Vol.2010-HPC-126 No.24
2010/8/4

Webブラウザを用いた長距離データ転送の高速化

井口 義己[†] 谷田 直輝[†] 稲葉 真理[†] 平木 敬[†]

近年 10Gbps の広帯域の高速ネットワークが普及してきたが，一般的な TCP を用いたアプリケーションは，特に長距離の高速ネットワークにおいて満足なスループット性能を出せないことが知られていた．我々のチームはこの問題を解決し LFN 上で高速なデータ通信を成功させ，ネットワークの有効利用の道を開いた．さらに，その成果を応用し，高速なデータ転送を行える Web ブラウザとして UsadaFox (Ultra-High-Speed file-Acquisition-system over Distance with Apache and fireFOX) を設計・実装し，日米間 LFN 上で高速なデータ転送を行えることを示した．UsadaFox の開発を通して，ネットワークアプリケーションを LFN 向けに高速化するためにはメモリ上のデータの扱いが重要だということを示した．高速データ通信を行う際に扱うデータは非常に大きいため，バッファ間のデータコピーを行う際には注意を払って実装を行う必要がある．本論文では UsadaFox で実際にデータ通信を行った際のバッファの挙動を計測し．適切なメモリ操作やバッファ量についての考察を行った．さらに，一般的なアプリケーションを長距離高速通信に対応させる際の指針を示した．

Performance Optimization of HTTP Data Transfer over Long Distance with Web Browser

Yoshiki IGUCHI[†], Naoki TANIDA[†], Mary INABA[†], and Kei HIRAKI[†]

With the rapid progress of high-bandwidth network technology, high-speed networks cover the entire world. It is well known that, in general, efficiency of data transfer between two points drastically drops as their distance grows. In our previous work, we proposed UsadaFox, which accelerates long distance data transfer. UsadaFox consists of modified Firefox browser and tuned Apache HTTP server. UsadaFox attained more than 6.5Gbps for file download between U.S. and Japan where round trip time was 136ms. We found it is important to reduce memory operations in the development process of UsadaFox. In this paper, we observed the behavior of buffers in UsadaFox. Based-on our observation, we show proper memory operations and appropriate size of the buffers for the network applications to support high-speed data transfer on high-speed large-delay networks.

1. はじめに

近年のネットワーク技術の向上に伴うネットワークの広帯域化は著しい．10Gbps のネットワークは，当初は大量のデータを扱う研究機関や大企業などごく限られた領域でのみ使用されていたが，ネットワーク需要の増加に伴いそれ以外の領域にも急速に普及しつつある．

高遅延高帯域ネットワークは LFN（Long Fat-pipe Network）と呼ばれる．ここ数年のネットワーク技術の発展とクラウド技術の普及に伴い，データをローカルではなくネットワーク上のストレージに保存するケースが増えているが，海外のサーバなどからの LFN を経由したデータ転送は，幾つかの原因により転送速度が上がらないことが知られている．そのため，LFN 上での高速データ転送への要求は高まっている．

我々は，2007 年に 10Gbps ネットワークにおける TCP を用いたデータ転送において，理論限界に近い 99%の帯域利用率を達成した[1]．さらに，ストレージ間の転送においても LFN を通して高速に通信が行えることを示した[2]．

次なる目標として，データ転送に特化した軽量なアプリケーションではなく，一般的なネットワークアプリケーションで高速通信を実現することに挑戦した．この目標を達成するため，高速なデータ転送を行える Web ブラウザベースのシステムとして UsadaFox を設計・実装した[3]．我々は，UsadaFox を用いて実際の日米間 LFN 上で 6.5Gbps の速度でデータ転送が行えることを示した．

