2007－HPC－111 (33)
2007／8／2

負荷ばらつきを考慮した MPIブロードキャスト通信の動的最適化に関する研究

栗原 康志[†1†2] 曽我 武史[†3] Hyacinthe Nzigou Mamadou[†1] 南里 豪志[†4]
末安 直樹[†5] 松本 透[†5] 井上 弘士[†6] 村上 和彰[†6]

[†1] 九州大学大学院システム情報科学府
[†2](財) 九州システム情報技術研究所
[†3] 財団法人福岡県産業科学技術振興財団
[†4] 九州大学情報基盤研究開発センター
[†5] 富士通株式会社ソフトウェア事業本部ミドルウェアコンポーネント事業部
[†6] 九州大学大学院システム情報科学研究院

概要
本研究では, 負荷バランスの不均衡によって生じるプロセス毎の到着遅れがブロードキャスト通信の性能を低下させる問題に注目し, それを解決する手段として, 負荷状況に応じて通信順序を調整する動的最適化手法を提案した. この手法では, まずプロセスの負荷情報としてルートプロセスからの到着時刻の遅れ時間を算出し, それに応じてブロードキャスト通信アルゴリズム内の仮想ランクへのプロセス割り当てを変更する. 本稿では, 提案手法のプロトタイプをPCクラスタ環境に実装し, 負荷の状況によって最大で40%程度, ブロードキャスト通信の性能を向上できることを確認した. さらに, 提案手法のブロードキャストを疎行列計算に適用することにより, 通信時間を最大25%程度削減できることを確認した.

Dynamic optimization of broadcast communication according to the load balance

Kouji Kurihara, [†1†2] Takeshi Soga, [†3] Hyacinthe Nzigou Mamadou, [†1] Takeshi Nanri, [†4] Sueyasu Naoki, [†5] Toru Matsumoto, [†5] Koji Inoue [†6] and Kazuaki Murakami [†6]

[†1]Graduate School of Information Science and Electrical Engineering, Kyushu University
[†2]Institute of System & Information Technologies/KYUSHU
[†3]Fukuoka Industry, Science & Technology Foundation
[†4]Research Institute for Information Technology, Kyushu University
[†5]Middleware Components Division, Software Unit, FUJITSU LIMITED
[†6]Graduate School of Information Science and Electrical Engineering, Kyushu University

Abstract
This work focuses on the problem that the load imbalance can decrease the performance of broadcast communication. To avoid the problem, the authors proposed a technique of optimization that adjusts the order of communications in a broadcast at runtime. In this technique, the information of the delay of each rank from the root rank is used to decide the optimal order. In This paper, a proto-type of this technique was implemented on a PC cluster, and showed that the optimization decreased the by 40% at maximum. In addition to that, it was confirmed to be able to reduce the communication time by about 25% or less by applying the broadcast of the proposal technique to the sparse matrix calculation.

1 はじめに

分散メモリ型並列計算機では，プロセッサ間の通信が性能のボトルネックとなることが多く，通信時間の削減が重要となる．特に複数プロセスで構成されるグループ全体で行う集団通信はプロセス数の増加に伴い通信の所要時間が増大するので影響が大きい．MPICH[2] 等，多くのMPI実装では集団通信を一対一通信の組み合わせで実現している．通常，これらの集団通信のアルゴリズムではプロセスが集団通信を呼ぶ順番に関係なく通信順序が決められている．各プロセスがほぼ同時に通信を開始することができる場合はこの通信順序で問題ないが，プロセスの負荷の不均衡のためプロセスごとに通信開始時刻が大きくずれている場合，通信開始時刻の遅いプロセスのために発生した遅延が他のプロセスに伝搬するため，性能が大きく低下する場合がある．

そこで本研究では，Binomial Treeによるブロードキャスト通信を対象としてプロセス毎の通信開始時刻のずれによる性能低下の改善手法を提案する．この手法は，各プロセスの負荷を考慮して，最適な通信順序で通信を行うことにより，ブロードキャスト通信の性能改善を図る．本稿では，提案手法による動的最適化機構を備えたブロードキャスト関数を実装し，PCクラスタを用いて，最適化の効果を検証する．

