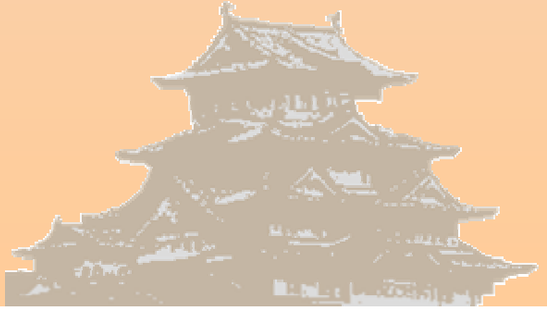
SPEC OMPの HPF化とその性能評価

姫路工業大学　大学院工学研究科
坂上仁志，森井宏幸

SPEC OMP2001

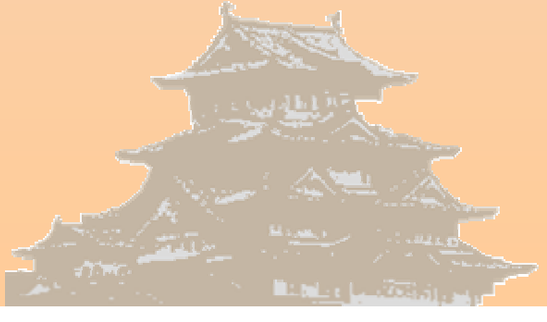
- SPECによって開発されたOpenMPによる共有メモリ型並列計算機システムの性能評価を行うベンチマークプログラムである.
  - OpenMPは，HPFと同様に指示文ベースである.
  - 単なるベンチマークコードではなく，実用的なコードである.
  - FORTRANコード8本とCコード2本からなる.
- HPFの可用性と並列性能を検証する.
- SPEC HPFの開発を目指す.



OpenMPの並列化

- 並列領域をHPFで並列化する.
  - PARALLEL,END PARALLEL指示文
  - DO指示文

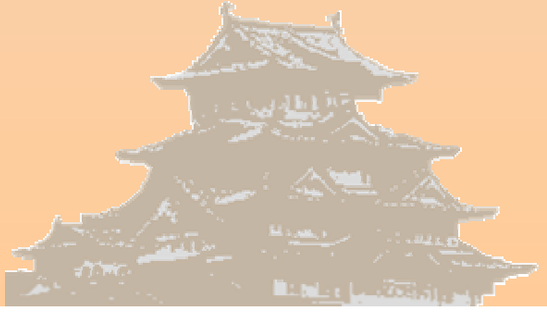
```
      …
!$OMP PARALLEL
!$OMP DO
      do j = 1, n
        do i = 1, m
          u(i,j) = p(i,j)
          v(i,j) = p(i,j+1)
        end do
      end do
!$OMP END DO
!$OMP END PARALLEL
      …
```



HPFによる並列化(1)

- データ分散と処理分割
  - PROCESSORS指示文
  - DISTRIBUTE指示文
  - INDEPENDENT指示文

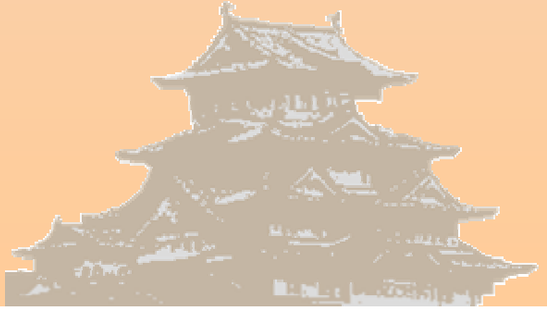
```
      …
!HPF$ PROCESSORS proc(NPROC)
!HPF$ DISTRIBUTE (*,BLOCK) ONTO proc :: u,v,p
      …
!HPF$ INDEPENDENT
      do j = 1, n
        do i = 1, m
          u(i,j) = p(i,j)
          v(i,j) = p(i,j+1)
        end do
      end do
      …
```



