BSD UNIXにおけるover Gbpsネットワーク処理性能の向上

清水健司† 小倉毅† 川野哲生†
君山博之† 丸山充†

近年, 汎用 PC 等の安価なプラットホームを用いた高速ネットワークの実現手法に関する研究が盛んに行われており, それに伴い, 数 Gbps の高速ネットワークを活用したアプリケーションが開発されている. 本論文では, この様なアプリケーションへの適用を目標として, 汎用 PC 上で動作するサーバ OS の実装法に起因するネットワークスループットへの影響に着目しながら, ネットワークプロトコル処理の高速化手法に関して報告する. まずサーバ OS として広く用いられる BSD 系 OS(FreeBSD) と Linux をインストールしたシステムに, ギガビット超の高速ネットワークを実現する 10GbE 対応インターフェース, 及び OC-48c(2.4Gbps) インターフェースを搭載し, 一次評価を行い, その結果を元に FreeBSD におけるネットワークスループットの低下原因を解析した. 特に FreeBSD に特有のバッファ管理方式である mbuf 構造に着目し, 我々が提案し実装を進めている「適応型デバイスドライバ」を用いてスループットを改善する実装を行った. 「適応型デバイスドライバ」とは, ホストの持つリソースの状態に応じて動的に振舞いを変更するデバイスドライバである. 我々の実装した手法により, ショートパケットにおいて 70% のパケット処理性能の改善, およびロングパケットにおいて 10% のスループットの改善効果を確認する事ができた.

Enhancement of over-Gbps protocol processing using BSD UNIX

KENJI SHIMIZU ,† TSUYOSHI OGURA,† TETSUO KAWANO,†
HIROYUKI KIMIYAMA† and MITSURU MARUYAMA †

Recent R&D activities in high-speed networks using general-purpose PCs lead to the emergence of applications demanding over-Gbps network bandwidth. In this paper, we report how we enhanced the network protocol processing of the BSD UNIX by focusing on the difference in the implementation of server-oriented OSs, especially in over-Gbps networks. Firstly, we conducted the preliminary evaluation of the network throughput using two systems which we installed Linux and FreeBSD in combination with over-Gbps network interface cards. The cards include ones for 10-gigabit Ethernet and for OC-48c(2.4Gbps) POS(Packet over SONET/SDH). By the obtained result of the throughput comparison, we found the mbuf memory management scheme is the primary cause lowering the achievable throughput in FreeBSD.

We proposed adaptive device drivers, which change dynamically their behaviors according to the host-resource usage states, to work around the mbuf's drawback, resulting in 70 % packet processing gain in the short-packet transmission and 10 % throughput gain in the long-packet transmission.

1. はじめに

近年,IP 網を用いたアプリケーションとして普及し始めた高品質な映像ストリーミングサービスや, 広域ストレージサービスの登場と共に, 安価な汎用 PC を用いた高速ネットワークの実現手法に関する研究が盛んに行われる様になった.

例えば,iSCSI を用いた高速なストレージサービスの提供を目指して, 専用ハードウェアエンジンを搭載した高速ネットワークインターフェースの開発が進められている[1].

また, 我々が現在開発している IP 網を用いた非圧縮映像の配信技術である i-Visto は, 非圧縮 HDTV 映像のストリーミング配信を実現するために,1.5Gbps 以上の帯域に加えて,1 映像フレーム以内という低遅延性を必要とする, 最も要求条件の厳しいアプリケーションのひとつと言える[2].

この様なシステムは,UNIX 互換のサーバ OS として広く用いられる Linux や BSD 系 OS を使用しており, 専用のハードウェアと OS によるシステムと比較して柔軟性が高いという利点があるが, ソフトウェア, ハードウェアの制約から高い性能を引き出す為には様々な課題が残されている. 特に,BSD 系 OS は高い信頼性により評価を受けているが, 高速なネットワークを実現することが困難である事が知られいる[3].

本論文では, まず Linux,FreeBSD 双方の OS と数 Gbps のネットワークを実現する 2 種類のネットワークインターフェースカード (以下,NIC) を使用しスループットの一次評価実験を行った.NIC には, 我々の開発した OC-48c(2.4Gbps) 対応 NIC[4] と市販の Intel 製 10GbE 対応 NIC[5] を用いた.

評価の結果, パケットの送信スループットを比較した結果,FreeBSD ではジャンボパケットを用いた時に,Linux にくらべて大きくスループットが劣る事が確認できた.

本論文では, まず一次評価の結果を受けて, FreeBSD におけるプロトコル処理性能の低下原因の明確化を行い, そのスループットへの影響を定量的に測定する. そして, これらの要因を回避し, スループットを改善する為に「適応型デバイスドライバ」を提案し, その効果を実験により明らかにする.

1.1 ネットワーク高速化に関する研究

パケット受信性能を改善するべく導入された手法である NAPI[6] は, パケット受信を示すハードウェア割り込みを検知すると, 一時的に割り込みの発生を禁止し, カーネルによるポーリングでパケットを受信する. 本手法により CPU 負荷が軽減され受信性能が向上するが, ポーリングで受信するパケットの量や, ポーリングの周期の調整に課題が残っている.

NIC 上のハードウェアアシスト機能を用いた高速化手法である TSO(TCP Segmentation Offloading) は, 通常 OS のプロトコルスタックで行うパケットの分割処理を NIC 上で行うことによって, プロトコル処理にかかる負荷を軽減する手法で,GbE をはじめ多くの NIC で採用されている. ただし, アプリケーションによっては効果が上がらない事が確認されており[7], 更に高負荷なアプリケーションと高速なネットワークを用いた定量的評価が必要である.

