Vol.2009-HPC-121 No.33
2009/8/6

アクセス性能を保証する並列ファイルシステムの提案とストレージサーバの設計

谷村勇輔[†1] 鯉江英隆[†1,†2] 工藤知宏[†1]
小島功[†1] 田中良夫[†1]

本研究では性能を指定したアクセス予約ができ，予約時間においてその性能を確実に保証する並列ファイルシステムを提案する．それは性能要求に基づいてファイルアクセス時のストライピング・パラメータを事前に決定し，利用するストレージデバイスへの予約，優先アクセスを提供する仕組みを備えたファイルシステムである．本稿では，アクセス予約とそれに関連して提案ファイルシステムが提供する機能を整理し，それを踏まえた全体のアーキテクチャを提示する．さらに，これまでの研究で開発した，ディスク領域の予約利用が可能なオブジェクトベース・ストレージ（ROBS）を用いてストレージサーバを設計し，基本 I/O におけるクライアントとストレージサーバ間の通信，I/O ストライピングの実装方法について報告する．

Proposal of Performance-Assured Parallel File System and Design of The Storage Server

Yusuke Tanimura,[†1] Hidetaka Koie,[†1,†2]
Tomohiro Kudoh,[†1] Isao Kojima[†1]
and Yoshio Tanaka[†1]

In this study, we propose a parallel file system which assures access performance specified by users when the access was reserved with performance, date and time parameters in advance. The file system automatically decides striping parameters according to the user's request, reserves the storage servers, and allows priority access to the storage servers during reserved time. This paper describes a basic concept of the access reservation of the file system and the overall architecture. In addition, the paper reports the design of the storage server using Reservable Object-based Storage (ROBS) that was developed in our previous work, the basic I/O between the client and the storage server, and implementation of parallel I/O.

1. はじめに

従来より，インターネットを介して提供されるサービスの多くはベストエフォートの性能を提供するものである．これはインターネットが提供するデータ転送性能が基本的にベストエフォートであったことに因るところが大きい．しかし，光パス網などの研究[1)] により，帯域性能が保証されたネットワークが利用できるようになりつつある．また，近年のグリッドやクラウドに見られるように，インターネット越しに何らかの IT サービスを利用するビジネスが普及しつつあり，その中でサービス品質の保証（SLA: Service Level Agreement）に基づいた課金について議論がなされるようになってきた．そのような品質が保証されたサービスを上位で行うことを可能にする IT 基盤システムの構成要素の 1 つとして，本研究ではアクセス性能を保証するストレージの実現を目指している．

アクセス性能を確実に保証するために我々がとったアプローチは，性能保証が必要なアクセスに対して予約利用を条件づける仕組みの導入である．必要性能と利用日時を指定して予約が行われるものとし，予約の有無に基づいてファイル・データが格納されたストレージサーバへのアクセスを制御する．つまり，予約済みのアクセスを優先し，ストレージサーバへのアクセスの集中を回避することで性能を保証する．単一のディスクの性能を上回る性能が必要な場合には，ストライピングを利用して性能を確保し，並列アクセスする複数のストレージサーバに対して予約を行うことになる．

現在，我々はそのような予約に基づいたアクセスによって性能が保証される機能を実装した並列ファイルシステムを提案し，その開発を進めている．ファイルシステムとして提供する理由は，ユーザがファイル，またはストレージ上に任意に確保した領域を指定してアクセス予約ができるのが望ましいと考えるためである．本稿では，まず提案ファイルシステムの利用例について述べ，それを念頭にファイルシステムの予約利用とそれに関連して提供する機能を整理し，ファイルシステム全体のアーキテクチャを提示する．そして，これまでの研究において開発した，データの書き込み領域を予約できるオブジェクトベース・ストレージ（ROBS: Reservable Object-based Storage）[2)] を用いたストレージサーバ，基本 I/O にお

