Blue Geneに適した格子QCDプログラムの超並列化

土井 淳[†] 寒川 光[†]
松古栄夫[††] 橋本省二[††]

Blue Gene は専用計算機の価格性能比で，汎用性を備えたスーパコンピュータを目的として開発された．組み込みシステム向けである PowerPC440 をコアとしたデュアルコアプロセッサであり，各 CPU に FPU を 2 つ搭載したダブル FPU という構造を持つ．また，チップ上に実装した通信機能によって相互接続することで，非常に大規模な並列計算機を構築することができる．我々は格子 QCD アプリケーションを，Blue Gene のダブル FPU 命令セットを効果的に利用し，MPI よりも低遅延時間で通信のできるトーラスパケット通信を直接利用することで，高い並列度を実現することに成功した．本論文ではこれらの Blue Gene に特有の手法について報告する．

Highly Parallelization of Lattice QCD Program for Blue Gene

Jun Doi,[†] Hikaru Samukawa,[†] Hideo Matsufuru[††] and Shoji Hashimoto[††]

Blue Gene is developed to realize a general purpose supercomputer with excellent price/performance like special purpose machine. It can configure very large scale parallel machine with interconnecting many compute nodes which consist of embeded PowerPC 440 core attached with double FPU. This paper report a success of tuning for Blue Gene effectively using double FPU capability and highly parallelization without using MPI in lattice QCD application paralellization.

1. はじめに

Blue Gene☆は，開発が計画された 1999 年当時，高性能汎用マイクロプロセッサを多数使用する並列型スーパコンピュータと，QCDSP のような専用計算機との価格性能比の大きな差に着目し，専用計算機の価格性能比を受け継いだまま汎用性を持つスーパコンピュータを目指して開発された．そのために，低電力で，発熱量の低い，低周波数の組み込み用プロセッサを専用 ASIC チップの中に含めた計算チップを作成し，これをできるだけ数多く搭載することで高密度実装を実現する超並列計算機である．このアプローチは，現在一般的に使用される並列化よりも高い並列度を実現するのに適しているが，MPI を用いて分散メモリ型のプログラムの並列化を行うのが問題によっては難しく，性能を引き出すのは容易ではない．

本論文は，格子 QCD プログラムのカーネル部分であるウィルソン・ディラック演算子の計算に，Blue Gene のダブル FPU を効果的に用い，通信処理に MPI の代わりにトーラス網通信を直接利用して，高いスケーラビリティを実現した方法を述べる．以下，2 章で Blue Gene の概要，3 章で格子 QCD プログラムの特徴，4 章で格子 QCD プログラムのダブル FPU による高速化と並列化の手法，5 章で計測結果および考察，6 章で本論文をまとめる．

2. Blue Geneの概要

Blue Gene の ASIC チップは，プロセッサコアに 2 つの PowerPC 440（PPC440）を持ち[1)]，それぞれのコアには，既存の浮動小数点演算ユニット（FPU）を 2 つ搭載した，ダブル FPU と呼ばれる機構を備える．さらに L2 キャッシュ，L3 キャッシュのディレクトリ，eDRAM による 4 MB の L3 キャッシュなどを持ち，トーラス網，グローバルツリー，グローバルイ

† 日本アイ・ビー・エム株式会社東京基礎研究所
Tokyo Research Laboratory, IBM Research
†† 高エネルギー加速器研究機構
High Energy Accelerator Research Organization

☆ IBM が米ローレンス・リバモア国立研究所に納入したスーパコンピュータ Blue Gene/L をベースにした，IBM eServer Blue Gene Solution を指す．本文では「Blue Gene」の略称を使用した．

ンタラプト，Gbit イーサなどの通信機能もソフトコアで実装している．

並列計算機としての基本構成は 1 筐体の半分の 512 計算ノードからなるミッドプレーンと呼ばれる構成である．ミッドプレーンでは，各計算ノードは XYZ 方向に相互接続され，8×8×8 の 3 次元トーラス☆がネットワークの基本となる．さらに複数のミッドプレーンをリンク ASIC を用いて接続することで，大規模な 3 次元トーラスによる並列計算機を構成することができる．

1 ミッドプレーンは，計算ノードを単位とした 512 ウェイか，CPU を単位とした 1024 ウェイの並列計算機として使用することができる．前者をコプロセッサ・モードと呼び，一方の CPU は MPI エンジンとして稼動する．後者を仮想ノード・モードと呼び，2 つの CPU がそれぞれ独立にユーザプログラムを実行する．理論最大性能は 1 CPU が 2.8 Gflop/s なので，仮想ノード・モードの場合，1 筐体（2048 CPU）で 5.6 Tflop/s である．

2.1 計算 ASIC

ベースとなる PPC440 は 32 ビットアドレッシングの組み込み用のプロセッサで，クロック周波数は 700 MHz である．この 32 ビット実装は多重プロセッササポートを持たない☆☆．組み込み用なので浮動小数点ユニットは持たないが，Auxiliary Processor Unit（APU）ポートを密結合できる仕様になっている．BG/L ではこの APU に 2 つの FPU を密結合している．

L1 は 32 KB/32 KB の命令/データキャッシュで，FPU に対しては 128 ビットのインタフェースを持つ．**図 1** に計算 ASIC チップの構成を示したが，2 つの L1 間にはコヒーレンシがない．これを補う目的で，ロックボックスや共有 SRAM などの同期の手段を備えている．共有 SRAM は 16 KB で 2 つの CPU コアから共有されるので，2 つの CPU 間の高速な通信に使用できる．L2 は容量が 2 KB と小さいがフル連想性を備え，128 B のキャッシュラインを 16 個保持でき，また 2 つの L2 間はスヌープによるコヒーレンシがある．L3 の実装は 4 MB の eDRAM により，2 つの CPU コアで共有される．メモリは DDR コントローラがチップ上にあり，チップあたり 512 MB の主記憶に接続される．

また，トーラス，グローバルツリー，グローバルイ

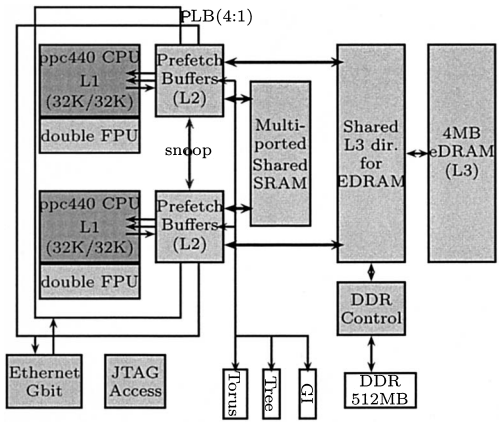

図 1 Blue Gene の専用 ASIC プロセッサ
Fig. 1 ASIC processor of Blue Gene.

