パケットペーシングによる全対全通信の最適化とシミュレーション評価

柴村英智[†1] 三輪英樹[†2] 薄田竜太郎[†1]
平尾智也[†1] 安島雄一郎[†2] 三吉郁夫[†2]
清水俊幸[†2] 石畑宏明[†3] 井上弘士[†4]

本論文では，複数のメッセージを同時送受信可能な 2 次元トーラス網上で，ネットワークのバンド幅を活用する全対全通信アルゴリズム（A2AT）の定量的な性能評価とパケットペーシングによる最適化について述べる．A2AT と従来の全対全通信アルゴリズムとの性能比較，ならびに，A2AT にペーシングを適用した実行性能について，NSIM と呼ぶインターコネクトシミュレータを用いて評価を行った．その結果，A2AT は他のアルゴリズムよりも良い通信性能を実現することがわかった．また，パケットペーシングによって高いリンクスループットを維持した高速な実行を達成でき，バンド幅を積極的に利用する A2AT のようなアプリケーションにペーシングが効果的であることを確認した．

Optimization and Simulation Evaluation of an All-to-all Communication using Packet Pacing

Hidetomo Shibamura,[†1] Hideki Miwa,[†2] Ryutaro Susukita,[†1]
Tomoya Hirao,[†1] Yuichiro Ajima,[†2] Ikuo Miyoshi,[†2]
Toshiyuki Shimizu,[†2] Hiroaki Ishihata[†3] and Koji Inoue [†4]

This paper presents quantitative performance evaluation and optimization of A2AT, an optimal all-to-all communication algorithm. This algorithm exploits the network bandwidth of 2D-torus network which nodes can transmit and receive multiple messages simultaneously. The A2AT is compared with conventional all-to-all algorithms and performance improvement by explicit packet pacing is examined by using an interconnection network simulator NSIM. The result shows that A2AT achieves good performance than the other algorithms and holds higher link throughput with packet pacing.

1. はじめに

並列処理における全対全通信は，個々の処理ノードが持つデータを全てのノードへ分配するため，インターコネクト（相互結合網）への負荷が非常に高い．そこで，全対全通信はインターコネクトの評価項目の一つとして利用されるとともに，これまでに数多くのアルゴリズムが考案されている．

近年では，ノードに複数の通信エンジンを搭載し，複数のリンクから同時にメッセージを送出可能なインターコネクトが提案されている[1)]．また，2 次元のメッシュ網やトーラス網を対象に，このような同時送信機構を活用する全対全通信アルゴリズム（A2AT）が提案されている[2)]．このアルゴリズムでは，全てのリンク上に，常に同数のメッセージ通信が重なり合うようスケジューリングされており，リンクバンド幅を公平に共有することで全体の通信効率を高めている．すなわち，高い通信性能を維持するためには，異なるメッセージ同士でリンクを公平に共有する必要がある．

一方，インターネット環境を基盤とするマルチメディア配信やグリッドコンピューティングの分野では，複数のバースト的なトラフィックによってネットワークの通信帯域を過剰に圧迫することによる性能の低下が問題となっている．これを解決するために，パケットの送出間隔を制御しトラフィックを平滑化する，ペーシン

†1 （財）九州先端科学技術研究所
Institute of Systems, Information Technologies and Nanotechnologies
†2 富士通株式会社
Fujitsu Ltd.
†3 東京工科大学
Tokyo University of Technology
†4 九州大学
Kyushu University

ⓒ 2011 Information Processing Society of Japan

グに関する研究や評価実験が盛んに行われている[3)–5)]．このようなペーシング技術を利用したインターネットワーキングでは非常に高いネットワーク利用効率を達成しており，高スループットも重要視されるインターコネクトへの応用展開が期待できる．一方で，並列計算機におけるインターコネクトへの応用はほとんどなく，BG/L ではバリア同期のペーシング性を利用した全対全通信にとどまっている[6)]．

本研究では，全対全通信のような高トラフィックとなるアプリケーションを対象に，インターコネクト上でのパケットペーシングの効果をシミュレーションによって定量的に評価することを目的とする．

本論文では，まず，A2AT や既存の全対全通信アルゴリズムの実行性能について比較評価を行う．次に，A2AT を対象に，様々なパケットの送出間隔（パケット間ギャップ）によるペーシングの有効性について評価する．そして，詳細なシミュレーション解析に基づき，A2AT の実行時間が全対全通信の理想実行時間に近づくよう，パケット間ギャップ値を最適化し，その効果を明らかにする．なお，ペーシングによる通信状況の変化を詳細に解析するために，64 から 256 ノード程度の小規模な 2 次元トーラス網を評価の対象とし，大規模システムについての評価は今後の課題とする．

以下，第 2 章では評価対象とする全対全通信アルゴリズムについて述べ，第 3 章では NSIM と呼ぶインターコネクトシミュレータを概説する．第 4 章では全対全通信アルゴリズムの比較実験やパケットペーシングによる評価実験について述べ，実験結果について考察する．そして，第 5 章でまとめる．

2. A2AT：最適全対全通信アルゴリズム

本研究では，文献 2) で提案されている最適全対全通信アルゴリズム（A2AT）に焦点を絞る．

A2AT は，ノード間を双方向リンクで接続した 2 次元のメッシュ網やトーラス網を対象とする．また，各ノードは，4 近傍に対してメッセージを同時送信／同時受信する機構を持つことを前提とする．なお，本研究における同時送信とは，ある期間中に複数のメッセージを並列に送信できることを意味し，同一時刻に送信を開始するものではない．同時受信も同様である．