昨年度達成された Internet2 Land Speed Record は,NIC によるパケット間ギャップ調整と,TCP マルチフローを活用している[8]. パケット間ギャップ調整により, パケット送信速度をネットワークボトルネック帯域以下に押え, パケットドロップの発生により TCP ウィンドウが小さくなるのを回避している. また,TCP マルチフローにより, パケット送信プロセスをパイプライン化し, 高速化を達成している.

このように様々な手法が考案されているが, 近年利用が可能となった 2.4Gbps や 10Gbps の回線速度をもつネットワークインターフェースと, 高速 CPU や PC アーキテクチャを組み

† NTT 未来ねっと研究所
NTT Network Innovation Laboratories

表 1 送受信に用いた PC の仕様

CPU	Xeon 2.0 GHz x 2
メモリ	512 Mbytes
マザーボード	Supermicro P4DL6 (FSB: 400 MHz)
PCI-X 拡張バス	64bit/66MHz for OC-48c 対応 NIC 64bit/133MHz for 10GbE 対応 NIC
OS: Linux system	Redhat Linux 9.0 and Linux kernel-2.4.21
OS: FreeBSD system	FreeBSD 4.8
ベンチマークソフトウェア	netperf-2.2pl4[12]

合わせた評価は十分に行われておらず☆1, 本論文で述べるギガビット超の領域で初めて顕在化する OS の実装の相違が与えるスループットへの影響の明確化は重要な課題であると考えられる.

2. FreeBSD と Linux を用いた送信スループットの一次評価

図 1 は Linux と FreeBSD で構築したシステムを用いたスループットの評価結果である. 評価は, 表 1 に示す仕様の PC を 2 台, それぞれ送信用, 受信用として接続し, ネットワークベンチマークソフトウェアを用いたスループット測定により実施した. 本測定には,UDP パケットを用いている. X 軸にメッセージサイズ (プロトコルヘッダをのぞいたパケットの大きさ),Y 軸にスループットをとりプロットした. ただし, それぞれ上図はスループットを単位時間あたりのパケット処理数 [kpps] で表し, 下図は単位時間あたりのデータ量 [Mbps] で表している.

10GbE を使用した図中左列では,Linux を用いた場合最大 4Gbps 程度のスループットが得られているが,FreeBSD では, 最大 2.5Gbps 程度でスループットが飽和している事が分かる. しかも,FreeBSD では,2kbyte までは Linux よりも高い値を示しているが, それ以上のパケットサイズを用いた途端に Linux との優劣が逆転し, 急激に差が拡がり始めている.

同様に, 図中右列においても,Linux を用いたシステムでは 9kbyte 以上のパケットを用いる事で,OC-48c の回線限界である 2.4Gbps の最大スループットを達成しているのに対して,FreeBSD では 1.9Gbps でスループットが飽和している. mbuf クラスタと呼ばれる FreeBSD のバッファサイズを 4kbyte に拡大するチューニングを施すことでスループットは 2.2Gbps まで改善するが, それでも Linux と比較して低いスループットにとどまっている☆2.

最大スループットに関する傾向は, どちらの NIC を用いた場合でも共通しており, 2kbyte を越えてパケットサイズを大きくすると,FreeBSD と Linux のスループットの差が拡がる現象が見られた. また, この傾向は従来のギガビットイーサネットでは検知できない速度領域で顕在化しており, ギガビット超の帯域に対応する NIC を用いた比較により始めて明らかとなる結果である.

また OC-48c 対応 NIC を用いた評価では, 単位時間あたりのパケット処理数を比較した場合, ショートパケットを用いた測定でも FreeBSD が Linux よりも低い値となっている. この現象は 10GbE 対応 NIC を用いた評価では現れない事から, デバイスドライバの実装方法が大きく影響していると考えられる.

2.1 一次評価結果の考察

一次評価の結果明らかになった, 最大スループットの差に関して考察する. OS の実装の相違のうち, ネットワーク性能に影響を与える要因として考えられるのは,

- タスクスケジューリングアルゴリズム
- I/O スケジューリングアルゴリズム
- プロトコルスタックの実装
- バッファ管理の実装

等が挙げられる.

ただし, スケジューリングアルゴリズムやプロトコルスタックへの修正は, 既存のソフトウェアとの共存性の観点から望ましくないと考えられる為, 本稿の検証対象からは外す事とする. 効果的な高速化手法の一つとして知られているゼロコピー通信に関しても, ユーザアプリケーションが実行するシステムコールの書き換えが必要という点で望ましくない☆3.

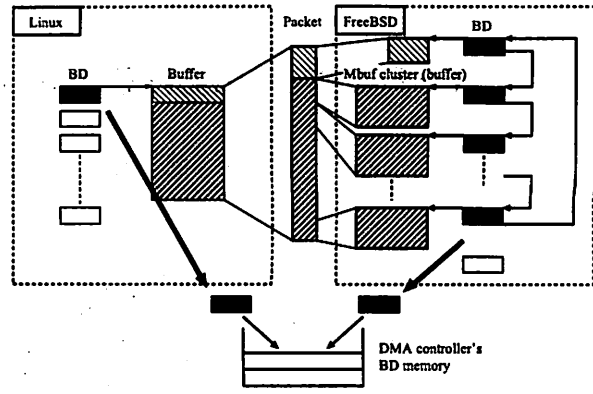


図 2 OS 間のバッファ管理手法の相違.