UsadaFox は Firefox と Apache を長距離高速通信向けに最適化した，Linux 上で動作するシステムである．LFN 上におけるデータ通信の高速化を，高速なストレージを用意し，MAC レイヤでの制御[1]や TCP レイヤのチューニング[4]を行い，メモリ操作を最適化することによって実現した．高速データ通信では大量のメモリ帯域が消費される．メモリ操作だけでメモリバス及びプロセッサ時間を占有してしまわないために，メモリ操作に関して注意を払い実装を行う必要があった．さらに，高速データ通信を行う際はバッファにも着目する必要があった．高速データ転送においては僅かな時間で大量のデータを受信するため，不適切なバッファ設定はバッファの詰まりを生じさせ，さらにパフォーマンス低下につながる．そこで，実際にバッファがどのように使われてどのようにデータが流れているか調査を行い，適切なバッファ量の選択やメモリ操作を行う必要があった．

本論文では UsadaFox を用いて LFN 上で高速通信を行い，バッファの使用量や通信速度について計測を行った．そしてネットワークアプリケーションにおける最適なパラメータについての考察を行った．

[†] 東京大学
The University of Tokyo

二章では，一般的なアプリケーションが LFN において高速データ転送を行う上の障害となりうる原因について述べる．三章では，我々が開発した UsadaFox が採用している 3 バッファモデルとその性質について述べる．四章では，UsadaFox を用いてバッファの挙動を調査し高速にデータを扱うことができるバッファの性質について考察した．続く五章にて，関連研究と UsadaFox の比較を行ない，六章で本論文の総括を行った．

2. LFN における高速データ転送実現への障害

2.1 TCP レイヤの問題

長距離 LFN 上での伝送では，距離に比例したパケットの遅延や中間スイッチでの遅延が累積し，短距離に比べ RTT が長くなる．たとえば東大との往復遅延時間を計測した場合，状態がよければ国内本州においては 30ms 以下の RTT で通信できているが，海外に目を向けるとアメリカの西海岸では 80ms 程度，ヨーロッパでは 200ms 程度，国や地域によっては 250ms を超えることもある．

長距離の高速 TCP 通信を行うためには，送受信側共に大きなバッファが必要であることは知られている．TCP は送信データが確実に受信側に届いたことを確認することができる．送信したデータが受信側に届くと，受信側は ACK パケットを送信側に返す．送信側は ACK パケットを受け取ることによって，送信したデータが無事に到達したことを確認できる．もし ACK を受け取ることが出来なかった場合は，そのデータが失われたと判断し，データを再度送信する．そのため，送信側はデータの送信後にも ACK を受信するまで再送用にデータを保持しておかなければならない．ACK を受信していないデータ，つまり送信側が無事到着したことが認識されていないデータをインフライトデータ（in-flight data）と呼ぶ．インフライトデータの最大値は BDP（bandwidth-delay product，帯域遅延積）と呼ばれ，帯域と RTT の積で表される．たとえば，RTT が 200ms の経路において 1Gbps = 125MB/s で通信する場合の BDP は 25MB となる．RTT が長いほど，帯域が広いほど，そのネットワークの BDP は大きくなる．そのため高速な通信を行うためには，送信側には大きいバッファが必要である．

さらに，送信側は受信側から通知された受信可能データ量に応じて送信速度をコントロールする働きがある．受信側から送信側に通知される受信可能データ量のことを RWIN（Receive Window，受信ウィンドウ）と呼ぶ．送信側は，インフライトデータを通知された RWIN の値以下に抑える．たとえば，RTT が 200ms の経路において RWIN が 1MB の場合，送信速度は 5MB/s = 40Mbps に制限されてしまう．RWIN は受信側のソケットバッファの一部が割り当てられるため，RWIN を制限しないためにも，受信ソケットバッファの値も多く確保しなくてはいけない．

