jh210036-NAH

FMO プログラム ABINIT-MP の高速化と超大規模系への対応

望月 祐志（立教大学）

概要
本課題では、A64FX スーパーコンピュータ「不老」 Type I をプラットフォームとしてフラグメント分子軌道(FMO)プログラム ABINIT-MP の高速化、ならびにフラグメント総数 1 万を超える超大規模系への対応を図ります。高速化に関しては、標準的な 2 次摂動計算（FMO-MP2/6-31G*）ジョブに対してコスト分析を行い、2 電子積分の生成ルーチン群に対して SIMD 化等のチューニングを施して 20-30%の高速化を得ました。また、通信周りの改善も図り、て、ジョブの加速としては従前比で 1.2～1.5 倍となりました(計算対象に依存)。計算対象の拡大に関しては、不要な情報の保持配列を削減し、1.1 万フラグメントの扱いを可能としました。これらの改善を反映した Version 2 Revision 4 を 2021 年 9 月にリリースし、「不老」や「富岳」などの HPCI 拠点でライブラリ公開しました。改良は、その後も続けており、最新のローカル版での加速は 1.7 倍となっています。また、水のクラスターでは 2 万フラグメントも達成しています。こうした、改良活動は 2022 年度も jh220010 として継続されています。

1. 共同研究に関する情報

(1) 共同研究を実施した拠点名

名古屋大学

(2) 共同研究分野

超大規模数値計算系応用分野

(3) 参加研究者の役割分担

望月が全体の進捗を管理すると共に、テスト計算を手掛けています。また、学会発表などの情報発信も行っています。中野（国立医薬品食品衛生研究所）と坂倉（計算科学振興財団）は、望月と連携してプログラムの改良を進めています。渡邊（HPC システムズ）は性能の詳細評価を担当しています。片桐と大島（名古屋大学）は「不老」の責任者として計算機科学の立場から適切なアドバイスを行っています。大学院生（秋澤、山梨、森下）は教員の指導を受けてコミットしています。

2. 研究の目的と意義

2020 年度に「富岳」の試行的利用期間で実施された新型コロナウイルスの特別プロジェクトの g9330001 課題「新型コロナウイルス関連タンパク質に対するフラグメント分子軌道計算」では、多数の計算化学的な成果を論文や学会発表などで出すことが出来ました。しかし、ABINIT-MP の改良すべき点として、**(1)** 高速化、**(2)** 超大規模系への対応、の 2 つが顕在化しました。本課題（jh210036-NAH）では、この 2 点の改良に取り組むことが大きな目的です。

(1)では、2020 年 6 月にリリースした Open Version 1 Revision 22（以下、Ver. 1 Rev. 22 と略記）に比して、「不老」 Type I や「富岳」などの A64FX スパコン上で、FMO の応用計算で常用される FMO-MP2 ジョブで 2 倍の加速を第一段階の目標として作業を進めました。具体的には、コストの多くを占める 2 電子積分の生成に係るルーチン群の SIMD 化などの改善を積み重ねました。

(2)では、分子動力学（MD）シミュレーションから生成されるタンパク質の液滴状の水和モデル(対イオンを含む)を対象とし、Ver. 1 Rev. 22 では不可能だったフラグメント総数で 1 万を超える系の扱いを意図しました（この場合、タンパク質本体のアミノ酸残基数は 2 千～2.5

千程度)。対策としては、ここまで可視化解析のためにサポートしてきた専用 GUI である BioStation Viewer のために保持していたフラグメント数の自乗依存性を持つ情報配列を複数削除することにしました。結果的に、BioStation Viewer との接続は放棄することになりました。

これらの改良作業を加えていく ABINIT-MP は、Ver. 2 シリーズとして Ver. 1 と同様に HPCI 拠点でライブラリプログラムとして一般利用のために公開する方針を取っています。2021 年 9 月には、Rev. 4 として初のリリースを行いました。この版では(1)に関して、標準的な 2 次摂動レベルの FMO-MP2/6-31G*ジョブでの速度向上は Ver. 1 Rev. 22 との比較で 1.2～1.5 倍となりました(計算対象に依存します)。(2)では、総数 1.1 万フラグメントのタンパク質液滴モデルの FMO 計算が可能となり、扱える規模的には Ver. 1 Rev. 22 の 2 倍を超えました（成果の学術論文の[1]に記載)。

