Xeonプロセッサ向けLinpackベンチマーク最適化手法とその評価

成瀬 彰[†] 住元真司[†] 久門耕一[†]

本論文では，PCクラスタのプロセッサとしてよく使用されるXeonプロセッサ向けのLinpackベンチマーク高速化手法について述べる．特に，Linpackベンチマーク実行時間の大半を占める倍精度行列積（DGEMM）の最適化手法について説明する．最適化はオープンソースの数値演算ライブラリであるATLASをベースに行い，キャッシュ容量，TLBエントリ数，システムバス負荷の均等化を考慮してDGEMMカーネルを作成し，それに合わせてDGEMMカーネル呼び出し部分を変更した．最適化したDGEMMを2.4 GHz Xeonプロセッサを搭載したFujitsu PRIMERGY L250で評価した結果，DGEMMそのものの性能を測定する正方行列積プログラムでは4.33 GFlops（実行効率：90.2%）を記録した．また，Linpackベンチマークでは4.13 GFlops（実行効率：86.0%）を記録した．これは従来最速とされていたGOTO BLASを5%上回る性能である．

Optimization and Evaluation of Linpack Benchmark for Xeon Processor

Akira Naruse,[†] Shinji Sumimoto[†] and Kouichi Kumon[†]

In this paper, we explain the optimization technique of matrix multiplication (DGEMM) for Xeon processor, that is heavily used in Linpack benchmark. Our optimization technique is based on reducing cache misses and D-TLB misses and averaging FSB (Front Side Bus) loads. We have applied this technique to an open-source numerical library ATLAS. The benchmark results using our DGEMM implementation show better performance than GOTO BLAS that is famous for a fast DGEMM. On a Fujitsu PRIMERGY L250 system equipped with 2.4 GHz Xeon processors, performance of square matrix multiplication program attains 4.33 GFlops (90.2% efficiency) and performance of Linpack benchmark attains 4.13 GFlops (86.0% efficiency).

1. はじめに

多数のPCを高速なネットワークで結合したPCクラスタは，価格性能比が非常に良いこともあり，科学技術計算分野において重要なプラットフォームの1つとなっている．2003年秋のTOP500[1]においては，上位10システムのうち6システムがPCクラスタであり，そのうち3システムがプロセッサとしてXeonプロセッサを搭載したシステムであった．

本論文では，このPCクラスタシステムのプロセッサとして広く用いられているXeonプロセッサ向けに，TOP500で使用されるLinpackベンチマークを高速化した手法について述べる．特に，Linpackベンチマーク実行時間の大半を占める倍精度行列積（DGEMM）の最適化について述べる．DGEMMは，BLAS[2]のLevel 3で規定される倍精度行列積用のサブルーチンである．BLASのLevel 3では多数のサブルーチンが規定されているが，その多くは行列積に帰結できることが知られている[3]．単にLinpackベンチマークで使われているからだけではなく，行列積をチューニングすることでLevel 3 BLASサブルーチンの大部分をチューニングできるという特徴があるため，DGEMMはこれまで多くのグループの研究対象となってきた．

DGEMMの最適化は，オープンソースの数値演算ライブラリであるATLAS[4]をベースに行った．Xeonプロセッサのハードウェアリソース（キャッシュサイズ，TLBエントリ数，レジスタ本数）とシステムバスの負荷均等化を考慮してDGEMMカーネルを設計・実装し，それに合わせてDGEMMカーネルの呼び出し部分を変更した結果，2.4 GHz Xeonを搭載したFujitsu PRIMERGY L250において，DGEMMそのものの性能を示す正方行列積プログラムで4.33 GFlops（実行効率：90.2%）と高い性能を記録した．また，Linpackベンチマークでは4.13 GFlops（実行効率：86.0%）を記録した．

本論文の構成は次のとおりである．2章ではXeonプロセッサの特徴について述べる．3章ではLinpackベ

† 富士通研究所
Fujitsu Laboratories

表 1 Xeon プロセッサの特徴
Table 1 Characteristices of Xeon processor.

拡張命令（SSE2）	倍精度 SIMD 演算命令
FSB 動作周波数	400/533 MHz
FSB 帯域	3.2/4.3 GB/s
HW プリフェッチ	8 ストリーム
L1 キャッシュサイズ	8 KB (D)
L2 キャッシュサイズ	512 KB
TLB エントリ数	128 (I) + 64 (D)

ンチマークの現状について述べ，Linpack ベンチマークの実装の 1 つである HPL[5] と，HPL が使用する行列積サブルーチン（DGEMM）の特徴について説明する．4 章では，我々が提案する Xeon プロセッサ向けの DGEMM 最適化手法について説明する．5 章では最適化した DGEMM の評価結果について述べる．6 章では結論を述べる．

