Wisteria/BDEC-01（Odyssey）における OpenMP によるプログラミング入門（その 1）

中島研吾

東京大学情報基盤センター

0. はじめに

2021 年 5 月 14 日に運用を開始した「計算・データ・学習」融合スーパーコンピュータシステム（Wisteria/BDEC-01）［1,2］のうち，シミュレーションノード群（Odyssey）における OpenMP プログラミングについて 2 回（もしかしたら 3 回）に分けて連載する．Odyssey は FUJITSU Supercomputer PRIMEHPC FX1000」20 ラックから構成され，「A64FX」を 7,680 ノード（368,640 コア）搭載する。各ノードは倍精度浮動小数点演算で 3.3792 TFLOPS の理論ピーク性能，合計ピーク性能は 25.9 PFLOPS である．各ノードは 32 GiB の HBM2 メモリを搭載し，シミュレーションノード群（Odyssey）の総メモリ容量は 240 TiB，総メモリバンド幅は 7.8 PB／秒である．各ノードはバイセクションバンド幅が 13.0 TB／秒のノード間相互結合ネットワーク（Tofu インターコネクト D）で結合されている．本連載では，A64FX の各ノード（48 コア搭載）における OpenMP 並列プログラミングについて紹介する．

紙面の都合もあるため，詳細は当センターで実施している講習会「OpenMP によるマルチコア・メニィコア並列プログラミング入門（Wisteria/BEDC-01（Odyssey，A64FX 搭載））[1,2]」のホームページ［3,4］をご覧頂きたい．講習会の詳細な資料の他，ビデオ動画も公開されている．Wisteria/BDEC-01 の登録ユーザーは講習会資料で紹介しているサンプルプログラムを自由にコピーして学習することが可能である．

第 1 回は対象とするアプリケーションの概要と，その OpenMP 並列化について紹介する．

1. 本稿の目的

有限要素法，差分法等の科学技術アプリケーションは最終的には大規模な疎行列を係数とする連立一次方程式を解くことに帰着される．安定で効率的な解法の研究開発は重要な技術的課題であり，特に昨今では大規模並列計算をターゲットとした前処理付き反復法に関する研究が盛んである〔5〕．大規模なマルチコアクラスタ，メニィコアクラスタではノード間のメッセージパッシング（例：MPI），ノード内のスレッド並列（例：OpenMP）に基づくハイブリッド並列プログラミングモデルが広く使用されている〔5〕．Odyssey でも，大規模並列計算実行にあたっては，OpenMP/MPI によるハイブリッド並列プログラミングモデルの適用が推奨されている。

本稿は，ハイブリッド並列化に向けて，各ノードにおける OpenMP 並列化のための注意事項について説明する．本稿で扱う題材は有限体積法によるアプリケーションである．本稿の内容は単一のアプリケーションに特化した内容であるが，基本的な考え方は様々な分野に適用可能である．

本稿では特に有限体積法によってポアソン方程式を解くことによって得られる疎行列を係数行列とする大規模連立一次方程式を代表的な非定常反復法である共役勾配法（Conjugate Gradient

[1] https://www.cc.u-tokyo.ac.jp/events/lectures/173/

[2] http://nkl.cc.u-tokyo.ac.jp/seminars/multicore2021/

Method，CG 法）［6］の並列化に主眼を置いている．ハイブリッド並列化において重要なのは，各ノード内におけるスレッド並列化である． 以下，有限体積法の基礎的な考え方から始めて詳細に説明する．なお，OpenMP の文法等については文献〔7-10〕を参照されたい．