ンタラプトなどの通信のハードウェアは，32 ビットのメモリ空間にメモリマップされており，CPU からアクセスすることができる．

2.2 ダブル FPU アーキテクチャ

PowerPC アーキテクチャを拡張して 2 重化された浮動小数点ユニットを持ち，32 要素（各要素は 64 ビット長）を持つレジスタファイルを 2 セット持つ．既存の PowerPC 浮動小数点演算命令は 1 次側のレジスタファイルに作用し，拡張された命令は 2 次側のみか，1 次側と 2 次側の両方に同時に作用する．1 次側と 2 次側で違う命令を同時に実行することはできず，1 次側と 2 次側に同時に作用する命令を使った場合のみに，双方の FPU が同時に動作する．

また，拡張されたダブル FPU の演算命令は倍精度浮動小数点演算のみである．PowerPC の単精度浮動小数点演算命令を使用する場合は 1 次側のみに作用する．単精度の演算は，ロード/ストア時に倍精度/単精度に変換する命令と，単精度への丸めを行う命令を組み合わせることで，ダブル FPU の命令を利用できる．

2 つの FPU を用い，積和演算を行った場合，1 プロセッサが 1 クロックで 4 浮動小数点演算を行うので，CPU あたりのピーク性能は動作周波数の 4 倍である 2.8 Gflop/s になる．追加された 72 命令の内訳はロード命令が 12，ストア命令が 13，レジスタ間の移動命令が 11，比較命令が 2，変換命令が 3，それに 32 の浮動小数点演算命令である．

2.2.1 ロード命令とストア命令

ダブル FPU の 2 つのレジスタファイルにデータをロードしたり，ストアしたりするために，ダブル FPU 用に拡張されたロード/ストア命令が用意されている．通常の PowerPC のロード/ストア命令は，1 次側のレ

☆ 計算ノードを 3 次元の格子状に並べ XYZ の 3 軸方向に隣接する計算ノードどうしを相互結合し，ドーナツのように端と端のノードどうしも相互接続したネットワーク網．

☆☆ キャッシュコヒーレンシとメモリ同期化命令（lwarx，stwcx）を持たない．

ジスタファイルのみに作用する．拡張されたロード/ストア命令は，1 次側，2 次側に同時に 2 つの値をロード/ストアするパラレルロード/ストア命令と，2 次側に作用するロード/ストア命令である．

パラレルロード/ストア命令は，メモリ内で連続したアドレスに格納されている 2 つの値を，1 次側と 2 次側のレジスタファイルに同時にロード，あるいは逆にストアするような命令である．倍精度浮動小数点数の配列に対して 2 つの値，16 B 分を，単精度浮動小数点数の配列に対して 2 つの値，8 B 分を，1 命令で 1 次側，2 次側に対して一度にロード/ストアすることができる．単精度の場合は，倍精度への変換が同時に行われる．ただし，このとき，ロード元，またはストア先の 2 つの値が，16 バイト境界をまたぐようなアドレスに配置されていないことが必要となる☆．倍精度の場合，16 バイトの倍数のアドレスから始まる 2 つの値に対してパラレルロード/ストアが適用できる．

また，パラレルロード/ストアには，ロード/ストア時に，1 次側と 2 次側で値を入れ替える命令が用意されている．また，1 次側，2 次側のレジスタ間で値をコピーしたり入れ替えたりする移動命令も用意されている．

2.2.2 ダブル FPU の浮動小数点演算命令

2 重化された浮動小数点レジスタファイルをクロスして使用することで，複素数演算などを高速に実行可能な豊富な命令セットが定義された[2]．2 つの FPU は**図 2** のように，1 次側と 2 次側のレジスタファイルから値を入れ替えて演算器に入力できるような仕組みを持つ．それぞれの FPU のレジスタファイルからの入力以外に，それぞれの FPU 間で入力をクロスできるような構成になっているので，単純に 2 つの FPU で同じ演算を行うだけではなく，片側のレジスタの値を共通して使用したり，値を 2 つの FPU 間で入れ替えたりといった演算命令が用意されている．演算器は通常の PowerPC の FPU と同じように，1 命令で積算と加算を行う積和演算を実行する．

ダブル FPU の演算命令は大きく分けて**表 1** のような 5 つに分類され，それぞれ，積和演算の形のものと，加減算のみ，積算のみの組合せがある．表 1 の A，C，B は図 2 中の A，C，B に対応し，A_p は 1 次側を，A_s は 2 次側を表す．

たとえば配列に対しての DAXPY の計算（$Y = aX + Y$）をダブル FPU を用いて処理する場合，ループ処理の前にあらかじめ 1 次側に a をロードしておき，ループ内の処理では X と Y の配列から 2 つずつパラレルロード命令を使って，1 次側と 2 次側にロードし，クロスコピー命令に分類される fxcpmadd 命令を使うことで，

$$Y_i = aX_i + Y_i$$
$$Y_{i+1} = aX_{i+1} + Y_{i+1}$$

の 2 つを同時に計算する．

このようにダブル FPU を使うことで，複素数演算でなくても，2 の倍数でアンロール可能なループ処理を効率良く処理することができる．

☆ 16 バイト境界をまたぐ場合もパラレルロード/ストアはできるが，1000 サイクルオーダのペナルティを受ける．

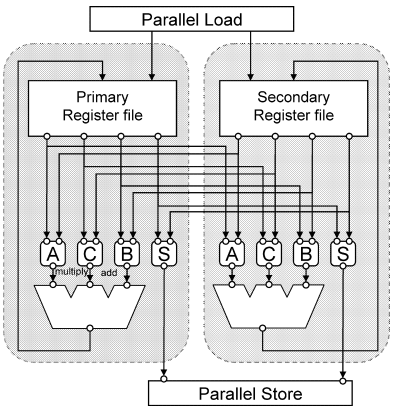

図 2 ダブル FPU
Fig. 2 Double FPU.

