# HPFからMPIへのトランスレータ ffhpf

富士通（株）ソフトウェア事業本部
岩下英俊
<iwa@soft.fujitsu.com>

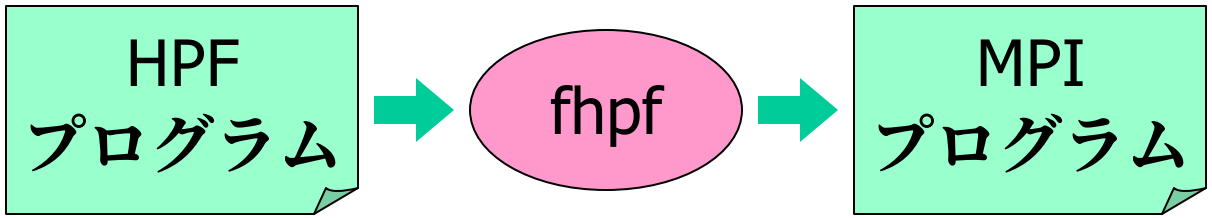
2004年3月12日

# 内容

- fhpfの特徴とご使用方法
- 新技術のご紹介
- 機能と性能
- ライセンスに関して
- まとめ

# 特徴

- トランスレータ方式
- MPI呼出しを直接生成




“MPIプログラムを自動生成するツール”

- 実行時ライブラリなし

→FortranとMPIが動く環境なら、どこでも実行可能

# fhpfの変換の実例

```
% fhpf block.hpf -f77
fhpf V1.1.3/MPI -- HPF translator for Solaris system
block.hpf -> block.mpi.f
%
```

入力ファイル：block.hpf

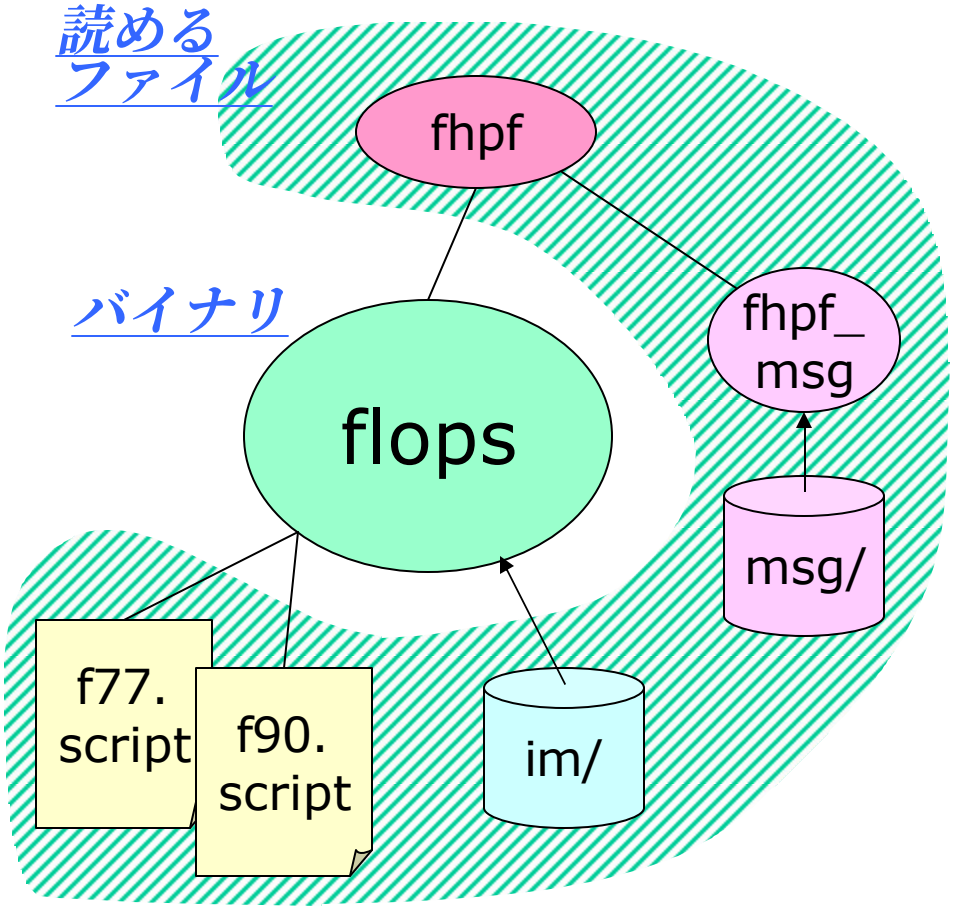
```
      integer A(100)
!hpf$ processors P(4)
!hpf$ distribute A(block) onto P

!hpf$ independent
      do i=n1,n2
        A(i)=i
      enddo
      end
```

