iSCSI を用いた IP ネットワークストレージシステムのトレース解析

山口 実靖† 小口 正人‡ 喜連川 優†

†東京大学 生産技術研究所
‡お茶の水女子大学 理学部 情報科学科

要 旨

本稿では iSCSI を用いた IP-SAN のアクセストレースシステムの提案と, それを用いた性能向上に関する考察について述べる. FC-SAN の欠点を補う SAN として IP を用いる IP-SAN や iSCSI が期待を集めるようになっているが, IP-SAN は性能が FC-SAN より劣るという問題点も指摘されている. IP-SAN は多段のプロトコルで構成され, かつ サーバ計算機とストレージ機器が協調して動作する分散システムとなっている. よって, これらの統合的な解析の実現が重要である. 本稿では, 我々の実装した統合的なトレースシステムについて説明を行い, それを実際に高遅延環境における並列 iSCSI アクセスに適用し性能劣化原因の発見および問題の解決により並列 iSCSI アクセスの性能を向上が可能であることを示す.

Trace Analysis of iSCSI-based IP Network Storage System

Saneyasu Yamaguchi† Masato Oguchi‡ Masaru Kitsuregawa†

†Institute of Industrial Science, The University of Tokyo
‡Dept. of Information Sciences, Faculty of Science, Ochanomizu University

Abstract

In this paper, we propose an IP-SAN access trace system and present performance improvement using the system. IP-SAN and iSCSI are expected to remedy problems of FC-based SAN. In IP-SAN systems using iSCSI, servers and storages work cooperatively, thus integrated analysis of servers and storages can be considered important. We explain our integrated trace system and show that the system can point out the cause of performance degradation.

1 はじめに

計算機システムが扱うストレージの大容量化に伴いストレージ管理費用が増大し, これが大きな問題の一つとなっている [1, 2]. この問題に対する解決策の一つとして, SAN (Storage Area Network) の導入によるストレージ集約が提案された. 計算機群が使用するストレージを一箇所に集約して配置し各計算機がネットワーク (SAN) 経由でこれを使用する運用手法を取ることにより, ストレージ管理費用は大幅に削減される. この効果は高く評価されておりすでに多くの企業で SAN が導入されている. しかし, FC を用いる現世代の SAN ("FC-SAN") は, (1)FC 管理技術者が少ない, (2)FC は接続距離に限界がある, (3)FC 導入費用は高い, (4)FC の相互接続性は必ずしも高くない, などの問題も指摘されており, Ethernet と TCP/IP を用いて構築する IP-SAN が 次世代 SAN として期待を集めている. IP-SAN は, IP 技術者が多い, 接続距離に限界がない, 導入費用が低い, 相互接続性が高いなどの利点が期待されている. 逆に欠点としては, 性能が FC-SAN より低い, サーバ計算機の CPU 使用率が高くなる, などが指摘されている [3, 4, 5]. IP-SAN は LAN 環境では FC-SAN に匹敵する性能を提供できるとも言われる [6, 7] が高遅延環境下における性能

