Committed-Choice 型言語 Fleng の WS クラスタ処理系における負荷分散機構

4 E − 6

大内 敦夫、田中 英彦
東京大学大学院工学系研究科

1 はじめに

Committed-Choice 型言語 Fleng は、細粒度並列処理を目的として作られたプログラミング言語である。著者らは Fleng 言語の処理系を WS クラスタ上に実装し、実行速度の改善を目的に改良を行なってきた [1]。本研究では、この処理系における負荷分散機構に着目し、その改良によってどの程度実行速度が改善されるかを検討する。

2 Fleng 言語

Committed-Choice 型言語の一種である Fleng のプログラムは、次のような形をしたホーン節から構成されている。

$$H : -B_1, B_2, \cdots, B_n. \ (n \geq 0)$$

Fleng プログラムの実行は次のようにして行なわれる。

最初にいくつかの初期ゴールが与えられ、ゴールプールと呼ばれる待ち行列に格納される。ゴールプールから適当なスケジューリングによってゴールを 1 つ選び、それに対応する定義節を一つだけ選び出す。そして、そのボディ部の各述語 $B_1, B_2, \cdots, B_n$ を新たなゴールとし、ゴールプールに格納する。この新たなゴールを生成する操作をリダクションと呼ぶ。これを、ゴールプール中のすべてのゴールが処理されるまで繰り返す。

Fleng では、単一代入変数や、定義節の選択に必要な変数のバインドを待つサスペンド・アクティベートの機構等により、プロセス間の排他制御や同期をあまり意識することなくプログラムを作成することができる。また、プログラムがリダクションによって非常に小さい単位に分割され、それらを可能な限り並列に実行しようとすることから、Fleng は細粒度の並列処理を記述することが可能である。

3 WS クラスタ

WS クラスタは、複数台のワークステーションをネットワークによって接続し、並列に動作させる計算機システムである。システムのコンポーネントとして、既存のワークステーションや Ethernet 等のネットワークを用いるため、専用の並列計算機に比べて、システムの作成が容易・安価である反面、通信によるレイテンシが大きいといった欠点がある。

Study of Load Distribution for the Runtime System of Committed-Choice Language Fleng on WS Cluster
Atsuo OUCHI, Hidehiko TANAKA
Graduate School of Engineering, The University of Tokyo

著者らが過去に行った実験 [2] では、SparcStation20 を Ethernet で接続した WS クラスタの場合で、1 回の通信に要する時間は 1 回のリダクション操作に要する時間の 1000 倍程度であるという結果が出ている。従って、WS クラスタによるプログラムの並列実行によって速度向上を図るためには、WS 間の通信を可能な限り避け、レイテンシに見合うだけの速度向上を得られる場合のみ、複数の WS による並列実行を行なうことが必要となる。

4 処理系における負荷分散機構

WS クラスタ用 Fleng 処理系に当初実装した負荷分散機構は、プログラム中に明示的にゴールの WS 間転送を指示することによって行う静的負荷分散のみであった。しかし、この方法では、例えば quicksort の場合で、6 台の WS を使用しても最大で 1 台の場合の 2 倍程度の速度向上しか得られない [1] といったように、プログラムの実行速度向上が理想比に比して小さい。

またその他にも、以下のような点が問題となる。

- 実行時環境を負荷分散に反映させることが難しい。例えば、速い WS と遅い WS の混合したクラスタと、均一な速度の WS だけから成るクラスタとでは、最適な分散方法は異なる筈である。
- プログラムにとって最適な負荷分散が、あらかじめプログラマに分かっているとは限らない。負荷分散に関する記述をプログラマが行わなくても、自動的に適切な負荷分散が行なわれる方が望ましい。

以上のことから、本研究では WS クラスタ処理系に動的な負荷分散を行なうような機構を付加することを考える。その際、前述の通り、通信レイテンシが非常に大きいシステムであることを考慮し、WS 間で転送するゴールは、粒度が十分に大きく、そのゴールの実行に大量の通信が発生しないようなものであることが求められる。そのため、プログラムを解析し、ゴールの粒度やゴール間の通信データ量を推定して、適切なゴールのみを分散の対象とする。

4.1 プログラムの静的解析

並列論理型言語[1]におけるプログラムの静的解析の研究は過去にいくつかなされているが [3][4]、これらはゴールの粒度の解析のみを行なっており、通信量についての解析は行なっていない。

[1] Committed-Choice 型言語は並列論理型言語の一種である

ゴールが複数のWSに分散している場合、ゴール間の通信はそれらの間の共有変数によって行なわれる。この変数に対して複数のWSから頻繁にアクセスが行なわると、アクセスによる通信レイテンシの為に実行速度が低下する。従って、共有変数による通信が多いゴール群は同一のWS上で実行し、相互の通信量の小さい複数のゴール群をWS間に分散させるといった区別が必要となると考えられる。

