Vol.2018-HPC-167 No.14
2018/12/17

AI・ビッグデータ処理におけるオブジェクトストレージを用いたデータステージングの評価

谷村 勇輔[1,a)]　遊佐 佳一[1]　高野 了成[1]　浜西 貴宏[1]

概要：
大規模な HPC システムにおいて，計算ノード内のメモリやディスクにファイルを保持して高速に処理を行うためには，計算ノード群とシステム全体から利用できる共有ストレージとの間のデータステージングが必要である．しかし，近年は共有ストレージに直接アクセスするのではなく，I/O のバッファリングやフォワーディング手法と組み合わせたステージング技術の研究が進んでいる．また，共有ストレージの大規模化傾向により，POSIX に準拠したファイルシステムでなく，Amazon S3 互換のインタフェースを持つオブジェクトストレージに注目が集まっている．本稿では，AI・ビッグデータ処理，HPC が融合した計算環境において，S3 互換のオブジェクトストレージの可能性と効果的に利用するための手法を明らかにすることを目指し，S3 を用いたデータステージングの性能や課題について調査した結果を報告する．

1. はじめに

近年，高性能計算（HPC）分野において，仮想化やコンテナ技術等を利用して，利用者が自由にカスタマイズできる環境を提供するクラウド型運用の導入が進む一方，ビッグデータ解析分野においては大規模化の必要性から HPC 技術の導入が進んでいる．そうした両者の融合傾向は，人工知能（AI）処理の需要の高まりにより，HPC での採用が多かった GPU 等のアクセラレータの活用が産業分野に広まったことで，さらに促進されている．

そのような状況を踏まえて HPC ストレージに目を向けると，HPC 向けに設計された並列ファイルシステムは特に大量の小さなファイル群へのアクセス性能が十分にスケールせず，大規模化に対するコストが見合わなくなってきている．ストレージクラスメモリや高速 SSD をローカルストレージとして活用する研究 [1] や，バーストバッファ [2] や I/O フォワーディング技術 [3, 4] が研究されるようになり，POSIX 準拠のファイルアクセス・インタフェースを計算ノード全体に展開する必然性が低下しつつある．セキュリティに関しては，機微な情報を扱う応用においてパーミッションレベルのアクセス制御が不十分と認識されつつあり，他で運用されるシステムと連携するためにも，自システム内の計算ノードへのデータ提供との一貫性を保ちつつ，他システムとのデータインポート／エクスポートを安全かつ容易に行える仕組みの検討が行われる等，認証機構の再整備が必要となっている．

一方，ビッグデータ解析やクラウドにおいては，Amazon S3（Simple Storage Service）[5] に代表されるオブジェクトストレージが利用されるようになり，大容量ストレージを安価に構築でき，かつ全体性能がスケールすること，アプリケーション毎の暗号化等も組み込みやすいことから注目を集めている．ただし，オブジェクトストレージは次世代のストレージ基盤として有望ではあるが，性能を引き出すための手間や標準インタフェースが定まっていなこと，他システムと連携するための認証機構が不十分であることから，バックアップやアーカイブ用途が中心であり，システム全体の主ストレージシステムとして積極的に用いられるに至っていない．当然ながら HPC 用途のシステムでも採用が進んでいない．

そこで本研究では，AI 処理を対象に HPC とビッグデータ解析が融合した次世代のデータプラットフォームにおいて，S3 互換のオブジェクトストレージの性能を最大限に引き出して従来的な並列ファイルシステムを上回る高い性能を達成するとともに，使い勝手においても並列ファイルシステムと遜色ないレベルを実現できるかという問いへの答えを明らかにすることを目的とし，その過程において，オブジェクトストレージを効果的に利用する手法も明らかにすることを目指す．それに向けて，本稿では実運用されている産総研の AI クラウドシステムと S3 互換ストレージを用いて，データステージングにおける性能特性を理解する

[1] 国立研究開発法人 産業技術総合研究所
[a)] yusuke.tanimura@aist.go.jp

表 1 S3 互換のソフトウェアの例

Client tools and services	S3 tools, S3 browser, Jungle Disk, FOBAS, Dropbox, etc.
Storage systems	RADOS Gateway, Swift, Scality, Riak CS, Cloudian, etc.
Cloud management platforms which adopt/integrate the above systems and tools	OpenStack, CloudStack, etc.

