情報処理学会論文誌　コンピューティングシステム　Vol. 1　No. 1　51–60 (June 2008)

精度混合型 Krylov 部分空間反復法における疎行列ベクトル積の Cell BE 上での実装と性能評価

木原崇智[†1]　多田野寛人[†1]　櫻井鉄也[†1]

大規模非エルミート疎行列を持つ線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の効率的な解法として，前処理つき Krylov 部分空間反復法がある．同反復法は前処理部分に多くの計算時間を要する場合が多い．精度混合型 Krylov 部分空間反復法は，前処理演算を単精度で行っても最終的に倍精度の解が得られる方法である．多項式前処理を適用した場合には，前処理は行列ベクトル積の繰返しで得られる．一方，Cell Broadband Engine は単精度演算においてきわめて高い演算性能を持つマルチコアプロセッサである．本論文では，Cell Broadband Engine 上での単精度行列ベクトル積の実装方法とその高速化手法について示し，精度混合型 Krylov 部分空間反復法を Cell Broadband Engine 上で実装する場合の性能を，数値実験により評価する．

Implementation and Performance Evaluation of Sparse Matrix Vector Multiplication for Mixed Precision Krylov Method on the Cell BE

Takanori Kihara,[†1] Hiroto Tadano[†1] and Tetsuya Sakurai[†1]

Large sparse linear systems $A\boldsymbol{x} = \boldsymbol{b}$ arise in many scientific applications. Preconditioned Krylov subspace iterative methods are often used for solving such linear systems. The computational cost of the preconditioning part is sometimes large. The preconditioner is calculated with single precision instead of double precision in a mixed precision Krylov subspace iterative method. Speed up of matrix vector multiplication is very important for acceleration of the method. The Cell Broadband Engine provides extremely high performance single precision floating operations. We have implemented and tested the single precision sparse matrix vector multiplication on the Cell Broadband Engine. The performance was evaluated by several numerical experiments.

1. はじめに

科学技術計算の様々な分野で，大規模非エルミート疎行列を係数行列に持つ線形方程式

$$A\boldsymbol{x} = \boldsymbol{b}$$

が現れ，同方程式の求解に多くの時間を要する．したがって，同方程式を効率良く高速に解くことが求められ，様々な解法が研究されている．

Krylov 部分空間反復法[3),5),14)] は大規模線形方程式の代表的な解法である．一般的に同反復法を用いる場合，収束性を向上させるために前処理を適用する．代表的な前処理として，不完全 LU 分解[11)] や多項式前処理[11)] があげられる．多項式前処理は，行列ベクトル積を繰り返すことで得られる．この場合，行列ベクトル積の計算時間が反復法全体の計算時間の大部分を占める．したがって，行列ベクトル積の高速化が不可欠である．

文献 12) において，これらの前処理部分に対する計算を単精度で行っても，最終的に倍精度の解が得られる方法が提案された．同方法により，多項式前処理を適用する場合には単精度行列ベクトル積の高速化が重要となる．

一方，Cell Broadband Engine（Cell BE）[13)] は，SCEI，東芝，IBM が共同開発した非対称型のマルチコアプロセッサで，SCEI が販売する家庭用ゲーム機「PLAYSTATION3」などに搭載されており，単精度浮動小数点演算に関してきわめて高い性能を持つ．

本論文では，問題対象を QCD（量子色力学）シミュレーション[7)] で現れる大規模線形方程式とする．同方程式を多項式前処理つき Krylov 部分空間反復法で解く場合に，計算時間の大部分を占める単精度行列ベクトル積を Cell BE 上で実装し，同方程式を高速に解くことが本論文の目的である．疎行列ベクトル積の高速化については，これまでも様々な方法が提案されてきた[6),9),10),15)]．本論文では「Cell BE を用いた単精度行列ベクトル積の実装方法」について示し，Cell BE 上で同方程式を解く場合の性能を評価する．

2. 精度混合型 Krylov 部分空間反復法

2.1 前処理部分の単精度化

前処理つき Krylov 部分空間反復法において，前処理部分の計算を単精度で行っても，最終的に倍精度の解が得られる方法が提案された[12)]．

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, Tsukuba University

前処理は左前処理と右前処理に大別される．前処理部分の演算を単精度で行った場合に含まれる摂動の影響を考えると，左前処理を適用した場合には本来満たされるべき解と残差の関係が満たされず単精度の解しか得られない．一方，右前処理を適用した場合には，本来満たされるべき解と残差の関係が保たれるため倍精度の解が得られる．

線形方程式を前処理つき Krylov 部分空間反復法で解く場合，しばしば前処理部分の計算時間が支配的となるため，前処理部分の高速化が重要となる．文献 12) の方法により，前処理部分の計算を倍精度から単精度に置き換え可能であることは，計算の高速化において大きな意味がある．さらに，この単精度前処理を並列環境で実装する場合には，データ転送量が半分になり転送時間のボトルネックが緩和されるなどの利点もある．

2.2　多項式前処理

本論文では，Krylov 部分空間反復法の前処理として多項式前処理を適用する．以下に多項式前処理について述べる．

多項式前処理において，前処理行列 M は係数行列 A の多項式で定義される．その多項式の 1 つとして Neumann 級数展開がある．行列 A が $\|I-A\|<1$ を満たすとき，A^{-1} は Neumann 級数展開により

$$A^{-1}=(I-(I-A))^{-1}=\sum_{j=0}^{\infty}(I-A)^j$$

と表される．多項式前処理では，m 次までの以下の近似式を前処理行列 M とする．

$$A^{-1}\approx M=\sum_{j=0}^{m}(I-A)^j. \tag{1}$$

本論文で問題対象とした QCD シミュレーションで現れる線形方程式を，多項式前処理つき BiCGSTAB 法[14] で解く場合，多項式前処理（次数 5）に要する時間は全体の約 8 割を占める．行列の次元数がより大規模になると，それにともない多項式前処理の最適な次数も大きくなるため，さらに前処理部分の割合は増加する．