常に全てのリンクを利用し，ある時刻において各リンクは全体を通して同数のメッセージで共有するように送受信がスケジューリングされている．自ノード座標を $(0,0)$ とし，送信先ノード座標を (x,y) と表す場合，A2AT では，$|x|+|y|=$ ホップ数 となる 4 つのノードに対して同時送信を行う．ここで，自ノードからの送信を座標軸方向と象限方向に大別する．座標軸方向への送信は，**図 1** のように $(i,0)$, $(-i,0)$, $(0,i)$, $(0,-i)$ となり，象限方向への送信は，**図 2** のように (i,j), $(-i,-j)$, $(-j,i)$, $(j,-i)$ となる．

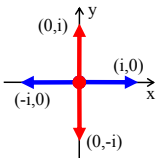

図 1 座標軸方向への送信
Fig. 1 Sending toward four axes

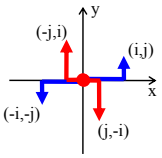

図 2 象限方向への送信
Fig. 2 Sending toward four quadrants

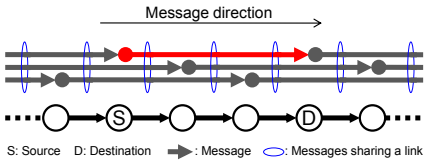

図 3 $+X$ 方向におけるリンクの共有
Fig. 3 Link sharing on $+X$ direction

全ノードが一斉に同様の送信を行うと，メッセージは経路上のリンクで他のメッセージと重なり合う．座標軸方向の送信では，$+X$ 方向へのリンクは i 個（= ホップ数）のメッセージで共有される．例えば，3 ホップの送信では，**図 3** のように各リンクは 3 つのメッセージでリンクを共有する．$-X$, $+Y$, $-Y$ 方向についても同様で，これらは他方向の通信を妨げることはない．一方，象限方向への送信時は，経路上のリンクを $(i+j)$ 個（= ホップ数）のメッセージで共有する．

このように，A2AT では，常にホップ数と同数のメッセージが各リンクを経由し，リンクバンド幅を公平に共有することで通信効率の向上を図っている．

図 4 に A2AT のアルゴリズムを示す． アルゴリズム全体では，4 方向に対して 4 メッセージを送信した後，4 方向から 4 メッセージを受信することを繰り返す．また，4 つの受信がすべて完了しない限り，次の送信は行われずブロックされる．そして，自ノードを中心に送信先ノードを放射状に広げる． これらは，6 つのプログラムフェーズから成り，ノード数の奇偶数やメッセージの送信方向に応じて実行されるフェーズが時々刻々と変化する．

3. NSIM：インターコネクトシミュレータ

現在，我々は NSIM と呼ぶインターコネクトシミュレータを開発している[7)]．NSIM は，数万ノードを接続する次世代インターコネクトの性能評価を目的とし，設計・開発現場での実践的な利用を念頭に，実用時間

ⓒ 2011 Information Processing Society of Japan

```
Xsize = Ysize = n = mat_size ;

for ( i=0; i<n; i++ ) {
  if ( 0 < i && i <= (n-1)/2 ) {
    // Phase 1
    recv( -i,  0, msg_size, NIC0, &r_req[0] );
    recv(  i,  0, msg_size, NIC1, &r_req[1] );
    recv(  0, -i, msg_size, NIC2, &r_req[2] );
    recv(  0,  i, msg_size, NIC3, &r_req[3] );
    send(  i,  0, msg_size, NIC0, &s_req[0] );
    send( -i,  0, msg_size, NIC1, &s_req[1] );
    send(  0,  i, msg_size, NIC2, &s_req[2] );
    send(  0, -i, msg_size, NIC3, &s_req[3] );
    waitall( 4, s_req, s_stat );
    waitall( 4, r_req, r_stat );
  }

  if ( 0 < i && i <= (n-1)/2 ) {
    // Phase 2
    recv( -i, -i, msg_size, NIC0, &r_req[0] );
    recv(  i,  i, msg_size, NIC1, &r_req[1] );
    recv(  i, -i, msg_size, NIC2, &r_req[2] );
    recv( -i,  i, msg_size, NIC3, &r_req[3] );
    send(  i,  i, msg_size, NIC0, &s_req[0] );
    send( -i, -i, msg_size, NIC1, &s_req[1] );
    send( -i,  i, msg_size, NIC2, &s_req[2] );
    send(  i, -i, msg_size, NIC3, &s_req[3] );
    waitall( 4, s_req, s_stat );
    waitall( 4, r_req, r_stat );
  }

  for ( j=0; j<n; j++ ) {
    if ( 0 < j && j < i && i <= (n-1)/2 ) {
      // Phase 3
      recv(  i,  j, msg_size, NIC0, &r_req[0] );
      recv( -i, -j, msg_size, NIC1, &r_req[1] );
      recv(  j,  i, msg_size, NIC2, &r_req[2] );
      recv( -j, -i, msg_size, NIC3, &r_req[3] );
      send( -i, -j, msg_size, NIC0, &s_req[0] );
      send(  i,  j, msg_size, NIC1, &s_req[1] );
      send( -j, -i, msg_size, NIC2, &s_req[2] );
      send(  j,  i, msg_size, NIC3, &s_req[3] );
      waitall( 4, s_req, s_stat );
      waitall( 4, r_req, r_stat );
    }
    // Phase 4
    recv( -j,  i, msg_size, NIC0, &r_req[0] );
    recv(  j, -i, msg_size, NIC1, &r_req[1] );
    recv( -i,  j, msg_size, NIC2, &r_req[2] );
    recv(  i, -j, msg_size, NIC3, &r_req[3] );
    send(  j, -i, msg_size, NIC0, &s_req[0] );
    send( -j,  i, msg_size, NIC1, &s_req[1] );
    send(  i, -j, msg_size, NIC2, &s_req[2] );
    send( -i,  j, msg_size, NIC3, &s_req[3] );
    waitall( 4, s_req, s_stat );
    waitall( 4, r_req, r_stat );
  }
}
```