2. Xeon プロセッサ

本章では Xeon プロセッサの特徴について述べる．

Xeon プロセッサは，Pentium4 プロセッサを SMP 対応にしたサーバ向け IA-32 プロセッサであり，基本的な特徴は Pentium4 プロセッサと同等である．Xeon プロセッサの主な特徴を**表 1** に示す[6),7)]．

近年の PC の用途変化にともない，Pentium4 プロセッサにはデジタル画像処理やビデオ再生等で必要となる機能が強化されている．それにともなって拡張命令（SSE2）で倍精度 SIMD 演算命令が追加され，システムバス（FSB）性能も大幅に強化された結果，Pentium3 と比べて科学技術計算分野でも使いやすいプロセッサとなった．

一方，キャッシュサイズは大きくなっていない．データ L1 キャッシュに関しては，Pentium3 の 16 KB より小さくなっている．SSE2 用のレジスタ数も従来どおり 8 本で，科学技術計算分野で使用するにはもの足りない．また，文献 8) が指摘しているとおり，L2 キャッシュサイズとデータ TLB のエントリ数がアンバランスである．ページサイズが 4 KB の場合☆，256〜512 KB サイズの配列データを繰り返し走査すると，L2 キャッシュはヒットするが，D-TLB はミスするという現象が発生する．

1 世代前の Pentium3 に比べると，HW プリフェチによるメモリレイテンシ隠蔽が可能であり，ピーク浮動小数点演算性能は動作周波数の 2 倍で，かつ動作周波数が他プロセッサに比べて速いこともあり，高い潜在能力を備えているといえるが，まだハードウェアリソースで物足りない面もある．理論性能に近い性能を引き出すには，相当の工夫が必要なプロセッサといえる．

☆ ページサイズは 4 KB が一般的．

3. Linpack

本章では Linpack ベンチマークの特徴について述べる．

Linpack ベンチマークは，ガウス消去法を用いて非対称実数密行列の連立一次方程式を解く際の演算性能を測定するものである．HPL（High-Performance LINPACK Benchmark）[5] は，分散メモリ型並列計算機用の Linpack ベンチマーク実装の 1 つである．以下，HPL と HPL が用いる数値演算ライブラリに関して述べる．

3.1 HPL

HPL は，C 言語実装の Linpack ベンチマーク実行プログラムで，実行には MPI（Message Passing Interface）[9] 準拠のメッセージ通信ライブラリが必要である．文献 10) が示すとおり，HPL には多数のパラメータが存在し，問題サイズ（行列サイズ）だけでなくブロックサイズや通信パターン等をパラメータで変更することが可能である．使用する数値演算ライブラリやネットワーク性能・トポロジ等，測定対象システムの構成に応じてパラメータを設定できるため，TOP500 の上位にランクするシステムでも使用されている．もちろん 1 プロセッサでも使用できる．

HPL は行列計算部分に数値演算ライブラリとして BLAS（Basic Liner Algebra Subprograms）[2] もしくは VSIPL（Vector Signal Image Processing Library）[11] 準拠のライブラリを使用することができる．本論文では BLAS を対象とした．HPL 実行時の数値演算ライブラリの占める役割は非常に大きい．文献 12) が指摘しているとおり，HPL 実行時間の 8〜9 割は数値演算ライブラリの行列積が占めている．HPL の実行性能は行列積の性能次第であるともいえる．

3.2 DGEMM

DGEMM は Level 3 BLAS で規定される倍精度行列積のサブルーチンであり，HPL 実行時間の大部分を占めている．この DGEMM を最適化することで，Linpack ベンチマークの実行性能を上げることができる．

一般的な DGEMM 実装例について述べる．行列積 $C = AB + C$ の計算方法としては，コアループでのキャッシュミスや TLB ミスを削減するため，行列 A，B，C をブロックサイズでサブ行列に分割し，サブ行列単位で行列積を計算するのが一般的である．サブ行列

```
for (j=0; j<N; j+=NB) {
    for (i=0; i<M; i+=NB) {
        copy_gather(C[j][i], wC);
        for (k=0; k<K; k+=NB) {
            copy_gather(A[i][k], wA);
            copy_gather(B[j][k], wB);
            dgemm_kernel(wA, wB, wC);
        }
        copy_scatter(wC, C[j][i]);
    }
}
```

図 1 DGEMM カーネル呼び出し処理例
Fig. 1 DGEMM kernel caller example.

```
for (i=0; i<NB; i++) {
    for (j=0; j<NB; j++) {
        for (k=0; k<NB; k++) {
            wC[j][i] += wA[i][k] * wB[j][k];
        }
    }
}
```

図 2 DGEMM カーネルの例
Fig. 2 DGEMM kernel example.

