Cell BE 向け連立一次方程式の求解プログラムの開発

田原 慎也
東京工業大学 大学院理工学研究科

Cell Broadband Engine 向けとして，連立一次方程式を高速に解くプログラムの開発に取りくんだ．Cell Broadband Engine 特有のアーキテクチャは高い理論ピーク性能を誇るが，性能を十分に引き出すためにはそのアーキテクチャを熟知した上で様々なテクニックを駆使しなければならない．今回開発したプログラムは，縦ブロック分割 LU 分解を用いて係数行列を分解し，前進消去と後退消去により解を求めるアルゴリズムを用いた．係数行列の LU 分解では，32×32 のブロックごとに DMA 転送を行い処理を進めることで，データの利用効率を高め，種々の高速化手法を適用し易くしている．高速化手法としては，SIMD 化，DMA 転送時間の隠蔽，ループアンローリングなどを利用した．

1 はじめに

Cell Broadband Engine(以下 Cell) は，1 つの汎用的なプロセッサコアと 8 つのマルチメディア処理に適したシンプルなプロセッサコアを持つ，ヘテロジニアスなマルチコア・プロセッサである．その特徴的なアーキテクチャは，非常に高い理論ピーク性能を誇るが，性能を十分に引き出すことは難しい．今回，Cell Speed Challenge 2008 の規定課題部門において，Cell 向けにプログラム開発を行った．その開発過程や性能を引き出すための高速化手法などをここに述べる．

1.1 規定課題

まず，規定課題について説明する．今回出題された課題は,「連立一次方程式の求解」である．次のように，係数行列 A，右辺ベクトル B が与えられたときに解ベクトル X を求めるプログラムを作成する．

$$AX = B \tag{1}$$

解ベクトルを求めるまでの計算時間を測定し，それによりプログラム性能を決定する．競技チームは，このプログラム性能を競うことになる．

1.2 データ形式と初期配置

係数行列 A は，$N \times N$ 行列で各要素は $float$ 型である．右辺ベクトル B と解ベクトル X は，$M \times N$ 行列である．ここで，N は 32 の倍数という制約が設けられている．これらの行列は，メインメモリ上に A，X，B の順で確保されており，さらにその合計サイズと同等の空き領域が与えられている．これらのデータの総容量は $80MB$ 以下である．その先頭アドレス（ポインタ buf）や行列のサイズ N や M などの情報は，プログラム実行時に SPE に渡され利用することができる．

2 設計

連立一次方程式の解法は，直接法と反復法の 2 つに大きく分かれる．直接法は，所定の回数の演算を実行すれば必ず求解できる解法で，ガウス消去法や LU 分解を用いた解法が有名である．反復法は，解に適当な初期値を与えて正しい解へと収束させる方法で，直接法より高速なことが多いが必ず求解できるとは限らない．代表的な解法としては，ヤコビ法やガウス・ザイデル法などがある．

今回作成したプログラムは，直接法である LU 分解を用いて連立一次方程式を解くプログラムである．どんな連立一次方程式に対しても求解可能で，多くの並列化手法が研究されていることからこの解法を選択した．また解ベクトルが複数ある場合にも，LU 分解は一度だけで済むために有利である．

