情報処理学会論文誌　コンピューティングシステム　Vol. 2　No. 2　144–151 (July 2009)

Blue Gene/P における第一原理分子動力学ソフトウェア PHASE の最適化

今井晴基[†1]　森山孝男[†1]

IBM Blue Gene®/P は，IBM Blue Gene® ソリューションの最新機種であり，ノード内 4 コアでのメモリ共有が可能となるなどいくつかの機能向上が図られている．本論文では，この Blue Gene/P において第一原理分子動力学ソフトウェア PHASE の最適化を行った．単体ノード性能の向上のために，キャッシュブロッキングのブロックサイズの選定，SIMD 演算命令利用のためのコード修正，FFT ライブラリの置き換えを行った．また，スケーラビリティ向上のため，MPI と OpenMP のハイブリッド並列化を行った．オリジナルコードに対して約 1.8 倍の単体ノード性能の向上を実現し，また，ハイブリッド並列化によりスケーラビリティが向上することを確認した．

The First Principles Molecular Dynamics Software PHASE on the Blue Gene/P

Haruki Imai[†1] and Takao Moriyama[†1]

The IBM Blue Gene®/P system is the latest machine in the IBM Blue Gene® family. One of its new functions makes it possible to share memory among the 4 cores within a node. In this paper, we optimized PHASE, a first principles molecular dynamics system for the Blue Gene/P. To improve single node performance, we used cache blocking, modified the code to generate SIMD instructions, and replaced the FFT library. To improve the scalability, we used hybrid parallelization using MPI and OpenMP. As a result, we improved the performance about 1.8-fold compared with the original code performance and achieved good scalability with hybrid parallelization.

†1 日本アイ・ビー・エム株式会社東京基礎研究所
IBM Tokyo Research Laboratory

1. はじめに

従来の CMOS スケーリング技術による単体プロセッサの性能向上の限界が近づく状況においても，システム全体の性能向上を実現するために並列計算機が広く利用されている．理化学研究所が中心となって行っている「次世代スーパーコンピュータの開発と利用プロジェクト」（以下，次世代スパコンプロジェクト）[1] においても超並列計算機は必要不可欠なものとなっている．このような超並列計算機を有効活用するためには，アプリケーションの開発段階からその並列性などを検討する必要がある．東京大学生産技術研究所が中心となって行われた革新的シミュレーションソフトウェアの研究開発プロジェクト（RSS21）[2] では，それらを考慮しつつ，大規模計算を行う実用的なソフトウェアの開発が行われてきた．開発されたソフトウェアの 1 つとして第一原理分子動力学ソフトウェアの PHASE がある．PHASE は，SC07（the International Conference for High Performance Computing, Networking, Storage and Analysis）において，地球シミュレータでの性能報告が Gordon Bell 賞のファイナリストに選ばれ[3]，また，次世代スパコンプロジェクトにおいても 21 本のターゲットアプリケーションソフトの 1 つとして選定されるなど[1]，国内外で評価されているアプリケーションである．また，小松ら[4] により，さらなる超並列化への取り組みも検討されている．超並列計算機の 1 つに IBM Blue Gene® ソリューション（以下，Blue Gene）がある．Blue Gene は，Top 500 リスト（2008 年 6 月版）[5] において，Top10 のシステムのうち 4 つを占めるなど，超並列計算の有力なソリューションの 1 つであり，超並列計算のプラットフォームとして，実際にさまざまな分野のアプリケーションの動作実績とその最適化の報告がなされてきた[6),7]．その最新機種である Blue Gene®/P（以下，BG/P）[8] は Blue Gene®/L（以下，BG/L）の後継機種として多くの設計思想を受け継ぎ，プロセッサ動作周波数やメモリシステムなどにいくつかの機能向上が図られたシステムである．本論文では，この BG/P における PHASE の最適化方法，および，その結果を報告する．以下，2 章では，BG/P のアーキテクチャについて解説する．3 章では，PHASE について，4 章では BG/P における一般的な最適化手法に関して述べる．5 章では，それらの手法の PHASE への適用方法と実際の実行結果から性能向上について考察する．6 章で本論文のまとめを述べる．

2. Blue Gene/P のアーキテクチャ

BG/P は，BG/L の後継機種としてその多くの設計思想を引き継いでいる．**表 1** に BG/P

表 1　BG/L と BG/P のハードウェア構成比較
Table 1　Feature comparison between the BG/L and BG/P systems.

