3N-7

論理型言語向き並列計算機KPRのプロセス管理方式

平田博章、山本雅亮、柴山 潔、萩原 宏
(京都大学・工学部)

1. はじめに

KPRは、論理型言語プログラムの実行モデルとして我々が提案した『並列リダクション・モデル』に基づいて設計された論理型言語向きマルチプロセッサ・システムである。KPRが対象とする言語はOR並列型の論理型言語であり、論理型言語の持つOR並列性は並列リダクション・モデルのO(Or)プロセスによって自然に実現されるが、AND並列性については、S(Stream)プロセスによって、疑似的に実現される。つまり、Sプロセスでは、節本体のAND関係にあるゴール列を逐次的に処理する。このとき、各ゴールの呼び出し(複数プロセスの起動)に対して一般には複数の解が返されるので、1つのゴール列を処理する過程では、複数プロセスのストリームが生成され、それらが見かけ上並列に実行される。

本稿では、KPRのプロセス管理、特に、動的負荷分散方式とその問題点、および動的負荷分散を支援するKPRのハードウェア機構について述べる。

2. KPRのシステム構成

KPRは図1に示すようなシステム構成をとる。二分木状のネットワークの葉の部分には各要素プロセッサ(PE)が、中間ノードにはNNU (Network Node Unit)が配置されている。各PEはOプロセスを処理するORP (Or Reduction Processor)、Sプロセスを処理するARP (And Reduction Processor)の2種類の専用プロセッサの対から成り、また、PCU (Process Control Unit)と呼ばれる専用ユニットがPE内における種々のプロセス管理を担う。

KPRにおける動的負荷分散は各PE内のPCUと各NNUが分担して実現する。KPRのシステム規模が大きくなると、1台の特別なユニット(プロセッサ)がシステム全体の状況を集中管理しながら負荷分散を行うのは現実的ではない。そこで、各PCU、NNUがそれぞれ局所的な情報を用いて互いに協調し、システム全体として負荷をうまく分散させる方式を考えた。

負荷分散を行う上で、NNUとPCUは次のような役割分担を行う。すなわち、NNUはPE(PCU)からのデマンド(新しいプロセスを割り付けることを要求するメッサージ)をどのPEに配るかという点に関して責任を負い、一方、PCUは新しいプロセスを自分自身(PE)で処理するのか、あるいは、ネットワークを介して他のPEに委託するのかを決定する点に関して責任を負う。

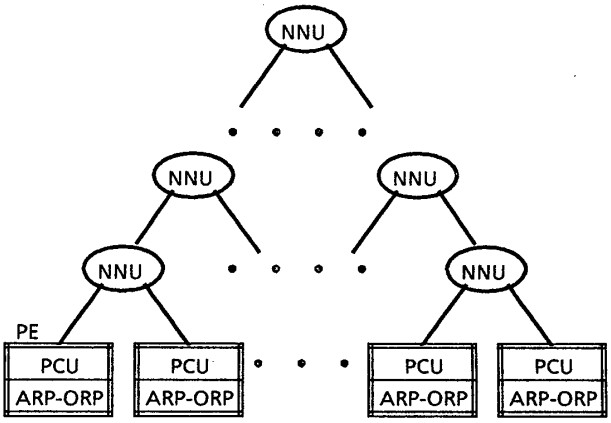

図1 KPRのシステム構成

3. 動的負荷分散支援ハードウェア機構

KPRが二分木状のネットワークを採用していることを利用し、各NNUは、自分を根とする右部分木、左部分木、自身より上位の部分という3つの単位で負荷の大きさを管理する。各NNUは隣接するNNU(またはPCU)との間で互いの負荷情報を受け渡しすることによって、システム全体の状態を反映した負荷分散を行う。各PCUから知らされるPEの負荷情報は、いくつかのNNUを介して、間接的に他のPCU(PE)に伝えられる。従って、各PEが他のすべてのPE個々の負荷を知ることはできないが、少なくとも、自分よりも負荷の小さいPEが他に存在するかどうかの判断を下すことはできる。PCUは、この負荷情報をもとに、ネットワークへデマンドを送出するか否かを決定する。

NNUは負荷管理と連動する高速バス・スイッチ機構を備えており、隣接したNNU(PCU)から送られてきたデマンドに対して、負荷状態や通信パス長を考慮してその伝播方向を決定し、ネットワーク・バスの高速スイッチングを行う。PCUが発したデマンドの送出先は、複数のNNUが各NNU間のバスを順次切り換えて行く方式により、動的に決定される。

Process Management of a Logic Programming Language-Oriented Parallel Machine: KPR
Hiroaki HIRATA, Masaaki YAMAMOTO, Kiyoshi SHIBAYAMA, Hiroshi HAGIWARA
Kyoto University

