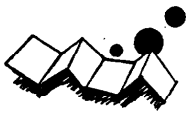
解　説



# 偏微分方程式解析のためのマイクロプロセッサ複合体†

星　野　　力††

## 1. はじめに

70年代において本格化したLSI技術の進展は各分野へ大きなインパクトを及ぼしつつあり計算機設計の分野でも変革期であるといわれている．私達，科学技術計算に携わってきた者にとっても，今までのお仕着せの汎用計算機ではなく，オーダメイドの計算機を持つという可能性が出てきた．

この記事は私達が，どういう考えで科学技術用並列計算機の試作を始めたか，という一つのケーススタディとしてお読みいただきたい．本来解説記事としては公平なレビューと展望に努めるべきであるが，専門家でない私には少し荷が重すぎる．むしろ私達が現在試作しているシステムに即して書いた方が面白いのではないかと思ったので，若干の自己宣伝も含まれていることをお許しいただきたい．

## 2. ニーズ

確かな統計を持っているわけではないが，私の印象では科学技術用高速大型計算機のCPU時間のかなりの部分は，多次元偏微分方程式を解くのに費やされているのではないだろうか．この種の問題を解くニーズは大きくて際限がなく，新設されたばかりの大型計算機をすぐにジョブであふれさせてきた．計算規模のイメージをうるために，一例をあげると，流体の運動方程式（Navier-Stokes 方程式）を空間 $10^6$ ノードにわたって，1ステップの時間積分を行うのに，CDC 7600で82秒を要するといわれている[1]．現象をシミュレートするには，かなりのステップ数の積分を行わねばならぬから，全部で約数十〜数百時間，商業ベースで約数千万〜数億円ほどの計算料を要するだろう．これを支払うのは容易ではないから，大低は資金に見合ったサイズに縮少されるのが普通である．記憶容量の制約は最近の仮想記憶方式とメモリ価格の低下で以前よりはゆるやかな制約となっているので，供給不足となっているのはやはり計算速度の方である．

偏微分方程式の数値積分に帰着する問題は，理工学の全分野にわたって存在し，若干の例をあげると，連続体力学の Navier-Stokes の方程式，連続の方程式，プラズマ物理の電磁流体力学方程式，電磁界の Maxwell 方程式，量子力学の Schrödinger 方程式，種々の分野に現われる拡散方程式，熱伝導方程式などがある．最近の巨大科学では装置は大型化し，巨額の建設費を要する．たとえば核融合実験装置や大型風洞で約100〜1,000億円，原子力発電プラントは2,000億円もする．そこでは理論に基づく数値計算と実験の合致が追求され，実験が不可能な場合は実験の代用となりうる数値シミュレーションの技術が要求されている．

いずれにしても，計算機にものを言わせたようなシミュレーションへのニーズは今後ますます巨大化するであろう．

よく現われる偏微分方程式（以下 PDE と略す）は多次元多変数で，しばしば非線形である．次元を増すにつれて問題のサイズが幾何級数的に増大し，困難が飛躍的に増す．これを「次元の呪」という．たとえ10倍速い計算機が出現しても，呪われているユーザは予算さえ許せば問題のサイズをすぐ大きくし，それでも十分満足することがないので，まるで無限の計算需要があるかのようである．

## 3. 問題の特徴

これらの PDE 系に共通した著しい特徴は，いわゆる「近接作用」によって場が形成されていることである．ある位置における密度とか電位とかは近傍におけるそれらの値との関係で決っており，いわゆる遠隔作用は存在しない．数学的には，微分方程式で表現されることになり，差分近似を行えば，データ間の相互作

† Microprocessor Complex for the Analysis of Partial Differential Equations by Tsutomu HOSHINO (Institute of Atomic Energy, Kyoto University).

†† 京都大学原子エネルギー研究所

用は極めて局所的に生ずるのみである．たとえばラプラスの方程式 $\Delta f=0$ を例にとると，ある $k$ 回目のくり返しでのノード $i,j$ における値は，前回のまわりのノードにおける値から，

$$f^{(k)}(i,j)=\{f^{(k-1)}(i-1,j)+f^{(k-1)}(i,j-1)+f^{(k-1)}(i,j+1)+f^{(k-1)}(i+1,j)\}/4 \quad \cdots\cdots(1)$$

のように表現される．これをすべてのノード $i,j$ について計算し $\|f^{(k)}-f^{(k-1)}\|<\varepsilon$ になるまで $k$ に関してくり返して収束解をうる．

通常の計算機ではすべての計算は逐次的に行われるが，本来(1)式は並列に計算可能なものである．並列処理によっても「次元の呪」が本質的に解消するものではないが，もしノード数に見合うプロセッサを用意出来れば計算時間はノード数に無関係になる．

またこれは自然現象の「自然」なシミュレーションでもある．自然現象は，すべての点で同時に進行している．この同時性を出来るだけ損わないようにシミュレートすることによって無駄のない計算が行えるという希望がある．

並列計算については古くから考えられていたもので，歴史的叙述は成書[2),3)]にゆずり，また最近の並列処理の動向についても，よく整理された解説記事[4)~7)]があるので参照されたい．

