2H−1

PIE64の並列処理管理カーネルにおけるスケジューリング,負荷分散の検討

日高 康雄 小池 汎平 田中 英彦
{hidaka,koike,tanaka}@mtl.t.u-tokyo.ac.jp
東京大学 工学部

1 はじめに

我々は,大規模知識処理のための高並列推論エンジン－PIE64－[1]の研究,開発を進めている.PIE64は,コミッティッドチョイス型言語であるFleng[7]を主要な記述言語とする並列計算機であり,64台の推論ユニット(IU－Inference Unit)を,2系統の自動負荷分散機能を持つ相互結合網[4]で接続した構成をしている.

我々は,並列処理で新たに必要となる処理は,要素プロセッサ間の通信と負荷分散やスケジューリングのような管理であり,これらが並列処理オーバヘッドの根本的原因であると考え,それらを本来の計算処理と並列に処理することにより,従来のプロセッサ高速化技術の下で並列処理オーバヘッドを効果的に吸収する処理モデルを検討した.そして,PIE64の要素プロセッサである推論ユニットに,協調動作する3種類のプロセッサがそれぞれ「計算」,「通信」,「管理」を担うというアーキテクチャを採用した.[2]ユーザの記述したプログラムは推論処理プロセッサ(UNIRED[6])で,通信処理はネットワークインタフェースプロセッサ(NIP[5])で,並列処理管理は管理プロセッサ(MP－Management Processor)で実行される.(図1)管理プロセッサには,汎用のRISC型高速マイクロプロセッサであるSPARCを使用し,様々な並列処理管理アルゴリズムの実験を可能とする汎用性を重視している.

管理プロセッサは,「並列処理管理カーネル」と呼ばれるプログラムを実行する.並列処理管理カーネルは,いわゆるオペレーティングシステムの核に相当するが,細粒度並列処理を効果的に支援するために,特に並列処理管理,すなわち,細粒度プロセス(スレッドまたはゴールに相当する.以下ではスレッドと呼ぶ)に対するスケジューリングと負荷分散に重点を置いているのが特徴である.(図2)並列処理管理カーネルの全体構成を図3に示す.

本稿では,並列処理管理カーネルにおけるスケジューリング,負荷分散の検討を行なう.

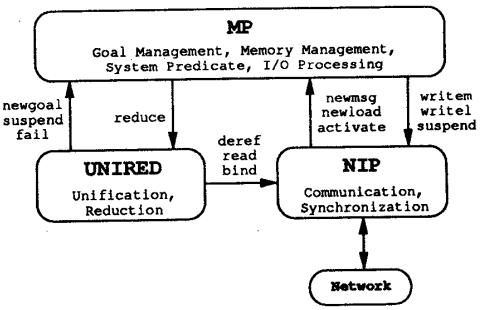

図1: 推論ユニット内の分散協調処理

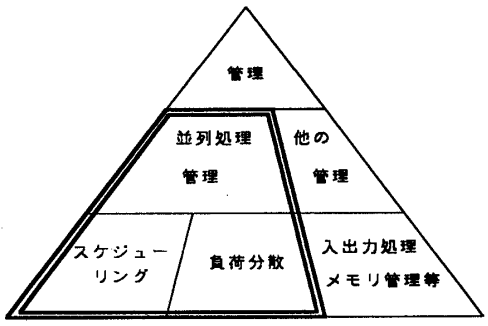

図2: 並列処理管理カーネルの役割

2 理想的なスケジューリング,負荷分散の定性的検討

本当に最適なスケジューリング,負荷分散を求めるのは,全てのスレッドの定量的な挙動がわかっている場合でも,現実的な時間内で求めるのは困難な場合が多い.また,スレッドの生成が動的に行なわれる場合は,実行時にスケジューリング,負荷分散を行なう必要があり,その処理は軽くなければならない.

ここでは,目標とする理想的なスケジューリング,負荷分散に求められる事柄を列挙して定性的に考察し,実際のスケジューリング,負荷分散のアルゴリズムにどのような要素を採り入れるべきかを検討する.