ABINIT-MP Ver. 2 Rev. 4 の公開に関しても鋭意推進し、理研-計算科学研究センターの「富岳」、名古屋大学の「不老」 Type I 、東京大学の Wisteria/Odyssey の A64FX スパコンはもちろん、複数の HPCI 拠点にライブラリプログラムとして整備しました。Xeon 系スパコンでは、(1)は有効ではありませんが、(2)の恩恵は受けるのと、別動プロジェクトで実施した機能追加・改良が反映されています。Ver. 2 Rev. 4 を広く一般利用可能としたことは、公益に適う成果と考えます。一方で、BioStation Viewer の既存ユーザーの利便も考慮し、Ver. 1 Rev. 22 もそのまま併存させています。

本課題の 3 つ目の内容は、(3) フラグメント間の相互作用エネルギー（IFIE）の可視化マップ（IFIE-map）の深層学習です。これは、タンパク質の構造がα-ヘリックス、β-シートを持つかによってアミノ酸残基間の IFIE-map 上で特徴的なパターンを呈することに着目し、複数のタンパク質の MD-FMO 連携シミュレーションで生成された画像ファイルを TensorFlow で学習させ、画像ファイルからα-ヘリックスとβ-シートの存在を判定させるものです。実行では、「不老」Type II の NVIDIA の GPU を用いて CPU（Xeon）のみの場合に比べて 11 倍の加速を得ました。今後、MD-FMO 連携での大規模計算が常態化すると結果データが膨大になるため、直截な統計的解析（平均や標準偏差など）だけでなく、特異値分解やテンソル分解を使って物理化学的に重要な数値指標を抽出する必要が増しますが、それと共に一旦可視化した上で特徴量を深層学習から演繹するアプローチも有用になると思われます。今回の(3)の試行は、その端緒と位置づけられます。

3. 当拠点公募型研究として実施した意義

ABINIT-MP の研究開発は 20 年以上の長きに渡りますが、Ver. 1 系までは、計算機科学の専門家との直接的なコラボレーションはせずに、望月、中野、坂倉らといった計算化学・量子化学のバックグラウンドを持つ者が主要コーダーとなってきました。しかし、「富岳」に代表される近年のメニーコア/超並列スパコンの利活用を図るためには、計算機科学の研究者との共同研究が不可欠になってきました。そこで、名古屋大学の情報基盤センターの教員で、高性能計算を専門としている片桐、大島との協調を 2020 年度下期から始めて本課題の提案に至りました。

片桐らが管理している「不老」 Type I は「富岳」に比べて比較的混雑していないため、A64FX での性能評価とコード改良を繰り返すには好適なプラットフォームと言えます。また、動作周波数が 2.2 GHz で全ノードが固定されていることも好条件と言えます。また、大島が担当する「不老」 Type II は機械学習向けのスパコンで、TensorFlow の GPU での実行のセットアップも直截です。

このように、本課題の活動の拠点として名古屋大学は人員的にも計算機環境的にも整って

います。同時に、本課題は計算化学・量子化学と計算機科学のインタープレイの意義も帯びており、学際性を重視する JHPCN の目的にも合致しています。

4. 前年度までに得られた研究成果の概要

該当しません。

5. 今年度の研究成果の詳細

先ず、最も利用される FMO-MP2 計算のフローについて記します（"Recent Advances of the Fragment Molecular Orbital Method – Enhanced Performance and Applicability", ed. Y. Mochizuki, S. Tanaka, K. Fukuzawa, (January 2021, Springer)を参照)。最初に、フラグメントの単体（モノマー）の環境静電ポテンシャル（ESP）が自己無撞着電荷（SCC）条件を達成するまでハートリーフォック（HF）計算が反復されます（いわゆるモノマーSCC サイクル)。SCC 到達後、各モノマーで MP2 計算が（一度だけ）実行されます。この後、モノマー段階で決せられた ESP 下で、フラグメントの（近接する）対（ダイマー）のリストに対して HF と MP2 が行われて、最終的なエネルギーが得られます。図 1 は、PDB-ID=6LU7 の新型コロナウイルスのメインプロテアーゼで測定した FMO-MP2/6-31G*計算の並列スケーリング性能で、橙色が「富岳」での結果となります。このように ABINIT-MP の並列性能は十分に高いです。実の