## 4. 専用機の時代

現在の高速計算機はそろそろ素子の速度限界に近づいていると言われている一方，周知のように LSI 技術は安価で性能価格比のよいマイクロプロセッサを実現した．もしこれを多数結合してもシステムの性能価格比が低下しないようにし，大きな計算能力が実現出来れば巨大なシミュレーションのニーズに十分こたえることが出来るだろう．

後述の RAND コーポレーションで提案されている Navier-Stokes 方程式専用計算機では1万個のワンチップ LSI プロセッサを直角アレイ状に並列結合して，前述の CDC 7600 で 82 秒かかる計算を 0.35秒で行うと見積られている[1)]．ハードウェアのコストは1チップ当り 10~100 ドル見当と見積られており[8)] 1万個のチップの値段だけで最大2億円ぐらいであろう．これは汎用大型計算機での1ケースに要する計算料と同程度であり，大型風洞の建設費の約 1/100 である．このことから大型装置の建設費の端数程度の金額で専用の計算機を開発することも可能になりつつあることがわかる．

計算機のハードウェア価格の高かった時代では，汎用ハードウェア上で多種多様なジョブを走らせるという，いわゆる共同利用という形態が経済的であり，汎用性が計算機設計の常識であったが，これはそろそろ過去の概念となりつつあるのではないだろうか．たとえ PDE 系のシミュレーションに目的を限定しても，科学技術の分野では十分汎用とも言える．またユーザはそういう専用機の開発費を負担しうるようになってきているのではないかと思う．

元来，計算機は数値計算上のニーズから生れたものと言われている．その後技術的経済的理由から科学技術計算は汎用機といういわば既製服を着せられてきた．とくべき問題に対してハードウェアは最適なものからほど遠い存在であった．計算機設計者とユーザの間にはなにがしかのコミュニケーション・ギャップすら生れるに至っている．

もうそろそろ，われわれは初心に帰り，計算機をその用途との密接な関連において見直してよい頃ではないだろうか．ユーザから見れば，計算機は計算すべき問題があってはじめて存在意義をもつものなのである．

## 5. 開 発 例

### 5.1 SMS[7),9)]

汎用科学技術計算機として Siemens 社で開発されている SMS (Structured Multimicroprocessor System) を紹介する．最大 128 のプロセッシングユニット (Intel 8080) をもつシステムである．少なくとも一つのプロセッサを含み並列動作を行う単位モジュールを以下プロセッシングユニット PU と呼ぼう．特徴は SMIMD (Switched Multiple Instruction-Multiple Data Stream) 方式を採用しているので，コントローラによって (i) 制御フェーズ，(ii) 通信フェーズ，(iii) 実行フェーズの3フェーズに構成が切り換えられる．(i)(ii) のフェーズでは各 PU 中に含まれるメモリ CM がバスに結合され，CM 間でデータ交換が行われる．(iii) のフェーズでは CM は PU 内にとりこまれ，各 PU は独立に計算を行う．通信用メモリ

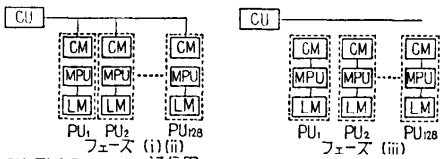



図-1 SMS の構成

CM が共通メモリ的機能を果すわけである．性能としては，バスの転送能力 0.1 MBPS, 128 PU 全体として演算能力 32 MIPS(8 ビット)，0.03 MFLOPS (32 ビット，MFLOPS=million floating point operations per second) である．応用例として1時間先の北半球の天気予報の計算 (2,000 ノード) を3分で行うとされている．

収束計算中のデータは単一バスを経て通信されるため計算規模が大きくなるとバスの混雑が生じる．プロセッサをビットスライス型としバスの転送能力を 3.5 MBPS に向上させた SMS-3 システムの例では，250 MIPS, 18 MFLOPS の性能とされているが，天気予報の例では計算時間の 41% がバス上のデータ交換に費やされている．これでは，これ以上 PU を結合しても巨大計算用としては実用になりにくい．やはり汎用性にとらわれすぎた単一バスの採用が災しているのではないだろうか．中規模システム SMS 201 (算術演算 LSI つき) でも 0.55 MFLOPS の性能であり，最新のミニコンの性能と (多分性能価格比も) 大差ないことになる．

### 5.2 DAP[10]

DAP (Distributed Array Processor) は ICL 社で開発されたものである．実験機 Pilot DAP は 32×32 =1,024 個の PU をもつ．もっと多数の PU たとえば 128×128=16,384 個にまで拡張することは容易である．

各 PU は1ビットの汎用レジスタQ，アクティビティレジスタAとキャリーレジスタCをもち，1ビットの演算を行うユニットで，4Kビットのメモリをもつ．PU 間結合は上下左右の4方向のみに1ビット幅で行われる．行と列方向にデータハイウェイをもち，マスターコントローラと通信する．マスターコントローラでデコードされた制御信号は，アレイの一辺のサイズ (たとえば32) のビット幅をもち，各 PU に放送され，これにより全 PU が同一動作を行う．すなわち Flynn の分類[11] による SIMD (Single Instruction-Multiple Data Stream) 方式である．各 PU

