5J-05

データ駆動型ノイマンマシン（DDNM）を前提とする整数ベンチマークにおける命令レベル並列度

田中 美喜　谷川 一哉　高橋 隆一　吉田 典可
広島市立大学 情報科学部 情報工学科

1 はじめに

データ駆動型ノイマンマシン (DDNM) は，広いメモリバンド幅を確保し，内部的に，多数のデータ駆動型の演算セルを備えることによって，ノイマン型コンピュータを高速化，高信頼化する試みである [2][3][4][5].

今回は，このDDNMアーキテクチャを前提とする**SPECint95**ベンチマークにおける命令レベル並列度を論じる．

2 DDNMの構成

図1にDDNMの構成を示す．モジュールフェッチによってモジュールバッファにプログラムモジュールが取り込まれる [4][5]．NDD(Neumann Data Driven) コンバータはノイマン型のプログラムのデータ依存解析を行い，真のデータ依存関係のみを抽出する．DDNMにおいて外部とのデータの受渡しを行なうのはパケットマネージャである．プログラムの初期データはパケットマネージャに渡され，演算命令はオペレーションアロケータに渡される．最終的な結果がパケットマネージャに返され，結果が出力される [3][4].

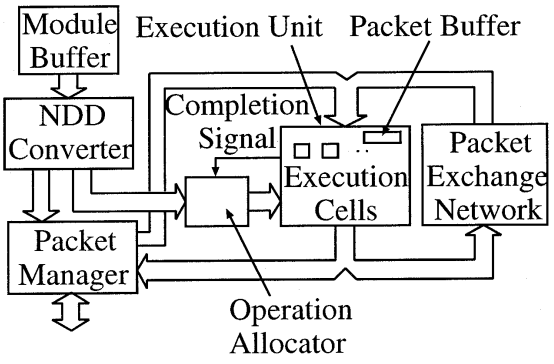

図 1: DDNMの構成図

3 DDNMの制御

オペレーションアロケータは演算セルに命令を割り付ける．また，演算セルからステータス信号を受け取ることにより演算セルの状態を管理する．ステータス信号には演算の終了信号と演算セルの障害情報が含まれる．

オペレーションアロケータは空いている演算セルに命令を割り付ける．演算セルは受け取った命令のオペランドのデータが揃い次第，命令の実行を開始する．

4 命令レベル並列度

本稿では，実行中の命令が他の命令に依存しないことによる命令シーケンス間でのオーバーラップの可能性を命令レベル並列度 (ILP:instruction-level parallelism) と呼ぶ．プログラム自身の持つ並列性をできる限り多く抽出し，並列度を増加させることで，コンピュータの性能を向上させることができる．

通常のノイマン型コンピュータにおいて，命令レベル並列度を増加させるための最も単純な方法は，ループのイテレーション間の並列性を活用することである．この並列性はループレベル並列性 (loop-level parallelism) と呼ばれることがある [1]．ループレベル並列性を活用する方法として，ループアンローリングやベクトル命令の使用がある．

4.1 分岐の解決

本稿では，命令系列で，出口を除いて分岐命令がなく，かつその入口を除いて外から分岐されることのない処理対象を基本ブロックと呼ぶ．

図2に基本ブロック間の制御フローを例示する．矢印は特定の条件下での基本ブロックの実行順序を表す．

いま，この順に基本ブロックが実行されるものとして，current block（BB0）の分岐命令の実行が終了して初めて次の基本ブロック（BB1）が実行される場合を分岐の解決数0という．BB1の分岐命令の実行が終了するとBB2の実行が開始される．

分岐の解決数1の場合，まずBB0とBB1の実行が開始される．BB0の分岐命令の実行が終了するとBB2の実行が開始される．さらにBB1の分岐命令の実行も終了するとBB3の実行が開始される．

分岐の解決数2の場合，最初にBB0，BB1とBB2の実行が開始される．BB0の分岐命令の実行が終了するとBB3の実行が開始される．

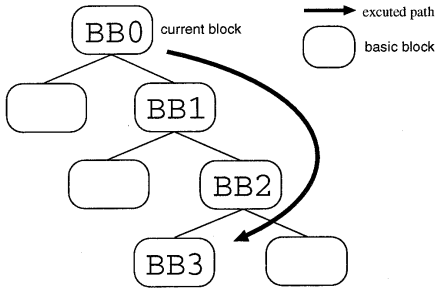

図 2: 分岐の解決

4.2 基本ブロック長の相違

基本ブロック長は，長い方が多くの並列度を得られると期待される．

分岐の解決数を一つ増すと一つ多く先の基本ブロックを見ることができる．分岐命令数が少なく基本ブロック長が長いと，分岐の解決数が同じでもプログラムを広範囲にわたって見ることになる．このことにより，オーバーラップの可能性を見い出しやすくなり，クリティカルパス長を短くすることができると考えられる．

図3に，入力とする機械語に対し，ループアンローリングを行なっていない命令 (a) と，行なっている命令 (b) の相違を示す．図中の四角は基本ブロック，節点は命令，枝は依存関係を示す．破線枝は基本ブロックを跨ぐ依存関係である．黒い節点は分岐命令を表す．

(a) と (b) は同じ処理である．上から3つ目の基本ブロックが並列実行の対象となったとき，(a) ではループ

1回の繰り返し分の短い基本ブロックまでしか並列度抽出の対象とならないのに対し，(b) では2回の繰り返し分の命令が並列性を見い出す対象に加わる．その結果，実行ステップ数が9から7に減少している．