偶数ノード数時の実行フェーズ

```
if ( n % 2 == 0 ) {
  for ( k=1; k<= n/2-1; k++ ) {
    // Phase 5
    recv( -n/2,  -k, msg_size, NIC0, &r_req[0] );
    recv(  n/2,   k, msg_size, NIC1, &r_req[1] );
    recv(  -k, -n/2, msg_size, NIC2, &r_req[2] );
    recv(   k,  n/2, msg_size, NIC3, &r_req[3] );
    send(  n/2,   k, msg_size, NIC0, &s_req[0] );
    send( -n/2,  -k, msg_size, NIC1, &s_req[1] );
    send(   k,  n/2, msg_size, NIC2, &s_req[2] );
    send(  -k, -n/2, msg_size, NIC3, &s_req[3] );
    waitall( 4, s_req, s_stat );
    waitall( 4, r_req, r_stat );
  }
  // Phase 6
  recv(  n/2,  n/2, msg_size, NIC0, &r_req[0] );
  recv( -n/2,    0, msg_size, NIC1, &r_req[1] );
  recv(    0, -n/2, msg_size, NIC2, &r_req[2] );
  send( -n/2, -n/2, msg_size, NIC0, &s_req[0] );
  send(  n/2,    0, msg_size, NIC1, &s_req[1] );
  send(    0,  n/2, msg_size, NIC2, &s_req[2] );
  waitall( 3, s_req, s_stat );
  waitall( 3, r_req, r_stat );
}
```

図 4 A2AT のアルゴリズム
Fig. 4 Algorithm of A2AT

内でシミュレーションを完了することを目指している．

これまでに開発されてきたシミュレータの多くは，実際の並列計算機から取得した通信ログや人工的に生成した通信パターンを利用することが多い．一方，インターコネクトのトポロジや通信性能の差異によってプログラムの挙動が動的に変化する場合は，正確なシミュレーションを行うことが難しい．また，大規模インターコネクトのシミュレーション時には，相当する通信ログの取得限界問題が障壁となる[8),9)]．

そこで，アプリケーションの振る舞いに忠実に従ったシミュレーションを行うために，NSIM では MGEN プログラムと呼ぶ MPI 互換のインタフェースで記述したプログラムを入力とし，シミュレーション時にそのプログラムを駆動させることによって通信パターンを生成している．また，インターコネクトに関する仕様を設定ファイルで与えるため，実機のみならず未知のインターコネクトの性能評価にも利用できる．

シミュレーションの終了後，プログラムの予測実行時間，リンクスループット，レイテンシ，バッファ利用率，リンクビジー率，通信履歴といった数多くの統計情報を出力する．通信履歴は，MPICH2[10)] に含まれる Jumpshot の入力形式に変換し可視化できる．

4. 評 価 実 験

4.1 評価対象システム

本研究で仮定する評価対象システムは，近年のスーパーコンピュータ相当の機能・性能を有するものとし，**表 1** に示す仕様を NSIM に与える．なお，現在の主流となる技術・性能を反映した値を用いており，表中の仕様を持つシステムは実在しない．

表 1 評価対象システムの仕様
Table 1 Specification of evaluation system

パラメータ	設定値
トポロジ	2 次元トーラス網
ルーティング方式	次元順+dateline
パケット転送方式	virtual cut-through
フロー制御方式	クレジットベース
ノード間リンクバンド幅	4GB/s（単方向分）
ルーティング計算時間 (RC)	4ns
仮想チャネル設定時間 (VA)	4ns
スイッチ設定時間 (SA)	4ns
フリット転送時間 (ST)	4ns
スイッチ遅延時間	78ns
ケーブル遅延時間	10ns
MTU	2KiB
パケット長	32B〜2KiB(MTU)
パケットヘッダ長	32B
仮想チャネル数	2
仮想チャネルバッファ	8KiB (MTU×4)
フリット長	16B
NIC 数	4（同時送受信可能）
DMA 転送レート	16GB/s
メモリバンド幅	16GB/s
MPI 関数処理オーバヘッド	200ns

4.2 実験 1：全対全通信アルゴリズムの性能比較

本実験では，A2AT（以下，a2at）と従来の全対全通信アルゴリズムの基本的な実行性能を比較する．まず，$n \times n$ の 2 次元トーラス網における全対全通信の理想実行時間 (T_{ideal}) は，バイセクションリンクを通過する総データ量 Q とバイセクションバンド幅 B_{bisect} に用いて次式で求められる．

$$T_{ideal} = \frac{Q}{B_{bisect}}$$

ここで，1 ノードに送信するメッセージサイズを L_{mesg}，単方向のリンクバンド幅を B_{link} とすると，Q と B_{bisect} は，それぞれ，

$$\begin{aligned} Q &= \left(\lfloor\frac{n}{2}\rfloor n\lceil\frac{n}{2}\rceil n + \lceil\frac{n}{2}\rceil n\lfloor\frac{n}{2}\rfloor n\right)L_{mesg} \\ &= 2n^2\lfloor\frac{n}{2}\rfloor\lceil\frac{n}{2}\rceil L_{mesg}, \\ B_{bisect} &= 4nB_{link} \end{aligned}$$

となり，理想実行時間は以下の式で求められる．

$$T_{ideal} = \frac{n}{2}\left\lfloor\frac{n}{2}\right\rfloor\left\lceil\frac{n}{2}\right\rceil\frac{L_{mesg}}{B_{link}}$$

この式から，評価対象システム上で 1MiB メッセージの全対全通信を行った場合の理想実行時間は，パケットヘッダの転送オーバヘッドも考慮すると，8×8 では 17.0ms ，同様に，9×9 では 24.0ms となる．

ⓒ 2011 Information Processing Society of Japan

表 2 実験 1 の評価パラメータ
Table 2 Evaluation parameters for exp.1

パラメータ	設定値
アルゴリズム	a2at, pwx, ring
ノード数	64 (8 × 8), 256 (16 × 16), 1,024 (32 × 32), 4,096 (64 × 64)
使用 NIC 数	4: a2at, 1: pwx, ring
メッセージサイズ	64MiB 領域 ÷ ノード数 †

† 問題サイズを同一にするため.