	BG/L	BG/P
Node		
Processor	PowerPC440	PowerPC450
Frequency	700 MHz	850 MHz
#Cores	2	4
L1 cache (private)	32 KB	32 KB
L3 cache (shared)	4 MB	8 MB
Main memory size	512 MB/1 GB	2 GB
Memory bandwidth	5.6 GB/s	13.6 GB/s
Network		
Bandwidth (Torus)	2.1 GB/s	5.1 GB/s
	(175 MB/s × 12)	(425 MB/s × 12)
Bandwidth (Tree)	700 MB/s	1.7 GB/s
	(350 MB/s × 2)	(850 MB/s × 2)

と BG/L の主なハードウェア構成を示す．プロセッサは，Power PC440（以下 PPC440）から Power PC450（以下 PPC450）になり，そのプロセッサ周波数は 700 MHz から 850 MHz に増強され，ノード内のプロセッサコア数は倍の 4 つになった．それにともない，メインメモリ，および L3 キャッシュサイズもそれぞれ倍になり，また，プロセッサ周波数あたりのメモリバンド幅とノード間のネットワークバンド幅も倍になっている．BG/P のノードあたりのピーク性能は 13.6 G FLOPS であり，1 筐体 1,024 ノード（4096 コア）では約 14 T FLOPS となる．これは，BG/L の 1 筐体に対して 2.43 倍のピーク性能である．BG/P は電力あたりの演算性能が優れていることにも特徴があり，Green500 リストの上位にも位置している[9]．BG/L ではノード内の 2 つのプロセッサコアはコヒーレントなメモリ共有を行うことができなかったが，BG/P ではキャッシュコヒーレンスが実装され，ノード内の 4 つのコアは 4 ウェイの SMP 構成をとることで共有メモリ並列で実行可能となった．BG/P では，ノードの実行モードとして，4 つのコアがメモリを共有し，1 プロセス 4 スレッドで動作させる SMP モード，ノード内のメモリを 2 つに分割し，2 つのコアでメモリを共有し，2 プロセス 2 スレッドで動作させる DUAL モード，BG/L と同様に 1 コアそれぞれに 1 プロセスずつ，ノード内 4 プロセスで動作させる VN モードがある．ノード内の並列化には OpenMP やコンパイラによる自動並列化を利用可能である．これらのモードは実行ごとに指定可能であり，アプリケーションの並列化の特性や利用メモリサイズなどにより，最適な構成を選択することができる．ノード間のネットワークは BG/L 同様に，隣接ノード間の通信に利用される 3 次元トーラスネットワークとブロードキャスト通信に利用されるツリーネットワークから構成されているが，BG/P では各ノードに新たに DMA エンジンが搭載され，通信が DMA にオフロードされることでプロセッサの通信処理を軽減することができるようになった．PPC450 には，PPC440 同様にデュアル・パイプラインの浮動小数点演算ユニット（Dual FPU）が搭載されている．この Dual FPU では SIMD 型の浮動小数点演算を実行可能である[10]．BG/P でノード単体性能を引き出すためには，この SIMD 演算命令の効率的な利用が不可欠である．BG/P では各プロセッサに 256 個のパフォーマンス・カウンタが取り付けられており，計算量やキャッシュの使用状況など同時に 256 個のイベントを観測可能となっており，アプリケーションのプロファイリングに利用可能である．

3. PHASE

PHASE は，密度汎関数理論に基づく電子状態計算ソフトウェアである．擬ポテンシャル法の導入により平面波基底の数を減らすことで計算量の減少が図られている．ソフトウェアは RSS21 のソフトウェア公開サイト[2] からダウンロード可能である．また，PHASE で利用する擬ポテンシャルデータを作成するソフトウェア（CIAO）や入力データを作成する GUI 統合環境ソフトウェア（PHASE-Viewer）なども同サイトにおいて公開されており，実用性も考慮されたソフトウェアである[⋆1]．プログラムコードは主に Fortran90 で書かれており，MPI によりバンド数と k 点での並列化がなされている[3),4)]．さまざまなプラットフォームに対応するための Makefile や FFT ライブラリのラッパールーチンが用意されている．また，コード内，主に計算カーネル部分には，ディレクティブを挿入することで，各プラットフォームの最適化への対応も図られている．

4. Blue Gene/P における最適化

4.1 ノード単体性能の向上

一般にスカラ計算機におけるノード単体性能の向上には，キャッシュやパイプラインの利用効率に関する最適化が行われる．最適化の例としては，配列アクセスのストライドの最小化，キャッシュブロッキング，ループ展開によるスケジューリングなどがあげられる．これらの最適化の一部はコンパイラが行うが，すべてコンパイラでできるわけではなく，手

⋆1 PHASE，PHASE-Viewer は東京大学生産技術研究所が中心となって行った「革新的シミュレーションソフトウエアの研究開発プロジェクト」において開発されたソフトウェアである．以下のサイトからダウンロード可能である．http://www.ciss.iis.u-tokyo.ac.jp/rss21/result/download/index.php#download_4

```
real(8)  a, x(n), y(n)
call alignx(16, x(1))
call alignx(16, y(1)))
do i = 1, n
  y(i) = y(i) + a * x(i)
end do
```

図 1　DAXPY コード
Fig. 1　DAXPY code.