本稿の構成は以下の通りである．第2章で提案手法について説明する．第3章で実装方法を示し，第4章で実験について説明し，第5章で実験結果の評価，考察を述べる．最後に第6章でまとめ，今後の課題について述べる．

2 到着時刻に応じた通信順序入れ替えによるブロードキャスト通信の高速化

2.1 Binomial Treeによるブロードキャスト通信

ブロードキャスト通信とは一つのプロセスが持っているデータをグループ内の全てのプロセスに配布する集団通信である．本稿では，ブロードキャスト通信のアルゴリズムとして，MPICH等で採用されているBinomial Treeを対象とする．Binomial Treeは，P個のプロセスにおけるブロードキャストを$logP$回のステップで行う．各ステップではデータを持っているプロセスがデータを持っていないプロセスにデータを送付する．このアルゴリズムではプロセスを仮想的な番号に割り付け，通信を行う．この番号を仮想ランクと呼ぶ．一方，MPIプログラムではプロセスの識別子としてランクが割り当てられる．本稿では，このランクを実ランクと呼ぶ．MPIの実装ではこの実ランクと集団通信内部の仮想ランクとの対応を，相対的に行う．対象のデータを最初に所有しているプロセスの実ランクを仮想ランク0に割り当て，他の実ランクはルートプロセスの実ランクからの相対位置で割り当てを決定する．図1に16プロセスによるBinomial Treeの仮想ランクの例を示す．各節点の番号は仮想ランクを表し，右肩の数字がデータの配布されるステップである．

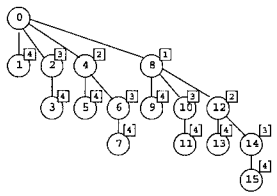

図1: Binomial Treeアルゴリズム

2.2 プロセス負荷の不均衡によるブロードキャスト通信時間への影響

図2にプロセス数が4の場合のブロードキャストの様子を示す．プロセスの負荷が均等な場合，通信の開始が同時に行われるので実ランクと仮想ランクの対応が性能に影響することはない．しかし，ブロードキャスト通信の直前でプロセスの負荷が不均等な場合，通信の開始にずれが生じ，集団通信の完了が遅くなることがある．例えば，最初にデータが配布されるプロセスがブロードキャスト通信直前で他のプロセスより多くの処理を行う場合(図3)，データの受信が遅れるため，次のプロセスにデータを配布するのが遅れる．このように，負荷の状況によってブロードキャスト通信の通信時間が長くなる．一方，最後にデータが配布されるプロセスがブロードキャスト直前で他のプロセスより多くの処理を行う場合(図4)，データが送られてくるまでその処理を行うことができる．さらに，そこで通信の遅れが生じたとしても他のプロセスへの通信の遅れの影響が少ないのでブロードキャスト通信時間への影響も小さい．

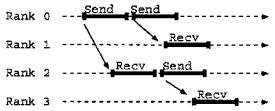

図2: 負荷が均等な場合

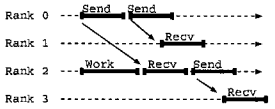

図3: データを中継するランクの負荷が重い場合

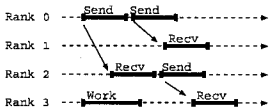

図4: データを中継するランクの負荷が軽い場合

2.3 仮想ランク最適化問題

第2.1項で述べたようにBinomial Treeによるブロードキャスト通信は仮想ランクを用いて行われる．また，第2.2項から各プロセスの負荷状況によりBinomial Treeによるブロードキャスト通信の性能が大きく変化することが分かった．しかしながら，従来手法ではルートプロセスからの相対距離により仮想ランクが決められている．そのため，各プロセスの負荷が考慮されないのでブロードキャスト通信の性能が大きく低下する場合がある．よって，各プロセスの負荷状況を考慮して各プロセスへ仮想ランクを適切に割り当てることによりこの問題点を解決できるので第2.2項で述べた性能低下を防ぐことができると考えられる．以下で各プロセスの負荷状況を考慮した最適仮想ランク割り当てを行う方法を述べる．

2.4 最適仮想ランク割り当て方法

早い段階でデータを中継する仮想ランクほど，遅延によってブロードキャスト通信全体の時間に与える影響が大きいので，中継の順序に応じて仮想ランクに優先度を設定し，遅延の小さいプロセスを優先的に配置する．

