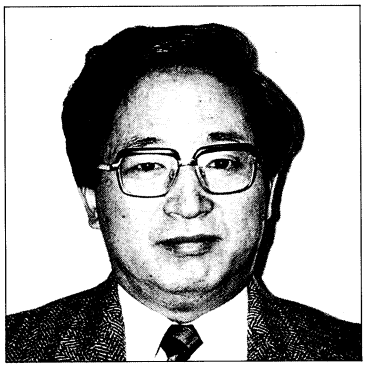
3-7 並列コンピュータ



田中　英彦

ABSTRACT

並列処理の研究は長い歴史を持つが，伝統的アプローチ，言語からのアプローチ，専用マシンからのアプローチ，計算モデルからのアプローチに分けられる．本稿は，これらの流れを概観すると共に技術をアルゴリズムから実装に至るまで分類してそれらの現状をまとめている．ポイントは，アルゴリズム，プログラミング，並列性の抽出，割付とスケジューリング，同期機構，高速アクセス機構，高密度実装である．最後に，並列コンピュータの将来について，発展の見通しを述べると共にそこに至るまでの重要な技術要素をまとめている．

キーワード：並列コンピュータ，並列処理，並列プログラミング，計算機アーキテクチャ

1. はじめに

並列処理研究の歴史は古い．1920年リチャードソンが，天気予報の計算をするのに64,000人の人を円形劇場に集めて，中央の指揮者の指令下でそろって計算をすすめるという構想を発表している．電子技術を用いたコンピュータの出現は大分後になるが，その当時の並列処理は，一つの数値の演算をけたごと逐次進めてゆくことから，複数けたそろって演算を進めるのが並列処理であった．現在は素子技術も進み，何万というプロセッサを集めたシステムが考えられている時代である．

ここでは，並列処理研究の流れ，並列処理技術を概観し，今後の展望を行う．

田中英彦：正員　東京大学工学部電気工学科
Parallel Computer. By Hidehiko TANAKA, Member (Faculty of Engineering, The University of Tokyo, Tokyo, 113 Japan).
電子情報通信学会誌　Vol.75 No.11 pp.1230-1234 1992年11月

2. 並列処理の要素技術

並列コンピュータを実現するための技術を，処理の表現レベルによって分類すれば図1のようになる．表現レベルの内，並行タスク表現とは処理を分析して論理的に並列実行ができる部分を抽出したプログラムであり，静的実行コードとは実行前に各プロセッサに割付けられるコードで，動的実行コードは，実行中に各所の状況に応じて変化する実際の動作を表したコードである．それより下のレベルは，ハードウェア上の動作を表現したものである．

3. 並列処理研究の流れ

並列処理技術にはいくつかの流れがあり，同じ技術でもその流れによって見方が異なる．

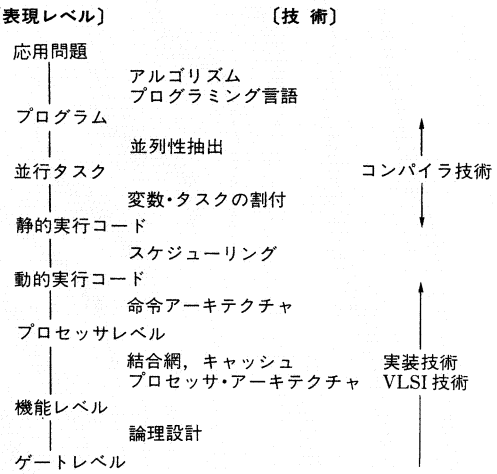


図1　処理の表現レベルと対応技術

3.1　伝統的アプローチ

Fortranを代表とする言語で書かれた科学技術計算を並列に処理することによって高速化を計ろうとするもので，並列性の明示的扱いを特徴とする．スーパコンピュータの演算パイプラインを複数並べて並列性をも抽出しようというものや，実行コードを分析して命令レベルにおける2〜4くらいの小規模な並列効果を得ようとするスーパスケーラ技術[用語]もこの流れ内にある．

