RDMA Storage I/O向けカーネル通信APIの設計

大江和一[†] 渡辺高志[†]
中島耕太[†] 西川克彦[†]

RDMAでデータ転送を行うStorage I/Oに適したカーネル通信APIの検討を行い，Scoreクラスタシステムソフトウエアの PM2をベースにカーネルからのアクセスを可能にしたKernel PM2を設計した．このKernel PM2にはPM2でサポートされている機能に加えて複数の離散領域を同時に登録&転送できる機能とCallbackによるイベント通知機能を追加した．InfiniBand 4x対応富士通製HCAを用いて試作を行い，iozoneから325 MB/secの性能を得ることに成功した．

Design of the Kernel Level Communication Interface for Storage I/O Using RDMA

Kazuichi Oe,[†] Takashi Watanabe,[†] Kohta Nakashima[†] and Katsuhiko Nishikawa[†]

This paper presents the design of Kernel PM2. The Kernel PM2 is based on the PM2 of SCore cluster system software, and is added new functions. One function can register the distributed memory areas at the same time and return a memory handle. The distributed memory areas can be tranferred at the same time by using the memory handle. Another function can inform communication events by using callback. We have implemented the Kernel PM2 on Fujitsu's InfiniBand 4x HCA and confirmed to turn the 325 MB/sec throughput performance by iozone benchmark.

1. はじめに

現在，SAN（Storage Area Network）を構築するうえで主流になっているのがFC（Fiber Channel）である．FC SANでストレージシステムを構築する場合に問題となるのは主にコスト高になることである．

そこで近年，TCP/IP上でiSCSIを使うことでIP上にSANを構築（IP SAN）する方法が提示されてきた．しかし，TCP/IPを前提にするIP SANでは性能上の問題でハイエンドには向かないとされている．

このIP SANの問題を解決する方法としてIB（InfiniBand）のようなRDMAをサポートしたネットワーク上でのSRP（SCSI RDMA Protocol）やiSER（iSCSI RDMA over Ethernet）を使用したSAN構築が提案されている．SRPやiSERでSAN構築ができればFC SANのコスト高の問題も解決でき，1つのネットワークでSANとLANの両方が構築できるメリットもある．10 GbEthernetでもRDMAサポートが予定されており，今後SANを構築するうえで有望な方法だと著者らは考えている．

このSRPやiSERでSAN構築を行う場合，イニシエータ側はカーネル内部からの通信APIを使用してターゲット側とのアクセスを行っていくことになりカーネル内部からの通信APIの機能，性能がストレージ性能に大きな影響を与えることになる．本稿では，著者らが開発を進めているWire Speed Storage（WSS）のイニシエータ側通信APIとしてkDAPL（kernel Direct Access Programming Library）を使用して実装を行った場合の問題点の報告，kDAPLにおける問題点を解決したKernel PM2の設計およびWSSと組み合わせた実験結果の報告を行う．

2. WSSの概要

最初に評価に使用したWSS[12]について簡単に紹介しておく．

WSSは，StorageEngine（SE），CacheEngine（CE），RealStorage（RS）の3種類のコンポーネントを低レイテンシ・高スループットネットワークで相互結合した構成をとる．さらに，SRPイニシエータとも同一

† 株式会社富士通研究所
Fujitsu Laboratories Ltd.

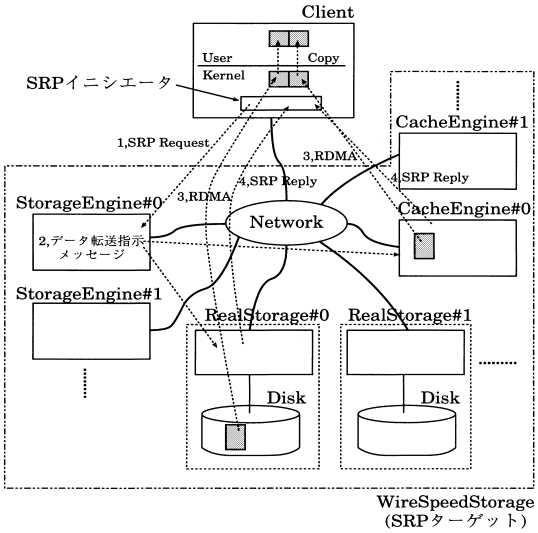

図 1 WSS アーキテクチャ
Fig. 1 WSS architecture.

ネットワークで結合する（**図 1** 参照）．

SRP イニシエータは SCSI ドライバとしてクライアント側に組み込まれる．

SE は SRP イニシエータ側から見える仮想ボリュームと実際にデータを管理している CE または RS 領域とのマップを管理しており，SRP イニシエータからのアクセス要求を受け付けると，実際にデータを管理している CE または RS を検索し，該当コンポーネントにデータ転送命令を送出する．

CE は WSS のストレージキャッシュとして機能し，SE からの指示に基づいて SRP イニシエータ側領域にデータ転送を実行する．データ転送実行後，SRP reply も直接 SRP イニシエータ側に送信する．

