第I部

序論

ここでは、本書の構成の全体について記述する。それは、High Performance Fortran 2.0 版 (第 II 部に記述される) と、HPF 公認拡張仕様 (第 III 部に記述される) に共通の用語と概念を定義し、以下の章のために必要な基礎を与える。

第1章　概要

本書は、High Performance Fortran (HPF) 言語で表現されたプログラムの形式を明示し、その解釈を規定する。HPF は確立された国際標準 Fortran 規格への拡張および変更の集合として設計されている。本書の公表時点では、基準として使われている標準は “Fortran 95” (ISO/IEC 1539:1997) と非公式に呼ばれている。その文書への参照は次のように行なわれる: その文書の 13.11.6 は、F95:13.11.6 として参照される。

本章では、言語の目的と適用範囲を概説し、HPF 言語モデルを紹介し、言語の主要機能を強調し、HPF 1.1 と HPF 2.0 の間の差異を説明し、そして本書の残りの部分の手引きを与える。

1.1　High Performance Fortran の目的と適用範囲

HPF 言語の開発の背景にある主要な目的は:

- データ並列プログラミング (単一スレッド、グローバルな名前空間、疎同期並列計算) のサポート
- 異なったアーキテクチャにまたがる可搬性
- 不均一なメモリアクセスコストをもった並列計算機上で高性能 (ただし、他の計算機上での性能も阻害しない)
- 基準として Fortran 規格 (現在は Fortran 95) を使用
- 他言語 (例えば、C) および他のプログラミングパラダイム (例えば、MPI を使用したメッセージ通信) とのオープンなインタフェースと相互引用性

二次的な目的は:

- 限られた時間内での実装可能性
- Fortran および C の将来の標準化活動への入力の提供
- 進んだ機能を一貫した方法で言語に付加するための発展的な道筋の提供

言語定義の最初の版 HPF 1.0 は 1993 年 5 月に公開された。HPF 1.0 に定義されていた多くの言語機能は今や Fortran 95 言語規格に吸収されている (例えば、単純 `FORALL` 文および構文、`PURE` 手続)。したがって、これらの機能はもはや HPF 2.0 の定義の中では詳細には記述されない。HPF 言語の進展 (1.0、 1.1 と 2.0 版を通して) についての情報と、HPF 2.0 と HPF 1.1 の相違点の列挙は 1.4 節にある。

1.2 HPF 言語モデル

HPF の重要な目的は、多様な並列計算機間にまたがるプログラムの可搬性を達成することである。そのためには、HPF プログラムがすべての計算機上で動作するだけではなく、ある並列計算機上で高い性能をもつ HPF プログラムは、他の並列計算機上でも相応のプロセッサ数で適度に高い性能を達成できなければならない。そうでなければ、ある計算機上で高い性能を達成するために費やされたプログラマの努力は、HPF プログラムを他の計算機に移植した場合に無駄になってしまう。共有メモリ計算機と分散メモリ計算機は異なる低レベル操作 (primitives) を使用するが、これらの計算機上での並列プログラムの性能に影響する基本的な要因に関しては広い類似性が存在する。したがって、ある程度高いレベルの HPF プログラムについて、異なる計算機上で高い効率を達成することは実現可能な目的である。並列プログラムの性能に影響する基本的な要因は、利用可能な並列度、データローカリティの利用、適当なタスク粒度の選択、などである。HPF はこれらの要因に関して、プログラマがコンパイラに指示する機構を提供する。

HPF の最初の版は Fortran 90 の拡張として定義されていた。HPF 2.0 は現在の Fortran 規格 (Fortran 95) の拡張として定義される。HPF の将来の改定版は、ISO で承認されしだい Fortran 規格の進展を包含し、それらと一貫性をもつ予定である。

Fortran を基準にして、HPF 言語機能は次の 4 つに分類できる:

- HPF 指示文
- 新言語構文
- 新ライブラリルーチン
- 言語の変更および制約

HPF 指示文は、コンパイラに対して実装戦略を示唆したり、プログラムについての事実を表明するための構造化された注釈として現れる。正しく使われた場合には、それらは実行される計算の効率にのみ影響を与え、プログラムによって計算される値は変えない。HPF 指示文の形式は、文の先頭にある注釈先頭語を削除することによって、将来の Fortran 規格がこれらの機能を完全な文として言語の中に包含することができるように選ばれている。

