7-2

# PlayStation3による連立一次方程式解法の効率化

日大生産工(院)　○峯　　和也
日大生産工　　角田　和彦

## 1. はじめに

2006年、SCE(Sony Computer Entertainment)から、特殊なアーキテクチャを持つ独自プロセッサCell/B.E.(Cell Broadband Engine)が搭載された家庭用ゲーム機"PlayStation3"(以下PS3)が発売された事は記憶に新しい。

国内ではCellプロセッサ(以下Cell)のプログラミングコンテスト"Cellチャレ(Cellスピードチャレンジ･Cellチャレンジ)"が3年間に渡り開催され、年内には東芝からCell搭載のテレビが発売される事になっているなど、マルチコアCPUの注目は一層高まっている[1]。

本研究では連立一次方程式の解法プログラムをSIMD演算やDMA転送など[2]の手法を用いPS3向けに書き換え、計算処理の効率化を図ることを目的としている。

## 2. 連立一次方程式

$n$個の未知数$x_1,\dots,x_n$に関する$m$個の線形の連立方程式を連立一次方程式という[3]。

$$\left.\begin{aligned} a_{11}x_1+\cdots+a_{1n}x_n &= b_1 \\ a_{21}x_1+\cdots+a_{2n}x_n &= b_2 \\ \vdots \qquad\qquad & \quad \vdots \\ a_{m1}x_1+\cdots+a_{mn}x_n &= b_m \end{aligned}\right\} \quad (1)$$

係数$a_{ij}$と$b_i$ $(i=1,2,\dots,m\ ,j=1,2,\dots,n)$は定数で、$a_{ij}$を要素とする行列$A$を係数行列という。ここで行列$A$と$\boldsymbol{x},\boldsymbol{b}$を以下のように置くと、式(1)は式(4)のように書ける。

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \quad (2)$$

$$\boldsymbol{x}={}^t[x_1x_2,\dots,x_n],\boldsymbol{b}={}^t[b_1,b_2,\dots,b_m] \quad (3)$$

$$\boldsymbol{Ax}=\boldsymbol{b} \quad (4)$$

## 3. プログラミングについて

式(4)のような連立一次方程式を解くプログラムをCell上で高速化することを考える。

Cellの性能をフル活用するにはPPEとSPEの二種類のコアを協調させる必要があり、プログラミングにも様々な手法やテクニックが必要となってくる。

問題を個々のSPEに分割して転送する際にDMA転送が必要であったり、SIMD演算でデータを扱う際にはそのデータがメモリ上で16バイト境界の番地に揃えて配置されている必要があったり、SPE自体が持っているメモリ領域であるLS(Local Store)の容量が256Kバイトなので転送するデータを最小限のものにしなくてはならないなど非常に多くの問題がある。

以下の項目でDMA転送やSIMD演算について詳しく触れる。

## 4. SIMD演算

SIMD(Single Instruction Multiple Data)演算は、スカラ演算に比べて計算処理を少なくすることができる演算手法である[4]。

例えば、配列の足し算を行う場合スカラ演算では配列のそれぞれの要素に対して演算をするので4回の演算処理が必要だが、SIMD演算を使用すると4回の足し算を1回の演算処理で実行できるという利点がある(図1参照)。

これによって演算回数を短縮でき、全体の処理時間を削減することができる。

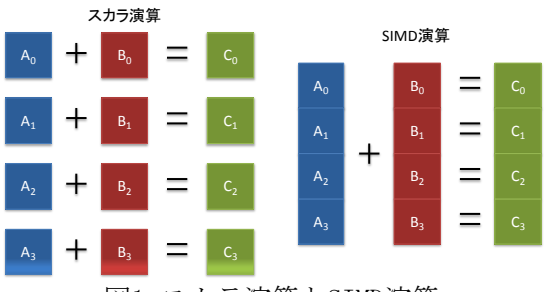


図1. スカラ演算とSIMD演算

---

An Efficient Approach of Simultaneous Linear Equations by Using PlayStation3

Kazuya MINE and Kazuhiko KAKUDA

### 4-1. VMX

VMX(Vector Multimedia Extensions)とはCellのPPEに搭載されているSIMD演算用の命令セットである。

VMXを用いたSIMD演算も可能だが、PPEとSPEを協調させる方が遥かに処理性能を引き出すことができるので通常はSPU Intrinsicsを用いてSIMD演算をする。