ための評価を行い，前記プラットフォームにおける既存オブジェクトストレージの課題を整理した．具体的には，用いる S3 クライアント・ソフトウェア，同時アクセス数，オブブジェクト・サイズ，暗号化の有無等の要素による性能の違い，既存の並列ファイルシステムを用いた場合との性能の違いを実験を通して示した．

2. オブジェクトストレージ

2.1 概念

オブジェクトストレージは，大量の非構造データを低コストで，かつ容易に扱うことに主眼を置いて設計されたストレージであり、ビッグデータ解析やクラウド等での利用を中心に発展してきた．オブジェクトストレージでは，フラットな名前空間において各データはオブジェクト（Object）として扱われる．1 度書き込まれたデータの改変を許さない仕様により，従来的なファイルシステムよりずっと簡単な仕組みの実装で済むことから，低コスト，高いスケーラビリティ，高い可用性，広域ネットワークでのデータの一貫性保証等が可能になっている．また，ファイルサイズ上限の撤廃や Erasure Coding 等を用いたデータの自動修復機能が組み込まれることも多い．最近では，マルチテナントでの運用に柔軟にかつ安全に対応するため，高度なメタデータ管理機能の実装も検討されている．

多くのオブジェクトストレージの実装において，アクセスインタフェースとして REST API が提供されている．SNIA の CDMI [6] 等，API の標準化を図る動きもあるが，現在広く利用されているのは次節で述べる Amazon S3 互換のインタフェースである．

2.2 Amazon S3 とその互換インタフェース

Amazon S3 はインターネット上のどこからでもアクセス可能であり，Web Services に基づくアクセスインタフェースを提供する Amazon のオンライン・ストレージサービスであり，クラウドにおけるオブジェクトストレージの先駆をなした存在である [5]．そのストレージ規模は年々拡大しており，2012 年には 1 兆個以上のオブジェクトの格納に至っている [7]．Amazon S3 の内部実装は公開されていないが，S3 の成功により，**表 1** に例示するように，S3 が提供する REST API の互換インタフェースを持つクライント，ストレージシステム，ストレージサービスが多数開発されてきた．その結果，S3 互換インタフェースがオブジェクトストレージの実質的な標準インタフェースとして広く認識されるに至っている．S3 の REST API では，GET や PUT，POST 等の HTTP メソッドがそれぞれオブジェクトのダウンロードやアップロード等のオブジェクト操作に対応する仕様となっている．

3. オブジェクトストレージを用いたデータステージングの検討

3.1 機能要件

本研究では，まず既存の S3 互換ソフトウエアの問題を整理して必要な改良を施すため，AI・ビッグデータ解析を HPC システム上で実行する想定のもと，その際に行うデータステージングの機能要件を洗い出す．

HPC クラウド（ASGC）[8] や AIST AI Cloud（AAIC）[9] におけるこれまでの我々の運用経験により，AI の研究においては，比較的小さなサイズのファイルを大量に処理することが多いことが分かっている．ビッグデータ解析においても Volume（データサイズ）よりも Velocity や Variety が主のアプリケーションが存在する．そのような場合，共有ストレージ上にファイルを置くと，共有ストレージへの I/O リクエスト数が過剰になり性能低下を招く恐れがあるため，解析前後に共有ストレージと計算ノード間のデータステージングを行うのが望ましい．しかし，複数のノードへのデータステージングは，データの分散方法をアプリケーションレベルで指示する必要があったり，並列ステージングを自分でプログラミングする必要があったりする．さらに，機微な情報を扱う可能性があるため，共有計算環境において安全にデータを扱えるよう，求められるセキュリティレベルと性能を考慮して適切なデータ保護を実現する仕組みが必要である．

まとめると，AI・ビッグデータ解析向けのデータステージングでは次の要件が重要といえる．

- 比較的小さなサイズのファイルを大量にステージングできること．また，それが十分に高速であること．
- 負荷分散を考慮し，指定された数の複数ノードへのステージングができること．
- 機微なデータを扱うための安全なデータ転送，および保存ができること．