いくつかの新言語機能は Fortran 構文および解釈の直接の拡張として定義されている。新 HPF 言語機能は、第一種言語構造であり、プログラムによって計算される結果に直接に影響するという点において HPF 指示文とは異なっている。

計算機能をもつ HPF ライブラリは、高性能計算のために有用であると証明されているルーチンへの標準インタフェースを定義する。これらの追加関数には、マッピング問合せ、ビット操作、配列集計、配列集計拡散、配列累計、それに配列ソート関数がある。

Fortran 95 に対する少数の変更と制約が定義されている。最も重要な制約は、HPF のデータ分散機能と両立しないことから、順序結合と記憶列結合の使用に対して課せられた制約である。しかし、ある HPF 指示文に明示することによって、順序結合と記憶列結合を維持することも可能である。

1.2.1 データマッピング指示文

HPF における並列性の基本モデルは、グローバルな共用アドレス空間をもつ単一スレッドデータ並列実行モデルである。Fortran の配列記述と FORALL 文はデータ並列計算を指定する自然な方法である。加えて、HPF は INDEPENDENT 指示文を備えている。INDEPENDENT 指示文は、ループの繰返しをまたぐ依存関係がなく、したがって並列実行可能であることを表明している。

データローカリティの利用は、単一プロセッサのワークステーションであろうと、ワークステーションのネットワークであろうと、並列計算機であろうと、高性能計算機で高い性能を得るためには決定的に重要である。不均一メモリアクセス (NUMA) 並列計算機上では、プロセッサメモリ間に効果的にデータを分散することは、データ移動のオーバヘッドを減らすために非常に重要である。HPF の基本的な機能の一つはデータマッピングを利用者が指定できる機能である。HPF は、並列計算機を抽象プロセッサの一次元あるいはそれ以上の次元の矩形構成とみなす機能を備えている。プログラマは、異なる配列間で配列要素の相対的な整列および論理プロセッサ格子上への配列の分散を指定することができる。データマッピングは、並列性能を最適化するときにコンパイラの助けとなるが、プログラムの意味には影響を与えない HPF 指示文を使って指定される。以下の簡単な例で説明する。

```
      REAL A(1000,1000)
!HPF$ PROCESSORS procs(4,4)
!HPF$ DISTRIBUTE (BLOCK,BLOCK) ONTO procs :: A
      DO k = 1, num_iter
         FORALL (i=2:999, j=2:999)
           A(i,j) = (A(i,j-1) + A(i-1,j) + A(i,j+1) + A(i+1,j))/4
         END FORALL
      END DO
```

このプログラムは、一つの二次元浮動小数点配列 A を使った単純な Jacobi 緩和法を記述したものである。HPF 指示文は構造化された注釈として現れている。PROCESSORS 指示文は 4×4 のプロセッサの論理格子 proc を指定している。DISTRIBUTE 指示文はコンパイラが配列 A をその各次元に沿って等しい大きさのブロックに分割することを推奨している。この結果、 250×250 の要素を含む 4×4 個のブロックができ、各プロセッサに 1 ブロックずつ割り振られる。PROCESSORS と DISTRIBUTE 指示文の詳細は、後の第 3 章で説明される。

外側の DO k ループは num_iter 回だけ Jacobi 緩和ステップを繰り返す。内側のループは Fortran 95 の FORALL 構文を使っている。それは、ループ本体を i と j に対して 2 から 999 までの範囲のすべての値について実行することを指定している。FORALL では、左辺の変数への代入が行われる以前にすべての繰返しについて (すなわち、i と j に対して 2 から 999 までの範囲のすべての値について) 右辺の式の評価が行われる必要がある。

このプログラムを 16 台のプロセッサからなる分散メモリ計算機上で実行する場合、HPF コンパイラは各プロセッサがグローバルな配列 A の一部をローカルに含むような SPMD コードを生成する。内側の FORALL は並列実行され外側の k ループは逐次実行される。各プロセッサは、他のプロセッサのローカルメモリにマップされている A の部分に含まれる「境界」の

要素を必要とする。HPF コンパイラは、必要なプロセッサ間通信を行う低レベル操作を、生成された SPMD コードに挿入する。グローバルな名前空間をもつ単一スレッドデータ並列モデルでは、プログラマは、並列化とデータ分割のための戦略をより高い抽象度で容易に指定できる。抽象的なグローバルな名前空間から個別のプロセッサのローカルメモリへの (アドレス) 変換や、明示的なプロセッサ間通信の管理などの退屈な低いレベルの処理の詳細はコンパイラに任されている。

