ストレージコネクタ spark-ceph-connectorの 書き込み性能の改善

2020/09/25(Fri.) 第176回 HPC研究発表会

高橋 宗史 [1)] 建部 修見 [2)]

1) 筑波大学 大学院 システム情報工学研究科 コンピュータサイエンス専攻 HPCS研究室
2) 筑波大学 計算科学研究センター

発表の構成

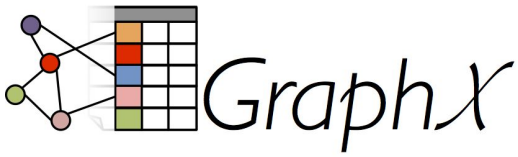
1. 研究の背景
2. 課題と研究の目的
3. 提案手法
4. 予備実験
5. 実験結果
6. まとめ
7. 関連研究
8. 今後の課題

研究の背景

1. 研究の背景
2. 課題と研究の目的
3. 提案手法
4. 予備実験
5. 実験結果
6. まとめ
7. 関連研究
8. 今後の課題

研究の背景: Apache Spark

- **Apache Spark** (Zaharia et al., 2012, USENIX NSDI '12)[1][2]
  - Hadoopが苦手とする反復的な処理が高速
  - 機械学習を含む充実した標準ライブラリ
  - データ解析の広い分野で人気
  - spark-operatorやKubeflowに対応し、Kubernetesを含む幅広いプラットフォームに対応



[1] Spark: Cluster Computing with Working Sets | USENIX HotCloud '10 - Usenix https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets
[2] Apache Spark™ - Unified Analytics Engine for Big Data - https://spark.apache.org/

研究の背景: 分散オブジェクトストレージ Ceph

- **Ceph** (Sage A. Weil, 2006, USENIX OSDI '06) [3]
  - Linux Foundation 傘下の Ceph Foundation で RedHat 等を中心に活発に開発が行われているオープンソースの分散オブジェクトストレージ
  - 特徴: メタデータサーバーが不要なRADOSをベースとし、高いスケーラビリティを持つ

[3] Ceph: a scalable, high-performance distributed file system, Weil, Sage A. et al. OSDI '06
DOI: 10.5555/1298455.1298485

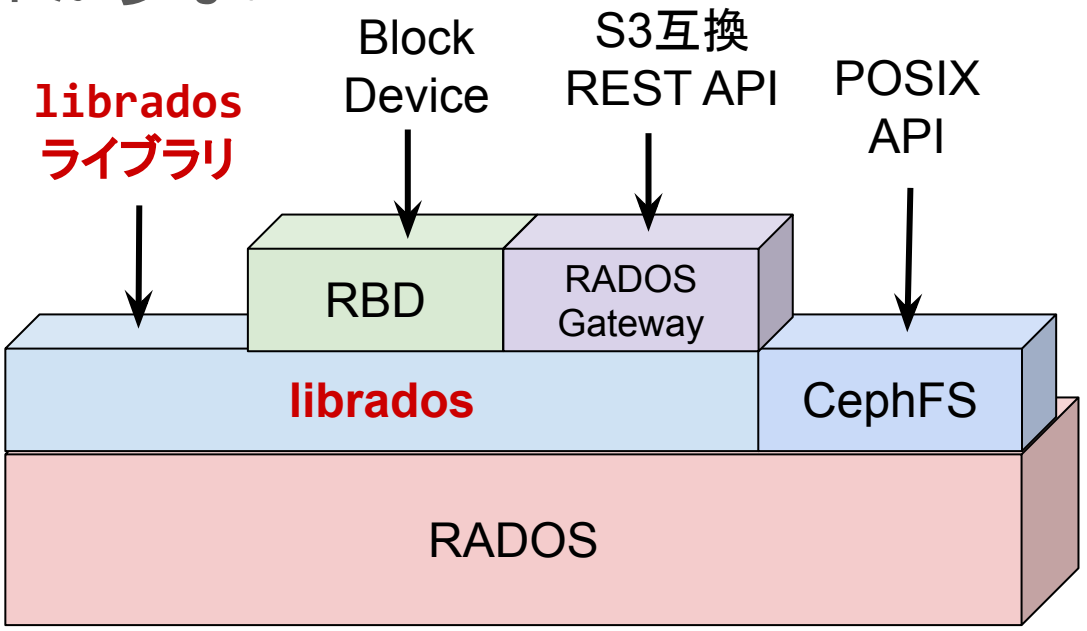
研究の背景: 分散オブジェクトストレージ Ceph

- **Ceph** (Sage A. Weil, 2006, USENIX OSDI '06) [4]
  - 利用例: CERN, Yahoo などでの10 - 100 PB級の大規模なデータの格納など[4][5]
  - 最新版で完全にコンテナ化され、Rookを利用したKubernetesクラスター上でのストレージクラスタの自動構築が可能 [6]

[4] Storage for Open Infrastructure. Dan Van Der Ster | CERN IT Storage Group
https://indico.cern.ch/event/860733/contributions/3625111/attachments/1938852/3214034/2019-intro-ceph.pdf
[5] Yahoo Cloud Object Store - Object Storage at Exabyte Scale | Yahoo Engineering
https://yahooeng.tumblr.com/post/116391291701/yahoo-cloud-object-store-object-storage-at
[6] Rook: Open-Source, Cloud-Native Storage for Kubernetes - https://rook.io/

研究の背景: Ceph のインターフェイス

- Cephは主に4種類のインターフェイスを提供
  - 仮想ブロックデバイス (RBD)
  - S3 互換 REST API (RADOS Gateway)
  - CephFS (POSIX API)
  - RADOS ネイティブオブジェクト (librados)
    - 最もオーバーヘッドが少ない



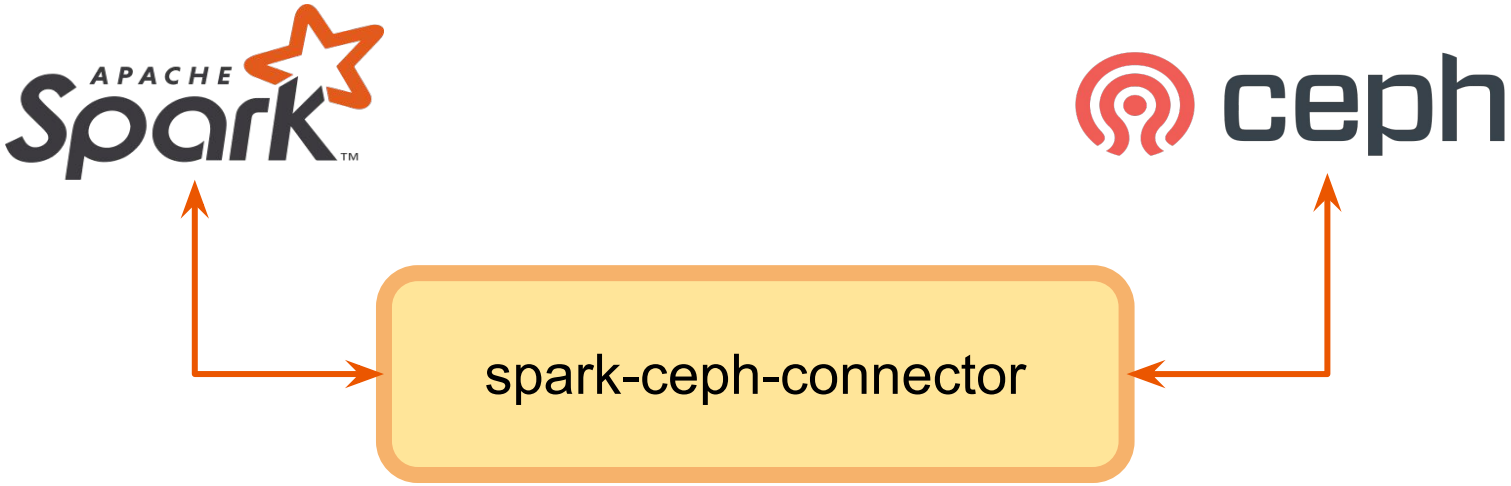
http://docs.ceph.com/docs/mimic/architecture/ を元に作成

課題と研究の目的

1. 研究の背景
2. 課題と研究の目的
3. 提案手法
4. 予備実験
5. 実験結果
6. まとめ
7. 関連研究
8. 今後の課題

課題と研究の目的: spark-ceph-connectorの開発

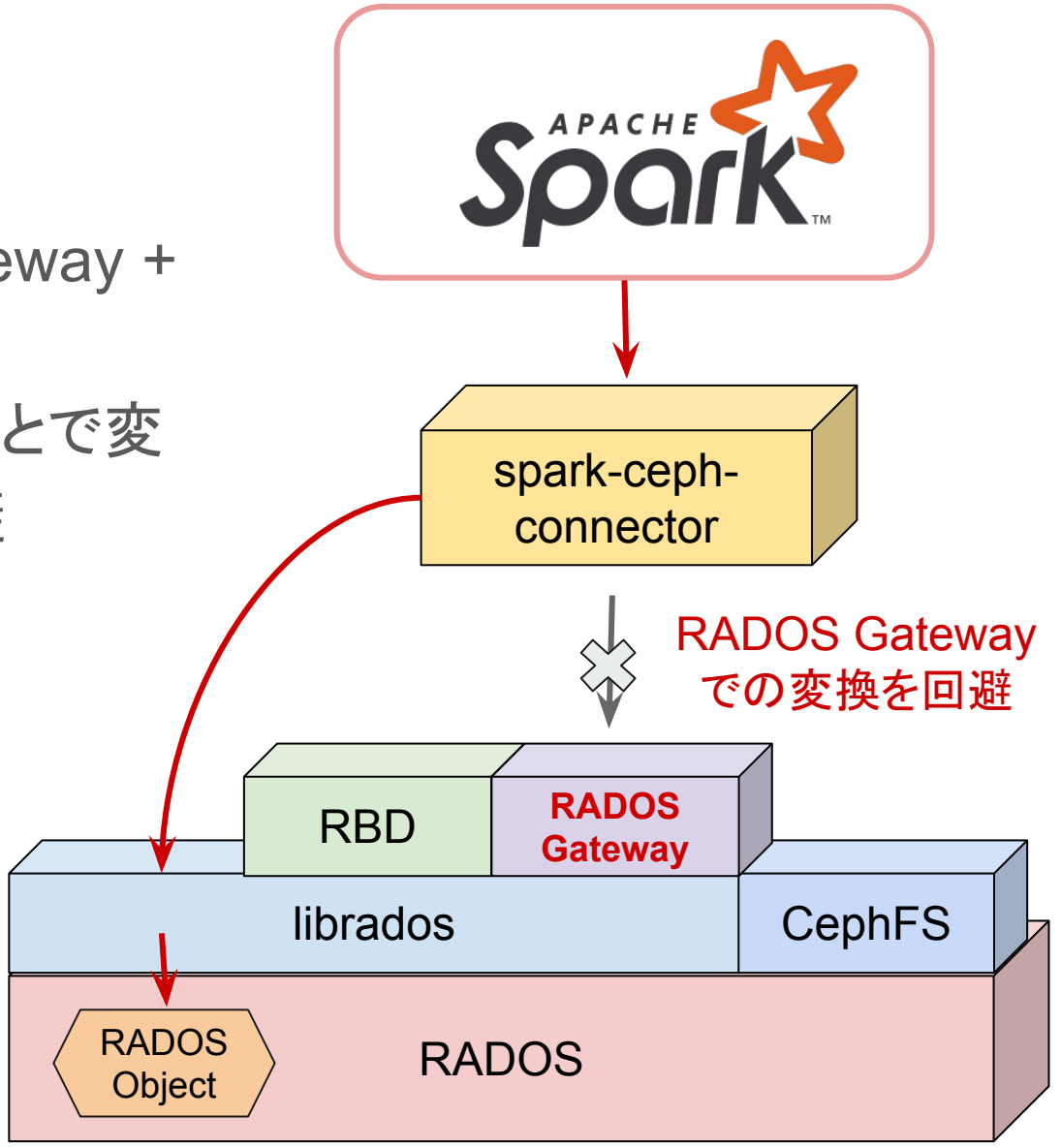
- Cephの大規模データをApache Sparkから有効に活用することを目的にCephを有効に活用できるストレージコネクタspark-ceph-connectorを設計・実装した [7]



