リストベクトル処理における擬似ベクトルプロセッサPVP-SWの評価 *

4P－5

廣野哲、森本貴之、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。そのためスカラプロセッサではデータキャッシュが有効に働かず、主記憶アクセスペナルティのために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]。また、データの参照がランダムアクセスになり、データ参照の空間的局所性が少ないためにキャッシュが有効に働かないリストベクトル処理においても、PVP-SWは良い性能を示すことが過去に報告されている [3]。しかし、従来のリストベクトル処理の評価は、リストによって間接参照されるデータ間に依存関係が無いという仮定のもとに行なわれていた。

本研究では、データ間に依存関係が見られるようなリストベクトル処理を、PVP-SWで効果的に行なう手法を示し、計算機シミュレーションによる性能評価よりその有効性を示す。

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、主記憶の擬似パイプライン化の3つの機能追加を行なっている。PVP-SWでは、将来必要となる浮動小数点データを予め浮動小数点レジスタにプリロードすることによって、主記憶アクセスペナルティを隠蔽する。プリロード命令は、キャッシュを介すことなく直接主記憶–浮動小数点レジスタ間でデータ転送を行なう(キャッシュヒット時にはキャッシュからロードする)。

3. PVP-SWにおけるリストベクトル処理

リストベクトル処理では、1つの配列要素を参照するのにリストの参照と、リストを用いた間接参照の2つのメモリアクセスが必要となる。この2つのアクセスは逐次に行なう必要があるため、主記憶アクセスペナルティが性能に与える影響は大きくなる。PVP-SWでは、浮動小数点データの主記憶アクセスレーテンシはプリロード機能によって効果的に隠蔽する。また、通常連続アクセスとなるリストデータのアクセスレーテンシを隠蔽するためには、キャッシュプリフェッチ機能を用いる(PVP-SW with Prefetching Cache[3]、これをPVP-SW(PC)と表す)。

ここでは式(1)を例として、PVP-SWにおけるリストベクトル処理を説明する。

$$A(L(I)) = B(L(I)) + A(L(I)) \qquad (1)$$

まず、リストデータ$L(I)$をキャッシュにプリフェッチしてくる。$L(I)$は通常メモリ上に連続に割り当てられているので、1回のプリフェッチで転送されるブロックサイズ分の全てのデータを有効利用することができる。その後$L(I)$をロードし、アドレス計算の後、プリロード命令を用いて$A(L(I))$をロードする。

しかし、$\exists I, J; I \neq J$において$L(I) = L(J)$が成り立つならば、式(1)はAに関してデータ依存関係が生じる。この場合には、$A(L(I))$のストアが行なわれた後に、$A(L(I+1))$のロードを行なう必要があるので、プリロード命令を用いて$A(L(I+1))$の主記憶アクセスレーテンシを隠蔽することはできない。このような場合は通常のベクトルプロセッサにおいてもベクトル化できず、効率良く処理を行なうことができていない。

そこでPVP-SW(PC)においては、上記のような依存関係の生ずるおそれのあるデータ$A(L(I))$に関しては、キャッシュプリフェッチを利用する。即ち、まず上記と同様に、$L(I)$をロードし、$A(L(I))$のキャッシュへのプリフェッチの後、通常のロード命令を用いて$A(L(I))$をロードする。この方法により、依存関係があるデータに関してはキャッシュの恩恵を受けることができる。また、実際に依存関係が生じていないデータに関しても、このようにすることにより、通常のキャッシュプリフェッチの効果が期待できる。ただし、$A(L(I))$のアクセスはランダムアクセスになるので、プリフェッチされるデータが全て有効に利用されるわけではない。

4. 性能評価

4.1 評価モデル

PVP-SW(PC)の性能を検討するために、次の3つのプロセッサモデルに対して性能比較を行なった。

<Original> PA-RISC 1.1 Architecture[4]。

<Prefetch> <Original>にキャッシュへのプリフェッチ機能(software prefetch)のみを追加したモデル。

<PVP-SW(PC)> 提案する擬似ベクトルプロセッサ。キャッシュプリフェッチ機能も有するPVP-SW。

<Prefetch>、<PVP-SW(PC)>においては、主記憶には理想的なパイプライン・メモリを仮定し、bank-conflictの影響は無いものする。主記憶スループットは8 byte/MC (Machine Cycle)とする。

また、prefetch-bufferは無限にあるものとし、プリフェッチ命令の処理はパイプライン的に処理されるもの

---

*Performance Evaluation of the Pseudo Vector Processor in List Vector Computation

†Akira Hirono, Takayuki Morimoto, Hiroshi Nakamura, Taisuke Boku, Kisaburo Nakazawa

‡Institute of Information Sciences and Electronics, University of Tsukuba

とする。データキャッシュはmulti-portを仮定し、キャッシュのスループットは十分にあるものとした。

その他に、全モデルに対し以下の仮定をおいた。

- 命令発行: 2命令同時発行のスーパースカラ方式。但し、ロード/ストア命令(プリロード/プリフェッチ命令も含む)は同時に1命令しか発行できない。
- 命令キャッシュ: all hitを仮定。
- データキャッシュ: 容量256 Kbyteの1st-levelキャッシュ。ブロックサイズ32 byte。ダイレクトマップ方式、write back、write allocation方式を採用。

