2H−9

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 ～SPECint95ベンチマークテストの試み～

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

DDNM（Data-Driven Neumann Machine）はソフトウエアにとってノイマン型コンピュータであるが，内部的にデータ駆動型の演算ユニットを備えて高速化，高信頼化を図ろうとする試みである[1],[2],[3]．プログラムモジュール単位でフェッチができる程度のメモリバンド幅を確保し，各プログラムモジュールはクリティカルパスでの処理速度を目標性能としている．

筆者らはノイマン型コンピュータの機械語をデータ駆動型の演算ユニットを前提とする形に変換する**NDD**（Neumann-Data-Driven）コンバータを考え，真の依存関係のみを抽出することを考えた．本稿では**DDNM**においても従来技術のスケジューリングと同様な制御が可能なことを示し，SPECint95ベンチマークテストの「碁」（getmove()関数）の評価結果を示す．

2　DDNMアーキテクチャ

図1に**DDNM**の構成を示す．モジュールバッファにはノイマン型コンピュータの機械語命令と初期データが格納される．**NDD**コンバーターはデータ駆動型マシン用にプログラムを変換する．パケットマネージャーは初期データを演算セルに渡す機能と，得られた結果を出力する機能を持つ．**DDNM**において外部とデータの受渡しを行なうのはパケットマネージャーである．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．データパケット交換ネットワークは，演算セルから出力される結果を次の演算セルに分配する．**DDNM**中を流れるパケットはタグフィールドとデータフィールドから構成される．タグフィールドのプロセスIDにより，複数のプログラムが同時実行可能である．

2.1　プログラムの型変換

NDDコンバータによる型変換を図2に示す．**NDD**コンバータは真の依存関係のみを抽出する．入出力部分はパケットマネージャーに渡され，演算実行部分はオペレーションアロケータに渡される．演算実行部分にはフロー依存のデータ依存関係だけが残る．

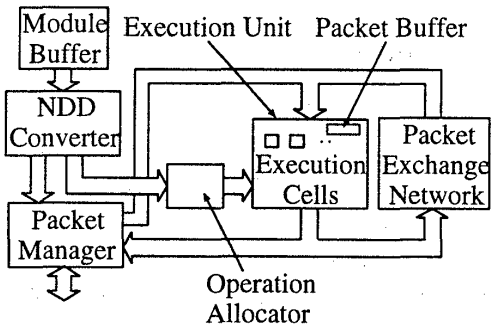

図1：**DDNM**の構成

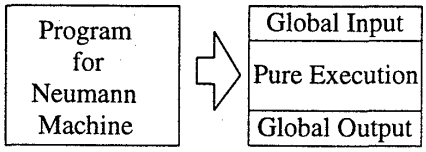

図2：**NDD**コンバータによる型変換

NDDコンバータによる型変換はレジスタリネイミングと同様であるといえる．レジスタはデータフローグラフのアークに変換される．ある命令でレジスタへの書き込みが行なわれる度に新しいタグを割り当てる．レジスタが使用される場合はその命令より先行する命令でそのレジスタが置き換えられたタグを使用する命令に置き換える．フラグレジスタのタグにはフラグを設定する演算の結果に割り当てられたタグと同じタグが使用される．

ストア命令は，書き込むアドレスをストアされるデータのタグで置き換える．ロード命令は参照するメモリアドレスがタグで置き換えられていなければ，そのデータに新しいタグを与える命令に変換される．タグで置き換えられていれば，レジスタに割り当てるタグをメモリアドレスのタグで置き換える．この場合，命令は生成されない．分岐命令は，フラグレジスタのタグを参照する命令で置き換える．即値を使用する命令は，即値に対し新しいタグを割り付ける命令と，即値のタグを参照する命令とに変換される．

2.2　スケジューリング

あるパケットを使用する演算が演算セル上に割り付けられていない場合，そのパケットはパケットバッファに格納される．パケットバッファを導入することにより，演算ユニットの演算セル数を任意の個数に設定することが可能となる．

Scheduling Strategy for Data-Driven Neumann Machine
Kazuya TANIGAWA, Ryuichi TAKAHASHI and
Noriyoshi YOSHIDA
Faculty of Information Sciences, Hiroshima City University

演算ユニットに投入する命令数を変化させることにより実行を制御することができる．このことは従来のスケジューリングに相当する．データフローグラフの深さが浅い方から優先的に演算を割り付けるスケジューリングは従来技術のASAP(As Soon As Possible)[4]と同様である．

