オンデマンド及びバックグラウンド複製による ファイルサーバ移行時の停止時間削減

松沢 敬一[1,2]　早坂 光雄[1]　品川 高廣[2,3]

概要：ファイルサーバの移行は，格納ファイルの複製を伴うため長時間を要する．この間ファイルアクセスが停止すると，利用者の利便性を著しく損なう．本研究では，異機種間でのファイルサーバの移行時に，ファイルアクセスの停止時間を短縮するための移行方式を提案する．提案方式では，まず接続先サーバを移行先サーバに切り替える．その後アクセス要求時に応答に必要なファイルのみ移行元からオンデマンドで複製する方式と，低負荷時に未アクセスファイルをバックグランドで複製する方式を併用する．これにより，ファイルアクセスを停止すること無く時間のかかる複製をおこなえるようにする．性能評価の結果，移行に伴うアクセス停止時間を，一般的なファイルサーバのアクセス許容応答時間である 30 秒以内に短縮できることを確認した．

Reducing Downtime on File Server Migration using On-Demand and Background Replication

Keiichi Matsuzawa[1,2]　Mitsuo Hayasaka[1]　Takahiro Shinagawa[2,3]

1. はじめに

IT 機器は時間経過とともに年間故障率が上昇することが知られており [1]，ストレージ機器を含む IT 機器の寿命は一般に 3〜5 年程度とされる [2]．そのため，長期運用される IT システムにおいてストレージ機器の移行は定期的に発生する．ファイルサーバにおいては，機器の刷新が新しい機器の導入理由の上位に挙がっており [3]，さらに機器選定の上で移行のサポートを重要視するストレージ管理者もいるなど [4]，移行は利用者の関心事の一つとなっている．

ファイルサーバの移行には現行サーバが保持する既存のデータを新規サーバへ異機種間で複製する必要がある．ファイルサーバには平均で 400TB 超のデータが格納されており [3]，このような大容量データの複製には数日を要する．この間クライアントのファイルアクセスを停止することは利用者の利便性を著しく損なう．一般に，クライアントにおけるファイルサーバに対するアクセス許容応答時間は 30〜60 秒程度であるとされている [5], [6]．よって，移行に際して利用者の利便性を損ねないためには，クライアントによるファイルアクセスの停止時間を 30 秒以下に押さえ，かつ移行完了後に移行に伴う性能オーバヘッドを残さない複製手法が求められる．

ファイルサーバ間のデータ複製には，2 章で述べるようにいくつかの手法が提案されている．しかし異機種間のファイルサーバ移行に適用するには，いずれの方式も課題が残る．例えば，同一機種でなければ移行できない，継続的にデータ更新が行われるワークロードにおいてファイルアクセス停止時間を十分に短くできない，移行完了後も性能オーバヘッドが残る，などである．

本報告は，先行してクライアントの接続先を移行先サーバに切り替え，その後データ複製を行う Post-Copy 方式により，移行時の停止時間を削減できるファイルサーバ移行方式を提案する．本方式では，アクセス対象ファイルが

[1] (株) 日立製作所 研究開発グループ
Hitachi, Ltd. R&D Group.
[2] 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[3] 東京大学 情報基盤センター
Information Technology Center, The University of Tokyo

表 1: データ複製手法分類

複製方法	複製単位	
	ファイル単位	ボリューム単位
Pre-Copy 方式	rsync [7] Robocopy [8]	SnapVault [9] Storage vMotion [10]
Post-Copy 方式	提案手法	ImageStreaming [11]
GNS	X-NAS [12] NAS Switch [13]	N/A
アーカイブデータ復元	DAB [14] Cumulus [15]	N/A

移行先サーバに存在しない場合にオンデマンドで対象ファイルを移行元サーバから複製する On-Demand Replication (ODR) と，未アクセスファイルを移行先サーバの低稼働時に複製する Background Replication (BGR) を併用して，データを複製する．本方式により，クライアントのアクセス停止時間を接続先切り替えの短時間のみに抑え，以後クライアントからのアクセス要求への応答と並行してデータ複製を行う．また，複製に必要な機能は移行先のサーバに備え，移行元のサーバを変更せずに済むようにして，異機種間での移行を可能にする．さらに，完了後にはファイルアクセス時にオーバヘッドを生じない移行を実現する．

本研究では，Linux 及び Samba を用いて，NFS と SMB プロトコルを対象とした複製方式の実装を行い，性能評価実験を行った．その結果，移行に伴うクライアントのファイルアクセス停止時間を，一般的なファイルサーバに対するアクセス許容応答時間である目標の 30 秒以内に短縮できることと，移行完了後のアクセスにオーバヘッドを生じさせないことを確認した．

2. 関連研究

本章では，ファイルサーバ移行の関連研究について述べる．表 1 はファイルサーバ移行のためのデータ複製手法を分類したものである．

2.1 ボリューム単位複製

本方式はサーバ間でボリューム単位の複製を行う方式である．SnapVault [9] では，移行元サーバにおけるデータ更新をボリューム単位で移行先に複製することで両サーバのボリュームの格納データを同期させ，サーバ移行を実現する．Storage vMotion [10] や ImageStreaming [11] は仮想マシン移行の手法であり，プロセッサやメモリに加えボリュームの複製も行う．

