Vol.2014-HPC-145 No.14
2014/7/28

共有受信キューに基づくスケーラブルなプロセッサ間通信の提案

谷本 輝夫[1,a)]　小野 貴継[1]　中島 耕太[1]　三吉 貴史[1]

概要：プロセッサの多コア化により計算機クラスタにおけるノードあたりのノード間通信を行うプロセス数は増加傾向にある．これにより，システムの性能向上やプログラマビリティの向上が期待できる一方で，さらなるプロセス間通信のスケーラビリティ向上が必要となる．本研究では，ノード間のプロセス間通信として，InfiniBand の RC (Reliable Connection) を対象に SRQ (Shared Receive Queue) を用いる際，従来必要であった割り込みを必要としないフロー制御を提案する．提案法は一回のフローコントロールに要する遅延を 95%削減し，アプリケーションの実行トレースを用いたシミュレーションにより SRQ のバッファサイズを最大 86%削減，スケーラビリティを最大 7 倍向上可能であることを確認した．

1. はじめに

プロセッサのコア数は増加傾向にある．60 個以上のコアを持つコプロセッサなど，多数のコアを持つ製品も登場した [2]．これに伴い，計算機クラスタを構成するノードの搭載コア数は増加傾向にあり，スループットの向上が期待される．搭載コア数の増加によりプロセス間通信の実行時間全体への影響は大きくなる．なぜなら，並列度の向上によりプロセス間通信が増加し，通信が実行時間に占める割合が増加するからである．今後はさらなるコア数の増加が予想されることから，メニーコア CPU を活用するためのプロセス間通信は益々重要となる．

分散環境におけるノード間通信の構成は様々あるが，本研究では計算プロセスが通信を行う構成について考える．この構成はノード間通信の低遅延化を図ることができる一方でノード内の通信プロセス数が多くなる．通信には送受信データを一時的に格納しておくためのバッファが必要である．このバッファはノード内の通信プロセス数が増加すると無視できないサイズになり，接続可能なプロセス数（通信のスケーラビリティ）を制限する．したがって，通信のスケーラビリティ向上のためにはバッファサイズを削減する必要がある．

InfiniBand ではスケーラビリティを向上するために，複数の送信元からのパケットを格納する Shared Receive Queue (SRQ) がサポートされている．SRQ にはフロー制御がないため，SRQ が溢れてパケット廃棄が起きる可能性がある．従来法では受信側で割り込みを用いて溢れを防ぐ手法が提案されている [7]．しかしながら，割り込みにかかる時間に到着するパケットを受け取るため多くのバッファが必要であり，スケーラビリティ改善の余地がある．

そこで，本研究では割り込みを必要としないフロー制御によりバッファサイズを削減し，通信のスケーラビリティを向上を図る．具体的には，本研究ではバッファ溢れ時に送信側へ停止要求を送信し，受信側でパケットを受信する準備が完了した後に再送を要求する．評価の結果，フロー制御に要する遅延を 95%削減し，アプリケーションの実行トレースを用いたシミュレーションにより SRQ のバッファサイズを最大 86%削減可能であることが分かった．

本論文の構成は以下の通りである．第 2 章でインタコネクトのスケーラビリティ問題について述べ，第 3 章では提案法について述べる．第 4 章では提案法の評価を行い，第 5 章で関連研究について述べ，第 6 章でまとめる．

2. インタコネクトのスケーラビリティ

HPC におけるクラスタの有力なインタコネクトとして InfiniBand が挙げられる．Top500[8] のうち，InfiniBand を採用したシステムは 224 システムあり，最大勢力である．そこで，本研究では InfiniBand を対象にスケーラビリティの向上を目指す．

2.1 InfiniBand とそのスケーラビリティ問題

InfiniBand ではホストは HCAs (Host Channel Adapters) によってネットワークに接続される．送受信は

[1] 富士通研究所
Kawasaki, Kanagawa 211–8588, Japan
[a)] tanimoto.teruo@jp.fujitsu.com

Send Queue，Receive Queue (RQ) からなる Queue Pair (QP) に対して，Work Queue Request (WQR) をキューに追加することによって制御する．

パケットを受信するためには，受信プロセスが RQ に WQR を追加する．受信プロセスは HCA が受信したパケットを WQR が指すメモリ上の受信バッファに書き込めることを保証しなければならない．したがって，受信プロセスは事前にドライバを経由して受信バッファをページアウトされない領域として登録しておく必要がある．