優先度は以下のように設定する．Binomial Treeでは子の数が多い節点の仮想ランクほどデータを配る相手が多いので，遅延によってブロードキャスト通信全体の時間に与える影響が大きい．したがって，早いステップでデータを受け取る仮想ランクの優先度を高く設定する．ステップだけで優先度を決定すると，複数プロセスに同じ優先度が設定される．しかしながら，Binomial Treeでは図2のランク1とラ

ンク 3 のように同じステップでもデータを受け取る時刻が異なる．これは，Send 開始から Recv 開始まで少なくともネットワークの遅延時間分遅れるのでルートプロセスからの転送回数が少ないランク 1 の方がメッセージ到着時刻が早くなるためである．したがって，同じステップの仮想ランク同士ではルートプロセスからの転送回数が少ない仮想ランクの優先度を高く設定する．以上をまとめると次のようになる．

- 早いステップでデータを受け取る仮想ランクほど優先度を高く設定する
- 同一ステップでデータを受け取る場合はルートプロセスからの転送回数が少ない仮想ランク (=仮想ランクの値が小さい) ほど優先度を高く設定する

3 ブロードキャストの動的最適化実装

第 2.4 節で述べた仮想ランクでブロードキャストを行うためのブロードキャストの実装方法を述べる．実装の概要は以下の通りである．

1. 仮想ランク表を用いてブロードキャストを実行
   - 初期は MPICH で実装されている相対距離による仮想ランクでブロードキャストを行う
2. 各プロセスにおけるブロードキャスト直前の負荷情報を取得する
3. 一定回数ブロードキャストを実行する毎にそれまでの各プロセスの負荷情報を収集し，最適な仮想ランク割り当てを決定して，仮想ランク表を更新する

3.1 負荷情報

ブロードキャスト直前の負荷状況を示す情報として，本研究ではブロードキャスト関数への到着遅れ時間を用いる．到着遅れ時間は各プロセスがルートプロセスからどれだけ遅れてブロードキャスト通信を開始したかを示す．各プロセスはブロードキャスト直前の負荷を処理し終えてからブロードキャスト通信を開始するので，この時間が大きいほど負荷が重いとみなすことができる．ルートランクのブロードキャスト通信開始時刻を S_0，ランク i のブロードキャスト通信開始時刻を S_i とすると，ランク i の到着遅れ時間 T_i は

$$T_i = S_i - S_0$$

で求めることができる．この到着遅れ時間を各プロセスで算出するため，本稿で実装したプロトタイプでは，ルートランクの通信開始時刻 S_0 をブロードキャストメッセージと並行して送信する．

3.2 仮想ランク表の生成．更新

本研究では仮想ランクへのプロセスの割り当てを動的に変更するため割り当て情報を示す仮想ランク表を用いる．そこで各プロセスの実ランクと仮想ランクの対応を示した仮想ランク表を各プロセスで生成，更新する．仮想ランク表の更新は全プロセスの到着遅れ情報を基に行う．

本稿の実装では，一定回数のブロードキャスト通信の度に MPI_Allgather 関数を用いて各プロセスに全プロセスの到着遅れ情報をコピーする．そして，その情報を基に 2.4 節で示した手法によって割り当てを決定し，各プロセスで仮想ランク表を作成する．生成した仮想ランク表は次回のブロードキャスト通信から適用される．

3.3 時刻同期

各ノードの時刻には一般的に数 msec 以上のずれがある．一方，メッセージサイズやプロセス数が小さい場合のブロードキャスト通信は数 msec 以下で終わる場合がある．したがって，ブロードキャスト通信の最適化に利用する到着遅れ時間の計算には少なくとも 1msec 未満の精度が求められるため各ノードの時刻を同期させる必要がある．本稿の実装では，NTP(Network Time Protcol) と同様のアルゴリズムを用いて，実ランク 0 と各ランクの時刻同期を行う．時刻同期の際には InfiniBand のような通信性能が安定した通信路の利用を想定している．この場合，2 μ sec 以下の精度で各ノードの時刻を同期させることができる．

