

並列化コンパイラ向け共通インフラストラクチャの研究

→自動並列化の研究

→→プロセッサレベル並列化の研究

→→→SIMD 並列化

1. 目的

マルチメディアアプリケーションでは、PCM 音声や、画像のビットマップイメージのように、比較的サイズの小さい均質なデータが多数連続しているものを扱うことが多い。そのため、近年のプロセッサでは、マルチメディア処理向けショートベクタ SIMD(Single Instruction Multiple Data stream)拡張命令を持ち、1 命令で複数のデータを扱うようにして高速化を達成しようとしている。このような拡張命令の例としては、以下のようなものがある。

旧 DEC	Motion Video Instruction Extensions for Alpha
MIPS	MIPS 3D
SUN	Visual Instruction Set
Motorola	Velocity Engine (AltiVec)
Intel	MMX, SSE, SSE2, SSE3

しかしながら、これらの拡張命令を既存のコンパイラが活用する場面は非常に限定されており、多くの場合はプログラムの開発者がアセンブラによる記述や、`intrinsic`等の言語拡張の使用といったプロセッサ依存性の方法を用いて、拡張命令を利用するようにプログラムを書く必要がある。

本小テーマでは、SIMD 型のマルチメディア向け拡張命令を活用するようにプログラムを変換する方式を開発し、並列化コンパイラの共通インフラストラクチャの一部として利用可能な形にすることを目標とする。

2. 概要

- 第 1 期は、SIMD 並列化方式の検討や実験と、それを第 1 版の共通インフラストラクチャへ組み込んだプロトタイプを作成を行い、主に共通インフラストラクチャの仕様に対する検討を行った。
- 第 2 期は、第 2 版の共通インフラストラクチャへの SIMD 並列化の組み込みを行い、さらに対象命令セットの展開を図った。

3. 方法

SIMD 並列化系の設計にあたっての目標は次の通りである。

- それぞれの言語のフロントエンドに対して、SIMD 並列化のための言語仕様の拡張

を要求しない。

- SIMD 並列化系では「ループ展開」や「入れ子 IF の引き剥がし」のようなソースコードレベルで実施可能なベクトル化処理を行わず、単純な IF 変換と同形オペレーションのまとめ上げに徹する。

SIMD 型拡張命令を使った最適化は、従来のベクトルプロセッサ向けの最適化技術を応用可能な部分もあるが、ベクトル命令とは異なる特徴がいくつかある。

- SIMD 型拡張命令の並列度はあまり大きくないが、実行コストも低い。
- 整数型データに対しては、飽和演算や差の絶対値、乗算の上位ワード等、特殊な演算を行うことができる。
- データサイズの混合を許している。例えば、
 - MOV 命令や論理演算等では、同一の命令が複数のサイズを扱える(例えば 64 ビット演算は、8 ビットオペランド 8 個とも 16 ビットオペランド 4 個とも解釈可能)
 - 入力と出力のサイズが異なる場合がある
 - 一部のオペランドだけ変更する命令がある
- 多くの SIMD 型拡張命令では、比較演算の結果が全ビットが 0 か 1 の真偽値として得られ、これをマスクとして選択演算を行う。

そのため、次のような 3 つの最適化方法を LIR から LIR への変換系として実装することにした。

- peephole optimization(針穴式最適化)の拡張による SIMD 命令の生成
- 論理演算を利用した if 変換
- 特殊な SIMD 命令へ置き換えるためのパターンマッチ

また、SIMD 命令のコード生成には、TMD 記述による通常命令と同様のコード生成系を用いる。このために、第 1 版の LIR の仕様に対していくつかの拡張を提案し、第 2 版で採用された。

4. 成果

成果を年度ごとにまとめると、以下のようになる。

- 平成 12 年度
 - SIMD 命令の調査

最新のマイクロプロセッサの持つ SIMD 型拡張命令に関する技術文書を入手し、基本的特徴などのほか、単に SIMD 型命令であること以外に特徴を持つような特殊な命令についてもまとめた。(文献[1]参照)

