[高速化支援]

第 1 原理電子構造計算コード VASP の SX-ACE 向け最適化

山下 毅 [†1], 長谷川 正之 [2], 青山修也 [2], 千葉貴則 [2], and 西館 数芽 [‡2]

[1] 東北大学　情報部情報基盤課
[2] 岩手大学　理工学部　システム創成工学科電気電子通信コース

概要

パラメータを用いずに物質の電子構造を調べる第 1 原理電子構造計算法は、物性物理学の分野における非常に強力な研究手法であり、最近では理論研究者のみならず広く実験研究者にも使われるようになってきた。本稿では密度汎関数法に基礎をおく第 1 原理電子構造計算コード VASP のスーパーコンピュータ SX-ACE 向けの最適化と、並列コンピュータ LX 406Re-2 での実行結果の比較について述べる。

1　はじめに

物性理論の精密化とスーパーコンピュータの演算性能の飛躍的向上に伴い、密度汎関数法（Density Function Theory, DFT）[1] に基礎をおく第 1 原理電子構造計算法は物質科学の分野において頻繁に使用される研究ツールとなってきた。この方法は物質の物理的・化学的特性を調べるための非常に強力な手段を提供してくれる。通常の意味での経験的パラメータを用いないため、たとえば経験的理論が未だ存在していない特殊な環境下における物質の物理的・化学的特性をも予測できる。現在では、WIEN2k、CASTEP、VASP[2] などのさまざまな先進的な DFT コードが入手可能である。Vienna Ab-initio Simulation Package (VASP) は、平面波を基底とする DFT の枠組みにおける第 1 原理電子構造計算コードである [3, 4, 5]。原子の内殻は、PAW 法（Projector augmented-wave method）で記述し、DFT ハミルトニアンの対角化に効率的なアルゴリズムが採用されている。ここでは物質科学の分野でユーザーが非常に多い第 1 原理計算コード VASP に焦点をあてる。

高性能科学技術計算の追求は、近年の並列計算機の隆盛をもたらした。現在用いられている DFT コードの大部分が、多数の計算ノードから構成される大規模並列計算機での実行を前提にプログラムされている。しかしながらメモリの閾値や大規模な並列化におけるオーバーヘッドのために、単純に計算ノード数を増加させてもそれに比例した速度向上を得ることはできない。この問題に対する一つの解は、各計算ノードに搭載されているプロセッサあたりの性能を高めることである [6, 7]。

SX-ACE は NEC 製のベクトル型スーパーコンピュータである [8, 9]。4 コアを有するベクトルプロセッサは、コアの演算性能とメモリバンド幅のバランスの取れたシステムとして設計されている。ただし、その高い演算性能を引き出すためには、コンパイラの最適化を促進させるためのコード修正が必要で

† yamacta@tohoku.ac.jp
‡ nisidate@iwate-u.ac.jp

ある。本稿では VASP.5.4.1（vasp.5.4.1.24Jun15.tar.gz）を対象とし、SX-ACE 環境への移植作業および、SX-ACE 向けコンパイラに対応したコード修正、およびコードのベクトル高速化について述べる。また、FFT ライブラリは付属の Furth ライブラリのソースコードをコンパイルして使用した。

2　SX-ACE 用コンパイラ向け移植作業とベクトル高速化

2.1　SX-ACE 用コンパイラの仕様

SX-ACE 用のコンパイラとして、NEC 製の FORTRAN90/SX（ISO/IEC 1539-1:1997 準拠の Fortran90/95 コンパイラ）、NEC Fortran2003（ISC/IEC 1539-1:2004 準拠の Fortran2003 コンパイラ）、および、C++/SX（ISO/IEC 1539-1:2004 準拠の C,C++ コンパイラ）が利用可能である。

今回移植の対象とした VASP.5.4.1 は一部 Fortran2003 規格を使用したソースコードがあるが、現時点では FORTRAN90/SX コンパイラの自動ベクトル化/最適化機能が Fortran2003 コンパイラよりも優れているため、本稿では前者のコンパイラの利用を仮定する。Fortran2003 機能を利用するためのオプションを指示することにより、FORTRAN90/SX（rev.534）コンパイラでも VASP.5.4.1 をコンパイルすることが可能である。また、C,C++ コンパイラとして C++/SX（rev.112）コンパイラを使用する。