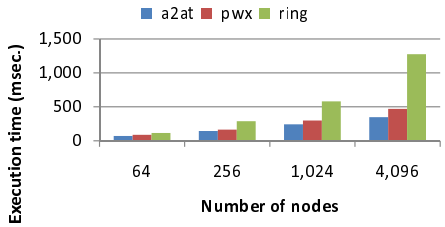

図 5 各アルゴリズムの実行時間
Fig. 5 All-to-all execution time

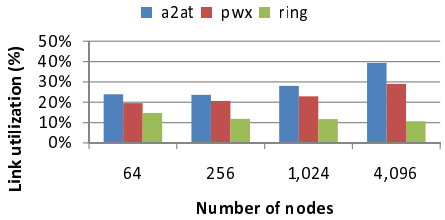

図 6 各アルゴリズムのリンク利用率
Fig. 6 All-to-all link utilization

比較するアルゴリズムには，pairwise exchange (以下，pwx) と ring を用いる．また，**表 2** に実験 1 に用いた評価パラメータを示す．

4.2.1 実験 1：結果と考察

ノード数に応じた各アルゴリズムの実行時間を**図 5** に示す．この結果から，a2at が pwx や ring よりも実行性能が良いことがわかる．また，**図 6** に示すリンク利用率も a2at の方が高い．しかし，a2at は複数の NIC を利用した同時送受信を行うためリンクを利用する機会が多い．それにも関わらず他のアルゴリズムと比較するとリンク利用率が低いといえる．

ここで，実行中のリンクスループットの時系列変化に着目する．**図 7** は，a2at-8x8 のリンクスループットの推移であり，次元方向 ($+X$, $-X$, $+Y$, $-Y$) 毎の平均と全体平均（AVE.）に分けている．

実行開始時は 4 近傍への 1 ホップの送信であるため，すべてのリンクは 1 メッセージのみを転送する．したがって，スループットは最大（4GB/s）となる．しかし，ホップ数が増えるとルータへの過度のパケット投入によりルータ内の VC(仮想チャネル) バッファ

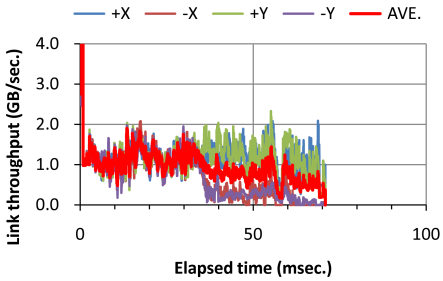

図 7 a2at-8x8 のリンクスループット（実行時間：71.3ms）
Fig. 7 Link throughput of a2at-8x8

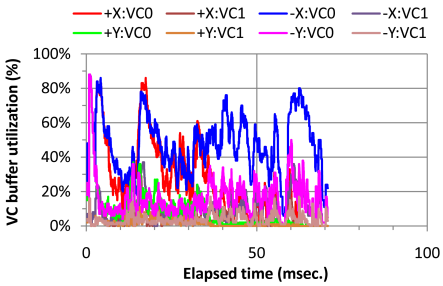

図 8 a2at-8x8 の平均 VC バッファ利用率
Fig. 8 Average VC buffer utilization of a2at-8x8

が飽和し通信が滞り始める．これを裏付けるために，実行中の VC バッファの平均利用率の変化を**図 8** に示す．この図から，実行開始後，$+X$,$-X$,$+Y$,$-Y$ 方向の仮想チャネル番号 0 のバッファ利用率が急激に高くなっていることが確認できる．

図 7 において，実行の前半は各リンクとも同数のメッセージで共有されるため各次元のスループットはほぼ同じである．しかし，後半（32msec. 以降）は各次元で偏り，全体的に平均バンド幅が低下している．これは，ノード数が偶数のためである．A2AT は，偶数ノード数時にプログラム後半部で最遠部（自ノードを中心とした場合に最も遠くなる辺部分）への送信を行う．その際，一般的な次元順ルーティングではリンクを公平に共有する方向へ送信できず，リンクを共有するメッセージ数が偏るためである．**図 9** に，a2at-9x9 のリンクスループットの推移を示す．この図から，奇数ノード数時はリンクを共有するメッセージ数に偏りが起こらないため，各次元のスループットは一貫して同じになっていることが確認できる．

4.3 実験 2：パケットペーシングによる性能改善

A2AT では，リンクを共有するメッセージ数は高々ホップ数である．よって，理論的には，自ノードからのパケットの送出時に（(ホップ数 −1) × パケット長）のリンク転送に要する時間以上の非送出期間を設ける

ⓒ 2011 Information Processing Society of Japan

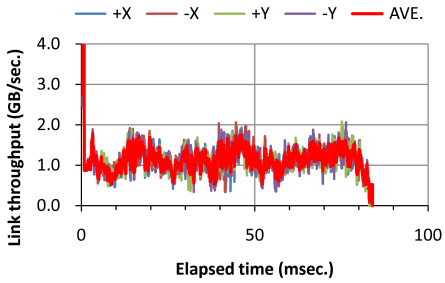

図 9 a2at-9x9 のリンクスループット（実行時間：84.0ms）
Fig. 9 Link throughput of a2at-9x9

表 3 実験 2 の評価パラメータ
Table 3 Evaluation parameters for exp.2

パラメータ	設定値
アルゴリズム	a2at
ノード数	8×8，9×9，···，16×16
メッセージサイズ	1MiB
パケット間ギャップ	（ホップ数 −1）+ ギャップバイアス
ギャップバイアス	−2.0 ～ +16.0 (ステップ：1.0)
	0 ～ +6.0 (ステップ：0.125)