に対する行列積ルーチンは DGEMM カーネルと呼ばれる．ブロックサイズが NB の場合の DGEMM カーネルの呼び出し処理を**図 1** に示す．また，DGEMM カーネルの例を**図 2** に示す．

図 1 の DGEMM カーネル呼び出し処理例ではループ順序が $j \to i \to k$ となっているが，これは実装依存である．また，ライン競合によるキャッシュミスや TLB ミスを減らすため，DGEMM カーネルを呼び出す前に行列データをワーク領域 wA，wB，wC にコピーして連続メモリアドレス上に行列データを配置している．転置が必要な場合にはこのときに同時に行われる．行列 A，B，C のどれをワーク領域にコピーし，どれをコピーしないかは実装依存であり，DGEMM チューニングポイントの 1 つである．

図 2 の DGEMM カーネル例は非常にシンプルな例である．多くの場合，分岐ペナルティを削減するために k ループは完全に展開される．また，演算あたりのロード命令数を少なくするために i ループと j ループはレジスタ本数に応じてアンローリングされる．コア部分はアセンブラ相当で記述されることが多い．

3.2.1 性 能 比 較

BLAS 準拠の数値演算ライブラリは多数存在する．Xeon プロセッサで使用できる BLAS 準拠ライブラリの中では，下記の 3 種類が有名である．

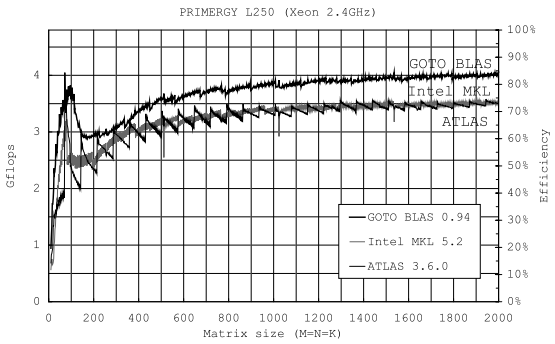

図 3 正方行列積による DGEMM 性能比較
Fig. 3 DGEMM performance comparison.

- ATLAS [4)]
- Intel MKL（Math Kernel Library）[13)]
- GOTO BLAS [14)]

ATLAS はテネシー大学の Whaley らが開発したライブラリで，対象プロセッサのハードウェアリソースに適したライブラリを自動生成することで有名である．ソースはオープンソースで配布されている．Intel MKL はベンダ提供ライブラリで，スレッドで並列化されている．ユーザがプログラムを並列化しなくても，SMP マシンであれば自動で複数プロセッサを使うことができる．GOTO BLAS はテキサス大学の後藤ら☆が開発しているライブラリであり性能に定評がある．

この 3 種類の BLAS 準拠ライブラリを使い，2.4 GHz Xeon を搭載した PRIMERGY L250 で倍精度の正方行列積プログラムを実行した結果を**図 3** に示す．図 3 は，横軸が行列サイズを，縦軸が演算性能を示している．ほぼすべての行列サイズにおいて，GOTO BLAS が最も良い性能を示しており，最高で 4.06 GFlops を記録している．実行効率は 84.5%である．

4. DGEMM 最適化

本章では，我々が行った DGEMM 最適化手法について述べる．最適化は，オープンソースである ATLAS をベースに行った．以下，DGEMM カーネルの最適化と，DGEMM カーネル呼び出し部分の最適化に分けて説明する．

4.1 DGEMM カーネルの最適化

最適化した DGEMM カーネルの概要を**図 4** に示す．図 4 には k ループが記述されているが，実際には k ループは完全に展開され存在しない．

☆ 後藤氏は Alpha プロセッサ用高速 BLAS の開発者として有名．

```
for (j=0; j<N; j++) {
   for (i=0; i<156; i+=6) {
      for (k=0; k<156; k++) {
         C[j][i  ] += wA[i  ][k] * wB[j][k];
         C[j][i+1] += wA[i+1][k] * wB[j][k];
         C[j][i+2] += wA[i+2][k] * wB[j][k];
         C[j][i+3] += wA[i+3][k] * wB[j][k];
         C[j][i+4] += wA[i+4][k] * wB[j][k];
         C[j][i+5] += wA[i+5][k] * wB[j][k];
      }
      prefetch(wB[j+1][i]);
   }
}
```

図 4 最適化した DGEMM カーネル
Fig. 4 Optimized DGEMM kernel.

