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

2008－HPC－118（13）
2008／12／17

性能モデルによる予測を併用した Alltoall アルゴリズム動的選択技術の評価

南里 豪志[†1] Hyacinthe Nzigou Mamadou[†2] Feng Long Gu[†3] 村上 和彰[†4]

[†1] 九州大学情報基盤研究開発センター, [†2] 九州大学大学院システム情報科学府,
[†3] 九州大学大学院総合理工学研究院, [†4] 九州大学大学院システム情報科学研究院

概要

プロセス並列処理における集団通信の性能は実装アルゴリズムに依存する．本研究では，性能モデルを用いてアルゴリズムの選択肢を絞り込むことにより，実行中に効率良く複数のアルゴリズムから最速なものを選択する技術を提案する．この技術を 4 種の並列計算機に実装して評価した結果，本手法によってほとんどの場合で最速アルゴリズムが選択されることを示した．また，メッセージサイズが大きい場合，性能モデルを用いてアルゴリズムを絞り込むことによって選択に要するコストを低減できることを示した．

Evaluation of Dynamic Algorithm Selection with Performance Prediction Models on Alltoall Operation

Takeshi Nanri[†1] Hyacinthe Nzigou Mamadou[†2] Feng Long Gu[†3] Kazuaki Murakami[†4]

[†1] Research Institute for Information Technology, Kyushu University,
[†2] Graduate School of Information Science and Electrical Engineering, Kyushu University,
[†3] Interdisciplinary Graduate School of Engineering Sciences, Kyushu University,
[†4] Faculty of Information Science and Electrical Engineering, Kyushu University

Abstract

Performances of collective communications depends on the algorithms that construct them. This research introduces a technique that efficiently selects the best algorithm at runtime efficiently by using performance model of each algorithm to reduce the number of candidates. The results of experiments on four parallel computers showed that the technique could select the fastest algorithm in most cases. Those results also revealed that in relatively large message sizes, the reduction of candidates with performance models could decrease the cost for selecting algorithm.

1 はじめに

MPI 等のプロセス並列による並列処理モデルは，物理的に分散した並列処理環境に対する親和性の高さから，特に大規模並列計算において広く用いられている．このモデルによる並列処理性能に最も大きく影響を与える要因の一つが，プロセス間でデータを共有するために行う通信時間である．特に複数のプロセスによるグループ内でデータを交換する集団通信は，一般にプロセス数に応じて所要時間が増大するため，計算機の高並列化に伴って，より効率の良い実装が求められている．通常，集団通信はグループ内の一対一通信を組み合わせることによって実装されている．この組み合わせ方，すなわち集団通信の実装アルゴリズムは，処理性能に大きく影響するため，様々な手法が提案されてきた．しかし，転送するデータ量や参加するプロセス数によってアルゴリズムの優劣が変化するため，適応的なアルゴリズム選択を行う必要がある．

従来，アルゴリズム選択の手法としては，静的なものが用いられてきた．これは，計算機の初期設定時等，実際にアプリケーションを動作させる前に，計算機の基本性能や各アルゴリズムでの性能を計

測することによって, アルゴリズム選択の方針を決定するものである. このアルゴリズム選択の方針として広く用いられている手法に, プロセス数や転送データ量について閾値を定め, その閾値に応じて使用するアルゴリズムを変える, というものがある. しかしながら, 特に計算機が大規模になるにつれて, アプリケーション実行時の状況による処理性能への影響が大きくなっており, 静的な選択では不十分となってきている. 実行時の状況によって影響を受ける原因としては, アプリケーションの負荷バランス, プロセスの配置による通信性能の変化, 同時に実行される他のジョブによる資源の競合等が挙げられる.
そこで我々は, 実行時に集団通信のアルゴリズムを選択する技術について研究を行っている. 同様の技術として, Faraj らが提案した StarMPI [1] がある. これは, 実行時に各アルゴリズムの性能を計測し, 最適なものを選択する技術である. 本研究では, この StarMPI において問題となっている各アルゴリズムの性能計測コストについて, 事前に性能予測モデルを用いたアルゴリズムの絞り込みを行うことによる低減を図る. 本稿では, この性能予測モデルによる絞り込みを併用した最適アルゴリズム動的選択技術を, 科学技術計算で多用される Alltoall 通信に対して実装し, 効果について計測, 評価を行う.