表 1 ダブル FPU 演算命令の分類
Table 1 5 types of double FPU instructions.

命令の分類	説明
パラレル命令	2 つの FPU で同じ演算を行う $A_p \times C_p \pm B_p$ $A_s \times C_s \pm B_s$
クロス命令	2 つの FPU で A の値を入れ替えて演算を行う $A_s \times C_p \pm B_p$ $A_p \times C_s \pm B_s$
クロスコピー命令	2 つの FPU で共通の A の値を使用する $A_p\ (A_s) \times C_p \pm B_p$ $A_p\ (A_s) \times C_s \pm B_s$
非対称クロスコピー命令	クロスコピー命令の後に片方の結果を反転する $\pm(A_p\ (A_s) \times C_p \pm B_p)$ $\mp(A_p\ (A_s) \times C_s \pm B_s)$
複素演算命令	複素数演算のための特殊な命令 $\pm(A_s \times C_s \pm B_p)$ $\mp(A_s \times C_p \pm B_s)$

2.3 MPI の実装

Blue Gene のメッセージ交換ライブラリは MPICH2 を基礎にしている．通信ソフトウェアは，上位に MPI 層（レイヤ），下位にネットワークをアクセスするパケット層を備え，中間には 1 層のメッセージ層を持つ[3]．トーラス通信はメッセージ層を通して bgltorus

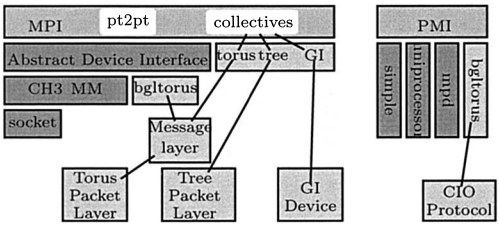

図 3 MPICH2 BG/L Roadmap
Fig. 3 MPICH2 BG/L Roadmap.

というデバイスとして，CH3 や MM の階層に並んで追加される（**図 3**）．また collective 通信用には，最適化されたパスを実装している．これも図 3 に示した．コミュニケータに COMM_WORLD を使用した場合の縮約通信では，ツリーのハードウェアが使用できる．

下位の相互結合網にアクセスするパケット層は，トーラス/ツリーデバイスの FIFO をアクセスして，初期化，パケット送信，パケット受信の 3 つの機能を実現する．機能はすべてノンブロッキングに行われ，パケット層は状態を持たない．

トーラスのパケットサイズは 32 B の倍数で，ヘッダ込みで 256 B までである．ヘッダは 16 B で，各パケットにはハードウェアが 14 B の CRC をつける．ヘッダには，データのサイズやあて先などの情報が記述され，計算ノードからは，XYZ 正負の 6 方向いずれかのトーラス FIFO にパケット送信すると，あて先へのルーティングはトーラスのハードウェアが自動的に処理をし，次々に隣接する計算ノードをたどってあて先に到着する．トーラスは XYZ 正負の 6 方向に隣接する計算ノードどうしを双方向に接続し，それぞれ独立に 1.4 Gbps の速度で通信できる．

ツリーのパケットサイズは 256 B で，各パケットにはハードウェアが 10 B の CRC をつける．ツリーのハードウェアは四則演算，論理演算の回路を持ち，縮約通信の演算を効率良く処理できる．

MPI 層とパケット層を結ぶのがメッセージ層である．これはユーザメモリ領域に置かれたメッセージをパケット化してパケット階層に置く仕事と，その逆（アンパケット化）を行う．

3. 格子 QCD プログラムの特徴

QCD（quantum chromodynamics，量子色力学）は強い相互作用の理論であり，核子などのハドロンを構成するクォークやグルーオンのダイナミクスを記述する．しかしこれを解析的に解くことは困難で，ハドロンの様々な性質を QCD から導くには，格子 QCD に基づいた数値シミュレーションが広く用いられている．

3.1 格子 QCD

格子 QCD 理論は 4 次元ユークリッド空間の格子上で定義され，経路積分を用いて量子化される[4)~6)]．数値シミュレーションでは，ゲージ場の配位 $\{U_\mu\}$ を Monte Carlo 法によって確率的に生成し，それらの配位の上で様々な相関関数を計算することによって，物理量を求めることが可能となる．

格子 QCD の基本的な自由度は，ゲージ場（グルーオン）とクォーク場である．ゲージ場は $SU(3)$ 群☆の元であり，3×3 の複素行列によって表され，4 次元ユークリッド格子の，格子点と格子点を結ぶ“リンク”の上に定義される（リンク変数とも呼ばれる）．クォーク場は格子点（サイト）上に定義され，3 成分の色の自由度と 4 成分のスピノルの自由度を持つ，$3\times 4 = 12$ 成分の複素ベクトルとして表される．

経路積分では，クォーク場は反交換関係を満たす数として定式化されるが，このような数を計算機上で直接用いるのは有効ではない．実際に様々な物理量を計算するために必要なのはクォークの伝播関数であるため，この伝播関数を求めることが基本的である．たとえばハドロンの質量は，クォークの伝播関数から構成されるハドロンの相関関数の，時間方向に対する減衰から求めることができる[6)]．伝播関数は，クォークの作用に現れるディラック演算子の逆行列である．格子上のディラック演算子としては，様々な形のものが提案されているが，本論文では最も基本的な形である，Wilson によって提唱されたもの[4)] を用いる．それ以外のディラック演算子に対しても，本論文で示した手法は適用できる．

