並列プログラミング言語の設計と実現

A Parallel Programming Language : -Design and Implementation-

小畑　正貴(*)　金田悠紀夫(**)　田中　敏幸(***)　前川　禎男(**)
MASAKI KOHATA　YUKIO KANEDA　TOSHIYUKI TANAKA　SADAO MAEKAWA

(*)　岡山理科大学理学部　(**)　神戸大学工学部　(***)　(株)シャープ
Okayama Univ. of Science　Kobe Univ.　Sharp co.,ltd.

1．まえがき

並列計算機システムについては、そのアーキテクチャ・プログラム言語・並列処理アルゴリズムなどについて以前より多くの研究がなされている。そして、近年におけるハードウェア技術の発達、特にマイクロプロセッサの出現により、いくつかのシステムが実際に作成され、研究および一部の応用に利用されるようになっている［1，2］。しかしながら、並列処理アルゴリズムを記述でき、それをマルチプロセッサ上で効率よく実行できる並列プロラミング言語に関しては実現が遅れており、並列計算機システムのより幅広い応用に対するさまたげとなっている。

マルチプロセッサシステムは、その結合方法や制御方法によってさまざまな形をとるため、どんなシステムに対しても効率よく動作する言語を実現することは、一般的に困難である。これまでに実現された並列処理記述言語の例として、アレイ結合型のＩＬＬＩＡＣⅣに対するIVTRAN［3］や、疎結合分散型システム用のConcurrent　C［4］などがあげられる。

本稿で提案する並列プログラミング言語は、共有メモリ結合による並列計算機システムを対象にしている。このシステムは、最も一般的な結合形態の一つであり、実現が容易である。

本言語は、C言語［5］に並列処理機能を付加する形で設計され、１６ビットマイクロプロセッサを用いた試作マルチプロセッサシステム［6］上に実現されている。本稿では、本言語の機能と実現法、および試作機上での処理系について述べる。

また、並列プロラミング言語においては、記述の容易さ・正確さの点と、実行効率の点が問題であり、例題のプログラム例とその実行結果から、これらの問題について考察する。

2．並列プログラミング言語の設計

2．1　対象システム

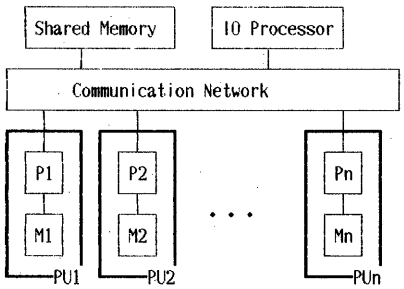


図1．ハードウェア・モデル
fig 1. hardware model

対象とする並列計算機システムのハードウェアモデルを図1に示す。プロセッサ（P）とメモリ（M）とからなるプロセッシングユニット（ＰＵ）が複数、コミュニケーションネットワークを通して共有メモリに結合された形で、一般的なマルチプルセッサシステムの一形態である。ＰＵ間での通信はネットワークを通して直接、あるいは共有メモリを通して間接的に行うことができる。

システムを構成するプロセッサとメモリ間での通信コストをC（P，M）で表すとすると、このシステムにおいては一

般に次の関係が成り立つ。

$$C(Pi,Mi) < C(Pi,SM) < C(Pi,Mj),\ i \neq j$$

言語設計においては、この関係をもとに、実行効率の向上を考慮する。

また、入出力はＩＯプロセッサが行うものとする。

２．２　基本設計

言語設計における基本概念を以下に示す。

（１）記述性の良さやプログラミングのし易さの点から、Ｃ言語と上位互換性を持たせる。

（２）システム全体に対して一体化したプログラムの記述ができる。また、プロセッサ台数によらないプログラムが書ける。

（３）数値計算などにおける大容量データを効率よく取り扱うことができる。

（４）できるだけ構造化された形で並列動作制御機能を組み込む。

以上の点を基本としてＣ言語に対するデータ構造と制御構造の拡張を行っている。拡張された機能を以下に示す。

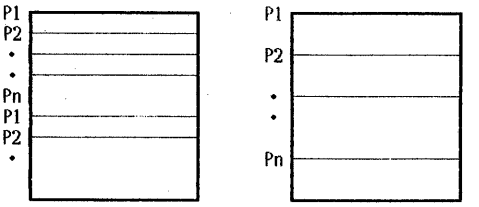
２．３　データ構造の拡張

（１）記憶クラス（universal,shared）

universal で宣言された変数は共有メモリ上におかれ、sharedで宣言された変数は各ＰＵ内のメモリにおかれる。この２つの記憶クラスの変数はすべての関数とすべてのプロセッサからアクセス可能であるが、アクセスに必要なコストには２．１節で述べた関係がなりたつ。sharedは多量のデータを分配して処理する場合に効果的である。

