Vol.2018-HPC-167 No.15
2018/12/17

高性能計算に向けた 分散オブジェクトストレージ Ceph の性能評価

高橋 宗史[1,a)]　建部 修見[2]

概要：本研究では，分散オブジェクトストレージ Ceph に対して，S3 インターフェイスを提供する RADOW Gateway 用いた場合の，各種ストレージ・指標を用いた性能評価を行った．ストレージとしては，HDD および PCIExpress 接続の SSD (RevoDrive 3 X2) を用いた．また，性能測定の指標としては，オブジェクトのサイズ，リクエストを発行するワーカー数，オブジェクトゲートウェイのキャッシュの有無，および，それらの指標に対する読み込み・書き込みリクエストについて検討した．実験の結果，より高性能なストレージを利用することで，オブジェクトの書き込み性能を大きく向上させられること，並列アクセスが行われる場合には，オブジェクトのサイズが性能に大きな影響を与える場合があること，そして，サイズが比較的小さなオブジェクトに対して，オブジェクトゲートウェイのキャッシュが性能向上に大きく寄与することを確認できた．

キーワード：分散オブジェクトストレージ，Ceph，オブジェクトゲートウェイ，性能評価

1. はじめに

従来，大規模なデータセットを利用したビッグデータの処理基盤としては，Apache Hadoop[2] や Apache Spark[3] などのデータ処理フレームワークが広く使用されてきた．これらのフレームワークは，主に Hadoop のために開発されたファイルシステム Hadoop Distributed File System(HDFS) にデータを格納し，その上でデータ処理を行うことが多い [1], [10].

Hadoop は数千台のスケールのストレージクラスタを構成することが可能であるが，非常に大規模なデータを格納する場合には，ストレージクラスタの運用が煩雑であったり，ストレージの利用効率が低いといった問題があった．

そのため，近年では，大規模なデータを格納する手段として，スケーラビリティが高く，ストレージ利用効率の高い，分散オブジェクトストレージが活用されるようになってきている．

ストレージ利用効率の点では，HDFS はストレージクラスタの複数のノードに渡ってデータのレプリカを作成するが，分散オブジェクトストレージでは，より効率のよい Erasure Coding に対応するシステムが増えている．

Hadoop や Spark などのデータ処理フレームワークと，分散オブジェクトストレージと仲介する役割を担うために，Amazon S3A コネクタや Alluxio(Tachyon)[4] などの仮想ストレージが開発されており，オブジェクトストレージと接続するインターフェイスとしては，S3 インターフェイスがデファクトスタンダードとなっている．

分散オブジェクトストレージは，基本的にはオブジェクト単位の操作を行う限定された API のみを提供し，階層管理をなくしたフラットな構造を採用することで，スケーラビリティを向上させている．一般的な POSIX 互換ファイルシステムとは異なる特性を持つため，高性能な計算のために十分な性能を発揮するために，特に，可搬性の観点からも，多くのシステムから利用される S3 インターフェイスを利用した場合の特性を明らかにすることが重要である．

また，ビッグデータの解析を HPC で行う需要があるため，オブジェクトストレージ自体の性能が求められるようになってきている．

オブジェクトストレージを利用する手段としては，様々な選択肢がある．まず，パブリッククラウドが提供するオブジェクトストレージのサービスが挙げられる．サービスの例としては，Amazon Web Service が提供するオブジェクトストレージサービス Simple Storage Service (S3)[7] や，Google Cloud Platform が提供する Google Cloud Storage (GCS)[8] が有名である．

これらのサービスを提供するソフトウェアはクローズ

[1] 筑波大学 情報学群 情報科学類
[2] 筑波大学 計算科学研究センター
[a)] shuuji3@hpcs.cs.tsukuba.ac.jp

ドであるが，一方で，オープンソースソフトウェアで実装されたオブジェクトストレージシステムも広く利用されている．特に，大規模なオブジェクトストレージシステムとして実環境でも利用されているソフトウェアとしては，OpenStack のプロジェクトの 1 つとして開発されている Swift[17] や，Linux Foundation 傘下のプロジェクトの 1 つとして，Ceph Foundation の下で開発が行われている Ceph[16] が挙げられる．

Ceph は，パブリックおよびプライベートクラウドのサービスのバックエンドとして活用されている．たとえば，パブリッククラウドの Digital Ocean では，オブジェクトストレージサービスのバックエンドとして Ceph を利用しており [20]，プライベートクラウドの例としては，HPC 上でのオンデマンドの計算資源に対する需要に答えるために，OpenStack を用いて構築したプライベートクラウド上に，大規模な Ceph のオブジェクトストレージサービスを構築し，運用している，Minnesota Supercomputing Institute による Stratus の事例がある [19]．

