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

2006－ARC－167 (33)
2006－HPC－105 (33)
2006／3／1

光ネットワーク環境におけるMPI集団通信

滝澤 真一朗[†] 松岡 聡[†,††] 中田 秀基[†††,†]

光バーストスイッチングネットワークでは、通信を行う前に光パスコネクションを確立し、通信終了後にはコネクションを解放しなければならない。コネクション確立・解放コストは合計して平均10msほど要する。そのため、MPIアプリケーション等、コンピュートインテンシブなアプリケーションで集団通信を実行する際にはコネクション確立・解放のオーバーヘッドが顕著になる。そこで、MPI集団通信において通信発生とは独立してコネクション確立・解放を行うことにより、オーバーヘッドを削減する手法を提案する。本手法ではノードの持つポート数に応じてコネクション確立・解放方法を変え、同時コネクション確立を行い高速実行を実現する。本手法と単純に通信のたびにコネクションを張る方法とで、数値解析によるアルゴリズム単体の比較、およびシミュレータを用いて仮想的に実アプリケーションを実行した場合の比較を行い、本手法の優位性を示した。

MPI Collective Communication on All Optical Network

Shin'ichiro Takizawa,[†] Satoshi Matsuoka[†,††]
and Hidemoto Nakada[†††,†]

On the Optical Burst Switching network, it is necessary to establish an optical path connection between nodes before communication and release it after communication. This process takes about 10 ms on average. For this reason, in compute-intensive applications, like MPI applications, an execution of collective communication is heavily influenced by the cost of establishing and releasing a connection. We propose a method which establishes and releases connections independent of communication occurrence to reduce cost in collective communication. In this method, we accomplish fast execution by changing algorithms and simultaneously connecting based on ports on node. The evaluation result shows our proposed technique performs superior to the method which establishes connections whenever communication occurs.

1. はじめに

大容量・超広帯域通信を可能にする光ネットワーク技術がグリッドやクラスタなどのハイパフォーマンスコンピューティング分野で利用され始めている。例えば、OptIPuterプロジェクト[1]では光ネットワーク環境での、分散アプリケーション開発・実行環境の構築、効率の良い集団通信を実現するプロトコル、ストレージシステムなどの研究開発を行っており、火星探索データの可視化等に利用されている[2]。光ネットワークは大容量データの高速転送を実現するので、データインテンシブアプリケーション実行環境として注目されている。

現在利用可能な光ネットワーク技術である、波長パススイッチ、光バーストスイッチ[3]はどちらも回線交換方式のため、通信時にはあらかじめ該当ノード間で通信用コネクションを確立しておく必要がある。より詳細には、通信前にはノード間でコネクションを確立し、通信終了後にはそのコネクションを解放しなければならず、確立・解放に合わせて最大20ms、平均10msほど要する[4]。I/Oにボトルネックが存在するデータインテンシブアプリケーションでは、このコストは通信時間に対して相対的に小さくなり、無視できるものかもしれないが、コンピュートインテンシブアプリケーションの場合は問題になる。例えば、NAS Parallel Benchmarkの各種プログラムでは、集団通信の場合も含め、MPI関数の実行コストがマイクロ秒のオーダーになることが多く、単純に通信のたびにコネクションの確立・解放を行うのであれば、最低でも10倍実行時間が増加する。複数ノードとの複数回同時通信を行う集団通信では特に問題になる。

本研究では、光ネットワーク上でMPIアプリケーションを実行する際に、集団通信において、関数内部で通信とは独立してコネクション管理を行うことでコネクション確立・解放回数を減らし、実行時間増加を

† 東京工業大学
Tokyo Institute of Technology
†† 国立情報学研究所
National Institute of Informatics
††† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

