勉強会資料
# 並列処理の基礎

吉良 裕司
平成16年5月10日

## 1.並列処理

### 1.1 並列処理とは
並列処理とは、処理すべき問題を複数の小部分に分割し、それらを複数のプロセッサで手分けして同時に処理することで、全体の処理時間を短縮しようとするものである。このことを可能とする機構を持った計算機を並列計算機と呼ぶ。

### 1.2 目的
上記の並列計算機が求められる背景として、以下の点が挙げられる。
①超高速計算機の絶対性能の向上
②価格性能比の向上
③スケーラビリティ
①については、LSI(大規模集積回路)チップの内部クロック周波数ほどの速さを、チップの外、つまり、プリント基板上で維持することはできないという、単一プロセッサの実行速度の限界がある。
②については、単一プロセッサで超高速動作を実現するにはコストが非常にかかるが、安価で性能の低いプロセッサを並列に組むことができれば、同じ能力を低コストで実現可能になるといえる。
③については、並列計算機では、プロセッサの数を必要に応じて変えることで、求める性能を得ることができるということである。

### 1.3 ハードウェアとソフトウェア
並列処理技術をハードウェアとソフトウェアとに分けると、並列ハードウェアは、プロセッサをどのように結合するかであり、並列ソフトウェアは、プログラムをどのように手分けして実行させるかということになる。

### 1.4 空間並列・時間並列
並列アーキテクチャは、空間方向の並列化(空間分割型並列処理)と時間方向の並列化(時分割型並列処理)に大別することができる。
前者は、多数のプロセッサを空間的に配列したもので、後者は、演算装置などをいくつかの基本ステップに分解して 、各基本演算操作(FetchやDecode等)を時分割で多数のデータに対して施すようにしたものである。

### 1.5 粒度とトレードオフ
プロセッサ数を増やす、つまり、並列度を増すと確かに性能は大きくなるが、通信負荷も増大して効率は悪くなるというトレードオフが起こる。
また、プロセッサの数と、そのサイズとの間にもトレードオフが存在する。そこで、各プロセッサに割り当てられる計算仕事負荷を表す、粒度(granularity,grain size)によって、プロセッサの数とサイズを決定する。
粒度は、粗粒度、細粒度、中粒度に大別される。
粗粒度は、プログラムの最も外側の繰り返しループの展開など、プログラム制御の最も高いレベルの計算プロセスを含んでおり、少数の大型・複雑・高性能プロセッサ構成を意味する。対して、細粒度は仕事の単位が式の計算、さらには単一の算術論理演算のような低レベルであり、多数の小型・簡単・低性能プロセッサ構成を意味する。VLIWといった命令レベル並列処理がこれに当たる。中粒度はその中間の粒度である。マルチプロセッサ型並列計算機では、プロセッサ間では中から粗粒度あたりの並列処理を行い、プロセッサ内では細粒度並列処理を行っている。

## 2.機械命令の起動形態による分類

計算機アーキテクチャは、機械命令の起動形態で分類すると、コントロール駆動、データ駆動、要求駆動に大分類できる。コントロール駆動は、SISD方式、SIMD方式、VLIW方式、演算パイプライン方式、マルチプロセッサ方式に分類できる。また、データ駆動方式、要求駆動方式が非ノイマン型並列計算機と呼ばれるのに対して、SISD以外のコントロール駆動方式はノイマン型並列計算機と呼ばれる。また、マルチプロセッサ方式、データ駆動方式、要求駆動方式は、MIMD方式に属する。
以下、各方式の説明を述べる。

### 2.1 コントロール駆動方式(control driven)

操作部とオペランド部からなる機械命令プログラムの実行が、ひとつの制御装置によって制御される。制御装置中のプログラムカウンタ(PC)の指す機械命令を読み出し、解読(デコード)後、オペランド部と操作部によって指定された操作を行う。次に実行する命令は、PCに1を加算したアドレスもしくは分岐命令先のアドレスから読み出す。このように、コントロール駆動方式では、機械命令の実行が逐次的になされる。

コントロール駆動型計算機では制御の流れは単一であり、演算装置の構成によって、以下のように分類される。

### 2.1.1 SISD方式(Single Instruction Single Data Stream)

演算装置が一つ設置されており、一つの機械命令の実行によって一つの演算結果が得られる。

SISD方式の代表例として汎用計算機を上げる。汎用大型計算機は特にこれといった特徴はないが、全てのことを無難にしかもかなり高速にこなすといったタイプの計算機である。過去のソフトウェアの継続利用という大きな足かせがあり、機械命令レベルのアーキテクチャは1964年のIBM360の発表以来ほぼ固定化されている。従って、レジスタ・トランスファ・レベル以下のレベルで高速化が図られている。汎用大型計算機の論理方式上の工夫による高速化は命令パイプライン方式の改善にかかっている。

命令パイプライン方式は、CPU内での命令実行を複数の工程に分解し、これをパイプライン処理するものである。1つの命令の実行が、例えば4段階からなるとすると(Fetch,Decode,Execute,Write back)、この4つの工程を4ステージのパイプラインとして構成し、命令を次々実行できるようにする。

### 2.1.2 SIMD方式(Single Instruction Multiple Data Stream)

SISDに対して、演算装置が多数設置されており、それらが一つの制御部から発せられて演算命令を同時に実行する。演算装置が非常に多いとき、真価を発揮する。

演算装置は互いに結合網で結合されている。結合網には、すべてのプロセッサ対の間を接続する完全結合や、その他にも、リング、メッシュ、トーラス、ツリー、ハイパーキューブなどがある。

SIMDでは、定型的な並列処理の高速化を達成できるので、科学技術計算処理、画像・信号処理など、ベクトルや配列といった、集合に対する演算を必要とする分野に適している。

