2007.03.02 2版

HPF 推進協議会

http://www.hpfpc.org/

fhpf
ユーザーガイド

プログラミング言語 HPF（High Performance Fortran）および HPF/JA の言語仕様や、HPF に関係する最新の情報は、HPF 推進議会（HPFPC）のホームページ:

http://www.hpfpc.org/

から入手してください。fhpf を紹介しているプラズマ・核融合学会誌 Vol.82 No.9（2006 年）の連載コラムも、上記のサイトから参照できます。

1 ソフトウェアの特徴と構成

1.1 特徴

fhpfは、HPFプログラムを入力として、Fortranプログラムコードを出力する、ソース-to-ソースのコンパイラ（トランスレータ）です。出力コードにはMPI1.1 APIに準拠したMPIライブラリの呼出しが含まれます[1]。

fhpfの出力は、ハードウェアやFortranコンパイラやMPIライブラリの種類に依存しません[2]。任意のFortranコンパイラを使って翻訳し、任意のMPI環境の上で実行することが可能です。

1.2 ソフトウェアの構成

本ソフトウェア（以降、fhpfと呼びます）には、Linux 版と Solaris 版があります。どちらも、以下のファイルから構成されています。

- fhpf（シェルスクリプト）
  全体を制御するシェルスクリプトです。fhpf と fhpf_msg を呼び出します。

---

[1] 標準のMPIでは使用できない四倍精度（16 バイト）実数型や四倍精度（32 バイト）複素数型を取り扱うため、一部MPI1.1 APIにない機能を使用しています。MPICHや富士通のMPIではサポートされている機能です。

[2] LAM/MPIの現在の版では四倍精度実数型や四倍精度複素数型のサポートが不完全であるため、ドライバ制御スクリプトを修正して生成コードを若干変更することをお勧めします。詳しくはインストールガイドの「4.インストール手順」を参照してください。

- flops（バイナリファイル）
  コンパイラ本体です。ドライバ制御スクリプトを参照しながらソースプログラムの翻訳を行います。必要に応じて IM ライブラリを参照します。
- fhpf_msg（シェルスクリプト）
  メッセージデータを参照し、翻訳時のエラーや警告のメッセージを出力します。
- f77.script, f90.script（ドライバ制御スクリプト）
  flops の動作を制御するデータです。
- IM ライブラリ（im ディレクトリ配下）
  翻訳時に必要に応じて Fortran コードに展開されるライブラリです。中間コードの形式で格納されています。
- メッセージデータ（msg ディレクトリ配下）
  メッセージ出力のためのデータが格納されています。
- サンプルプログラム（sampleディレクトリ配下）
  HPFプログラムのサンプルと出力例が格納されています。使用方法は「4サンプルプログラム」を参照してください。

2　翻訳と実行の方法

実行ファイルを作成するには、fhpf による翻訳（HPF 翻訳）と、Fortran コンパイラによる翻訳・結合（Fortran 翻訳）の2段階の作業が必要です。HPF 翻訳の結果は、MPI ライブラリ呼出しを含む Fortran プログラムです。Fortran 翻訳は、通常の MPI プログラムの翻訳と同じです。

作成した実行ファイルは、通常の MPI プログラムと全く同じように実行できます。

2.1 HPF 翻訳

取り扱うプログラムには、FORTRAN77 モードと Fortran90 モードの 2 通りがあります。これらは以下のように識別されます。

<table>
<tr><td colspan="2">オプション-f77 が有効</td><td>FORTRAN77 モード</td></tr>
<tr><td colspan="2">オプション-f90 が有効</td><td>Fortran90 モード</td></tr>
<tr><td rowspan="2">-f77,-f90 とも指定なし</td><td>入力ファイル名のサフィックスが
.f77 または.hpf77</td><td>FORTRAN77 モード</td></tr>
<tr><td>入力ファイル名のサフィックスが
.f または.f90 または.hpf</td><td>Fortran90 モード</td></tr>
</table>