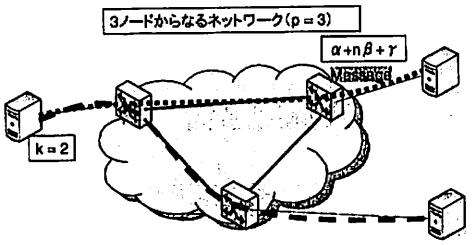

図1 ネットワークモデル図

抑える手法を提案する。その後、数値解析による評価、シミュレータを用いた実アプリケーション実行の評価を通して、提案手法と単純に通信のたびにコネクション確立を行う方法とで比較を行い、提案手法の優位性を示す。

2. 光ネットワークモデル

MPI アプリケーションを実行するネットワーク環境として光バーストスイッチングネットワークを想定する。光バーストスイッチングネットワークは回線交換方式であるため、通信前にコネクションを確立し、通信終了後には解放する必要がある。今回、光ネットワーク環境を以下に示す通りにモデル化する。

光ネットワーク網 p 個のノードが光ネットワーク網を介して互いに通信可能である。ネットワーク内の光スイッチの配置・配線は通信に影響を与えないと仮定し、考慮しない。任意の 2 ノード間のコネクションは並列に確立可能である。コネクションが確立された 2 ノード間では全二重通信が行われる。2 ノード間の通信遅延はどの 2 ノードを取っても等しいものとし、α とする。また、単位バイトあたりの転送時間を β とし、コネクション確立・解放にかかる時間を合計して γ とする。

ノード 各ノードは k 個のネットワークインターフェースを持ち、同時に最大 k ノードと通信できる。この k をポート数と定義する。通信を行う前に、送信ノードは受信ノードに対してコネクションを張り、終了後に解放する。

このモデルを図に示したものが図 1 である。

このネットワーク環境での一対一通信のコストを、「**コネクション確立時間 + 通信時間 + コネクション解放時間**」と定義する。ここで通信時間は転送データ量を n とすると $\alpha + n\beta$ となるので、通信コストは $\alpha + n\beta + \gamma$ となる。

3. MPI 集団通信の設計

3.1 設計方針

MPI アプリケーションを前章で定義した光ネットワーク上で実行する場合を考える。このとき、単純に通信発生ごとにコネクション確立・解放を行うのであれば、複数回一対一通信を実行する集団通信関数で大幅な性能低下が発生することが予想できる。そこで、MPICH 等既存の MPI 実装で用いられている集団通信アルゴリズムを拡張し、通信とは独立してコネクション確立・解放を行うアルゴリズムを設計する。コネクション確立・解放は、(1) 通信発生前に必要なコネクションを、可能ならば複数同時に確立する手法、(2) 1 度確立したコネクションを、相手との通信が全て完了するまで利用し続ける手法、を組み合わせて行う。また、アルゴリズムによってはポート数に応じてコネクション確立方法を変更する。ポート数により同時コネクション数、通信数が変化するためである。

アルゴリズム設計に際して、以下の仮定を置く。

- 1 ノードあたり、1MPI プロセスを実行する
- ノード数は 2 のべき乗個である
- 計算に参加する全てのノードにおいてポート数は等しい
- 複数のポートを持つ場合、複数ノードに対するコネクション確立・解放は並列に実行可能であり、その場合の実行時間は 1 つのコネクションを確立・解放した時間と等しい

以降では、MPI 集団通信アルゴリズムとして用いられている、Linear、Ring、Recursive Doubling、Binomial Tree の各アルゴリズムに、コネクション確立・解放機構を組み込んだアルゴリズムを示し、その実行コストを示す。また、通信のたびにコネクション確立・解放を行う場合 (Naive な手法) の実行コストも示す。MPI 集団通信ではこれら 4 種類以外にもさまざまなアルゴリズムが利用されているが[5]、これらアルゴリズムのステップ実行順を逆にしたもの、複数組み合わせたもの、通信相手が異なるだけのものがほとんどであるため、この 4 種類のみに着目した。