用　語　解　説

スーパスケーラ技術　命令レベルの並列処理を行うことで，処理性能を向上させるプロセッサ技術．

並列推論マシン　推論を演算の基本要素とする並列コンピュータ．

オブジェクト指向言語　記述したいものすべてをオブジェクトとしてとらえ，それらの間のインタラクションでプログラムを構成するための言語．

タスク　処理の単位．

データフローマシン　演算に必要なデータの生成状況に応じて演算を制御する並列コンピュータ．

キャッシュ　よく使うデータのコピーを置いておき，実効的にメモリアクセスを高速化するための記憶装置．

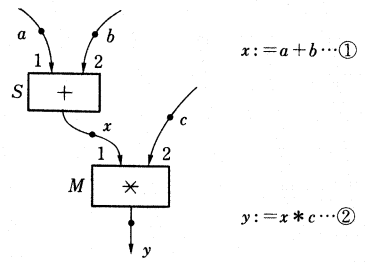


図2　データ駆動の原理

3.2　言語からの汎用マシンアプローチ

あらゆる演算は，それに必要なデータがそろったらすぐに始めることができるというデータ駆動の原理は，並列処理の基本である．例えば，図2おいて，データa，bの双方がそろえば十の演算を始めることができる．従って文①と②の実行順は，文の出現順序ではなく，各演算データの存在のみによって定まり，それをチェックする機構があれば自然な形で並列性を抽出できる．

そのための言語は基本的に関数型言語で，それに近い論理型言語を用いたマシン，例えば第5世代コンピュータのプロトタイプ，並列推論マシンPIM[(1)用語]もこの流れにあり，更に，オブジェクト指向言語を用いて並列処理を行おうとするアプローチもある．

3.3　専用マシンからのアプローチ[(2)]

解きたい問題が明確である場合，その問題を分析し，それに向いた並列コンピュータを構成するもので，過去作られて実用に供されているのはほとんどがこれである．画像処理，信号処理，通信交換処理，CAD，データベース等に用いられており，規則的な演算でわかりやすい並列性を対象としたものが多く，実用的にも成功しているものが多い．

3.4　計算モデルからのアプローチ

動物の脳に代表される神経系の素子であるニューロンの動作モデルとしては，図3のようなものが考えられている．これを組み合わせることによって構成されるニューロネットは，学習能力に優れている．この発展形としてコンピュータを考えることが研究されているが，こ

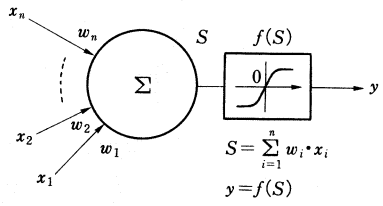


図3　ニューロンのモデル

れも大変自然な並列システムである．

4.　並列処理技術[3]

4.1　アルゴリズム

線形計算に対しては，多くのアルゴリズムが提案されており，並列計算機上に実装され評価も行われている．また，規則的な並列のパイプライン演算をモデル化したものにシストリック（Systolic）アルゴリズムがある．これは，行列計算，信号処理，画像処理等に適しており，それ用のLSIも作られている．

化学反応をモデル化した形，すなわち多くの粒子が近接粒子と勝手に“反応する”形に応用問題を表現し，高並列に問題を解くアルゴリズムも最近提案され今後の発展が期待される．

4.2　プログラミング

応用問題をプログラムに表現する場合，プログラムから並列処理部分を容易に抽出でき，またそれを効率良く処理できる必要がある．

演算の並列性をプログラマが明確に表現するための言語の例は，PARAGRAM[4]や，新しく仕様の定められたFortran 90である．前者では，配列演算や手続きの並列実行を簡潔に記述できる機能を持っている．

一方，データフロー言語や並列処理型言語では並列性の明示は不要で，各文は原則として並列実行される．

しかし，並列性は，細かい命令レベルの並列性だけでは不十分で，実際にはより大きな単位で区分けする必要も存在する．このような区分けのヒントとして期待されるのが，オブジェクト指向言語[用語]を用いて記述されるオブジェクトの単位である．

記述性における他の観点は，並行タスク間の同期問題である．従来の手続き型言語では，同期をプログラマが明示的に記述する必要があり，これがプログラムにバグを生じさせる大きな要因であるが，データフローに基づく言語では，その必要性がなく，エラーが発生し難いといわれている．第5世代プロジェクトで開発された並列オペレーティングシステムPIMOS[1]がその一例である．

一般に，並列プログラムのデバグは難しいといわれ，プログラムの動作をさまざまな観点から能率良く眺めるためのツール[5]が有効であるが，プログラムを仕様から自動合成する手法も将来は期待できよう．

4.3　並列性の抽出

Fortran系の言語に対してはDOループの並列化技法がある．一つは，DOループの各繰返しにおける処理間に依存関係がない場合，それを並列化するもので，他は，制約がある場合，各繰返し間の同期を取ることによって，ある程度の並列処理を可能とするものである．

