動的スケジューリング可能な一般化静的順序制御機構についての検討

6G－7

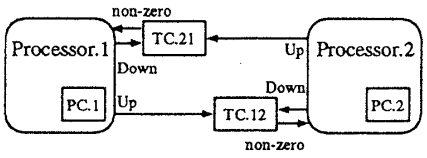
服部基保†　有田隆也†　石井直宏†　曽和将容‡
†名古屋工業大学　‡電気通信大学

1. はじめに

細粒度並列計算機を効率よく実行するための同期機構として，我々は「一般化静的順序制御機構」[1]を提案している。この機構はカウンタ結合網により構成されており、実行可能である命令を極めて高速に検出できる。しかし、実行タイミングの動的変動に対する影響を最大限に押えることが可能ではあるが、各命令の処理ユニットへの割り当てをコンパイル時に行うので、なお、不要な待ちが生じる可能性がある。本論文では、各命令の処理ユニットへの割り当てを実行時に行うための簡単なハードウェアを付加した一般化静的順序制御機構を採用した細粒度並列計算機アーキテクチャを提案する。これにより、処理時間の変動が存在する場合の効率的な実行やコンパイルを含めた全処理時間の短縮が期待できることをシミュレーション評価に基づいて示す。

2. 一般化静的順序制御機構

一般化静的順序制御機構は、図1のような構成になっている。この機構は、各命令の処理ユニットへの割り当ての済んだプログラムを、各処理ユニットにあるプログラムカウンタを用いてフェッチし、カウンタ結合網を用いて実行可能である命令を検出し、実行する。このカウンタをトークンカウンタと呼び、そのカウンタのアップ、ダウン、そして、その値が0であるかの検出で同期をとっているため、実行可能である命令を極めて高速に検出できる。このトークンカウンタは、PNプロセッサ[2]でも採用されている。



TC:token counter　PC:program counter

図1　処理ユニット2台の一般化静的順序制御機構

3. 提案するアーキテクチャ

提案するアーキテクチャ(図2)では、複数の処理ユニットはフェッチ・実行可能命令検出ユニット(FU)と実行ユニット(PU)から構成される。フェッチユニットは、命令間の本質的な先行関係に関する情報が付加されたプログラムを最大並列度を持つ定並列プログラムとしてフェッチし、トークンカウンタを用いて実行可能命令検出ユニットで実行可能判別を高速に行う。このため、FUはプログラムの最大並列度分の台数を想定する。実行可能になった命令は、セレクタ機構で実行時にPUへの割り当てを行い、PUは各命令を実行する。PUは、一般化静的順序制御機構とは異なりFUより少ない台数用意すればよい。また、トークンカウンタを用いるために、命令を実行し終ってから、PUはその命令をフェッチしたFUへ実行終了信号を送信する必要がある。

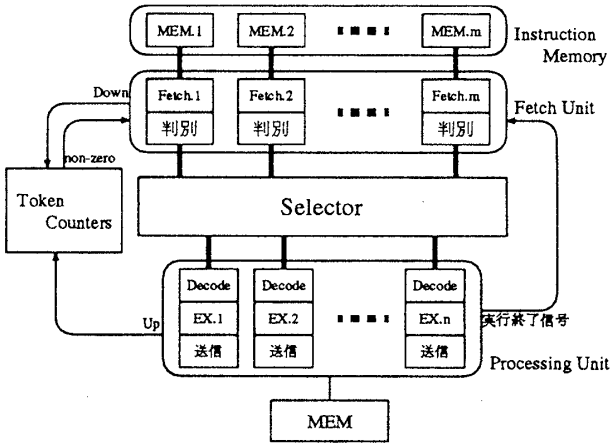


図2　提案するアーキテクチャ

セレクタの機構(PUへの割り当て方法)を、次の6通りについて、検討する。

方法1　実行可能な命令群の中から、番号の小さいFUでフェッチされた命令順に割り当てる。FUとPUを接続しているスイッチの状態(ONOFF)を制御することによって、命令を割り当てる。そのスイッチの状態は、各FUでのフェッチした命令の状態(実行可能、不可能)と、各PUの状態(命令実行中、待機中)の信号で決定する。

方法2　実行可能な命令群の中からランダムに割り当てる。各FU、PUの状態の信号とランダムに反転する信号で、スイッチの状態を決定する。

方法3　実行可能になった順に割り当てる。同時に実行可能になった場合は、番号の小さいFUでフェッチされ

*A Study of Static Synchronization Mechanism with Facilities for Dynamic Scheduling
Motoyasu HATTORI †　Takaya ARITA †
Naohiro ISHII †　Masahiro SOWA ‡
†Nagoya Institute of Technology
‡University of Electro-Communications