3.2 クライアントの選択

表 1 に示したように，現在多くの S3 互換クライアントが開発されているが，前節の要件を満たすためには高性能で必要機能，あるいは必要機能を実装可能な下地が備わっ

表 2 S3 互換クライアントの比較

	Support of multipart data transfer	Caching support	Encryption support
s3fs	Enabled in upload when file size > 20MB. (Default: part size = 10MB, parallel count = 5)	In-memory metadata caching and on-disk data caching are supported.	Server-side (SSE-C, SSE-KMS and SSE-S3)
Goofys	Enabled in upload when file size > 5MB with auto-tuned parameters.	Some metadata and data can be cached in memory.	Server-side (SSE-KMS and SSE-S3)
S3QL	No need. File contents are splitted into small blocks and each is stored as an object. (Default: block size = 10MB)	Metadata is always cached as a local SQLite database. Block-based data caching is enabled.	Client-side and server-side (SSE-S3)
s3cmd	Available. Users need to specify the options. (Default: part size = 15MB, no parallel options)	None	Client-side and server-side (SSE-KMS)

た S3 クライアントを選択することが重要である．本稿では次に述べる 4 つのソフトウェアに着目することとし，それぞれの特徴や違いを調査した後，予備性能評価（4 節）を行う．なお，本研究では S3 に焦点を当てるが，これらのツールは S3 以外のオブジェクトストレージへのアクセスもサポートしている．

s3fs [10]，Goofys [11]，S3QL [12] は，いずれも FUSE（Filesystem in Userspace）を利用して S3 のインタフェースを隠蔽し，従来的な POSIX に基づいたファイルシステムのインタフェースをユーザに提供するツールである．つまり，ユーザ権限で S3 のバケット（Bucket）をマウントして，ローカルのファイルのようにオブジェクトにアクセスすることを可能にする．s3fs や S3QL は多くの POSIX 操作をサポートしているのに対し，Goofys は POSIX 操作よりも性能を重視した設計となっている．また，s3fs や Goofys を通して格納したファイルはバケット内でオブジェクトとして保持されるため，他の S3 クライアントからアクセス可能であるが，S3QL はバケット内に独自フォーマットでデータを格納するため，S3QL でしか取り出すことができない．s3fs と S3QL は Python，Goofys は Go で書かれている．

s3cmd [13] は，S3 互換のオブジェクトストレージに対してコマンドラインでアクセスを可能にする Python で書かれたツールである．S3 の各 API に対して比較的単純なマッピングとなっており，s3cmd 内部において転送パラメータのチューニングやキャッシング機構等は実装されていない．

表 2 に各ソフトウェアのマルチパートデータ転送，キャッシング，暗号化に関する提供機能を挙げる．マルチパートデータ転送は大きなサイズのファイルを転送する際にファイルを分割し，かつ並列に転送する機能である．本稿での評価結果には含めていないが，S3 を利用する上で重要な機能の 1 つである．キャッシングは単純なデータステージングだけを考えた場合は不要であるが，ステージング前後に多少のアクセスを伴う場合は重要になる．暗号化には，クライアント側での暗号化（CSE: Client-side encryption）とストレージサーバ側での暗号化（SSE: Server-side encryption）

