擬似ベクトルプロセッサによるリストベクトル処理

2L−1

位守弘充　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では、データ領域が非常に大きく、データ参照に時間的局所性が少ないため、プロセッサの高い処理能力に貢献するはずのキャッシュメモリが有効に働かないことが多い[4]。そのためプロセッサの実効性能はキャッシュミス時の主記憶アクセスペナルティにより著しく低下する。

これに対し小容量のベクトルレジスタを備え、マルチスレッド機能等を追加し、またload/store pipelineを強化してスーパースカラパイプラインを切れ目なく有効に働かせようというマイクロベクトルプロセッサアーキテクチャも提案されている[5,6]。

我々は、スーパースカラ処理方式の利点を活用し、従来のスカラプロセッサとの上位互換性を保ちながらこれらの問題を解決した擬似ベクトルプロセッサを提案している[2,3]。

本稿では、擬似ベクトルプロセッサに対しさらに汎用レジスタのレジスタウィンドウ化を行うことにより、上位互換性を保ちながらリストベクトル処理を効率よく扱う手法を提案する。そしてこの手法の簡単な評価を行い、その有効性を示す。

2.擬似ベクトルプロセッサ

擬似ベクトルプロセッサでは、ベクトル計算機と等価な処理を行なうためにスーパースカラプロセッサに以下の機能追加を行なっている。

・浮動小数点レジスタのレジスタウィンドウ化
・主記憶のパイプライン化
・プリロード機能(プリロード命令の導入)

これにより図1のような処理が可能となる。

すなわち1つのループは概念的にはロード/演算/ストアという3つのphaseに分割できる。擬似ベクトルプロセッサでは、この各phaseをレジスタウィンドウを切り替えながらpipeline的に処理することで高速化を計る(phase pipelining[3])。

例えば、i loopの実行には以下の3つのphaseが3つのウィンドウを使って処理される。

・次のi+1 loopで用いられるデータを次ウィンドウ(window 3)のレジスタにロードする命令を発行する(preload phase)。
・i loopの実行を現ウィンドウ(window 2)で処理する(execution phase)。
・i-1 loopで計算されたデータを前ウィンドウ(window 1)から主記憶にストアする命令を発行する(poststore phase)。

繰り返しの終了時に各phaseの用いるレジスタウィンドウを切り替えて処理することにより主記憶のアクセスレーテンシーを隠すことが可能になる。

スーパースカラ方式によりデータ供給と演算を並列に実行することで、演算パイプラインを切れ目なく稼働させることができる。これにより1つのベクトル命令の処理を複数のスカラ命令を垂直マイクロプログラミング的に動作させることで上位互換性を保ちながらレジスタ数の拡張を行ない、ベクトル処理と等価な処理が可能となる(擬似ベクトル処理)。

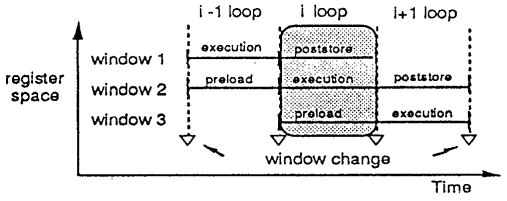

図1 擬似ベクトル処理の原理

3.リストベクトル処理への応用

非常に大きなsparse matrix問題では、通常0でない要素のみを主記憶に格納し、そのアドレスを保持するベクトルを用意している。このベクトルをリストベクトルという。

例として以下のようなリストベクトル演算を取り上げる。

$$A(i) = B(L(i)) + const$$

この演算においてキャッシュメモリが有効に働かない場合には、L(i)、B(L(i))という2回の主記憶へのアクセスが生じることになり、性能低下が著しい。

そこで前章で述べたphase pipelingを拡張し、L(i)も予めプリロードしておくことを考える。そのために、汎用レジスタウィンドウのレジスタウィンドウ化を新たに行なう。

汎用レジスタウィンドウの構造は、浮動小数点レジスタと同一のものとし、現在activeであるウィンドウナンバーも一致しているものとする。

この機能追加によるリストベクトル処理の原理を図2に示す。

今、i loopがwindow jのレジスタ空間を用いて演算を実行するとする。この時i loopは以下の4つのphaseを実行する。ここで( )内に示すのは新たに追加する命令である。

・GR listLoad phase
2つ先の汎用レジスタウィンドウ(g-window j+2)へL(i+2)をロードする。(GR listLoad)
・FR listLoad phase
1つ先のg-window j+1へ既にロードされているL(i+1)を用いて、1つ先の浮動小数点レジスタウィンドウ(f-window j+1)へB(L(i+1))をロードする。(FR listLoad)
・execution phase
現在activeであるf-window jに既にロードされているB(L(i))を用いて実行を行なう。
・FR listStore phase
1つ前のg-window j-1にあるi-1を用いて、計算結果のB((L-1)) + constを1つ前のf-window j-1よりストアする(FR listStore)

各phaseが異なるレジスタ空間を利用しながら1ループを構成し、1ループの終了ごとに汎用/浮動小数点レジスタウィンドウの両方を同時に切り変えていくことで図3に示すphase pipeliningが実現できる。

このようなリストベクトル処理においても、数種類の命令を追加するだけで実現可能であり、また既存のアーキテクチャとの上位互換性を保つことができる。

List Vector Processing on the Pseudo Vector Processor
Hiromitsu IMORI, Hiroshi NAKAMURA, Taisuke BOKU, Kisaburo NAKAZAWA
Institute of Information Science and Electronics, University of Tsukuba