それぞれのモードに対して、fhpf の動作は以下の通りです。

	FORTRAN77 モード	Fortran90 モード
入力プログラム	FORTRAN77 に含まれない書	

の制限	式(引用仕様宣言、形状引継ぎ配列など)が含まれていてはなりません。	
出力プログラム	FORTRAN77 プログラムとなります[3]。	Fortran90 プログラムとなります。Fortran 翻訳には、90 仕様をサポートするコンパイラを使用してください。
出力ファイル名(デフォルト)	入力ファイルと同じ名前で、サフィックスが.mpi.f になります。	入力ファイルと同じ名前で、サフィックスが .mpi.f90 になります。

fhpf コマンドの使用方法は以下の通りです。

fhpf {*オプション*|*入力ファイル名*} ...

入力ファイルは、同時に複数指定することができます。オプションには、以下のものがあります。

- ドライバ制御オプション

-f77|-f90	入力ファイル名のサフィックスに関わりなく、同時翻訳するすべてのプログラムを FORTRAN77 モードまたは Fortran90 モードとします。 -f77 FORTRAN77 モードとします。 -f90 Fortran90 モードとします。
-h|-help	fhpf の使用方法を表示します。実行は行いません。
-o *出力ファイル名*	出力するファイルの名前を指定します。入力ファイルが複数ある場合でも、出力は指定した1ファイルにまとめられます。
-q	ドライバが出すメッセージを抑止します。
-show	ドライバが参照する制御スクリプトを表示します。
-stdout	出力先をファイルでなく標準出力とします。

- コンパイル制御オプション

[3] 出力に自動割付け配列が含まれる場合があります。自動割付け配列はFORTRAN77 の範囲を超えますが、gnu g77 コンパイラではサポートされています。

-Am	入力プログラムがモジュールを使用しているときに指定します。モジュールがファイルを跨いで引用される場合や、同じファイル内でもモジュールの引用が定義よりも前にある場合には、必ず指定が必要です。このオプションを指定すると、カレントディレクトリに <モジュール名>.pmi という名前のファイルが生成されます。
-Fixed|-Free	入力ファイルが固定形式か自由形式かを指定します。デフォルトでは固定形式となります。どちらの場合でも、出力ファイルは固定形式としても自由形式としても解釈可能な Fortran プログラムとなります。
-I *ディレクトリ*	インクルードファイルを検索するディレクトリを指定します。
-w	入力ファイルが固定形式のとき、255 カラム目までの文字を有効と見なします。
-K{ *サブオプション*},...	最適化に関わるサブオプションを指定します。1 つの-K に続けてサブオプションを複数指定する場合には、サブオプションをコンマで区切って空白を空けないで並べます。

- -Kのサブオプション

-indep	ローカルアクセスの自動判定を抑止します。
indep (デフォルト)	INDEPENDENT 指示のない DO ループも、可能な限り並列化されます。(INDEPENDENT 指示の自動化)
noindep	INDEPENDENT 指示のない DO ループは、並列化されません。DO ループの並列化には、INDEPENDENT 指示が必須となります。
on (デフォルト)	ON 指示のない DO ループも、できる限り並列化されます。(ON の自動化)
noon	ON 指示のない DO ループは、並列化されません。並列化したい DO ループの並列化には、INDEPENDENT と ON の両方の指示が必須となります。
local (デフォルト)	LOCAL 指示のないデータの定義と使用について、コンパイル時にローカルアクセス(通信不要)の判定が行われます。(LOCAL 指示の自動化)

nolocal	LOCAL 指示のないデータの定義と使用について、ローカルアクセス(通信不要)の判定はすべて実行時に行われます。

【例 1-1】

HPF プログラムファイル abc.f は HPF 翻訳され、ファイル abc.mpi.f90 に出力されます。

```
% fhpf abc.f
```

【例 1-2】

Fortran 翻訳に FORTRAN77 コンパイラを使用する場合、オプション-f77 を使います。入力ファイル hoge77.f は Fortran90 仕様を含むことはできません。出力ファイル名は hoge77.mpi.f となります。

```
% fhpf -f77 hoge77.f
```