InfiniBand の仕様では 5 つのトランスポートレイヤプロトコルが定義されている．本研究では最も良く利用されているコネクション型で送達保証のある Reliable Connection を対象とする．

Reliable Connection はコネクション型の通信プロトコルであり，すべての QP は 1 対 1 で接続される．したがってプロセスは通信相手ごとに一定量のバッファを用意する必要がある．プロセスごとのバッファサイズが通信相手プロセス数に比例し，スケーラビリティが問題となる．

2.2 Shared Receive Queue (SRQ) によるスケーラビリティ向上

第 2.1 節で述べた問題を解決するため SRQ がサポートされた．SRQ は複数の QP からのメッセージを書き込むことができる RQ である．これにより，プロセスが必要とする受信バッファ量は通信相手数によらなくなった．

SRQ の実際の利用事例として，MPI 実装への適用が挙げられる [6], [7]．S. Sur らは、MPI 実装の中で広く利用されている MVAPICH へ SRQ を適用することにより，16,000 ノードの構成においてプロセスあたりのメモリ使用量を 10 分の 1 に削減している [7]．

2.3 SRQ の課題

SRQ にはフロー制御がないという問題点がある．SRQ は複数の送信元が 1 つの SRQ に書込みを行う．そのため，送信側で受信側の残りバッファ数を管理することができず，溢れてしまうことがある．バッファが溢れると，Receiver Not Ready (NAK-RNR) が送信側に通知される．送信側 HCA は NAK-RNR を受け取ると，タイムアウト制御で予め設定された回数再送を試みる．

タイムアウトによる再送が発生すると高速なプロセス間通信を実現できない．なぜなら，再送しても SRQ に格納される保証がないため頻繁な再送ができず，タイムアウトの遅延によりアプリケーション性能が低下するためである．したがって，SRQ のバッファが溢れないように受信側で常に WQR を追加する必要がある．

バッファの枯渇を防ぐため MVAPICH への適用では割り込みを用いて WRQ を追加する手法が提案されている [7]．SRQ のバッファ数に閾値を設け，残りバッファ数が閾値

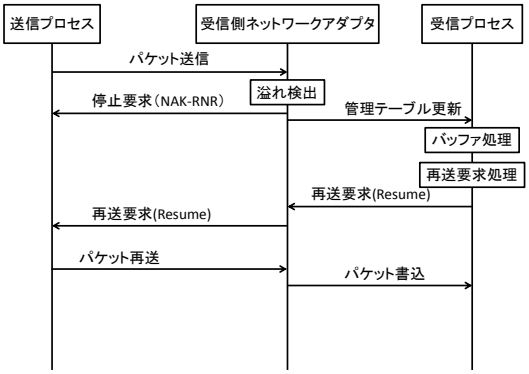

図 1 再送シーケンス

以下になると HCA から CPU に割り込みイベントを起こして受信プロセスにバッファを追加させる．この手法で必要となるバッファサイズは式 (1) で表される．

$$\text{BufferSize[B]} = (t_{int}\text{[s]} + t_{setev}\text{[s]}) \times \text{BandWidth[B/s]} \quad (1)$$

t_{int} と t_{setev} はそれぞれイベントが起きてからスレッドが動作し始めるまでの時間と HCA へ割り込みイベントをセットするのに要する時間であり，BandWidth はネットワークのバンド幅である．割り込みや HCA へのアクセスを伴うことから，バッファの枯渇を避けるために閾値を大きく設定する必要がある．結果として多くのメモリリソースを要し，スケーラビリティが低下するという問題が生じる．

3. SRQ のフロー制御

第 2 章で示した SRQ の問題点を解決するため，SRQ バッファ枯渇時の再送コストを低減させるフロー制御を提案する．割り込みを用いないフロー制御を適用することによって必要な受信バッファ量を削減し，通信のスケーラビリティを向上する．

3.1 停止・再送要求によるフロー制御

既存の SRQ の問題の原因は再送をタイムアウトで制御し，送信側を停止できない点である．そのため受信側で割り込みを用いて SRQ が溢れないようにしなければならず，結果として多くのメモリリソースを必要とする．そこで，プロトコルに停止要求と再送要求を加えたフロー制御を提案する．提案するフロー制御の再送シーケンスを図 1 に示す．再送シーケンスは以下のとおりである．