RS はディスク装置を内蔵しており，WSS のデータ格納庫として機能する．

このような構成のアーキテクチャにより，1）ワイヤスピードに近い高性能，2）高コストパフォーマンス，3）高スケーラビリティを実現した．

3. kDAPL による SRP イニシエータの実装と評価

3.1 SRP+kDAPL+IB を使用した理由

現在使用できる最も帯域の大きいネットワークは 10 Gbps クラスであり，IB は比較的入手しやすくかつ RDMA をサポートしているので IB で実装を行うことにした．IB NIC ボードとしては IB 4x 対応富士通製 HCA を使用した．通信 API としては DAPL を使用した．DAPL はカーネルからの通信 API をサポートしており，著者らが実験を開始した時点では他にカーネルからアクセス可能な API がなかったため使用した．Storage I/O プロトコルとしては SRP を使用した．著者らはすでに SRP イニシエータを cLAN 上に実装 & 評価[12] しており，コード等が流用できることが SRP を使用した最大の理由である．ストレージ業界の動向としては主に運用上の問題で iSER が主流になりつつあるが[7),8)]，性能評価という観点では SRP も iSER も大差ないと著者らは考え，SRP での評価を行った．

表 1 DAPL 試作システムの基本仕様

Table 1 Specification of DAPL prototype system.

PC	CPU: Xeon 2 GHz × 2 Chipset: Server Works GC LE Memory: 2 GB
Network	Nic: InfiniBand 4x 対応富士通製 HCA Switch: RedSwitch RS-8020 HCA Driver: Intel-SF IBA 1.120 DAPL: NetApp-SF beta 2.02
OS	Linux 2.4.18

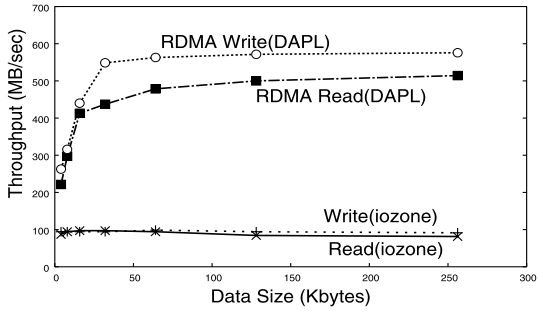

図 2 iozone 実行結果
Fig. 2 Iozone execution result.

3.2 性能評価

WSS は SE，CE の 2 ノードで起動した．さらに SRP イニシエータをクライアントとして使用するノードに起動した（合計 3 ノード）．各ノードともに同一仕様の PC を使用した（**表 1** 参照）．クライアントから SRP イニシエータに対応する特殊ファイルに対して iozone[4] を実行することで性能評価を行った．データ転送はすべて CE から行われるようにした．その結果，100 MB/sec 前後のスループットとなり，DAPL RDMA のスループット（600 MB/sec 前後）の 1/6 程度の性能しか得られないことが分かった（**図 2** 参照）．

実験に使用した Linux-2.4.18+SRP イニシエータに対して iozone からアクセスを行うと iozone からの 1 回のアクセスサイズとは無関係に SRP イニシエータのレベルでは 1 回の Scsi_Cmnd にほぼ 4 KB

× 32（128 KB）の離散領域が渡されてくることが分かった☆．SCSI 共通層および buffer cache 層が転送要求を束ねて下位ドライバに IO を出しているためと考えられる．

そこで Read 128 KB 実行時のシステム内部の遅延時間測定を行った（**図 3** 参照）．測定結果より 1）CE 処理時間（672 us），2）Reg Mem（Register Memory）☆☆（260 us），3）カーネル空間〜ユーザ空間コピー（236 us），4）3 回のメッセージ通信レイテンシ（204 us）の順に遅延時間が大きいことが分かった．Write 128 KB の測定も行ったが Read と同等であったことも報告しておく．

4 KB 単位で分割された 32 バッファアドレスが上位から SRP イニシエータに 1 回の Scsi_Cmnd で渡され，このアドレスリストが WSS 側に通知されるため RDMA は 4 KB 単位で 32 回実行される．RDMA 4 KB × 32 の実行時間は CE 処理時間の大部分を占める．RDMA の遅延時間だけでスループットを計算すると 197 MB/sec である．

Reg Mem は RDMA の対象となる領域をページングの対象にならないようにロックし IB の NIC である HCA ☆☆☆に登録する遅延時間である．Reg Mem は Source 側 Destination 側の両方で操作が必要である．SCSI 上位からは 1 回の Scsi_Cmnd で 4 KB × 32 の離散領域への転送を指示されたものが渡されるため Reg Mem が 32 回実行される．1 回あたりの Reg Mem の遅延時間は 8 us 前後だが 32 回実行されるため 260 us の遅延となった．

今回実装した SRP イニシエータでは，SCSI 上位に転送完了を通知した後に Dereg Mem（Deregister Memory）処理を実行するためカーネル空間〜ユーザ空間コピーに隠蔽されてユーザからみえる性能に直接影響していないが，この Dereg Mem の遅延時間も大きい（525 us）ことも報告しておく．なお，CE 側のキャッシュ領域は初期化時にあらかじめ Reg Mem できるため問題にならない．

クライアント→SE，SE→CE，CE→クライアント通信で発生する 3 回のメッセージ通信レイテンシはコンポーネント間での Request/Reply 送受信にともなうものである．DAPL でのメッセージ受信は Polling と Callback のどちらかで実装することが可能だが，WSS では CPU を占有しない Callback で実装を行った．SRP イニシエータでは SCSI ドライバとして実装するために CPU を占有しない Callback での実装が必須になることと，SE，CE に関してもサーバとして動作させるため Callback を使用した方が実装が容易なことがその理由である．4 KB × 32 の SRP request は 2 KB 程度の大きさになることが分かっており，**図 4** の Callback での遅延時間とほぼ一致することが読み取れる．