これらのボリューム単位の複製は，移行前後において同一のボリュームを利用できる場合に有効である．しかし異機種間の移行においては，ボリューム上のファイルシステムが同一とは限らないため適用できない．

2.2 Pre-Copy 方式

Pre-Copy 方式は，移行元サーバから移行先サーバへデータの複製を完了してからクライアントの接続先を変更する方式である．この方式はファイルストレージ製品の移行手順として広く利用される．rsync [7] や Robocopy [8] は，ファイル単位の Pre-Copy によりディレクトリツリーを複製するツールである．

Pre-Copy 方式において，全データを移行先にもれなく複製するためには，複製中の移行元サーバにおいて，クライアントからデータが更新される場合の対処が必要となる．そこで，Pre-Copy 方式では一般的に，全てのデータが移行元・移行先で一致する状態になるまで，複製中に更新された領域の再複製を行う．しかし，絶えず更新が行われるワークロードにおいては，再複製の繰り返しにより複製が長時間化する．また，複製中の更新の有無を確認する間，クライアントによるファイルの更新を禁止する必要が生じる．多数のファイルを含むサーバにおいて，この確認にはファイルアクセスの目標停止時間である 30 秒を大きく超える時間がかかる．

2.3 Post-Copy 方式

一般に大容量データを備えるサービスを移行する際，全データの複製完了を待たずに移行先のサービスを開始する方法として，Post-Copy 方式の複製が広く使われている．Post-Copy 方式では，移行開始以降のデータへのアクセスは移行先に対して行う．移行先がアクセス対象のデータを保持しない場合，オンデマンドで移行元からデータを複製しアクセス要求に応答する．例えばメモリを対象とした Post-Copy 方式の複製は Unix 系 OS におけるプロセスのメモリ Copy on Write [16] や仮想マシンのライブマイグレーション [17] で用いられている．ボリュームへの適用例としては，前述の ImageStreaming [11] が相当する．

Post-Copy 方式は，サービス停止時間を短く抑え，かつ更新頻度の高いデータ領域において Pre-Copy 方式で生じる複数回のデータ複製を抑えられる利点がある．その反面，複製未完了のデータへのアクセスに対し，複製を完了するまで応答できないため応答時間がその分伸びる．つまり，Post-Copy 方式では短時間の複製待ちを何度も発生させる一方，長時間の複製待ちを抑止できるという利点がある．

2.4 Global Name Space を用いた複製方式

Global Name Space(GNS) は，複数のサーバのディレクトリツリーを束ねる仮想化層を用いて，単一のディレクトリツリーを構築する方式である．この仮想化層は，個々のファイルの格納先サーバを管理しており，仮想化層で受信

したクライアントのアクセス要求を格納先サーバに転送する機能を備える．クライアントはこの仮想化層を介することで，ファイルが格納されたサーバを認識することなくアクセスできる．

本方式は，この GNS と同様の仮想化層とデータ複製を組み合わせることで移行を実現する．仮想化層によりバックグラウンドでサーバ間のファイル複製を行いつつ，クライアントからのアクセスを複製の状態に応じ移行先・移行元いずれか適切なサーバに転送することで，クライアントはファイルの複製状況や所在を認識することなくアクセスを継続できる．本方式の先行研究には X-NAS[12] や NAS Switch[13] が挙げられる．

本方式の問題点には，移行元サーバが元々 GNS 構成下で運用されている必要があることや，ファイルアクセスに仮想化層を挟むために継続的に性能オーバヘッドが生じることがある．

2.5 アーカイブデータ復元方式

DAB[14] や Cumulus[15] はファイルサーバが別の機器にアーカイブしたデータを，別のファイルサーバで読み込むことでサーバ間のデータ複製を行う．これらの方式は，移行元と移行先がアーカイブデータの形式を共有している必要があるため，異機種サーバ間の移行には適用できない．

2.6 従来方式の課題

従来の複製方式で解消できていない，ファイルサーバ移行で解消すべき課題は下記の 3 つにまとめられる．

(1) 異なるファイルシステム間でも，移行元サーバには手を加えず適用できる．
(2) クライアントのアクセス停止時間を 30 秒以下にする．
(3) 移行完了後に，複製のためのオーバヘッドを残さない．

3. 提案移行方式

提案方式では，Post-Copy 方式をファイル単位のデータ複製に適用し，サーバ移行を行う．本方式では，複製をファイル単位で行うことで移行元サーバのファイルシステムに非依存な複製を行い，また Post-Copy 方式を用いることでクライアントのアクセス停止時間を抑える．

ファイル単位でディレクトリツリーを複製するには，個々のファイルのデータ・メタデータに加えディレクトリにより構成される階層構造を複製する必要がある．両者を Post-Copy 方式で複製するため，本方式ではスタブ管理機能を用いてファイルとディレクトリの複製状況を管理する．

3.1 データ複製方式

提案方式は，On-Demand Replication (ODR) と，Background Replication (BGR) を併用し，ファイル共有対象のディレクトリツリーを複製する．

3.1.1 ODR

ODR は，Post-Copy 方式をファイルの複製に応用した方式で，ディレクトリツリー全体の複製が未完了の状態で移行先サーバにおいてアクセス要求を受け付け，アクセス要求があった時点で応答に必要なデータだけを移行元サーバから複製する方式である（以後，この移行元サーバから移行先サーバへのデータ複製をリコールと呼ぶ）．これにより，クライアントからのアクセス停止時間は，接続先を移行元サーバから移行先サーバへ切り替えるわずかな時間に留めることができる．