3.2 Linear

Linear は MPI_Bcast や MPI_Scatter などの一対全通信で利用されるアルゴリズムである。これは Root プロセスが他のプロセスに対して逐次に送受信を行うアルゴリズムである。よって、Naive な手法では、1 回の通信コストが $\alpha + n\beta + \gamma$ なので、p プロセス全てにメッセージが行き渡るまでの時間は以下になる。

$$(p-1)(\alpha + n\beta + \gamma) \quad (1)$$

提案手法では、ポートが複数ある場合にはポート数分ずつコネクションを確立し、転送、解放を行う、ことを繰り返す。図 2 にポート数が 3 の場合のアルゴリズムの動作を示す。ステップ数は $\lceil \frac{p-1}{k} \rceil$ になり、実行コストは

$$(p-1)(\alpha + n\beta) + \lceil \frac{p-1}{k} \rceil \gamma \quad (2)$$

となる。$\lceil x \rceil$ は x より大きい最小の整数を意味する。

3.3 Ring

Ring は MPI_Allgather などの全体全通信で利用さ

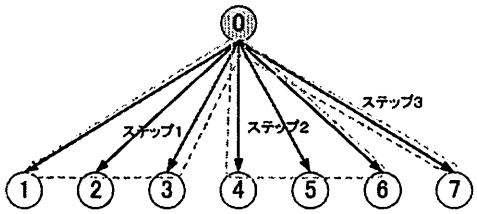

図 2 ポート数が 3 の場合の Linear アルゴリズム

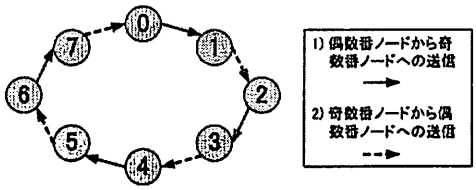

図 3 ポート数が 1 の場合の Ring アルゴリズム

れるアルゴリズムである。これは各プロセスはランク ID の連続するプロセスと接続し、仮想的なリングを構築してメッセージをリレーするアルゴリズムである。このアルゴリズムでは各プロセスは 1 ステップあたり 2 つのコネクションを必要とする。このため、Naive な手法、およびポート数が 1 の場合はステップ内でコネクションの張替えを行う必要がある。そこで、この場合はまず偶数ランクプロセスがメッセージを送信し、その後奇数ランクプロセスがメッセージを送信するという手法を取る（図 3）。ステップあたり 2 回通信が必要なため、各プロセスがそれぞれサイズ $\frac{n}{p}$ のメッセージを持っている場合の MPI_Allgather の実行コストは以下になる。

$$2(p-1)\left(\alpha+\frac{n}{p}\beta+\gamma\right) \tag{3}$$

提案手法では、ポートが複数ある場合には通信開始前に隣接プロセスとコネクションを張り、全ての通信の終了後にコネクションを解放する方法を取る。この場合は、通信にはコネクション確立・解放時間は影響せず、実行コストは

$$(p-1)\left(\alpha+\frac{n}{p}\beta\right)+\gamma \tag{4}$$

となる。

3.4 Recursive Doubling

このアルゴリズムは MPI_Barrier などの全体全通信で利用されている。これはアルゴリズムの第 i ステップにおいて、各プロセスはそれぞれ 2^{i-1} 離れたプロセスとメッセージ交換するアルゴリズムである（図 4）。各プロセスがそれぞれサイズ $\frac{n}{p}$ のメッセージを持つときの、Naive な手法による MPI_Allgather は、アルゴリズムのステップ数は $\log p$ だが、各プロセスは $(p-1)$ 個のメッセージを受信するため、実行コストは以下になる。

