社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－ARC－167（7）
2006－HPC－105（7）
2006／2／27

# 大規模SMPクラスタにおける固有値ライブラリの通信最適化について

今村俊幸[†]　山田進[††]　町田昌彦[††]

分散メモリ型並列計算機での密行列固有値計算において, その主要な部分をなすハウスホルダー 3 重対角化や逆変換の bcast, allreduce, 行分割から列分割への再分散操作など集合通信のコストが大きい. 1 ノードが 128 プロセッサからなる SGI Altix3700Bx2 などの, 大規模な共有メモリ並列計算機やそのクラスタでは, フラット MPI による並列化が性能的に優れている場合がある. ベンダ提供の MPI は集合通信においてかなりのチューニングがされていると期待したいが, 著者らの経験からはまだまだ不十分といえる. 本研究は, copy, split-merge 操作を再帰的に組み合わせかつ, Altix の結合トポロジーを考慮した通信順序の選択により, bcast, allreduce 等の集合通信の最適化を図るとともに, 集合通信に対する自動チューニング展開への足がかりとする.

# Optimized Communication on a Large SMP Cluster for an Eigenvalue Library

Toshiyuki Imamura,[†] Susumu Yamada[††]
and Masahiko Machida[††]

On solving an eigenvalue problem for dense-matrices on a distributed-memory parallel computer, time-cunsuming parts of Householder transform and its back-transform are collective communication, bcast, allreduce and row-column data-redistributions. As some large SMP cluster systems, on which 100+ processors are installed like an SGI Altix3700Bx2, offer better performance on a flat-MPI programming, vendor supplied MPI libraries are expected to perform excellently, however, it shows insufficient performance. This paper covers, the construction of collective communications by recursive use of copy, split and merge operations, communication optimization of bcast and allreduce with a consideration of the network topology, and an idea of the auto-tuning methodology for the collective communcation.

## 1. はじめに

大規模な SMP クラスタがスパコンの標準的なアーキテクチャになり, Dual-Core/Dual-CPU のクラスタシステムが比較的安価に構成可能となってきたが, 標準的なプログラミングスタイルは分散メモリ型並列計算機でのメッセージパッシングが継承されることが多い. これは, OpenMP などのスレッドによるプログラミングにおいて, データのローカリティが依然性能を左右していることに因している.

特に, 本研究において開発を進めている Altix 上の固有値ライブラリではスレッドによる開発は困難と判断される. 主な理由は first touch policy によって格納されたメモリを常に同一のスレッドがアクセスするわけではなく, ライブラリからはそれを知ることが極めて困難だからである. そのため, 共有メモリ型並列計算機ではあるものの, 場合によっては (実はかなり多くのアプリケーションで) 分散メモリ型計算機とみなした方が性能を引き出しやすい傾向にある.

この様な背景のもとで, 高速な固有値計算ライブラリを作成する場合に, 通信ライブラリ (MPI) のコストを減らすことも同時に考えていかなくてはならなくなった. 後述するが, アプリケーションでは集合通信が重要な位置を占めることが知られているが, ベンダ提供の MPI の集合通信が十分なものなのかまずは疑っていきたい. 従来アルゴリズムと新規アルゴリズムの整理, 近年自動チューニング集合通信ライブラリと呼ばれる枠組みを視野に入れながら, SMP クラスタ上の高速な固有値ライブラリ開発について言及する.

## 2. 固有値計算に現れる集合通信

近代的な高性能アルゴリズムが開発され, 現在, 実対称行列の固有値計算での主要コストは前処理 Householder 三重対角化と逆変換となっている. その内, 主要な計算部分は, 行列-ベクトル積, 2m ランク更新, 行

† 電気通信大学電気通信学部情報工学科
Department of Computer Science, The University of Electro-Communications
†† 日本原子力研究開発機構システム計算科学センター
CCSE, Japan Atomic Energy Agency

