iSCSI ストレージアクセスのトレースシステム

喜連川 優
情報理工学系研究科電子情報学専攻

概 要

FC-SAN の欠点を補う SAN として TCP/IP と Ethernet を用いる IP-SAN や iSCSI が期待を集めている. しかし, iSCSI を用いた IP-SAN システムはサーバ計算機とストレージ機器が通信を行い協調して動作する分散協調システムであること, IP-SAN のプロトコルスタックが多段であることなどが原因で, IP-SAN システムの振る舞いの把握は容易では無い. 本研究では, 分散システムである IP-SAN システム全体の振る舞いの把握を可能とす統合的トレースシステムの構築を行う. 実装したトレースシステムを IP-SAN に適用した結果, IP-SAN システムの振る舞いが視覚的に確認できるようになった. また, IP-SAN の性能劣化原因を明らかにすることができ, その解決により性能が約 4 倍に向上された.

1 はじめに

ストレージの管理コストの削減手法として SAN (Storage Area Network. ストレージ専用の高速ネットワーク) を用いたストレージの集約が提案された. ストレージを一箇所に集約し, 各計算機からは SAN を用いてストレージに接続する手法は管理コストを大幅に削減することが可能であり, すでに多くの企業で導入されている. しかし, 現世代の SAN は FC (Fibre Channel) を用いた FC-SAN であり FC 管理技術者が少ない, FC には接続距離に限界がある, FC の導入コストが高いなどの問題点も明らかになってきた. そこで, これらの問題を解決する SAN として Ethernet と TCP/IP を用いて構築する IP-SAN や, そのためのデータ転送プロトコル iSCSI が提案され大きな期待を集めている. IP-SAN は, IP 管理技術者が多い, IP は接続距離に限界がない, 導入コストが低いなどの利点が期待されている. 逆に IP-SAN の欠点としては, 性能が FC-SAN より劣ることと, CPU 使用率が高いことが指摘されており, これらの解決は非常に重要である. しかし, IP-SAN は SCSI over iSCSI over TCP/IP over Ethernet という多段で複雑なプロトコルスタックで構成されており, さらにサーバ計算機とストレージ機器が協調動作する分散協調システムであるため, IP-SAN の振る舞いの把握は困難である. 結果として性能向上の考察が十分になされていない.

そこで我々は, 文献 [1] において TCP 層などの個別の層を観察できる解析システムを構築した. さらに本研究では, 複雑な構成の IP-SAN の包括的な振る舞いの把握を可能とする “iSCSI トレースシステム” の構築を目指す. そして, 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し, 性能向上の実現を目標とする.

2 IP-SAN トレースシステム

本章では本研究で提案する “iSCSI ストレージアクセスのトレースシステム” について説明を行う. IP-SAN においてサーバ計算機のアプリケーションから発行された I/O 命令は, サーバ計算機において, ファイルシステム層やブロックデバイス層やキャラクタデバイス層, SCSI 層, iSCSI 層, TCP/IP 層, Ethernet 層を経由し, ストレージ機器において, Ethernet 層, TCP/IP 層, iSCSI 層, SCSI 層を経由し HDD デバイスにアクセスを行う. そして, この経路を逆方向に経由しアプリケーションに応答が返されることになる. そこで 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, 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

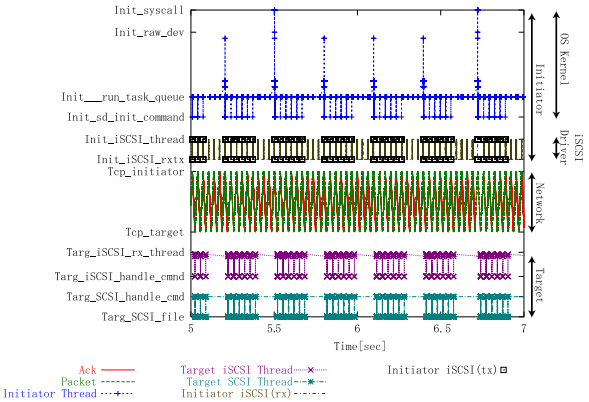

図 1: iSCSI アクセスのトレース

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 が提供する iSCSI reference implementation (ver. 1.5.02) 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 1 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 Target 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実行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処理における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 高遅延環境における並列アクセスのトレース

本章では, 前章で紹介を行った “iSCSI ストレージアクセスのトレースシステム” を実際に並列 iSCSI アクアセスに対し適用し, 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 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. イニシエータ, ターゲット, 遅延装置はすべて PC 上に構築し, イニシエータとターゲットには Linux を, 遅延装置 には FreeBSD をインストールした.

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, 同ベンチマーク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, 片道遅延時間は 16m 秒とした.