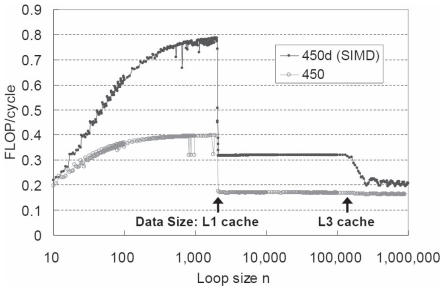

図 2　DAXPY の性能
Fig. 2　DAXPY perfornamce.

動のコード修正により対応する必要がある．BG/P もスカラ計算機であるため，ノード単体性能の向上のために，これらと同様の最適化を行う必要がある．代表的な計算カーネルである DAXPY（**図 1**）は 1 ループ内で倍精度の配列要素を 2 個参照する．ループサイズ n に対しては，$16n$ バイトの配列データが参照されるため，これらがキャッシュに収まるとき最も性能が良くなる．**図 2** に BG/P における DAXPY のループサイズ n と 1 サイクルあたりの実効 FLOP 数を示す．VN モード 32 ノードで実行し，ノード内で同時に 4 プロセス走らせたときの結果である．グラフの 450d は SIMD 演算命令を利用したときの結果，450 は利用しないときの結果である．図 2 を見ると，ループサイズが小さいときは，ループのオーバヘッドが大きいため，良い性能を得られていないが，ループサイズの増加にともない，2000 付近まで性能が向上している．ループサイズが 2000 のときの 2 つの配列サイズの合計は L1 キャッシュサイズの 32 KB とほぼ等しい．2000 を超えると L1 キャッシュからデータが溢れるため，パフォーマンスが急激に落ちてしまう．L3 キャッシュは各プロセスあたり 2 MB まで利用可能であるため，対応するループサイズ 125000 付近で L3 キャッシュのサイズを超えるとさらに性能が悪化する．BG/P のキャッシュの最適化においては，L1 キャッシュに収まるように配列サイズを調整することが重要である．

BG/L，BG/P は SIMD 型演算命令を実行可能であるため，ノード性能を向上させるためには，これらを効率的に利用する必要がある．以下，SIMD 演算命令を効果的に利用する方法を DAXPY の場合を例にして示す．この SIMD 命令のロード・ストア命令を実行するためには，キャッシュのアーキテクチャの制限からロード・ストア対象のアドレスが 16 バイト境界にアラインされている必要がある．しかし，コンパイラはすべてのアラインメントを解釈するのは困難である．そのため，ユーザが図 1 のようにアラインメントの指示文 alignx を利用することでコンパイラへの指示を行い，コンパイラが SIMD 命令を出力しやすくする必要がある．さらに intrinsic 関数と呼ばれる実行命令に対応した組み込み関数を利用することにより，複雑でコンパイラが解釈しにくい計算でも直接的に命令を生成することができる[11]（DAXPY の場合はコンパイラが SIMD 命令を出力するため intrinsic 関数を使う必要はない）．

4.2　スケーラビリティの向上

一般に超並列計算機でのスケーラビリティの向上には，並列度が増加しても通信量が増加しない計算アルゴリズムの選択，アムダールの法則に基づき，非並列化部分の少ない計算アルゴリズムの開発・実装が求められるが，ここでは，BG/P の特徴である．MPI と OpenMP によるハイブリッド並列化利用時のスケーラビリティ向上に関して検討を行う．一般に MPI プログラムは並列度が上がるとそれにともないプロセス間の通信量が増加し，スケーラビリティを阻害する．そのため，ノード内の OpenMP による並列化も組み合わせたハイブリッド並列化を利用することにより，プロセス数の増加にともなう通信量の増加を抑えられ，より良いスケーラビリティを得られる可能性がある．また，OpenMP ディレクティブを利用すると MPI とは異なる部分の並列化を行うこともできる[12]．また，MPI の並列数が計算アルゴリズムや入力データにより限定されていて，最大並列数よりも利用可能なプロセッサ数が多い場合は，OpenMP による並列化を利用することで，より多くのプロセッサを利用でき，スケーラビリティを確保できる可能性がある．