本複製方式では，移行先サーバにおいて階層ストレージ管理で用いられるスタブ管理の考え方を用いて ODR を実現する．スタブ管理では，スタブと呼ぶ特殊な状態のファイル（以下ディレクトリも同様に対象とする）を扱う．スタブ状態のファイルは，自身のディスクにデータの全てまたは一部を保持せず，代わりに移行元サーバにあるファイルを参照するための位置情報を保持する．ODR では，スタブ状態のファイルへのアクセス要求をフックし，その契機でリコールを行う．ファイルサーバにおいて ODR を実現するためのスタブ管理の詳細は 3.2 節で述べる．

3.1.2 BGR

ODR はアクセス要求時点で初めて対象ファイルを複製するため，アクセスされないファイルは複製されずディレクトリツリー内の全ファイルの複製が完了しないという問題点がある．この問題点を解消するため，バックグラウンドで未アクセスファイルの複製を行う BGR を併用する．

BGR は ODR を用いて実現する．移行先サーバ内で，ディレクトリツリーを探索し，全ファイルのデータを読み込む巡回プログラムを動作させる．この巡回プログラムにより，全ファイルが順次ファイルがリコールされる．巡回プログラムはファイル共有サービスの負荷が低い時間帯に実行することで，クライアントのアクセス性能への影響を抑えて複製できる．

3.1.3 BGR の進行状況管理

移行元サーバの格納データ量が多く，全データを複製し終えるまでに時間がかかる場合，BGR の間欠実行を要するケースがある．例えば計算機システムの稼働率の高い昼間を避け，夜間のみ数日かけて BGR を行う場合が相当する．この場合，日中 BGR の非実行中にクライアントによりファイルが更新されても，最終的に全データをもれなく複製した状態にできなければならない．そのため，巡回プログラムを途中停止・再開しながら実行できるよう進行状況管理が必要となる．

本方式では，ディレクトリツリー内にある複製未完了のファイル数カウントを保存し，ディレクトリツリー全体の複製完了・未完了の判断に用いる．巡回プログラムがディレクトリツリー全体を探索し終えた時，このカウントが 0 であれば複製が完了したとみなす．そうではない場合，複

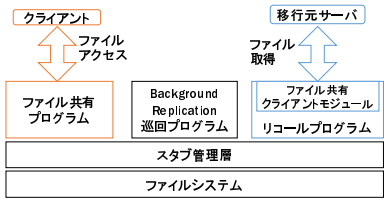

図 1: ソフトウェア構成図

製未完了と判断し再探索を行う．巡回プログラムが探索を行ったにもかかわらずカウントが 0 にならないケースとして，クライアントによるファイル・ディレクトリの移動により，階層の一部を巡回プログラムが探索せず見逃される場合がある．

巡回プログラムは停止時には探索済みディレクトリの情報は保存せず，停止後再開すると再度ルートディレクトリから全ディレクトリを探索する．この方式では，再探索を繰り返すため CPU やディスクの処理オーバヘッドを生じるが，巡回プログラム動作時以外にはオーバヘッドを生じないという利点がある．再探索を不要とする方法として，常時クライアントによるファイル・ディレクトリの移動を監視し，その移動に追従して探索を行うことも考えられるが，移動の監視は通常運用中のオーバヘッド増につながるため採用しない．

3.2 ファイルサーバ向けスタブ管理

本章では，3.1 節で述べた複製方式を可能とする，ファイルサーバ向けのスタブ管理について説明する．

3.2.1 ソフトウェア構成

図 1 に移行先サーバのソフトウェア構成を示す．

ファイル共有プログラムはファイルシステム上のファイル群を，ファイル共有プロトコルを介しクライアントと共有するプログラムで，ファイルサーバが一般的に備えるプログラムである．

リコールプログラムはスタブ管理層からの要求に応え，移行元サーバからデータ及びメタデータを取得し移行先サーバへ複製するプログラムである．本プログラムは移行元サーバへのアクセスプロトコルに対応したクライアントモジュールを持つ．

巡回プログラムは 3.1.2 節で述べた BGR 用の巡回プログラムである．

スタブ管理層はファイルの状態管理を行い，その状態に応じてファイル・ディレクトリへのアクセスをフックする機能を備える．

ファイルシステムは一般的に OS が備えるファイルシステムである．

3.2.2 ファイル・ディレクトリ状態管理

ODR を行うにあたってクライアントアクセスへの応答時間を短く抑えるため，アクセス要求時に移行元サーバから

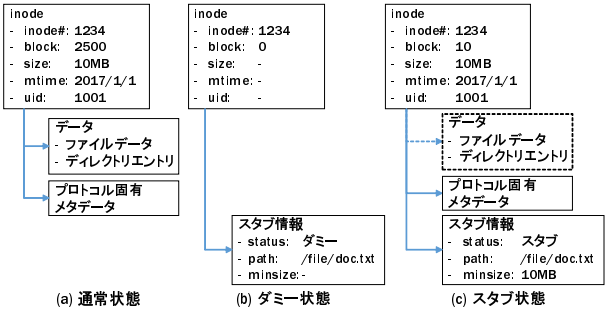

図 2: ファイル・ディレクトリの内部データ構造

図 3: 複製ファイル・ディレクトリの状態遷移

ら複製するデータ・メタデータを必要最小限に留めなければならない．そこで本スタブ管理層では，データ及びメタデータを部分的に複製した状態を保持できるようにする．そのため，ディスク内に格納するファイル及びディレクトリを，複製の状況に応じて初期状態，ダミー状態，スタブ状態，通常状態の 4 つの状態で管理する．