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

けるストレージサーバとクライアント間の通信や I/O ストライピングの実装方法について報告する．

2. 予約に基づきアクセス性能を保証するファイルシステムの利用例

アクセス性能を保証する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．また，そのような応用のいくつかでは，保証を受けるためにアクセスに先立って予約を行う仕組みの導入は十分に許容範囲であると考える．例えば，本研究で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，提案ファイルシステムを用いて構築したストレージの適用を目指している．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．映像配信サービスを提供する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．その時，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．

配信以外では，データのバックアップをスケジュールに基づいて行うシナリオも考えられる．バックアップデータを格納するストレージへのアクセスは常にスケジュールされるため，予約を行うことができる．ストレージの予約だけを行うのではなく，予約に基づいて性能保証を行うネットワーク[3),4)] と組み合わせることで，遠隔地のデータセンタに定期的にバックアップを行うようなディザスタ・リカバリのシナリオにも適用可能である．

3. 提案ファイルシステム

本研究が提案するファイルシステムは，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．その基本的なアイデアは次のようなものである．まず，ファイルシステムは予約要求を受け取ると，そこで指定された性能が得られるようにファイル I/O のストライピングの有無，およびストライピング・パラメータを自動的に決定し，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）の予約を行う．そして，予約時間には予約を行っているクライアントからのアクセスを優先し，性能を保証するのである．本節ではこのアイデアをもとに，予約と予約に関連してファイルシステムとして提供する機能を整理し，それに基づいて設計したファイルシステム全体のアーキテクチャを述べる．

3.1 提 供 機 能

3.1.1 対象とするファイルアクセスの特徴とスループットの保証

2 節で述べた利用例におけるファイルアクセスの特徴を検討した結果，提案ファイルシステムではファイルアクセスに関して次の想定を置くこととする．

(1) ファイルサイズは数百 MB 以上と比較的大きい．
(2) 個々のアクセスではファイル全体，またはその大部分に対する逐次的なアクセスが中心である．
(3) アクセス全体で見ると Write Once，Read Many のアクセスパターンである．

(1) と (2) の想定を踏まえて，提案ファイルシステムが提供する性能保証は，予約の際に指定されたファイルの「読み出し（READ）」および「書き込み（WRITE）」におけるスループットの保証とする．そして，本稿では以降，スループットの保証を得るための予約を「性能予約」と呼ぶ．提案ファイルシステムでは予約が成立しており，かつ (1) と (2) の両方を満たすアクセスに対してスループットを保証する．

さらに，(2) と (3) の想定を踏まえて，ファイルへの書き込みについては追加書き込み（Append）のみを提供することとする．この制限を設けることで，後述するスナップショットの更新を容易にし，提案ファイルシステムの仕組みを単純化することができる．

3.1.2 データが書き込めることの保証

書き込めることの保証とは，データを書き込むための空きがストレージデバイスに存在することの保証である．この保証は，書き込みアクセスが性能予約を行っていたとしても実際にストレージデバイスに空きがなければアクセスが失敗するのを回避するために必要である．そこで，提案ファイルシステムでは書き込めることの保証を行うものとし，それを実現する機能として「Bucket」の概念を導入する．

Bucket とは，ユーザがアクセス時の性能およびデータサイズを指定して作ることができるストレージ領域である．ユーザは各 Bucket に対して任意の名前を付けることができ，Bucket を指定してその中に複数のファイルを作成することができる．ユーザ固有の領域であるため，ファイルの消去や消去した領域の再利用も自由であり，ユーザが空き領域のサイズを確認した上で書き込みを行えば，領域不足に陥ることはない．

図 1 に示すように，各 Bucket はユーザが指定した性能を満たすようにストライピング・パターンが決定されており，それに従ってストレージデバイス（図では Disk）上の領域が予約されている．提案ファイルシステムでは事前に Bucket を作成し，Bucket に対して性能予約を行うことでスループットの保証と書き込めることの保証の両方を得ることができ

