5J-06　データ駆動型ノイマンマシン（DDNM）を前提とする浮動小数点ベンチマークにおける命令レベル並列度

村上 和希　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，外部的にはノイマンマシンだが，内部的にデータ駆動型の演算ユニット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 [4][5][6]. **DDNM** は単一のプロセスに対しては，数百から数千の命令を同時発行可能なスーパスカラプロセッサとして動作する.

半導体技術の進歩によりシステム全体を載せることのできるほどの大規模な LSI が実現可能となった．筆者らは，システムを載せるのではなく，数百から数千を越える命令を同時発行可能な単一のプロセッサが実現可能になったと考えている.

本稿では，SPECfp95 ベンチマークにおける命令レベル並列度を具体例として，この規模は，通常，複数プロセスによって享受されるべきであることを示す.

2 DDNM の概要

図 1に **DDNM** の構成 [4] を示す．モジュールフェッチによってノイマン型のプログラムはモジュール単位でフェッチされ，モジュールバッファにとりこまれる．NDD 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．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．データ駆動型の演算ユニットは，初期データをもとに指令された演算を依存関係とハードウエア資源が許す限界の処理速度で実行する．最終的な結果はパケットマネージャに返され，結果が出力される [5][6].

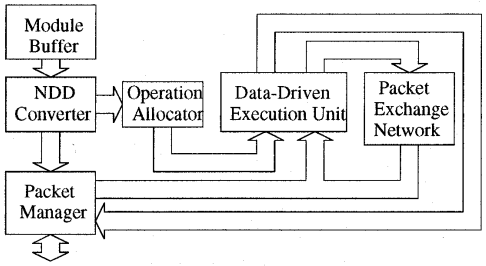

図 1: DDNM の構成図

3 NDD コンバータ

NDD(Neumann Data Driven) コンバータは真のデータ依存関係の抽出に際し，処理過程で用いられるレジスタやメモリを，データフローグラフのアークに変換する.

図 2に，レジスタがアークに変換される様子を示す．タグ（tag0,tag1,...）はデータパケットのフィールドの一部である．即値データは，データ依存解析を行った範囲で，共用されると仮定している.

```
add r0,r1,r2 ⟶  add   tag0,tag1,tag2
sub r2, 1,r3 ⟶  const   1,tag3
                sub   tag2,tag3,tag4
add r4, 1,r4 ⟶  add   tag5,tag3,tag6
and r3,r4,r3 ⟶  and   tag4,tag6,tag7
```

図 2: レジスタのアークへの変換

図 3に，メモリがアークに変換される様子を示す．ロードストアの動作は行われず，データが直接受け渡しされる．即値データの扱いはレジスタの場合と同様である．即値データの実現方法としては，データ駆動型演算ユニット内の演算セルに即値を出力させる方法が考えられる.

```
add r0,r1,r2 ⟶  add   tag0,tag1,tag2
st  r2, adr1
ld  adr1,r4
add r4, 1,r4 ⟶  const   1,tag3
                add   tag2,tag3,tag5
add r4, 1,r4 ⟶  add   tag5,tag3,tag6
```

図 3: メモリのアークへの変換

レジスタやメモリをアークに変換することは，通常は，従来型マシンでのフォワーディング [1],[2] しか行わないという設計と見なすことができる.

ハードウエア資源の制約で，直接の受け渡しができない場合，データはパケットバッファに保存される [5].

4 分岐の解決数

ノイマン型のプログラムのどれだけの範囲を変換し，データ駆動型演算ユニットに送り込むかのひとつの指標が分岐の解決数である.

本稿では，ノイマン型のプログラムの命令系列で，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ものを基本ブロックと呼ぶ.

分岐の解決数が 0 の場合，データ駆動型の演算ユニットには，基本ブロック単位で処理が投入される．解決数が 1 の場合は，ひとつ先の基本ブロックまでが投入される．解決数が n の場合は n 個先までの基本ブロックが投入される．可能な分岐先をすべて投入してしまう方法が最も単純である．完全な分岐予測と等価であるという考え方から，本稿では，分岐の解決数を予測数（number of predictions）とも呼ぶ.

5 命令レベル並列度

浮動小数点ベンチマークである **SPECfp95** を調べて，高い命令レベル並列度をもつ処理が見いだされた.