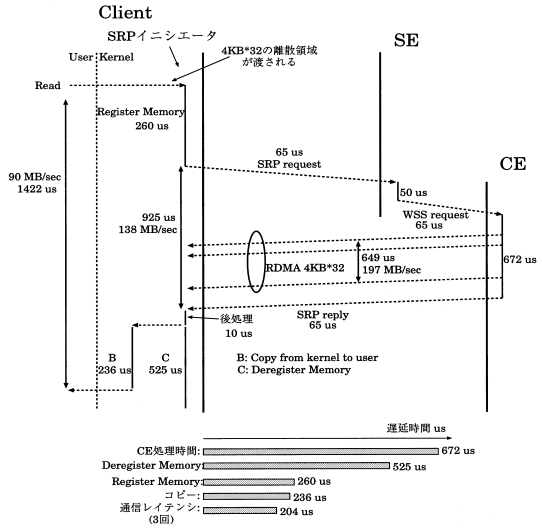

図 3 iozone 実行時の内部遅延時間（Read 128 KB）
Fig. 3 Internal latency at the time of iozone execution (Read 128 KB).

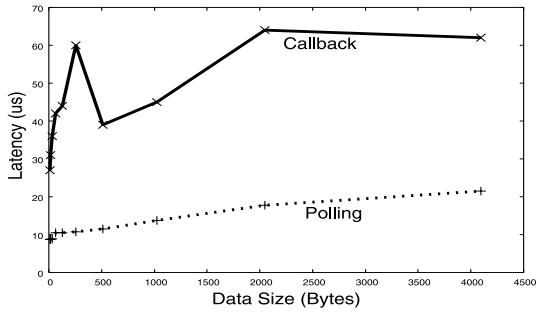

図 4 DAPL 片道レイテンシ
Fig. 4 DAPL one-way latency.

3.3 DAPL 実装における課題

3.2 節の結果より，SRP イニシエータからの性能を向上させるためには，1）4 KB × 32 での RDMA スループット性能の向上，2）Reg Mem, Dereg Mem 遅延時間の削減，3）メッセージ通信レイテンシの削減，が有効であることが分かる．これらは WSS と組み合わせた SRP イニシエータだけではなく，RDMA を前提にした SRP や iSER などのストレージプロトコルでのイニシエータ& ターゲット実装における課題に

☆ 実験した範囲では 4 KB × 32（128 KB）より大きなバッファは渡されてこなかった．
☆☆ DAPL での dat_lmr_create 相当．
☆☆☆ Host Channel Adapter

なると著者らは考えている．

4. Kernel PM2 の設計と評価

4.1 課題の解決方法

3.2 節の評価で DAPL は特に Callback 時の処理負荷が重く，ソース規模も大きくて複雑なため 3.3 節で提示した課題を解決するベースとしては不適当と著者らは考えた．そこで DAPL で使用した IB 4x 対応富士通製 HCA をサポートしており，DAPL よりソース規模が小さい PM2[3] を使用することにした．PM2 は DAPL より通信性能が良いこと[11] も分かっており，PM2 を使用することでレンテンシ削減の課題は解決できると考えた．

4 KB × 32 での RDMA スループット性能の向上および Reg Mem，Dereg Mem 遅延時間削減に関しては，性能を十分に引き出すために HCA firm レベルでの解決を検討し，4 KB × 32 の離散領域を 1 つのメモリハンドルとして firm に登録して firm 内で領域を切り替えて RDMA を行う方法を考えた．こうすれば 1 回の Reg Mem，Dereg Mem 実行で 4 KB × 32 の離散領域への登録が完了し，firm レベルでアドレス切替えを行うため連続領域に対して RDMA を行うのに近い性能が期待できる．

4.2 Kernel PM2 の設計

SRP イニシエータの実装を行うにはカーネルからのアクセス機能が必要になる．PM2 にはカーネルからの API がないので新たに Kernel PM2 を規定して実装した．Kernel PM2 で User PM2☆から追加&修正した主な機能は，1）離散領域登録解放機能，2）Callback によるイベント通知機能，である．このうち離散領域登録解放機能は 4.1 節で示した firm 修正に対応する機能である．

今回設計した Kernel PM2 の構成を **図 5** に示す．User PM2/Kernel PM2 Library，PM2 driver，HCA の階層で構成される．PM2 driver は，User PM2/Kernel PM2 Library からの通信要求に応じて HCA にアクセスを行うドライバである．

4.2.1 離散領域登録解放機能

4.1 節で提示した解決方法に従い，firm 修正および対応する Kernel PM2 関数の追加を行った．firm 側には複数の離散領域を渡して 1 つのアドレスハンドルを返す機能を追加した．さらにそのアドレスハンドルを与えて RDMA を実行すると firm 内部でアドレス切替えを行いアドレスハンドルで指定される複数の離散領域にデータを転送する機能を追加した．