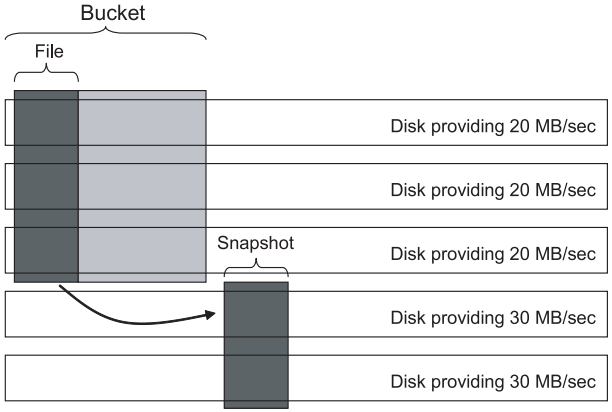

図 1 Bucket とスナップショットの概念

る．また，Bucket を作らずに提案ファイルシステムにデータを書き込むことも可能である．しかし，その場合は書き込めることの保証は得られないため，割り当てられたストレージデバイスの空きが十分でなく，途中でデータが書き込めなくなる可能性が残るベストエフォート型のサービスとなる．

3.1.3 性能保証アクセスとベストエフォートアクセスの共存

提案ファイルシステムではファイルシステムとしての汎用性を考慮し，予約機能は PVFS[5] や Lustre[6] などの並列ファイルシステムの基本機能を拡張する形で実現する．そして，予約を必要としないアクセスに対してはそのような並列ファイルシステムと同程度の機能を提供し，性能保証アクセスに影響を与えない範囲でベストエフォートアクセスを受け付けることとする．

ベストエフォートアクセスが提案ファイルシステムに受け付けられる条件は，その時間においてストレージサーバの性能に余裕がある場合である．また，アクセスが受け付けられたとしても，予約済みのアクセスがストレージデバイスを優先利用している時間帯は（ストレージデバイスが提供可能なスループット − 予約アクセスが使うスループット）分の性能に制限される．さらに，アクセス中に予約済みのアクセスによってストレージデバイスが占有される状況が発生すると，ストレージデバイスが提供可能なスループットに余裕ができるまでアクセスがブロックされる．

3.1.4 即 時 予 約

提案ファイルシステムでは事前に予約されていないが，現時点以降のできるだけ早い時間を予約開始時刻として，予約ベースのファイルアクセスを受け付ける機能を提供する．ただし，ある一定時間内に希望の予約がとれない場合はアクセスが失敗する．そのような予約をここでは即時予約と呼ぶ．即時予約の仕組みは性能予約だけでなく，Bucket の作成にも応用する．事前に Bucket を作成していなくても，アクセスの時点で必要なサイズの Bucket を作成してそこにデータを書き込む使い方が可能である．

即時予約の利点は，事前の予約がない場合にでもスループットの保証や書き込めることの保証が得られることである．前節で述べたようにベストエフォートアクセスでは，アクセスが受け付けられたにも関わらず実際の読み出しや書き込みがブロックされる可能性があるが，即時予約の機能はこの問題を解決する．

3.1.5 予約に基づいたスナップショット作成

2 節で述べた映像配信のシナリオを考えると，同時間帯に特定のファイルに対して多数の読み出し予約を行う状況が考えられる．しかし，ストレージデバイスが受け付けられる予約数には上限がある．そこで，提案ファイルシステムでは図 1 に示すように，そのようなファイルのスナップショットを別のストレージデバイスの集合に作成する機能を提供する．スナップショットの作成に際しては元のファイルと同等の性能（読み出しスループット）が得られることを担保する．つまり，内部的にはストライピング・パラメータが変わる可能性がある．図 1 の例では，元のファイルが 3 つのディスクを用いて 60[MB/sec] の性能を提供している状況において，より高速な 2 つのディスクを用いて 60[MB/sec] 以上の性能を出すようにスナップショットを作成している⋆1．

