オープンソフトウェアによるNetwork Attached Storageの性能の解析および改善に関する一試み

田胡和哉[†,☆] 根岸康[†] 奥山健一[†,☆☆]
村田浩樹[†] 松永拓也[†]

Linuxおよび Samba等の，いわゆるオープンソフトウェアを用いて実装された，NFS，CIFSプロトコルによるファイルサービス機能を持つ Network Attached Storage（NAS）の性能を，スループットを指標として解析した．実用的な条件でプロセッサネックになる場合があることが確認された．プロセッサが消費される原因として，単一のものとしてはファイルシステムとプロトコルスタックの間で行われるデータコピーが占める割合が最も大きく，全体の 20%以上を占めることが判明した．データコピーのオーバヘッドを解析すると，同一のデータに対して行われる複数回のデータコピーの回数を削減しても，完全にコピーを排除しないかぎり性能の向上は得られないことが測定された．実際にNFS，および，CIFSプロトコル向きに完全にコピーを排除した zerocopy 機構を実現することにより，10%以上の全体性能の向上が確認された．本稿では，性能向上の理由に関する解析，および，改善結果について述べる．

Performance Analysis and Improvement of Network Attached Storage Built from Open Software

Kazuya Tago,[†,☆] Yasushi Negishi,[†] Kenichi Okuyama,[†,☆☆]
Hiroki Murata[†] and Takuya Matsunaga[†]

The performance and causes of overheads of Network Attached Storage (NAS) which is built from open software components including Linux and Samba have been analyzed. The analysis showes that the processor is one of the major bottleneck sources. The overhead from copying payload data is found to be the largest single source of the processor overhead. We also found that reducing multiple times of data copy into single copy produces little performance gain, and we need to eliminate all of payload data copy by processor to get much larger performance gain. We built zerocopy mechanisms for both CIFS and NFS protocols by modifying Linux kernel and evaluated them. It has been shown that around 10% of improvement in throughput is attainable from the zerocopy mechanisms.

1. はじめに

ネットワーク利用の高度化にともない，種々の専用目的のサーバの利用がさらに広がっている．本稿では，その典型的なものの 1 つである Network Attached Storage（NAS）[7]の効率の良い実現方式について提案する．NASは，ネットワークに接続されたファイルサーバとしての機能と，バックアップ等のデータ管理のためのソフトウェアを統合し，利用者個々人がデータを個別に管理するより，全体としての運用費用（Total Cost of Ownership）を少なくすることを目的としている．それを実現する方法として，専用のハードウェアやオペレーティングシステムを用いる方法と，既存の PC に基づいたハードウェアと，Linux 等の汎用目的のオペレーティングシステムを用いる方法が利用されている．ここでは，後者の方法に焦点をあてる．

性能の面からみると専用の構成要素を用いた前者の方法が有利である．しかしながら，PC向けの汎用ハードウェアは，競争的な環境のもとで個々の部品の性能が急速な進歩をとげている．また，PC向けのソフトウェアは，いわゆる，オープン化が進み，これも技術的な進歩をとげているとともに，付加価値の高い有用なソフトウェアが多くの開発者によって次々開発され，無料で公開されている．したがって，汎用構成要素を

† 日本IBM
IBM Japan
☆ 現在，東京工科大学
Presently with Tokyo University of Technology
☆☆ 現在，フェニックス・テクノロジーズ
Presently with Phenix Technologies

用いた方法は，価格を低くできるとともに，機能が改善される早さの点では有利であると考えられる．

その一方において，PC 用の汎用のオペレーティングシステムでは，NAS のような，多くのクライアントからの要求を高いスループットで処理する，サーバとしての運用の目的に最適化されているとは必ずしも限らない．オープンソフトウェアの発展に追従し，今後開発されるソフトウェアがほぼそのまま利用できる汎用性が保たれる範囲内で，NAS としての適性の判定と，必要があれば，改善をほどこしてゆく必要がある．

本稿では，スループットを評価基準として，Linux オペレーティングシステムによって構成され，Network File System（NFS）[23]，および，Common Internet File System（CIFS）[22] ファイルアクセスプロトコルによるサービスを実現する NAS の性能を解析する．NAS の目的はできるだけ多くのクライアントに対してサービスを提供するものであり，たとえば，個々のファイルアクセスのレイテンシによる評価よりも，全体のスループットによる評価が，利用目的から見た重要度が高いことによる．これによって，接続可能なクライアントの数の目安が得られ，利用者による投資の基準とすることができる．

我々は，実際に PC を用いて種々の条件下で，NAS システムの性能を分析した．それによれば，ファイルアクセスプロトコルの種類やディスク装置の性能によっては実用的な条件下でもプロセッサネックとなりうること，プロセッサネックになった場合にはデータのコピーによるオーバヘッドが大きいこと，コピーによるオーバヘッドはコピー回数を削減するだけでは不十分で 0 回とする必要があること，等が明らかになった．また，Linux のシステムコールインタフェースを変えることなくこのコピーを除去し，性能を改善することができた．