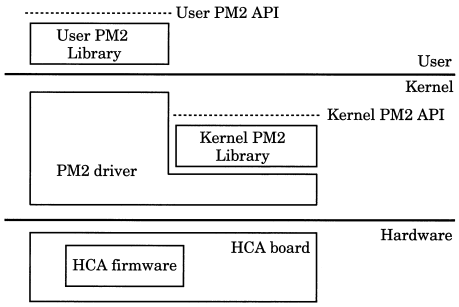

図 5 Kernel PM2 の実装
Fig. 5 Design of Kernel PM2.

```
int kpmMLockV(kpmContext *ctxt, int rmt_node,
              int sgnum, struct scatterlist sg[],
              kpmAddrHandle *hndl_out);
int kpmMUnlockV(kpmContext *ctxt, kpmAddrHandle hndl);

struct scatterlist {
        char *address;
        struct page *page;
        unsigned int offset;
        dma_addr_t dma_address;
        unsigned int length;
}
```

図 6 kpmMLockV/kpmMUnlockV の仕様
Fig. 6 Specification of kpmMLockV/kpmMUnlockV function.

Kernel PM2 側は複数の離散領域を渡して 1 つのアドレスハンドルを返す kpmMLockV/kpmMUnlockV（**図 6** 参照）を規定した．kpmMLockV は複数の離散領域へのアドレスと長さが書き込まれた scatterlist へのポインタを引数にとり，1 つのアドレスハンドルを返す．

kpmMLockV で獲得したアドレスハンドルを kpmReadWithOffset/ kpmWriteWithOffset☆☆に与えるとそのアドレスハンドルに対する firm への RDMA 要求が呼び出され，アドレスハンドルに対応した複数の離散領域への RDMA が実行される．

kpmMLockV により登録された領域は，RDMA 転送時には連続した 1 つの領域として操作できる．したがって，4 KB × 32 の離散領域を転送する際には kpmMLockV，kpmReadWithOffset または kpmWriteWithOffset，kpmMUnlockV を各々1 回発行すればよい．このため 3 回の firm 呼び出しによって離散領域が転送できる．

4.2.2 Callback によるイベント通知機能

User PM2 における送受信完了等のイベント待ちは Polling のみである．カーネル内部で CPU を占有する

☆ PM2 で規定された仕様をここでは User PM2 と呼ぶことにした．

☆☆ User PM2 の pmRead/pmWrite と同等な関数．offset を与えることで MLock された任意のアドレスから転送を指定できることが異なる．

```
void rdma_write(kpmContext *kpmc,
                struct scatterlist *sg,
                int sgcnt,
                int len,
                kpmAddrHandle rmp)
{
    int err;
    kpmAddrHandle lmp;

    kpmMLockV(kpmc,0,sgcnt,sg,&lmp);
    kpmWriteWithOffset(kpmc,0,rmp,0,lmp,0, len,
                       (kpmCbWrite)writeDoneCb,NULL);
    kpmMUnlockV(kpmc,lmp);
    return;
}

void writeDoneCb(kpmContext *kpmc,
                 int err,
                 void *arg)
{
    printk("##### WRITE DONE(CB)\n");
    return;
}
```

図 7 RDMA Write の実装例
Fig. 7 The example of RDMA Write implementation.

Polling でのイベント待ちはカーネルの他処理に影響を与えるため Kernel PM2 での Polling でのイベント待ちは望ましくない．この問題を解決するため，カーネル内部からのメッセージ送受信，RDMA 完了通知をあらかじめ登録された関数に対して Callback する機能を実装した．

4.2.3 Kernel PM2 における送受信処理

Kernel PM2 における RDMA Write 実装例を **図 7** に示す．User PM2 との違いは scatterlist によって SCSI 上位から渡された複数の離散領域を kpmMLockV を 1 度呼び出すことで Reg Mem してしまうこと，および User PM2 の pmWrite に相当する kpmWriteWithOffset に実行完了時に呼び出す callback 関数へのポインタを加えることである．さらに pmWrite ではアドレスハンドルの先頭領域からしか RDMA を行うことができなかったが，kpmWriteWithOffset ではアドレスハンドル内の任意の位置から RDMA できるよう仕様拡張も行った．pmRead に対応する kpmReadWithOffset も同等の仕様拡張を行った．

4.3 性 能 評 価

4.3.1 試作システムの概要

WSS を **表 2** に示した環境で動作させた．NIC は DAPL での評価と同じく InfiniBand 4x 対応富士通製 HCA に今回新たに開発した PM2 専用ファームウエアをインストールして使用した．PM2 専用ファームウエアは IB-Ether ブリッジを介して 10 GbE スイッチで接続される仕様になっている（**図 8** 参照）．今回の評価までに IB-Ether ブリッジを入手できなかったため対向接続で評価を行った．対向接続になったため，SE と CE は同一ノード上で動作させた（**図 9** 参照）．

4.3.2 基礎性能評価

WSS からの性能評価を行う前に DAPL で問題となったメッセージ通信レイテンシ，Reg Mem/Dereg Mem 遅延時間，離散領域への RDMA 性能の評価を行った．

表 2 PM2 試作システムの基本仕様
Table 2 Specification of PM2 prototype system.