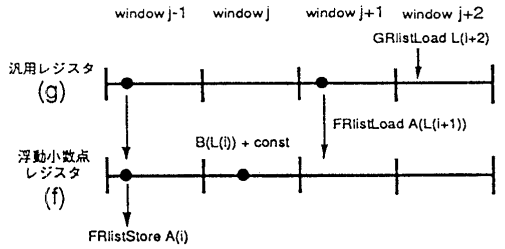

図2 リストベクトル処理の原理

●：レジスタにデータが格納されていることを示す
g - window j は汎用レジスタwindow jを示す

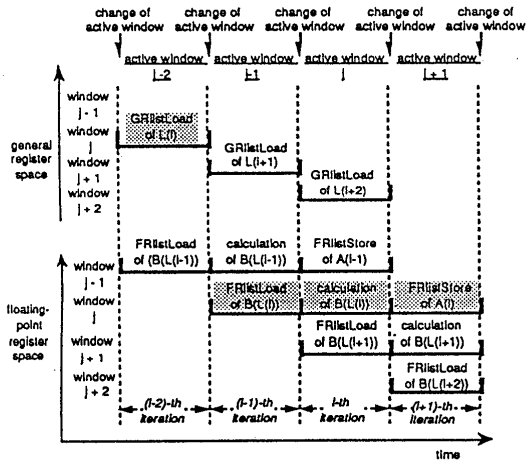

図3 リストベクトル処理のphase pipelining

4. 評価

4.1評価対象モデル

以下の3つのプロセッサモデルに対して評価を行なった。モデルprefetchは提案するモデルpvpとの比較のために新たに仮定したものである。

(basic)　PA-RISC1.1 Architecture

(prefetch)
主記憶のパイプライン化とprefetch命令を機能追加し、データキャッシュへのプリフェッチをコンパイラがサポートしたもの[1]。

(pvp)
擬似ベクトルプロセッサに主記憶のパイプライン化、リストベクトル処理用の汎用/浮動小数点レジスタのレジスタウィンドウ化を行なったもの。

4.2評価環境

object codeは各プロセッサモデルに対し、最適なcodeをハンドコンパイルにより生成した。

評価においては以下のことを仮定した。

・命令発行:2命令発行のスーパースカラ方式
・データキャッシュ:block size 16Bとし、basic、prefetchではline conflictは起こらないとする。また、prefetchにおいてはmulti-port cacheを仮定する。
・主記憶:prefetch、pvpではbank conflictが生じないようにデータは最適に割り当てられているとする。また、主記憶へのアクセスレーテンシーを20 MC (Machine Cycle)とし、load/store pipelineのバンド幅を8 byte/MCと仮定する。
・データ依存関係:先行命令が浮動小数点演算なら5MC以上、先行命令がロード命令ならcache hit時に2MC以上、cache miss時ならメモリアクセスレーテンシー以上、後続命令の発行を遅らせるとする。

4.3評価結果

A(i)=B(L(i)) + constを例として取り上げる。1ループあたりの処理サイクルにより評価を行ない、評価指標としてFLOPC(floating-point operations per cycle)を用いる。図4にload/store pipelineが1本の場合と、2本の場合の評価結果を示す。

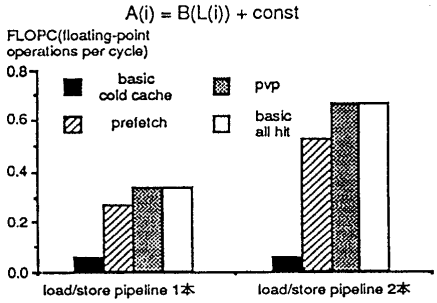

図4 各モデルの評価結果

memory access latency = 20 MC

図4よりpvpはbasicのcold cacheと比較してload/store pipelineが1本の場合で5倍、2本の場合で11倍程度性能が向上している。また、ソフトウェアによりプリフェッチを行なうprefetchと比較しても約1.25倍の性能改善が期待でき、アクセスレーテンシーが20サイクル程度ならデータが全てキャッシュに入っていると仮定しているall hitと全く性能が変わらない。

同様の結果が他のリストベクトル処理についても得られるが、ここでは省略する。

5.おわりに

擬似ベクトルプロセッサに既存のアーキテクチャと上位互換性を保ちながら汎用レジスタのレジスタウィンドウ化も行なうことで、通常のスーパーコンピュータでサポートされている程度のリストベクトル演算が効率よく扱えることを示した。

また、ベンチマークを用いた性能評価によりその有効性を確認した。

参考文献

[1]位守他,"擬似ベクトル処理向きメモリアーキテクチャの一提案",情報処理学会研究報告,ARC-91-8,1991
[2]位守他,"浮動小数点レジスタウィンドウを用いた擬似ベクトル処理",情報処理学会第44回全国大会
[3]中村他,"レジスタウィンドウとスーパースカラ方式による擬似ベクトルプロセッサの提案",並列処理シンポジウム JSPP'92 論文集,1992
[4]M.L.Simmons,et al,"Performance Evaluation of the IBM RISC System/6000:Comparison of an Optimized Scalar Processor with Two Vector Processors" ,Proceedings of Supercomputing'90,pp132-141
[5]村上他,"マイクロベクトルプロセッサアーキテクチャの検討",情報処理学会研究報告,ARC-92-6,1992
[6]橋本他,"「順風」:MSF型ベクトル・プロセッサ・プロトタイプ",並列処理シンポジウム JSPP'92 論文集,1992