データコピーが性能に有害であることは，古くから知られている事実である．また，我々の試みと前後して，特に Linux のファイルシステムや通信機構において，データコピーを削減する実装が種々試みられている [13]．しかしながら，Linux を用いた NAS の性能に関する総合的な解析や，性能改善の試みの有効性に関する検討は十分に行われているとはいいがたい状況である．以下において，性能改善の試みの有効性について解析を試みる．また，今後さらに性能を改善するための方策について検討を行う．

2. 性能の解析

ここでは，汎用の Linux オペレーティングシステムと PC ハードウェアを用いた NAS の性能について検討する．このようなシステムでは，多くの場合，NFS，および，CIFS によるサービスが利用されているので，議論はこの 2 つの場合に絞る．以下では，プログラムの解析による定性的な議論，巨視的なベンチマークによるボトルネックの同定，微視的な解析による改善箇所の同定を行う．

2.1 システムの動作

2.1.1 CIFS プロトコルによるサービスの実現

Windows オペレーティングシステム以外のオペレーティングシステムを用いて CIFS プロトコルのサービスを実装する際には，オープンソフトウェアである Samba プログラム [17] が利用される場合が多い．Samba は，通常のアプリケーションプログラムとして実装されており，オペレーティングシステムが提供するネットワーク通信機能とファイルアクセス機能を用いて，CIFS プロトコルによるファイルアクセスサービスを実現する．Samba は，実行時には，クライアントに 1 対 1 に対応するプロセスを生成し，クライアントごとの要求を個別に処理するユーザプログラムとして動作する．以下では，Samba による CIFS プロトコルのサービス全体を含めて CIFS と記すことにする．また，Samba によって起動されたユーザプログラムを Samba デーモンとよぶことにする．

Samba デーモンは，クライアントから，open，close，read，write 等のファイルアクセス要求が 1 回あるたびに，通常のソケットインタフェースを利用して要求のパケットを受信し，その内容に従ってシステムコールを発行して処理を実行した後に，返答のパケットを送出することを繰り返す．以下では，NAS のクラインとが NAS に対してファイル入出力を行うユーザデータをペイロードと総称する．

性能上，ボトルネックになるうる可能性の高いシステム資源として，ディスク装置とプロセッサがあげられる．

2.1.2 NFS プロトコルによるサービスの実現

NFS プロトコルの場合には，処理はすべてオペレーティングシステム核内で行われる．CIFS と比較した際の主な特徴は以下のようである．

(1) NFS の read，write 要求の処理にはシステムコールは利用されない．したがって，システムコールによるオーバヘッドは生じない．NFS プロトコルは，システム核内にある NFS デーモンプロセスによって実現される．NFS デーモンプロセスは，RPC 機構を用いてクライアントとの通信を実現している．

(2) NFS に特徴的な点として，プロトコル規定上，コミットライトが必要である点があげられる．すなわち，データ量のいかんにかかわらず，クライアントからの write 要求ごとに，メタデータ，ペイロードデータの双方の二次記憶装置への書き込みを完了することが求められており，ディスク装置への書き込み処理に関しては揮発性のキャッシュを利用することができない．これにより，ディスク書き込み処理の実行頻度は高くなりやすいことが予想される．ディスク装置は，このような頻繁なコミットライトを実行すると，著しく性能が低下する場合がある．

以下では，簡単のために，NFS プロトコルによるサービスの実現全体を含めて NFS と記すことにする．NFS は，V2 と V3 規格が現在利用されている．以下では，V2 規格を対象とする．

2.2 測定と解析

2.2.1 測 定 方 法

実際にシステムの測定を行うことにより，ボトルネックを同定し，その解消法について検討する．測定のためのベンチマークとして，NFS では SPECsfs [21),24)]，CIFS では NetBench [26)]が標準的に広く利用されている．これらはいずれも実際のアプリケーションが動作しているシステムの動作状況を解析しそれに対応した負荷パターンを生成するよう設計されており，実運用性能との乖離が少ない点，他の測定との比較が行える点等から，これらを利用した．

商用の NAS に近いハードウェア構成で測定を行う．具体的には，

- 500 MHz クロックの Pentium III プロセッサ
  512 Kbyte L2 キャッシュ250 MHz クロック駆動．
- Linux オペレーティングシステム
  Ver.2.2.14 を用いる．ファイルシステムとして，以下特にことわらない限り，EXT2 システムを用いる．
- SCSI インタフェースによって接続されたディスク群
  IBM 社製の 7,200 rpm 18 Gbyte SCSI ディスクを用いる．ディスク群制御機構において，不揮発性メモリによるキャッシュは使用しない．
- 1 Gbps のイーサネットインタフェース
  3COM 社製 3C985-B および Intel 社製 Pro1000 ハードウェアを利用した．
- 256 Mbyte のメモリ