図 2 に各状態におけるディスク内に保持するデータ構造を示す．一般的なファイルシステムでは，図 2 **(a) 通常状態**に示すように，ファイルの一般的なメタデータを格納する inode と，そこから参照されるデータ本体（ファイルデータ及びディレクトリエントリ）から構成される．また，プロトコル固有のメタデータとして，ファイル共有プログラムがアクセス権限などを格納する．

本スタブ管理方式ではそれに加えて，図 2 **(b) ダミー状態**及び **(c) スタブ状態**に示すように，ファイル毎にスタブ情報を保持する．スタブ情報には，ファイルに対応する移行元サーバ内のファイルのフルパス名（path）と状態フラグ（status），及び複製後最小ファイルサイズ（minsize）を保持する．複製後最小ファイルサイズについては，4.5.2 節で説明する．

図 3 に移動先サーバにおけるファイル・ディレクトリの状態遷移を示す．状態は，クライアントからのアクセスに応じて順に遷移する．移行先サーバにおけるファイル (及びディレクトリ) の各状態は下記の通りである．

初期状態　移行先サーバは移行元サーバの格納ファイルに関し何ら情報を持たない状態である．親ディレクトリが参照されると，次のダミー状態のファイルが生成される．

ダミー状態　移行先サーバが，移行元サーバにおけるファイルのパスだけを認識した状態である．移行先サーバには，最小限のメタデータとして，inode 番号とスタブ情報が格納される．この状態のファイルのメタデータが参照されると，次のスタブ状態に遷移する．

スタブ状態　移行先サーバが，ファイルのメタデータ全て及びデータの一部を保持した状態である．移行先サーバには，inode の全メタデータとプロトコル固有メタデータ，スタブ情報，そしてデータの一部を保持する．ファイルのデータはブロックの有無を複製状態と対応付けて管理しており，ブロックがホールであれば，その領域は複製未完了とみなす．全てのデータブロックがホールではなくなると，そのファイルは次の通常状態に遷移する．

通常状態　移行元サーバからファイルを複製した状態である．元々移行元に存在せず，移行先で新規作成されたファイル・ディレクトリも通常状態とみなす．

移行先サーバ内の全ファイル・ディレクトリが通常状態になると，以後移行元サーバへのアクセスは生じないため，ディレクトリツリー全体の複製が完了したとみなせる．

本スタブ管理層は，ファイルシステム全体で 3.1.3 節で述べた複製未完了のファイル数カウントを管理する．上記の 4 状態のうち，ダミー状態とスタブ状態のファイル・ディレクトリがカウント対象となる．

3.2.3　親ディレクトリエントリアクセス時処理

ダミー状態の親ディレクトリエントリが初めて参照されると，親ディレクトリエントリが通常状態に遷移する (後述) と同時に，親ディレクトリ内の全ファイル (及び子ディレクトリ) がダミー状態で生成される．

ダミー状態ファイルの生成は，スタブ管理層の依頼によりリコールプログラムが行う．リコールプログラムは，親ディレクトリエントリを参照し，そこに格納されたファイル名毎に，ダミー状態のファイルを生成する．同時に，複製未完了のファイル数カウントを更新する．

ダミー状態ファイルのスタブ情報内のパス名は，親ディレクトリの移行先サーバにおけるパスではなく，スタブ情報内のパスからの相対位置で設定する．これはこの親ディレクトリが，移行中にクライアントのアクセスにより移動しても，移行元と移行先サーバのファイルの対応関係を保持するためである．例えば親ディレクトリのスタブ情報内のパスが/A/B/C であり，ディレクトリエントリに D というファイルがあるならば，生成するダミー状態ファイルのスタブ情報には/A/B/C/D というパスを格納する．

ダミー状態ファイルは，親ディレクトリエントリ中の情報だけで生成でき，ディレクトリ内の個々のファイルのメタデータ取得を要しない．そのため移行元サーバとの通信が不要であり，ディレクトリ内のファイル数が多い場合も高速に完了する．

3.2.4　メタデータアクセス時処理

ダミー状態のファイル (及びディレクトリ) のメタデータがアクセスされると，スタブ管理層の依頼により，リコールプログラムは対象ファイルをスタブ状態に遷移させる．

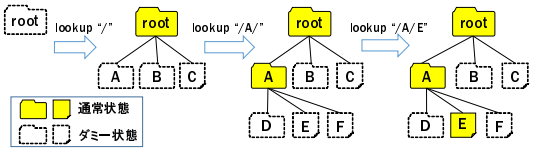

図 4: ファイルアクセスに伴うパス構成要素順次複製

リコールプログラムはスタブ情報内のパス名を用いて移行元サーバから各種メタデータを複製する．

3.2.5　ファイルデータアクセス時処理

スタブ状態のファイルデータ (及びディレクトリエントリ) がアクセスされると，スタブ管理層は，アクセス位置のブロックを確認し，ブロックがホールである場合，リコールプログラムにデータの複製を依頼する．リコールプログラムは移行元サーバからファイルデータを読み取り，ファイルシステムに書きこむことでブロックを埋められた状態にする．