バッファ管理の実装に関する調査によると,BSD 系 OS で採用されている mbuf 構造を用いたバッファ管理手法では, パケットはヘッダ部分とコンテンツ部分の 2 つの領域に分離されて格納され, その後, パケットのコンテンツ部分が MINCLSIZE で定義されるサイズよりも大きい場合は☆4, コンテンツ部分が更に 2kbyte 毎に分割されて別々の領域に格納される. 本領域は,mbuf クラスタと呼ばれる.

一方 Linux では, 最大 128kbyte の連続アドレスを持つバッファ領域を確保する事ができ IPv4 パケットを分割する事なく単一の領域に格納する事ができる☆5. 本領域はスラブアロケータ[11] と呼ばれるキャッシュ機構により割り当てられる. また, パケットのヘッダとコンテンツは同一のバッファに格納されてデバイスドライバに渡される. これらを簡易に図示したものが図 2 である.

デバイスドライバには, 図中の斜線を引いたボックスのような形でパケットが渡される. また, 図中の黒い四角はバッファディスクリプタ (以下,BD) を表している.BD はデバイスドライバ内で管理され, バッファ領域の情報を格納している.BD の役割に関しては,5.1 節にて説明する.

この FreeBSD における mbuf 構造をベースとしたバッファ管理手法は, 以下に説明する点でネットワークスループットを低下させる原因となる可能性がある.

(1) プロトコルスタックにおける高い処理負荷
大きなパケットを 2kbyte 毎の複数のバッファで管理するため, Linux にくらべて, 全ての mbuf クラスタのリストを辿りながらプロトコル処理をおこなう必要があり高負荷である.

(2) ハードウェア割り込み負荷
一般的に,NIC は DMA 転送の完了をデバイスドライバに通知する為にハードウェア割り込みを用いる. デバイスドライバは, その割り込みハンドラとして, 転送済みのバッファを解放する関数を登録しておき, 割り込みの発生を検知すると同時に実行する. カーネルでは, そのような割り込みに起因する処理は高い優先度で実行されるる為, タスク切り換えの為のスケジューリングやコンテキストスイッチ等が実行され, 余計な CPU 処理が必要となる. 特に FreeBSD ではパケットが複数の mbuf クラスタに分割されるために, 発生するハードウェア割り込みの数が多くなり, 割り込み処理の負荷がより高く

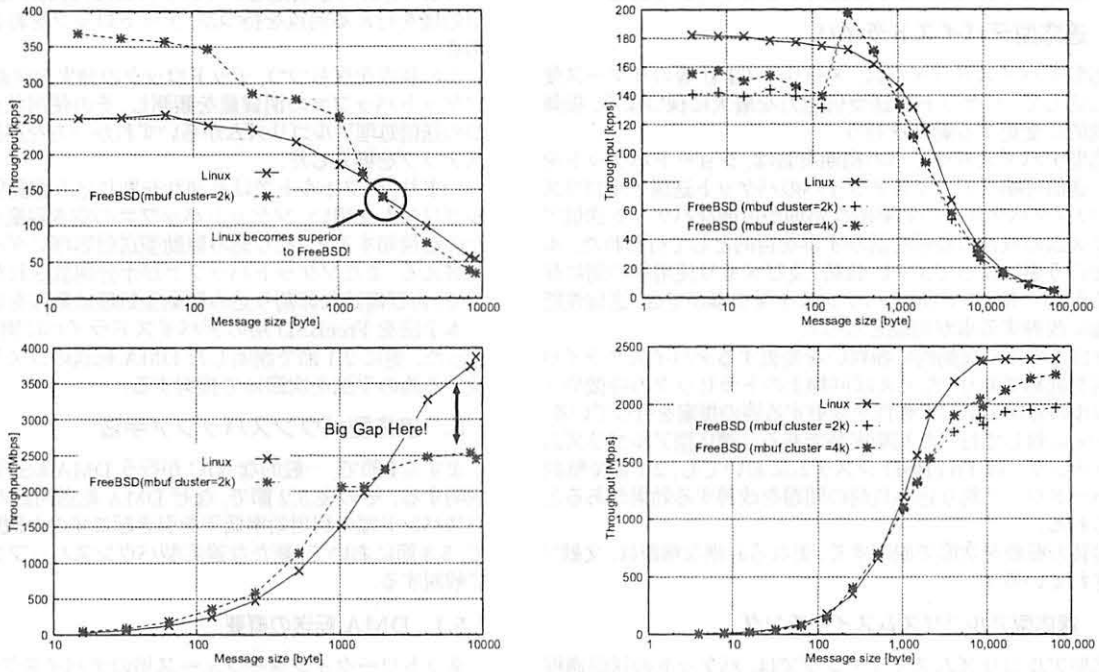
☆1 汎用 PC に搭載可能な 10Gbps 対応ネットワークインターフェースは, 我々が開発した文献[9] のもの以外は,10GbE にのみ対応しており, また市場にも数種類程しかない.

☆2 mbuf クラスタの大きさは,MCLSHIFT というパラメータによって, 4kbyte まで大きくする事が可能である

☆3 Linux,FreeBSD 双方のデバイスドライバを開発する上で, バッファのバイトアライメントの問題により FreeBSD のスループットが低下する事があった. アライメントのとれていないバッファの為に, 新たにメモリ領域を確保しなおして, そこにパケットをコピーするという余分な処理が必要となる為である.FreeBSD では, 1byte アライメントであるのに対して,Linux では 8byte アラインメントであった.