からなるシステムを用いて測定を行った．サーバは，1 Gbps のイーサネットに接続されており，ブリッジと 100 Mbps のイーサネットを経由して複数のクライアントに接続される．NFS では Linux，CIFS では Windows クライアントを最大 40 台まで利用して測定を行った．

NFS の性能は，SPECsfs ベンチマークでは，クライアントからの要求によって生ずる RPC の，単位時間あたりの実行回数で計測される．CIFS の性能は，NetBench ベンチマークでは，入出力のスループットによって計測される．

ベンチマークの実行時に，システムの資源の消費状況を観測するためにいくつかのツールを作成した．たとえば，Pentium プロセッサは，クロックによって駆動される 64 ビットのカウンタを保持しており，カウンタの値をユーザモードであってもカーネルモードであっても参照することができる．これを用いることにより，プロセッサクロックの分解能を持つ計測タイマが実装できる．核内，および，ユーザアドレス空間内にカウンタ値を記録するログを設け，後にマージして集計することにより，処理の実行を時系列で詳細に追跡することができるようにした．

2.2.2 CIFS の解析

ベンチマークを実行してプロセッサ，および，ディスク装置のビジー率を計測する巨視的な測定によれば，プロセッサがボトルネックとなることが判明した．**図 1** に，ベンチマークを実行している際のプロセッサの利用率を示す．この図において，横軸は NAS に接続されているクライアントの数であり，縦軸は，プロセッサの利用率と，NAS 全体のスループットがプロットされている．NetBench ベンチマークは，クライアントの数に比例して NAS に対する処理要求が増加するように作られており，クライアントの数に全体のスループットが比例しないのは，NAS の能力が飽和していることを示す．この測定では，クライアント

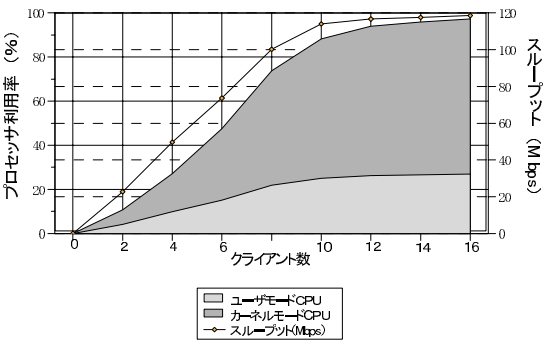

図 1 CIFS におけるプロセッサ消費
Fig. 1 Processor utilization for CIFS execution.

を 15 台接続し，総スループットが 120 Mbps になった場合に，プロセッサの利用率が 100%に近くなり性能が飽和している．

これに基づいて，ベンチマークを実行し微視的な解析を行った．プロセッサの消費量を，システムコールの種類ごとに分けた場合のそれぞれの総計，Samba デーモン，プロトコルスタック，ファイルシステム，データコピーに分けた場合のそれぞれの総計，および，各々の内訳について計測した．それによれば，単一の原因としては，データコピーによるものが最も大きいことが判明した．すなわち，全体のプロセッサ消費時間の約 4 分の 1 を，ファイルの read，write システムコール，および，ソケットの send，receive システムコールにそれぞれ 1 回ずつ含まれるデータコピーが占めている．

このほかの事実として，観測の過程において，データコピーによるプロセッサの消費，プロトコルスタックの実装全体によるプロセッサの消費の総計，および，Samba デーモン全体によるプロセッサの消費の総計がほぼ同程度であること，select システムコールの実装において不要に待ちループの実行回数が多く，実行全体の 7%程度のプロセッサ資源を消費すること，システムコールが最大で毎秒 31,000 回発行されていたがこれによるシステムコールの呼び出し機構のオーバヘッドは小さいこと，等が判明した．

2.2.3 NFS の解析

測定を行った条件下では，NFS ではディスク資源がボトルネックとなっていることが確認された．また，プロセッサの利用率はシステム性能に比例して増大しており，条件によっては，プロセッサ資源がボトルネックとなりうる可能性もあることが確認された．**図 2** に，プロセッサの利用率を示す．この測定では，1 GHz のクロックの CPU を用いている．CPU の利用率は線形に増大しているが，全体の性能は 4,300 ops./sec. の付近で飽和していることが観測された．プロセッサの消費は CIFS の場合と同様に，種々の原因によるが，データコピーによる部分が単一の原因として最も大きな部分を占めることが判明した．

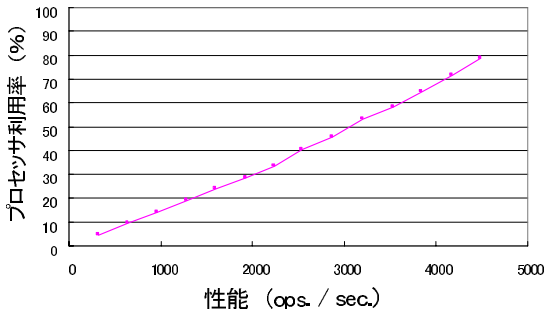

図 2 NFS におけるプロセッサ消費
Fig. 2 Processor utilization for NFS execution.