2 関連研究

前述の通り集団通信は並列処理の性能に大きく影響するため, 様々なアルゴリズムが提案されている [2, 3]. これらの優劣は, プロセス数や転送データ量によって変化するため, どれか一つのアルゴリズムだけを利用するのは性能上問題がある. そこで通常は, 集団通信のアルゴリズムを複数用意し, プロセス数や転送データ量に応じて適切なものを選択する, という実装手段が取られている [4, 5]. しかし, 集団通信の性能は負荷バランスやネットワークの混雑等, 実行時の状況にも影響を受ける. 特に計算機が大規模になると, その影響も大きくなる. これに対して Faraj らは, このような実行時の状況に応じて適応的に集団通信アルゴリズムを選択する技術を提案した [1]. これは, 繰り返し呼び出される集団通信について, 最初の方の繰り返しを使って各アルゴリズムの性能を測定し, その結果に応じて以降の呼び出しで使用するアルゴリズムを選択するものである. この技術では, 最適なアルゴリズムに対して数倍〜数十倍の時間を要するアルゴリズムも選択肢に入れているため, 性能測定時のコストが問題である.
本研究は, 動的アルゴリズム選択において, 選択肢となるアルゴリズムを性能予測モデルを用いて絞り込むことにより, アルゴリズム選択技術の効率向上を図る.

3 Alltoall アルゴリズムの性能予測

3.1 一対一通信性能モデル P-LogP

基本的に集団通信は一対一通信を組み合わせることによって実装される. そこで本研究では, 集団通信アルゴリズムの性能予測手段として, 一対一通信の性能モデル P-LogP [6] を基にして集団通信の性能予測モデルを作成する.
P-LogP は, プロセッサ数 (P), プロセス間通信遅延 (L), 送信オーバヘッド (Os(m)), 受信オーバヘッド (Or(m)), 及び次のメッセージが送信可能となるまでの時間 (g(m)) の 5 個のパラメータによってプロセス間通信の所要時間をモデル化するものである. このモデルは, メッセージサイズ m に応じて Os, Or, g を設定することにより, 精度の高い性能予測を行うことが出来る.

3.2 P-LogP を用いた Alltoall アルゴリズムの性能予測

Alltoall は, 各プロセスが送信データをプロセス数と同じ数のブロックに分け, 全プロセスにプロセス番号に応じたブロックを送信する全体通信である. この集団通信は特に通信量が多い処理であるため, 様々なアルゴリズムが提案されている [5]. 著者らはそのうち Simple Spread, Bruck, Recursive Doubling, Pair, Pair with Light Barrier, Pair with MPI Barrier, Ring について, P-LogP モデルによる性能予測モデルを作成した [7]. 表 1 に各アルゴリズムの性能予測モデルを示す.

表 1: Alltoall アルゴリズムの性能予測モデル

Algorithm	Performance Model
Simple Spread	$(L \times g(m)) \times (P-1)$
Bruck	$(L+2g(m)(\log_2 P+1)) \times \log_2 P$
Pair	$(L + g(m) + min(O_s(m), O_r(m)) \times (P-1))$
Pair with Light Barrier	$(L + g(m) + min(O_s(m), O_r(m)) \times (P-1)) + max(O_s 0, O_r(0)) \times (P-2)$
Pair with Barrier	$(L + g(m) + min(O_s(m), O_r(m)) \times (P-1)) + (L+2g(0)) \times \log_2 P$
Recursive Doubling	$(L+2g(2m) \times \log_2 P) \times \log_2 P$
Ring	$(L+2g(m)) \times (P/2-1))$

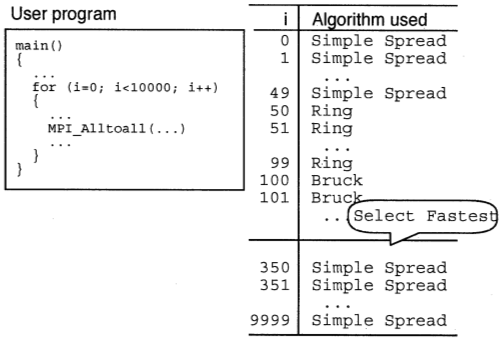

図 1: 集団通信アルゴリズム動的選択技術

4 性能予測モデルを併用した集団通信アルゴリズム動的選択技術

本節では, 本研究で開発した集団通信アルゴリズム動的選択技術について説明する.

この技術は, Star-MPI で提案された技術を基にしており, 実行中に同じ集団通信を繰り返し呼び出すプログラムにおいて, 最初の数十〜数百回分の呼び出し時に実際に一つずつアルゴリズムを試すことによって, 最適なアルゴリズムを選出する.