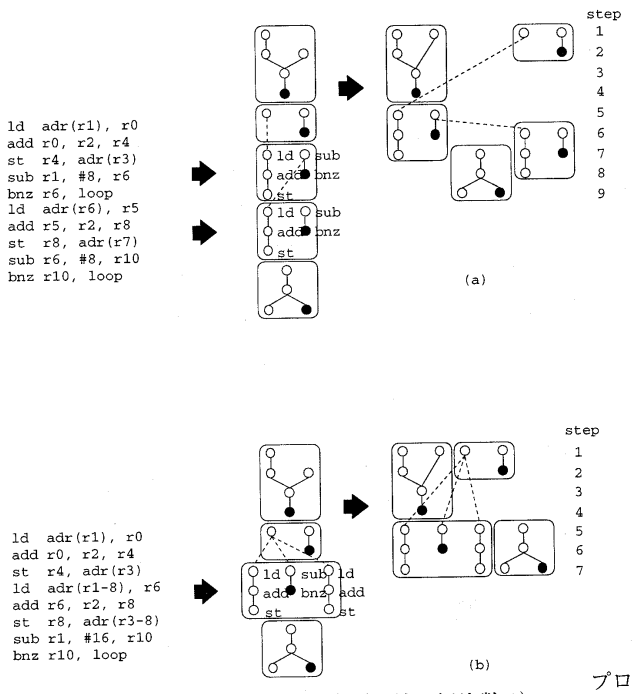

図 3: 基本ブロック長による相違（分岐の解決数 1）

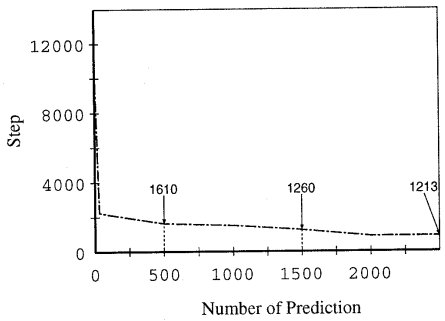

図 4: “碁” のプログラム (-O2) の評価結果

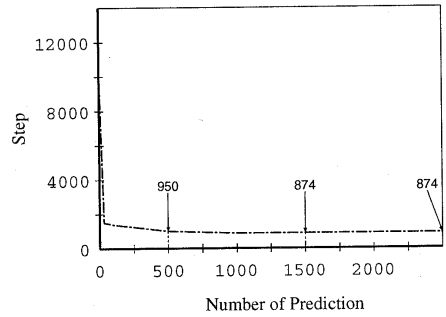

図 5: “碁” のプログラム (unroll) の評価結果

4.3 コンパイルオプションの影響

ここではSPECint95ベンチマークテストgo（碁）のプログラムのgetmove関数の評価結果を示す．この関数は碁の一手を打つ部分に相当する．コンパイル時のオプションは`-O2`及び`-O2 -f`*unroll-all-loops*を用いた．評価結果を図4，図5に示す．横軸は分岐の解決数である．縦軸はDDNMにおける実行ステップ数を表す．ILPの最大値は約2500であった．

ステップ数の最小値においてはオプションによる差はほとんど見られなかった．しかし，ステップ数の減少の仕方を見ると，`-O2`に比べ`-O2 -f`*unroll-all-loops*のほうが早く減少していることがわかる．

表1にオプションによる平均基本ブロック長の違いを示す．ここで，

平均ブロック長 = 全命令数/(分岐命令数 + 1)

である．

表 1: go のプログラムの平均基本ブロック長

オプション	全命令数	分岐命令数	平均基本 block 長
-O2	23459	3942	5.9
unrolling	21353	3278	6.5

`-O2 -f`*unroll-all-loops*オプションでは`-O2`よりも平均基本ブロック長が長くなっている．

5 まとめ

DDNMの性能をプログラムの実行ステップ数という観点から評価した結果を示した．同じ操作を実行するプログラムでも，基本ブロック長の長いプログラムを入力プログラムとすることで，少ない分岐の解決数でも高い性能が得られることが分かった．

NDDコンバータの構成方法をより詳細に調べることなどが今後の課題である．

参考文献

[1] Hennessy J. L. and Patterson D. A.: Computer Architecture A Quantitative Approach 2nd edition, Morgan Kaufmann Publishers, Inc. (1996)

[2] Ryuichi Takahashi and Noriyoshi Yoshida: "Diagonal Examples for Design Space Exploration in an Educational Environment CITY-1" Proc. 1999 International Conference on Microelectronic Systems Education pp.71-73 (1999)

[3] 谷川一哉，高橋隆一，吉田典可：データ駆動型ノイマンマシン (DDNM) におけるスケジューリング，第58回情処全大 2H-9 (1999)

[4] 小椋祐治，高橋隆一，吉田典可：データ駆動型ノイマンマシン (DDNM) の構築，第58回情処全大 2H-10 (1999)

[5] 小椋祐治，高橋隆一，吉田典可：ノイマン型コンピュータのデータ駆動型マシン技術を用いた高速化，第56回情処全大 2N-4 (1998)

[6] 谷川一哉，高橋 隆一，吉田 典可：データ駆動型ノイマンマシン (DDNM) における乱流計算 ～ SPECfp95 ベンチマークテストの試み～，第50回中国支部連大 102004 p.269 (1999)

[7] 小椋 祐治，高橋 隆一，吉田 典可：データ駆動型ノイマンマシン DDNM における単一プロセス実行～パケット交換ネットワークの一論理構成手法～第50回 中国支部連大 102005 p.270 (1999)