3. データコピーによるオーバヘッドの削減

3.1 データコピーによるオーバヘッドの性質

プロセッサによるオーバヘッドの大きな原因はデータコピーによるものであり，これを削減することを試みる．現状ではクライアントからのファイル入出力要求を処理するために総計で 2 回のデータコピーが行われている．たとえば，read 要求の処理では，ファイルキャッシュから Samba デーモンのバッファへのコピーが 1 回目，Samba デーモンのバッファからプロトコルスタックのバッファへのコピーが 2 回目に対応する．**表 1** に，CIFS の場合に，sendfile システムコールを利用して 2 回のデータコピーを 1 回に削減した際のオーバヘッドの削減状況を示す．削減の方法については 3.3 節で述べる．

表 1 は，クライアントにおいて NAS 上のファイル 200 Kbyte の読み込みを行ったものである．2 回のデータコピーの総計時間が 1,516 μsec．1 回に削減したデータコピーの総計時間が 1,363 μsec である．後者において，1 回目のデータコピーは，ディレクトリ名等の制御情報のコピーに対応しており，従来の 2 回コピー方式の 1 回目にも含まれていることから補正の意味でここに記してある．2 回目のコピーが，ファイルキャッシュからプロトコルスタックのバッファへの直接のデータコピーに対応している．

ここで明らかなように，データコピーを 2 回から 1 回に削減することによる効果はほとんどない．これは，プロセッサの L2 キャッシュによる効果であると考えられる．L2 キャッシュでは，ライトバック方式が用いられているために，同一のデータ転送命令であっても，実際には，1 回目の転送はメモリー > キャッシュの転

表 1 コピーの所要時間
Table 1 Execution time of data copy.

	1 回目の所要時間 (μsec.)	2 回目の所要時間 (μsec.)	総計 (μsec.)
従来の 2 回コピー式	823	693	1,516
sendfile による 1 回コピー式	66（制御情報）	1,297	1,363

送，2回目の転送はキャッシュー > メモリの転送が行われている．これが，1回に削減されると，メモリー > キャッシュの転送が行われ，キャッシュにあるデータはその後除々にメモリに書き戻される．書き戻しが起こる間，システムに対するオーバヘッドが生じることにより，全体として，2回の転送が行われている場合と比較してほとんど差が生じないことになる．実際に，1回に削減したシステムでNetBenchを実行したが，測定誤差の範囲内でデータコピー2回のオリジナル版と同一性能値であった．

Sambaによる処理の場合，2回のデータコピーの間データがL2キャッシュに残っている程度に隣接した間隔でデータコピーが行われていることになる．2回のデータコピーの間には多量のデータの移動が行われておらず，また，コンテクストスイッチが起こる可能性も低いことが確認されているので，上記の理由によってコピー回数を1回削減するだけでは十分な性能改善が得られなくなっているものと推測される．

この事実により，データコピーによるオーバヘッドを削減するためには，少なくとも，プロセッサによるデータコピーを完全に排除する必要があることが判明した．

3.2 zerocopy 機構の実現

NASにおいて，プロセッサによるユーザデータのコピーをすべてなくすためにシステムに追加された機構を，zerocopy機構とよぶことにする．クライアントからのwrite要求の処理をzerocopy化することはシステムの変更が大規模になりすぎるので，read要求処理に限ったzerocopy機構を検討することにする．すなわち，write要求処理では，サイズを予測することができない複数の受信パケットを，コピーせずに1つのファイルブロックにまとめあげる必要があるのに対して，read要求処理ではあらかじめサイズが確定している1つのファイルブロックを複数の送信パケットに分割すればよいので，はるかに容易に実現できる．以下では，read要求処理のzerocopy機構を単にzerocopy機構と記すことにする．

Linuxを対象として，CIFS，および，NFSのzerocopy機構を実現した．我々の試みと同時期にzerocopy機構実現の試みが種々行われており[13]，オープンソフトウェアとして入手可能であるので，ここでは動作を理解するために必要な，複数の実現に共通の技術項目についてのみ記述する．

(1) **CIFS用zerocopy機構**

ディスク上のユーザデータをネットワークに送出するためには，sendfileシステムコール[10]を用いるようにSambaの実装を変更する．sendfileは，ファイシステムとネットワークの間で直接データを転送する機能を提供している．このシステムコールを用いると，CIFSを実現しているSambaデーモンの階層からみてデータコピーなしにユーザデータの送出が行われるようになる．しかしながら，オペレーティングシステムの核内では，sendfileの実装において，プロトコルスタックが用いるバッファとファイルシステムのキャッシュの間で1回のデータコピーが行われている．このため，この変更によって2回のデータコピーが1回に削減されたことになる．