出力ファイル：block.mpi.f

```
  PROGRAM ORG__MAIN___
  INCLUDE 'mpif.h'
  INTEGER*4 ORG_RANK
  INTEGER*4 ORG_SIZE
  INTEGER*4 ierrX0
  COMMON /ORG_COMMON/ORG_RANK,ORG_SIZE
  CALL mpi_init(ierrX0)
  CALL mpi_comm_rank(MPI_COMM_WORLD,ORG_RANK,ierrX0)
  CALL mpi_comm_size(MPI_COMM_WORLD,ORG_SIZE,ierrX0)
  CALL ORG___MAIN___()
  CALL mpi_finalize(ierrX0)
  END
```

```
SUBROUTINE ORG___MAIN___()
INCLUDE 'mpif.h'
INTEGER*4 a(0:24)
INTEGER*4 i
INTEGER*4 n1
INTEGER*4 n2
INTEGER*4 ORG_RANK
INTEGER*4 ORG_SIZE
INTEGER*4 spmdX0
INTEGER*4 spmdX1
INTEGER*4 spmd_startX0
INTEGER*4 spmd_endX0
INTEGER*4 spmd_stepX0
INTEGER*4 ierrX0
COMMON /ORG_COMMON/ORG_RANK,ORG_SIZE
spmdX0 = mod(ORG_RANK,4)
spmdX1 = 25*spmdX0
IF (spmdX0.LE.(n1+(-1))/25) THEN
  spmd_startX0 = n1+(-1)-spmdX1
ELSE
  spmd_startX0 = 0
ENDIF
IF (spmdX0.LT.(n2+(-1))/25) THEN
  spmd_endX0 = 24
ELSE
  spmd_endX0 = n2+(-1)-spmdX1
ENDIF
spmd_stepX0 = 1
DO i=spmd_startX0,spmd_endX0,1
  a(i-0) = (i+spmdX1)-(-1)
ENDDO
END
```

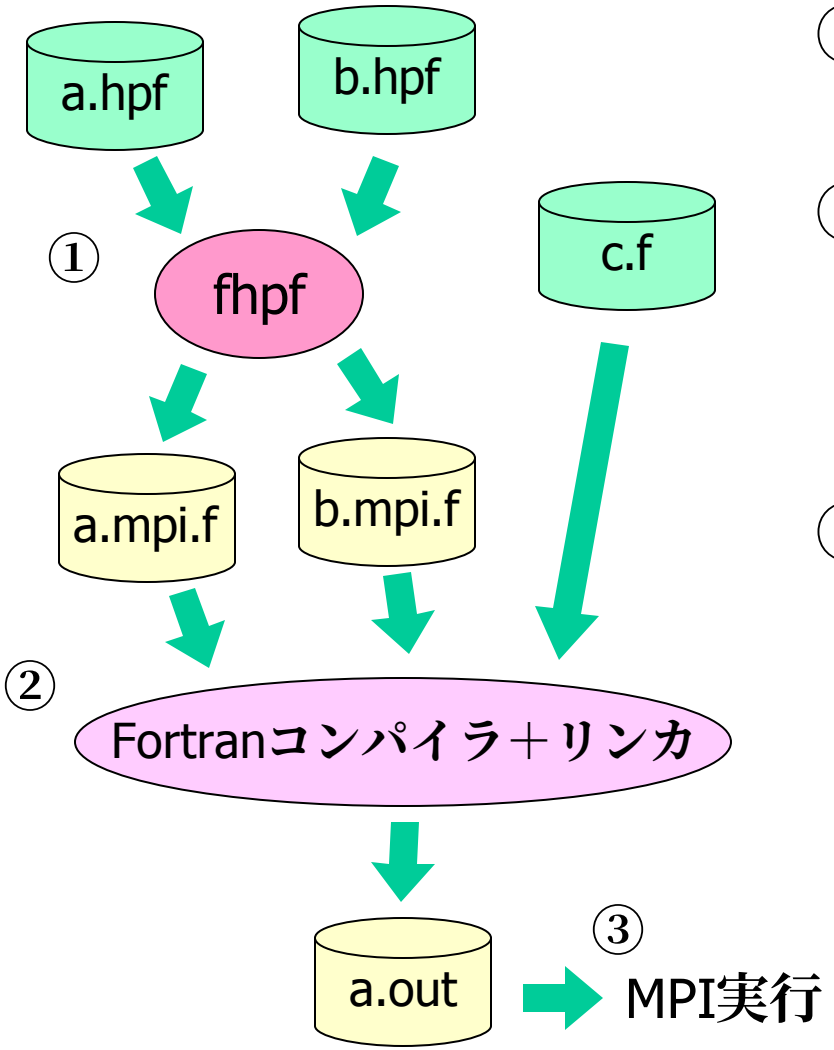
# ソフトウェア（hpf V1.1.3）の構成




| | Linux版 | Solaris版 |
|---|---|---|
| CPU | Intel IA32系 | SPARC系 |
| OS | Linux 2.4以降 | Solaris 7以降 |

- fhpf (shell script)
  - 利用者コマンド
- flops (binary)
  - HPFコンパイラ本体
- f77.script, f90.script
  - flopsの制御スクリプト
- im/ 配下
  - 翻訳時生成ライブラリのテンプレート（中間言語表現）
- fhpf_msg (shell script)
  - メッセージ出力用フィルタ
- msg/ 配下
  - メッセージデータファイル

# 使用方法(1) 翻訳～実行




① HPF翻訳

【例】`fhpf a.hpf b.hpf`

② Fortran翻訳（MPIと結合）

- 任意のF77, F90コンパイラ
  確認済：Intel, GNU, Fujitsu

【例】`mpif77 a.mpi.f b.mpi.f c.f`

③ 並列計算機で実行

- MPI1.1 APIが動作する環境
  確認済：MPICH/Linux,
  LAM-MPI/Linux,
  Fujitsu MPI/PRIMEPOWER）

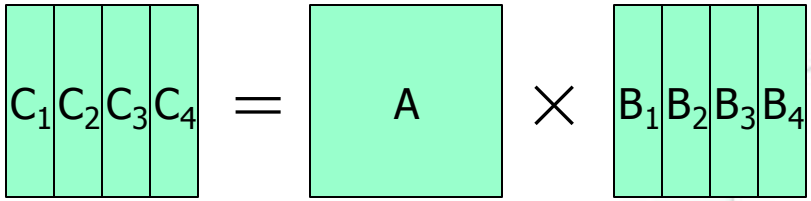
【例】`mpirun -np 4 a.out`

# 使用方法(2) FORTRAN77モード

- HPFはF90ベース ⇔ Linux環境はg77利用者が多い
- オプション -f77 を提供
  - 入力は 原則として)F90仕様を含まないHPFに限定
  - Interface blockがない → 手続間の整合性は利用者責任