☆4 この値は,/usr/src/sys/sys/mbuf.h で定義されている. デフォルト値は 213byte

☆5 ゼロコピー通信等では, ページ毎にパケットのコンテンツを格納し, ページサイズより大きいパケットは, ページのリスト構造により管理される. この場合, 連続領域は 4kbyte となる.



(a)Intel PRO/10GbE LR Server Adapter を用いた評価　　(b)OC-48c 対応カードを用いた評価

図 1　ギガビット超の帯域対応 NIC を用いたスループットの一次評価.

なると考えられる.

(3)　PCI バスバンド幅の利用効率の低下
PC のメインメモリから NIC 上のメモリへ DMA 転送を実行する為に, デバイスドライバにより NIC 上の DMA コントローラのレジスタ操作や, 管理するバッファ情報を伝える為の BD の操作が PCI バスを介して行われる. この処理は,DMA 転送の前後におこなわれる為, 複数の mbuf クラスタを Scatter/Gather DMA 転送により個別に転送しなければならない FreeBSD では,PCI バスバンド幅の利用効率がより低下すると考えられる.

上記 (2) の割り込み負荷の問題は,Linux においても特にショートパケットの転送時に観測する事ができた. この問題を解決するために我々は Linux において,「適応型デバイスドライバ」というコンセプトのもと, あらたなデバイスドライバを実装し, 送信処理性能を改善している[4]. 本手法は,FreeBSD 用のデバイスドライバにも同様に適用可能であると考られる. 本手法は 4 節において説明する.

(3) の PCI バス利用効率低下の問題は 5.2 節で扱う. この問題は,FreeBSD 特有である為, 新たなスループットの改善手法を検討し, 実装を行う.

実装したデバイスドライバを用いて行ったスループットの評価結果を 6 節で示し, 解析を行う.

この様に, デバイスドライバのみの修正によりスループットを改善する事ができれば, カーネル本体を修正することによるコンパチビリティの問題を回避することができる. ただし, デバイスドライバの修正には, ハードウェアの詳細な情報が必要となるため, 一次評価で用いた市販の 10GbE 対応 NIC で効果を確認する事は困難である. よって, 最大スループットに関して同様な傾向を示した OC-48c 対応 NIC を対象として以降の評価を進める.

3. OC-48c PCI カードについて

本カード (図 3) は, 汎用 PC のマザーボードとしては一般的な, 64bit/66MHz の PCI 拡張スロットに対応し, 最大 2.4Gbps の広帯域通信に対応する.

ハードウェアの構成を図 4 に示す. 各部の機能は以下の通りである.

(1)　MUX/DEMUX/CDR 部☆
入力されたデータのシリアル-パラレル変換を行う. また,SONET/SDH における OC-48 の 2488.320 MHz クロックの生成を行う.

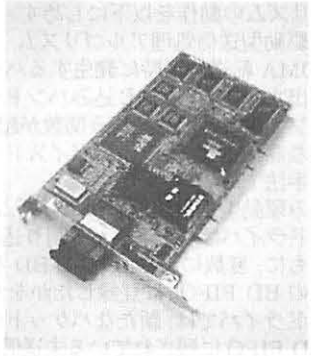


図 3　OC-48c 対応ネットワークインターフェースカード

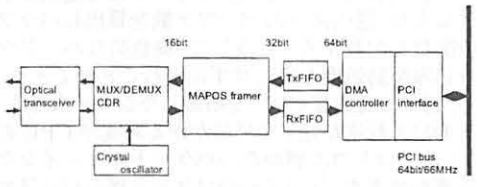


図 4　OC-48c 対応ネットワークインターフェースカードの機能ブロック図

(2)　MAPOS フレーマ部
SONET/SDH フレーミング処理, および HDLC フレーミング処理を行う. 本機能は, 我々独自に設計した ASIC により実現されている. 本フレーマは,DMA コントローラ部と 512 kbyte の FIFO メモリを介して接続されている.

(3)　DMA コントローラ部/PCI インターフェース部
カード上の FIFO メモリ (図中では TxFIFO と RxFIFO) とホストのメインメモリ間での DMA 転送を実現する. また,PCI バスと内部回路間のブリッジとして動作する.

我々のカードの DMA コントローラ部, 及び PCI インターフェース部は, 一般的な高速 NIC で使用されている物と同一で特殊な機能は実装されていない. また, 物理レイヤおよびデータリンクレイヤの実装[10] に関しては, その他の POS(Packet-over-SONET/SDH) に分類されるプロトコルとほぼ同様であ

☆ multiplexer/demultiplexer/clock and data recovery

る. よって本論文における議論は,10GbE や OC-192c POS などより高速な NIC に関しても適用する事が可能である.

4. 適応型デバイスドライバ

適応型デバイスドライバは, メモリや CPU 等のリソース使用率に応じて, パケットの送受信能力を最大に保つよう, 振舞いを動的に変更する動作を行う.

適応型デバイスドライバの初期実装は, ショートパケットを高速に送信可能なデバイスドライバのパケット送信アルゴリズムと, ロングパケットの効率的な処理が可能なパケット送信アルゴリズムの双方の長所を活かす事を目的として行われた. 本実装を行う事によって,CPU 負荷, 及びメモリ使用率の間に存在するトレードオフの間でバランスを保つ事ができ, 送信性能を大幅に改善する事が可能となった.