次の例はスカラ代入文によって起こる暗黙の通信の例である。この例の目的は、並列実行での通信の必要性に対してデータの分散指定がどう関わっているかを示すことである。説明は必ずしも実際の翻訳の過程を反映しているわけではない。

以下のプログラムを考える:

```
      REAL a(1000), b(1000), c(1000), x(500), y(0:501)
      INTEGER inx(1000)
!HPF$ PROCESSORS procs(10)
!HPF$ DISTRIBUTE (BLOCK) ONTO procs :: a, b, inx
!HPF$ DISTRIBUTE (CYCLIC) ONTO procs :: c
!HPF$ ALIGN x(i) WITH y(i+1)
      ...
      a(i) = b(i)                    ! 代入 1
      x(i) = y(i+1)                  ! 代入 2
      a(i) = c(i)                    ! 代入 3
      a(i) = a(i-1) + a(i) + a(i+1)  ! 代入 4
      c(i) = c(i-1) + c(i) + c(i+1)  ! 代入 5
      x(i) = y(i)                    ! 代入 6
      a(i) = a(inx(i)) + b(inx(i))   ! 代入 7
```

この例では、PROCESSORS 指示文は 10 個のプロセッサの一次元構成を指定している。DISTRIBUTE 指示文は、1 プロセッサ当たり 100 個の連続する要素から成るブロックを構成するように、配列 a、 b および inx を 10 個のプロセッサに分散することをコンパイラに推奨する。配列 c は、procs(1) には c(1), c(11), ..., c(991) 、procs(2) には c(2), c(12), ..., c(992) 、などとなるように各プロセッサに循環的に分散される。配列 x と y のプロセッサへの完全なマッピングは指定されていない。しかし、それらの相対的な整列は、ALIGN 指示文によって示されている。ALIGN 指示文は、コンパイラによって選ばれる y の実際の分散にかかわらず、x(i) と y(i+1) はすべての i の値に対して同じプロセッサに置かれるよう推奨している (y(0) と y(1) は x のどの要素とも整列していない)。PROCESSORS、DISTRIBUTE、および ALIGN 指示文の詳細は第 3 章で説明される。

代入 1 (a(i) = b(i)) について、a と b が同一の分散であることは、すべての i について a(i) と b(i) が同じプロセッサにマップされるべきであることを指定している。したがって、この文の実行ではデータ値のプロセッサ間通信は必要でない。

代入 2 (x(i) = y(i+1)) では本質的に通信は必要ない。この場合、配列のどのような分散に対しても二つの配列の相対的な整列は代入文と一致している。

代入 3 (a(i) = c(i)) は最初の代入と非常に似ているように見えるが、a と c の分散が

異なっているため、通信の必要性についてはまったく異なっている. 配列要素 a(i) と c(i) は i の可能な値のたった 10%の場合しか同一プロセッサにマップされていない。(このことは第 3 章にある BLOCK と CYCLIC の定義からわかる。) 各要素は $\lfloor (\mathtt{i}-1)/100 \rfloor = (\mathtt{i}-1) \bmod 10$ が成立する場合のみ同じプロセッサに置かれている。例えば、$\mathtt{i} = 1$ または $\mathtt{i} = 102$ の場合にはこの代入は本来の通信を含まない (すなわち、a(i) と c(i) の両方とも同じプロセッサにある)。しかし、$\mathtt{i} = 2$ の場合には通信が必要である。

代入 4 (a(i) = a(i-1) + a(i) + a(i+1)) では、配列 a への参照は i の可能な値の 98%の場合に同じプロセッサ上で行われる。これの例外は、$k = 1, 2, \ldots, 9$ に対して $\mathtt{i} = 100 * \mathtt{k}$ となる場合 (a(i) と a(i-1) が procs(k) 上にあり、a(i+1) が procs(k+1) にある場合) と、$k = 1, 2, \ldots, 9$ に対して $\mathtt{i} = 100 * \mathtt{k} + 1$ となる場合 (a(i) と a(i+1) が procs(k+1) 上にあり、a(i-1) が procs(k) にある場合) である。この代入文では各プロセッサ上の「境界」要素についてのみ通信が必要である。

