Cell/B.E. での連立一次方程式の求解の高速化

松本 和也
会津大学大学院 コンピュータ理工学研究科

Cell スピードチャレンジ 2008 の規定課題部門では，連立一次方程式の求解という課題が出題された．本稿では，その規定課題部門に向けて開発を行い，結果，部門 3 位と相なったプログラムについての解説を行う．開発したプログラムでは，ブロック幅 32 の right-looking 法による LU 分解のブロック化アルゴリズムを用いた．求める解ベクトルの本数が多い問題においては，解ベクトルを複数本まとめて求めることで，他の参加チームと比べて高い性能を発揮した．しかし，コンテストの実行環境では，プログラムの非対称密行列における最大性能は，70.99 Gflop/s と，理論ピーク性能の 40 パーセントの性能を発揮するにとどまった．

1 はじめに

Cell Broadband Engine (Cell/B.E.) は，ヘテロジニアスなマルチコアプロセッサである．Cell/B.E. 上で高速に動作するプログラムを書くのためには，そのアーキテクチャの理解と，並列プログラミングの知識が必要とされる．

Cell スピードチャレンジ 2008[1] の規定課題部門においては，Cell/B.E. を搭載した Cell リファレンスセット [2] 上で，与えられた連立一次方程式の解をどれだけ速く求められるか，という課題が出題された．

規定課題の概要を記述する．連立一次方程式 $AX = B$ が与えられたとき，その解行列 X を求めよ．ここで，係数行列 A は $n \times n$ 行列であり，右辺行列 B と解行列 X は，それぞれ $n \times m$ 行列である．また，係数行列のサイズ n は 32 の倍数に限定され，計算に使える SPE の数は 7 基で固定であった．

本稿では，その規定課題部門に向けて開発を行い，3 位入賞を果たしたプログラムについての解説をする．

2 アルゴリズム

この節では，開発した連立一次方程式の解を求めるプログラムで用いたアルゴリズムについて説明する．

2.1 LU 分解

コンテストでは，LU 分解により連立一次方程式を解く方法を用いた．LU 分解による解法は，分解を行うのに $O(n^3)$ の時間計算量を必要とするが，最初に LU 分解してしまえば，後の計算量が $O(n^2)$ となる．ゆえに，同一の係数行列を用いて複数の連立一次方程式を解く場合には，特に有効な方法である．

実際には文献 [3] で述べられている，right-looking 法による LU 分解のブロック化アルゴリズムを基にしたアルゴリズムを用いて，行列 A を下三角行列 L と上三角行列 U に分解する．このアルゴリズムはブロック単位で LU 分解を行う．課題では問題のサイズが 32 の倍数に限られているので，ブロック幅を 32 とするブロック化アルゴリズムを実装した．図 1 は，その実装したブロック化アルゴリズムの，K 回目の繰り返しでの行列の更新イメージを表す．大まかな分解手順は以下の通りである．

1. C を L_0, L_1, U_0 に分解する．
2. $L_0U_1 = D$ を前進代入法で解くことにより，U_1 を得る．
3. L_1, U_1 の行列積を用いて E を更新し，$E' = E - L_1U_1$ を得る．
4. 1-3 を $N - 1$ 回 ($N = n/b$, b はブロック幅) を繰り返した後に，残りの $b \times b$ 行列を分解すれば，係数行列を LU 分解した行列が求まる．

このブロック化アルゴリズムは，行列積演算が主要な演算となるので，高い性能が発揮できると考え，実装するアルゴリズムとして採用した．また，課題では，非対称密行列，対称密行列，帯行列などのように，様々な性質を持つ行列が係数行列として与えられたが，行列の種類に応じて処理内容を変更させることは行わなかった．