図 4 に, この技術を利用した Alltoall 関数を呼び出すプログラムの実行例を示す. この例では, ユーザプログラムで呼び出される 10000 回の Alltoall のうち, 最初の 50 回については Simple Spread アルゴリズムを用い, 次の 50 回については Ring アルゴリズム用いる, というように, 実装されているアルゴリズムを規定回数ずつ試し, 所要時間を記録する. この, アルゴリズムを試行する回数はプログラム実行前に指定する. 全アルゴリズムについて計測が終了すると, 計測結果から最速のものを選出する. それ以降は, この選出されたアルゴリズムを用いて Alltoall 通信を行う.

この手法では, 最終的に最適なアルゴリズムを選択することが可能だが, 計測対象のアルゴリズムの中には, 最適なアルゴリズムの数倍〜数十倍の時間を要するものが含まれている場合があり, このコストによる性能低下が問題となる. これに対して本研究で開発したアルゴリズム動的選択技術は, 計測対象のアルゴリズムを性能予測モデルを用いて絞り込むことにより, アルゴリズム選択時のコスト低減を図る.

この技術は, Grouping, Learning, Running の 3 フェーズで構成される. 各フェーズの動作は以下の通りである.

Grouping Alltoall が呼び出された際, そのメッセージサイズ m による最初の呼び出しである場合にこのフェーズが実行される. このフェーズでは 3.2 節で紹介した性能モデルを用いて各アルゴリズムの性能を予測し, その中から次の Learning フェーズにおける性能計測の対象とするアルゴリズムの選択肢を絞り込む.

フェーズ内の具体的な処理としては, まず, 呼び出された Alltoall のメッセージサイズ m に対して, 性能モデルで用いる P-LogP のパラメータ L, g(0), g(m), Os(0), Os(m), Or(0), Or(m) を計測する. 計測には文献 [6] で紹介されている計測ツールを用いる. その後, これらのパラメータを使って得られた各アルゴリズムの性能予測結果から閾値を設定し, 予想処理時間が閾値を越えないアルゴリズムを, Learning フェーズに性能計測対象グループとして抽出する. 閾値の決定の方法として, 本稿の実験では予想処理時間が最長のものと最短のものの平均を用いている.

Learning Grouping フェーズで絞り込まれたアルゴリズム群について, 図 4 に示すような手法で所要時間を計測し, 記録する. 全アルゴリズムについて規定回数の計測が終了するま

で, 各 Alltoall 呼び出しに対してこのフェーズを実行する.

計測がすべて終了すると, 最も高速なアルゴリズムを次の Running フェーズで使用するアルゴリズムとして選出する. アルゴリズムの選出に用いる各アルゴリズムの所要時間としては, 全プロセスの所要時間の平均値を用いる.

Running 選出されたアルゴリズムを用いて Alltoall 通信を行う. このフェーズは Learning フェーズが終了してアルゴリズムが選択された後, 各 Alltoall 呼び出しに対して実行する.

実際の計算環境では実行中に状況が大きく変化することも考えられるため, 定期的に Alltoall の所要時間と Learning フェーズで得られた所要時間を比較する. もし, 計測した時間が Learning フェーズ時の所要時間から大きく変動した場合, 他のアルゴリズムの方が高速なった可能性があるので, 再度 Learning フェーズに入り, アルゴリズムを選択する.

なお, この 2 回目以降の Learning フェーズでは, 前の Learning フェーズの結果に応じて計測対象アルゴリズムをさらに絞り込む. 例えば本稿の実験では, 所要時間が最速のアルゴリズムより 20%以上遅いアルゴリズムを次の Learning フェーズの計測対象アルゴリズム群から外している.