代入 5 (c(i) = c(i-1) + c(i) + c(i+1)) は、表面的には代入 4 と同じに見えるがまったく異なった通信動作をする。c の分散が BLOCK ではなく CYCLIC であることから、c(i)、c(i-1) および c(i+1) の三つはすべての i の値に対して三つの異なったプロセッサにマップされている。したがって、実装方法にかかわらず、この文は右辺の参照のために少なくとも 2 回の通信が必要である。

最後の二つの代入は通信の必要性に関して非常に限られた情報しかもっていない。代入 6 (x(i) = y(i)) では使用可能な情報は x(i) と y(i+1) が同じプロセッサにあるということだけである。これからは x(i) と y(i) の関係についての論理的な帰結は導けない。したがって、さらなる情報が無ければ、この文のために実行時に必要な通信に関しては何も言うことができない。代入 7 (a(i) = a(inx(i)) + b(inx(i))) では、a(inx(i)) と b(inx(i)) が常に同じプロセッサ上にマップされていることが証明できる。同様に、a(i) と inx(i) が一緒にマップされることを簡単に推論できる。しかし、inx に格納されている値の知識なしには a(i) と a(inx(i)) の関係は分からない。同様に、a(i) と b(inx(i)) の関係も分からない。

1.3 HPF 2.0 言語機能の概要

本書で定義される言語は、二つの主要部分から成る:

- HPF 2.0 言語 (第 II 部)
- HPF 2.0 公認拡張仕様 (第 III 部)

HPF 2.0 言語は言語仕様が公開されてから 1 年以内に実装可能と思われる機能を含んでいる。これらは、基本データ分散機能、データ並列機能、組込みおよびライブラリルーチン、それに外来機構を含んでいる。公認拡張仕様は、特定の要求を満足するが、初期のコンパイラの実装ではサポートされそうにない進んだ機能を含んでいる。

1.3.1 HPF 2.0 言語機能

1.3.1.1 データ分散機能 (第 3 章 と 第 4 章)

大抵の並列アーキテクチャおよび逐次アーキテクチャでは、データのアクセスにローカリティがある場合に最も高い性能を発揮する。Fortran 規格に暗黙的に含まれている記憶順序の逐

次性は、アーキテクチャが要求するローカリティとしばしば相反する。これを避けるために、HPF では、データの相互位置付け (ALIGN) と、メモリ域または抽象プロセッサ間でのデータの分割 (DISTRIBUTE) を記述する機能を含んでいる。コンパイラは、すべてのデータはプログラム内の任意の時点で単一の値をもつという意味的な制約に従う限り、データのメモリ割付けを改善するためにこれらの記述を利用することができる。第 4 章では、マッピング機能が副プログラム境界を越えてどのように作用するかを定義する。

HPF の目的の一つは Fortran と互換性を保つことであるが、Fortran の順序結合と記憶列結合を完全に維持することは、HPF の中でデータの分散を通して高性能を得るという目的と両立しない。第 3 章 と 第 4 章では記憶列結合と順序結合に関係する制約と指示文を説明する。

1.3.1.2 データ並列実行機能 (第 5 章)

並列計算を明示的に表現するために、 HPF は INDEPENDENT 指示文を定義している。INDEPENDENT 指示文はプログラムの特定部分にある文が逐次的依存性を示さないことを表明する。それは、正しく使われた場合にはその部分の意味を変えず、最適化のためにより多くの情報を言語処理系に与える可能性がある。INDEPENDENT 指示文の REDUCTION 節は、交換則と結合則が成立する演算によって更新される変数を識別するために使用される。REDUCTION 節は、変数の更新の累算が順序に依存しないようなループの場合に、集計演算がもつ並列性を利用可能にする。

1.3.1.3 外来プログラム単位 (第 6 章)

HPF は高レベル機種独立言語として設計されていることから、直接表現するのが困難または不可能な操作が存在する。例えば、あるアプリケーションではある機種上で高度にチューニングされたシストリック通信が役に立つかもしれない。HPF のグローバルなアドレス空間ではこれをうまく表現できない。HPF では、外来機構を使って、明示的なメッセージ通信サブルーチンライブラリや他の言語 (例えば C) での記述のように、他のパラダイムで記述された手続と容易にインタフェースを取ることができる。

1.3.1.4 組込み関数と標準ライブラリ (第 7 章)

大規模並列計算機での経験から、並列アルゴリズムの設計に有用な多くの基本操作が知られている。Fortran の組込み配列関数はこれらのいくつかに応えている。HPF では、組込み関数および標準ライブラリ関数としていくつかの並列操作を言語定義に追加している。さらに、並列実行の制御に有用ないくつかのシステム問合せ関数も備えられている。