3.4 動的最適化による効果の予測

本研究の提案手法はブロードキャスト実行中に取得する負荷情報を基に仮想ランク割り当てを決定するので，最適化が適用されるのはその次に行われるブロードキャスト通信からである．したがって，本研究の提案手法の効果が得られるプログラムはループ内のブロードキャストで同じ負荷状況がある程度続くプログラムに本手法が有効であると考えられる．

4 PC クラスタによる実験

本節では，実環境においてオーバーヘッドを含めた本研究の提案手法の有効性を確認するためにベンチマークプログラムを用いた実験を行う．

4.1 実験環境

計算環境としては理化学研究所情報基盤センターに設置された RSCC(RIKEN Super Combined Cluster) の一部を用いた．使用したのは Intel Xeon 3.06GHz を 2 基と 4GB のメモリを搭載した PC 128 ノードで構成される部分であり，計算ノード間は InfiniBand で接続されている．MPI ライブラリ及びコンパイラは，富士通社製のものを利用した．また，MPI プログラム実行では 1 ノードに 1 プロセスを割り当てた．本稿の実装では提案手法による最適化の適用頻度を 20 回に 1 回とする．

4.2 ベンチマークプログラム

2 つのベンチマークプログラムを用いて実験を行った.

- 実験 1: 擬似負荷プログラム
  - このプログラムは実ランクがプロセス数の半分であるプロセスに意図的に負荷を与えるプログラムである. 意図的に負荷を与えることにより, 提案手法の効果を最大に得ることができる状況を作ることができる. このプログラムについて負荷の大きさやメッセージサイズに応じたブロードキャスト通信の所要時間を計測する. プロセス数は 32, 64, 128, メッセージサイズは 640B, 2560B, 10KB, 40KB, 160KB, 640KB, 2560KB, 10240KB, 40960KB と変化させる. 負荷としては密行列積の計算を用い, 行列のサイズと行列積の回数によって負荷の重さを調節する.
- 実験 2: 疎行列積プログラム
  - このプログラムは圧縮された巨大な疎行列の行列積を求めるプログラムである. データとして用いる疎行列は CCS(Compressed Column Storage) 形式で 3 つの配列に格納されており, 大きく 3 つの処理の過程を持つ. まず, ランク 0 のプロセスはこの 3 つの配列を利用して圧縮された各行を 1 行ずつ元の大きさの行に戻す. 次に, 元に戻した行を各プロセスに配布する. この時の配布にブロードキャストを用いる. 他の各プロセスは割り当てられた列を同じく 3 つの配列から元の列に戻す. 最後に, 戻した列とランク 0 から配布される行との積を求め, その結果を再び圧縮する. 以降, ランク 0 のプロセスが各行を元の大きさに戻す作業とランク 0 以外のプロセスの積を求める作業は並列に行われる. 行列積の計算が終了後, ランク 0 が各プロセスの圧縮された結果を集め, 一つの疎行列のデータにまとめる. 本実験で用いた疎行列データとしては, Matrix Marke[11] に掲載されている BCSSTK32 を用いた.

4.3 実験結果 1

図 5 に負荷が 50×50 の行列積であるときの実験 1 の結果を示す. 実験環境において 50×50 の行列積の所要時間は 500 μ 秒であった. プロセス数は 128 である. 横軸がメッセージサイズで縦軸がブロードキャスト通信の所要時間である. 図のグラフはそれぞれ以下の値を示している.

- Original: 従来手法のブロードキャストの所要時間
- Optimize: 提案手法のブロードキャストの所要時間
- Optimize(ideal): Optimize からプロファイルコスト, 最適化コストを除外したブロードキャストの所要時間

Optimize(ideal) の値は以下の関数を用いて算出した.