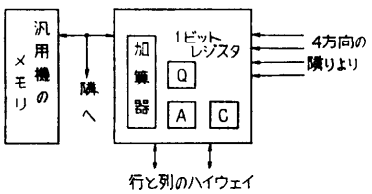



図-2 DAP の PU の構成

のもつメモリは同時に親コンピュータ (ICL 2900) の主記憶の一部となっており，アレイプロセッサが記憶装置にうめこまれていて一種のインテリジェント・ストレージとなっている．専用の DAP-FORTRAN により並列アルゴリズムを記述する．1PU の性能は浮動加減算 148 μs，同乗算 305μs，平方根 215 μs，全システムの性能は 32×32 元逆行列 29 ms，FFT (1,024 点複素) 14 ms である．応用例での性能比は，有限要素法で 2～6×IBM 360/195，3次元 MHD 計算で 14～30×IBM 360/91，パターンマッチング問題で 300×360/195，多体シミュレーションで 10×CDC 7600，索表問題で 3×CRAY-1 と報告されている．性能は高く，かなり実用規模に近いものと言える．なお SIMD 方式には後述のような欠点があると思われる．

### 5.3 RAND の計画[9]

DAP と似た PU 間結合の構造を採用している例として，カルフォルニア工大のサザランド教授により提案され，RAND コーポレーションで検討されているシステムを紹介する．特徴は流体力学の Navier-Stokes 方程式専用であることで，PU 間データ通信は隣接の4PU との間に限定されており，1PU は 16 ピンの1チップにすることを考えている．1PU は固定小数乗算を 5 μs，隣接 PU へのデータ転送を 3 μs で行い，256 語 (64 ビット) のメモリをもつ．DAP や ILLIAC-IV と同様の SIMD 方式である．流体シミュレーションの計算時間の例は専用機の時代の項で述べた．CDC 7600 の約 100～1,000 倍の性能を目指している．PU の数は 1～10 万個を想定しており，100×100 のアレイにしたときの大きさは 8×8 フィートで壁にかけられるぐらいの大きさである．PU 数が大きいため1チップ化は必須条件で，それも出来るだけサイズを小さくする配慮をしているらしい．その理由は LSI のコストが全ハードウェア中にしめる割合が小さくなっており，むしろボード，コネクター，ケーブル，フレームなどの幾何学的設計がコスト上重要になってきているためである[12]．

この提案にもとづいてコンピュータメーカ，チップメーカ，ユーザを集めて開かれたワークショップ (1977年3月)[9] では，結論として技術的には十分実現可能で，3～5年以内に汎用超高速機よりずっと安い価格で実現するとしているが，まだこのプロジェクトは予備的研究段階にあると聞く．

## 6. PACS のアーキテクチャ

私達の試作した PDE 専用のマイクロプロセッサ複合体 PACS (Processor Array for Continuum Simulation)[13] のアーキテクチャについて述べ，あわせてこの種の試みのもつ一般的な問題点，可能性について言及する．

### 6.1 プロセッサ間通信

汎用性を目指すシステムでは，プロセッサはシステム中の他のプロセッサ，メモリその他の資源に自由にアクセス出来るようになっているのが普通である．一方 PDE 問題ではデータは隣接した PU 間でのみ通信されるとしてよい．PACS では PU は2次元直角格子状に配置し，各 PU は原問題の空間の1ないし数ノードより成るブロックの計算を担当する．隣接したプロセッサ間に通信用メモリ CM を設け，これを両方のプロセッサのアドレス空間に含める．隣接プロセッサのクロックを入れ換え，CM を一つのクロックに同期した時分割で両方のプロセッサと結合するようにした．この方式は I/O 用のハードを特に必要とせず安価でありまた転送はメモリアクセスと同じで高速である．PACS の構成を図-3 に示す．PU はコントロールユニット CU の制御をうける．CU のアドレス空間内に各 PU のプログラム領域が同一のアドレスを持って含まれており，通信方式は CM と同じであるが各 PU へ向って一斉の放送が可能である．共通バスは CU-PU 間通信に限って使用される．PU 中のバッファメモリへのアクセスは CU の優先的制御の下に行われるので，CU へのアクセス競合という問題は生じない．本来，汎用機との結合は I/O インタフェース経由とせずに DAP のように汎用機のアドレス空間内に PU のメモリを含めてしまうのがよい．そうすれば汎用機が初期値や制御用コマンドをメモリ中に書きこんで CU に制御を移せば，PU がメモリ上で並列計算を行い，収束すれば解が汎用機のメモリ上にのっていることになる．

固有値問題を解くときによく現われるノルムの計算では，全 PU にわたる加算が必要となることが多いが，これも CU で行わなくとも，PU 上で並列に行える．すなわちアレイ上のデータを一方向へ $2^r$ 回転送し各 PU で加算するという手続きを $r=0,1\cdots\cdots \log_2 N$ 回にわたって行うと全 PU に和が現われる[4]．このためにはアレイの両端の PU 間を結ぶ CM を設け，領域をエンドレスにする必要がある．したがって