PC	CPU: Xeon 2 GHz × 2 Chipset: Server Works GC LE Memory（SE，CE 側）: 2 GB Memory（SRP イニシエータ側）: 1 GB
Network	Nic: InfiniBand 4x 対応富士通製 HCA User PM2: 自主開発 Kernel PM2: 自主開発
OS	SCore 5.6.1（Linux 2.4.21）

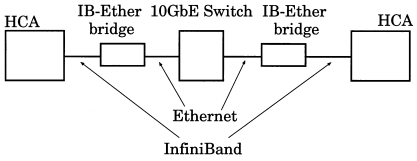

図 8 IB-Ether ブリッジの概要
Fig. 8 IB-Ether bridge structure.

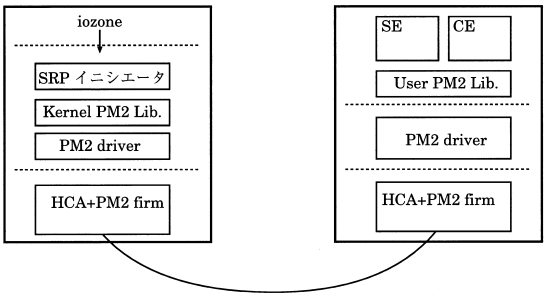

図 9 PM2 での実験環境
Fig. 9 PM2 experiment system.

PM2 レイテンシに関しては，SRP イニシエータ〜SRP ターゲット間で発生する Kernel〜User 通信と WSS 内部で発生する User〜User 通信での通信レイテンシを測定した．その結果 15-20 us 前後となり DAPL Callback の 1/3〜1/4 となることが確認された（**図 10** 参照）．

4.2.1 項で説明した離散領域登録解放機能の遅延時間を測定した．1 つの離散領域を 1 KB，2 KB，4 KB とし離散領域数を 32 までで kpmMLovkV/kpmMUnlockV の遅延時間の測定を行ったところ領域数にかかわらずつねに 20〜22 us 程度となった（**表 3** 参照）．

kpmMLovkV で登録した領域への RDMA 性能の測定も行った．1 つの離散領域が 1 KB，2 KB，4 KB の場合で離散領域数を 32 まで変化させて測定を行ったところ，4 KB × 32 Read で 579 MB/sec の性能が得られた（**図 11**）．

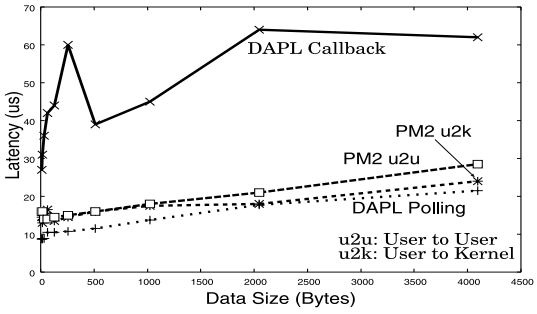

図 10 PM2 片道レイテンシ
Fig. 10 PM2 one-way latency.

表 3 kpmMLovkV 遅延時間
Table 3 kpmMLovkV latency.

	kpmMLovkV	kpmMUnlockV
1 KB × 1〜32	22 us	20-21 us
2 KB × 1〜32	22 us	20-21 us
4 KB × 1〜32	22 us	20-21 us

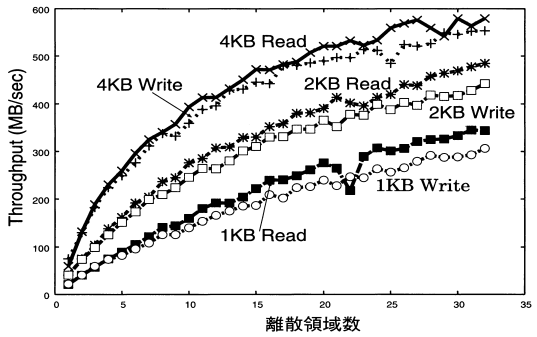

図 11 離散領域からの RDMA 性能
Fig. 11 The RDMA performance from distributed domain.

4.3.3 iozone 性能評価

4.3.3.1 概　　要

総データ転送量を 512 MB にして 1 回のデータサイズを 4〜128 KB まで変化させて iozone を実行した．その結果 Read で 226 MB/sec，Write で 166 MB/sec の性能が得られた．kpmMLockV の効果を検証するため，kpmMLock を使用した場合の性能も測定した．その結果 Read/Write ともに 20 MB/sec 程度の性能しか得られないことが分かった（**図 12**）．kpmMLock で転送を行った場合 4 KB 単位での RDMA となり 図 11 より Read 59 MB/sec，Write 75 MB/sec がネットワーク上の上限値となる．また，kpmMLock が複数回実行されるため Reg Mem にともなう遅延時間も増加すること，WSS の実装が kpmMLock 転送で最適化されていないこともあり 20 MB/sec 程度の性能となった．DAPL より性能が低い理由は，DAPL は 4 KB

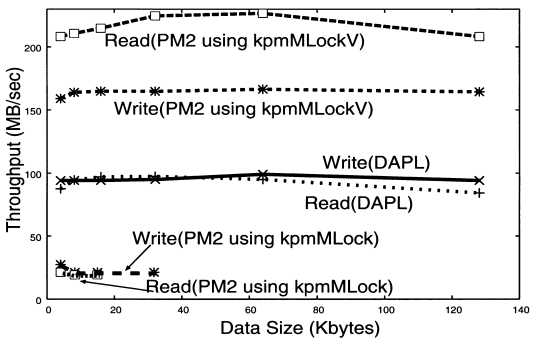

図 12 iozone 実行結果（1）
Fig. 12 Iozone execution result (1).