HPFによる並列化(2)

- 通信の指示による通信効率向上
  - SHADOW，REFLECT指示文
  - ON-HOME-LOCAL指示文

```
      …
!HPF$ SHADOW (0,0:1) :: p
      …
!HPF$ REFLECT p
!HPF$ INDEPENDENT
      do j = 1, n
!HPFJ ON HOME(u(i,j)), LOCAL BEGIN
        do i = 1, m
         u(i,j) = p(i,j)
         v(i,j) = p(i,j+1)
        end do
!HPFJ END ON
      end do
      …
```



SWIMの並列化

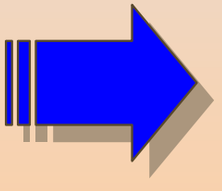
- 浅水方程式による海水のシミュレーションプログラムである.
- 同一のDOループ内で結果を格納する配列の添字が異なる場合に並列性能が悪い.
  - Owner Computes Ruleとの矛盾

```
      dimension a(10),b(9)
!HPF$ PROCESSORS proc(2)
!HPF$ DISTRIBUTE (BLOCK) ONTO proc :: a,b
      do i = 1, 9
        a(i+1) = ...
        b(i)   = ...
      end do
```

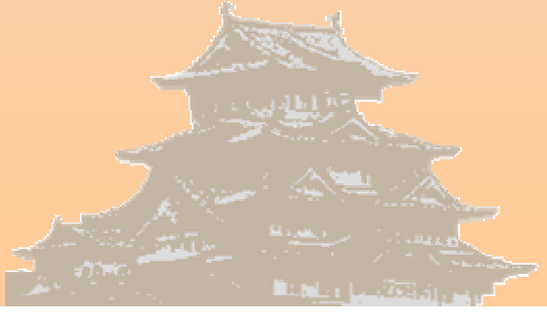
並列性能低下の原因

- OCRと矛盾する場合は，一時変数を動的に確保して計算後，本来の変数にコピーする．
  - 並列化はできるが，大きなオーバーヘッド

```
do i = 1, 9
  a(i+1) = ...
  b(i)   = ...
end do
```



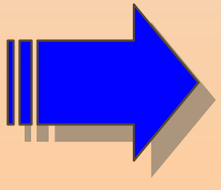
```
allocate(atemp)
do i = 1, 9
  atemp(i) = ...
  b(i)   = ...
end do
do i = 1, 9
  a(i+1) = atemp(i)
end do
deallocate(atemp)
```



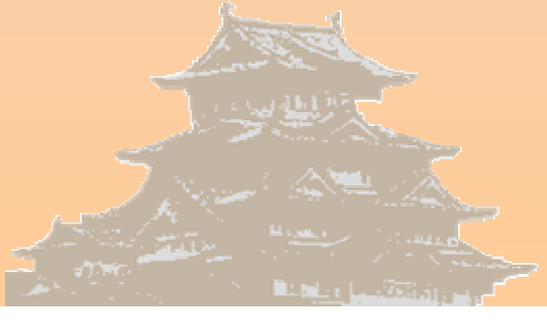
問題の解決方法(1)

- 添字が異なる配列の処理を別々のDOループに分ける.
  - ソースを修正する必要があるが，容易に並列性能を改善できる.
  - ベクトル処理の効率が低下する場合がある.

```
do i = 1, 9
  a(i+1) = ...
  b(i)   = ...
end do
```



```
do i = 1, 9
  a(i+1) = ...
 end do
 do i = 1, 9
  b(i)   = ...
end do
```



問題の解決方法(2)

- OCRとDOループ処理が矛盾しないよう配列の分散をALIGN指示文で調整する.
  - 指示文のみで並列性能を改善できる.
  - 異なるDOループで最適なALIGNが異なる場合は,対応できない.

```
!HPF$ ALIGN b(i) WITH a(i+1)
    ...
    do i = 1, 9
      a(i+1) = ...
      b(i)   = ...
    end do
```

両方の方法の併用

- ベクトル処理の効率を考慮する.

