論　文

iSCSI リモートストレージアクセスの性能向上を実現する手法の提案と実装

比嘉　玲華[†]　松原　幸助[††]　岡廻　隆生[††]　山口　実靖[†††]
小口　正人[†]

Suggestion and implemention for the improvement of iSCSI remote storage access

Reika HIGA[†], Kosuke MTSUBARA[††], Takao OKAMAWARI[††], Saneyasu YAMAGUCHI[†††], and Masato OGUCHI[†]

あらまし　リモートバックアップの手法として iSCSI を適用したいという期待が産業界からも高まっている．しかし，iSCSI は高遅延環境において性能が劇的に劣化してしまうということが知られている．iSCSI は複雑なプロトコルスタックとなっており，性能劣化の問題を解決するためにはネットワーク上を飛び交うパケットを解析するとともに，各層を網羅的に解析することが必要になる．そこで本稿においては，パケット解析と各層を解析するシステムツールを提案し実装を行った．このシステムツールを用いて iSCSI リモートストレージアクセスを解析し，カーネルにおいて性能劣化におちいる条件分岐のコードを特定し最適化を行った．
その結果，RTT20ms においてデフォルト時と比較して約 7 倍の性能向上を実現した．この性能は，理論値に極めて近い性能であり高遅延環境において性能劣化の問題を解決することに成功した．最適化においては iSCSI ドライバに対してのアプローチではなく，TCP 層においてのアプローチをとったというより汎用性能高いアプローチを採用している．

キーワード　iSCSI，IP-SAN，Linux kernel

1. ま え が き

コンピュータシステムにおける処理データ量の増大に伴い，効率的にストレージを管理したいという要望が高まっている．そこで SAN (Storage Area Network) が登場し，広く用いられるようになった．現在主流として用いられているのは，ファイバチャネルを用いた FC-SAN であるが，構築や管理が非常に高価であるため，より経済的でコストパフォーマンスの高い SAN が望まれている．そこで，次世代 SAN として期待されているのが Ethernet と TCP/IP を用いて構築する IP-SAN である．iSCSI はその IP-SAN の代表的なプロトコルであり，SCSI コマンドを TCP/IP パケットでカプセル化する規格である [1] [2]．IP-SAN は，IP 技術者が多い，接続距離に限界がない，比較的安価構築可能である，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 よりも低い，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．

また，テロや自然災害を考えた場合，データをできるだけ遠くに書き込むというリモートバックアップは非常に重要であり，リモートバックアップのプロトコルの研究は学術的興味のみならず，産業界からも強く望まれている．よりコストパフォーマンスの高いものが利用される昨今の時勢を考えた場合，リモートバックアップのプロトコルとして iSCSI の適用が望まれる．

しかし現状において iSCSI は，図 1 のように複雑な階層構成のプロトコルスタックで処理しており，バースト的なデータ転送も多いことから，通常のソケット通信と比較して，特に高遅延環境においては性能の劣化が著しいことが知られている．実際に本論文で用いた実験環境においても高遅延環境を構築し iSCSI の性能測定を行った．その結果を示した図 2 によると，ソ

[†] お茶の水女子大学
Ochanomizu University
[††] ソフトバンクテレコム株式会社
SOFTBANK TELECOM Corp.
[††] 工学院大学
Kogakuin University

ケット通信時においては高遅延環境下においてもほぼ性能を保っているのに対して，iSCSI 通信においては，高遅延環境になるにつれて性能が劇的に劣化していったという結果が得られた．

アクセスブロックサイズを 4MB 程度の大きな値としたとき，理論的には図 2 が示す程度の性能がだせるはずである．しかし，高遅延環境下では劇的な性能劣化という現象が起きている．

そこで，本研究においては，リモートバックアップへの iSCSI の適用を考え，シーケンシャルライトアクセス性能の向上に焦点を絞り，高遅延環境下における iSCSI リモートストレージアクセス性能解析を行った．

iSCSI は複雑なプロトコルスタックからなるためにすべての層にまたがる解析を行う必要がある．そこで，SCSI 層，iSCSI 層，TCP 層，Ethernet 層における解析や最適化を行った．その結果，性能に一番影響を与えたのは TCP 層における最適化であった．ソケットバッファの割り当てがうまくいかずパケットが断続的に送信されてしまっていることが原因であった．カーネル解析を行い，条件分岐箇所をカーネルコード上で特定し，そこに性能向上のための適切なソフトウェアモジュールを挿入した．その結果，RTT20ms においてデフォルト時の iSCSI と比較して約 7 倍の性能向上という結果を得られた．この値は，理論値に極めて近い性能であり，高遅延環境下における iSCSI 性能劣化という問題を解決できたということができる．