我々はこのような動的に振舞いを変更するデバイスドライバの応用を進めており, たとえば回線上のトラヒックの特徴やプロトコルの分布に応じて動作を変更する等の提案を行っている.

Linux に対して行った初期実装である「適応型アルゴリズムスイッチング」は FreeBSD システムにおいても, 2.1 節で解説したハードウェア割り込み負荷の問題を改善する効果があると考えられる.

本実装の概要を次節で説明する. 更なる詳細な解説は, 文献[4] に記されている.

4.1 適応型アルゴリズムスイッチング

適応型アルゴリズムスイッチングでは, パケットの送信過程においてソケットバッファ領域の使用率を監視し, バッファの空き容量に応じて, デバイスドライバに実装された 2 つのパケット送信処理アルゴリズムのうち, 片方を選択して実行する.

2 つのアルゴリズムの動作を以下にしめす.

(1) 割り込み駆動型送信処理アルゴリズム
NIC が DMA 転送修了時に発生するハードウェア割り込みを検出することで, 割り込みハンドラとして登録されたバッファ領域の開放を行う関数が起動される. これは一般的な高速 NIC やそのデバイスドライバで採用されている手法である.

(2) 非割り込み駆動型送信処理アルゴリズム
デバイスドライバはハードウェア割り込みの発生を禁止するとともに, 変数により, 幾つの BD を DMA コントローラ上の BD FIFO☆に登録したかをカウントする. デバイスドライバでは, 新たなパケットの送信要求とともに本 BD FIFO に残されている未送信パケットの数をレジスタより読み取る. このレジスタに残っている未送信 BD の数と, デバイスドライバによる送信要求数の差分により, 送信済みのバッファ数を算出しバッファの開放関数を起動する. こうして, 高負荷なハードウェア割り込み駆動処理を実行せずに済むことができる.

しかし, 近年の高速な CPU を搭載したシステムでは,NIC が実行する DMA 転送が完了までにかかる時間が,CPU の処理速度, 即ち上位プロトコル層からのパケットキューイング速度に追いつく事ができず, ドライバやソケット層のバッファを大量に消費することが知られている.

特にジャンボパケットの転送時は, データ転送の完了までにより長い時間が必要となるため顕著となる. その結果ソケットバッファ不足が頻繁に引き起こされ, アプリケーションからの新たな送信パケットがソケット層でドロップする.

ソケットバッファが不足した場合, 各々の送信処理アルゴリズムでは以下のように振舞う.

(1) 割り込み駆動型送信処理アルゴリズム
ソケットバッファが不足した場合でも,DMA 転送終了後のハードウェア割り込みを検知し, バッファの開放関数が実行されると同時に, 送信処理が再開する.

(2) 非割り込み駆動型送信処理アルゴリズム
ソケット層でパケットの送信要求がストップしてしまう為, ドライバでのバッファ開放関数が実行されなくなる. その結果, ソケットバッファの空き容量がいつまでも回復せずに, 最終的には送信処理が停止するデッドロックが発生する.

つまり,(2) の送信処理アルゴリズムは, 負荷が低く, 高速な送信処理を行える利点を持つが, デッドロックをおこす危険性がある.

この利点を保ちつつ, デッドロックの発生を回避するために, ソケットバッファの消費量を観測し, その使用状況に応じて,2 つの送信処理アルゴリズムからいずれか一方を選択するというステップを導入した.

つまり, デフォルトでは高速な非割り込み駆動型送信処理アルゴリズムを用い, ソケットバッファの空き容量が少なくなったのを検知すると, 割り込み駆動型送信処理にダイナミックに切替える. またソケットバッファが十分開放されたのを契機として, 再び高速な非割り込み駆動型処理に動作を切替える.

本手法を FreeBSD 用のデバイスドライバに実装し, 評価を行った. 更に 2.1 節で説明した DMA 転送のコストの問題を解決する為の手法を次節にて提案する.

5. 適応型バウンスバッファ手法

まず 5.1 節で, 一般的な NIC が行う DMA 転送処理に関して説明する. その後,5.2 節で, なぜ DMA 転送回数の増加が PCI バスバンド幅の利用効率低下を引き起こすのか説明する. そして, 5.3 節において, 新たな適応型バウンスバッファ手法について解説する.

5.1 DMA 転送の概要

ネットワークインターフェース用のデバイスドライバは, 通常, 複数の BD を用いてアプリケーションから送信要求があったパケットの物理アドレスやパケット長を格納し, その状態を管理している (図 2).

そして, デバイスドライバが実際に送信を行う際に, この BD のアドレスを NIC 上の DMA コントローラのレジスタに設定し, 同時に DMA 転送を開始させるレジスタを操作する.

DMA コントローラは, まずセットされた BD のアドレスを元に,BD の本体を NIC 上のメモリへ DMA 転送する. その後, 取得した BD の本体に記述された送信パケットの物理アドレスやパケット長を参照して, 送信するパケットの本体をホストのメインメモリから NIC のオンボードメモリへと転送する.

5.2 DMA 転送のコスト

DMA 転送を行っている間,PCI バスには, 以下の様に実データの転送以外の処理に費されるバスサイクルが存在する.