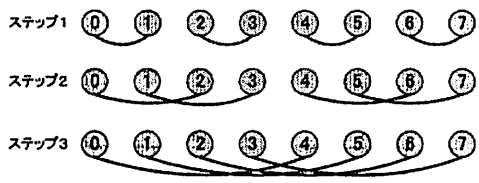

図 4 Recursive Doubling アルゴリズム

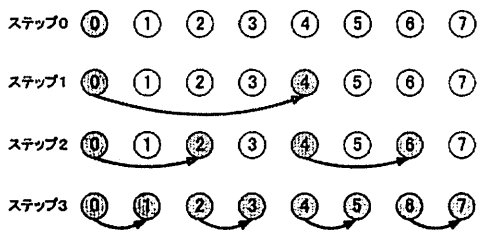

図 5 Binomial Tree アルゴリズム

$$(\log p)(\alpha+\gamma)+(p-1)\frac{n}{p}\beta \tag{5}$$

このアルゴリズムではプロセスは毎ステップ異なるプロセスと通信を行う。そこで、ポート数分ずつ必要なコネクションを事前に確立し、通信を行い、確立した分のコネクションを解放する、ことの繰り返しを行う手法を提案する。これにより、コネクション確立・解放回数は $\lceil\frac{\log p}{k}\rceil$ となり、実行コストは

$$(\log p)\alpha+(p-1)\frac{n}{p}\beta+\lceil\frac{\log p}{k}\rceil\gamma \tag{6}$$

となる。

3.5 Binomial Tree

このアルゴリズムは MPI_Bcast などの一対全通信で利用されている。これは二項木（Binomial Tree）を用いて、あるステップでメッセージを持つプロセスは後続するステップにおいてメッセージ送信者になるアルゴリズムである（図 5）。ランク 0 のプロセスが持つサイズ n のメッセージを他のプロセスに MPI_Bcast する場合に、Naive な手法を用いた通信を行うと、実行コストは以下になる。

$$(\log p)(\alpha+n\beta+\gamma) \tag{7}$$

このアルゴリズムもプロセスは毎ステップ異なるプロセスと通信を行うので、Recursive Doubling と同様にコネクションの事前確立を行う手法を提案する。これにより、コネクション確立・解放回数は $\lceil\frac{\log p}{k}\rceil$ となり、実行コストは

$$(\log p)(\alpha+n\beta)+\lceil\frac{\log p}{k}\rceil\gamma \tag{8}$$

となる。

表 1　数値評価での変数設定

変数	値
p	32
k	2,3,4
n	1 ～ 1M
α	0.0001
β	8×10^{-9} (1000 Mbps 相当)
γ	0.01

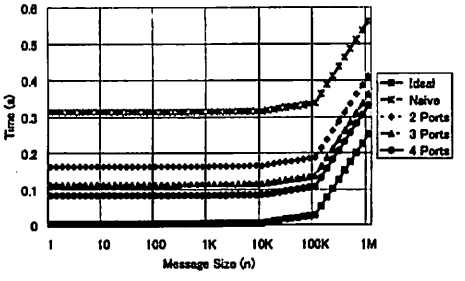

図 6　Linear の実行コスト

4. 評　　価

提案アルゴリズムを、実行コスト式を用いた数値による評価、およびシミュレーションによるアプリケーション実行時の性能評価を行った。

4.1 数値評価

各アルゴリズムについて、ポート数を変えた場合の提案手法（2 Ports、3 Ports、4Ports）の実行コストを、Naive な手法（Naive）による実行コスト、コネクション管理の必要ない場合（Ideal）の実行コストと、メッセージサイズを変数に比較した。各アルゴリズムの Ideal の場合のコスト式は、それぞれ式 2、4、6、8 で γ を 0 とした式である。アルゴリズムの数値評価を行うにあたり、各変数を表 1 に示す値に設定した。

Linear の実行コストを図 6 に示す。Naive な手法に対して、提案手法の方が最低でも 2 倍ほど高性能だと確認できる。どの手法においても、メッセージサイズが 10K を越えたあたりから、コネクション確立・解放遅延よりも通信時間の方が実行コストにしめる割合が大きくなり始めることがわかる。

