参考資料:並列 OS

4 回生 山添博史

1996 年 4 月 24 日 (水)

1 はじめに

分散 OS に関する書籍は多いが、並列 OS に関する書籍が見つからないため、主に雑誌記事を参考にして、並列 OS についてまとめた。並列 OS に要求される課題 2 点と、実際の並列 OS の例 2 件を示す。

2 スケジューリング

マルチプロセッサの主な存在意義はプログラムの高速実行であることを考えると、スケジューリング問題が重要となる。

2.1 計算機資源からみた分類

時分割 (Time-multiplexing) 方式と空間分割 (Space-multiplexing) 方式の 2 つに大別できる。時分割方式は、基本的にはシステム全体のプロセッサ群を 1 つのプロセスに割り当て、プロセス内のスレッドを同時実行し、タイムスライスを使いきったら次のプロセスに割り当てるものである。一方、空間分割方式は、プロセッサ群をグループに分割して、グループ対応にプロセスを割り当てるものである。

2.2 割り当て決定時期からみた分類

プログラム実行時にのみプロセッサ割り当てを決定する静的方式と、実行中にも決定する動的方式の 2 つがある。割り当てのオーバーヘッドを考慮しても動的方式が優れているという報告がある。

2.3 並列処理実行環境モデル

ユーザ・プログラムは、ユーザまたは並列化言語処理系によって複数のアクティビティに変化し、それらがプロセッサにマッピングされて並列実行される。このとき、アクティビティはまずカーネルの提供する仮想プロセッサ (virtual processor) にマッピングされ、さらに仮想プロセッサはカーネルのスケジューラによって実際の物理プロセッサ (physical processor) にマッピングされる。

軽い実行環境モデルとして、スレッド・モデルが注目されている。スレッドとはコンテクストのことで、プログラムカウンタ・レジスタの内容・ファイル資源の割り当て情報など、プロセッサが命令を実行しているときの状態を表す情報の総称である。このスレッドをどのレベルで管理するかに関して、次の 2 つが考えられた。

2.3.1 カーネルレベル・スレッド・モデル

カーネルが仮想プロセッサとしてスレッドを提供し (このスレッドをカーネル・スレッドという)、ユーザ・アクティビティはカーネル・スレッドに 1 対 1 にマッピングされて実行されるモデルである。

2.3.2 ユーザレベル・スレッド・モデル

ユーザ空間でスレッドの生成、消滅などのスレッド管理を行うものである。ユーザ空間内の実行時ルーチンでユーザ・スレッドをスケジューリングできる。すなわち、ユーザ・スレッドのスケジューリング方策をそのプログラムに適した方策にユーザが定義できるという観点からは、効率的な並列実行が行える可能性がある。

3 仮想ページ配置問題

3.1 メモリ・アーキテクチャ

現在のマルチプロセッサを、メモリ・アーキテクチャの観点から大きく分けると、次の 3 つになる。

3.1.1 UMA(Uniform Memory Access)

物理メモリは共有メモリでかつ、どのプロセッサからも任意のアドレスにほぼ同じ時間でアクセスできるアーキテクチャである。

3.1.2 NUMA(Non-Uniform Memory Access)

物理メモリは共有メモリでかつ、プロセッサからのメモリアクセス時間が均一でないアーキテクチャである。各プロセッサから等距離でアクセスできる共有メモリ (グローバル・メモリ) があるかどうかにより、ローカル/リモート・メモリ・アーキテクチャと、ローカル/グローバル/リモート・メモリ・アーキテクチャに別れる。

3.1.3 NORMA(NO Remote Memory Access)

共有メモリをもたず、プロセッサからのアクセスはプロセッサ対応のローカル・メモリに対して行われるアーキテクチャである。

3.2 仮想ページ配置の方法

NUMA 型マルチプロセッサでは、プロセッサからのメモリアクセス時間が物理アドレスによって異なるので、仮想ページをどの物理ページにマッピングするかという仮想ページ配置問題がプログラムの実行時間に大きな影響を与える。仮想ページ配置方法は以下の 2 つに分類できる。

静的方法 仮想ページは 1 つの物理ページにマッピングされたあとは、物理メモリが不足して 2 次記憶に追い出される場合を除いて、そこに固定される。

動的方法 仮想ページの物理ページへのマッピングをプログラム実行中に動的に変更する。

静的方法は実現方法が単純であり、管理のオーバヘッドが少ないが、物理メモリアクセスの不均一性を十分に活かしきれていないという問題がある。動的方法は、物理メモリアクセスの不均一性を考慮して、極力ローカル・メモリにマッピングしようとする方法であり、管理のオーバヘッドが生じるという欠点があるが、不均一性を活用できる可能性があり、最近研究されている。以下、動的方法について述べる。

一般に、共有する仮想ページの動的管理として、以下の 2 つがある。

1. ページを複製する方法 :仮想ページをアクセスするすべてのプロセッサのローカル・メモリに仮想ページをコピーして、アクセス時間を短縮しようとする方法である。

2. ページを移動する方法：仮想ページをアクセスするプロセッサの中でいずれか1つのプロセッサのローカル・メモリに仮想ページをマッピングする方法である。他プロセッサからはリモート・アクセスとなるので、アクセス時間がかかる。