[7] 高橋 宗史, 建部 修見: “分散オブジェクトストレージ Ceph のための Spark ストレージコネクタの設計”, 情報処理学会 第171回 HPC 研究会報告 (HPC171), Vol. 2019-HPC-171, No. 1, Sep. 2019.

課題と研究の目的: spark-ceph-connector

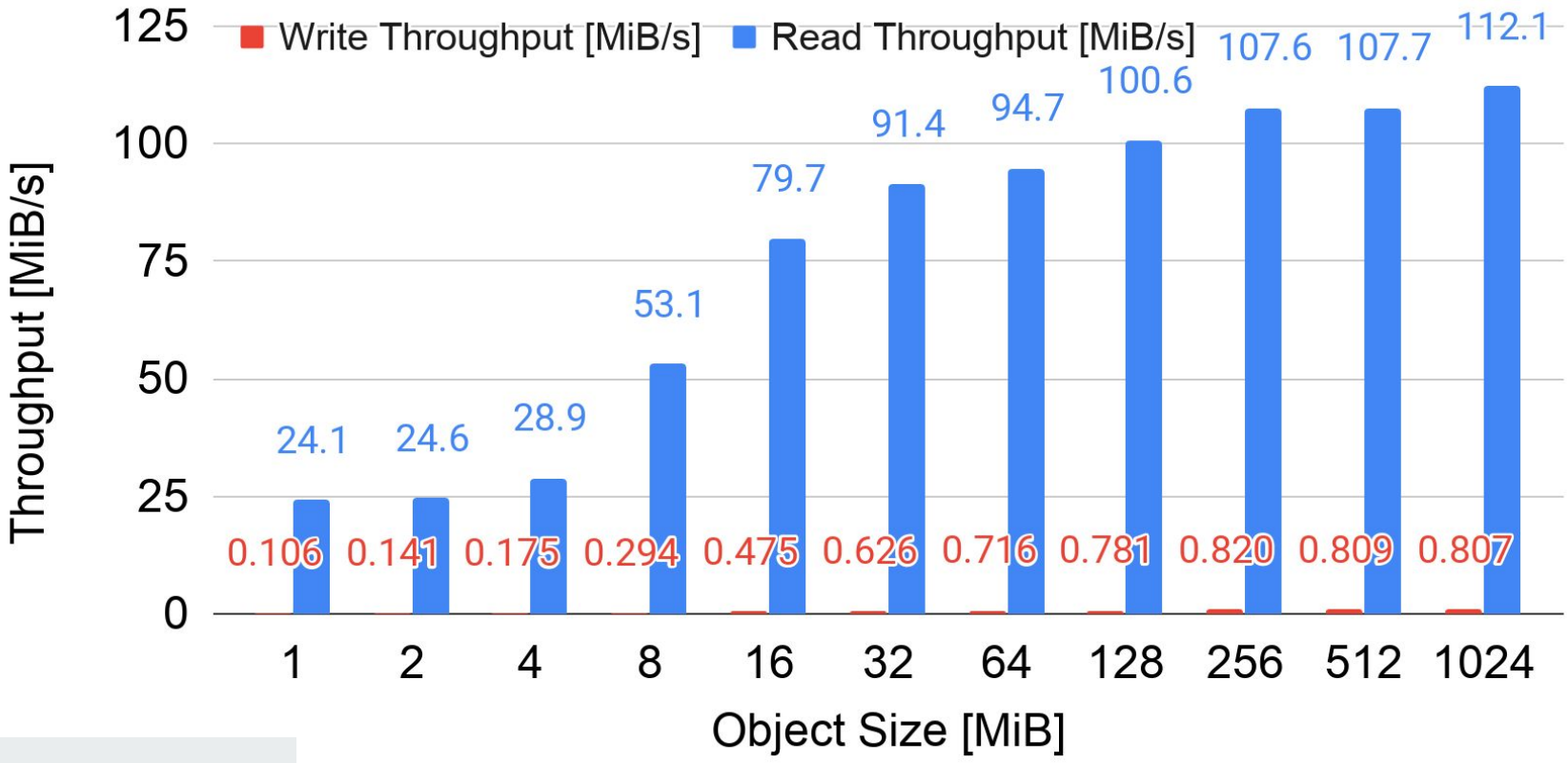
- ストレージコネクタspark-ceph-connector
- 従来: Ceph RADOS Gateway + S3オブジェクトへの変換
- libradosを直接利用することで変換のオーバーヘッドを回避



ceph

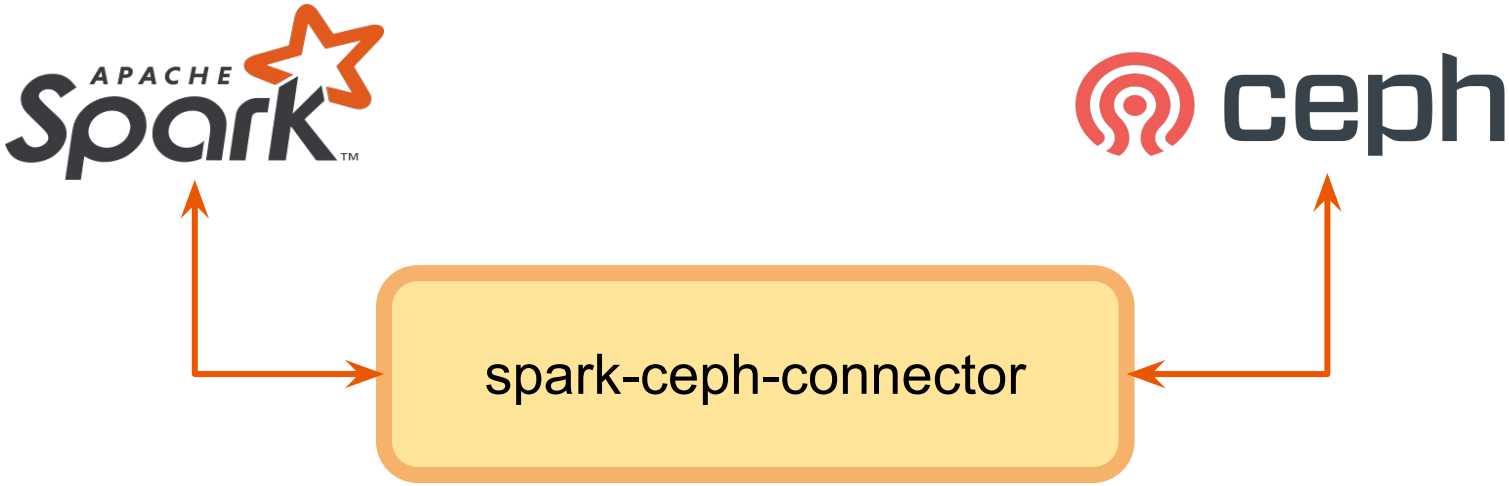
課題と研究の目的: 書き込み性能極めて低い問題

- 改善前のspark-ceph-connectorに対する読み込みと書き込みの性能測定の結果[7]
  - 読み込み性能はベース性能に性能を発揮
  - **書き込み性能が極めて低い** (< 1 MiB/s)



課題と研究の目的: 本研究の目的

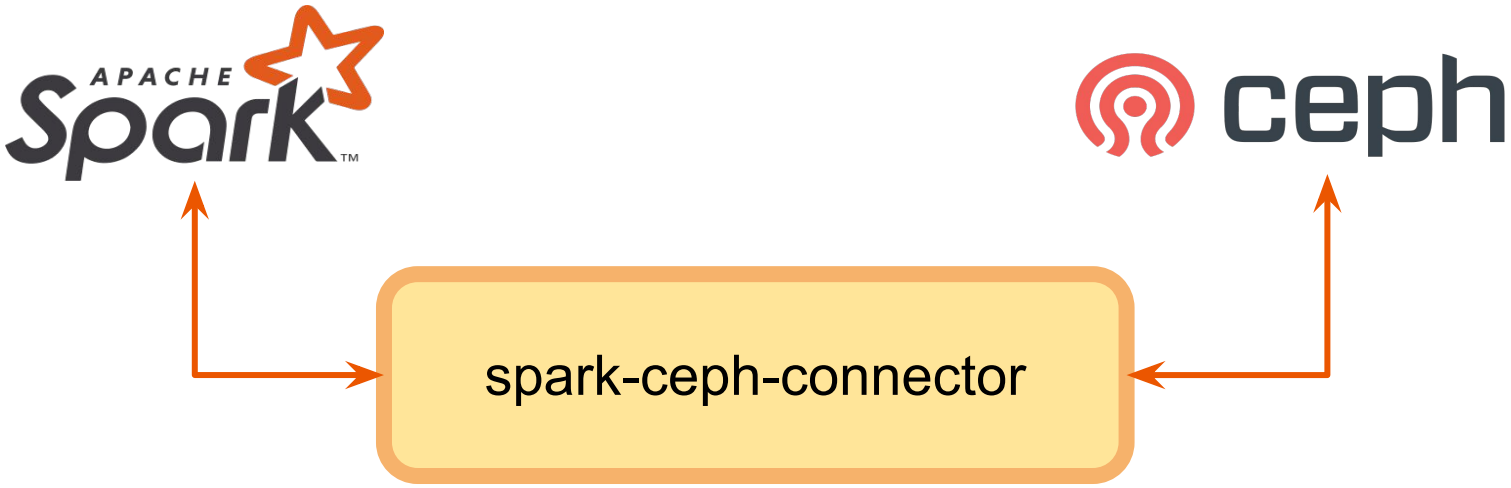
- 書き込み性能が極めて低い原因について、Ceph・Apache Spark特有の性質の分析・対処
- ストレージコネクタspark-ceph-connectorの書き込み性能の改善方法の提案