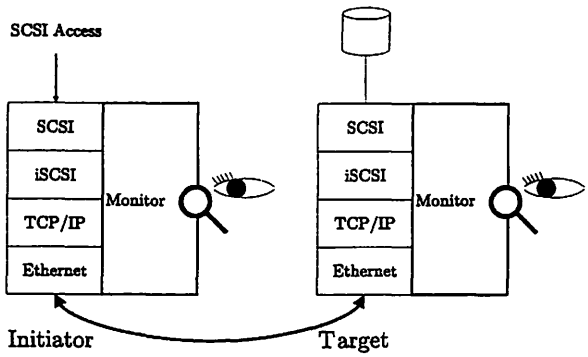

図 1: iSCSI プロトコルスタックと解析システム

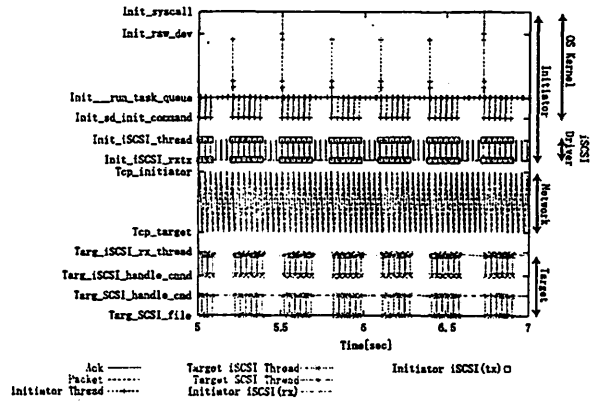

図 2: iSCSI アクセスのトレース

の劣化が大きく [1, 8], この解決は非常に重要であると考えられる. そこで, 本稿ではトレースシステムを用いた IP-SAN の性能向上手法を提案する. そして, 実際に高遅延環境下における IP-SAN システムに対して提案システムを適用し, その有効性を示す.

IP-SAN 用のデータ転送プロトコルとしては iSCSI[9, 10] が 2003 年 2 月に IETF[9, 10, 11] に承認され, これが標準的なデータ転送プロトコルとなっている. そこで本稿では iSCSI ストレージアクセスのトレースシステムについて述べる. iSCSI では, SCSI プロトコルを TCP プロトコルの中にカプセル化し IP ネットワーク上で転送するブロックレベルのプロトコルである. 多くの場合, 物理層, トランスポート層には Ethernet が用いられ, 代表的なプロトコルスタックは図 1 の様に SCSI over iSCSI over TCP/IP over Ethernet となる.

本稿は以下の様に構成されている. まず, 第 2 章において "IP-SAN トレースシステム" を提案する. 次に, 第 3 章において提案システムを実際に IP-SAN システムに対し適用し, その有効性を示す. そして, 第 4 章で関連する研究の紹介を行う. 最後に, 第 5 章において本稿のまとめを述べる.

2 統合トレースシステムの提案

本章では本稿で提案する "IP-SAN アクセストレースシステム" について説明を行う. iSCSI ストレージアクセスは個別に動作するサーバ計算機とストレージ機器が協調して実行されるため, この双方の振る舞いを統合的に解析することが重要であると考えられる. また iSCSI ストレージアクセスは多段のプロトコルで構成されるが, 全層を経由して行われるため全層が性能劣化原因となる可能性があり, 性能向上について考察するにはこれら全層を網羅的に観察する必要がある. そこで図 1 の様に iSCSI プロトコルスタック全層の振る舞いを観察できるモニタシステムを構築し [12], サーバ計算機とストレージ機器において個別に記録された iSCSI ストレージアクセスのトレース記録を統合的に解析することを可能とするトレースシステムを構築した. 実装には, オープンソースである Linux OS(ver. 2.4.18) とニューハンプシャー大学 InterOperability Lab[13] が提供する iSCSI reference implementation (ver. 1.5.02) [14] を用い (以下この iSCSI 実装を "UNH" と呼ぶ), これらのソースコードにモニタ用コードを適用することにより解析システムを実現させた.

図 2 に, 当システムを用いて iSCSI シーケンシャルアクセスのトレース結果を可視化した例を示す. 同図の縦軸は iSCSI アクセスのプロトコルスタックの遷移を表している. すなわち, 上から順に, (1) アプリケーションによるシステムコールの発行, (2)raw デバイス層, (3)SCSI 層, (4)iSCSI 層, (5)TCP/IP 層, (6)Ethernet によるパケットの転送, (7)TCP/IP 層, (8)iSCSI 層, (9)SCSI 層, (10)HDD デバイスアクセス, を表している. (1)〜(5) が, サーバ計算機内における処理であり, (7)〜(10) がストレージ機器における処理である. 本トレース例では, ファイルシステムを用いずに raw デバイスを用いた. また使用した iSCSI ターゲットは "ファイルモード" で動作させたため最下位層の HDD デバイスアクセスは実際はファイルアクセスのトレースとなっている. 横軸が時間の経過を表しており, iSCSI ストレージアクセスの各層における各処理の消費時間等を視覚的に確認することが可能となる. また, 大きなブロックサイズのシステムコールが各層で細分化されている様子や, 待ち状態にある処理の把握なども可能となる. 同図の例では 2MB のシステムコールが発行されており, これを raw デバイスが 512KB ごとの 4 要求に分割し, 4 要求が完了した時点で上位層にシステムコールの完了を通知していることや, raw

デバイスから発行された 512KB の要求が 32KB の SCSI 命令に分割されていること, 細分化された要求を用いてネットワークに処理要求が送出されている様などを観察することが可能である.

