ハイブリッドクラウド環境の拠点間転送量を削減するファイル更新範囲を用いた細粒度差分転送方式の提案

鴨生 悠冬[1,2] 早坂 光雄[1] 高田 昌忠[1] 山内 利宏[2]

概要：近年，ハイブリッドクラウドの普及に伴い，バックアップや，データ分析・生成 AI 向けのデータ集約等を目的とした拠点間ファイル転送が増加している．一方，WAN 帯域がボトルネックとなり，転送時間が長大化する．本稿では，実際の更新範囲，転送先に存在する同一データ，および転送対象データ間の類似性を転送量削減に利用できることに着目する．これらをそれぞれ利用する細粒度差分転送の方式として，書込み時に記録した更新範囲のみを転送するバイト差分転送，転送先と同一のデータを送らない転送データ重複排除，および類似するデータをまとめて圧縮する転送データ圧縮の 3 つの方式を提案する．このうちバイト差分転送を試作して評価した結果，更新が小さい場合（102.5MiB のファイルの 0.01%を更新），既存手法 rsync に比べて転送量を約 92%，帯域 100Mbps の条件での転送時間を約 87%削減した．

キーワード：ハイブリッドクラウド，拠点間ファイル転送，差分転送，オブジェクトストレージ

A Fine-Grained Delta Transfer Method Using File Update Ranges to Reduce Inter-Site Transfer Volume in Hybrid Cloud Environments

YUTO KAMO[†1†2] MITSUO HAYASAKA[†1]
MASANORI TAKADA[†1] TOSHIHIRO YAMAUCHI[†2]

Abstract: Inter-site file transfer is increasing with the spread of hybrid cloud environments, while WAN bandwidth remains a bottleneck. This paper focuses on three kinds of information usable for reducing transfer volume: the actual update ranges, data identical to that already present at the destination, and similarity among the data to be transferred. We propose fine-grained delta transfer consisting of three methods that exploit these respectively. We implemented and evaluated the first method, byte-level delta transfer. For small updates (0.01% of a 102.5 MiB file), compared with rsync, an existing delta-synchronization tool, it reduced the transfer volume by about 92 % and the transfer time under 100 Mbps by about 87 %.

Keywords: hybrid cloud, inter-site file transfer, delta transfer, object storage

1. はじめに

ハイブリッドクラウドの普及に伴い，オンプレミス拠点とクラウドの間でファイルを転送する機会が増えている．例えばバックアップ，データ分析基盤へのデータ集約，生成 AI の学習データ収集などがある．これらの用途では，拠点間を接続する WAN の帯域が転送時間を支配し，クラウド事業者の課金にも影響する．したがって，転送するデータ量そのものを削減する技術の重要性が高まっている．

転送量削減の基本的な考え方は，既に転送先に存在するデータを再送しないことである．しかし，既存の手法にはそれぞれ制約がある．rsync[1]に代表される内容照合型の差分同期は，差分を発見するために，更新が小さい場合でもファイル全体の走査とチェックサムの交換を必要とする．rclone[2]に代表されるクラウドストレージ同期ツールは，更新されたファイルを全体として再アップロードする．S3 互換オブジェクトストレージへの転送では，Multipart Upload のパート単位（最小 5MiB）が転送の粒度となるため，数 KiB の更新であっても 5MiB 以上の転送が発生する．また，ZFS・Btrfs の増分 send[3][4]のように変更ブロックを追跡する技術は，内容照合なしに差分を特定できるが，ボリュームやスナップショットを単位とし，同じ形式の複製先を前提とするため，部分更新 API を持たない S3 互換オブジェクトストレージへファイル単位で反映できない．

一方，ファイルシステム（FS）への書込み処理の時点では，どのバイト範囲が実際に書き換えられたかという情報を取得できる．本情報を用いれば，内容の照合を行わずに，更新されたバイト範囲のみを転送対象として特定できる．

本稿では，FS への書込み時に得られる更新範囲を利用する細粒度差分転送を提案する．提案は 3 つの方式からなり，中核となるバイト差分転送は，更新範囲のみを転送する．後段には，転送先データとの同一性を利用する転送データ重複排除と，転送対象データ間の類似性を利用する転送データ圧縮を置く．これらの後段方式については，更新範囲

1 (株)日立製作所
2 岡山大学

ⓒ2026 Information Processing Society of Japan

により処理対象を限定する構成の設計を示す．

本稿の主な貢献は次のとおりである．

(1) FS への書込み処理で記録した更新範囲を入口とする細粒度差分転送方式の枠組みを示した．その中核であるバイト差分転送について，差分集約オブジェクトと，部分更新 API を持たない S3 互換オブジェクトストレージへの反映方式を設計・試作した．後段の重複排除・圧縮は設計と机上評価・予備評価にとどまる．

(2) バイト差分転送の試作実装を rsync・rclone と比較実測し，小さな更新（102.5MiB のファイルの 0.01%を更新）で rsync に比べて転送量を約 92%，帯域 100Mbps での転送時間を約 87%削減できることを示した．

本稿の構成は次のとおりである．2 章で関連研究を整理し，3 章で提案方式を述べる．4 章で評価を示し，5 章で議論と限界を述べ，6 章でまとめる．

2. 関連研究

転送量削減に利用する情報と，転送先での保存形態に基づいて関連研究を整理する．

差分同期 : rsync[1]は，受信側が保持する旧版ファイルをブロックに分割し，そのチェックサムを送信側へ渡す．送信側は移動窓により一致するブロックを探索し，一致部分の参照情報と不一致部分のリテラルデータを送ることで，新版を再構成する. ファイル全体を再転送せずに済む一方，差分の発見は内容の照合によるため，送信側でのファイル走査と署名の交換が必要である．本研究の方式 1 は，書込み時に記録した更新位置と長さを転送時に直接取り出す点が異なる．

低帯域 FS : LBFS[5]は，内容依存の可変長チャンクと衝突耐性ハッシュを用い，送信対象と同一のチャンクがサーバの FS またはクライアントキャッシュに存在する場合に，当該チャンクの送信を省く．同名ファイルの旧版だけでなく，他のファイルに含まれる同一内容も利用できる．本研究の方式 2 はこの考え方に基づくが，方式 1 が抽出した更新範囲を入力とすることで，チャンク化とフィンガープリント（FP）計算の対象量を削減する点を検討する．

WAN 最適化レプリケーション : Shilane らは，バックアップデータの WAN レプリケーションを対象として，重複排除に加えて類似領域をデルタ圧縮する stream-informed delta compression を提案し, 重複排除およびローカル圧縮に対して追加で約 2 倍の圧縮効果を報告している[6]．同一性と類似性を段階的に利用する点で本研究に最も近い．したがって，重複排除と圧縮を組み合わせること自体は本研究の新規性ではない．本研究との差は，その前段に更新範囲を第三の情報として置き，後段の処理対象そのものを限定する点にある．

類似検出とデルタ圧縮 : Douglis らは，明示的な版関係が与えられていないオブジェクト集合から類似する基準データを検出してデルタ符号化へ利用する考え方を示した[7]. Kulkarni らは大規模ファイル集合における冗長性除去を扱い[8], Finesse は重複排除後のデルタ圧縮に向けた細粒度の特徴局所性による高速な類似検出を提案している[9]．本研究の方式 3 も類似判定には既知技術を利用するため，類似検出アルゴリズムそのものを新規性の中心には置かない．方式 3 は，類似チャンク間で専用のデルタ符号化を行うのではなく，類似チャンクを近接配置して汎用圧縮の窓に収める点が異なる．