（２）配列（m,b,s）

行列計算などにおける多量のデータは、各プロセッサに分配され、並列処理される。この際データの分配法として２つの方法を考える。１つは配列のあるインデックス値を台数で割った剰余を使って、端から順に配っていく方法であり、もう１つは、全体を台数個のブロックに等分し、分配する方法である。これらはそれぞれ m,bを使って宣言する。２次元配列における例を図２（ａ），（ｂ）に示す。また、スパース行列を取り扱う場合には、配列のサイズのかわりに [s]と宣言することで非ゼロ要素のみがリスト構造で内部表現され、記憶領域が節約できる。



(a) a[100:m][100]　　(b) a[100:b][100]

図２．配列要素の分配
fig 2. destributions of array

（３）ＰＵ間での参照（ＰＵ：変数）

他のＰＵの shared クラスの変数を参照する場合、それが前述の m,bによって分配された配列ならば、そのインデックス値によってどのＰＵか一意的に決定できる。しかし、すべてのＰＵ上に同じ名前でとられている変数に対しては特別なＰＵ指定が必要になる。このため、変数名の前にＰＵ指定プリフィックスを付けて相手ＰＵを指定する。

２．４　制御構造の拡張

（１）単一パス指定（ single(ＰＵ) 文:）

従来の並列動作の記述法には、直列部分から並列部分への移行に着目したものが多い（fork-join,cobegin-coend など［７］，［８］）。本言語では逆に、並列実行中の直列部分に着目した記述を行う。これは本言語が記述の対象として、潜在的に高い並列性をもつ問題を考えている点からくるものである。

単一パスは single(ＰＵ番号) 文：で表される。内部の文は指定されたプロセッサのみが実行する。

（２）仕事の分配（forall（式１；式２；式３；ＰＵ）

並列化できるループに対する分担処理を示す。各ＰＵは、forallループのうち、自分の分担のみを実行する。この文は以下の文と同等である。

```
式１；
while （式２） {
  single（ＰＵ）
    文；
  式３；
  }
```

特にＰＵを指定しない時は、端から順に分配されるものとする。

（３）一斉同期（sync）

sinc文は全ＰＵでの一斉同期を示す。これ以後の文の実行は、これ以前の実行が全ＰＵで終了するまで遅らされる。実行の様子を図３に示す。

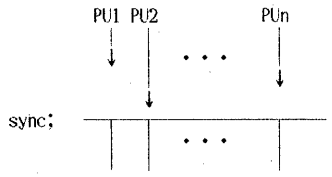


図3．同期
fig 3. synchronization

（4）通信（send（変数，式），receive（変数），
release （変数）

各プロセッサはこれらの文によって、共有変数についての同期をとる。send文は、式の値を共有変数に代入し、値の確定を全プロセッサに知らせる。receive 文は、その値の確定を知る。もし確定していなければ、確定するまで待つ。また release 文は、その変数値の使用を終えたことを知らせる。全プロセッサが使用を終えると、その変数は書き換えが可能となる。