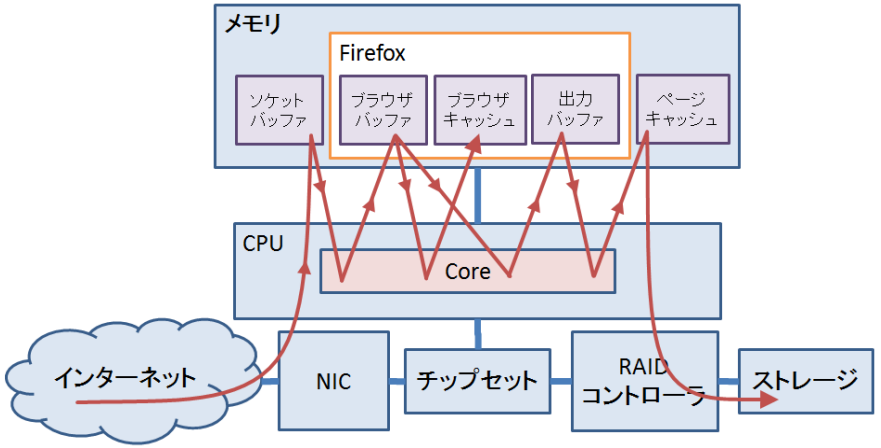

図 1：Firefox のデータフロー

2.2 アプリケーションの問題

アプリケーションの実装に関しての問題点も存在する．大きな問題として挙げられるのは，メモリとプロセッサ間のバス幅は有限であり，データを読み書きするたびにこのバスが使われるということである．高速データ通信でメモリに対して操作を行うためには，短時間に大量のデータを処理しなくてはならず，バスを圧迫する．高速なデータ転送を維持するためはメモリへの読み書き回数を減らし，バスが飽和しないようにしなくてはいけない．図 1 は Firefox を用いてファイルのダウンロードを行った時のデータフローである．メモリコピーが複数回行われていることが示されている．無駄なメモリコピーはリソースを無駄に消費し，ボトルネックの原因となる．

3. UsadaFox とその設計

3.1 UsadaFox

UsadaFox は，HTTP サーバとブラウザで構成された高速データ転送システムである．改造 Firefox と最適化 Apache を組み合わせた，LFN の帯域を有効活用したデータ転送を行えるシステムである．Web ブラウザをベースにしているため，ワンクリックで高速なデータ転送を行うことができる．UsadaFox は，Linux 上で動作し，特殊なパーツやハードウェアではなく一般的な汎用 PC パーツで構成されたサーバ上で性能を発揮することができる（表 1）．

UsadaFox ブラウザでは，ソフトウェア内部のデータフローとして 3 バッファモデル

表 1　UsadaFox を構成するハードウェア

CPU	Intel Core i7 940
マザーボード	ASUS Rampage II GENE
チップセット	Intel X58 chipset
メモリ	6 GB DDR3 SDRAM
NIC	Chelsio S310E-CR NIC
RAID Controller	Adaptec ASR-51245
SSD	Intel X25-E Extreme 32 GB×6 RAID0 にて使用

（図 2）を採用した．ネットワークより受け取ったデータは，3 つのバッファを経由してからストレージに保存される．「データ入力時のソケットバッファ」「アプリケーション内部のバッファ」「データ保存時のファイルページキャッシュ」である．一般的に経由するバッファ数が少ないとバッファ間のコピーのためのメモリ操作が減るため，パフォーマンスが上がる．しかし，ネットワークアプリケーションが何らかの処理や保存を受信データに対して行うことを考えると，極端に少ないバッファで設計を行うのは保守性の面から難しくなる．我々は，UsadaFox にて，3 つのバッファ数で LFN にて高速通信が出来ることを実証した．入力，処理，保存の 3 つのバッファを持つことは，アプリケーションの保守性とパフォーマンスを両立させた妥当な設計である．

そのバッファ構成をそれぞれどのように設計するかは，パフォーマンスを大きく左右する．本論文では，LFN 上で大量のデータを扱うネットワークアプリケーションを対象に，この 3 バッファモデルを用いた際の最適な構成を検討する．3.2 節から 3 バッファモデルの特性を検討したのち，次章にて 3 バッファモデルの挙動を実験にて確かめる．