2.2　FORTRAN90/SX 向けのコード修正

2.2.1　変数定義部の修正

FORTRAN90/SX コンパイラでは、変数の定義部で組込関数を利用できないため、リスト 1 に示す 3 つのソースコードについて、変数の定義部では型宣言だけを行い、実行部で値を代入するように修正を行った。

リスト 1　変数定義部

```
[src/vdwforcefield.F]
（6047行目）定義部で組込関数のCMPLXを使用できない
  COMPLEX(q) :: zdumm0=CMPLX(0._q),zdumm1=CMPLX(1._q),zdummM1=CMPLX(-1._q)

（7824行目）定義部で組込関数のSQRTを使用できない
  REAL(q),PARAMETER :: two_over_sqrtPI=2._q/SQRT(PI),four_over_sqrtPI=2*two_over_sqrtPI

[src/bse_te.F]
（137行目）定義部で組込関数のREALを使用できない
  REAL(mq),PRIVATE, PARAMETER :: HARTREE  = 2._mq*REAL(Rytoev,mq)

[/src/dmatrix.F]
（1321行目）定義部で組込関数のSQRTを使用できない
  REAL(q), PARAMETER :: SQPi54=4._q*SQRT(PI/5._q),SQPi52=2._q*SQRT(PI/5._q),
                        mSQPi52=-2._q*SQRT(PI/5._q),SQPi15=6._q*SQRT(PI/15),
                        mSQPi15=-6._q*SQRT(PI/15)
  関連してDATA文で定義される配列XIXJ2YLMも実行部で値を代入する
```

2.2.2　WRITE 文の修正

FORTRAN90/SX コンパイラの仕様に基づき、リスト 2 に示す 2 つのソースコードについて、WRITE 文中で/=を使用する行のコメントアウトと、WRITE 文の直後のカンマを削除した。

リスト 2　WRITE 文

```
[/src/gridq.F]
（337行目）WRITE文中で/=を使用できない
  WRITE(0,*) GRID%MPLWV /= SIZE(C,1)

[src/minimax.F]
（182行目）WRITE文の直後のカンマ（,）が文法エラー
  IF ( INU >= 0 ) WRITE(*,'(" Number of grid points forced to ",I4)'),NNU
```

2.2.3　変数の二重定義の修正

変数が二重定義されており、FORTRAN90/SX コンパイラではコンパイル時にエラーとなるため、定義部分の片方をコメントアウトした。

リスト 3　変数の二重定義

```
[src/base.F]
（169行目から190行目）dipolが2重定義でエラーになるので、定義部分をコメントアウト
```

2.2.4　外部関数呼出しの修正

関数 volume を呼び出しているが、他ソースコードで定義されていないためコンパイル時にエラーとなる。実際の利用はないようなので呼び出し部分をコメントアウトした。

リスト 4　外部関数呼出

```
[vasp.5.4.1/src/subdftd3.F]
（724行目）外部関数 volume をリンクできない
  実際の利用はないようなので、コメントアウト
```

2.3　FORTRAN90/SX 向け高速化

2.3.1　関数 rane のインライン展開

SX-ACE 向けの簡易性能解析情報ツールである、FTRACE を用いたプロファイリングにより、関数 rane が多数回呼び出されおり、この呼び出しオーバーヘッドのコスト割合が高いことが分かった。このオーバヘッドを削減するために、関数 rane のインライン展開を試みた。

インライン展開を行うにはコンパイルオプションに-pi を指定することで同一ファイル内のインライン展開が行われる。複数ファイル間のインライン展開を行うには -pi のサブオプション expin に対象ソースコード名もしくは対象ソースコードの保存されたディレクトリ名を指定する。-pi オプションの記述内容についてはリスト 8 に示す。

関数 rane ではインライン展開を阻害する SAVE 属性の利用があったため、変数を新たな module を定義して変数の値を保持させることに変更した。リスト 5 に関数 rane のインライン展開を促進するために修正を行ったソースコードを示す。呼び出し側ではこの module を利用する宣言と変数の初期化を行うことで、インライン展開が可能となり、呼び出しオーバヘッドの削減を行うことが出来た。