×

```
do i = 1, 9
  a(i+1) = p(i)*2
  b(i)   = -p(i)/2
end do
...
do i = 1, 9
  a(i) = 0
  b(i) = 1
end do
```

```
    ...
!HPF$ INDEPENDENT
    do i = 1, 9
      a(i+1) = p(i)*2
    end do
!HPF$ INDEPENDENT
    do i = 1, 9
      b(i)   = -p(i)/2
    end do
    ...
!HPF$ INDEPENDENT
    do i = 1, 9
      a(i) = 0
      b(i) = 1
    end do
```

○

```
!HPF$ ALIGN b(i) WITH a(i+1)
    ...
!HPF$ INDEPENDENT
    do i = 1, 9
      a(i+1) = p(i)*2
      b(i)   = -p(i)/2
    end do
    ...
!HPF$ INDEPENDENT
    do i = 1, 9
      a(i) = 0
    end do
!HPF$ INDEPENDENT
    do i = 1, 9
      b(i) = 1
    end do
```

SWIMの性能評価

- OpenMPと同程度の並列性能を得ることができた.
  - NEC SX-5（大阪大学CMC）

		実行時間 [秒]			高速化率		
		改善前	改善後	OpenMP	改善前	改善後	OpenMP
プロセッサ数	1	505.3	294.7	283.2	1.00	1.00	1.00
	2	265.9	153.3	150.4	1.90	1.92	1.88
	3	183.7	109.2	103.2	2.75	2.70	2.74
	4	144.4	82.3	81.4	3.50	3.58	3.48
	8	81.8	51.3	48.4	6.18	5.74	5.85

MGRIDの並列化

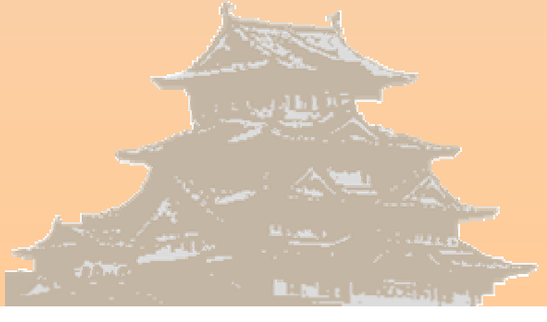
- マルチグリッドにより3次元のポテンシャル場を計算をするプログラムである.
- 主プログラムと副プログラムで引数の次元数が異なるため，並列化することができなかった.

主プログラム

```
dimension big(NALL),is(k),il(k)
•••
do i= 1, k
  callsub( big(is(i)), il(i) )
end do
•••
stop
end
```

副プログラム

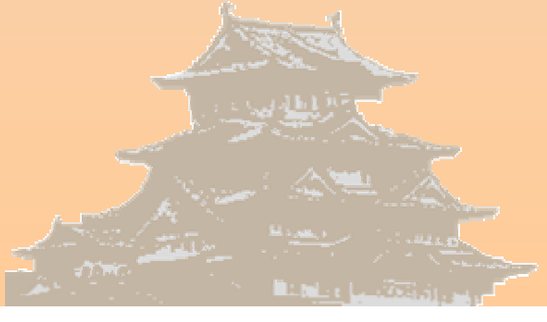
```
subroutine sub( a, m )
dimension a(m,m,m)
•••
return
end
```



一時配列を用いた問題解決

- 動的に配列を確保／分散し，非分散の仮引数から値を複写して，並列化する．

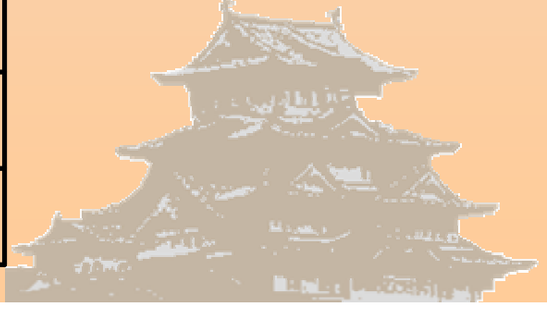
```
      subroutine sub( aa, m )
      dimension aa(m*m*m)
!HPF$ PROCESSORS proc(NPROC)
!HPF$ DISTRIBUTE (*,*,BLOCK) ONTO proc :: a
      allocatable a(:,:,:)
      allocate(a(m,m,m))
      a ← aa
      ...
      ...
      aa ← a
      deallocate(a)
      return
      end
```