3.2 ソケットバッファ

入力時の第一のバッファは，カーネル空間内に配置されるソケットバッファである．ネットワークから受信したデータを一時的に溜め，受信データの揺らぎを吸収する役割を持つ．また，このバッファは入力の緩衝領域としての役割だけではなく，TCP の RWIN として使われる領域も兼ねる．そのため，ソケットバッファには BDP の値にさらに緩衝領域として必要である容量を加えた値を必要である．このように，長距離通信においては送信速度が制限されることのないようにソケットバッファのサイズを決める必要がある．

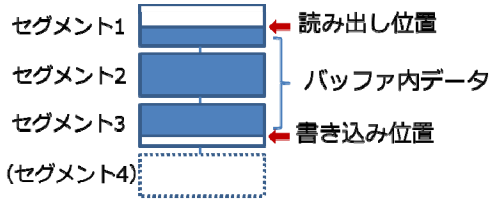

図 2 セグメントバッファ

3.3 アプリケーションバッファ

アプリケーションはカーネル空間上のソケットバッファの内容を直接読めないため，送られたデータを扱うためには，ユーザ空間内のアプリケーションバッファに存在する必要がある．recv などのシステムコールを通じて，カーネル空間内のソケットバッファ内のデータをユーザ空間のバッファへコピーする．

UsadaFox 内部において，このバッファはセグメントバッファ（図 3）として実装されており，複数個のセグメントの集合をバッファとして扱う．セグメントは必要になった時点でメモリ内に確保され，不要なセグメントは随時解放する．この方式には，必要なセグメントのみメモリ上に確保されるため，省メモリであるという利点がある．ただし，読み書きはセグメント境界を跨いで行えないため，セグメントサイズを小さくした場合は読み書きの際のオーバーヘッドの負荷が高くなる．

アプリケーションバッファは，コンピュータ内部で完結しているため，瞬間的なブロックが発生してもパフォーマンスロスに繋がらない．しかし，このバッファの制御は OS ではなくアプリケーション側で行うため，アプリケーションの開発においては長時間のブロックが発生しないように留意して設計を行う必要がある．

3.4 ページキャッシュ

ストレージへファイルとしてデータを書き出す際には，ページキャッシュを経由してストレージに書き込まれる．このページキャッシュは出力時のバッファとしての役割を果たし，OS によって非同期にストレージに書き込まれる．

アプリケーションバッファからページキャッシュへは，mmap を用いてアクセスを行う．mmap は Linux のシステムコールであり，ページキャッシュ領域に対してユーザ空間からのアクセスを可能にする[5]．アプリケーションは，アプリケーションバッファのデータを読み出し，データに対して処理を行ない，処理済みのデータをページキャッシュに書き出す．

書き出されたデータは，OS が非同期にストレージへのフラッシュを行う．高速データ通信では，短時間に大容量のデータを書き込む必要がありこのキャッシュへも随時大量のデータが保存される．そのため，ページキャッシュをフラッシュするアルゴ