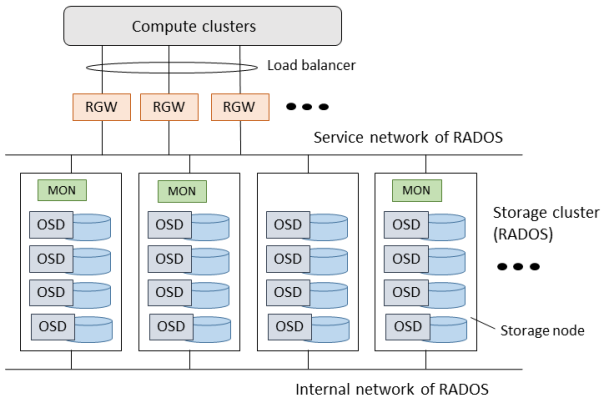


図 1 RADOS Gateway を用いたオブジェクトストレージの構成

がある．S3 において SSE には暗号鍵の管理方式にいくつかの選択肢が存在し，API で指定可能となっている．

3.3 オブジェクトストレージの構成例

HPC システムにおけるデータステージングを考えた時，性能を考慮すれば当然，クラウドのサービスを利用するのではなく，HPC システムに近い場所にストレージを構築する必要がある．オンプレミスで用いることのできる S3 互換のオブジェクトストレージとして，本研究では Ceph の RADOS Gateway（RGW） [14] に着目する．RGW は RADOS（Reliable Autonomic Distributed Object Store） [15] のフロントエンドとして動作し，S3 互換のアクセスインタフェースを提供するソフトウェアである．**図 1** に示すように，RADOS はストレージサーバ上のディスクへのアクセスを仲介する OSD デーモンと，OSD デーモンの状態監視を行う MON デーモンからなるストレージクラスタのソフトウェアである．RGW は 1 つのストレージクラスタ（RADOS）において複数稼働させることができ，同時アクセス数や負荷に応じて増やすことが可能である．ただし，その際はロードバランサ等を合わせて用意することになる．これまでの我々の RGW の運用や評価を通して，RGW が十分な性能を提供できることは分かっており，本稿の 4 節の評価実験では RGW で構築したオブジェクトストレージを用いる．

3.4 並列ファイルシステムを用いた場合との違い

単純なデータステージングのみを考えた場合，ユーザにとって，共有ストレージのアクセスインタフェースがオブジェクトストレージであるか並列ファイルシステムであるかに大きな違いはない．また，ADIOS [16] や MPI-IO [17] 等の HPC 向けの上位 I/O ライブラリを介してデータステージングを行う場合は，下位のストレージのインタフェースはユーザから隠蔽されることになる．ただし，現在はそうした既存ライブラリの多くが S3 互換インタフェースをサポートしていない問題がある．

並列ファイルシステムにおいて，ストライピングは 1 つのファイルアクセスを高速に行う手法であり，アクセス単位でストレージサーバあるいはデバイスへの I/O 要求のマッピングが指定可能である．オブジェクトストレージのマルチパートデータ転送は，データを分割して並列に転送する点においてはストライピングと同じであるが，ロードバランサやフロントエンドサービス（RGW 等）を介してストレージサーバやデバイスに I/O が負荷分散される設計となっており，1 つのアクセスを高速化するために I/O 要求を各デバイスにマッピングできない仕様である．

4. オブジェクトストレージを用いたデータステージングの予備性能評価

4.1 性能評価の方法

データステージングにおける S3 互換アクセスの性能特性を整理し，改善に向けた指針を定めるため，以下の項目について性能評価および比較を行うこととした．

- S3 クライアント・ソフトウェアによる性能の違い
- 複数オブジェクトの同時転送における性能
- オブジェクト・サイズが性能に与える影響
- 暗号化の有無による性能の違い
- 既存の並列ファイルシステムを用いた場合との性能差

性能測定においては，単純なデータステージングを想定することとし，各 S3 クライアント・ソフトウェアのキャッシュや暗号化，圧縮に関するオプションは全て無効とし，データ転送時の並列数を除き，オプションはデフォルト値を用いた．そして，1MB のサイズのファイルを 1000 個用意し，それらをステージイン（計算ノードへのダウンロード）あるいはステージアウト（計算ノードからのアップロード）するのに要した時間（walltime）を測定した．ただし，4.4 節や 4.6 節の実験ではファイルサイズやファイル数を変更して測定を行った．データ転送の際は，いくつかのファイルを同時に転送する方法も試し，本稿ではこれを並列数と呼ぶ．また，転送データは計算ノード側ではメモリベースで構成した/tmp に格納し，計算ノード側の I/O がボトルネックとならないようにした．

表 3 AAIC インタラクティブ・ノードのスペック

CPU	Intel Xeon E5-2630L v4 × 2
Memory	128GB
Network for GPFS	Infiniband EDR, 100Gbps
Network for S3	Ethernet, 1Gbbps
OS	CentOS 7.4
Singularity	v2.5.1