本稿は以下のように構成されている．まず，第 2 章において本実験環境と，本稿で提案するシステム解析ツールについて紹介する．次に第 3，4 章において本システムツールを用いた性能解析を行う．解析結果に基づいた性能評価を第 5 章で行い，第 6 章において本稿のまとめと今後の課題を述べる．

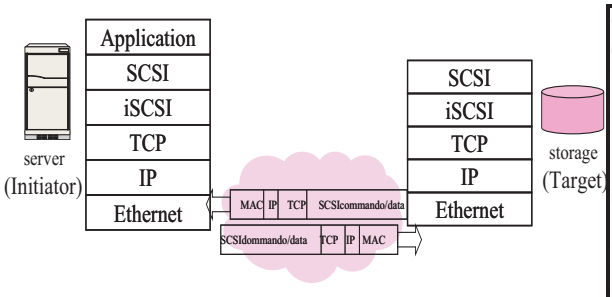

図 1　Configuration of iSCSI

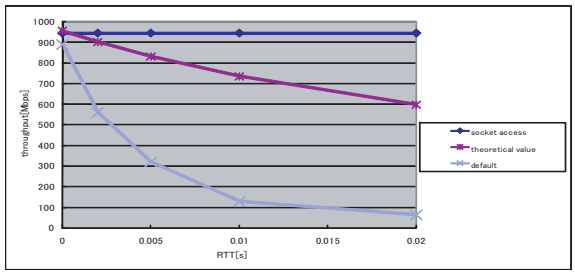

図 2　Throughput comparison with default iSCSI

2.　システムについて

2.1　実験システム

本研究において,Initiator と Target 間は GigabitEthernet で接続し，TCP/IP コネクションを確立した．Target のストレージには SAS ディスクを用い RAID コントローラによる RAID0 構成で接続した. 使用した実装システムと実験環境を図 3 および表 1 に示す.

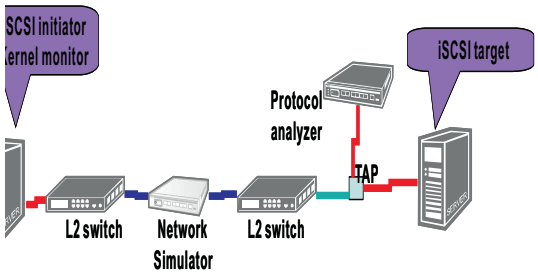

図 3　実装システム概要

表 1　実 験 環 境

OS	Red Hat Enterprise Linux 2.618-8.e.15
CPU	Quad Core Intel Xeon 1.6GHZ
Main Memory	2GB
NIC	Intel PRO/1000PT Server Adaptor on PCI Express
HDD	73GB SAS × 2(RAID0)
RAID Controller	SAS5/iR
iSCSI	Initiator : open-iscsi-2.0-865
	Target : iSCSI Enterprise Target(IET)-0.4.15
Network Analyzer	ClearSight Network Recorder
Network Simulator	ANUE

2.2 解析システムツールの提案

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ため，全層を経由して行われることから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解析する必要がある．そこで本稿において図 4 のようなシステムツールを構築した．このシステムツールを使用することで，ネットワーク上を飛び交うパケット解析と Initiator におけるカーネル解析を行うことが可能となる．

システムツール構成要素の一つに我々のオリジナルのツールである「カーネルモニタ」がある．これは TCP カーネルの振舞をモニタするツールである．カーネル内部の TCP ソースにモニタ関数を挿入しカーネルを再コンパイルすることで，一般にはユーザ空間からは見ることができない TCP のパラメータ情報などを可視化できるというツールである．これによりモニタできるようになった値には，輻輳ウィンドウやソケットバッファキュー長といった TCP パラメータ情報や log trace timestamp がある．その他のシステムツール構成要素としては，tcpdump コマンド，ネットワークアナライザ [3] がある．これらによって，ネットワーク上を飛び交うパケットを解析し，パケット情報を得ることが可能になる．本稿においては，このシステムツールを効果的に使用しすべての層を解析，最適化を行うことで，iSCSI 遠隔ストレージアクセスの性能低下の原因がどこにあるのかを詳しく解析していく．