さらにデータコピーを削減して，zerocopyとするためには，プロトコルスタックのバッファとファイルキャッシュのペイロードの保持方法を変更すること，ファイルシステムで割り当てられたバッファの開放をプロトコルスタック内で行えるようにすること，例外処理が相互に影響しないようにすること，等のオペレーティングシステムの実装に対する変更が必要になるので，これを実施した．

ペイロードの保持方法として，具体的には，以下の方法をとった．データ全体のうち物理的に連続している部分領域を，そのページ番号，ページ内のオフセット，サイズの3つ組みによって表現し，複数の部分領域の集合によって，1つのバッファを表現するようにした．通信パケットとディスクブロックの間ではデータの蓄積方法が異なるが，これによってデータ領域を単一の方法で表現することが可能になり，ネットワークインタフェースハードウェアのDMAのギャザ機構を用いて1回で物理入出力を行うことができるようになる．

zerocopy機構の実現において必要なもう1つの項目として，TCPパケットに付加することが義務付けられているチェックサムの生成がある．従来は，プロセッサによってデータをコピーするのにあわせてプロセッサによってチェックサムの計算を行っていたが，この処理がなくなるのにともなって，通信ハードウェアによってチェックサムを生成し，TCPパケットに記入してから送出するように制御ソフトウェアを変更した．チェックサムをハードウェアによって計算する機能は，すべての通信ハードウェアに実装されているわけではないが，1 Gbpsのイーサネットインタフェースハードウェアでは実装される場合が多い．今回利用した通信ハードウェアにはいずれも実装されている．

以上の変更により，ファイルキャッシュにあるペイロードが，プロセッサによってコピーされることなくそのまま通信ハードウェアによってネットワークに送出されることになる．

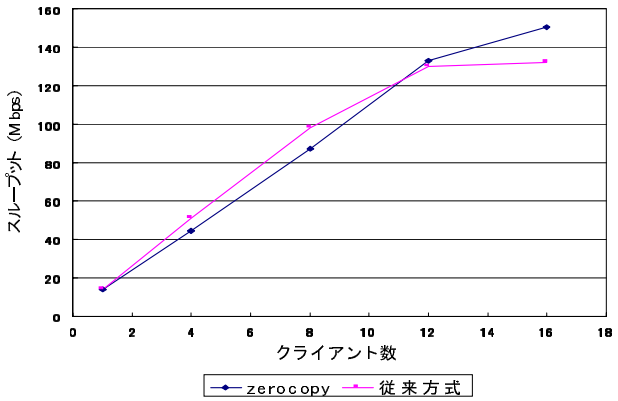

図 3 zerocopy による CIFS 性能の改善
Fig. 3 Improvement of CIFS performance by zerocopy with executing NetBench.

(2) **NFS 用 zerocopy 機構**

NFS の実装では，RPC 機構の内部で引数のマーシャリングを行う際にデータコピーがおこる．これも，sendfile の zerocopy 実装と同様にプロトコルスタックが用いるデータバッファと RPC の引数バッファの間で同一のペイロードデータの保持方法をとるように変更し，データコピーを削除した．この変更は，RPC の実装を変更することによって行われ，その仕様は変更されていない．

3.3 評　　価

(1) **CIFS 用 zerocopy 機構の効果**

CIFS 用 zerocopy 機構を用いた場合と用いない場合のシステム性能の比較を**図 3** に示す．図 1 と同様に，横軸はクライアント数であり，縦軸はすべてのクライアントに対するサービスによる総計のスループットを示している．プロセッサ資源が飽和する 15 クライアント付近で，最大スループットが 12%改善している．Miller は，我々と独立に，sendfile システムコールによる read 要求処理の zerocopy 化を実現した[13]．Miller は，実装を公開しているが，性能の改善が図られる理由の解析は行っていない．彼らの実装についても評価を行ったところ，測定誤差の範囲内で同一の性能を得た．本稿で行った議論は，そのまま適用可能であると考えられる．Miller は，NFS の zerocopy 化は行っていない．

(2) **NFS 用 zerocopy 機構の効果**

NFS 用 zerocopy 機構を用いた場合，CPU の利用率が 10〜15%軽減されることが観測された．

さらに，LAN で連結された複数のサーバノード上で単一のファイルイメージを実現する機能を持つファイルシステムである GPFS[18]を用いて NFS サービスを実現し，SPECsfs 値を測定した．Pentium III 1 GHz プロセッサ 2 個による SMP 構成，2 Gbyte メモリ，

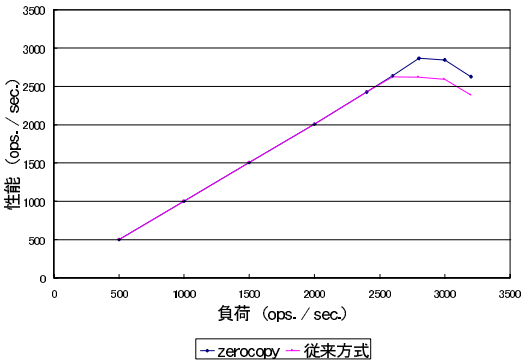

図 4 zerocopy による NFS 性能の改善
Fig. 4 Improvement of NFS performance by zerocopy with executing SPECsfs.

