Vol.2014-DPS-161 No.11
Vol.2014-EIP-65 No.11
2014/9/19

複数NICを連携制御する既存手法の評価

吉田 泰三[1]　山内 利宏[1]　谷口 秀夫[1]

概要：通信路の通信速度の向上により，複数のネットワークインタフェースカード（以降，NIC と略す）を使用し，1 台の計算機から多量のデータを送信できる手法が提案されている．このような場合，NIC 間の負荷分散をうまく行うことが求められる．本稿では，Linux で実現されている Ethernet チャネルボンディングの balance-rr モードと balance-tlb モードについて，NIC 間での負荷分散をうまく行っているか否かを示すために，両モードの通信性能を評価した結果を報告する．

キーワード：複数 NIC，Ethernet チャネルボンディング，負荷分散，オペレーティングシステム

1. はじめに

通信路の通信速度の向上に伴い，10Gigabit Ethernet や 100Gigabit Ethernet といった高速な通信機器が開発されている．Ethernet 以外にも，Myrinet[1] では 10Gbps，InfiniBand（× 4 QDR）[2] では 40Gbps といった高速通信が実現されており，いずれも HPC 分野で使用されている．また，現在も通信路高速化の研究開発は行われており，今後更なる通信速度の向上が予想される．しかし，高速な通信機器は，非常に高価である．10Gigabit Ethernet の環境構築を例に必要な経費を計算する．10Gbps のネットワークインタフェースカード（以降，NIC と略す）は 1 枚あたり 10 万円以上，10Gbps 対応の LAN ケーブルは 1 本 1 万円以上，10Gbps 対応のスイッチングハブは 100 万円以上の費用を要する．このため，高速な通信環境の構築は，容易とは言い難い．

そこで，廉価な複数の NIC を連携制御し，一台の計算機から多量のデータを送信できる手法 [3][4] が提案されている．廉価な NIC の例として，1Gbps NIC がある．1Gbps NIC は，一般に広く使用されており，環境構築に要する費用も合計で約数万円である．複数の NIC を連携制御する場合，NIC 間の負荷分散をうまく行うことが求められる．文献 [3][4] の手法では，複数の NIC を仮想的に一つに見せることにより，連携制御を実現している．複数の NIC を仮想的に一つに見せる手法は，大きくラウンドロビン分散方式と負荷分散方式に分類できる．ラウンドロビン分散方式は，送信パケットをラウンドロビンで NIC に振り分ける．また，負荷分散方式は，送信負荷情報に基づき，NIC 負荷の均一化を行う．

本稿では，複数 NIC を連携制御する手法として Linux で実現されている Ethernet チャネルボンディング [4] の balance-rr モードと balance-tlb モードを対象とした定量的な評価結果について述べる．評価では，NIC 間の負荷分散をうまく行っているか否かを示すため，通信性能を明らかにする必要がある．このため，両モードにおいて，データ送信プロセスの数，搭載する NIC の枚数，および NIC の種類を変更し，TCP/IP 通信を行った場合のスループットと平均 CPU 使用率を示す．

[1] 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

2. 複数の NIC を仮想的に一つに見せる手法

2.1 分散方式

複数の NIC を連携制御し，1 台の計算機から多量のデータを送信する手法が提案されている．文献 [3][4] の手法は，複数の NIC を仮想的に一つに見せることにより，連携制御を実現している．複数の NIC を仮想的に一つに見せる手法は，大きく以下の二つに分類できる．

（1）ラウンドロビン分散方式

ラウンドロビン分散方式は，送信パケットをラウンドロビンで NIC に振り分ける．ラウンドロビン分散方式の実現例として，FreeBSD の Lagg 機能における Round-robin モード，Linux の Ethernet チャネルボンディングにおける balance-rr モードがある．

（2）負荷分散方式

負荷分散方式は，送信負荷情報に基づき，NIC 負荷を均一化する．負荷分散方式の実現例として，Linux の Ethernet チャネルボンディングにおける balance-tlb モードと

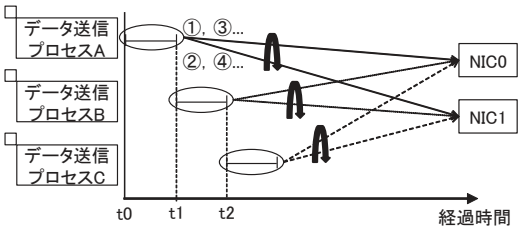


図 1 balance-rr モードにおける NIC 選定例

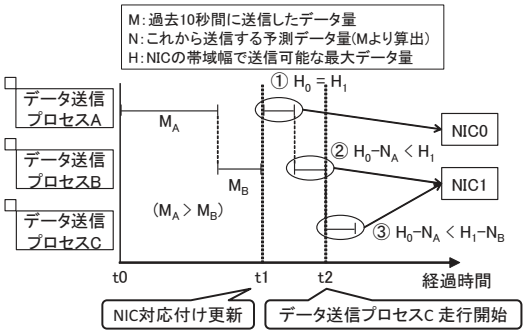


図 2 balance-tlb モードにおける NIC 選定例

balance-alb モードがある．

Linux の Ethernet チャネルボンディングは，両分散方式を実現している．このため，Ethernet チャネルボンディングの balance-rr モードと balance-tlb モードを対象とし，両方式の説明と比較を行う．