（5）相互排除（lock（変数）文）

一般に相互排除は共有変数に対するアクセス時に必要となる。ここでは、変数ｖに対する相互排除を、これに鍵をかけるという意味で

lock(v) ｖを含む文；

と表現する。この文を実行する間は、ｖに対する相互排除領域となる。

さらに、複数の変数に対する相互排除は

lock(a,b,c) 文；

のように表し、処理系でロックの順序を決定することによりデッドロックを防ぐ。

３．処理系の実現

本言語の処理系を試作並列計算機システム上に実現した。以下、試作機の構成、各機能の実現法、試作機上での実行環境について述べる。

３．１　並列計算機システム

本システムのハードウェア構成を図３に、メモリ構成を図４に示す。

（1）プロセッサ

インテル社の１６ビットマイクロプロセッサ８０８６に数値演算プロセッサ８０８７を付加したマルチプロセッサ構成となっている。

（2）メモリ

本システムはアクセス法の異なる３種のメモリを持つ。ローカルメモリ（LM）は各ＰＵに固有のメモリで、他からア

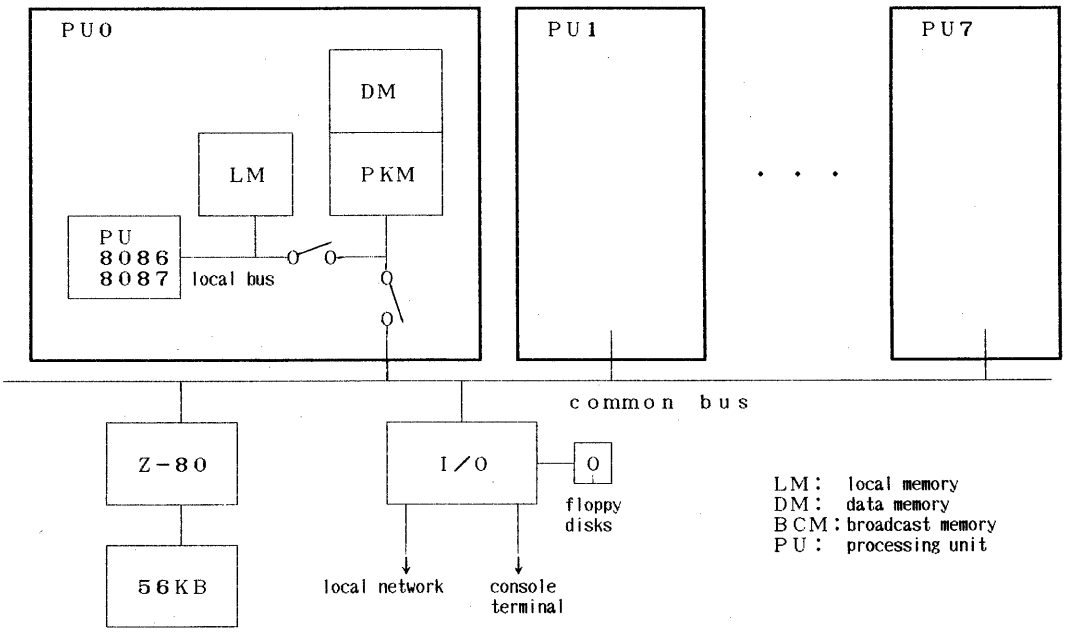


図４．並列計算機システム
fig 4. the multiprocessor system.

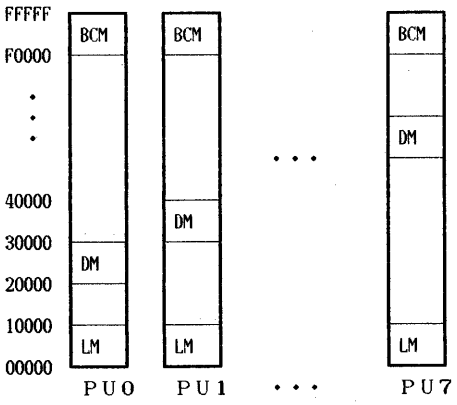


図5．メモリ構成
fig 5. memory configuration

クセスされない。データメモリ（DM）は各ＰＵ内で自分のプロセッサからアクセスされるほか、バスを通して他のプロセッサからもアクセスされる。ブロードキャストメモリ（ＢＣＭ）は放送転送機能を持っており、書き込み時には全ＰＵの同一アドレスに同じデータが自動的に転送される。

これら３種のメモリは、ハードウェア上ではそれぞれ図３の位置にある。また、アドレス空間上では、ＬＭとＢＣＭはそれぞれ下位と上位の同一アドレスにあり、ＤＭはシステム全体でユニークにアドレス付けされる。

（３）ソフトウェア

ＯＳにはＣＰ／Ｍ－８６を用いており、ソフトウェア開発時にはＰＵ０のみが動いている。実行時にはＰＵ０を含む全ＰＵに起動がかけられ、実行が終わると再びＰＵ０のみに制御がもどされる。