【例】行列積 C = A×B

- $C_1=A\times B_1, \cdots, C_4=A\times B_4$ に分割し、各々Fortranで計算



# 使用方法(2) FORTRAN77モード 続き)

HPF77プログラム　my_matmul.f

```
      real A(100,100),B(100,100),C(100,100)
!hpf$ processors P(4)
!hpf$ distribute B(*,block) onto P
!hpf$ distribute C(*,block) onto P
      read(*,*) A,B
      call my_matmul_sub(C,A,B)
      write(*,*) C
      end
```

FORTRAN77プログラム　mmm_sub.f

```
      subroutine my_matmul_sub(C,A,B)
      real A(100,100),B(100,25),C(100,25)
      do 10 j=1,25
        do 10 i=1,100
          C(i,j)=0.0
          do 10 k=1,100
 10         C(i,j)=C(i,j)+A(i,k)*B(k,j)
      return
      end
```

```
% fhpf -f77 my_matmul.f                                  ! HPF翻訳
% mpif77 my_matmul.mpi.f mmm_sub.f -o my_matmul          ! F77翻訳と結合
% mpirun -np 4 my_matmul                                 ! 4並列実行
```

# 内容

- fhpfの特徴とご使用方法
- 新技術のご紹介
- 機能と性能
- ライセンスに関して
- まとめ

# 新技術(1) 整列の正規化

①プロセッサ形状を正規化

②テンプレートを生成

入力プログラム

```
!hpf$ processors P(4)
  real A(1:30),C(1:29),D(1:15)
  real F(1:10,1:30)
!hpf$ template T(30)
!hpf$ distribute A(block) onto P
!hpf$ distribute T(block) onto P
!hpf$ align C(I) with A(I+1)
!hpf$ align D(I) with A(2*I)
!hpf$ align F(*,I) with T(I)

!hpf$ independent
  L1: do I=2,29
!hpf$   on home(A(I)) begin
    A(I)=C(I)*I+F(I,I)
!hpf$   end on
  end do

!hpf$ independent
  L2: do K=1,10
!hpf$   on home(D(K)) begin
    D(K)=A(2*K)
!hpf$   end on
  end do

  end
```

③テンプレートへの直接整列に

④テンプレートをHOMEに

③ A(**I-1**)=C(I)*I+F(I,**I-1**)

③ D(**2*K-1**)=A(**2*K-1**)

正規化結果（イメージ）

```
!hpf$ processors P(0:3)
  real A(0:29),C(0:29),D(0:29)
  real F(1:10,0:29)
!hpf$ template T1(0:29)
!hpf$ distribute T1(block(8)) onto P
!hpf$ align A(I) with T1(I)
!hpf$ align C(I) with T1(I)
!hpf$ align D(I) with T1(I)
!hpf$ align F(*,I) with T1(I)
!hpf$ independent
  L1: do I=1,28
!hpf$   on home(T1(I)) begi
    A(I)=C(I+1)*(I+1)+F(I+1,I)
!hpf$   end on
  end do

!hpf$ independent
  L2: do K=1,19,2
!hpf$   on home(T1(K)) begin
    D(K)=A(K)
!hpf$   end on
  end do
  end
```