2. アプリケーション（P3D）の概要

本稿で対象とするアプリケーションは図 1 に示す差分格子によってメッシュ分割された三次元領域において，以下のポアソン方程式を解くもので P3D と呼んでいる：

$$\Delta\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = f \tag{1}$$

$$\phi = 0 \,@\, z = z_{\max} \tag{2}$$

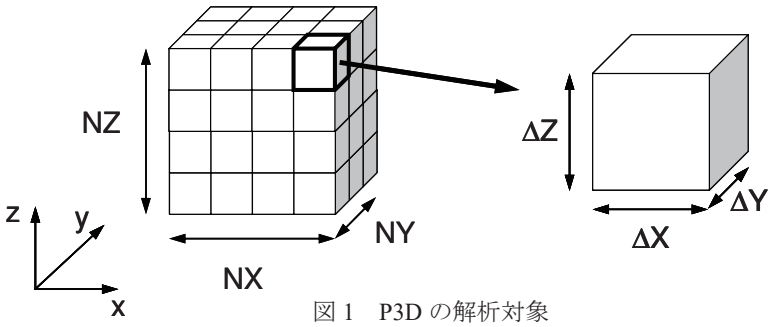


図 1　P3D の解析対象

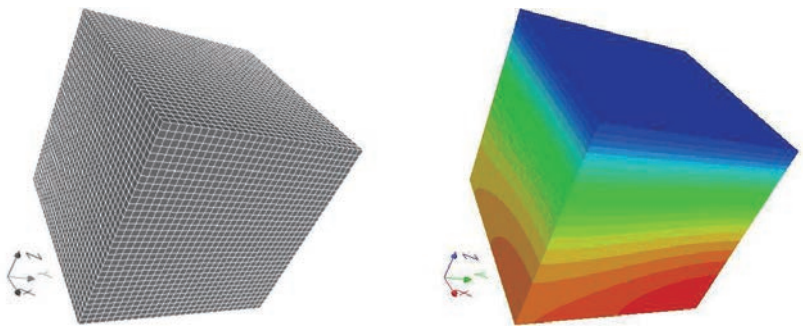
差分格子の各メッシュは直方体（辺長さはΔX，ΔY，ΔZ），X，Y，Z 各方向のメッシュ数は NX，NY，NZ

ここで，式（1）の右辺の f は体積あたりのフラックス（flux，流束）項で，以下に示すような空間分布を有すると仮定している：

$$f = dfloat(i_0 + j_0 + k_0) \tag{3}$$

$$i_0 = XYZ(icel,1), \quad j_0 = XYZ(icel,2), \quad k_0 = XYZ(icel,3) \tag{4}$$

式（4）における *XYZ(icel, k)*（k=1,2,3）は X，Y，Z 方向の差分格子のインデックスで各メッシュが X，Y，Z 方向の何番目にあるかを示している．支配方程式（1）を境界条件（2）とフラックスの条件（3），（4）を適用して解いた計算結果の例（ϕの分布）を図 2 に示す．



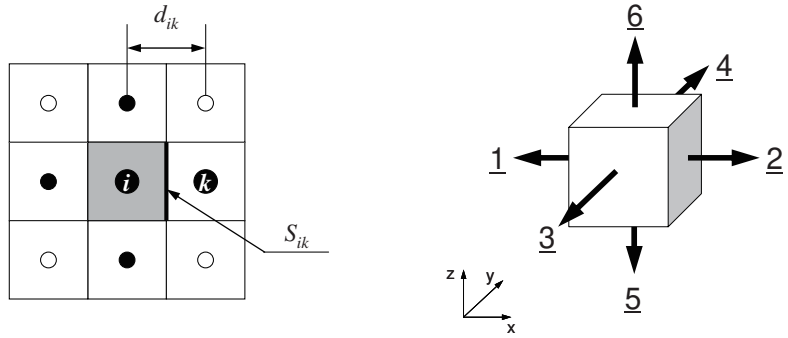
メッシュ分割図　　計算結果分布図（ϕ）

図 2　P3D の計算結果例（32^3=32,768 メッシュ）

形状は規則正しい差分格子であるが，プログラムの中では，一般性を持たせるために，有限体積法に基づき，非構造格子型のデータとして考慮する．図 2 における任意のメッシュ i の各面を通過するフラックスについて，式 (1) により以下に示す式 (5) のような釣り合い式が成立する：

$$\sum_{k=1}^{6}\left[\frac{S_{ik}}{d_{ik}}(\phi_k-\phi_i)\right]+V_i f_i=0 \tag{5}$$

ここで，S_{ik}：メッシュ i と隣接メッシュ k 間の表面積，d_{ik}：メッシュ i-k 重心間の距離，V_i：メッシュ i の体積，f_i：メッシュ i の体積あたりフラックスである（図 3（a）参照）．三次元問題の場合，各直方体メッシュは 6 個の面を持っているため，隣接メッシュ数は（最大で）6 であり，式（5）左辺第一項は k=1~6 の和となっている．図 3（b）は，三次元問題における各メッシュの局所面番号（すなわち隣接メッシュの番号付けのルール）である．