一方，前処理つき Krylov 部分空間反復法では，前処理行列 M とベクトルとの積が現れる．式 (1) より，多項式前処理の場合はこの部分は行列ベクトル積を繰り返すことで得られる．以上より，同線形方程式を高速に解くためには，単精度行列ベクトル積の高速化が不可欠となる．

3. Cell Broadband Engine

Cell BE は単精度浮動小数点演算において，きわめて高い演算性能を持つことで知られている．日本では 2006 年 11 月の PLAYSTATION3（PS3）の発売にともない，高性能の Cell BE 搭載マシンが安価で入手可能となった．本章では，Cell BE の構造[13] および，Cell BE で計算を行ううえで留意しておかなければならない Cell BE の特徴について示す．

3.1　Cell BE の構造

図 1 に Cell BE の構造を示す．Cell BE は，PowerPC Processor Element（PPE）と呼ばれる 1 基の制御系プロセッサコアと，Synergistic Processor Element（SPE）と呼ばれる 8 基の演算系プロセッサコアで構成される非対称型のマルチコア CPU である．PPE は，PowerPC アーキテクチャとほぼ 100%の互換性を持ち，主にオペレーティングシステムの駆動や SPE の制御を担う．一方 SPE は，PPE のような複雑なプログラム制御は苦手であるが，計算を単純に繰り返すマルチメディア系の処理を得意とする．

各プロセッサコアは，Element Interconnect Bus（EIB）と呼ばれる高速なバスで接続されている．また，EIB はメインメモリや外部入出力デバイスとも接続されており，各プロセッサコアは EIB を経由してデータアクセスを行う．

SPE でのプログラム実行は，SPE 内の Local Store（LS）と呼ばれる 256 KB の内部メモリ上で行われるため，メインメモリからプログラムやデータを LS に転送する必要がある．このデータ転送には Direct Memory Access（DMA）転送を用いる．DMA とは，機

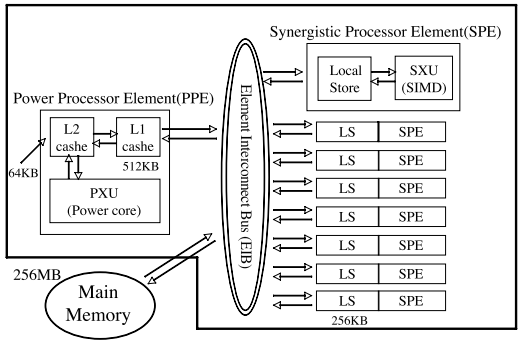


図 1　Cell BE の構造
Fig. 1　Architecture of the Cell BE.

械語の命令群によらず，メモリとメモリ（またはメモリと I/O デバイス）の間で直接データを転送することである．DMA 転送には転送元のデータが格納されている先頭アドレスとそのバイト長が必要となる．各 SPE はこれらの情報を持ちさえすれば，それぞれ Memory Flow Controller （MFC）と呼ばれる制御ユニットに専用の DMA コントローラを持つため，適宜メインメモリから自身の LS に必要なデータを DMA 転送することが可能である．

3.2　Cell プログラミングの特徴

SPE は Single Instruction Multiple Data（SIMD）系のアーキテクチャで，ベクタデータ型を用い複数データを 1 命令で処理可能である．Cell BE で扱うベクタデータは 16 バイト固定であるため，単精度浮動小数点演算では 4 つのデータを 1 命令で処理できる．

並列計算においては，しばしばデータ転送時間がボトルネックとなるが，メインメモリと LS 間のデータ転送で用いられる DMA 転送の遅延は，ダブルバッファリングと呼ばれる手法により隠蔽することが可能である．これは，バッファを 2 組用意し，一方のバッファに対し演算を行っている裏で，もう一方のバッファに対し DMA 転送を行うことで実現する．

SPE は SIMD 命令，かつダブルバッファリングを駆使したプログラミングにより単精度演算においてきわめて高い演算性能を持つ．Cell BE は，複数個の SPE を効率的に活用することにより高い演算能力を発揮する．

一方，PPE は SPE に比べ演算性能が低い．したがって，PPE に負担のかかる実装は非効率であり，Cell BE で高い性能を出すためには，可能な限り多くの計算を SPE で行わせることが重要である．

SPE スレッドの生成には多くの時間を要する．したがって，1 度スレッドを生成したらそのスレッドを何度も使い回すことが望ましい．同一のスレッドを使って，PPE から繰り返し SPE を呼んだり，異なる仕事をさせたりする場合には，SPE はつねにフラグ待ちの状態にしておき，PPE は SPE に行わせる仕事に応じて異なるフラグを SPE に送信する．SPE は受け取るフラグに応じて，行うべき仕事を判断する．本論文ではこのフラグの通信に Mailbox 通信を利用した．Mailbox 通信は 32 ビットのデータを PPE と SPE の間で双方向に送受することができる．