提案手法

提案手法: 概要

- 提案手法の概要
  1. Cephのreplication factorと、Committerのアルゴリズムの変更による改善
  2. ストレージコネクタspark-ceph-connectorの内部バッファサイズ変更による改善
  3. 書き込みデータのSpark上のシリアライズ方法の変更による改善

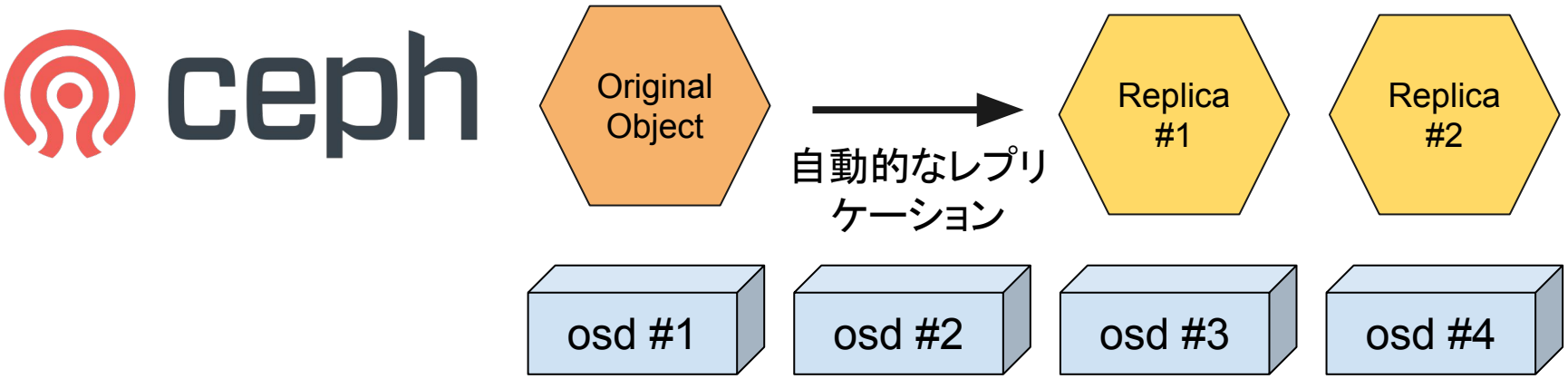


提案手法: replicationとCommitterの変更による改善

- Cephのレプリケーションファクターの変更
  - デフォルト: 3 → 1
  - 一時データの複製を回避
- FileOutputCommitter のコミット方法
  - デフォルト: アルゴリズム v1 → アルゴリズム v2
  - 2回のrename()を1回に半減

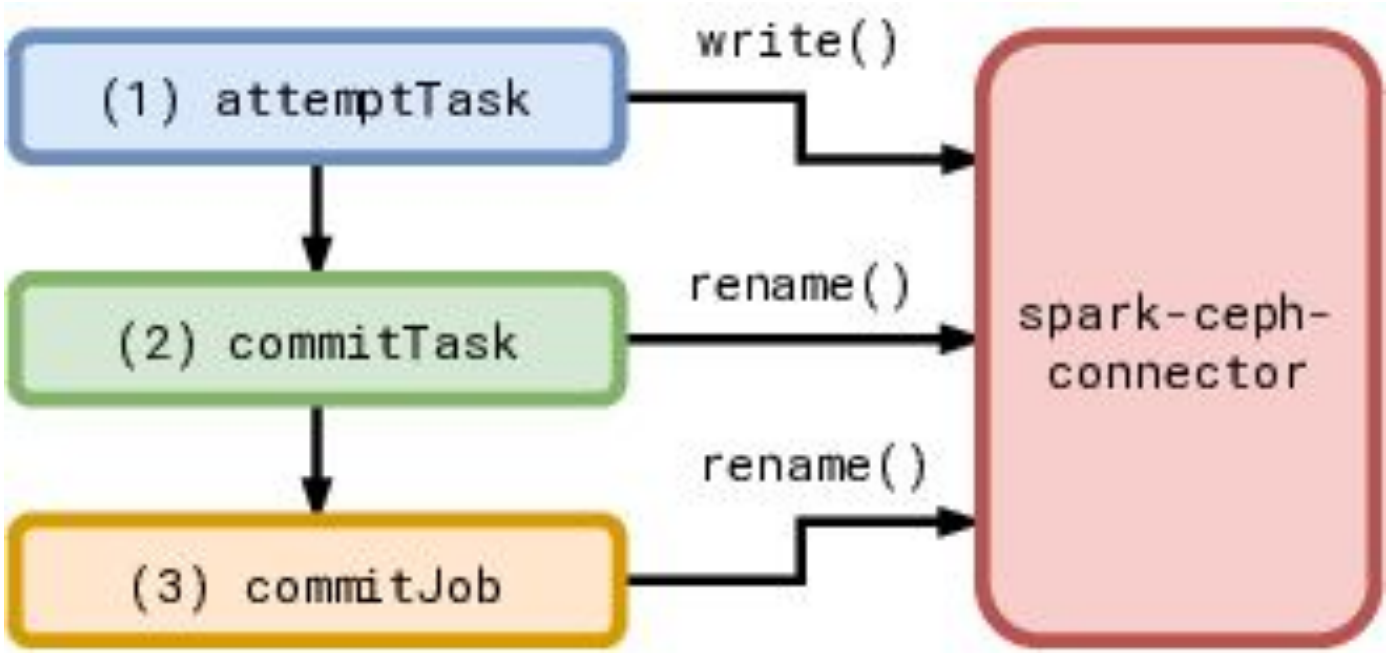
提案手法: Cephのreplication factor

- Cephのレプリケーション
  - CephがベースとするRADOSでは、オブジェクトのレプリケーションが自動的に実行される
  - 一時データの複製が書き込み性能に影響



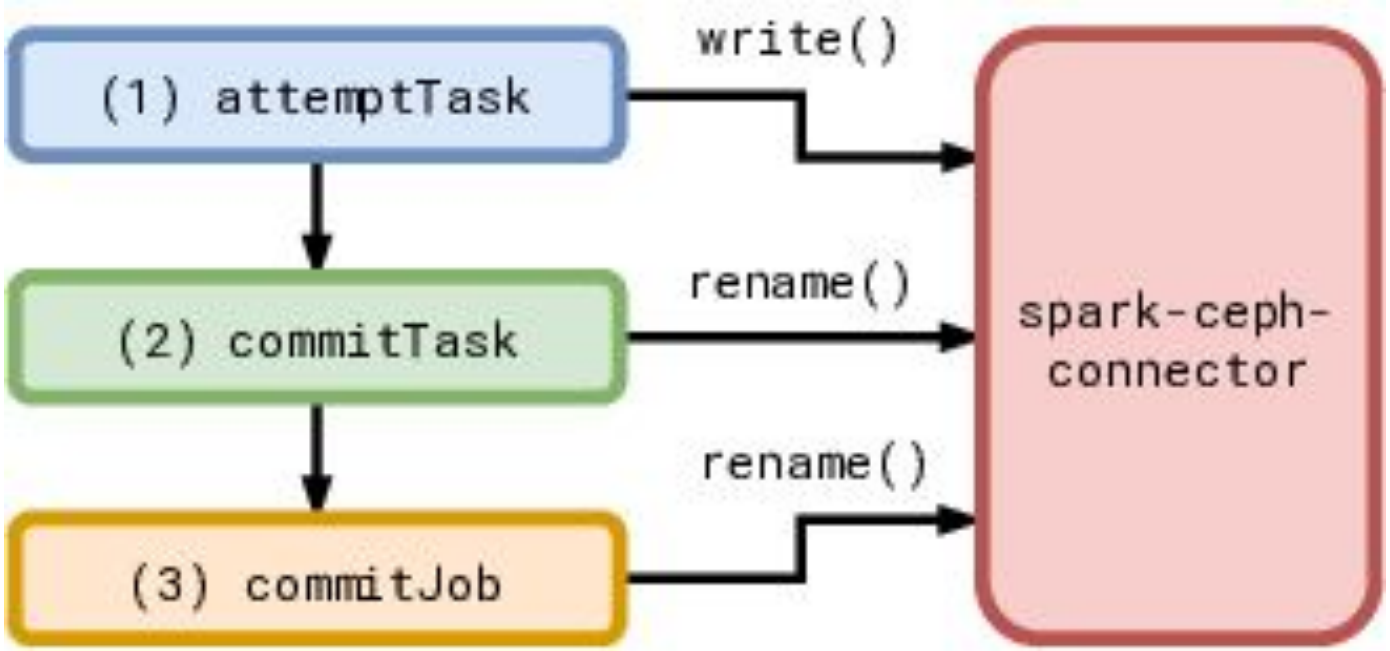
提案手法: FileOutputCommitterのコミットステップ

- 3つのコミットステップが実行される
  - (1) attemptTask() → write() メソッド
  - (2) commitTask() → rename() メソッド
  - (3) commitJob() → rename() メソッド