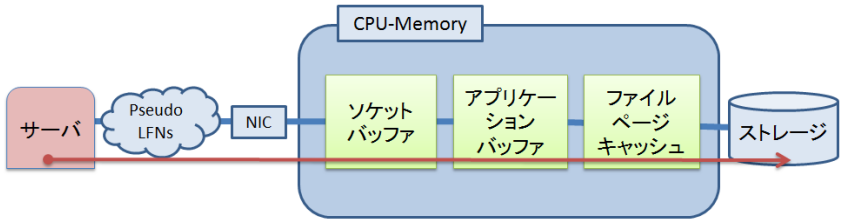

図 3　実験環境でのデータの流れ

リズムが適切ではないとストレージへの書き込み速度が劣化してしまう可能性が考えられる．

特にストレージへの書き込みがボトルネックとなっている場合は，ストレージに対して随時書き出しを行わせないと，ページキャッシュが滞って後続する入力をブロックしてしまうため，パフォーマンスが出ない．その対策として，UsadaFox では，madvise を用いてカーネルに対して書き込みを行ったページキャッシュを随時フラッシュするように指示を行った．madvise も Linux のシステムコールであり，mmap を使用してマッピングしたメモリ領域の使い方をカーネルに対して予告し，キャッシュの扱い方を指示することができる[6]．たとえば，今回のようなダウンロードしたデータをストレージに書き出すような用途では，一度 mmap を使って書き込んだ領域に対して再度アクセスすることはない．その場合は，madvise を用いてストレージに対して連続的なアクセスを行うことを予告する．それを受け取ったカーネルは，その mmap 領域に対しては，書き込みが行われたページをすぐにフラッシュする挙動を示すようになる．

4. 実験

以上のような各バッファの特性を踏まえて，UsadaFox を用いて各バッファの実際の使用状況の計測を行った．さらに，以下の点に関して検証を行った．

- LFN 上での高速なデータ転送時におけるバッファの挙動の調査
- パフォーマンスが劣化する条件の検証

実験は，ネットワークエミュレータを用いて構成した擬似 LFN 上で行った．UsadaFox のサーバとクライアントをネットワークエミュレータを経由させて繋ぎ，その上でデータ転送実験を行った（図 3）．ネットワークエミュレータには Anue H-Series Network Emulator を用いた．このエミュレータはネットワークに人工的な遅延を，片方向ずつそれぞれ約 0-800ms の範囲で任意に設定することができる．今回の実験では遅延量は 100ms ずつ，往復で 200ms に設定した．

40GB/s のファイルの転送を行ったときの転送速度およびソケットバッファ及びアプリケーションバッファの使用量を測定した．転送時には，送信側 NIC においてペーシング[1][3]を行ない，ストレージの最大速度である 6.5Gbps で送信速度に制限をかけている．

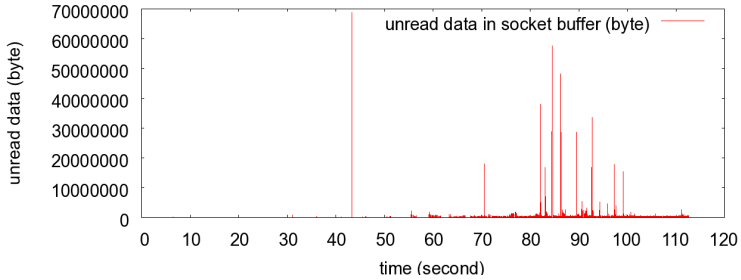

図 4　ソケットバッファの使用量の推移

表 2　ソケットバッファを変化させた時の受信速度の推移

SO_RCVBUF の設定 [MB]	25	50	75	100	125	150
平均速度 [MB/s]	235.42	470.36	701.95	806.78	811.66	811.68
RWIN サイズ [MB]	46.87	93.75	140.62	187.50	234.37	281.24
RWIN / RTT [MB/s]	234.35	468.75	703.10	937.50	1171.35	1406.20

4.1 ソケットバッファの実験と結果

ソケットバッファはネットワークから受信したデータが格納される．RWIN が BDP を下回るとスループット低下につながる．最大限のパフォーマンスを発揮させるには，緩衝領域として必要なバッファサイズに加え，BDP サイズを加えたサイズを受信バッファに確保する必要がある．この実験では，setsockopt システムコールを用いて SO_RCVBUF のパラメータの値に余裕をもたせた 600MB を設定した．そうすることにより，受信ソケットバッファには，SO_RCVBUF で指定した値の倍である 1200MB から，管理領域を引いた領域が割り当てられる[7]．RWIN はその受信ソケットバッファのうち，未使用部分の一部が割り当てられる．

ソケットバッファの使用量の推移を調べた．ソケットバッファの使用量を求めるために，Linux の TCP ソケットのパラメータである copied_seq，rcv_nxt という 2 つのパラメータを読み取った．copied_seq は，バッファ中のアプリケーションにすでにコピーされたデータの終端を示し，rcv_nxt は受信済みのデータの終端を示す．この 2 つの値の差を計算することによってソケットバッファの使用量を求めることができる．測定は，カーネルがパケットを受信した際に呼び出される内部関数である tcp_recvmsg