1. 同期オーバヘッドの低減(スケジューリング)
   記号処理においては,データ依存関係を静的に求められない場合があり,全てを静的にスケジューリングするのは困難であるため,データ依存関係を持つスレッドの間では,同期をとって実行順序を保証しなければならない.すなわち,未確定の変数を参照した場合は,コンテクストスイッチを行なってそのスレッドの実行をサスペンドさせる必要がある.しかし,コンテクストスイッチはそのままオーバヘッドとなるため,なるべくサスペンドを起こさないスレッドをスケジューリングし,同期オーバヘッドの低減をはかるのが良い.
2. 並列度の維持,抑制と通信量の低減(スケジューリング,負荷分散)
   並列度が低い時には,多くのスレッドをフォークするようなスレッドを優先的にスケジューリングして並列度の向上をはかるべきである.また,負荷分散も頻繁に行なうべきであるが,やみくもに負荷の分割を行なうとリモートメモリ参照が増えて通信量の増加につながるため,並列度の低下を招かない範囲で通信量を減らすべく,負荷の分割を適切に行なうのが良い.[3]逆に,並列度が充分に高い時には負荷分散を抑制し,リモートメモリ参照や負荷分散のための通信量の低減をはかるべきである.
3. ヒープメモリの枯渇の回避(スケジューリング,負荷分散)
   データの生成をするスレッドと消費をするスレッドがある時に,生成のスピードの方が早いと,ヒープメモリが枯渇して実行終了に至らない場合がある.要求駆動やbounded bufferなどのプログラミング手法も存在するが,プログラマの負担になり,同期オーバヘッドの増加や並列度の低下にもつながるため,できれば,スケジューリングを適切に行なって,ヒープメモリの枯渇を回避するのが望ましい.すなわち,ガベージコレクション直後のヒープメモリの残量に応じて,生成型スレッドのスケジューリングの

On scheduling and load distribution of the parallel processing management kernel of PIE64
Yasuo HIDAKA, Hanpei KOIKE, Hidehiko TANAKA
University of Tokyo, Faculty of Engineering

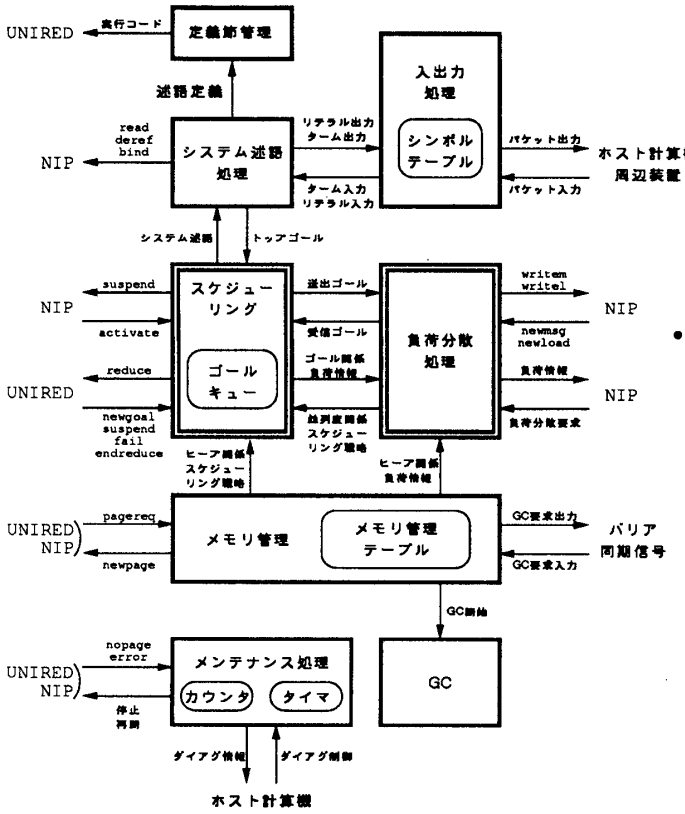

図 3: 並列処理管理カーネルの全体構成

優先度を動的に変化させるのが望ましい. また, 負荷分散においても, 生成型のスレッドに対しては, ヒープメモリ使用量を負荷量とした負荷分散も採り入れるのが良い.

4. 負荷のバランス (負荷分散)
   各推論ユニットのかかえるスレッド数に不均衡があると, アイドル状態の推論ユニットが多くなり, 実行時間が増加する. また, 各推論ユニットのヒープメモリ使用量に不均衡があると, 一括型のガベージコレクションの間隔が短くなり, やはり実行時間が増加する. PIE64 の相互結合網は自動負荷分散機能を持つためこれを有効に活用し, スレッド数とヒープメモリ使用量のバランスをとる必要がある.
5. ユーザ指定の優先度 (スケジューリング)
   優先的処理や投機的計算などを可能とするために, ユーザがスケジューリングの優先度を指定できる枠組も用意すべきである.