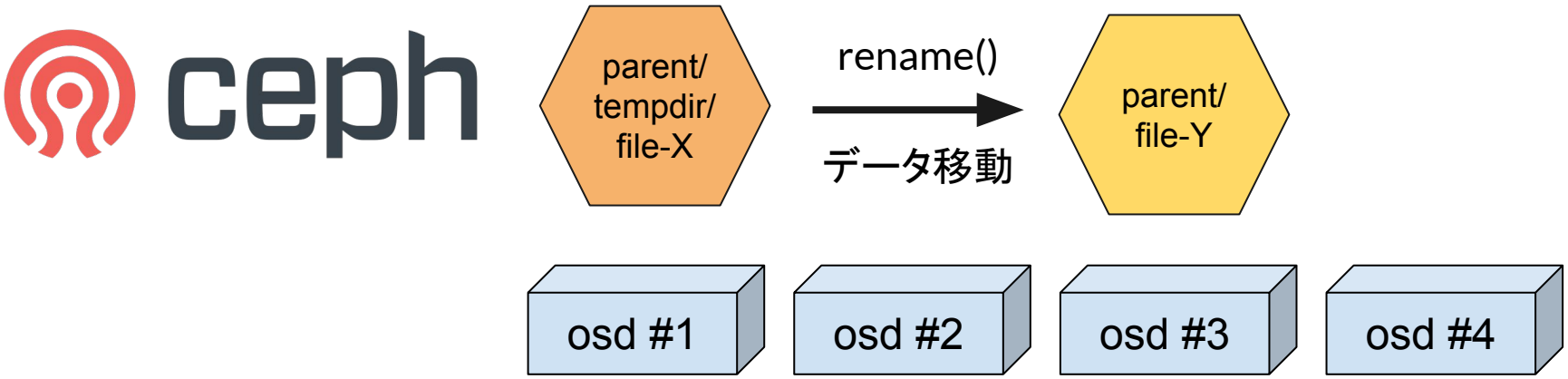
提案手法: FileOutputCommitterのコミットステップ

- rename() メソッド
  - HDFSなどのファイルシステム
    →ファイル名を変更するメタデータ操作だけの軽量な操作



提案手法: FileOutputCommitterのコミットステップ

- rename() メソッド
  - Cephはオブジェクト配置アルゴリズム RADOS がベース
    - Ceph の設計上、rename により、オブジェクト全体の保存場所の変更が必須
    - → オブジェクト全体の複製が発生

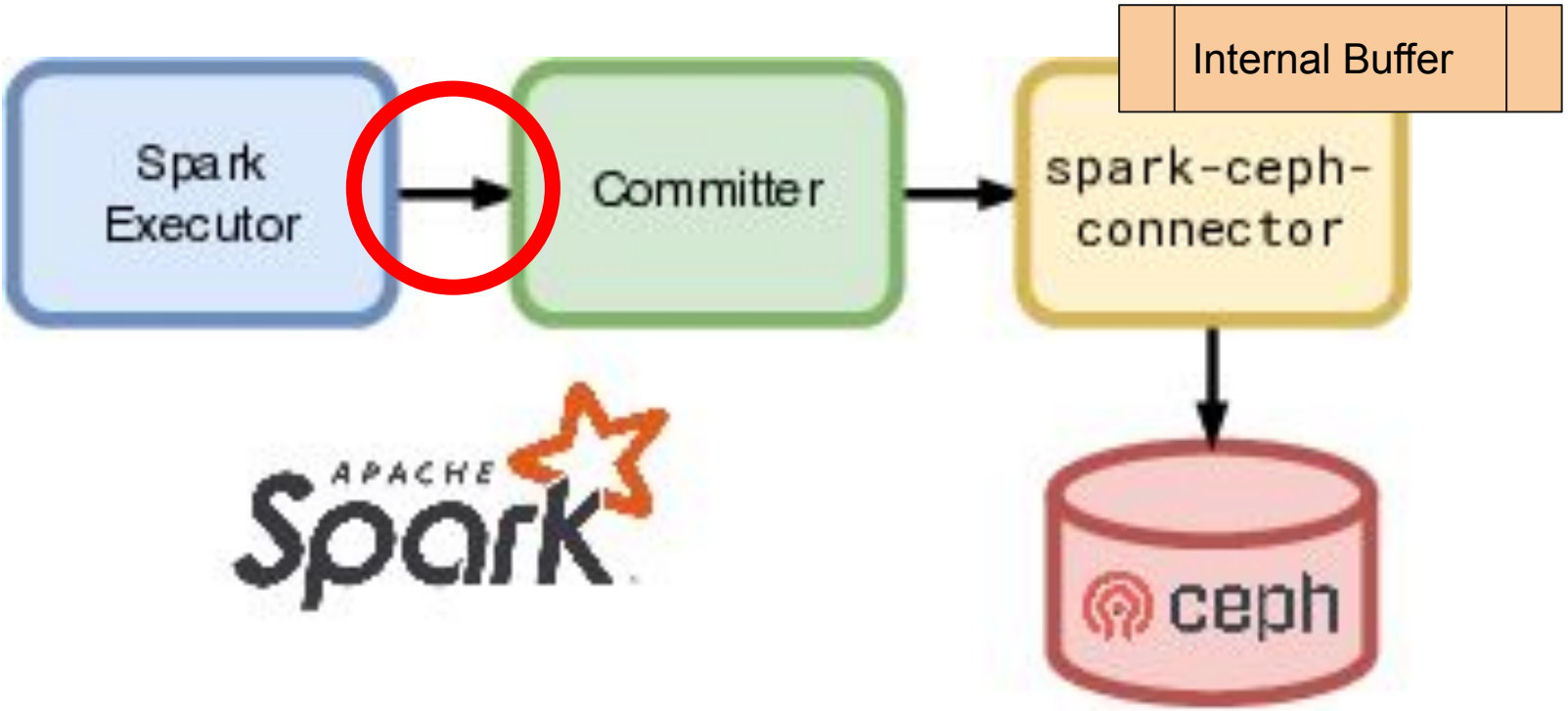


提案手法: オブジェクト形式の利用とバッファ拡張

- 転送データチャンクサイズが小さい問題
  - データの書き込みメソッドの変更
  - Javaネイティブのオブジェクト形式で保存
- spark-ceph-connector の内部バッファのサイズ拡張
  - 4 KiB → 4 MiB に拡張
  - バッファサイズの不足に対処

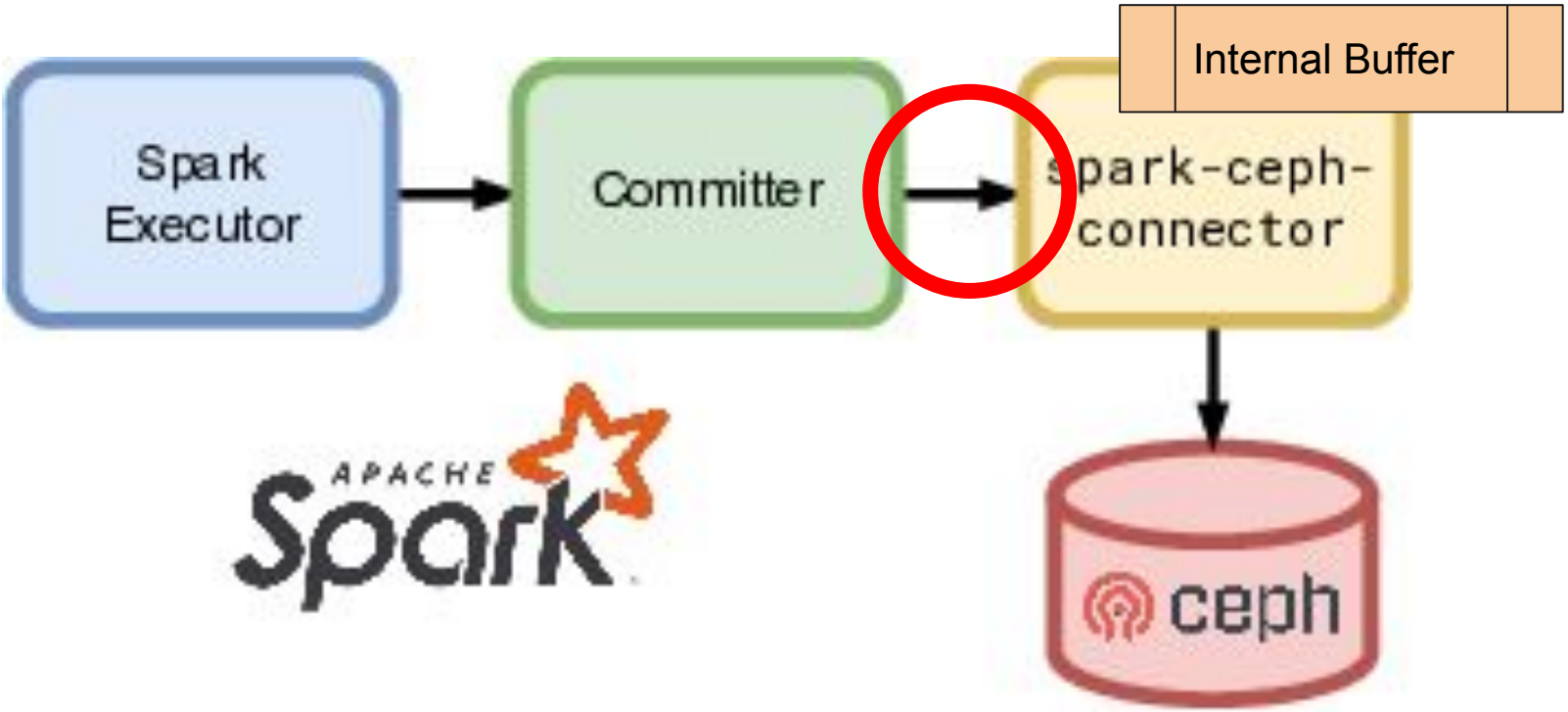
提案手法: FileOutputCommitter のコミット方法

- 1. Spark Executor → Committer の処理
  - Spark Executor:
    - データ処理によって生成されたデータI/Oの命令を発行する