2.1.1 並列化

このブロック化アルゴリズムはブロック単位で各 SPE に処理を割り振ることができる．手順 1 では行方向にサ

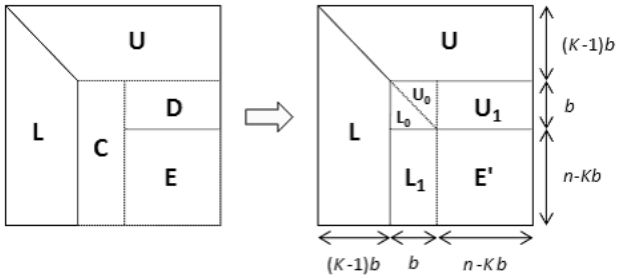

図 1: ブロック化 LU 分解アルゴリズムのの K 回目の繰り返しにおける更新イメージ

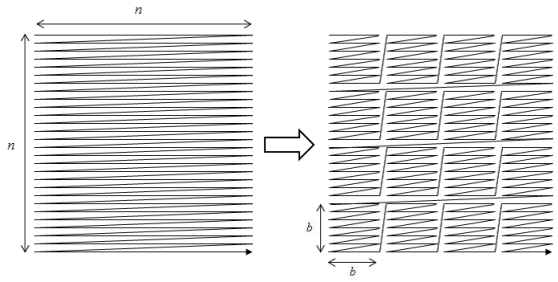

図 2: データレイアウトの変換

イクリック分割し，手順 2,3 では列方向にサイクリック分割して，各 SPE に処理を割り振り，並列的に処理を行わせた．

2.1.2 軸選択

LU 分解を行う場合には，解精度を確保するために，軸選択が必要である．プログラムでは，部分軸選択を行っている．軸選択後には行交換が必要であるが，実装では，使用しているすべての SPE を用いて DMA 転送を行い，実際に交換させている．また，右辺行列 B の行交換については，交換したインデックスの情報を保存しておき，LU 分解後の解行列 X を求める段階で，解ベクトルのデータを SPE にロードしてきた直後に行うようにした．

2.1.3 前進代入と後退代入

LU 分解後に前進代入，後退代入により解を求めるにあたっても，ブロック幅 32 のブロック化アルゴリズムを用いた．

前進/後退代入は，解ベクトル単位で各 SPE に処理を割り振ることができ，その処理同士に依存関係がなく，並列度が高い．また，実装では，1 回の LU 行列の転送で，8 本を上限として，できる限り多くの本数の解ベクトルをまとめて求めるようにした．これにより，求める解ベクトルの本数 m が多い場合には，DMA 転送量を削減できた．

3 最適化

本節では，開発したプログラムで用いた最適化技法について述べる．プログラムの開発にあたっては, 文献 [4, 5] 等を参考にした．

3.1 データレイアウトの変換

係数行列 A のデータは，行優先 (row-major) で並んでいるが，プログラムは LU 分解前に，メインメモリの空き領域を利用して，その並び方を図 2 のように，ブロックデータレイアウト [6] と呼ばれる並び方に変換している．この変換により，それぞれの 32×32 のブロックのデータが連続したメモリ領域に整列される．結果，レイアウト変換を行わずに DMA リストを用いてブロックデータの転送を行うよりも DMA 転送が効率化され，性能が向上した．予選ラウンドから決勝ラウンドにかけての性能向上は，このデータレイアウトの変換を行ったことに依るものが大きく，問題によっては倍以上の性能向上がみられた．

3.2 ループアンローリング

計算が集中している部分においては，できる限りのループ処理をアンロールし，ループカウンタの更新等にかかるオーバヘッド等を軽減させることで性能向上を図った．

ループアンローリングを行うことは，他の最適化技法である命令の SIMD 化，パイプライン最適化などの効果を高める上でも重要である．

3.3 DMA ダブルバッファリング

SPE ではダブルバッファリングと呼ばれる，複数のバッファを用意し，DMA 転送と演算処理を平行して行わせる技法を用いることで，その転送時間の遅延を隠蔽することができる．プログラムでは，図 1 の D, E を更新する部分と，前進/後退代入において，この技法を用いた．

