数値計算アルゴリズム：科学技術計算のためのマルチコアプログラミング入門

プログラミング実習編

中島研吾

東京大学情報基盤センター

プログラミング実習

- L1におけるMETHOD=3（対角スケーリング）に相当するプログラムに対して:
  - OpenMP並列化を実施する，時間測定関数挿入
  - 行列格納形式をELLにする

- 計算速度を元のプログラムと比較してみる
- 時間があればプロファイラも使用してみよ

ファイルコピー：元のプログラム

```
>$ cd <$O-TOP>

>$ cp /home/ss/aics60/C/ex-c.tar .
>$ cp /home/ss/aics60/F/ex-f.tar .

>$ tar xvf ex-c.tar
>$ tar xvf ex-f.tar

>$ cd multicore
```

以下のディレクトリが出来ていることを確認

```
      ex
```

これらを以降　<$O-ex>

元のプログラムのありか on FX10

- 所在
  - <$O-ex>/src, <$O-ex>/run
- コンパイル，実行方法
  - 本体
    - cd <$O-ex>/src
    - make
    - <$O-ex>/run/ex-sol（実行形式）
  - コントロールデータ
    - <$O-ex>/run/INPUT.DAT
  - 実行用シェル
    - 自分で作成すること

手 順

- ex以下のプログラムは全く並列化されていない（L1からMETHOD=3の場合のみ取り出したもの）
- まずOpenMP並列化する
  - SMPindexGを使っても使わなくても良い
- 次にELL化する
  - poi_gen
  - solver_PCG

変更例（poi_gen）

全ての行において非零非対角成分の数=6（=3+3）

```
!C
!C-- 1D array

      allocate (indexL(0:nn), indexU(0:nn))
      indexL= 0
      indexU= 0

      do icel= 1, ICELTOT
        indexL(icel)= 3
        indexU(icel)= 3
      enddo

      do icel= 1, ICELTOT
        indexL(icel)= indexL(icel) + indexL(icel-1)
        indexU(icel)= indexU(icel) + indexU(icel-1)
      enddo

      NPL= indexL(ICELTOT)
      NPU= indexU(ICELTOT)

      allocate (itemL(NPL), AL(NPL))
      allocate (itemU(NPU), AU(NPU))

      itemL= 0
      itemU= 0
         AL= 0.d0
         AU= 0.d0
```

変更例(poi_gen)

行列成分が0になる部分に対応する列番号:ICELTOT+icou

```
      icou= 0
      do i= 1, ICELTOT
      do k= 1, 3
        if (itemL(indexL(i-1)+k).eq.0) then
          icou= icou + 1
          itemL(indexL(i-1)+k)= ICELTOT + icou
        endif
        if (icou.eq.N2) icou= 0
      enddo
      enddo

      icou= 0
      do i= 1, ICELTOT
      do k= 1, 3
        if (itemU(indexU(i-1)+k).eq.0) then
          icou= icou + 1
          itemU(indexU(i-1)+k)= ICELTOT + icou
        endif
        if (icou.eq.N2) icou= 0
      enddo
      enddo
```

icou: 1-N2
N2:余り大きくない数(例えば256)に設定する

PHI(ICELTOT+N2)

変更例（solver_PCG）

行列ベクトル積：いくつかの実装を試してみよ（1/3）

```
      do i= 1, N
        VAL= D(i)*W(i,P)

        do k= indexL(i-1)+1, indexL(i-1)+3
          VAL= VAL + AL(k)*W(itemL(k),P)
        enddo

        do k= indexU(i-1)+1, indexU(i-1)+3
          VAL= VAL + AU(k)*W(itemU(k),P)
        enddo

        W(i,Q)= VAL
      enddo
```

```
W(ICELTOT+N2,4)
```

変更例（solver_PCG）

行列ベクトル積：いくつかの実装を試してみよ（2/3）

```
      do i= 1, N
        VAL= D(i)*W(i,P)

        do k= 1, 3
          kk= (i-1)*3 + k
          VAL= VAL + AL(kk)*W(itemL(kk),P)
        enddo

        do k= 1, 3
          kk= (i-1)*3 + k
          VAL= VAL + AU(kk)*W(itemU(kk),P)
        enddo

        W(i,Q)= VAL
      enddo
```

```
W(ICELTOT+N2,4)
```

変更例（solver_PCG）

行列ベクトル積：いくつかの実装を試してみよ（3/3）

```
      do i= 1, N
        VAL= D(i)*W(i,P)

        do k= 1, 3
          kk= (i-1)*3 + k
          VAL= VAL + AL(kk)*W(itemL(kk),P)
     &             + AU(kk)*W(itemU(kk),P)
        enddo

        W(i,Q)= VAL
      enddo
```

```
      do i= 1, N
        VAL= D(i)*W(i,P)

        do k= 1, 6
          kk= (i-1)*6 + k
          VAL= VAL + AMAT(kk)*W(itemLU(kk),P)
        enddo

        W(i,Q)= VAL
      enddo
```