5.　PHASE への適用

5.1　実 行 環 境

本研究では，BG/P を最大 1 筐体 1024 ノード（4096 コア）利用した．また，コンパイラは，IBM XL Fortran Advanced Edition for Blue Gene/P，V11.1，数学ライブラリは ESSL/ESSL SMP ver.4.3，PHASE は ver.6.01 を利用した．入力データの作成には

PHASE-Viewer（Ver.3.00）を利用し，PHASE コード内に同梱されている Si の 8 原子のサンプルデータ Si8 の単位格子を x，y，z の軸各方向に積み重ね，512 原子の Si512 を作成した．さらにそれを基準として 2 倍，4 倍し，1024 原子数（Si1024），2048 原子（Si2048）を作成した．Si512 はバンド数 1,536，平面波数 31,439，Si1024 はバンド数 3,072，平面波数 62,999，Si2048 はバンド数 6,144，平面波数 126,039 を使用した．波動関数のカットオフは 9.00Ry を利用した．Brillouin zone 内の試行 k 点は Γ 点 (0, 0, 0) のみとし，そのため並列化はバンドのみで行った．波動関数の更新後の部分対角化は行っていない．計算は収束させず，SCF 計算ループを 10 回だけ実行した．

5.2　ノード単体性能の向上

ここでは，PHASE の計算カーネルのキャッシュブロッキングの性能評価を行うことで BG/P で利用すべきブロックサイズを決定する．また，BG/P の SIMD 演算命令を利用するためのコード変更を行い，ノード単体性能の向上を実現する．Si512 データとオリジナルコードを利用し，128 プロセスでサブルーチンごとの実行時間プロファイルを取得した．結果として，文献 3)，4) と同様に擬ポテンシャルの非局所項と波動関数の積，波動関数と射影関数の内積，グラム・シュミットの直交化，FFT のルーチンでほぼ 8 割の実行時間を占めていることが分かった．そのため，ノード単体性能の向上のため，これらの部分の最適化を行うことにした．各ベンダの FFT ライブラリはそれぞれ少しずつサブルーチン名，引数，データ構造が異なっているため，公開されている PHASE のコードはそれら各種の FFT ライブラリに対応するラッパルーチンが用意されている．しかし，BG/P で利用可能な数学ライブラリの ESSL には対応していなかったため，新たにそのラッパルーチンを作成した．一般に FFT ライブラリのデータ長 n は $n = 2^h 3^i 5^j$ などになっている必要があるため，PHASE コードには，データ長をそれらに整合させるルーチンが含まれている．しかし，ESSL の FFT ライブラリはそのデータ長 n の制限が多少異なり，$n = 2^h 3^i 5^j 7^k 11^m$ $(1 \leq h \leq 25 : i \leq 2 : j, k, m \leq 1)$ であるため，これに整合させるロジックを追加した．次に実行時間のかかるルーチンの 1 つである擬ポテンシャルの非局所項と波動関数の積のコードを利用し，キャッシュブロッキング時の最適なブロックサイズの決定と SIMD 演算命令の出力のためのコード変更を行う．この部分のコードの構造を**図 3** に示す．

PHASE は，各カーネル部分でキャッシュブロッキングが行われており，入力ファイルにキャッシュサイズを指定するとそれをもとにブロックサイズが計算されるようになっている．このループの場合，キャッシュに入れておくべき配列要素は，y1(bs:be)，y2(bs:be)，z(bs:be,1:np,1)，z(bs:be,1:np,2) である．y1(bs:be)，y2(bs:be) は，Loop A でキャッシュ

```
do bs=1, n, bsize
  be=bs+bsize-1
        :
  do k = 1, 4
    do j = 1, 4
      t = tt(j)                          Loop A
      do i = bs, be<Loop for plain-waves>
        y1(i)=y1(i) + a1*s1(i)*ss(i, t)
        y2(i)=y2(i) – a1*s2(i)*ss(i, t)
      end do
    end do
        :                                Loop B
    do m = 1, np <Loop B1 for wavefunctions>
      a = x(m)
      do i = bs, be  <LoopB2 for plain-waves>
        z(i, m, 1) = z(i, m, 1) + a*y1(i)
        z(i, m, 2) = z(i, m, 2) + a*y2(i)
      end do
    end do
        :
  end do
end do
```

図 3　擬ポテンシャルと波動関数の積部分のカーネル構造
Fig. 3　Structure of a kernel code “Products of pseudo potential and wavefunctions”.

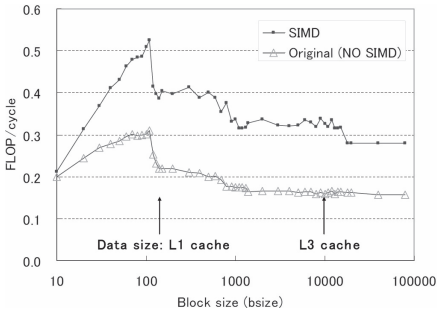

図 4　擬ポテンシャルと波動関数の積部分のブロックサイズとその性能
Fig. 4　Block size and the performance of “Products of pseudo potential and wavefunctions”.