3 適用による提案システムの評価

本章では提案トレースシステムを実際に高遅延環境下における並列 iSCSI ショートブロックアクセスに対し適用し, その有効性を示す.

3.1 並列アクセス実験

iSCSI イニシエータ (サーバ計算機) と iSCSI ターゲット (ストレージ機器) を Gigabit Ethernet で接続し IP-SAN 環境を構築し性能測定実験を行った. イニシエータとターゲットの間には人工的な遅延装置として FreeBSD Dummynet[15] を配置し擬似的な高遅延環境を実現した. iSCSI 実装としては, UNH 実装を用いた. iSCSI プロトコルの振る舞いの影響とストレージデバイスの振る舞いを明確に分離するために, ターゲットはファイルモードで動作させ, ファイル内容がメインメモリにキャッシュされている状態で計測を行った. イニシエータ, ターゲット PC の仕様は下記の通りである. CPU は Pentium4 2.8GHz, メインメモリは 1GB, OS は Linux 2.4.18-3, NIC は Intel PRO/1000 XT Server Adapter(Gigabit Ethernet Card). 遅延装置の PC の仕様は下記の通りである. CPU は Pentium4 1.5GHz, メインメモリは 128MB, OS は FreeBSD 4.5-RELEASE, NIC は Intel PRO/1000 XT Server Adapter.

本実験環境において以下ような並列ショートブロックアクセス実験を行った. サーバ計算機からストレージ機器に対して iSCSI 接続を確立し, その raw デバイスに対してシーケンシャルにショートブロックアクセス (raw デバイスに対してシステムコール "read()") を行うベンチマークプログラムを作成し, 同プログラムを複数プロセス同時に起動し全プロセスの合計性能を測定した. ブロックサイズは 512 バイトとし各プロセス 2048 回システムコールを発行した. 片道遅延時間は 16m 秒とした.

3.2 実験結果

解析実行前の初期状態において上記の実験を行い図 3 の "can_queue=2, queue_depth=8" の結果を得た. 横軸が並列に動作させたベンチマークプロセスの数である. 縦軸は合計性能を表し, 単位時間における全プロセスの合計トランザクション数である (トランザクション数は "512 バイトのシステムコールの回数" を表す). シングルプロセス時の性能は 31.0 [Trans/sec] であり, これは往復遅延時間 0.032 秒の逆数とほぼ等しく期待通りの性能が得られている. 同様に, 2 プロセス並列時の性能も 62.0 [Trans/sec] となり 2 プロセス合計でほぼ 2 倍の性能が得られていることが確認された. これらに対し並列プロセス数を 3 以上に向上させても合計性能は 2 並列時と同程度となり, 多段プロトコルスタックで構成される iSCSI ストレージアクセスのいずれかの層で並列動作数を 2 に制限していると考えられる.