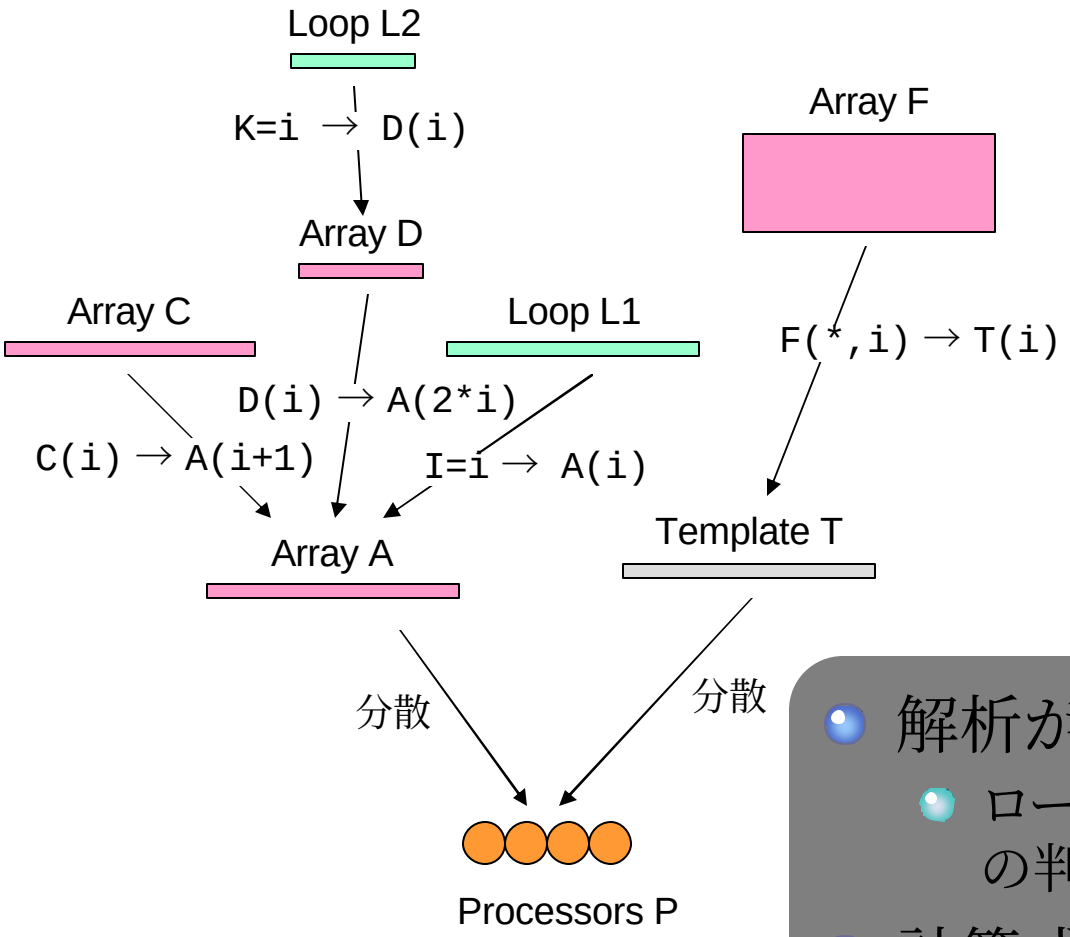
③形状とマッピングを正規化

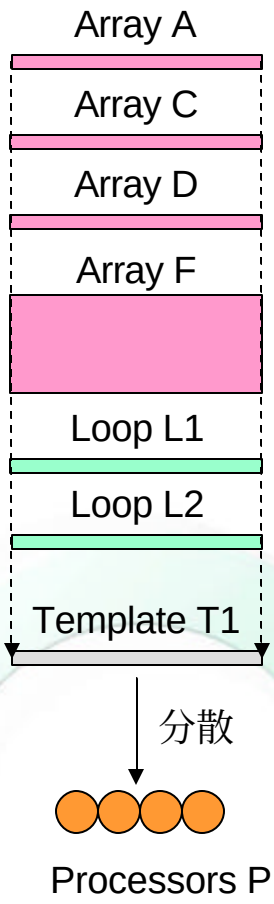
④ループパラメタを調整

④ I→I+1

④ K→(K+1)/2

# 新技術(1) 整列の正規化 続き)



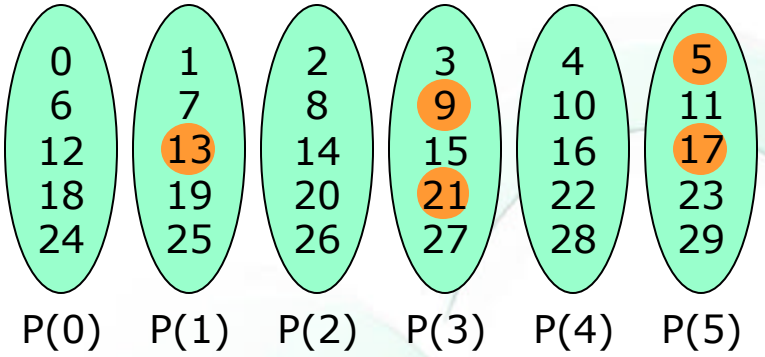

- 解析が容易に
  - ローカルアクセスの判定など
- 計算式が簡単に
  - ループ分割後の上下限の計算など

# 新技術(2)　翻訳時生成ライブラリ

- 作り付けの実行時ライブラリを全廃
  - 通信はすべてMPIの直接呼び出し
  - その他の必要なライブラリは、コンパイル時に作る。

【例】Cyclic分散で、増分付きループパラメタの変換

```
!hpf$ processors P(0:5)
      real X(0:22)
!hpf$ distribute X(cyclic) onto P

!hpf$ independent
      do I=5,22,4
        X(I)=1.0
      enddo
      end
```




```
do I=?,?,?
  X(I)=1.0
enddo
```

GCDの計算などを含み、
コンパイラでは生成が困難

# 解説資料(2) 翻訳時生成ライブラリ(続き)

fhpfの実際の出力

```
･･･省略）･･･
      IF ((org_cyclic_f(5,4,6,mod(ORG_RANK,6))).NE.-1) THEN
        spmd_startX0 = (5+(org_cyclic_f(5,4,6,mod(ORG_RANK,6)))*4-(mod(O
     &RG_RANK,6)))/6
      ELSE
        spmd_startX0 = 2147483647
      ENDIF
      spmd_endX0 = (28-(mod(ORG_RANK,6)))/6-1
      spmd_stepX0 = 4/(org_gcd_f(6,abs(4)))
      DO i=spmd_startX0,spmd_endX0,spmd_stepX0
        x(i) = 1.0e0
      ENDDO
      END
      FUNCTION org_cyclic_f(i1,i3,pnum,pno)
･･･省略）･･･
      n_max = pnum/(org_gcd_f(pnum,abs(i3)))
･･･省略）･･･
      END
      FUNCTION org_gcd_f(m,n)
･･･省略）･･･
      m2 = m
      n2 = n
 1000 CONTINUE
      k = mod(m2,n2)
      IF (k.NE.0) THEN
        m2 = n2
        n2 = k
        GOTO 1000
      ENDIF
      org_gcd_f = n2
      END
```

翻訳時生成ライブラリ#１

翻訳時生成ライブラリ#２（ユークリッド互除法）

# 内容

- fhpfの特徴とご使用方法
- 新技術のご紹介
- 機能と性能
- ライセンスに関して
- まとめ

# HPF機能サポート範囲

| HPFの指示文 | HPF/VPP5000 | fhpf V1.1.3 |
|---|---|---|
| DISTRIBUTE | ○ | ○ |
| block-cyclic, indirect | × | ○（全種別対応） |
| allocatable配列 | × | × |
| ポインタ・構造体 | × | × |
| 転写的指示 | × | × |
| ALIGN | ○ | ○ |
| allocatable配列 | × | × |
| ポインタ・構造体 | × | × |
| 重複分散 | × | × |
| PROCESSORS | ○ | ○ |
| NUMBER_OF_PROCESSORS使用 | × | ○ V1.1.3から |
| TEMPLATE | ○ | ○ |
| SEQUENCE | ○ | ○ |
| INHERIT | ○ | × |
| INDEPENDENT | ○ | ○ |
| HPF組込み・library | ○（一部制限） | ×（ほぼ未対応） |