伝播関数は CG 法のような反復法によって求める．反復法のカーネルである行列ベクトル積の部分は，ウィルソン・ディラック演算子とクォーク場ベクトルの積となる．格子 QCD シミュレーションでは，クォークの伝播関数の計算が計算時間のほとんどを占めるため，この演算が高速化の鍵となる．

本論文ではこのウィルソン・ディラック演算子を作用させる部分の高速化と並列化について報告する．反復法としては BiCGSTAB 法を用いた．

3.2 ウィルソン・ディラック演算子

ウィルソン・ディラック演算子を示す．

$$D(n,m) = \delta(n,m) - \kappa\sum_{\mu=1}^{4}\Big\{(1-\gamma_\mu)U_\mu(n)\delta(n+\hat{\mu},m) + (1+\gamma_\mu)U_\mu^\dagger(n-\hat{\mu})\delta(n-\hat{\mu},m)\Big\} \quad (1)$$

☆ SU(3) は 3 次元の特殊ユニタリー群．

$n=(n_x,n_y,n_z,n_t)$ は 4 次元格子上の格子点を表す．格子間隔 a は 1 ととったので n_μ は整数となる．$\hat{\mu}$ は μ-方向への単位ベクトル（$\mu=x,\ y,\ z,\ t$）である．κ はある実数のパラメータであり，クォークの質量と関係する．ゲージ場を表す行列 $U_\mu(n)$ は 3×3 の複素数行列で，クォーク場 ψ を格子間隔 a だけ移動させる平行移動子であり，ψ の色の自由度部分に対して作用する．γ_μ は 4×4 の複素数スピノル行列である．$1+\gamma_\mu$ および $1-\gamma_\mu$ は，ψ のスピノル自由度に対して作用し，4 成分スピノルを 2 成分スピノルに射影する．

クォークの伝播関数 $S_q(n,0)$ は，線型方程式

$$\sum_m D(n,m)S_q(m,0)=\delta(n,0) \tag{2}$$

を解いて得られるが，反復法ではこれを $v(n)=\sum_m D(n,m)w(m)$ という演算を繰り返して解く．

$D(n,m)$ の $w(m)$ に対する作用は，点 n に対し $\pm\mu$ 方向に隣接するサイトの w に対する演算からなる．$+x$ 方向を例にとる．係数を除けば，次のような演算を行うことになる．

$$v(n)^j_a=\sum_{b=1}^{3}\sum_{k=1}^{4}(1-\gamma_x)^{jk}U_x(n)_{ab}w(n+\hat{x})^k_b \tag{3}$$

ここでは式 (1) では省略した添字を省略せずに書いた．$a,\ b$ (= 1–3) は色の自由度を表し，$j,\ k$ (= 1–4) はスピノルの自由度を表している．

$(1-\gamma_x)$ と $U_x(n)$ は作用する空間が異なるため交換可能であるので，まず $(1-\gamma_x)$ を作用させる．$w(n+\hat{x})$ を色の 3 成分ごとにまとめた小ベクトルを w_j (j = 1–4) とすれば，スピノル空間の作用は次のように表される．

$$\begin{pmatrix}w'_1\\w'_2\\w'_3\\w'_4\end{pmatrix}=\begin{pmatrix}1&0&0&I\\0&1&I&0\\0&-I&1&0\\-I&0&0&1\end{pmatrix}\begin{pmatrix}w_1\\w_2\\w_3\\w_4\end{pmatrix} \tag{4}$$

ここで I は対角成分が i である 3×3 の複素小行列である．実際にはこのような 16×16 行列を定義してベクトルに掛けるのではなく，$w'_1=w_1+Iw_4$ のように成分ごとに求める．$w'_4=-Iw'_1$，$w'_3=-Iw'_2$ に注意すれば，左辺は 2 成分スピノル分の自由度（サイトあたり 6 成分複素ベクトル）しか持っていないので，$(1-\gamma_x)$ が射影行列となっていることが分かる．したがって，色空間に作用するゲージ場行列 $U_x(n)$ は w'_1，w'_2 に作用させるだけでよく，その後で 4 成分スピノルの形（式 (3) の左辺）に戻す．並列化により通信が必要な場合も，この 2 成分スピノルの形で転送する．このような式 (3) に対応する演算を，uxp という関数によって行うものとする．他の方向に関しても同様な構造であり，対応して関数 uxm（$-x$ 方向），uyp，uym，uzp，uzm，utp，utm が定義される．

4. Blue Gene における格子 QCD の並列化

本章ではじめに格子 QCD のウィルソン・ディラック演算子について，ダブル FPU 命令セットを用いて複素数の演算を行う方法について説明し，次に Blue Gene における格子 QCD の並列化について説明する．

4.1 ダブル FPU によるウィルソン・ディラック演算子の最適化

本論文では，すべての演算について最適なダブル FPU 命令を利用し，可能な限り，ロード/ストアのタイミングや，データ通信のタイミングを考慮したパイプラインのスケジューリングを行うために，C++ からインラインアセンブラを利用してコーディングを行った．ループ処理や関数の記述，データ構造などの定義は C++ 言語で記述した．

ダブル FPU 命令セットを用いて複素数の演算を行うには，複素数の配列は，実数部，虚数部の順に複素数が連続するものを使用する．つまり，C++ では，double _Complex で定義される配列を用いる．このような配列を用いることで，ダブル FPU セットのパラレルロード/ストア命令を用いて，1 つの複素数を 1 つのロード/ストア命令で，同時にロード/ストアすることができる．ダブル FPU を用いて複素数の演算を行う場合，1 次側 FPU が実数部の演算を，2 次側 FPU が虚数部の演算を行う．

格子 QCD のウィルソン・ディラック演算において，最も計算量を必要とする演算は，3×3 のゲージ場行列と，クォーク場スピノルの 3 成分小ベクトルの乗算である．式 (1) にあるように，すべての格子点について，4 次元 xyzt の正負の方向計 8 方向に隣接する格子点についてこの乗算を行う．このとき，負の方向の格子点については，ゲージ場行列のエルミート共役行列を乗じる．

