Cell スピードチャレンジ 2008 で用いた高速化手法

須藤 郁弥 † 本田 耕一 † 松田 健護 † 坂内 恒介 † 柴田 悠希 † 小林 隼人 ‡ 石野 明 ‡ 篠原 歩 ‡

† 東北大学工学部 電気情報・物理工学科 ‡ 東北大学大学院 情報科学研究科

Cell スピードチャレンジ 2008 規定課題部門では，連立一次方程式求解の高速化に取りくんだ．本開発では求解アルゴリズムとして LU 分解法を用いた．また，高速化として LU 分解のブロック化，SIMD 演算の利用，ループアンローリング，DMA ダブルバッファリング，帯行列 LU 分解の計算量削減などを行った．結果として，係数行列が密行列で与えられた場合には 7 SPE 利用時に最大 77 Gflops の性能を実現できた．これは，ピーク性能 179.2 Gflops（動作周波数 3.2 GHz）の 43% にあたる．さらに，帯行列 LU 分解では密行列と比較して 2 倍から 5 倍の高速化に成功した．

1 規定課題概要

Cell スピードチャレンジ 2008 の規定課題は連立一次方程式の求解であった．コンテストでは，与えられた係数行列 A（$n \times n$ 行列）および右辺行列 B（$n \times m$ 行列）に対し，$AX = B$ を満たす解行列 X を高速に求めることが課題であった．係数行列 A は行方向に，右辺行列 B は列方向に要素が格納されて与えられ，行列の各要素は float 型であった．また，問題の制約として，入力データは利用可能であるメインメモリの容量 80 MB の半分の 40 MB 以下であり，n は 32 の倍数と定められていた．

2 求解アルゴリズム

我々のプログラムでは求解アルゴリズムとして LU 分解法を採用した．処理の流れは以下の通りである．

1. **係数行列 A の LU 分解**
   LU 分解により，$A = LU$ を満たす下三角行列 L および上三角行列 U を求める．LU 分解は right-looking 法 [1] により行った．
2. **前進代入**
   LU 分解の結果から，前進代入により $LY = B$ となる行列 Y を求める．
3. **後退代入**
   後退代入により $UX = Y$ を解き，解行列 X を得る．

3 LU 分解の高速化

本節では，LU 分解に対して行った主な高速化について述べる．実装にあたっては [2, 3, 4] を参考にした．

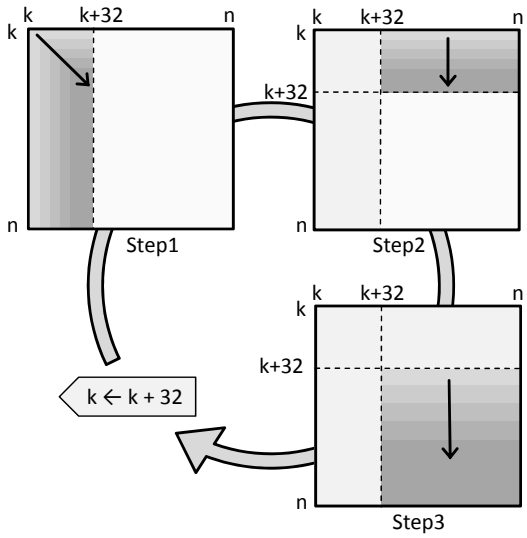

図 1: 縦ブロックガウス法の処理

3.1 LU 分解のブロック化

LU 分解には，縦ブロックガウス法 [1] と呼ばれるブロック化アルゴリズムを用いた．縦ブロックガウス法では，データを列方向に分割し，定めたブロック幅ごとに LU 分解を行う．今回，問題サイズ n が 32 の倍数という制約から，ブロック幅は 32 とした．図 1 は $(n-k) \times (n-k)$ 領域 $(k = 0, 32, ..., n-32)$ を更新する際の処理を示したものである．各ステップで行う処理を次に示す．

- **Step1**
  Step1 では更新する領域の左 32 列を先行的に LU 分解する．計算は行方向にサイクリック分割して各 SPE へ割り当てた．このとき，問題サイズの制約から各 SPE の担当領域はローカルストア (以下 LS)

に格納可能なサイズとなるため，Step1 におけるメインメモリとのデータ転送は Step1 の開始時および終了時のみでよい．また，SPE 内に列データを全て保有しているため，SPE 間の通信のみでピボッティングを実現できる．

- **Step2 および Step3**
  Step2，Step3 では残りの領域を更新する．計算は列方向にブロック分割して各 SPE へ割り当て，一回あたりのデータ転送量がなるべく大きくなるようにした．ここで，Step2 の計算結果は LS に格納可能なサイズとなる．この計算結果は Step3 の計算に用いるため，LS に記憶しておくことでこの分のデータ転送を削減できる．また，ブロック化により更新が 32 段ずつ進むようになるため，データ転送回数は 1/32 にできる．さらに，多段の情報を用いた更新となるため，演算が 2 重ループから 3 重ループになり，後に述べるループアンローリングの効果が大きく表れた．