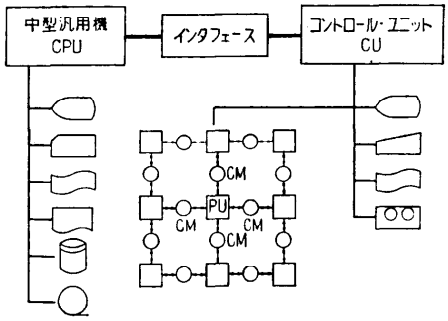



図-3 PACS-9 の構成

CU にはこれらの並列計算のフェーズ（すなわち収束計算，ノルム計算等）の切り換えを行う仕事がある．

### 6.2 プロセッサの同期

PACS は MIMD (Multiple Instruction-Multiple Data Stream) 方式[11]を採用しており各 PU は独立のプログラムをもちうる．もちろん PU 内のメモリ容量が許せばプログラムはすべて同一として CU より放送モードでロードし，実行時には各 PU ごとに条件判別を行って別々のパスを通るように出来る．各 PU はアルゴリズム・レベルでは非同期的に動作し，計算が終り次第まわりの PU と通信を行う．データの受け渡しは，通信メモリ上にある1個のデータ・レディ・フラグによって制御する．くり返し計算に同期したデータのやりとりなのでこの方式で十分である．

こうして各 PU のうち一番計算の遅いものに歩調を合わせて全体の計算が進んでゆく，収束判定は各 PU ごとに行い．全部の PU で収束すれば CU は全 PU にストップをかけ収束したデータを共通バス経由でとり込む．

### 6.3 オーバヘッド

PU 数が多くなるにつれてシステムの性能価格比は悪化してゆくと一般にいわれているのは，共通資源へのアクセス競合などにより全部の PU が有効に働かないからである．PACS ではそのような可能性は少ないが予想されるオーバヘッド（1個のプロセッサで逐次的に処理するときには現われない計算能力のむだ）としては（i）PU 間通信時間によるもの，（ii）PU 内計算時間のばらつきによるもの，（iii）CU による並列計算フェーズの切り換えによるもの，が考えられる．

（i）は前述の通信手順をふむことによって単なるメモリアクセス以上の時間を要することによるもので，これはかなり問題のノードのとり方に依存し，一般的

議論が困難であるが，一般的に言って PU 数が増え一つの PU が担当するノード数が減ってくると，(PU が受け持つ領域の表面積対体積比が増すと)このオーバヘッドは増す．$\beta_1$=(PU 間通信に含まれる余分な時間)/(PU 内計算時間) とする．通信速度を低下させないためには，通信路の幅を狭めたり，ソフトウェア制御のポート経由で行う通信はまずいと言える．1PU 当りのノードのとり方をきめておけば，$\beta_1$ は全 PU 数Nに無関係である．

(ii)のオーバヘッドは，収束計算の進み方は一番遅い PU によってきまるため，速く計算がすむ PU はアイドル状態に入ることになって生ずる．すべてのノードについて計算のアルゴリズムが同一ならば．このオーバヘッドはデータによる算術演算速度のばらつきによって生じる．$\beta_2$=(平均アイドル時間)/(平均 PU 内計算時間) とする．演算速度の統計的ばらつきには上限が存在するのでNが増すにつれて $\beta_2$ は飽和すると考えられる．むしろ深刻なのは，問題によっては特殊な長時間の計算を要する領域が存在し，たとえ，それが一点でも全体の性能は低下することである．これはその点の計算そのものを短かくする以外に有効な解決策はないだろう．

(iii)については，普通は PU アレイ上のくり返し計算がほとんどの計算時間を占めると思われるので，あまり考慮しなくてよいだろう．フェーズ切り換えといっても単に PU 内にすでにロードされているプログラムの別のスタート番地へ飛びこむだけの手順である．しかし，もし下手なプログラマがこれをループの深いところで行えば，そしてまたメモリ不足からそのつどプログラムのロードを伴うのなら，深刻なオーバヘッドを生ずるだろう．

並列計算フェーズの切換えを伴わない収束計算における，並列処理の効率（すなわち全 PU の働作時間に対する有効な計算時間の率）$\alpha$ は，$\alpha=1/(1+\beta_1+\beta_2)$ によりきまる．したがってN個の PU によって計算能力は，$\alpha$N 倍に増す．

私達は今までの経験から，特定の問題を与えられれば，上のようなオーバヘッドを評価し，十分効率のよい設計が出来ると予想している．

### 6.4 システム・プログラム

オペレーティング・システムの開発については，大きな負担にならないであろう．ソフトウェアによる複雑な制御を必要としないし，スタンドアロン型ではなく，汎用システム中の一演算装置 (Number Cruncher) として働くからである．