３．２　各機能の実現法

（１）データ構造

Ｃにおける自動変数と外部変数はＬＭ上にとられ、拡張されたクラスである universal変数と shared 変数はそれぞれＢＣＭとＤＭ上にとられる。

ＢＣＭ上のすべての変数にはセマフォが対応づけられ、これによって send,receive,release および lock が実現される。

また、スパース宣言された配列に対しては非ゼロ要素のみを

（インデックス，値，ポインタ）

からなるセルの連結リストで内部表現する。

（２） single , forall

各プロセッサにシステム変数として pnum,inum,lc を持たせる。pnumは全プロセッサ数を示し、inumは自分のプロセッサ番号を示す。これらは実行時にＯＳから与えられる。lcはforallループ内でのループカウンタとして働く。

single文でのＰＵ指定は、たとえば次のように書かれる。

single (i % pnum)

（３）同期

共有メモリ上にカウンタ（初期値= pnum ）を設ける。各プロセッサは sync 文でこれをデクリメントし、０にならなければふたたび値が pnum に戻るのを待つ。最後に０にしたプロセッサがこれを pnum に戻し、全プロセッサに起動をかける。このデクリメントはバスロックを用いて相互排除で行う。

（４） send ,receive ,release

universal 型変数に付加されたセマフォ（sem1,sem2)を用いる。sendでは sem2 が０になるのを待って変数に式の値を代入し、sem1,sem2 を pnum にする。receive では sem1 が０でなくなるのを待ち、sem1をデクリメントしてから使用に入る。release では sem1 が０になるのを待ち、sem2をデクリメントする。

release で sem1 が０になるのを待つのは、いったん全ＰＵがその変数を使用することを保証し、同一ＰＵが一度 releaseした値を再び使用しないようにするためである。

（５） lock

同じ変数に対して send 文と併用しないものとし、セマフォに対するテストアンドセット命令によって実現する。

（６）実行

実行は main() で指定された関数から、全ＰＵで平行して開始される。

３．３　試作機上での実行環境

プログラムの作成からコンパイル・実行まですべてＣＰ／Ｍ－８６上でおこなわれる。コンパイラはＣで記述されており、出力としてＡＳＭ８６へのアセンブリ言語を生成する。このあと、ＡＳＭ８６およびＧＥＮＣＭＤにより実行を行う（図６）。

