

並列計算機 PAX による数値解析

星 野 力

(筑波大学構造工学系)

川合敏雄

(慶応義塾大学理工学部)

1. まえがき

元来計算機はまず目的があって作られた。バスカル、バベジ、ENIAC の歴史はそれを物語っている。ところが ENIAC 以降の電子計算機は汎用機となり、主としてビジネスのために用いられている。科学技術計算は今ではその片隅に寄生する小さな存在である。

けれども科学技術計算の需要が小さいわけではない。ノイマンを計算機に駆り立てたあの流体計算を例にとりて概算すれば、小規模のシミュレーションでも、所要演算量は 10^9 FLOP をこえる。科学、技術のあらゆる分野で 10^9 FLOPS をこえる計算機による「計算実験」が切望されている。科学技術への利用という問題を持つた目で見直せば、お仕着せの汎用計算機路線に安住しているだけではないと反省させられる。

10^9 FLOPS を超え、 10^{10} FLOPS 機をめざすにはどのような道があろうか、直ちに思いつくのは素子・デバイスの一層の高速化・小型化である。GaAs や Josephson 接合の開発がその方向であるが、あと 1~2 桁の改善しか期待できないと言われている。

もう一つの道は並列処理である。これも新しい考えではなく、ビット、命令、タスク処理の多様化はすでに採用されており、パイプライン方式も一種の並列処理である。また、自然の問題の中にあり並列性を利用した並列計算機の考えも、リチャードソンの数値的気象予報工場の夢の中に古くから登場していたのである。彼は 64000 人の計算手を大劇場に集め、地球上の地域を分担して近似計算手とデータを交換しつつ計算する様子を 1922 年に描いている^[1]。これはハードウェアに擬似すれば 64000 台の Processing Unit (PU) を二次元格子状に結合して MIMD (Multiple Instruction - Multiple Data) 方式で自分の仕事を実行するものであり、自然そのものの並列性にもとづくアーキテクチャといえる。

リチャードソンの夢は 50 年後に ILLIAC-IV としてハードウェアになった^[2]。

ただし SIMD (Single Instruction - Multiple Data) 方式という点では差があった。全 PU が同時に同一命令をする(かしないか)という点で窮屈な点が MIMD に劣る。このプロジェクトはハードウェアでいくつかの画期的進歩の芽を育て、その波及効果は今日にも及んでいるが、目標を達せなかったという点では失敗とされている。ハードウェアの技術の未熟のため、大規模システムの信頼度は低かった。しかしさらに本質的には並列アルゴリズムや並列言語に関するソフトウェア面での経験不足を挙げるべきであろう。我々は教訓として問題 (Application)、アルゴリズム (Algorithm)、計算機アーキテクチャ (Architecture) の三つの A の関係について、深い考察が必要なことに気をつけなくてはならない。

今日では LSI 技術の急激な進歩により、高度のマイクロプロセッサを多数並べた超並列方式ハードウェアの建設が可能となりつつある。また SIMD でなく柔軟に MIMD で計算する方式も可能である。こうして超高速で経済的な科学技術用計算機はいよいよ現実のものになりつつあると考えられる。ハードウェアにも問題はもちろんあるが、ソフトウェアの問題が超並列方式の鍵を握っている。

PAX は 9 台、32 台 [3]、128 台 [4] 並列へと発展し、また商用の PAX 64J [5] も設置されている。その主目的は科学技術計算用としてソフトウェアの問題を実証的に研究することにある。以下に現在までの成果を、アルゴリズムに重点をおきつつも、やや総合的に紹介する。

2 科学技術計算

2.1 計算の対象・問題の型

自然は原子より成る。原子、分子、粒子は他の粒子と力を及ぼしつつ発展するのが自然の姿である。もちろん粒子の数は非常に多く、衝突も非常に頻繁だから現象を統計的に扱って連続体と見做すこともある。このときだけ空間に連続的に分布する密度とか温度のような量を考える。こうして粒子モデルの他に分布モデルが現れる。実は粒子といえども量子力学までまかのばれば波動関数という分布系で記述されている。だから分布系一元論も原理的には可能といえる。しかし計算の立場からは分布系も離散化して解くしかない。だからすべては離散化した系（粒子系も含めて）に一元化される。

離散化した量の間の相互の関係にはいろいろある。まず分布系の場合に近接作用が普通である（近接型）。粒子系では引き合い屋などからすぐわかるように遠隔作用が普通である（遠隔型）。ほかに電気回路のように、素子の入出力の結合関係が不規則なもの（不規則型）もあるし、また相互作用のない独立型もある。四型のいずれにせよ自然を記述する変数は相互作用しつつ時間的に変化する。

自然のもう一つの特徴は、系の自由度（独立な変数の数）が膨大なことである。流体の場合に格子点の数は $(100)^3$ 以上を要するだろう。その各点に密度・速度・圧力・温度などがある。分子力学では粒子数は $(100)^3$ 個は必要であろう。これらの量は法則に従って自律的に自らの未来を決定してゆく。したがってそれを並列的に計算することが出来る。

自然がこのようなものであると考えれば、それを対象とする計算機は多数の演算装置がデータを交換しつつ並列に計算するアーキテクチャが当然のものと考えられる。ただし早合点してはならないのは、自然の構造と計算の構造（アルゴリズム）が必ずしも 1:1 に対応しない点にある。自然が分布系であっても、離散化の方法が差分法なら近接型、境界要素法なら遠隔型、固有関数に展開すれば独立型または不規則型、有限要素法なら近接型または不規則型になる。粒子間の力も近接型の方程式から求めることもある。なぜ現象とは違う型のアルゴリズムを考える必要が生ずるのかを以下で考えよう。