DGEMM カーネル最適化のポイントは 3 つである．

- アンローリング
- ブロックサイズ
- システムバス負荷の均等化

以下，それぞれについて述べる．

4.1.1 アンローリング

DGEMM カーネルにおいては，i ループと j ループのどちらか一方，もしくは双方をアンローリングして，演算あたりのロード命令を削減することができる．i ループ，j ループのアンローリング数をそれぞれ u_i と u_j とすると，u_i と u_j は下記条件を満たす必要がある．

$$(u_i * u_j) + min(u_i, u_j) + 1 \leq \text{レジスタ数} \quad (1)$$

基本的にはアンローリング数は多い方が望ましいが，Xeon プロセッサの FP レジスタ数は 8 本である．この条件を満たす (u_i, u_j) 組合せは**図 5** に示すとおり 12 種類ある．そのうち $(1,6)$，$(2,2)$，$(6,1)$ のアンローリングで作成した 3 種類の DGEMM カーネルの性能評価結果を**表 2** に示す．評価には ATLAS の DGEMM カーネル性能測定プログラムを用いた．プログラムは Linpack 実行時の配列アクセスパターンを模擬するように変更した．ブロックサイズは 156 である．

演算数あたりのロード命令数は，$(2,2)$ の場合が 0.50 で，$(1,6)$ と $(6,1)$ の場合が 0.58 である☆．これで判断すると $(2,2)$ が良さそうだが，評価の結果は $(6,1)$ が最も良くなった．

主な原因は行列 wB の参照のストリーム数と考えられる．$(2,2)$，$(1,6)$ の場合，行列 wB の参照はそれぞれ 2 ストリーム，6 ストリームとなる．そのため，行列 wB のデータをメモリからキャッシュに転送するのに HW プリフェッチを使えない可能性が高い☆☆．また，システムバス負荷の偏りもストリーム数が増えるほど大きくなる．さらに，j ループが 1 巡する間にアクセスするデータ量が増えるため，キャッシュの利用効率が落ちる可能性もある．

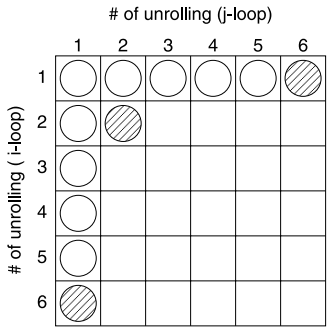

図 5 選択可能なアンローリング方式
Fig. 5 Unrolling patterns.

表 2 DGEMM カーネルの性能評価（NB=156）
Table 2 Performance evaluation of 3 DGEMM kernels.

(u_i, u_j)	GFlops	(実行効率)
$(1,6)$	3.72	(77.5%)
$(2,2)$	4.03	(84.0%)
$(6,1)$	4.33	(90.3%)

最終的には全 12 種類の組合せをひととおり評価したうえで，$(6,1)$ のアンローリングを選択した．$(6,1)$ の場合の命令列の一部を**図 6** に示す．

4.1.2 ブロックサイズ

ブロックサイズは DGEMM カーネルの処理対象であるサブ行列の大きさを決める係数である．このブロックサイズは，基本的には DGEMM カーネルの j ループ 1 巡で使用するデータがキャッシュに収まる範囲で最大のものを選択するのが望ましい．行列 C と行列 wB は 1 列分がキャッシュに収まればよいが，行列 wA は全データがキャッシュに収まる必要がある（**図 7**）．そのため，ブロックサイズ NB は次の条件を満たす必要がある．

$$8\,Byte * (NB * (NB + 2)) < \text{キャッシュ容量} \quad (2)$$

また，アンローリング数は $(u_i, u_j) = (6,1)$ としたので，ブロックサイズは 6 の倍数にするのが望ましい．

☆ 加算と乗算は別演算として算出．レジスタ間のロード命令はカウントせず．

☆☆ 4 KB ページ内に 2 ストリーム以上ある場合，HW プリフェッチは機能しない．

```
....
movapd   wB[j][k]    reg6
movapd   wA[i][k]    reg7
mulpd    reg6        reg7
addpd    reg7        reg0
movapd   wA[i+1][k]  reg7
mulpd    reg6        reg7
addpd    reg7        reg1
movapd   wA[i+2][k]  reg7
mulpd    reg6        reg7
addpd    reg7        reg2
movapd   wA[i+3][k]  reg7
mulpd    reg6        reg7
addpd    reg7        reg3
movapd   wA[i+4][k]  reg7
mulpd    reg6        reg7
addpd    reg7        reg4
movapd   wA[i+5][k]  reg7
mulpd    reg6        reg7
addpd    reg7        reg5
....
```

図 6 DGEMM カーネルの命令列
Fig. 6 DGEMM kernel instruction sequence.