**表 1** MPI 版固有値ライブラリ (Householder) の通信コスト内訳 (Altix3700Bx2 で 32CPU 使用)

| 次元 | N=1000 | N=4000 | N=6000 |
|---|---|---|---|
| 計算全体 (秒) | .187 | 1.745 | 4.303 |
| Bcast (秒 (%)) | .030(13) | .140(8.0) | .247(5.7) |
| Allreduce (秒 (%)) | .081(43) | .561(32) | 1.158(26) |
| 再分散 (秒 (%)) | .021(11) | .218(12) | .442(10) |

**表 2** 集合通信アルゴリズム, 文献 9) より抜粋

| 集合通信 | アルゴリズム |
|---|---|
| Broadcast | 逐次, Chain, Binary, Binomial |
| Scatter | 逐次, Chain, Binary, Binomial |
| Gather | 逐次, Chain, Binary |
| Reduce | Gather+演算, Chain, Binary, Binomial, Rabenseifner |
| Allreduce | Reduce+bcast, Allgather+演算, Chain, Binary, Binomial, Rabenseifner |
| Allgather | Gather+bcast |
| Alltoall | Gather+scatter, 巡回 |

列-行列積である. Hendrickson[1] や Katagiri[2] らの報告にあるように, i) 行列-ベクトル積の並列計算には Allreduce が必要で, その他, ii) reflector 共用のため Broadcast, iii) 2 次元分割を行う場合には行-列分割間での再分散機能が必要となる (場合によってはscatter, allgather を用いる). いずれにしても, 集合通信が重要な位置を占める. 表 1 は Householder 変換を Altix3700Bx2(32CPU) で実行した際の, MPI 関数ごとの内訳である (ただし, 再分散は関数単体ではない). 集合通信, 特に allreduce のコストが多くの部分を占めていることは明らかである. CPU 数が増加すれば, この割合も増加する. バンド幅が比較的広い Altix ですらこういった状況にある. つまり, 帯域の厳しい多くのアーキテクチャで通信最適化の緊急性が高い.

## 3. 集合通信のアルゴリズム

### 3.1 従来アルゴリズム

集合通信のアルゴリズムは Vadhiyar の論文 9) によくまとめられている. 多くの MPI 実装はこの中の何れかを, 固定もしくは選択的に使用している.

### 3.2 基本操作とアルゴリズム

$\log_2 P$(P:プロセス数) の複雑さで実現される集合通信アルゴリズムが知られているが, それを系統的に整理し全アルゴリズムを構成することを行っていく.

[**Notation**]

- $\mathsf{R}$: 集合通信におけるルートプロセスを意味するが, 本稿ではあるステージにおける送信元プロセスの集合をさす.
- $P = 2^p$: 集合通信に参加するプロセス数.
- $A_{[i_1:i_2]}@r$: ランク $r$ のプロセス上に存在するデータ構造 $A$ を指す. 下添字は配列要素の範囲.
- $A \mapsto B$: $A$ から $B$ へのデータ通信を意味する.

[**基本操作**]

- $\mathsf{C}^{(d)}$ (Copy): ルートプロセス ($r \in \mathsf{R}$) は $2^d$ 離れたプロセス ($r' = r \oplus 2^d$) へ全データを送信.

$$\mathsf{C}^{(d)} : A_{[*]}@r \mapsto A'_{[*]}@r'. \tag{1}$$

- $\mathsf{S}^{(d)}$ (Split): ルートプロセス ($r \in \mathsf{R}$) は $2^d$ 離れたプロセス ($r' = r \oplus 2^d$) に対して, $f_{r,d}[1:N]$ で決まるデータ範囲を送信する.

$$A_{f_{r,d}[1:N]}@r \mapsto A'_{f_{r,d}[1:N]}@r'. \tag{2}$$

- $\mathsf{M}^{(d)}$ (Merge): ルートプロセス ($r \in \mathsf{R}$) は $2^d$ 離れたプロセス ($r' = r \oplus 2^d$) に全データを送信し, 受信側は $f_{r,d}[1:2N]$ で決まる範囲に格納する.