MGRIDの性能評価(1)

- 動的な配列の確保／解放およびデータ複写のオーバーヘッドのため，OpenMP程の並列性能が得られなかった.
  - そのオーバーヘッドを測定すると約80秒

		実行時間 [秒]		高速化率	
		HPF	OpenMP	HPF	OpenMP
プロセッサ数	1	331.1	230.2	1.00	1.00
	2	207.1	118.6	1.60	1.94
	3	160.3	81.1	2.07	2.84
	4	141.0	63.1	2.35	3.65
	8	95.2	36.0	3.48	6.39

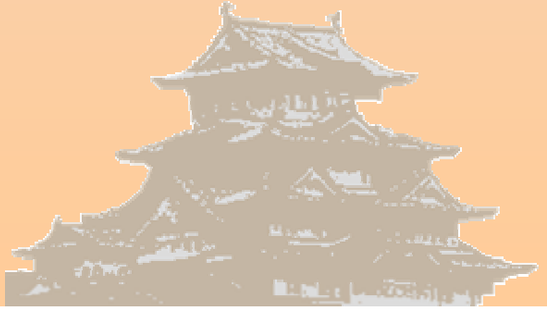


MGRIDにおける性能改善

- 大きさごとに別々の分散配列を用意して，呼び出し時に振り分けれる．
  - 動的な配列の確保／解放およびデータ複写は必要ない．
  - 大幅なソース修正が必要である．

```
allocate(a1(il(1),il(1),il(1))
allocate(a2(il(2),il(2),il(2))
...
allocate(ak(il(k),il(k),il(k))
```

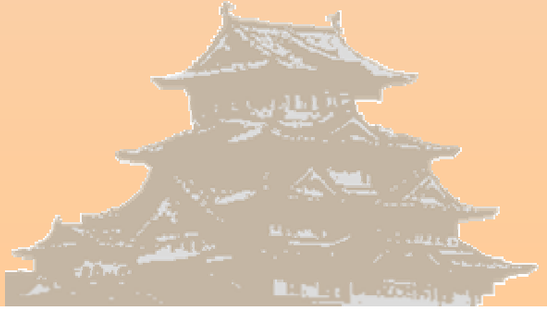
```
do i = 1, k
  if( i.eq. 1 ) then
    call sub( a1, il(i) )
  else if( i.eq. 2 ) then
    call sub( a2, il(i) )
  ...
  else if( i.eq. k ) then
    call sub( ak, il(i) )
  end if
end do
```



ifcの拡張機能による改善

- intel fortran compilerの拡張機能
  - loc：アドレスの取得
  - %val：引数のcall by valueによる結合

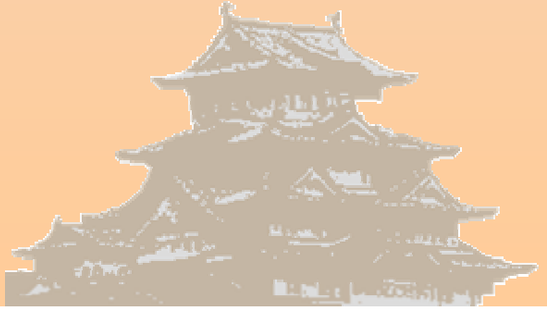
```
allocate(a1(il(1),il(1),il(1))
allocate(a2(il(2),il(2),il(2))
...
allocate(ak(il(k),il(k),il(k))
la(1) = loc( a1 )
la(2) = loc( a2 )
...
la(k) = loc( ak )
...
do i = 1, k
   callsub( %val(la(i)), il(i) )
end do
```