さらに，ユーザの手間を削減し，確実にスナップショットを作成できるようにスナップショットの作成自体を予約できる機能も合わせて提供する．ユーザは対象ファイルとスナップショットの作成期限を指定して，スナップショットの作成を提案ファイルシステムに要求する．要求を受け取った提案ファイルシステムは，ファイルサイズやストレージデバイスの空き状況を考慮してスナップショットの作成を適切にスケジュールし，それに伴う読み出しと書き込みの予約を行う．もし，うまくスケジュールできない場合はスナップショットの作成要求がエラーとなる．スケジュールできた場合，予約開始時刻になると提案ファイルシステムにより自動的にスナップショットの作成が開始される．

⋆1 この例はあくまで概念的なものであり，実際にはオーバーヘッドなどを加味した性能の算出が必要である．

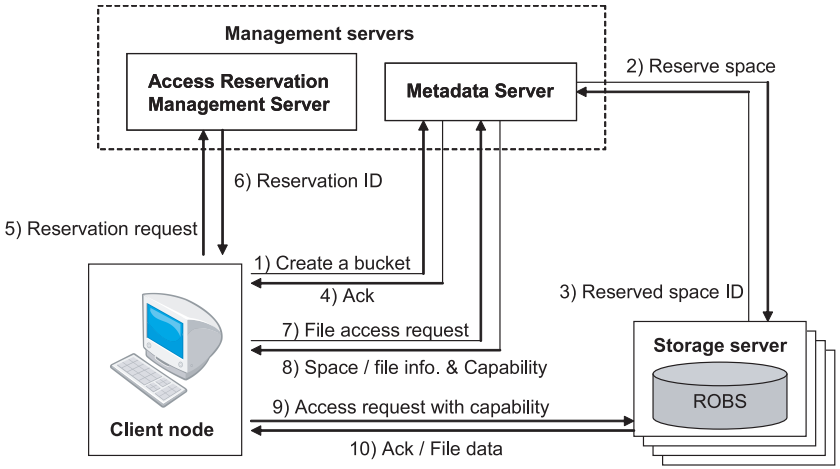

図 2 提案するファイルシステムの構成

3.1.6 ファイルの削除における制限

提案ファイルシステムでは，性能予約を受け付けているファイルは削除できないものとする．そのようなファイルを削除する場合は，全ての予約時間が終了するまで待つか，あるいは全ての予約をキャンセルしてから削除を実行する必要がある．また，ファイルの削除に関連して，提案ファイルシステムでは全てのファイルに対してライフタイムの仕組みを導入する予定であるが，本稿ではこれに関する説明を省略する．

3.2 アーキテクチャ

本研究ではこれまでに述べた機能を実現できるように，提案ファイルシステムの全体的なアーキテクチャについて検討した．以下ではシステムを構成する各コンポーネントとファイルシステムの基本的な動作の概略について述べる．

3.2.1 コンポーネント

提案ファイルシステムの設計においては他の並列ファイルシステム[5),6)] のアーキテクチャを参考にし，**図 2** のような構成をとることとした．クライアント，ストレージサーバ，そして管理サーバから構成され，管理サーバはさらにメタデータサーバとアクセス予約管理サーバを用意する．各コンポーネントの機能は次の通りである．

- ストレージサーバ（SS: Storage Server）
  ストレージサーバはディスクなどのストレージデバイスを１つ以上管理し，クライアントに対してストレージサービスを提供する．そして，予約の有無やクライアントのアクセス権限の情報をもとにストレージデバイス，またはそこに保存されたデータへのアクセスを制御する．提案ファイルシステムにおいては多くの場合，ストレージサーバは複数存在する．
- メタデータサーバ（MDS: Metadata Server）
  メタデータサーバは提案ファイルシステムにおけるファイルやディレクトリ，Bucket の名前空間を管理する．加えて，ファイルデータを格納したストレージサーバやファイル・パーミッションなどのファイルのメタ情報と，サイズや使用量などの Bucket の情報を一元管理する．
- アクセス予約管理サーバ（ARMS: Access Reservation Management Server）
  アクセス予約管理サーバは性能予約に関する情報，すなわち全てのストレージサーバに対するアクセスの予約を一元管理する．
- クライアント
  クライアントは予約またはファイルアクセスを実行するプログラムである．提案ファイルシステムでは予約に関係するコマンドとアプリケーションを開発するための I/O ライブラリを提供する．