(1) BD の転送の為に費されるバスサイクル
(2) DMA コントローラが DMA 転送を開始するまでの, バスアービトレーションサイクル
(3) 一定量のデータを転送し終った後に挿入される WAIT サイクル

2) のアービトレーションサイクルの長さは,DMA コントローラの仕様により決定する. 一方,3) の WAIT サイクルが挿入されるまでに転送可能なデータの量はマザーボード上の PCI コントローラの仕様に依存する.

DMA 転送の回数が多くなる事により, 実データの転送以外に使用されるバスサイクルが増え,PCI バスバンド幅の利用効率の低下, つまりスループットの低下を引き起こす.

DMA 転送回数は, 単一のジャンボパケットを 1 度の DMA 転送のみで送信可能な Linux よりも,Scatter/Gather 機構を用いて, 複数に分割された mbuf クラスタから個別に DMA 転送を行う FreeBSD の方が多くなる. 特に, 我々の OC-48c 対応 NIC は 64kbyte という長大な MTU をサポートし, IPv4 の最大パケットサイズである 64kbyte のパケットを分割せずに高速な通信を行う事ができるものの,FreeBSD ではその恩恵が受けられず, 2 節で示した様な結果になったと考えられる.

5.3 適応型バウンスバッファ手法の動作

図 5 を用いて,FreeBSD 用デバイスドライバに実装した適応型バウンスバッファ手法を説明する.

デバイスドライバはカーネルとは別に, 独自のバッファ領域をデバイスの初期化と同時に確保する. 本領域は特別なメモリ割当ルーチンを用いる事により, 連続したアドレス空間で且

☆ デバイスドライバにより登録された BD を格納するオンボードメモリ.

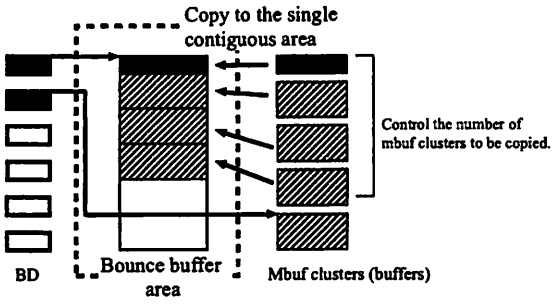


図 5 適応型バウンスバッファ手法

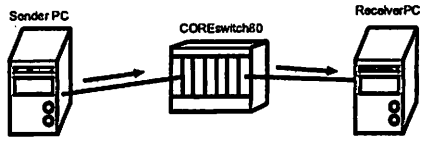


図 6 実験環境

つ,64kbyte の広大なサイズを持っている. これをバウンスバッファ領域と呼ぶ.

デバイスドライバが上位のプロトコル層からパケットを受け取ると, パケットヘッダと複数の mbuf クラスタに分割されて格納されているパケットの本体を, 本バウンスバッファ領域にコピーする. コピーは bcopy 関数を用いて行った.

その後デバイスドライバはバウンスバッファ領域の物理アドレスを BD に設定し, BD のアドレスを DMA コントローラのレジスタに登録する. あとは通常の手順と同様に DMA 転送を開始する. こうする事で,FreeBSD での DMA 転送の回数を減らす事ができ,DMA 転送の回数が Linux と同数となる.

ただし, ここでデータのコピー操作のコストに注意する必要がある. 通常, メモリ内でのデータのコピーは CPU が介在するため,CPU 負荷を増大させ, ネットワーク性能の低下を引き起こす原因となる.

我々は, バウンスバッファ領域にコピーを行うバッファの数を変更可能とする事で過度のコピーを避けるよう制御を行った. 具体的には, 簡易なベンチマークにより, データのコピーにかかる CPU 負荷を測定し, その結果に応じてコピーするバッファの数を変更する.

6. 実装したシステムの評価

図 6 に示す実験環境を用いて, 適応型デバイスドライバの効果を検証した. 表 1 の仕様の送受信 PC 間を MAPOS[10] 対応スイッチである CORE switch 80 を介して接続した. このスイッチを介した場合,2.5 μs 程度のディレイが発生したが, 本評価に影響を与える値では無いと考えられる. また,PC 間のラウンドトリップタイムを,FreeBSD に付属の ping を用いて測定した所, 50 μs であった.

6.1 適応型アルゴリズムスイッチングの効果

4.1 節で解説した適応型アルゴリズムスイッチングの効果を図 7 に示す. 本グラフは図 1 と同様にスループットをパケット処理数 (上図,[kpps]) とデータ量 (下図,[Mbps]) で表したものである。

実験結果から以下の効果がある事がわかった.

(1) パケット処理能力を示す上図において, ショートパケットでの転送能力が最大 70% 向上しているのが分かる*. 特に, ショートパケットでは送信バッファを多く消費しないことから, 測定期間中完全に非割込み駆動型送信処理が行われ,CPU 負荷の削減効果が高かったと言える。MINCLSIZE(213byte) よりも大きなパケットの転送時は, ハードウェア割り込み削減の効果はなくなり, 通常の送信アルゴリズムと同等の性能となった.
MINCLSIZE よりも小さなパケットは,MLEN** と言う定数で定義された大きさのバッファに格納される一方, それよりも大きなパケットは,2kbyte の mbuf クラスタに結合, あるいは分割されて保存される為, プロトコル処理が行われる単位や DMA 転送が行われる単位が 2kbyte となる.
よって,200byte 付近から 2kbyte の間のサイズのパケットでは,CPU 処理負荷の低下や DMA 転送効率が向上し, グラフに見られる様にパケット処理能力が飛躍的に向上していると考えられる.