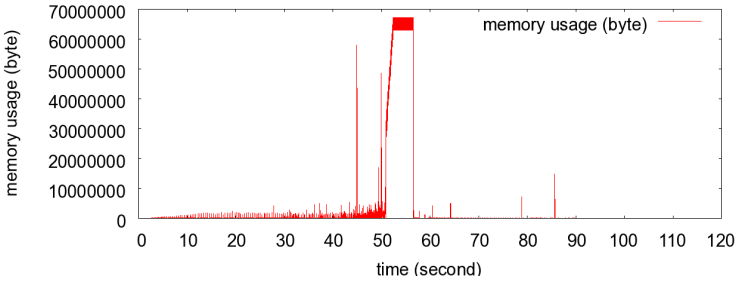

図 5　アプリケーションバッファの使用量の推移

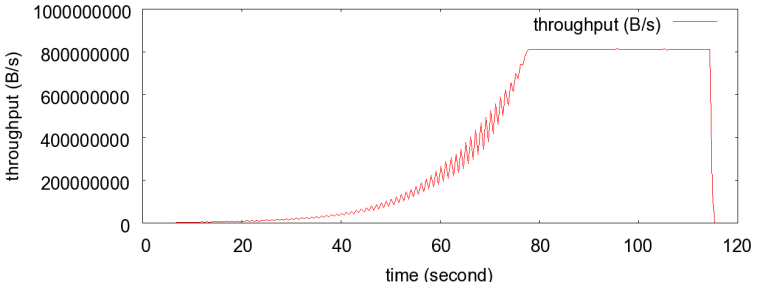

図 6　転送時のスループット

にフックを掛け，この関数が呼び出される度に目的のパラメータの値を記録することにより行った．

ソケットバッファの使用量の推移を測定した．図 4 がその結果である．113 秒で 40GB のデータの転送を完了した．バッファ使用量は低い値で安定しているが，瞬間的なバースト状態が観測された．その中で最も大きいものは開始後 43 秒地点で，約 68.8MB 弱のバッファを使用していた．このケースでは 6.5Gbps のスループットを得るためには，理論上 BDP は6.5Gbps × 200ms = 162.5MBであるため，231.3MB 以上のソケットバッファが確保されていればよいと考えられる．

また，同様の実験を何度か行ったところ，このような瞬間的なバーストがソケットバッファにはしばしば現れた．後半の速度が出た段階でのバーストが多いが，図 4 の 43 秒地点のケースと同様の加速段階のバーストも発生するケースも存在した．

ソケットバッファサイズと最大速度の関係を得るために，ソケットバッファサイズを変更しながらスループットを計測した．スループットは速度が最大速度付近で安定している転送後期の 30 秒の平均を計測した．また，転送開始時（バッファにデータが含まれていない状態）の RWIN も記録した．RWIN は TCP パラメータの rcv_wnd の値を取得した．

実験の結果が表 2 である．RWIN のサイズは SO_RCVBUF で指定した値の約 1.88 倍が設定されることがわかる．6.50Gbps 時の BDP である 166.4MB より大きい RWIN が設定されていれば，6.49Gbps で転送が行えることがわかった．また，SO_RCVBUF が小さい場合は，RWIN/RTT の値に速度が制限されることがわかった．

4.2 アプリケーションバッファの実験と結果

アプリケーションはカーネル空間のソケットバッファ上を直接操作できないので，recv などのシステムコールを通してアプリケーションバッファにコピーを行う．

UsadaFox 内にバッファの使用量を調査するために，アプリケーションバッファに対する書き込み・読み込みにフックを掛け，使用量の記録を行った．アプリケーションバッファのサイズは 64.0MB に設定した．図 5 は，図 4 を記録した際に同時に記録したアプリケーションバッファの使用量のグラフである．