正の方向の格子点について，ゲージ場行列 U と，クォーク場スピノルの 3 成分小ベクトル w の乗算は次のような演算である．

$$\begin{aligned}v_1&=U_{1,1}*w_1+U_{1,2}*w_2+U_{1,3}*w_3\\v_2&=U_{2,1}*w_1+U_{2,2}*w_2+U_{2,3}*w_3\\v_3&=U_{3,1}*w_1+U_{3,2}*w_2+U_{3,3}*w_3\end{aligned}$$

それぞれの要素は複素数であるので，9 個の複素数の

乗算および，6 つの複素数の加算で表現される．さらに，複素数の乗算は 4 つの実数の乗算および 2 つの加算からなり，この処理は 66 回の実数の演算からなる．

ここで，複素数の乗算，$U_{1,1} * w_1$ をダブル FPU 命令セットを用いて実行すると，

fxpmul $v_1, U_{1,1}, w_1$
fxcxnpma $v_1, U_{1,1}, w_1, v_1$

という 2 つの命令で計算することができる．ここで，複素数の乗算 $U_{1,1} * w_1$ を次のように書くとする．

$\mathrm{Re}(v_1) = \mathrm{Re}(U_{1,1}) * \mathrm{Re}(w_1) - \mathrm{Im}(U_{1,1}) * \mathrm{Im}(w_1)$
$\mathrm{Im}(v_1) = \mathrm{Re}(U_{1,1}) * \mathrm{Im}(w_1) + \mathrm{Im}(U_{1,1}) * \mathrm{Re}(w_1)$

“fxpmul $v_1, U_{1,1}, w_1$” は，クロスコピー命令であり，1 次側 FPU のレジスタの値，$U_{1,1}$ の実数部を，複素数 w_1 に乗じ，複素数 v_1 に代入するという演算で，

$\mathrm{Re}(v_1) = \mathrm{Re}(U_{1,1}) * \mathrm{Re}(w_1)$
$\mathrm{Im}(v_1) = \mathrm{Re}(U_{1,1}) * \mathrm{Im}(w_1)$

の演算に相当する処理を 1 つの命令で実行する．また “fxcxnpma $v_1, U_{1,1}, w_1, v_1$” は，複素命令であり，$U_{1,1}$ の虚数部を複素数 w_1 の実数部と虚数部を入れ替えたものに乗じ，1 次側 FPU の値，実数部の符号を反転してから，複素数 v_1 に加算して代入する演算で，

$\mathrm{Re}(v_1) = \mathrm{Re}(v_1) - \mathrm{Im}(U_{1,1}) * \mathrm{Im}(w_1)$
$\mathrm{Im}(v_1) = \mathrm{Im}(v_1) + \mathrm{Im}(U_{1,1}) * \mathrm{Re}(w_1)$

を 1 命令で実行する．

同様に v_1 の計算の残りの “$+U_{2,1} * w_2 + U_{3,1} * w_3$” については，fxpmul 命令を積和演算の形にした fxcpmadd 命令を用いて次の 4 命令で演算する．

fxcpmadd $v_1, U_{2,1}, w_2, v_1$
fxcxnpma $v_1, U_{2,1}, w_2, v_1$
fxcpmadd $v_1, U_{3,1}, w_3, v_1$
fxcxnpma $v_1, U_{3,1}, w_3, v_1$

よって，3×3 のゲージ場行列と 3 成分のクォーク場スピノルの乗算は，66 回の実数の演算を 18 個のダブル FPU 命令で処理することが可能である．

同様に，負の方向の格子点について，ゲージ場行列のエルミート共役行列 $\bar{U}$ と，クォーク場スピノル w の乗算は，次のような演算である．

$v_1 = \bar{U}_{1,1} * w_1 + \bar{U}_{1,2} * w_2 + \bar{U}_{1,3} * w_3$
$v_2 = \bar{U}_{2,1} * w_1 + \bar{U}_{2,2} * w_2 + \bar{U}_{2,3} * w_3$
$v_3 = \bar{U}_{3,1} * w_1 + \bar{U}_{3,2} * w_2 + \bar{U}_{3,3} * w_3$

このとき，共役複素数と複素数の乗算 $\bar{U}_{1,1} * w_1$ は，次の 2 つのダブル FPU 命令によって計算できる．

fxpmul $v_1, U_{1,1}, w_1$
fxcxnsma $v_1, U_{1,1}, w_1, v_1$

2 つの複素数の乗算の場合と 2 つ目の命令が違うだけである．

共役複素数と複素数の演算は，

$\mathrm{Re}(v_1) = \mathrm{Re}(U_{1,1}) * \mathrm{Re}(w_1) + \mathrm{Im}(U_{1,1}) * \mathrm{Im}(w_1)$
$\mathrm{Im}(v_1) = \mathrm{Re}(U_{1,1}) * \mathrm{Im}(w_1) - \mathrm{Im}(U_{1,1}) * \mathrm{Re}(w_1)$

であるので，2 つ目の命令において，符号を反転するのを虚数部にするだけの違いである．同様に，残りの部分についても，積和演算を用いて，正の方向の場合と同じく，全部で 18 個のダブル FPU 命令で計算することができる．

ウィルソン・ディラック演算において，これらのゲージ場行列とスピノルの色自由度の乗算に加えて，スピノル自由度に対する演算（式 (4)）がある．この演算は，4×3 成分スピノルを 2×3 成分スピノルに射影する部分と，その 2×3 成分スピノルにゲージ場行列を作用させた後で，4×3 成分スピノルの形に戻す部分からなる．

これらの演算はそれぞれ，複素数の加減算または積和演算で実行することができ，前後の処理ともに，クォーク場スピノルそれぞれの成分について 1 つのダブル FPU 命令で計算できる．

