擬似ベクトルプロセッサPVP-SWにおけるリストベクトル処理*

2C－4

若林拓、広野哲、山崎浩太、中村宏、朴泰祐、中澤喜三郎†
筑波大学 電子・情報工学系‡

1. はじめに

大規模科学技術計算の分野では、データ領域が非常に大きく、データの時間的局所性が少ない。これらの計算を従来のスカラプロセッサで処理すると、キャッシュミスが頻発して、主記憶アクセスのペナルティが増大し、性能が著しく低下する。

この問題に対処するため、我々は浮動小数点レジスタをスライドウィンドウ化した擬似ベクトルプロセッサ PVP-SW(Pseudo Vector Processor based on Slide-Windowed Registers) を提案し、その有効性を既に確認している [1][2]。

本稿では、リストベクトル演算を効率良く処理するために、擬似ベクトルプロセッサに対し、キャッシュへのプリフェッチ及び汎用レジスタのスライドウィンドウ化の二種類の独立した方式を導入することを提案する。そして、簡単な評価結果とそれらの有効性を示す。

2. 擬似ベクトルプロセッサ PVP-SW

擬似ベクトルプロセッサ PVP-SW[1][2] は、既存のスカラアーキテクチャに対し、浮動小数点レジスタのスライドウィンドウ化、プリロード命令(機能) の追加、主記憶のパイプライン化の三つの機能追加を行なっている。浮動小数点レジスタのスライドウィンドウ構成を図 1に示す。これにより、既存のアーキテクチャとの上位互換性を保ちながら、浮動小数点レジスタを増加でき、高速な擬似ベクトル処理が可能になる。

PVP-SW では以下の 3 ステップの処理がソフトウエアパイプライン的に行なわれる。

- 処理 (演算) に用いるデータを、現ウィンドウから u 個先のウィンドウへロード。
- 既にロードされた値を現ウィンドウで処理 (演算)。
- 現ウィンドウから v 個前のウィンドウの演算結果をストア。

より長い主記憶アクセスレーテンシを隠蔽するには、上の u の値を大きくすればよい。

3. リストベクトル処理への応用

ベクトルの参照に間接指標が必要となるリストベクトル処理においては、リストベクトルのロード、リストベクトルの示すアドレスからのロードが必要となる。少なくとも二回のロードを逐次的に行なわなくてはならないので、性能に与える影響が大きい。そこで、PVP-SW

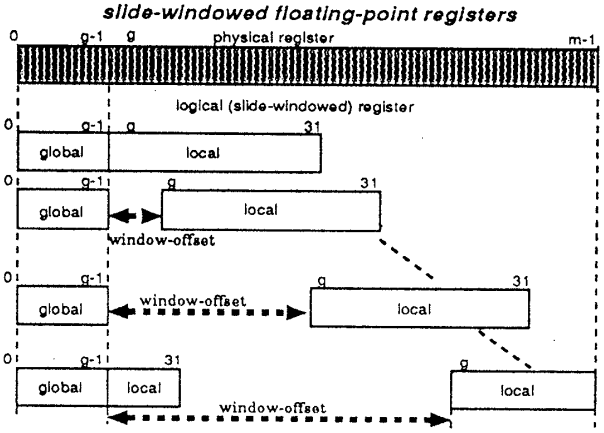

図 1: 浮動小数点レジスタのスライドウィンドウ構成

において、リストベクトル処理も効率良く処理できるように、次の 3.1 節及び 3.2 節で述べるような、二種類の独立した拡張処理方式を提案する。

ここでは、式 (1) の処理を例として説明する。

$$B(L(I)) = A(L(I)) + B(L(I)) \quad \{\forall I, J; I \neq J \Rightarrow L(I) \neq L(J)\} \tag{1}$$

式 (1) の {} 内の条件を満たさない場合、通常のベクトルプロセッサにおいてベクトル化できないので、ここでは対象としない。

3.1 キャッシュプリフェッチ方式の導入

PVP-SW を拡張した PVP-SWPC(PVP-SW with Prefetching Cache) では、リストベクトルの要素となる整数データを、予めデータキャッシュにプリフェッチすることによって、主記憶アクセスレーテンシを隠蔽する。浮動小数点数データのアクセスレーテンシは、浮動小数点レジスタのスライドウィンドウ構成を用いて隠蔽する。PVP-SW からの拡張点は、キャッシュへのプリフェッチ命令の追加のみである。

式 (1) に示す処理は以下のステップで実行される。

- $L(I+u+v)$ をキャッシュにプリフェッチする。
- キャッシュに存在する $L(I+u)$ をロードし、$A(L(I+u))$、$B(L(I+u))$ の値を u 個先のウィンドウへプリロードする。
- 現ウィンドウ内のレジスタにある $A(L(I))$、$B(L(I))$ の値を処理 (演算) する。
- キャッシュに存在する $L(I-w)$ をロードし、w 個前のウィンドウ内のレジスタにある演算結果 $B(L(I-w))$ の値をストアする。

3.2 汎用レジスタのスライドウィンドウ化

PVP-SW を拡張した PVP-SWSW(PVP-SW with Slide-Windowed Genaral Registers) では、浮動小数点

*List Vector Processing on the Pseudo Vector Processor

†Taku Wakbayashi, Akira Hirono, Kohta Yamazaki, Hiroshi Nakamura, Taisuke Boku, Kisaburo Nakazawa
‡Institute of Information Sciences and Electronics, University of Tsukuba