ソフトウェア開発が必要なのは，汎用機を用いたクロス・コンパイラである．コンパイラ言語にどういう並列処理特有の機能をもたせるかという問題がある．現在の PACS では，PU の制御，通信等はすべてユーザにまかされている．現在の設計では相互排斥に気をつかうような critical section は存在しないが，くり返し計算中はむしろ全部の PU がつき合って走る必要があり，収束判定後部分的に PU を停止したりすると，それに隣接している PU でデッドロックが生じたりする．したがって将来標準的な手続が固まってくると，適当なマクロ・ステートメントを設定することも十分ありうる．

PACS では各 PU 内のプログラムはすべて同一とし CU より一斉放送で PU へロードすることを原則としている．したがって PU 内のプログラムの記述は局所的で，PDE の差分近似式そのまま，またはそれに近いものになり，コンパイラも複雑なアレイ上の計算を記述する必要はないと予想される．

### 6.5 ユーザプログラム

ユーザにとってより本質的なことは問題をとく手続をいかに並列プロセッサアレイ上に展開するかということであろう．

もちろん並列処理に適した数値計算アルゴリズムは今までも多く研究されているが，ユーザから見れば MIMD 流に並列処理を行う上で指導原理となるシステム分解原理がほしいところである．

直観的には自然が現実に行っているのに出来るだけ忠実なシミュレーションを行うことが一つのコツといえる．しかし PDE の初期値問題 $\dot{x}+Ax=b$ はそれでよいとしても，定常問題 $x=A^{-1}b$ や，逆問題すなわちある解 $x$ を結果する条件 A,b を求める問題（いわば非自然問題）はこれではすまない．自然現象（初期値問題）では割算が現われないと言われるが，これらの非自然問題では割算が現われるし，相互作用も近接点間に限られない．これらの問題のためにはまた別のアーキテクチャが必要と思われる．

PACS は隣接プロセッサと主に通信するという特徴を生かせる問題ならば，特に PDE に限らず応用出来る．若干の工夫を要する問題と考えられるものに3階以上の偏微分方程式がある．

有限要素法は，一旦有限要素のとり方がきまると，通信すべき相手のノードは固定されるが，それを PU アレイ上にどう写像するかが一般には難かしい．しか

し十分なノード数を用いた差分近似が実用的になれば，そもそも有限要素法のニーズが解消，もしくは減少するのではないか．回路解析，信号処理，線形計算では一定した通信相手が一般には定義出来ないか，またはアレイ上への写像がむづかしい．

これに関連して直角格子アレイをピラミッド状に重ねた構成が案外一般性をもつという指摘は面白い[14]．たとえば，4×4=16 個の MPU (平社員) の上位に 1 個の MPU (係長) を置き，さらに 16 個の係長 MPU の上に個の課長 MPU を置くという構成を何段かくり返すわけである．同じレベルで隣接している MPU 間の通信は直接に，二つ以上はなれた MPU とはより上位の MPU を経由して通信を行う．

この構成によれば，PDE はもちろん解けるうえに，線形計算や回路解析のように必ずしも隣接通信に限定されない目的にも使える可能性がある．たとえば回路解析を例にとれば各サブシステム間の結合が出来るだけなくなるように (結合が局在するように) システム分解を行い．これに基づいて各ノードを各 MPU に割り振るとオーバヘッドはそれほど増えないだろう．

また PDE 解析に現われるノルム計算や収束判定なども，この上位 MPU を使って簡単に出来るかもしれない．

### 6.6 SIMD 対 MIMD

科学用並列計算機は SIMD 方式が有望という説をよく見かけるが果してそうであろうか．SIMD 方式により各 PU のハードウェアは簡単化するというメリットがあるが，現在の LSI 技術を前提にすればチップ内のロジックが簡単になってもそれほど最終的なハードウェアのコストにひびいてこない．

一方ソフトウェアに関しては，SIMD 方式の方が作りにくい．すなわち，すべての PU が同時に同じ動作を行うか，または停止するという建前であるから，各 $PU_i$ ごとに異なった分岐動作するときに問題が生じる．いま $PU_i$ 中のデータを $d_i, c_i, e_i$ とし図-4

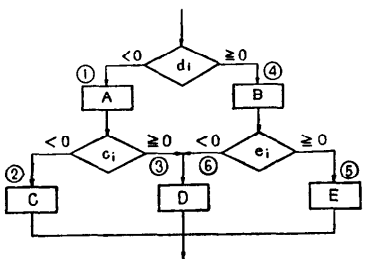



図-4 フローの分岐と合流の例

のような分岐があったとしよう．まず，$d_i \geqq 0$ である $PU_i$ はコントローラからの制御を受けつけないようにし (すなわち停止またはアイドル状態)，$d_i < 0$ である $PU_i$ でAを実行する．次に $d_i < 0$ なる $PU_i$ を停止し，$d_i \geqq 0$ なる $PU_i$ でBを実行することになる．すなわち処理は逐次的になってくる．もし 図-4 のようにAやBの先に更に分岐があれば同一動作を行う PU 群は更に細分化されて，ますます並列処理から遠のいてくる．