4.2 評価ベンチマーク

ベンチマークプログラムとして、Livermore Fortran Kernels (LFK) #14 (1-D PIC) のリストベクトル処理部分、及び、文献[5]より引用した5つのカーネルを採用する。

```
LFK #14: RH(IR(K)   ) = RH(IR(K)   ) + fw - RX(K)
         RH(IR(K)+1) = RH(IR(K) + 1) + RX(K)

   L1: A(L(I)) = B(L(I)) + A(L(I))
   L2: A(L(I+2)) = B(L(I+1)) + A(L(I))
   L3: A(L(I+2)) = B(L(I+1)) * R + A(L(I))
   L4: A(L(I+3)) = B(L(I+2)) * C(L(I+1)) + A(L(I))
   L5: 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))
```

図1: 評価ベンチマーク

演算で用いる配列データはキャッシュの容量を十分越えるサイズ(各512 KB)とし、それに合わせてリストデータも拡張した。リストデータ($IR(K)$ 及び $L(I)$)は、LFK #14のリストベクトル作成アルゴリズムを用いて作成した。リストは32K個の要素を持ち、1〜64Kの範囲の値をとる。リストベクトル$L(I)$を用いた$A(L(I))$の参照においては, 配列Aの64K個要素のうちの約30%がアクセスされ、アクセスされる要素への平均参照回数は約1.68である。

4.3 評価結果と考察

図2に各モデルのリストベクトル処理の評価結果を示す。性能指標にはFLOPC (Floating point Operation per Cycle)を用いた。

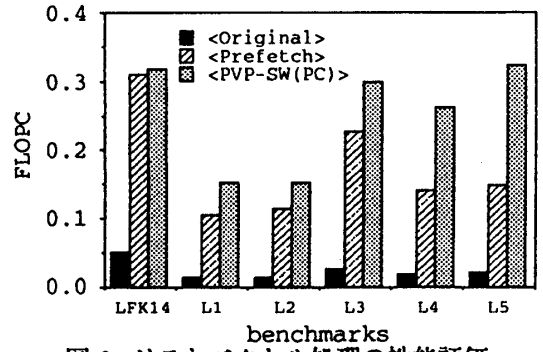

図2: リストベクトル処理の性能評価

<Original>は主記憶アクセスレーテンシ隠蔽の機構を何ら持たないので、他のモデルと比べて著しく性能が悪い。<PVP-SW(PC)>、<Prefetch>はプリロード、キャッシュプリフェッチ機能により良い性能を示しており、各々<Original>の約6.2〜15.7、6.1〜8.6倍の性能を示している。

また、<Prefetch>は<PVP-SW(PC)>よりも性能が低く、最悪の場合で<PVP-SW(PC)>の約45.9 %の性能しか出ていない。

これはプリフェッチ命令実行のオーバーヘッド、及び不要なデータもブロック転送されるために生ずる主記憶スループット不足が主な原因である。LFK #14においては、$RX(K)$のアクセスは連続であり、またRHに関しても$RH(IR(K))$と$RH(IR(K)+1)$の2要素は連続にアクセスされる。このように、データアクセスに空間的局所性が多く存在する場合には、<Prefetch>においても<PVP-SW(PC)>と同程度の性能を達成することができる。しかし、L1〜L3において<Prefetch>では、依存関係が無い$B(L(I))$が、ランダムに参照されるにも関わらずキャッシュにプリフェッチされる。そのため、ブロック転送により不要なデータもプリフェッチせざるを得ず、主記憶スループットが不足して性能が低下する。<PVP-SW(PC)>では、$B(L(I))$に関してはプリロード命令を用いて必要なデータのみを主記憶からレジスタへ直接転送するので、この問題は生じない。L4、L5ではランダムアクセスとなるデータ参照が増えるので、<Prefetch>は<PVP-SW(PC)>に比べてさらに性能が低下する。

プリフェッチ命令はキャッシュに影響を与えるので、L5のようにプリフェッチするデータの種類が多いと、主記憶スループット不足の他に、未使用キャッシュラインの追い出し、ダーティ・ブロックの書き戻し等の悪影響を引き起こし易くなる。<PVP-SW(PC)>においては、ストアされるデータのみに関してプリフェッチ命令を利用し、依存関係の無いデータはプリロード命令で参照するので、上記の問題を最小限に抑えることができるのである。

5. おわりに

本稿ではPVP-SWにおけるリストベクトル処理方式(prefetch to cacheも活用する方式)を提案し、性能評価を行なった。評価結果より、依存関係のあるリストベクトル処理においてもPVP-SW(PC)は有効に働くことが確認できた。

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、アーキテクチャ研究室諸氏に深く感謝します。なお、本研究は一部文部省科学研究費(創成的基礎研究 07NP0401、及び 奨励研究(A) 07780222)によるものである。

参考文献

[1] H.Nakamura, et al., "A Scalar Architecture for Pseudo Vector Processing based on Slide-Windowed Registers", Proc. of ICS '93, pp.298-307, 1993

[2] 位守 弘充 他, "スライドウィンドウ方式による擬似ベクトルプロセッサ", 情報処理学会論文誌, Vol.34, No.12, pp.2612-2623, 1993

[3] H.Nakamura, et al., "Pseudo Vector Processor for High-Speed List Vector Computation with Hiding Memory Access Latency", Proc. of TENCON '94, pp.338-342, 1994

[4] Hewlett-Packard Company, "PA-RISC 1.1 Architecture and Instruction Set Reference Manual", Manual Part Number 09740-90039, 1990

[5] W.F.Wong, et al., "Supercomputer Performance Evaluation using Six Benchmarks", Proc. of TENCON '94, pp.1107-1111, 1994