J. Plasma Fusion Res. Vol.82, No.8 (2006) 534-537

連載コラム High Performance Fortran で並列計算を始めよう

1．HPF とは？

妹尾義樹，渡邉國彦[1)]
NEC インターネットシステム研究所，[1)]地球シミュレータセンター

（原稿受付：2006年5月19日）

1.1 はじめに

最近の高性能コンピュータシステムは，地球シミュレータから，PC クラスタにいたるまで，ほとんどが多数のプロセッサを結合した並列計算機です．しかも，後述するように分散メモリ構成を採用しているため，プログラムを並列化して高速性能を得るには大変な労力がかかります．HPF（High Performance Fortran）は，これを簡単にするために開発されたプログラミング言語です．通常の Fortran あるいは Fortran90言語で記述されたプログラムに HPF で定義された指示文を付け加えることで，分散メモリ構成の並列計算機で簡単に高い性能が得られるように設計されています．

HPF は利用者にとってプログラミングが簡単ですが，その分コンパイラには並列化についての高度な解析技術が要求されます．世界の一線の研究者やユーザが集まった HPF フォーラムで1992年に言語仕様が決められ[1]，日米欧で競ってコンパイラが開発されましたが，なかなか高性能のものができず，欧米では2000年を境に HPF の実用化をあきらめざるを得ない状況になっています．一方，日本ではもともとベクトルプロセッサ向けの自動ベクトル化コンパイラを世界で始めて実用化した高い技術力を有しており，この蓄積を活用して NEC と富士通が並列ベクトルマシンや地球シミュレータ，PC クラスタ向けの高性能な実用コンパイラの開発に成功しています．2002年には SC2002国際会議において，地球シミュレータを用いた HPF による高速計算の研究成果が Gordon Bell Award の言語賞を受賞しました[2]．1997年から日本の先端ユーザとメーカが共同で行った JAHPF（Japan Association for High Performance Fortran）における HPF/JA 拡張仕様の策定[3]，そして2001年から開始された HPFPC（HPF 推進協議会：http://www.hpfpc.org/）の活動も，この実用化に大きく貢献しています．

本シリーズでは，6 回にわたる連載で，HPF を用いた並列プログラミングについて，初級から上級まで，実際のプログラミング事例を参照しながらわかりやすく解説したいと思います．

1.2 分散メモリと共有メモリにおける並列プログラミング

超高速計算のための分散メモリシステム上の並列プログラミングなんていうと，何だか難しそうですが，並列プログラミングそのものの考え方は非常に簡単です．例で説明するとわかりやすいので，複数の板前さんが協力して，にぎり寿司を作るケースを考えてみましょう．簡単のために1人前の上にぎりを作るのに，(1)ネタの処理，(2)寿司を握る作業，(3)盛り付けが必要だとします．

共有メモリシステム上の並列処理はいわば，大きなまな板を複数の板前さんが共有して作業することに相当します．

Fig. 1 に示すように，1つのまな板の上に並べられた食材を処理して8人前の上にぎりを作ります．まな板の上の

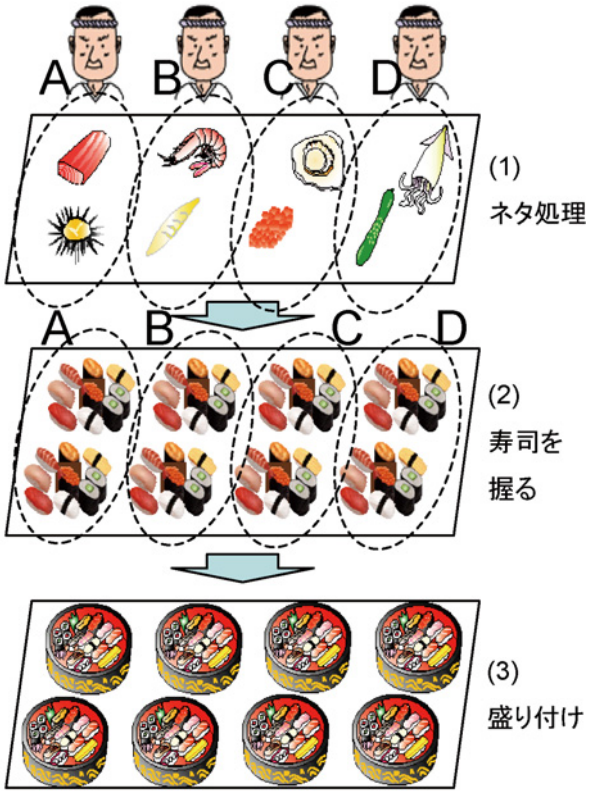

Fig. 1　共有メモリ（1つのまな板）の並列処理の例．

Let Us Start Parallel Processing Using High Performance Fortran ! 1. What is HPF ?
SEO Yoshiki and WATANABE Kunihiko