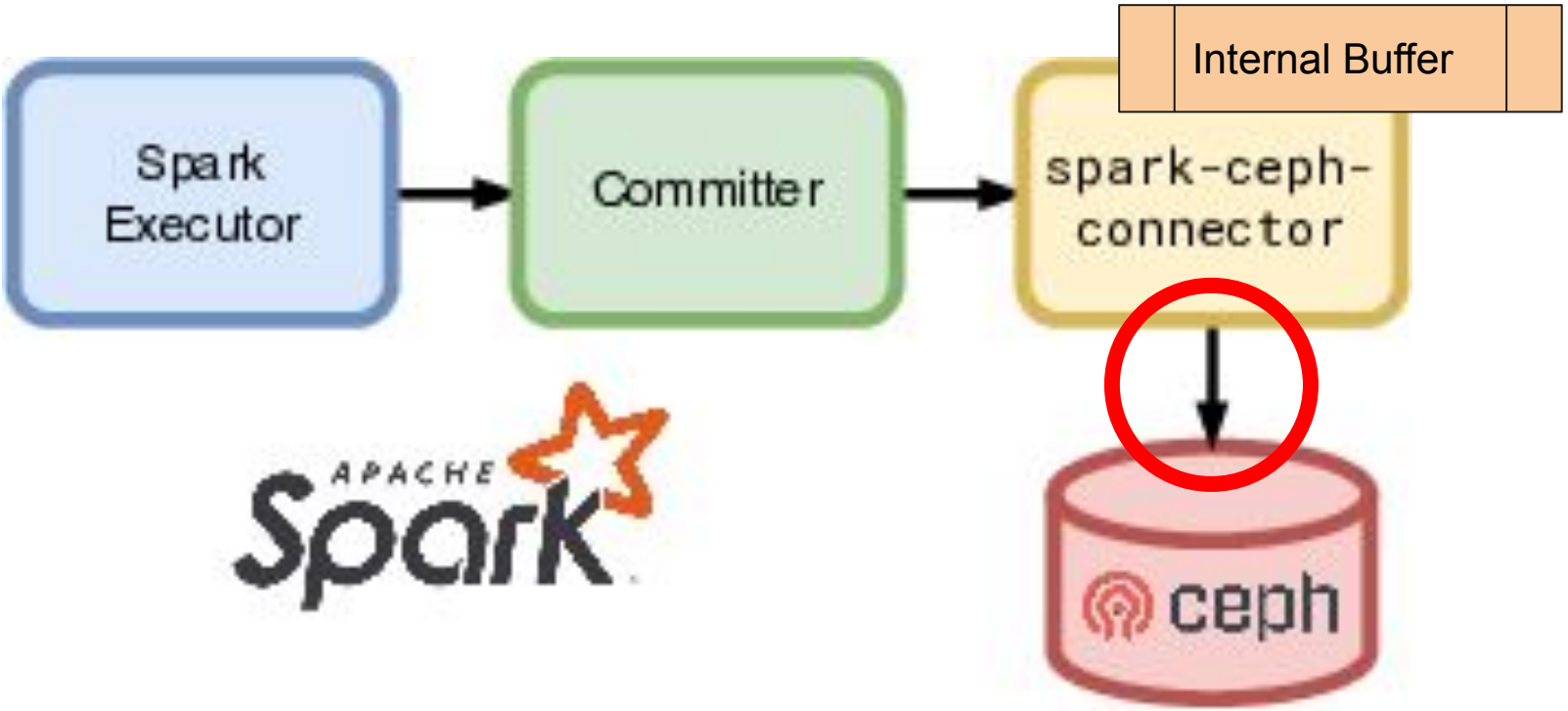
提案手法: FileOutputCommitter のコミット方法

- 2. Committer → コネクタの処理
  - Committer:
    - ストレージコネクタに対してデータを渡して書き込み処理を依頼



提案手法: FileOutputCommitter のコミット方法

- 3. コネクタ → ストレージの処理
  - コネクタがlibradosを使用してデータI/Oの処理を実行
  - Cephクラスタにオブジェクトを書き込み



提案手法: DataFrameを使用したシリアライズの利用

- DataFrameを使用したシリアライズの利用
  - 最適な書き込みデータのシリアライズ方法を検証
- シリアライズ形式
  - JSON形式
  - Apache ORC形式
  - Apache Parquet形式
- Apache ORC, Apache Parquet
  - データ解析用の列指向のシリアライズフォーマット
  - Apache ORC: "High-Performance Columnar Storage for Hadoop"
  - Apache Parquet: "efficient columnar data representation available to any project in the Hadoop ecosystem"

予備実験

1. 研究の背景
2. 課題と研究の目的
3. 提案手法
4. 予備実験
5. 実験結果
6. まとめ
7. 関連研究
8. 今後の課題

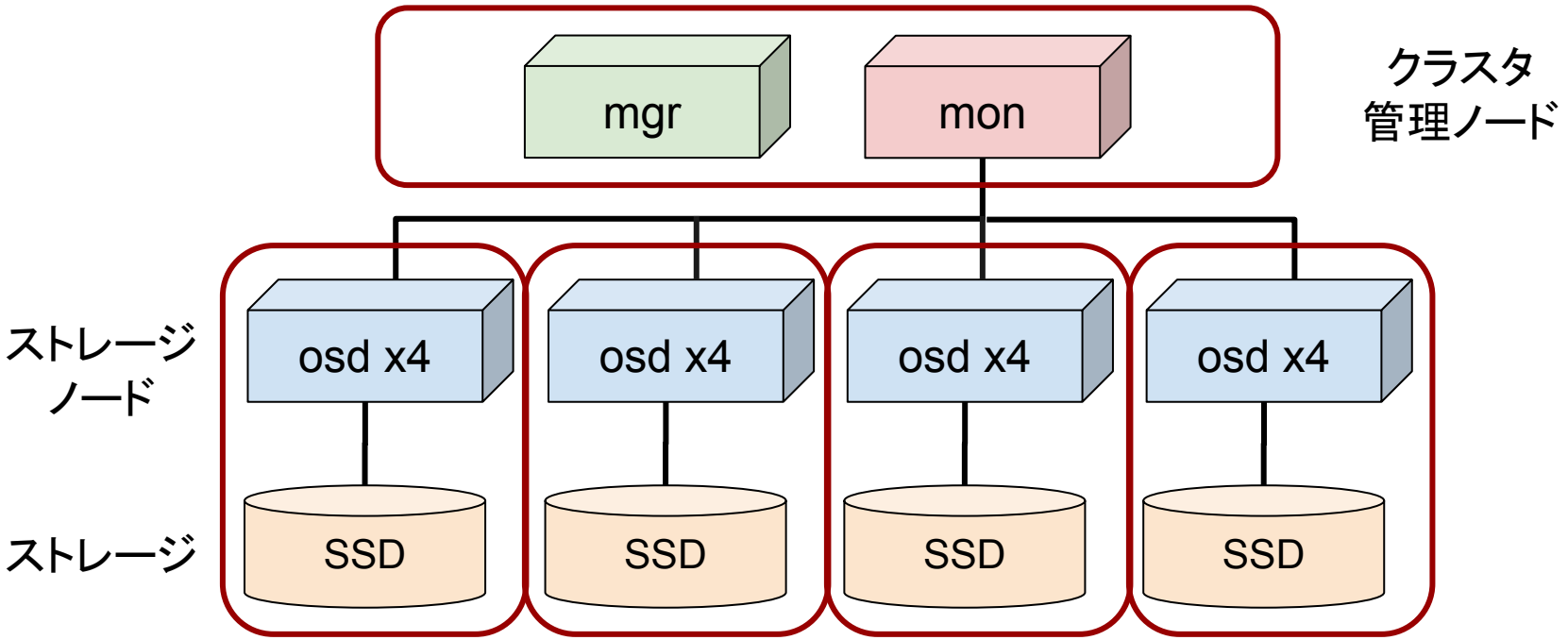
予備実験: 実験環境

- Ceph クラスタを構成する各ノードの環境

CPU	Intel Xeon CPU E5620 @ 2.40GHz x2
メモリ	DDR3-1333 ECC 4 GB x6 (24 GB)
ネットワーク	10 Gb Ethernet
ストレージ	RevoDrive 3 X2 SSD 240 GB (60GB x4)
OS	Ubuntu 20.04 LTS
Linux Kernel	Linux 5.4.0-42-generic (x86-64)
Ceph	ceph v15.2.3 (d289bbd) octopus (stable)

予備実験: クラスタの構成

- Ceph クラスタを5ノードから構成
  - 管理ノード (1ノード): mon, mgr プロセスを実行
  - ストレージノード (4ノード): osd プロセスを実行



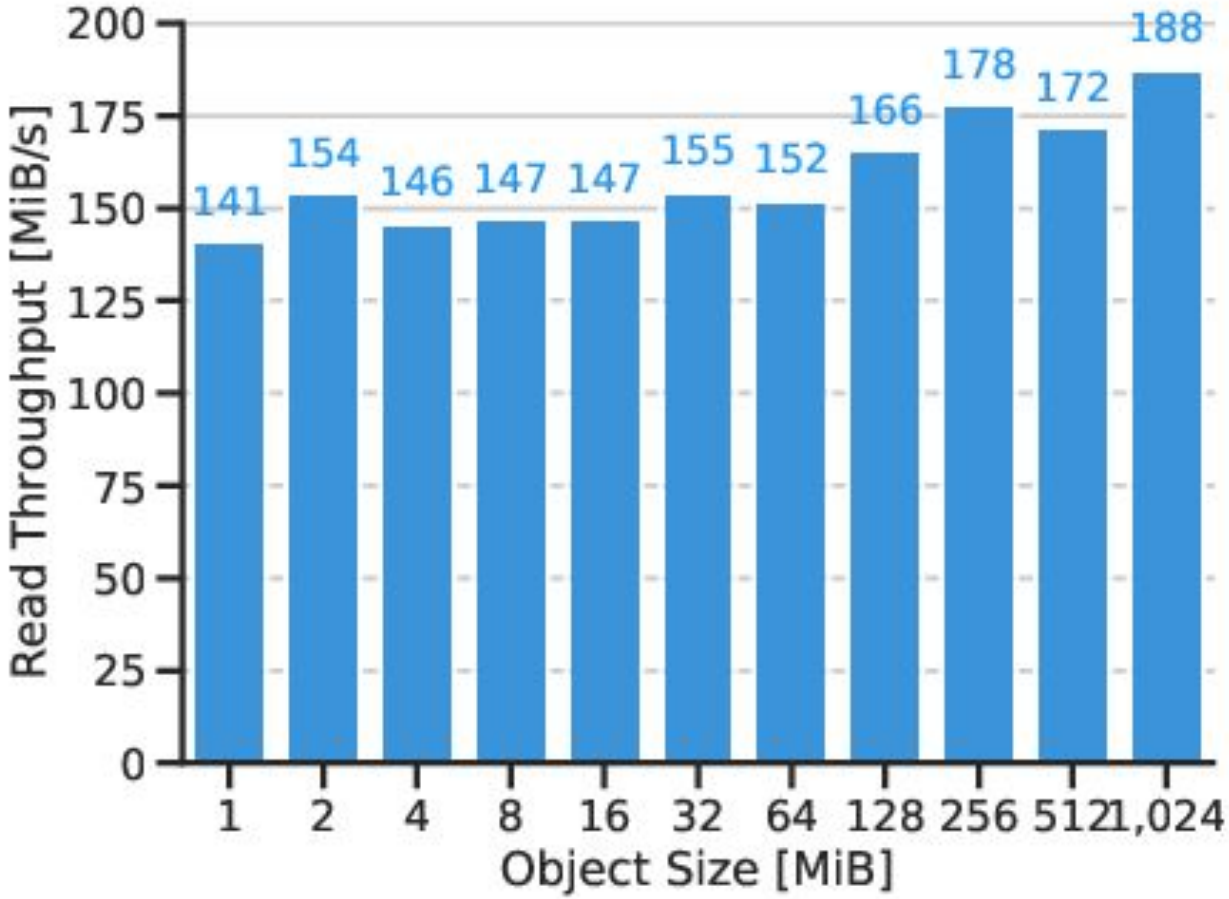
ceph

予備実験: ベース性能の測定

- Cephに標準で付属するRADOS Benchを使用して測定
  - RADOS Bench: librados を利用したRADOS オブジェクトの直接のI/O性能を測定
- シーケンシャルRead/Writeの性能を測定
- Read/Writeのオブジェクトサイズを1 MiB - 1,024 MiB に変化させて、実験環境のCephクラスタにおける性能を確認