- Bcast: 従来手法のブロードキャスト関数
- Bcast(noopt): 従来手法のブロードキャスト関数にプロファイル機能を付加したもの
- Bcast(opt): プロファイル機能, 最適化機能を持ったブロードキャスト関数
- Bcast(afteropt): 最適な仮想ランクで行うブロードキャスト関数にプロファイル機能を付加したもの

まず, プロファイルコストを P_cost とするとプロファイルコストは式 (1) で求められる.

$$P\_cost = Bcast(noopt) - Bcast \quad (1)$$

また, 最適化頻度を毎回に設定し, 最適化コストを O_cost とすると最適化コストは式 (2) で求められる.

$$O\_cost = Bcast(opt) - Bcast(afteropt) \quad (2)$$

以上より, Optimize(ideal) の値は式 (3) となる.

$$Optimize(ideal) = Optimize - (P\_cost + O\_cost) \quad (3)$$

また, 図 6 に従来手法と提案手法のブロードキャストによる最適化による性能向上率を示す. 最適化による性能向上率とは従来手法に比べてコストを考えない提案手法を適用することによりどれだけ所要時間が削減されたかの指標である. 最適化による性能向上率は式 (4) で計算した.

$$((Original - Optimize(ideal))/Original) \times 100 \quad (4)$$

次に, 提案手法のブロードキャストを行う際に必要となるプロファイルコストと最適化コストを図 7 に示す. 図 7 はメッセージサイズが 40KB の時のものである. 縦軸がそれぞれのコストに要する時間を示し, 横軸がプロセス数を示している.

実験結果ではプロファイルコスト, 最適化コストを含んだ場合に提案手法による効果がほとんど得られなかったが, これは最適化の頻度を 20 回に固定したことが原因だと思われる. 同じ負荷状態で多数のブロードキャスト通信を実行するプログラムでは, 最適化の頻度をより少なく調整することにより, 効果が得られる期待できる.

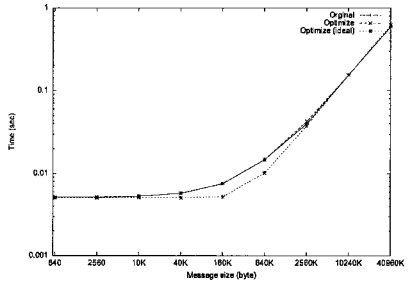

図 5: ブロードキャスト所要時間

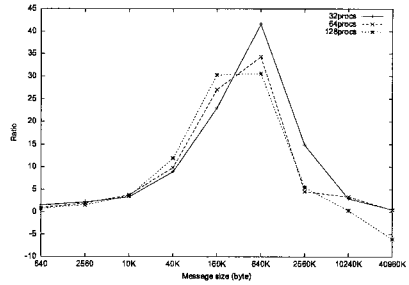

図 6: 提案手法の性能向上率

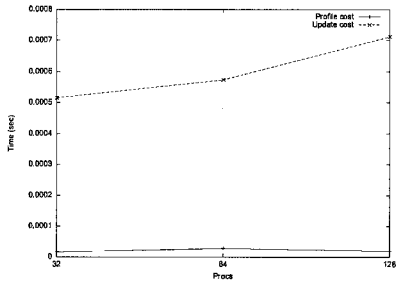

図 7: 提案手法におけるプロファイル，最適化コスト

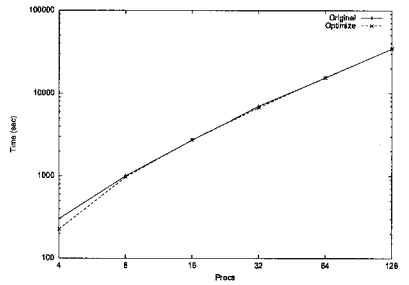

図 8: ブロードキャストの所要時間

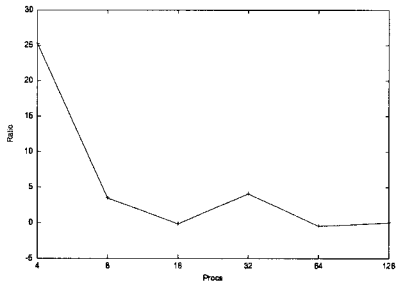

図 9: 提案手法によるブロードキャストの所要時間削減率

4.4 実験結果 2

図 8 にブロードキャスト通信の所要時間を示す．縦軸が時間を示し，横軸がプロセス数を示している．図の Original が従来手法のブロードキャストを用いたときの所要時間を示し，Optimize が提案手法のブロードキャストの所要時間を示している．また，図 9 は最適化による所要時間削減率を示す．縦軸が従来手法に対する提案手法のブロードキャストの所要時間削減率を示し，横軸がプロセス数を示している．