チャンク重複排除型バックアップ : S3 等を保存先とする Restic，Kopia，Duplicati[10][11][12][13]は，増分バックアップの実用的な比較対象である．いずれもファイルをチャンク（Restic・Kopia は内容依存の可変長，Duplicati は固定長ブロック）に分割して既存内容を再利用し，複数チャンクをまとめて格納する独自形式のリポジトリに保存する．対して本研究は，FS 反映では元のファイル名のまま，S3 反映では UUID をキーとする本体・属性・ディレクトリの名前一覧を別オブジェクトとして保持する．これらのツールとの比較は今後の課題とする．

内容照合なしに変更箇所を特定する技術には，1 章で述べた ZFS・Btrfs の増分 send[3][4]に加え，スナップショット間で変化したブロックを非同期に複製する SnapMirror[14]や，仮想ディスクの変更ブロック追跡（VMware CBT）[15]がある．これらは方式 1 と考え方を共有するが，FS 全体やボリュームを単位とし，同じ形式の複製先への反映を前提とする．本研究は，ファイル単位・バイト範囲で記録した更新範囲を，部分更新 API を持たない S3 互換オブジェクトストレージへ反映し，さらに重複排除・類似圧縮の入力の限定に用いる点で異なる．

以上より，本研究の新規性は，差分転送・重複排除・類似検出・圧縮という個別の要素技術ではなく，書込み処理から得られる実際の更新範囲を起点とする構成にある．具体的には，更新範囲のみを抽出・集約して転送し，部分更新 API を持たない S3 互換オブジェクトストレージへ反映する方式を設計・試作し，バイト差分転送が有効となる条件を実測で示す．さらに，同一性と類似性を利用する後段の処理対象と適用判断を更新範囲によって限定する設計を示す．ただし，後段との併用効果および適用判断の有効性の実測検証は今後の課題である．

3. 提案方式

3.1 提案方式の概要

提案方式は，転送量削減に次の 3 つの情報を利用する．

- 更新範囲：どのバイト範囲が実際に更新されたか.
- 同一性：転送先に同一のデータが存在するか.
- 類似性：転送対象のデータ間に圧縮可能な類似性があるか.

これらにそれぞれ対応する方式として，方式 1（バイト

差分転送），方式 2（転送データ重複排除），方式 3（転送データ圧縮）を定義する．方式 1 は，更新範囲を転送対象の決定に用いると同時に，方式 2,3 への入力を限定する役割をもつ．

提案方式のシステム構成を図 1 に示す．転送元拠点のファイルストレージでは，プロトコル処理とファイルシステムの間に置いた IO Hook がアプリケーションのファイル操作を捕捉し，書込みの位置と長さを更新範囲として管理情報に，作成・削除・名前変更・切詰め・属性変更等のファイル操作を，操作の種別と対象ファイルを発生順に記録する操作ログ（oplog）に記録する．dataMover は，oplog から転送ファイル，管理情報から転送対象データを決定し，ファイル実体から差分を読み出す．dataMover は方式 1〜3 に対応するバイト差分転送，転送データ重複排除，転送データ圧縮の処理部を有し，方式 2 では転送先拠点の FP 索引との間で FP の照合（要求・応答）を行う．処理結果は差分集約オブジェクトとして WAN を介して転送先拠点の受信処理へ送られ，受信処理は展開・復元を行ったうえで，FS または S3 互換オブジェクトストレージへ反映する．

本稿の評価範囲は方式ごとに異なる．方式 1 は試作実装による実測評価の対象とする．方式 2 は設計を示し，既存の測定結果を前提とした机上評価にとどめる．方式 3 は設計を示し，並べ替えと圧縮の処理のみを実装した予備評価を行う．3 つの方式を併用したときの効果は評価しておらず，今後の課題である．

3.2 処理の流れと適用判断

3 つの方式の処理の流れと適用判断を図 2 に示す．3 つの方式は，転送元から転送先への一連の流れの中で次の順に組み合わせる．

- 転送対象の決定（方式 1）：IO Hook が記録した更新範囲と oplog から，転送対象の差分を決定する．
- 方式 2 の適用判断：重複排除による転送量の削減期待値（差分サイズと転送先との重複率の積）が，照合と参照に要する通信量（FP，要求・応答，および重複チャンクの参照）を上回る場合のみ方式 2 を適用する．
- 重複排除（方式 2）：差分をチャンクに分割し FP を算出し，転送先の索引と照合する．重複と判定されたチャンクは本体を送らず，その FP を参照として差分集約オブジェクトに格納する．
- 類似チャンクの近接配置（方式 3）：重複と判定されなかったチャンク，および方式 2 を適用しなかった差分について，差分のサイズが圧縮ウィンドウを超える場合に，削減見込みが構成情報のサイズを上回る類似チャンクのみを近接配置する．
- 圧縮と転送（方式 3）：並べ替えの適用の有無にかかわらず，非重複チャンクのデータと重複チャンクの参照を差分集約オブジェクトにまとめて圧縮し，WAN 上の HTTP 要求で転送する．

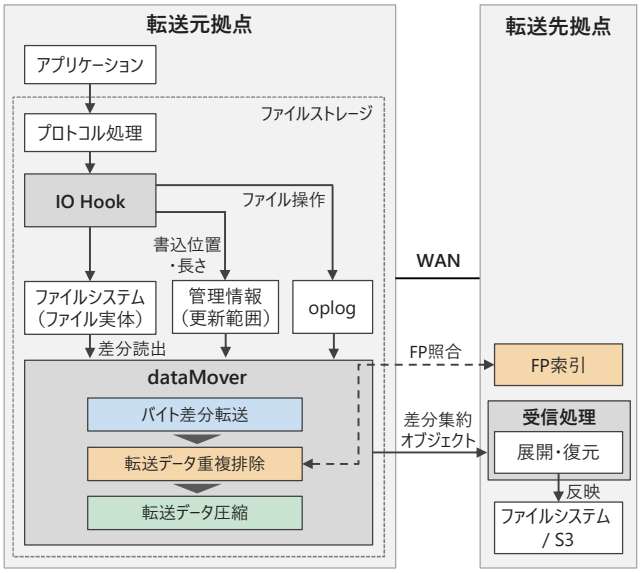

図 1　細粒度差分転送方式のシステム構成

Figure 1　System architecture of fine-grained delta transfer.

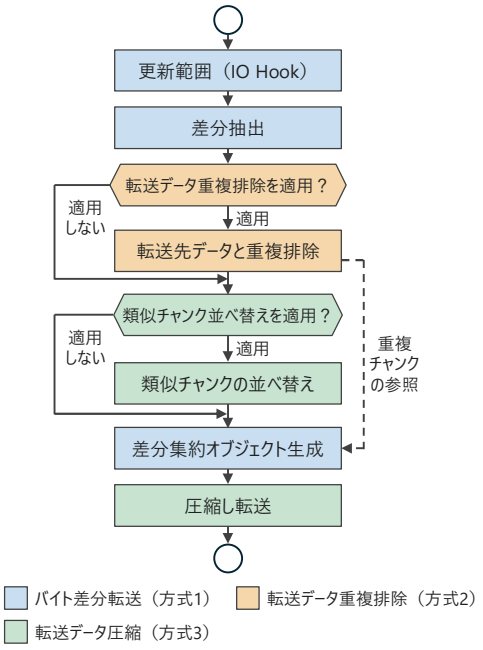

図 2　3 つの方式の処理の流れと適用判断

Figure 2　Processing flow and applicability decisions.