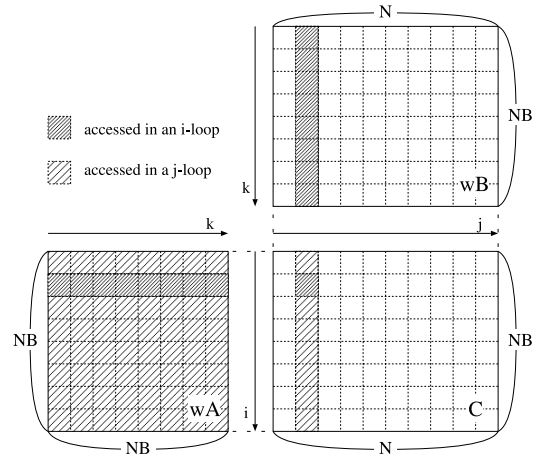

図 7 DGEMM カーネルの j/i ループ内で参照されるデータ
Fig. 7 Data access in j/i-loop of DGEMM kernel.

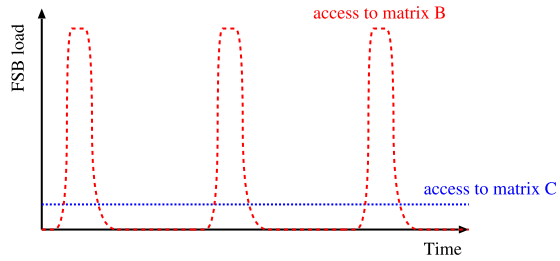

図 8 DGEMM カーネル内におけるシステムバス負荷の偏り
Fig. 8 Unbalance FSB load in DGEMM kernel.

Xeon プロセッサの L2 キャッシュ容量は 512 KB であるが，D-TLB のエントリ数は 64 本しかない．ページサイズが 4 KB の場合，D-TLB がカバーできるのは最大 256 KB となる．L2 キャッシュに収まるサイズであっても，256 KB を超えると D-TLB ミスが発生する．そこで，L2 キャッシュ容量は 256 KB と見なすことにする．

L2 キャッシュ容量を 256 KB と考えると，ブロックサイズは 180 が上限となる．180 を上限としていろいろなブロックサイズを試した結果，156 をブロックサイズとした場合に安定して高い性能を示すことが分かった．156 を超えるブロックサイズでは，ブロックサイズ 156 の場合を超える性能を示すこともあったが，性能が不安定であった．不安定の理由は，配列データのメモリ配置により，キャッシュでライン競合が発生する可能性が高いためと思われる．

ブロックサイズを 156 とすると，安定して高い性能を出せることが確認できたので，我々はブロックサイズを 156 とした．

4.1.3 システムバス負荷の均等化

DGEMM カーネル内では，行列 wA データの参照時にはキャッシュミスは発生しない．キャッシュミスが発生する可能性があるのは，行列 wB データの参照と行列 C データの参照 · 更新のときである．特に，行列 wB データの参照は性能に対して大きな影響がある．

行列 wB データの参照について考える．行列 wB 内の参照カ所は j ループで決まり，i ループ内で参照される行列 wB データは同じである（図 7）．したがって，i ループの 1 巡目だけキャッシュミスが発生し，2 巡目以降はキャッシュミスが発生しないことになる．j ループ 1 巡の間に参照される行列 wB の総データサイズは 1,248 B である☆．システムバス負荷の観点から考えると，1 KB 強の負荷が偏って存在することになる．**図 8** にその概念図を示す．

システムバス負荷は均等であった方が良い性能が出る．そこで，行列 wB の参照で発生するシステムバス負荷を均等化するため，プリフェッチを導入した（図 4）．次の j ループで参照する行列 wB のデータを，i ループ内において少しずつプリフェッチすることにより，i ループの 1 巡目に偏っていたシステムバスへ

☆ $1,248\ Byte = 156 * 8\ Byte$

```
for (k=0; k<K; k+=NB) {
    copy_gather(B[0][k], wB);
    for (i=0; i<M; i+=NB) {
        copy_gather(A[i][k], wA);
        dgemm_kernel(wA, wB, C);
    }
}
```

図 9 最適化した DGEMM カーネル呼び出し部
Fig. 9 Optimized DGEMM kernel caller.

の負荷が均等化される．

このプリフェッチの実装には，プリフェッチ命令ではなくロード命令を用いた．Xeon プロセッサのプリフェッチ命令では，D-TLB ミスが発生する場合にはプリフェッチが実行されず，メモリからキャッシュへデータが転送されない．一般的にはこの仕様は正しい．D-TLB ミスペナルティは大きいので，あえて D-TLB ミスを発生させてまでプリフェッチを実行する必要性は少ないからである．しかし，本ケースのように，プリフェッチしたデータをほぼ確実に使用する場合には，このプリフェッチ命令の仕様では意図した効果が得られない．そのため通常のロード命令を用いた．