3.2.2 Bucket の作成と Bucket に対する書き込みの動作の概略

提案ファイルシステムの機能を利用して性能保証がなされたアクセスを行う手順とファイルシステムの動作の概略を説明する．まず，データを書き込むにあたり Bucket を作成する．図 2 に示すようにクライアントはメタデータサーバに対して Bucket 名，Bucket のサイズと利用期間，Bucket に対してデータを書き込む際のスループット，そして Bucket に作成したファイルを読み出す際のスループットを指定して 1)Bucket の作成を要求する．要求を受け取ると，メタデータサーバはクライアントの要求に見合うように，すなわち指定された時刻 t_{start} から t_{end} において以下の 2 つ条件を満たすようにストレージサーバを探索する．

$$CThput \geq \sum_{i=1}^{n} SThput_i - overhead \quad (1)$$

$$CSize \geq \sum_{i=1}^{n} SSize_i \quad (2)$$

ここで $CThput$ はクライアントが要求するスループット，$SThput_i$ はストレージサーバ i が保証する最低スループットである．同様に，$CSize$ はクライアントが要求するディスク

領域のサイズ，$SSize_i$ はストレージサーバ i が提供するディスク領域のサイズである．n がストライプ数であり，$overhead$ がストライピングによるオーバーヘッドである．

メタデータサーバは利用可能なストレージサーバを選び出し，利用するストレージサーバとストライピング・パラメータ n を決定すると，それらのストレージサーバに対して 2) ディスク領域の予約（確保）を行う．ストレージサーバは予約を受け付けると 3) 予約 ID をメタデータサーバに返すので，メタデータサーバは各予約領域に対する予約 ID と Bucket 名のマッピング情報を内部的に記録し，4)Bucket の作成が完了した旨をクライアントに通知する．

作成した Bucket に対して書き込みを行う場合，クライアントはファイルの作成をメタデータサーバに要求する際に 7)Bucket 名を送信する．メタデータサーバはその Bucket に対するクライアントの書き込み権限を確認し，その権限の情報が記された Capability を作成する．そして，Bucket のために予約されている領域を保持するストレージサーバの情報と Capability を 8) クライアントに送信する．クライアントは各ストレージサーバに 9) アクセスし，受信した Capability を提示して書き込みを行うことになる．

3.2.3 性能予約とそれに基づいたファイルアクセスの動作の概略

次に，作成した Bucket に対する書き込みの性能予約を行う．図 2 に示すように，クライアントはアクセス予約管理サーバに対して，書き込みを行う日時と書き込み時に保証してもらいたいスループットを指定して 5) 予約要求を実行する．アクセス予約管理サーバは，クライアントのスループットの要求が Bucket が提供できるスループットを超えていないことを確認し，Bucket の予約領域が存在するストレージサーバに対してアクセス予約を行う．この予約情報はアクセス予約管理サーバの内部で管理される．予約が完了するとアクセス予約管理サーバはその予約に対応する ID を発行し，6) クライアントに通知する．

予約時間において，クライアントはメタデータサーバに対して 7) ファイルの作成要求を行い，利用する Bucket 名と上記の予約 ID を送信する．メタデータサーバは予約 ID の正当性と Bucket に対する書き込み権限を確認し，それらの情報が記された Capability を作成する．以降は，性能予約を行わずに Bucket に対して書き込みを行う場合と同じ手順になる．

作成したファイルに対する性能予約は，仕組みとしては Bucket に対する書き込みの性能予約とほぼ同じである．Bucket の場合は予約領域が存在するストレージサーバに対してアクセス予約を行うが，ファイルの場合はファイルデータが存在するストレージサーバに対して予約を行う．予約時間におけるアクセスでは Bucket 名の代わりにファイル名を送信する．そして，クライアントが受け取る Capability はファイル・パーミッションに基づいた情報が記されることになる．