2.2 アルゴリズムの型

偏微分方程式で記述される分布型。たとえば

$$\frac{\partial A}{\partial t} = \frac{\partial A}{\partial x} + \frac{\partial^2 A}{\partial x^2} + S \quad (1)$$

を解くのに、時間については前進差分、空間については中心差分をと、

$$\frac{A_i^{n+1} - A_i^n}{\Delta t} = \frac{A_{i+1}^n - A_{i-1}^n}{2\Delta x} + \frac{A_{i+1}^n - 2A_i^n + A_{i-1}^n}{(\Delta x)^2} + S_i^n \quad (2)$$

とすれば、点 i における $n\Delta t$ 後の値 A_i^{n+1} は近接点の現在の値 (A_{i-1}^n, A_{i+1}^n) から定まる。定常問題はその定常値、固有値問題は（時間微分を二階化するが、時間を $\sqrt{\Delta t}$

と虚数におまかえて) その振動数をみればよい。こうして近接型を発展的に解くアルゴリズムが出发点となる。

ところがこの解法では数値解析的安定性の点から Δt の値が制約を受ける。 Δt の大きさは「自然時定数」を空間分点数の1乗ないし2乗で割ったものであるが、一つの問題の中で多くの自然時定数が混っているとすればその最小の時定数で Δt が抑えられる。最小と最大の時定数の比が著しく大きいとすれば、必要な時間だけ計算するのに要する時間ステップが過大となって實際上計算不可能となる。このような「stiff」の問題に対する一つの対策は、時間を後退差分として

$$\frac{A_i^n - A_i^{n-1}}{\Delta t} = \frac{A_{i+1}^n - A_i^n}{2\Delta x} + \frac{A_{i+1}^n - 2A_i^n + A_{i-1}^n}{(\Delta x)^2} + S_i^n \quad (3)$$

とおく「陰解法」である。これは $(n-1)$ 時点の値から n 時点の値を求めるもので、 A^n についての連立方程式を解く手数は大きい。安定性の心配が要らない。

連立方程式を解く必要性はこのほかにも定常値だけがほしいときに発生する。線形化された連立方程式は行列の形であらわされ、近接型は帯行列、遠隔型は密行列、不規則型は疎行列となる。その並列解法は型によって異なり、それぞれ反復法と直接法がある。いずれの型にせよ、各PUは特定の変数の計算に責任を持ち、それに必要な他のPUの変数を放送(遠隔の場合)または近接転送によって入手する。偶奇(多色)SOR、ガウスザイデル、共役勾配法は近接作用の問題を反復法で解くのに使うことができ、ガウスジョルダンは密行列を直接法で解くのに適する。そのまゝでは並列処理が難しい漸化式(三項対角方程式のガウス消去)などは、よく知られたCyclic reduction法による並列化が可能である。ただしこれは計算機によってほとんど使いものにならない(後述)。不規則結合の問題は密行列の一種と見做せば、並列計算ができないうけではない。しかし無駄な計算や転送の比率が多く、一般には並列向きとは言えない。したがって一般の疎行列問題のために一般的に有効性の保証できる並列計算機はHEPやデータフロー型にならざるをえないであろう。けれども不規則型問題は、逐次式計算機でも難しいことが多く、難しさは並列式に限った話ではない。

アルゴリズムはアーキテクチャとアプリケーションを結びもので、アーキテクチャがアプリケーションとかけ離れているとまたアルゴリズムはそのしわ寄せを受けて難しくなる。これまでは不思議とは思わなかったが、格子型アーキテクチャの出現によって負がつく一例は有限要素法における節点番号である。汎用機のメモリ構造は一次元的であり、処理は逐次的である。このことから節点番号も一次元的な通し番号になっているので、原問題における近接性が見えなくなり、帯幅が不規則になり、三次元問題が解まにくく、多次元的な番号づけの方が並列レイ型では自然にアルゴリズムが記述出来る。汎用的な逐次式アーキテクチャが、かえって自然な幾何な構造表現を妨げていることもあるという例である。

3. 並列アルゴリズム

3.1 評価の方法

これまでの並列アルゴリズム研究は

- (1) 演算要素の数Pは無限であり、
- (2) 記憶装置で他の演算要素からデータは瞬時に取り寄せることができ、
- (3) 演算にはすべて同じ時間がかかる。

というような仮定にもとづくものが多かった^[6]。これでは以下のような並列処理にとって切実な問題には答えられない。

- (1) どの仕事をどの演算要素が担当するか。
- (2) 演算要素をどう結合し、その間でデータをいかに交換するか。
- (3) 演算要素間のデータ転送、演算装置の遊休などによる無駄(オーバーヘッド)はどの位か。
- (4) そのオーバーヘッドは演算要素の内部で行われる有効な計算に比して定量的に無視できるかどうか。
- (5) これらの無駄も含めた計算所要時間はいくらか。