オンプレミスのシステムとは異なり，パブリッククラウドのサービスはインフラに対する設備投資や維持費を必要としないが，それらの費用はデータ保管料や転送量に転嫁されているため，大規模なデータを保管しようとすれば，使用方法によっては TOC は大きくなる可能性がある．

また，インターネット経由のデータ転送には，転送量がかかる以外にも，ネットワークのレイテンシが高かったり，データ転送速度が低いという問題がある．そのため，頻繁にアクセスが必要な解析用のデータは，依然として，オンプレミスに構築されたオブジェクトストレージクラスタを利用することも多い．

ただし，再利用の頻度が少ない，長期保管用のデータなどに対しては，オブジェクトストレージの中でもコールドストレージという種類のサービスを利用することが，費用を抑えるために有効な手段の 1 つである．実際の研究データの利用パターンに基づいた，パブリッククラウドのコスト試算を行うケーススタディも実施されており，パブリッククラウドのストレージ性能は年々向上していることがわかっている [9]．しかしながら，クラウド外へのデータ転送コストは比較的大きく，オンプレミスの環境で計算に比較的多くのデータが必要な場合や，高いパフォーマンスを必要とするケースでは，パブリッククラウドのデータ伝送コストや速度の問題を回避するため，オブジェクトストレージシステムをオンプレミスに構築することが，未だに有効であると考えられる．

前述したように，オブジェクトストレージのアダプタとしては S3 互換の API 利用されることが多い．また，S3 インターフェイスを利用することで，オブジェクトストレージシステムをオンプレミスのシステムや，ハイブリットクラウドに組み込んで利用するユースケースも考えられる．

そこで，本研究では，オープンソースの分散オブジェクトストレージ Ceph が提供する S3 インターフェイス (RADOS Gateway) を対象に，オンプレミスのオブジェクトストレージサービスで利用されることを念頭に置き，いくつかのストレージを利用した場合の基本的な性能を測定し，その特徴を明らかにすることを目的とする．そして，オンプレミスのオブジェクトストレージシステムを構築する際の参考に供したい．

2. 関連研究

大規模なデータ解析のためのデータ転送の高速化を行う研究としては，HDFS ファイルシステムと S3 インターフェイスの間にデータアクセスパタンを考慮したインテリジェントなコネクタを実装することで，オブジェクトストレージへのアクセスを最適化する研究がある [26]．この研究は，S3 インターフェイス自体の性能改善ではなく，HDFS ファイルシステムから S3 インターフェイスへのアクセス数を削減することにより，データ転送の高速化を図るものである．

Ceph のストレージシステムに対する性能評価を行っている研究としては，小規模な OpenStack クラウド上に構築した Ceph クラスタを対象に，Ceph の RBD および RADOS インターフェイスの性能を測定した研究 [23] や，Yahoo! Cloud Serving Benchmark (YCSB)[6] を用いて，RADOS インターフェイスの CRUD の性能を測定した研究がある [24]．これらは，Ceph が提供する librados またはブロックデバイスのためのインターフェイスである RBD を使用したものであり，本研究が対象とする RADOS Gateway とは，インターフェイスの特性が異なる．

RBD および RADOS に対して，オブジェクトサイズ，OSD 数，ジャーナル用のストレージを変化させた場合の詳細な性能評価を行った研究もある [25]．こちらはより詳細な性能特性の分析を行っているが，同様に，異なるインターフェイスを対象にしている．

3. 分散オブジェクトストレージ Ceph の概要

実験の準備として，Ceph のアーキテクチャの概要について述べる．Ceph は，オープンソースにより開発が行われている，スケーラブルなソフトウェア定義型の分散オブジェクトストレージシステムである．Ceph のクラスタは，コモディティハードウェアの上に構築することが可能である．実環境においても，数 10 - 数 100PB のサイズまでの運用例があり [21], [22]，エクサバイト規模の容量にまでスケールが可能であるとされている．

Ceph は RADOS (Reliable Autonomic Distributed Object Store) と呼ばれるオブジェクトストアを基盤に構築されている．RADOS の内部では，クラスタ内でのオブ