2.2 NIC 選定方法

2.2.1 balance-rr モードにおける NIC 選定方法

balance-rr モードにおける NIC 選定例を**図 1** に示す．send() システムコール（以降，send() と略す）により送信されるデータは，OS 内の Maximum Segment Size（1,460 Bytes）ごとに区切られた送信バッファに格納される．また，ヘッダに IP アドレスやポート番号などが格納された後，パケットは送信される．balance-rr モードは，送信データをパケット単位で送信側計算機上に搭載されている NIC へラウンドロビンで振り分ける．図 1 の場合，t0 からデータ送信プロセス A が走行しており，パケットが NIC0 と NIC1 に対して，順番に振り分けられている．また，t1 と t2 から走行するデータ送信プロセス B と C についても，同様にパケットが振り分けられる．

2.2.2 balance-tlb モードにおける NIC 選定方法

balance-tlb モードにおける NIC 選定例を**図 2** に示す．balance-tlb モードは，データ送信プロセスを最も余力のある NIC に対応付ける．対応付けの後，データ送信プロセスが「これから送信する予測データ量（N）」を対応付けられた NIC の余力から引く．なお，「これから送信する予測データ量（N）」は，データ送信プロセスが「過去 10 秒間に送信したデータ量（M）」から 1 秒間あたりに送信したデータ量を算出したものである．

また，10 秒に 1 回，データ送信プロセスと NIC の対応付けを解除し，現在の負荷に応じて，再対応付けを行う．図 2 に示したデータ送信プロセスと NIC の対応付けの順番を以下に示す．なお，図 2 において，NIC の帯域幅で送信可能な最大データ量を H としている．

（1）t1 でデータ送信プロセス A と NIC の対応付けを行う．このとき，両 NIC の余力はともに最大のため，若番である NIC0 に対応付ける．次に，NIC0 の余力からデータ送信プロセス A がこれから送信する予測データ量 N_A を減算する．これにより，NIC0 の余力は，$H_0 - N_A$ となる．

（2）t1 でデータ送信プロセス B と NIC の対応付けを行う．このとき，「NIC0 の余力 < NIC1 の余力」，つまり「$H_0 - N_A < H_1$」であるため，NIC1 に対応付ける．次に，NIC1 の余力からデータ送信プロセス B がこれから送信する予測データ量 N_B を減算する．これにより，NIC1 の余力は，$H_1 - N_B$ となる．

（3）t2 でデータ送信プロセス C からのデータ送信要求が発生したため，NIC との対応付けを行う．このとき，$N_A > N_B$ より，「NIC0 の余力 < NIC1 の余力」，つまり「$H_0 - N_A < H_1 - N_B$」であるため，NIC1 に対応付ける．次に，NIC1 の余力からデータ送信プロセス C がこれから送信する予測データ量を減算する．しかし，データ送信プロセス C は，過去 10 秒間にデータを送信していない．この場合，NIC1 の余力から 1 Byte を減算する．したがって，NIC1 の余力は，$H_1 - N_B - 1$ となる．

3. データ連送時の送信性能

3.1 評価内容

2.2.1 項で述べたように，send() により送信されるデータは，OS 内の送信バッファに格納された後，パケットとして送信される．実際の通信では，画像や動画をはじめとして，多くのデータを送信する場合がある．

本章では，データを連続して送信する場合の送信性能について述べる．具体的には，TCP/IP 通信において，連続してデータを送信した場合の send() の処理時間を測定し，送信性能を明らかにする．

3.2 評価方法

評価では，送信側計算機上でデータ送信プロセス，受信側計算機上でデータ受信プロセスを走行させた．データ送信プロセスは，send() により対応するデータ受信プロセスへ 1 KB のデータを 100 万回送信する．また，各 send() の前後でタイムスタンプを取得する．一方，データ受信プロセスは，recv() システムコールにより，データを連続して受信する．

データ送信プロセスの走行終了後，send() の前後で取得したタイムスタンプの差分から send()1 回あたりの処理時間を計算した．なお，計算結果より，各 send() は，周期性

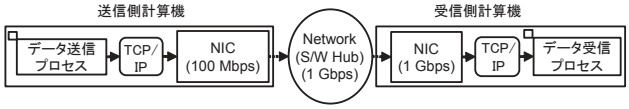


図 3 評価環境（データ送信プロセス:NIC=1:1，100Mbps NIC）

表 1 計算機性能

計算機	CPU	メモリ	NIC
送信側	Pentium 4 (2.4GHz)	1,024 MB	DE500-BA（100Mbps）
			Intel PRO 1000MT
受信側	Celeron D (2.8GHz)	256 MB	Intel PRO 1000GT

をもって発行されていたため，中盤の 5,000 回（50 万 1 回目から 50 万 5,000 回目まで）のみを抜粋し，評価の対象とした．

3.3 評価環境

評価環境（データ送信プロセス:NIC=1:1, 100Mbps NIC）を**図 3** に示す．また，計算機性能を**表 1** に示す．送信側計算機には，100Mbps NIC（DE500-BA）を搭載し，受信側計算機には，1Gbps NIC（Intel PRO 1000GT）を搭載した．送信側計算機と受信側計算機は，LAN ケーブルとスイッチングハブ（Allied Telesis GS924L V2，1000BASE-T 対応）を介して接続した．なお，各計算機の OS は，いずれも Linux 2.6.35 とした．

3.4 結果と考察

各 send() 発行時のタイムスタンプを**図 4** の（A）に示す．図 4 の（A）より，周期的に send() の処理時間が長大化していることがわかる．破線部に注目すると，328 回連続して send() を発行した後，send() の処理時間が長大化していることがわかる．