3.2 SIMD 演算の利用

Cell では 16 バイト長のベクタデータに対する演算を 1 命令で実行する SIMD 演算が利用できる．行列の各要素が float 型（4 バイト）であるため，SIMD 演算により 1 命令で 4 要素の処理を行うことができる．プログラム中の演算に SIMD 演算を用いることで，高速化ができた．

3.3 ループアンローリング

プログラム中でループアンローリングを行うことで，高速化の効果が見られた．さらに，本開発ではループアンローリングを行った際に，場合に応じて以下の操作を行った．

1. **SIMD 演算の入れ子化**
   ループ中の SIMD 演算で更新されるベクタが一定である場合，SIMD 演算を入れ子にした．この操作により，多くの問題で解精度の向上が見られた．ここで，図 2 中の a, b, c はメモリ上のベクタデータとする．

2. **一時変数の利用**
   配列 a に対し，*(vector float*)& a[k] と記述することで，配列中の連続した 4 要素をベクタとして扱うことができる．この場合，図 3 に示すように

```
for(i=0;i<n;i+=2){
  c = spu_nmsub(a_{i+1}, b_{i+1}, c);
  c = spu_nmsub(a_i, b_i, c);
}
```

⇩

```
for(i=0;i<n;i+=2){
  c = spu_nmsub(a_i, b_i,
      spu_nmsub(a_{i+1}, b_{i+1}, c));
}
```

図 2: SIMD 演算の入れ子化

```
*(vector float*)&a[k] =
  spu_nmsub(a_1, b_1,
      *(vector float*)&a[k]);
*(vector float*)&a[k+4] =
  spu_nmsub(a_2, b_2,
      *(vector float*)&a[k+4]);
```

⇩

```
vector float t_1 , t_2;
t_1 = spu_nmsub(a_1, b_1,
      *(vector float*)&a[k]);
t_2 = spu_nmsub(a_2, b_2,
      *(vector float*)&a[k+4]);
*(vector float*)&a[k]   = t_1;
*(vector float*)&a[k+4] = t_2;
```

図 3: vector float 型の一時変数の利用

vector float 型の一時変数に SIMD 演算の結果を格納してから，元のベクタに戻すようにすることで，展開分の演算が高速化できた．

3.4 DMA ダブルバッファリング

DMA 転送が必要な部分では DMA ダブルバッファリングを用い，演算中に次に必要な要素の読み込みや結果の書き込みを行うことで通信時間を隠蔽した．

3.5 帯行列 LU 分解の計算量削減

我々は帯幅の小さい帯行列について，計算する必要のない領域が現れることに注目し計算量の削減を図った．

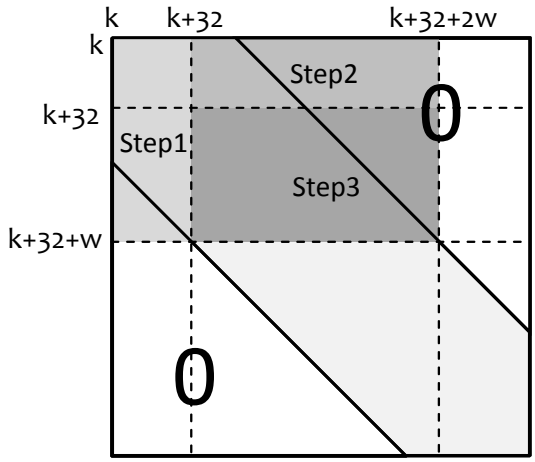

図 4: 帯行列 LU 分解の計算量削減

帯行列の性質を利用した計算量削減を行うため，LU 分解を開始する前に次の手順で行列の半帯幅 w を調べた．半帯幅 w は各対角要素の上下左右に並ぶ非零の要素数が w 個以下であることを保証する．

1. $w = 0$ で初期化する．
2. SPE の id を用いて $i = id$ として，第 i 行について，行の左端および右端から数えて最初に非零の要素が現れる位置をそれぞれ調べ，対角要素の左に並ぶ非零の要素数 w_{Li} および右に並ぶ非零の要素数 w_{Ri} を求める．
3. 各 SPE のもつ $\max(w_{Li}, w_{Ri})$ のうち最大のものを w とする．閾値 σ に対し，$w > \sigma$ ならば，密行列として LU 分解を行う．
4. 全ての行を調べ終わったら半帯幅として w を出力する．そうでないならば $i = i + 7$（7 SPE 利用時）として，2 へ戻る．

w を用いると，LU 分解の各ステップで計算が必要な領域は図 4 のように表すことができる．Step 1 において，ピボット交換は第 k 行から第 $(k + 32 + w)$ 行の間で行われるため，ピボット交換後も第 $(k + 32 + w)$ 行以降の要素はすべて 0 である．また，範囲中の最下行とのピボット交換が起こった場合でも，Step 2 の領域で非零の要素は第 $(k + 32 + 2w)$ 列までに収まる．さらに，Step 3 の計算では Step 1 および Step 2 の領域を参照するため，それぞれで考慮した行・列についてのみ計算すれば十分である．