(2) Linux では,9kbyte より小さなパケットの転送でのみ効果が現れていたのに対し,FreeBSD ではそれよりも大きなパケットの転送でもスループットが向上しているのが分かる. これは,FreeBSD では通常のアルゴリズムを用いた場合, 大きなパケットでも割り込みの数が多く, それ故 Linux にくらべて割り込み削減の効果が高かった事が考えられる.

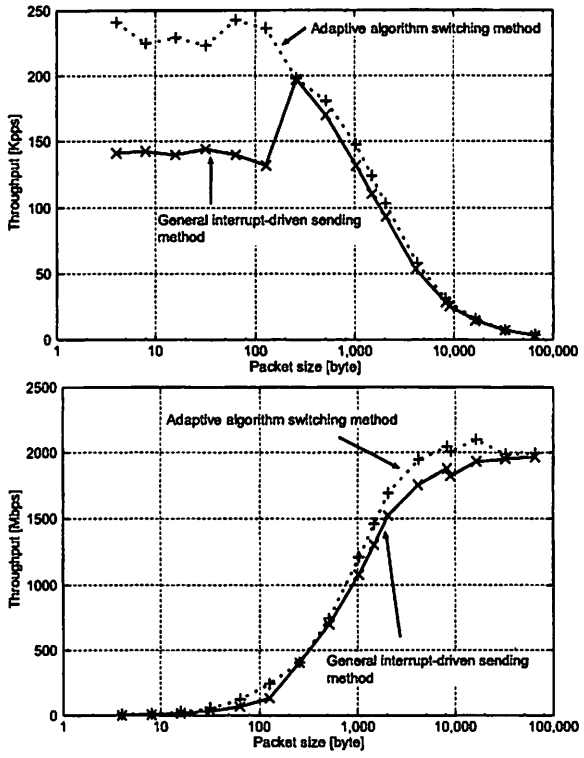


図 7 適応型アルゴリズムスイッチングの効果

6.2 DMA 転送のコストの定量的評価

適応型バウンスバッファ手法の評価の前に, まず DMA 転送のコストがどの程度スループットに影響を与えるのかを明らかにする. 定量的に測定するために, 以下のように修正したデバイスドライバを使用した.

DMA 転送コストの測定用修正デバイスドライバの動作

デバイスドライバは, 上位のプロトコル層からパケットの送信要求を受けると, パケットの中味を無視して廃棄し, パケットのヘッダ部分のみを DMA に登録する. かつ, あらかじめ用意しておいたダミーのパケットをパケットの中味として登録する. このダミーのパケットはデバイスドライバの初期化時に作成され, 有意なデータは含まれておらず, かつメインメモリ上では 1 つの連続したアドレス空間に格納されている. このデバイスドライバを用いれば,Linux と同様な DMA 転送の振舞い, つまり Scatter/Gather DMA 転送を行わない場合のスループットを測定できる.

かつ, 修正デバイスドライバを用いても, カーネルのプロトコル処理は FreeBSD のプロトコルスタックがそのまま用いられている為,DMA 転送のコストとともに, mbuf 構造を用いたプロトコル処理のコストも定量的に評価する事が可能である.

つまり,

- 修正デバイスドライバで得られたスループットと Linux でのスループットの差分により,mbuf 構造をベースとした

* Linux では, 最大でも 50% 程度の向上であった.
** MLEN は/usr/src/sys/sys/mbuf.h に定義されている.

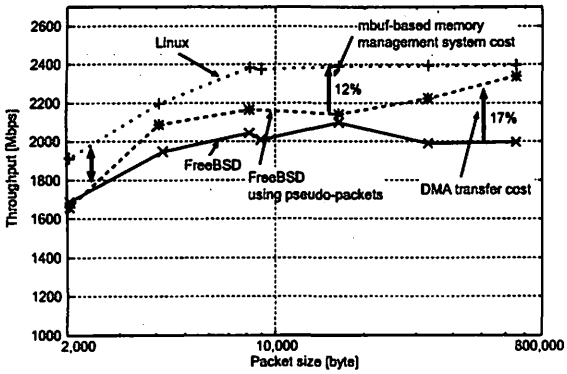


図 8 DMA 転送のコストの評価

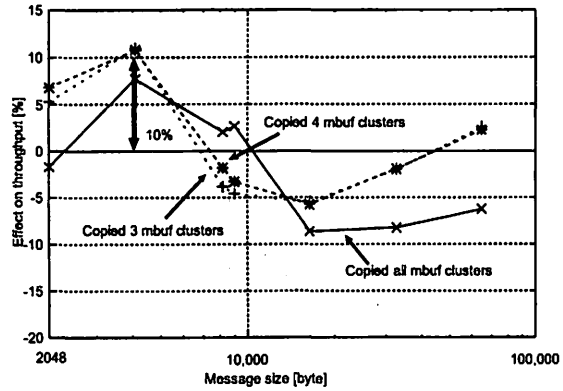


図 9 バウンスバッファ手法の効果

バッファ管理システムで行われるプロトコル処理のコストが明らかになる.

- 修正デバイスドライバで得られたスループットと通常の FreeBSD でのスループット (図 7) の相違により,DMA 転送のコストが明らかになる.

本評価の結果 (図 8) より以下の知見がえられた.