スーパスケーラ技術のように，命令レベルの並列処理を行う場合，各命令コード位置を動かしたりコピー配置することによって実効並列度を上げることも行われる．

データフロー言語に対する並列性抽出は容易でその並列性は高いが，その実現オーバヘッドを減らすために最近は，実行コードを分析し，通常の逐次処理ですむところはそうして，処理のオーバヘッドを減らす工夫が取られる．

並列論理型言語に対しては，すべてが並列実行の対象となるため，実行可能な処理単位とデータ待ちの処理単位とを分けて，実行を制御する．

4.4　割付とスケジューリング

コンパイラで抽出した並行処理単位（タスク[用語]とよぶ）を，どのプロセッサ上でどのような順序で処理するかがこの問題であり，性能に大きな影響を持つ．

各タスクの実行時間と先行順序関係があらかじめわかっている場合はそれを解析して優れた

スケジュールを効率よく生成できる近似解法が存在する．また，タスク間の情報授受関係が既知の場合には，通信コスト等を最小化することで良いスケジュールを求めることができる．

しかし，論理型言語のようにタスクが動的に生成されその性質が前もってわかり難い場合は，動的負荷の監視や，前もっての走行経験を解析する方法が検討されている．

また，タスクの割付けのヒントをプログラムが指定する手法（pragmaとよぶ）もあり，有効な場合もあるが一般にはなかなか困難である．

4.5 同期機構

並列処理においては，並行して動いている多くの活動の間の同期が非常に重要な操作である．並行に動かした全タスクのすべてが完了するのを待つ（バリア同期）等の同期が必要で，それを支援するためにハードウェア同期機構を設けることも考えられている．

一方，データフローマシン[用語]においては，図2の①のような演算に着目したとき，先に到着したデータ（a）を記憶しておき，その後，対応するデータ（b）が到着すると，識別子Sをキーにして記憶を連想探索して読み出し，演算を実行する．また，各データをメモリに蓄えておく場合，図5のようにタグを設け，そのデータが書き込まれたときに立てる．読出操作はタグを調べ，立っていれば読み出すがそうでなければ待ち，読み書き間の同期を取る．

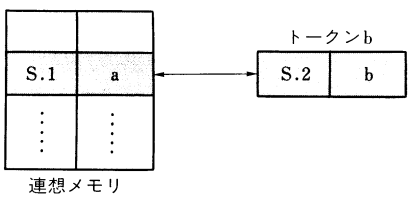


図4　データフローマシンにおける演算の制御

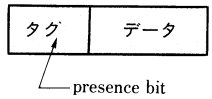


図5　Iストラクチャの構成要素

4.6 高速アクセス機構

並列プログラミングからみれば，主記憶はすべてのプロセッサに共有でアクセスは速いことが望ましい．そのためにキャッシュ[用語]が用いられ，それに伴って複数キャッシュ間の無矛盾性維持制御機構が必要となる（図6）．

より大規模構成では，図6のようなグループをクラスタとし，それを複数集めてハイパーキューブ，メッシュ等の結合網で結ぶことが行われる．物理的に単一の主記憶は難しくなり，分散構成が取られることが多い．

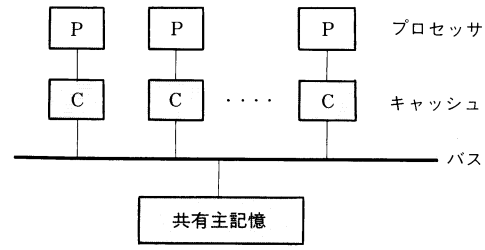


図6　スヌープキャッシュ

4.7 高密度実装

コンピュータの速度は，ゲートの遅延と信号の物理的伝搬時間の和によって定まる．

デバイスの進歩はこれらの要求を満たし，ここ10年の内に，CMOSで10^8トランジスタ，250 MHzの論理素子，メモリで1チップ4 Gbitくらいのものが手に入ると予想されている．しかし，より高並列で高性能なシステムを実現する要求は限りなく，ウェーハスケールインテグレーションや，それらの間の3次元実装によってパス長を減らす工夫等が重要になろう．チップの電力削減も放熱のために重要で，現在チップ駆動電圧も大幅に下げる努力が続けられている．

5. 並列コンピュータの例

表1に，ここ数年内に作られた並列コンピュータの例を示す．方式の欄中，SIMDは単一の命令を多くのデータに対し並列に施す形の方式，MIMDはいわゆる複数のプロセッサがそれぞれに動く形の方式を示す．SIMD型は制御