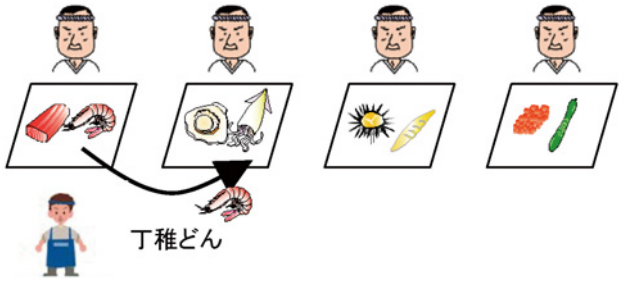

Fig. 2　個別まな板（分散メモリシステム）の例.

すべての食材はそれぞれの板前さんが手を伸ばして取ることができます．食材をどのように配置するかは気にせずに，どの板前さんがどの作業を分担するかだけを決めておけば，並列に作業を行なうことができます．この作業分担のことを「計算マッピング」と言います．たとえば，図のように 4 人が目の前にあるネタの処理を行なった後に，すべての種類の寿司を 2 人前分ずつ握って，それらを盛り付けてもいいし，あるいは寿司を握る作業について，A さんは巻き物が得意だから，A さんはかっぱ巻きとイクラを担当するという風に種類ごとに分担して握ることも考えられます．いずれにせよ，手を伸ばせばすべての食材を取れるので，ある板前が処理したネタを他の板前が使うとしても，特に何も考える必要はありません．共有メモリシステム上の並列処理は，このように作業分担だけを指定することでプログラミングできます．

では，今度は分散メモリの例を考えてみましょう．分散メモリは，Fig. 2のように，それぞれのプロセッサが独自のメモリ（まな板）を持つシステムです．それぞれの板前（プロセッサ）は自分のまな板の上のものしか処理できません．また，すべての食材やにぎった寿司などを 1 つのまな板に置くことはできません．仮に容量的に置けたとしても，1 人の板前しか処理できないことになります．そこで，食材（データ）を分割して配置する必要があります．この分割と配置を「データマッピング」と呼びます．この例では，2

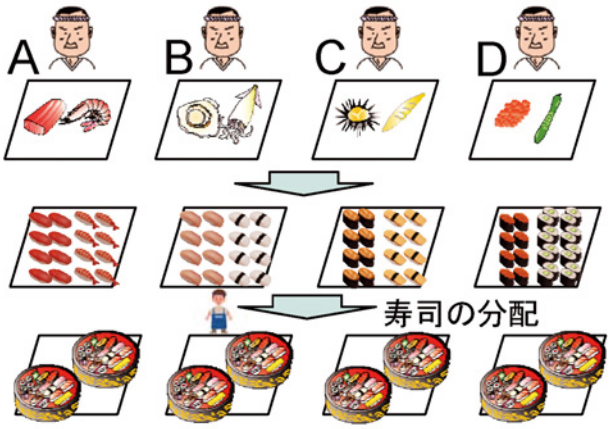

Fig. 3　個別まな板（分散メモリシステム）の並列処理の例（各板前が別々のネタを握る場合）.

Fig. 4　個別まな板（分散メモリシステム）の並列処理の例（全員で同じネタを握る場合）.

種類ずつのネタがそれぞれのまな板に分割配置されています．もし，他の板前のまな板の食材（この例ではエビ）を処理しようとすると，丁稚どんに頼んで運んでもらう必要があります．

この例では，並列処理を Fig. 3 に示すように行うことができます．板前はそれぞれが自分のまな板にある 2 種類ずつのネタを処理し，そのネタで寿司を握り，握った寿司を丁稚どんにたのんで分配してもらい[1]，これをそれぞれが盛り付けて完成となります

しかし，ここで仮に，カッパ巻きとイクラをつくる手間が他の 2 倍かかるとします．すると，板前 D が作業を終える前に，板前 A，B，C の 3 人は処理を終え，遊んでしまうことになります．これでは効率があがりません．そこで，処理の分担（計算マッピング）を変更して，Fig. 4 のようにすべての板前が同じネタを握るようにすることで，より早く作業を終えることができます．この場合，寿司ネタを処理したら，その材料を丁稚どんに頼んで 4 人のまな板に配ってもらい，処理することになります．握った寿司は，すでに 2 人前ずつがそれぞれのまな板にできているので，盛り付けの前にはデータの分配は必要ありません．