た命令順に割り当てる。この方法では、一つの高速なキューを用い、ＦＵは実行可能になった順にキューに命令を入れていく。同時に実行可能になった場合は、番号の小さいＦＵでフェッチされた命令順に入れる。ＰＵはキューから命令を取り出して実行する。キューを用いているため、同時に複数の命令をＰＵに送信できない。

方法４　実行可能になった順に割り当てる。同時に実行可能になった場合は、ランダムに割り当てる。高速なキューを用い、ＦＵは実行可能である命令をラウンドロビン方式でキューに入れ、ＰＵは方法３と同様である。

方法５　実行可能な命令の中からＣＰ長の長い順に割り当てる。ＣＰ長を実行前にあらかじめ計算しておいて、その値をタグとして付加する必要がある。タグを見て、大きい順に割り当てなければならないため、ハードウェア量が大きくなる。なお、今回は分岐命令を使用していないサンプルプログラムのみ評価している。

方法６　ＣＰ法で静的にスケジューリングして命令を並びかえ、実行する。ＦＵとＰＵがパイプライン化した一般化静的順序制御機構といえる。

4. シミュレーション評価

３で示した６通りの方法に関して、５つのサンプルプログラムでシミュレーション評価を行った。処理時間の変動がある場合は、メモリをアクセスする命令の１５％が変動するとして評価を行った。

分岐を含まない三つのプログラムの平均(C1)(括弧内は個々のプログラム)、実行前に回数の不確定な独立した５つのループを含むプログラム(C2)、４０回ループするプログラム(C3)での処理時間の変動がある場合の方法６に対する相対実行時間(表１)と、変動がない時に対するある時のC1の実行時間の増加率と処理時間の変動がある場合のC3のＰＵ利用率(表２)を示す。

表１より、C1では、方法６に比べて平均では数％悪いが、方法２～４が良いプログラムもある。C2では８～２５％よい。これは、実行前に命令のＣＰ長がわからないためにうまくスケジューリングできないからである。C3では、６～８％悪い。これは、スケジューリングしたプログラムと比較して、ループ一回につき分岐命令を一回余分に実行しなければならないためである。しかし、ＰＵ利用率は、表２より方法６よりも７～１０％高くなっている。割り当てを実行時に行う方法(方法１～５)は、静的にスケジューリングして実行する場合(方法６)より、処理時間の変動時に性能低下を平均すると５０％押えている。

方法１～５を比較すると、方法５、方法４、方法３か２の順で良く、方法１が一番悪い。方法１と３は、実行する前のＦＵへの命令の配置によって実行時間がかなり左右される。そのため、実行時間のばらつきが大きい。方法４と方法２とは、１～２％ぐらいしか変わらない。また、方法５はハードウェア量が大きく、方法４は高速なキューを用いているため複数の命令を同時にＰＵに割り当てることができない。以上の検討により、現段階では方法２が良いと判断する。

	相対実行時間		
	C1	C2	C3
方法１	1.088 (1.014,1.067,1.168)	0.916	1.069
方法２	1.013 (0.950,1.044,0.999)	0.774	1.086
方法３	1.014 (0.943,1.053,0.990)	0.757	1.048
方法４	1.003 (0.935,1.040,0.981)	0.758	1.085
方法５	0.955 (0.935,0.965,0.948)		
方法６	1.000 (1.000,1.000,1.000)	1.000	1.000

表１　方法６に対する相対実行時間

	C1の増加率 [%]	ＰＵ利用率 [%]
方法１	4.1　(6.0, 4.1, 3.1)	90.7
方法２	8.1　(14.5, 7.4, 5.8)	89.2
方法３	7.2　(16.1, 5.3, 5.9)	92.4
方法４	8.1　(16.5, 7.0, 5.9)	89.3
方法５	9.4　(15.2, 8.8, 7.3)	
方法６	14.6　(23.2,12.7,13.2)	82.1

表２　C1実行時間の増加率とC3のＰＵ利用率

5. おわりに

一般化静的順序制御機構の拡張手法を示し、シミュレーションにより比較した。処理時間の変動が存在する場合、シミュレーション結果やハードウェアのことを考えると、方法２がよいことを示した。また、実行するプログラムによっては、かなり実行時間が短縮できることを示した。このことより、コンパイルを含めた全処理時間の短縮も期待できる。

参考文献

1. 高木浩光, 有田隆也, 曽和将容, "問題が持つ先行関係のみを保証する高速な静的順序制御機構", 情報処理学会論文誌,Vol.32,No.12,pp.1583-1592(1991).

2. 有田隆也, 伊藤広明, 曽和将容, "複数命令流をもつ機能分割型スーパスカラプロセッサのシミュレーション評価", 電子情報通信学会論文誌,Vol. J74-D-I,No. 9,pp.635-643 (1991).