社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－ARC－182 (14)
2009－HPC－119 (14)
2009／2／27

予約利用可能なオブジェクトベース・ストレージの設計

谷村勇輔[†] 鯉江英隆[†,††] 工藤知宏[†]
小島功[†] 田中良夫[†]

予約に基づいたアクセスを行うことで，データアクセスの性能などのサービスレベルを保証できる並列ファイルシステムの開発に向けて，その下位コンポーネントとなるオブジェクトベース・ストレージの設計を行った．設計においては，予約時間に確実にデータが書き込めることを保証するために，オブジェクトファイルシステムであるEBOFSを拡張して，ディスクスペースを事前に確保する仕組みを追加した．また，予約時間においてオブジェクトへのアクセスに対する性能保証が必要になるため，ディスクおよびEBOFSの性能について予備実験を行い，安定して高スループットを得るための課題を明らかにした．今後はこれらの結果をもとに設計の一部を改善し，オブジェクトベース・ストレージの開発を進める予定である．

Design of The Object-based Storage Allowing Data Access Reservation

Yusuke Tanimura,[†] Hidetaka Koie,[†,††] Tomohiro Kudoh,[†]
Isao Kojima[†] and Yoshio Tanaka[†]

The final goal of this research is to develop a parallel file system which guarantees the service level in data access (e.g. performance) if a user makes a reservation for the access. In this work, we present design of an object-based storage which is a lower component of the targeted parallel file system. Our design uses EBOFS, which is an object file system, and extends the architecture by adding a function of the advanced disk space reservation. We also present the preliminary experiment results which show the performance stability of EBOFS, because the object access must be stable during the reserved time in our scenario. The results raise several issues for gaining high throughput from disks constantly. Improvement and implementation of our current design by considering the results are our future work.

1. はじめに

複数のストレージサーバを束ねて高性能かつ大規模な容量を提供する並列ファイルシステムは，扱うデータ規模の増大などに対応するため，HPC分野に留まらず，ビジネス分野でも注目を集める技術である．しかし，ビジネス用途のシステムでは，システムが提供するサービス品質が非常に重要であり，そのシステムの構成要素となる並列ファイルシステムに対して一定以上のサービス品質を要請する場合がある．例えば，同時刻に複数のアクセスが集中することにより，並列I/Oに使われるディスクが競合して性能の低下が生じたり，ディスク容量の不足により，並列I/Oに使えるディスク数が制限を受けて必要な性能が確保できないといったことが起きないようなサービスを提供しなくてはならない．この問題に関して，既存の並列ファイルシステムでは並列I/Oにおけるディスクアクセスのスケジューリングやディスク使用量の平滑化が研究されているが[1),2)]，100%に近い性能保証を行うことは容易ではない．

そこで，我々は予約利用を前提とすることで確実に性能を保証するアプローチをとる．つまり，並列ファイルシステムがストレージ資源へのアクセスを時間軸で管理し，ユーザはデータアクセスの日時と必要性能を予約しておくことで，予約時間においては指定した性能を享受できる仕組みを実現する．性能保証はストライピングに用いるI/Oを制御することで行い，また，データが書き込めることも同時に保証するために書き込みスペースの予約もできるようにする．

本研究では現在，このような並列ファイルシステムの開発の第一段階としてオブジェクトベース・ストレージの開発を行っている．オブジェクトベース・ストレージは，並列ファイルシステムのストレージサーバにおいてデータを格納するのに用いられる下位コンポーネントであり，HPC向けのLustre[3)]やPanFS[4)]，Ceph[2)]などの並列ファイルシステムでも利用されている．下位コンポーネントをオブジェクトベース・ストレージと

† 産業技術総合研究所 / National Institute of Advanced Industrial Science and Technology (AIST)
†† 数理技研 / SURIGIKEN Co., Ltd.