（a）メッシュ i と隣接メッシュ k の関係　　（b）局所面番号，隣接メッシュナンバリング

図 3　隣接メッシュとの関係

式（5）を図 1 のような三次元領域に適用すると，メッシュ i について式（6）が得られる：

$$\begin{aligned}&\frac{\phi_{k=1}-\phi_i}{\Delta x}\Delta y\Delta z+\frac{\phi_{k=2}-\phi_i}{\Delta x}\Delta y\Delta z+\frac{\phi_{k=3}-\phi_i}{\Delta y}\Delta z\Delta x+\frac{\phi_{k=4}-\phi_i}{\Delta y}\Delta z\Delta x+\\&\frac{\phi_{k=5}-\phi_i}{\Delta z}\Delta x\Delta y+\frac{\phi_{k=6}-\phi_i}{\Delta z}\Delta x\Delta y=f_i\Delta x\Delta y\Delta z\end{aligned} \tag{6}$$

これを整理すると，式（7）のようになり，三次元差分法の場合と同様の式が得られる：

$$\frac{\phi_{k=1}-2\phi_i+\phi_{k=2}}{\Delta x^2}+\frac{\phi_{k=3}-2\phi_i+\phi_{k=4}}{\Delta y^2}+\frac{\phi_{k=5}-2\phi_i+\phi_{k=6}}{\Delta z^2}=f_i \tag{7}$$

式（5）にもどって，これを整理すると，式（8）が得られる：

$$\left[\sum_{k=1}^{6}\frac{S_{ik}}{d_{ik}}\right]\phi_i-\left[\sum_{k=1}^{6}\frac{S_{ik}}{d_{ik}}\phi_k\right]=+V_i f_i \tag{8}$$

これは各メッシュ i について成立する式であるので，全メッシュ数を N とすると，N 個の方程式を連立させて，境界条件を適用し，連立一次方程式 $A\phi=b$ を解くことに帰着される．式（8）の左辺第一項は A の対角項，第二項は非対角項，式（8）の右辺は b に対応している．式（8）からもわかるように，各メッシュ i に対応する非対角成分の数は最大 6 個であるので，係数行列 A は疎（sparse）な行列となる．

3. CG 法について

式(8)を連立させて得られる連立一次方程式 $A\phi=b$ を解く方法として，本稿では反復法（Iterative Method），特に Krylov 部分空間法といわれる，非定常型の反復法を使用する〔6〕．係数行列 A は，式（7）からも類推されるように，対称かつ正定（Symmetric Positive Definite）であり，このような行列に対しては，通常は共役勾配法（Conjugate Gradient Method，CG 法）が使用される〔6〕．Krylov 部分空間法の収束は係数行列の性質（固有値分布）に強く依存するため，前処理を適用して固有値が 1 の周辺に集まるように行列の性質を改善する．前処理行列を M とすると，$\tilde{A}=M^{-1}A$ ，$\tilde{b}=M^{-1}b$ として（$[M]^{-1}$ を右から乗ずる場合もある），すなわち $\tilde{A}\phi=\tilde{b}$ という方程式を代わりに解くことになる．M^{-1} が A^{-1} をよく近似した行列であれば，$\tilde{A}=M^{-1}A$ は単位行列に近くなり，それだけ解きやすくなる．前処理付き CG 法のアルゴリズムの概要はリスト 1 のようになる：

```
Compute r(0)= b-Ax(0)
for i= 1, 2, …
    solve Mz(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)=z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 z(i)
    endif
    q(i)= Ap(i)
    αi  = ρi-1/p(i) q(i)
    x(i)= x(i-1) + αi p(i)
    r(i)= r(i-1) - αi q(i)
    check convergence |r|
End
```

リスト 1　前処理付き CG 法（共役勾配法，Conjugate Gradient Method）のアルゴリズム［6］

「solve Mz(i-1)= r(i-1)」の部分で，前処理行列 M を係数行列とする連立一次方程式を解く必要がある．本稿では，前処理手法として単純だが広く使用されている対角スケーリング（Diagonal Scaling）を使用する．点ヤコビ法（Point Jacobi Method）と呼ばれることもある［6］．この手法は，前処理行列 M として，元の係数行列 A の対角成分のみを取り出した対角行列（Diagonal Matrix）D を使用するものである．D の逆行列の各成分は対角成分の逆数となるので，リスト 1 の「solve Mz(i-1)= r(i-1)」では，ベクトル r の各成分を対角成分で割ることによって z が得られる．近くスケーリング・点ヤコビは，係数行列の対角成分が非対角成分より十分に大きい，「対角優位」な行列の場合，特に有効である．対角スケーリング・点ヤコビは，後述するように並列化も容易である．