4.　Cell BE 上での行列ベクトル積の実装方法

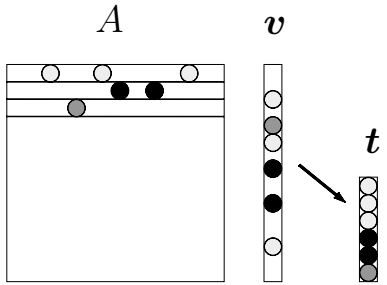
本章では Cell BE 上での行列ベクトル積（$A \times \boldsymbol{v}$）の実装方法について示す．ここで行列 A は大規模疎行列であり，行列データは Compressed Row Storage（CRS）形式[1] で格納されているとする．

Cell BE で高い性能を得るためには，効率的に SPE を活用することが不可欠である．しかしながら，SPE の LS は 256 KB ときわめて小さいため，大規模な行列ベクトル積においては掛けるベクトル 1 本すら LS に格納できない．したがって，行列ベクトル積を SPE で計算させるためには，行列の一部とベクトルの一部をメインメモリから繰り返し SPE の LS に DMA 転送し計算させる必要がある．以下に，Cell BE 上での行列ベクトル積の実装方法を 2 つあげる．

4.1　パッキングを用いた実装

ベクトル $\boldsymbol{v}$ はそのままでは LS に格納できないため，ベクトル $\boldsymbol{v}$ から，行列の n 行分（$n \ll$ 次元数）の非ゼロ要素に対応する計算に必要な要素だけを取り出し，ベクトル $\boldsymbol{t}$ にパックする．**図 2** にパッキングのイメージを示す．そして $\boldsymbol{t}$ が得られたら，SPE の LS に行列 A の n 行分の疎行列データと，$\boldsymbol{t}$ を DMA 転送する．そして SPE で行列 n 行分の計算が終わったら，結果をメインメモリに DMA 転送する．同様の処理を（A の次元数/n）回繰り返し，$A \times \boldsymbol{v}$ を計算する．

パッキングによる方法の利点は，SPE での演算において，ベクトル $\boldsymbol{t}$ が計算に必要なデータのみ連続して並んでいるため，ベクトル $\boldsymbol{t}$ をそのままベクタデータ型に変換でき SIMD 命令が利用できる点である．また，計算に必要なデータのみを抽出するため，データ転送量も少ない．しかしながら，PPE でのベクトル $\boldsymbol{t}$ の生成の負担が大きく，複数の SPE に対して処理が間に合わないという問題点がある．



○：非ゼロ要素

図 2　ベクトルのパッキング
Fig. 2　Packing of a vector.

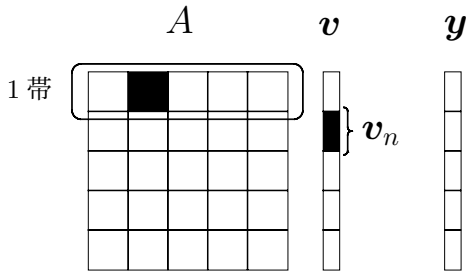


図 3　行列のブロック分割化
Fig. 3　Partitioning of a matrix.

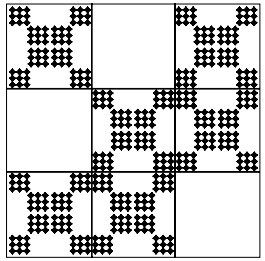
4.2　ブロック化を用いた実装

図 3 のように，行列を小行列（以下，ブロックと呼ぶ）に分割し，ブロックとそれに対応するベクトル $\boldsymbol{v}_n$ を SPE の LS に DMA 転送し計算する．横 1 行のブロック（以下，1 帯と呼ぶ）は 1 つの SPE で計算させる．したがって，SPE では各ブロックの行列ベクトル積の結果を順々に足し合わせれば，1 帯分の行列ベクトル積の結果が得られる．SPE は 1 帯分の結果が得られたら，結果をメインメモリに DMA 転送する．

1 帯の計算を始める際に，あらかじめ少なくとも 1 帯分のブロック行列データそれぞれの先頭アドレスとそのバイト長を SPE に知らせておくことが重要である．これにより SPE は，1 ブロック分の計算が終われば次のブロックの計算に必要なデータは勝手に DMA 転送で取得可能であるため，1 帯の計算が終わるまで PPE はまったく関与する必要がない．PPE はある SPE から計算終了のフラグが得られたら，次の帯の計算に必要なデータを SPE に送る．

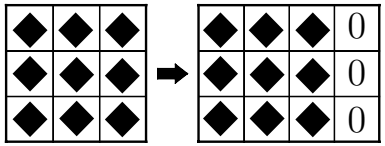
また行列 A は疎行列であるため，ブロック分割するとブロックの中の要素がすべてゼロであるブロック（以下，ゼロブロックと呼ぶ）が現れる．ゼロブロックの場合，SPE が DMA 転送でデータを要求すると，行列データの通信量はほぼゼロであるが，ブロックに対応するベクトル $\boldsymbol{v}_n$ は転送される．このとき SPE で計算は行われず，$\boldsymbol{v}_n$ の転送が無駄な処理となり遅延の原因になる．そこで，ブロックを 1 つの要素と見なした CRS 形式[1] のデータを持ち，ブロック内に非ゼロ要素があるブロック（以下，非ゼロブロックと呼ぶ）のみを計算に反映させる．これにより，ゼロブロックは無視されベクトル $\boldsymbol{v}_n$ の無駄な転送が回避できる．

行列のブロック化による方法の利点は，PPE での負担が小さく，SPE 並列計算の台数効果が期待できる点である．また，ゼロブロックに対応するベクトル要素は転送されないため，計算に反映されないベクトル要素の転送量が小さい．しかしながら，A は疎行列であるため，ベクトル $\boldsymbol{v}$ の計算に使用する要素は連続に並んでいない．したがって，そのままではベクタデータ型に変換できず SIMD 命令が使えないという問題点がある．



◆：非ゼロ要素

図 4　QCD 行列の非ゼロ構造
Fig. 4　Nonzero structure of the QCD matrix.