して抽象化することにより，ディスク領域の管理とファイル断片の管理レイヤを分離し，前者をストレージデバイス側の実装，後者を上位のファイルシステム側の実装で行える．そのため，より高度なファイル共有やセキュリティ機能を上位のファイルシステムで実装可能であり，同様に，予約機能についても実装を容易にするなどの利点があると考えている．

本稿では，上述の予約機能を上位コンポーネントで実現するための，ディスクベースの事前確保を行う仕組みを備えたオブジェクトベース・ストレージの設計について報告する．さらに，性能保証に向けて，開発中のオブジェクトベース・ストレージの性能の安定性について予備評価を行った結果を示し，今後の開発に向けた課題を報告する．

2. 予約利用できる並列ファイルシステム

2.1 概　　要

予約利用できる並列ファイルシステムとは，性能と日時を指定した予約がなされていれば，予約時間において確実に性能を保証する機能を備えるファイルシステムである．予約要求時に指定した性能が得られるようにI/O時のストライピング・パラメータが自動決定され，アクセスされるストレージサーバおよびストレージデバイス（ディスクなど）の予約がなされる．そして，予約時間においてそれらのストレージデバイスに対する優先的なアクセスを可能にする．また，書き込みアクセスはストレージデバイスに空きスペースがなければ，結果として性能を保証することができなくなるため，書き込みスペースの予約も合わせて行える機能を提供する．

2.2 利　用　例

このようなファイルシステムはサービス品質を管理し，品質によって課金額を設定するようなビジネスなど，一定の応用範囲があると考えている．特に，我々は次世代の高精細映像配信が要求する，高バンド幅のストリームを多数のユーザに提供するシナリオにおいて，開発したファイルシステムを適用することを目指している．そのシナリオでは，利用者が時刻を指定して映画などの見たい映像の視聴予約を行う．サービス提供側は視聴予約に合わせてストレージとネットワークを同時予約し，映像配信環境を整える．例えば，視聴開始までにコンテンツをリポジトリから配信拠点まで確実に転送したり，視聴時間において，コンテンツが格納されているストレージから視聴端末までのスループットを保証したりするのに予約機能が利用される．

2.3 システム構成とオブジェクトベース・ストレージの位置づけ

図1に開発する並列ファイルシステムの構成を示す．これは，ごく一般的な並列ファイルシステムの構成であり，クライアント，ストレージサーバ，管理サーバの3つのコンポーネントからなる．予約はクライアントから管理サーバに対して，ファイルのアクセス日時とアクセス時に保証してもらいたい性能を指定して予約要求が実行されることで行われる．管理サーバは要求された性能に従ってI/Oの並列度を決定し，そのI/Oに用いるストレージサーバを予約する．予約が成立すれば，予約チケットがクライアントに返される．その後，予約時間において，クライアントは予約チケットを提示して並列ファイルシステムにアクセスすることで，予約されているストレージサーバを優先的に利用することができる．

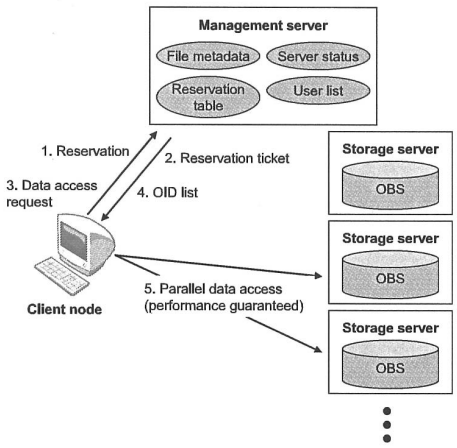

図1　予約利用できる並列ファイルシステムの構成

本システムでは，先に述べたように，並列ファイルシステムのストレージサーバの実装にオブジェクトベース・ストレージ（OSD）を用いる．図1の管理サーバはファイルメタデータとして，ファイルを構成するオブジェクト識別子（OID）を管理する．一方，オブジェクトのディスク上の物理レイアウトは個々のオブジェクトベース・ストレージにて管理する．オブジェクトベース・ストレージは性能予約の機能を実現するために，さらに次の機能を持たなくてはならない．