たとえばある種の並列計算法は転送が多くて実効がよらないというような報告^[7]をみると、あまりかに作業分担が不適切な場合があり、注意しなくてはならない。

3.2 並列処理の性能指数

演算要素を P 台、問題規模を N とする。問題規模とは未知数、粒子、格子点の数などである。この問題を逐次式計算機で解くのに要する時間を T_1 秒とし、並列計算機で解くと T_P 秒かかったとする。速度向上比 S は

$$S = T_1 / T_P \quad (4)$$

と定義する。次に

(5)

$$E = S / P$$

を効率と定義する。これはまた $T_{net} = (\text{全 } P \text{ の正味計算時間の累計})$, $T_{ovh} = \text{平均無駄時間}$ として

$$E = T_{net} / (P \cdot T_P) = 1 - \frac{T_{ovh}}{T_P} \quad (6)$$

ここに無駄時間の内訳は、 P 台間データ転送、作業不均等のための待ち時間などである。 $T_1 = T_{net}$ のとき式(5)と式(6)の E の定義は一致する。これは逐次処理と並列処理の正味計算が同内容(同アルゴリズム)であるときである。もしアルゴリズムの違いがあれば、言語やアルゴリズムの差によるソフトウェア効率(並列用と逐次用のアルゴリズム差が効くことがあるが、逐次計算の方が無駄が多いこともある。)を用いて式(5)の E は $E = E_a E_s$ と表すべきである。ここに E_a は式(6)で表されるアーキテクチャ効率 E である。 T_P , S や E は一般に N , P のほかハードウェアの特性値・アルゴリズム・問題などの関数であり、これをスケーリング則と呼ぶ。とくに N , P 依存性については3つの立場から論ずることが出来る。

- (1) すでに並列計算機を買った人は、 P が一定との条件で $E(N)$ 等を論ずる。
- (2) コンピュータ学者は問題 N が与えられたとして、 P が多くなると効率がいかに低下するかを考える。
- (3) 大規模計算ユーザーは、できるだけ大まかい問題を解きたい。すなわち N/P をメモリなどで決まる実用限界と固定し、 P を変数として効率を検討する。

3.3 並列処理のレベルと粗さ (Granularity)

並列演算のレベルには計算機ハードウェアから応用問題までいろいろある。問題固有の並列性を *inherent parallelism*, 計算機が実現する並列性を *machine parallelism* と呼び区別する。現在では前者が後者より圧倒的に並列度が高い。計算機中では論理演算、ミクロマクロの命令の実行、項・式・手順・副問題・全問題のレベルで並列に処理される。たとえば行列 $A \cdot B$ の並列処理では N^2 の演算要

素により項 $a_{ik} a_{kj}$ を並列に、内積 $\sum_k a_{ik} a_{kj}$ を N コの演算要素により、1つの行 ($\sum_k a_{ik} a_{kj}$, $j=1, \dots, J$) を N コの演算要素により計算させることも考えられる。また行列の積という仕事が部分行列に対していくつかの演算要素ばかり分けて実行することも考えられる。しかし現実には提供される *machine parallelism* P は与えられたものであるとすれば、なるべく効率のよいレベルまたは粗さに分けることになる。

3.4 コンシステントな並列アルゴリズム

問題規模 N が無限大の極限で並列処理時間 T_p の複雑さの N に関するオーダー $O(T_p(N))$ が逐次処理のそれ $O(T_1(N))$ と等しいような並列アルゴリズムをコンシステントであるという^[8]。並列計算機の教科書^[9]ではカスケード加算や三重対角方程式の *cyclic reduction* では $N \rightarrow \infty$ で逐次近似より低速になると書かれているが、それはこれらのアルゴリズムがコンシステントでないからである。巨大な N をもつ問題を並列計算機で解こうというのに、そのようなものは役に立たない。コンシステンスは大切な概念である。

カスケード加算について考えよう。 $N = 2$ (整数) コの数を加えるのに、 $P = N$ コの演算要素がリング状につないである並列計算機を考える。1PUが1データを持つ。まず一定方向に全PUが1つ隣のPUに自分のデータを送り、受け取ったデータを自分のデータに加える。めいめいが隣の2数の和を得たわけである。次にその小計を全PUが一斉に一定方向に2つ隣のPUに送り自分の小計に加えて中計を作る。再び中計を4つ隣に送る。これを $\log_2 N$ 回くり返すと総計ができる。加算は $\log_2 N$ 回で済む。データ転送は、一つ隣に送るのを1単位として $N-1$ 回必要である。一方逐次計算機では $N-1$ 回の加算が必要となる。(転送に相当するものとしてメモリへのアクセスオーバーヘッド(バンクコンフリクト)がありうるが、逐次型に花をもたせて考慮しないことにする)。これより $O(T_p) = O\{C \log_2 N + D(N-1)\}$, $O(T_1) = C(N-1)$ よりカスケード加算はコンシステントのように見える。

けれども現実には $N \gg P$ であり P は一定数なので $O(T_p) = O\left\{\frac{C \log_2 N}{P} + \frac{D(N-1)}{P}\right\}$ となり、カスケード加算をデータ単位で単純に行えばコンシステントにならないのである。しかしこのアルゴリズムをコンシステントにする方法がある。まず1コのPUはあらかじめ自分の持つ N/P コのデータの小計を作る。この部分は完全に並列にゆく。 N が大きいとき、この部分は $O\{C(N/P-1)\}$ で計算の主部となる。このあと小計をカスケード加算する部分は $O\{C \log_2 P + D(P-1)\}$ となり、全体のアルゴリズムはコンシステントとなる。この方法は一般化することが可能で、後述のADIアルゴリズムのGEQRによる並列化がその例である。またパイプライン式コンピュータで総和をとるときも、同様に P (パイプ段階) ごとの部分和をとり、あとで逐次的に部分和を加算しているが、これもコンシステントな方法である。