に残った状態ならば，Loop B2（平面波数のループ）で再利用される．z(bs:be,1:np,1)，z(bs:be,1:np,2) は Loop B1（波動関数のループ）でループ長の np 回再利用される．**図 4** にこのコードのブロックサイズ bsize（= be-bs+1）を変更したときの性能 [FLOP/cycle]

変化のグラフを示す．Si512 のデータを 128 プロセスで実行したときの結果である．グラフの SIMD は後に述べる SIMD 演算命令出力のための変更を行った後の結果である．np は 12 であったため，ブロックサイズ 110 ではキャッシュに入るべき配列サイズは 23 KB であり，これは L1 キャッシュの容量に近い値である．10000 を超えると L3 キャッシュからも溢れるため，さらに性能が悪化している．この結果から実際のコードにおいても L1 キャッシュに配列要素を収めるようなブロックサイズを利用することが望ましいことが分かる．

次に SIMD 演算命令の利用を検討する．図 3 の配列 s1，s2，y1，y2 は，図 3 のコードの前方で allocate 文により確保されており，y1(1) など各配列の最初の要素は 16 バイト境界にアラインされている．しかし，キャッシュブロッキングのために配列を分割すると，最初にアクセスされる配列要素 y1(bs) はブロックサイズ次第で 16 バイト境界にアラインされていない可能性がある．一方，ブロックサイズは，キャッシュに収まるように計算されるだけであり，多少のサイズの変更は可能である．そのため，ブロックサイズを偶数に限定し，最初にアクセスされる y1(bs) が必ず 16 バイト境界にアラインされるようにした．他の配列 ss，z はアクセスされる配列要素が実行時に決まるため，**図 5** のようにバージョニングし，コードを切り替えることにした．図 5 のサブルーチン sub_align_ee は，配列要素 z(bs, m, 1)，z(bs, m, 2) が両方とも 16 バイト境界にアラインされている場合に実行され，図 3 の LoopB2 がサブルーチン化されている．このループでの配列は 3 次元であるが，配列アクセスは 1 次元方向のみであるため，sub_align_ee 内部では 1 次元配列で記述した．このようにすることで，コンパイラにとってコードの見通しが良くなり，より良いコードを出力しやすくなる．

```
do m = 1, np
  a = x(m)
  if (z(bs, m, 1) at 16 bytes boundary) then
    if ( z(bs, m, 2) at 16 bytes boundary) then
      call sub_align_ee(z(bs, m, 1), z(bs, m, 2),
                        y1(bs), y2(bs), a, be-bs+1)
  else
   :
  endif
  else
        :
```

図 5　アラインメント確認用コード
Fig. 5　Code for alignment check.

図 2 の DAXPY においては，SIMD 命令利用時の性能向上率はほぼすべてのループ長で 2 倍になっている．図 4 では，L1 キャッシュから溢れた後 L3 キャッシュに収まる部分での性能向上率は 2 倍で DAXPY の場合と変わらないが，L1 キャッシュに収まる範囲での性能向上率が 1.6 倍程度と DAXPY の場合より悪くなっている．図 4 のグラフ SIMD の実行コードは，バージョニング，サブルーチン化などを含んでいるため，L1 キャッシュに収めるためにブロックサイズを小さくすると，アライメント確認やサブルーチン化した部分を通る回数が増え，それらのオーバヘッドが増える．性能向上率の悪化はそのためと考えられるが，依然として L1 キャッシュに収めるブロックサイズを利用することが望ましいことが分かる．**図 6** に擬ポテンシャルの非局所項と波動関数の積の部分に対する最適化の効果を示す．横軸は最適化の種類であり，1 つ右に行くごとに最適化手法を追加している．オリジナルコードに対してキャッシュブロッキングにより 46%，SIMD 命令の利用により 41%の性能向上が得られ，全体で 68%の性能向上が得られた．波動関数と射影関数の内積においても計算カーネルの構造は，擬ポテンシャルの非局所項と波動関数の積とほぼ同じであり，同様の変更を加えると同様の最適化の効果が得られた．しかし，グラム・シュミットの直交化部分では，L1，L3 キャッシュに入れようとするとブロックサイズは非常に小さくなり，それにともなうループのオーバヘッドが大きくなってしまい性能が向上しないことが分かった．そのため，グラム・シュミットの直交化部分ではキャッシュブロッキングは行わないことにした．**図 7** に Si512 のデータの全体を実行したときの結果を示す．128 プロセスで実行した場合を示している．最適化の結果，全体で 45%（1.8 倍）の性能向上を実現した．**図 8** に SIMD 命令の実行率を示す．ここでは全演算量に対する SIMD 演算命令による演算量の割合とロード・ストアされた全要素数に対する SIMD 命令でロード・ストアされた要素数