- SRQ が枯渇しパケット廃棄が起きた時に送信元を停止させる．
- 受信側が SRQ にバッファを追加しメッセージを受け取れる状態になると，受信側から廃棄したパケットの送信元に対して再送要求を送信する．
- 再送要求を受け取った送信元が再送を行う．

受信側 HCA の動作フローを図 2 に示す．停止要求の送

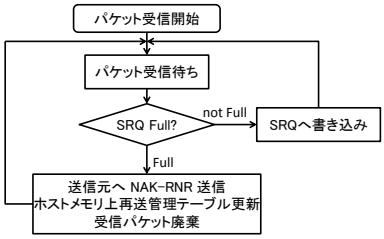

図 2 受信側 HCA 動作フロー

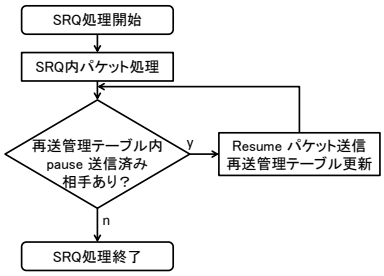

図 3 受信プロセス動作フロー

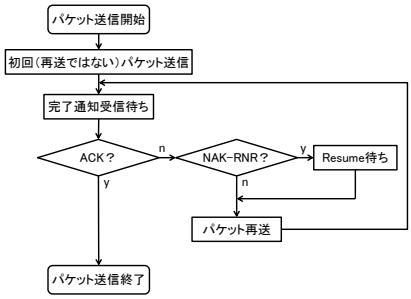

図 4 送信プロセス動作フロー

信と再送管理テーブルへの記録は受信側 HCA によって行う．受信プロセスの動作フローを図 3 に示す．バッファの追加と再送要求の送信は受信プロセスが管理する．受信プロセスは再送要求を送信後，再送管理テーブルを更新する．送信プロセスの動作を図 4 に示す．受信側からの停止要求を受けると，該当 SRQ 宛の送信を停止し，再送要求を受けるとパケットを再送する．

提案法と従来法の本質的な違いは受信側が送信側を止められることである．従来法では SRQ が枯渇しないよう受信側でバッファを追加しなければならないのに対し，提案法では受信側が準備できるまで送信を止めることことができる．したがって，受信バッファのサイズは任意のサイズにすることができる．しかしながら，バッファサイズとスループットの間にはトレードオフがある．従来法との比較のため，スループットが低下しないためのバッファサイズは式 (2) によって求められる．

$$\mathrm{BufferSize[B]} = t_{poll}\mathrm{[s]} \times \mathrm{BandWidth[B/s]} \qquad (2)$$

t_{poll} はポーリングにより受信パケットを取得し，残りバッファ数を計算するのに要する時間であり，BandWidth はネットワークのバンド幅である．従来法と比較して，割り込みやイベントの登録にかかる時間に比べてポーリングに要する時間は短いためバッファの削減が期待できる．

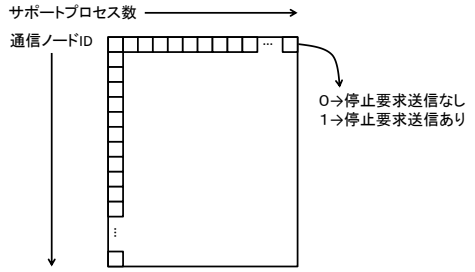

図 5 再送管理テーブルの構成

提案法において，受信プロセスは通信処理をいつ実行するか不明であることからバッファが不足し通信が停止するという懸念が生じる．バッファが不足する状況は受信プロセスが計算処理を行なっているか，バッファの追加が追いつかないかのどちらかにより起きる．このような場合は送信側を止める戦略を取るべきである．なぜなら，従来法では割り込みのオーバヘッドにより元々処理能力を超えている受信プロセスの処理をさらに阻害するためである．

3.2 再送管理テーブルよる再送要求

効率的な停止・再送を実現するためには，必要な通信相手だけに要求を送信できる仕組みが必要である．停止や再送のたびにマルチキャストを行うことも考えられるが，通信相手数が増加するほど無駄な停止要求や再送要求パケットが増加する問題がある．実際に要求を送る必要があるのは廃棄パケットの送信元のみである．そこで，受信側ホストメモリ上に再送管理テーブルを用意し，SRQ のバッファが不足した際当該パケットの送信元に対して停止要求を送信すると共に再送管理テーブルに記録する．