並列ゆえの無駄をカバーするためには、 P は十分大きいことが必要である。いまそのための P の条件を求めよう。^[10] 逐次計算で計算所要量 aN の問題が並列計算では所要量が全体で $bf(N)$ になったとする。これを P 台のPUで並列処理すると速度向上比は $S = aN / \{bf(N)/P\}$ となり、 $S > 1$ のためには

$$P > \frac{a}{b} \frac{f(N)}{N}$$

でなくてはならない。 P が低い値(5とか10)の場合、この条件はみたされないこ

とが多い。パイプライン式コンピュータでcyclic reductionが使い物にならないのは、並列度が低いためであり、アルゴリズムのためではない。コンシステントなアルゴリズムでは $f(N)/N$ は有限にとどまるので、台数を増せば原理的には必ず並列計算機の方が逐次計算機より速くなる。

4 PAX

PAXはユーザーからの発想で、実用的に役立つことを目指した格子型並列計算機である。近接作用系はもちろん、遠隔型、独立型の問題に効率よく適用できる多目的科学技術計算機である。この範囲だけでも解くべき巨大計算は多い。MIMD方式としたのはいろいろな点でSIMDに勝るためである。

分布系の問題では担当する領域の変数により計算法が異なるし、粒子系の問題では条件ジャンプが多い。もっともPAXの使い方の実際を見ると、同期をとりながらのMIMDであって、擬MIMDと言うべきだとの指摘もある。並列処理の粗さはILLIAC-IVにおけるSIMDよりはるかに大まい。純正MIMDのモードで使うこともできるが一貫して完全に非同期で働くプログラムは作りにくく、デバックが困難なことが多い。

同期はハードウェアにより全PU間でとられ、大した時間を要しない。同期に時間をかけないために同期を1つのセマホ(信号)とするデータ転送も速い。放送は遠隔作用型アルゴリズムや初めに共通のプログラムを全PUに渡すのにも使われる。

PAXの構成図やPUの詳細[3][4][5][11]についてはここでは省略し、表1にハードとソフトの概要をまとめた。

トラス状に端と端を結んだのは、ノルム計算、遠隔型問題や同期境界条件の問題に極めて有用であったからである。

2次元格子として、1次元や3次元をとらなから、理由はデータ転送にある。Pコ(のPU)を1次元に並べると、その最大距離は $P/2$ であるが、2, 3次元では $P^{1/2}/2$, $P^{1/2}/2$ である。たとえば $P=1024$ とすると、これは512:16:5となる。1次元から2次元への改善は著しいが、2次元から3次元への改善は僅かである。一方3次元実装は技術的に困難である。3次元の結合に $P>10^4$ となる将来の課題である。

ユーザーはホスト用プログラムをFORTRANで、PU用プログラムをSPLMで書いてきた(PAX32と128の場合)。PAX64Jではメモリ空間は統一され、ホスト用を主プログラム、PU用をサブルーチンとしてFORTRANで統一的に書くことになった[5]。このFORTRANはDEC提供のふつりのものであり、特別な並列言語を持ってはいない。

ふつうは全PUに共通のプログラムをロードする(実行はPUごとに後分かれすることもある)。PU用プログラムは隣接のCommunication Memory(CM)への代入、CMからの読み込みを記述する。このとき局所的にデータは授受され、全PUにわたる動作はない。分布系の場合に分担した領域の隣接点のデータを隣接PUからもらってくる必要があるが、このようなルーチン作業は既成のサブルーチンと呼ぶことによって可能である。

PAXのソフトウェアは使いやすく考えやすい。熟練の度合と問題の難易にもよるが1~30日で使いこなすに至る。現在のソフトは単純だが役立っている。しか

しもっと凝った知的並列言語の可能も考えられよう。

5 今までに行われた応用

5.1 実験的方法論

以下の実験的問題にPAXを適用した。もちろん実験機の制約のため問題の規模 N とハードウェアの規模 P は小さいが、それを外挿するためのデータを実験的にとって、スケーリング則(実行時間や効率を N , P その他のパラメータの関数として表したもの)を作るようにつとめた。上に述べてきたようなオーダーの議論も大切だが、並列機の性能を具体的に定めるのはそれにかかる係数の大まきである。それによつて並列機の設計において留意すべき技術的パラメータが明らかになるわけである。

5.2 実験・経験の結果

表2に応用問題と主要結果をまとめた。これらは典型的な実用問題を代表的なアルゴリズムで解いている。近接型でない物理的問題やアルゴリズムには、PAXは不得意であるとの考えが従来から存在するが、必ずしもそうでないことがこの表から見られるであろう。むしろ規則的な大規模問題はすべて効率よく解ける、と簡潔に要約できよう。

この表のパラメータ P と N の定義はすでに前に述べた通りである。以下効率を外挿するとすは N/P を一定に保ち、 $P \rightarrow \infty$ として論ずる。(大規模計算ユーザーの立場)。

(1) SOR法によるポアソン方程式[4][11]

陽的反复解法、近接型の地域分担モデル。

オーバーヘッドは近接データ交換であるが、各PUの正味の計算は分担する点の数 N/P に比例し、交換量は周辺の点の数に比例するから、効率はPUあたりの格子点の数 N/P が多くなれば向上し、 N/P を一定とすれば効率は一定となり台数に比例する速度向上となる。PAXの最も得意な問題である。

(2) ADI法によるポアソン方程式[2]

x , y 方向の一次元問題を交互に解いて二(三)次元問題を解こうという方法である。ILLIAC-IVでは有名なデータの斜め配置(*Skewed-Storage*)が考案された[6]。これにより図1のように P 台の1次元PUアレイ(リング状結合)により、ADIは並列度 P で計算出来る。もちろん共有メモリをもつコンピュータ(現在のスーパーコンピュータや、ILLIAC-IVでも)では、メモリ・プロセッサ(またはパイプ)間転送時に、データの転置をやってしまうので、この*Skewed Mapping*が実際に使われた例は少ないと思われる。しかしデータをPUに分散してもつ高並列アレイ型では、この*Skewed Mapping*は(考慮に値するが)次の点で魅力に欠ける。