リスト 5　インライン展開による高速化

```
[vasp.5.4.1/src/random.F]
  関数raneの定義において、インライン展開を阻害するSAVE属性の利用をmoduleに変更する。
  module icallmを設定し、icallの値を保持させる。
  同じくインライン展開を阻害するDATA文を代入文に変更する。

[vasp.5.4.1/src/main.F]
  icallの値を保持するために、use icallmの記述とicall=0で初期化。
```

2.3.2　コンパイラ指示行によるベクトル高速化

ソースコード中の直後の DO ループに依存性がないことを示す、インテルコンパイラ向けの指示行（!DIR$ IVDEP）が記述されている箇所に、FORTRAN90/SX コンパイラ向けの同種の指示行（!CDIR NODEP）を挿入することで、コンパイラによる DO ループのベクトル化が行われるようにした（リスト 6）。

リスト 6　コンパイル指示行によるベクトル化

```
[vasp.5.4.1/src/fft3dlib.F]
  !DIR$ IVDEPが指定されているループ箇所（57箇所）に
  !CDIR NODEPを記述してベクトル化する。
```

2.4　SX-ACE 向け Makefile

VASP に付属の Makefile の動作は、.F90 の拡張子を持つソースコードからプリプロセッサがオプションに従って.f90 のソースコードを生成し、このソースコードをコンパイルする、という流れでオブジェクトファイルを生成する。

SX-ACE 向け高速化では、ファイル間のインライン展開を行うことで、関数・サブルーチン呼び出しの部分が DO ループ内で展開され、ベクトル化が促進されることがある。そのため make 時はコンパイル時間を短縮するために、最適化オプションを最低レベルの-C ssafe として.f90 のソースコードを生成した後に、リスト 8 に示すスクリプトファイルで相互ファイル間のインライン展開の指定と、最大の最適化オプション-C hopt を指定してコンパイルとリンクを行った。

リスト 7　SX-ACE 向け makefile.include と src/makefile

```
[vasp.5.4.1/makfile.inclued]
CPP_OPTIONS=-DMPI -DHOST=\"NECSX\" -DMPI -DPGF90 -Davoidalloc -DscaLAPACK -Duse_collective\
            -USX -Dpro_loop -DMPI_BLOCK=80000
CPP         = sxcpp -P $(FUFFIX)$(SUFFIX)$(CPP_OPTIONS)
FC          = sxmpif90
FCL         = sxmpif90
FFLAGS      = -f5 -f2003
OFLAG       = -Cssafe
BLAS        = -lblas
LAPACK      = -llapack
BLACS       = -lblacsF90init -lblacs -lblacsF90init
SCALAPACK   = -lscalapack -lblacsF90init -lblacs -lblacsF90init -llapack -lblas
OBJECTS     = fft3dfurth.o fftmpi.o fftmpi_map.o fft3dlib.o
OBJECTS_01 += fft3dfurth.o fftmpi.o
OBJECTS_02 += fft3dlib.o
CC_LIB      = sxcc
CFLAGS_LIB =
FFLAGS_LIB = -Cvopt

[vasp.5.4.1/src/makefile] sxmpif90 の最適化オプションと拡張自由形式のオプションに変更
OFLAG_1=-Cssafe -f5
OFLAG_2=-Cssafe -f5
OFLAG_3=-Cssafe -f5
OFLAG_4=
```

fft3dlib.f90 と vdw_nl.f90 については、インライン展開のネスト数が 3 を超えた場合、コンパイラの最適化に要するメモリ不足により、コンパイル途中で処理が中断した。また force.f90 と fock.f90 については最適化オプションのレベルによって計算結果が異なることが分かった。このため、これらのファイルに対しては他のファイルとは異なるオプションを指定してコンパイルを行っている。