分散メモリシステムでは，上記の丁稚どんに相当するのが通信ライブラリと呼ばれるものです．MPI（Message Passing Interface）[4]は，丁稚どんにデータの移動を頼むインタフェースを規定しており，これを用いてユーザが仕事の分配もネタのやりとりも，すべてを直接記述する方法が MPI プログラミングです．

1.3　HPF プログラミング

Fig. 3, Fig. 4 で説明したとおり，分散メモリにおいては，並列処理を行なうために，データの分散配置（データマッピング）と処理の分担（計算マッピング）の両方を決める必要があります．また，分散配置を変更するためには通信ライブラリを用いたデータ転送を行う必要があります．

HPF 言語の基本的な考え方は，データマッピングをユー

[1] 握った寿司を 2 つずつ他のまな板に移す作業．実際には，移動先にデータを保持する作業領域（配列）をそれぞれ確保して，ここにデータをコピーする操作になります．

ザがプログラムで明示的に指定し，計算マッピングやデータ転送の処理はコンパイラ（処理系）に任せようというものです．Fig. 3，Fig. 4を比べてみていただきたいのですが，違いは図の中段の握った寿司の配置だけです．Fig. 4 では，元々，板前が何を握るかという処理分担を変更したかったのですが，自分のまな板の上に作るべき寿司は，自分が作るということにすれば，寿司の配置（データマッピング）を決めることで，作業分担も自動的に決まることになります．この，「自分のまな板に作るべき寿司は自分が作る」という決まりを「オーナーコンピュートルール（Owner Computes Rule）」と言います．プログラムの用語で説明すると，「代入文の左辺のデータが割り当てられたプロセッサが，その実行を分担する」ということになります．ここで注意すべきは，左辺のデータが割り当てられたプロセッサと右辺のそれが一致するとは限りません．Fig. 4 の例ではネタの配置が，握る寿司の配置と整合しませんよね．この場合，データの分配のための通信が必要になりますが，これはコンパイラが自動的に処理してくれます．

1.4 データマッピングの指定

HPF は，Fortran90言語を基本としており，これに主にデータマッピングのための指示文を追加することでプログラムします．指示文は Fortran90の言語仕様ではコメントとして解釈されますので，並列実行を考えなければプログラムをそのまま Fortran90コンパイラで処理して逐次実行することもできます．1.2章の寿司の例について，どのようにデータのマッピングを指定するか説明しましょう．

Fig. 5に寿司ネタ配列 A（8）のデータマッピングを指定する様子を示します．processors 指示文は対象となる分散メモリの構成を指示するもので，対象となるまな板の指定ですね．この場合には 1 次元の 4 要素を持つ構成が指定されています．最初の！hpf＄は HPF の指示文記述であることを示します．そして distribute 指示文でデータマッピングを指定します．A（block）というのは 1 次元配列を block 分割してマッピングする指定であり，8 要素を 4 つに等分して分配します．block 分割の他には cyclic 分割というものがあり，仮に A（cyclic）と指定されていれば Fig. 6 のようにトランプ札を 4 人に配るような分配になります．

次に握った寿司のデータマッピングについて見てみましょう．こちらのデータは 2 次元配列 B（8, 8）として Fig. 7 のように宣言されているとします．縦方向が一次元目で横方向が 2 次元目とします．

これを Fig. 3 の中段にあるようなデータマッピングにするには 2 次元目で block 分割します．B（＊, block）の 1 次元目の＊はその次元を分割しないという意味です．この例では 2 次元目で等分割して，それぞれのまな板に 2 種類の 8 つの寿司が並べられます．これを Fig. 4 の中段に示すようなデータマッピングにしたいならば，B を 1 次元目で block 分割して B（block, ＊）とすればよいでしょう．

基本的に HPF では，通常の Fortran90で記述されたプログラムの並列化は，主要データ（寿司ネタ，握った寿司，盛り付けた寿司）について，Fig. 5，Fig. 7 に示すような distribute 指示文によってデータマッピングを指定するだけで行なうことができます．それぞれの板前に Fig. 3 のように別の種類の寿司を握らせるか，Fig. 4 のように 2 人前のすべての種類を握らせるかは配列 B の block 分割を 2 次元目にするか 1 次元目にするかでプログラマが選択できます．それ以外の計算マッピングやネタを分配するための通信処理はすべてコンパイラが自動的に行なってくれます．

1.5 むすび

今回は，HPF についての連載の第 1 回目として，HPF 言語の概要と基本的考え方について，寿司の例でわかりや

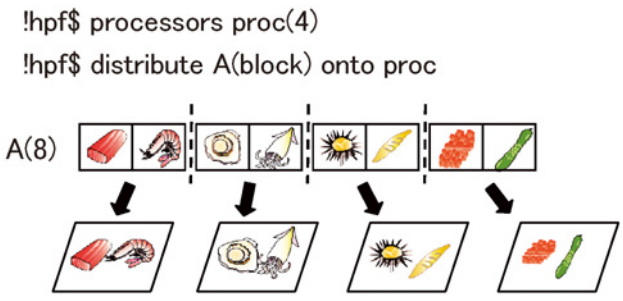

Fig. 5 寿司ネタの block 分割による distribute の例.