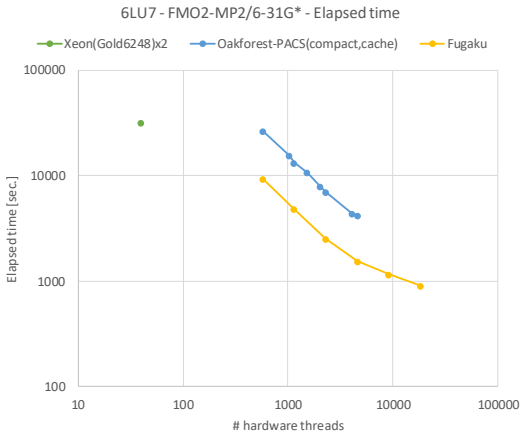

図 1. FMO-MP2 ジョブのスケーリング性能例

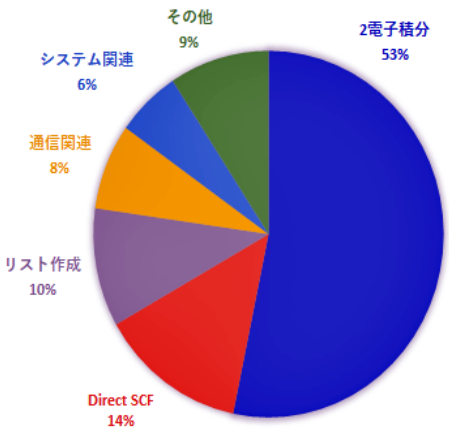

図 2. FMO-MP2 ジョブのコスト例

ところ、フラグメントの粒度を揃えた $Trp_{383}His$ モデルでは、実行並列化率は 99.999723%、並列化効率は 95.147045%に達しています。従って、ジョブ全体の高速化には演算部分の改良が直截なルートになります。