（ペーシングする）ことで，他のパケットと干渉することなく交互にリンクを利用することが可能である．ここで，パケット送出時に n パケットのリンク転送に要する時間だけ待たせる場合を，**パケット間ギャップ** $= n$（ただし，$n \geq 0$）と定義する．なお，パケット間ギャップが 0 の場合，パケットは連続して送出される．

本実験では，A2AT に対して全体の実行を最小化するパケット間ギャップ値を調査し，パケットペーシングの効果を明らかにする． なお，ノード数の増加によってホップ数が増えると，通信混雑も増加すると考えられる．そこで， 理論的に最適なパケット間ギャップ値である（ホップ数 −1）に，様々な**ギャップバイアス**を加えた実行を行う．すなわち，ギャップバイアスを変化させ，理論的に最適なパケット間ギャップ値を中心とした評価実験を行う．**表 3** に実験に用いた評価パラメータを示す．

パケット間ギャップ値は，NSIM が有する高レベル API 関数で設定し，刻み幅を 1/8 とすることで最小でパケット長の 1/8 のギャップを挿入する．

4.3.1 実験 2：結果と考察

図 10 に，各ノードサイズにおいて，−2 から +16 までギャップバイアスを変化させた場合の A2AT の実行時間を示す．そして，ペーシングの効果が顕著に表れているギャップバイアスが 0 から +6 について詳細に調査した結果を**図 11** に示す．これらのグラフから，a2at-8x8 から a2at-16x16 まで，ペーシングによって実行性能が向上していることが確認できる．

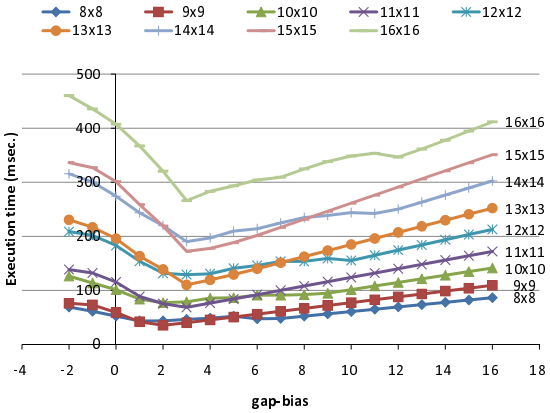

図 10 各ギャップバイアスにおける実行時間
Fig. 10 Execution time for each gap-bias

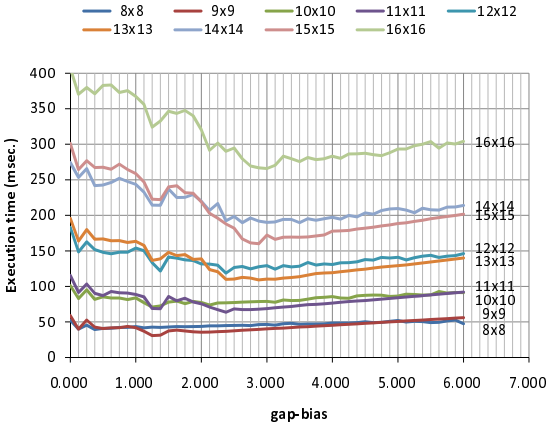

図 11 各ギャップバイアスにおける実行時間（詳細）
Fig. 11 Execution time for each gap-bias (details)

一方，A2AT の実行時間を最小化する最適なパケット間ギャップは，前節で示した理論値（ホップ数 −1）よりも大きいことがわかる．また，最適値は一意には定まらず，ノード数に応じて変化することがわかった．

ノード数が偶数と奇数の違いに注目すると，例えばギャップバイアスが 3 の場合に，偶数ノード数の 14×14 よりもノード数が多い奇数ノード数の 15×15 の方が，実行時間が小さくなっている．これは，実験 1 で述べたように，奇数ノード数ではリンクを共有するメッセージ数の偏りが発生せず，ペーシングが効果的に作用しためである．

ギャップバイアスに対する平均リンク利用率を**図 12** に示す．ギャップバイアスが 0 の場合に a2at-8x8 に着目すると，リンク利用率が 25%（図 6 参照）から 30%以上に増加している．すなわち，ペーシングによってパケットが順調に流れ出したためである．また，ギャップバイアスが 1.250 から 1.375 付近で，全体的

ⓒ 2011 Information Processing Society of Japan

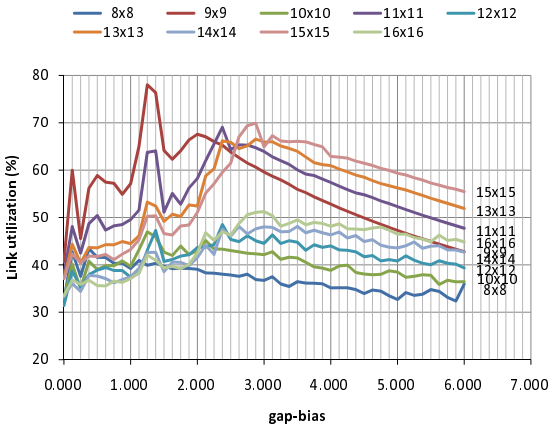

図 12 各ギャップバイアスにおける平均リンク利用率
Fig. 12 Average link utilization for each gap-bias

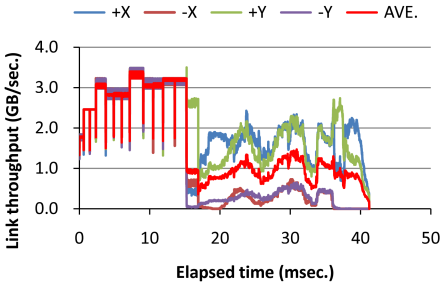

図 13 ペーシングを適用した 2at-8x8 のリンクスループット（実行時間：42.7ms）
Fig. 13 Link throughput of a2at-8x8 with pacing