$$A_{[1:N]}@r \mapsto A'_{f_{r,d}[1:2N]}@r'. \tag{3}$$

ここで, 範囲を返す関数 $f_{r,d}$ は次のように定義する.

$$f_{r,d}[1:N] = \begin{cases} [1:\frac{N}{2}] & \bar{r} < 2^d \\ [\frac{N}{2}+1:N] & \bar{r} \geq 2^d \end{cases}$$
$$\bar{r} := \mathrm{mod}(r, 2^{d+1}) \tag{4}$$

上記操作の直後に受信データに指定された演算 $o$ を作用させ $A$ を更新する ($A \leftarrow o(A, A')$). また, 上記操作の直後に, $\mathsf{R}$ に対して受信側プロセス ($r'$) の追加や, 特定の送信側プロセスの削除などの操作を指定する.

これら基本操作を組み合わせることで, broadcast や reduction(gather や scatter なども含めた) の集合通信全般に関する従来型アルゴリズムを定式化することができる. ただし, 上記基本操作は 2 ベキ以外の距離にあるプロセスには送受信をしないので, 全てのアルゴリズムを表現できるわけではない.

#### 3.2.1 Bcast

Broadcast(bcast) は, $o(A, A') := A'$ とし, 受信側のプロセスを全て $\mathsf{R}$ に加えることで実現される. 初期状態で $\mathsf{R} = \{0\}$ とする☆. 以下, bcast において $o$, $\mathsf{R}$ の更新方法, 初期状態は同一とする.

[**Broadcast(Binomial tree)**]

$$\mathsf{Bcast} := \mathsf{C}^{(p-1)} \cdots \mathsf{C}^{(0)} = \prod_{i=0}^{p-1} \mathsf{C}^{(i)} \tag{5}$$

受信プロセスは必ずしも昇順である必要がなく, $\mathsf{C}$ が連続する場合は順序を変更してもよい.

場合によっては, binomial tree アルゴリズムよりも次に挙げる Rabenseifner[5] のアルゴリズムの方が優

☆ ルートが 0 でない場合は, そのランクとの排他的論理和をとり新たなランクとすることで結合トポロジーを変更することなく, ルートを 0 にできる.

れている (本来 reduce 用だが bcast にも適用可能).

**[Broadcast(Rabenseifner)]**

$$\mathsf{Bcast} := \mathsf{M}^{(0)} \cdots \mathsf{M}^{(p-1)} \mathsf{S}^{(p-1)} \cdots \mathsf{S}^{(0)} = \prod_{i=p-1}^{0} \mathsf{M}^{(i)} \prod_{i=0}^{p-1} \mathsf{S}^{(i)} \quad (6)$$

本アルゴリズムはレイテンシが 2 倍に増大するため, ある程度メッセージサイズが大きな領域でなければ適用されない. 通信発行回数を削減しつつ通信コストを軽減するハイブリッドアルゴリズムが考えられる☆

**[Broadcast(Hybrid1)]**

$$\mathsf{Bcast} := \mathsf{M}^{(0)} \cdots \mathsf{M}^{(j)} \mathsf{C}^{(p-1)} \cdots \mathsf{C}^{(j+1)} \mathsf{S}^{(j)} \cdots \mathsf{S}^{(0)} = \prod_{i=j}^{0} \mathsf{M}^{(i)} \prod_{i=j+1}^{p-1} \mathsf{C}^{(i)} \prod_{i=0}^{j} \mathsf{S}^{(i)} \quad (7)$$

本ハイブリッドアルゴリズム 1 は $\mathsf{S}^{(*)}$ を $p-1$ 段まで行わず $j$ 段で止めておき, 中間に $\mathsf{C}^{(*)}$ 操作を挟むものである. C, S, M を混合させより一般化したアルゴリズムを次の様に定式化できる.

