Cell/B.E.による疎行列ベクトル積の高速化

吉田　淳*　　藤井　昭宏†　小柳　義夫‡
工学院大学*　工学院大学†　工学院大学‡

1. はじめに

従来のプロセッサ開発では，コアの構造が複雑化し，消費電力の増加，発熱量の増加，高コスト化を招き，1基のコアの性能向上は限界に達している．そのため，近年は複数のコアで処理を並列に実行できるマルチコアCPUが主流となっている．また，様々な数値シミュレーションで現れる連立一次方程式の求解において，疎行列ベクトル積が計算時間の大部分を占める．そこで，浮動小数点演算において高い演算性能をもつことで知られているCell/B.E.を用いれば，疎行列ベクトル積の高速な演算処理を実現することが期待できる．しかし，Cell/B.E.の性能を最大限に発揮するには，Cell/B.E.のアーキテクチャを意識した高度なプログラミングテクニックを要する．
そこで，Cell/B.E.プログラミングを容易に実現するために開発された並列化フレームワーク，" Cell Superscalar "が注目されている．そこで本研究では，Cell Superscalarを利用して，疎行列ベクトル積の高速化をすすめ，性能，プログラム作成の容易性について評価を行う．

2. Cell/B.E.の構造

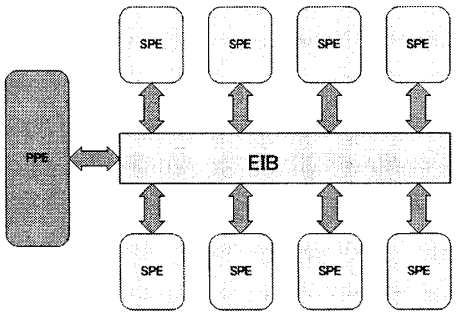

図1.　Cell/B.E.の概要図

Cell/B.E.は1個の制御系プロセッサコアと8個の演算系プロセッサコアを組み合わせた非対称型マルチコアである．

制御系プロセッサコアはPowerPC Processor Element (PPE)と呼ばれ，8個の演算系プロセッサコアはSynergistic Processor Element (SPE)と呼ばれる．SPEはそれぞれ，Local Store (LS)と呼ばれる容量が256KBの専用のメモリをもっている．各プロセッサコアは、EIB (Element Interconnect Bus)と呼ばれる高速なバスで接続されている．[1]

3. Cell Superscalar (CellSs)

Cell/B.E.プログラミングにおいて，ネックとなるのがDMAによるメモリ転送やスケジュールの管理である．Cell/B.E.の性能を発揮するには細かいチューニングが必要であり，プログラマの負担が大きい．よって，その負担を軽減できるプログラミング手法が提案されている．
プログラミングを容易にするためにBarcelona Supercomputing Centerが開発したのがCell Superscalar(CellSs)である．
CellSsは既存のソースコードに独自のpragma文を付加し，SPEで実行させる計算をtaskとして明示することで，複数のSPEで並列処理される効率のよいコードを自動で生成させるためのフレームワークである．
従来のCell/B.E.プログラミングであるIBMのCell SDKでは，PPE用，SPE用，2つのソースコードを用意する必要があったが、CellSsでは，1つのソースコードだけ用意すればよい．
図2で，1つのソースコードから実行ファイルを生成するまでの処理の流れを示す．

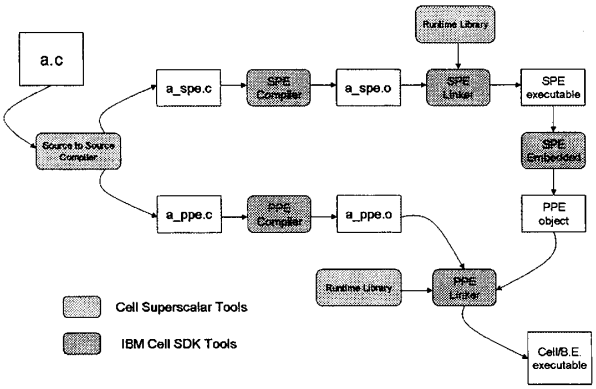

図2. CellSsの処理の流れ

Acceleration of Sparse-Matrix-Vector Product on the Cell/B.E.