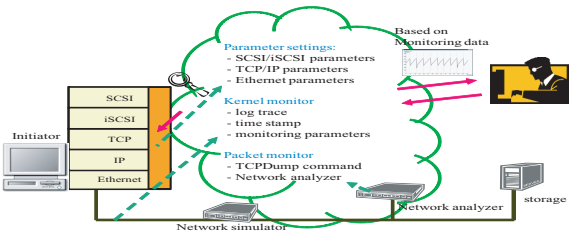

図 4　解析システムツール概要

3. システムツール適用による TCP 層以外の解析

本章においては，既存研究における，システムツールを用いた TCP 層以外の箇所の解析と最適化について説明する [4]．SCSI/iSCSI 層と Ethernet 層における最適化を行った結果，RTT=20 m s においてはデフォルト時と比較して約 5 倍の性能向上を得られた．よりよりパフォーマンスを得るために，バーストアクセスを行ったが，高遅延環境下ではやはり性能低下が確認された．システムツールを用いた解析の結果，パケット送信の断続が確認された．この断続が大きな性能低下の要因だと考えられる．また，TCP ACK がきっかけでパケット送信再開が始まっているから，パケット送信断続の原因は TCP 層のどこかにあると確認された．

4. システムツール適用による TCP 層の解析

トランスポート層に性能劣化の要因が考えられる場合，一般的には，広告ウィンドウ，輻輳ウィンドウ，ソケットバッファのいずれかである．そこで，その全てをシステムツールを使用し網羅的に解析していく．本章においては，特に，ソケットバッファについて焦点を絞り解析を行う．

その際の環境設定としては，遅延時間を RTT=20ms，アクセスブロックサイズを 4MB，広告ウィンドウを通信の妨げにならない値の 5MB に設定した．

4.1 ウィンドウ解析

実際の iSCSI 通信時における広告ウィンドウ，輻輳ウィンドウの値を解析した [4]．その結果，広告ウィンドウ，輻輳ウィンドウともにパケット送信の断続を引き起こす値ではないことが分かった．

4.2 ソケットバッファ解析

前記の解析結果により，パケット送信断続の原因は，広告ウィンドウ，輻輳ウィンドウの両者ではないことがわかった．そこで，本章においては送信処理におけるソケットバッファのキュー長の振舞を解析していく．図 6 から図 9 は，Initiator 側で tcpdump コマンドを実行したときのパケットの送出と，カーネルモニタを用いて記録したキューの長さを比較したものである．横軸を時間，縦軸をキューの長さ，第二縦軸をパケットの送出番号とする．このとき ACK の縦軸は意味を持たずタイミングのみの表記とする．

4.2.1 TCP 送信処理

iSCSI は下位のトランスポート層に TCP を用いる．Linux OS のカーネル内部の TCP 実装において，送信データを格納するソケットバッファは図 5 のように，送信キューにつながれていて処理されるのを待つが，送信されても確認応答 (ACK) を受信するまでは解放されない．

送信キューは sock 構造体の sk_write_queue メンバで，次に送り出すデータのソケットバッファを指すのが sk_send_head ポインタである．このうち，キューの先頭から sk_send_head の手前までのソケットバッファは，「送信されたが確認応答がまだないために解放できない部分」である（再送キュー：状況によっては再送される）．sk_send_head から先には，これから送信するデータのソケットバッファがつながれている．セグメントを送信したら，sk_send_head をずらしていく．