- 性能保証を行うため，各オブジェクトへのアクセスに対して安定した性能を提供できなければならない．
- データが書き込めることを保証するため，オブジェクトのディスクスペースを事前に確保する機能を備えていなければならない．
- 予約期間が終了しているオブジェクトのディスクスペースを自動的に回収する仕組みを備えていなければならない．

性能については本研究では映像配信への応用に焦点を当て，扱うファイルサイズは数MB～数GBと大きく，逐次的な読み出しが中心であることを想定する．以降では，これらの要件に基づいて設計したオブジェクトベース・ストレージについて説明する．

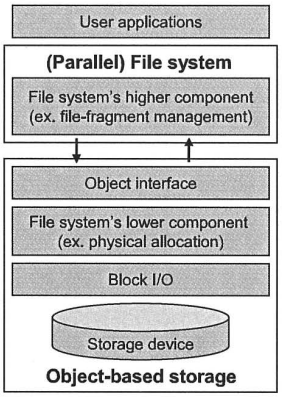

図 2 オブジェクトベース・ストレージのモデル

3. ディスクスペースの予約が可能なオブジェクトベース・ストレージの設計

3.1 オブジェクトベース・ストレージ

オブジェクトベース・ストレージは**図 2** に示すように，従来的なブロック I/O のインタフェースでなく，格納するデータと関連するメタデータを 1 つのオブジェクトとしてアクセス可能にするインタフェースを提供する[5]．これは，ブロック I/O に比べてアクセス性能が低下する反面，メタデータが分離され，その上位の並列ファイルシステムにおいてスケーラブルなセキュリティを構築するのを容易にする．また，ファイルベースの I/O に比べると，ずっと高速なアクセスを提供可能である．通常，オブジェクトはデータとメタデータから構成され，サイズは可変である．メタデータにはファイルシステムなど外部から登録・アクセスができるものとオブジェクトベース・ストレージ内部で管理されるものの 2 種類がある．

オブジェクトベース・ストレージはファイルシステムの設計を容易にすることが大いに期待され，Object Storage Device（OSD）として T10[6] や SNIA[7] などで標準化が進められている．また，Intel の OSD/iSCSI 参照実装[8] などのソフトウェアでの実装や，標準インタフェースを備えず，独自インタフェースを持つオブジェクトベース・ストレージ／ファイルシステムの実装が存在する．本研究では，その中の 1 つである EBOFS（Extent B-tree based Object File System）を利用し，予約に基づくデータアクセスに対応するため，ディスクスペースの事前確保の仕組みを拡張実装したオブジェクトベース・ストレージを開発する．

3.2 EBOFS

EBOFS[9] は，オブジェクトインタフェースを備えるファイルシステムであり，その実装において Extent と B+tree を利用している．ブロックデバイスに直接アクセスしてオブジェクトを格納するが，ユーザ空間で動作し，カーネルを修正することなく利用可能である．EBOFS の特徴は，スーパーブロックや B+tree の管理領域を 2 つ持つことでそれらのソフトアップデートを実現していること，複数のオブジェクトをまとめて更新するトランザクションをサポートしていること，メモリキャッシュを用いてディスクへの書き込みをノンブロッキングに行うことなどである．EBOFS を並列ファイルシステムに利用した例としては Ceph[2] が挙げられる．Ceph では，EBOFS の上位に RADOS（Reliable, Autonomic Distributed Object Store）と呼ばれる分散オブジェクト・ストレージを構築し，さらにその上位にファイルシステムを構成する設計となっている．

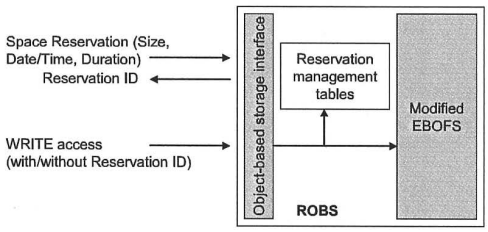

図 3 ROBS の概要

3.3 ディスクスペースの予約機能の追加

本研究ではディスクスペースの予約機能を持つ，Reservable Object-based Storage（ROBS）を開発する．ここで追加する予約機能は，予約において利用時間を指定するものであることから，ROBS では全てのディスクスペースを期限付きで（利用時間を指定して）提供するモデルを導入することとした．これは，予約したディスクスペースやそこに作られたオブジェクトに利用期限を設けるだけでなく，ディスクスペースの予約なしに作られたオブジェクトが使うディスクスペースも期限付きで提供されるものである．