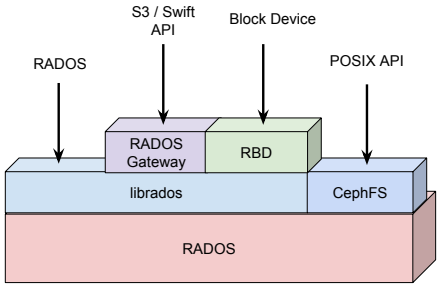

図 1　Ceph の各種インターフェイス *1

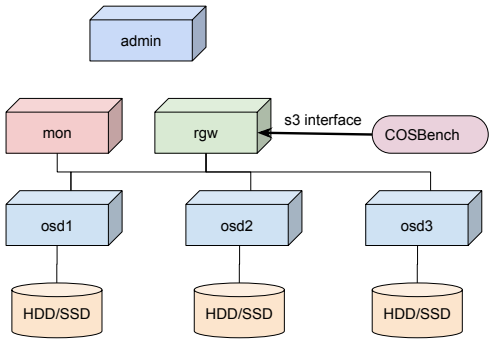

図 2　Ceph クラスタの構成

ジェクトの配置場所を決定するために，CRUSH(Controlled Replication Under Scalable Hashing) アルゴリズムを利用している．これは，オブジェクト名のハッシュと最新のクラスタマップを元に，格納場所を確定的に高速に計算することができるアルゴリズムであり，この特徴により，Ceph は高いスケーラビリティを実現している [12], [13].

Ceph は，システム利用者に対して複数のインターフェイスを提供することができる．図 1 に示すように，librados ライブラリを使用した RADOS オブジェクトへの直接アクセスによる利用ができる他，librados の上に構築されるブロックストレージデバイス (RDB) のインターフェイス，CephFS が提供する POSIX 互換のファイルシステムのインターフェイス，そして，RADOS Gateway が提供する S3 および Swift 互換の API 経由でのオブジェクトゲートウェイのインターフェイスを提供している．

さまざまな形態のストレージ利用方法を提供することができるという特徴のため，OpenStack クラウド上のコンポーネント Cinder や Glance のバックエンドとしても広く活用されており，Cinder のバックエンドとしては，OpenStack で構築されたクラウドの約 2/3 で利用されている [18]．また，OpenStack プロジェクトの下で開発が行われている Swift と並んで，OpenStack クラウド上のオブジェクトストレージのバックエンドとしても多く利用されている [11].

4. 実験: Ceph RADOS Gateway の性能評価

Ceph RADOS Gateway の性能評価を行う．

4.1 実験環境

HDD と SSD を利用した場合の性能を測定するために，ストレージとしては，各ストレージノードに，HDD および PCI Express 接続の SSD である RevoDrive 3 X2 を，それぞれ 1 台ずつ設置した．

*1 http://docs.ceph.com/docs/mimic/architecture/ を元に作成

表 1　実験環境

コンポーネント	説明
CPU	Intel Xeon CPU E5-2630 v4 @ 2.20GHz x2
メモリ	DDR2 FB-DIMM 667 MHz 4GB x 8
ネットワーク	10 GbaseT/Full
HDD	SEAGATE SAS HDD ST9500430SS 500 GB
SSD	RevoDrive 3 X2 PCI-Express SSD 240 GB
OS	CentOS Linux release 7.5.1804 (Core)
Linux	3.10.0-862.14.4.el7.x86_64
Ceph	13.2.1 mimic (stable)

Ceph のクラスタ構成図を図 2 に示す．また，クラスタの各ノードの実験環境を表 1 に，各ノードの役割と搭載されたストレージを表 2 に示す．

本研究では，6 台のノードからなる Ceph クラスタを構築し，実験を行った．クラスタを構成する各ノードの役割とストレージについて説明する．

まず，admin ノードは，管理用のノードであり，Ceph クラスタのセットアップに利用される．このノードだけは，Ceph の機能は提供しない．

次に，mon ノードは，Ceph を構成するクラスタ全体のノード・OSD(Object Storage Daemon)・rgw など，クラスタを構成するすべてのコンポーネントの情報を含むクラスタマップを管理する．ここで，クラスタマップは，CRUSH アルゴリズムによって使用され，オブジェクトの配置場所を決定するための入力情報の 1 つとして利用される．OSD とは，通常物理ストレージ 1 台に対して 1 プロセス実行されるデーモンであり，実際のストレージへのアクセスを司る役目を持つ．

そして，osd ノードは，この OSD が実行されるノードである．本実験では，3 台の osd ノードを使用し，同時に実行する OSD は 1 ノードにつき 1 プロセスであり，HDD と SSD を切り替えて実験を行った．

最後に rgw ノードは，osd ノードに格納された Ceph 特有のオブジェクト形式である RADOS オブジェクトを，S3/Swift 互換 API として提供する RADOS Gateway を実行するノードである．