本研究においては，キューの長さ（waiting ACK のキューの先頭から tail まで）について議論していく．

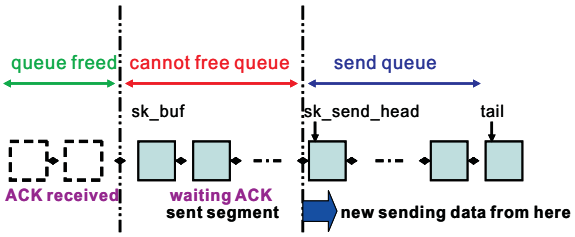

図 5　送信キュー

4.2.2　送信パケットと送信キュー

ソケット通信においては，高遅延環境の場合も性能がほぼ維持されるのに対して，iSCSI 通信においては高遅延環境の場合は性能が劇的に低下してしまう．そこで，両通信においてキューの振舞を比較することで解析をおこなった．iSCSI 通信は sg_dd コマンド [5]，ソケット通信は Iperf を使用し測定をおこなった．

図 6 から，iSCSI 通信においてはキュー長の最大値が約 300 であり，その後 0 から 300 を推移することがわかる．図 7 からは，ソケット通信においてはキュー長の最大値は約 1300 であり，その後 200 から 300 を推移することがわかる．このように，両通信において明らかな振舞の違いが確認された．

4.2.3　詳細な送信キューの振舞解析

本節において，図 8 以降は TCP ACK コマンドを含めて詳細に解析をしていく．ACK はタイミングのみの表記とする．図 8 は iSCSI 通信時の定常状態以降のソケットバッファの振舞を示している．パケット送出停止から約 RTT 経過後に Target からの ACK が受信され，ソケットバッファの ACK 待ちキューが解放され，その後送信キューの成長とパケットの送信再開が行われている．さらにその後キューの成長が停止

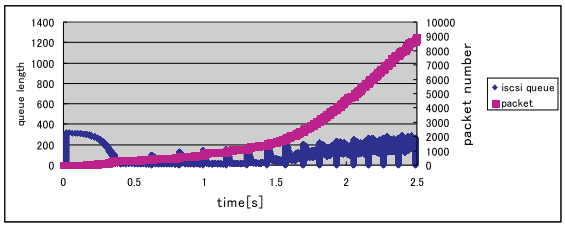

図 6　Queue state in iSCSI access in 20ms RTT

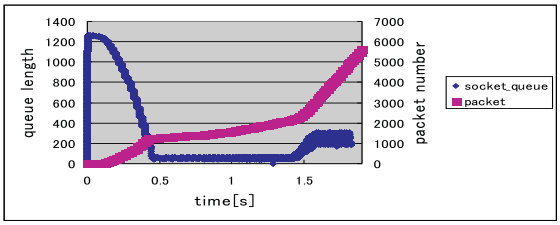

図 7　Queue state in socket access in 20ms RTT

し，データを送信し尽くしたことでパケット送信の再停止が生じていることがこれらの図からわかる．

比較のために，図 9 にソケット通信における拡大図を紹介する．キューが 1300 に到達するとキューの成長は止まるが，パケット送信の断続は生じていない．このことは，高遅延環境における iSCSI 通信とソケット通信の性能差の原因になっていると考えられる．

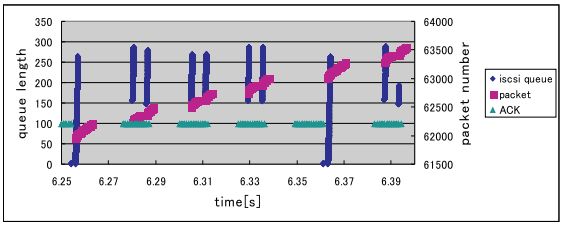

図 8　Enlarged part of queue state in iSCSI access

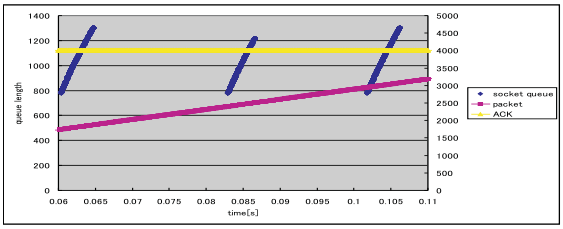

図 9　Enlarged part of queue state in socket access

4.2.4 iSCSI 通信時のソケットバッファキュー長解析考察

ソケット通信と iSCSI 通信の高遅延環境におけるキュー長の振舞が明らかに異なることが確認された．ソケット通信においては，通信の開始時に大きな値までキューが成長しパケットは送出され続けるのに対して，iSCSI 通信においてはキュー長が約 300 で頭打ちになり，パケット送出が断続的になってしまっている．ソケット通信においては高遅延環境においても高いスループットを維持するのに対して，iSCSI 通信においてはスループットの劇的な低下が生じる原因になっていると考えられる．つまり，ソケット通信時においては，TCP にデータが割り当てられるとバッファが随時割り当てられキュー長が伸びていくのに対して，iSCSI 通信の場合は，割り当てられるバッファが存在してもバッファ割り当てられることなく，キュー長が伸びていかないことがある．