4. 動的負荷分散方式

負荷分散方式を考える上で重要な問題は、並列処理効果と通信オーバヘッドとの間のトレードオフであり、適切な処理単位(granularity)が設定されなければならない。KPRが採用している二分木状ネットワークの欠点の1つは、ランダムに通信を行った場合、木の根に近いノードのトラフィック量が大きくなることである。従って、KPRにおける通信には、局所性を持たせる必要がある。また、KPRでは論理プログラムの本質的な実行処理と、プロセッサ間通信とを時間的にオーバラップさせる方式で通信のオーバヘッドを軽減している。

以下に、KPR上での動的負荷分散方式を考える場合に配慮すべき事項について述べる。

①ユーザによる指定

負荷分散に関してユーザがプログラム中で何らかの指示を与える方式では、本質的な論理プログラミング以外の要素に気配りする必要があり、また、プログラミング結果としてのプログラムもKPRのシステム構成(PEの台数など)に大きく依存したものとなりがちである。従って、KPRでは、基本的には、負荷分散はシステム側で制御·実現することにし、ユーザの指示は、せいぜい、子プロセスを親プロセスと同じPEで処理するか、あるいは、別のPEで処理するかを指定できる程度に制限する。

②負荷の定義

負荷分散を考えるとき、一般には、各PEの待ち行列長を「負荷」の度合いとすることが多い。KPRの場合でも、基本的には、負荷を待ち行列長と定義する。しかし、負荷分散機構が必要とする負荷情報とは、他のPEに自分があとどれくらいの仕事を請け負うことができるかを提示する性質のものであるため、単純に負荷を待ち行列長とする定義では充分でない。つまり、PE内のローカル·メモリにどれくらいの空き領域があるかについての情報についても配慮する必要がある。

あるPEが他のPEにデマンドを送出しようとしても、これを受け入れる余裕のあるPEがなければ、自分自身で処理するか、他のPEに余裕ができるまで待たなければならない。この場合には、待ち行列長よりもむしろメモリ領域の方が問題となる。PEが、どのような状況になれば、ネットワークに対して「自分は、本当にこれ以上デマンドを受け付けることはできない」と宣言すればよいかという問題に対する最適な解を見つけることはなかなか難しい。

③並列性の組み合せ的爆発

KPRはOR並列型言語を対象としているため、全解探索を要する問題に対して、特に充分な適応性を備えていると考えられる。しかし、何の制限も加えずに幅優先的にAND/ORプロセス木を展開·探索したのでは、処理を待つ多くのプロセスの環境がKPRのメモリを埋め尽くすという事態も予想される。このような並列性の爆発を防ぐため、(i)デマンドよりもイベント(解)による処理の方を優先して実行する、(ii)デマンドに対しては、古いプロセスよりも新しいプロセスの方を優先して処理するなど、PE内では深さ優先的なスケジューリングを行う。また、これによって、通信を行う場合の局所性も高くなることが期待できる。

④負荷情報の表現ビット数

NNU間、あるいはNNUとPCUとの間での負荷情報の受け渡しは、専用の信号線を用いて行う。NNUでデマンドのルーティングを行う際、この信号線のデータ幅を多数ビットにして細かく負荷の比較を行っても、僅少差で負荷の少ないところに対してデマンドが集中するという現象が生じることも考えられ、また、ハードウェアのコストも高くつく。従って、できるだけ少ない本数の信号線で有効な負荷分散が行えるようにしなければならない。

PCUでは、待ち行列長を適当なビット数の表現形式に単純に線形変換するだけでなく、対数をとる、あるいは表現形式の尺度を動的に変更するなど、種々の方式を実際に試みる予定である。

⑤ARP-ORP対構成の利用

KPRのPEはARPとORPの対を中心に構成される。これは、論理プログラムを実行する場合には原則としてSプロセスとOプロセスが交互に起動されることを意識したハードウェア構成上の工夫である。デマンドをPE外部に出すか否かを決定する際に、局所性を考慮するにしても、ARPとORPを独立に扱うことは可能である。逆に、SプロセスからOプロセスへ(またはその逆)のデマンドは、必ず同じPE内の対のプロセッサに割り付けるといった戦略を採ることも可能である。結局、これはgranularityをどれ位に設定するのかという問題に帰着される。

⑥節本体の長さと負荷分散

Sプロセスへは複数の解が返され、これによって、また新しいプロセスが起動される。一般に、節本体の長さが長く(ゴール数が多く)なると、返されてくる解の数も多くなる。これは通信の集中を招き、また、通信の局所性がかえって適切な負荷分散を妨げるという好ましくない結果を生む。従って、述語呼び出しのオーバヘッドと負荷分散効果とのトレードオフを考慮した上で、非常に長い本体を持った節に対しては、コンパイル時にこれを書き直すなどの最適化処理を適用する必要があろう。

5. おわりに

以上、KPRの動的負荷分散方式について述べた。今後、シミュレーションによって前節の①~⑥を中心に負荷分散方式を検討してゆく。応用プログラムには、n-queenなどのゲーム、論理シミュレーション、構文解析·文生成プログラムなどを考えている。なお、負荷分散の様子は、これら応用プログラムに大きく影響されると考えられ、従って、応用プログラムの並列度のチェックや負荷分散評価に必要なツールの開発も行っている。