【例 1-3】

HPF プログラムファイル a1.f と a2.f は HPF 翻訳され、結果はファイル a12.f にまとめて出力されます。オプション-o は出力ファイル名を指定します。

```
% fhpf a1.f a2.f -o a12.f
```

【例 1-4】

オプション-stdout は、標準出力への出力を指定します。以下は例 1-2 と同じ効果があります。

```
% fhpf -stdout a1.f a2.f >a12.f
```

2.2 Fortran 翻訳・結合

MPI ライブラリが提供する翻訳コマンドを使用するのが便利です。使用する MPI のドキュメントを参照してください。

HPF 翻訳で生成される Fortran プログラムは、固定長形式でも自由長形式でも翻訳可能です。

結合編集時には、HPF 翻訳した結果のファイルと、逐次 Fortran プログラムファイルやオブジェクトファイルを混在させることもできます。

【例 2-1】

例 1-1 のように作成された Fortran コード abc.mpi.f90 を翻訳し、MPI ライブラリを結合します。出力ファイル名は abc とします。翻訳には mpif90 コマンドを使用します。

オプションについては、使用する MPI や Fortran コンパイラのマニュアルを参照してください。

```
% mpif90 abc.mpi.f90 -o abc
```

【例 2-2】

例 1-2 のように作成された FORTRAN77 コードを翻訳し、MPI ライブラリを結合します。翻訳には mpif77 コマンドを使用し、Fortran コンパイラには/usr/bin/g77 を選択しています。出力ファイル名は通常 a.out となります。オプションについては、使用する MPI や Fortran コンパイラのマニュアルを参照してください。

```
% mpif77 -fc=/usr/bin/g77 hoge77.mpi.f
```

【例 2-3】

富士通 PRIMEPOWER を使って、例 1-1 のように作成された Fortran コード abc.mpi.f90 を翻訳し、MPI ライブラリを結合します。翻訳には mpifrt コマンドを使用し、オプションにより64ビットアドレスモードを選択しています。翻訳時オプションについては、富士通の MPI 使用手引書と Fortran 使用手引書を参照してください。

```
% mpifrt -KV9 -o abc abc.mpi.f90
```

2.3 並列実行ファイルの実行

並列実行ファイルは通常の MPI の実行可能ファイルと同じように実行できます。実行方法の詳細は、実行する計算機で使用する MPI のマニュアルを参照してください。

備考

並列実行ファイルは、同じプログラムを Fortran コンパイラだけで翻訳した実行ファイルと比較して、スタック領域を多く使用する傾向があります。そのため、スタック領域不足による実行時エラーが発生しがちです。並列実行ファイルの実行時には、スタック領域を十分に確保することをお勧めします。

スタック領域の大きさの設定には、limit(1)コマンドを使用します。例えば csh では、以下のようにスタックサイズを 50 Mbyte に設定できます。詳しくは Linux または Solaris のオンラインマニュアルなどを参照してください。

```
% limit stacksize 50M
```

【例 3-1】

mpirun を使って、4 並列の並列実行ファイル a.out を実行する場合。

```
% mpirun -np 4 a.out
```

【例 3-2】

SCore Version 5.0.1 を使って、4 並列の MPI 実行ファイル a.out を実行する場合。この例では、in.dat を標準入力と結合し、out.dat を標準出力と結合しています。

```
% scrun -nodes=4 scatter == ./a.out :=in.dat =:out.dat
```

【例 3-3】

富士通 PRIMEPOWER で 8 並列の並列実行ファイル abc を実行する場合、以下のように記述できます。fhpf で生成した MPI プログラムは、高速な limited モードで実行可能です。

```
mpiexec -mode limited -n 8 abc
```