① 並列度は、物理空間の1次元方向格子点数が上限となるが、応用問題(たとえばNavier-Stokes方程式)では全空間の格子点数の並列度で処理される部分がほとんどを占める。

② 1次元*mapping*では、1台のPU当たりのメモリと速度への要求が過酷である。むしろCyber Plus(スーパーコンピュータ×16台リング結合)のようなアーキテクチャに向いている。

PAXのような高並列指向のマシンには、やはり物理空間をそのままPUアレイ

上に写像する図2のような直接写像が自然である。ここではよこ(たて)1列のPUによる三項対角方程式の並列処理が問題となる。しかしこれもガウス消去を併用した *cyclic reduction* で処理できる[12]。この並列ADIは *cyclic reduction* を用いているにも拘らずコンシステントである。その具体的アイデアはPUの分担する領域の境界上の点の値がわかれば内点計算は独立に並列計算ができることにある。そこで

① ガウス消去法によりPU領域の内点を消去し、境界点間方程式の係数を求める。

② *cyclic reduction* により、未知の境界点の値を求める。

③ その結果を使って内点の値を後退代入により求める。

の順で計算する。①③は完全に並列であり $O(N^2/p^2)$ の計算量と $O(N/p)$ の転送量をなす(問題サイズ $N \times N$, PUアレイサイズ $P \times P$ としている)。②はデータ転送を伴い、PUの遊びも生ずる部分である。それでも内点消去のおかげで *cyclic reduction* の問題規模が P と小さく、②の部分の計算量は $O(\frac{N}{P} \log_2 P + \frac{N(P-1)}{P})$ となっている。全体のスキームは $O(N^2/p^2)$ であり逐次処理のそれ $O(N^2)$ と比べて、コンシステントであることがわかる。

(3) Beam-Warming 法による Navier Stokes 方程式

この空気の学のための陰解法は本質的にADI法であり、PAXでうまくいく。他方 Mac Cormack 法は著名な陽解法であり、よく工学で使われている SOLA スキームは陽的解法(Navier Stokes を陽解法、連続の式はSOR(陽的反复)による陰解法で計算する)でありもちろんPAXで効率よく解ける。

(4) Gauss-Jordan 法による線形連立方程式[11]

未知数をPUが担当する。したがって行列を横に切って、PUはいくつかの行と対応するベクトルの要素を担当する遠隔結合型アルゴリズム。密行列の場合は無駄なく、逐次式計算機と同じ計算量で済む。オーバーヘッドはピボット選択のためにPU間の最大値探索と、ピボット情報の放送の際に生ずる。前者は $\sqrt{P}$ 依存、後者は N^2 依存の作業量であるが、計算量 N^3/p に比較すれば小さい。コンシステントであり、効率は一定値に漸近する。

(5) Conjugate Gradient 法による連立一次方程式[11]

反復解法。未知数をPUが担当する。PUは自分の担当未知数を計算するのに必要な定数は内蔵しているが、関連未知数を近接もしくは遠隔(問題が近接型か遠隔型かによる)のPUから受け取る必要がある。しかしこれらは効率をゼロにするものではない。何故なら正味計算は $N/p = n$ として $O(nN) = O(n^2P)$ となりデータの放送の部分は $O(nP)$ であるからである。

(6) モンテカルロ法によるスピンモデル[13]

スピンの向きを、その近接のスピンの向きから統計的に決めようという磁性体シミュレーションである。素粒子研究のための格子ゲージ理論でも似たモデルの大規模計算が必要とされる。いうまでもこれはPAXのための良い問題であり、近接型発展解法で単純に効率よく解ける。オーバーヘッドは相関距離計算のために遠隔データまでもらってくることに生ずるが定量的には問題にならない。格子ゲージモデルでは多次元空間を2次元のアレイへ写像するので効率は1に漸近する。

(7) 分子力学[14]

これも物性論、化学で重要性を増しつつある分野である。分子間の力が与えら

わていれば古典力学に従ってその運動を追跡し、相転移や物性値、化学反応を予知する。

PUへの仕事の割り当ては二種類ある。ラグランジュ方式ではPUは粒子を担当する。その運動のために必要は他の粒子の位置をPUアレイ中で循環させる。力の計算が所要演算量の主要部でこれは N^2/p 回に比例する。遠隔粒子の力は無視するとしても、距離判定のためにやはり $O(N^2)$ の計算が必要である。データ循環は $O(N(P-1)/p)$ にすぎないので効率はいに漸近する。

これに対してオイラー方式では領域分担である。運動の結果粒子が領域境界を越えるか否かを監視して必要なら隣のPUに移動してやるとか、PUの担当粒子数に不均一が生じて一部に遊びが生ずるという無駄が生ずる。力を計算するためのデータ転送は近隣のいくつかのPUでのみ行えばよい。もし近接(ポテンシャル到達範囲内)の粒子数が N に比例すればやはり $O(N^2/p)$ となり、もし N に比例せず一定ならば $O(N/p)$ となりラグランジュ方式に比べて有利になる。二つの方法の優劣は問題のパラメータに依存して一概には言えない。

(8) アラズマの粒子シミュレーション(FFT)