- 転送先での反映：図 1 に示す受信処理がオブジェクトを展開し，並べ替えを行った場合は元の配置に復元する．重複チャンクの参照が含まれる場合は FP 索引から格納先を求めて同一データを複製し，差分をファイルシステムまたは S3 互換オブジェクトストレージへ反映する．

このように，更新範囲は転送対象を決めるだけでなく，後段の照合・圧縮の処理対象を限定し，方式 2 と方式 3 の適用判断の基礎となる．

ⓒ2026 Information Processing Society of Japan

3.3 方式 1：バイト差分転送

方式 1 は, 図 1 のうち IO Hook, 管理情報, oplog, dataMover のバイト差分転送部，および転送先拠点の受信処理によって実現する．以下では，差分抽出，差分集約オブジェクト，転送と反映の順に述べる．

3.3.1 差分抽出

IO Hook は，アプリケーションによるファイル I/O の各要求を捕捉する．IO Hook は，ファイルごとに，オフセット，長さ，状態の組からなる範囲のリストを管理情報として管理する．書込みを受けると，書込み範囲と重なる範囲の状態を，未転送の更新を表す DIRTY とする．ファイルの末尾を超える書込みでは，DIRTY の範囲を末尾に追加する．転送時には，DIRTY の範囲を統合して転送対象のバイト範囲の集合を得る．転送が完了した範囲は未更新の状態に戻す．転送中に同じ範囲へ新たな書込みがあった場合，その範囲は DIRTY のまま残り，次回の転送対象となる．

転送時にはファイル実体から最新の内容を読み出すため，転送先には記録時点以降の内容が反映される．作成，削除，名前変更，属性変更，切詰めは，IO Hook が oplog として dataMover へ通知する．転送先のファイルは UUID で識別するため，名前変更は範囲のリストに影響しない．差分の発見に内容の照合を用いないため，rsync のような，差分を発見するためのファイル全体の読み込みとブロックチェックサムの計算は不要である．

3.3.2 差分集約オブジェクト

抽出した複数のバイト範囲は，一つの差分集約オブジェクトにまとめて転送する（図 3）．オブジェクトは, ヘッダ，エントリ表，およびデータ列からなる．ヘッダは，エントリ表とデータ列の長さを保持する．各エントリは，対象ファイルを識別する UUID，操作種別，オフセット，長さを保持する．ファイルパスは含めず，UUID により転送先のファイルを指定する．

この形式により，多数の小さな更新範囲を 1 回の HTTP 要求で転送でき，範囲ごとに要求を発行する場合に生じる往復遅延とプロトコル固定費の繰り返しを避けられる．一方, エントリ表のサイズは更新範囲の個数に比例するため，断片化した更新ではオーバーヘッドが増える．

3.3.3 転送と反映

差分集約オブジェクトは，1 回の HTTP 要求で転送先の受信処理へ送り，反映先に応じて以下の処理を行う．

FS への反映では，UUID からファイルハンドルへの対応表を引き, エントリが示すオフセットへ pwrite で書き込む．

S3 互換オブジェクトストレージへの反映では，オブジェクトの部分更新が API として提供されないため，次のように構成する．対象が 5MiB 未満の場合は，GET により取得した内容に差分を適用して PUT する．5MiB 以上の場合は，オブジェクトを 5MiB 単位の格子をもつ Multipart Upload として再構成し，差分を含む区画のみ GET と UploadPart を

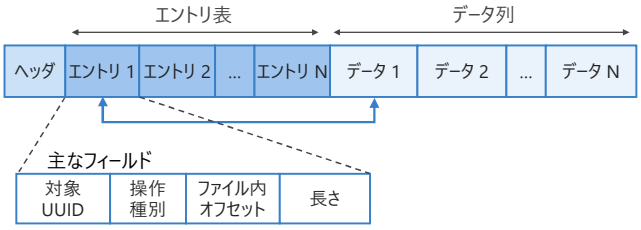

図 3　差分集約オブジェクトの構造

Figure 3　Structure of the aggregated delta object.

行い，それ以外の区画は UploadPartCopy で複製する．転送先のオブジェクトキーは UUID から一意に定め，データ本体，属性，ディレクトリの名前一覧を別のオブジェクトとして保持する．このため，名前変更ではディレクトリの名前一覧のみを更新し, 本体は書き換えない. Multipart Upload による再構成は完了時に一括して置き換わるため，読み手には更新前または更新後の状態のみが見える．属性と名前一覧は，取得時の ETag を条件とする条件付き PUT で更新し，同時更新を検出した場合は上書きせずに再実行する．

これにより, WAN を越える転送量を更新範囲のみに抑えつつ，転送先での部分更新を実現する．ただし，転送先内部では，差分を含む区画数に比例した GET と，それ以外の区画数に比例した UploadPartCopy が発生する．

3.4 方式 2：転送データ重複排除

方式 1 が抽出した差分をローリングハッシュで可変長チャンクに分割し, 各チャンクの SHA-256 で計算した FP を，転送先が保持する FP の索引と照合して，転送先に既に存在するチャンクの本体を送らない．チャンク長は，重複の検出率と FP の通信量の兼ね合いから，一般に数 KiB 程度が適当である．重複と判定されたチャンクについては，本体の代わりにその FP を参照として差分集約オブジェクトに格納する．転送先の FP 索引は，FP と同一データの格納先（ファイルとオフセット）を対応付けて保持し，受信処理は索引から格納先を求めて同一データを複製し，反映する．このため，照合の応答は重複の有無のみでよい．照合の対象を更新範囲内のデータに限定できるため，照合の処理量と通信量を抑えられる．適用の判断は 3.2 節に示した条件による．本方式は基本設計のみで実装しておらず，評価は 4.2 節の机上評価による．

3.5 方式 3：転送データ圧縮

方式 2 で重複と判定されなかったチャンク，または方式 2 を適用しなかった差分を分割したチャンクについて，チャンク分割で計算したローリングハッシュ値のうち，最小チャンク長以降で得られた値の大きいものから 4 つをスケッチとして求め，スケッチの 4 値がすべて一致するチャンクを類似とみなし，チャンク単位で並べ替えて近接配置する．これにより，汎用圧縮アルゴリズムの圧縮ウィンドウ内に類似チャンクが収まり, 冗長性が検出されやすくなる．並べ替えたチャンク列を差分集約オブジェクトとして圧縮

ⓒ2026 Information Processing Society of Japan

し，並べ替えを元に戻すための構成情報を付与して転送先で復元する．並べ替えの適用は転送元で判断し，方式 2 の適用判断とは独立である．差分の総サイズが圧縮ウィンドウ以下であれば，元の並びのままで全チャンクが参照可能なため並べ替えない．そうでない場合は，元の並びで圧縮ウィンドウより離れて位置する類似チャンクについて，近接配置による圧縮後サイズの削減見込みが，そのチャンクの並べ替えに要する構成情報のサイズを上回る場合に限り並べ替え，それ以外のチャンクは元の並びのまま残す．

判断に用いるスケッチの計算量は差分のサイズに比例し，方式 1 の更新範囲によって限定される．類似チャンク間で専用のデルタ符号化を行う方式に比べ圧縮率で劣る可能性がある一方，基準チャンクの索引と保持への依存を抑えられる．

4. 評価

4.1 方式 1 の実測評価

4.1.1 評価の方針と環境

3 つの方式は単独で評価し，方式間で数値を直接比較しない．方式 1 は試作実装による実測，方式 2・3 は，それぞれ既存の測定結果を前提とした机上評価，並べ替えと圧縮の処理のみを実装した予備評価である．

次の 3 点を実測により検証する．

- H1 : 更新範囲を用いると，更新が小さいほど転送量を削減できる．
- H2 : 転送量の削減は，帯域が制限された条件で転送時間の短縮に反映される．
- H3 : 複数の更新範囲を一つの差分集約オブジェクトにまとめたとき，ヘッダとエントリ表による転送量の増加率は, payload のサイズと更新範囲の個数に依存する．payload が極小の場合を除き，この増加率は数%以下に収まる．

表 1 に示す 5 方式を比較する．byte は提案方式（更新されたバイト範囲のみを転送）である．part5m と whole は同一実装上のベースラインで，それぞれ更新を含む 5MiB 区画単位（オブジェクトストレージの Multipart Upload のパート単位に相当），ファイル全体単位で転送する．

転送先での保存形態が異なる方式を直接比較することは避け，反映先を揃えた 2 組，すなわち，FS へ反映する組（byte/part5m/whole/rsync）と，S3 互換オブジェクトストレージへ反映する組（byte/part5m/whole/rclone）で比較する．

送信側・受信側として，同一の仮想環境上の 2 台（各 4vCPU，メモリ 8GB，Ubuntu 24.04.4，kernel 6.8.0-136）を用いた．両機は同一のサブネットに接続し，MTU は 1500 とした．測定は他の通信の影響を受けうる共有基盤上で行った．TSO/GSO/GRO は有効である．帯域制限なしの条件では，iperf3 による実効帯域は約 8.7Gbps，往復遅延は約 1〜2ms であった. 帯域制限の条件では, 両側の NIC に netem

表 1　各ファイル転送方式の比較
Table 1　Comparison of file transfer methods.