(1)の FMO-MP2 ジョブの高速化の作業では試行錯誤を容易にするため、小型で粒度を揃えた Ala_9Gly モデルを用い、2 ノードで 12 スレッド×8 プロセスで実行することにしました。基底関数は、6-31G*です。図 2 は、このジョブのコスト分析の結果で、"2 電子積分"の生成が全コストの 58%を占め、生成された積分からハ HF 計算を行うための"リスト作成"（添え字処理とパッキング）と"Direct SCF"とで 24%となっています（約 3/4 のコスト)。MP2 計算の実体は、生成された 2 電子積分の 4 段での線形変換で、DGEMM 処理が可能なことと、積分生成が（この小さな系では）一度で済むので目立ちません。そこで、積分生成を高速化の第一対象としました。ABINIT-MP の積分ルーチン群は小原の漸化式アルゴリズムに基づき、4 個の軌道タイプの組み合わせに応じて自動生成ツールによって作成されています。A64FX 系の高速化の基本レシピの一つは SIMD 化ですので、OCL 指示詞の挿入と一部作業配列のスカラ変数化を、コストが相対的に大きい(ss|ss)～(ss|sd)の 15 個のルーチンに対して手動で行い、コンパイルオプションも見直しました(Kfast 有効など)。図 3 に、実際に手を入れた積分ルーチンの具体例とし

```
subroutine sub_sssp(zetam,pm,dkabm,etam,qm,dkcdm, &
     ma,mb,mc,md,ngij,ngkl,a,b,c,d,sint,tv)
  !
  !        Nov.05,'02
  !                T.NAKANO & Y. ABE
  !
  use constant
  use auxiliary_integral_table                                  !
  use integral_parameter                                        !
  implicit none                                                 !
  real(8),intent(in)::zetam(*),pm(3,*),dkabm(*), &              !
       etam(*),qm(3,*),dkcdm(*)                                 !
  integer,intent(in)::ma,mb,mc,md,ngij,ngkl                     !
  real(8),intent(in)::a(3),b(3),c(3),d(3),tv                    !
  real(8),intent(out)::sint(*)                                  !
  !---------------------------------------
  integer npq,nrs,ix
  real(8) p(3),q(3),qd(3),pq(3),wq(3),f(0:max_m), &
       dkab,zeta,dkcd,eta,ze,rz,re,rho,a0,tt
  integer ts,i,j,k,l,m
  real(8) delta,t_inv
  real(8) ssss(0:1),f0,f1,qd1,qd2,qd3,wq1,wq2,wq3

  sint(1:3) = 0.0_8
                               !ocl eval
                               !ocl fp_relaxed                  !
                               !ocl fp_contract                 !
                               !ocl noswp
                               !ocl eval_concurrent
                               !ocl SIMD
  do npq=1,ngij
    if (abs(dkabm(npq)) > tv) then
      do nrs=1,ngkl
        if (abs(dkabm(npq)*dkcdm(nrs)) > tv) then
          ze  = 1.0_8/(zetam(npq)+etam(nrs))
          a0  = dkabm(npq)*dkcdm(nrs)*sqrt(ze)
          rz  = etam(nrs)*ze
          re  = zetam(npq)*ze                                   !
          rho = zetam(npq)*rz

          do i=1,3                                              !
!           qd(i) = qm(i,nrs)-d(i)                              !
            pq(i) = qm(i,nrs)-pm(i,npq)                         !
!           wq(i) =-re*pq(i)
          end do

            qd1 = qm(1,nrs)-d(1)
            qd2 = qm(2,nrs)-d(2)
            qd3 = qm(3,nrs)-d(3)
            wq1 =-re*pq(1)
            wq2 =-re*pq(2)
            wq3 =-re*pq(3)

            tt = (pq(1)*pq(1)+pq(2)*pq(2)+pq(3)*pq(3))*rho

            if (tt <= 38.0_8) then       ! Tf = 2*m+36 (for the case of m=1)
              ts    = 0.5_8+tt*fmt_inv_step_size
              delta = ts*fmt_step_size-tt

              f(0) = ((fmt_table(3,ts)*inv6*delta  &
                   +   fmt_table(2,ts)*inv2)*delta &
                   +   fmt_table(1,ts))*delta      &
                   +   fmt_table(0,ts)
              f(1) = ((fmt_table(4,ts)*inv6*delta  &
                   +   fmt_table(3,ts)*inv2)*delta &
                   +   fmt_table(2,ts))*delta      &
                   +   fmt_table(1,ts)
              f0   = ((fmt_table(3,ts)*inv6*delta  &
                   +   fmt_table(2,ts)*inv2)*delta &
                   +   fmt_table(1,ts))*delta      &
                   +   fmt_table(0,ts)
              f1   = ((fmt_table(4,ts)*inv6*delta  &
                   +   fmt_table(3,ts)*inv2)*delta &
                   +   fmt_table(2,ts))*delta      &
                   +   fmt_table(1,ts)
            else
              t_inv = inv2/tt
              f(0) = sqrt(pi_over2*t_inv)                       !
              f(1) = t_inv*f(0)                                 !
              f0   = sqrt(pi_over2*t_inv)
              f1   = t_inv*f0
            end if
            !-----------------------------------------------------
            !  ERI code generator Ver.20020228
            !  2002/02/28
            !  T. Nakano
            !
            !  (sssp)
            !
            ssss(0:1)=f(0:1)*a0                                 !
            ssss(0)=f0*a0
            ssss(1)=f1*a0
            do l=1, 3                                           !
              sint(l) = sint(l)+qd(l)*ssss(0)+wq(l)*ssss(1)     !
            end do                                              !
            sint(1) = sint(1)+qd1*ssss(0)+wq1*ssss(1)
            sint(2) = sint(2)+qd2*ssss(0)+wq2*ssss(1)
            sint(3) = sint(3)+qd3*ssss(0)+wq3*ssss(1)
            !-----------------------------------------------------
          end if
        end do
      end if
    end do
  end subroutine sub_sssp
```

図 3. コードを改良した 2 電子積分生成ルーチン（ss|sp）

て(ss|sp)のソースを貼りました。Ver. 2 Rev. 4 への高速化改良では、この他にダイマー計算を静電近似（Dimer-ES）するステップの改良、プリント量のコントロール等も行っています。