’04/9月サポート見通し

’04/9月サポート見通し

# HPF機能サポート範囲（続き）

| HPFの指示文 | HPF/VPP5000 | fhpf V1.1.3 |
|---|---|---|
| EXTRINSIC接頭辞 | ○ | × |
| 明示的引用仕様なしでの他言語objectの結合 | × | ○ |
| REDISTRIBUTE | ○（制限あり） | × |
| REALIGN | × | × |
| RANGE | × | × |
| SHADOW | ○ | ○ |
| フルshadow | ○ | × |
| SUBSET | ○ | ○ |
| ON | ○ | △（Independent-doのbodyのみ可） |
| EXT_HOME節 | ○ | ○ V1.1.3から |
| RESIDENT | ○ | ○ |
| TASK_REGION | ○ | × |
| ASYNCHRONOUS | ○ | ○ |
| REFLECT | ○ | ○ |
| LOCAL | ○ | ○ |
| INDEX_REUSE | ×（機能しない） | ×（機能しない） |

’04/9月サポート見通し

’05/3月サポート見通し

# 性能的な課題

- 入出力文
  - マスタプロセッサだけで処理
  - マスタプロセッサへの／からの通信に改善の余地
- ループ内でのスカラ通信
  - 通信一括化が未実装
    →ASYNCHRONOUS構文のご利用を

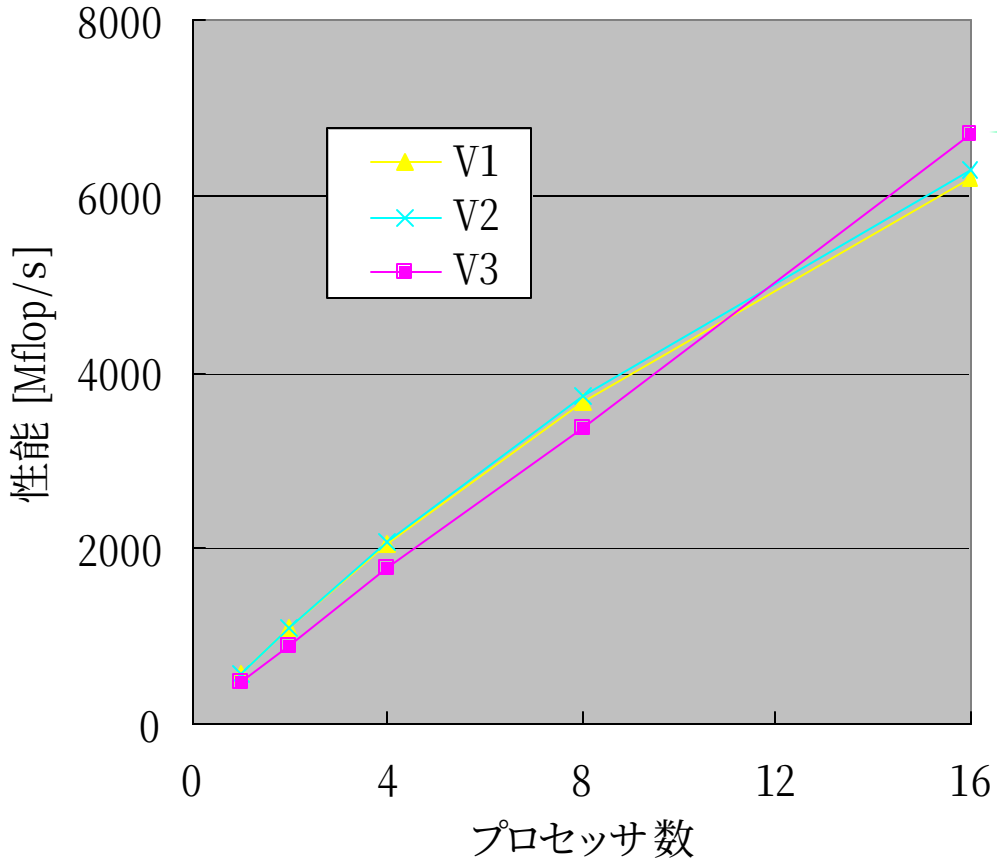
# HimenoベンチマークのHPF化

- 3通り作成

| 版 | 作成者 | 作成方法（所要時間） | コードの特徴 |
|---|---|---|---|
| V1 | VPPF熟知<br>HPF熟知 | ソース：姫野98 VPPF版<br>1) 並列の観点でソース理解（30分）<br>2) HPFに移植（30分）<br>3) 姫野xp流に修正（10分） | •ほとんどのデータは重複<br>•変数pはshadow/reflect |
| V2 | 同上 | ソース：姫野xp Fortran版<br>先にV1を作成し前提知識あり。<br>1) HPFで並列化（2時間） | •全データがblock分割<br>•変数pはshadow/reflect |
| V3 | MPI歴5年<br>HPF初めて | ソース：姫野98 C+MPI版<br>1) データ配置を重複から分散に変更（1週間）<br>2) C+MPI→Fortran+MPI（1日）<br>3) Fortran+MPI→HPF（1日） | •概ねV2と同じ。変数gosaの周辺の計算が違う?<br>•C流の添字の並び順序 |