Ring の実行コストを図 7 に示す。提案手法に対して、Naive な手法は遥かに性能が劣るが、これは提案手法ではコネクション確立・解放は 1 回のみだが、Naive な手法では 62 回実行されることによる。そのため、最低でも実行コストは 0.6 以上要する。

Recursive Doubling の実行コストを図 8 に示す。Recursive Doubling では 3 ポートの場合と 4 ポートの場合の実行時間が等しい。これは、コネクション確立の回数がそれぞれ 2 回になるためである（$\lceil \frac{5}{3} \rceil = \lceil \frac{5}{4} \rceil = 2$）。

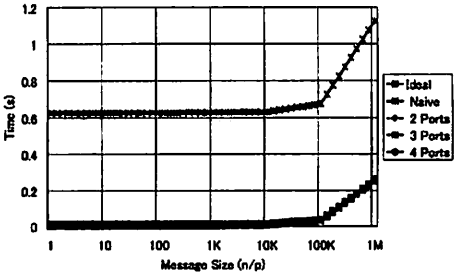

図 7　Ring の実行コスト

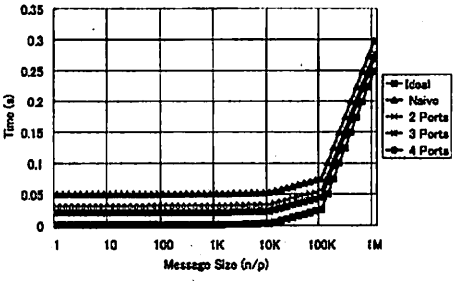

図 8　Recursive Doubling の実行コスト

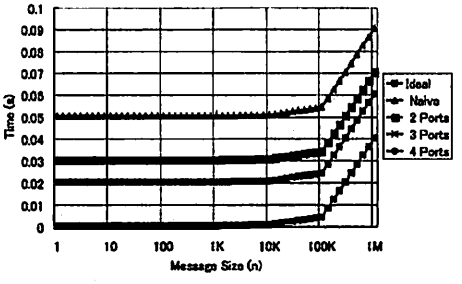

図 9　Binomial Tree の実行コスト

Binomial Tree の実行コストを図 9 に示す。Binomial Tree ではメッセージサイズが 100K を越えたあたりから、コネクション確立・解放遅延よりも通信時間の方が実行コストにしめる割合が大きくなり始める。この理由は、式 8 より通信時間:$\alpha + n\beta$ にかかる係数が $\log p$ であり、他のアルゴリズムに比べ小さいためである。

4.2 アプリケーション実行評価

シミュレータを作成し、提案アルゴリズムを用いた場合と Naive な手法の場合の、MPI アプリケーショ

表 2 NPB:FT の集団通信

集団通信関数	呼び出し回数	実装したアルゴリズム
MPI_Alltoall	6	Recursive Doubling
MPI_Barrier	1	Recursive Doubling
MPI_Bcast	2	Binomial Tree
MPI_Reduce	6	Binomial Tree

表 3 シミュレータの環境設定

項目	値
ネットワークバンド幅	1000Mbps
ポート数	2 ～ 4
通信遅延	100 μ s
コネクション確立・解放遅延	10ms

ン実行時間の比較を行った。

4.2.1 シミュレータの設計

作成したシミュレータは MPI アプリケーションを、CPU のみ使用する部分（CPU イベント）、MPI 関数呼び出し部分（MPI イベント）に分け、この繰り返しであるとして、先頭から順に実行し、総実行時間を求める。CPU イベントのコストは対象アプリケーションを実環境で実行したときの CPU 利用時間であり、MPI イベントのコストは関数内部で行われる通信の時間を計算して求める。通信時間のコスト式は 2 章で提案した式と等しく、既にコネクションが確立している場合には $\alpha + n\beta$ であり、確立していない場合は $\alpha + n\beta + \gamma$ である。