表1　並列コンピュータの例

システム名	組　織	方　式	プロセッサ Max（台）	メモリ	結　合　網
SIGMA-1	ETL	データフロー	128	256 kW/SE	クロスバ＋オメガ
CM-2	Thinking Machines	SIMD	1ビット×64 k	256 kbit/PE	ハイパーキューブ
GF 11	IBM	SIMD	576	256 kB/PE	メンフィススイッチ
NCUBE 2	NCUBE	MIMD	8192	4 MB/PE	ハイパーキューブ
TC 2000	BBN	MIMD	504	16 MB/PE	バタフライスイッチ
EM 4	ETL	データフロー	80	4.75 MB/PE	サーキュラーオメガ
Adenart	松下	MIMD	256	2 MB/PE	ツイステドクロスネット
SUPRENUM	GMD	MIMD	256	8 MB/PE	トーラスメッシュ＋バス
QCDPAX	筑波大	MIMD	480	4 MB/PE	2次元格子
CAP	三　菱	SIMD	4096	2 kB/PE	2次元格子
R 256	NTT	MIMD	256	1.25 MB/PE	変形キューブ
AP-1000	富士通	MIMD	1024	16 MB/PE	2次元トーラス＋リング・階層バス
MasPar	MasPar	SIMD	4ビット×16 k	16 kB/PE	格子＋クロスバ
J-machine	MIT	MIMD	64 k	18 kB/PE	3次元格子
PIM/P	富士通/ICOT	MIMD	512	256 MB/8 PE	バス＋ハイパーキューブ
CM-5	Thinking Machines	MIMD	16 k	32 MB/PE	変形トリー
Paragon	intel	MIMD	4096	128 MB/PE	2次元メッシュ

が容易であるという特徴がある．また，プロセッサ台数は，実験機に対しては実装台数，商用機に対しては仕様上の最大台数を示している．メモリ欄のPEはプロセッサ要素，SEはデータフローマシン用の構造記憶要素を示す．

6．今後の展望

並列処理は，まず応用と効果が明白な領域から実用に入っているが，今後は並列コンピュータが標準となってゆくのではなかろうか．最初はユーザインタフェースの機能向上，入出力部分の機能向上等を目的として複数プロセッサを用いるところから始まり，順次ユーザプログラムの並列処理に至る．専用が増えるのは当然として，今後は汎用コンピュータが並列化されてゆくのは時代の趨勢であろう．

もちろんこれには，コンパイラ，デバッガ等の発達，並列を基本とするOSの普及が必要であるが，ここに至ってこれらの見通しが見えてきたように思われる．後は，より多くの分野への利用経験の蓄積を通して，技術の深化，システム統合，実用化を定めることが大切である．分散処理と並列処理の自然な融合が進み，利用者に取ってわかりやすい処理のビューが提供されることになるのであろう．

21世紀を眺めれば，今後の高度技術を支える汎用のツールは高性能なコンピュータである．設計したものや自然をそのままにシミュレート可能な超並列コンピュータは，正に今求められているシステムで，これが実現されれば設計法を大きく変えるであろう．従来のコンピュータとは全く異なった特性を持つニューロネットワークとの協力もいわゆる広義のコンピュータの能力を格段と広げるに相違ない．

今，正に時代は，並列なのである．

文　献

（1）Report on ICOT Research Results：Parallel Inference Machine PIM, Operating System PIMOS, Proc. Intl. Conf. on Fifth Generation Computer Systems 1992 (June 1992).

（2）特集号：“並列処理マシン”，情報処理，28，1（1987-01）．

（3）特集号：“並列処理技術”，情報処理，27，9（1986-09）．

（4）特集：マルチプロセッサスーパーコンピュータPHIの研究開発”，情報処理，33，5（1992-05）．

（5）舘村，小池，田中：“並列論理型言語Flengのマルチウィンドデバッガ Hyper DEBU”，情処学論，33，3，pp. 349-359（1992-03）．

田中　英彦（たなか　ひでひこ）（正員）

昭40東大・工・電子卒．昭45同大学院博士課程了．同年東大工学部講師，昭46助教授，昭62教授．専門は計算機アーキテクチャで，並列推論エンジン，分散処理，AI等の研究を行っている．工博．著書「非ノイマン型コンピュータ」，「計算機アーキテクチャ」など．