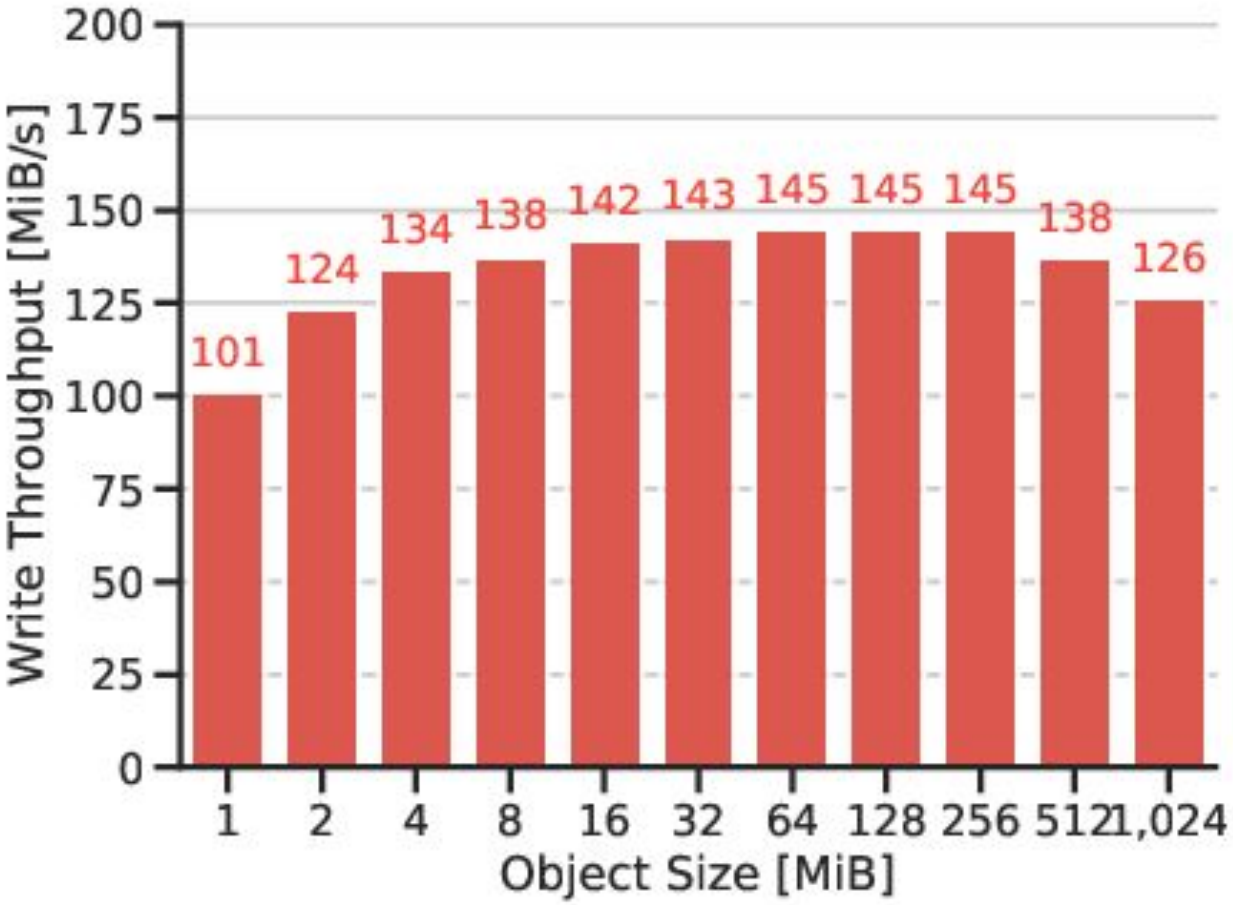
予備実験: 読み込みベース性能の測定

- 読み込み性能
  - 140 MiB/s を上回る安定した性能



予備実験: 書き込みベース性能の測定

- 書き込み性能
  - 約 100 MiB/s～約 145 MiB/sまでの性能を発揮



実験結果

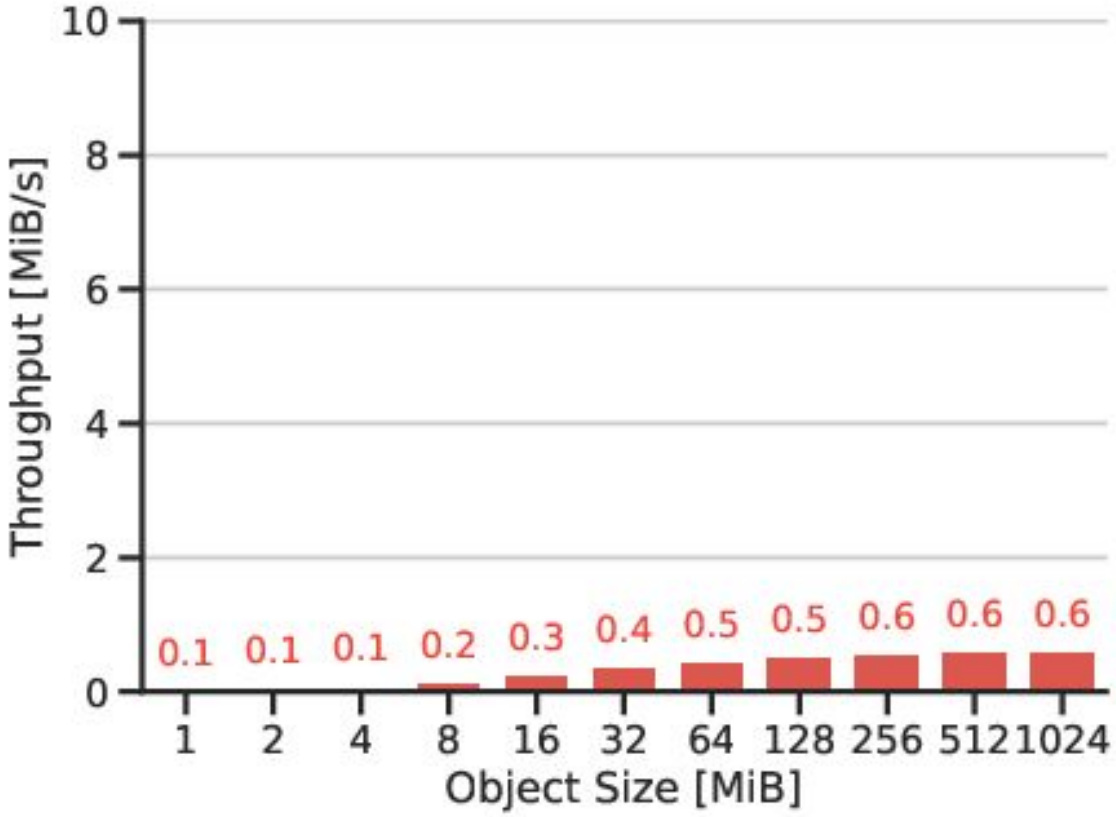
1. 研究の背景
2. 課題と研究の目的
3. 提案手法
4. 予備実験
5. 実験結果
6. まとめ
7. 関連研究
8. 今後の課題

実験 - 実験の方法

- Apache Sparkからspark-ceph-connector経由でCephクラスタにオブジェクトを書き込む性能を測定
- オブジェクトデータの準備
  - Ceph上にオブジェクトデータを準備
  - キャッシュ機能を利用して、Apache Spark上に書き込みデータをキャッシュ
  - キャッシュしたデータをspark-ceph-connector経由で Cephクラスタへ書き込み
- 生成したデータのオブジェクトサイズ
  - 1 - 1,024 MiB