表 2　Ceph ストレージクラスタの各ノードの役割

ノード名	役割	ストレージ
mon	クラスタ監視ノード	-
osd1	ストレージノード	HDD x1，SSD x1
osd2	ストレージノード	HDD x1，SSD x1
osd3	ストレージノード	HDD x1，SSD x1
rgw	オブジェクトゲートウェイ	-
admin	管理用ノード	-

表 3　bonnie++ を用いた HDD および SSD のディスク性能

アクセスタイプ	HDD	SSD	SSD / HDD 性能比
Sequential Read	132 MB/s	453 MB/s	3.43
Sequential Write	102 MB/s	390 MB/s	3.82

表 4　iperf を用いたノード間通信速度の計測結果

転送データ量	10.8 GBytes
バンド幅	9.25 Gbits/s

4.2　ストレージの基本性能の測定

はじめに，実験に使用した HDD および SSD に対して，ベンチマークを実行し，基本性能を確認した．

基本性能は，bonnie++[14] を用いて測定した．使用した bonnie++ のバージョン は，1.97 である．計測結果を表 3 に示す．読み込み・書き込みともに，HDD に対して、SSD は約 3-4 倍の性能があることを確認した．

次に，ノード間ネットワークの通信速度を，iperf[15] を用いて確認した．使用した iperf のバージョン は， 2.0.12 である．計測結果を表 4 に示す．バンド幅は 9.25 Gbits/s を達成しており、10 GbaseT の性能が問題なく発揮できていることを確認した．

4.3　性能測定ツール

オブジェクトストレージの性能測定には，COSBench[5] を使用した．COSBench のアーキテクチャを図 3 に示す．

COSBench は，オブジェクトストレージの性能測定のために Intel により開発されたオープンソースのベンチマークツールであり，Controller と Driver の 2 つのコンポーネントからなる．ユーザーは，あらかじめ作成したワークロードファイルを Controller に送信する．Controller は，ワークロードをキューに保存し，Driver に対して順次ワークロードの実行命令を発行する．そして，命令を受けた Driver が，オブジェクトゲートウェイに対して順次リクエストを発行し，その性能を測定する．

本実験では，ノード間のネットワーク帯域がオブジェクトサイズに対して十分大きく，ネットワークの性能によりストレージの性能が制限される可能性は低いと考えられるため，オブジェクトゲートウェイを実行しているノードに Driver を配置し，測定を行った．

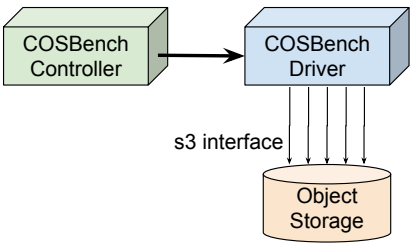

図 3　COSBench のアーキテクチャ

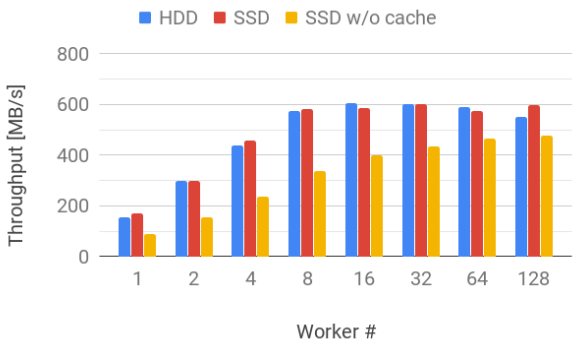

図 4　測定 1: GET・リクエストワーカー数を変化させた場合

4.4　実験の目的と評価指標

オブジェクトゲートウェイの性能特性を調べるための指標としては，オブジェクトのリクエストに対する読み込み (GET) および書き込み (PUT) のアクセスパタン，リクエストを発行するワーカー数，および，リクエストするオブジェクトのサイズを考える．

リクエストのパターンによる性能の違いを調べることで，アプリケーションの要求に応じたオブジェクトストレージの性能の違いを明らかにする．また，ワーカー数を変化させることで，オブジェクトゲートウェイの並列アクセスに対する性能を確認する．そして，リクエストするオブジェクトのサイズによる性能の変化を見ることで，データをオブジェクトに分割する際などに使用するべき，適切なサイズを確認することができると考えられる．