4. P3D（単体 CPU 版）の処理内容と OpenMP による並列化

続いて単体 CPU 用版の P3D[3]〔11〕の処理内容を簡単に説明する．プログラムは Fortran 版，C 版があり，それぞれについて詳細な資料が準備されいるので詳しくはそちらをご覧いただきたい［11］．係数行列は対称であるが，対角成分，非ゼロ非対角成分に分け，上下三角成分を全て記憶している．行列の格納形式としては，非零成分のみを効率的に格納できる Compressed Row Storage（CRS）［6］という方法が採用されている．各成分は表 1 のように記憶されている：

表 1 Compressed Row Storage（CRS）による疎行列格納法

変数・配列名	型	Fortran	C	内　容
ICELTOT, N	整数	-	-	全メッシュ数
NPLU	整数	-	-	全ゼロ非対角成分数
D	倍精度実数	(N)	[N]	係数行列の対角成分
X	倍精度実数	(N)	[N]	解ベクトル
W	倍精度実数	(N,4)	[4][N]	CG 法で使用されるベクトル，添字は R,Z,P,Q など「リスト 1」中に現れる配列名に対応，DD は対角成分の逆数
B	倍精度実数	(N)	[N]	右辺ベクトル
indexLU	整数	(0:N)	[N+1]	各行の非ゼロ非対角成分数，一次元圧縮配列（CRS）
itemLU	整数	(NPLU)	[NPLU]	各行の非ゼロ非対角成分に対応した列番号，一次元圧縮配列（CRS）
AMAT	倍精度実数	(NPLU)	[NPLU]	各行の非ゼロ非対角成分 一次元圧縮配列（CRS）

表 1 に示した CRS 法によって一次元圧縮配列に格納された疎行列成分を使用して CG 法に現れる行列ベクトル積 $A\,p = q$ の計算を実施すると，リスト 2 のようになる．

・Fortran

```
do i= 1, N
      VAL= D(i)*W(i,P)
      do k= indexLU(i-1)+1, indexLU(i)
        VAL= VAL + AMAT(k)*W(itemLU(k),P)
      enddo
      W(i,Q)= VAL
    enddo
```

・C

```
for(i=0; i<N; i++) {
     VAL = D[i] * W[P][i];
     for(k=indexLU[i]; j<indexLU[i+1]; j++) {
         VAL += AMAT[k] * W[P][itemLU[k]];
     }
     W[Q][i] = VAL;
 }
```

リスト 2　CRS 法によって格納された疎行列成分による行列ベクトル積 $Ap = q$ の計算

[3] http://nkl.cc.u-tokyo.ac.jp/files/fvm.tar

前処理付き CG 法（リスト 1）の処理内容は，大きく分けて以下の 4 種類である：

① 前処理（対角スケーリング）
② ベクトルの内積
③ ベクトルの実数倍の加減
④ 行列ベクトル積

前処理に対角スケーリング・点ヤコビ法のような簡単な手法を適用すると，各処理は OpenMP の指示行（directive）を挿入するのみで簡単に並列化できる．IC 法，ILU 法等のデータ依存性を含むような前処理を適用する場合には，リオーダリングによって並列化を抽出する必要があるが，このような場合については，いずれ稿を改めて紹介したい．

リスト 3，リスト 4 は，Fortran，C においてリスト 1 中の①～④に相当する処理を OpenMP によって並列化した例である．基本的にリスト 1 の各演算は，行列ベクトル積も含めて，各ベクトルの N 個の成分に対する個別の処理であるので，OpenMP 指示行の挿入のみで並列化が実現できるのである：

① 前処理

$solve\ Mz^{(i-1)}= r^{(i-1)}$

```
!$omp parallel do
      do i= 1, NP
        W(i,Z)= W(i,R) * W(i,DD)
      enddo
!$omp end parallel do (省略可能)
```

② 内積

$\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$

```
     RHO= 0.d0
!$OMP PARALLEL DO PRIVATE(i) REDUCTION(+:RHO)
      do i= 1, N
        RHO= RHO + W(i,R) * W(i,Z)
      enddo
!$OMP END PARALLEL DO (省略可能)
```

③ ベクトル実数倍加減

$x^{(i)}= x^{(i-1)} + \alpha_i\ p^{(i)}$

```
!$omp parallel do
      do i= 1, NP
        X(i)= X(i) + ALPHA * W(i,P)
      enddo
!$omp end parallel do (省略可能)
```