3COM イーサネットインタフェース，IBM 社製 ProFibre Storage Array RAID，SCSI ディスク 24 個の構成で行った計測では，2,630 ops./sec. が 2,870 ops./sec. に，9.1%性能が改善されることが測定された．測定結果を，**図 4** に示す．横軸は，ベンチマークプログラムが発生したファイルアクセスの要求量であり，縦軸は実際に得られたスループットである．GPFS は，サーバノードを複数用いてスケーラブルなスループットを実現できる一方，サーバノード間のファイルキャッシュコヒーレンシを維持するために，EXT2 等の単一のノード用の既存ファイルシステムと比較すると，ノードあたりより多くのプロセッサ資源を消費していることによる．複数ノード構成の NAS を構築する際には，NFS に対しても zerocopy 技術が有効であることが推察される．

4. 議　　論

オープンソフトウェアを用いた NAS の性能の改善

方式全体について概観し，zerocopy 技術をこの観点から位置づけてみる．オープンソフトウェアの性能改善の技術の妥当性を判定するにあたっては，その効果のみならず，既存部分との親和性の良否も考慮に入れる必要がある．たとえば，Linux で動作する NFS プロトコルのサーバや Samba はカーネルの開発とは別個に行われており，それらの間のインタフェースが大きく変更されるような改良は個々の開発者たちに受け入れられにくい．ファイルシステムやプロトコルスタック等のシステム要素間や，システムコールのインタフェース仕様の変更必要性の程度と，変更によって得られる効果の双方を視野に入れた判断がきわめて重要になってくる．

オープンソフトウェアを用いた NAS 全体の性能に関して解析を行った例は多くないが，それを構成する個々の要素の性能に関しては，きわめて多くの研究がなされている．具体的には，

(1) ファイルシステムのアクセススループットやレイテンシ向上の試み[6),8),16),19),25)]，
(2) CIFS，NFS 等のファイルアクセスプロトコルの仕様改善[12)]，
(3) 基盤となるハードウェアやオペレーティングシステムの一般的な性能の改善，

があげられる．本稿では，(3) に焦点をあてた．

基盤となるハードウェアやオペレーティングシステムの性能の改善の試みのなかでも，特に NAS と関連が深いと考えられるものは，

(1) DMA やコピーによるデータの移動経路，すなわち，データパスの効率改善，
(2) プロトコルスタック等システム機能の一部の専用ハードウェアへのオフローディング，
(3) 割込みやシステムコールによって実現される，システム制御機構の実装効率の改善[5)]，

がある．

オープンソフトウェアを用いた NAS の動作においても，アプリケーションプログラム自体の実行による負荷は比較的軽微で，ファイルシステムおよび通信システムに過大な負荷が生ずる．このようなシステムの動作形態は，ほかにも，Web サーバ等ネットワーク用サーバではよく見られるものである[11),14),20)]．このような負荷におけるシステム性能は，データパスの構造や性能と深く関連している．データパスの改善は，物理入出力の方式に関連するものと，システムソフトウェアの構造に関連するものがあり，ここでは，後者を試みた．

UNIX や Linux のファイルシステムおよび通信システムは，オペレーティングシステム核内でバッファリングを行い，アプリケーションプログラムにはその内容をコピーして渡す，いわゆるコピーセマンティクスのインタフェースを基本として構成されている．これによって，アプリケーションプログラムの作成が容易化される一方，zerocopy 技術を一般的な方法で導入することは難しくなる．UNIX の zerocopy 技術に関して，90 年代の中葉から多くのことが試みられてきた．はじめに，ネットワークの高速化にともなって通信機構の zerocopy 化が試みられた．たとえば，DMA によって主記憶にデータを転送するかわりに，通信ハードウェア上にメモリを用意し，アプリケーションがそれに直接アクセスすることによって zerocopy を実現することが試みられた[4)]．これらの経験をふまえて，最近の通信ハードウェアでは，DMA におけるギャザスキャッタ機能等を装備して高能率なデータ転送が行えるように配慮されている．また，最近では，特に Web サーバの実現性能に関連して，ファイルシステムとプロトコルスタックを結合した系全体に関して zerocopy を実現することが試みられている[3)]．その主なものは，仮想記憶やシステムコールに関する新たなインタフェースを定義して統一的な方法でデータコピーを除去しようとするものである．ファイルマップを拡張する方法[3)]や新たなデータ構造を実現する方法[15)]が提案されているが，インタフェースの変更による影響が大きく，広く用いられる最終的な解を得るには至っていないのが現状である．