なお，上記の Capability を用いたアクセス制御のモデルは T-10 のオブジェクトベース・ストレージに関する標準[7] に基づいたものである．メタデータサーバとストレージサーバではあらかじめ信頼できる方法で鍵の共有または鍵交換を行っており，それらの鍵を用いてメタデータサーバは作成した Capability に署名を施し，各ストレージサーバはクライアント経由で受け取った Capability の正当性を検証できるものとする．

4. ファイル I/O の設計と実装

3 節で述べた提案ファイルシステムのアーキテクチャに基づき，クライアントからストレージサーバに対して行うことになる I/O の仕組みについて検討した．そして，ストレージサーバの機能を具体化し，クライアントとストレージサーバ間のプロトコル，I/O ストライピングについて次のような実装を行うこととした．

4.1 ROBS を利用したストレージサーバの開発

提案ファイルシステムではストレージサーバが安定した性能を提供することを前提とし，ストレージサーバに対するアクセス制御を行うことで性能保証を実現しようとしている．特に，SSD（Solid State Drive）などのフラッシュベースのストレージデバイスは，書き込み性能にはまだ問題があるものの，読み出し性能においては従来の機械的な駆動部を持つ HDD が苦手とするランダムアクセスにおいても安定した性能を提供する．本研究では，そのようなストレージデバイスが提供する安定性能を損なわないように，ストレージデバイスに直接アクセスするオブジェクトベース・ストレージを開発し，それをもとにストレージサーバを開発する．

これまでの研究において我々は，ディスクス領域を予約（確保）できる機能を備えたオブジェクトベース・ストレージ（ROBS: Reservable Object Storage）[2] を開発している．ROBS が提供する主な API を**表 1** に示す．ROBS では create_reserve() により指定したサイズのディスク領域を予約することができる．予約に対しては ID が返されるので，予約領域に対してデータを書き込む際にはその ID を用いて attach() を実行して書き込み先を切り替えた後，write() を行うという使い方をする．今回開発するストレージサーバは，図 2 に示すように ROBS を内部に利用し，クライアントやメタデータサーバとのインタフェースを備える構成である．ストレージサーバはメタデータサーバやクライアントからのコマンドを受信すると表 1 の API を呼び出して，管理下にあるストレージデバイスの領域を確保したり，データをオブジェクトとしてストレージデバイスに書き込んだり，読み出したりする．

表 1 ROBS の主な API

create_reserve()	Reserve new space of given size.
modify_reserve()	Change the existing reservation.
destroy_reserve()	Cancel the existing reservation.
read()	Read the object data with offset and length.
attach()	Set a write-pointer to the reserved space.
write()	Write data to the object (append only).
remove()	Remove the object.
sweep()	Collect spaces of expired objects.
stat()	Get status of the object.
get_attr()	Get attributes of the object.
set_attr()	Set attributes of the object.

4.2 基本 I/O におけるクライアントとストレージサーバ間の通信

表 2 にストレージサーバに対する予約とファイルアクセスに関する主なコマンドを示す．また，**図 3** にはファイルアクセスのコマンドシーケンスと ROBS の API 呼び出しへのマッピングを示す．ファイルアクセスにおいては，まず最初にクライアントはメタデータサーバに 1a)OPEN を要求し，書き込み領域の情報またはファイルデータ（オブジェクト）に関する情報と Capability を得る．次に，クライアントはストレージサーバに接続し，1b)OPEN を要求する．それを受信したストレージサーバはクライアントが提示した Capability の正当性を検証する．書き込みアクセスでは，Capability に ROBS のディスク領域を予約した時の ID が含まれているので，それを用いて ROBS の attach() を実行する．OPEN に成功するとクライアントは 2)WRITE 要求を行う．この WRITE 要求はブロッキング操作であり，ROBS の write() が完了してから ACK をクライアントに返す．これは ROBS の write() がブロッキングかつアトミックな操作になっているためである．