RDMA で 200 MB/sec 以上の性能（図 2 参照）が出ているためである．なお，データが 32 KB までしか取得できていないのは，WSS および PM2 の実装問題である．クライアント側のキャッシュヒットの影響を調べるため WSS 側の統計情報を利用して WSS から転送したデータ量の調査も行い，すべて WSS からのデータ転送であることも確認した．

4.3.3.2 Read/Write 処理の内訳

Read 128 KB 実行時のシステム内部の遅延時間測定も実施した．その結果 3.3 節で問題になった RDMA 性能，Reg Mem 遅延時間，通信レイテンシが大幅に改善され，SRP イニシエータから WSS に SRP request を出して SRP reply が返ってくるまでのスループットが 454 MB/sec となることが分かった．SRP イニシエータで実行される Reg Mem 等の遅延時間を加えてもスループットが 391 MB/sec に達することが分かった．カーネル空間からユーザ空間へのコピーのため iozone から見える性能が 227 MB/sec となることも分かった（**図 13**）．今回の評価では機材の問題で SE と CE を同一ノードで実行させたため 図 3 との公正な比較はできない．図 10 より PM2 での片道レイテンシが 15 us 程度であることが分かっている．それで机上値になってしまうが，図 13 の SE+CE に本来発生する 15 us の通信遅延時間を加えた 297 us，431 MB/sec が SRP イニシエータから見える WSS の性能となる．図 3 より DAPL では 925 us，138 MB/sec であり，Reg Mem の遅延時間も 260 us（DAPL）から 22 us（PM2）へ大幅に短縮できており著者らが提案した方法の有効性が示された．Write 128 KB も SRP イニシエータから先は RDMA の向きが Read と逆になることを除くと同じ遅延時間で動作したことも報告しておく．今回の評価では使用した OS（表 2 参照）の特性のため，著者らが調査した限りでは大部分の Scsi_Cmnd が 4 KB × 32 の離散領域を渡してきた．そのため 4 KB × 32

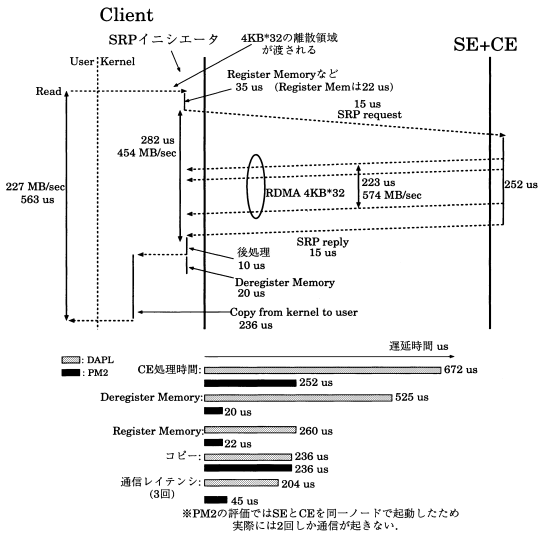

図 13 iozone 実行時の内部遅延時間（Read 128 KB）
Fig. 13 Internal latency at the time of iozone execution (Read 128 KB).

を中心に評価を行ったが，ここまでの測定結果より本稿で提案した方法は 4 KB × 32 だけではなく任意の離散領域が渡されてくる場合においても有効だと著者らは考えている．

4.3.3.3 性能チューニング

デバイスあたりの命令多重度（cmd_per_lun）☆を 1 から 2 に増やし，readahead ☆☆のページ数を 31 から 255 に増やした環境での iozone の測定も行った．その結果 Read で 325 MB/sec，Write で 197 MB/sec の性能が得られた（**図 14**）．cmd_per_lun は Write 性能向上に，readahead は Read 性能向上に効果があった．Read 325 MB/sec 時の SRP イニシエータ内部の動作確認を行い，複数の命令がオーバラップして動作していることが確認できた．SRP イニシエータを含むクライアント側の処理と WSS 側の処理がオーバラップして動作したため 図 12 と比較して大幅に性能が向上した．Read 性能を支配しているのはクライアント側のカーネル空間からユーザ空間へのコピー（550 MB/sec）と WSS のスループット（454 MB/sec）であるので，コピーと WSS へのアクセスをより効果的にオーバラップさせることで 325 MB/sec より性能を伸ばすことは可能だと考えている．これは Read 実行時のプロファイルを取得したところ（**表 4** 参照）全体の約 2 割が idle であることからも裏付けられる．idle の次に割合の高い file_read_actor は内部でカーネル空間からユーザ空間へのコピーを実行している．

☆ Scsi_Cmnd を同時に渡せる数.
☆☆ Read 時にブロックを先読みする機能.

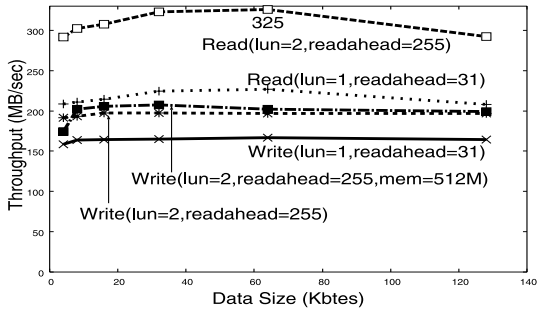

図 14 iozone 実行結果 (2)
Fig. 14 Iozone execution result (2).

