条件分岐のSIMD化手法のCellへの適用

岩田顕† 漆尾有史‡ 桑原寛明† 國枝義敏†
†立命館大学情報理工学部 ‡立命館大学大学院理工学研究科

1 はじめに

近年，シングルコアCPUの性能向上が，消費電力や発熱量の問題で限界になり難しくなってきた中で，マルチコアCPUが誕生した．その中でも複数種のコアをもつCPUをヘテロジニアスマルチコアという．その一例としてソニー・コンピュータエンタテインメントとIBMと東芝により開発されたCell Broadband Engine [1]（以下，Cellと呼ぶ）がある．Cellは2種類のプロセッサコアをもち，一方はPPE（PowerPC Processor Element）と呼ばれる64ビットPowerPCと命令互換性を有した汎用プロセッサであり，もう一方はSPE（Synergistic Processor Element）と呼ばれる単純なマルチメディア系の処理を目的とするベクトルプロセッサである．CellにはSIMD（Single Instruction Multiple Data）命令セットとしてPPEにはVMX命令，SPEにはSPU_SIMD命令が実装されている．Cellは PPE1基と SPE8基を1チップ上に搭載しており，PPEで処理の制御を行い8基のSPEに並列処理させることで高い演算性能を発揮する．

Cellの性能を高度に引き出すためには処理を並列化するとともに，Cellの特性を理解した上での様々な最適化を行う必要があり，プログラマにかかる負担が大きい．その一例として，SIMD命令セットを用いたデータ並列化がある．SIMD命令セットを用いれば，1命令で16バイト分のデータを同時に処理できるため，大量のデータを高速に処理することが可能となる．しかし，条件分岐やループのような複雑な制御構造をSIMD並列化する場合は命令の置き換えだけでなく制御構造の書き換えも必要となる．

本稿では，文献[2]で提案された条件分岐のSIMD並列化手法をCellに適用し，速度向上率に対しての評価を行い，その有効性を検証する．

2 条件分岐のSIMD並列化

文献[2]で提案された条件分岐のSIMD並列化手法について説明する．

スカラで書かれた条件分岐のプログラムをSIMD並列化するには，制御構造のSIMD並列化を行う必要がある．条件分岐は条件の成否により，then節とelse節のどちらの処理を実行するか決まるので，各要素ごとに異なる処理を実行する必要がある．しかし，SIMD命令は1命令で複数の要素に対して同じ処理は実行できるが，各要素ごとに異なる処理を実行することはできないからである．

制御構造のSIMD並列化を単純にするために次の制約が設けられている．

- SIMD命令で扱えないデータや演算はSIMD並列化対象に含まれない．
- プログラム構造の書き換えによって例外を起こしうる演算は発生しない．
- ループの入口となるブロックは1つである．
- ループの出口となるブロックは1つである．

次に制御のSIMD並列化を行う際に必要な命令が定義されている．

vselect（v_1,v_2,v_{cond}）

ベクトルv_1，v_2の各要素からベクトルv_{cond}の内容が偽の要素ではv_1の要素を取り出し，真の要素ではv_2を取り出して新たなベクトルを生成する演算であると定義する．

条件分岐のSIMD並列化を行う．(1)条件の成否にかかわらずthen節とelse節の両方の処理をこの順で行う．このとき，データ依存関係が新たに生じないように処理内に存在する代入文は全て個別の一時変数への代入に置き換える．(2)最後に上記のvselect()命令を用い，分岐条件の比較結果に従い両方の処理の結果を合成する．

3 Cellへの適用

2章で述べた手法をCellに適用する．Cellでは主に演算を行うのはSPEなので，SPU_SIMD命令セットを用いてSIMD並列化を行う．SPU_SIMD命令セットでは，vselect()命令と同等の機能を持つ命令として

spu_sel(v_1,v_2,v_{cond})

命令がある．この命令は，v_{cond}の各ビットについて1ならばv_1から，0ならばv_2から対応するビットを選択する命令である。

以下，サンプルプログラム（図1）を例に述べる．まず，1章で述べたとおりCellでは16バイト分のデータ

A Simdization of Conditionals for Cell Processor
Akira Iwata†, Yushi Urushio‡, Hiroaki Kuwabara†and Yoshitoshi Kunieda†
†College of Science and Engineering, Ritsumeikan University
‡Graduate School of Science and Engineering, Ritsumeikan University

```
1 for (i = 0 ; i < n ; i++){
2   if (a[i] > 0)
3     a[i] = a[i] + 1;
4   else
5     a[i] = a[i] * (-1);
6 }
```