### 4-2. SPU Intrinsics

SPU IntrinsicsとはCellのSPEに搭載されているSIMD演算用の命令セットである。

SIMD演算を行う場合、通常はPPEのVMXは使用せずこちらのSPU Intrinsicsを使用する。

### 4-3. DMA転送

DMA(Direct Memory Access)転送とは、プロセッサを介すことなく周辺装置とメモリ間のデータ転送を高速に行うための手段である。

PS3の場合、SPE内部に搭載されているMFC(Memory Flow Controller)によって、SPEプログラムの実行とは独立して行われる。

### 4-4. プログラムの例

例として配列aの数値を配列bに10万回足し合わせるプログラムを用意した(図2参照)。

通常は配列の足し算がfor文の中で4回呼び出されるが、SIMD演算を用いたプログラムではspu_addというベクタの各要素を加算するためのSIMD組み込み関数が1回だけ呼び出されている(図3参照)。

```
#include <stdio.h>
#include <spu_intrinsics.h>

int main()
{

  signed int a[4] __attribute__((aligned(16))) = {1,2,3,4};
  signed int b[4] __attribute__((aligned(16))) = {0};

  int i;

  /* 100000回の加算演算 */
  for (i = 0; i < 100000; i++) {
    b[0] += a[0];
    b[1] += a[1];
    b[2] += a[2];
    b[3] += a[3];

    }
}
```

図2. 通常のプログラム

```
#include <stdio.h>
#include <spu_intrinsics.h>

int main()
{

  signed int a[4] __attribute__((aligned(16))) = {1,2,3,4};
  signed int b[4] __attribute__((aligned(16))) = {0};

  vector signed int* va = (vector signed int* ) a;
  vector signed int* vb = (vector signed int* ) b;

  int i;

  /* 100000回の加算演算 */
  for (i = 0; i < 100000; i++) {
       *vb = spu_add(*va, *vb);

    }
}
```

図3. SIMD演算化プログラム

### 5. 効率化結果

上記のプログラムをPS3上で実行し、演算に要した時間を図4にまとめた。SIMD化によって約2.33倍の処理性能向上が確認できた。

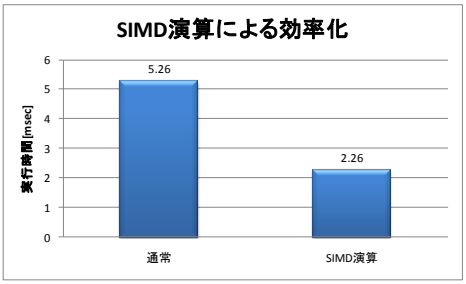


図4. SIMD演算による効率化結果

### 6. おわりに

SIMD演算やDMA転送などの手法を用いて、PS3上で連立一次方程式を効率的に解くためのプログラミングに関する検討を行った。

難易度の高いテクニックがいくつも必要になってくるが、Cellの真価を発揮させるためには必要不可欠なことばかりなので、積極的に活用し今後の研究に生かして行きたい。

### 参考文献


[1] Cell Speed Challenge 2008
"http://www.hpcc.jp/sacsis/2008/cell/"

[2] 塩田紳二, 安田絹子, 國司光宣, 平初, 川井義治, 古坂大地, 青柳信吾, 八重樫剛史, PLAYSTATION3 Linux 完全攻略ガイド, インプレスジャパン(2007)

[3] 水島二郎, 柳瀬眞一郎(共著), 理工学のための数値計算法[第2版], 数理工学社, (2002)

[4] FIXSTARS PLAYSTATION®3
"http://cell.fixstars.com/ps3linux/"