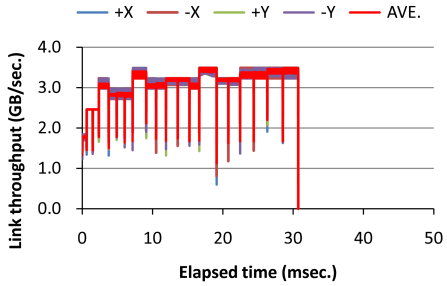

図 14 ペーシングを適用した a2at-9x9 のリンクスループット（実行時間：30.7ms）
Fig. 14 Link throughput of a2at-9x9 with pacing

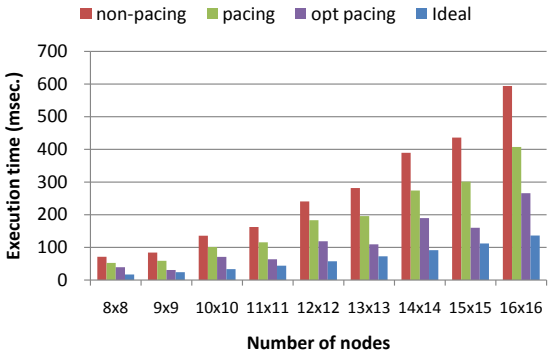

図 15 パケットペーシングの効果
Fig. 15 Effeciency of packet pacing

にリンク利用率が急激に増加している．この点については，第 4.4.1 節で議論する．さらにギャップバイアスが大きくなるとリンクを流れるパケットが疎らになり，リンク利用率が徐々に低下している．以上の解析によって，各ノード数における最適なギャップバイアスが求められた．

次に，ノード数が 8×8 と 9×9 の場合について言及する．**図 13**，および**図 14** に，それぞれのリンクスループットの推移を示す．これらは，ギャップバイアスが 1.250 で，実行時間が最も小さくなる場合である．

それぞれの図において，不定間隔でスパイク状にリンクバンド幅が低下している箇所がある．これは，A2AT では，4 つの送信方向に応じていくつかのプログラムフェーズを変えながら実行しており，メッセージの受信待ちと送信開始処理によって通信がないためである．なお，a2at-9x9 の場合では，20 回のフェーズ実行を行っており，リンクバンド幅の出力は順次 20 個のシーケンスとして分類できる．

両図から，実行前半部は同様のリンクスループットの変化を示しており，後半部はノード数の奇偶数による差異が明確である．また，ペーシングを行わない場合（図 7，図 9 ）と比較して約 3 倍以上向上し，大きな変動も無く安定した通信が行われていることがわかる．その結果，a2at-9x9 の実行時間は 84.0ms から 30.7ms へと 2.74 倍速くなっており，理想実行時間の 127.9%になっている．

図 15 に，本実験におけるパケットペーシングの効果をまとめる．各ノードにおける棒グラフは左から，ペーシングなし（non-pacing），ギャップバイアス無しのペーシング（pacing），最適なパケット間ギャップ値によるペーシング（opt-pacing）による実行時間であり，最後は理想実行時間（Ideal）である．この図から，ノード数が 16×16 まで増加しても，最適なパケット間ギャップ値を用いたペーシングは，ペーシングを利用しない場合と比較して 2 倍以上の高速化を達成していることがわかる．また，最適なパケット間ギャップ値によるペーシングは，ギャップバイアス無しのペーシングと比較して，ノード数の増加に対する実行時間増加の割合が少ない．これは，ホップ数が増えたことによる通信混雑の増加をギャップバイアスによって抑制していると考えられる．

以上の結果から，A2AT に対するパケットペーシン

ⓒ 2011 Information Processing Society of Japan

表 4 実験 3 の評価パラメータ
Table 4 Evaluation parameters for exp.3

パラメータ	設定値
アルゴリズム	a2at
ノード数	9×9 (81)
メッセージサイズ	1MiB
パケット間ギャップ	（ホップ数 −1）＋ ギャップバイアス
ギャップバイアス	−3.0〜+3.0（ステップ：0.125）

グの有効性を確認した．一方，全体で一様のギャップバイアス値を与えているため，フェーズによってはパケット間ギャップが大きすぎるといった，最適な実行ができていない可能性がある．

4.4 実験 3：シーケンス毎のチューニング

本実験では，a2at-9x9 の実行をフェーズ毎に細分化し，各フェーズに最適なパケット間ギャップを与えて実行することで全体性能の向上を図る．

まず，A2AT が実行する一連のフェーズ群に対してシーケンス番号を与える．なお，一つのフェーズに複数のシーケンス番号が付与される場合もある．次に，ギャップバイアスを様々に変化させながら，各シーケンスを NSIM で単体シミュレーションする．これによって，各シーケンスを最適に実行するギャップバイアスが得られる．**表 4** に本実験に用いたパラメータを示す．そして，最適なギャップバイアスによる各シーケンスを連続して実行することで，a2at-9x9 全体の最適な実行とする．

4.4.1 実験 3：結果と考察

図 16 に，ギャップバイアスに対する各シーケンスの実行時間を示す．この図から，シーケンスを最適に実行するギャップバイアスは，シーケンス毎に異なることがわかる．また，同じフェーズの実行であっても，シーケンス番号が異なる．すなわち，ホップ数が異なるとギャップバイアスも違う結果となった．

ギャップバイアスが小さい場合はパケット間の送出間隔が狭いため，シーケンスの実行は遅い．ギャップバイアスが大きくなるにつれてリンク競合が減り，パケットが順調に流れ出すため，それぞれのシーケンスの実行は速くなる．しかし，さらにギャップバイアスを大きくすると逆に遅くなり出していることがわかる．