再送管理テーブルの構成を図 5 に示す．テーブルは受信側ホストメモリ上に SRQ ごとに作成する．テーブルに記録すべき情報は「どのノードのどのプロセスからのパケットを破棄し停止要求を送ったか」である．予めサポートするノード数，ノードあたりプロセス数を設定しておき，ノードのアドレスとノード内プロセス番号で送信元プロセスを並べた 2 次元のテーブルを用意する．このテーブルに要する容量は $4K$ ノード，ノードあたり 256 プロセスをサポートする場合に $32K \times 256 \times 1bit = 1MB$ である．

4. 評価

提案法によるスケーラビリティ向上の可能性を調べるため，InfiniBand の RC を用いて手法を模擬するプログラムを作成し，フロー制御にかかる遅延の評価を行った．

評価結果をもとに，MPI の非同期通信を用いたアプリケーションの実行トレースに対して本手法を適応し，フロー制御のペナルティを計算することでバッファ削減効果

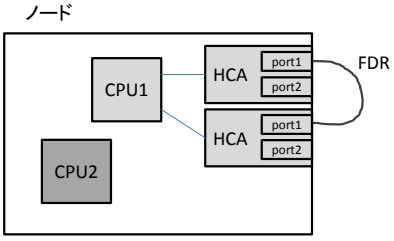

図 6 ハードウェア構成

表 1 ハードウェアパラメータ

CPU Model	Intel Xeon E5-2690 (8C16T) × 2
CPU Freq.	2.90GHz
Memory	64GB
HCA	Mellanox FDR (ConnectX-3)

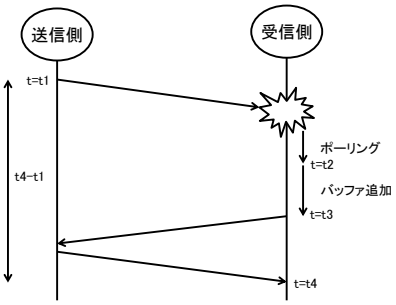

図 7 提案法の動作

の評価を行った．

4.1 一回のフロー制御に要する遅延

提案するフロー制御の通信バッファ削減効果の見積もりに用いるために一回のフロー制御に要する遅延の評価を行った．

4.1.1 評価環境

評価に用いたハードウェアの構成を図 6 に，パラメタを表 1 に示す．1 つの筐体に 2 つの HCA を搭載し，FDR ケーブルで接続した．2 つの HCA は 1 つの CPU (図 6 の CPU1) に接続されている．

本稿における評価では割り込みを利用するプログラムが含まれる．CPU は一定時間アイドルが続くと C ステートと呼ばれる電源状態を遷移させ，消費電力の小さいスリープ状態になる．スリープ状態では割り込みを受けてから動作し始めるまでの遅延が動作状態 (C0) よりも大きくなる．そのため，C ステートが動作状態から遷移しないようにした．

4.1.2 フロー制御の動作

SRQ 溢れが 1 回起きた際に要する遅延と，必要なバッファサイズの評価を行った．提案法（図 7）では，

- 送信プロセスが SRQ を溢れさせるパケットを送信（時刻 $t1$）
- 受信プロセスがパケット到着を検知（時刻 $t2$）

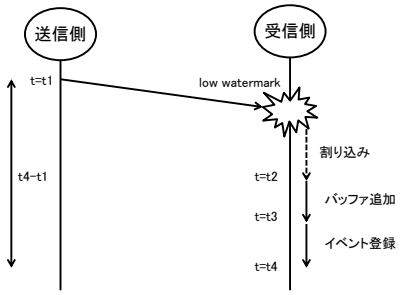

図 8 従来法の動作

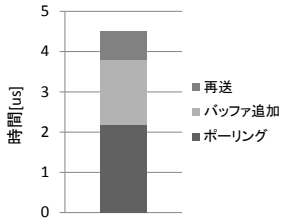

図 9 提案法の遅延

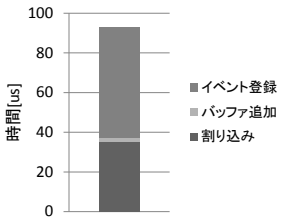

図 10 従来法の遅延

- 受信側プロセスがバッファを追加完了（時刻 $t3$）
- 送信側プロセスに対して再送要求を送信し，それを受信した送信側が廃棄されたパケットを再送し，受信プロセスが受信（時刻 $t4$）