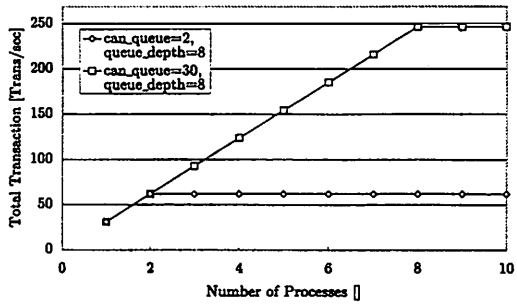

図 3: 実験結果 A : 並列 I/O の合計性能

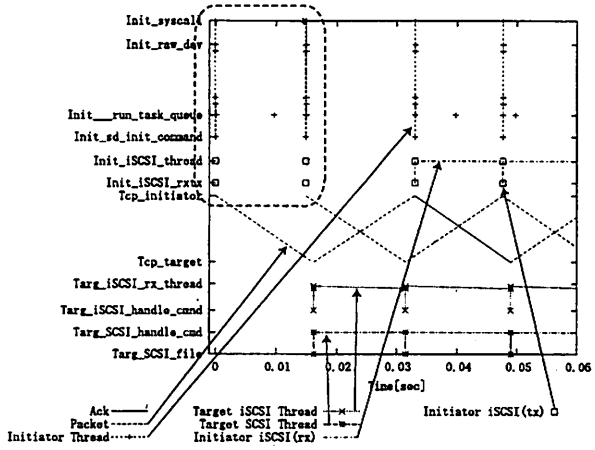

図 4: 並列アクセスのトレース図 (default) : A

3.3 並列アクセスのトレース

前節の 3 プロセス並列アクセス実験のトレースを可視化し図 4 を得た. "Initiator Thread" が切断して記してあるのは OS のプロセススケジューラのコンテキストスイッチが発生しプロセスが停止/再開したことを表している (同様の理由で 0.010 秒, 0.040 秒などにおいてトレースがプロットされている). 同

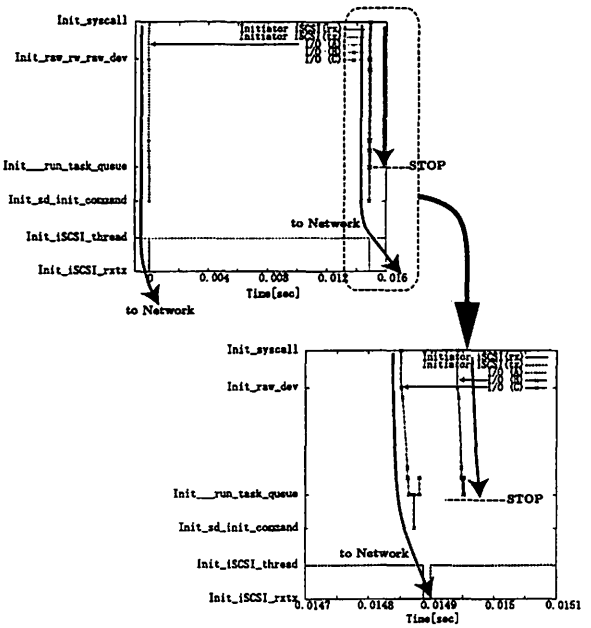

図 5: 並列アクセスのトレース図 (default) : B

図よりイニシエータ (サーバ計算機) からターゲット (ストレージ機器) に対し 1 往復時間に内に同時に 2 個の I/O 要求しか送出されておらず, 並列度制限はイニシエータ側に存在すると予想される.

次に並列度を制限している部分の巨視的な解析を行う. 図 4 左上波線部を拡大し可視化したものを図 5 の左上に記す. そして, その波線部の拡大図を同図右下に記す. 同図では, 並列して動作している 3 個の I/O 要求を "I/O(A)", "I/O(B)", "I/O(C)" と別の線として記した.

図 5 左上より I/O(A), (B), (C) は順に時刻 0.000 秒, 0.015 秒, 0.015 秒にシステムコールを発行していることが確認できる. ターゲットからの応答を待たずに 1 往復時間 (32ms) 内に 3 個のシステムコールが発行されていることからシステムコールの発行において並列性が制限されていないことが分かる. また, "I/O(A)" のトレースより "I/O(A)" の要求は raw デバイス層, SCSI 層, iSCSI 層, TCP/IP 層を経由し実際にネットワークに送出されていることも確認できる. 次に, 同図右下より "I/O(C)" のシステムコールは時刻 0.01485 秒に発行され, 各層を経由し要求はネットワークに送出されていることが確認できる. これに対し "I/O(B)" のシステムコールは時刻 0.01494 秒に発行され, raw デバイス層を経由し raw デバイスによる命令は発行されているが SCSI 層における SCSI 命令の発行に到っていないことが確認でき, SCSI 命令の同時発行上限が 2 となっていると予想できる.