**[Broadcast(Hybrid2)]**

$$\mathsf{Bcast} := \mathsf{T}^{(p)} \mathsf{F}^{(p)} \quad (8)$$

$$\begin{cases} \mathsf{F}^{(i)} := \mathsf{C}^{(i)} \mathsf{G}_1 \mathsf{F}^{(i-1)}, \ \mathsf{T}^{(i)} := \mathsf{G}_2 \\ \mathsf{F}^{(i)} := \mathsf{S}^{(i)} \mathsf{G}_1 \mathsf{F}^{(i-1)}, \ \mathsf{T}^{(i)} := \mathsf{G}_2 \mathsf{M}^{(i)} \end{cases} \quad (9)$$

ここで $i > 0$ であり, $\mathsf{F}^{(0)} := \mathrm{id.}, \mathsf{T}^{(0)} := \mathrm{id.}$ とする. 式 (9) の上下 2 通りいずれかを選択し, $\mathsf{G}_1\mathsf{G}_2 = \mathsf{T}^{(i-1)}$ を満足する任意の組み合わせに対して再帰的に定義することを意味する.

本アルゴリズムを用いると, プロセス数 4 ($p = 2$) で CC, MSC, CMS, MCS, MSMS, MMSS の 6 種類 (図 1). $p = 3$ で CCC, MSCC, CMSC, MCSC, MSMSC, MMSSC, CCMS, MSCMS, CMCS, MCCS, MSMCS, MMSCS, CMSMS, MCSMS, MSMSMS, MMSSMS, CMMSS, MCMSS, MMCSS, MSMMSS, MMSMSS, MMMSSS の 22 種類, $p = 4$ では 90 種類となる.

#### 3.2.2 Reduce

Reduce(reduction) では, Bcast と同様の送信方法が適用できるが, 更新演算 $o$ に簡約演算に指定された演算, R の初期状態を全プロセス, さらに R 更新ルールは '送信プロセスを集合から除き受信プロセスを集合に加える' で定式化できる.

#### 3.2.3 Allreduce

Allreduce は, 更新演算 $o$ に簡約演算に指定された演算, R の初期状態を全プロセス, さらに R の更新ルールは全プロセスを常に有効とすることで定式化できる.

### 3.3 Gather/Allgather

Gather は, $\prod_{i=0}^{p-1} \mathsf{M}^{(i)}$ に対して Reduce と同じ R 更新ルールを適用することが最も単純なアルゴリズムである. また, R の更新ルールを '常に有効にする' を用いることで Allgather となる.

#### 3.3.1 Scatter

Scatter は, $\prod_{i=p-1}^{0} \mathsf{S}^{(i)}$ で実現可能である (R 更新ルールは Bcast と同様).

#### 3.3.2 そ の 他

本稿で詳細は省略するが, MPI で規定されている All 系のその他の集合通信についても, 同様に基本操作の組み合わせで定式化は可能である. Tipparaju[12] らの提案する All-to-all 集合通信アルゴリズムでは, ランクの付け替えを行えば S と M の混合と解釈できる.

### 3.4 最適パラメータの選択

前節で示したように, Bcast のみでも通信アルゴリズムが多数存在する. Vadhiyar らが提唱する集合通信ライブラリの自動チューニングの枠組み[8] では, { 集合通信, PE 数, メッセージサイズ, アルゴリズム } を指定した場合の最良セグメントサイズ候補を決定し☆☆, その情報をもとに { 集合通信, PE 数 } に対する最良アルゴリズムでのサンプリングを行う (最終的にはこの 2 フェイズを繰り返し, 最良パラメータの精度を上げていく). この様に, 自動チューニングの枠組みには個々のパラメータに応じた性能測定が不可欠であり, 前節で示した集合通信の形式化のもとでは, アルゴリズムそのものが PE 数に応じて爆発的に増加するため性能測定が非現実的となる.

一方, 予めパラメータ探索空間を絞り込むための性能モデルを導入し, 性能測定に必要なサンプリング数を削減することが自動チューニングのみならず集合通信通信ライブラリ作成における重要な部分となる[4),8),10)].