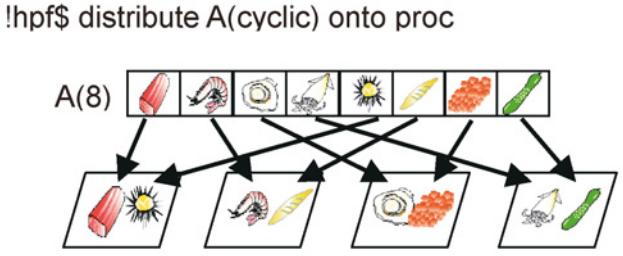

Fig. 6 寿司ネタの cyclic 分割による distribute の例.

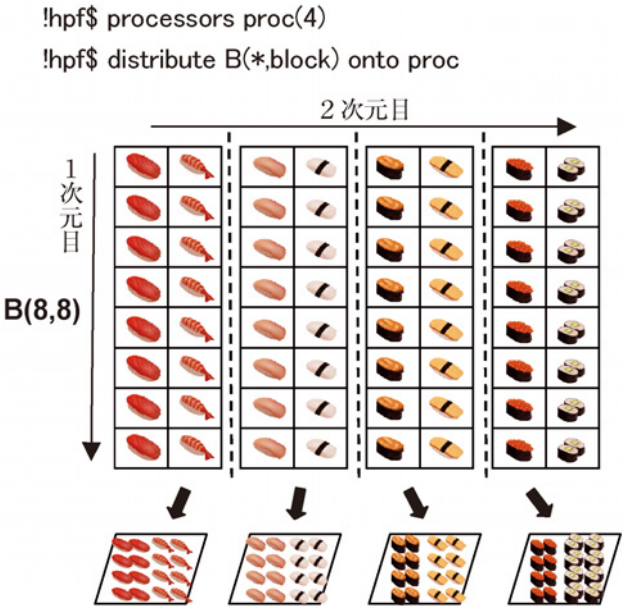

Fig. 7 2 次元構成の握った寿司データと，2 次元目 block 分割による distribute の例.

すく説明しました．いよいよ次号からプログラミングや，プログラムの高速化手法など，より具体的に解説したいと思います．ご期待下さい．

参考文献

［1］High Performance Fortran Forum, High Performance Fortran Language Specification, January 1997.

［2］H. Sakagami, H. Murai, Y. Seo and M. Yokokawa, 14.9 TFLOPS Three-Dimensional Fluid Simulation for Fusion Science with HPF on the Earth Simulator, in *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing* (Baltimore, Maryland), IEEE Computer Society Press., 2002.

［3］Y. Seo, H. Iwashita, H. Ohta, and H. Sakagami, HPF/JA: Extensions of High Performance Fortran for Accelerating Real-world Applications, Concurrency and Computation: Practice and Experience, Vol. 14, 555 (2002).

［4］Message Passing Interface Forum, MPI: A Message Passing Interface Standard, June 1995. (http://www-unix.mcs.anl.gov/mpi/).

妹尾義樹（せおよしき）

1961年生まれ．1986年京都大学工学研究科情報工学専攻修士課程修了．同年 NEC 入社．以来研究開発部門において，スーパーコンピュータの研究開発に従事．特に並列処理アーキテクチャ，分散メモリマシンのための並列化コンパイラ，並列アルゴリズム，グリッドコンピューティングに興味を持つ．現在インターネットシステム研究所総括マネージャー．1994年から1995年まで米国 Rice 大学 CRPC 客員研究員．工学博士．1988年情報処理学会論文賞，2002年 Gordon Bell Award 受賞．

渡邉國彦（わたなべくにひこ）

京都大学理学部，名古屋大学大学院理学研究科を修了後，名古屋大学プラズマ研究所，UCLA，コンサルタント会社，広島大学，核融合科学研究所を経て，昨年10月に地球シミュレータセンター　シミュレーション理工学研究領域長に就任．様々なプラズマにおけるシミュレーション研究を行ってきた．