4.5　実験の結果と評価

4.5.1　測定 1: GET・リクエストワーカー数を変化させた場合

条件

はじめに，Ceph RADOS Gateway に対して，読み込みを行う GET リクエストのみを行うワークロードを実行した．

まず，オブジェクトゲートウェイに対して，HDD および SDD を使用したストレージクラスタのそれぞれの場合で，オブジェクトサイズを 1 MB に固定して，GET リクエストのみを発行するワークロードを用いて測定を行った．こ

のとき，ワーカー数を，1，2，4，8，16，32，64，128 と変化させ，並列アクセスに対する性能を調べた．

RADOS Gateway には，同一のオブジェクトゲートウェイへのアクセスを高速化する目的で，内部にキャッシュの機構が設けられている．この機能を無効化した場合の性能も，その他の条件を同じにして確認した．これにより，キャッシュ機能ががオブジェクトゲートウェイへのアクセスに対して寄与する程度を明らかにすることができると考えられる..

結果

実験の結果を，図 4 に示す．「HDD」は，HDD を使用した 3 台のストレージノードを使用した場合，「SSD」は，SSD を使用した 3 台のストレージノードを使用した場合，「SSD w/o cache」は，さらに，オブジェクトゲートウェイのキャッシュを無効にした場合を表している．

ワーカー数が同じ場合，HDD と SSD とで，有意な性能差は見られなかった．

スループットは，ワーカー数が 1 の時，約 150-170 MB/s であり，ワーカー数が 8 以上になると，約 600 MB/s で飽和した．また，ワーカー数を 128 まで増加させた場合でも，600 MB/s を上回ることはなかった．

また，RADOS Gateway のキャッシュを無効にした場合，オブジェクトサイズが 10 MB までは，最大で約 65%の性能の低下が見られたが，10 MB 以上のときは，大きな性能の低下はなかった．

評価

表 3 に示したように，SSD の性能は HDD より十分にも大きいにもかかわらず，GET リクエストの並列度を変えても，性能差が見られなかった．これは，RADOS Gateway が GET リクエストの性能のボトルネックになっており，ストレージの性能を十分に発揮できなかったためだと考えられる．

一方，ストレージの種類にかかわらず，並列度を上げることにより性能が向上すること，8 並列以上のときは大きな性能差はないが，16 並列付近で最も高い性能が発揮されることを確認できた．

4.5.2 測定 2: GET・オブジェクトサイズを変化させた場合 (ワーカー数 1)

条件

次に，リクエストを発行するワーカー数を 1 に固定し，GET するオブジェクトのサイズを 10 KB，100 KB，1 MB，10 MB，100 MB，1 GB と変化させて，オブジェクトサイズによるゲートウェイの読み込み性能の変化を調べた．

結果

実験の結果を，図 5 に示す．

ワーカー数が 1 で，並列アクセスがない場合には，オブジェクトサイズを大きくすることで，スループットが向上した．

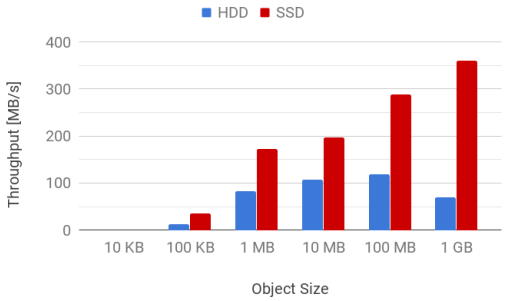

図 5 測定 2: GET・オブジェクトサイズを変化させた場合 (ワーカー数 1)

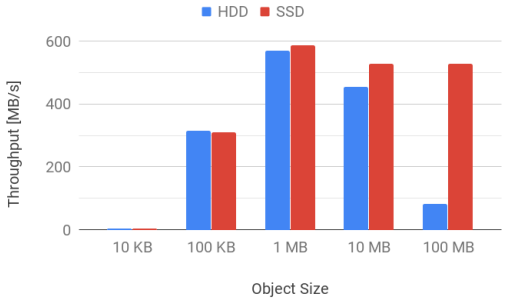

図 6 測定 3：GET・オブジェクトサイズを変化させた場合 (ワーカー数 16)

ストレージによる違いに着目すると，オブジェクトのサイズが 100 MB を下回る場合には，SSD が HDD より高い書き込み性能を発揮した．しかしながら，オブジェクトのサイズが 100 MB 以上になると，HDD 書き込み性能が SSD を上回った．

評価

ワーカーが 1 つのみで，並列アクセスがない場合には，オブジェクトサイズを大きくするほど，スループットが向上することが確認できた．これは，オブジェクトゲートウェイに対するリクエストのオーバーヘッドが相対的に小さくなるためだと思われる．