アプリケーションの実行イベント列は MPI プロファイリングライブラリを用いて、対象アプリケーションを実行することにより生成する。その際に、CPU イベントのコストも取得する。MPI イベントに関しては、MPI 関数が呼び出されるタイミングだけを記録する。宛先プロセス、データサイズ、さらに集団通信の場合にはルートプロセスのランク、グループのサイズ等、通信に必要な情報は独自に関数呼び出し時に取得するようにした。これらの情報を元に、通信パターンを模倣して実行時間を計算する。

4.2.2 NPB、FT での評価

提案アルゴリズムを実アプリケーションに適応した場合の評価を行うため、Nas Parallel Benchmarks (NPB) FT をシミュレータ上で実行した。問題サイズは A、プロセス数は 16 とした。FT は集団通信のみ行い、呼び出される集団通信の種類は表 2 に示すとおりである。表にはシミュレータ用に実装した MPI イベントに使用したアルゴリズムも載せてある。シミュレータの環境設定を表 3 に示す。

結果を表 4 に示す。実測値はギガビットイーサネット、通信遅延約 100 μ s の環境での結果である。Ideal はコネクション確立・解放遅延が存在しない場合での結果なので、実測値と等しい値をとることが期待されるが、この誤差は転送データサイズがそれほど大きく

表 4 NPB：FT シミュレーション結果

環境	実行時間
実測値	2.27s
Ideal	1.82s
Naive	2.67s
2 ports	2.33s
3 ports	2.33s
4 ports	2.16s

無いために、実環境では 1000Mbps を完全には活用できていないためだと考えられる。提案手法と Naive な手法の結果を見ると、数値解析の結果と同様に、Naive な手法が一番時間がかかり、提案手法ではポート数を増やすにつれ性能向上が見られる。

5. 関連研究

文献[6] では、光ネットワーク技術を利用した計算機環境として、各ノードが回線交換型光ネットワークインターフェースと、パケット交換型電気ネットワークインターフェースで互いに通信する環境を提案している。このシステムでは、一対一データ転送はまず電気ネットワークで開始される。トラヒックを監視しつつ、通信データ量が増加した場合には、光ネットワークを用いた通信に切り替える。しかし、光ネットワーク通信に切り替えようとしても、波長リソースが足りずに切り替えることができない場合があるため、古い使用されていない光パスを解放し、波長リソースに空きを作る機構も設置している。このシステムでは、一対一通信に対してはこのような賢い手法を採用しているが、集団通信は常にパケット交換型電気ネットワークで行う。本研究では集団通信を光ネットワーク上で行うことを考えており、この点で異なる。

文献[7),8] では、MPI アプリケーションの通信発生を予測し、通信開始前にあらかじめ必要なコネクションを確立するための、さまざまな予測アルゴリズムを提案し、各アルゴリズムの予測ヒット率を評価している。本研究では、MPI アプリケーションの実行ログより、通信間隔時間がマイクロ秒のオーダーになることが多いことを確認したので、通信予測はコネクション確立遅延を隠蔽することに適さないと判断した。

文献[9),10] では、回線交換型光ネットワーク環境での集団通信アルゴリズムを提案し、数値による解析を行っている。本件研究と同様に、ノードのポート数に着目したアルゴリズムの考案を行っているが、本研究ではアルゴリズムを実環境に実装することを考えており、既存の MPI ライブラリとの共存を考慮して設計を行っているが、文献で提案されている手法は複雑であり、実装には困難だと思われる。

6. おわりに

光バーストスイッチングネットワーク上でコンピュートインテンシブなMPIアプリケーションを実行する際には、光パスコネクション確立・解放による遅延のため、実行時間が大幅に増加する問題がある。この問題は特に、複数のノードと同時に通信を行う集団通信に影響する。そこで本研究では、既存のMPI集団通信アルゴリズムに対して、実行ノードが複数ポートを持つ場合に、通信とは独立してコネクション管理を行う機構を組み込み、実行時間増加を抑える手法を提案した。数値解析による評価、シミュレータを用いたNPB：FTの実行による評価を通して、提案手法は、通信のたびにコネクションの確立・解放を行う手法に対し、全体的に優れた性能を示し、更にポート数が多い場合に特に良い性能を示すことを確認した。