◆: 非ゼロ要素

図 5　SIMD 化のためのゼロパディング
Fig. 5　Zero padding for SIMDization.

4.2.1　SIMD 命令を利用するための工夫

本論文では，QCD シミュレーションで現れる行列を扱う．**図 4** は QCD シミュレーションで現れる行列の非ゼロ構造の一部分を示した図である．図 4 より，3×3 のブロックを最小単位とした構造が確認できる．そこで，3×3 の密行列に着目する．Cell BE の場合，扱えるベクタデータ型は 16 バイト固定であるため，単精度データであれば 4 データ 1 セットがベクタデータ型の 1 変数となる．したがって，3×3 ブロックのままではベクタデータ型に変換ができない．そこで，**図 5** のようにデータ 3 つごとにゼロをパディングし，3×4 のブロックとする．これにともない，ベクトル $\boldsymbol{v}$ も要素 3 つごとにゼロをパディングする．これらのデータをベクタデータ型とし，SIMD 命令により演算の高速化を図る．なお，ゼロ

パディングにより行列の非零要素数は 4/3 倍となるが，3×4 ブロックを 1 要素と考えることで，行列の次元数を本来の 1/3 と見なすことが可能となり，CRS 形式[1] のインデックス（col_ind）のデータ量は 1/9 倍，ポインタ（row_ptr）のデータ量は約 1/3 倍となる．これにより，単精度の場合には合計の行列データ量を削減できるという利点がある．また実装面の工夫として，3×3 の構造から 1 度レジスタにフェッチしたベクトルデータを 3 回使い回すことが可能であり，ロードの回数を削減できる．

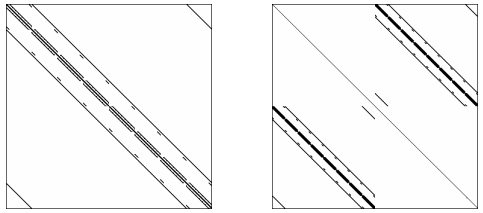
5. 数値実験

5.1 実験環境

数値実験は SONY PLAYSTATION3（CPU: PPE & SPE 3.2 GHz, Memory: 256 MBytes）上で行い，開発言語には C 言語を用いた．コンパイラおよびコンパイラオプションは “Compiler: ppu-gcc & spu-gcc，Compile option: `-O3`” とした．実験対象は QCD シミュレーションで現れる線形方程式とした．係数行列 A（非ゼロ構造：**図 6** (a)）は Matrix Market[8] にある `conf5.4-00l8x8-1000`（次元数：49,152）であり，右辺ベクトルは $\boldsymbol{b}=[5.4,5.4,\cdots,5.4]^{\mathrm{T}}$ とした．

行列ベクトル積の計算には A に対し Red-Black オーダリングを行った行列（非ゼロ構造：図 6 (b)）に対して，SSOR 前処理[4] を施して得られた行列（次元数：24,576）を用いた．

Krylov 部分空間反復法は BiCGSTAB 法[14] を用い，前処理には多項式前処理[11] を適用した．このとき多項式前処理の計算は単精度で行い，その他の部分の計算は倍精度で行った．BiCGSTAB 法の収束判定値は $\|\boldsymbol{r}_k\|_2/\|\boldsymbol{b}\|_2\le1.0\times10^{-15}$ とし，多項式前処理の次数 m は 5 とした．ここで $\boldsymbol{r}_k$ は，BiCGSTAB 法の第 k 番目の残差である．



(a) Natural ordering.　(b) Red-Black ordering.

図 6　QCD 行列の非ゼロ構造
Fig. 6　Nonzero structure of QCD matrix.

また，SPE スレッドは 1 度生成したら何度も使い回せるため，行列ベクトル積 1 回の実行時間には SPE スレッドの生成・破棄の時間は含めない．

Cell プログラミングには，Cell Toolkit Library（CTK）[2] と呼ばれる Cell プログラム開発支援のためのオープンソースライブラリを利用した．CTK は SPE プログラムの実行制御や複数 SPE プログラム，PPE プログラム間の並列処理のための基本的な API を提供する．また，CTK を使って開発されたアプリケーションは，IBM，Sony，東芝の各種 Cell Linux 環境上でソースコードレベルの互換性を持つという利点もある．

本論文では以下で，データ転送時間や PPE での処理時間なども含めた全体の計算時間を「実行時間（runtime）」と呼び，SPE での行列ベクトル積部分の演算のみに要した時間を「演算時間（computation time）」と呼ぶこととする．

5.2 実験結果

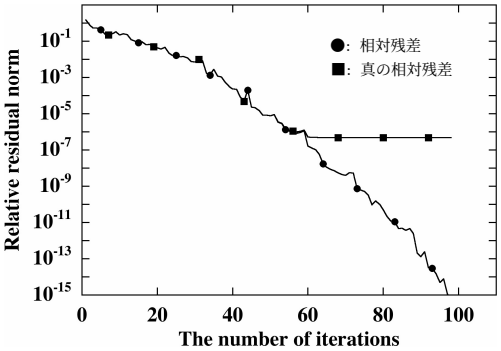
まず，単精度前処理の有効性について示す．**図 7** は多項式前処理つき BiCGSTAB 法の相対残差履歴である．相対残差は漸化式から求めた残差ベクトルのノルムであり，真の相対残差は各ステップごとで得られた近似解から，残差ベクトルを計算してそのノルムをとったものである．図 7 より左前処理の場合には，前処理部分の計算を単精度で行ったことによる摂動の影響で，真の相対残差が単精度でとどまってしまうことが分かる．一方，右前処理を適用した場合には前処理部分の計算を単精度で行っても本来満たされるべき解と残差の関係が保たれるため倍精度の解が得られることが分かる．なお，図 7 の実験は PPE 上で行った．