4.2 3 次元トーラス網を用いた格子 QCD の並列化

Blue Gene の計算ノードは 3 次トーラス網で接続されている．つまり各計算ノードは，XYZ 軸方向の 3 方向の正負で合計 6 方向について，隣接する計算ノードと双方向で接続されており，物理的に一番端にある計算ノードは，同じ軸方向の反対側の端にあるノードと接続され隣のノードとして扱うことができる．以下，ハードウェアの構成としての方向を XYZT で表し，プログラムが扱っている問題（格子 QCD）における方向を xyzt で表す．

このように計算ノードが 3 次元トーラス網で接続されていると，構造格子で離散化した 3 次元空間に周期境界的であるような問題を解く場合に，単純に構造格子を xyz 方向にそれぞれ等分することで，効率良く並列化を行える．格子 QCD もそのような問題であり，構造格子を等分し，それぞれ隣接ノードに対して，境界部分のデータをやりとりすることで並列化が可能である．

ただし，格子 QCD は 4 次元の問題であるので，工夫が必要である．Blue Gene の計算ノードには 2 つの CPU が載っており，仮想ノード・モードで実行する場合，同一チップ内の 2 つの CPU 間で仮想的な 4 次元目（T）の方向のトーラスを形成していると考えることができる．したがってミッドプレーンの場合，計算ノード単位ではなく，CPU 単位で考えると $8 \times 8 \times 8 \times 2$ の 4 次元トーラスを利用できる．

我々は，4 次元目の T 方向に，格子の x 方向を割り

当てることにした．格子は，3×3 のゲージ場行列および，4×3 成分のクォーク場スピノルの 4 次元配列からなり，これらの配列は，xyzt の順にデータが格納される．つまり配列のデータを処理する際のループは外から tzyx の順になり，x 方向の処理はループの一番内側となる．同一計算ノード内の 2 つの CPU 間におけるデータ交換は，双方の CPU から参照可能なメモリ空間を介して行うことができるため高速であり，ループの一番内側でノード間通信が発生しないことで速度を稼げると判断したからである．残りの格子方向の yzt は，それぞれ 3 次元トーラスの XYZ に割り当てることで，各計算ノードは，3 次元トーラス網で物理的に隣接する計算ノードとのみのデータ交換で処理できるようになり，他の通信と経路が重なるようなこともなく，3 次元トーラス網を最大限に活かすことが可能となる．

たとえば，解こうとする格子 QCD の問題の 4 次元格子の大きさが $48 \times 48 \times 48 \times 48$ の場合，これをミッドプレーンで処理する場合，各 CPU 上の 4 次元格子の大きさは，$24 \times 6 \times 6 \times 6$ という大きさになる．この問題の場合，各 CPU の 4 次元格子の大きさを $24 \times 3 \times 3 \times 3$ までさらに分割していくことが可能で，ミッドプレーン 8 個分，つまり 4 筐体（8192 CPU）まで並列化が可能である．

4.3 トーラスパケット通信によるデータ交換

ウィルソン・ディラック演算子は，式 (1) のように，各格子点から隣接する 8 つの格子点のデータを参照して演算を行う．並列化によって格子を分割すると，各計算ノードに割り当てられた部分格子の端の格子点から元々隣にあった格子点を参照する場合に，部分格子に隣の格子点を含む計算ノードとデータ通信を行う必要が生じる．

このとき，4.2 節で述べたように，仮想的な 4 次元トーラスに格子を割り当てることで，格子の yzt 方向の隣の格子点は，必ず隣接する計算ノードに存在するので，隣の計算ノードとの通信のみでデータ交換を処理できる．また，x 方向の隣の格子点は，同一計算ノード内の別の CPU が持っているので，ここではデータ通信は必要ない．

ところで，すでに示したように，スピノル空間の行列（$1 \pm \gamma_\mu$）が射影行列であるという性質を利用すると，境界部分において隣の計算ノードに送信するのは，2×3 成分のスピノルデータ，つまり，6 個の複素数，96 B と小さなデータで済むことになる．

Blue Gene は MPI を利用してノード間のデータ交換を行うことができるが，2.3 節で述べたように，MPI の実装は階層が深く，小さなデータを頻繁に送受信すると，実際のデータ送受信にかかる時間に比べて，メッセージ層のパケット化，アンパケット化のように MPI 関数内の処理時間が大きくなり，パフォーマンスの低下を引き起こす．

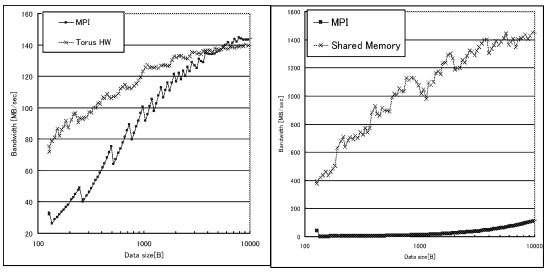

図 4 MPI とトーラスパケット通信の通信速度比較
Fig. 4 Comparison of bandwidth: MPI vs. low level communication.

これを防ぐためには，なるべく大きなデータを一度に送るようにすればよいが，境界部分の計算結果を大きなバッファに溜め込んで一度に送受信しようとすると，せっかく 1 つのデータは小さく，1 次キャッシュに楽々収まるようなものであるのに，計算結果をバッファに保存したり，バッファから読み込んだりするためのメモリアクセスが増えて，パフォーマンスを低下させる原因になってしまう．

また，仮想ノード・モードを用いて MPI 通信を行う場合，各 CPU がそれぞれ MPI 通信の処理を行う必要があるため，非同期の送受信を行うことができない．もともと Blue Gene のトーラスのハードウェア自身は XYZ 正負の 6 方向に同時にデータを送信できるので，これを直接利用できれば，通信時間を重ねることで短縮できる．

そこで，我々は MPI によるデータ送受信の代わりに，トーラスのハードウェアを直接利用してデータ交換を行うことで，通信時のパフォーマンスの低下を最小限に抑えることにした．