以上のように帯行列 LU 分解の計算量を削減できる．

```
for(j=0;j<i;j++)
  b[i] -= a[i][j]*b[j];
```

⇩

```
temp = b[i];
for(j=0;j<i;j++)
  temp -= a[i][j]*b[j];
b[i] = temp
```

図 5: スカラ変数の利用

4 前進代入・後退代入の高速化

前進代入・後退代入においても SIMD 演算や DMA ダブルバッファリングなどの基本的な高速化を行った．本節では，特に前進代入・後退代入に対して効果が見られた高速化手法について述べる．

4.1 解ベクトルの複数本処理

係数行列の読み込み回数を削減するため，各 SPE で一度に 3 本ずつ解ベクトルを読み込んで処理を行うようにした．このとき，生じるループに対して，LU 分解と同様のループアンローリングを行い，さらに高速化を図った．

4.2 スカラ変数への格納

前進代入および後退代入では，配列の 1 要素に繰り返しアクセスを行った．この際，図 5 のように配列の要素をスカラ変数に格納しそのスカラ変数に対してアクセスを行うことで，処理時間を削減する事ができた．

4.3 右辺行列の行交換

右辺行列は要素が列方向に配置されているため，行交換を行いたい要素を取り出すためにはその周辺の要素ごと取り出す必要があり，行交換の効率は大変悪い．そのため，係数行列で行ったピボット交換の操作を保管し，前進代入・後退代入の処理時に一括して解行列のピボット交換を行った．

表 1: 決勝ラウンド結果

問題	n	m	時間	順位	行列の性質
1	768	1	14	3 位	非対称密行列
2	224	896	8	3 位	非対称密行列
3	2,688	1	186	2 位	非対称密行列
4	256	20,000	120	4 位	非対称密行列
5	1,792	1	74	3 位	対称密行列
6	32	131,072	64	4 位	対称密行列
7	2,720	1	FAILED		三重対角行列
8	2,240	224	113	2 位	帯行列
9	2,656	100	212	3 位	対角優位行列
10	3,200	7	62	1 位	ブロック行列

時間は msec

5 評価結果

以上の高速化を行ったプログラムを用いて，$n = 3200$, $m = 1$ として係数行列が密行列である場合の求解を行った．この場合の性能は 77 Gflops であり，動作周波数 3.2 GHz，7 SPE 利用時のピーク性能 179.2 Gflops の 43% の性能を実現できた．また，Cell スピードチャレンジ 2008 規定課題部門本選の結果は表 1 の通りであった．問題 7 では，[5] を参考に実装したアルゴリズムを用いたところ誤差が大きくなってしまい正しい解を出力できなかった[1]が，それ以外の全ての問題で 4 位以内に入ることができた．さらに，係数行列のサイズ n が大きい問題（問題 3）および，係数行列が帯行列で与えられた問題（問題 8，10）では，いずれも 2 位以内となった．

問題 8，問題 10 に対し，計算量削減を行わなかった場合の求解時間はそれぞれ 192 msec，284 msec であった．よって，計算量削減により，求解時間は問題 8 ではおよそ半分に，問題 10 では 1/5 にまで短縮できたことが分かる．また，提出したプログラムでは先に述べた領域よりも余分に計算を行っていたため，実際には求解時間をさらに短縮できる．具体的には，問題 8 で 103 msec，問題 10 で 42 msec に短縮できることが確認できた．ちなみに，問題 7（三重対角行列）に対し同方法を用いた場合の求解時間は 28 msec であり，さらにこの場合は十分な解精度が得られた．計算量削減を行わない場合の求解時間は 186 msec であったため，この場合も計算量を大幅に削減できていることが分かる．

本開発では，LU 分解の Step 2，Step 3 において各 SPE の担当領域をブロック分割により決めることで，一度になるべく大きいサイズを読み込むようにし，データ転送の効率化を図った．このため，問題 3 のように n が大きい場合には十分な効果が得られたが，n が小さい問題では分割により各 SPE の担当領域も小さくなるために効率が落ちてしまったと考えられる．対策としては，行方向にもブロック幅を考慮して 32×32 のブロック単位で処理するようにし，係数行列のデータレイアウト変換を行うことが考えられる．

[1] 練習問題，予選問題の三重対角行列に対しては正しい解を出力した

6 まとめ

LU 分解法による連立一次方程式求解の高速化を行った．LU 分解のブロック化などによりデータ転送の回数を削減した上で DMA ダブルバッファリングを行い，また演算に SIMD 命令やループアンローリングを用いることで，様々な問題に対して高速に解を出力するプログラムを作成することができた．また，係数行列が帯行列であった場合には効率よく計算量の削減を行うことができた．

参考文献

[1] 規定課題問題作成 WG．マルチコアプログラミングコンテスト Cell スピードチャレンジ 2008 規定課題参考資料，2008.

[2] 東芝セミコンダクター社 Cell オンラインチュートリアル

[3] Sony Computer Entertainment Inc. SPU C/C++ 言語拡張 Version 2.3, 2006.

[4] FIXSTARS Cell プログラミングチュートリアル

[5] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery．丹慶勝市ほか訳．Numerical Recipes in C．技術評論社，pp．62-63，1993.