表 4 Ceph RADOS Gateway を構成する機器のスペック

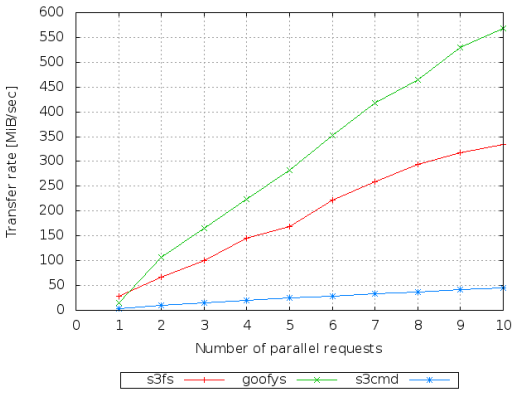
Frontend node	Supermicro SuperServer ×2
CPU	Intel Xeon E3-1230v3 (4 cores, 3.3GHz) ×2
Memory	32GB (DDR3-1600)
Disk	Intel SSD 530 120GB
10GbE adaptor	Intel X520-DA2
OS	Ubuntu 12.04.5
Storage node	Supermicro custom server ×10
CPU	Intel Xeon E5-2630Lv2 (6 cores, 2.4GHz) ×2
Memory	32GB (DDR3-1600)
System disk	Intel SSD 530 120GB or 240GB
Journal disk	Intel SSD 530 480GB ×2
Data disk	Seagate ST2000NM0023 ×5 (SAS 6G connect, 7200RPM, 2TB) HGST HUH721010AL4200 ×3 (SAS 6G connect, 7200RPM, 10TB)
10GbE adaptor	Intel X520-DA2
OS	Ubuntu 12.04.5
10GbE switch	Arista Networks 7124SX ×2

4.2 評価環境

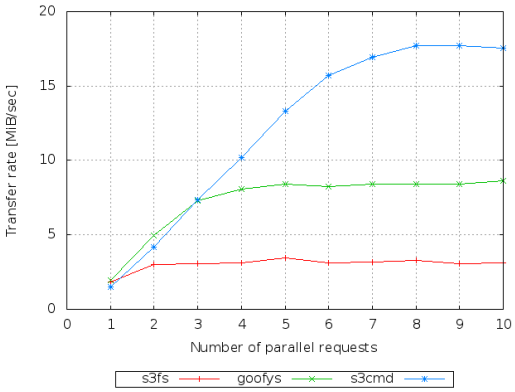
性能評価は，AAIC のインタラクティブ・ノードと RGW を用いて構築したオブジェクトストレージを用いて，その間でデータステージングを実行して時間を測定する方法とした．それぞれの機器のスペックを**表 3** と**表 4** に示す．

評価に用いたオブジェクトストレージは Ceph version 0.94.9 の RADOS および RGW を用いて構築した．RADOS は 80 個の OSD からなり，合計の物理容量は 400TB であるが，3 コピーのレプリカ設定を採用しているため，サービス向けの実効容量は約 133TB である．オブジェクトストレージ本体は 2 系統のネットワークを持ち，レプリカ作成等の RADOS 内通信と S3 を提供するフロントエンドノードが接続するサービスネットワークを分離している．フロントエンドノードは Active-Standby，つまり 1 ノードのみの構成である．フロントエンドノードのネットワークは 10GbE であり，AAIC のインタラクティブ・ノードまで 10Gbps で接続されている．

オブジェクトストレージとの比較として用いたファイルシステムは，AAIC の共有ストレージを構成する Spectrum Scale（GPFS) version 4.2.1 である．AAIC の計算ノードからは，4 つのファイルシステムノードにアクセス可能であり，データ転送に利用する接続は IB-EDR である．GPFS