方式	転送の粒度	反映先	備考
byte（提案）	更新されたバイト範囲	FS / S3	提案方式
part5m	更新を含む 5MiB 区画	FS / S3	ベースライン
whole	ファイル全体	FS / S3	ベースライン
rsync 3.2.7	ブロックチェックサムによる差分	FS	ssh 経由，圧縮なし
rclone v1.60	ファイル全体	S3	圧縮なし

delay 5ms rate 100Mbit を設定し，往復遅延約 11ms，iperf3 による実効帯域 99.8〜100.4Mbps, 再送 0 を確認した. rsync は-t（更新時刻の保持），--inplace（転送先の直接書換えにより byte と反映方法を揃える），--no-whole-file（差分アルゴリズムを常に使用）を，rclone は並列化による差を除くため--transfers 1 と--checkers 1 を指定した．S3 互換オブジェクトストレージには MinIO（RELEASE.2025-08-29T21-27-49Z）を用い，試作実装の HTTP 転送には libcurl 8.5.0 を用い, 差分集約オブジェクトは multipart/form-data 形式で送信した．試作では FUSE を用いて IO Hook を実装した．

転送量は，送信側ホストの nftables による外部計測（IP 層以上の送受信合計）を主指標とする．各方式の通信は宛先ポートにより区別して計測した. payload および HTTP 本文のサイズは，試作実装が出力する計測ログから取得し，内訳として併記する．時間は，転送コマンドの開始から終了までの経過時間であり，byte については，更新範囲の抽出，差分集約オブジェクトの生成，転送，および転送先での反映の完了までを含む．初期データの配置に要する転送は計測対象から除く．

単一ファイルの評価では，初期サイズ 102.5MiB のファイルを対象とし，更新率 8 水準（0.01，0.1，1，5，10，20，50，100%）と追記 5 水準（1KiB，64KiB，1MiB，5MiB，50MiB）の計 13 条件に，5MiB 境界に整列した場合の挙動を確認するための整列追記 1 条件（初期サイズ 100MiB，追記 1KiB）を加えた 14 条件を用いた．初期サイズを 5MiB の倍数からずらしているのは，末端に端数の区画が生じる一般的な場合を主条件とするためである．更新は，初期サイズに対する比率に相当するバイト数を 4KiB 単位で，重複しない一様ランダムな位置に書き込む. 書込みは 4KiB 単位に切り上げるため，実際の更新量は名目の比率より大きくなる（たとえば名目 0.01%は 3 個，すなわち 12,288B で実効 0.0114%である）．本稿では名目の比率で条件を示す．各条件 5 反復とし，反復ごとに乱数シードを変更した．中央値を主値，最小〜最大を変動の範囲として報告する．同一の反復番号では，全方式が同一の更新位置・内容を用いる．全試行について，転送後のファイルの SHA-256 が送信元と一致すること，終了コードが 0 であること，および計測区間終了後の通信が 0B であることを確認した．

ⓒ2026 Information Processing Society of Japan

以上の構成で，単一ファイルは 2 反映先×2 帯域×4 方式（反映先ごとの組）×14 条件×5 反復=1,120 試行，複数ファイルは 2 反映先×2 帯域×4 方式×4 条件×5 反復=320 試行を実施した．以下の値はいずれも 5 反復の中央値である．

4.1.2 転送量

更新率および追記量に対する転送量を，FS 反映について図 4 に，S3 反映について図 5 に示す．表 2 に代表的な条件の値を示す．

byte の転送量は更新率にほぼ比例し，全条件で whole とほぼ同量以下であった．更新率 0.01%では 14.2KiB で，whole（105,129KiB）の 0.013%（約 1/7,400）にあたる．更新率 100%では byte と whole がほぼ同量（100.02%）となり，差分集約オブジェクトのヘッダおよびエントリ表による増加は，この測定の精度では現れなかった．更新が全域に及ぶ場合でも byte が whole を実質的に上回らないのは，連続する更新範囲が転送時に 1 エントリへ統合されるためである．以上より H1 を支持する．

part5m は，更新率 0.01%で whole の 14.6%，0.1%で 75.6%となり，1%以上では whole と区別できなくなった．更新が 4KiB 単位でファイル全体に分散する本ワークロードでは，更新率が 1%に達するとほぼすべての 5MiB 区画が更新を含むためである．この結果は，オブジェクトストレージのパート単位の転送では，更新が小さくても削減効果が限定的であることを示す．

追記では，初期サイズが 5MiB の倍数でない場合，part5m は byte に末端の端数区画（約 2.5MiB）を上乗せした転送量となった（追記 1KiB で 2,569KiB，byte は 2.8KiB）．初期サイズを 100MiB に整列させた条件では，part5m は byte と同量（2.8KiB）であった．すなわち part5m の挙動は，追記位置と区画境界の関係に強く依存する．

rsync の転送量は，小さな更新・追記で byte の 3〜47 倍であった（追記 1KiB で 46.6 倍，更新率 0.01%で 12.4 倍）．理由は 2 点ある．第一に，rsync は受信側の旧版ファイルをブロックに分割し，各ブロックのチェックサムを送信側へ送る．既定のブロック長はファイルサイズの平方根程度であり，102.5MiB のファイルではブロック長約 10KiB，ブロック数約 1 万，チェックサムは計約 122KiB（1 ブロックあたり約 12B）となり，更新の大きさによらず送られる．第二に，4KiB の更新でも更新を含むブロック全体（約 10KiB）がリテラルデータとして送られる．これに対し byte は，書込み時に記録した更新範囲のみを送り，照合のための通信を要しない．更新率が上がるとこの固定的な転送量の比率が小さくなり，更新率 100%および追記量 5MiB 以上では byte とほぼ同量となった．

rclone の転送量は，更新・追記の大きさによらず更新後のファイル全体に相当する約 100〜153MiB であり，byte との比は 1.0 倍（更新率 100%）から 37,790 倍（追記 1KiB）に及んだ．

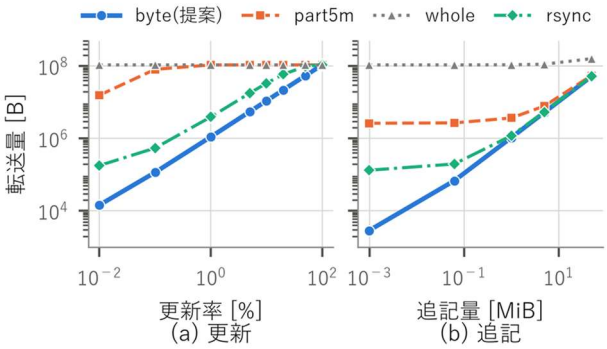

図 4　FS 反映：転送量（IP 層以上）

Figure 4　Transfer volume (FS destination).