5.1 110.applu

110.applu は, 流体力学におけるアプリケーションを想定した, 偏微分方程式の浮動小数点ベンチマークプログラムである [7]. モデルとされた問題は, 以下の行列式による表現に帰着させている [3]:

$$K^n \Delta U^n = R^n \tag{1}$$

ここに, ΔU は, $\Delta U^n = U^{n+1} - U^n$ であるとして, Taylor 展開による近似から定まる差分であり, R とともに $N = 5 \times (N_x - 2) \times (N_y - 2) \times (N_z - 2)$ 長のベクトルである. この行列 K^n は, さらに,

$$K^n = D^n + Y^n + Z^n \tag{2}$$

のように対角, 下三角, 上三角の和で表現され,

$$U^{n+1} = U^n + [1/\omega(2-\omega)]\Delta U^n \tag{3}$$

なる, SSOR(Symmetric Successive Over-Relaxation) が用いられている.

5.2 jacld 関数

高い命令レベル並列度が見いだされたのは, この SSOR 反復 (iteration) における下三角部分を計算する, jacld 関数においてである. この jacld 関数は各メッシュ点で成立する

$$A_{i,j,k}\Delta U_{i,j,k-1} + B_{i,j,k}\Delta U_{i,j-1,k} + C_{i,j,k}\Delta U_{i-1,j,k} + D_{i,j,k}\Delta U_{i,j,k} + \ldots$$

の $A(U_{i,j,k-1})$,$B(U_{i,j-1,k})$,$C(U_{i-1,j,k})$,$D(U_{i,j,k})$ を求めているが, これらは SSOR 反復において, 更新される前の U の値から計算されるためである.

解析に用いているプログラムの実行時間の都合で, $(N_x \times N_y \times N_z) = (5\times5\times5), (6\times6\times6), (7\times7\times7)$ の場合の平均命令レベル並列度 (ILP average) を図 4に示す. 横軸は分岐の解決数である. 図は, たとえば, 80 個先までの基本ブロックを予測できたとして 7×7×7 の場合で約 3700 個の命令を並列実行可能であることを表している.5×5×5, 6×6×6, 7×7×7 それぞれの下の括弧の中は, 実行された, (分岐命令数, 全命令数) である. たとえば, 5×5×5 の場合は, 全部で 3 万 2 千の命令が実行され, そのうちの分岐命令が 65 個であったことを表している.

同じベンチマークプログラムにおいて誤りを計算する error 関数における平均命令レベル並列度 (ILP average) は, 5 × 5 × 5 で分岐の解決数 1500 の場合で 200 以下であった. 実行された命令数は約 3 万で, そのうちの約 3 千が分岐命令であった.

分岐の解決数が 1500 というのは, 分岐先をすべて投入しておくと仮定すると, 2^{1500} という基本ブロックの投入を意味する. 数千という高い並列度が見いだされた jacld 関数の場合でも, 分岐の解決数 20 を得るために 2^{20} 個の基本ブロックを投入することは現実的でない.

6 まとめと今後の課題

SSOR 反復において, 下三角行列を計算する jacld 関数には高い並列度が見いだされたが, 同じプログラムの error 関数では十分な並列度が認められなかった. したがって, データ駆動型演算ユニットに多数の演算セルを配置

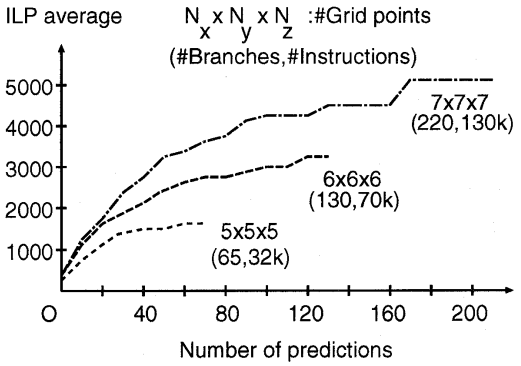

図 4: jacld 関数における命令レベル並列度

しても, 単一のプロセスを対象としていては, 利用効率の悪いことが分かった. 他方で, 多数の演算セルを配置しても, この間の通信 (Packet Exchange Network) の遅延時間が大であることが分かっている. また, 多数の演算セルのどれかが故障を起こしただけで, 対応する処理が中断されることは許されない.

分岐の解決を実現する実際的な方法, 通信の負荷の軽減, 高信頼化の具体的な方法を見いだす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] Kogge P. M.:The Architecture of Pipelined Computers, McGraw-Hill, New York(1981)

[3] Barszcz E.,Fatoohi R.,Venkatkrishnan V. and Weeratunga S.:"Solution of Regular, Sparse Triangular Linear Systems on Vector and Distibuted-Memory Multiprocessors" NASA Techinical Report RNR-93-007(1993)

[4] 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)

[5] 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)

[6] 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)

[7] 高橋隆一：システム開発と設計, サイエンス社 (1996)