本稿でも，同一の問題領域に関して検討を行った．しかしながら，ここでは，新規のシステムインタフェースを導入することはせず，既存の sendfile システムコールの実装方式を改善することのみで，所期の目的を達成している[14)]．NAS の実装に問題を限れば，新たなシステムコールインタフェースは必要ないことが分かる．

物理入出力に関連したデータパスの効率の改善には，周辺機器とメモリの間のデータ移動に関して単なる DMA とは異なる新たな機構が必要になる．代表的なものとして，InfiniBand[9)]があげられる．この規格により，入出力の種類ごとに事前に指定した論理アドレスへ，ソフトウェアの関与なしにデータを転送することができる．これにより，たとえば周辺機器からユーザの論理アドレス空間に直接データを転送することを可能にしている．

ネットワークプロトコルの処理を専用のハードウェアによって実現することも行われている．たとえば，Alacritech 社は，TCP/IP プロトコルの処理をすべて

周辺機器内で実行する製品[2]を開発している．このような装置の効果は，プロセッサの負荷を軽減することが大きい．また，ディスク装置に関しても，装置自体にファイルシステムを実装したり[1]，そのインタフェースを変更して，より多くの機能をディスク装置自体が分担するようにしたりする規約が検討されている．2章での測定によれば，ネットワークプロトコルの実装のオフロードの効果が大きいことが予想される．

このとき，周辺機器への機能のオフロードは，zero-copy と密接な関係にある点に配慮が必要である．我々の測定によれば，NAS の負荷パターンでは，TCP/IP のプロトコル処理に要する時間と，そこで転送されるペイロードデータをメモリ間でコピーする時間とはほぼ同水準にある．したがって，機能のオフロードによって本体プロセッサと周辺機器の間の機能分担が従来とは変わるにもかかわらず，たとえば DMA 等の既存の物理入出力機構やシステムコールのインタフェースを維持する目的で，データの交換においてプロセッサによる新たなコピーを行わなければならない場合には，性能改善の程度は限定されたものとなってしまうことが予想できる．

オープンソフトウェアを用いた NAS の性能改善に関する方針をまとめてみる．

(1) プロセッサによるデータコピーをすべて排除する，zerocopy 化を図ることは有効である．
(2) zerocopy 機構を実現するためには，NAS を実現する観点からは，あらたなシステムコールを設ける等のシステムインタフェースの拡張は必要ない．また，システムコールを実現するためのオーバヘッドが少ないこと等の理由によりシステム全体の構造を大きく変更する必要性も少ないことが分かる．オペレーティングシステムの実装の改善の範囲内でも十分有効な結果が得られる．たとえば，select システムコールの実装の改善は考慮に値する．
(3) さらに性能の改善を図るためには，プロトコルスタックのオフローディングが有効な手段であると考えられる．ただし，そのためには，同時に zerocopy 機構が維持されなければならず，物理入出力機構のインタフェースの妥当性について再度検討する必要がある．

5. おわりに

本稿では，Linux と Samba によるオープンソフトウェアを用いた NAS の性能について議論を行った．ファイルシステムと通信機構の負荷が大きく，実用的な条件でプロセッサネックになる場合があることが確認された．さらに，プロセッサによるデータコピーの性能に与える影響について解析を試みた．それによれば，データコピーによるオーバヘッドは，単一の原因によるプロセッサのオーバヘッドとしては最も大きなものであり，その削除によってシステム性能が 1 割以上改善できることが確認できた．また，データコピーの回数は 0 にすることが必要であり，0 回にならない削減では効果がきわめて限定されること，Linux の既存のシステムコールの実装を変更することによってこのような削減が可能であること，等が判明した．