読み出しアクセスでは 1b)OPEN 要求において attach() は不要である．OPEN が成功した後，クライアントは 3)READ 要求を実行する．3)READ 要求ではその都度，読み出し開始位置（オフセット）と読み出すデータ長を指定することができ，ストレージサーバ内部ではそれを単純に ROBS の read() にマップして実行する．上述の 2)WRITE は Append のみに制限されているが，3)READ は任意の位置からオブジェクトのデータを読むことが可能である．

1)〜3) の一連のアクセスにおいて，クライアントとストレージサーバは TCP 接続を維持し，クライアントが 4a)CLOSE 要求を行うことによりそれが切断される．この時，クライアントはメタデータサーバにも 4b)CLOSE 要求を行う．要求を受信したメタデータサーバは，CLOSE されたアクセスが書き込みであった時，ストレージサーバに対して 5)VERIFY_SIZE を発行して最終的にストレージサーバに書き込まれたオブジェクトのサイズを確認する．

表 2 ストレージサーバに対する主なコマンド

Client → Storage Server	MDS → Storage Server
OPEN	CREATE_RESERVE
CLOSE	MODIFY_RESERVE
READ	DESTROY_RESERVE
WRITE	VERIFY_SIZE
REMOVE	CREATE_SNAPSHOT
GET_ATTR	
SET_ATTR	

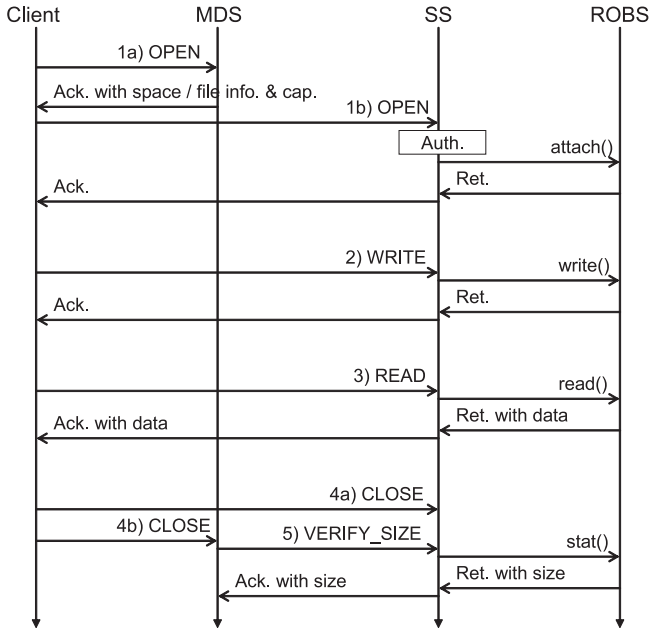

図 3 ファイルアクセスにおけるコマンドシーケンス

4.3 ストライピング

提案ファイルシステムでは単体のディスク性能を上回る性能要求に対してストライピング，すなわち複数のストレージサーバに並列にアクセスして性能を得る手法を用いて対応する．ストレージサーバはオブジェクトベース・ストレージとして設計されているため，ストライピングにより複数のストレージサーバに分散されるファイルデータは，サーバ毎に 1 つ以上のオブジェクトとして格納されることになる．メタデータサーバはそれらのオブジェクトのリストに対して ID を振り，これを Global Object ID（GOID）として管理し，内部においてファイルパスと GOID のマッピング情報を持つ．

GOID は，クライアントからのファイルの新規作成要求（Create）の際に発行され，クライアントを介してストレージサーバに伝えられる．ストレージサーバでは GOID を ROBS の Collection ID（CID）に指定して Collection を作成し，その Collection に関連づけられたオブジェクトを作成する．GOID をそれぞれのオブジェクトに関連づけておくことで，万が一，メタデータサーバが故障してもストレージサーバからファイルを復旧できる．また，Collection の利用は，クライアントやストレージサーバにおけるオブジェク管理を単純化する利点もある．

提案システムでは現在 RAID-0 相当のストライピングの開発を進めている．RAID-5 相当のパリティを用いたストライピングについてはまだ検討中であり，耐故障性の仕組みと合わせて検討を行っていく予定である．