5.2.1 パッキングによる実装

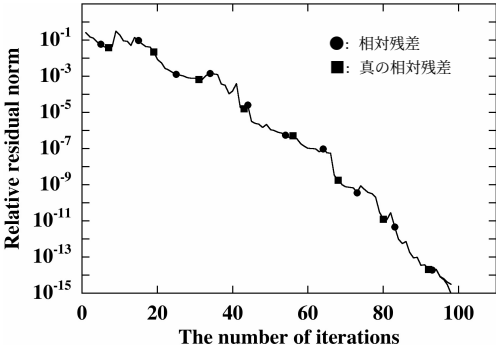
4.1 節で説明したベクトル $\boldsymbol{t}$ を，PPE で行列の次元数分生成するのに要する合計時間は 72,307 μsec であった．一方，PPE 単体で通常の単純な行列ベクトル積を行わせた結果は 21,023 μsec であった．したがって，「間接参照＋代入」という操作は演算に比べ非常に遅く，パッキングによる方法では行列ベクトル積の高速化は期待できない．

5.2.2 ブロック化による実装

まず，本実験における SPE でのループアンローリングの効果について示す．ループアンローリングとは，1 回のループ内で命令列を複数回分並べることで，ロード/ストアの回数やループのオーバヘッドを削減させるための手法である．最適なループアンローリングの展開数は，マシンや問題によって異なる．本実験で用いた行列は 3×3 の非ゼロ構造を持つことから分かるように，1 行あたりの非零要素数は 3 の倍数となっている．具体的にはブロック化によるブロックの 1 行あたりの非零要素数は（0，3，6，24）の 4 パターンとなる．ここで通常最適な展開数が 4 ～ 8 程度であること，および 1 行あたりの非零要素数が少なく



(a) Left preconditioning.



(b) Right preconditioning.

図 7　多項式前処理つき BiCGSTAB 法の残差履歴
Fig. 7　Relative residual norm histories for the BiCGSTAB with the polynomial preconditioner.

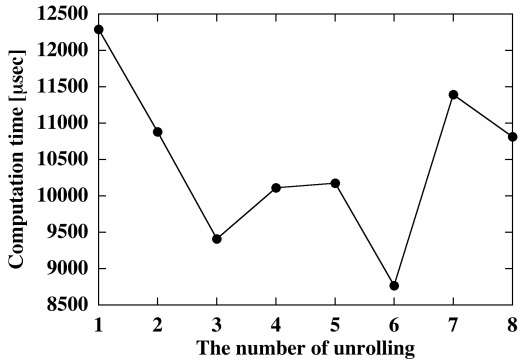


図 8　ループアンローリングの効果
Fig. 8　Effect of loop unrolling.

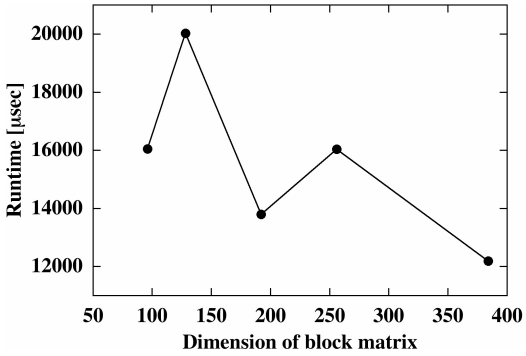


図 9　ブロック行列の次元数に対する行列ベクトル積 1 回の実行時間
Fig. 9　Runtime of a matrix-vector multiplication for dimension of block matrix.

端数処理の影響が大きいことを考慮すると，本実験での最適な展開数は 3 もしくは 6 であると予想できる．

図 8 は本実験における SPE でのループアンローリングの効果を示しており，横軸はループアンローリングの展開数，縦軸は行列ブロック化によるブロックの次元数が 384 のときの SPE での行列ベクトル積 1 回の演算時間である．図 8 より，本実験ではループアンローリングの展開数が 6 のとき最も高速であり，上記の予想は正しかったことが分かる．以下の実験では，展開数を 6 とした．

図 9 は行列ブロック化によるブロックの次元数（96，128，192，256，384）に対する，行列ベクトル積 1 回の実行時間である．次元数が大きくなるに従って，実行時間が短くなる傾向がある．ただし，LS の制限があるためこれ以上次元数を大きくすることは不可能であった．ここで，ブロックの次元数（96，128，192，256，384）に対する非ゼロブロック内の最小非零要素数はそれぞれ（144，2，288，2，576）であった．これより次元数が 128 と 256 のときに実行時間が大きくなったのは，ブロック化した際に非零要素数が極端に少ないブロックが生じるため効率が悪くなるからだと考えられる．以下の実験では，ブロック