本研究で対象とするシステムの一つ Altix3700Bx2 は NUMA であるためローカルとリモートのメモリアクセスではそのコストは大きく異なる. 分散メモリ型並列計算機の多段ネットワークの結合とほぼ同等と考えてよい. 各ノードは 2CPU が主記憶を共有する形で SHUB コントローラに接続する. さらに, 4 ノードが Router ユニットに NUMAlink4(6.4GB/s) で結合され CR ブリックをなしている. 各 CR ブリックは 8 個の NUMAlink4 ルータポートを持ち, 外部ルータと接

☆ Barnet[6] らのハイブリッドアルゴリズムもこのアルゴリズムと考えられる.

☆☆ Vadhiyar らの手法ではパイプライン方式の送信を基本とするため, セグメントサイズが重要となる

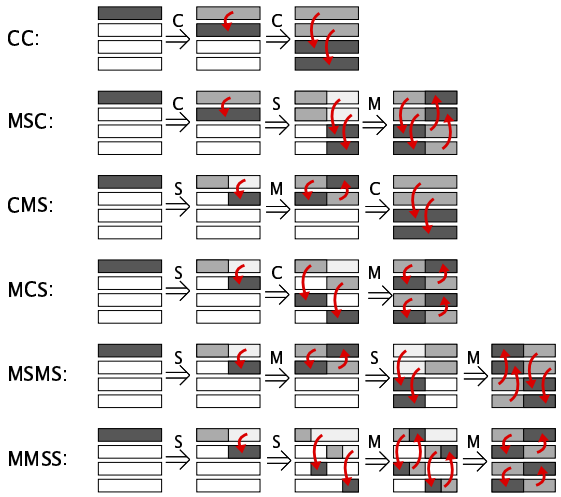


**図 1** $p = 2$ における Hybrid2 アルゴリズムでの Bcast の全通信パターン

続し NUMA を構成する[13),14)] さらに, 複数 NUMA システムが結合する☆.

したがって, 送受信モデルは

- SHUB を共有するレベル,
- CR ブリックを構成するレベル,
- それ以外の NUMA 内レベル,
- NUMA システム間レベル

の 4 段階を用意しなければならない.

また, 単方向の送受信についても 1 プロセス対 · 複数プロセス対が送受信する場合を考える必要がある. 双方向の送受信も同様に考えなくてはならない. Altix3700Bx2(Itanium2 1.6GHz, 128CPU NUMA システム) で,

- 単方向送受信 (1 プロセス対)
- 単方向送受信 (複数プロセス対)
- 双方向送受信 (複数プロセス対)

について, プロセス間距離 1,2,4, ···, データ長 1,2,4, ··· と 2 のベキ語長 (1 語は 8B) で変化させた際の通信時間を測定したものが図 2 である. これらの結果から, Altix3700Bx2 の通信モデルを線形モデル $(\alpha + \beta x)$ で近似するとパラメタ $\alpha, \beta$ は表 3 となる. もう一つのプラットフォーム PC クラスタ (Xeon-D 2.8GHz, 4 ノード, GbE スイッチ結合) での 1) ノード内 2PE, 2) ノード間通信のパラメタも表 3 に併せて示した.

これら通信モデルをもとに, 各アルゴリズムに対して通信時間を予測できる. 具体的には, { データ長, PE 数, 集合通信 } が与えられたときの各送信アルゴリズの通信時間を算出することになる. 全アルゴリズムの中からそれを最小とするアルゴリズム (C,S,M の組み合わせ列) を最良アルゴリズムとして決定すればよい.

ランクの決め方もパラメタ探索空間に影響するので, 考慮が必要である. 全パターンで $P!$, 結合トポロジを保持する場合は $p!$ の可能性が存在する. MPI が初期に振り分けるランクにはある種の結合トポロジーを反映しているはずである. そこで, パラメタ探索において厳密性には欠けるが, 次の 2 種類のランクのつけ方に限定しパラメタ探索を実施する.

(1) **FFNL(Far First Near Last):** ハイパーキューブ網のトポロジ下で物理的に遠方のリンクから近接リンクの順にランクを振り分ける.

(2) **NFFL(Near First Far Lasr):** FFNL とは逆に物理的に近接するリンクから遠方リンクの順にランクを振り分ける.