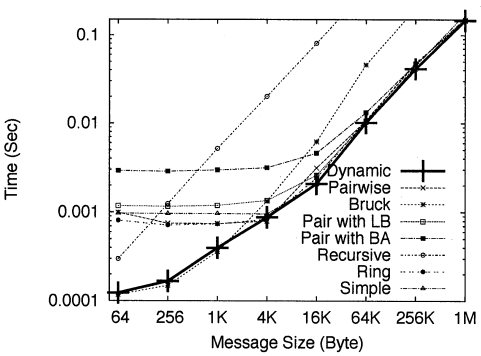

図 2: PRIMEGY(4core × 8nodes) の所要時間

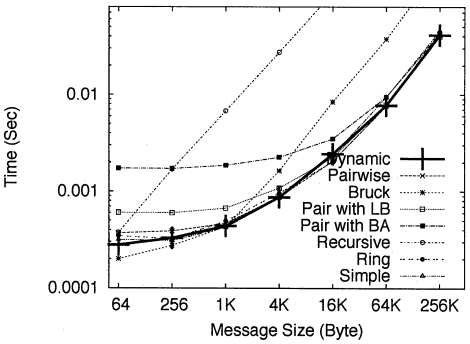

図 3: PRIMEQUEST(32core) の所要時間

5 実験

5.1 実験環境

本研究で開発したアルゴリズム動的選択技術について, 九州大学情報基盤研究開発センターの 3 式の並列計算機 (PRIMERGY, PRIMEQUEST, SR11000) と, 理化学研究所の RSCC(Riken Super Combined Cluster) 上に実装し, 評価を行った. PRIMERGY は 4 コアの Intel Xeon 3.0GHz と 8GB の RAM を搭載したノードで構成され, OS は RHEL WS 4, コンパイラと MPI ライブラリは富士通製で, ノード間ネットワークは InfiniBand である. PRIMEQUEST は 64 コアの Intel Itanium2 1.6GHz と 128GB の RAM を搭載したノードで構成され, OS は RHEL AS 4, コンパイラと MPI ライブラリは富士通製である. SR11000 は 16 コアの IBM Power5 1.9GHz と 128GB の RAM を搭載したノードで構成され, OS は IBM AIX 5L, コンパイラと MPI ライブラリは IBM 製で, ノード間ネットワークは専用のクロスバスイッチである. RSCC は 2 コアの Intel Xeon 3.06GHz と 4GB の RAM を搭載したノードで構成され, OS は RedHat 8, コンパイラと MPI ライブラリは富士通製で, ノード間ネットワークは InfiniBand である. なお, RSCC ではノード内の 2 コアのうち 1 コアのみを使用して実験した.

5.2 動的アルゴリズム選択の効果

各計算機上で, 本研究で開発したアルゴリズム動的選択技術を適用した Alltoall 関数の所要時間を, メ

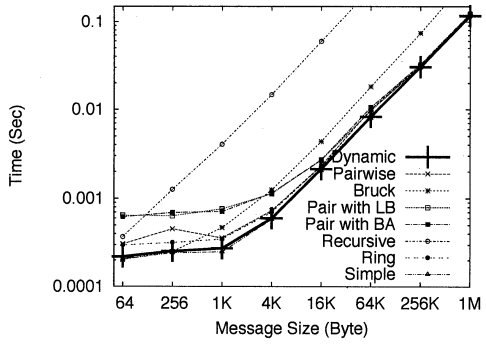

図 4: SR11000(16core × 2node) の所要時間

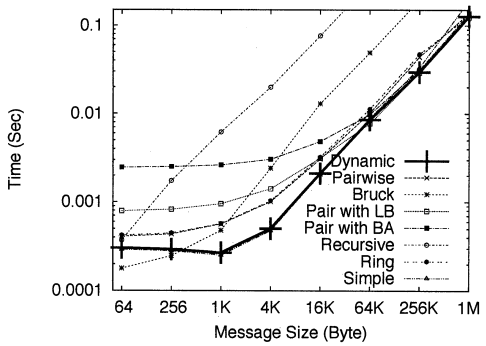

図 5: RSCC(1process × 32nodes) の所要時間

ッセージサイズを変えて計測した結果を図 2,3,4,5 に示す．各計算機とも 32 コアを使用した．計測は Alltoall 関数を 20000 回呼び出し，1 回当たりの所要時間について全プロセスの平均を取った．また，比較のため，各アルゴリズムを直接呼び出した場合の所要時間についても同様に計測した．図の横軸がメッセージサイズ，縦軸が所要時間 (秒) である．また，Dynamic は本研究の動的選択技術を適用した Alltoall による所要時間，Pairwise, Bruck, Pair with LB(Light Barrier), Pair with BA(Barrier), Recursive(Recursive Doubling), Ring, Simple(Simple Spread) は，それぞれ各アルゴリズムを直接呼び出した時の所要時間である．

これらの図より，メッセージサイズの相違によってアルゴリズム間の優劣が変化していることが分かる．例えば図 2 でメッセージサイズが 1KB 以下では Bruck が最速だが，4KB 以上になると Ring や Simple Spread の方が高速になる．

また，PRIMERGY と SR11000 は CPU コアを複数搭載したノードをネットワーク接続した階層的な構造での通信であるのに対し，PRIMEQUEST はノード内通信のみ，RSCC はノード間通信のみである等，計算機構成が違うため，計算機によってアルゴリズムの優劣が入れ替わるメッセージサイズが異なる．

このようにアルゴリズム間の優劣が複雑に変化する中で，Dynamic はほとんどの場合において各アルゴリズムの中の最も高速なものに近い性能を出していることが分かる．これは，本研究のアルゴリズム動的選択技術によってほぼ正確に最適なアルゴリズムを選択できていることを示している．なお，図 3 や図 5 で，メッセージサイズが 64 バイトの時に Dynamic の所要時間と最速のアルゴリズムの所要時間の差が大きい．これは，最適化に要するコストによるものであると考えられる．