### 2.1.3 VLIW方式(Very Long Instruction Word)

命令の1語長を長くして(128ビット以上)、1語の中に複数の命令を詰め込む。そして、各命令で演算機やメモリを独立に制御する。

VLIWでは、同時に実行する命令の組み合わせは、コンパイル時に決め、その組み合わせでVLIW命令コードを生成する。この組み合わせは実行時に変更されることはない(静的スケジューリング)。

実行時にスケジューリングを行わないので、制御回路は簡単になり、高速化が達成できる。その代わりに、プログラムの解析を行い、並列実行可能な命令を組み合わせてコードを生成する特殊なコンパイラが必要となる。

この方式に対照的な方式として、スーパースカラがある。これは、同時に実行する命令の組み合わせが実行時に動的に決まる(動的スケジューリング)。

### 2.1.4 演算パイプライン方式(ベクトルプロセッサ)

先に述べた命令パイプライン方式とほとんど同じだが、演算パイプライン方式は、浮動小数点計算に対してパイプライン処理を行うものである。一般に、ベクトルプロセッサと呼ばれる超高速数値計算を目的とするスーパーコンピュータに使われている。近年では、多数のベクトルプロセッサを同時に動かす「並列ベクトル型」が主流になっている。

最大性能が得られるのは、各ステージでの処理時間が等しいときである。大量データが入力されて結果が出るまでに時間がかかるが、結果が出始めると連続的に結果が得られる。

### 2.1.5 マルチプロセッサ

複数のプロセッサが相互接続され、各プロセッサが個々に別々のプログラムを実行できるもの。特徴としては、同一処理を多数のプロセッサで分担して高性能化を図る(負荷分散)、異なる機能をそれぞれのプロセッサで分担して高性能化を図る(機能分散)、デュアルシステムやフェイルセーフシステムなどによる信頼性の向上を図る、必要に応じた性能をプロセッサ台数で調整できる(スケーラビリティ)などがある。

マルチプロセッサの構成要素は、プロセッサ(複数)、メモリ(単一または複数)、それらを接続するネットワークである。メモリアクセスを基本に構成を分類すると、次の3つに分けることができる。

①UMA(Uniform Memory Access)

共有メモリを持つもので、対照型マルチプロセッサとも呼ぶ。基本的に、プロセッサとメモリの間にネットワークが入っていて、全プロセッサが全メモリに対して同等にアクセス可能。

②NUMA(Non Uniform Memory Access)

分散共有メモリ型で、非対称マルチプロセッサとも呼ぶ。基本的にメモリ同士がネットワーク接続されている。プロセッサは全メモリにアクセス可能だが、メモリの位置によってアクセス時間が違う。そのために、データの配置によって実行効率が変わる。

③NORA（NO Remote Access）
　他のメモリへのアクセスができないもので、分散メモリ型である。基本的にプロセッサ同士をネットワーク接続した構成となる。

　共有メモリは、全プロセッサからアクセスできるメモリで、メモリと各プロセッサがネットワークを通して接続されている。共有メモリ型システムの多くは、共有メモリの接続にバスを使っている。共有メモリに対して、各プロセッサが個別に持ち、自分だけがアクセス可能なメモリをローカルメモリと呼ぶ。
　分散メモリ型マルチプロセッサは共有メモリを持たず、それぞれ個別にメモリを持ったプロセッサ同士をネットワークで接続した構成をとっている。

## 2.2 データ駆動方式

　データ駆動方式には、PCがなく、各機械命令はそのオペランド・データが揃うと実行される。したがって、実行される命令の系列は、データの依存関係に基づいており、原理的には、プログラムに内在するすべての並列性を実行時に同時に引き出すことができる。また、副作用(*)がなく、関数型言語に適している。ベクトル化が容易でないような、科学技術計算分野や人工知能用言語PROLOGの処理系などの、非定型的処理の高速処理が期待されている。しかし、メモリ構成などを中心に解決すべき問題点が山積みされている。
　上記の副作用(*)とは、ノイマン型並列計算機で起こる、プログラムの変数の値が、特にサブルーチンの実行によって、書き換えられることにより変数値の変更が間接的に行われることである。これによって、変数と値の関係はますます把握しにくくなる。

## 2.3 要求駆動方式

　機械命令はその結果が要求されるときに起動される。要求された命令からの結果は、要求した命令に返される。結果を返す過程はデータ駆動的になされる。つまり、要求駆動方式は、要求が次々に伝えられていく過程と、結果が返されていく過程に分けることができ、後結果が返されていく過程は、データ駆動方式と全く同様な機構で実現できる。
　純粋なデータ駆動方式と比較して、要求駆動方式は、次のような特徴を持つ。
①必要に応じて式の評価を開始することができ、不必要な式の評価を行う必要がない。
②式の評価を最外側から行うことができ、遅延評価を容易に行うことができる。
　要求駆動方式では、実行過程が、処理してみないと分からない動的なもので、あらかじめデータ・フロー・グラフにコンパイルできない場合、要求を伝える過程で解釈実行しながらデータ・フロー・グラフを生成する。したがって、データ駆動方式がコンパイラ向きなのに対して、要求駆動方式はインタプリタ向きなものとなっている。要求駆動方式は関数型言語の項書き換えモデルや論理型言語PROLOGの実行制御過程との親和性がよいとされている。

## 参考文献


富田　眞治 「並列計算機構成論」 昭晃堂
小畑　正貴 「たのしくできる並列処理コンピュータ」 東京電機大学出版局
奥川　峻史 「並列計算機アーキテクチャ」 コロナ社
富田研新入生セミナー 「並列処理の基礎」（平成14年度）