リスト 8　sxmake.sh

```
(vasp_stdをコンパイル・リンク)
cd ./build/std
rm -f *.o *.F vasp
sxmpif90 -Chopt -f5 -f2003 -c -pi nest=5 line=1000 expin=./ ./*.f90
sxmpif90 -Chopt -f5 -f2003 -c -pi nest=3 line=1000 expin=./ fft3dlib.f90
sxmpif90 -Chopt -f5 -f2003 -c -pi nest=3 line=1000 expin=./ vdw_nl.f90
sxmpif90 -f5 -f2003 -c -pi nest=5 line=1000 expin=./ force.f90
sxmpif90 -f5 -f2003 -c -pi nest=5 line=1000 expin=./ fock.f90

sxmpif90 -o vasp base.o mpi.o smart_allocate.o xml.o constant.o jacobi.o main_mpi.o ... \
   (その他必要なオブジェクトファイル)
... fft3dfurth.o fftmpi.o fftmpi_map.o fft3dlib.o main.o \
 -Llib -ldmy -lscalapack -lblacsF90init -lblacs -lblacsF90init -llapack -lblas -llapack -lblas

cp ./vasp ../../bin/vasp_std
```

3　計算結果の比較

3.1　対象としたベンチマークファイル

ベンチマークファイルとして、スウェーデンのリンショーピン大学（Linköping University）国立スーパーコンピューティングセンターのペーター博士（Peter Larsson, PhD）が、Cray XC-40（通称 Beskow）において VASP のベンチマーク用に設定した系を用いる。[*1] 対象として GaAsBi-512 と呼ばれる系を使用した。これは、GaAs（格子定数 5.6537 Å）の 1 つの As のみを Bi で置換したものである。この系を表現するインプットファイルを用いて、SX-ACE と当センターの並列コンピュータ LX 406Re-2（Intel®Xeon E5-2695v2）との計算結果を比較した。LX 406Re-2 向けのコンパイルに使用した makefile.include は vasp.5.4.1.24Jun15.tar.gz に同梱のものを使用し、Intel®コンパイラおよび MKL ライブラリ（インテル®Parallel Studio XE 2016）を用いてコンパイルとリンクを行った。

インプットファイルの 1 つである、計算条件のキーワードを指定するファイル、INCAR ファイルの共通パラメータは以下の通りである。

リスト 9　INCAR ファイルの共通パラメータ

```
SYSTEM=GaAs    ! 系の名前
ISTART=0       ! 始めから計算
ICHARG=2       ! 原子位置から電荷密度を推測
PREC=Accurate  ! 計算精度の設定
ENCUT=313      ! カットオフエネルギー [eV]
ISPIN=1        ! スピンは考慮しない
ISMEAR=-5
ALGO=fast
LCHARG=.FALSE.
LWAVE=.FALSE.
LREAL=Auto
NELMIN=1
NELM=20
EDIFF=1E-4
NBANDS=1536
```

3.1.1　SX-ACE での計算結果

SX-ACE は 1 ノードあたり 1CPU と 64GB の主記憶を搭載し、1CPU は 4 コアで構成される。1 ノードの理論演算性能は 276GFLPOS、メモリバンド幅は 256GB/sec である。

SX-ACE での実行時に INCAR に記載した並列化パラメータは以下の通りである。1 ノードあたり 4 プロセス、32 ノードを用いて合計 128 プロセスで実行を行った。

[*1] https://www.nsc.liu.se/~pla/blog/2015/01/13/vaspstudy-crayxc40/

リスト 10　SX-ACE での並列化に関するパラメータ

```
NCORE=4
KPAR=4
NSIM=1
NPAR=32
```

SX-ACE の実行時にはゼロ割演算の警告を抑制するために、リスト 11 に示す環境変数をバッチリクエストのスクリプトファイル内に追記した。

リスト 11　実行時環境変数

```
#PBS -v F_ERROPT1="252,253,0,0,2,2,2,2"
```

リスト 12　SX-ACE での実行結果（OUTCAR ファイルの抜粋）

```
  total drift:                                -0.000019     -0.000019      0.000000
 -----------------------------------------------------


 FREE ENERGIE OF THE ION-ELECTRON SYSTEM (eV)
 ---------------------------------------------------
 free  energy   TOTEN  =     -2110.33769038 eV

 energy  without entropy=    -2110.33769038  energy(sigma->0) =    -2110.33769038

               Total CPU time used (sec):      930.070
                         User time (sec):      929.570
                       System time (sec):        0.500
                      Elapsed time (sec):      933.133
                      (イタレーション回数は20回)
```