図 3 に ROBS の構成概要を示す．ROBS では，予約のためにディスクスペースの管理方法が修正された EBOFS を内部で利用し，EBOFS の外側に予約管理表を持つ．ROBS のインタフェースは基本的に EBOFS をラップしたものであるが，予約に関するインタフェースを加えられる．以下に ROBS の設計の詳細を述べる．

3.3.1 ディスクスペースの管理方法

EBOFS は**図 4** のようにオブジェクトを格納する．オブジェクトの管理データを安全に更新するため，スーパーブロックと B+tree 管理用のビットマップは 2 つ用意されている．B+tree はオブジェクトの参照とフリー Extent を管理し，その領域は初期化直後はスーパーブロックの後に配置されるが，オブジェクト数が増加して領域が不足するとデータエリアに追加領域が作成される．EBOFS の Extent の最小単位（ブロック）は 4KB であり，オブジェクトにはこの単位でディスクスペースが割り当てられる．

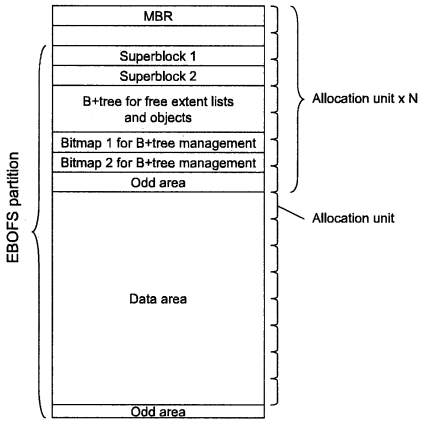

図 4 EBOFS のディスクスペースの管理方法

これに対して ROBS では，まずオブジェクトへのディスクスペースの割り当て単位を 4KB より大きなサイズに変更する．以降では，この割り当て単位を AU（Allocation Unit）と呼ぶ．AU の値は，EBOFS のフォーマット時に簡単な計測プログラムを実行して，使用するディスクの性能特性に合わせて決定するものとする．格納するオブジェクトのサイズがある程度大きければ，これによるディスクスペースの割り当て効率の低下は小さく，AU を 4KB に固定した場合よりも性能面で有利である．

次に，全ての AU のオフセットを固定にする．各オフセットは EBOFS の初期化時に決定され，ストレージデバイスの先頭（あるいは，EBOFS のパーティションの先頭）からの AU の倍数となる．AU のオフセットが固定であるため，例えばスーパーブロックや B+tree の領域等をまとめて AU の N 倍と考えると，B+tree 管理用のビットマップとデータエリアの間には半端エリア（Odd area）ができる．

予約に対応するため，ROBS では**図 5** のように「a) ディスクスペースの管理表」を持ち，時間軸に沿って未割り当ての AU の個数を記録する．また，「b) 予約表」においては予約毎に，予約したディスクスペースのサイズに対応する AU の個数を記録する．予約したディスクスペースには複数のオブジェクトを作成可能であり，各オブジェクトに対してどの AU を実際に割り当てるかについては，オブジェクトの書き込み時に決定する．

また，オブジェクトの格納においては，onode と呼ばれるオブジェクトのメタデータも格納する必要がある．onode は 1 つの AU に納まるほどの大きさであることが予想されるため，onode の格納場所としては半端領域にあるフリー Extent を優先的に割り当てる．半端領域が存在しない場合は AU を割り当てることにする．

3.3.2 予約処理の流れ

予約は上位のツール（ファイルシステムなど）が

a) Disk space management table

Time	# of free AUs
20090110190000	1000
20090114090000	800
20090115090000	1000

c) Reservation-Object mapping table

Reservation ID	Object ID
rid0001	oid003

b) Reservation table

Reservation ID	Start time	End time	# of reserved AUs	# of used AUs
rid0001	20090114090000	20090115090000	200	0