もし分岐した先の処理内容が同じものがあったら，それらを逐次的に処理するのは無駄だから goto 文を許してもらうことにしてフローを合流させDを実行するのは 1 回にすれば効率がよいがそのためには③からのフローを一旦おあづけにして⑥からのフローと同時に処理することになる．もしこれがもっと複雑になると，どういう PU の制御をすればよいのだろうか．構造化プログラムとやらで，何とか切りぬけられるのだろうか．

また各 PU ごとにデータの並び方が (発生がそもそも) ランダムになっていてこれを再配列しないと並列処理が出来ないことがあるだろう．この再配列を一般に SIMD で並列処理するのは難しい．ILLIAC-IV のコンパイラの例[15]を見ても，これらの問題についてお手あげらしくてユーザにすべて下駄をあづけている．結局ソフトウェアコストが高くつきハードウェアの簡単化のメリットなど打ち消されてしまうのではないだろうか．

この種の困難は PU が (マイクロ) 命令レベルで同期して動くという SIMD 方式の宿命であるが，またシミュレーションの方法としては極めて不自然なものである．自然現象は並列に動いているといっても，一斉に同じことをやったり，部分的に停ったりはしない．

一方 MIMD 方式は非同期方式であるから，命令レベルでは各瞬間に各 PU は別々のことをしてもよい訳で，より自然現象に近いシミュレーションが出来る．しかし解くべき問題には PU 間の相互作用が存在する以上何らかの同期をより上位のレベルでとらねばならないだろう．

私達の経験した原子炉炉心計算では並列化とその同期は，物理現象とのアナロジで容易に行えたし，またプログラム上もずっとわかり易い．これをもし SIMD 流にコーディングされたら，ずっと解りにくいし，実現も難かしいと思われる．MIMD 方式でプロセジュ

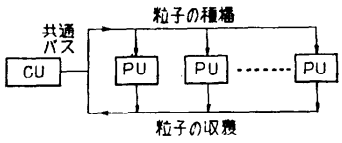



図-5 PACS のパイプライン的動作

アあるいはタスクレベルの同期をとる仕事はユーザの領分であって，いわば「物理」に属することと考えている．

### 6.7 ラグランジュ的シミュレーションへの応用

流体や多粒子系のシミュレーションには2種類のアプローチがあり，

（i） オイラー的定式化：空間に固定された座標系をとり，各点での場を表現する．

（ii） ラグランジュ的定式化：流れまたは粒子に乗った座標系によって場を表現する．

オイラー的な場合については今までに述べた通りであるが，ラグランジュ的シミュレーションを PACS で行うには，少し違ったモードの制御が必要である．まずそのまえに単純なケースとして相互干渉のない粒子のモンテカルロ法シミュレーションについて述べよう．

たとえば中性子輸送現象では粒子間の衝突は無視され，モンテカルロ法による計算では各粒子ごとに独立に確率実験が行われ集計される．したがって各PU間のデータ通信は行わないことにし，各PUはCUより乱数で発生させた一つの粒子をもらって（たねまき），その軌道を追跡し1つの粒子が死滅すればその結果をCUへ返し（収獲）また新しい粒子をもらう．すなわち各PUは1つのパイプラインを形成するわけである．粒子の運動を計算する時間の方が粒子のたねまき収獲に要する時間よりはるかに長いから，もしN個のPUがあり各PUがいつも1個の粒子を担当するとすれば，計算能力は単一PUのほぼN倍になるだろう．

ところで各粒子間には相互干渉が存在するのが一般的である．たとえば荷電粒子の輸送では各粒子の作る電磁界がまた各粒子の運動に影響を与える．運動方程式の1ステップの積分を行うまえに，その点における電磁界の値が必要となるが，これをCUで計算していたのでは計算のペースはCUで決ってしまう．しかし幸にして電磁界は線形で重ね合わせが効くから，むしろ各粒子の位置で並列に計算出来るはずである．すなわち全粒子の現位置と，前回の位置（電流分布を出すのに必要）を全PUへ放送する．これはCU共通バス経由の放送と，各PU間の通信路を使った回覧板方式が考えられる．グリーン関数がうまく求まるとすれば粒子の位置情報と自分自身の位置とから電界と磁界が各PUごとに計算出来るだろう．

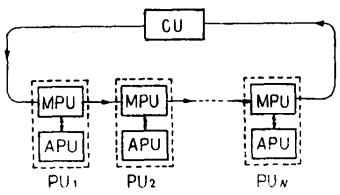



図-6 回覧板方式によるデータ交換

この位置情報の流通は浮動小数点演算素子で計算中に MPU を使って出来るだろう．もちろん MPU へのアクセスの整理は解決しておかねばならない．

### 6.8 性能評価，性能予測

PACS の現状は 3×3=9 個の PU より成り，MPU はモトローラ MC 6800，メモリは各 PU 当り，7.25 Kバイトをもつ．浮動小数点演算はソフトウェアで行っている．原子炉の拡散方程式（ポアソン方程式）を3次元 432 ノード上で解いたときの実測を表-1 に示す．言語やコーディングの方式が違っているので一概に比較は出来ないが部品のみで約 150 万円ぐらいの PACS-9が定価約 4,000 万円ぐらいの汎用 CPU の約 2/3 の能力に達しているというのは極めて印象的である．PU 内の計算にほとんどの時間を費やしているため相対的にオーバヘッドが少なく，並列処理効率は約 $\alpha$=99% ぐらいである．