1.3.2 HPF 2.0 公認拡張仕様

1.3.2.1 データマッピングの拡張機能 (第 8 章)

拡張されたマッピング機能は、実行時におけるデータの動的な再整列と再分散 (REALIGN、REDISTRIBUTE、 DYMNAMIC 指示文)、プロセッサの部分集合へのデータマッピング、ポインタと構造型成分のマッピング、およびデータの不規則分散のサポート (GEN_BLOCK、 INDIRECT

分散) を包含し、データマッピングをより自由に制御することを可能にしている。さらに、配列が取り得る可能な分散の範囲の情報 (RANGE 指示文) や、ステンシルを使った隣接計算に含まれる配列で使用されるバッファの大きさの情報 (SHADOW) を、プログラマがコンパイラに与えることができる機構が定義されている。

1.3.2.2 データ並列とタスク並列の拡張機能 (第 9 章)

ON 指示文によって計算の分割を明示的に指定できる。計算を実行するように推奨するプロセッサは、プロセッサ構成の明示的に指定された部分集合として、または間接的に、データ実体またはテンプレートがマップされているプロセッサの集合として、のどちらかで指定できる。

コンパイラが効率的なコードを生成するのを助けるために、プログラマによって ON 指示文と共に使われる RESIDENT 指示文が定義されている。RESIDENT 指示文は、ON 指示文の有効範囲内の指定された実体へのすべてのアクセスが実行中のプロセッサにローカルに閉じることを表明するために使用される。TASK_REGION 指示文は、重なりのない部分プロセッサの間でプログラムの異なった部分を同時実行することを利用者が指定できるようにする。

1.3.2.3 非同期入出力の拡張機能 (第 10 章)

入出力と計算の同時実行を可能にするために、書式なしデータに対する非同期直接READ/WRITE のための拡張が定義されている。非同期入出力のために、Fortran の READ/WRITE 文に非ブロッキング実行を指定する入出力制御パラメタが追加され、新しい文 (WAIT) が導入されている。

1.3.2.4 組込みおよびライブラリ手続の拡張 (第 12 章)

HPF 組込み関数およびライブラリルーチンに対する公認拡張仕様は、ほとんどがマッピング問合せ手続に関係している。いくつかの新しい問合せルーチンが定義されている。HPF 2.0 言語で定義されている他のルーチンは、部分プロセッサへのマッピング、GEN_BLOCK、 INDIRECT、DYMNAMIC 分散などの拡張マッピング機能に関する問合せを行えるように拡張されている。Fortran の TRANSPOSE 組込み関数の汎用化も定義されている。

1.3.2.5 HPF 外来仕様の公認拡張仕様 (第 11 章)

第 11 章には、公認 HPF 2.0 拡張機能として多くの特定の外来インタフェースが定義されている。これらは、異なった並列処理モデルへのインタフェース (SPMD 並列処理のための LOCAL や単一プロセスでの逐次実行のための SERIAL) はもちろん、他言語 (例えば、C および FORTRAN 77) との相互引用性を容易にするインタフェースも含んでいる。外来機能において有用なライブラリルーチンは 11.7 節に定義されている。HPF フォーラムによって公式に認知されているがフォーラムによって定義または保守されていない付加的な外来インタフェースは、附属書 F と G に含まれている。このような外来インタフェースの公式認知のための方針と方法は附属書 E に記述されている。

1.4 HPF 1.1 からの変更点

HPF 2.0 は、様々な点で HPF 1.1 と異なっている:

言語の再分割: 新しい文書は、HPF 2.0 と公認拡張仕様の二つの構成要素を記述している。HPF 2.0 言語は、幅広く比較的迅速に実装されることが期待される。公認拡張仕様は、HPF 2.0 の一部ではないが、利用者の要求に応じて、コンパイラ技術の成熟に伴って、将来の実装に含まれるかも知れない。

Fortran 規格に採用された機能: Fortran 90 ではなく Fortran が拡張のための基準言語として定義される。このことは、HPF は 1995 年改訂版 Fortran に追加されたすべての機能を含むことを意味する。この改訂版では HPF 1.1 のいくつかの機能は Fortran 規格の一部になった。したがって、Fortran に対する HPF 拡張としては、もはや現れない。