以下に調査を行った SIMD 型拡張命令セットの名称と、その実装されている プロセッサの例を示す。

ベンダー	名称	プロセッサの例
SUN	VIS™ Instruction Set	UltraSPARC-I 以降
Compaq (DEC)	Motion Video Instructions	Alpha 21264
Intel	MMX® Technology	MMX テクノロジ対応 Pentium Processor, Pentium II 以降
Intel	Streaming SIMD Extensions	Pentium III 以降
Intel	Streaming SIMD Extensions 2	Pentium 4
AMD	3DNow!™ Technology	K6-2 以降
AMD	Enhanced 3DNow!™ Technology	Athlon, Duron
Motorola	AltiVec™ Technology	MPC7400, MPC7410 (PowerPC G4)
日立	浮動小数点グラフィック強化命令	SH7750
MIPS	MIPS V ISA Extension (現 MIPS_64 の一部)	R5000
MIPS	MDMX (Mips Digital Media Extension)	MIPS64 5Kc
MIPS	MIPS-3D(TM) ASE (Application Specific Extension)	MIPS64 R20K, MIPS64 20Kc

➤ 命令の意味の詳細な記述

SIMD 型マルチメディア向け拡張命令セットの中には、飽和演算や丸めつき乗算など、複合した演算、サイズを意識した演算を行う命令がある。これらを活用するには、従来のような、高級言語の演算子とそれに対応する命令といった、単純化された命令記述では困難である。また、並列度は低いコストも小さいという SIMD 型命令の特長を生かすためには、通常命令を含めて命令スケジューリングを行う必要があり、通常命令も含めた依存関係を詳細に解析する必要がある。そのため、通常命令も含めて、命令の意味の詳細な記述を行った。具体的には、次のアーキテクチャの命令セットについて、フラグ、オーバーフロー時の処理などを含めた詳細な記述を行った。

- SPARC Version 8
- SPARC Version 9 (64 ビット, VIS Instruction Set を含む)
- Intel の 32 ビットアーキテクチャ非特権命令(32 ビットモードのみ)(Intel MMX/SSE/SSE2, AMD Enhanced 3DNow!を含む)
- PowerPC (AltiVec を含む)

- 平成 13 年度

- SIMD 並列化系プロトタイプの実装

先述の 3 つの最適化方法を実現しつつ、正しくコード生成を行うために、次のようなフェーズ分けによってプロトタイプ実装を行った。

- 論理演算を利用した if 変換
- 基本ブロックを DAG に変換
- DAG を基本操作(SIMD 命令で個々の要素に適用できる操作)に分解
- 基本操作を複数まとめて SIMD 命令を組み立て
- 命令とレジスタの制約(デスティネーションはソースの一方と同じなど)を適用
- SIMD レジスタ割り当て

ここで使われている中間表現は、インフラストラクチャ基盤部で用いている低水準中間表現(LIR)に不足する機能を追加しており、実装も異なるため、SIR(S-expression Intermediate Representation)と呼び、区別する。SIR の設計は、命令の意味をより正確に表現することを目標としており、最適化で扱いやすいことを目指した、基盤部の低水準中間表現とは相違がある。

- 平成 14 年度

- SIMD 並列化系プロトタイプインフラストラクチャへの組み込み

前年度に作成した SIMD 並列化系プロトタイプを第 1 版のインフラストラクチャへ組み込んだ。並列化系は、コンパイラパスにおける実命令割り付け前の LIR レベル最適化の最後の段階に置く。並列化系で用いている SIR からインフラストラクチャ基盤部の LIR へのデータ形式の変換を行って、インフラストラクチャ基盤部と接続を行った。生成されるコードは、基盤部の当時のターゲットであった Sparc/VIS 命令セットを対象とした。

このプロトタイプは、任意のプログラムのコンパイルが可能で正しいコードを生成できるほどの頑健さを有するには至らなかったが、これを使った予備評価を行い、特定のマルチメディアプログラムにおいて最大で 5 倍以上の速度向上を確認することができた。