開始後 43 秒地点にてバーストが発生し，最大時で 58MB のバッファが使われている．これは，ソケットバッファ内での 43 秒地点から始まるバーストとの関連性が考えられる．

また，49 秒地点から 56 秒地点のあたりまでバーストが発生している．アプリケーションバッファの最大容量である 64.0MB と 60.0MB を使用量が激しく上下している．バッファの最大容量である 64.0MB に到達した後，最大の読み出し単位である 4.0MB を読み込む，という動作を連続して繰り返しているためである．

この転送実験では，同時にスループットも計測した（図 6）．0.5 秒毎に受信したデータ量を測定した．速度の増加時（約 10-79 秒地点）にスループットが 0.5 秒単位で上下している．これは，RTT が 200ms であるため 200ms 周期でバーストが発生しているのが原因である．バッファのバーストが発生している時においても，スループットへの影響はみられないことがわかる．

4.3 考察

制限速度である 6.5Gbps 以下では，スループットは RWIN/RTT にほぼ抑えられているということがわかった． RTT:200ms のネットワークにて 6.5Gbps の性能を出すためには, BDP である 162.5MB 以上を RWIN に指定するために, 162.5/1.88 = 86.44 MB 以上のサイズを SO_RCVBUF に指定する必要がある．また，少なくとも今回の実験範囲ではそれ以上のバッファサイズを指定しても，パフォーマンスへの影響はほとんどなかった．

ソケットバッファで発生しているバーストは，RWIN を考えても余裕があるためスループットへの影響が出ないことがわかった．また，アプリケーションレイヤのバーストはソケットバッファにて吸収され, スループットへの影響がないことがわかった．

アプリケーションバッファにおいてバーストが発生し，バッファを埋めてしまう場合でも軽微な程度であればパフォーマンスに影響を与えないことがわかった．また，ソケットバッファにおいて瞬間的なバーストが発生した場合でも，すぐにアプリケーションバッファ，さらにファイルキャッシュへとデータが流れ，バーストが吸収されることがわかった．軽微なバーストは日常的に発生しているが，次のバッファにすぐに吸収されるため，前のバッファを詰まらせる問題やパフォーマンスに対する影響は発生しないことがわかった．つまり，バッファ間のメモリコピーが頻繁に行われている場合は，バッファのサイズを小さくすることができる．逆に，バッファ間のメモリコピーの頻度が少ない場合には，バッファを大きく取る必要があると推察できる．

また，43 秒付近でソケットバッファのバーストが発生しているが，このバーストはその時点の転送レートを考えると不自然に大きい．NIC からソケットバッファへのコピーにおいて数十 MB 程度の何らかのバーストが発生している可能性が考えられる．

また，アプリケーションバッファの 49-57 秒付近のバーストに関しては，ソケットバッファの様子を見ても詰まった様子はないため，おそらくファイルキャッシュ側に何らかの問題があるのだと考えられる．

5. 関連研究

GridFTP [8] は，グリッドコンピューティング環境向けのデータ共有システムである．FTP を拡張したプロトコルを用いて，複数の TCP ストリームを用いた並列データ転送を行う．GridFTP は並列 TCP ストリームを用いてストリーム全体での速度向上を図る試みであるが，我々は UsadaFox を用いて，単一ストリームの転送においても，バッファに対して適切な設計を行うことで高速なデータ転送が行えることを示した．

FastSoft E シリーズ [9] は，FastSoft 社による長距離データ転送のためのアプライアンスである．サーバ側にハードウェアを置くことによって通常の TCP を FastTCP [10] に変換して長距離においてもデータ転送を行うことができる．我々は，UsadaFox においてソフトウェアレベルでの変更を適切に施せば，通常の TCP においても高速なデータ転送を行うことができることを示した．UsadaFox には，ハードウェアの用意やネットワーク構成の変更を行う必要がないというメリットがある．