### 3.5 ノード間 · ノード内通信アルゴリズム選択の戦略/パイプライン処理

一般的にノード間通信とノード内通信のコストは大きく異なるため, ノード間通信とノード内通信を並行処理 (オーバラップ) させることが多くの論文や実装系で検討されている. この場合, CPU 負荷やバッファ溢れの問題を解消するために, データをセグメント化しパイプライン処理することがされている. 本稿では, ノード間通信を特に特別視することなくデータの一括送受信を行う実装としている.

送受信アルゴリズムについても, ノード間/ノード内を区別して行うべきという研究成果が報告されてい

☆ ここで, NUMA は共有メモリシステムの最大構成部分を指すものとする. 1 筐体 (40U ラック) には最大 8CR ブリック (64CPU) が搭載される. また, CR ブリック間結合は多段ファットツリートポロジのデュアル構成となる. NUMA 間も NUMAlink4 での結合である.

**表 3** 線型モデル $\alpha + \beta x$ による性能モデル $(\alpha, \beta)$, 左: Altix3700Bx2, 右:PC クラスタ.

| | $d = 0$ | $0 < d \leq 2$ | $2 < d$ | $d = 0$ | $0 < d$ |
|---|---|---|---|---|---|
| 1) 単方向 (1 プロセス) | (2.77, 4.79E-3) | (4.05, 7.61E-3) | (4.46, 8.37E-3) | (25.4, 26.5E-3) | (111., 103.E-3) |
| 2) 単方向 (複数プロセス) | (2.91, 4.60E-3) | (3.92, 9.96E-3) | (3.92, 22.13E-3) | (35.0, 24.32E-3) | (111., 164.E-3) |
| 3) 双方向 (複数プロセス) | (3.34, 8.71E-3) | (5.43, 13.08E-3) | (5.88, 26.89E-3) | (26.8, 41.2E-3) | (63.6, 294.E-3) |

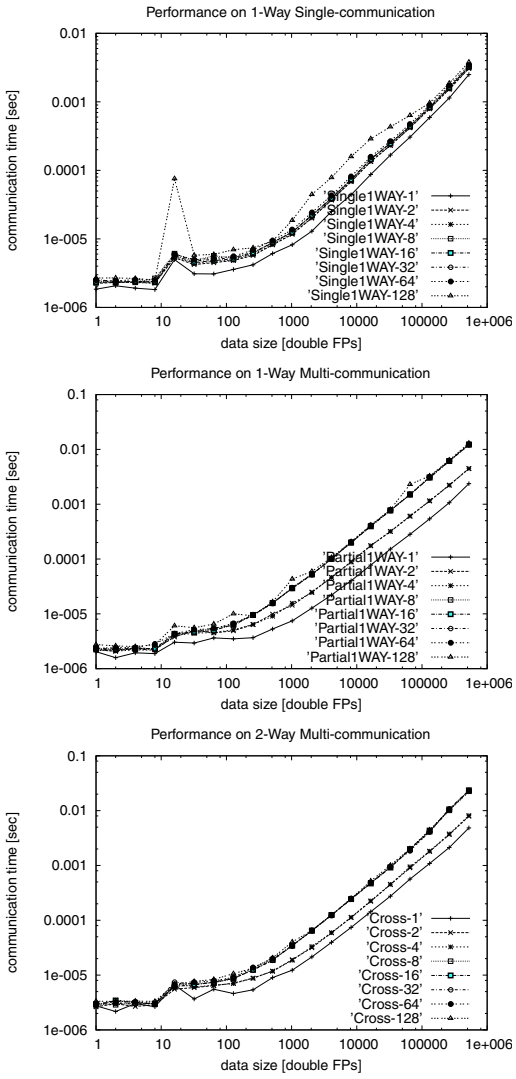


**図 2** Altix3700Bx2 の MPI 通信性能

る[3),4)]. PC クラスタや地球シミュレータの様に SMP の構成要素プロセッサが全て同時に通信を行えない場合には有効に働く可能性がある. 一方, Altix は NUMA を構成する 128CPU が同時に他の NUMA に通信を行っても十分な帯域を確保しているため, ノード間通信を特別視する必要はない.