現在 APU-LSI をつけ加え若干の改良を加えた 32 個の PU より成る PACS-32 システムを試作中である．PACS-32 について，拡散方程式を例にとって評価すると 0.38 MFLOPS，$\alpha$=83%，性能価格比=0.09 MFLOPS/M ¥ となり，ほぼ新型ミニコンクラスの性能がえられると予想している．また PASCAL 的なコンパイラを開発中である．

一般に並列システムの性能は PU 数Nに比例しないと言われているが，PACS 流のアーキテクチャではアクセス競合が起る資源はとくにないこと，その他の

表-1 PACS の性能

| 計 算 機 | PU 数 | 浮動演算 | $\alpha$ | 1 反復計算時間 | 全ノード数 |
|---|---|---|---|---|---|
| 中型汎用機 | 1 | ハードウェア | 100% | 1.6秒 | 432 |
| PACS-9 | 9 | ソフトウェア | 99 | 2.4 | 532 |
| PACS-32 | 32 | ハードウェア | 83* | 0.12* | 1,536 |

* 予測値

オーバヘッドもNに依存しないことから，全システムの性能は，ほぼNに比例して，すなわち各 PU レベルの性能価格比を保って，問題が要求する能力を実現することが出来よう．

従来からある並列処理への批判として，ジョブ中には並列化出来る部分は比較的少なく，並列化メリットは少ないというのがある[4]．これは命令レベルの並列化の話であるが，これをPUアレイによる PDE 解析にあてはめて見ると，たとえば並列化可能手続が全体の 2/3 のジョブがあったとして最大限並列化を行っても計算時間は 1/3 を割ることはない，と言うことになろう．

現存するいわばノード数nが中途半端の大きさであるようなジョブではこの理由で並列化メリットの少ないものもあるだろう．しかしユーザの心理としてはそれが可能ならノード数nを常に増そうと望んでいるのである．PDE 解析は大部分並列化しうる部分と考えてよい．たとえば私達が行った 432 ノードによる原子炉炉心シミュレーションでは並列化しうる部分が全体の 93% を占めた．これは PDE 解析ではかなり一般的に言えることであると思う．

## 7. 将来の課題と展望

数百〜数万個の PU を結合するためには，1ボードの PU を 1〜3 チップぐらいに LSI 化しなければならない．そうすればさらにチップ当りの性能価格比が上り，数万 PU の結合も容易になろう．

PU 個数Nに比例する性能がえられるという前提に立てば単位 PU の性能価格比がそのままシステムの性能価格比になる．現在1ボードを構成している約 100 個の LSI, IC 等を1万円の LSI にまとめられるとすれば，CRAY-1 の約数十倍の性能価格比がえられると期待される．より高速の素子を使えば，絶体的な性能は増すが性能価値比は果してどうなるだろうか．

この1チップ化 PU が出来ないと，実用化へのブレークスルーが生じないだろう．マイクロプロセッサ，アドレスつきの 16K バイトのメモリ，浮動小数点演算 LSI，外部へは約 100 ピンの1ボード PU を 1〜3 個のチップにまとめることである．現在これに合う仕様のチップはなく，またチップメーカもマーケットニーズをそれほど認識しているとも思えない．一方ほとんどのユーザは並列計算機などは遠い世界のことと思っており，たとえ関心があっても市場に現われてから使うことを考えているだけである．これではブレークスルーは生じないし，多分アメリカの軍需産業が手をつけるのを待つだけということになる．私達の経験ではチップを組み上げてゆくことと，ソフトウェア・システムにはあまり技術的困難はないという印象なので，ここで一つ，日本の LSI 技術によって将来の科学技術計算機のイニシアティブをとってはどうであろうか．

次に考えるべき課題は実装と RAS の問題である．実装方法については，素人の頭の中にあるアイディアにすぎないがプリント板を廃止してチップを直接コネクター中にはめこんではどうだろうか．そして子供のブロック遊びのようにブロック化した PU を組み合せて，自分の望む形状を実現するというのはどうだろうか．先日やってきた米国系コネクターメーカの話では，これは現在でもそろそろ可能ということであった．もし1チップ化が可能なら，いっそうのことコネクターも廃止して LSI のパッケージ自身をはめ合せ可能にしてはどうか．先日見つけた英国の文献[16]でそういう絵まで描いてあるのがあった．

もう一つの RAS に関しては1万 PU ぐらいのシステムから問題になってくるように思う．もし自動訂正機能や場合によっては 2 out of 3 のように多重化が必要であれば若干のコスト高になるかもしれない．誤りを起したブロックは CU またはブロック上の LED に表示すれば，修理は単にブロックの差しかえですむ．全体の形状は2次元アレイの場合，上下左右でエンドレスにするためにドーナツ状またはトポロジカルにそれと同等の形状にするのがよいと思われる．