「不老」 Type Ⅰ上（lang/tcsds-1.2.31 環境）での FMO-MP2 ジョブのタイミングですが、Ala_9Gly の 6-31G*基底では従前の Ver. 1 Rev. 22 の 156.6 秒に対し、Ver. 2 Rev. 4 では 116.4 秒となり、加速としては 1.4 倍が得られました。Chignolin、Crambin など他のタンパク質でもテストしてみましたが、加速としては 1.2 倍～1.5 倍となりました。また、基底関数を 6-31G*よりも短縮が長い cc-pVDZ とすると、加速が向上する傾向も見られました。これは、SIMD 化がより有効に働くようになるためと思われます。

大規模系でのテストとして、PDB-ID=6VXX の新型コロナウイルスのスパイクタンパク質について、「富岳」で Ver. 1 Rev. 22 と Ver. 2 Rev. 4 の比較をしてみました。アミノ酸残基数は 3.3 千になりますので、FMO-MP2/6-31G*ジョブで 8 ラック（3072 ノード）を使いました（48 スレッド、ノードあたり 1 プロセス）。ABINIT-MP のプリント末尾のタイミングリストを図 4 に示します。Ver. 1 Rev. 22 に比べて Ver. 2 Rev. 4 では 1.2 倍速くなっていますが、モノマーSCC に対応する"Monomer SCF"の時間が半分以上の時間を占めていることに気付きます。この状況は、FMO-MP2 計算でアミノ酸残基の数が千を超えてくると顕在化しますので、SCC 反復回数を削減する工夫も必要です。もう一点、Ver.

Ver. 1 Rev. 22

```
====================
   ## TIME PROFILE
====================

      Elapsed time: Monomer SCF     =         2028.7 seconds
      Elapsed time: Monomer MP2     =           15.0 seconds
      Elapsed time: Monomer (Total) =         2068.6 seconds
      Elapsed time: Dimer ES        =          353.9 seconds
      Elapsed time: Dimer SCF       =          362.4 seconds
      Elapsed time: Dimer MP2       =          302.6 seconds
      Elapsed time: Dimer (Total)   =         1603.4 seconds
      Elapsed time: FMO (Total)     =         3672.1 seconds

## Time profile

Number of cores (total)    =   3072
Number of cores (fragment) =      1

THREADS (FRAGMENT)         =     48


Total time =          3759.3 seconds
```

Ver. 2 Rev. 4

```
====================
   ## TIME PROFILE
====================

      Elapsed time: Monomer SCF     =         1801.6 seconds
      Elapsed time: Monomer MP2     =           14.2 seconds
      Elapsed time: Monomer (Total) =         1839.1 seconds
      Elapsed time: Dimer ES        =          314.2 seconds
      Elapsed time: Dimer SCF       =          335.7 seconds
      Elapsed time: Dimer MP2       =          294.6 seconds
      Elapsed time: Dimer (Total)   =         1188.5 seconds
      Elapsed time: FMO (Total)     =         3027.7 seconds

## Time profile

Number of cores (total)    =   3072
Number of cores (fragment) =      1

THREADS (FRAGMENT)         =     48


Total time =          3090.8 seconds
```

図 4. 「富岳」での PDB-ID=6VXX の FMO-MP2/6-31G*ジョブのタイミング比較

2 Rev. 4 では Ver. 1 Rev. 22 に比してダイマーの個々のステップと総計時間の「差」が小さくなっていますが、これは結果集計の通信量の若干の低減とプリント量のコントロールが奏功しています。

次に、(2)の超大規模系への対応についてです。BioStation Viewer の可視化結果解析では、チェックポイントファイル（CPF）というテキストフォーマットで ABINIT-MP から書き出される情報ファイルが必要です。データとしては、IFIE をはじめとするフラグメントの対（自乗性の大きさ）のデータが並びます。テキストとしているのは、エンディアン問題を回避するためですが、対象系が大きくなるとファイル容量が急増します。実のところ、前出のスパイクタンパク質 6VXX の場合、BioStation Viewer での CPF 読み込みはメモリが 64GB あるカスタム PC でなければ不可でした。MD-FMO 連携による多サンプル構造計算も考えれば、前述のように可視化解析に固執することはデメリットが大きいです。そこで、BioStation Viewer 用の配列を削除し、FMO 計算実行時のメモリ要求量を削減する決断をし、Ver. 2 系では可視化解析を放棄しました。