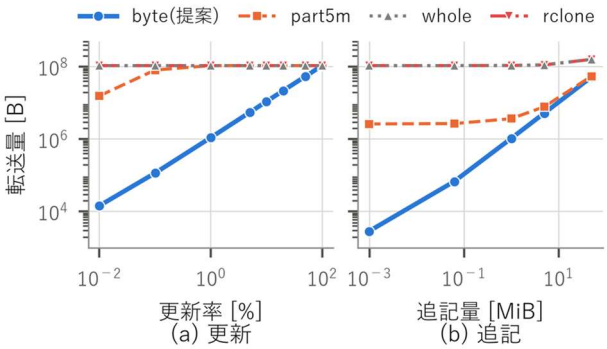

図 5　S3 反映：転送量（IP 層以上）

Figure 5　Transfer volume (S3 destination).

表 2　代表条件の転送量（KiB，IP 層以上）

Table 2　Transfer volume for representative conditions.

(a)　FS 反映

条件	byte	part5m	whole	rsync	rsync/byte
更新 0.01%	14.2	15,385	105,129	175	12.4
更新 0.1%	113	79,488	105,136	532	4.7
更新 1%	1,083	105,123	105,131	3,846	3.6
更新 100%	105,139	105,134	105,118	106,022	1.0
追記 1KiB	2.8	2,569	105,139	130	46.6
追記 1KiB（整列*）	2.8	2.8	102,565	124	44.4
追記 50MiB	51,285	53,851	156,428	51,747	1.0

(b)　S3 互換オブジェクトストレージ反映

条件	byte	part5m	whole	rclone	rclone/byte
更新 0.01%	14.2	15,385	105,130	105,103	7,416
更新 0.1%	113	79,484	105,125	105,102	928
更新 1%	1,082	105,124	105,127	105,102	97.2
更新 100%	105,138	105,134	105,132	105,100	1.0
追記 1KiB	2.8	2,569	105,131	105,102	37,790
追記 1KiB（整列*）	2.8	2.8	102,573	102,539	36,868
追記 50MiB	51,286	53,851	156,407	156,374	3.0

*) 初期サイズを 100MiB（5 MiB の倍数）としたファイルへの追記

ⓒ2026 Information Processing Society of Japan

WAN 区間の転送量は，反映先が FS の場合と S3 互換オブジェクトストレージの場合とでほぼ一致した（更新率 1% の byte で，それぞれ 1,083KiB と 1,082KiB）．WAN を流れるのは同一の差分集約オブジェクトであり，転送量が反映先に依存しないことを確認した．

4.1.3 転送時間

帯域 100Mbps・往復遅延約 11ms の条件における転送時間を，FS 反映について図 6 に，S3 反映について図 7 に示す．表 3 に代表値を示す．

whole の転送時間は，更新では更新率によらずほぼ一定であり，追記では追記後のファイルサイズに応じて増加した（FS 反映で 10.0〜15.5 秒）．更新の場合の値（10.2〜10.3 秒）は 102.5MiB を 100Mbps で送る下限（約 8.6 秒）に近い．これに対し，byte の時間は転送量に比例して増加し，更新率 0.01%で 0.13 秒，50%で 5.81 秒，100%で whole と同等の 10.43 秒となった．更新率 0.01%および追記 1KiB では，byte は whole の約 1/77 の時間で完了した．byte の時間が反復間の変動を超えて whole を上回る条件は，FS 反映・S3 反映のいずれにおいても，全 14 条件で現れなかった．以上より H2 を支持する．

rsync は，小さな更新・追記で約 1.0〜1.2 秒を要し，byte の 4.0〜7.8 倍であった（表 3 (a)）．更新率 50%以上では whole に近づき（更新率 50%で 9.06 秒），更新率 100%では rsync（9.89 秒）が byte（10.43 秒）をわずかに下回った（比 0.95）．更新が全域に及ぶ場合，差分の照合で削減できる転送量がなく，両者の差は反復間の変動と同程度である．追記量 50MiB では byte とほぼ同じであった．

小さな更新で rsync が byte より時間を要する要因として，転送量の差（表 2）に加え，更新量によらない送受信側の全体走査・チェックサム計算と，ssh の鍵交換・認証，rsync の起動・ネゴシエーション，ファイル一覧・チェックサム交換に伴う逐次往復が考えられる．byte は TCP 接続の確立と 1 回の HTTP 要求の計 2 往復程度で転送するため，往復遅延が大きい WAN では，小さな更新における優位がさらに大きくなると定性的に見込まれる．rclone も転送先の一覧取得とアップロードの要求をファイルごとに発行する．ただし，本評価の帯域制限下の往復遅延は約 11ms の 1 条件のみであり，時間の内訳と遅延増大時の効果は未測定である．各要因の寄与の定量化と他の遅延条件での検証は今後の課題とする．なお，byte の S3 反映における区画ごとの要求は転送先ホスト内で完結し，WAN の往復遅延には依存しない．

rclone は転送量に比例した時間を示し（更新で約 9.7 秒，追記 50MiB で 14.2 秒），S3 反映の byte の 2.9〜9.6 倍を要した（表 3 (b)）．主因はファイル全体の再アップロードであり，102.5MiB を 100Mbps で送る下限（約 8.6 秒）が大半を占める．byte の S3 反映の時間は転送先での再構成を含むが，それを含めても rclone より短い．

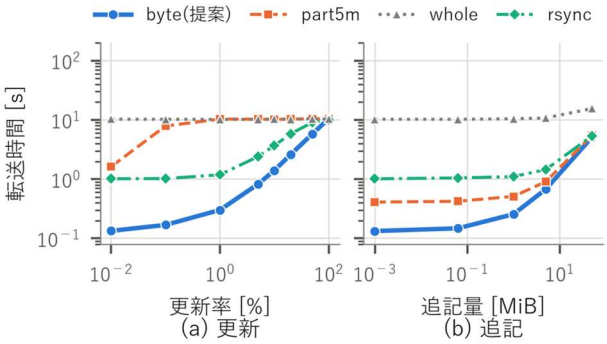

図 6　FS 反映：転送時間（帯域 100Mbps）

Figure 6　Transfer time under 100 Mbps (FS destination).

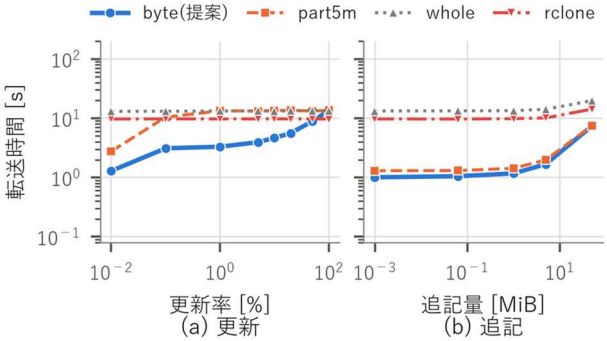

図 7　S3 反映：転送時間（帯域 100Mbps）

Figure 7　Transfer time under 100 Mbps (S3 destination).

表 3　代表条件の転送時間（秒，帯域 100Mbps）

Table 3　Representative transfer times (100 Mbps).