レジスタだけではなく、汎用レジスタも図1に示すようなスライドウィンドウ構成とすることで、整数/浮動小数点数データの双方のアクセスレーテンシを隠蔽する。

PVP-SWSW において、式(1)は次のようなステップで処理される。

- $L(I+u+v)$ を汎用レジスタの $(u+v)$ 個先のウィンドウへプリロードする。
- 汎用レジスタの u 個先のウィンドウ内のレジスタにある $L(I+u)$ をアドレスとして、$A(L(I+u))$、$B(L(I+u))$ の値を浮動小数点レジスタの u 個先のウィンドウへプリロードする。
- 現ウィンドウ内のレジスタにある $A(L(I))$、$B(L(I))$ の値を処理(演算)する。
- 汎用レジスタの w 個前のウィンドウにある $L(I-w)$ をアドレスとして、浮動小数点レジスタの w 個前のウィンドウ内の演算結果 $B(L(I-w))$ の値をストアする。

4. 評価

4.1 評価モデル

次の4つのアーキテクチャを想定して評価を行なう。

<original> PA-RISC1.1 Architecture

<prefetch> original にキャッシュへのプリフェッチ機能のみを追加したアーキテクチャ。

<PVP-SWPC> 浮動小数点レジスタのスライドウィンドウ化(総数64)、キャッシュへのプリフェッチ機能の追加を行なったもの(3.1節)。

<PVP-SWSW> 浮動/汎用レジスタのスライドウィンドウ化を行なったもの(3.2節)。浮動/汎用レジスタ総数は、ともに64とする。

4.2 評価環境

図2の7つのベンチマーク[3]を用い、各モデルに最適なコードをハンドコンパイルにより作成した。

評価における仮定は次の通りである。

- 命令発行は、2命令発行のスーパースカラ方式。
- データキャッシュは、スループット 8byte/MC、ブロックサイズ 32byte とする(MC:マシンサイクル)。
- <prefetch>と<PVP-SWPC>において、プリフェッチされた整数データは、その後、関連する浮動小数点数のロード/演算/ストアの動作が終わるまで、キャッシュから flush out されないものとする。
- 主記憶は<original>以外はパイプライン化されており、スループットは 8byte/MC である。

4.3 評価結果

図2のベンチマークそれぞれの実行をシミュレートし、主記憶アクセスレーテンシによる性能を評価した。

```
L1: B(L(I))=A(L(I)+B(L(I))
L2: B(L(I+2))=A(L(I+1))+B(L(I))
L3: B(L(I+2))=A(L(I+1))*R+B(L(I))
L4: B(L(I+3))=A(L(I+2))*C(L(I+1))+ B(L(I))
L5: R=R+A(L(I+1))*B(L(I))
L6: R=R+(A(L(I+2))*B(L(I+1))+C(L(I)))
L7: C(L(I+7))=((A(L(I+6))+E(L(I+5)))
              +(B(L(I+4))+F(L(I+3)))
              -(C(L(I+2))+D(L(I+1))))*G(L(I))
```

$$\forall I, J; I \neq J \Rightarrow L(I) \neq L(J)$$

図 2: 使用したベンチマーク

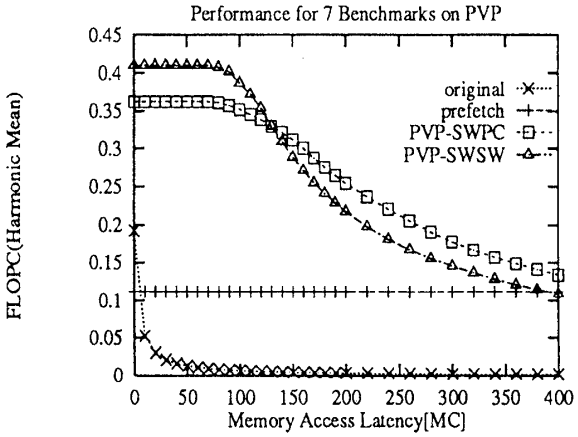

図 3: リストベクトル処理の性能評価

評価指標には FLOPC(FLoating-point Operation Per Cycle)を用いる。図3に7つのベンチマークの調和平均を示す。

<PVP-SWPC>と<PVP-SWSW>は、<original>と比べ、かなり性能が良い。アクセスレーテンシが50MCの点で比較すると、<PVP-SWPC>は約30倍、<PVP-SWSW>は約34倍になっている。また、<prefetch>と比べても、それぞれ、約3.2倍、約3.7倍である。

<PVP-SWPC>と<PVP-SWSW>を比較すると、ピーク性能は前者がやや低い。これは、前者ではプリフェッチ命令を余分に挿入する必要があるためである。レーテンシが70MC 以上の場合に両者とも性能が低下するのは、レジスタ数の不足によりそれ以上のレーテンシを隠蔽できないからである。また、前者ではキャッシュへのプリフェッチにより整数データに対するレーテンシを無限に隠蔽できると仮定しているのに対し、後者では汎用レジスタの不足により整数データに対するレーテンシを隠蔽できない。130MC 付近では両者の性能が逆転するのは、このためである。

5. おわりに

本稿では、擬似ベクトルプロセッサ PVP-SW において、リストベクトル処理を効率良く行なうための二種類の方式を提案した。いずれの方式も、従来のスカラプロセッサに比べ高い性能が得られることが判明し、それらの有効性を確認できた。

謝辞

本研究に関し貴重な御意見をいただいた筑波大学西川博昭助教授並びに中澤研究室諸氏に深く感謝します。

参考文献

[1] 李航他,“スライドウィンドウ方式に基づく擬似ベクトルプロセッサの評価”, 第47回情処全大, 4G-5

[2] H.Nakamura et al., “A Scalar Architecture for Pseudo Vector Processing based on Slide-Windowed Registers”, Proc. of ICS '93, pp.298-307

[3] W.F. Wong, ”Hardware for the Fast Computation of the Elementary Functions”, Dr. Thesis, Univ. of Tsukuba, 1993