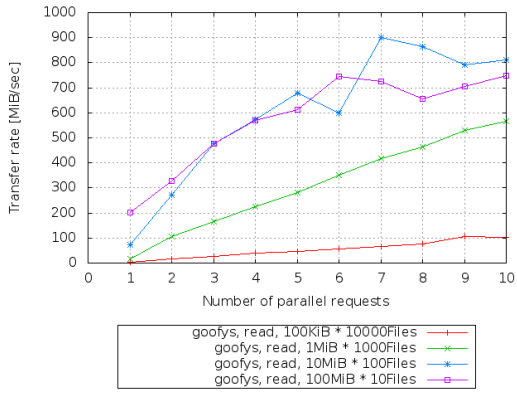


a) ダウンロード性能（1MB × 1000 ファイル）

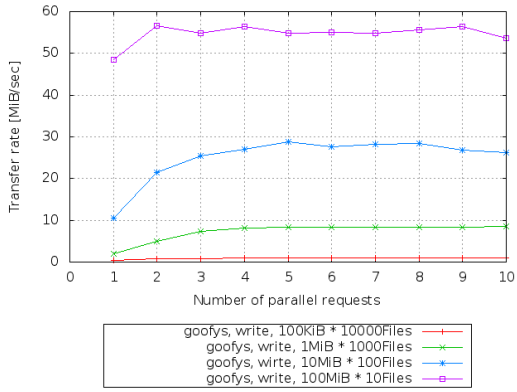


b) アップロード性能（1MB × 1000 ファイル）

図 2 S3 クライアント・ソフトウェアの比較



a) ダウンロード性能



b) アップロード性能

図 3 オブジェクトサイズによる影響

の内部は 72 個の NSD からなり，合計の実効容量は約 4PB である．

AAIC では，AI 処理向けにユーザが環境をカスタマイズして処理を実行する仕組みとして Singularity コンテナの環境を提供し，その利用を推奨している．このため，本性能評価でも全てのデータステージング操作および測定を Singularity コンテナ内で行った．ただし，本性能評価の実施時点において，AAIC のコンテナ内で FUSE を用いたマウントができない制約があるため，FUSE を用いたマウントはコンテナ外で実行し，そのマウントディレクトリをコンテナ起動時に bind マウントする方法を用いた．

4.3 S3 クライアントの比較結果

図 2 に s3fs，Goofys，s3cmd の性能を比較した結果を示す．ダウンロードは FUSE ベースの S3 クライアントが良い性能を示し，Goofys が最も良い結果だったのに対し，アップロードでは並列数が 4 以上の場合は s3cmd が最も良い性能を示した．アップロードはダウンロードに比べると転送速度が遅く，並列数を増やしても十分な速度向上を得られなかった．

4.4 オブジェクトサイズによる性能への影響

次に Goofys に着目し，合計転送サイズは一定のまま，ステージング対象のファイルサイズ（本評価ではオブジェクトサイズと等しい）を変えた場合の性能を測定した．**図 3** に示すように，ダウンロードとアップロードの両方において，ファイルサイズが一定以上大きくなければ十分な性能を得ることができず，ファイルサイズが小さい時の性能は非常に低い結果となった．

4.5 暗号化処理による性能への影響

図 4 は CSE 利用時のダウンロード（Read）およびアップロード（Write）の性能測定の結果である．S3QL の CSE は，PyCrypto を用いて AES-256bit（CTR モード）の暗号化を行っており，s3cmd の CSE は，GPG を用いて AES-256bit（CFB モード）の暗号化を行っている．本結果では，アップロードに関しては s3cmd が明らかに高速であり，ダウンドードに関しては並列数によって S3QL と s3cmd の優劣に違いがあった．ただし，先に示した暗号化処理を行わない場合の転送性能に比べると，s3cmd においては 2~2.5 倍の差が見られた．より高速な他の S3 クライアントと比べると，暗号化の有無による性能差はさらに大きいともいえる．

4.6 GPFS との比較結果

図 5 は，S3 互換オブジェクトストレージを用いたデータステージングと GPFS を用いたデータステージングの性能を比較した結果である．GPFS を用いたデータ転送は標