の次元数を 384 とした．

図 10 はダブルバッファリングを行った場合と行わなかった場合（以下，シングルバッファリングと呼ぶ）の単精度行列ベクトル積 1 回の実行時間の結果である．また，**図 11** はシングルバッファリングの場合の実行時間の内訳を示している．ここで，凡例の “DMA transfer” は DMA 転送時間，“Other” は PPE での作業時間や SPE での配列の初期化などに要した時間，“Computation” は SPE での行列ベクトル積部分のみの演算時間である．図 10 と図 11 より，ダブルバッファリングにより DMA 転送時間がほとんど隠蔽されたことが分かる．しかしながら，実行時間の大部分を演算時間 “Computation” が占めているため，ダブルバッファリングにより実現した高速化は，全体の実行時間に対しては小さな割合であった．したがって，行列ベクトル積のさらなる高速化のためには演算部分の高速化が重要である．

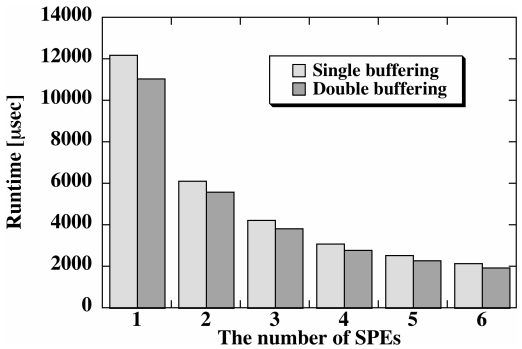


図 10　SPE の数に対する行列ベクトル積 1 回の実行時間
Fig. 10　Runtime of a matrix-vector multiplication for the number of SPEs.

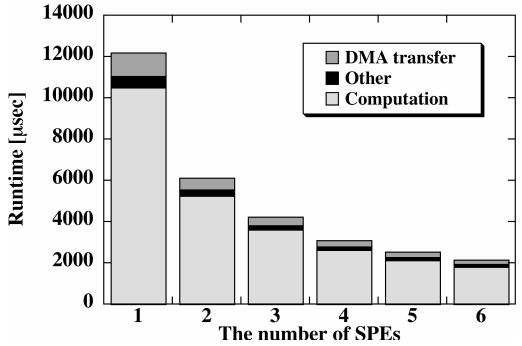


図 11　SPE の数に対する行列ベクトル積 1 回の実行時間の内訳
Fig. 11　Details in runtime of a matrix-vector multiplication for the number of SPEs.

ここで参考に，Dell Precision Workstation 470 （CPU: Intel Xeon 3.2 GHz，RAM: 2.0 GBytes，OS: Red Hat Enterprise Linux WS 4）上で，コンパイラおよびコンパイラオプション “Compiler: GNU Fortran ver.3.4.3，Complier option: `-O3`” を用い，本実験で用いた行列の行列ベクトル積を行ったところ，計算時間は 5,703 μsec であった（ループアンローリングあり，SSE 命令なし）．一方，PS3 では「ダブルバッファリング・6 SPE 使用」の場合の実行時間は 2,166 μsec であった．したがって，上記 Xeon の結果と比べ 2.6 倍以上高速であった．これは QCD 行列を用いた場合の結果であるが，QCD と非ゼロ構造が似ている行列であれば,「ブロック化＋ダブルバッファリング」で同程度の性能が期待できる．

「SIMD 命令を利用した場合の実験結果」

図 12 は SIMD 命令を使った場合の本実験における SPE でのループアンローリングの効果を示している．横軸はループアンローリングの展開数，縦軸は行列ベクトル積 1 回の SPE での演算時間である．図 12 より，本実験では展開数が 8 の場合が最も高速であったことが分かる．SPE は 128 bit のレジスタを 128 本有するため，問題にもよるがループアンローリングの最適展開数は多くなる傾向がある．以下の実験では，SIMD 命令を利用する場合にはループアンローリングの展開数を 8 とした．

図 13 は「シングルバッファリング・ダブルバッファリング」「SIMD 命令を利用しない場

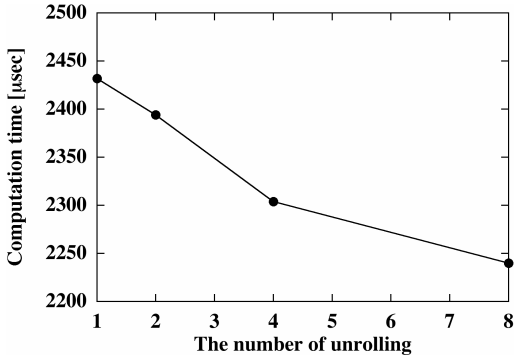


図 12　SIMD 版のループアンローリングの効果
Fig. 12　Effect of loop unrolling in case of using SIMD.

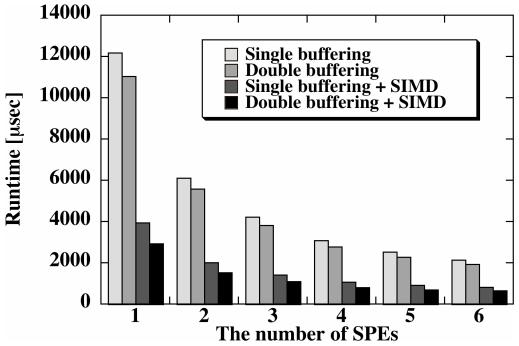


図 13　SPE の数に対する行列ベクトル積 1 回の実行時間
Fig. 13　Runtime of a matrix-vector multiplication for the number of SPEs.