3 並列処理管理カーネルにおけるスケジューリング, 負荷分散

上記の考察に基づいた, 並列処理管理カーネルのスケジューリングや負荷分散の戦略の概要を以下に示す.

- スケジューリング
  実行するスレッドの選択は, まず, ユーザ指定の優先度, 次に, 並列度やヒープメモリ使用量で決まる動的な優先度, そして, データ依存関係が静的に検出できる箇所の実行順序の順番で行なう. 動的な優先度の変更の仕方を, 表 1にまとめる. フォーク型のスレッド, 生成型のスレッドの検出, データ依存関係に従った実行順序の決定は, コンパイラで静的に行なう. また, データ依存関係が静的に決まらないが, 実行前に参照する変数がわかっているスレッドは, 実行を開始してからコンテクストスイッチ, サスペンドを行なうのではなく, 実行前にまず参照する変数にサスペンド登録を行ない, 変数の値が確定してから実行を開始することによって, コンテクストスイッチのオーバヘッドの低減をはかる.

表 1: スケジューリング優先度の動的な変更

ヒープメモリ残量	並列度	
	低い	高い
多い (全 IU)	フォーク型を高く	
少ない (他 IU)	フォーク型を高く	
少ない (自 IU)	生成型を低く フォーク型を高く	生成型を低く

- 負荷分散
  どのスレッドを負荷分散させるかは, プロファイラやコンパイラによって, リモートメモリ参照が少なくなるように静的に選択する. そして, 並列度が低い時のみ, 選択されたスレッドを実際に負荷分散させ, 並列度が高い時は負荷分散を行なわずに, そのスレッドを自 IU で実行する. 但し, 並列度が高い時でも自 IU のヒープメモリ残量が少ないときは, 生成型のスレッドを負荷分散させる. 負荷分散でスレッドを渡す先は, 相互結合網の自動負荷分散機能によって決定する. 2 系統の相互結合網にはそれぞれ, スレッド数に関する負荷情報とヒープメモリ使用量に関する負荷情報を流す. ヒープメモリ残量の少ない IU がひとつもない時は, 全ての負荷分散はスレッド数に関する負荷情報に従って行ない, ヒープメモリ残量の少ない IU が存在する時は, 生成型のスレッドの負荷分散をヒープメモリ使用量に関する負荷情報に従って行ない, それ以外の負荷分散をスレッド数に関する負荷情報に従って行なう.

4 おわりに

本稿では, PIE64 の並列処理管理カーネルにおけるスケジューリング, 負荷分散について検討を行なった. 今後は, 並列処理管理カーネルの実装を行ない, アルゴリズムの細部の違いによる影響などを評価して, 並列処理管理アルゴリズムの改良を行なっていく.

なお, 本研究は文部省特別推進研究 No.62065002 の一環として行なわれている.

参考文献

[1] 小池, 田中: “並列推論エンジン PIE64”, 並列コンピュータアーキテクチャ, bit 臨時増刊, Vol.21, No.4, 1989, pp. 488-497.

[2] 日高, 小池, 田中: “並列推論エンジン PIE64 の推論ユニットのアーキテクチャ”, コンピュータシステム研究会 CPSY90-44, 電子情報通信学会, Vol.90, No. 144, July 1990.

[3] 日高, 小池, 舘村, 田中: “実行プロファイルに基づく PIE64 の負荷分散方式”, 並列処理シンポジウム '90 T2-3, 情報処理学会, May 1990.

[4] 高橋, 小池, 田中: “並列推論マシン PIE64 の相互結合網の作成および評価”, 並列処理シンポジウム '90 A1-1, 情報処理学会, May 1990.

[5] 清水, 小池, 田中: “並列推論マシン PIE64 の推論ユニット間通信”, 計算機アーキテクチャ研究会 79-4, 情報処理学会, Nov. 1989.

[6] 島田, 下山, 清水, 小池, 田中: “推論プロセッサ UNIREDII の命令セット”, 計算機アーキテクチャ研究会 79-5, 情報処理学会, Nov. 1989.

[7] Nilsson, M. and Tanaka, H.: *Massively Parallel Implementation of Flat GHC on the Connection Machine*, Proc. of the Int. Conf. on Fifth Generation Computer Systems, 1988. p1031-1040.