表 4 Read 実行時のカーネルプロファイル
Table 4 Kernel profile at the time of Read execution.

function	CPU time (%)
IDLE	20.46
file_read_actor	14.10
上記以外の OS 実行時間	65.02
USER 実行時間	0.42

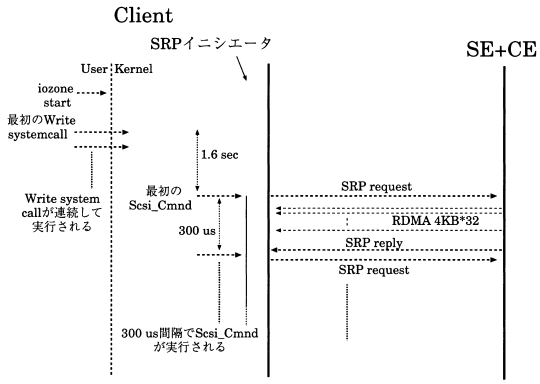

図 15 クライアント側 Write 動作内訳
Fig. 15 Client side Write operation details.

4.3.3.4 Write 性能が伸びない理由について

Write 性能（197 MB/sec）が Read ほど伸びない理由の調査も行った．iozone を strace から実行することで，最初の write systemcall を実行してから実際に SRP イニシエータに Scsi_Cmnd が届くまでの遅延時間を測定した．その結果，最初の write systemcall 実行から最初の Scsi_Cmnd を SRP イニシエータが受け取るまでに 1.6 sec かかっていることが分かった．最初の Scsi_Cmnd が届いた後は約 300 us 間隔で Scsi_Cmnd が実行されていることも分かった（**図 15**）．iozone からの write systemcalll 実行でまず buffer cache に書き込み buffer cache が枯渇するまでは SRP イニシエータに I/O 要求をカーネルが出さないことが 1.6 sec の

表 5 Write 実行時のカーネルプロファイル
Table 5 Kernel profile at the time of Write execution.

function	CPU time (%)
IDLE	29.73
generic_file_write	11.10
上記以外の OS 実行時間	58.45
USER 実行時間	0.72

理由だと考えている．buffer cache 量を減らすことで 1.6 sec を削減できるのではないかと予想し，クライアント側の主記憶を 512 MB にして実行してみた．その結果ピークが 208 MB/sec となり 10 MB/sec 程度向上したが大幅な向上にはならなかった（図 14）．次に Write 実行時のカーネルプロファイルを取得してみたところ（**表 5**）全体の 3 割が idle であることが分かった．idle の次に割合が高い generic_file_write は内部でユーザ空間からカーネル空間へのコピーを実行している．今回の評価では idle が発生する原因の特定までは行わなかったが，Write 性能を向上させるためには idle 時間の削減が有効だと考えている．cmd_per_lun を 3 以上にした測定も行ったが，2 の場合と大きな違いがなかったことも報告しておく．

今回の評価では試さなかったが，Read，Write ともに Direct I/O [10] を使うことで性能がさらに向上する可能性がある．Kernel PM2 の効果でボトルネックがカーネル空間〜ユーザ空間コピーを含めたクライアント側実装に移ってきており，Direct I/O 使用によるコピー削減が効果的と考えられる．なお，Direct I/O における 1 回のデータサイズをある程度大きくしておかないと 図 11 の結果より十分な RDMA 性能が得られなくなるので注意する必要がある．

5. 関 連 研 究

RDMA を使用した Storage I/O の研究に関して述べる．

オハイオ州立大学では IB 上で iSCSI プロトコルをベースにデータ転送部を RDMA に置き換えた試作を行い，最大 417 MB/sec のスループットを取り出すことに成功している [7]．Reg Mem 遅延時間の問題に関しては，Zero-Cost Kernel Memory Registration （ZKMR）を提案している．これは OS 初期化時に SCSI driver が転送に使用する領域をあらかじめ HCA に登録しておくことで遅延時間の削減を図っている．複数の離散領域への RDMA に関しては，1 回の RDMA におけるイニシエータ側の領域がなるべく連続になるように scatterlist を再構成してターゲット側に転送要求を出すことで解決している．

ZKMR に関しては Reg Mem 遅延時間を削減する有効な方法の 1 つであるが，著者らが提案したカーネル通信 API を機能拡張する方法と比較してカーネル内部関数の変更が必要で移植性に問題が残ると考えている．イニシエータ側の領域がなるべく連続になるように scatterlist を再構成する提案に関しては，カーネル上位で生成される scatterlist に性能が左右されることになり，著者らが提案した方法のようにつねに同じ性能が得られないと考えられる．最大 417 MB/sec のスループットに関しては，測定条件の記述が不十分なため著者らの測定結果との比較は困難であるが，ドライバレベルの性能ということならば著者らも 454 MB/sec のスループットを達成しており，同等レベルであるといえる．