中盤 5,000 回のうち 188 回目から 515 回目を抜粋した場合の send() の処理時間を図 4 の（B）に示す．図 4 の（B）より，周期的に send() の処理時間が少し長くなっていることがわかる．破線部に注目すると，25 回連続して send() を発行した後，send() の処理時間が少し長くなっていることがわかる．

中盤 5,000 回のうち 224 回目から 248 回目を抜粋した場合の send() の処理時間を図 4 の（C）に示す．図 4 の（C）より，25 回の send() は，1 回あたりの処理時間の変動が小さいことがわかる．

図 4 の（A），（B），および（C）より，send()1 回あたりの処理時間は，大きく三つに分けることができる．

ここで，send() の処理流れを**図 5** に示す．TCP/IP 通信において，データを連続して送信する場合，大きく以下の三つの処理が行われる．

（1）送信するデータをユーザ空間から OS 内の送信バッファへコピー

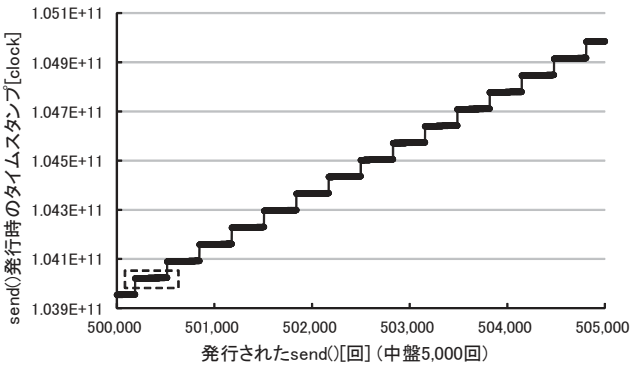
（2）送信バッファ内のデータを NIC のデバイスドライバ内のリングバッファへ格納

（3）NIC からパケットを送信

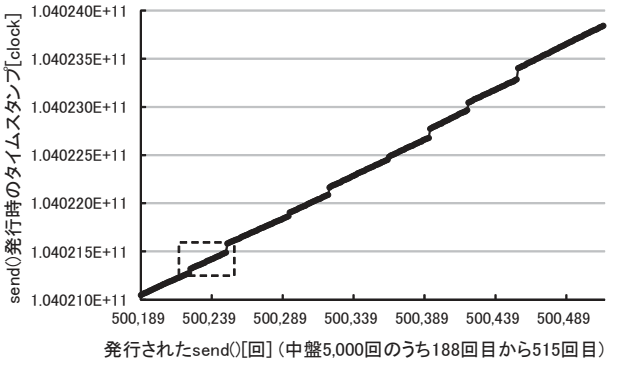
send()1 回あたりの処理時間は，上記三つの処理と対応していると考えられる．send()1 回あたりの処理時間を**表 2** に示す．

図 4 の（A）で発行された処理時間の長大化した send() は，（3）NIC からリングバッファ内のパケットを送信する際の処理時間を含む send() だと考えられる．また，NIC からのパケット送信に要する処理時間は，約 27.40ms であった．

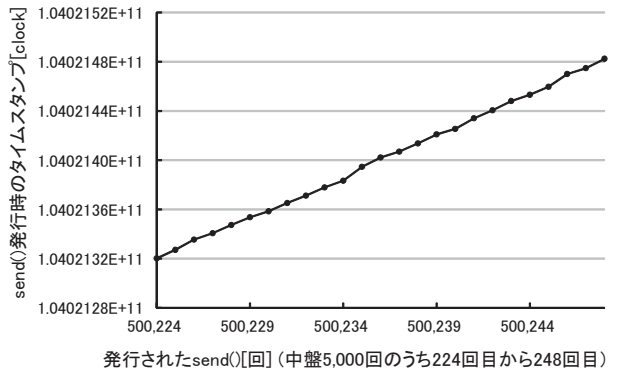
次に，図 4 の（B）で発行された少し処理時間の長い



（A）中盤 5,000 回分の send()



（B）中盤 5,000 回のうち 188 回目から 515 回目



（C）中盤 5,000 回のうち 224 回目から 248 回目

図 4 各 send() 発行時のタイムスタンプ

表 2 send()1 回あたりの処理時間

分類	最大値 [ms]	最小値 [ms]	中央値 [ms]	平均値 [ms]	分散	標準偏差
(1) NIC からパケットを送信	27.48	27.27	27.45	27.40	7.03×10^{-9}	8.39×10^{-5}
(2) パケットを NIC のデバイスドライバ内のリングバッファへ格納	0.094	0.0060	0.028	0.027	2.77×10^{-10}	1.67×10^{-5}
(3) 送信データを OS 内の送信バッファへコピー	0.0059	0.0017	0.0027	0.0029	7.48×10^{-13}	8.65×10^{-7}

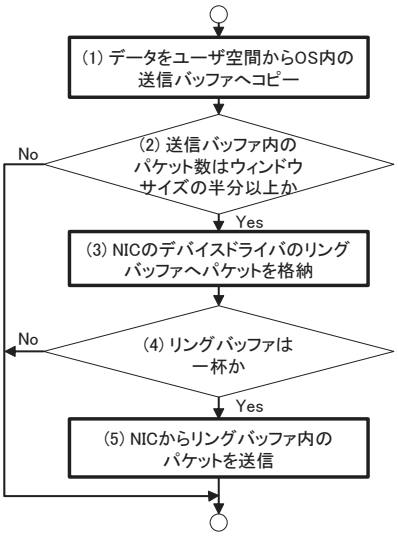


図 5 send() の処理流れ

send() は，(3) の処理を含まず，(2) 送信バッファ内のパケットを NIC のデバイスドライバのリングバッファへ格納する際の処理時間を含む send() と考えられる．リングバッファへのコピーは，リングバッファが一杯になるまで繰り返される．また，リングバッファへのコピーに要する処理時間は，約 0.027ms であった．