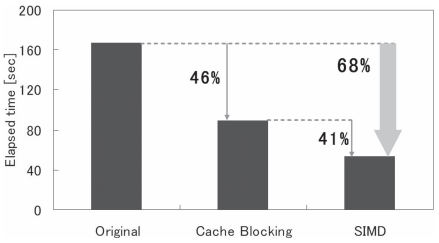

図 6　擬ポテンシャルと波動関数の積部分の最適化の効果（Si512 128 プロセス）
Fig. 6　Optimization results only for “Products of pseudo potential and wavefunctions” (Si512 128 procs).

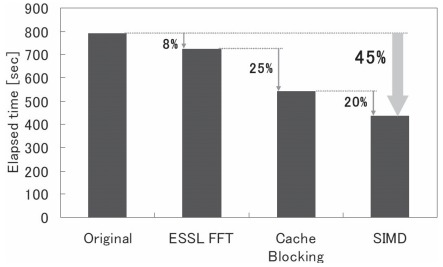

図 7　プログラム全体の最適化の効果（128 プロセス）
Fig. 7　Optimization results for whole program (128 procs).

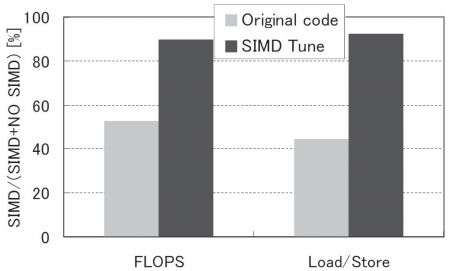

図 8　SIMD 実行率（128 プロセス）
Fig. 8　SIMD ratio (128 procs).

の割合で示している．これらは実行時のハードウェアカウンタの値に基づき計算した結果である．SIMD 命令出力のための変更により，9 割以上の SIMD 命令実行率を得ることができた．

5.3　スケーラビリティの向上

PHASE には，すでにいくつかのベンダ独自のディレクティブが挿入されており，ハイブリッド並列化がなされているが，BG/P で利用可能な OpenMP ディレクティブや IBM ディレクティブは挿入されていない．今回，計算カーネル内のループにディレクティブを挿入し実際に実行することで，BG/P におけるハイブリッド並列化すべきループを決定し，スケーラビリティの向上を実現した．ディレクティブとして OpenMP ディレクティブを利用した．今回 OpenMP による並列化箇所として，図 3 の Loop A と Loop B1，Loop B2 を検討した．LoopA と LoopB2 のループ長はプロセス数が増加しても変化しないが，LoopB1

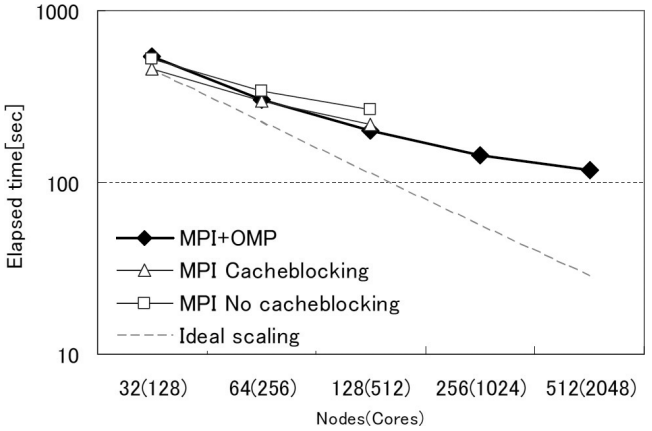

図 9　Si512 のスケーラビリティ
Fig. 9　Scalability of Si512.