に要する時間（$t4 - t1$）を一回のフロー制御の遅延とした．本来は $t1$ はパケットを送信した時刻ではなく，受信側 HCA に到着した時刻を用いるべきだが，この時刻は取得できないため送信時刻を用いた．これは従来法でも同様である．式 2 において $t_{poll} = t2 - t1$ である．

一方，従来法（図 8）では，

- 送信プロセスがバッファ数が閾値を下回るパケットを送信（時刻 $t1$）
- 受信プロセスが割り込みを受け取り動作開始（時刻 $t2$）
- 受信プロセスがバッファを追加完了（時刻 $t3$）
- 受信プロセスが HCA にイベントをセット（時刻 $t4$）

に要する時間 ($t4 - t1$) を一回のフロー制御の遅延とした．式 2 において $t_{int} = t2 - t1, t_{setev} = t4 - t3$ である．

本稿における評価ではパケットサイズを 64[B] とした．また，一度に追加するバッファ数を 64 個とした．

4.1.3 結果

提案法及び従来法の評価結果を図 9, 図 10 に示す．縦軸は動作に要する時間であり，単位は us である．それぞれのバーはフロー制御に必要な動作に要する時間を積み上げたものである．フロー制御には提案法で 4.51[us], 従来法で 92.8[us] かかる．

フロー制御にかかる時間の内訳を見ると，提案法においてポーリングによりパケットを受け取るまでに要する時間は 2.18[us] である（t_{poll}）．従来法において受信プロセスが割り込みを受けて動作を開始するまでに 35.0[us]（t_{int}）, HCA にイベントを登録するために 55.9[us] 必要と

表 2　評価環境

CPU Model	Intel Xeon E5-2697 v2 (12C24T) × 2
CPU Freq.	2.70GHz
Memory	124GB
HCA	Mellanox FDR (ConnectX-3)
number of nodes	8
OS	RHEL 6.4
MPI	mpich 1.4.1 (gcc)

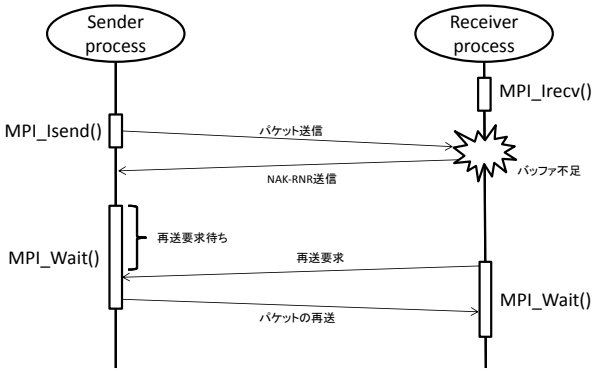

図 11　MPI 非同期通信におけるバッファ溢れ時の動作

する（t_{setev}）．割り込みに 35[us] 要するのは一般的に割り込みにかかる時間が数 us であることを考えると大きい．これは，ユーザレベルのプログラムが ibv_get_async_event() で取得する SRQ_LIMIT_REACHED イベントを HCA が CPU に対して通知するまでの遅延が大きいためと考えられる．

4.2　MPI アプリケーションにおけるバッファ削減効果

4.1 節の結果に基づき，アプリケーションの実行トレースに対して提案法を適用した場合の SRQ のバッファ削減効果の評価を行った．ベンチマークアプリケーションとして NAS Parallel Benchmarks[1] を用い，その中でも非同期通信（MPI_Isend と MPI_Irecv）を用いる bt, sp を使用した．アプリケーションの入力サイズとしてクラス A を用い，実行トレースの取得には MPE を用いた．MPE を用いることで MPI 関数がコールされた時刻およびコールした rank を取得することができる．

表 2 に実行トレースを取得した計算機環境を示す．実行環境は 8 ノードからなり，合計で 192 個の CPU コアを有する．

4.2.1　非同期 MPI 通信への適用

送信プロセスからのパケットがバッファ溢れを起こさない場合，送信プロセスは MPI_Wait コール時に NAK が届いていないことを確認し，送信が成功したと判断する．図 11 にバッファ溢れが起きた場合の動作を示す．送信されたパケットにより SRQ が溢れた場合，送信プロセスは MPI_Wait コール時に NAK-RNR の受信を確認し，パケットがドロップしたことを検出する．すると送信プロセスは