最後に，図 4 の（C）で連続して発行された処理時間の変動が小さい send() は，(2) と (3) の処理を含まず，(1) 送信するデータをカーネル空間から OS 内の送信バッファへコピーする際の処理時間を含む send() と考えられる．送信バッファへのコピーは，送信バッファ内のパケット数がウィンドウサイズの半分以上になるまで繰り返される．また，送信バッファへのコピーに要する処理時間は，約 0.0029ms であった．

以上より，連続してデータ送信を行った場合は，上記三つの処理時間を含む send() を周期的に発行することがわかる．

4. balance-rr モードと balance-tlb モードの性能比較

4.1 評価内容

balance-rr モードと balance-tlb モードの NIC 選定方式の違いによる性能差を示すため，以下の評価を行った．

(1) スループット
(2) 平均 CPU 使用率

balance-tlb モードは，10 秒ごとにデータ送信プロセスと NIC の対応付けを更新する．このため，時間経過によるスループットの変化を評価した．また，通信では，NIC の I/O 処理がボトルネックになる場合と CPU 処理がボトルネックになる場合がある．このため，スループットとともに平均 CPU 使用率も評価した．

4.2 評価の組合せ

評価は，以下の要素の組合せを変更し，行った．

(1) データ送信プロセスの数
(2) NIC の枚数
(3) 送信側計算機上の NIC の種類（転送速度）

4.1 節で述べたように，NIC の I/O 処理がボトルネックになる場合と CPU 処理がボトルネックになる場合を想定し，転送速度の異なる 100Mbps NIC と 1Gbps NIC を使用した．NIC の基本性能を把握するため，モードを設定せず，データ送信プロセスと NIC 枚数が 1 対 1 の場合において，100Mbps NIC と 1Gbps NIC を 1 枚ずつ使用し，スループットと平均 CPU 使用率を評価した．

次に，balance-rr モードと balance-tlb モード設定時におけるスループットと平均 CPU 使用率を評価した．NIC は，100Mbps NIC と 1Gbps NIC を使用した．また，両モードにおいて，複数の NIC を制御する上でデータ送信プロセス数と NIC 枚数は，重要な要素となる．このため，100Mbps NIC を使用する場合と 1Gbps NIC を使用する場合において，データ送信プロセス数と NIC 枚数が 2 対 4，3 対 3，および 4 対 2 の合計六つの組合せで評価した．

4.3 評価方法

評価では，3.2 節で述べたデータ送信プロセスとデータ受信プロセスに加え，親プロセスを使用した．親プロセスは，fork() システムコールにより，データ送信プロセスを複数走行させる．

親プロセスを走行させると，データ送信プロセスごとにタイムスタンプを取得できる．取得したタイムスタンプをもとに，各データ送信プロセスの send() 発行時のタイムスタンプを時系列順にまとめる．次に，最初の send() が発行された時刻を 0 秒とし，10 秒単位でスループットを計算する．一方，平均 CPU 使用率は，複数のデータ送信プロセスが同時に走行している区間を対象に取得した．

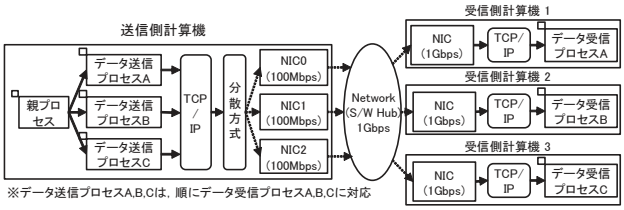


図 6 評価環境（データ送信プロセス:NIC=3:3，100Mbps NIC）

なお，各組合せにおいて，複数のデータ送信プロセスが同時に一定時間以上走行する状況を作るため，データ送信プロセスの send() 発行回数を変更した．100Mbps NIC を使用する場合は 100 万回とし，1Gbps NIC を使用する場合は 1,000 万回とした．ただし，データ送信プロセスと NIC が 2 対 4，かつ 100Mbps NIC を使用する場合は，250 万回とした．

4.4 評価環境

評価環境 (データ送信プロセス:NIC=3:3, 100Mbps NIC) を**図 6** に示す．図 6 は，送信側計算機上に 3 枚の 100Mbps NIC を搭載し，受信側計算機と 1 対 1 対応するデータ送信プロセスを 3 つ走行させたときの様子である．なお，データ送信プロセスの数，NIC の枚数，および NIC の種類は，組合せにより変化する．また，データ送信プロセスの数に合わせて，受信側計算機の台数も変化する．

計算機性能は，表 1 と同様である．送信側計算機と受信側計算機は，3.3 節と同様に，LAN ケーブルとスイッチングハブ（Allied Telesis GS924L V2，1000BASE-T 対応）を介して接続した．各計算機の OS も，3.3 節と同様に，Linux 2.6.35 とした．

4.5 基本性能

データ送信プロセス 1 つと NIC1 枚を使用した場合のスループットを**図 7**（A），平均 CPU 使用率を（B）に示す．

100Mbps NIC を使用した場合，スループットは約 94Mbps であり，平均 CPU 使用率は 4.6%であった．NIC の帯域の 94%以上を使用しており，かつ平均 CPU 使用率が低いことから，NIC の I/O 処理がボトルネックになっていることがわかる．