# 性能評価(1) Himeno-M／SMP

姫野ベンチHPF版、サイズM
PRIMEPOWER HPC2500 1.5GHz (ノード内)
fhpf V1.1.3, Parallelnavi Fortran2.3, MPI 6.1

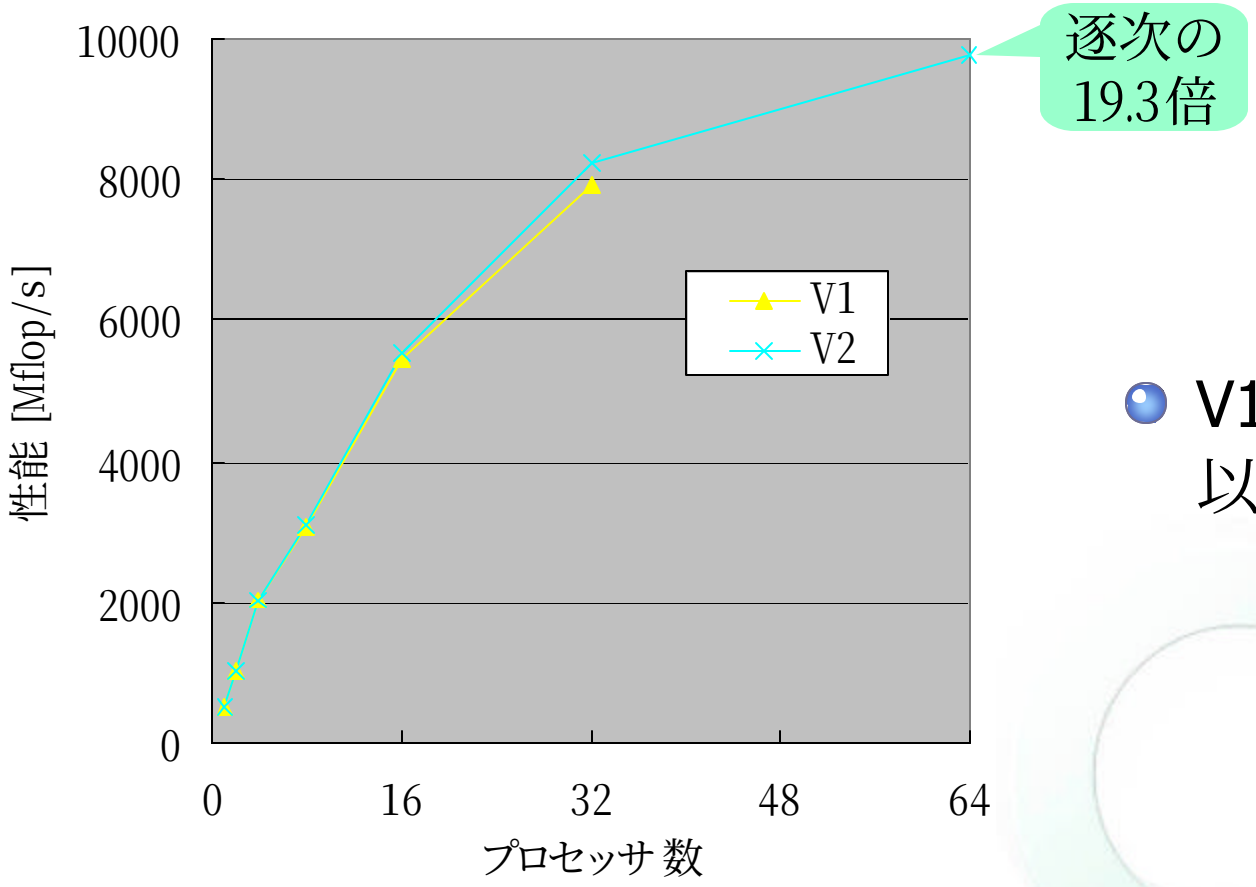



逐次の
14.5倍

- 共有メモリのノード内での実行結果。

# 性能評価(2) Himeno-L／SMP

姫野ベンチHPF版、サイズL
PRIMEPOWER HPC2500 1.5GHz (ノード内)
fhpf V1.1.3, Parallelnavi Fortran2.3, MPI 6.1