Ver. 2 Rev. 4 の段階では、このメモリ量の削減は第一段階ですが、インフルエンザウイルスのヘマグルチニン（HA）と Fab 抗体 2 個の複合体、PDB-ID=1KEN（アミノ酸残基の数は 2.2 千）の水和モデルで総数 1.1 万フラグメントの系が計算可能と可能となりました。図 5 に、「不老」 Type I の 1 ラックを使って Ver. 2 Rev. 4 で実行した FMO-MP2/cc-pVDZ ジョブのタイミングを示します。cc-pVDZ の方が 6-31G*よりもメモリを要し、限界テストをするには好適で、9.2 時間で完走しています。ただ、ここで気になるのは、既に触れたダイマーの総計時間の「差」が非常に大きくなっている点で、集計の通信に改善が必要なことを示しています。3 次の摂動（MP3）計算

```
====================
   ## TIME PROFILE
====================

      Elapsed time: Monomer SCF     =        14546.6 seconds
      Elapsed time: Monomer MP2     =           32.5 seconds
      Elapsed time: Monomer (Total) =        14741.5 seconds
      Elapsed time: Dimer ES        =         4021.8 seconds
      Elapsed time: Dimer SCF       =         7215.9 seconds
      Elapsed time: Dimer MP2       =         2492.4 seconds
      Elapsed time: Dimer (Total)   =        18240.6 seconds
      Elapsed time: FMO (Total)     =        32982.1 seconds

## Time profile

Number of cores (total)    =    384
Number of cores (fragment) =      1

THREADS (FRAGMENT)         =     48

Total time =         33120.9 seconds
```

図 5. 「不老」 Type I での PDB-ID=1KEN の水和モデルの FMO-MP2/cc-pVDZ ジョブのタイミング

表 1. ABINIT-MP の HPCI 拠点での整備状況（2021 年 12 月）

拠点	CPU type	V1 R22	V2 R4
北大	Xeon	○	○
東北大	SX-AT, Ryzen	○	
JCAHPC	Xeon Phi	○	
東大	A64FX, Xeon	○	○
東工大	Xeon	○	○
JAMSTEC	SX-AT、Ryzen	○	○
分子研	Xeon	○	○
名大	A64FX	○	○
阪大	SX-AT	○	
理研CCS	A64FX	○	○
FOCUS	Xeon	○	○
九大	Xeon	○	○

では、テンソル縮約のオーダーが 4 から 6 に上がってメモリ要求も増えますが、この 1KEN 水和モデルが MP3 でも実行可能となりました。「富岳」の 8 ラック利用では、FMO-MP3/cc-pVDZ ジョブが 6.7 時間で完走しました。MP3 が出来ると MP2.5 スケーリングにより、精度の高い CCSD(T)なみの算定が可能となるので実行が確保できた意味は大きいです。

Ver.2 Rev. 4、それに併存とした Ver. 1 Rev. 2 の HPCI 拠点での 2021 年 12 月時点でのライブラリ整備状況を表 1 に示します（拠点、CPU タイプ、版名は略記）。ほぼ全国をカバーしており、一般利用者の利便が図られていることを分かっていただけるはずです。なお、Ver. 1 Rev. 22 の SX-Aurora TSUBASA（SX-AT）のベクトル化対応版は NEC との別プロジェクトの成果でもあります。

さて、ここで Ver. 2 Rev. 4 のリリース後の改良について簡単に紹介します。(1)については、HF 計算での Fock 行列構築のアルゴリズム変更を行いました。図 6 に、当該部分の擬似コードを示しますが、添字の同値性を $(1/2)^n$ (n=1,2,3)で考慮して if 分岐を排したループへの変更により、HF 部分で 30%の高速化が得られています。Ala_9Gly の FMO-

```
do p=ixi1,ixi2
  do q=ixj1,ixj2
    do r=ixk1,ixk2
      do s=ixl1,ixl2
        ix=ix+1
        val = sint(ix)
        if((abs(val) <= tv)) cycle
        fock(q,p)=fock(q,p)+dc(s,r)*val*2.d0! クーロン項
        fock(s,r)=fock(s,r)+dc(q,p)*val*2.d0
        fock(r,p)=fock(r,p)-dc(s,q)*val*0.5d0! 交換項
        fock(s,p)=fock(s,p)-dc(r,q)*val*0.5d0
        fock(r,q)=fock(r,q)-dc(s,p)*val*0.5d0
        fock(s,q)=fock(s,q)-dc(r,p)*val*0.5d0
      end do
    end do
  end do
end do
```