データ複製の結果，ファイル内にホールであるブロックがなくなると，リコールプログラムはこのファイルを複製完了したとみなし，ファイルを通常状態に遷移させ，複製未完了のファイル数カウントを更新する．

3.2.6　ファイルアクセスにおけるパス構成要素の順次複製

ここまで述べたファイル・ディレクトリの状態遷移を用いると，移行先サーバのファイル共有にダミー状態のルートディレクトリだけ作成しておくことで，以後移行先サーバはパス構成要素を順次複製するため，任意のファイルのアクセス要求に応答可能となる．

図 4 に，移行先サーバにダミー状態のルートディレクトリだけある状況において，ファイル”/A/E” にアクセスする例を示す．アクセス先ファイルのパス名解決のため，まずルートディレクトリのディレクトリエントリを参照することで，ルートディレクトリ下にダミー状態のファイル・ディレクトリ A,B,C が生成される．続けてディレクトリ A のディレクトリエントリを参照することで同様にダミー状態のファイル・ディレクトリ D,E,F が生成される．最終的にパス名の示すファイル E をアクセスすると，E がスタブ状態を経て通常状態となりアクセスに応答できる．

3.3　移行手順

本節では，ここまで述べたデータ複製手法とスタブ管理機能を用いて，ファイルサーバを移行する手順について説明する．移行手順におけるクライアント・移行元サーバ・移行先サーバの接続関係を図 5 に示す．

移行先サーバの設置　移行開始前は，図 5**(a) 移行実施前**が示す通りクライアントは移行元サーバに接続している．その状態で，ファイルサーバの管理者は移行先サーバの設置を行い，移行元サーバと対応してファイルシステム及びファイル共有を構築する．続けて，移

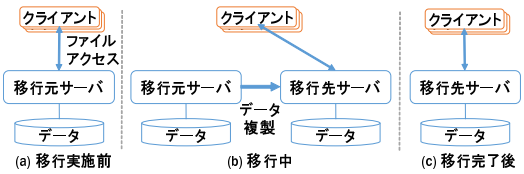

図 5: ファイルサーバの移行手順

行元サーバに合わせ，ネームサーバなど運用に必要な外部サーバの設定を行う．次に，移行先サーバが移行元サーバの全ファイルを参照するため，移行元サーバにファイルアクセス用アカウントを作成する．最後に，移行先サーバのファイル共有にダミー状態のルートディレクトリを作成する．ここまでの手順により，ODR による複製が可能となる．

クライアントの接続先変更　各クライアントの接続先を，移行元サーバから移行先サーバに変更する．この接続先変更作業中，クライアントはファイルサーバ上のファイルを参照できないため，ファイルのアクセス停止時間となる．

以後図 5**(b) 移行中**が示す通り，クライアントは移行先サーバに接続してファイルにアクセスする．移行先サーバにクライアントが要求したファイルが未複製の場合，ODR によりファイルは移行元サーバから複製される．

Background Replication の実行　BGR の巡回プログラムを実行する．BGR がクライアントのファイルアクセス性能に及ぼす影響を押さえるため，巡回プログラムの実行はクライアントによるファイルアクセスが少ない時間帯のみ実施したり，巡回プログラムが占める I/O 帯域に上限を設定したりする．

移行元サーバの停止　移行先サーバのファイル共有において複製未完了ファイル数カウントが 0 となると，データの複製が完了したとみなす．以後移行元サーバを参照することはないため，移行元サーバを停止して撤去し，図 5**(c) 移行完了後**の状態とする．

3.4　プロトコル固有機能への対応

ファイル共有プロトコルは，固有の機能やメタデータを備える場合がある．例えば NFS のハードリンクや，SMB のアクセス権限管理が相当する．これらの固有機能は，リコールプログラムにおける移行元サーバからのデータ・メタデータ取得処理をプロトコル毎に作成し，対応する．具体例については 4.5 節で説明する．

3.5　提案方式の制限

本方式はファイル共有プロトコルを介し移行元から移行先サーバにファイルを複製するため，ファイル共有プロトコルで参照できない機能やデータに関しては複製できない．例えば，移行元サーバがスパースファイルや圧縮のようなデータ容量を削減する機能を備えていても，移行先サーバではその削減機能を再現できず，inode に格納されたファイルサイズに応じたディスク容量を必要とする．

4.　実装

本研究では，Linux を用いたファイルサーバを移行先サーバとし，提案方式を実装した．本章ではこの移行先サーバにおける各構成要素について説明する．

4.1　ファイル共有プログラム

ファイル共有プログラムには，Linux で一般的に用いられるものを利用した．NFS に対しては Linux カーネルの nfsd，SMB に対しては Samba 4.1.0 を用いた．

4.2　リコールプログラム

リコールプログラムは，ユーザー空間で動作するアプリケーションとして C 言語で実装した．リコールプログラムとスタブ管理層は，Unix Domain Socket を用いて双方向のリクエストキューペアを共有し，互いにリコールやスタブ情報の変更などの処理要求を送受信する．リコールプログラムは複数スレッドで動作し，受信した処理要求パケットを並列に処理していく．データの転送効率を向上するため，リコールプログラムはクライアントのアクセス先領域に対し，10MB 単位でリコール対象の領域を拡大し，複製する．ファイル共有クライアントモジュールとしては，NFS に対しては Linux カーネルの NFS クライアント，SMB に対しては Samba に付属する libsmbclient を用いた．