3.2 実験結果

初期状態において上記の実験を行い図 2 の “can_queue=2(default)” 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

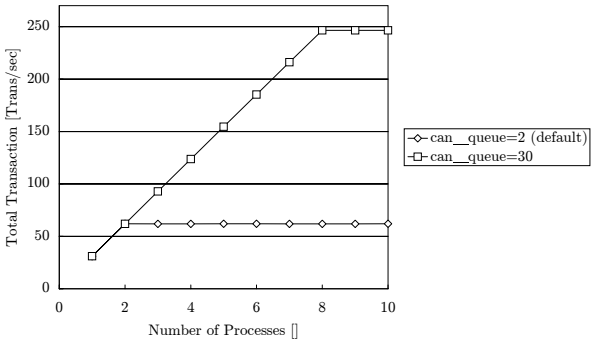

図 2: 実験結果 : 並列 I/O の合計性能

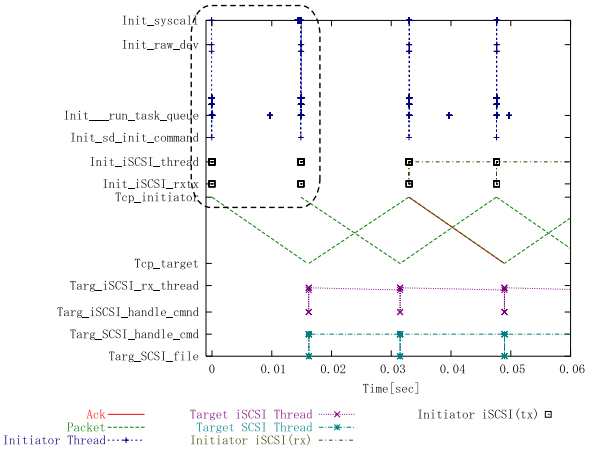

図 3: 並列アクセスのトレース図 (default) : A

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. 2 プロセス並列時には 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, プロセス数の上昇に対する合計性能の向上は 2 プロセスで飽和する結果となった. これは, 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 に制限していることが原因と考えられる.

3.3 並列アクセスのトレース

まず並列度を制限している部分の巨視的な解析を行う. 前節の 3 プロセス並列アクセス実験のトレースを提案トレースシステムを用いて可視化すると図 3 の様になった. 同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いないことが確認できる. すなわち, 並列度を制限している層はサーバ計算機側に存在すると予想される.

図 3 左上波線部を拡大し可視化したものを図 4 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 本の 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 4 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. このことから, 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おり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 層を経由し要求が実際にネットワークから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 となっていることが確認できる.

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)) の分岐点は SCSI 層である図 5 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である命令数 `host_busy` 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

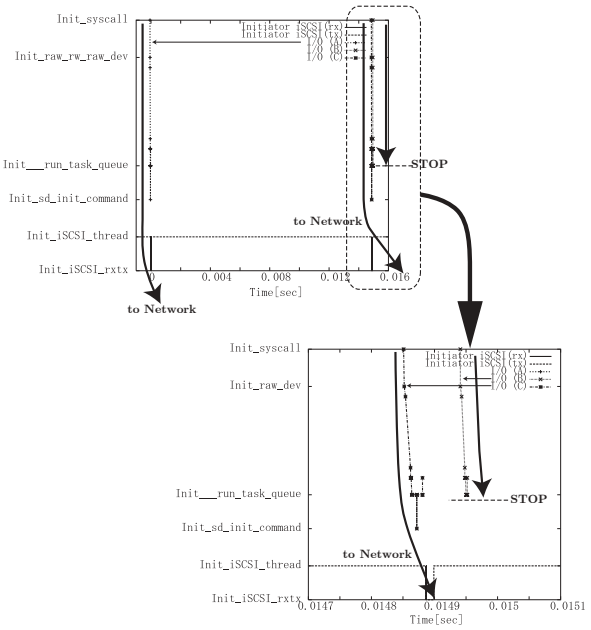

図 4: 並列アクセスのトレース図 (default) : B

SCSI 命令数 can_queue 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 5 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.

3.4 発見された問題点の回避と性能の向上

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 30 に増加させ第 3.1 節の実験を行い図 2

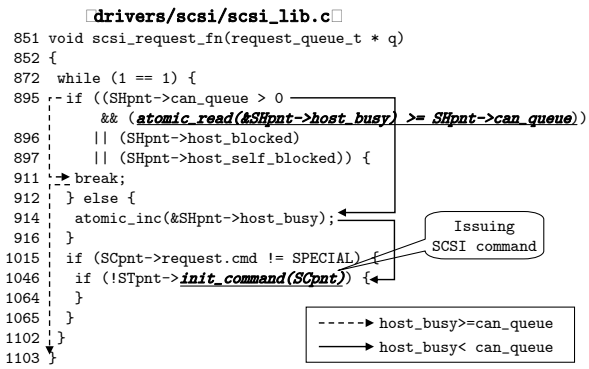

図 5: Linux SCSI 層のトレース : ”drivers/scsi/scsi_lib.c”

の “can_queue=30” の実験結果を得た. 同図より, 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り, 並列度 8 以上においては 4 倍の性能向上が実現させられた.

このように, 提案トレースシステムを用いて統合的なアクセストレース解析を行うことにより, 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 おわりに

本研究では iSCSI トレースシステムを提案し, それを実 IP-SAN 環境に適用し有効性を検証した. 提案システムの適用により 約 4 倍の性能向上が実現され, 提案システムの有効性が確認された. 今後はファイルシステムや実 HDD デバイスを用いるより現実的な IP-SAN システムを観察できるトレースシステムの実装を進めていく予定である. また, “can_queue” 値増加後の並列度を 8 に制限している原因についても考察を進めていく予定である.

参考文献

[1] 山口実靖, 小口正人, and 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.