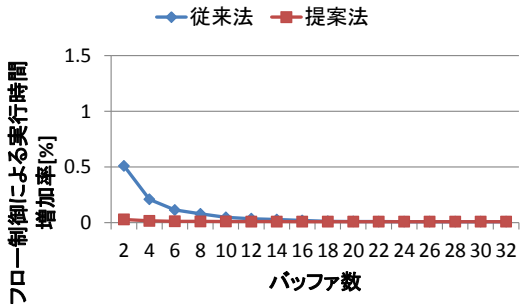

図 12　bt (100 プロセス) のバッファサイズと実行時間増加率の関係

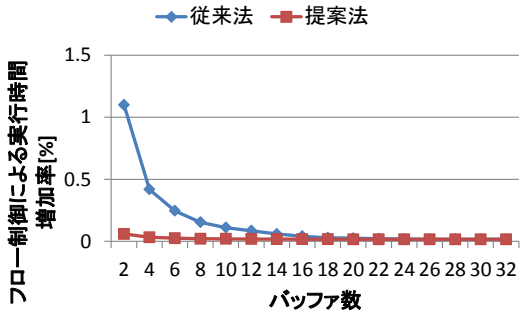

図 13　bt (196 プロセス) のバッファサイズと実行時間増加率の関係

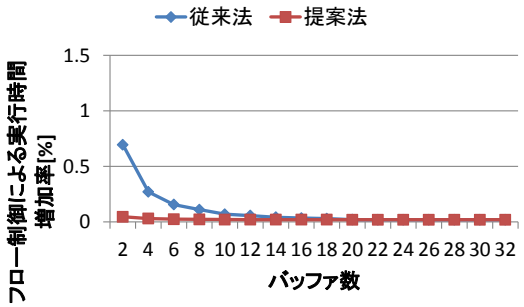

図 14　sp (100 プロセス) のバッファサイズと実行時間増加率の関係

受信プロセスから再送要求が送られるまでブロックして待つ．ある時点で受信プロセスが通信処理を行い，バッファを追加した後に再送管理テーブルを参照して再送要求を送信する．この再送要求を受け取った送信プロセスはバッファ溢れを起こし，廃棄されたパケットを再送する．再送パケットは必ず SRQ に受信されるものとし，送信プロセスは MPI_Wait を抜ける．

今回のシミュレーションでは，各 MPI 通信において一つのパケットが送受信されるものとし，その際に 32KB の固定長のバッファが消費されるものとした．提案法の評価パラメータとして，一回のフロー制御に 4.51[us] かかるものとし，従来法では一回のフロー制御に 92.8[us] かかるものとした．

4.2.2　結果

図 12,13,14,15 はバッファ数を変えた時の並列数が 100, 196 の時のフロー制御による実行時間の増加率を示したものである．提案手法を適応することにより，フロー制御の

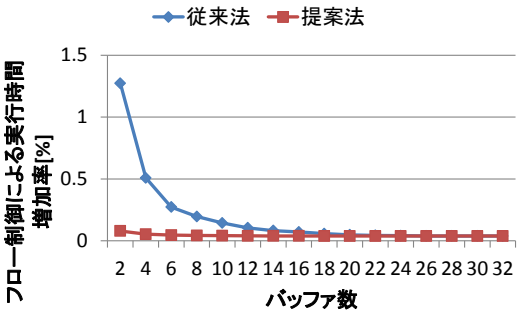

図 15 sp (196 プロセス) のバッファサイズと実行時間増加率の関係

ペナルティを削減可能であることがわかる．フロー制御のペナルティの実行時間に対する影響の許容値を 0.1%とすると，バッファ削減率は bt, sp の 196 プロセス時で 83%, 86%となる．プロセス間通信に必要なリソースがバッファサイズによって決まると仮定すると，提案法では従来法に比べてスケーラビリティを最大 7 倍向上させることが可能であると言える．

