新入生勉強会 並列計算機

岡村 大

2004/05/17

1 並列計算機の概要

並列計算機とは複数の CPU を同時に動作させることで、単一ジョブの高速化をはかった計算機のことである。

1.1 並列処理の基礎

高性能の並列計算機を構成するためには、並列アルゴリズム、並列プログラミング言語、並列コンパイラ、並列オペレーティングシステム、並列ハードウェアを総合的に研究することが必要となる。

一般にある問題を並列計算機で実行するまでの処理過程は次の 4 段階に大別できる。

1. ある問題に対する最適の並列アルゴリズムの選択
2. 並列プログラミング言語によるそのアルゴリズムの記述
3. コンパイル
4. 計算機による実行

プログラム効率は、アルゴリズムそのものの他に、アルゴリズムのプログラム化におけるプログラミング技術、コンパイラの性能に大きく依存する。アルゴリズムの並列度と計算機の並列度が合致すれば、もっとも効率の良いプログラムができる。

1.2 並列処理の基本原理

並列計算機アーキテクチャは原理的に、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 (時間分割型並列処理) の 2 つに分けることができる。

空間分割型並列処理には、多数のプロセッサを空間的に配置したプロセッサアレイ方式やマルチプロセッサ方式が用いられる。N 個のプロセッサを並列に動作させるとほぼ N 倍の性能が得られる。時間分割並列処理には、演算操作をいくつかの基本ステップに分解して、各基本演算操作を時分割で多数のデータに対して行うようなパイプライン方式が用いられる。処理時間が Nt かかる演算操作を M 段に分解することで、単一データの処理時間は変わらないが、一連のデータの処理について見れば Nt/M の時間で 1 データを処理していることになり、時間あたりの処理量が増えるので全体の処理が高速化される。

1.3 並列処理の効率

アムダールの法則 単一プロセッサで最良の直列アルゴリズムによって処理した場合の実行時間を T_1 、N 個プロセッサで並列アルゴリズムににによって処理した場合の実行時間を T_N とすると、$S_N = T_1/T_N$ で定義される速度向上比は、直列アルゴリズムの中でどうしても並列化できない処理の割合を α とおくと、以下の式で与えられる。

$$S_N = \frac{N}{1 + (N-1)\alpha}$$

この関係をアムダールの法則という。

ここで、$N \to \infty$ のとき、$S_N \to 1/\alpha$ となり、α の値が並列アルゴリズムの性能に大きな影響を与えることが分かる。一般に α はベクトルに対する性能長さ n の関数であり、効率の良いアルゴリズムは、$n \to \infty$ のとき、$\alpha(n) \to 0$ すなわち $S_N = N$ となって、プロセッサ数 N に比例した線形の速度向上が見込めるようになる。

n のベクトルの処理にかかる時間 t は、実際の計算機に対し一次近似で以下の式で表せる。

$$t = \frac{n + n_{1/2}}{r_\infty}$$

r_∞ は 1 秒間に実行される等価なスカラー演算の単位ではかった最大計算速度で、$n \to \infty$ のときこの性能が得られる。この値は使用される計算機技術の特性もあらわし、クロック周期に反比例する。

$n_{1/2}$ は最大性能の半分を達成するのに必要なベクトル長で半性能長と呼ばれる。この値は計算機アーキテクチャの並列度を示す。

1.4 プロセス間通信と同期

並列処理では、複数のプロセスが強調しながら仕事を実行する。そのためプロセス間でのデータ通信を行い、実行のタイミング同期をとる必要がある。同期の方式には、共有メモリに於かれた共有変数による方式と、メッセージ交換による方式がある。

共有変数による同期

共有メモリを介したプロセス間通信は、送信プロセスがデータを共有変数に書き込み、受信プロセスがその共有変数を読み出すことによって同期を取る。その同期の取り方には、2つのプロセスが同時に共有変数にアクセスすることの無いように制御する相互排除同期と、受信プロセスが有る条件が満たされるまで待つ条件同期がある。

同期によるオーバーヘッドを小さくするために、同期にはハードウェアを設けるのが普通だが、セマフォのようなソフトウェアで実現される同期機構を併用することが多い。セマフォは以下の2つの基本操作によってアクセスされる整数変数 $s(\geq 0)$ である。

- $P(s)$:プロセスを監視し、$s > 0$ になるまでプロセスの実行を遅らせ、$s > 0$ になれば s を1減らす。

- $V(s)$:s を1増やす。

具体的なセマフォの使用例を示す。プロセスAとBを並列に実行しているとき、Aを実行中にBとの同期が必要なところで一旦 $P(s)$ を呼び出し一時プロセスを停止し、Bが同期が取れるところまで実行された時点で $V(s)$ を呼び出し、Aの実行を再開する。これによって同期の取れた動作が可能となる。

メッセージ交換による同期

共有変数の読み書きの代わりにプロセスが直接メッセージを送受信することでデータの同期を取る方式。。

2 並列計算機の諸方式

2.1 並列計算機の分類

計算機は制御駆動型、データ駆動型、要求駆動型の3つに大別される。現在の商用計算機は、すべての系列が制御装置によって解読され実行される制御駆動型であり、フォンノイマン型とも呼ばれる。

2.1.1 Flynn の分類

SISD 単一の命令を順番にメモリから取り出して、単一のデータの流れに対して処理を施す方式。最近のプロセッサは処理のパイプライン化が当たり前におこなわれているが、並列処理が単一の CPU 内の演算切れベルの処理なので、この方式に属するとされる。

SIMD 多数の演算装置に対し単一の命令をブロードキャストし、複数のデータに対して同一の処理を行う方式。単純な処理に対しては、低コストで高い並列性を得ることが出来ることから、画像処理等の専用マシンとしては有利だが、すべての演算装置が単一の命令しか実行できないため、複雑な処理を行う汎用マシンには向かない。

MISD 単一のデータに対して、同時に複数の命令が実行される方式。このカテゴリに入る計算機は存在しない。

MIMD 独立の命令実行能力を持った複数の CPU が、互いにデータを交換しながら複数のデータフローに対して処理を行う方式。それぞれのプロセッサが独立に動作するので、汎用性・柔軟性が高いが、同期操作が複雑となりプログラミングが困難な場合が多い。最近の並列計算機の主流である。

2.2 共有メモリシステムに関する分類

MIMD 型の計算機はバラエティに富んでいるため、共有メモリシステムに着目して以下の3つのタイプに分類する。

UMA(均一アクセスモデル) 全てのプロセッサがアドレス空間を共有し、かつ同一時間でアクセス可能な共有メモリのモデル。プロセッサ数が 10 程度間電お小規模汎用システムで、この方式がとられている。既存のシステムとの互換性が高い。

NUMA(不均一アクセスモデル)　UMA よりもプロセッサ数が増えた場合、均一時間ですべての共有メモリにアクセスしようとすると、アクセス時間が大きくなってしまう。これを改善するために、メモリを分散し、近くのメモリは高速に、遠くのメモリは結合網を介してアクセスし、有る程度遅延を許すようにした方式。データアクセスの局所性を利用すれば、遠隔メモリのアクセスで起こる遅延をかなり避けることが出来る。また、UMA のシステムがそのまま利用できるという利点がある。

NORA(無遠隔アクセスモデル)　共有メモリを物理的に持たず、プロセッサ毎にローカルメモリを持ち、メッセージ転送によって他のプロセッサと交信する方式。

　以上の 3 方式を比較すると、UMA は汎用性に優れるが大規模システムの構築は難しい。逆に NORA は大規模な数値計算アドの専用目的で利用される。NUMA はこの 2 つの中間に位置する。