Parallelnavi 環境の下でバッチ実行を行うには、例えば以下のような NQS スクリプトを用意し、qsub(1)コマンドを使ってジョブキューに投入します。

```
#!/usr/bin/csh
# @$-oi                          # 統計情報を標準出力ファイルへ
# @$-eo                          # 標準エラー出力を標準出力ファイルへ
# @$-q queue8                    # ジョブキューの指定
# @$-lp 1 -lP 8          # スレッド並列なし、8プロセス確保
# @$-lM 100gb                    # メモリ使用量（総量）100Gbyte を確保
# @$-ls 100mb                    # スタック使用量（プロセス当り）100Mbyte を確保

cd $QSUB_WORKDIR

setenv PATH /opt/FSUNf90/bin:/opt/FSUNaprun/bin:/opt/FJSVmpi2/bin
setenv LD_LIBRARY_PATH /opt/FSUNf90/lib:/opt/FJSVmpi2/lib:/opt/FSUNaprun/lib
setenv LD_LIBRARY_PATH_64 /opt/FJSVmpi2/lib/sparcv9

#setenv GMP_INFO_TRANSPORT              # MPI 動作設定（MPI 使用手引書を参照）

echo '*** JOB Start ***'
date
mpiexec -mode limited -n 8 abc    # limited モードで 8 並列実行
date
```

3 HPF プログラミング

3.1 プログラミング上の注意事項

3.1.1 最適化レベルに関して

デフォルトでループの自動並列化を行います(V1.4 以降)。オプション(2.1節)によって自動化を止めて利用者の指示だけに従わせることもできます。

Independent-DO ループ内で変数のシャドウ領域を参照するための ON HOME 指示と LOCAL 指示は、Independent-DO ループの直前に REFLECT 指示文がある場合には省略可能です(V1.4 以降)。

3.1.2 プロセッサ数に関して

DISTRIBUTE 指示文で ONTO 以降を省略すると、実行時に決まるプロセッサ数と同じ大きさを持つ 1 次元プロセッサ配列に対する分散と見なされます。つまり、例えば配列 A の分散に関して、

```
!HPF$ DISTRIBUTE A(*,*,BLOCK)
```

と

```
!HPF$ PROCESSORS P(NUMBER_OF_PROCESSORS())
!HPF$ DISTRIBUTE A(*,*,BLOCK) ONTO P
```

は同じ意味を持ちます。2次元以上のプロセッサ配列に対する分散では、ONTO 以降を省略することはできません。

PROCESSORS 指示文で指定したプロセッサ配列の大きさと、実行時に指定するノード数(mpirun のオプション-np で指定する数値など)は、一致しなければなりません。PROCESSORS 指示文で複数のプロセッサ配列が指定されているとき、それらの次元数や形状は一致する必要はありませんが、大きさは一致しなければなりません。

3.1.3 手続き呼び出しに関して

手続境界での自動再マッピングは、明示的引用仕様(Interface ブロックなど)がある場合に限って行われます。呼出し側とサブプログラムで引数のマッピングが一致しない場合には、明示的引用仕様が必要です。逆に、明示的引用仕様がない場合には、再マッピングによる性能的オーバヘッドが生じないことが保証されます。

原則として、HPF 手続を Independent-DO ループの中で呼び出すことはできませんが、手続き内で入出力や分散配列の参照がない場合に限っては許されます。

Fortran または C の逐次実行手続は、Independent-DO ループの中で呼び出すことができます。HPF プログラムから呼び出される Fortran プログラムを定義するときは、接頭辞 `EXTRINSIC('FORTRAN','LOCAL')`を使用するか、fhpf を通さないで翻訳し結合してください。

3.1.4 その他の制限事項

宣言指示文の出現順序に制限があります。変数名、テンプレート名およびプロセッサ名は、それらが使用されるよりも前に宣言されていなければなりません。つまり、ALIGN 指示文より先にそのターゲットに関する TEMPLATE 指示文、DISTRIBUTE 指示文などがなければならず、DISTRIBUTE 指示文より先にその分散先プロセッサに関する PROCESSORS 指示文がなければなりません。