また，オブジェクトのサイズが比較的小さい場合には，ゲートウェイのキャッシュを有効にすることで，性能の向上に大きく寄与することがわかった．

一方，オブジェクトサイズが大きい場合に，予想に反して，SSD よりも HDD の書き込み性能が高くなる原因については，明らかにすることができなかった．

4.5.3 測定 3：GET・オブジェクトサイズを変化させた場合 (ワーカー数 16)

条件

次に，節 4.5.1 の結果を踏まえて，ワーカー数を，並列アクセス時に最も高い性能が発揮された 16 に固定し，GET するオブジェクトのサイズを 10 KB，100 KB，1 MB，10 MB，100 MB と変化させて，GET リクエストのみを発行

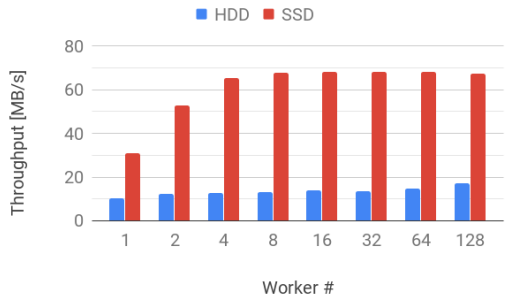

図 7　測定 4: PUT・リクエストワーカー数を変化させた場合

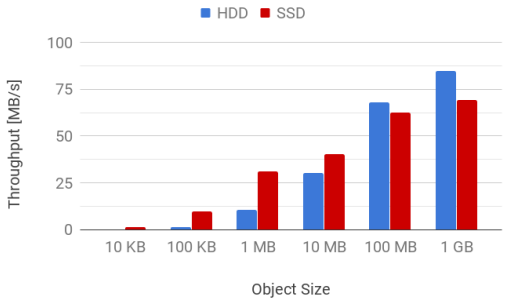

図 8　測定 5: PUT・オブジェクトサイズを変化させた場合 (ワーカー数 1)

するワークロードを用いて測定を行った．

結果

実験の結果を，図 6 に示す．ワーカー数 16 の場合，SSD においては，オブジェクトサイズが 1 MB のときに最大で，約 600 MB/s のスループットが出ている．また，オブジェクトサイズを 10 MB，100 MB と大きくしても，スループットが 600 MB/s を超えることはなかった．

一方，HDD では，SSD と同様にオブジェクトサイズが 1 MB のときにスループットが最大であるが，オブジェクトサイズをさらに大きくすると，性能が大きく低下した．

評価

節 4.5.2 において，ワーカー数 1 の場合と比較すると，並列アクセスが多く発生する状況では，オブジェクトサイズを大きくすることが，必ずしも性能向上に寄与しないため，適切なオブジェクトを適切なサイズにすることが重要であることがわかった．

HDD の場合にオブジェクトサイズを大きくなると，性能が低下している．これは，HDD のでは，SSD に比較して読み取りのアクセス時間が長くなるために，3 つのストレージからなるクラスタに 16 並列でアクセスが発生すると，アクセスの発生するために性能が大きく低下するのだと思われる．

4.5.4　測定 4: PUT・リクエストワーカー数を変化させた場合

条件

PUT のワークロードでも，GET の場合と同様に実験を行った．まず，オブジェクトサイズを 1 MB に固定して，PUT リクエストを発行するワーカー数を，1，2，4，8，16，32，64，128 と変化させて測定を行った．

結果

実験の結果を，図 7 に示す．ワーカー数にかかわらず，SSD は HDD の約 3-5 倍の書き込み性能を発揮している．

HDD の場合，ワーカー数が増えると性能がわずかに向上した．SSD の場合は，ワーカー数が 1 から 4 まで増えると，2 倍以上の性能向上が見られたが，ワーカー数を 4 以上にしても，70 MB/s を超えることはなかった．

評価

節 4.5.1 で示されたように，RADOS Gateway を介した読み込みの性能は，HDD と SSD とで有意な差が見られなかったが，書き込みの性能に関しては，ストレージとして HDD の代わりに SSD を使用することで，大幅な性能向上を実現することができた．

また，SSD の場合に，ワーカー数が 4 以上のときには，書き込み性能が飽和した．これは，本実験の Ceph クラスタのストレージおよびストレージノードがそれぞれ 3 台であるため，ストレージ台数以上の同時書き込みを行うことができないためだと考えられる．

4.5.5　測定 5: PUT・オブジェクトサイズを変化させた場合 (ワーカー数 1)

