非数値計算プログラムにおける命令レベルの並列性について

4E−5

白川 健治　　今井 徹
(株)東芝 総合研究所

1 はじめに

VLIW、superscalerに代表されるアーキテクチャでは、1つのMPUに複数の演算器を持ち、複数の機械語命令を並列に実行することにより高速化を図っている。このようなアーキテクチャは科学技術計算のような、高い並列度を持つプログラムに関しては有効であるが、非数値計算分野においては必ずしも有効ではない。

非数値計算プログラムに存在する並列度を調査するために、既存のRISCプロセッサをもとに、複数の演算器を持つプロセッサを仮定し、そのプロセッサ向きのコンパイラを作成した。そして、このコンパイラの生成するコードを評価することにより、命令レベルでの並列度を求めた。

本稿では、この結果と共に、並列化に影響するアーキテクチャ上の制約を評価し、アーキテクチャの決定に対する示唆を与える。

2 仮想プロセッサ

既存のRISCプロセッサをもとに、その演算器を複数化したもの想定する。完全にスケジュールされたコードを実行するスーパースカラ・プロセッサである。そのアーキテクチャは、

- 整数演算器は2つ
- 浮動小数点演算器はない
- ロード命令は2サイクルである
  パイプライン処理される。
- ブランチは2サイクル
- リソースロッキングはない

とする。

演算器を2つに限定したのは、並列度は低いと予想し、2つの演算器に命令を供給できるかを評価するためである。

加えて、並列化に影響する次の2つの条件について評価した。

メモリとのポート数に関して:

メモリポートが1つの場合と2つの場合を比較する。

2ポートの場合、loadとstoreの同時実行条件として

- <load load> <load store> <store store>
  の同時実行を許す
- <load load> <load store> の同時実行を許す
- <load load> の同時実行を許す

場合を比較する

ブランチ命令のディレイド命令の実行について:

- ブランチ命令の次のサイクルの2命令をディレイド命令とする場合(ディレイド命令は最大3命令となる)
- ブランチ命令の次の命令をディレイド命令とし、その次の命令をアボートする場合

について考察する。

3 最適化(並列化)手法

仮定したプロセッサ用の非数値計算プログラムを対象とするコンパイラおよび並列化オプティマイザを作成した。一般に行なわれるコンパイラの最適化手法に加えて、

- 命令列のスケジューリング
  リソースロッキングがないため完全にスケジュールする必要がある。
- スケジューリングに向いたレジスタアロケーション
  レジスタ割つけによる不必要なリソースロッキングを回避する。
- ディレイド命令の有効利用

を行なう最適化を施した。ソフトウェアパイプライニングの対象となるプログラムは少ないと判断し、パイプライニングは採り入れなかった。

4 評価結果

評価は並列化を行なうコンパイラの生成するコードの静的な命令数をもって行なう。ベースとしたプロセッサと仮想プロセッサでの、

(静的な命令数) * (各命令のサイクル数)

を測定し、その比をもって並列度とする。

Instruction Level Parallelism in Non-numerical Applications
Kenji Sirakawa, Toru Imai
Research & Development Center, Toshiba Corp.

Stanford Benchmark Suite を対象として評1価を行なった。その結果を表に示す。

この結果から次の2点が言える。

1) 非数値計算プログラムにおける命令レベルの並列度は1.5程度である。

2) メモリアクセスの制約については、2命令同時実行の場合には2ポートであることが望ましいが、load命令と、store命令を同時に実行することは必ずしも必要ではない。

1)の要因として、非数値計算プログラムでは、ブランチが多いことが挙げられる。ブランチが多いとスケジューリングの対象となる命令が少なくなるため並列に実行可能な命令が少なくなる。VLIWに見られるトレース・スケジューリングの手法はこれを回避するものであるが、あまり現実的ではない。ソフトウェア・パイプライニングもブランチの多い命令列では有効ではない。更にここで仮定したプロセッサでは、ブランチ命令のディレイド命令は2、3命令であることも並列度を下げる要因となっている。ブランチ実行のメカニズムの工夫が望まれる。

2) load命令とstore命令が同時実行可能でなくとも良いのは、store命令の頻度が小さいからである。load命令とstore命令を同時に実行する場合には、メモリアドレスの競合を考慮しなくてはならないが、同時実行による並列化を期待できないことからこれを禁止しても良いと思われる。

5 おわりに

非数値計算を主体とするプログラムの場合、命令レベルでの並列度は1.5程度である。したがって、ここで対象としたプログラムに関しては演算器は2つで十分であると言える。この結果は、VLIW、superscaler の得意とする数値計算プログラムとここ対象とした非数値計算プログラムをバランス良く実行するプロセッサのアーキテクチャ決定の指針となると期待する。

ここでの評価は、静的な命令数で行なったが、これを実行時の動的な評価と結び付けることが必要である。大規模なプログラムでの並列度の調査では動的な評価は難しい。動的な評価と静的な評価を結び付けることができれば、シミュレーションの効率化となる。静的な評価では把握できない要素である、命令の実行頻度、ブランチの実行頻度および条件ブランチの成否についての統計をもとに静的な評価から動的な評価を予測したい。

表. Stanford Benchmark Suite における並列度

program		bubble	mult	perm	puzzle	quick	tower	tree
1ポート		1.37	1.56	1.39	1.30	1.37	1.35	1.21
2ポート	A	1.43	1.56	1.39	1.41	1.41	1.35	1.23
	B	1.43	1.56	1.42	1.41	1.41	1.35	.1.24
	C	1.43	1.56	1.42	1.41	1.41	1.36	1.26
	D	1.43	1.56	1.42	1.41	1.41	1.36	1.26

A : <load, load> を許す
B : <load, load>, <store, store> を許す
C : <load, load>, <load, store> を許す
D : <load, load>, <load, store>, <store, store> を許す