また，1Gbps NIC を使用した場合，スループットは約 784Mbps であり，平均 CPU 使用率は 29.9%であった．NIC の帯域を 78%しか使用できなかった原因は，CPU 以外のハードウェア性能であると考えられる．

4.6 性能比較

balance-rr モードと balance-tlb モード設定時におけるスループットを**図 8** に示す．また，balance-rr モードと balance-tlb モード設定時における平均 CPU 使用率を**図 9** に示す．

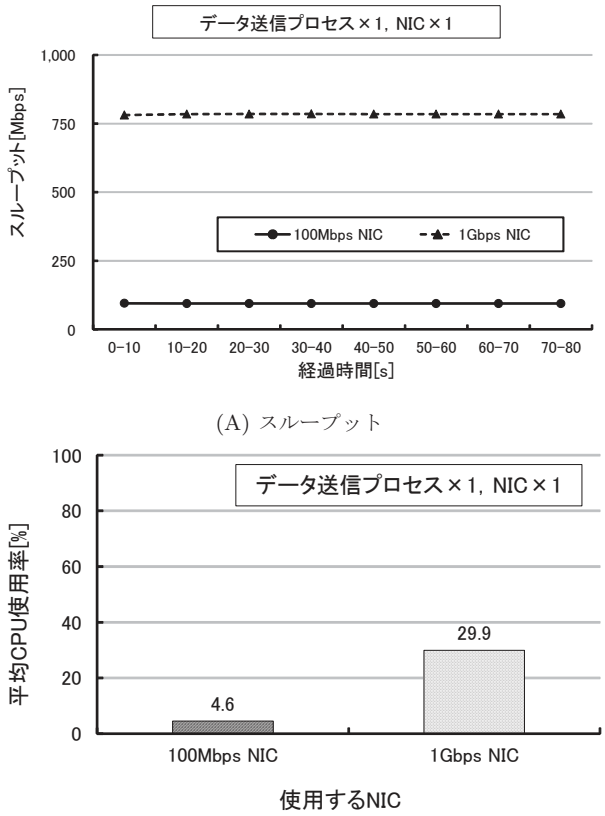


図 7 100Mbps NIC と 1Gbps NIC を使用した場合の基本性能

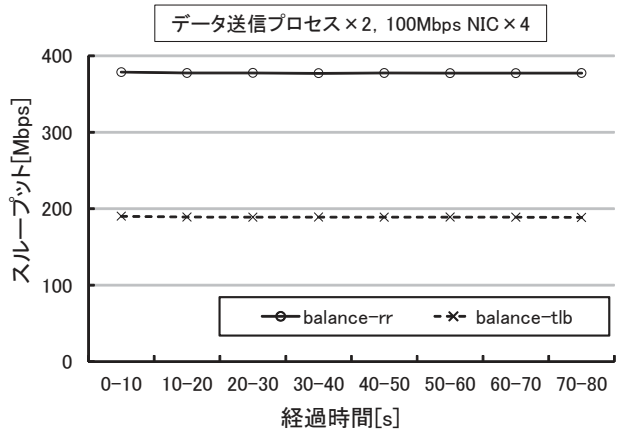
複数の 100Mbps NIC を使用した場合について，測定結果を示す．

（A1）2-4-100 と図 9 より，100Mbps NIC を使用した場合の balance-rr モードのスループットは約 377Mbps であり，平均 CPU 使用率は 75.3%であった．また，balance-tlb モードのスループットは約 188Mbps であり，平均 CPU 使用率は 11.6%であった．balance-rr モードは，balance-tlb モードと比較して，スループットが約 189Mbps 高く，平均 CPU 使用率が 67.0%高かった．

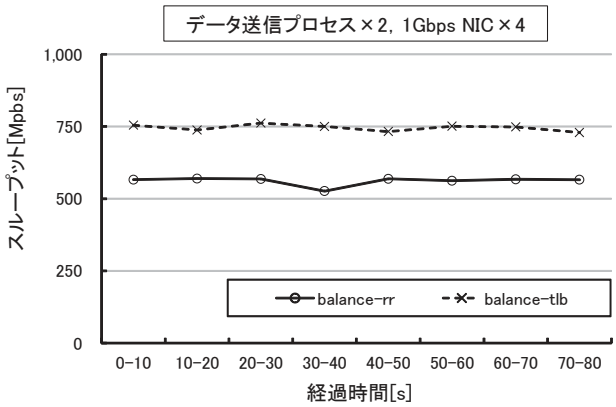
（B1）3-3-100 と図 9 より，100Mbps NIC を使用した場合の balance-rr モードのスループットは約 282Mbps であり，平均 CPU 使用率は 34.0%であった．また，balance-tlb モードのスループットは約 283Mbps であり，平均 CPU 使用率は 21.9%であった．balance-rr モードは，balance-tlb モードと比較して，スループットが同程度であり，平均 CPU 使用率が 12.1%高かった．

（C1）4-2-100 と図 9 より，100Mbps NIC を使用した場合の balance-rr モードのスループットは約 188Mbps であり，平均 CPU 使用率は 12.8%であった．また，balance-tlb モードのスループットは約 188Mbps であり，平均 CPU 使用率は 11.6%であった．balance-rr モードは，balance-tlb モードと比較して，スループットが同程度であり，平均 CPU 使用率が 1.2%高かった．