4.3　Background Replication 巡回プログラム

BGR のための巡回プログラムは，ユーザー空間で動作するアプリケーションとして C 言語で実装した．このアプリケーションはファイル共有毎に 1 プロセス動作する．

4.4　スタブ管理層及びファイルシステム

本実装においては，スタブ管理層は Linux カーネルの VFS 層に組み込んだ．これはファイルアクセスを低オーバヘッドでフックするためである．スタブ情報は，ファイルの拡張属性に格納した．

4.5　プロトコル固有処理

本節ではプロトコル固有で対応を要する処理について述べる．

4.5.1　アクセス権管理

プロトコルやその前提となるファイルシステムによって，アクセス権管理の方法や Access Control List(ACL) の形式は異なる [18]．本実装では，対応プロトコルのうち最も細粒度でアクセス権を管理する SMB プロトコルで ACL

を取得し，独自に設計したマッピングルールにより他プロトコルにおけるアクセス権判定を行う．

4.5.2 truncate 要求対応

本スタブ管理では，ファイルブロックのホール状態を複製要否の判定に利用している．そのため，クライアントがスタブ状態のファイルに NFS の truncate 要求を行いファイルを拡大した場合，移行先サーバで明示的にホール状態のブロックが作成され，以後のアクセスにおいてそのブロックは複製未完了領域と誤判定されてしまう．本実装では，対策として inode 内に保持されるファイルサイズとは別に，ファイルが初めてスタブ状態になった以後のファイルサイズの最小値をスタブ情報に格納する．この最小値以上のホール状態のブロックは，複製未完了領域ではなく移行先サーバで作成されたホールと見なし，誤判定を回避する．

4.5.3 ハードリンク対応

本方式では移行元と移行先サーバのファイルの対応関係をスタブ情報に格納したパス名で管理するため，互いにハードリンクされた複数のファイルはそれぞれ別のファイルとして複製されてしまう．そこで本方式では，これらのファイルを複製後も同一の inode にハードリンクされた状態を復元させる．

リコールプログラムはスタブ状態のファイルを生成する過程でリンク数が 2 以上のファイルを検出した場合，ハードリンクされた異なるパス名のファイルが後に複製されるケースに備え，専用の隠しディレクトリに，複製対象の inode 番号を名前とするファイルを作成する．既に同名のファイルが存在する場合，別のパスにある同 inode にハードリンクされたファイルが存在するため，新規にスタブファイルを作成する代わりに，対象ファイルをハードリンクで作成することで複数ファイルが互いにハードリンクされた状態を再現する．

5. 性能評価

本章では，提案手法を用いて NFS と SMB を対象としてファイルサーバの複製及び移行を行い，その際生じる停止時間と性能オーバヘッドを評価する．

5.1 実験環境

実験に用いた機器を表 2 に示す．移行元サーバ，移行先サーバ，クライアントはいずれも 1 台で，互いにネットワーク接続されている．I/O の帯域不足により早期に性能が頭打ちになることを防ぐため，NVMe SSD と 10GbE NIC を用いた．ここで，移行先サーバでは OS が未対応の NVMe SSD を SCSI デバイスのエミュレーションを介し利用するため，仮想マシンモニタを用い仮想マシン上で動作させた．ファイル共有のプロトコルには，NFSv3 及び SMB3.0 を用いた．移行元サーバ及びクライアントは，両プロトコル

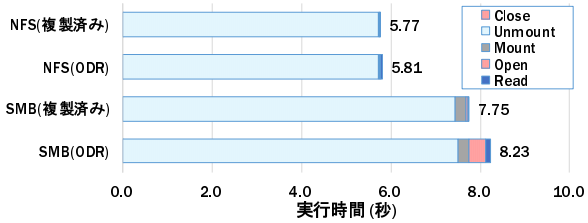

図 6: 移行時のファイルアクセス停止時間

の評価において OS のみ入れ替え，同一のハードウェアを用いた．

5.2 移行におけるクライアントのアクセス停止時間

本節では，図 5 における **(a) 移行実施前**から **(b) 移行中**への遷移の過程で，ファイルアクセスに 30 秒以上の停止時間が生じないことを実験する．

本実験では，まず移行元サーバに 100GB の大ファイルを格納し，クライアントから接続して先頭から 10MB ずつ順次読み込みを行う．途中で接続先変更を模擬するため，クライアントはファイルを半分まで読み込んだ後，ファイルのクローズ (Close)，移行元サーバのアンマウント (Unmount)，移行先サーバのマウント (Mount)，ファイルオープン (Open)，続きのオフセットから 10MB 読み込み (Read) を順に実行し，アクセスを継続する．移行元サーバで最後に読み込みを行った後，移行先サーバの最初の Read 完了までの時間をアクセス停止時間とみなし，その間の各手順の経過時間を計測する．

測定は ODR を用いてファイル複製を行いながらファイル読み込みを継続する場合と，移行元と移行先でデータが複製完了済みである状態で，単に接続先を切り替えてファイル読み込みを継続する場合を比較し，ODR におけるアクセス停止時間を評価する．