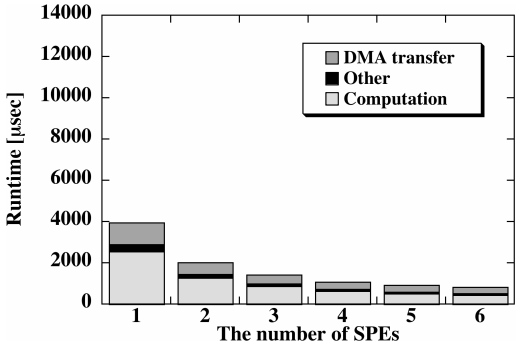


図 14　SPE の数に対する SIMD 版の行列ベクトル積 1 回の実行時間の内訳
Fig. 14　Details in runtime of a matrix-vector multiplication for the number of SPEs (SIMDver.).

合・利用した場合」それぞれの行列ベクトル積 1 回の実行時間の結果である．SIMD 命令により大幅な高速化が実現できたことが確認できる．ここで，シングルバッファリングの場合の SIMD 命令利用時の実行時間の内訳を**図 14** に示す．図 14 から，図 11 の場合と比べ SPE での演算時間 “Computation” が大幅に減少したことが分かる．SIMD 命令を利用するためのゼロパディングにより演算量は 4/3 倍となったが，SIMD 命令により “Computation” は 4.1 倍高速化された．なお，SIMD 命令により “Computation” が 4 倍以上高速になる理由として，コンパイラのオプティマイズが効きやすくなるなどの理由がある．

また，図 14 においては，使用する SPE の数が多くなると，DMA 転送時間 “DMA transfer” が図 11 の場合と比べ大きくなる傾向が見られた．6 SPE 使用時の図 11 の “DMA transfer” は 206 μsec であり，図 14 の “DMA transfer” は 301 μsec であった．これは，SIMD 命令により演算時間 “Computation” が高速化され，メモリバンド幅の影響が出てきたためだと考えられる．図 11 では “DMA transfer” に比べ “Computation” の割合が圧倒的に大きかったが，図 14 においては，6 SPE 使用時には “DMA transfer” と “Computation” の割合がほぼ同程度となった．また，このときダブルバッファリングで隠蔽不可能な “DMA transfer” の割合が大きくなった．

なお,「SIMD・ダブルバッファリング・6 SPE 使用」の場合の実行時間は 671 μsec であった．一方，上記 Xeon において，SSE 命令を利用し，ループアンローリングにより最適化を行った場合の結果は 3,532 μsec であった．

ここで，Cell BE のメモリバンド幅と本論文での実装方法による疎行列ベクトル積の演算性能の関係について考察する．文献 15) に，IBM 3.2 GHz Cell blade（using 8 SPEs）上での実装による “密行列密行列積” と “疎行列ベクトル積” の実験結果（Flops 値）が示されている．文献 15) によると，“密行列密行列積” の場合の Flops 値は，単精度演算では 204.7 Gflops であり，倍精度演算では 14.6 Gflops である．この場合，単精度演算は倍精度演算の約 14 倍の性能が出ている．一方，SPE の理論ピーク性能は単精度演算が倍精度演算の約 14 倍である．したがって，両者の値は一致している．

これに対し，文献 15) での実非対称疎行列における “疎行列ベクトル積” の Flops 値（4 種の平均）は，単精度演算では 3.59 Gflops であり，倍精度演算では 2.52 Gflops である．したがってこの場合，単精度演算は倍精度演算に比べ約 1.4 倍しか性能が出ていない．これは “疎行列ベクトル積” の場合，メモリバンド幅が性能を支配しており，Cell BE 本来の演算性能を発揮するためにはメモリバンド幅が不足しているためだと考えられる．本論文の数値実験においても「SIMD・ダブルバッファリング・複数 SPE 使用」の場合，メモリバンド幅の影響が生じ，演算性能向上を妨げた．

一方，本論文で提案した実装方法では,「SIMD・ダブルバッファリング・6 SPE 使用」の場合，14.2 Gflops の演算性能が得られた．この結果が，上記文献 15) の “疎行列ベクトル積” の結果と比べ 4 倍程度高い理由として，本論文の方法が，QCD シミュレーションで現れる行列の非ゼロ構造に特化した実装方法であること，および，文献 15) の結果は実疎行列を扱っているのに対し，本論文では複素疎行列を扱ったため，Byte/Flop 値が有利であるためと考えられる．たとえば，$c = c + a \times b$ という計算を考えた場合，a，b，c が単精度実

数の場合は Byte/Flop 値は 8 で，a，b，c が単精度複素数の場合は Byte/Flop 値は 4 であり，単精度複素数の場合の Byte/Flop 値は単精度実数の場合の 1/2 倍である．

以上から，疎行列を扱う場合には，Byte/Flop 値を小さくするための実装上の工夫が重要であることが分かる．また同様に，倍精度と単精度の Byte/Flop 値を比べた場合は，単精度が倍精度の 1/2 倍である．したがって，Byte/Flop 値を小さくするためにも精度混合型 Krylov 部分空間反復法の有効性は高いと考えられる．

6. ま と め

多項式前処理つき Krylov 部分空間反復法の前処理計算を単精度で置き換えた場合，右前処理の場合には単精度化での誤差が近似解の精度に影響を与えず，倍精度の解が得られた．多項式前処理は行列ベクトル積の繰返しで得られる．多項式前処理を高速に行うため，単精度行列ベクトル積を Cell BE 上で実装した．

ベクトルのパッキングによる方法では，4.1 節で述べたベクトル $\boldsymbol{t}$ の生成に多くの時間を要してしまい性能が出なかった．一方，行列のブロック化による方法では，SIMD 命令を利用しなくても比較的高い性能が得られた．