なお、現在の版では、関数・構造体および演算子の一部は実現されていない。

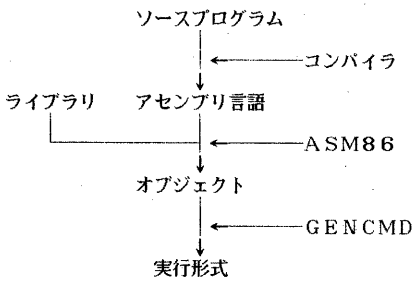


図6．実行手順
fig 6. exsecuting steps

実行時には、同じプログラムがロードされ並列に起動がかけられる。

4．プログラム例

4．1　プログラミング

並列計算が効果的である場合として、大容量データを分配して処理する問題が考えられる。この場合のプログラムは、

・初めにデータを分配しておき、

・自分の割り当て分のみを実行する。

・１台の中では通常の直列実行をする。

という形で記述できるが、このために以下のようなことがらが必要となる。

・仕事を分担する。

・共通に使われる変数を共有領域に置く。

・共有変数の値の確定を保証する。

共有変数の値の確定を保証する手段としては、

・syncを使って一斉同期をとる。

・send,receive,releaseによって同期をとる。

ことが考えられる。後者は前者より高並列性を得ることができるが、プログラムは複雑になる。

4．2　ガウス消去法のプログラムと実行結果

プログラムの例として、連立方程式　Ａｘ＝ｂ　に対するガウス消去法を考える。入力および前進消去部に対する本言語によるプログラムを図７に示す。

また、試作システムによる実行時間と速度向上を図８，図９に示す。測定に用いた係数行列は１００×１００の密行列で、８０８７による浮動小数点計算を行っている。

```
shared float a[100:m][101];                          1
universal float aa[101];                             2
main() {                                             3
int i,j,k,n=100;                                     4
single(0)                                            5
   for(i=0;i<n;++i)                                  6
      for(j=0;j<n+1;++j)                             7
         a[i][j]=getf();                             8
sync;                                                9
for(k=0;k<n;++k) {                                   10
   single(k%pnum)                                    11
      for(i=k;i<n;++i)                               12
         aa[i]=a[k][i]=a[k][i]/a[k][k];               13
   sync;                                             14
   forall(i=k+1;i<n;++i;i%pnum)                      15
      for(j=k+1;j<n+1;++j)                           16
         a[i][j]=a[i][j]-a[i][k]*aa[j];              17
   sync;                                             18
   }                                                 19
        ・・・
```

図７．ガウス消去法のプログラム
fig 7. program for gauss elimination

台数	1	2	4	8
時間（秒）	54.5	28.5	15.4	8.8
速度	1	1.92	3.54	6.15

図８．実行時間
fig 8. exsecution time

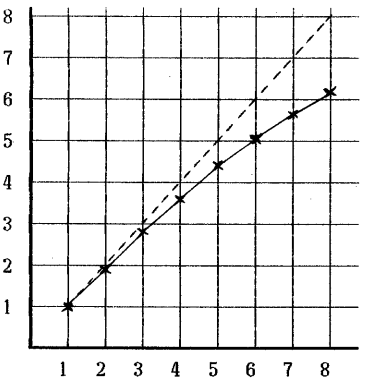


図９．速度向上
fig 9. speed up

4．3　評価

プログラミングにあたり、従来の直列処理言語に対してよぶんに必要なのは、

・共有変数の宣言

・並列部と単一部の指定

・共有値のコピー

・仕事の分担

・同期

に関する部分である。一斉同期を多用すれば比較的容易に並列プログラムが得られるが（図7）、同期部での待ちが多くなるため、実行効率は多少悪くなっている（図9）。

実行待ちがおこるのは同期および相互排除の部分である。これらの実行待ちは send,receive を使ったり、実行順序を変えたりすることによって減少させることができるが、プログラミングはより複雑になる。

5．むすび

共有メモリを持つマルチプロセッサシステム用の並列プログラミング言語の開発と実装例について述べた。本言語の特徴をまとめると次のようになる。

（1）共有メモリ結合によるシステムに広く対応できる。

（2）C言語を拡張しており、記述性・移植性がよい。

（3）数値計算向きであり、大容量データを効率よく処理することができる。

本言語の処理系は、試作マルチプロセッサ上で実現され、数値計算を中心とした並列アルゴリズムの研究に利用されている。構造解析の問題については他講座の学生にも使用してもらっているが、比較的容易に慣れていけるようである。また、実行効率に関しては、多くの問題について、8台の時に6～7倍の速度向上が得られている。

4．3で述べたように、正しくかつ効率のよい並列動作を簡単に記述することは、一般に困難である。これを実現するためには、コンパイラによる自動並列化や並列実行の最適化が必要であり、今後の大きな課題である。

参考文献

[1] 村岡，坂間：並列処理技術，信学誌，Vol.63,No.10 pp.1064-1071，(1980)

[2] A.K.Jones,P.Schwarz:Experience Using Multiprocessor Systems-A Status Report,Computing Surveys, Vol.12,No.2,pp.121-165,(1980)

[3] 加藤，苗村：並列処理計算機，オーム社，(1876)

[4] 安藤ほか：分散型システム記述用言語 Concurrent C の設計とその実現，情報処理論文誌，Vol.24,No.1, pp.30-39,(1983)

[5] カーニハン，リッチー：プログラミング言語C，石田訳，共立出版，(1981)

[6] 小畑，金田，前川：ブロードキャストメモリ結合形マルチマイクロプロセッサシステムの試作，情報処理論文誌，Vol.24,No.3,pp.351-356,(1983)

[7] P.B.Hansen:Concurrent Programming Concepts, Computing Surveys,Vol.5,No.4,pp.223-245,(1973)

[8] G.W.Andreus,F.B.Scheneder:Concepts and Notation for Concurrent Programming,Computing Surveys, Vol.15,No.1,pp.3-43,(1983)