*Jun Yoshida　Kogakuin University
†Akihiro Fujii　Kogakuin University
‡Yoshio Oyanagi　Kogakuin University

図 2 で示した CellSs の Source-to-Source コンパイラが task 間のデータ依存性を分析し，並列に処理できる task を自動で見つけ，多数の task のスケジューリングも CellSs に任せることができる．[2][3]
図 3 に，ブロックサイズごとに SPE で並列化する行列加算のプログラム例を示す．

```
#pragma css task input(A, B) inout(C)
void mat_add( int A[BSIZE][BSIZE], int B[BSIZE][BSIZE], int C[BSIZE][BSIZE])
{
    int i, j;

    for (i = 0; i < BSIZE; i++)
    {
        for (j = 0; j < BSIZE; j++)
        {
            C[i][j] = A[i][j] + B[i][j];
        }
    }
}
```

図 3.プログラム例

4. 疎行列ベクトル積の高速化

疎行列を格納するデータ構造については，CRS 形式と呼ばれる方法を採用した．CRS 形式では，非ゼロ要素のみを演算に使用するので，無駄な計算を省くことができる．
Cell/B.E.の性能を十分に発揮するには，大量の演算を要する部分を各 SPE に分担させることが重要である．しかし，SPE は 256KB のメモリ容量しかなく，大規模な疎行列になると，ベクトル一本を SPE に送るのも困難である．そこで本研究では，疎行列データを小さなブロックに分割し，CellSs を使って PPE が処理の終わった SPE に対して，繰り返しブロック毎のデータをタスクとして振り分けるようにする．ブロックの大きさは固定し，ブロック行列とブロック行列内のデータは CRS 形式で保持する．[4]

5. 数値実験と評価

計測に用いる疎行列データは，Matrix Market というウェブサイトから入手する．実験では，行列サイズ 8192×8192，非ゼロ要素数 41746 の Matrix Market 形式の疎行列データを使用した．ブロックの大きさは 64×64 とした．
実行には，IBM 社製の Cell/B.E.プロセッサ「PowerXCell™ 8i」を搭載したマシンを使用した．倍精度浮動小数点計算機能が強化されており，同動作周波数では従来の Cell/B.E.に比べ，理論値で 5 倍の倍精度浮動小数点計算性能を発揮する．
実験は，SPE を最大で 8 個使用し，SPE の個数ごとに処理時間を計測した．
図 4 に実験結果を示す．

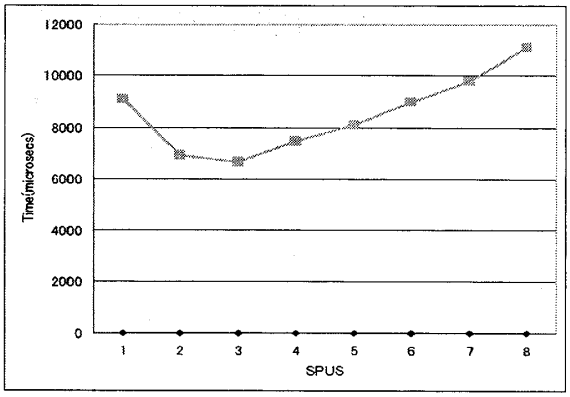

図 4.実験結果

実験結果より，SPE の個数が 3 個までは，個数が増えるごとに高速化できているが，4 個以上になると個数が増えるごとに処理時間が遅くなってしまうことがわかる．原因として 2 つ考えられる．まず，一つのブロック内の非ゼロ要素数が少ないので，転送量に対して計算量が少なく，転送と処理のバランスが悪い．次に，小さなブロックに分割したので，メモリと LS 間の転送が頻繁に起き，メモリにアクセスが集中し，帯域不足になっている．以上 2 点より，コアの個数による並列化によって，コアの個数が多くなると十分な高速化が得られなかったと考えられる．転送と処理のバランスの改善，アクセス集中の回避を行う設計が必要となる．

6. まとめ

CellSs を使うことで，Cell/B.E.上でのプログラミングを容易に実現することができたが，課題も多い．今後の課題として，実験結果から考察した改善点を検証し，プログラムを改良して十分な高速化を得る．
さらに，CellSs は SPE に実装された組み込み関数を使用できるので，細かいチューニングを行う．また，SIMD 演算を実装して，演算回数を減らすことで高速化を図る．
IBM の Cell SDK によるプログラムとの比較も行う予定である．

参考文献

[1] 「Cell プログラミングチュートリアル」
http://cell.fixstars.com/ps3linux/index.php/Cell プログラミングチュートリアル
[2] Barcelona Supercomputing Center：Cell Superscalar
http://www.bsc.es/plantillaG.php?cat_id=179
[3] IBM developerWorks Japan：
Cell Superscalar はどうだい？
http://www.ibm.com/developerworks/jp/power/library/j_pa-cellss/
[4] 櫻井 鉄也 多田野 寛人: 精度混合演算を用いたCell プロセッサ上での高性能線形計算