図 6. Fock 行列構築の擬似コード

MP2/6-31G*ジョブでは、Ver. 2 Rev. 4 の 116.4 秒が 104.3 秒に短縮されました。さらに、レジスタスピルの低減のためにループ分割を幾つかの積分生成ルーチンに実施したところ 95.8 秒になり、Ver. 1 Rev. 22 に対して 1.6 倍の加速となりました。加えて、モノマーSCC の反復回数の低減を意図し、アンダーソン外挿法のオプションを追加したところ、時間は 91.1 秒、加速は 1.7 倍に達しました。他の系でも Ver. 2 Rev. 4 からの有意な改善が得られています。2022 年度、継続して高速化と性能評価を進めていきます。

(2)については、水のみのクラスターですが、2 万フラグメントを超えた MP2 計算も実行できています。2022 年度の作業では、タンパク質水和系で 2 万を超えたいところです。

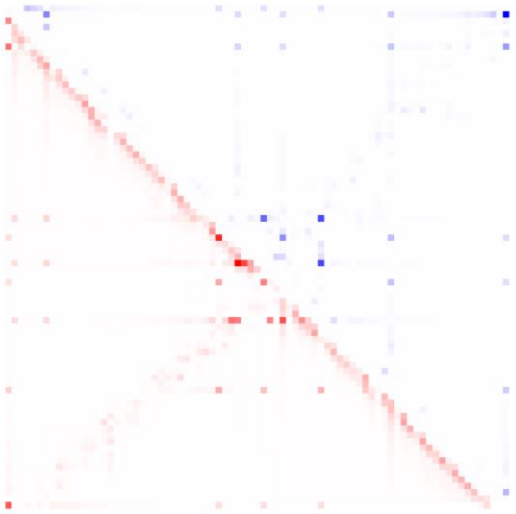

図 7. IFIE-map の例（α-ヘリックス）

最後に、(3)のタンパク質セットの IFIE-map の TensorFlow を使った深層学習について報告します。図 7 に、PDB-ID＝1A91 のα-ヘリックスを持つタンパク質のアミノ酸残基間の IFIE-map を例示します（下 3 角の赤が安定化、上 3 角の青が不安定化に対応）。α-ヘリックスの場合、対角線付近に有意なピクセルが集中する傾向があります。一方、β-シートでは、安定化領域で逆対角線方向のピクセルが目立ちます(図示は略)。これらの画像を TensorFlow に学習させ、画像だけからα-ヘリックスとβ-シートの存在を判定させます。以前の報告（Saitou et al., ChemBio-Informatics J. 18 (2018) 58-69）では、PC ベースでの解析であったため、画像のダウンコンバージョンをせざるを得ませんでした。今回は、「不老」 Type II 上に TensorFlow の環境を整え、IFIE-map の画像をそのまま用いて学習させました。元の IFIE データは前報のものを踏襲しています。ここでは、GPU による加速の結果を記しますが、Xeon のみの場合に比べて GPU を利用すると 11 倍高速に処理できました。今回は予備的なレベルでの検証ですが、今後、大規模な MD-FMO 連携シミュレーションを実施していく際、IFIE-map に対する深層学習による解析も可能であることが確認できたと考えています。

6. 今年度の進捗状況と今後の展望

(1)の A64FX での高速化については、リリースした Ver. 2 Rev. 4 では従前の Ver. 1 Rev. 22 に比して FMO-MP2 ジョブで 1.2～1.5 倍の加速を得ており、その後の追加改良版では 2 倍近い加速となるケースもあります。次に、(2)の超大規模系への対応については、総数 1.1 万フラグメントの FMO-MP2 だけでなく FMO-MP3（「富岳」でテスト）が実行可能となりました。(1)と(2)、さらに HPCI 拠点でのライブラリ整備まで含めた総合的な進捗の自己評価は 90 点になります。また、(3)の IFIE-map の TensorFlow による深層学習については試金石的な意味合いですが、GPU での加速を実証できたので、こちらも 90 点とします。