4.2.5 条件分岐箇所の特定

iSCSI 通信とソケット通信の両者の両者においてどのようにカーネル内の処理が行われているのかを，カーネルモニタのログを詳しく追っていったところ，通過する場所は違うものの、両者ともに timeout が生じていることがわかった．ソケット通信においては，図 10 における 1415 行目の schedule() を通過し，iSCSI 通信においては 1439 行目の schedule() を通過する際にタイムアウト待ちに入り RTT 後の ACK で再開されている．iSCSI 通信においては不必要なタイムアウト待ちが頻繁に生じているということも確認された．

両者ともにタイムアウト待ちに入っているのにも関わらず，ソケット通信においては性能は維持されたままで，iSCSI 通信においては性能低下を引き起こしている．その原因としては，ソケット通信の場合はソケットバッファのキューに十分なデータが保持されているからパケット送出は途切れないということが考えられる．

そこで，不必要なタイムアウト待ちを解消するために，コードを追跡していった．その結果を図 11 に示す．その結果，sk_stream_memory_free 関数が条件分岐となっていることが分かった．sk_stream_memory_free 関数を図 12 に示す．この関数は，sk_wmem_queued と sk_send_buf の大きさを比較して値を返している．sk_wmem_queued が大きい場合にタイムアウト待ちに陥ることが解析の結果わかったので，sk_send_buf の値を最適化するソフトウェアモジュールを挿入した．その結果，不必要なタイムアウト待ちが格段に減った．この時，カーネルの改変によって，悪い副作用は確認されていない．

```
1400 fastcall signed long __sched schedule_timeout(signed long timeout)
1401 {
1402     struct timer_list timer;
1403     unsigned long expire;
1404
1405     switch (timeout)
1406     {
1407     case MAX_SCHEDULE_TIMEOUT:
1415         schedule();                                   Socket access
1416         goto out;
1417     default:
1425         if (timeout < 0)
1426         {
1427             printk(KERN_ERR "schedule_timeout: wrong timeout "
1428                    "value %lx from %p\n", timeout,
1429                    __builtin_return_address(0));
1430             current->state = TASK_RUNNING;
1431             goto out;
1432         }
1433     }
1435     expire = timeout + jiffies;
1436
1437     setup_timer(&timer, process_timeout, (unsigned long)current);
1438     __mod_timer(&timer, expire);
1439     schedule();                                       iSCSIaccess
1440     del_singleshot_timer_sync(&timer);
1441
1442     timeout = expire - jiffies;
1443
1444  out:
```

図 10　kernel ソースコード

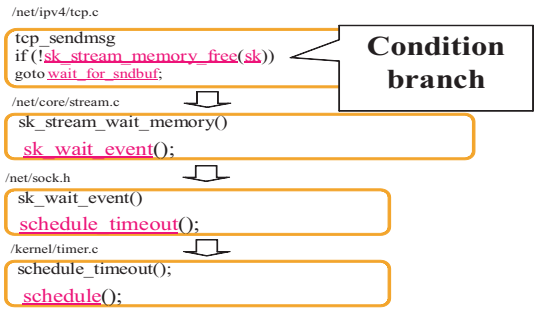

図 11　trace code

```
linux+v2.6.18.5/include/net/sock.h#L446

static inline int sk_stream_memory_free(struct sock *sk)
{
 return sk->sk_wmem_queued < sk->sk_sndbuf;
}
```

図 12　sk_stream_memory_free

5. 評　　価

本章では sk_send_buf の値を最適化するソフトウェアモジュールを挿入したことでどのように振る舞いが変わったのかみていく．

5.1 キュー長変化

図 13 は sk_send_buf 最適化後のキュー長の振る舞いを表したものである．図 6 と比較してキュー長の最大値が大きくなっており，明らかに振る舞いが変化していることがわかる．