図 1: サンプルコード

```
1 //voneは全要素が1のベクトルデータ
2 //vminusは全要素が -1のベクトルデータ
3 for (i = 0 ; i < n/4 ; i++){
4   mask1 = spu_cmpgt(vector_a[i], 0 );
5   a1 = spu_add(vector_a[i], vone);
6   a2 = spu_mul(vector_a[i], vminus);
7   vector_a[i] = spu_sel(a1, a2, mask1);
8 }
```

図 2: SIMD 並列化したサンプルコード

を一度に処理できるため，int 型変数 a は 4 要素のベクトルデータ vector_a に置き換える．それとともに，図 1 の for ループの繰り返し判定 i < n は，図 2 で i < n/4 とする．条件分岐 a[i] > 0 の演算結果を分岐条件変数 mask1 に代入する (図 2 の 2 行目)．次に条件が真の場合の処理 (図 1 の 3 行目) を先に変換する．前章で述べたとおり，条件分岐先の代入文は全て一時変数に対しての代入に置き換える必要がある．この例では，ベクトル a の一時変数として a1 を用意する．変換後は図 2 の 3 行目となる．同様に，条件が偽の場合 (図 1 の 5 行目) の処理も変換し，図 2 の 4 行目となる．最後に一時変数 a1 と a2 の合成処理を行う (図 2 の 5 行目)．合成した変数の代入先はベクトル化された元の変数なのでここでは vector_a に書き戻す．以上で SIMD 並列化が完了する．

4 評価・考察

SIMD 並列化する前のスカラのプログラムと SIMD 並列化したプログラムを PLAYSTATION3(OS は Fedora6、ライブラリは SDK2.1) 上で SPE1 基のみで実行し，速度向上率を測定する．図 3 は実行するプログラムの雛形であり，

1. 処理 1，2 が代入文の場合
2. 処理 2 がネストした条件分岐の場合
3. テスト 2 の条件 1 の成否に極端な偏りがある場合

の 3 通りの簡単なテストプログラムを用いて，評価実験を行った．その結果を表 1 に示す．すべてのプログラムが SIMD 並列化して速度向上しているが，1，2，3 の順に速度向上率が低下している．テスト 1 と 2 の比較から，命令数の差が原因と考えられる．条件分岐の SIMD 並列化ではすべての処理を行ってから結果を合

```
1 for (i = 0 ; i < n ; i++){
2   if ( 条件1 )
3     処理1
4   else
5     処理2
6 }
```

図 3: テストプログラムの雛形

表 1: 速度向上率

テスト番号	スカラ (msec)	SIMD(msec)	速度比率
1	3.583	1.146	3.126
2	4.014	1.711	2.345
3	3.156	1.711	1.844

成するため，処理量に偏りがあると実行命令数に大きな差が生まれる．テスト 2 では，処理 2 に条件分岐が入っているため条件の成否による命令数の差が大きくなる．また，合成命令は追加した命令なので条件分岐がネストしていくと合成回数が増えていき命令数が増加する．テスト 2 と 3 の比較から，条件 1 の成否の偏りにより命令数の差が増えたことが原因と考えられる．テスト 3 では，3 の方が条件 1 の成立する割合を多くしている．そのためスカラ実行の時間は短くなっているが，SIMD 並列化した場合の時間は変わらず，その結果速度向上率が低下している．以上の結果から，複雑な条件分岐では SIMD 並列化しても速度向上せずに実行時間が遅くなる可能性があると考えられる．

5 おわりに

本稿では，Cell プログラムでの条件分岐の SIMD 並列化手法について述べた．今回の Cell への適用では，[2] のプログラムの変換手法との変更点は無かった．しかし，条件分岐の構造次第で速度が向上せずに低下する場合もあるため，すべての条件分岐を単純に SIMD 並列化するだけでは有効性が低いと考えられる．今後の課題としては，条件分岐の条件の成否や then 節と else 節の処理量の偏りを解析し，SIMD 並列化が有効な場合を判断することが挙げられる．

参考文献

[1] Sony Computer Entertainment Inc. Cell broadband engine architecture version 1.01. http://cell.scei.co.jp/pdf/CBE_Architecture_v101_j.pdf.

[2] 廣松 悠介, 黒田 久泰, 金田 康正, 複雑な制御構造を持つプログラムの SIMD 命令セットによる最適化, 情報処理学会論文誌,Vol.48, No.SIG 4(PRO 32), pp. 62–72 ,2007.