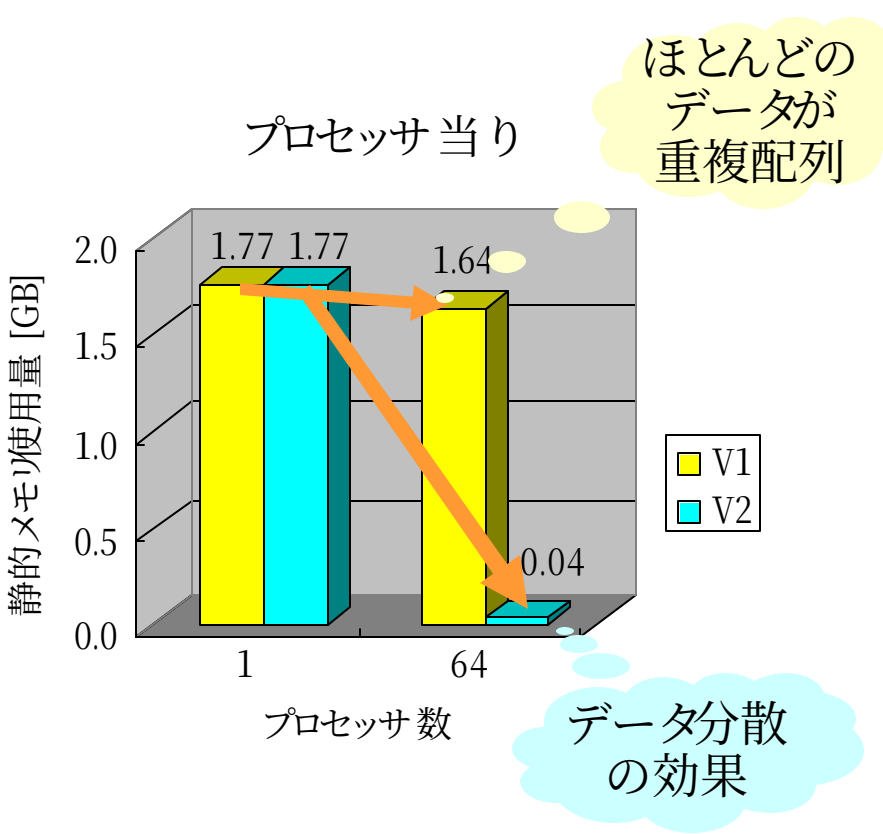

- V1では64プロセッサ以上でメモリ不足

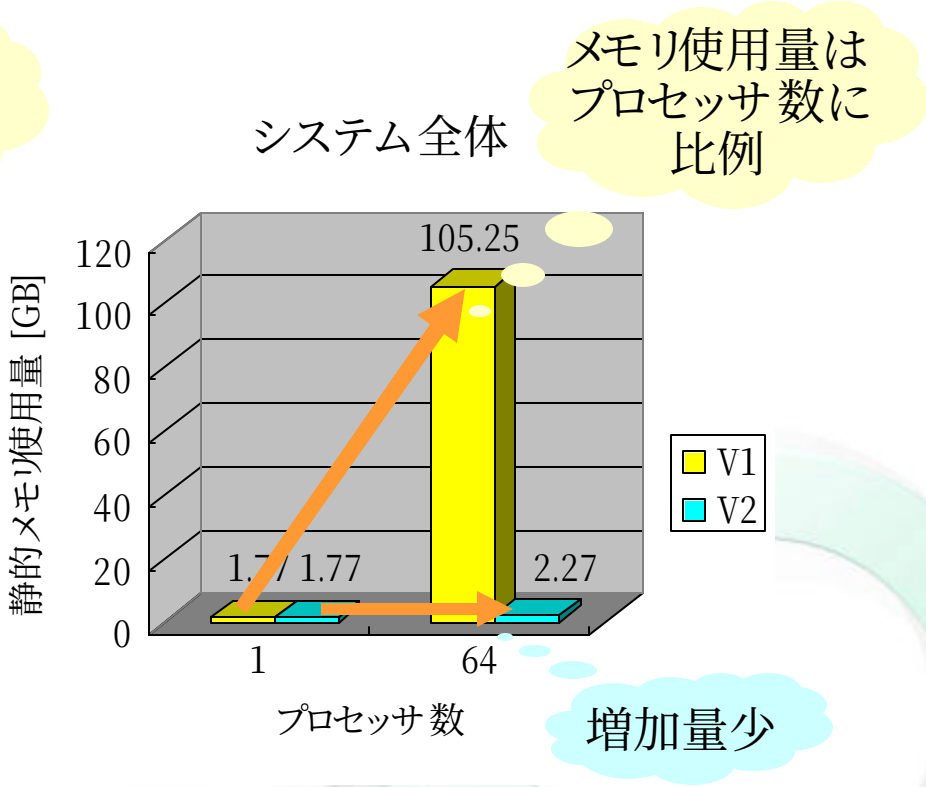
↓

理由は

# 性能評価(2) Himeno-L／SMP 続き)

## 静的なメモリ使用量の比較

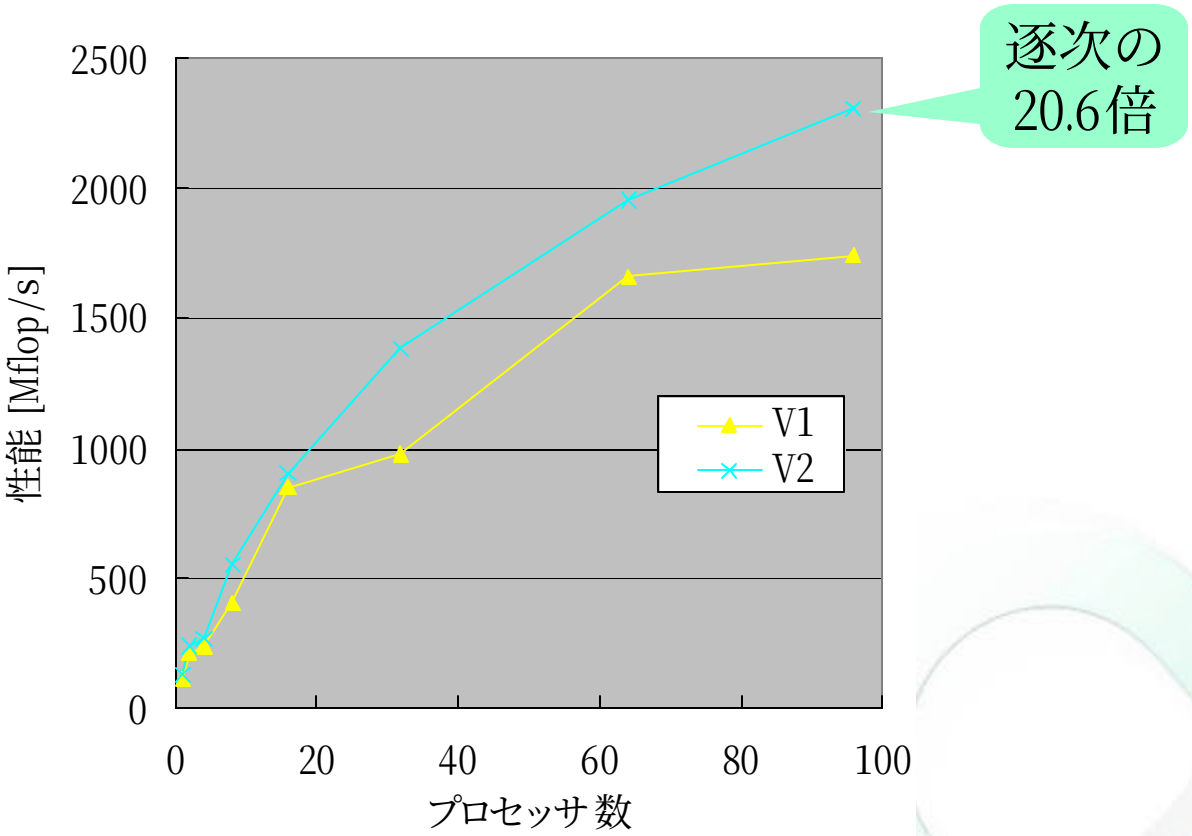



SMPでは、重複配列はできるだけ避ける

# 性能評価(3) ブレードサーバ

姫野ベンチHPF版、サイズM
Linuxサーバ（Mobile Pentium III）
fhpf V1.0, g77, LAM-MPI



# 内容

- fhpfの特徴とご使用方法
- 新技術のご紹介
- 機能と性能
- ライセンスに関して
- まとめ

# 代表的なライセンス形態

| | | ソフトは無償 *1 | 再配布可能 | ソース入手可能 | ソース変更自由 | 備考 |
|---|---|---|---|---|---|---|
| 商用ソフト | | | | | | バイナリもコピー禁止。 |
| 試用版ソフト | | yesまたは条件付き | 条件付き | | | fhpfの現在の形態。 |
| シェアウエア | | 条件付き | yes | | | 利用者は使ってみてから購入または破棄。モラルに依存。 |
| パブリックドメイン | | yes | yes | | | 作者が所有権を放棄。日本と欧州では法的に認められない。 |
| フリーウェア（無料バイナリ） | | yes | yes | | | バイナリのみ配布。リバースエンジニアリングなどの行為は禁止。 |
| 無料ライブラリ | | yes | yes | yes | | 改変はライセンス違反。クラスライブラリ、ヘッダファイルなど。 |
| オープンソース | BSD方式 | yes | yes | yes | yes | ソースは勝手に利用可能。ソース修正は開発元に還元しない。 |
| | GPL *1 | yes | yes | yes | yes | GPLソースを利用したソフトウェア全体がまたGPLとなる。 |
| | LGPL *2 | yes | yes | yes | yes | LGPLソースを利用したソフトウェアは、その利用部分がLGPLとなると共に、いくつかの義務を負う。 |

検討中

*1 配布手数料やサービスは別。例えば、RedHatはLinuxの配布が収入源の一つ。
*2 GNU General Public License。GNUのgccやLinux OS（の大部分）など。
*3 劣等GPL（Lesser GPL）。GNUのライブラリの大部分。

# フリーウェア化に向けて

広くご意見を頂戴したく思います。