5.3 性能予測モデルによるアルゴリズム絞り込みの効果

次に，本研究のアルゴリズム動的選択技術において Grouping フェーズを介さずに全てのアルゴリズムを Learning フェーズの対象とした場合の所要時間を計測し，提案手法による所要時間と比較する．これにより，性能予測モデルを用いてアルゴリズムの選択肢を絞り込むことによる効果を評価する．

実験結果を表 2 に示す．これは各計算機について Grouping フェーズによるアルゴリズムの絞り込みを行った場合 (model) と，行わなかった場合 (nomodel) の所要時間を計測した結果である．使用 CPU コア数は，いずれも 32 コアである．

表より，メッセージサイズが 4KB 以上の場合，アルゴリズムの絞り込みを行うことによる性能向上が得られることが分かる．しかしメッセージサイズを 256 バイト以下にすると，アルゴリズム絞り込みによって性能が低下する．これは，アルゴリズム絞り込みによるコストと効果の関係によるものである．

表 2: 性能モデルによるアルゴリズム選別の効果

Size(Byte)	PRIMEQUEST		PRIMERGY		SR11000		RSCC	
	model	nomodel	model	nomodel	model	nomodel	model	nomodel
64	0.279	0.276	0.123	0.120	0.220	0.225	0.304	0.183
256	0.328	0.317	0.166	0.201	0.253	0.253	0.293	0.275
1024	0.438	0.461	0.397	0.420	0.273	0.273	0.269	0.275
4096	0.871	0.939	0.877	1.026	0.598	0.653	0.500	0.548
16384	2.437	2.412	2.079	2.318	2.143	2.515	2.112	2.328
262144	7.727	8.893	10.17	10.89	8.300	8.843	8.522	8.562
1048576	41.09	45.94	41.33	44.63	30.44	32.55	29.56	33.51

(msec)

アルゴリズム絞り込みは, Learning フェーズにおける遅いアルゴリズムの性能計測を回避することによって最適化コストの削減を図るが, 性能モデルを用いて予測を行うため, P-LogP のパラメータを計測するコストが必要である. 前節の図でも明らかなように, アルゴリズム間の性能差はメッセージサイズが大きくなるにつれて増加する. そのため, メッセージサイズが十分に大きく, アルゴリズム絞り込みの効果が P-LogP パラメータの計測コストを上回る場合, 全体として性能が向上する.

6 むすび

集団通信の性能向上を目的として, 実行時の計測結果を用いた動的アルゴリズム選択技術を提案した. 実験の結果, ほとんどの場合で最速のアルゴリズムが選択できることを示した. また, アルゴリズムの選択肢を絞り込むことによる効果について検証した. 今後は, 使用ノード数による影響や, 動的な計算機環境の変化による影響について検証する予定である.

参考文献

[1] A. Faraj, X. Yuan, and D. Lowenthal, "STAR-MPI: Self Tuned Adaptive Routines for MPI Collective Operations," In Proceedings of the 20th ACM International Conference on Supercomputing (ICS06), 2006.

[2] P. Mitra, D. Payne, L. Shuler, R. van de Geijn, and J. Watts, "Fast Collective Communication Libraries," In Proceeding of the Intel Supercomputing User's Group Meeting, 1995.

[3] R. Rabenseifner, "Optimization of Collective Reduction Operations," In Proceedings of ICCS, LNCS 3036, pp. 1-9, 2004.

[4] V. Bala, J. Bruck, R. Cypher, P. Elustondo, A. Ho, C.- T. Ho, S. Kipnis and M. Snir, "A portable and Tunable Collective Communication Library for Scalable Computers," IEEE Transaction on Parallel and Distributed Computing , pp. 154-164, 1995.

[5] Ahmad Fajad , Xin Yuan, "Automatic Generation and Tuning of MPI Collective Communication Routines," The 19th annual ACM International Conference on Supercomputing, 2005.

[6] T. Kielmann, H. Bal, and K. Verstoep, "Fast measurement of LogP parameters for message passing platforms," In IPDPS Workshops, volume 1800 of Lecture Notes in Computer Science, pp. 1176-1183, 2000.

[7] H. Nzigou Mamadou, G. de M. Baptista Domingues, T. Nanri and K. Murakami. "Performance Analysis and Linear Optimization Modeling of All-to-all Collective Communication Algorithms," In Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing, pp. 203-210, 2007.