LU 分解を用いた連立一次方程式の解法は，次のような流れである．まず，係数行列 A を下三角行列 L と上三角行列 U に LU 分解する．

$$LUX = B \tag{2}$$

ここで UX を C とおいて，

$$LC = B \tag{3}$$

を C について解く．これを前進消去という．最後に，

$$UX = C \tag{4}$$

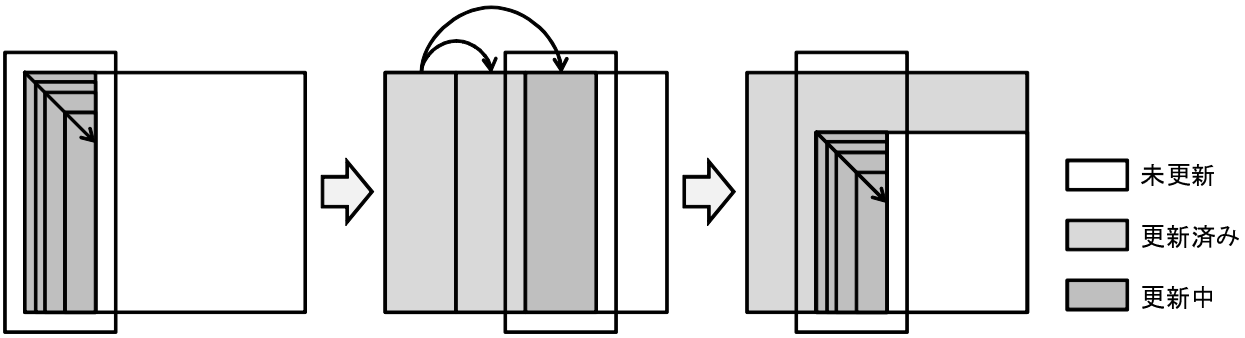

図 1: 縦ブロック分割 LU 分解

を X について解く．これを後退消去という．前進消去や後退消去は，三角行列を利用しているため計算量が少なくて済む．LU 分解には大きく分けて right-looking 法と left-looking 法，crout 法の 3 つのアルゴリズムが存在する．作成したプログラムは，right-looking 法をもとにしている．

2.1 縦ブロック分割 LU 分解

LU 分解の並列化にあたって，どの作業領域をどのプロセスに割り当てるか，どのような情報をどんな方法で同期するかなど考慮する必要がある．例えば，行列の分配方法としては，ブロック分割やサイクリック分割，ブロック・サイクリック分割などがある．

係数行列 A の LU 分解において，係数行列 A をブロック幅 n を用いて $n \times n$ の小行列のブロックに分割し，ブロック単位で計算することが望ましい．これは，LU 分解を行列積で実現することが可能で，データの利用効率が高くなるからである．また，ブロックごとに SPE を割り当てて計算させることで，並列に処理を進めることが安易になる．しかし，これではピボッティングがブロック内のみで局所的に行われるため，解の誤差が大きくなる可能性が高い．この誤差の拡大をなくすために，ピボッティングは列全体で行う縦ブロック分割並列 LU 分解の考えを取り入れた．係数行列 A を幅 n のブロック列に分割し，最左ブロック列に対して LU 分解の処理を行う．次に，この最左ブロックの情報を用いて他のブロック列を更新する．このとき，個々のブロック列は独立して処理可能なので並列に処理を進めることが可能である．

3 高速化手法

実装したプログラムの主な高速化手法について述べる．

3.1 ブロック化

ブロック化により，DMA 転送の回数を減らすことができる．また，問題によっては行列同士の計算でデータの再利用性が高まり，後述する SIMD 演算による高速化の恩恵を受けやすくなる．

前述したように，LU 分解はブロック化アルゴリズムを用いている．このときのブロック幅は，問題の制約から 32 とした．各 SPE のデータ保存領域である Local Store(以下 LS) は 256KB と限られているため，$n \times n$ のブロック単位で DMA 転送を行い，処理を進めることにしている．更新領域の最左ブロック列は，すべての SPE を用いて更新する．このときのピボッティングに関しては，更新を行いながら各 SPE で次のピボッティングの候補となる行を挙げておき，1 列更新が完了する度に同期をとり，最終的なピボッティングの行を決定する．最左ブロック列以外は，それぞれのブロック列を各 SPE が担当して更新する．これらのブロック列には依存関係がないので，依存関係を気にすることなく並列処理が可能である．

前進代入や後退代入もブロック化により高速化をはかった．係数行列は，$n \times n$ のブロック単位，右辺行列は，$n \times 4$ のブロック単位で DMA 転送し処理を行う．ただし，右辺行列において，m が 4 場合より小さい場合や 4 で割り切れずに余りがでた場合には，$n \times 1$ のブロック単位とする．