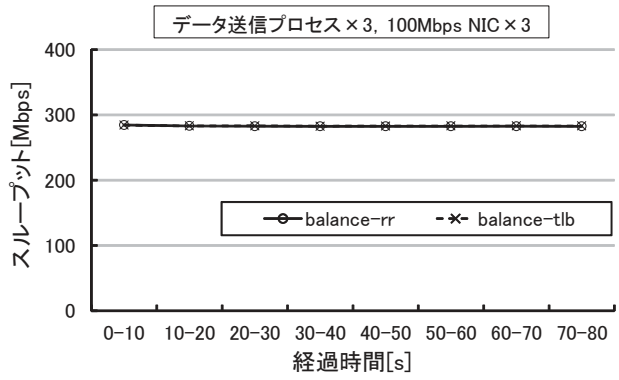
（A1）2-4-100，（B1）3-3-100，および（C1）4-2-100 よ



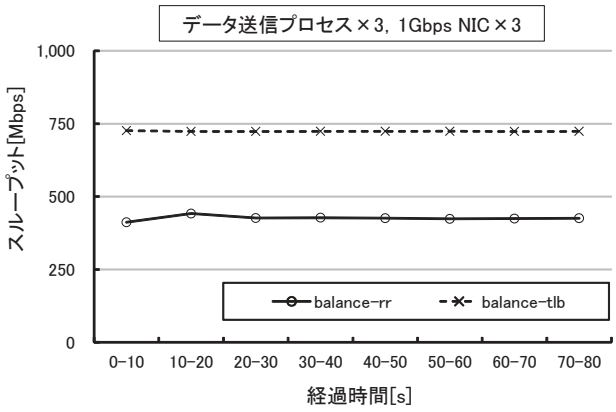
（A1）2-4-100



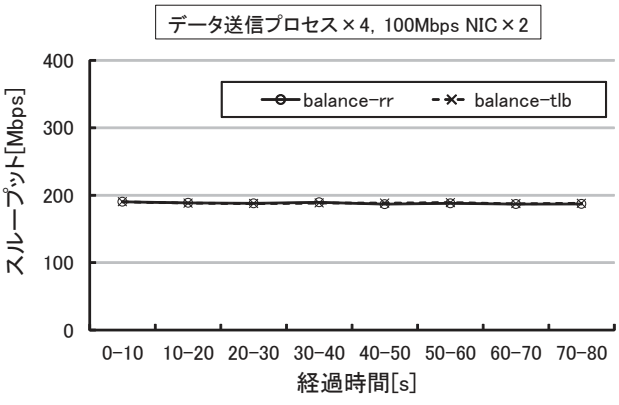
（A2）2-4-1000



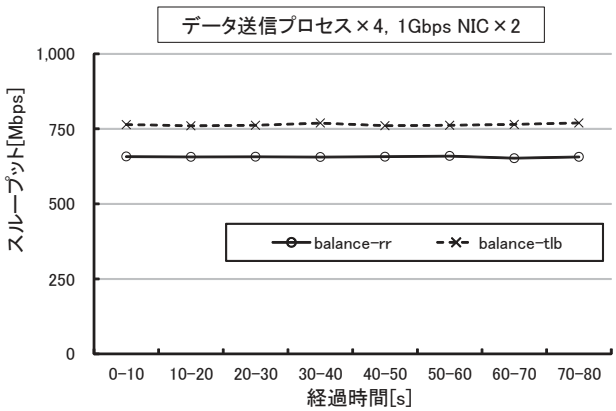
（B1）3-3-100



（B2）3-3-1000



（C1）4-2-100



（C2）4-2-1000

図 8 balance-rr モードと balance-tlb モード設定時におけるスループット
（データ送信プロセス数-NIC 枚数-NIC 転送速度）

り，以下のことがわかる．

（1）複数の 100Mbps NIC を使用した場合，CPU 処理ではなく，NIC の I/O 処理がボトルネックとなる．このため，balance-rr モードと balance-tlb モードのスループットに差が生じない．なお，NIC の枚数がデータ送信プロセスの数より多い場合，balance-rr モードは全 NIC の帯域を使用できる．一方，balance-tlb モードは，データ送信プロセスと対応付けられた NIC しか使用できない．つまり，（A1）2-4-100 の場合は，NIC2 枚分の帯域しか使用できず，全 NIC の帯域を活かせない．

（2）balance-rr モードの平均 CPU 使用率は，balance-tlb モードの平均 CPU 使用率より高い．これは，balance-rr モードでは，balance-tlb モードより，ソフト割り込みの処理時間が長くなっているためである．この原因は，balance-rr モードでは，複数の NIC から各プロセスに送信完了割り込みが発行されるからであると推察できる．一方，balance-tlb

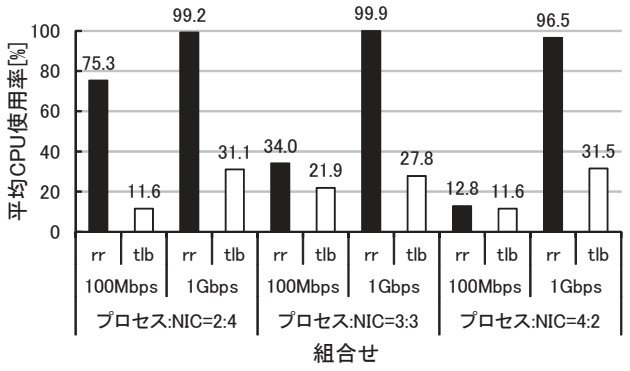


図 9 balance-rr モードと balance-tlb モード設定時における平均 CPU 使用率

モードは，データ送信プロセスと対応付けられた NIC のみからしか送信完了割り込みが発行されず，ソフト割り込みに要する処理時間が balance-rr モードと比べて短い．