4 性能評価

この節では，まず，LU 分解を用いて連立一次方程式を解く場合に必要な演算量について考察する．次に，提出したプログラムの，規定課題部門の決勝ラウンドでの結果を示す．その後，プログラムの LU 分解の性能と，前進/後退代入の性能についての評価結果を述べる．そして，複数本の解ベクトルを，一回の LU 行列の転送でまとめて求めることの有効性を示す．

4.1 演算量

LU 分解を行うのに必要な演算量は，

$$\frac{2}{3}n^3 + O(n^2) \tag{1}$$

であり，前進代入と後退代入に必要な演算量は，合わせて

$$(2n^2 + O(n))m \tag{2}$$

である．

式 (1,2) から，問題のサイズ n が解ベクトルの本数 m に比べて極端に大きい場合には，LU 分解の演算量の方が前進/後退代入の演算量より多くなり，LU 分解の性能が，問題の処理時間を決めることがわかる．一方，m が n より大きい場合には，前進/後退代入の演算量がより多くなり，前進/後退代入の性能により，問題を処理する速さが決定されることがわかる．

4.2 決勝ラウンド結果

表 1 に規定課題部門の決勝ラウンドにおける，提出したプログラムの結果を示す．解ベクトルの本数 m が多い問題の場合は，問題 2,4 では 1 位と，良い結果を残せた．他チームと比べて，プログラムの前進/後退代入の性能が高いことがわかる．ただし，係数行列のサイズ n が 64 以下の場合は，処理方法が違うために，問題 6 のような場合には，それほど高い性能が出なかった．

4.3 評価結果

図 3(a) は非対称密行列において，問題のサイズ n が 3232 で，解ベクトルの数 m が 1 本の場合のプログラムの性能を表す．この性能測定はプログラムの LU 分解の性能を調べるために行った．SPE を 3 個用いて計算させた場合には，58.84 Gflop/s と，3 個の場合の理論ピーク性能 (76.8 Gflop/s) の 77 パーセントの性能を発揮している．また，SPE の数が 4 個の場合も 67.66 Gflop/s と，確実な性能向上がみられる．しかし，SPE の数が 5 個以上の場合になると，ほとんど性能向上がみられない．特に SPE の数が 7 個の場合には，70.99 Gflop/s と，理論ピーク性能 (179.2 Gflop/s) の 40 パーセントの性能しか発揮できていない．この性能の上げ止まりは，DMA 転送量が，Cell/B.E. のコアエレメント同士を繋ぐ EIB の帯域幅を越えてしまい，演算時間よりも DMA 転送時間に，より多くの時間を費やしているためと思われる．

図 3(c) は，非対称密行列において問題のサイズ n が 512 で，解ベクトルの数 m が 4096 本の場合のプログラムの性能である．この性能測定は前進/後退代入の性能を調べるために行った．図 3(a) の結果と違い，SPE の数の増加にほぼ比例して性能が向上している．

図 4 に，一回の LU 行列の転送で，解ベクトルを複数本まとめて求めることの有効性を確かめるために，まとめて求める数を変化させて，性能を測定した結果を示す．図 4 からもわかるように，解ベクトルの本数が 4096 と多い場合においては，まとめて 5 本以上の解ベクトルを求めた場合には，1 本ずつ求める場合と比べて，3.6 倍程度の性能向上を達成できる．

5 まとめ

開発したプログラムでは，ブロック幅 32 の right-looking 法による LU 分解のブロック化アルゴリズムを用いた．データレイアウトの変換，ループアンローリング，DMA ダブルバッファリング等の最適化技法を用いて，プログラムを高速化した．

表 1: 決勝ラウンド結果 (時間は sec 単位)

問題	n	m	時間	順位	行列の性質
1	768	1	0.012	2	非対称密行列
2	224	896	0.006	1	非対称密行列
3	2688	1	0.195	3	非対称密行列
4	256	20000	0.075	1	非対称密行列
5	1792	1	0.070	2	対称密行列
6	32	131072	0.057	3	対称密行列
7	2720	1	0.181	5	三重対角行列
8	2240	224	FAILED		帯行列
9	2656	100	FAILED		正定値行列
10	3200	7	0.310	2	ブロック行列