Fast And Secure Protocol (FASP) [11] は Aspera 社による長距離データ転送のためのプロトコル，およびそのプロトコルを用いた長距離データ転送を高速に行うソフトウェアのアプライアンスである．ブラウザ等の既存ソフトウェアと連携し，FASP に対応したサーバからのダウンロードを FASP ソフトウェアが代行し高速なダウンロードを行ことができる．転送には，UDP 上にて独自の輻輳制御と信頼性を付加したプロトコルを用い，長距離において高速なデータ転送を行うものである．しかし我々は UsadaFox では，バッファ等に対して適切な改善を行えば通常のブラウザソフトウェアからでも十分な高速データ通信が行えることを示した．サーバに関しても Apache 等の一般的な HTTP サーバを流用することができ，特別なソフトウェアを用意しなくともブラウザ単体で高速なダウンロードを行うことができる．

6. まとめ

本論文では，UsadaFox の開発を通して，ネットワークアプリケーションが LFN を用いて性能を出すための方法論を示した．高速データ通信を行う際に重要であるメモリ上のデータの扱いに着目し，UsadaFox のバッファの挙動を測定しながらパフォーマンスへの影響を調べた．そして，アプリケーションが LFN においてパフォーマンスを維持するために必要なバッファの条件について考察を行った．

今後は，擬似 LFN ではなく実際の LFN 上でも実験を行い，実環境に近いネットワークでの挙動についても調査していきたい．また，ページキャッシュの挙動についても調査を行ない，バッファの相互関係をより深く調べていきたい．

謝辞

UsadaFox を使用した実験を行う際において多くの助言や支援を頂きました，慶應大学の加藤朗氏，NICT の長谷部克幸氏，東京大学の玉造潤史氏，下見淳一郎氏，石井康雄氏，小泉賢一氏にこの場を借りてお礼申し上げます．なお，本研究は科研費基盤 (S)21220001 の助成を受けて行われた．

参考文献

1) Yoshino, T., Sugawara, Y., Inagami, K., Tamatsukuri, J., Inaba, M. and Hiraki, K.: “Performance optimization of TCP/IP over 10 gigabit ethernet by precise instrumentation,” *In Proc. of the 2008 ACM/IEEE conference on Supercomputing*, IEEE Computer Society, pp. 1-12, (2008) .
2) 谷田直輝，稲葉真理，平木敬: “TCP による長距離ディスク間データ転送の高速化”，情報処理学会研究報告, Vol.2009-ARC-184 No.23, pp. 1-7, (2009).
3) Iguchi, Y., Tanida, N., Koizumi, K., Inaba, M., Hiraki, K.: *USADAFOX*: “Ultra-High-Speed file-Acquisition-system over Distance with Apache and fireFOX,” *In Proc. of TERENA NETWORKING CONFERENCE 2010*, (2010).
4) 玉造潤史，吉野剛史，稲上克史，菅原豊，稲葉真理，平木敬: “10 ギガビットネットワーク上での高効率 TCP/IP 通信の実現”，情報処理学会研究報告, Vol.2006-HPC-107 No.87, pp. 299-304, (2006).
5) Linux manpage of mmap(2).
6) Linux manpage of madvice(2).
7) Linux manpage of TCP(7).
8) Bill A., Joe B., John B., Ann L. C., Ian F., Carl K., Sam M., Veronika N., Darcy Q., Steven T.: “Data Management and Transfer in High-Performance Computational Grid Environments,” *Parallel Computing Journal*, Elsevier, pp.749-771, (2002).

Vol.2010-HPC-126 No.24
2010/8/4

9) E Series TCP optimization products - FastSoft, Inc.
http://www.fastsoft.com/software-appliances/
10) Wei D., Jin C., Low H. S., Hedge S.: FAST TCP: Motivation, Architecture, Algorithms, Performance, *IEEE/ACM Transactions on Networking*, IEEE Press, pp. 1246-1259, (2006).
11) High-speed file transfer software – Aspera - fasp technology overview
http://www.asperasoft.com/en/technology/fasp_overview_1/fasp_technology_overview_1