(a)　FS 反映

条件	byte	part5m	whole	rsync	rsync/byte
更新 0.01%	0.13	1.63	10.28	1.02	7.8
更新 1%	0.30	10.30	10.18	1.19	4.0
更新 100%	10.43	10.45	10.27	9.89	0.95
追記 1KiB	0.13	0.41	10.25	1.01	7.8

(b)　S3 互換オブジェクトストレージ反映

条件	byte	part5m	whole	rclone	rclone/byte
更新 0.01%	1.29	2.76	13.06	9.69	7.5
更新 1%	3.30	13.32	13.23	9.71	2.9
追記 1KiB	1.01	1.30	13.21	9.71	9.6

帯域制限がない条件（約 8.7Gbps）では，byte と whole の時間差は縮小した．たとえば追記 1KiB では，byte0.091 秒に対し whole1.336 秒で，比は 14.7 倍にとどまる（100Mbps では 77.1 倍）．帯域がボトルネックでない場合，転送量の削減は時間の短縮に直結しない．

S3 反映の byte の時間は，FS 反映より約 1〜3 秒長い（更新率 1%で 3.30 秒に対し 0.30 秒）．これは，転送先の受信処理が MinIO に対して 5MiB 格子のうち差分を含む区画の GET と UploadPart，およびそれ以外の全区画の

ⓒ2026 Information Processing Society of Japan

UploadPartCopy を実行するためである．この処理は転送先ホスト内で完結し，WAN 区間の転送量には現れない．この区間を含めても，更新率 0.01%では byte は whole の約 1/10 の時間（1.29 秒対 13.06 秒）で完了した．

4.1.4 転送量の内訳

図 8 に，payload に対する byte における差分集約オブジェクト，HTTP 本文，IP 層以上の各サイズの増加率を示す．破線は，固定費 C=1,824B（追記 1KiB における IP 層以上の増加分）が一定と仮定した場合の C/payload である．

IP 層以上の増加率は，追記 1KiB（payload1,024B）の 178%から，追記 50MiB の 0.17%まで，約 3 桁減少した．追記では差分集約オブジェクトの増加分が 128B，HTTP 本文までの増加分が 614B で一定であり，増加率は payload にほぼ反比例して右下がりとなる．ただし IP 層以上の増加分は，payload が大きいほどパケット数が増えるため 1,824B〜約 87KiB と増加し，実測は C/payload の曲線より緩やかに下がる．一方，更新では，IP 層以上の増加率は更新率 0.01%（payload12,288B）で 18%，更新率 1%以上では 1.4〜2.9%であり，追記のように payload に反比例しては下がらない．これは，更新範囲の数に比例するエントリ表が増加分に含まれるためである．

試作実装では，差分集約オブジェクトのヘッダは 32B，エントリは 1 個あたり 96B の固定長である．更新では 4KiB の書込みごとにエントリが生じるが，隣接する書込みは 1 エントリへ統合される．ファイルを 4KiB 単位の n ブロックとし，更新率を p とすると，エントリ数の期待値は約 $n \cdot p(1-p)$であり，ヘッダとエントリ表による増加率は約 $2.3\% \times (1-p)$となる．実測でも，payload に対して更新率 0.01%で 2.6%（ヘッダの寄与を含む），1%で 2.3%，100%で 0.04%未満であり，更新率が大きくなるにつれて低下し，全域の更新では 1 エントリに統合されてほぼ 0 となった．いずれの条件でも増加は payload の数%以下であり，H3 を支持する．以上より，複数の更新範囲を一つの差分集約オブジェクトとして転送することで，更新量が一定以上あればプロトコル固定費とエントリ表の影響を小さく抑えられる．ただし，集約後の payload 自体が極小の場合には，HTTP ヘッダ，TCP/IP ヘッダ，接続確立の通信などの固定費の比率が相対的に大きくなる．

4.1.5 複数ファイル

単一ファイルの結果が，多数のファイルを扱う場合にも成り立つかを確認するため，表 4 に示す大×10 と小×1,000 について，更新と追記の計 4 ワークロードを評価した．

転送量と転送時間を，FS 反映について図 9 に，S3 反映について図 10 に示す．転送時間のエラーバーは最小〜最大を表す．大×10 では，byte の転送量は rsync の 1/39（更新）〜1/80（追記），rclone の 1/27,870〜1/71,660 であった．帯域 100Mbps での更新の転送時間は，FS 反映で byte 0.16 秒に対し rsync 2.07 秒，S3 反映で byte 6.10 秒に対し rclone 45.64 秒であり，多数の大きなファイルでも更新範囲を直接得る方式の優位は保たれた．

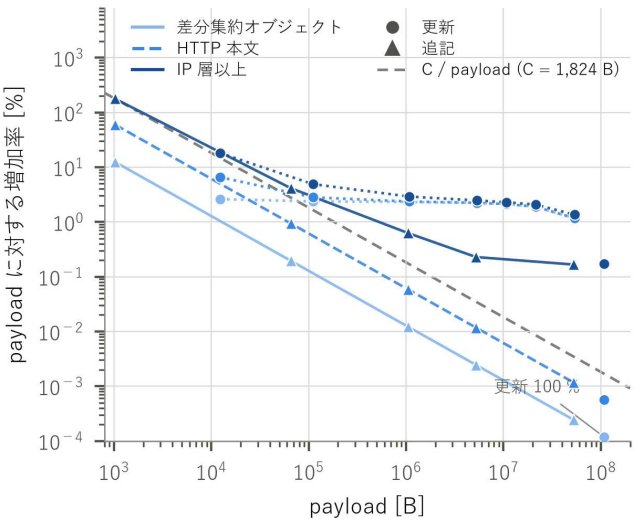

図 8　byte の転送量増加率（FS 反映）

Figure 8　Transfer-volume overhead of byte (FS destination).

表 4　複数ファイルのワークロード

Table 4　Multi-file workloads.

名称	構成	更新	追記
大×10	100MiB ×10 ファイル	5 ファイルの各 3 箇所に 1KiB を更新	5 ファイルに各 1KiB を追記
小×1,000	64KiB ×1,000 ファイル	100 ファイルの各 1 箇所に 1KiB を更新	100 ファイルに各 1KiB を追記

一方，小×1,000 では，FS 反映における byte の転送量は rsync の 1/2.7〜1/2.4 にとどまった．ファイルが小さく，rsync が交換するチェックサムの量も少ないためである．S3 反映では byte は rclone の約 1/70 であった．rclone の転送量は更新した 100 ファイル全体を送る whole より約 2 割多く，ファイルごとの要求の固定費が加わるためと考えられる．また，各ファイルが 5MiB 区画内に収まるため，part5m は whole と同量となった．

小×1,000 の転送時間は，帯域よりもファイル数に応じた処理と往復回数に左右される．FS 反映では，byte は 100 ファイル分の更新範囲を 1 回の HTTP 要求で送るため，ファイルごとにチェックサムを交換・照合する rsync の約 1/4.5 の時間（更新で 0.21 秒対 0.94 秒）で完了した．S3 反映では，ファイルごとに要求を発行する rclone が byte の約 2.9 倍の時間を要した．一方，byte も転送先で各ファイルの GET・PUT を要するため，whole との差は小さかった（4.61 秒対 4.79 秒）．

4.2　方式 2 の机上評価

方式 2 により転送量と転送時間がどの程度改善するかを，既存の測定結果に基づく前提をおいて机上で試算する．方式 2 が対象とするのは，転送先に既に存在するデータとの重複である．転送対象の内部での重複は転送元のみで除去

ⓒ2026 Information Processing Society of Japan

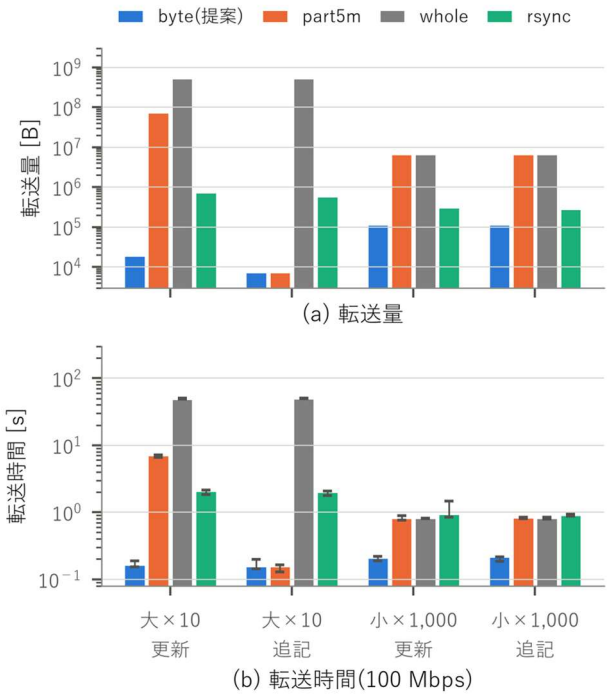

図 9　FS 反映：複数ファイルの転送量・時間

Figure 9　Multi-file transfer volume and time (FS).