条件

次に，リクエストを発行するワーカー数を 1 に固定し，PUT するオブジェクトのサイズを 10 KB，100 KB，1 MB，10 MB，100 MB と変化させて，オブジェクトサイズによるゲートウェイの書き込み性能の変化を調べた．

オブジェクトの書き込みのみの場合は，オブジェクトゲートウェイのキャッシュの影響はないため，測定は行わない．

結果

実験の結果を，図 8 に示す．ワーカー数が 1 の場合は，オブジェクトサイズが大きくなるにつれて，スループットも向上した．

評価

オブジェクトサイズの増加によってスループットが向上する傾向は，節 4.5.2 の GET リクエストのみの実験と同様である

4.5.6　測定 6：PUT・オブジェクトサイズを変化させた場合 (ワーカー数 16)

条件

次に，節 4.5.4 の結果を踏まえて，ワーカー数を，最も高い性能が発揮される 16 に固定し，PUT するオブジェクトのサイズを 10 KB，100 KB，1 MB，10 MB，100 MB，1 GB と変化させて，PUT リクエストのみを発行するワー

図 9　測定 6：PUT・オブジェクトサイズを変化させた場合 (ワーカー数 16)

クロードを用いて測定を行った．

結果

実験の結果を，図 7 に示す．HDD の場合は，オブジェクトサイズが 10 MB までは，比較的低い性能であったが，10 MB および 100 MB の場合は，同程度に高い性能を発揮した．SSD の場合は，1 MB の場合でも，10 MB，100 MB と同程度に高い性能を発揮した．

評価

ワーカー数が 1 の場合と同様に，並列アクセス時にも，オブジェクトのサイズが小さすぎると，十分な性能が発揮できないことが確認できた．そのため，オブジェクトサイズを適切な大きさにすることが重要であることが分かった．

また，高性能なストレージを利用することで，より小さいサイズのオブジェクトの書き込み時にも，書き込み性能の劣化を抑えられることが明らかになった．

5. まとめと今後の課題

本研究では，分散オブジェクトストレージ Ceph を対象に，S3 インターフェイスを提供する RADOS Gateway を利用した，オブジェクトベースのストレージアクセス性能を測定し，評価した．

その結果，以下のことが明らかになった．

- より高性能なストレージを利用することで，オブジェクトの書き込み性能を大きく向上させられること，
- 並列アクセスが行われる場合には，オブジェクトのサイズが性能に大きな影響を与える場合があること，
- サイズが比較的小さなオブジェクトに対して，オブジェクトゲートウェイのキャッシュが性能向上に大きく寄与すること，

一方で，複数ワーカーによるオブジェクトの GET アクセス時に，高性能なストレージを利用した場合であっても，オブジェクトの読み込み性能が向上することを確認できなかった．これは，RADOS Gateway が性能のボトルネックとなっているため，ストレージの性能を引き出すことができなかったためだと思われる．このボトルネックを明らかにすることで，オブジェクトの読み込み時にもストレージ性能を発揮し，オブジェクトストレージ全体の性能を改善できる可能性がある．

また，1 つのワーカーによりオブジェクトの PUT アクセス時に，オブジェクトのサイズが 100 MB を上回る場合には，予想に反して，HDD 書き込み性能が SSD を上回った．この原因については明らかにすることができなかった．

本研究では，3 台のストレージノードとして，HDD，SSD を複数台用い，ストレージを使用するプログラムも複数プロセス起動して性能を測定したが，S3 インターフェイスを提供するオブジェクトゲートウェイは 1 のみで実行した．S3 インターフェイスのためのプロセスを複数起動し，ロードバランシングを行うことで，書き込み時の性能を改善できる可能性が考えられる．

謝辞　本研究の一部は，JST CREST JPMJCR1303，JST CREST JPMJCR1414，JSPS 科研費 17H01748 および富士通研究所との共同研究の助成を受けたものです．