4.2 DGEMM カーネルの呼び出し最適化

最適化した DGEMM カーネルに合わせて設計した DGEMM カーネル呼び出し部を**図 9** に示す．以下に設計のポイントを示す．

- ループ順序：
  行列 A データの再利用を重視し，ループ順序は $k \to i \to j$ とした．
- 行列データのワーク領域へのコピー：
  行列 A，B データはワーク領域 wA，wB にコピーするが，行列 C データはワーク領域へのコピーをしない．k ループを最外ループとしたため，行列 C データはワーク領域にコピーする意味がない．
- 行列 B のワーク領域へのコピー回数削減：
  j ループ方向で行列を分割するのをやめ，行列 B データのワーク領域 wB へのコピー回数を削減した．その結果，DGEMM カーネルの呼び出し部からは j ループがなくなった．ワーク領域 wB のサイズが大きくなるという短所がある．

5. 評　　価

本章では性能評価の結果を示す．評価環境は**表 3** に示すとおりで，マシンは 2.4 GHz Xeon プロセッサを搭載した PRIMERGY L250 を使用した．

5.1 正方行列積による DGEMM 性能評価

正方行列積プログラムを実行した結果を**図 10** に示す．図 10 は，横軸が行列サイズを，縦軸が演算性能を示している．比較対象は既存の BLAS ライブラリの中で最も性能の良かった GOTO BLAS である．

表 3 評価環境
Table 3 Evaluation environment.

Machine	Fujitsu PRIMERGY L250
Processor	Xeon 2.4 GHz (dual)
L1 (D)/L2	8 KB/512 KB
Chipset	Intel E7500
FSB	400 MHz (3.2 GB/s)
Memory	2 GB (PC1600, dual channel)
Network	GbE (Intel e1000)
Switch	Dell POWERCONNECT 5224
OS	Redhat Linux 7.3 (kernel: 2.4.19)
MPI	SCore MPICH 1.2.4 (SCore 5.4)
Compiler	gcc 2.96
Compile options	-O3 -fomit-frame-pointer -funroll-all-loops

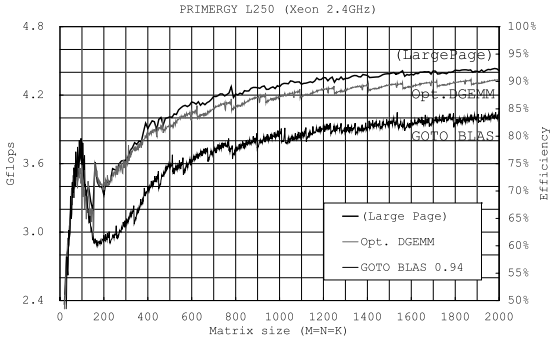

図 10 正方行列積による最適化 DGEMM の性能評価
Fig. 10 Optimized DGEMM performance evaluation.

ほぼすべての行列サイズにおいて，我々が最適化した DGEMM は GOTO BLAS を上回る性能を示した．表 4 にそれぞれの平均性能と最高性能を示す．実行効率を比較すると，最適化 DGEMM は GOTO BLAS を最高性能で 5.9 ポイント，平均性能で 7.4 ポイント上回っている．

5.1.1 ラージページの効果

一般的に，Xeon プロセッサ搭載システムで使用されるページサイズは 4 KB であるが，Xeon プロセッサは 4 MB ページサイズ（ラージページ）をサポートしている．ラージページにすると TLB ミスが減り，TLB ミスがネックとなるプログラムにおいては性能向上が期待できる．最適化した DGEMM においても，ラージページを用いると

- 行列データのワーク領域へのコピー，
- 行列 C データへの参照・更新，

の際に発生する TLB ミスが減り，性能が上がる可能性がある．

表 4 正方行列積実行時の平均性能と最高性能
Table 4 Average and maximum performance of matrix multiplication (DGEMM).

	平均性能		最高性能	
	GFlops	(実行効率)	GFlops	(実行効率)
GOTO BLAS	3.69	(76.8%)	4.06	(84.5%)
Opt. DGEMM	4.04	(84.2%)	4.34	(90.4%)
(LargePage)	4.12	(85.8%)	4.43	(92.4%)

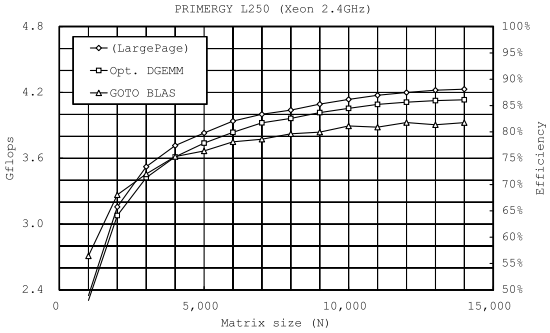

図 11 1 プロセッサでの Linpack 測定結果
Fig. 11 Linpack result on single processor.

表 5 1 プロセッサでの Linpack 測定結果（N=14,000）
Table 5 Linpack result on single processor.