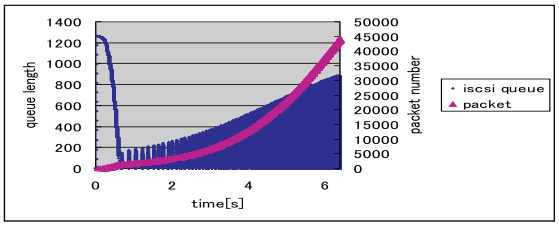

図 13 Queue state in iSCSI access with optimized socket buffer in 20ms RTT

5.2 パケット送出量変化

図 14 は sk_send_buf 最適化後のネットワーク上を飛び交うパケット解析を表したものである．write10 コマンドから次の write10 コマンドまでが 4MB のパケットを示している．図 8 においては，4MB のパケットが約 1 MB ずつに断続して送信されていたのに対して，図 14 では断続が解消されている．

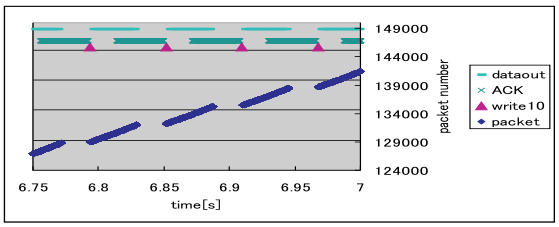

図 14 Analysis of packet on Initiator with optimized socket buffer in 20ms RTT

5.3 スループット変化

図 15 は sk_send_buf 最適化後のスループットを表したものである．RTT=20ms においてデフォルト時と比較して約 7 倍の性能向上が達成されたこと，またその値は理論値と匹敵する性能であることから，iSCSI リモートストレージアクセスの高遅延環境における性能劣化問題は解決されたということができる．

6. まとめと今後の課題

接続距離に制限がなく IP-SAN を比較的安く容易に構築可能というメリットを生かして，産業界からも高く期待されているリモートバックアップのプロトコルに iSCSI を適用することが望まれている．しかし，iSCSI には高遅延環境において性能が劇的に劣化してしまうという問題がある．そこで，本稿において，複雑なプロトコルスタックを持つ iSCSI の解析を行うために，iSCSI が高遅延環境で性能低下する原因を解析し性能向上を実現するシステムツールの提案，実装をおこなった．性能低下原因となる条件分岐箇所を特定し，その箇所に性能向上のために適切なソフトウェアモジュールを挿入した．

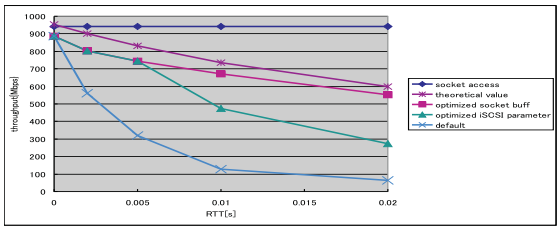

図 15 Throughput comparison with optimized socket buffer

解析の結果，性能向上に一番影響を与えたのは TCP 層におけるソケットバッファの最適化であった．RTT=20ms においてデフォルト時の約 7 倍の性能向上が確認され，高遅延環境における性能劣化問題の解消が達成された．

性能向上の際に，iSCSI ソフトウェアに対して最適化を行うのではなく，その下の TCP 層に対して最適化を行ったことは，より汎用性の高いアプローチであり，特定の iSCSI ソフトウェアに限った対処ではない．よって，我々のとった最適化，システムツールは幅広いケースに対応が可能であると考えられる．

そこで，Linux OS，Open iSCSI 以外の環境を構築し実験を行ってこれを実証したい．

文　献

[1] iSCSI Specification, http://www.ietf.org/rfc/rfc3720.txt?number=3270
[2] SCSI Specification, http://www.danbbs.dk/˜dino/SCSI/
[3] http://www.toyo.co.jp/clearsight/product/analyzer.html
[4] Reika Higa et al. :”Optimization of iSCSI Remote Storage Access through Multiple Layers,” TeNAS’2009 in conjunction with AINA-09, pp.612-617, May 2009
[5] http://sg.torque.net/sg/sg3_utils.html/