- 想定利用者／要件
  - 実アプリの研究開発者、利用者
    - フリーでも高品質であること。
    - 有償サービスとの連携。カスタマイズ、サポートなど。
  - コンパイラ、ソフトウェアの研究者
    - 改造、実験が容易であること。
    - 構成が簡単であること。
  - とりあえず持っていたい人、記念に欲しい人
    - 入手容易であること。

# フリーウェア化に向けて（続き）

- 使用許諾条件の大筋の案
  （内容を約束するものではありません。）
  - 本ソフトは一式無償配布。ただし、
    - Linuxライブラリは、利用者の所有するものをdynamic linkする。
    - 有償モジュールが将来別途提供されるかも。
  - 使用、複製、改変は自由。（改変し）再配布するときは、
    - 著作権とライセンスの表示義務
    - 販売は不可
    - 本ソフトを（改造し）他のソフトへ組み入れる場合も、改変に相当
  - 使用して得た結果を公表する場合、本ソフト名などの表示

# まとめ

- fhpfは、
  - HPFからFortran+MPIを生成する、トランスレータ
- 新機能紹介
  - 正規化　・・・解析が簡単に、後続処理が楽に
  - 翻訳時ライブラリ生成　・・・MPI直接呼び出しを実現
- 機能・性能強化は継続中
  - 動的再マッピング、タスク並列を順次サポート
  - SMP上、Linuxクラスタ上で評価
- フリーウェア化を検討中
  - ご意見を広く募集します。

# 参考文献など


- Hidetoshi Iwashita, Kohichiro Hotta, Sachio Kamiya and Matthijs van Waveren. Toward a Lightweight HPF Compiler, HiWEP2002 in The 4th International Symposium on High Performance Computing (ISHPC), May 2002.
- 中田育男、「コンパイラの構成と最適化」。朝倉書店 (1999)
- 姫野ベンチ (http://w3cic.riken.go.jp/HPC/HimenoBMT/index.html)
- GNU's Not Unix! - GNUプロジェクトとフリーソフトウェア財団 (FSF) (http://www.gnu.org/home.ja.html)
- マイクロソフト、いわゆる「ハロウィン文書」。 (http://www.post1.com/home/hiyori13/freeware/halloween.html)

[5] Jan Sandred著、でびあんぐる監訳、「オープンソースプロジェクトの管理と運営」。オーム社 (2001)