3.1.2　LX 406Re-2 での計算結果

LX 406Re-2 は 1 ノードあたり 2CPU と 128GB の主記憶を搭載し、1CPU は 12 コアで構成される。1 ノードの理論演算性能は 460.8GFLOPS、メモリバンド幅は 119.4GB/sec である。

LX 406Re-2 での実行時に INCAR に記載した並列化パラメータは内容は以下の通りである。1 ノードあたり 24 プロセス、8 ノードを用いて合計 192 プロセスで実行を行った。

リスト 13　LX 406Re-2 での並列化に関するパラメータ

```
NCORE=24
KPAR=4
NSIM=24
NPAR=2
```

リスト 14　LX406 での実行結果（OUTCAR ファイルの抜粋）

```
  total drift:                                -0.000019     -0.000019     -0.000000
 -----------------------------------------------------


 FREE ENERGIE OF THE ION-ELECTRON SYSTEM (eV)
 ---------------------------------------------------
 free  energy   TOTEN  =     -2110.32291858 eV

 energy  without entropy=    -2110.32291858  energy(sigma->0) =    -2110.32291858

               Total CPU time used (sec):     1003.195
                         User time (sec):     1001.277
                       System time (sec):        1.919
                      Elapsed time (sec):     1010.096
                      (イタレーション回数は13回)
```

SX-ACE と LX 406Re-2 でそれぞれ実行した結果、イタレーション回数はそれぞれ 20 回、13 回と差異があったが、最終的な系のエネルギーの値は小数点第一位まで一致する結果が得られた。

4 VASP 実行オブジェクトの提供

サイバーサイエンスセンターでは VASP のソフトウェアライセンスを保有する利用者に、VASP.5.4.1 の SX-ACE 向けバイナリと LX 406Re-2 向けバイナリを提供しています。利用方法については共同利用支援係（連絡先は本誌の表紙内側を参照）までお問合せください。

5 まとめ

本稿では VASP.5.4.1 の SX-ACE 向け移植方法と高速化について述べた。当センターでは SX-ACE は最大で 1,024 ノード（2,048 プロセス並列）で、60TB メモリを利用した実行が可能である。SX-ACE で大規模な分子モデルの解析を行っていただき、利用者の研究ツールとしてご活用いただければ幸いである。

参考文献

[1] P. Hohenberg and W. Kohn. *Phys. Rev.*, 136:B846, 1964.

[2] Belgium Center for Molecular Modeling, Ghent University. *Comparing Solid State DFT Codes, Basis Sets and Potentials*. `https://molmod.ugent.be/deltacodesdft`, 2016.

[3] G. Kresse and J. Hafner. *Phys. Rev. B*, 47:558, 1993.

[4] G. Kresse and J. Furthmuller. *Phys. Rev. B*, 54:11169, 1996.

[5] Jürgen Hafner. *Journal of Computational Chemistry*, 29:2044, 2008.

[6] R. Egawa, S. Momose, K. Komatsu, Y. Isobe, H. Takizawa, A. Musa, and H. Kobayashi. Early evaluation of the SX-ACE processor. SC14: The international conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA., 2014. The extended abstract can be found here: `http://sc14.supercomputing.org/sites/all/themes/sc14/files/archive/tech_poster/poster_files/post196s2-file3.pdf`.

[7] S. Momose, T. Hagiwara, Y. Isobe, and H. Takahara. The brand-new vector supercomputer, SX-ACE. In *Proceedings of 29th International Supercomputing Conference (ISC'14)*, pages 199–214. Springer, 2014.

[8] 萩原孝 浜口博幸 山信田恒. スーパーコンピュータシステム SX-ACE のハードウェア. *SENAC*, 48-1:5–14, 2015.

[9] 山下毅 森谷友映 佐々木大輔 齋藤敦子 小野敏 大泉健治 岡部公起 江川隆輔 小林広明. スーパーコンピュータシステム SX-ACE の紹介. *SENAC*, 48-1:39–46, 2015.