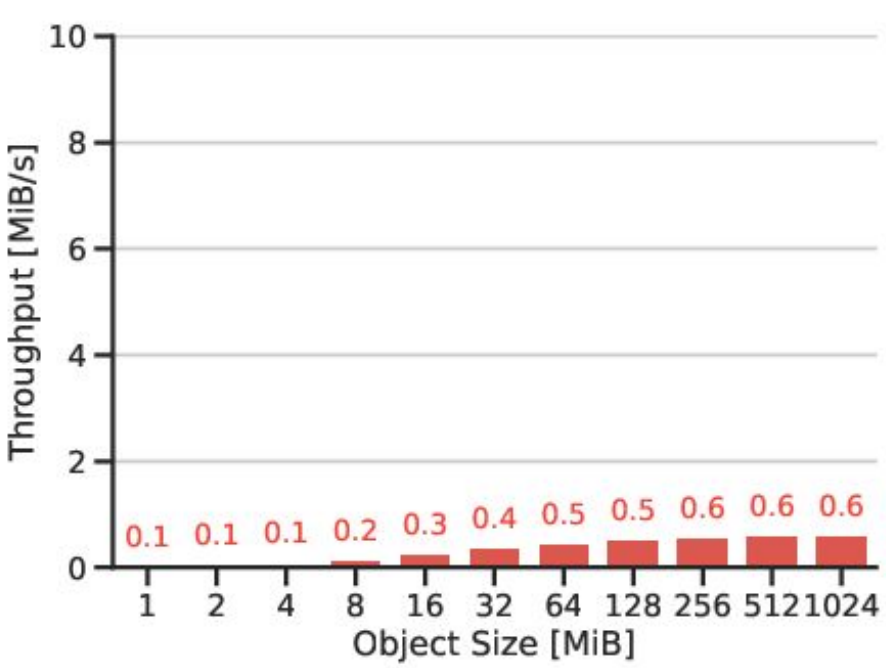
実験1: replicationとCommitterの変更による改善

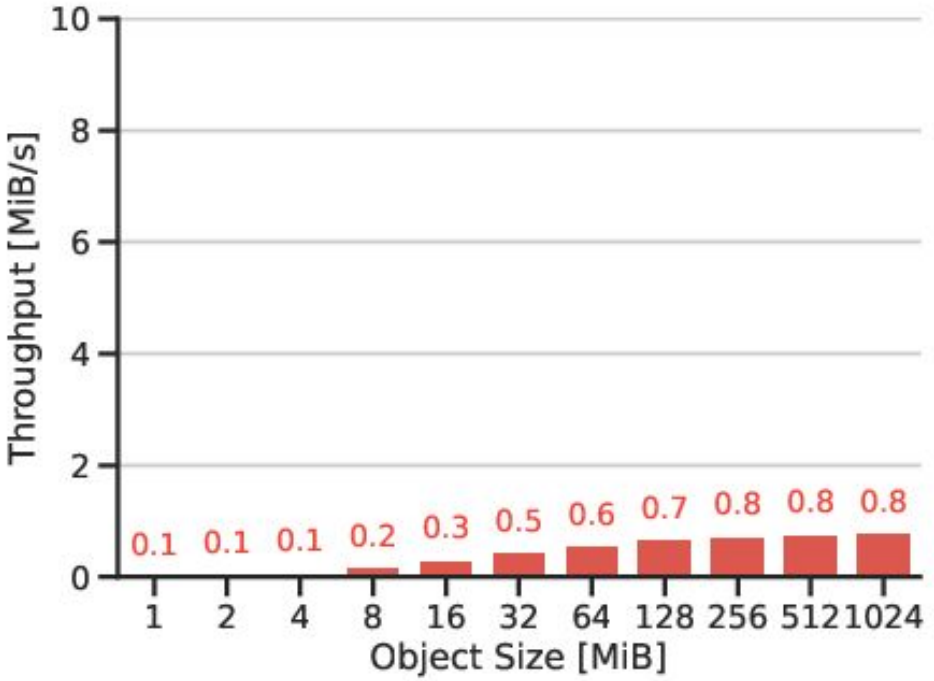
- ベース性能
  - replication factor: 3
  - FileOutputCommitter: アルゴリズムv1



replicationとCommitterの変更による改善

- replication factorの変更による改善
  - replication factor: 3→1
  - FileOutputCommitter: アルゴリズムv1

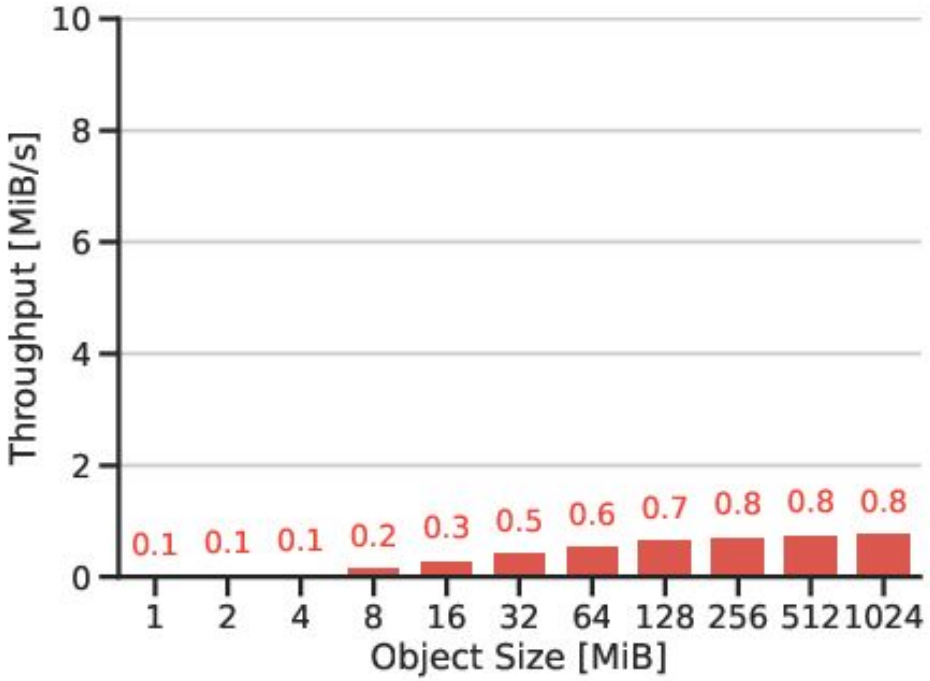


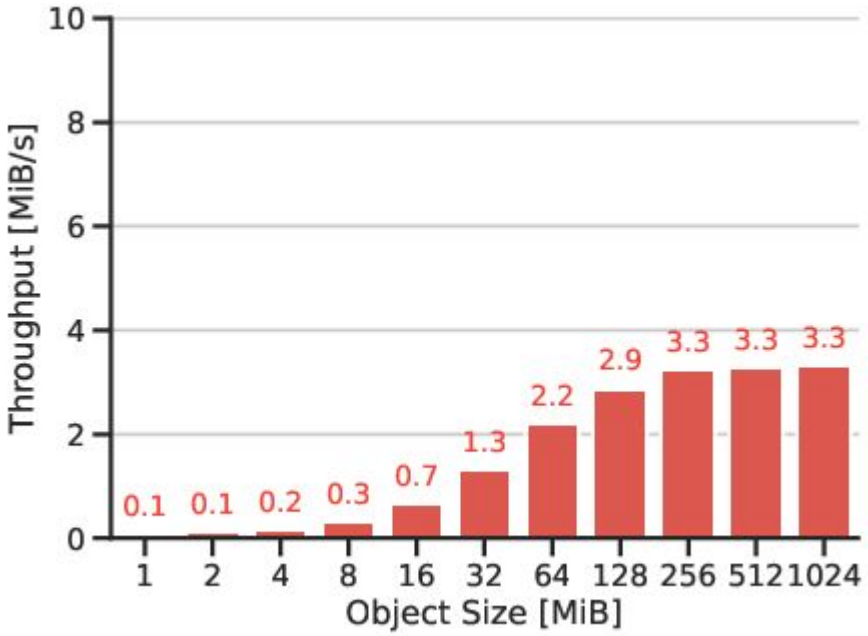


変更前 → 変更後

replicationとCommitterの変更による改善

- replicationとアルゴリズムの変更による改善
  - replication factor: 3 → 1
  - FileOutputCommitter: アルゴリズムv1 → v2

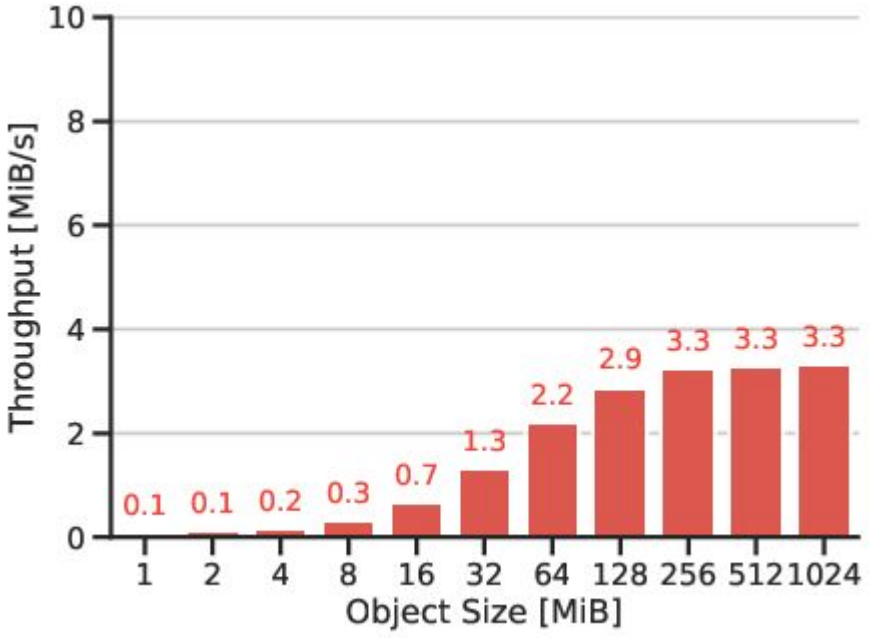


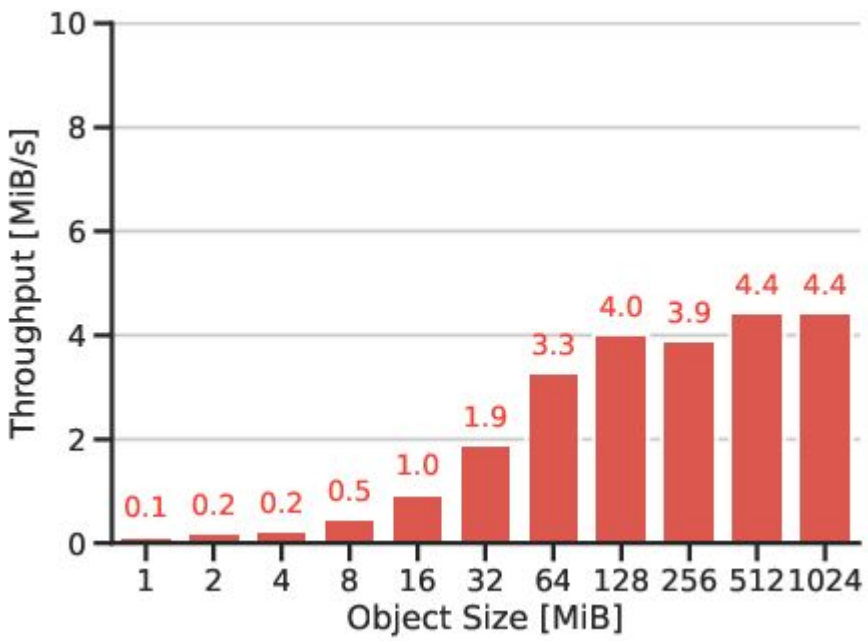


変更前 → 変更後

オブジェクト形式の利用とバッファ拡張による改善

- Javaネイティブオブジェクト形式の利用による改善
  - 約 3.3 MiB → 最大 4.4 MiB/s に改善
  - ストレージコネクタ内部バッファの飽和が明らかに