プロセス数が少ない場合では提案手法の効果を得ることができたが，プロセス数が多くなると提案手法の効果はほとんど得ることができなかった．この疎行列積プログラムでは負荷のばらつきは疎行列データの非ゼロ要素のばらつきに依存するため，疎行列データによって効果の傾向が変わると予想される．

5 考察

本章では PC クラスタによる実験を受けて，提案手法についての考察を述べる．

5.1 プロセス数，メッセージサイズ，負荷の量と最適化の効果の関係

どんな要素が最適化の効果に影響を及ぼしているかを考える．実験結果 1 の図 6，から，プロセス数，負荷の量が同じでもメッセージサイズが変化すると最適化の効率も変化するということが分かる．また，図 9 から，メッセージサイズ，負荷の量が同じでもプロセス数が変化すると最適化の効果も変化することが分かる．さらに，図 5，6 からプロセス数，メッセージサイズが同じでも負荷の量が変化すると最適化の効果も変化するということが分かる．したがって，プロセス数，メッセージサイズ，負荷の量といっ

たそれぞれの要素が最適化の効果に影響することが分かる．

5.2 最適化コストについて

実験2では最適化の効果をほとんど得ることができなかった．したがって，提案手法に要するコストが大きいと考えられる．そのコストの中でも最適化コストが大きい部分を占めていることが図7から分かる．プロファイルコストは提案手法のブロードキャストを実行する毎に発生するコストであるため削減することはできない．一方，最適化コストは提案手法のブロードキャストを実行する毎に発生するコストではなく，仮想ランク表の更新，つまり最適化を行う毎に発生するコストである．したがって，最適化頻度に依存するコストであることが分かる．このため，最適化頻度を減らすことにより最適化コストを削減することができる．

6 まとめと今後の課題

本稿では各プロセスの負荷状況を考慮するBinomial Treeアルゴリズムのブロードキャスト通信の方法を提案した．また，本手法によるブロードキャスト関数のプロトタイプを実装し，実環境上での効果を検証した．今後の課題としてベンチマークプログラムや実際のアプリケーションでの効果の調査を予定している．また，メッセージサイズが大きい場合，メッセージを分割してパイプライン的にブロードキャストを行うアルゴリズムの方が効果が良い場合がある．そのようなアルゴリズムへの適用も検討する．

謝辞 本研究の実験結果は，理化学研究所のRIKEN Super Combined Cluster Systemを使用したものである．また，本研究は「ペタスケール・システムインターコネクト技術の開発」プロジェクト(文部科学省「次世代IT基盤構築のための研究開発」の研究開発領域「将来のスーパーコンピューティングのための要素技術の研究開発」(平成17〜19年度)の一つ)[1]によるものである．

参考文献

[1] http://www.psi-project.jp/, PSI プロジェクト

[2] http://www-unix.mcs.anl.gov/mpi/mpich/, MPICH

[3] R.Thakur and W.D.Gropp, "Improving the Perfomance of Collective Operations in MPICH", 10th European PVM/MPI User's Group Meeting, pp.257-267,2003.

[4] R.Rabenseifner, "Optimization of Collective Reduction Operations", LNCS 3036/2004, Proceedings of Computational Science - ICCS 2004: 4th International Conference, pp.1-9, 2004.

[5] V.Tipparaju, J.Nieplocha, "Optimizing All-to-All Collective Communication by Exploiting Concurrency in Modern Networks", SC 2005, 46.

[6] A.Faraj, X.Yuan, "Automatic generation and tuning of MPI collective communication routines", ICS 2005, pp.393-402.

[7] http://www.abc-lib.org/, ABC Lib

[8] Sathish S.Vadhiyar, Graham E.Fagg, Jack Dongarra, "Automatically Tuned Collective Communications", 2000 ACM/IEEE conference on Supercomputing, article No.3.

[9] A.Faraj, X.Yuan, and D.Lowenthal, "STAR-MPI: Self Tuned Adaptive Routines for MPI Collective Operations.", ICS 2006, pp.199-208.

[10] Peter S. Pacheco, Parallel Programming with MPI, Morgan Kaufmann Publishers,, 1997.

[11] http://math.nist.gov/MatrixMarket/, MatrixMarket