今後の課題として、以下を考えている。

- 現実の光バーストスイッチングネットワーク環境では、スイッチが提供する波長リソースに制限が設けられているため、多数のノードが同時に通信を行うとリソースを使い果たし、コネクションを確立できなくなることがある（輻輳発生）。集団通信では同時に多くの通信が発生するので、輻輳に対してロバストなアルゴリズムを考案する。
- 実際のネットワーク環境では、他のアプリケーションの影響により、ポート数、ネットワークの空き波長リソースは時間と共に変化する。ポート数、コネクション数の変化に対応した集団通信アルゴリズムを考案する。
- 近々、光バーストスイッチングネットワークのテストベッドが利用可能になる。提案手法をMPICH等の既存のMPI実装に移植し、実環境での評価を行う。

謝辞 本研究を進めるにあたり、光バーストスイッチングネットワークについての助言を下さった、NTT未来ネット研究所フォトニックトランスポートネットワーク研究部の方々に感謝いたします。

参考文献

1) Smarr, L. L., Chien, A. A., DeFanti, T., Leigh, J. and Papadopoulos, P. M.: The OptIPuter, *Commun. ACM*, Vol. 46, No. 11, pp. 58–67 (2003).
2) Schwehr, K. D., Nishimura, C., Johnson, C. L., Kilb, D. and Nayak, A.: Visualization Tools Facilitate Geological Investigations of Mars Exploration Rover Landing Sites, *IS&T/SPIE Electronic Imaging Proceedings*, pp. 16–20 (2005).
3) 超高速フォトニックネットワーク開発推進協議会: フォトニックネットワーク革命 - 世界最先端のIT国家実現のキーテクノロジー, 超高速フォトニックネットワーク開発推進協議会 (2002).
4) 佐原明夫, 笠原亮一, 山崎悦史, 相澤茂樹, 古賀正文: 光バーストスイッチングネットワークの輻輳制御 -JGN II けいはんなテストベッド実験 -, 信学技報, 社団法人 電子情報通信学会 (2005).
5) Thakur, R. and Gropp, W.: Improving the Performance of Collective Operations in MPICH, *Preprint ANL/MCS-P1038-0403* (2003).
6) Barker, K. J., Benner, A., Hoare, R., Hoisie, A., Jones, A. K., Kerbyson, D. J., Li, D., Melhem, R., Rajamony, R., Schenfeld, E., Shao, S., Stunkel, C. and Walker, P.: On the Feasibility of Optical Circuit Switching for High Performance Computing Systems, *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, Washington, DC, USA, IEEE Computer Society, p. 16 (2005).
7) Afsahi, A. and Dimopoulos, N. J.: Communications Latency Hiding Techniques for a Reconfigurable Optical Interconnect: Benchmark Studies., *Conference on Applied Parallel Computing (PARA '98)*, pp. 1–6 (1998).
8) Afsahi, A. and Dimopoulos, N. J.: Hiding Communication Latency in Reconfigurable Message-Passing Environments., *13th International Parallel Processing Symposium / 10th Symposium on Parallel and Distributed Processing (IPPS / SPDP '99)*, pp. 55–60 (1999).
9) Afsahi, A. and Dimopoulos, N. J.: Collective Communications on a Reconfigurable Optical Interconnect, *International Conference On Principles Of Distributed Systems 1997(OPODIS '97)*, pp. 167–182 (1997).
10) Afsahi, A. and Dimopoulos, N. J.: Analysis of a Latency Hiding Broadcasting Algorithm on a Reconfigurable Optical Interconnect, *Parallel Processing Letters*, Vol. 12, No. 1, pp. 41–50 (2002).