測定結果を図 6 に示す．ODR を用いた場合，30 秒を大きく下回る停止時間でファイル読み込みを継続できた．Open 及び Read は移行先サーバで ODR によるデータ複製が生じ停止時間が伸びるが，その増分は最大でも SMB の 0.48 秒と影響は限定的である．NFS・SMB いずれも Unmount に時間がかかっているが，これは OS がファイルシステムのアンマウントに際しページキャッシュ・ファイルキャッシュの解放を行うためである．これらのキャッシュ解放時間は，搭載メモリ容量に比例してさらに伸び，30 秒を超える可能性があるが，キャッシュは事前に明示的に破棄しておくことで短縮可能である．

これらの結果より，提案方式は複製未完了のファイルへのアクセスの応答時間を若干増加させるが，それでもアクセスの停止時間は目標の 30 秒に対し非常に短く済むことを確認できた．

表 2: 実験機器

項目	移行先サーバ (括弧内は VM の構成)	移行元サーバ	クライアント
台数	1	1	1
CPU	Xeon E5-2603 × 2 (8 vCPU)	同左	同左
メモリ	64GB (16GB)	64GB	同左
データ用ディスク	NVMe SSD × 3 (LSI Logic SAS)	NVMe SSD	なし
データ用 NIC	10GbE Copper (VMXNET3)	同左	同左
VMM/OS	(VMM) VMWare ESXi 6.5 (Guest OS) Linux 2.6.30.1	(NFS) Ubuntu Server 16.04 LTS (SMB) Windows Server 2016	同左

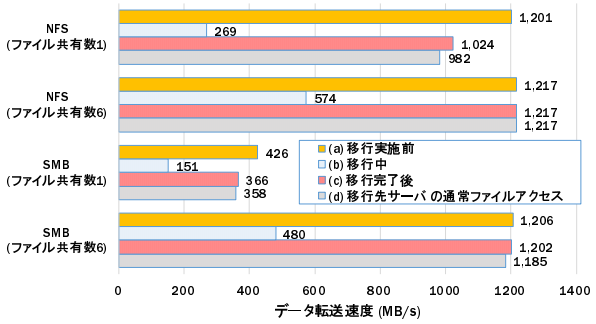

図 7: On-Demand Replication 実行中のファイルアクセス性能

5.3 ODR 実行中のファイルアクセス性能

本節では，スタブ管理及び ODR の導入による性能オーバヘッドを評価する．そのため，1GB のファイルを 100 個格納したファイル共有を作成し，クライアントからファイルをシーケンシャルに読み込み，その転送性能を測定した．クライアントはファイル共有毎に 1 ファイルずつ読み込みを行っており，同時にアクセスするファイル共有数は単一 (1) と複数 (6) でそれぞれ実施した．測定を行う構成は，図 5 が示す 3 つの状態と，本複製方法を用いず，**(d) 移行先サーバの通常ファイル**を参照する場合の計 4 通りである．

図 7 に測定結果を示す．**(a) 移行実施前**及び **(d) 移行先サーバの通常ファイルアクセス**は，それぞれスタブ管理や ODR を伴わないファイルサーバのアクセス性能を示す．

ODR を生じる **(b) 移行中**のケースでは，NFS・SMB 共に移行元・移行先サーバの性能より大幅に低い転送性能となった．ODR 実行時，クライアントのファイル読み込み要求に対し，移行先サーバは移行元からファイルを読みこみ，自身のファイルシステムに書き込んだ後，クライアントに応答する．そのため **(b)** の性能は両サーバの性能値 **(a)**，**(d)** の半分以下となることが想定であり，この測定値はそれを裏付ける結果となった．

一方，**(c) 移行完了後**と **(d) 移行先サーバの通常ファイルアクセス**の測定値がほぼ等しいことより，一度 ODR を実行したファイルを再度読み込む場合には余分なオーバヘッドが生じないことが確認できた．なお，**(d)** より **(c)** の方が性能が良いケースが見られるが，これは両者の対象ファイル生成時の I/O パターンの差により，ディスク上のブロック配置に差が生じたことが原因であると考えられる．

5.4 考察

本章で実施した実験より，提案手法が 2.6 節で述べた従来方式の課題を解決出来たことを確認する．

本実験では，Linux と Windows を用いた移行元サーバからそれぞれ移行を行えたことから，提案手法が移行元サーバの機種に依存しないことを確認できた．

5.2 節に示す測定結果より，クライアントのファイルアクセス停止時間を目標の 30 秒以下で達成できる見込みを得た．今後の課題として，多数のクライアントが同時に切り替えを行う場合や，多様なワークロードにおける効果の評価がある．

移行に伴うオーバヘッドの観点では，5.3 節が示す通り，一度複製が完了したファイルに対しては移行完了後にはオーバヘッドが残らず，通常ファイルと同程度のファイルアクセス性能を出せることを確認できた．

提案手法は，Pre-Copy 方式が必要とする長時間のアクセス停止時間を必要としない代わりに，個々の複製未完了ファイルのアクセス時に分散的に応答時間をわずかに増加させる手法と言える．ただし，その応答時間の増分は許容時間 30 秒に対し大幅に短く，かつ複製が完了したファイルに大しては性能オーバヘッドを生じないため，ファイルサーバの移行に適している．