クーロン力を受けて運動する電荷の集団はアラズマ物性の基礎であり、巨視的アラズマ挙動のための物性値を提供するために必要とされる。計算は電荷分布から場を求め、場から電荷分布を求めるという二段階をくり返す。第1段階はポアソン方程式を解く仕事、第2段階は単なる *particle push* であって分子力学(ラグランジュ方式)と異ならない。

第1段階は波数空間で解いた。周知のようにポアソン方程式は波数空間では単なる割算で解けるが、そのためには現実空間との間にフーリエ変換と逆変換が必要になる。PUアレイ上で高速フーリエ変換を並列処理することはILLIAC-IVで実用された。FFTではデータ転送が頻繁に起こり、その状況はちょうどカスカード知と同じである。したがって効率はこの部分については $1/p$ に低下する。また粒子分担の不均一による効率低下もある。しかし定量的にスケーリングを考えると、今のPAXのPUハードウェア性能を仮定すると p が数千程度までは、実用的な計算ができる。

(9) 河川の不定流のシミュレーション

河川の水位変動の方程式は一次元移流型の偏微分方程式で記述される。そこでPUも一次元的結合と見做し、パイプライン的处理により、この近接型方程式を解けばよい。PAXによるパイプライン処理は一般的(MIMDパイプ)で、IF文

IF $p < t$ then (p, t) の位置を計算する. else 何もせぬ
(こゝに p はPU番号, t は時間ステップ)

で可能である。

移流方程式は性質が悪く、数値解析にはノウハウが多く、河川学者には固有の解き方がある。この差分方程式は前後4点とリンクする漸化式であり、通常のパイプラインスーパーコンピュータでは処理は困難である。

(10) 論理回路シミュレーション

LSI開発に現れるこの問題は不規則型結合のきどわい問題である。PUは大体同じ数のデバイスを担当し、データは結合表によって非同期に転送し、データフロー的に処理する。近接PU間のデータ転送はまたFIFOにより行われるとしたエミュレーションも行った。一般にPUアレイを横切る転送オーバーヘッドは $\sqrt{p}$

に比例するので効率 E は $E = 1/(1+c\sqrt{p})$ の形になる。すなわちある P_t を境にして

$$E = O(1/\sqrt{p}) \quad P > P_t$$

$$= O(1) \quad P < P_t$$

すなわち少なくとも P 台の PU を用いて $\sqrt{p}$ 倍にはなるのである。このタイプの一般的な不規則問題の例として目下 LSI 設計の配置配線問題を解いている。

(II) その他

一次元気力学、二次元 MHD 、二次元の海岸波、三次元沸騰水型炉心シミュレータ(均など分布系、第2種 *Volterra* 型積分方程式(遠隔型)、磁場中の電荷の漏洩確率(独立型)などを計算した。

5.3 要約

(a) 命令系を近接モデルで解くとまは効率は高い。

(b) 粒子系を遠隔モデルで解くのも、うまくゆく。

(c) 粒子系を領域分担で解くとまは分担均一化が問題になる。これにはモデュラ画像(格子点 (i, j) を $PU(mod(i, P_i), mod(j, P_j))$ に割り当てる画像。ただし PU アレイサイズを $P_i \times P_j$ とする)を検討することが残された課題である。

(d) 大規模問題 (N/p の大きい) ならば① *cyclic reduction*, ② ノルム計算, ③ 内積、和、ビット選択、④ *FFT* などの効率は高い。

(d, 1) もしノルム計算のために PU アレイを横切ら転送 $O(\sqrt{p})$ が生ずるとま正味計算 $O(N/p)$ (一定) に対してオーバーヘッド比率は $p^{1/2}$ に比例して増大する。しかし PU アレイの次元 q (または2) よりも高次元 (q 次元) の空間を直接画像すると正味の計算は $O(p^{q/2-1})$ となる。たとえば $q=3$, $q=2$ では正味計算は $O(\sqrt{p})$, $q=1$ のとき $O(p)$ となり、 $O(\sqrt{p})$ のオーバーヘッドと同じかそれより高次になる。従って効率は一定値 $1/p$ に漸近する。高次元連続体モデル、連立方程式の行分担、格子ゲージ理論がこの例である。

(d, 2) もしノルム計算ではなく、全データが PU アレイを横切ら転送に参加するとき、効率は低下する。たとえば *FFT* がこの例である。

(e) たとえ並列アルゴリズムが、コンシステントでなくても、それをコンシステントにする方法があると期待される。すなわち逐次処理の複雑さを $O(S(N))$ としたとき、

① PU 内部の前処理 $O(S(N/p))$ により未知数を $O(g(p))$ に縮小する。

② 縮小した未知数を $O(g(p))$ の並列処理により求める。

③ 代入して内部未知数を $O(S(N/p))$ で求める。

(f) 一般的な $MIMD$ バイアライン処理が、簡単に出来る。

(g) 不規則型問題の非同期処理も結構実用的 ($O(\sqrt{p})$ の速度向上) である。

7. おわりに

並列処理の世界は人類にとって未知であり、多くの神話が語られている。神話(迷信)を退治するには実証あるのみである。近接格子型の PAX は、近接型のみならず遠隔型、独立型、および一部の不規則型にも十分有用であることが実証された。確かに残された課題は多いが、(固有値問題、自動画像プログラムシステムなど)、今までになかった応用から少なくとも「 PAX 型はデータ転送がネックに

な、て実用化しない」というのは、迷信にすぎないことは明白であろう。

また新しい並列アルゴリズムの開発にも多くの可能性が残されていると思われる。本文中で述べたコンシステント化アルゴリズムというのは、一つの指導原理になりうると思われる。