- 基盤部への提案

SIR と LIR の相違点のうち、多機種への対応や、より進んだ最適化のために、基盤部でも有益であると思われるものとして、以下のような指摘や提案を行った。

- シフト演算子の意味の詳細化
- フラグレジスタの表現
- フラグ用の演算子
- 条件判定演算子
- 変換演算子の詳細化
- 遅延分岐

これらのうちのいくつかは、第 2 版のバックエンドの仕様に取り入れられた。また、SIMD 並列化の対象命令セットアーキテクチャとして Sparc/VIS は特殊であるので、第 2 期では IA32/SSE2 や PowerPC/Altivec のような一般的なメディア処理向け SIMD 拡張命令セットをターゲットとして含めるべきであると提案した。

➤ データサイズ推論

先に述べたとおり、この研究における SIMD 並列化では入力言語に SIMD 並列化のための特別な拡張を要求しないため、多くのプログラミング言語が仕様として有する「汎整数拡張(integral promotion rules)」を施した場合と同等の演算結果を、integral タイプ より狭いデータサイズの演算器やレジスタを用いて処理できるようにするための、プログラムコード解析が必須である。我々は、そのための解析法を開発した。この解析の結果を用いることにより、コード生成系は最適な処理データ幅を選択することが可能となり、処理の並列度の向上、無駄なサイズ変換命令の挿入の省略、より高度な処理内容をもつ命令へのマッチングが可能となり、実行性能の向上を図ることができるようになった。

- 平成 15 年度

➤ プロトタイプ of 第 2 版バックエンドへの組み込み

第 1 期で作成したプロトタイプのターゲットアーキテクチャ非依存部を、第 2 版のバックエンドに組み込んだ。

➤ データサイズ推論の実装

第 1 期で開発したデータサイズ推論のプロトタイプを実装し、第 2 版バックエンドへ組み込んだ。またこの成果を情報処理学会研究会論文誌に発表した[2]。

➤ IA32/SSE2 向け SIMD 最適化系の試験実装

第 2 期のターゲットアーキテクチャとして加わった IA32 の SIMD 拡張命令セットである IA32/SSE2 を対象とした、意図したコードに対して適切な SIMD 命令を生成するレベルの SIMD コード生成系のプロトタイプを実装した。

➤ SIMD ベンチマークの設計

実際のソースコードが公開されているプログラムやベンチマークを対象として、SIMD 並列化が有効である箇所の抽出やコードの評価を行ってきた。その結果として

判ったのは、そのような箇所は実行時間に占める割合が小さいか、現在の SIMD 並列化技術では SIMD 命令効果的な適用が不可能であるかのいずれかである場合がほとんどであった。そこで、以下の要件を満たすことを目指したベンチマークの設計と試験実装を行った。

- コンパイラによる SIMD 命令活用の充実度の確認
- コンパイラによる SIMD 命令の誤用の検出
- ターゲットのプロセッサの SIMD 命令の実行性能の測定
- 同一の処理内容に関して、コンパイラが得てとする記述方法のプログラマへの提示

また、この成果を SACSIS2004 で発表した[3]。

- 平成 16 年度

参考文献

[1] 藤波順久, 阿部正佳: "[SIMD 型拡張命令をもっと使った最適化への道のり](#)", 第 43 回プログラミング・シンポジウム 報告集, 情報処理学会, 平成 14 年 1 月

[2] 鈴木貢, 藤波順久, 福岡岳穂, 渡邊坦, 中田育男: "マルチメディア SIMD 命令活用のためのデータサイズ推論," 情報処理学会 論文誌:プログラミング Vol.45, No.SIG5(PRO21), pp1-11(2004-5)

[3] 室田朋樹, 鈴木貢, 渡邊坦:"SIMD ベンチマークの設計," 先進的計算基盤システムシンポジウム SACSIS 2004 論文集, pp159-160(2004-5)

[4] Mitsugu Suzuki, Nobuhisa Fujinami, Takeaki Fukuoka, Tan Watanabe, Ikuo Nakata: "SIMD Optimization in COINS compiler infrastructure," Proc. 8th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'05), (2005-1) to appear