6. おわりに

本稿では，クライアントのファイルアクセス停止時間を削減できるファイルサーバの移行方式を提案し，実装及び性能評価を行った．従来方式では，移行元と移行先が同一機種でなければならない，クライアントのファイルアクセスが長時間停止する，移行後に性能オーバヘッドが残る，と言った課題があった．提案方式では，クライアントからアクセス要求があったファイルをその時点で複製する ODR と，アクセス要求のないファイルをサービス負荷の低い時間帯に複製する BGR を組み合わせてデータを複製する．また，ファイル単位での Post-Copy 方式複製を行うため，スタブ管理機能を用いてディレクトリツリーを段階的に複

製する．本方式により，異機種ファイルサーバ間において，クライアントのアクセス停止時間を目標の 30 秒以下に抑え，かつ完了後にオーバヘッドを残さない移行を実現した．

今後の課題は以下のとおりである．本報告における性能評価は，基本的なワークロードに留まっている．今後本方式の有効性の検証には，多数のクライアントや多様なワークロード下の複製性能評価といった，より実環境に即した構成に対する評価が必要である．また，提案手法はクライアントからファイルサーバへのマウント/アンマウントを実行するため，アプリケーションは一度ファイルをクローズする必要がある．SMB が備える透過フェールオーバ機能など，プロトコルのサポートを利用し，アプリケーションに透過的な移行を実現することが考えられる．

参考文献

[1] Schroeder, B. and Gibson, G. A.: Understanding Disk Failure Rates: What Does an MTTF of 1,000,000 Hours Mean to You?, *In ACM Transactions on Storage*, Vol. 3, No. 3, pp. 8:1–8:31 (2007).

[2] Intel, Inc.: Planning Guide: Updating IT Infrastructure (online), available from ⟨http://www.intel.com/content/dam/www/public/us/en/documents/guides/server-refresh-planning-guide.pdf⟩ (accessed 2017-10-10).

[3] TechTarget: Snapshot 1: New NAS buys motivated by performance and outdated hardware, *Storage Magazine*, Vol. 16, No. 2, p. 12 (2017).

[4] TechTarget: NAS trifecta: Price, features and performance, *Storage Magazine*, Vol. 16, No. 8, p. 14 (2017).

[5] JM Project: Linux Programmer's Manual: NFS (online), available from ⟨https://linuxjm.osdn.jp/html/util-linux/man5/nfs.5.html⟩ (accessed 2017-10-10).

[6] Olougouna, E. A.: SMB 2.x and SMB 3.0 Timeouts in Windows (online), available from ⟨https://blogs.msdn.microsoft.com/openspecification/2013/03/27/smb-2-x-and-smb-3-0-timeouts-in-windows/⟩ (accessed 2017-10-10).

[7] Tridgell, A. and Mackerras, P.: The rsync algorithm, Technical Report TR-CS-96-05, ANU Research Publications (1996).

[8] Microsoft TechNet: Command-Line Reference Robocopy (online), available from ⟨https://technet.microsoft.com/en-us/library/cc733145(v=ws.11).aspx⟩ (accessed 2017-10-10).

[9] Rikard, R.: SnapVault Best Practices Guide, Technical Report 3487, NetApp (2012).

[10] Mashtizadeh, A., Celebi, E., Garfinkel, T. and Cai, M.: The Design and Evolution of Live Storage Migration in VMware ESX, *In Proceedings of the 2011 USENIX Annual Technical Conference*, pp. 1–14 (2011).

[11] QEMU Wiki: Image Streaming API (online), available from ⟨https://wiki.qemu.org/Features/ImageStreamingAPI⟩ (accessed 2017-10-10).

[12] Yoshiko, Y., Shinichi, K., Atsushi, E., Jun, O. and Tatsuo, H.: Concept and evaluation of X-NAS: a highly scalable NAS system, *In Proceedings of 20th IEEE Mass Storage Systems and Technologies*, pp. 219–227 (2003).

[13] Wataru, K., Satoshi, Y., Takashi, T., Jun, I., Yoshihide, K., Kouji, Y., Kazuaki, F. and Toshihiro, N.: NAS switch: a novel CIFS server virtualization, *In Proceedings of 20th IEEE Mass Storage Systems and Technologies*, pp. 82–86 (2003).

[14] Jun, N., Atsushi, S. and Masaaki, I.: Directory-Aware File System Backup to Object Storage for Fast On-Demand Restore, *International Journal of Smart Computing and Artificial Intelligence*, Vol. 1, No. 1, pp. 1–19 (2017).

[15] Vrable, M., Savage, S. and Voelker, G. M.: Cumulus: Filesystem Backup to the Cloud, *In ACM Transactions on Storage*, Vol. 5, No. 4, pp. 14:1–14:28 (2009).

[16] Tanenbaum, A. S.: *Modern Operating Systems*, Prentice Hall Press, 3rd edition (2007).

[17] Hines, M. R., Deshpande, U. and Gopalan, K.: Post-copy Live Migration of Virtual Machines, *In ACM SIGOPS Operating System Review*, Vol. 43, No. 3, pp. 14–26 (2009).

[18] Hitz, D., Allison, B., Borr, A., Hawley, R. and Muhlestein, M.: Merging NT and UNIX Filesystem Permissions, *In Proceedings of the 2nd USENIX Windows NT Symposium* (1998).