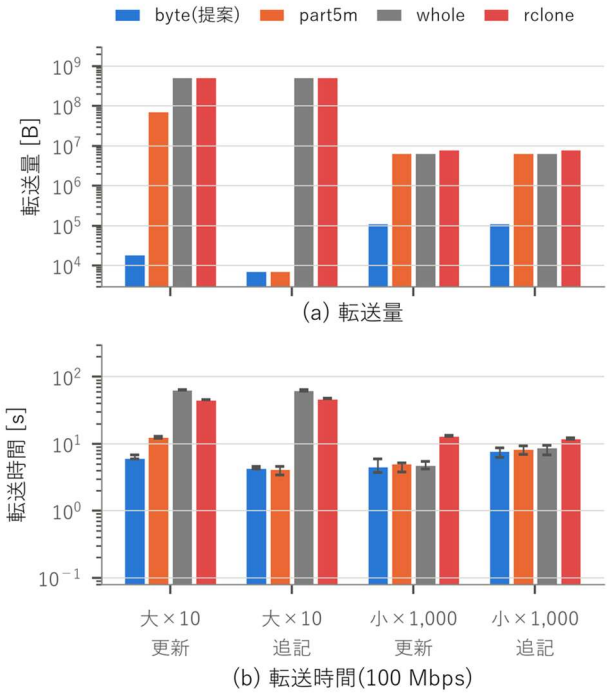

図 10　S3 反映：複数ファイルの転送量・時間

Figure 10　Multi-file transfer volume and time (S3).

できるが，転送先の既存データとの重複は転送先との照合なしには検出できない．試算の前提を次に示す．

- 転送対象：方式 1 が抽出した差分のデータ量 100GiB
- 重複率：転送対象のデータ量のうち転送先に同一チャンクが存在する割合を 34.3%と仮定する．根拠は，実運用ディレクトリ（885,065 ファイル，約 742.3GiB）を 4〜8KiB の可変長チャンクに分割した際の重複排除率（1−重複排除後のデータ量÷重複排除前のデータ量）であり，転送先の既存データとの重複も同程度と仮定した．
- チャンクと FP：平均チャンク長は上記の測定における平均値の約 5.5KiB とする．FP はチャンクの SHA-256 のハッシュ値（32B）である．
- 照合：1 要求あたり 10,000 個の FP をまとめて送り，応答は 1FP あたり 1 ビットの存在判定とする．重複チャンクの参照として差分集約オブジェクトに格納する FP も転送量に含める．
- 処理時間：上記の測定における 1 チャンクあたりの時間約 107µs（分割位置の決定 23.2µs，SHA-256 の計算 27.5µs，FP の索引との照合 56.2µs の合計）を用い，8 スレッドで並列に実行する．索引との照合は転送先で行うが，転送元と同じ処理性能を仮定する．
- 回線：帯域 100Mbps，往復遅延 11ms（4.1 節の評価環境と同じ）とする．

表 5 に試算結果を示す．チャンク数は 100GiB÷5.5KiB ≈ 1,907 万であり，FP の総量は 1,907 万×32B ≈ 0.57GiB（転送対象の約 0.57%），応答データは約 2.3MiB である．重複チャンクの参照として格納する FP は 1,907 万×0.343×32B ≈ 0.2GiB である．本体の転送量は 100×(1−0.343) = 65.7GiB となり，FP，応答および参照を加えた正味の転送量は約 66.5GiB，削減率は約 33.5%である．FP の送信による増加は，本体の削減に比べて十分小さい．

転送時間は，回線上の送信時間，チャンク分割・FP 計算・照合の処理時間，照合の往復時間の和とする．送信時間は，

重複排除なしで 100GiB÷100Mbps≈2.39 時間，ありで 66.5GiB÷100Mbps≈1.59 時間である．後者には，処理時間 1,907 万×107µs÷8 スレッド≈255 秒（転送先の索引照合は約 134 秒）と，往復時間 1,907 要求×11ms÷8 並列≈2.6 秒が加わる．合計は約 1.66 時間で，約 30.5%の短縮となる．

重複率を r とすると，FP と応答の量（約 0.57GiB）と処理時間・往復時間（計約 0.07 時間）は r によらず，FP 参照の量は約 0.57r GiB となるため，転送量は 100×（1−r）+0.57+0.57r GiB となり，r が約 0.57%を上回れば転送量は削減される．たとえば r=90%では，転送量を約 88.9%，転送時間を約 85.9%削減できる．処理時間・往復時間は帯域によらず一定であり，本条件では転送時間の約 4%にとどまるが，帯域が大きくなるほどその比率は増す．

4.3　方式 3 の予備評価

方式 3 の効果を見積もるため，類似チャンクを並べ替えて圧縮する処理のみを実装し，評価した．対象は，大規模ストレージシステムの共有ディレクトリ（84,886 ファイル，約 40.9GiB）である．評価の方法と指標を次に示す．

ⓒ2026 Information Processing Society of Japan

表 5　方式 2 の机上評価（帯域 100Mbps）
Table 5　Desk estimate of method 2 (100 Mbps).

項目	重複排除なし	重複排除あり
転送対象データ[GiB]	100.0	65.7
FP・応答・参照データ[GiB]	-	0.77
転送量の合計[GiB]	100.0	66.5
回線上の送信時間[h]	2.39	1.59
チャンク分割・FP 計算・照合[h]	-	0.07
照合の往復[h]	-	<0.01
転送時間の合計[h]	2.39	1.66

- チャンク分割 : 連続する 48B のデータを計算対象とするローリングハッシュにより, 4〜12KiB の可変長チャンクに分割する.
- スケッチ：分割の際に計算したローリングハッシュ値のうち，最小チャンク長（4KiB）以降で得られた値の大きいものから 4 つとする．チャンクの一部が更新されても，更新されていない領域から得た値は変わらないため，類似チャンク間でスケッチは一致しやすい.
- 類似判定：全チャンクをスケッチの 4 値をキーとしてソートし，隣接するチャンクの間でスケッチの 4 値がすべて一致する場合に類似と判定する．ただし，いずれかの値が 0 の場合は類似としない．類似の判定はスケッチの完全一致によるものであり，距離や類似度のしきい値は用いない.
- 並べ替え：類似と判定したチャンクをチャンク列の先頭にまとめ，スケッチの降順に並べる．スケッチが等しいチャンクはファイルごとにまとめ，同一ファイル内はオフセット順とする．類似と判定されなかったチャンクは，元の順序のままその後ろに置く．並べ替えは，1GiB の圧縮単位ごとではなく，データ全体を対象として行う.
- 圧縮：チャンク列を先頭から 1GiB ずつ LZ4 で圧縮する．並べ替えなしの条件も同じ 1GiB ずつ LZ4 で圧縮しており, 両条件の違いは並べ替えの有無のみである.
- 圧縮率：1−（圧縮後のサイズ÷圧縮前のサイズ）.

表 6 に結果を示す．並べ替えを行わない場合の圧縮率 58.7%に対し，類似チャンクの並べ替えを行った場合は 77.2%であった．LZ4 の参照可能な範囲（辞書）は 64KiB であり，チャンク長（4〜12KiB）の数個分にすぎない．並べ替えなしでは，類似するチャンクがデータ中で離れて存在すると，この範囲に収まらず冗長性が検出されない．類似チャンクを近接配置すると, これらが同一の範囲に収まり，冗長性が検出されやすくなる．表 6 の差は，これによるものと考えられる.