- 64kbyte のダミーパケットを使用した場合, スループットは約 17% 向上し,Linux と同様の値に引き上げられた. これが Scatter/Gather DMA 転送を用いた時の DMA 転送のコストであると言える. また, この実験結果より 2kbyte 以上の全パケットサイズに対して,DMA 転送のコストがスループットの低下を招く主要な原因であると確認できた.
- 64kbyte よりも小さいパケットを用いた場合, ダミーパケットを用いても Linux よりも低いスループットしか得られなかった. たとえば,16kbyte のパケットを転送した場合,12% 程度スループットが低かった. この原因は主に mbuf 構造を用いたバッファ管理のコストによるものと考えられる.

6.3 バウンスバッファ手法の効果

実験では, バウンスバッファ領域へコピーされる mbuf クラスタの数を,3 及び 4 として評価を行った. より少ない mbuf クラスタで構成されるパケットに関しては, 全ての mbuf クラスタがコピーされた時点で即時にコピーを終了する.

また, 全ての mbuf クラスタがバウンスバッファ領域にコピーされる場合も同様に評価を行った.

結果を図 9 に示す. X 軸はメッセージサイズ, 即ちヘッダ部分の大きさをのぞいたパケット長である. Y 軸には, スループットの変化をパーセンテージで表した.

バウンスバッファ手法を導入する事によって,4kbyte のパケットでは,10% 程度スループットが改善した事がわかる. しかし,64kbyte のパケットではコピー量によって, 改善はほとんど見られないか, 逆にスループットが低下している. 特に, 全ての mbuf クラスタをコピーするようなコンフィグレーションを行った場合, コピーにかかる CPU 負荷のコストが,DMA 転送の回数を減らした事による改善効果を上まわるために, スループットが低下したと考えられる.

以上より本手法によるスループットの改善効果は, メモリのバンド幅や CPU の処理速度に依存する事が分かる.

7. ま と め

10GbE や OC-48c(2.4Gbps) POS 等のギガビット超の帯域をもつネットワーク環境下において BSD 系の OS が Linux と比較して, 最大ネットワークスループットが低い事に着目し, カーネル内の調査や実験を通してその要因を明らかにした.

BSD 系 OS の mbuf 構造を用いたバッファ管理法に起因するスループット低下に関しては, 割り込み処理負荷, 及び DMA 転送のコストが大きく寄与しており,OC-48c 対応 NIC 用のデバイスドライバを用いてその影響を定量的に明らかにした.

スループットの低下を軽減するための適応型デバイスドライバを提案し, 実験を通してネットワークスループットを改善する効果を確認した.

(1) 適応型アルゴリズムスイッチング手法では, ソケットバッファ使用率に応じた送信処理アルゴリズムの選択を行う事にで, パケット処理性能を最大 70% 向上した.
(2) 適応型バウンスバッファ手法では,DMA 転送数を減少させる事でスループットが最大 10% 向上した.

今後は, 適応型デバイスドライバの適用範囲を広め, より多くのリソースを監視することで, アプリケーションが安定して広帯域ネットワークを活用できるよう改良を進める予定である.

また, インターフェース速度の高速化の観点から, 我々が新規開発した OC-192c POS,10GbE-LAN/WAN PHY の 3 種類のプロトコルに対応した NIC を用いて更なる評価を進めている. 本論文の一次評価で示したように,10Gbps に対応した NIC の場合には回線速度限界よりも PCI-X バスの帯域限界によりネットワークスループットが決まるため, 適応型デバイスドライバを用いた時の DMA 転送コストの軽減効果による PCI-X バス利用効率の向上効果がより大きく影響すると考えられる. また単位時間あたりのパケット処理数も増すため,CPU 処理負荷を更に軽減する TSO(TCP Segmentation Offloading) の様な新たな手法も必要になる可能性がある. 特にショートパケットを用いた場合でも, ワイヤーレートでのプロトコル処理を可能とするシンプルなハードウェアアシスト機能の実現に向けた研究を行う予定である.

参 考 文 献

1) "Chelsio", http://www.chelsio.com/
2) "NTT News Release", http://www.ntt.co.jp/news/news01e/0110/011026.html
3) "FreeBSD network performance tuning", SUCON '04, September 2004, Zurich, Switzerland
4) K. Shimizu, T. Ogura, T. Kawano, H. Kimiyama and M. Maruyama, "OC-48c High-Speed Network PCI Card: Implementation and Evaluation," IEICE Trans. vol. E86-D, No. 11, pp. 2380-2389, November 2003
5) "Intel PRO 10GbE Server Adapters", http://www.intel.com/network/connectivity/products/10GbE.htm
6) J. H. Salim, R. Olsson, and A. Kuznetsov, "Beyond Softnet," in 5th Annual Linux Showcase (ALS) and Conference. USENIX, Nov 2001, pp. 165172.
7) Fujita Tomonori and Ogawara Masanori, "Performance optimized software implementation of iSCSI," in SNAPI'03, Sept 2003, LA, USA.
8) "Data Reservor", http://data-reservoir.adm.s.u-tokyo.ac.jp/
9) 清水, 小倉, 川野, 君山, 丸山, "OC-192c POS/ 10GbE デュアルモード対応高速通信カードの実装と評価," IN2004-28, pp.7-12.
10) K. Murakami and M. Maruyama, "IPv4 over MAPOS Version 1," RFC-2176, 1997.
11) J. Bonwick. "The slab allocator: An object-caching kernel memory allocator," In USENIX Summer, pages 8798, 1994.
12) "The Public Netperf Homepage," http://www.netperf.org/