図 5 予約管理表の例

ROBS に対して Reserve 要求を発行することで行うものとする．Reserve 要求には予約したいディスクスペースのサイズ，利用開始時刻，利用終了時刻が含まれる．Reserve 要求に対して，ROBS は要求された時間帯の空き AU の個数を予約管理表より取得し，要求されたサイズが確保できれば予約を入れる．そして，該当予約に対応する識別子を発行し，予約成立の返答として Reserve 要求元に送信する．

予約したディスクスペース（以降，予約スペースと記述）に書き込みを行う場合には，書き込みに先立って，予約時に発行された予約識別子を ROBS に提示する．ROBS は予約識別子を用いて予約管理表を検索し，予約スペースのサイズ（AU 数として記録）を越えない範囲で AU を割り当てる．当然ながら，割り当てた AU の分だけ，その予約識別子で使うことのできる AU 数は減少することになる．また，予約識別子と書き込んだオブジェクトの識別子の対応を図 5 の「c) 予約－オブジェクト関連表」に記録する．

オブジェクトが消去された場合には，該当オブジェクトに割り当てられていた AU を回収し，先ほどとは逆に，その AU の提供元である予約スペースの AU 数を増やす．一方，利用終了時刻を過ぎた予約スペースも回収対象となり，その予約スペースから作られた全てのオブジェクトも自動的に消去する．

3.3.3 予約なしの書き込みのサポート

ROBS はディスクスペースの予約を前提としたオブジェクトの書き込みを基本としているが，予約のない書き込みも受け付ける．ただし，条件として，書き込みを受け付ける時点においてそのオブジェクトを格納するだけの空きがディスクにあり，その空きを一定期間以上利用できる場合のみとする．一定期間は，オブジェクトのデフォルトのライフタイムとして ROBS の管理者によって設定される．ROBS は，予約のない書き込みに対して，予約されていないフリーの AU を割り当て，割り当てた AU 数分のディスクスペースを自動的に予約する．そして，その予約の利用期間をオブジェクトを書き込んだ時刻からオブジェクトのデフォルトのライフタイムまでと設定する．

3.3.4 オブジェクトのライフタイム

ディスクスペースの予約機能の追加と合わせて，

表 1　実験環境

CPU	Dual-core Opteron 1.8 GHz×2
Memory	4 GB
Disk interface	SATA 1.5Gb/s
Disk-1 (HDD)	HGST Deskstar P5K500, 7200RPM, Max transfer rate: 142.25 MB/sec
Disk-2 (SSD)	Intel X25-M, MLC, Sustained Sequential Read: 250 MB/sec, Sustained Sequential Write: 70 MB/sec
OS	CentOS 5.2 (Kernel v.2.6.18)

ROBS では，格納する全オブジェクトにライフタイムを設ける．ライフタイムが切れるとオブジェクトは自動的に消去され，そのオブジェクトに割り当てられた AU は回収される．ライフタイムはオブジェクトの書き込み時に設定できるが，明示的にライフタイムが指定されない場合は，ROBS の管理者によって設定されるデフォルトのライフタイムが使われる．また，オブジェクトが予約スペースから作られていれば，設定可能な最長ライフタイムは予約スペースの利用期間が終了するまでとなる．オブジェクトがディスクスペースの予約なしに作られている場合は，最長ライフタイムは ROBS の管理者によって設定された最長ライフタイムが適用される．ライフタイムの更新可能の有無や更新回数は ROBS の管理者によって決められ，更新が許されているような場合は，定期的にライフタイムを更新することで，オブジェクトを長期間格納することが可能である．