ノード間 · ノード内の通信手法の区別が必要な場合は, ノード数, ノード番号, ノード内プロセス番号をそれぞれ $N_{\mathrm{node}}$, $x$, $y$ としたとき, ランクを $x + yN_{\mathrm{node}}$ で定義すればよい. 前節の枠組みでカバーできる (図 1 の MSMS が, DualCPU-2 ノードに対応する).

## 4. Altix3700Bx & PC クラスタでの測定

通信モデル $t = \alpha + \beta x$ がプロセッサ間距離に無関係と仮定したときに, 全検索から最適アルゴリズムは Hybrid1 に限定されるという結果を得た. 本実験では各並列計算機のネットワーク特性に応じたアルゴリズム構成の中から, 最良アルゴリズムを探索し集合通信を測定しベンダ MPI(PC クラスタでは MPICH) と性能比較を実施する.

### 4.1 Bcast on Altix3700Bx2

Hybrid1 形式をリファレンスアルゴリズムとして, アルゴリズムの探索を実施したがいずれのケースでも Hybrid1 形式が最良となった. また, ランクのつけ方は NFFL の方がよい結果であった. 128CPU, 256CPU(128CPU の NUMA2 システム) での最良アルゴリズムの測定結果が図 3 の上段である. ベンダ提供のものと比較して 3 倍高速である.

### 4.2 Allreduce on Altix3700Bx2

Bcast 同様の結果が allreduce でも得られたが, 同一メッセージサイズであっても bcast とはアルゴリズム (パラメタ $j$) が異なる. ベンダ提供のものと比較して最大 7 倍程度高速である (図 3 中段).

### 4.3 Bcast/Allreduce on Xeon-D PC-cluster

PC クラスタでは経験的に最良とされるノード内 · ノード間アルゴリズムの組み合わせを実施した (区別しないアルゴリズムは最良ではなかった). Bcast は MPICH(1.2.7) よりも最大で 2.5 倍, allreduce は 1.4 倍高速となった (図 3 下段).

### 4.4 固有値ライブラリでの性能改善

表 1 で示した結果が, どの様に改善されるかを示したのが表 4 である. 通信の内訳は単一プロセスの通信所要時間のため増加を示したものがあるが, N=6000 では全体の計算時間は 14% 減少している. これだけの計算時間の削減はループ最適化では難しく, 通信最適化による性能改善の余地があることを示している.