今後，新たなシステム構成の変化にともなう改善効果について検討を続けていく予定である．

参考文献

1) Acharya, A., Uysal, M. and Saltz, J.: Active Disks: Programming Model, Algorithm and Evaluation, *ACM 8th ASPLOS*, pp.81–91 (1998).
2) Alacritech: http://www.alacritech.com （Alacritech 社のホームページ）.
3) Brustoloni, J.C.: Interoperation of Copy Avoidance in Network and File I/O, *IEEE INFOCOM 1999*, pp.534–542 (1999).
4) Edwards, A., et al.: User-space protocols deliver high performance to applications on a low-cost Gb/s LAN, *USENIX OSDI 1996*, pp.14–23 (1996).
5) Farrow, R.: Linux 2.5 Kernel Developers Summit, *;login:*, Vol.26, No.3, pp.5–11 (2001).
6) Ganger, G.R., et.al.: Soft Updates: A Solution to the Metadata Update Problem in File Systems, *ACM TOCS*, Vol.18, No.2, pp.127–153 (2000).
7) Gibson, G.A. and Van Meter, R.: Network Attached Storage Architecture, *Comm. ACM*, Vol.43, No.11, pp.37–45 (2000).
8) Hagmann, R.: Reimplementing the Cedar File System Using Logging and Group Commit, *ACM 11th SOSP*, pp.155–162 (1987).
9) InfinBand Trade Association. http://www.infinibandta.org（InfiniBand のホームページ）.
10) Linux Consortium. http://www.kernel.org/ （Linux のホームページ）.
11) Maltzahn, C., Richardson, K.J. and Grunwald, D.: Reducing the Disk I/O of Web Proxy Server Caches, *USENIX 1999 Annual Technical Conference*, pp.225–238 (1999).
12) Martin, R.P. and Culler, D.E.: NFS Sensitivity to High Performance Networks, *ACM SIG-*

METRICS, pp.71–82 (1999).
13) Miller, D.S.: http://ftp.kernel.org/pub/linux/kernel/people/davem/ZEROCOPY（Linux Zerocopy 実装のホームページ）.
14) Nahum, E., Barzilai, T. and Kandlur, D.D.: Performance Issues in WWW Servers, *IEEE/ACM Trans. Networking*, Vol.10, No.1, pp.2–11 (2002).
15) Pai, V.S., Druschel, P. and Zwaenepoel, W.: IO-Lite: A Unified I/O Buffering and Caching System, *USENIX OSDI 1999*, pp.15–28 (1999).
16) Rosenblum, M. and Ousterhout, J.: The Design and Implementation of a Log-Structured File System, *ACM TOCS*, Vol.10, Issue 1, pp.26–52 (1992).
17) Samba Team: http://samba.org（Samba のホームページ）.
18) Schmuck, F.B. and Haskin, R.L.: GPFS: A Shared-Disk File System for Large Computing Clusters, *USENIX FAST*, pp.231–244 (2002).
19) Seltzer, M.I., et al.: Journaling versus Soft Updates: Asynchronous Meta-data Protection in File Systems, *USENIX Annual Technical Conference*, pp.71–84 (2000).
20) Shriver, E., et al.: Storage management for web proxies, *USENIX 2001 Annual Technical Conference*, pp.203–216 (2001).
21) Performance Evaluation Corp.: http://www.specbench.org/osg/sfs97r1/（SPECsfs のホームページ）.
22) Storage Networking Industry Association: http:www.cifs2002.org/（CIFS 関連情報）.
23) Sun Micro Systems: RFC-1094（http://www.faqs.org/rfcs/rfc1094.html から参照可能）
24) Wittle, M. and Keith, B.E.: LADDIS: the Next Generation in NFS File Server Benchmarking, *Summer 1993 USENIX*, pp.111–128 (1993).
25) Zhou, Y., Philbin, J.F. and Li, K.: The Multi-Queue Replacement Algorithm for Second Level Buffer Caches, *USENIX Annual Technical Conference*, pp.91–104 (2001).
26) Ziff Davis Media: http://www.netbench.com（NetBench のホームページ）.

(平成 14 年 5 月 24 日受付)
(平成 14 年 12 月 3 日採録)

田胡 和哉（正会員）

1986 年筑波大学大学院工学研究科博士課程修了．工学博士．筑波大学電子情報工学系助手，東京大学工学部助手，日本 IBM 東京基礎研究所を経て 2002 年より東京工科大学片柳研究所助教授．オペレーティングシステムの構成方式に興味を持つ．1984 年情報処理学会論文賞．ACM 会員．

根岸 康（正会員）

1987 年東京工業大学理学部情報科学科卒業．1989 年同大学院修士課程修了．同年日本アイ・ビー・エム（株）入社．現在同社東京基礎研究所所属．コンピュータネットワーク，通信プロトコルに関する研究に従事．ACM 会員．

奥山 健一（正会員）

1989 年東京工業大学工学部制御工学科卒業．1991 年同大学院総合理工学研究科修士課程修了．同年日本アイ・ビー・エム（株）入社．現在，フェニックス・テクノロジーズ（株）勤務．

村田 浩樹（正会員）

1987 年早稲田大学理工学部電子通信学科卒業．1989 年同大学院修士課程修了．同年日本アイ・ビー・エム（株）入社．現在同社東京基礎研究所所属．グリッドに関する研究に従事．電子情報通信学会，ACM，IEEE 各会員．

松永 拓也（正会員）

1994 年東京大学工学部システム量子工学科卒業．1996 年同大学院情報工学専攻修士課程修了．同年日本アイ・ビー・エム（株）入社．現在同社大和ソフトウエア開発研究所所属．