のループ長はプロセス数の増加とともに小さくなる．プロセス数を増やしても，スケーラビリティを確保するためにはループ長の変わらない LoopA と LoopB2 のループを並列化することが望ましいと考えられる．しかし，実際に実行して両者を比較してみると LoopB2 よりも LoopB1 を並列化した方の性能が良かった．これは LoopB2 が最内ループであるため，ここにディレクティブを挿入すると，その実行回数が増え，スレッド生成，同期など並列化のオーバヘッドが増加してしまうためと考えられる．また，キャッシュブロッキングを行った状態に，ディレクティブを挿入し並列化しても，その効果を得られず，逆に並列化前よりも遅くなってしまった．これも，OpenMP ディレクティブの実行回数の増加によるオーバヘッドの影響と考えられる．そのため，今回ハイブリッド並列化実行時には，キャッシュブロッキングを行わないことにした．キャッシュブロッキングを行っていないため，ハイブリッド並列化の 1 スレッドの性能は，キャッシュブロッキングを行った MPI の 1 プロセスでの性能より悪いが，OpenMP による並列化の効果が得られればハイブリッド並列化による性能向上が期待される．実際に Si512 データを利用し計測した結果を**図 9** に示す．横軸はノード数であり，MPI のみの並列化の場合は各プロセッサに 1 プロセスずつノードあたり 4 プロセス割り当てたとき，MPI+OMP はハイブリッド並列化であり，各ノードあたり 1 プロセス 4 スレッドで走らせたときの結果である．縦軸は実行時間で対数表記している．MPI に関しては，キャッシュブロッキングの有無の両方をプロットした．Si512 のデータは

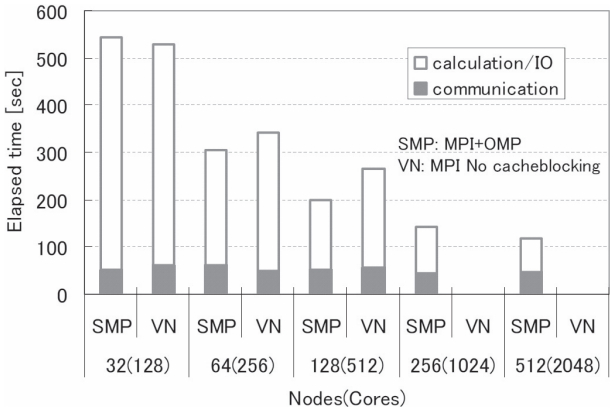

図 10　計算と通信の割合
Fig. 10　Communication versus. calculation/IO.

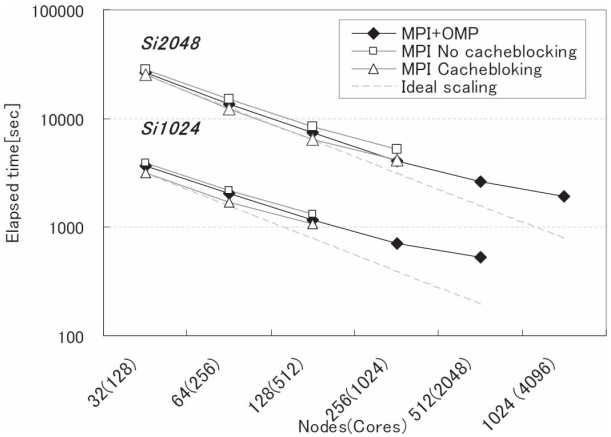

図 11　Si1024 と Si2048 のスケーラビリティ
Fig. 11　Scalability of Si1024 and Si2048.

その並列数の制限から 1024 プロセス以上で実行することはできなかった．通信や IO のタイミングなどにより各実行で性能のばらつく可能性があるが，512 プロセス，4 スレッドで 10 回計測したところ実行時間の誤差は 0.8%以内であったため，計測ごとの誤差はほぼないと考え，計測においては 1 回のみ計測することにした．キャッシュブロッキングなしの状態で MPI+OMP と MPI を比較すると，MPI+OMP のノード数が少ないときの実行時間は，MPI のキャッシュブロッキングなしとほぼ一致しているが，スケーラビリティはハイブリッド並列化の性能が良い．キャッシュブロッキングをすることでノード単体性能が向上するためグラフは下方に平行移動した形となるが，ノード数を増加させたときは依然としてハイブリット並列化の性能の方が良い．

図 10 に各ノードで実行時の通信と計算または IO の実行時間を示す．これを見るとノード数が増加してもほとんど通信時間は増加していないことが分かる．そのため，ハイブリッド並列時のスケーラビリティの向上は，実行 MPI プロセス数の増加が抑えられることによる通信時間の減少が原因ではないと考えられる．ここでの性能向上の原因は，MPI と OpenMP の並列化対象が異なっていることが考えられる．今回 OpenMP の並列化をしたループには LoopA のように MPI 並列化と無関係のループが存在する．このため MPI の並列化の効果が得られなくなったときでも OpenMP の並列化の効果を得ることができたと考えられる．次に大きなデータを利用する場合として，**図 11** に Si1024 と Si2048 の結果を示す．理想的なスケーリングを点線で表している．Si1024 と Si2048 も Si512 と同様の傾向を示しているが，データが大きい場合，より理想に近いスケーリングを実現していることが分かる．今回 OpenMP ディレクティブのオーバヘッドのため，ハイブリッド並列化時にキャッシュブロッキングを行うことができなかった．これを回避するためには，図 3 のコードの場合，最外ループで並列化し，ディレクティブの実行回数を減らす必要があると考えられる．しかし，これにはディレクティブの挿入だけではなく，依存関係などを解決するためのコード変更も必要であるため，今回は行わなかった．また，文献 4) のように波数方向にも MPI の並列化を行えば，キャッシュブロッキングに加え，さらなるスケーラビリティの向上が期待される．