	GFlops	(実行効率)
GOTO BLAS	3.93	(81.9%)
Opt. DGEMM	4.13	(86.0%)
(LargePage)	4.23	(88.1%)

使用した 2.4 系の Linux カーネルはラージページをサポートしていないが，ラージページを可能にするカーネル patch は何種類か存在する．その中の 1 つである super page [15] を用いてラージページ環境を作成，ラージページの効果を評価した．測定結果を図 10 と**表 4** に示す．ラージページを用いることにより，実行効率は最高性能で 2.0 ポイント，平均性能で 1.6 ポイント向上した．

5.2 Linpack ベンチマークによる評価

5.2.1 1 プロセッサ（UP）

1 プロセッサによる Linpack ベンチマークの実行結果を**図 11** と**表 5** に示す．図は横軸が行列サイズ（N）で縦軸が GFlops である．HPL のパラメータに関して，行列サイズは 2 GB メモリの 8 割程度を行列データに使用する 14,000 を最大とした[10]．ブロックサイズ（NB）は，GOTO BLAS は 112，最適化 DGEMM は 156 を選択し，それぞれが得意とするサイズに合わせた．

最適化 DGEMM を使用した場合の Linpack 性能は 4.13 GFlops を記録した．実行効率は 86.0%である．実行効率で GOTO BLAS を 4.1 ポイント上回った．さらに，ラージページを使用すると実行効率は 88.1%まで上がった．

表 6 2 プロセッサでの Linpack 測定結果
（N=14,000，(P,Q)=(1,2)）
Table 6 Linpack result on dual processors.

	GFlops	(実行効率)
GOTO BLAS	7.04	(73.3%)
Opt. DGEMM	7.43	(77.4%)

表 7 1 多重と 2 多重の Linpack 測定結果（N=10,000）
Table 7 Linpack result on 1 and 2 processes.

		GFlops	(実行効率)
GOTO BLAS	(1 多重)	3.91	(81.4%)
	(2 多重)	3.72	(77.5%)
Opt. DGEMM	(1 多重)	4.06	(84.5%)
	(2 多重)	3.89	(81.1%)

5.2.2 2 プロセッサ（DP）

2 プロセッサ（DP）使用時の Linpack ベンチマークの実行結果を**表 6** に示す．1 プロセッサ時と比べると，GOTO BLAS と最適化 DGEMM のどちらも実行効率が 8.6 ポイント低い結果となっている．原因としては以下のことが考えられる．

- システムバス負荷増によるメモリ latency の増加
- 1 プロセッサあたりの割当てデータ量が半減
- 並列化オーバヘッド
  - Panel Factorization で 1 プロセッサ待機
  - Panel Broadcast でのメモリコピー

原因切分けのため，1 プロセッサ使用 Linpack ベンチマークを 1 多重と 2 多重☆で実行した場合の性能を測定した．**表 7** にその結果を示す．1 多重と比べて 2 多重では実行効率が 3.4〜3.9 ポイント低下している．1 多重と 2 多重はプロセッサあたりの割当てデータ量は同じであり，また並列化オーバヘッドも存在しない．したがって，システムバス負荷増の影響で実行効率が 3.4〜3.9 ポイント低下したと考えられる．

なお，Xeon プロセッサの性能測定機能[6),7)] を用いた調査により，システムバス使用率は 1 多重の場合は 15%程度，2 多重の場合は 30%程度であることを確認している．

5.2.3 PC クラスタ

PC クラスタでの Linpack ベンチマーク測定結果を**表 8** に示す．MPI は SCore [16] MPICH 1.2.4（SCore 5.4）を使用し，ネットワークは PRIMERGY L250 に標準搭載の GbE（Intel e1000）を用いた．HPL のパラメータに関して，問題サイズ N は 29000，プロセ

☆ 2 多重：プログラムを同時に 2 つ起動．

表 8 PC クラスタでの Linpack 測定結果
(4 nodes/8 processors)
Table 8 Linpack result on PC cluster system.

	HPL パラメータ			測定結果	
	N	P	Q	GFlops	(実行効率)
Opt. DGEMM	29,000	2	4	26.9	(70.1%)

ス格子 (P, Q) は (2, 4) とした．また，LU 分解と再帰的 LU 分解のアルゴリズムは right-looking を選択した[10]．4 ノード/8 プロセッサ構成で，26.9 GFlops を記録した．実行効率は 70.1%である．

2 プロセッサでの実行結果と比較して，実行効率は 7.3 ポイント低い結果となった．主な原因としては Panel Factorization，Panel Broadcast における通信処理が考えられる．本評価ではネットワークに GbE を用いたため，latency が長く，バンド幅も狭い．高速なネットワークを使用することで，latency 短縮で主に Panel Factorization の通信処理を，バンド幅増で主に Panel Broadcast の通信処理を高速化できると見込んでいる．