5. 関連研究

InfiniBand を用いたクラスタ構成においてスケーラビリティに関する研究が行われてきた．SRQ を MPI に適用した研究では，MPI のプロセス間通信に SRQ を用いることで，プロセスあたりの通信に必要なメモリリソース量を削減し，スケーラビリティが向上することを示している [6], [7]．また，B-SRQ ではバッファサイズの異なる複数の SRQ を用いて送信パケットサイズに適したキューを使うことでリソースの効率化を図っている [4]．これらの研究がシステム全体のプロセス数の増加に関するものであるのに対し，本研究はメニーコアシステムにおけるノード内のプロセス数増加に対応するインタコネクトの探索を指向している点で異なる．

RC における通信では，各プロセスがすべての通信相手と QP によるコネクションを必要とする．RC のコネクションも一定のメモリリソースを必要とするため，スケーラビリティ問題が生じる．そのため，InfiniBand HCA の主要ベンダである Mellanox 社は eXtended Reliable Connection (XRC) と呼ばれる機能を提供している．XRC は，1 つの QP が複数の SRQ へメッセージを書き込めるようにし，すべてのプロセス間で必要だったコネクションを，ノード内のプロセス数と同等まで削減するものである．XRC を MVAPICH へ実装した研究では，コネクションの最適化に取りくんでいる [3]．また，OpenMPI へ実装に適用した研究では，XRC によりコネクション数を削減するだけでなく，B-SRQ におけるコネクションの増加問題を解決する技術として XRC を用いた X-SRQ を提案している [5]．ノード間のコネクションの最適化は本研究で提案した SRQ のフロー制御とあわせて用いることで，さらなるメモリリソースの削減が可能である．

6. おわりに

メニーコアシステムに対応したインタコネクトの実現のため，InfiniBand の SRQ のスケーラビリティを向上するフロー制御を提案した．受信側がバッファを追加し受信可能になった時点で再送することで，従来のタイムアウトによる再送に比べ再送にかかるコストを軽減する．従来法では受信側バッファが枯渇しないよう割り込みを用いてバッファを追加する．提案法を適用することで送信側を止めることができるようになり，割り込みを用いる必要がなくなるため，受信側バッファを削減することができる．評価の結果，一回のフロー制御に要する遅延を 95%削減し，SRQ のバッファサイズを最大で 86%削減可能であることが分かった．

本稿における評価は提案法を模擬するプログラムによる実機評価と，その結果と実アプリケーションの実行トレースに基づいたバッファ削減率の見積もりである．より多くのアプリケーションへ適用した場合のバッファ削減率の評価や，より大規模なシステムを想定した評価は今後の課題である．

参考文献

[1] Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Simon, H. D., Venkatakrishnan, V. and Weeratunga, S. K.: The nas parallel benchmarks, Technical report, The International Journal of Supercomputer Applications (1991).

[2] Chrysos, G.: Knights Corner, Intel ' s first Many Integrated Core (MIC) Architecture Product, *Proc. of the 24th Hot Chips* (2012).

[3] Koop, M. J., Sridhar, J. K. and Panda, D. K.: Scalable MPI design over InfiniBand using eXtended Reliable Connection, *Proc. of the CLUSTER'08*, pp. 203–212 (online), DOI: 10.1109/CLUSTR.2008.4663773 (2008).

[4] Shipman, G., Brightwell, R., Barrett, B., Squyres, J. and Bloch, G.: Investigations on InfiniBand: Efficient Network Buffer Utilization at Scale, *Proc. of the 14th EuroPVM/MPI*, pp. 178–186 (online), DOI: 10.1007/978-3-540-75416-9_28 (2007).

[5] Shipman, G., Poole, S., Shamis, P. and Rabinovitz, I.: X-SRQ - Improving Scalability and Performance of Multi-core InfiniBand Clusters, *Proc. of the 15th EuroPVM/MPI*, pp. 33–42 (online), DOI: 10.1007/978-3-540-87475-1_11 (2008).

[6] Shipman, G., Woodall, T., Graham, R., Maccabe, A. and Bridges, P.: Infiniband scalability in Open MPI, *Proc. of the 20th IPDPS*, (online), DOI: 10.1109/IPDPS.2006.1639335 (2006).

[7] Sur, S. and Panda, D. K.: Shared receive queue based scalable MPI design for InfiniBand clusters, *Proc. of the 20th IPDPS*, (online), DOI: 10.1109/IPDPS.2006.1639336 (2006).

[8] TOP500.Org: TOP500 November 2012, `http://www.top500.org/lists/2012/11/`.