次に, 並列度制限箇所の微視的な解析を行う. SCSI 命令が即時に発行される最初の 2 要求 (I/O(A), I/O(C)) と命令の発行が拒否される 3 個目の要求 (I/O(B)) のトレース分岐点は SCSI 層である図 6 の Linux カーネルの "drivers/scsi/scsi_lib.c" 部である. 同実装箇所は現在アクティブ (処理途中) である命令数 host_busy[1] と, 下位層 (本例では iSCSI ドライバ) が同時に受け付け可能なアクティブな SCSI 命令数 can_queue[2] の比較部となっている. UNH iSCSI 実装において "can_queue" の初期値は 2 となっており, アクティブ命令数 host_busy は 0 から開始される. 最初の 2 要求 (I/O(A),I/O(C)) のトレースでは host_busy はそれぞれ 0, 1 となっており図 6 における "host_busy<can_queue" に示される動作が記録された. すなわち, 同図 914 行目において host_busy をインクリメントし, 1046 行目において SCSI 命令が発行される動作が記録された. 3 個目の要求 (I/O(B)) においては host_busy が 2 であり "host_busy>=can_queue" に示される様に SCSI 命令が発行されない動作が記録された. 以上の微視的なトレース解析により, 並列度を制限し性能向上を妨げている原因は iSCSI ドライバにおける "can_queue" の値が不十分であることだと考えられる.

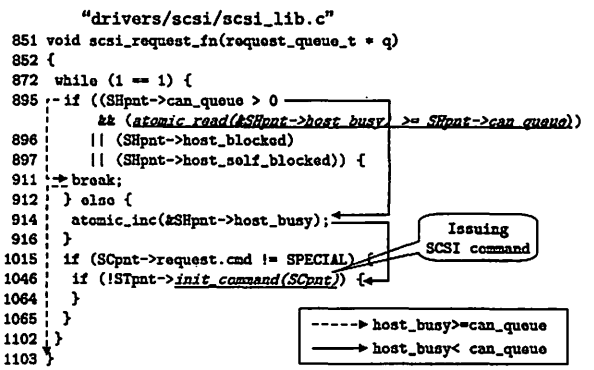

図 6: Linux SCSI 層のトレース: "drivers/scsi/scsi_lib.c"

3.4 発見された問題点の回避

前節の解析から高遅延環境における並列アクセスの性能を向上させるには "can_queue" の値を増加させることが重要であると考えられる. これを十分大きな値 (後述の理由により 8 以上) である 30 に増加

[1]Linux "drivers/scsi/hosts.h" にて "commands actually active on low-level" と解説されている

[2]UNH iSCSI 実装の "initiator/iscsi_initiator.c" にて "max no. of simultaneously active SCSI commands driver can accept" と解説されている

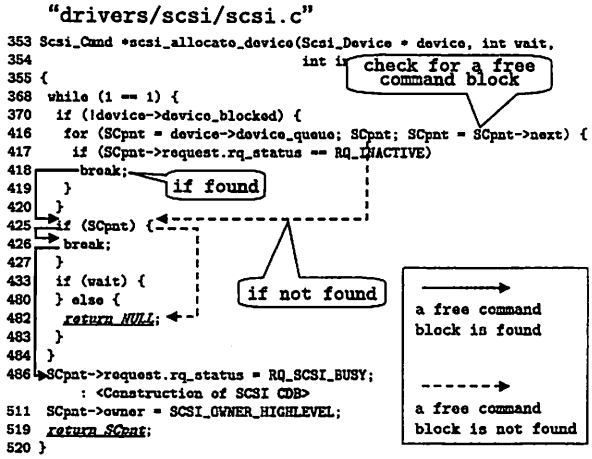

図 7: Linux SCSI 層のトレース: "drivers/scsi/scsi.c"

させ第 3.1 節の実験を行い図 3 の "can_queue=30, queue_depth=8" の実験結果を得た. 同結果より, 全プロセスの合計性能は並列度 8 までは線形に増加させることが可能となったことが確認できる.