④ 行列ベクトル積

$q^{(i)}= Ap^{(i)}$

```
!$omp parallel do private(i,VAL,k)
    do i= 1, N
      VAL= D(i)*W(i,P)
      do k= indexLU(i-1)+1, indexLU(i)
        VAL= VAL + AMAT(k)*W(itemLU(k),P)
      enddo
      W(i,Q)= VAL
    enddo
!$omp end parallel do (省略可能)
```

リスト 3　前処理付き CG 法の各処理と OpenMP による並列化（Fortran）

⑤ 前処理

solve $Mz^{(i-1)} = r^{(i-1)}$

```
#pragma omp parallel for private (i)
 for (i=0; i<N; i++) {
    W[Z][i] = W[R][i} * W[DD][i];
 }
```

⑥ 内積

$\rho_{i-1} = r^{(i-1)}\ z^{(i-1)}$

```
    RHO= 0.0;
#pragma omp parallel for private (i) reduction(+:RHO)
    for(i=0; i<N; i++){
      RHO= RHO + W[R][i] * W[Z][i];
    }
```

⑦ ベクトル実数倍加減

$x^{(i)} = x^{(i-1)} + \alpha_i\ p^{(i)}$

```
#pragma omp parallel for private (i)
 for (i=0; i<N; i++) {
    X[i] = X[i} + alpha * W[P][i];
 }
```

⑧ 行列ベクトル積

$q^{(i)} = Ap^{(i)}$

```
#pragma omp parallel for private (i,VAL,k)
 for(i=0; i<N; i++) {
     VAL = D[i] * W[P][i];
     for(k=indexLU[i]; j<indexLU[i+1]; j++) {
         VAL += AMAT[k] * W[P][itemLU[k]];
     }
     W[Q][i] = VAL;
 }
```

リスト 4　前処理付き CG 法の各処理と OpenMP による並列化（C）

次回（以降）は，この並列版プログラムの Odyssey 上での実行，性能評価，更なる最適化，高速化について紹介する．

（第 2 回（5 月号）へ続く）

参　考　文　献

[1] Wisteria/BDEC-01（「計算・データ・学習」融合スーパーコンピュータシステム）:https://www.cc.utokyo.ac.jp/supercomputer/wisteria

[2] 中島研吾, 塙敏博, 下川辺隆史, 伊田明弘, 芝隼人, 三木洋平, 星野哲也, 有間 英志, 河合直聡, 坂本龍一, 岩下武史, 八代尚, 長尾大道, 松葉浩也, 荻田武史, 片桐孝洋, 古村孝志, 鶴岡弘, 市村強, 藤田航平, 近藤正章,「計算・データ・学習」融合スーパーコンピュータシステム「Wisteria/BDEC- 01」の概要, 情報処理学会研究報告（2020-HPC-179-01）, 2021

[3] 東京大学情報基盤センター：第 173 回お試しアカウント付き並列プログラミング講習会「OpenMP によるマルチコア・メニィコア並列プログラミング入門（Wisteria/BDEC-01（Odyssey, A64FX 搭載））」, https://www.cc.u-tokyo.ac.jp/events/lectures/173/

[4] OpenMP によるマルチコア・メニィコア並列プログラミング入門（Wisteria/BDEC-01（Odyssey, A64FX 搭載）, http://nkl.cc.u-tokyo.ac.jp/seminars/multicore2021/

[5] Nakajima, K., Parallel Iterative Solvers of GeoFEM with Selective Blocking Preconditioning for Nonlinear Contact Problems on the Earth Simulator. ACM/IEEE Proceedings of SC'03, https://doi.org/10.1145/1048935.1050164, 2003

[6] Saad, Y.: Iterative Methods for Sparse Linear Systems Second Edition, SIAM, 2003

[7] OpenMP: https://www.openmp.org/

[8] Chandra, R.他，Parallel Programming in OpenMP，Morgan Kaufmann Publishers, 2001
[9] 牛島省，OpenMP による並列プログラミングと数値計算法，丸善，2006
[10] Mattson, T.G., Sanders, B.A., Massingill, B.L., Patterns for Parallel Programming, Software Patterns Series (SPS), Addison-Wesley, 2005
[11] P3D 関連資料
- ソースコード等： http://nkl.cc.u-tokyo.ac.jp/files/fvm.tar
- 解説資料（Fortran）：http://nkl.cc.u-tokyo.ac.jp/seminars/multicore2021/omp-f-01.pdf
- 解説資料（C）： http://nkl.cc.u-tokyo.ac.jp/seminars/multicore2021/omp-c-01.pdf