さらに QCD 行列の非ゼロ構造に着目した実装（ゼロパディング）を行い，SIMD 命令が利用できるよう工夫した．ゼロパディングにより計算量は 4/3 倍になるが，SIMD 命令を利用することで，SPE での演算時間は SIMD 命令を利用しない場合と比べ 4.1 倍高速になった．最も高い性能が得られたのは,「SIMD・ダブルバッファリング・6 SPE 使用」の場合で，671 μsec であった．このとき，SIMD 命令の利用により演算時間が高速化されたことで，メモリバンド幅の影響が生じ，DMA 転送時間がボトルネックとなった．

7. 今後の課題

Cell BE 上での行列ベクトル積をより高速に行うために，Byte/Flop 値を小さくするための実装上の工夫が必要である．また，最適なブロック次元数の自動決定も課題である．これは 1 帯あたりの非ゼロブロック数，および非ゼロブロック内の最小，最大非ゼロ要素数を利用すれば可能であると考えられる．本論文では問題対象を QCD シミュレーションで現れる線形方程式としたが，QCD 行列以外の疎行列ベクトル積の Cell BE 上での性能評価も今後行う必要がある．また，シングルコアの Xeon との比較だけでなく，最新の汎用型のマルチコア機との比較も求められる．さらに，多項式前処理つき BiCGSTAB 法全体の実装および性能評価のためや，実用規模での計算を実現するための Cell BE を利用したクラスタの構築も研究課題である．

参 考 文 献

1) Barrett, R., Berry, M. and Chan. T.F.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia (1994).
2) CTK: Cell ToolKit Library. http://ctk-dev.sourceforge.net/
3) Eisenstat, S.C., Elman, H.C. and Schultz, M.H.: Variational iterative methods for nonsymmetric systems of linear equations, *SIAM J. Numer. Anal.*, Vol.20, No.2, pp.345–357 (1983).
4) Frommer, A., Lippert, T. and Schilling, T.: Scalable parellel SSOR preconditioning for lattice computations in gouge theories, *European Conference on Parallal, Processing*, pp.742–749 (1997).
5) Gutknecht, M.H.: Variants of BiCGSTAB for matrices with complex spectrum, *SIAM J. Sci. Comput.*, Vol.14, pp.1020–1033 (1993).
6) 工藤　誠，黒田久泰，片桐孝洋，金田康正：並列疎行列ベクトル積における最適なアルゴリズム選択の効果，情報処理学会研究報告，Vol.2002, No.22, pp.151–156 (2002).
7) Kuramashi, Y., Aoki, S., Ishikawa, K., Ishikawa, T., Ishizuka, N., Kanaya, K., Tsutsui, N., Okawa, M., Taniguchi, Y., Ukawa, A and Yoshie, T.: 2+1 Flavor Lattice QCD with Lüscher's Domain-Decomposed HMC Algorithm, *Pos LAT2006:029* (2006).
8) Matrix Market. http://math.nist.gov/MatrixMarket/
9) Nishtala, R., Vuduc, R.W., Demmel, J.W. and Yelick, K.A.: Performance modeling and analysis of cache blocking in sparse matrix vector multiply, Technical report, University of California, Berkeley, EECS Dept. (2004).
10) Pinar, A. and Heath, M.T.: Improving Performance of Sparce Matrix-Vector Multiplication, *Proc. Supercomputing 99* (1999).
11) Saad, T.: *Iterative methods for sparse linear systems* (2nd edition), SIAM, Philadelphia (2003).
12) Tadano, H. and Sakurai, T.: On single precision preconditioners for Krylov subspace iterative methods. *Proc. LSSC'07*, Lecture Notes in Computer Science, Vol.4818, pp.721–728 (2008).
13) The Cell project at IBM Research. http://www.research.ibm.com/cell/home.html
14) Van der Vorst, H.A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, Vol.13, pp.631–644 (1992).

15) Williams, S., Shalf, J., Oliker, L., Kamil, S., Husbands, P. and Yelick, K.: Scientific Computing Kernels on the Cell Processor, *International Journal of Parallel Programming*, Vol.35, No.3, pp.263–298 (2007).

(平成 19 年 10 月 10 日受付)
(平成 20 年 1 月 27 日採録)

木原　崇智

2008 年筑波大学大学院博士前期課程システム情報工学研究科修了．同年，日本電気株式会社入社．主に，線形方程式，固有値問題の並列解法の研究に従事．日本応用数理学会 2007 年度年会若手優秀講演賞受賞．2008 年情報処理学会 HPCS2008 最優秀論文賞受賞．日本応用数理学会会員．

多田野寛人

2006 年筑波大学大学院博士課程システム情報工学研究科修了．博士（工学）．同年，独立行政法人科学技術振興機構研究員．2007 年，京都大学大学院情報学研究科 JSPS 助教．現在，筑波大学大学院システム情報工学研究科助教．大規模線形計算，主に連立一次方程式の反復解法と固有値問題の並列解法の研究に従事．2008 年情報処理学会 HPCS 最優秀論文賞受賞．日本応用数理学会会員．

櫻井　鉄也（正会員）

1986 年，名古屋大学大学院工学研究科博士課程前期課程修了．同年，名古屋大学工学部助手．1993 年，筑波大学電子・情報工学系講師．1996 年，同大学助教授．現在，筑波大学大学院システム情報工学研究科教授．独立行政法人産業技術総合研究所客員研究員．博士（工学）．大規模固有値問題の並列解法，数理ソフトウェアの利用支援環境の研究に従事．1996 年日本応用数理学会論文賞，2008 年 HPCS2008 最優秀論文賞受賞．日本応用数理学会，日本数学会，SIAM 各会員．