分散配列を DATA 文などで初期化することはできません。大きなデータはファイルから読ませるなどの方法で回避してください。

同じ翻訳単位内に出現する複数の ASYNCHRONOUS 指示文および ASYNC 接頭辞は、同じ非同期識別子を持つことはできません。

そのほか、HPF/JA1.0 仕様のうち以下の機能は未サポートです（ON 指示構文は V1.4.3 以降サポートされています）。

- 文字型変数および構造体のマッピング
- ALIGN 指示文による変数の複製
- 部分プロセッサに対する分散
- BLOCK 以外の分散に対する SHADOW 指示
- フルシャドウ
- TASK_REGION 指示構文
- 仮引数の転写的（transcriptive）なマッピング指定、INHERIT 指示
- 手続呼出し境界での再マッピングが必要な内部手続き
- HPF 組込み関数、HPF ライブラリ
- REDISTRIBUTE 指示、REALIGN 指示、DYNAMIC 指示
- INDEX_REUSE 指示
- 分散配列を左辺とする代入文での、配列構成子の使用

Fortran 言語仕様に関して、以下の制限があります。

- ASSIGN 文と割当て型 GOTO 文、ENTRY 文、選択戻り指定子
- WHERE 文、WHERE 構文
- 引数キーワード
- 総称指定のある引用仕様宣言

4 サンプルプログラム

本ソフトウェアがインストールされたディレクトリの下に、ディレクトリ sample が展開されます。サンプルプログラムとしてご利用ください。

サンプルプログラムは、姫野ベンチマークプログラム

http://accc.riken.jp/HPC/HimenoBMT/

のhimenoBMTxpをHPFで並列化したものです。

4.1 プログラムについて

ディレクトリ sample の下には、以下の HPF プログラムがあります。

(1) himenoBMTxp_HPFv1.f(インクルードファイルなし)

3次元空間のうち z 軸方向を block 分散するプログラムです。プロセッサ数はコンパイル時に指定する必要はなく、実行時にオプション(mpirun コマンドの-np など)で指定した値となります。

(2) himenoBMTxp_HPFv2.f(インクルードファイルは param_HPFv2.h)

3次元空間のうち z 軸方向を block 分散するプログラムです。プロセッサ数はインクルードファイルを修正して再コンパイルすれば変更することができます。

(3) himenoxp_dim3.hpf(インクルードファイルは param_dim3.h)

x 軸、y 軸、z 軸の3次元方向に block 分散するプログラムです。インクルードファイルの PARAMETER 文を修正して、各次元方向に任意の大きさで分割することができます。3次元方向の分割数を掛け合わせた値が、プロセッサ数となります。

4.2 fhpf のインストール確認の方法

それぞれのプログラムについて、HPF翻訳(2.1節)が正しく行われるか確認してください。翻訳時オプションなしでHPF翻訳した場合に出力される結果を、mpi_codeの下に置いています[4]。

4.3 プログラムの実行

インクルードファイルの最初の PARAMETER 文の選択により、問題規模を選択してください。

himenoxp_dim1.hpfとhimenoxp_dim3.hpfでは、実行するプロセッサ台数に合わせてインクルードファイルを修正する必要があります。このとき、インクルードファイルで定義したプロセッサ数と、実行時オプションで指定したノード数は一致していなければなりません。

サンプルプログラムは、Web で公開されている姫野ベンチマーク(himenoBMTxp)と同様に、ハードウェアの能力や並列度に寄らずほぼ一定の時間(1分程度)で終了する

[4] Linux版でもSolaris版でも結果は同じになります。LAM/MPIを使用するなどの理由でドライバスクリプトf90.scriptを修正している場合には、異なる結果となることがあります。

ようにできています。実行結果として、実行時間（elapsed time）をベースとした性能値（浮動小数点演算の実行頻度）が、Mflop/s の単位で出力されます。

以上