第 4.3.1 節では，a2at-9x9 の実行を最適化するギャップバイアス値は 1.250 であった．これは，1.250 における全てのシーケンスの実行時間の総和が最も小さくなることに結論できる．

次に，フェーズ毎に最適なパケット間ギャップ値を与えた a2at-9x9 のシミュレーションを行った．その際のリンクスループットの推移を**図 17** に示す．実行

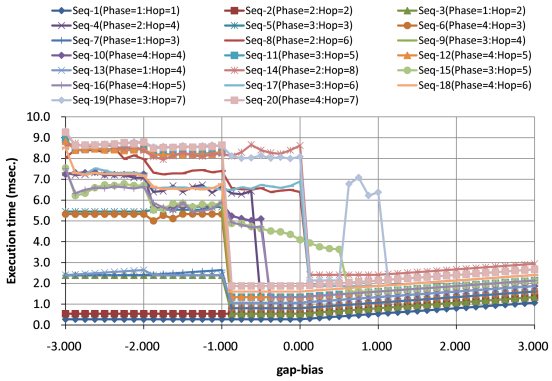

図 16 ギャップバイアスに対する各シーケンスの実行時間
Fig. 16 Execution time of sequence for each gap-bias

時間は 25.8ms であり，理想実行時間の 24ms に対して 107.5%と非常に高い実行性能を達成している．また，この図における各シーケンスの実行では，5 つのシーケンス（第 8，14，15，17，19 シーケンス）以外はリンクバンド幅を 100%利用しており，A2AT アルゴリズムの性能の高さを示している．この 5 つのシーケンスについて単体性能を確認したところ，それぞれの最大リンクスループットはすべて 3.5GB/s 前後となっており，第 15 シーケンス以外の 4 つの通信性能は順当である事がわかった．

そこで，第 15 シーケンスについて通信履歴や Jumpshot を用いた解析を行ったところ，第 14 シーケンス終了時刻のばらつき（以下，インバランス）が大きいことが原因であることが確認された．具体的には，第 14 シーケンスは他のシーケンスと比べてホップ数が 8 と最も大きく，各プロセスにおけるシーケンスの終了時刻に大きな差があった．そこで，すでに全ての受信を完了したプロセスは次の第 15 シーケンスを実行し，そのフェーズが発するパケットによって他のプロセスの第 14 シーケンスの通信を阻害していた．したがって，図 17 中，16〜17ms 付近の第 15 シーケンスの実行は，第 14 シーケンスと一部オーバラップしており，その結果として，リンクスループットは 3GB/s 付近まで低下している．

以上の結果から，A2AT は，各シーケンスの実行に最適なパケット間ギャップを与えることで，高い性能を発揮できることを確認した．一方，フェーズ実行終了時のインバランスが次のフェーズの実行に影響を与えることが明らかになった．この問題は，ノード数が大きくなると顕著に表れると考えられる．

4.5 実験 4：バリア同期によるインバランスの抑制

実験 3 では，第 14 シーケンスの終了時刻のインバランスによって，次のシーケンスの実行を妨げること

ⓒ 2011 Information Processing Society of Japan

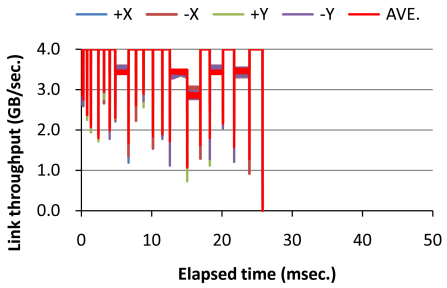

図 17 最適シーケンス実行におけるリンクスループット（実行時間：25.8ms）
Fig. 17 Link throughput of optimized sequence execution

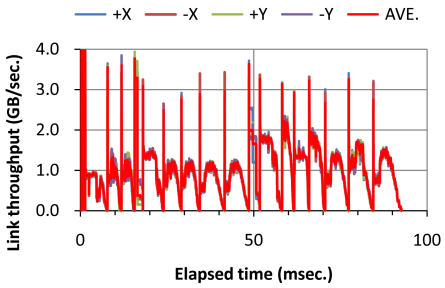

図 18 バリア同期を併用した最適シーケンス実行におけるリンクスループット（実行時間：92.6ms）
Fig. 18 Link throughput of optimized sequence execution with barrier syncronization

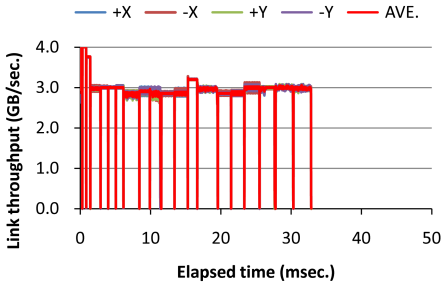

図 19 バリア同期を併用した最適シーケンス実行のリンクスループット（実行時間：32.8ms）
Fig. 19 Link throughput of optimized sequence execution with barrier syncronization

がわかった．そこで，本実験では，各シーケンスの終了後にバリア同期を挿入し，インバランスを抑制することで実行性能の向上を図る．

評価パラメータは実験 3 の表 4 と同じである．また，各シーケンスのギャップバイアスは実験 3 で取得したものを用いた．バリア同期には，ソフトウェア実装による dissemination アルゴリズムを使用した．このバリア同期のコストを事前に調査した結果，無負荷状態では 11.5us を要した．すなわち，a2at-9x9 での 20 シーケンス実行時には，218.5us のオーバヘッドとなるが全体の実行時間と比較すると無視できる．一方，本バリア同期によるインバランスも発生する．そこで，様々なインバランス環境下でバリア同期を実行し，その後のインバランス量を調査した結果，全て 3us 未満に収まることがわかった．