現時点の市販 LSI をベースに考えれば，絶対的な速度で大型機なみの性能を出すにはかなり多数の ($10^3$〜$10^4$個) の PU を結合せねばならない．しかしこの種の試みは何もスーパーコンピュータのみを目指すべきものとは思っていない．ニーズの項では十分述べなかったが，プラントの計装システムの一部としてオンライン・リアルタイム的なニーズが存在する．これはミニコンに比較して性能が上まわればよく，100 PU 程度で十分性能価格比の良いシステムが実現する．そうすればミニコンをコントローラとするスタンドアロン型単能シミュレータとして普及するかもしれない．それは潜在的なマーケットの大きさと，いかに良好な性能価格比を保てるかによるだろう．

## 8. おわりに

科学技術計算に重きをなす偏微分方程式のシミュレーション専用のマイクロプロセッサ複合体について，

とくに私達の進めている PACS システムに即して概観した．たった9個のプロセッサを結合させた段階なのに，少し針小棒大がすぎたかと反省している．

汎用機に対しても不公平な書き方をしたかもしれない．汎用機の存在理由がなくなることはない．専用と汎用という概念は補完し合うものであろう．すなわち，1問題1マシンという拡散し切ったイメージではなくて，似かよった問題は共通のマシンの上で解かれるはずである．この両者の混り具合は果してどのあたりにおちつくのであろうか．

また専用機は特殊な分野の特殊な応用であり計算機の本流ではないという考え方もあろう．また実社会では事務計算が主流で科学計算などお付き合い程度にすぎないという話もよく聞く．しかし科学計算のシェアが小さいのは，やむをえず事務用機の出来る範囲で科学計算をやっていたからではないだろうか．私達はやっと科学計算に適した計算機を持とうとしているのだと思う．

**謝辞** 川合敏雄氏は，並列マイクロコンピュータシステムの可能性について熱心に説かれ，PACS の試作の原動力となられた．東野純一・澤田和男・山岡彰の諸氏はPACS の具体的な設計製作にあたられた．文中私達と記したのはこれらの人々の寄与を含んでいるからである．ここに記して深謝する．

## 参 考 文 献

1) Weiman, C. F. R. and Grosch, C. E.: Parallel Processing Research in Computer Science: Relevance to the Design of a Navier-Stokes Computer, Proc. of the 1977 International Conf. on Parallel Processing, Wayne Univ. and IEEE Computer Society pp. 175-182, (1977).
2) 元岡 達(編)：計算機システム技術，オーム社 (昭和48年).
3) 加藤満佐夫，苗村憲司：並列処理計算機・超高速化へのアーキテクチャ，オーム社 (昭和51年).
4) 村岡洋一：並列処理概論(1)—(3)，情報処理，Vol. 16, pp. 65-72, pp. 137-144, pp. 239-245 (1975).
5) 情報処理，専用プロセッサの方式とシステム構成特集号，Vol. 18, No. 4 (1977).
6) 飯塚 肇：マイクロプロセッサ複合体の技術，電子通信学会誌 Vol. 60, No. 2, pp. 125-135 (1977).
7) 松崎 稔：各種の形態をとるマルチマイクロプロセッサシステムが次々に登場，日経エレクトロニクス pp. 52-75 (1977年12月).
8) Gritton, E. C., et al.: Feasibility of a Special-Purpose Computer to Solve the Navier-Stokes Equations, R-2183-RC, RAND Corp. (June 1977).
9) Kober, R. and Kuznia, Ch.: SMS 201-A Powerful Parallel Processor with 128 Microcomputers, EUROMICRO JOURNAL Vol. 5, pp. 48-52 (1979).
10) Flanders, P. M., et al.: Experience Gained in Programming the Pilot DAP, A Parallel Processor with 1024 Processing Elements, in Parallel Computer-Parallel Mathematics, pp. 269-273, IMACS, North Holland (1977).
11) Flynn, M. J.: Some Computer Organizations and Their Effectiveness, IEEE TCC-21, 9, pp. 948-960 (Sept. 1972).
12) サザーランド，I. E.，ミード C. A.：コンピュータサイエンスの変貌，サイエンス誌 pp. 142-158 (1977年11月).
13) 星野，東野，山岡，澤田：連続体シミュレーションのための並列計算機 PACS の試作 (I)(II)，C1, C2，日本自動制御協会第23回システムと制御研究発表講演会 (昭和54年).
14) Händler, W.: Aspects of Parallelism in Computer Architecture，文献 10) と同じ，pp. 1-8 (1977).
15) Lawrie, D. H., et al.: Glypnir—A Programming Language for Illiac IV, Comm. ACM 18, 3, pp. 157-164 (March 1975).
16) Billingsley, J.: A System Design for the Combination of a Number of Computing Elements into a Powerful Structure, in Large Scale Integration, EUROMICRO, North-Holland, pp. 270-273, (1978).

(昭和54年7月24日受付)