本評価の限界は次のとおりである. 3.5 節の適用判断は行わず，類似と判定したすべてのチャンクを並べ替えた．並べ替えを元に戻すための構成情報は圧縮後のサイズに含めていないが，チャンクあたり 4B とすると約 20MiB（並べ替えありの圧縮後の約 0.2%）にとどまる．また，データ全体（約 40.9GiB）を対象としたため，数 KiB〜数 MiB の差分では類似チャンクが少なく，効果が小さくなる可能性がある．並べ替えと圧縮に要する時間は測定していない.

表 6　方式 3 の予備評価（圧縮率）
Table 6　Method 3 compression results.

対象: 84,886 ファイル，約 40.9GiB．4〜12KiB 可変長チャンク，LZ4, 1GiB 単位.

条件	圧縮後[GiB]	圧縮率
並べ替えなし	16.9	58.7%
類似チャンクの並べ替えあり	9.3	77.2%

5. 議論と限界

方式 1 は，小さな更新，大きなファイル，帯域制約が重なる場合に有効である．一方, 更新率が高いと whole・rsync との差が縮小し，帯域制限がないと転送量削減が時間短縮に直結しない．小さなファイルが多い場合は，rsync が交換するチェックサムの量も少ないため転送量の差は縮まるが, 更新範囲を 1 回の要求にまとめて送ることで，転送時間では rsync に対する優位を保った．本評価の限界を以下に示す.

- 評価環境 : 単一の送受信ペアで, WAN の条件を同一サブネット上で netem により模擬した（往復遅延約 11ms, MTU1500 の 1 条件）. 実 WAN の輻輳, パケット損失, 遅延の変動は含まず，他の通信の影響を受けうる共有基盤上での測定である．また，TSO/GSO/GRO が有効なため，4.1.4 項の IP 層以上の増加分は下限値とみなすべきである.
- ワークロード：単一ファイルの更新は，更新位置を一様ランダムとする 4KiB 単位に整列した人工的な書込みであり，複数ファイルでも 1KiB の更新・追記（表 4）のみを用いた. ファイルサイズは約 100MiB と 64KiB のみである．更新が局所に集中する実アプリケーションでは，part5m の削減効果は本稿の値より大きくなると考えられる.
- 資源コスト：本評価は転送量と転送時間を対象とし，送受信両側の CPU 時間，読み込みバイト数，メモリ使用量は測定していない．書込み時に更新範囲を記録するコストも未測定であり，FUSE によるユーザ空間経由の影響を含む．したがって，rsync に対する読み込み I/O と CPU の削減，および書込み性能への影響は，本評価からは定量的に結論づけられない.
- S3 反映 : 単一ノードの MinIO で評価しており, クラウド事業者の S3 における遅延や，要求ごとの課金

（100MiB のファイルで約 21 要求）は含まない. また，Multipart Upload のパート数の上限（10,000）により，5MiB の格子のままでは約 48.8GiB を超えるオブジェクトに適用できない.

- 方式の併用 : 方式 2・3 は机上評価または部分的な実装による単独の評価であり，方式 1 と併用したときの効果，適用順序，損益分岐点は評価していない.

6. おわりに

本稿では，拠点間ファイル転送の転送量を削減するため，実際の更新範囲，転送先との同一性，転送対象間の類似性を利用する細粒度差分転送方式を提案した．このうちバイト差分転送を試作して評価した結果，小さな更新（102.5MiB のファイルの 0.01%を更新）で rsync に比べて転送量を約 92%, 帯域 100Mbps での転送時間を約 87%削減した．この結果は，転送時の内容照合に代えて，書込み時に記録した更新範囲から転送対象を決めることが，小さな更新と帯域制約が重なる場合に有効であることを示す．一方, S3 互換オブジェクトストレージへの反映では, WAN 転送量を削減しても転送先内部の再構成処理が残るため，適用判断では転送量と反映コストを分けて考える必要がある.

今後の課題として，3 つの方式の併用効果の評価，転送先の既存データとの重複排除の実測，更新範囲の記録を含む資源コストの測定，チャンク重複排除型バックアップツールとの比較，および実 WAN 環境での評価が挙げられる.

参考文献

[1]Tridgell, A. and Mackerras, P.: The rsync Algorithm, Technical Report TR-CS-96-05, Australian National University (1996).
[2]Rclone: Rclone Documentation（オンライン），入手先〈https://rclone.org/docs/〉（参照 2026-10-26）.
[3]Morris, R. and Cudhea, P.: ZFS Async Replication Enhancements, SNIA Storage Developer Conference (SDC 2015)（オンライン），入手先〈https://snia.org/sites/default/files/SDC15_presentations/file_sys/Morris_Cudhea_ZFS_Async_Replication_Enhancements-revision.pdf〉（参照 2026-10-26）.
[4]btrfs-send(8): Linux Manual Page（オンライン），入手先〈https://www.man7.org/linux/man-pages/man8/btrfs-send.8.html〉（参照 2026-10-26）.
[5]Muthitacharoen, A., Chen, B. and Mazières, D.: A Low-Bandwidth Network File System, Proc. 18th ACM Symposium on Operating Systems Principles (SOSP 2001), pp. 174–187 (2001).
[6]Shilane, P., Huang, M., Wallace, G. and Hsu, W.: WAN-Optimized Replication of Backup Datasets Using Stream-Informed Delta Compression, Proc. 10th USENIX Conference on File and Storage Technologies (FAST 2012) (2012).
[7]Douglis, F. and Iyengar, A.: Application-Specific Delta-Encoding via Resemblance Detection, Proc. 2003 USENIX Annual Technical Conference (USENIX ATC 2003), pp. 113–126 (2003).
[8]Kulkarni, P., Douglis, F., LaVoie, J. and Tracey, J. M.: Redundancy Elimination Within Large Collections of Files, Proc. 2004 USENIX Annual Technical Conference (USENIX ATC 2004), pp. 59–72 (2004).
[9]Zhang, Y., Xia, W., Feng, D., Jiang, H., Hua, Y. and Wang, Q.: Finesse: Fine-Grained Feature Locality Based Fast Resemblance Detection for Post-Deduplication Delta Compression, Proc. 17th USENIX Conference on File and Storage Technologies (FAST 2019), pp. 121–128 (2019).
[10]Restic: Restic Documentation（オンライン），入手先〈https://restic.readthedocs.io/〉（参照 2026-10-26）.
[11]Kopia: Architecture（オンライン），入手先〈https://kopia.io/docs/advanced/architecture/〉（参照 2026-10-26）.
[12]Kopia: Repositories（オンライン），入手先〈https://kopia.io/docs/repositories/〉（参照 2026-10-26）.
[13]Duplicati: Backup Size Parameters（オンライン），入手先〈https://docs.duplicati.com/technical-details/understanding-backup/backup-size-parameters〉（参照 2026-10-26）.
[14]Patterson, R. H., Manley, S., Federwisch, M., Hitz, D., Kleiman, S. and Owara, S.: SnapMirror: File-System-Based Asynchronous Mirroring for Disaster Recovery, Proc. 1st USENIX Conference on File and Storage Technologies (FAST 2002), pp. 117–129 (2002).
[15]Broadcom: Changed Block Tracking on Virtual Disks, Virtual Disk Development Kit Programming Guide, vSphere 9.1（オンライン），入手先〈https://techdocs.broadcom.com/us/en/vmware-cis/vsphere/vsphere-sdks-tools/9-1/virtual-disk-development-kit-programming-guide/backing-up-virtual-disks-in-vsphere/low-level-backup-procedures/changed-block-tracking-on-virtual-disks.html〉（参照 2026-10-26）.

ⓒ2026 Information Processing Society of Japan