次に，複数の 1Gbps NIC を使用した場合について，測定結果を示す．

（A2）2-4-1000 と図 9 より，1Gbps NIC を使用した場合の balance-rr モードのスループットは約 561Mbps であり，平均 CPU 使用率は 99.2%であった．また，balance-tlb モードのスループットは約 745Mbps であり，平均 CPU 使用率は 31.1%であった．balance-rr モードは，balance-tlb モードと比較して，スループットが約 184Mbps 低く，平均 CPU 使用率が 68.8%高かった．

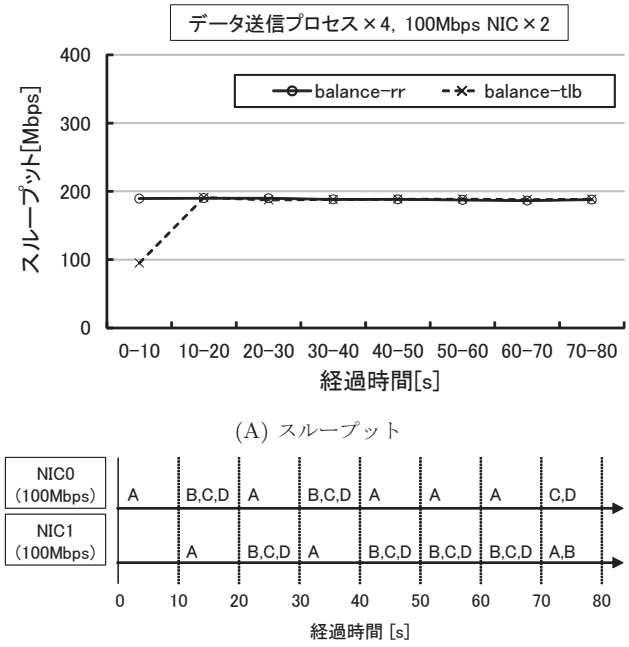
（B2）3-3-1000 と図 9 より，1Gbps NIC を使用した場合の balance-rr モードのスループットは約 425Mbps であり，平均 CPU 使用率は 99.9%であった．また，balance-tlb モードのスループットは約 724Mbps であり，平均 CPU 使用率は 27.8%であった．balance-rr モードは，balance-tlb モードと比較して，スループットが約 299Mbps 低く，平均 CPU 使用率が 72.1%高かった．

（C2）4-2-1000 と図 9 より，1Gbps NIC を使用した場合の balance-rr モードのスループットは約 656Mbps であり，平均 CPU 使用率は 96.5%であった．また，balance-tlb モードのスループットは約 764Mbps であり，平均 CPU 使用率は 31.5%であった．balance-rr モードは，balance-tlb モードと比較して，スループットが約 108Mbps 低く，平均 CPU 使用率が 65.0%高かった．

（A2）2-4-1000，（B2）3-3-1000，および（C2）4-2-1000 より，以下のことがわかる．

（3）複数の 1Gbps NIC を使用した場合，balance-rr モードでは，NIC の I/O 処理ではなく，CPU 処理がボトルネックになる．このときの平均 CPU 使用率は，100%近くになった．balance-rr モードは，ラウンドロビンで複数の NIC を使用する．これにより，ソフト割り込みに要する処理時間が長くなるためである．

（4）balance-rr モードのスループットは，balance-tlb モードのスループットより低い傾向にある．balance-rr モードは，balance-tlb モードと比較して，ソフト割り込みの CPU 使用率が増えており，これがスループット低下の原因の一つと考えられる．なお，原因の詳細な分析は，今後の課題とする．



(A) スループット

(B) データ送信プロセスと NIC の対応付け

図 10 balance-tlb モードにおいて NIC 負荷に偏りが生じた場合

（5）balance-rr モードと balance-tlb モードは，複数の 1Gbps NIC を使用しているにも関わらず，スループットが NIC1 枚分の帯域未満である．これは，4.5 節で述べたように，CPU 以外のハードウェア性能がボトルネックになっているためだと考えられる．

また，（C1）4-2-100 と（C2）4-2-1000 において，データ送信プロセスを複数の NIC にうまく分散できない場合を以下に示す．

（6）データ送信プロセス 4 つと NIC2 枚の場合，balance-tlb モードでは，更新周期のタイミングにより，データ送信プロセスのデータ送信量を反映できず，データ送信プロセスを複数の NIC にうまく分散できない場合がある．例として，**図 10** にデータ送信プロセス 4 つを 100Mbps NIC2 枚にうまく分散できない場合の実行結果を示す．先にデータ送信プロセス A を走行させ，10 秒後に残りのデータ送信プロセス B，C，および D を走行させる場合を考える．経過時間 10 秒の対応付けの更新において，データ送信プロセス A の送信したデータ量は，先に走行しているため，データ送信プロセス B，C，および D と比較して，十分大きい．一方，データ送信プロセス B，C，および D の送信したデータ量は，少量である．このため，NIC0 に 3 つのデータ送信プロセス（B，C，および D）が対応付けされる．図 10 の（A）では，スループットが NIC2 枚分の帯域

を活かせている．しかし，図 10 の（B）に示すように，10 秒から 70 秒の間，4 つのデータ送信プロセスを 2 枚の NIC に 2 つずつ分散できていない．