3 SPECint95ベンチマーク評価

今回の評価ではノイマン型コンピュータの機械語命令と初期データとしてはSPARC version8でのトレース結果を用いた．連続する基本ブロック数を変えてデータフローグラフを作成して演算ユニットに投入した．本稿ではそのことを分岐の解決と呼び，連続する基本ブロックの個数を分岐の解決数と呼ぶ．分岐の解決は分岐命令の実行が終了する度に行なわれる．分岐の解決は先行する分岐命令が全て終了するまで待たされる．本稿では演算セル数が加算無限個であるとした場合の碁の一手を打つ部分に相当する部分（getmove()関数）の評価結果を示す．トレースして得られたプログラムにあるレジスタ番号としては物理的なレジスタ番号に変換したものを使用した．

3.1 命令レベル並列度

図3にASAPでスケジューリングした場合の分岐の解決数と相対性能の関係を表す．ここに相対性能とは総命令数を分岐の解決により得られたデータフローグラフの深さの総和で除算したものを意味する．評価対象の全ての命令を投入した場合に40倍程度の性能が得られる．その場合の演算セル数（命令レベル並列度）は4044であった．相対性能が向上していない部分は基本ブロック間のデータ依存関係が強いため，多くの基本ブロックからデータフローグラフを作成しても並列に実行できる部分が少ない部分であると考えられる．

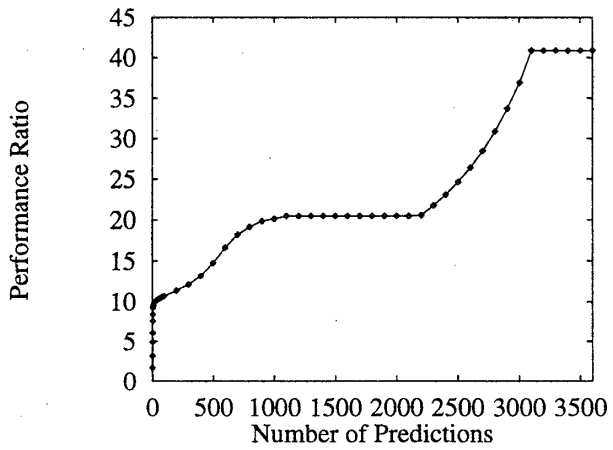

図3：分岐の解決数と性能の関係

3.2 パケットバッファ

図4に分岐の解決数とバッファサイズの関係を示す．あるタグを使用する演算が演算セル上に割り付けられていない場合にパケットはパケットバッファに格納される．分岐の解決は理想的な分岐予測を行なっている場合と等価だと考えられる．バッファサイズが減少している部分は，その前後にあるブロック間に強いデータ依存関係が存在している部分である．逆に増加している部分は，その前後にあるブロック間はそれよりも後にあるブロックにデータ依存関係がある部分である．

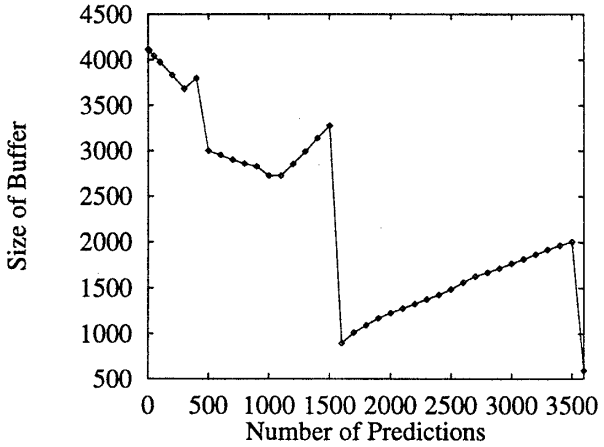

図4：分岐の解決数とバッファサイズの関係

4 まとめ

プログラムの型変換を行なうコンバータを導入し，スケジューリングと同様な制御が可能なことを示した．40倍程度という相対性能はプログラムモジュール程度というメモリバンド幅と比較して十分とはいえない．他のベンチマークプログラム，特に浮動小数点ベンチマーク評価を行なうこと，およびオペレーションアロケータによる制御を詳細に調べることが今後の課題である．

参考文献

[1] 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

[2] 高橋隆一，児島 彰，上土井陽子，吉田典可：マイクロコンピュータ設計教育環境City-1，情処研報 DA83-6, pp.41-48,1997

[3] 高橋隆一，吉田典可：システムのインテリジェント化を支えるディジタル設計教育，信学誌 Vol.81, No.9, pp.908-912, 1998

[4] 高橋隆一，吉村 猛：ハイレベルシンセシスの動向，信学論 A， Vol.J74-A, No.2, pp143-151(1991)