下問題の緩和に役立つ可能性がある。

【データ並列化法による SIMD 計算】

一般に、コンパイラの自動並列化機能を利用してコード中の多重ループ箇所の SIMD 化オブジェクトコード生成を行うには、最内ループが容易にループアンローリング可能であることが必要である。しかしながら通常の G 行列計算コードでは、最内ループが縮約計算であるためにそのループ長が外側ループ添字に依存しており、自動的には SIMD 化されない。そこで、ポテンシャルエネルギー面作成など、分子座標が異なる複数の入力データを並列に計算する様なデータ並列計算を対象とし、最内ループをデータ並列計算に割り当てることにより SIMD 化オブジェクトコード生成を試みた。また、SIMD 計算を効率的に実行するためには SIMD 演算対象のデータが連続領域に確保されている必要がある。そのため、G 行列計算部分では、積分計算に必要な全ての浮動小数点データや D, G 行列の全要素をデータ並列分だけ連続領域に多重化して保持するよう変更を行なった。

このデータ並列計算は前節の RT 並列化法とは全く異なる最適化方法であり、両方の実装を組合せる事が可能である。

【性能評価テスト】

上記の RT 並列化手法ならびにデータ並列化について (ss, ss) 積分を対象とした 逐次実行 G 行列計算の実装を行い、有効性の検証を行なった。RT 並列化手法において逐次実行する際にはデータ転送部分はメモリコピーにて代用した。使用コンピュータは Intel Xeon X5650 (2.67GHz, 6 コア, 倍精度浮動小数点 2 ウェイ SIMD 演算器) プロセッサ、DDR3 DIMM 16GB メモリからなり、コンパイル条件として Intel コンパイラ 12.1.0 による O3 最適化を使用した。性能計測には Perfmon2 ライブラリ [2] によるハードウエアカウンターデータの取得を行った。

複数の入力ファイルを対象としたデータ並列計算における SIMD 計算の有効利用に対する結果を図 2 に示す。6-31G 基底関数の $(H_2O)_8$ 分子を対象とし、分子構造パラメータのみ異なる 32 個の入力データを準備し、1~32 個の入力に対するデータ並列計算を行なった。グラフ中、折れ線は全実行浮動小数点命令数中の SIMD 浮動小数点命令数比を示しており、棒グラフはデータ並列を行わないプログラムを基準とした場合の、実行クロックサイクル数/データ並列度 に対する速度向上比を示している。この結果、データ並列が無い場合では、ほぼ 0% であった SIMD 化浮動小数点演算比が、並列度の増加に従い急増し、16 並列では最大 0.99 と、ほとんどの浮動小数点演算が SIMD 化される事が分った。実行クロックサイクル数を計測したデータ並列度あたりの速度向上比はデータ並列度が向上するに従い 2 に近づいており、整数演算等からなる逐次実行の影響もそれほど大きくはなく、全体として 2 ウェイ SIMD 演算器を使用した効率的な計算が実行されていることが分った。この結果から将来のウェイ数の増加に対しさらなる性能向上が可能であると期待される。

RT 並列化についての結果等については当日報告する。

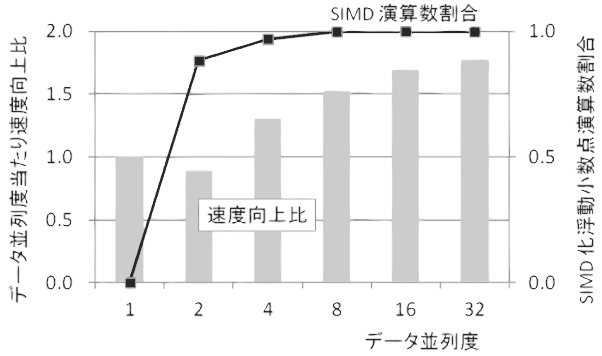

図 2: データ並列計算におけるデータ並列度あたりの速度向上率と
全浮動小数点演算数における SIMD 演算数の比

[1] H.Takashima *et al.*, *J.Comput.Chem.*, Vol.23, pp.1337-1346, 2002.

[2] http://perfmon2.sourceforge.net/