アルゴリズムの最終評価は、実用的な応用でなされるべきだというのが叙述の立場である。PAXは陽解法にすぎているとい、ても *Navier-Stokes* 方程式を並列処理した今のまでの経験では、陰解法 (*Beam-Warming* 法) の方が格段に優れていた。やはり、実験による実証が大切なゆえんである。

PAXはすでに4号機が稼働し、商業用の開発もすすんでいる。並列アルゴリズムの開発を志すすべての人々に、PAXは開放されている。

参考文献

1. L.F. Richardson; *Weather Prediction by Numerical Process*, Dover Publications, New York, 1965, Ch. 11/2, pp.219-220.
2. R.M. Hord (ed.), *THE ILLIAC IV The First Supercomputer*, Computer Science Press, Rockville MD USA, 1982, Ch. II, pp.3-17.
3. T. Hoshino, T.Kawai, T. Shirakawa, J. Higashino, A. Yamaoka, H. Ito, T. Sato, K. Sawada, PACS, A Parallel Microprocessor Array for Scientific Calculations, *ACM Transactions on Computer Systems*, Vol. 1 No. 3, 1983, pp. 195-221.
4. T. Shirakawa, et al., Processor Array PAX-128, *Transaction of the Institute of Electronics and Communication Engineers of Japan*, J67-D, 8, 1984. pp. 853-860.
5. T. Shirakawa and T. Hoshino, Processor Array PAX-64J, *Proc. of Architecture Work Shop in Japan*, Information Processing Society of Japan, Nov. 1984.
6. D. Heller, A Survey of Parallel Algorithms in Numerical Linear Algebra, *SIAM Review*, 20, 1978. pp.740-777.
7. W. M. Gentleman, Some Complexity Results for Matrix Computations on Parallel Processors, *J. ACM*, 25, 1978. pp. 112-115.
8. J.J. Lambiotte Jr., The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer, *ACM Transactions on Mathematical Software*, Vol. 1, No. 4, Dec. 1975, pp.308-329.
9. R.W. Hockney, C.R. Jesshope, *Parallel Computers*, Adam Hilger, Bristol, 1981, Ch.5, pp.268-269.
10. R.H. Barlow, D.J. Evans, J. Shanehchi, Comparative Study of the Exploitation of Different Levels of Parallelism on Different Parallel Architectures, 1982 International Conference on Parallel Processing, Aug. 24-27, 1982, pp.34-40.
11. T.Hoshino, T.Shirakawa, T. Kamimura, T. Kageyama, K. Takenouchi, H. Abe, S. Sekiguchi, Y. Oyanagi, T. Kawai, Highly Parallel Processor Array "PAX" for Wide Scientific Applications, 1983 International Conference on Parallel Processing, Aug. 23-26, 1983, pp.95-105.
12. T. Kamimura and T. Hoshino, Processing of Alternating Direction Implicit(ADI) Method by Parallel Computer PAX, *Transaction of Information Processing Society of Japan*, 26, 1, 1985. pp.19-24.
13. T. Hoshino, S. Majima, K. Takenouchi and Y. Oyanagi, Monte Carlo Simulation of A Spin Model on the Parallel Computer PAX, *Computer Physics Communication*, 34, 1, 1985, pp. 31-38.
14. T. Hoshino, K. Takenouchi, Processing of the Molecular Dynamics Model by the Parallel Computer PAX, *Computer Physics Communications*, Vol. 31, No. 4, 1984, pp. 287-296.
15. T. Hoshino and T. Shirakawa, Load Follow Simulation of Three-Dimensional Boiling Water Reactor Core by PACS-32 Parallel Microprocessor System, *Nucl. Technol.*, 56, 1982, pp. 465-477.
16. J.H. Ericksen, Iterative and Direct Methods for Solving Poisson's Equation and Their Adaptability to ILLIAC-IV, *CAC Document No. 60*, Center for Advanced Computation, Univ. of Illinois, 1972.

The diagram illustrates the relationship between a 2D grid of points and a 1D sequence of points. The 2D grid is labeled with indices 1, n, 2n, 3n, ..., N-n, N. The 1D sequence is labeled with indices 1, n, 2n, 3n, ..., N-n, N. The mapping is indicated by arrows and the text "物理空間" (Physical Space) and "空間" (Space).

2D Grid (Left):

A_1	A_2	A_3	$\dots$	A_p
B_1	B_2	B_3	$\dots$	B_p
C_1	C_2	C_3	$\dots$	C_p
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
Z_1	Z_2	Z_3	$\dots$	Z_p

1D Sequence (Right):

A_1	A_2	A_3	$\dots$	A_p
B_1	B_2	B_3	$\dots$	B_p
C_1	C_2	C_3	$\dots$	C_p
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
Z_1	Z_2	Z_3	$\dots$	Z_p

Arrows indicate the mapping from the 2D grid to the 1D sequence. The text "物理空間" (Physical Space) is written above the arrows, and "空間" (Space) is written below the arrows.