HPFコンパイラの内部仕様

- HPFコンパイラは，分散配列の情報をサブルーチンに引き渡すため，内部で引数を追加している．
  - プリコンパイル後のソースを修正

```
la$(1) = loc( a1$sd1 )
la$(2) = loc( a2$sd1 )
...
la$(k) = loc( ak$sd1 )
...
do i = 1, k
   callsub( %val(la(i)), il(i), %val(la$(i)), pghpf_type(25) )
end do
```



MGRIDの性能評価(2)

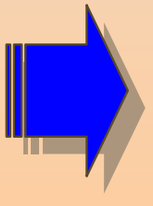
- 大幅に性能が改善した.
  - PCクラスタ(地球シミュレータセンター)
  - 計算回数(1/50)
- HPFコンパイラによる対応が必要である.

		実行時間 [秒]		高速化率	
		HPF	loc&%val	HPF	loc&%val
プロセッサ数	1	278.0	137.1	1.00	1.00
	2	151.2	84.5	1.84	1.62
	4	82.0	47.9	3.39	2.86
	8	47.3	27.8	5.88	4.93

APSIの並列化(1)

- 3次元流体方程式を解き，湖環境における汚染物質の拡散をシミュレーションする.
- 実引数と仮引数で次元数が異なる.
  - 変数ごとに別々に宣言するように修正した.

```
allocatable :: work(:)
allocate(work(ntotal))
…
lc=1
lstepc = 1 + nx*ny*nz
…
call run(nx,ny,nz, work(lc), work(lstepc))
…
subroutine run(nx,ny,nz,c,stepc)
dimension c(*),stepc(*)
…
call dctdx(nx,ny,nz,c)
…
```

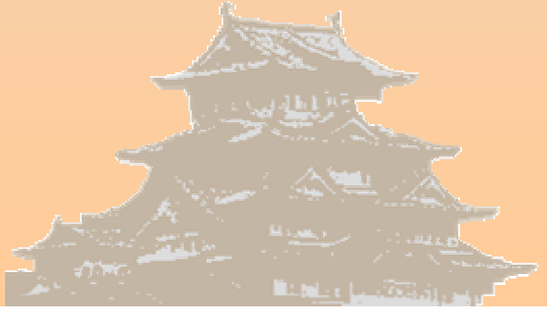


```
allocatable :: c(:,:,:),stepc(:,:,:)
…
allocate(c(nx,ny,nz))
allocate(stepc(nx,ny,nz))
…
call run(nx,ny,nz,c,stepc)
…
subroutine run(nx,ny,nz,c stepc)
dimension c(nx,ny,nz),stepc(nx,ny,nz)
…
call dctdx(nx,ny,nz,c)
…
```

APSIの並列化(2)

- 並列処理を行うべき配列の次元が，場所によって異なっている．
  - サブルーチン呼出し時の再マッピングで分散次元を変更した．

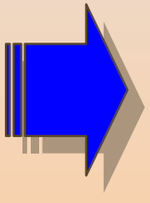
```
      ...
!HPF$ DISTRIBUTE (*,*,BLOCK) ONTO proc :: c
      ...
      ...（並列処理は3次元目）
      ...
      call advc(nx,ny,nz,c)
      ...
c-------------------------------------
      subroutine advc(nx,ny,nz,c)
      ...
!HPF$ DISTRIBUTE (*,BLOCK,*) ONTO proc :: c
      ...
      ...（並列処理は2次元目）
      ...
```