5.4 ま と め

本研究では，BG/P において，PHASE の最適化を行った．キャッシュブロッキングの最適なブロックサイズの決定，SIMD 演算命令の利用のためのコード変更，FFT の置き換えにより，単体ノード性能としてはオリジナルコードに対して 1.8 倍の性能を得ることができた．また，ハイブリッド並列化に際しては OpenMP ディレクティブを挿入するループを決定し，スケーラビリティが改善されることを確認した．データサイズを大きくすることでスケーラビリティが改善されることも確認した．

参 考 文 献

1) 次世代スーパーコンピュータ開発実施本部 Web サイト. http://www.nsc.riken.jp/index_j.htm
2) 革新的シミュレーションソフトウェアの研究開発 Web サイト. http://www.ciss.iis.u-tokyo.ac.jp/rss21/
3) Ohno, T., Yamamoto, T., Yamasaki, T., Kokubo, T., Sakaguchi, Y., Fukata, D., Azami, A., Uda, T. and Koga, J.: First-Principles Calculations of Large-Scale Semiconductor Systems on the Earth Simulator, *Proc. 2007 ACM/IEEE Conference on Supercomputing, Gordon Bell Finalist Awards*, Reno, USA (2007).
4) 小松秀実，山崎隆浩，市川真一：第一原理分子動力学・PHASE の超並列化—次世代スーパーコンピューティングに向けて，雑誌 Fujitsu 2008 年 9 月号（Vol.59, No.5）(2008).
5) TOP500 Supercomputer Sites. http://www.top500.org
6) Kumar, S., Huang, C., Zheng, G., Bohm, E., Bhatele, A., Philips, J.C., Yu, H. and Kale, L.V.: Scalable molecular dynamics with NAMD on the Blue Gene/L system, *IBM Journal of Research & Development*, Vol.52, pp.137–144 (2008).
7) Gygi, F., Draeger, E.W., Schulz, M., de Supinski, B.R., Gunnels, J.A., Anstel, V. and Sexton, J.C., et al.: Large-Scale Structure Calculations of High-Z Metals on the Blue Gene/L Platform, *Proc. 2006 ACM/IEEE Conference on Supercomputing, Gordon Bell Prize for Peak Performance winner*, Tampa, USA (2006).
8) IBM Blue Gene Team: Overview of the IBM Blue Gene/P project, *IBM Journal of Research & Development*, Vol.52, pp.199–220 (2008).
9) The Green 500 List. http://www.green500.org
10) Chatterjee, S., Bachega, L.R., Bergner, P., Dockser, K.A., Gunnels, J.A., Gupta, M., Gustavson, F.G., Lapkowski, C.A., Liu, G.K., Mendell, M., Nair, R., Wait, C.D., Ward, T.J.C. and Wu, P.: Design and exploitation of a high-performance SIMD floating-point unit for Blue Gene/L, *IBM Journal of Research & Development*, Vol.49, pp.377–389 (2005).
11) Sosa, C., et al.: IBM System Blue Gene Solution: Blue Gene/P Application Development. http://www.redbooks.ibm.com/abstracts/sg247287.html
12) Cappello, F. and Etiemble, D.: MPI versus MPI+OpenMP on the IBM SP for the NAS Benchmarks, *Proc. Supercomputing2000*, Dallas, USA (2000).

(平成 20 年 10 月 3 日受付)
(平成 21 年 1 月 4 日採録)

今井　晴基（正会員）

昭和 53 年生．平成 16 年筑波大学大学院システム情報工学研究科知能機能システム専攻博士前期課程修了．同年日本アイ・ビー・エム（株）入社．東京基礎研究所にて PC クライアント関連の研究開発，超並列計算機のコード最適化技術の研究開発に従事．

森山　孝男（正会員）

昭和 37 年生．昭和 62 年東京工業大学工学部情報理工学専攻修士課程修了．同年日本アイ・ビー・エム（株）入社．東京基礎研究所にて並列計算機のシステムソフトウェア，高速 3D 表示装置，ハイブリッドアーキテクチャ，超並列計算機の利用技術等に関する研究に従事．ACM 会員．