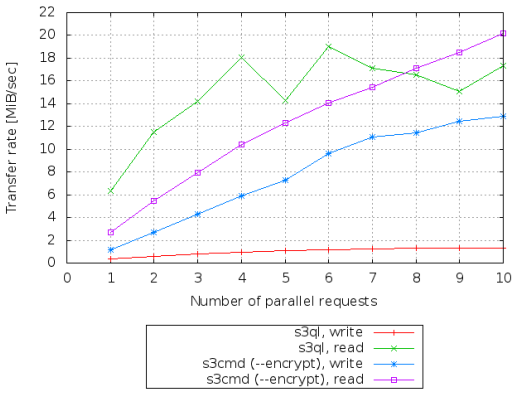


図 4 CSE 利用時の性能比較

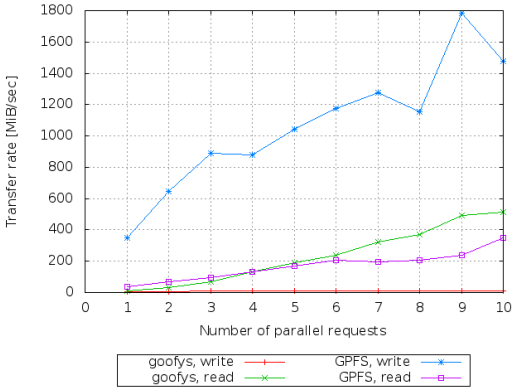


図 5 S3 と GPFS との性能比較

準的な cp コマンドを用いて行った．この結果より，アップロードは明らかに GPFS が高い性能を提供できたが，ダウンドードは並列数が増えた場合には S3（Goofys）が高い性能を示したことが分かる．

4.7 議論

本性能評価により，S3 を用いたデータステージングにおいて十分な性能を得るためにはオブジェクトサイズを大きくし，並列転送を行う必要があることが分かる．S3 はマルチパートデータ転送を提供しているが，これは大きなオブジェクトを分割して転送するためのものであり，小さなオブジェクトを大量に転送する用途には適用できない．このため，小さなオブジェクトをまとめて大きなオブジェクトにして転送する仕組みが必要であると考える．さらに，適切なオブジェクトサイズや並列数は，S3 クライアント・ソフトウェアや実行環境等に依存するため，それらを自動的にチューニングできる仕組みも合わせて必要である．

暗号化を行うことによる性能低下は非常に大きく，改善や使い方の工夫が必要である．並列処理による高速化のほか，今回は試していないが，SSE との性能を比較した上で CSE との適切な使い分けが必要だと考える．

データのステージインにおいては S3 は GPFS と同等以上の性能を示し，GPFS との置き換えの可能性を示したといえる．しかし，S3 はステージアウトが遅いため，現状では Read-only な場面での活用を考えざるを得ない．S3 のアップロード性能の高速化については，クライアントおよびストレージの双方でのキャッシングの活用等を考えていく必要があると考える．

5. 関連研究

大規模計算の実行に必要な I/O アクセスの高速化の研究としては，ファイルシステムのフロントにキャッシュ層を設けて I/O の削減や効率化を行ったり [2]，POSIX の制約を緩めたファイルアクセスを提案評価したり [18] する取り組みがある．脱 POSIX の方向は，オブジェクトストレージの活用を試みる本研究と同じであるが，本研究ではアクセスインタフェースに関する検討は研究対象とせず，オブジェクトストレージの S3 互換インタフェースを用いて，最大限に性能を引き出す取り組みに焦点を当てる．また，キャッシュ・アルゴリズムの開発も対象とせず，既存のキャッシュ層や I/O フォワーディング技術とオブジェクトストレージのインタフェースとの適合を試みていく．

S3 互換ストレージの研究としては，S3 API の上位に FUSE 等を用いて POSIX アクセスを効率的に実現する取り組みがある [10–12]．これらは従来的なアプリケーションからオブジェクトストレージを利用可能にすることを目的としているため，オブジェクトストレージの本来の特徴が失われている可能性が高い．本研究とは逆行する取り組みである．ただし，本稿ではデータ転送のツールとしてこれらを評価した．

6. まとめと今後の課題

本稿では，AI 処理とビッグデータ解析，HPC が融合した次世代のデータプラットフォームにおいてオブジェクトストレージの可能性と効果的に利用する手法を明らかにすることを目的とし，その事前調査として，実運用されている HPC 型のクラウドシステムにおいて，S3 を用いたデータステージングの性能特性を評価実験を通して示した．