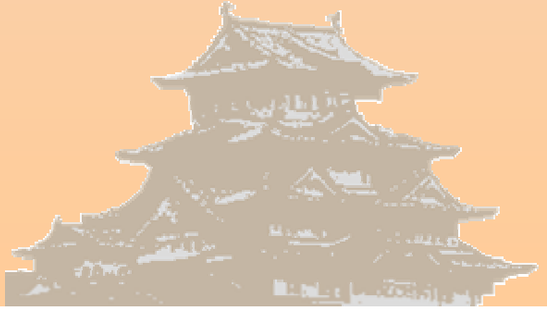
APSIの並列化(3)

- 副プログラムの並列呼び出し時に，分散配列の一部を渡す．
  - 形状引継配列を用いるように修正した．

```
      dimension c(nx*ny*nz)
      ...
      mlag = 1 - nxny
!$OMP PARALLEL DO
      do i = 1, nz
        mlag = mlag + nxny
       call sub(nx,ny,c(mlag))
      end do
!$OMP END PARALLEL
      ...
      subroutine sub(nx,ny,c)
      dimension c(nx,ny)
      ...
```



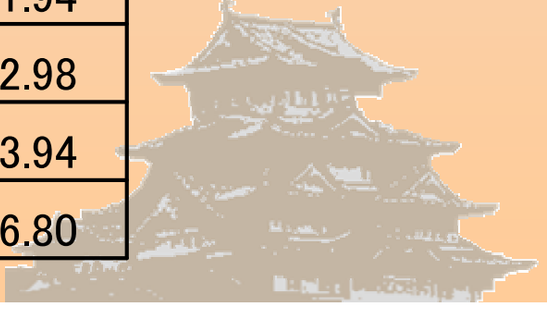
```
      dimension c(nx,ny,nz)
      ...
!HPF$ DISTRIBUTE (*,*,BLOCK) ONTO proc :: c
      interface
        pure extrinsic(fortran_local)
     &  subroutine sub(nx,ny,c)
      ...
      end interface
      ...
!HPF$ INDEPENDENT
      do i = 1, nz
       call sub(nx,ny,c(:,:,i))
      end do
      ...
      subroutine sub(nx,ny,c)
      dimension c(nx,ny)
      ...
```



APSIの性能評価

- OpenMP程の並列性能は得られなかった.
  - 形状引継配列から一時配列への複写
  - ベクトル長の減少
  - コンパイラが自動作成するエラー処理コード
- -Mnoentryオプションによりエラー処理コード生成を抑制し，OpenMPに近い性能が得られた.

		実行時間 [秒]			高速化率		
		HPF	noentry	OpenMP	HPF	noentry	OpenMP
プロセッサ数	1	1495.9	1020.5	928.4	1.00	1.00	1.00
	2	763.2	541.6	477.6	1.96	1.88	1.94
	3	524.3	358.9	311.6	2.85	2.84	2.98
	4	388.8	267.4	235.4	3.85	3.82	3.94
	8	271.0	171.2	136.6	5.52	5.96	6.80



指示文および追加･修正行数

- HPF指示文の行数は，includeを展開して計数している.
  - 実際の作業行数は，もっと少ない.

プログラム	オリジナル	OpenMP	HPF	追加･修正
SWIM(改善前)	約300	10	41	0
SWIM(改善後)	約300	10	77	20
MGRID	約400	84	127	208
MGRID(loc&%val)	約400	84	91	103+40
APSI	約4300	144	166	約400

並列化のまとめ

- SWIM
  - ソースを修正して，DOループの処理分割とOCR間の矛盾を解消する必要があった.
- MGRID
  - 実引数と仮引数で配列の次元数が異なるため，動的な配列確保とデータ複写を用いた.
  - ifcの拡張機能で効率の良い並列化を行った.
- APSI
  - 実引数宣言の修正および形状引継配列を用いた副プログラムの並列呼び出しを行った.

性能評価のまとめ

- SWIM
  - OpenMPと同等の性能が得られた.
- MGRID
  - 動的な配列の確保／解放およびデータ複写のオーバーヘッドが大きく，高い並列性能は得られなかった.
  - しかし，ifcの拡張機能を用いて改善できた.
- APSI
  - OpenMPに近い性能が得られた.