問題	1	2	3	4	5	6	7	8	9	10
N	768	224	2,688	256	1,792	32	2,720	2,240	2,656	3,200
M	1	896	1	20,000	1	131,072	1	224	100	7
行列の種類	非対称密	非対称密	非対称密	非対称密	対称密	対称密	三重対角	帯行列	対角優位	ブロック
時間 (msec)	29	11	372	204	176	381	288	294	334	580
順位	6	7	5	7	5	7	7	4	5	4

表 1: 決勝ラウンド結果

3.2 インデックスの作成

LU 分解において，ピボッティングにより係数行列の行交換が必要になる．その度にメモリ上のデータ配置を変更すると，多くの転送処理が発生してしまう．これを防ぐために，あらかじめ係数行列の行に関するインデックスを各 SPE の LS に配列としてつくっておき，行交換が発生する度に対応する行の配列要素の交換を行う．ただし，係数行列のサイズがある程度大きい場合は，メインメモリ上にインデックスをつくることにする．

係数行列からデータを持ってくる場合には，このインデックスを参照して DMA 転送する．また，LU 分解後はこのインデックスをもとに係数行列 B の行ソーティングを行う．

3.3 SIMD 演算

Cell には SIMD 命令があり，16 バイトのデータをまとめて 1 命令で実行することができる．例えば，要素が float 型の場合，4 要素に対して同じ演算を 1 つの SIMD 命令で処理可能である．SIMD 演算は，LU 分解，前進代入と後退代入の処理の核となる部分をはじめ，LU 分解後の右辺行列 B の行ソーティングにも利用している．

3.4 ループアンローリング

ループ処理を明示的にアンロールすることで，分岐の削減し豊富なレジスタを有効に利用して高速化をはかった．Cell には複雑な分岐予測機構がなく分岐ペナルティが高いため，他のプロセッサと比較して特に有効な高速化手法である．今回のような行列演算が主要な処理の場合は，ループ処理が多いのでループアンローリングの恩恵を受けやすい．

3.5 ダブル（マルチ）バッファリング

SPE は，SPU という演算処理を担当する機構と MFC という転送処理を担当する機構があり，それぞれ独立して動作することができる．これを利用して，複数のバッファを用意し，演算処理と DMA 転送処理を並行して行うことで転送にかかる時間を隠蔽することができる．これをマルチバッファリングといい，用意するバッファが２つの場合を特にダブルバッファリングという．プログラムでは，ブロック単位でのデータ転送・演算処理を行うような多くの部分でダブルバッファリングを利用している．特に，LU 分解のブロック列の更新には 1 ブロック列ごとに 2 つのバッファを用意しているので，1 つの SPE が複数のブロック列を同時並列的に更新するときにはマルチバッファリングとなっている．

4 評価結果

決勝ラウンドで出題された 10 の問題を今回作成したプログラムで解くのにかかった時間は図 1 のようになった．残念ながら，特に高速に解くような高順位の成績を残すような問題はなかったが，ある程度の性能を引き出しつつ，指定された誤差の範囲内ですべての問題を解くことができている．

5 まとめ

Cell 上で連立一次方程式を高速に解くプログラムの開発を行った．プログラムでは，縦ブロック分割 LU 分解を用いて係数行列を LU 分解し，前進代入と後退代入により解を求めるアルゴリズムを採用した．高速化手法としては，ブロック化によってデータの利用効率を高めて，さらに SIMD 演算やループアンローリングによるテクニックを用いた．また，複数のバッファを利用することで DMA 転送時間を隠蔽する手法も用いた．

参考文献

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

[2] Sony Computer Entertainment Inc. Cell Broadband Engine アーキテクチャ Ver 1.01, 2006.

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

[4] 今村俊幸. 分散メモリ型並列計算機における縦ブロック分割並列 LU 分解. 日本応用数理学会論文誌, Vol.8, No.3, pp. 373–388, 1998.