4.4 メタデータサーバとストレージサーバ間の通信

表 2 の右欄に示す予約コマンド（末尾が_RESERVE のコマンド）は ROBS が提供する書き込み領域の予約に関するものであり，提案ファイルシステムにおいては Bucket が作成される際に呼び出される．なお，性能予約はアクセス予約管理サーバで予約情報を持ち，Capability を用いたアクセス制御により実現されるため，表 2 の予約コマンドとは独立である．

その他にメタデータサーバは CREATE_SNAPSHOT の要求を行う．これはスナップショットの作成命令であり，要求と同時にスナップショットの元ファイルを構成するオブジェクトの読み出し権限が含まれた Capability がストレージサーバに送られる．CREATE_SNAPSHOT を受信すると，ストレージサーバは別のストレージサーバにあるオブジェクトを読み出し，自身のストレージデバイスにその複製を作成する．さらに表 2 に挙げたコマンド以外にも，メタデータサーバとストレージサーバ間では起動・終了のシーケンスやストレージデバイスの状態報告などに関わるコマンドが考えられるが，それらの詳細は検討中である．

5. おわりに

本稿では性能を指定したアクセス予約を受け付けて，予約時に指定した性能を確実に保証する並列ファイルシステムを提案した．提案ファイルシステムが保証する性能は，比較的大きなサイズのファイルに対して一定時間以上の連続アクセスを行った時のスループットである．さらに，性能予約と合わせてデータを書き込めることを保証するための Bucket の作成機能，性能保証アクセスとベストエフォートアクセスの共存方法，予約に基づいたスナップショットの作成機能などについて述べ，提案ファイルシステムの基本的なコンセプトとそれを実現するシステム全体のアーキテクチャを提示した．そして，そのアーキテクチャに基づいたファイル I/O の実装方法について報告した．

今後はファイル I/O の開発を進めてその性能や性能の安定性を検証するほか，メタデータサーバやアクセス予約管理サーバの開発，ストレージデバイスの状態を把握する仕組みの検討，耐故障機能の実現方法の検討などに取り組んでいく予定である．

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) Atsuko Takefusa, Michiaki Hayashi, Naohide Nagatsu, Hidemoto Nakada, Tomohiro Kudoh, Takahiro Miyamoto, Tomohiro Otani, Hideaki Tanaka, Masatoshi Suzuki, Yasunori Sameshima, Wataru Imajuku, Masahiko Jinno, Yoshihiro Takigawa, Shuichi Okamoto, Yoshio Tanaka, and Satoshi Sekiguchi. G-lambda : Coordination of a Grid scheduler and lambda path service over GMPLS. *Future Generation Computing Systems*, Vol.22, pp. 868–875, 2006.
2) 谷村勇輔, 鯉江英隆, 工藤知宏, 小島功, 田中良夫. 予約利用可能なオブジェクトベース・ストレージの設計. 情報処理学会研究報告 2009–HPC–119, pp. 79–84, 2009.
3) G-lambda Porject. http://www.glambda.net/.
4) 竹房あつ子, 中田秀基, 工藤知宏, 田中良夫, 関口智嗣. 多様な資源を事前予約で同時確保するためのグリッドコアロケーションシステムフレームワーク GridARS. 情報処理学会論文誌コンピューティングシステム（ACS20）, pp. 32–42, 2007.
5) PVFS. http://www.pvfs.org/.
6) Lustre. http://www.lustre.org/.
7) T10. SCSI Object-Based Storage Device Commands-2 (OSD-2) Working Draft. http://www.t10.org/.
8) Michael Factor, David Nagle, Dalit Naor, Erik Riedel, and Julian Satran. The

Vol.2009-HPC-121 No.33
2009/8/6

OSD Security Protocol. In *Proceedings of the Third IEEE International Security in Storage Workshop*, pp. 29–39, 2005.

9) 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.

10) Sage A. Weil. Leveraging Intra-object Locality with EBOFS. Technical Report UCSC CMPS-290S, 2004.