## 5. ま と め

集合通信のコスト削減の余地が十分あり, 今後も開発を継続すべきと認識された. アルゴリズムの選択

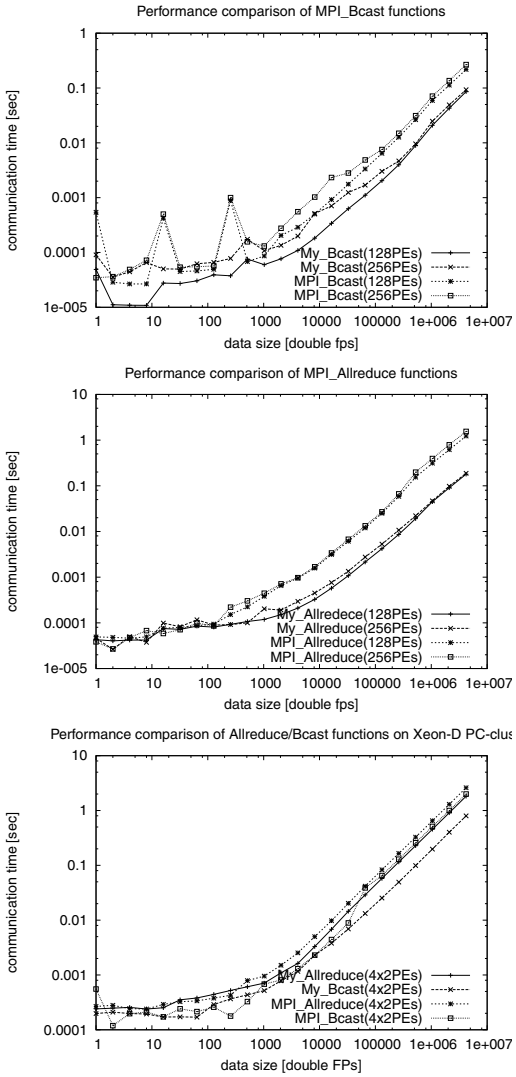


**図 3** 最良アルゴリズムとベンダ MPI(MPICH) との性能の比較, (上段: Bcast on Altix, 中段: Allreduce on Altix, 下段: Bcast と Allreduce on PC クラスタ)

**表 4** 固有値ライブラリの性能改善 (Altix3700Bx2 32CPU)

| 次元 | N=1000 | N=4000 | N=6000 |
|---|---|---|---|
| 計算全体 (秒) | .180 | 1.519 | 3.709 |
| Bcast (秒 (%)) | .029(16) | .183(12) | .333(8.9) |
| Allreduce (秒 (%)) | .074(41) | .426(28) | .836(23) |
| 再分散 (秒 (%)) | .021(11) | .205(13) | .410(11) |

などをカプセル化し, NIC のオフロード機能に任せるなど研究が発展する可能性は大きい. また, パラメタの自動探索機能 (性能測定機構も含めて) による自動チューニング機能など従来のループ最適化に加えて今後実施していく予定である.

## 参 考 文 献


1) B. Hendrickson, E. Jessup, C. Smith: Towards on Efficient Parallel Eigensolver for Dense Symmetric Matrices. SIAM J. Sci. Comput. 20(3) 1132-1154 (1999)
2) T. Katagiri, Y. Kanada: An Efficient Implementation of Parallel Eigenvalue Computation for Massively Parallel Processing. Parallel Computing, 27(14) 1831-1845 (2000)
3) S. Sistare, et al.: Optimization of MPI Collectives on Clusters of Large-Scale SMP's. Proc. of SC'99 (1999)
4) Wu M.-S., et al.: Performance Modeling and Tuning Strategies of Mixed Mode Collective Communications. Proc. of SC|05 (2005)
5) R. Rabenseifner: A New Optimized MPI Reduce Algorithm. `http://www.hlrs.de/mpi/myreduce.html`
6) M. Barnett, et al.: Interprocessor Collective Communication Library (InterCom). Proc. of Scalable High Performance Computing Conference, 357-364 (1999)
7) G.E. Fagg, S.S. Vadhiyar, J.J. Dongarra: ACCT: Automatic Collective Communication Tuning. Dongarra J. (eds): EuroPVM/MPI 2000, LNCS **1908**, 354-361 (2000)
8) S.S. Vadhiyar, G.E. Fagg, J.J. Dongarra: Performance Modeling for Self Adapting Collective Communications for MPI. Proc. of LACSI Symposium 2001 (2001)
9) S.S. Vadhiyar, G.E. Fagg, J.J. Dongarra: Towards an Accurate Model for Collective Communications. The International Journal of High Performance Cumputing Applications, 18(1) 159–167 (2004)
10) B.V. Protopopov, Anthony Skjellum: Shared-memory Communication Approaches for an MPI Message-passing Library. Concurrency - Practice and Experience, 12(9) 799-820 (2000)
11) V. Tipparaju, J. Nieplocha, D. Panda: Fast Collective Operations Using Shared and Remote Memory Access Protocols on Clusters. Proc. of IPDPS'03 (2003)
12) V. Tipparaju, J. Nieplocha: Optimizing All-to-All Collective Communication by Exploiting Concurrency in Modern Networks. Proc. of SC|05 (2005)
13) SGI Altix 3700 Bx2 User's Guide. Rev. 2, SGI (2004)
14) D.J. Mavriplis, M.J. Aftosmis, M. Berger: High Resolution Aerospace Applications using NASA Columbia Supercomputer. Proc. of SC|05 (2005)