5. 関連研究

複数の NIC をラウンドロビン方式で使用する手法 [5][6] が提案されている．PM/Ethernet-kRMA[5] では，OS 内の送受信バッファを直接ユーザに提供することで，データ移動のオーバヘッドを削減し，高帯域幅を実現している．また，RI2N++[6] は，受信側計算機に到着したパケット数をもとに，各 NIC に割り当てるパケットの割合を変更している．これらの手法は，ラウンドロビン方式で発生する受信側計算機上でパケットの到着順序が入れ替わる問題に，オーバヘッドがあることを前提で対処している．

複数の NIC を制御する上で性能を向上させる手法 [7]–[10] が提案されている．MiAMI[7] は，割り込みの親和性を考慮する手法であり，マルチコア環境において，データ送信プロセスと強く結びついているコア上で I/O デバイスにより作成された割り込みを処理し，キャッシュ効率を向上させている．10-Gigabit Ethernet TOE[8] は，TCP/IP Offload Engine により，TCP/IP のプロトコル処理を NIC が行い，CPU 負荷やメモリアクセス負荷を削減している．Zero Copy Socket 方式 [9] は，NIC メモリをカーネルのパケットバッファの代替とすることにより，CPU 負荷の削減と高スループットを実現している．また，文献 [10] は，CPU が NIC のレジスタをポーリングした際，結果の返却までに一定の待ちを加えることにより，パケット到着直後のレイテンシを削減している．これらの手法を複数の NIC を連携制御する手法と併用することにより，更なる性能の向上を期待できる．

6. おわりに

Ethernet チャネルボンディングの balance-rr モードと balance-tlb モードを対象とし，NIC 間での負荷分散をうまく行っているか否かを示すため，TCP/IP 通信でのスループットと平均 CPU 使用率の評価結果について述べた．評価は，データ送信プロセス数，NIC 枚数，および NIC 種類の組合せを変更し，データを連続して送信した場合における TCP/IP 通信での性能を測定した．

評価を行うにあたり，100Mbps NIC を使用し，データを連続して送信した場合の send() の送信性能を明らかにした．データを連続して送信する際の send() は，リングバッファ内のパケットを NIC から送信する処理（27.40ms），送信するデータを NIC のデバイスドライバ内のリングバッファへ格納する処理（0.027ms），および送信するデータを OS 内の送信バッファへコピーする処理（0.0029ms）に分けることができる．これらの処理を周期的に繰り返すことにより，データを送信する．

両モードの性能比較の結果，100Mbps NIC を使用した場合，NIC の I/O 処理がボトルネックになるため，balance-rr モードと balance-tlb モードのスループットは，同程度であることを確認した．ただし，balance-rr モードは，ソフト割り込みに多くの処理時間を要するため，平均 CPU 使用率が balance-tlb モードより高い傾向があった．一方，複数の 1Gbps NIC を使用した場合，balance-rr モードでは，CPU 処理がボトルネックになった．また，balance-rr モードと balance-tlb モードでは，複数の 1Gbps NIC を使用しているにも関わらず，スループットが NIC1 枚分の帯域未満であった．これは，CPU 以外のハードウェア性能がボトルネックになっているためだと考えられる．加えて，balance-tlb モードは，対応付けの更新のタイミングにより，データ送信プロセスを複数の NIC へうまく分散できない場合があることも確認した．

謝辞 本研究の一部は，科学研究費補助金基盤研究 (B) (課題番号：24300008) による．

参考文献

[1] Myricom: Myrinet, 入手先 ⟨http://www.myri.com/⟩(参照 2014-08-06).
[2] InfiniBand Trade Association: InfiniBand, 入手先 ⟨http://www.infinibandta.org/⟩(参照 2014-08-06).
[3] The FreeBSD Project, 入手先 ⟨http://www.freebsd.org/cgi/man.cgi?query=lagg(4)⟩(参照 2014-08-06).
[4] Thomas Davis: Linux Channel Bonding, 入手先 ⟨http://sourceforge.net/projects/bonding/⟩(参照 2014-08-06).
[5] Shinji, S. and Kouichi, K.: PM/Ethernet-kRMA: A High Performance Remote Memory Access Facility Using Multiple Gigabit Ethernet Cards, *Proc. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid* (*CCGird 2003*), pp.326–333 (2003).
[6] 米元大我，塙 敏博，三浦信一，朴 泰祐，佐藤三久：トラフィック量に適応する非対称マルチリンク Ethernet トランキング，情報処理学会論文誌，Vol.3，No.1，pp.25–37 (2010).
[7] Jang, H.C. and Jin, H.W.: MiAMI: Multi-Core Aware Processor Affinity for TCP/IP over Multiple Network Interfaces, *Proc. 17th Annual Symposium on High Performance Interconnects* (*HotI'09*), pp.73–82 (2009).
[8] Feng, W., Balaji, P., Baron, C., Bhuyan, L. N. and Panda, D. K.: Performance Characterization of a 10-Gigabit Ethernet TOE, *Proc. 13th Symposium on High Performance Interconnects*, pp.58–63 (2005).
[9] 小林伸治，安島雄一郎，新家正総：NIC メモリを用いた Zero Copy Socket 方式の提案，電子情報通信学会技術研究報告 CPSY，Vol.105，No.225，pp.31–36 (2005).
[10] Flajslik, M. and Roseblum, M.: Network Interface Designe for Low Latency Request-Response Protocols, *Proc. 2013 USENIX Annual Technical Conference*, pp.333–345 (2013).