6. まとめ

本論文では，Xeon プロセッサ向けに Linpack ベンチマークを高速化するため，Linpack ベンチマーク実行時間の大半を占める倍精度行列積（DGEMM）の最適化手法について述べた．DGEMM カーネルの設計は，キャッシュ容量，TLB エントリ数，システムバス負荷の均等化を考慮して行った．それに合わせて DGEMM カーネルの呼び出し処理を設計した．実装はオープンソースの数値演算ライブラリ ATLAS をベースに行った．本最適化の結果，既存ライブラリの中で最速の GOTO BLAS の性能を上回り，1 プロセッサの Linpack ベンチマークで 4.13 GFlops（実行効率：86.0%）を記録した．DGEMM そのものの性能を測定する正方行列積プログラムでは 4.33 GFlops（実行効率：90.2%）と，90%を超える実行効率を記録した．さらに 4 ノード/8 プロセッサの PC クラスタにおいて，ネットワークが GbE であっても 26.9 GFlops（実行効率：70.1%）の性能が出ることを示した．

また，ラージページに関しても評価し，Linpack ベンチマーク性能が実行効率で 2.1 ポイント向上することを確認し，行列積にはラージページが有効であることを示した．

今回は Xeon プロセッサを最適化の対象としたが，Itanium2 プロセッサや Opteron プロセッサに対しても本最適化手法は有効であると考える．

参考文献

1) Super Computer TOP500. http://www.top500.org/
2) BLAS (Basic Linear Algebra Subprograms). http://www.netlib.org/blas/
3) Gunnels, J.A., Henry, G.M. and van de Geijn, R.A.: A Family of High-Performance Matrix Algorithms, *Computational Science — 2001*, *Part I*, Lecture Notes in Computer Science 2073, pp.51–60, Springer (2001).
4) Whaley, R.C., Petitet, A. and Dongarra, J.: Autometed Empirical Optimization of Software and the ATLAS project. http://math-atlas.sourceforge.net/
5) Petitet, A., Whaley, R.C., Dongarra, J. and Cleary, A.: HPL — A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. http://www.netlib.org/benchmark/hpl/
6) IA-32 Intel Architecture Software Developer's Manual. http://developer.intel.com/
7) IA-32 Intel Architecture Optimization. http://developer.intel.com/
8) Goto, K. and van de Geijn, R.: On Reducing TLB Misses in Matrix Multiplication, FLAME Working Note #9, The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-2002-55 (2002).
9) Message Passing Interface Forum. http://www.mpi-forum.org/
10) 笹生 健，松岡 聡：HPL のパラメータチューニングの解析，ハイパフォーマンスコンピューティング，pp.91-22 (2002).
11) Vector Signal Image Processing Library. http://www.vsipl.org/
12) 西村祥治，荒木拓也，蒲地恒彦：Pentium III, Pentium4 における LINPACK ベンチマークチューニング手法，*SACSIS2003*，pp.129–136 (2003).
13) Intel Math Kernel Library. http://www.intel.com/software/products/mkl/
14) High-Performance BLAS by Kazushige Goto. http://www.cs.utexas.edu/users/flame/goto/
15) Linux Super Page. http://shimizu-lab.dt.u-tokai.ac.jp/lsp.html
16) SCore Cluster System Software. http://www.pccluster.org/

(平成 16 年 1 月 31 日受付)
(平成 16 年 5 月 9 日採録)

成瀬　彰（正会員）

1996 年名古屋大学大学院工学研究科修了（情報工学専攻）．同年富士通（株）入社．(株）富士通研究所にて IA サーバに関わる研究開発に従事．並列処理，性能最適化，計算機アーキテクチャに興味を持つ．

住元　真司（正会員）

1986 年同志社大学工学部電子工学科卒業．同年（株）富士通入社．(株）富士通研究所にて並列オペレーティングシステム，並列分散システムソフトウェアの研究開発に従事．1997 年より新情報処理開発機構に出向．コモディティネットワークを用いた高速通信機構の研究開発に従事．2002 年より（株）富士通研究所にて高速通信機構の研究開発に従事，並列分散システムのアーキテクチャ，システムソフトウェア等に興味を持つ．平成 12 年度情報処理学会論文賞受賞，工学博士（慶應義塾大学大学院理工学研究科）．

久門　耕一（正会員）

1979 年東京大学工学部電気工学科卒業．1981 年同大学大学院電子工学専門課程修士課程修了．1984 年同課程博士課程中退．同年（株）富士通研究所入社．現在，同社 IT コア研究所に所属．CPU，メモリ，並列計算機アーキテクチャに関する研究に従事．GCC，Linux カーネル等の改良にも興味を持つ．日本ソフトウェア科学会会員．