4F－2

ワークステーション・クラスタにおける動的なデータ交信／実行管理方法[1]

大澤　暁，　小松　秀昭[2]
日本アイ・ビー・エム㈱　東京基礎研究所

1．はじめに

近年では高性能なRISCチップを搭載したワークステーションが安価に購入できるようになり、以前は大型機で行われていた科学技術計算もワークステーション上で実行できるようになってきた。またHPF(High Performance Fortran)[1][2]などのデータパラレル向き分散処理記述言語も開発されつつあり、今後はワークステーションを複数台接続した分散システム上でも大規模な科学技術計算を実行するようになるものと思われる。

このようなシステムを想定した場合、各処理装置間に分散保持しているデータを必要に応じて送受信する必要が生じるが、これに伴うオーバヘッドをいかに小さくするかが並列化効率を考える上でのひとつの重要なキーとなる。そこで本研究ではワークステーション・クラスタ上でのデータパラレル向き分散処理記述言語の処理系を想定し、アプリケーション・プログラムを実行する上で必要なデータ交信の順序や関連する計算の実行順序を動的に制御し、またデータ交信と計算の実行を重複させることにより並列化効率を高める方法を提案する。そしてこの方法をHPFの実行時ライブラリの一部として実装したので、本稿ではその概要を報告する。

2．分散処理とデータ交信

分散システム上においてアプリケーション・プログラムを並列実行する場合、処理装置間のデータ交信に伴うオーバヘッドを小さくするために主に次のような方法がとられる。

(1)　言語処理系などがデータの交信量、相手などを解析して静的スケジューリングを行い、相当する実行コードを実行可能モジュール内に作成する。

(2)　言語処理系などが交信相手などの交信データに関する情報を表に作成し、実行時ライブラリが実行時にこの表を参照しながら動的に管理する。

(1)の方法は、静的スケジューリングのため実行時の制御は比較的簡単になるが、通信の相手方が実行時にしか決まらない場合などに対処できない。また種々の要因により送信側と対応する受信側で時間的なずれが生じる場合に、オーバヘッドが大きくなるおそれがある。

一方(2)の方法を用いた場合、(1)による場合の欠点は克服されるものの、実行時ライブラリの処理が繁雑化するという欠点がある。本研究では効率的にプログラムを実行するという観点にたち、データ交信と計算の実行を重複させることも考慮し、この両者を統合的に動的に管理する方法に着目することとした。

3．データパラレルと分散処理

分散システム上でデータパラレル処理を行う場合、各処理装置は分割して割り当てられているデータに対して計算を行う。例えば図1のプログラムを4台の処理装置で実行する場合、配列a,bおよびcは図2にように分割される。この時、代入分の左辺のデータを保持している処理装置が右辺の評価を行い、左辺に代入するのが一般的である。処理装置1がa(50,50)の計算を行う場合、b(51,51)およびc(49,51)が必要となるが、それらはそれぞれ処理装置4と3が保持しているので、計算実行前に当該データの交信が必要となる。これは簡単なプログラム例であるが、一般的には配列に関する計算と結果の代入に関し、どの処理装置とどのデータを交信する必要があるかを処理系が決定し、実行時にライブラリによりこれを制御する必要がある。

4．実行時における動的管理方法の概要

実行時にデータ交信や計算の実行順序を動的に管理するには、必要な情報を表などの形にして作成しておく必要がある。その情報とは次のようなものである。

(1)　データの交信が必要でないものと必要なものとに分類された計算セット（各処理装置が右辺を評価し左辺の部分配列に代入する配列要素の集合）の情報。

(2)　各処理装置がどこからどのデータを受信するか、またどこにどのデータを送信するかの情報。

この表はコンパイラが静的に作成できるものはこれを実行モジュール内に作成する。コンパイル時に解決できないものは空白などにしておき、実行時にライブラリ・ルーチンが動的に決定する。

並列化効率を高め、データ交信に伴うオーバヘッドを最小限に抑えるために、次の点を考慮する。

(1)　他の処理装置からデータが到着しているときは、その処理装置はデータ交信ができる状態にある

---

[1] A Method of Controlling Message Passing and Execution Orders Dynamically on a Cluster of Workstations

[2] Gyo Osawa and Hideaki Komatsu
IBM Research, Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242, Japan