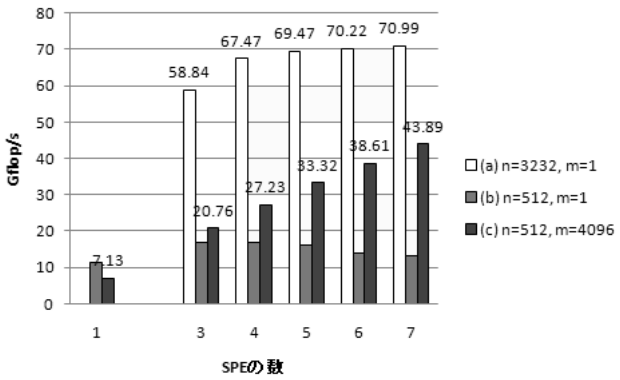

図 3: 非対称密行列におけるプログラムの性能; ただし (a) の SPE の数が 1 個の場合は同条件で実行不可のためデータなし．また SPE が 2 個の場合はプログラムの不備により実行不可のためデータなし．

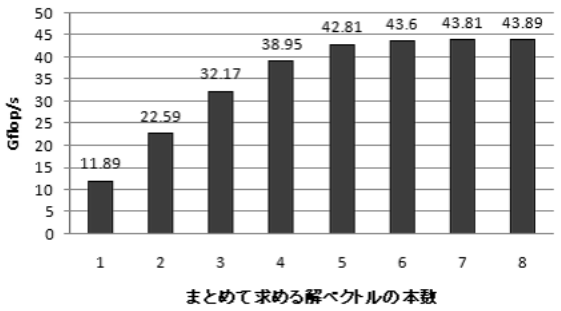

図 4: まとめて求める解ベクトルの本数の変化に伴う性能の変化; 測定に用いた SPE の数は 7 個で，他の条件は図 3(c) と同じ．

LU 分解後，前進/後退代入により解ベクトル行列を求める段階では，解ベクトルを複数本まとめて求めることにより，DMA 転送量の削減を図った．結果，解ベクトルの本数が多い問題においては，他の参加チームと比較して，高い性能を発揮した．しかし一方で，プログラムの最大性能は，理論ピーク性能の 40 パーセントである，70.99 Gflop/s の性能を発揮するにとどまり，最大性能が問われる問題では，それほど良い結果が得られなかった．

プログラムにはまだ性能を改善できる部分が残されている．例えば，今回の課題においては，係数行列のサイズが 32 の倍数に限定されるという条件から，ブロック幅を 32 としたが，代わりにブロック幅として 64 のように更に大きな値を用いれば，DMA 転送が効率化される等の理由から，性能向上が見込めるだろう．また，決勝ラウンドにおいては，解精度が問題作成ワーキンググループの作成したプログラムの 3 倍以下に抑えられず，”FAILED” となってしまった問題が 2 問存在した．処理する問題によっては，極端に解精度が低下するようである．原因を探り，改善を行う必要がある．

参考文献

[1] Cell スピードチャレンジ 2008 実行委員会. Cell スピードチャレンジ 2008. http://www.hpcc.jp/sacsis/2008/cell/, 2008.

[2] 上村剛, 大溝孝, 粟津浩一. Cell リファレンスセット概要. 東芝レビュー, Vol. 61, No. 6, pp. 25–29, 2006.

[3] Jack J. Dongarra and David W. Walker. Software libraries for linear algebra computations on high performance computers. *SIAM Review*, Vol. 37, No. 2, pp. 151–180, 1995.

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

[5] Sony Computer Entertainment Inc. SPU アセンブリ言語の仕様 Version 1.4, 2006.

[6] Neungsoo Park, Bo Hong, and Viktor K. Prasanna. Tiling, block data layout, and memory hierarchy performance. *IEEE Trans. Parallel and Dist. Syst.*, Vol. 14, No. 7, pp. 640–654, 2003.