圖2. 物理空間の直接字像

Component	Machines	
	PAX-32	PAX-128
Host Computer	TT1990/20	PAX-64J
Control Unit		PDP-11/24
Number of Processing Unit	32	- *
Processing Unit	Cosmo Terminal D	64
Microprocessor	MC6800	MC68B00
Clock Frequency	1 MHz	2 MHz
Attached Processor	AM9511A	AM9511A-4
Memory Capacity/PU	18K Byte	32K Byte
Memory Access Time	450 ns	150 ns
Peak Performance	0.5 MFLOPS	4 MFLOPS
Synchronize all PUs	138 μ s	47.5 μ s
Broadcast 1024 Byte Data	40 ms	40 ms
$A = A * B + C$	275 μ s	137.5 μ s
$A(I) = B(I) * C(I) + D(I)$	482 μ s	241 μ s
Solve linear equation with 256 variables	2.6 s**	17.1 s
by Gauss-Jordan method		
Cascade sum of data stored in all PUs	2.6 ms	1.2 ms
		0.672 ms

**** Currently 32 PUs are installed.**

**** for 64 variables.**

Table 2. Performance of PAX computer in application problems, obtained through the actual measurements of execution times

Application Problems (algorithm)	Features ^{*1} Data Transfer Mode / Operational Mode / Work Load / Mapping	Measurement				Prediction	
		Problem Size ^{*2} , $N_{\min} \sim N_{\max}$	Execution Time per iteration (s/iter) ^{*3} , $T_{\min} \sim T_{\max}$	Efficiency(X) $\alpha_{\min} \sim \alpha_{\max}$	PAX with P_0 PUs: PAX- P_0	Asymptotic Efficiency ($N/P = \text{const.}$ $P \rightarrow \infty$)	$P(\alpha = 50\%)^{*6}$ with fixing $N/P =$ $N_{\max}/P_0$
Poisson eq. (SOR)	Prx / Syn / Unf / 2Dr	16x16 ~ 48x48	0.0016 ~ 0.011	66 ~ 89	PAX-128	$O(1)$	∞^{*7}
		16x16 ~ 128x128	0.0031 ~ 0.105	48 ~ 95	PAX-64J ^{*4}	$O(1)$	∞^{*7}
Poisson eq. (ADI)	Prx,Rmt / Syn / Unf / 2Dr	32x32 ~ 48x48	0.134 ~ 0.247	65 ~ 72	PAX-128	$O(P^{-1/2})$	1600
		32x32 ~ 160x160	0.077 ~ 1.23	73 ~ 92	PAX-64J ^{*4}	$O(P^{-1/2})$	2000
Navier-Stokes(B-W)	Prx,Rmt / Syn / Unf / 2Dr	32x32 ~ 64x64	0.576 ~ 2.024	84 ~ 91	PAX-64J ^{*4}	$O(P^{-1/2})$	11000
(MacCormack)	Prx / Syn / Unf / 2Dr	32x32 ~ 64x64	0.241 ~ 0.903	97 ~ 99	PAX-64J ^{*4}	$O(P^{-1/2})$	$>10^6$
(SOLA)	Prx / Syn / Unf / 2Dr	16x32 ~ 32x64	1.93 ~ 7.31	74 ~ 87	PAX-128	$O(P^{-1/2})$	$>10^6$
Linear eq. (C-J)	Glb / Syn,Brd / Unf / 1Dr	128 ~ 640	3.9 ~ 202(s)	46 ~ 82	PAX-128	$O(1)$	∞^{*7}
Linear eq. (C-G)	Glb / Syn,Brd / Unf / 1Dr	128 ~ 640	0.102 ~ 0.553	34 ~ 73	PAX-128	$O(1)$	∞^{*7}
Monte Carlo spin	Prx,Rmt / Syn / Unf / 2Dr	32x32 ~ 80x80	0.927 ~ 8.51	70 ~ 95	PAX-128	$O(P^{-1/2})$	$>10^6$
U(1) lattice gauge	Prx / Syn / Unf / 2Dr	(2x2~8x8)x8x16	0.578 ~ 8.72	97 ~ 97	PAX-128	$O(1)$	∞^{*7}
Molecular dynam.							
(Lagrange scheme)	Glb / Syn / Unf / 1Dr, Lag	192 ~ 288	6.77 ~ 15.5	96 ~ 97	PAX-32	$O(1)$	∞^{*7}
(Euler scheme)	Glb / Syn / Nuf / 2Dr, Eul	192 ~ 288	8.23 ~ 17.4	75 ~ 76	PAX-32	$O(1/f)^{*5}$	∞^{*7}
Plasma particle							
field (FFT)	Rmt / Syn / Unf / 2Dr	512	0.129(s)	59	PAX-128	$O(P^{-1/2})$	600
particle push	Prx,Rmt/ Syn/ Nuf/ 2Dr,Eul	2048	0.083(s)	65	PAX-128	$O(1/f)^{*5}$	∞^{*7}
Unsteady flow	Prx / Syn,Ppl / Nuf / 1Dr	384	1.13	56	PAX-128	$O(1)$	∞^{*7}
Logic circuit	Rmt,Irg/ Asy,Dtf/ Nuf/ 2Dr	416 (gates)	0.0098(s)	5	PAX-128	$O(P^{-1/2})$	4

*1 Abbreviations in this column are as follows: Prx: Proximity, Rmt: Remote, Glb: Global, Irg: Irregular, Syn: Synchronous, Asy: Asynchronous, Ppl: Pipeline, Dtf: Data flow, Brd: Broadcasting, Unf: Uniform, Nuf: Nonuniform, 2Dr: 2-dimensional direct, 1Dr: 1-dimensional direct, Lag: Lagrangian scheme, Eul: Eulerian scheme.

*2 If not specified, in terms of lattice points or particles. *3 If not specified, in unit of (s/iteration, or s/time step).

*4 Currently 32 PUs are installed. *5 Factor f = Maximum to average particle densities. *6 $P(\alpha = 50\%)$ = Approximate number of PUs which displays 50% efficiency. *7 P will be limited by the propagation delay of synchronization signal, but in some value greater than 10^6 .