本研究では、前出の手法を基に、ゴール間の通信量の解析を加えて、高通信レイテンシシステムにおいて適切な負荷の分割を求める。手順としては以下のようになる(図1)。

1. Flengプログラムを読み込み、ゴールの親子関係を表すAND-ORグラフを作成する。
2. グラフ中から再帰構造を発見し、再帰構造全体をノードとする。
3. グラフの構造から各ノードの粒度を推定する。ノードの粒度は、一般に入力となる変数の大きさに依存するため、実行時に入力の大きさ、又はゴールの粒度そのものを情報として持つ必要がある。
4. 親子・兄弟ノード間の共有変数による通信量を解析する。
5. 以上の結果から、粒度に比して通信量の少ないような、転送の候補となるゴールを決定する。粒度が一定以上のサイズで、他のWSに転送してもすぐには終了しないこと、共有変数に対する読み出し/書き込みの頻度が小さく、他のWSとの通信があまり発生しないことが条件となる。

4.2 動的負荷分散

実行時環境に応じた適切な負荷分散を行うためには、実行時にどのように負荷を分散するか決定する動的負荷分散が必要である。本研究では、リダクションを行なうべきゴールを持たないWSが、他のWSに対して処理の分割を要求するon-demand型の負荷分散を行なう。

但し、WSクラスタの高通信レイテンシを考慮すると、負荷分散機構自身が多量の通信を発生するようなものであってはならない。従って、動的負荷分散機構としてあまり複雑なものを使用することは望ましくない。

本研究においては、ゴールを持たないWSが、ランダムに選んだ他のWSに対してゴールを要求するという戦略を用いる。この方法は、負荷分散に際してゴールの要求以外に通信が必要なく、最も単純な戦略の一つであるが、性能に関しては今後検討の余地が存在する。

5 おわりに

WSクラスタという高通信レイテンシシステムにおいて細粒度並列処理を行なう際に、負荷分散を効率的に行うための研究について述べた。現在性能評価を行なっている。

今後、負荷分散機構を含めた処理系全体の評価と各種高速化を行なう予定である。

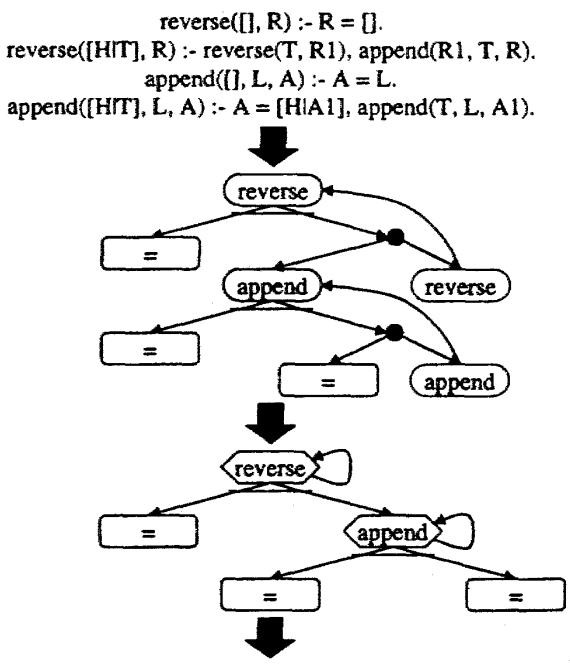

図1: naive reverseの解析例

参考文献

[1] 大内 敦夫, 荒木 拓也, 田中 英彦, "並列論理型言語Flengのワーステーションクラスタへの実装", 情報処理学会第52回全国大会, vol. 6, pp. 171–172, 1996.

[2] 大内 敦夫, 荒木 拓也, 田中 英彦, "Committed-Choice型言語Flengのワークステーションクラスタ上処理系の定量的評価", 情報処理学会第53回全国大会, vol. 1, pp. 349–350, 1996.

[3] X. Zhong, E. Tick, S. Duvvuru, L. Hansen, A. V. S. Sastry and R. Sundararajan, "Towards an Efficient Compile-Time Granularity Analysis Algorithm", in Proc. of the International Conference on Fifth Generation Computer Systems 1992, pp. 809–816, 1992.

[4] S. K. Debray, N.-W. Lin and M. Hermenegildo, "Task Granularity Analysis in Logic Programs", in Proc. of the ACM SIGPLAN'90 Conference on Programming Language Design and Implementation, pp. 174–188, 1990.