4.5.1 実験 4：結果と考察

最適なシーケンス実行にバリア同期を併せた結果を**図 18** に示す．実行時間は実験 3 における 25.8ms から 92.6ms へ大幅に悪化した．また，バリア同期による効果は見られず，A2AT の実行を阻害していることがわかる．ここで，最初の 3 つのシーケンスについては，ホップ数が 1 か 2 であるため，バリア同期によるインバランスの影響を受けていない．一方，残りの 17 個のシーケンスは 3 ホップ以上あり，すべてリンクスループットが不安定になっている．すなわち，バリア同期後の 3us のインバランスによって各フェーズの実行が崩れ，性能を十分に発揮できていないと考えられる．そこで，インバランスへの耐性を高めるために，インバランス環境下における各シーケンスの最適なギャップバイアス値を調査した．具体的には，インバランス量が 3us の状態を作り，その後最適なギャップバイアスを求めた．その結果，全シーケンスでのギャップバイアスは概ね増加し，最大値はホップ数が最も大きい第 14 シーケンスの 2.5 であった．

この調査結果に基づき，再度，a2at-9x9 を実行した結果を**図 19** に示す．この図から，バリア同期がもたらすインバランスの影響を抑制していることがわかる．また，実行時間，ならびにリンクスループットの平均値は，実験 3 の場合よりも低下しているが，実行時間は 32.8ms（理想実行時間の 136.7%）と，良好な性能を出しているといえる．しかし，ノード数が大きくなり，ホップ数が増えることによって，フェーズ実行後のインバランスが顕著になると考えられる．したがって，大規模システムでは，実験 3 のような最適化手法を採ることが難しく，バリア同期によるインバランスの抑制が効果的であると考える．

5. ま と め

本稿では，NSIM と呼ぶインターコネクトシミュレータを用いて，2 次元トーラス網での最適全対全通信アルゴリズムの一つである A2AT の基礎的な性能評価を行った．また，A2AT に対して積極的な（明示的な）パケットペーシングを適用し，その性能評価を行うとともに A2AT の性能の高さ，ならびにパケットペーシングの有効性を示した．そして，A2AT の各フェーズの実行にあわせたパケット間ギャップを設定することで，リンクバンド幅を十分に活用し，最大で

ⓒ 2011 Information Processing Society of Japan

理論実行時間の 107.5%で実行できることを確認した．以上の結果から，A2AT での通信は各フェーズ実行時刻のインバランスに敏感であることがわかった．したがって，実システムで利用するには OS ジッタの影響を考慮した評価も必要となる． 今後の課題は，大規模システムにおける評価，ならびに， 他の全対全通信や集団通信アルゴリズムなど，様々なアプリケーションについても評価を行う予定である．

謝辞 本研究を進めるにあたり，日頃からご協力頂く九州大学 村上和彰教授，青柳睦教授，南里豪志准教授に感謝の意を表す．なお，本研究の実験結果の一部は，九州大学情報基盤研究開発センターの研究用計算機システムを用いて取得したことを付記する．

参 考 文 献

1) Ajima, Y., Sumimoto, S. and Shimizu, T.: Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers, *Computer*, Vol. 42, No. 11, pp.36–40 (2009).
2) 高上治之，矢崎俊志，安島雄一郎，清水俊幸，石畑宏明：2 次元 Mesh ネットワーク・Torus ネットワーク上での最適全対全通信アルゴリズム，2010 年ハイパフォーマンスコンピューティングと計算科学シンポジウム (HPCS2010) 論文集，pp.83–90 (2010).
3) Aggarwal, A., Savage, A. and Anderson, T.: Understanding the Performance of TCP Pacing, *IEEE INFOCOM2000 Conference on Computer Communications*, pp. 1157–1165 (2000).
4) Kamezawa, H., M, N., Tamatsukuri, J., Aoshima, N., Inaba, M. and Hiraki, K.: Inter-Layer Coordination for Parallel TCP Streams on Long Fat Pipe Networks, *Proc. Intl. Conf. Supercomputing (SC2004)* (2004).
5) 高野了成，工藤知宏，児玉祐悦，松田元彦，石川裕，岡崎史裕：ギャップパケットを用いたソフトウェアによる精密ペーシング方式，情報処理学会論文誌 コンピューティングシステム， Vol.47, No.SIG7, pp.194–206 (2006).
6) Kumar, S., Sabharwal, Y., Garg, R. and Heidelberger, P.: Optimization of All-to-All Communication on the Blue Gene/L Supercomputer, *37th Int'l Conf. on Parallel Processing 2008, ICPP'08*, pp.320–329 (2008).
7) 三輪英樹，薄田竜太郎，柴村英智，平尾智也，眞木淳，稲富雄一，井上弘士，安島雄一郎，三吉郁夫，清水俊幸，安藤壽茂：NSIM:将来の大規模相互結合網を対象とした通信シミュレータの開発，情処研報，Vol.2010-HPC-125, No.5, pp.1–9 (2010).
8) 井上弘士，薄田竜太郎，安藤壽茂，石附茂，小松秀実，稲富雄一，本田宏明，山村周史，柴村英智， 于雲青，青柳睦，木村康則，村上和彰：大規模システム評価環境 PSI-SIM: 数千個のマルチコア・プロセッサを搭載したペタスケールコンピュータの性能予測，情処研報， Vol.2008, No.39, pp. 51–56 (2008).
9) Susukita, R., Ando, H., Aoyagi, M., Honda, H., Inadomi, Y., Inoue, K., Ishizuki, S., Kimura, Y., Komatsu, H., Kurokawa, M., Murakami, K. J., Shibamura, H., Yamamura, S. and Yu, Y.: Performance Prediction of Large-scale Parallel System and Application using Macro-level Simulation, *Proc. Intl. Conf. Supercomputing (SC2008)* (2008).
10) MPICH2: http://www.mcs.anl.gov/research/projects/mpich2/.

ⓒ 2011 Information Processing Society of Japan