今回のケースのように，隣接する計算ノードとの通信や，同一計算ノード内の CPU 間の通信に限った条件で，MPI による通信と，トーラスパケット通信を直接利用する場合のデータの転送速度の比較を**図 4** に示す．隣接ノードへの通信の場合，格子 QCD の場合のように 96 B と小さなデータの通信では，トーラスパケット通信が 3 倍程度高速である．また，同一計算ノード内での転送速度は，MPI の場合，隣接計算ノードへの転送と同じ程度しか出ておらず，MPI 処理のオーバヘッドの大きさがよく分かる．

トーラスパケット通信は，各計算ノードからトーラ

スの XYZ 軸の正負，計 6 方向の隣接計算ノードに対してパケットを非ブロッキング送信することができる．ただし，トーラス FIFO の容量は 1 KB と小さく，ここに収まる範囲内で非ブロッキング送信を行わなければならない．

トーラスパケット通信を利用してデータを送信するには，あて先やサイズを入れたパケットヘッダを用意し，メモリ空間にマッピングされたトーラス FIFO にパケットヘッダに続いてデータを書き込めばよいが，隣の計算ノードと通信する場合，送りたい方向の FIFO に書き込むだけでよい．受信も同様に XYZ 正負の 6 方向に対応したメモリ空間にマッピングされた FIFO からデータを読み込むことで受信ができる．

トーラスパケットのサイズは 256 B 以下で，これにはヘッダとフッタ計 32 B が含まれており，1 つのパケットで送信できるデータは 224 B までである．ウィルソン・ディラック演算において，境界部分の格子点の計算を行うときに，前述のように隣の計算ノードと 2×3 成分のスピノル，96 B のデータを送受信するので，トーラスパケット 1 つで 2×3 成分のスピノルを送受信することが可能である．

また，データを隣の計算ノードに送る前に，2×3 成分スピノルを作る計算を行うが，このときの計算結果をレジスタから直接 FIFO に書き込むことが可能であるので，計算が終わるとただちに送信することができるのも，MPI に比べてメモリアクセスを減らせる利点である．同様に受信するときも，FIFO からレジスタにロードしただちに次の計算に移ることができる．

このとき，送信は非ブロッキングであるので，FIFO にデータを書き出した後に他の計算などの処理や，他の方向へのデータの送信を重ねて行うことができる．これをうまく利用して，データの送信時間に別の処理を重ねると，通信時間の見た目を短縮できる．

なお，データの送信は 6 方向同時に行えるが，2 つの CPU で共有されるため，片方の CPU は XYZ の正の方向，もう一方の CPU は XYZ の負の方向に，というように 3 方向ずつ同時にデータを送信することにした．また，格子の x 方向は，2 つの CPU 間の共有のメモリ領域に対して，ノード間通信と同様に，2×3 成分のスピノルデータをレジスタから直接書き込み，2 つの CPU 間でバリア同期をとったうえで，相手のデータをレジスタに読み込むことで，データ交換が行える．

表 2 Blue Gene における格子 QCD プログラムの実効性能（対ピーク性能比）
Table 2 Performance of lattice QCD program on Blue Gene.

格子サイズ	1,024 CPU	2,048 CPU
$16 \times 16 \times 16 \times 32$	24.33%	22.78%
$24 \times 24 \times 24 \times 48$	29.23%	28.57%

5. Blue Gene における格子 QCD プログラムの実行結果

5.1 Blue Gene における測定

本論文で述べた Blue Gene に最適化した格子 QCD プログラム（ウィルソン・ソルバ）を用い，プログラム実行の経過時間を測定し，アルゴリズムの理論的な浮動小数点演算回数から実効性能を求めた．4 次元格子の大きさは，$16 \times 16 \times 16 \times 32$ および，$24 \times 24 \times 24 \times 48$ の 2 種類を用い，Blue Gene の 1 ミッドプレーン（1,024 CPU）および，1 筐体（2,048 CPU）を使い，仮想ノード・モードで実行した．

1 ミッドプレーンで実行する場合，仮想的な 4 次元トーラスは $2 \times 8 \times 8 \times 8$ であり，各 CPU の 4 次元格子サイズは，それぞれ，$8 \times 2 \times 2 \times 4$，$12 \times 3 \times 3 \times 6$ である．1 筐体で実行する場合，仮想的な 4 次元トーラスは $2 \times 8 \times 8 \times 16$ であり，各 CPU の 4 次元格子サイズは，それぞれ，$8 \times 2 \times 2 \times 2$，$12 \times 3 \times 3 \times 3$ である．

格子 QCD プログラムは，問題の格子データをファイルから読み込んで使用する．あらかじめ各 CPU 用にデータを分割しておき，各 CPU が実行時にそれぞれ専用のデータを読み込むようにした．このとき，ファイル入出力にかかる時間は実行時間の測定には含めず，ゲージ場行列を全 CPU が読み込んだ後に，全 CPU でバリア同期をとってから時間計測を開始し，すべての処理が終わってからバリア同期をとり時間計測を終了して経過時間を求めた．**表 2** に，各格子サイズについて，1 ミッドプレーンおよび，1 筐体で実行したときの，実効性能をまとめる．

比較のために，$24 \times 24 \times 24 \times 48$ の格子について，1 ミッドプレーンを使用し，4.3 節で述べたトーラスパケット通信を使用せずに MPI 通信を使いデータ交換を行ったところ，実効性能は，17.88%であった．ただし，MPI による実装の場合は，隣に送るべきデータをバッファに溜め込んで一度に送受信を行う方法をとった．

5.2 実効性能についての考察

次に $24 \times 24 \times 24 \times 48$ の格子について，1 ミッド

表 3 全経過時間に対する処理の占める割合
Table 3 Percentage of processes in lattice QCD.

処理内容	処理の占める割合
ウィルソン・ディラック演算	75.14%
反復法の代数演算	23.02%
反復法の全加算通信	1.84%

表 4 ウィルソン・ディラック演算子の理論的な実効性能の限界値
Table 4 Logical limitation of performance of Wilson-Dirac operator.