これら2つのうちどちらを用いるかは、仮想ページへのアクセス属性とアクセス頻度などに依存する。例えば、読み出ししか行わない仮想ページには方法1が適しているが、書き込みを行う仮想ページに方法1を適用すると、データの一貫性を保つ制御(コヒーレンス制御)が必要となり、このオーバヘッドが生じてしまう。

また、次のような例もある。1つの仮想アドレス空間内のスレッドa,bと、3つのデータ・オブジェクトA,B,Cを考える。データ・オブジェクトA,Bにはそれぞれaのみ、bのみがアクセスし、a,bで共有しないとする。データ・オブジェクトCはa,bで共有し、読み書きするとする。また、A,B,Cの大きさの合計は仮想ページの大きさ以下とする。このとき、処理系がA,B,Cを1つの仮想ページにマッピングすると、本来共有されないA,Bまでもが共有のための制御の対象となって、ローカル・メモリ間を移動することになり、効率が悪くなある。このとき、処理系は、少なくともA,BとCは異なる仮想ページにマッピングする必要がある。コンパイル時に静的に分かるデータアクセス属性はマッピングに少なくとも反映させる必要がある。

4 並列計算機TOP-1のOS

4.1 TOP-1

TOP-1は、10台のプロセッシング・ユニット(PU)をスヌープ・キャッシュを介して共有バスで結合した、いわゆるUMA(Uniform Memory Access)型の並列計算機である。各PUにはIntel i386DXと128KBのキャッシュが装備されており、キャッシュ間の一貫性はハードウェアによって保たれる。HDに直接アクセスできるのは、HD制御(HDC)カードに接続された特定のPUのみである。その他のI/Oや外界へのアクセスは、フロントエンド・プロセッサとして接続されているIBM PS/55を介して行われる。

4.2 TOP-1 OS

OS開発にあたっては、実現が比較的容易であるマスタ・スレーブ方式を用いている。つまり、OSのカーネル・コードは特定のKPU(Kernel PU:PU0)でのみ実行され、プロセスのユーザ・コード部分を並列に実行するUPU(User PU:PU1-PU9)によって呼び出される。このため、カーネル・モードとユーザ・モードが切り替わるときに同時にPUも切り替える必要がある。このモード切り替え用に、KrunQ, UrunQという2つのプロセス・キューを用意してプロセス管理を行っている。これはUnixのrunQ(runnable process queue)を拡張したもので、KrunQにはKPUでの実行を待っているプロセスが、UrunQにはUPUでの実行を待っているプロセスが入れられる。

5 並列言語KL1とOS

KL1は、荘園機能,実行優先順位指定機能,負荷分散指定機能など、並列マシン上でのプログラムの実行制御機能を備えた並列論理型言語である。これによって並列OSであるPIMOSを記述することを考える。

5.1 並列マシンの概要

KL1を起動させる並列マシンはマルチPSIおよびPIM(Parallel Inference Machine)である。マルチPSIは、逐次型推論マシンPSI(Personal Sequential Inference machine)のCPUを小型化し、最大64台まで

ネットワークで結合したマルチプロセッサシステムである。一方 PIM は、8 台程度のプロセッサを共有メモリで結合したクラスタとそのクラスタ間をネットワークで結合した階層構造のマシンである。

マルチ PSI のアーキテクチャ上の性質を示すと次のようになる。

- 各プロセッシングエレメント (PE) はローカルメモリを持つ。
- PE 同士はネットワークで接続される。
- PE 間では共有メモリをもたず、データ構造を共有する場合には別の PE 内部へのポインタが生じる。
- PE 内のアドレス系と PE 間にまたがるアドレス系は異なる。すなわち、PE を渡るポインタは、各 PE の出入口でアドレス系の変換が施される。

これらの性質は、PIM の各クラスタを 1 つの PE とみなした場合にも当てはまる。

5.2 PIMOS の概要

PIMOS はシングルユーザ、マルチタスクの機能を提供する集中単一型の OS であり、複数の PE 上の独立した OS が協調しながら処理を進めるような分散 OS ではない。

実行時には、スケジューラが各 PE 内にあるゴールスタックから実行可能なゴールを優先順位の高いものから取り出して順次実行していく。このとき、ゴールの実行が一部の PE に集中しないように調整することができる。これが KL1 の負荷分散指定機能で、ゴール単位にそのゴールを実行させるべき PE 番号を指定できる。また、この PE 番号の指定を実行時に与えることもできるので、適当な負荷分散の方針を KL1 自身で記述し実際の計算の進み具合いを見ながら適度な分散指定を決めることもある程度可能である。

参考文献

[1] 福田晃 :並列オペレーティング ・システム, 情報処理,Vol.34,No.49,pp.1139- 1149(Sep.1993).

[2] 福田晃 :並列 OS の研究動向, 情報処理,Vol.36,No.8,pp.726–728(Aug.1995).

[3] 河地谷清久仁 :並列計算機 TOP-1 の OS, 情報処理,Vol.36,No.8,pp.734–738 (Aug.1995).

[4] 宮崎敏彦 :並列論理型言語 KL1 の実現方式と並列 OS の記述, 電子情報通信学会論文誌,Vol.J71-D,No.9,pp.1423–1432(Aug.1988).