Reg Mem 遅延時間の問題については，論文で提案があった ZKMR 以外に Memory Registration Cache（MRC）と Fast Memory Registration（FMR）に関する紹介も記述されている．MRC は Dereg Mem が実行されてもすぐに実際の解放処理を行わずにしばらくキャッシュしておく方法であり，同じ領域が繰り返し利用される場合は効果があると考えられる．このアイデアは新情報処理開発機構☆が開発した PM2 にも Pin-down cache [9] として使われており，同じ領域に対して何度も RDMA するアプリケーション全般に有効なアイデアだといえる．著者らも kpmMLockV により 22 us まで Reg Mem 遅延時間を削減することに成功したが，さらに遅延時間を短縮する必要が生じたら MRC を採用することを考えている．

FMR はあらかじめ登録が必要なリソースの確保を行っておき，実際に登録が必要になると確保してあるリソースを使用して登録処理を行う方式である．FMR は iSER and RDMA over IP の実装等で使われている．

Voltaire 社では IB 上で iSER 実装に関する White Paper [8] を発表している．ここでは主に管理，運用面で SRP と比較した場合の iSER の優位性を提示している．

6. ま と め

DAPL 実装での課題であった，1）離散領域への RDMA 性能向上，2）Reg Mem 遅延時間の削減，3）通信レイテンシの削減は著者らが設計した Kernel PM2 により解決され，SRP イニシエータ内部から 454 MB/sec の性能が得られた．iozone からは Read で 325 MB/sec，Write で 208 MB/sec を取り出すこ

☆ 現在は PC クラスタコンソーシアムが開発を行っている．

とに成功し，Kernel PM2 に実装した離散領域を同時に Reg Mem し RDMA を実行する方法の有効性が示された．

DAPL 実装で問題となった主に通信 API による課題は解決され，ボトルネックはクライアント側カーネル実装に移った．今回の報告よりさらに性能を引き出すには Direct I/O 使用によるユーザ空間〜カーネル空間コピーの削減やカーネル内部の I/O 処理と WSS へのアクセスをより効果的にオーバラップさせることが重要であることも分かった．

今回は WSS を用いて評価を行ったが，本稿で報告した内容は SRP や iSER 等 RDMA を用いて Storage I/O を行うイニシエータ/ターゲット実装全般にあてはまるものであると著者らは考えている．

なお，本研究の一部は，東京大学石川研究室と富士通研究所との共同研究契約に基づいて行われた．

参 考 文 献

1) 大江和一，渡辺高志，西川克彦：10 Gbps 環境における Wire Speed Storage（WSS）の性能評価，SACSIS2004（ポスター）
2) Linux Infiniband Project. http://infiniband.sourceforge.net
3) SCore クラスタシステムソフトウエア，PC クラスタコンソーシアム. http://www.pccluster.org
4) IOzone Filesystem Benchmark. http://www.iozone.org/
5) http://www.t10.org/
6) OpenIB. org. http://www.openib.org/
7) Liu, J., Panda, D.K. and Banikazemi, M.: Evaluating the Impact of RDMA on Storage I/O over InfiniBand. http://nowlab.cis.ohio-state.edu/project/mpi-iba/ publication/ liu_san3.pdf
8) Voltaire: iSCSI RDMA による InfiniBand ファブリックの高性能 SAN 接続. http://www.voltaire.com
9) Tezuka, H., O'Carroll, F., Hori, A. and Ishikawa, Y.: Pin-down Cache: A Virtual Memory Management Technique for Zero-copy Communication, *IPDPS* (1998).
10) http://www.linuxforum.com/linux-scsi/dio.html
11) 住元真司，成瀬 彰，久門耕一，細江広治，清水俊幸：PM/InfiniBand-FJ：InfiniBand を用いた大規模 PC クラスタ向け高性能通信機構の設計，*SACSIS* (2004).
12) 大江和一，渡辺高志，西川克彦：Wire Speed Storage（WSS）アーキテクチャ，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG 1(ACS4), pp.100–114 (2004).
13) 大江和一，渡辺高志，西川克彦：Wire Speed Storage（WSS）アーキテクチャ，*SWoPP2002*, 信学技法， Vol.102, No.275, pp.1–6 (2002).
14) 渡辺高志，大江和一，西川克彦：ワイヤスピードストレージアーキテクチャ（WSS）における SCSI RDMA Protocol（SRP）の実装と評価，*SWoPP2002*, 信学技法，Vol.102, No.275, pp.7–12 (2002).

(平成 16 年 7 月 23 日受付)
(平成 16 年 11 月 21 日採録)

大江 和一（正会員）
1988 年九州大学工学部情報工学科卒業．同年富士通株式会社に入社．現在，株式会社富士通研究所にてストレージシステムの研究開発に従事．

渡辺 高志
2000 年早稲田大学大学院理工学研究科情報専攻（修士）修了．同年株式会社富士通研究所に入社．現在，株式会社富士通研究所にてストレージシステムの研究開発に従事．

中島 耕太（正会員）
2002 年九州大学大学院システム情報科学府情報工学専攻修士課程修了．同年富士通（株）入社．現在，株式会社富士通研究所勤務．高速インタコネクトに関する研究・開発に従事．

西川 克彦
1982 年東京大学工学部電子工学科卒業．同年株式会社富士通研究所に入社．以来図面認識の研究，ビデオサーバの研究開発等を経て，現在サーバ・ストレージシステムの研究開発に従事．映像情報メディア学会会員．