ということであるから、その処理装置との送受信を行う。
(2) データの到着がないときは他の処理装置にデータを送信し、自処理装置がデータ交信ができる状態にあることを知らせる。
(3) データ待ちの時に、データ交信が必要でない計算セットの計算を実行し、データ交信と計算の実行を重複させる。

5．動的管理アルゴリズムと実施例

図1のプログラムで、宣言された配列が4台の処理装置に図2で示されるように分割されている場合を想定する。この場合処理装置1に関して、表1に示される送受信データ管理表と表2に示される計算セット管理表が作成される。この場合、処理装置1は例えば処理装置2から部分配列b(51:51,3:50)を受信し、c(50:50,3:50)を送信する必要があることを示している。また表2において、a(50,50)の計算を実行するには処理装置3および4からデータを受信しなければならないことを示している。

この場合、図3に相当する機械語命令が実行モジュール内に生成される。実行時ライブラリ*get_iteration_set*では、順次以下の操作を行う。
① 表1の送受信データ管理表を参照しながら、処理装置毎に受信データが到着しているか確認し、到着している場合には受信し、管理表に受信済みのマークをする。また表2の計算セット管理表にもマークする。
② データを受信した場合、そのデータを送信した処理装置に対して送信するべきデータが存在する時にはこれを送信し、送信済みのマークをする。
③ その時点では受信するデータがない場合、自処理装置から送信するべきデータを順次送信し、送信済みのマークをする。そして①に戻る。
④ 次に表2の計算セット管理表を参照しながら、まだ実行していない計算セットでデータ受信済みのものを探し、その値をループ制御変数にセットし、計算済みのマークをする。
⑤ 計算セットが見つからない場合には、受信操作が不要である計算セットを探しその値をループ制御変数にセットし、計算済みのマークをする。
⑥ データ待ちなどの理由で現状ではいずれの計算セットも実行できない場合には、①に戻る。

6．動的管理アルゴリズムの実装

上述した管理アルゴリズムを、IBM Risc System/6000[3]ワークステーションを16台接続したクラスタ・システム上のHPF言語処理系における実行時ライブラリの一部として実装している。データ交信のプリミティブとしてはIBM AIX[4] Parallel Environmentの通信ライブラリを用いている。このライブラリでは非同期に通信できるプリミティブが用意されており、本研究におけるデータ交信の制御に使用している。現在は、動的にデータ交信を管理する部分の実装が終了しており、今後これを拡張して計算の実行順序も動的に変更し、統合的に管理するライブラリを構築する予定である。

参考文献

[1] High Performance Fortran Forum: High Performance Fortran Language Specification, Version 1.0 DRAFT (Jan. 1993)
[2] 郷田 修: High Performance Fortran, 情報処理 Vol. 34, No. 9

```
real a(100,100), b(100,100), c(100,100)
do i = 2, 99
  do j = 2, 99
    a(i,j) = b(i+1,j+1) - c(i-1,j+1)
  end do
end do
```

図1： プログラム例

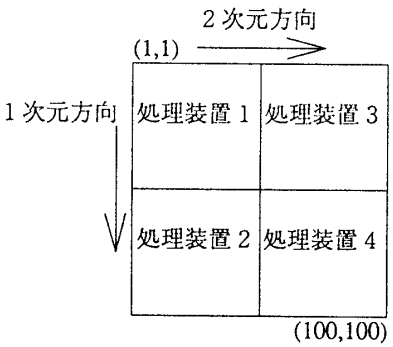

図2： 配列データa,bおよびcの分割

```
while(get_iteration_set(2, start, end))
  do i = start(1), end(1)
    do j = start(2), end(2)
      a(i,j) = b(i+1,j+1) - c(i-1,j+1)
    end do
  end do
end while
```

図3： 生成される機械語に相当するプログラム

処理装置番号	受信配列要素	送信配列要素
2	b(51:51,3:50)	c(50:50,3:50)
3	b(3:50,51:51) c(1:49,51:51)	なし
4	b(51:51,51:51)	なし

表1： 処理装置1における送受信データ管理表

配列aの計算セット	受信操作が必要な処理装置
(2:49,2:49)	なし
(50:50,2:49)	2
(2:49,50:50)	3
(50:50,50:50)	3、4

表2： 処理装置1における配列aの計算セット管理表

[3] IBMは米国IBM社の登録商標、Risc System/6000はIBM社の商標です。
[4] AIXはIBM社の商標です。