4. オブジェクトベース・ストレージの性能の安定性

4.1 実験方法

並列ファイルシステムにおいて性能保証を行うためには，下位レイヤで用いられるオブジェクトベース・ストレージが一定の性能を保証できなくてはいけない．そこで，オブジェクトベース・ストレージの開発に利用する EBOFS を用いて，オブジェクトサイズがある程度大きく，逐次的なアクセスを行う場合の性能の安定性について予備実験を行った．実験では，まずオブジェクトサイズによる性能の違いを調査し，その上でアクセスサイズを固定にし，連続的にアクセスを行った場合の性能の安定性を調べた．さらに，従来的な HDD を用いた実験だけでなく，異なる性能特性を持つと考えられる SSD（Solid State Disk）についても同様の実験を行った．

用いた実験環境は**表 1** の通りである．また，ディスクの Write Cache は無効とし，カーネルの I/O スケジューリングは CFQ（Completely Fair Queuing）を用いた．

4.2 オブジェクトサイズによる性能の違い

図 6 はオブジェクトサイズを変えて，オブジェクトの新規書き込みと読み出し性能をそれぞれ調べた結果である．オブジェクトサイズが小さい時には，メタデータアクセスのオーバーヘッドがアクセス時間全体に占める割合が大きいため低スループットであるが，オブジェクトサイズが大きくなるにつれてスループットが上がる．Disk-1 ではオブジェクトサイズが 8MB を越えるあたりまでスループットが上昇した．一方，Disk-2 は Disk-1 より常に良い性能を示し，オブジェクトサイズが 4〜8MB となるまでスループットが上昇した．

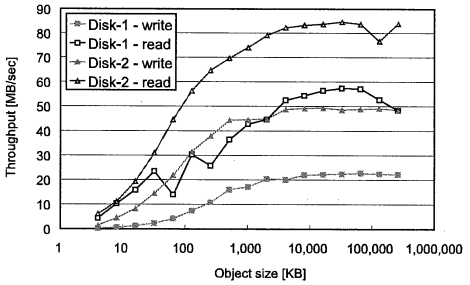

図 6　オブジェクトサイズとスループット

4.3 連続アクセスにおける性能の安定性

次に，EBOFS に連続アクセスした場合の性能の安定性を検証するために 2 種類の実験を行った．1 つ目の実験（1-obj）ではデータサイズを 16MB と固定し，同じオブジェクトに対して連続して 1024 回の追加書き込みを行い，16GB のオブジェクトの作成にかかる時間を計測した．そして，そのデータを同じく 1024 回に分けて読み出す時間を測定した．一方，2 つ目の実験（N-obj）では，同じオブジェクトにデータを書き込むのではなく，異なるオブジェクトにデータを新規に書き込むこととした．つまり，16MB のオブジェクトを 1024 個書き込むのにかかる時間，またそれらを読み出すのにかかる時間を測定した．本実験中はディスクを占有するため，異なるオブジェクトを作る場合でもオブジェクトのデータはディスク上にほぼ連続して格納されることになる．

図 7 に書き込み性能，**図 8** に読み出し性能の結果を示す．書き込みに関しては，1 つのオブジェクトに対する連続アクセスは，アクセス回数が多くなるとともに性能が大きく低下してしまった．それに対して，異なるオブジェクトにデータを書き込んだ場合はそのような性能低下が起きなかった．後者の実験では，Disk-1 の場合はスループットのばらつきが 10[MB/sec] 以内となり，その標準偏差は 1.29 であった．一方，Disk-2 のスループットのばらつきは 5[MB/sec] 以内に納まっているアクセスが多いものの，10[MB/sec] 以上性能が低下したアクセスがいくつか見られ，標準偏差は 1.88 となった．

読み出しに関してはデータの格納方法の違いによる性能差はない．スループットのばらつきは Disk-2 の方が少なく，N-obj の実験では 5[MB/sec] 以内に全てのアクセスが納まり，その標準偏差は 0.74 であった．