HPF 2.0 で除去または制約された機能: 現在までに実装されなかった HPF 1.1 のいくつかの機能は、除去することによって得られる単純さが機能の有効性を上回ることが経験により示されたため、言語から除去された。

HPF サブセットの削除: HPF 1.1 とは違って、HPF 2.0 はより早期の実装を目的とした推奨最小サブセット (すなわち、HPF サブセット) をもたない。ただし、元の HPF 1.1 サブセットは附属書に加えておく。

公認拡張仕様に移された機能: いくつかの言語機能は HPF 1.1 から公認拡張仕様の分類に移された。

HPF 2.0 の新機能: いくつかの新機能が基準言語に追加された。

新公認拡張仕様: 多くの最新機能が言語への公認拡張仕様として定義された。

外部機関で保持される認知された HPF 外来仕様: 最後に、本書では、HPF 関連 `EXTRINSIC` インタフェースという新しい範疇を認めている。そしてそれは、そのようなインタフェースのための適当な基準を満たすと承認されたが、公認拡張仕様には含まれない。そのようなインタフェースの内容の責任は、HPF フォーラムではなくそれを提案した機関にあるとされている。

これらの項目の各々は以下の節にまとめられている。

1.4.1 言語の分割

HPF フォーラムはしばしば相反する二つの重要な目的をもっていた:

- 利用者が要求する進んだ言語能力を供給すること。
- ベンダーによるコンパイラの早期開発を考慮すること。

両方の目的を満足するために、我々は言語定義を二つの部分に分けるという妥協を行った。HPF 2.0 は HPF 1.1 に非常によく似ており、本書が出てからだいたい 1 年以内に多くのベンダーによって効率的に実装されることが期待されている。実装のためにさらに多くの努力が

必要な進んだ機能は公認拡張仕様として集められている。実装者はできる限り迅速にこれらの機能をサポートするよう奨励される。また、利用者はかれらの要請をベンダーに知らせることによって、この過程を促進するよう奨励される。

1.4.2 Fortran 規格に採用された機能

HPF 1.1 の一部であった次の機能は、ISO Fortran の一部となったために本書からは削除された:

- 単純 FORALL 文および構文
- 手続の PURE 属性
- MINLOC と MAXLOC 組込み関数に省略可能な引数 DIM を追加するという拡張

1.4.3 HPF 2.0 で除去または制約された機能

次の機能は言語から削除された:

- 順序的配列は明示的にマップされることはない。
- 分散されたデータの再分散が必要な手続の呼出しでは、手続は明示的引用仕様をもたなければならない。
- INHERIT 指示文の扱いが、INHERIT と DISTRIBUTE の両方を同時に指定することができないように単純化された。
- ポインタの扱いが単純化された。

1.4.4 公認拡張仕様に移された機能

DYNAMIC 属性、REDISTRIBUTE 文、REALIGN 文が公認拡張仕様に移された。

1.4.5 HPF 2.0 の新機能

次の新しい機能が HPF 2.0 に導入された。

- INDEPENDENT ループの REDUCTION 節
- 新 HPF_LIBRARY 手続である SORT_DOWN と SORT_UP

1.4.6 新公認拡張仕様

公認拡張仕様は HPF 1.1 にはない次の機能を含んでいる:

- **部分プロセッサ**への実体のマッピング
- ポインタと構造型成分の明示的マッピング
- 新分散形式: GEN_BLOCK と INDIRECT

- 新指示文: RANGE、 SHADOW、 ON、 RESIDENT、TASK_REGION
- 追加の組込み手続: ACTIVE_NUM_PROCS、ACTIVE_PROCS_SHAPE、および汎用 TRANSPOSE 組込み関数
- 新 HPF_LIBRARY 手続: HPF_MAP_ARRAY と HPF_NUMBER_MAPPED。 HPF_ALIGNMENT、HPF_DISTRIBUTION および HPF_TEMPLATE の改訂
- 新しい WAIT 文および Fortran の READ/WRITE 文への入出力制御パラメタ追加による非同期入出力のサポート
- C および FORTRAN77 との相互引用性をサポートするための EXTRINSIC 機能の拡張

1.4.7 外部機関で保持される認知された HPF 外来仕様

本書では、二つの外部機関で保持される HPF 外来インタフェースが認知されている。

- HPF_CRAFT: SPMD パラダイムを HPF 機能に装備する。
- Fortran 77 ローカルライブラリ: Fortran 77 手続をローカルモードで呼び出すライブラリを定義している。