3.5 問題回避後の再解析

次に, 並列度が 8 に制限されている原因について考察する. 第 3.3 節同様に巨視的トレース解析を行うと, 制限箇所は同様に SCSI 層であることが確認された. また, 命令が実際に送出される最初の 8 要求と, 送出されない 9 個目の要求のトレースの分岐点は, Linux SCSI 層実装 "drivers/scsi/scsi.c" の `void scsi_allocate_device()` 部であった. その詳細を図 7 に記す. 同図は SCSI 命令ブロック作成部であり, 確保されているブロックキューから未使用ブロックを検索しそのブロックに SCSI 命令を格納し, それを返す関数である. 最初の 8 要求では "`a free command block is found`" に示されるトレース結果が記録された. すなわち, 416 行目において未使用ブロックを検索し, 417 行目においてそれを発見している (よって変数 SCpnt には発見されたブロックへのポインタが納められる). そして, SCSI 命令ブロック作成部 (同ファイル 486 から 511 行目) に移動し同関数は作成した SCSI 命令を返している. これに対し, 命令が発行されない 9 個目の要求では, "`a free command block is not found`" に示されるトレース結果が記録された. すなわち, 416 行目において未使用ブロックを検索したが発見されず (これにより変数 SCpnt は NULL となる), 482 行目において NULL を返している. 同関数は, "drivers/scsi/scsi_lib.c" 内の関数

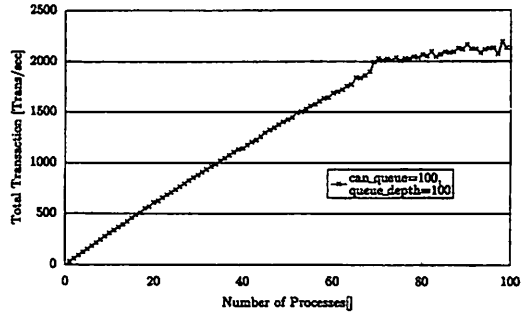

図 8: 実験結果 B : 並列 I/O の合計性能

`scsi_request_fn()` から呼び出されており, SCSI 命令ブロックが返された場合は SCSI 命令を発行し, NULL が返された場合は SCSI 命令を発行しないよう実装されている.

SCSI 命令キュー長 (Linux SCSI 実装内では "queue_depth" と呼ばれる) は SCSI 層の下位層により指定され, 今回の例では iSCSI 層の実装 (UNH iSCSI) の初期値 8 が使用された.

3.6 再解析に発見された問題の回避

前節の解析により "can_queue" 増加後の並列度制限原因は "queue_depth" であると考えられる. そこで, "can_queue", "queue_depth" をともに十分大きな値 (後述の理由により 69 以上) である 100 とし再度性能を測定し, 図 8 の結果を得た. 図より, 発見された並列度制限要因の解決により並列度はさらに上昇し, 並列度 69 まで合計性能はほぼ線形に上昇することが確認された. 並列度 69 における合計性能は 1992.9[Trans/sec] であり, 並列度 98 における合計性能は 2195.0[Trans/sec] であったため, 提案システム適用前の最適化を行っていない状態 (最大で 61.9[Trans/sec]) と比較し合計性能は 32.2 倍や 35.5 倍に向上されたこととなる.

このように, 提案トレースシステムを用いて IP-SAN システムの振る舞いを観察することにより性能劣化原因を的確に発見することが可能であり, それらの解決により性能を大きく向上させることが可能であることが確認された.

4 関連研究

iSCSI を用いた IP-SAN の性能の評価に関する研究としては, 文献 [1, 3, 4, 6, 7, 8, 16] があげられる. 文献 [1] は早期に SCSI over IP の性能評価を行った開拓的な研究である. Sarkar らは文献 [3, 4] におい

てCPU使用率に着目しiSCSIアクセスの性能についての評価を行い, iSCSIアクセスにおけるCPU使用率の増加の程度や, TOE(TCP Offload Engine)の貢献の限界などを示している. 文献[7]は, iSCSI, SMB, NFSの性能評価と比較を行っている. 文献[6]はiSCSI性能を評価しソフトウェア処理手法はFC-SANに匹敵する性能を提供できることを示している. 文献[8]は, IP接続ストレージのアクセス手法としてiSCSIとNFSの比較を行い, NFSに対するiSCSIの優位性等を示している. 文献[16]は, IPsecやSSLを用いるiSCSIの性能を評価している. 以上の研究は各種状況におけるiSCSI性能の評価を行ったものであり, iSCSIの性能を知る上で有用な研究であると言える. しかし, システムの外部から負荷を与え性能を評価したものでありシステム内部の振る舞いについて考察を行ったものではない. また, 性能の向上方法について十分な考察を行ったものではなく, 性能を向上させるには適切な能力の計算資源を用意する以外の方法を提供しない. よって, 性能向上方法の提供を目指す本研究はこれら既存の研究に対しても十分に有用であると考える.