本課題は 2022 年度も jh220010 として継続しており、ABINIT-MP 関係では引き続き、プラットフォームを「不老」 Type I として(1)と(2)を進めていきます。目標は、(1)の加速が 2 倍超え、(2)の対象系の拡大がタンパク質で 2 万フラグメント超え、としています。

(1)では、通信量の削減とモノマーSCC の加速をさらに進めます。また、A64FX 上での FMO-MD の実行を意図し、SIMD 化とループ分割の改造を核座標によるエネルギー微分（力）の計算も対象とする予定です。さらに余力があれば、抜本的な改善策として積分ルーチンの生成プログラムのロジックの変更も検討します。(2)では、メモリの削減を続けますが、作業はより慎重に行う必要があります。

jh220010 課題では、ABINIT-MP とは別に、二つの要素を含めました。一つは、DeepMind 社が開発・公開したタンパク質の構造予測システム AlphaFold2 の整備と活用です。2021 年度下期に「不老」 Type II 上で AlphaFold2 は利用可能となっていますが、タンパク質複合体の予測機能の向上などが 2022 年度も DeepMind で進められるはずですので、応じての更新作業が必要です。アミノ酸残基のシークエンスのみからタンパク質の構造が構築できれば、FMO 計算の対象とできる系も広がります。もう一つは、NVIDIA の量子コンピュータのシミュレータである cuQuantum の利用で、やはり Type II がプラットフォームになります。具体的には、ユニタリー結合クラスター(UCCSD)系の量子化学計算の実施を想定していますが、試行錯誤的な要素が多い試みになります。既に、FMO のような「分割して統合」のアプローチが量子化学計算のための量子回路の深度を浅くする点で注目を集めており、FMO 計算の「将来展開」を見据えての試みとも言えます。

7. 研究業績一覧

（発表予定も含む．投稿中・投稿予定は含まない）

(1) 学術論文 （査読あり）

[1] "FMO プログラム ABINIT-MP の整備状況 2021", 望月祐志*, 中野達也, 佐藤伸哉 , 坂倉耕太, 渡邊啓正, 奥脇弘次, 大島聡史, 片桐孝洋, J. Comp. Chem. Jpn. 20 (2021) 132-136.

(2) 国際会議プロシーディングス （査読あり）

該当ありません。

(3) 国際会議発表 （査読なし）

該当ありません。

(4) 国内会議発表 （査読なし）

[1] "FMO プログラム ABINIT-MP の高速化と大規模系への対応について"（オンラインポスター） 望月祐志*, 中野達也, 佐藤伸哉 , 坂倉耕太, 渡邊啓正, 奥脇弘次, 大島聡史, 片桐孝洋, 分子科学討論会 2021, 札幌, 2021/9/20.

[2] "FMO プログラム ABINIT-MP の整備状況 2021" （オンライン口頭） 望月祐志*, 中野達也, 佐藤伸哉 , 坂倉耕太, 渡邊啓正, 奥脇弘次, 大島聡史, 片桐孝洋, 日本コンピュータ化学会 2021 年秋季年会, つくば, 2021/11/3.

[3] "A64FX を用いたフラグメント分子軌道計算プログラムの性能評価"（オンライン口頭） 満田晴紀*, 片桐孝洋, 坂倉耕太, 望月祐志, 大島聡史, 永井亨, 第 84 回情報処理学会, 松山, 2022/3/4（学生奨励賞受賞：満田氏）.

(5) 公開したライブラリなど

ABINIT-MP Version 2 Revision 4 （2021 年 9 月） http://www.cenav.org/abinit-mp-open_ver-2-rev-4/

導入拠点リスト｛北海道大学, 東京大学, 東京工業大学, 海洋研究開発機構, 分子科学研究所, 名古屋大学, 理化学研究所-計算科学研究センター, （公財）計算科学振興財団, 九州大学｝.

(6) その他（特許, プレスリリース, 著書等）

[1] "FMO プログラム ABINIT-MP の A64FX スーパーコンピュータ向け高速化と大規模化", 望月祐志, 中野達也, 坂倉耕太, 計算工学ナビ Vol. 21 (2021 年秋号) 6, http://www.cenav.org/nldl/ からダウンロード可能.