参考文献

[1] Singh, Dilpreet and Reddy, Chandan K.: *A survey on platforms for big data analytics*, Journal of Big Data, Springer International Publishing (2014).
[2] Apache Hadoop (オンライン), 入手先 ⟨https://hadoop.apache.org/⟩, (参照 2018-11-13).
[3] Apache Spark™ - Unified Analytics Engine for Big Data (オンライン), 入手先 ⟨http://spark.apache.org/⟩, (参照 2018-11-13).
[4] Alluxio - Open Source Memory Speed Virtual Distributed Storage (オンライン), 入手先 ⟨https://www.alluxio.org/⟩, (参照 2018-11-13).
[5] Qing Zheng, Haopeng Chen, Yaguang Wang, Jian Zhang, and Jiangang Duan: *COSBench: cloud object storage benchmark*, In Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE '13), ACM, New York, NY, USA (2013).
[6] Cooper, Brian F. and Silberstein, Adam and Tam, Erwin and Ramakrishnan, Raghu and Sears, Russell: *Benchmarking Cloud Serving Systems with YCSB*, Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10), ACM (2010).
[7] Amazon S3 | AWS (オンライン), 入手先 ⟨https://aws.amazon.com/jp/s3/⟩, (参照 2018-11-13).
[8] Cloud Storage | Google Cloud (オンライン), 入手先 ⟨https://cloud.google.com/storage/⟩, (参照 2018-11-13).
[9] 吉田 浩，合田 憲人，上田 郁夫，原 隆宣，小杉 城治，森田 英輔，中村 光志，クラウドコールドストレージに対する 大規模実験データ格納のケーススタディ，情報処理学会研究報告 Vol.2018-HPC-165 No.8 (2018).
[10] J Sánchez and A Fernández Casaní and S González de la Hoz: *Distributed Data Collection for the ATLAS EventIndex*, 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015).
[11] T Bell, B Bompastor, S Bukowiec, J Castro Leon, M K Denis, J van Eldik, M Fermin Lobo, L Fernandez Alvarez, D Fernandez, Rodriguez, A Marino, B Moreira, B Noel, T Oulevey, W Takase, A Wiebalck and S Zilli:

Scaling the CERN OpenStack cloud, 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015).

[12] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, Carlos Maltzahn: *Ceph: a scalable, high-performance distributed file system*, Proceedings of the 7th symposium on Operating systems design and implementation, USENIX Association (2006).

[13] Weil, Sage A. and Brandt, Scott A. and Miller, Ethan L. and Maltzahn, Carlos: *CRUSH: controlled, scalable, decentralized placement of replicated data*, SC '06 Proceedings of the 2006 ACM/IEEE conference on Supercomputing Article No. 122, ACM (2016).

[14] Bonnie++ – Russell Coker's Documents (オンライン), 入手先 ⟨https://www.coker.com.au/bonnie++/⟩, (参照 2018-11-13).

[15] iperf2 | SourceForge.net (オンライン), 入手先 ⟨https://sourceforge.net/projects/iperf2/⟩, (参照 2018-11-13).

[16] Ceph Homepage - Ceph (オンライン), 入手先 ⟨https://ceph.com/⟩, (参照 2018-11-13).

[17] OpenStack Swift (オンライン), 入手先 ⟨http://swift.openstack.org/⟩, (参照 2018-11-13).

[18] OpenStack User Survey Analytics and Data (オンライン), 入手先 ⟨https://www.openstack.org/analytics⟩, (参照 2018-11-13).

[19] Bollig, Evan F. and Allan, Graham T. and Lynch, Benjamin J. and Huerta, Yectli A. and Mix, Mathew and Munsell, Edward A. and Benson, Raychel M. and Swartz, Brent: *Leveraging OpenStack and Ceph for a Controlled-Access Data Cloud*, Proceedings of the Practice and Experience on Advanced Research Computing, PEARC '18, Article No. 18, ACM, Pittsburgh, PA, USA (2018).

[20] Storage on DigitalOcean (オンライン), 入手先 ⟨https://github.com/digitalocean/navigators-guide/blob/master/book/03-backup/ch07-storage-on-digitalocean.md⟩, (参照 2018-11-13).

[21] object storage | Yahoo Engineering (オンライン), 入手先 ⟨https://yahooeng.tumblr.com/tagged/object-storage⟩, (参照 2018-11-13).

[22] Ceph 30PB Test Report (オンライン), 入手先 ⟨https://cds.cern.ch/record/2015206⟩, (参照 2018-11-13).

[23] X. Zhang, S. Gaddam and A. T. Chronopoulos: *Ceph Distributed File System Benchmarks on an Openstack Cloud*, 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), IEEE, Bangalore, (2015).

[24] J. Lee, C. Song and K. Kang: *Benchmarking Large-Scale Object Storage Servers*, IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), IEEE, Atlanta, GA, (2016).

[25] D. Gudu, M. Hardt and A. Streit: *Evaluating the performance and scalability of the Ceph distributed storage system*, 2014 IEEE International Conference on Big Data (Big Data), IEEE, Washington, DC, USA (2014).

[26] Gil Vernik, Michael Factor, Elliot K. Kolodner, Pietro Michiardi, Effi Ofer and Francesco Pace: *Evaluating the performance and scalability of the Ceph distributed storage system*, 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing , IEEE, Washington, DC, USA (2018).