格子サイズ	1024 CPU	2048 CPU
$16 \times 16 \times 16 \times 32$	32.2%	31.9%
$24 \times 24 \times 24 \times 48$	36.2%	35.9%

プレーンを使用した場合に，格子 QCD プログラムの処理別に全経過時間との比を測定した．処理別に経過時間の占める割合を**表 3** に示す．

このときの割合を用いて，格子 QCD のカーネルであるウィルソン・ディラック演算についての実効性能は，実行時間と論理的な計算量から計算すると，32.60%になる．

全体としての実効性能は 30%を切っているが，これは BiCGSTAB 法の処理によって，全体の実効性能が若干落ちているためである．この部分についてもダブル FPU を使って計算処理の最適化を行っているが，ウィルソン・ディラック演算子部分ほど理論的な実効性能が高くない処理が多いのと，全加算通信が含まれるのが原因である．

ここで，ミッドプレーンで $24 \times 24 \times 24 \times 48$ の格子を用いた場合，ウィルソン・ディラック演算子 1 回分の CPU あたりの計算量は約 0.93 Gflops で，このとき最適化されたダブル FPU 命令の数は約 280 K 命令であり，約 83%の実効性能になる．しかしながら，この性能はデータのロードおよびストアにかかる時間，および，計算ノード間データ通信にかかる時間を考慮に入れなかった場合なので，実効の限界はもっと低くなる．

上の条件で，ウィルソン・ディラック演算子において，ロードすべきデータの量は約 1.8 MB，ストアするデータ量は約 497 KB になる．これらをすべてダブル FPU 命令のパラレルロード/ストアを用いるとき，キャッシュメモリを有効に使えていると仮定し，複素数 1 個あたり 4 サイクルでロード/ストアできるとすると，ロード/ストアには約 562 K サイクルかかる．

隣の計算ノードとの通信時間は，トーラスパケット通信を利用すると，トーラスの FIFO とのアクセス時間を除いて，1 バイトのデータを送信するのに 4 サイクルかかる．上の条件では，格子の yz 方向に 55 KB，格子の t 方向に 28 KB のデータを送信する必要がある．このときに必要な通信時間は，それぞれ，221 K サイクル，および 111 K サイクルである．これら 3 方向への通信は並列に行えるので，最大のサイクル数である 221 K サイクルが通信時間として見えてくるが，ウィルソン・ディラック演算子に必要な命令数 280 K サイクルおよび，データのロード/ストアに必要な 562 K サイクルよりも小さいので，理論的には，通信時間は計算時間で隠すことができるので，データ通信自体にかかる時間は無視することにする．

しかし，FIFO のアクセス時間☆を考慮すると，3 方向分合計で約 52 K サイクルかかる．また，格子の x 方向のデータ交換には約 31 K サイクルを必要とする．

このとき，全体でデータアクセスに必要なサイクル数は，約 645 K サイクルで，PPC440 はロード/ストア命令は，演算命令と同時実行できるので，演算命令の時間はデータアクセスの時間に隠れ，約 645 K サイクルが実際に見えてくる処理時間となる．この場合，実効性能は約 36.2%となる．しかし，実際にはすべての演算命令とロード/ストア命令を同時に実行できるようにコーディングするのは困難であり，その分実際の実効性能が低くなっていると考えられる．表 2 のそれぞれの条件について，同様に計算した理論的な実効性能を，**表 4** にまとめる．

6. おわりに

Blue Gene におけるアプリケーションの並列化と最適化として，格子 QCD プログラムを取り上げ，格子 QCD のウィルソン・ディラック演算子について，ダブル FPU を用いて複素数演算を最適化した．また，仮想的な 4 次元トーラス網を利用して並列化し，MPI の代わりにトーラスパケット通信を直接利用することで，高い並列度を実現した．実際の実行結果では 2048 ウェイという高い並列度においても Blue Gene の性能が活かされており 30%近くの実効性能を得ることができた．

また，格子 QCD のように，ダブル FPU 命令を効率良く利用でき，隣接ノードとの通信が大半を占めるような問題に，本論文で述べたような最適化手法が適用でき，高い実効性能と並列度を実現できると考えられる．

☆ データのロード/ストアに必要な時間

参 考 文 献

1) Gara, A., et al.: Overview of the blue gene/l system architecture, *IBM J. Res. Develop.*, Vol.49, No.2/3, pp.195–212 (2005).
2) Chatterjee, S., et al.: Design and exploitation of a high-performance simd floating-point unit for blue gene/l, *IBM J. Res. Develop.*, Vol.49, No.2/3, pp.377–392 (2005).
3) Almási, G., et al.: Design and implementaion of message-passing services for the blue gene/l supercomputer, *IBM J. Res. Develop.*, Vol.49, No.2/3, pp.393–406 (2005).
4) Wilson, K.G.: *Phys. Rev.*, d10 (1974).
5) Wilson., K.G.: *New Phenomena in Subnuclear Physics*, Zichini, A. (Ed.), Plenum Press, New York (1977).
6) 岩崎洋一：物理学最前線 11，格子ゲージ理論，共立出版 (1985).

(平成 17 年 10 月 4 日受付)
(平成 18 年 1 月 31 日採録)

土井 淳（正会員）

1999 年早稲田大学大学院理工学研究科修了．1999 年日本アイ・ビー・エム（株）に入社．現在日本アイ・ビー・エム（株）東京基礎研究所に勤務．専門分野はコンピュータグラフィクス，CAD，モデリング，可視化，および，数値計算アプリケーションの最適化．

寒川 光（正会員）

1972 年早稲田大学理工学部機械工学科卒業．同年日本ユニバック（株）入社．1984 年日本アイ・ビー・エム（株）入社．現在東京基礎研究所勤務．数値解析，数値計算法，計算機アーキテクチャに関する仕事に従事．工学博士．計算工学会，日本シミュレーション学会各会員．

松古 栄夫

高エネルギー加速器研究機構計算科学センター助手．広島大学大学院修了，博士（理学）．専門は素粒子・ハドロン物理学の理論的研究，特に格子 QCD シミュレーション．

橋本 省二

高エネルギー加速器研究機構・素粒子原子核研究所・助教授．専門は素粒子物理学の理論的研究，特に格子ゲージ理論の数値シミュレーション．