藤田らの文献[5]は, iSCSIターゲットの内部の実装手法も考慮してiSCSI性能の評価を行い, OSのカーネルに変更を加える手法や低レベルインターフェイスを使用する実装手法が性能において優れていることを指摘してる. ターゲット実装に着目し詳細な考察を行っている点において, システム全体の考察を目指す我々の研究と目的が同じでは無いが, ターゲットシステム実装の詳細な考察を行った既存の研究として価値が高いと思われる.

5 おわりに

本稿では, サーバ計算機とストレージ機器の分散協調システムとして動作するIP-SANシステムのアクセストレースシステムを提案し, その有効性の検証を行った. その結果, 提案システムは多段プロトコルスタックで構成される分散システムIP-SANの振る舞いの把握を容易にし, 的確に性能劣化原因を発見することが可能であった. 本稿の例では多並列アクセス時に30倍以上の性能向上が実現され, 提案手法がIP-SANシステムの性能向上の実現に有効な手法であることが確認された.

今後は, ファイルシステムや実HDDデバイス使用時の解析, 提案システム適用のオーバーヘッドの計測, より複雑なアプリケーションの性能向上についての考察を行っていく予定である.

参考文献

[1] Wee Teck Ng et al. "Performance Evaluation and Improving of Sequential Storage Access using iSCSI Protocol in Long-delayed High throughtput Network". In *Proc. of IEICE The 14th Data Engineering Workshop*, March 2003.

[2] F. Neema and D. Waid. "Data Storage Trend". In *UNIX Review, 17(7)*, June 1999.

[3] Prasenjit Sarkar, Sandeep Uttamchandani, and Kaladhar Voruganti. "Storage over IP: When Does Hardware Support help?". In *Proc. FAST 2003, USENIX Conference on File and Storage Technologies*, March 2003.

[4] Prasenjit Sarkar and Kaladhar Voruganti. "IP Storage: The Challenge Ahead". In *Proc. of Tenth NASA Goddard Conference on Mass Storage Systems and Technologies*, April 2002.

[5] 藤田智成 小河原成哲. "iSCSIターゲットソフトウェアの解析". In 先進的計算基盤システムシンポジウム *SACSIS 2004*, May 2004.

[6] Stephen Aiken, Dirk Grunwald, and Andy Pleszkun. "A Performance Analysis of the iSCSI Protocol". In *IEEE/NASA MSST2003 Twentieth IEEE/Eleventh NASA Goddard Conference on Mass Storage Systems and Technologies*, April 2003.

[7] Yingping Lu and David H. C. Du. "Performance Study of iSCSI-Based Storage Subsystems". *IEEE Communications Magazine*, August 2003.

[8] Peter Radkov, Li Yin, Pawan Goyal, Prasenjit Sarkar, and Prashant Shenoy. "A performance Comparison of NFS and iSCSI for IP-Networked Storage". In *Proc. FAST 2004, USENIX Conference on File and Storage Technologies*, March 2004.

[9] "IETF IPS". `http://www.ietf.org/html.charters/ips-charter.html` .

[10] J. Satran et al. "Internet Small Computer Systems Interface (iSCSI)". `http://www.ietf.org/rfc/rfc3720.txt` , April 2004.

[11] IETF Home Page. `http://www.ietf.org/` .

[12] 山口実靖 小口正人 喜連川優. "iSCSI解析システムの構築と高遅延環境におけるシーケンシャルアクセスの性能向上に関する考察". 電子情報通信学会論文誌 *D-1*, 87, February 2004.

[13] University of new hampshire interoperability lab. `http://www.iol.unh.edu/` .

[14] iSCSI reference implementation. `http://www.iol.unh.edu/consortiums/iscsi/downloads.html` .

[15] L. Rizzo. dummynet. `http://info.iet.unipi.it/~luigi/ip_dummynet/` .

[16] Shuang-Yu Tang, Ying-Ping Lu, and David H. C. Du. "Performance Study of Software-Based iSCSI Security". In *1st International IEEE Security in Storage Workshop*, December 2002.