今後は今回明確になった課題を解決するため，既存の S3 クライアントの上位にデータステージング・ツールを作成して転送処理の最適化実装を進めていく．特に，小さなオブジェクト群のパッキング手法や自動並列化を行うとともに，マルチノードに対応したステージングやメモリベースで構成された I/O ノードとの連携も検討していきたい．

謝辞 本研究の一部は，NEDO の委託業務「次世代ロボット中核技術開発プロジェクト」の支援を受けて実施した．また，本研究の一部は，産総研・東工大 実社会ビッグデータ活用 オープンイノベーションラボラトリ（RWBC-OIL）の活動として実施した。

参考文献

[1] Jackson, A., Weiland, M., Parsons, M. and Homoelle, B.: Architectures for High Performance Computing and Data Systems using Byte-Addressable Persistent Memory, *CoRR*, Vol. abs/1805.10041 (2018).
[2] Bhimji, W., Bard, D., Burleigh, K., Daley, C., Farrell, S., Fasel, M., Friesen, B., Gerhardt, L., Liu, J., Nugent, P., Paul, D., Porter, J. and Tsulaia, V.: Extreme I/O on HPC for HEP using the Burst Buffer at NERSC, *Journal of Physics: Conference Series*, Vol. 898, No. 8, p. 082015 (2017).
[3] Ali, N., Carns, P., Iskra, K., Kimpe, D., Lang, S., Latham, R., Ross, R., Ward, L. and Sadayappan, P.: Scalable I/O Forwarding Framework for High-Performance Computing Systems, *Proceedings of the 2009 IEEE International Conference on Cluster Computing*, pp. 1–10 (2009).
[4] Vishwanath, V., Hereld, M., Iskra, K., Kimpe, D., Morozov, V., Papka, M. E., Ross, R. and Yoshii, K.: Accelerating I/O Forwarding in IBM Blue Gene/P Systems, *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*, pp. 1–10 (2010).
[5] Amazon S3: http://aws.amazon.com/s3/.
[6] SNIA: CDMI (Cloud Data Management Interface), http://www.snia.org/cdmi.
[7] Amazon S3 – The First Trillion Objects: https://aws.amazon.com/jp/blogs/aws/amazon-s3-the-first-trillion-objects/.
[8] Takano, R., Tanimura, Y., Takefusa, A., Hirofuchi, T. and Tanaka, Y.: AIST Super Green Cloud: Lessons Learned from the Operation and the Performance Evaluation of HPC Cloud, *International Symposium on Grids and Clouds* (2015).
[9] AIST AI Cloud: https://www.top500.org/system/179091.
[10] s3fs: https://github.com/s3fs-fuse/s3fs-fuse.
[11] Goofys: https://github.com/kahing/goofys.
[12] S3QL: https://bitbucket.org/nikratio/s3ql/.
[13] s3cmd: https://s3tools.org/s3cmd.
[14] RADOS Gateway: http://ceph.com/docs/master/radosgw/.
[15] Weil, S. A., Leung, A. W., Brandt, S. A. and Maltzahn, C.: RAODS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters, *Proceedings of the 2nd International Workshop on Petascale Data Storage*, pp. 35–44 (2007).
[16] Liu, Q., Logan, J., Tian, Y., Abbasi, H., Podhorszki, N., Choi, J. Y., Klasky, S., Tchoua, R., Lofstead, J., Oldfield, R., Parashar, M., Samatova, N., Schwan, K., Shoshani, A., Wolf, M., Wu, K. and Yu, W.: Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks, *Concurrency and Computation: Practice and Experience*, Vol. 26, No. 7, pp. 1453–1473 (2013).
[17] Thakur, R., Ross, R., Lusk, E., Gropp, W. and Latham, R.: Users Guide for ROMIO: A High-Performance, Portable MPI-IO Implementation, ANL/MCS-TM-234 (1998).
[18] Welch, B.: POSIX I/O High Performance Computing Extensions, *The 4th USENIX Conference on File and Storage Technologies* (2005).