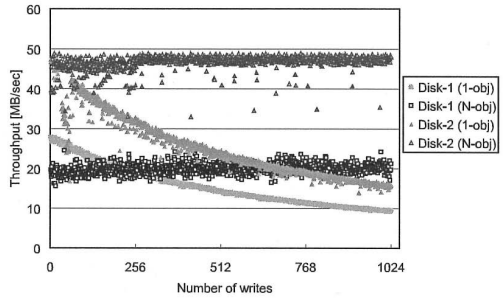

図 7 連続書き込みの性能

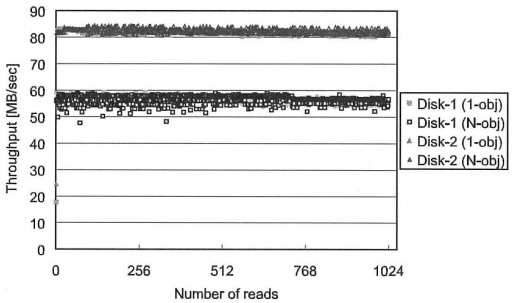

図 8 連続読み出しの性能

4.4 考　　察

本実験により，まず，ディスク性能を最大限に引き出して高スループットを得るには，各オブジェクトが一定以上のサイズでなければならないことが分かる．これはROBSのフォーマット時にAUサイズを決定するのと合わせて，簡単なベンチマークを実行して決定するのが良いと考える．次に，同一オブジェクトへの追加書き込みは徐々に性能が低下することが分かる．ROBSへのI/O単位毎にオブジェクトを作成することで追加書き込みを避けることも考えられるが，オブジェクトの分割は上位のファイルシステムにおいてメタデータ管理の手間を増やすことになる．性能低下の原因を究明し，EBOFSの改善も含めて今後の対応を考えたい．

一方，本実験で用いたディスクでは，SSDは読み出しにおいてHDDよりも安定した性能を示したが，書き込みでは時々，HDD以上の大きな性能低下が見られた．これについても原因の調査を行うとともに，ROBSやその上位のコンポーネントが持つバッファ等で性能低下を吸収できないか検討を進めたいと考えている．

5. まとめと今後の予定

本稿では，性能保証を行うことのできる並列ファイルシステムの実現に向けて，その下位コンポーネントであるオブジェクトベース・ストレージ（ROBS）の設計について報告した．ディスクスペースを事前に確保する仕組みを作るために予約やライフタイムの概念を導入し，EBOFSを拡張する形で設計を行った．一方，既存のディスクをEBOFSからアクセスした際の性能を調査し，並列ファイルシステムの上位コンポーネントに対して安定した性能を提供するための課題を示した．

今後は，それらの性能調査の結果をROBSの設計や実装に反映させ，並列ファイルシステムの上位コンポーネントの開発に取り組む計画である．そこでは，アクセス予約の管理方法やストライピングパラメータの決定方法など，性能保証を実現するための検討を進める．また，ROBSに対するアクセスの性能保証だけでなく，ROBSを実装するストレージサーバとクライアントとの間のデータパスの性能保証についても検討する予定である．

謝辞　本研究成果の一部は，文部科学省科学技術振興調整費「先端融合領域イノベーション創出拠点の形成 光ネットワーク超低エネルギー化技術拠点」委託研究によるものである．

参　考　文　献

1) Qingsong Wei, Wujuan Lin, Bharadwaj Veeravalli, and Lingfang Zeng. QoS-Aware Striping with Replication Model for Object-Based Multimedia. In *Proceedings of the Fourth International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI)*, pp. 114–121, 2007.
2) Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: A Scalable, High-Performance Distributed File System. In *Proceedings of the 7th Conference Operating Systems Design and Implementation (OSDI'06)*, pp. 307–320, 2006.
3) Lustre. http://www.lustre.org/.
4) David Nagle, Denis Serenyi, and Abbie Matthews. The Penasas ActiveScale Storage Cluster: Delivering Scalable High Bandwidth Storage. 2004.
5) Mike Mesnier, Gregory R. Ganger, and Erik Riedel. Object-Based Storage. *IEEE Communications Magazine*, August 2003.
6) T10. http://www.t10.org/.
7) SNIA. http://www.snia.org/.
8) Intel's Open Storage Toolkit. http://sourceforge.net/projects/intel-iscsi/.
9) Sage A. Weil. Leveraging Intra-object Locality with EBOFS. Technical Report UCSC CMPS-290S, 2004.