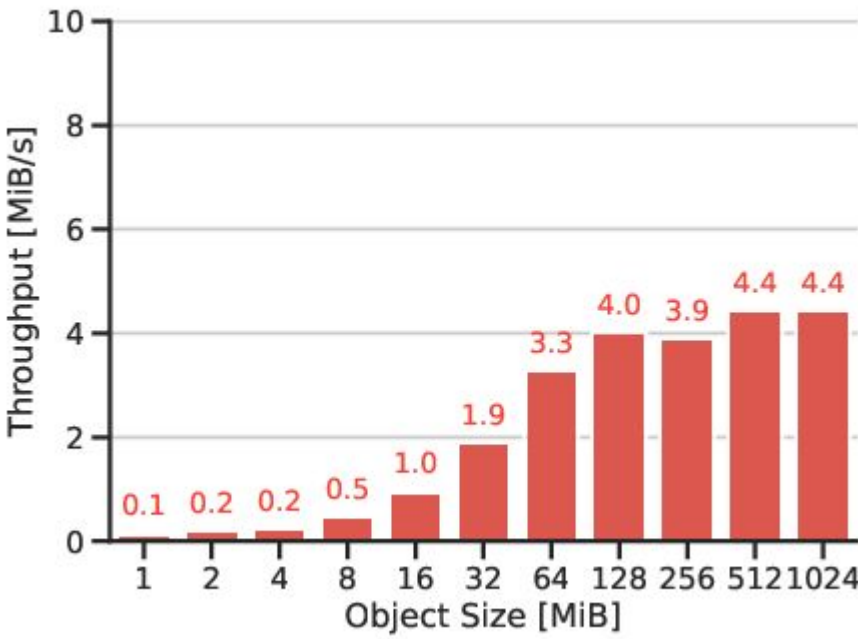


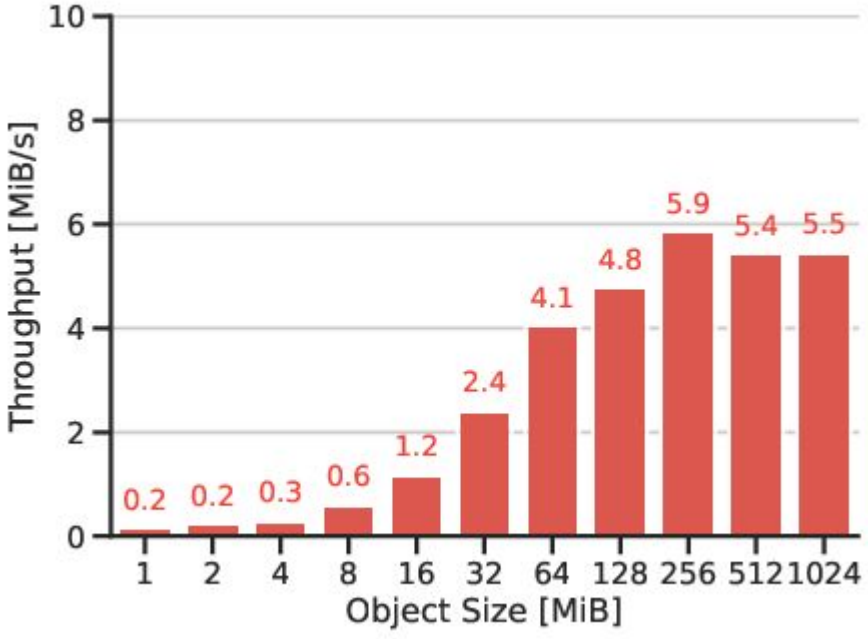


変更前 → 変更後

オブジェクト形式の利用とバッファ拡張による改善

- ストレージコネクタ内部バッファの拡張
  - バッファサイズ: 4 KiB → 4 MiB
  - バッファサイズが小さい影響が解消
  - 最大で約 1.8 倍の 5.9 MiB/s に性能向上

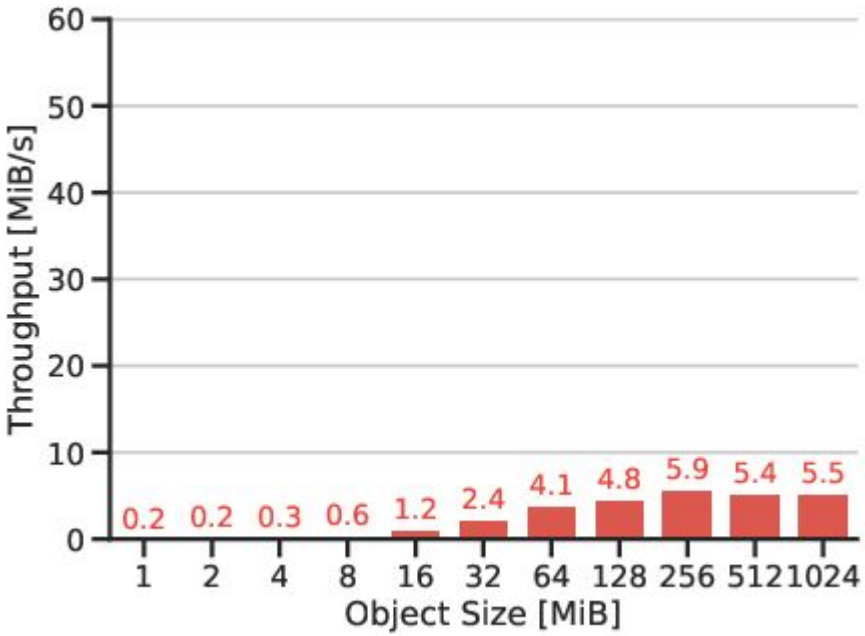




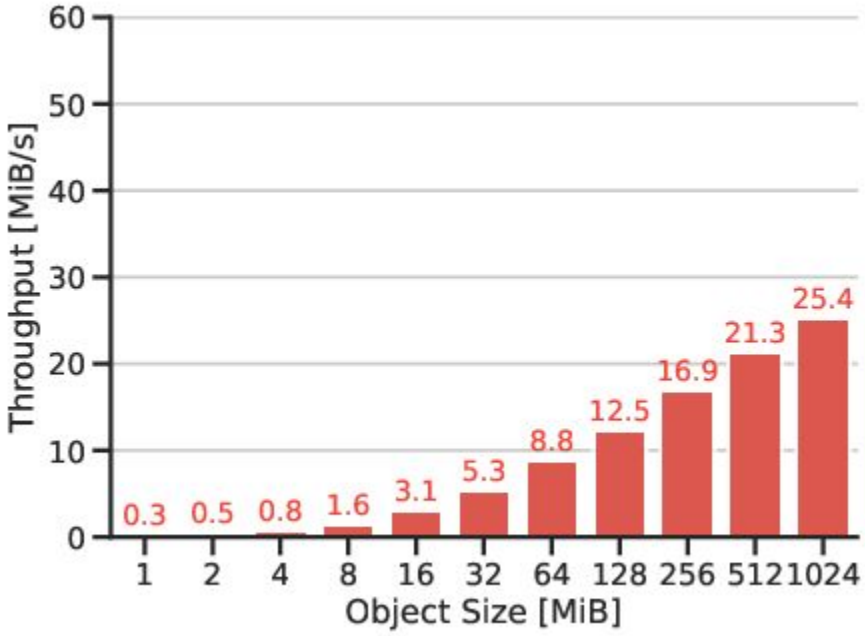
変更前 → 変更後

DataFrameを使用したシリアライズによる改善

- DataFrameのシリアライズ形式
  - JSON形式
  - テキスト形式だが、データチャンクが安定して 8 KiB で利用され、効率のよい書き込みが実現された



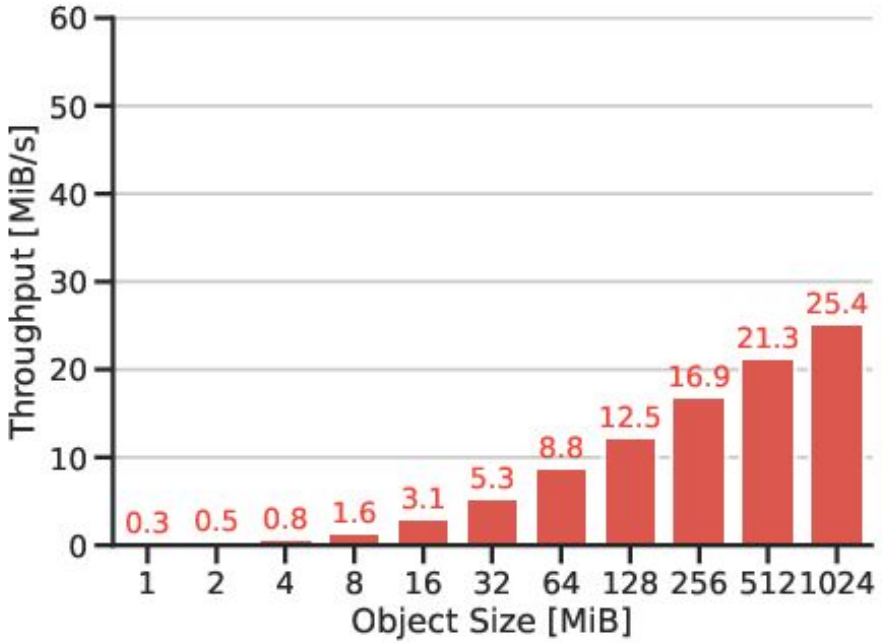
変更前

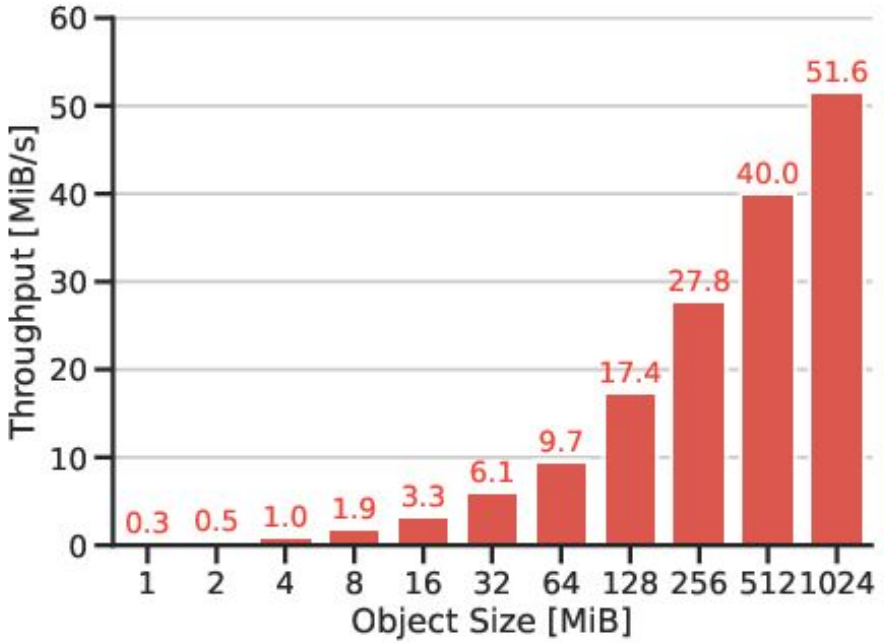


変更後

DataFrameを使用したシリアライズによる改善

- DataFrameのシリアライズ形式
  - DataFrameを使用したApache ORC形式
  - 最大で約 51 MiB/s

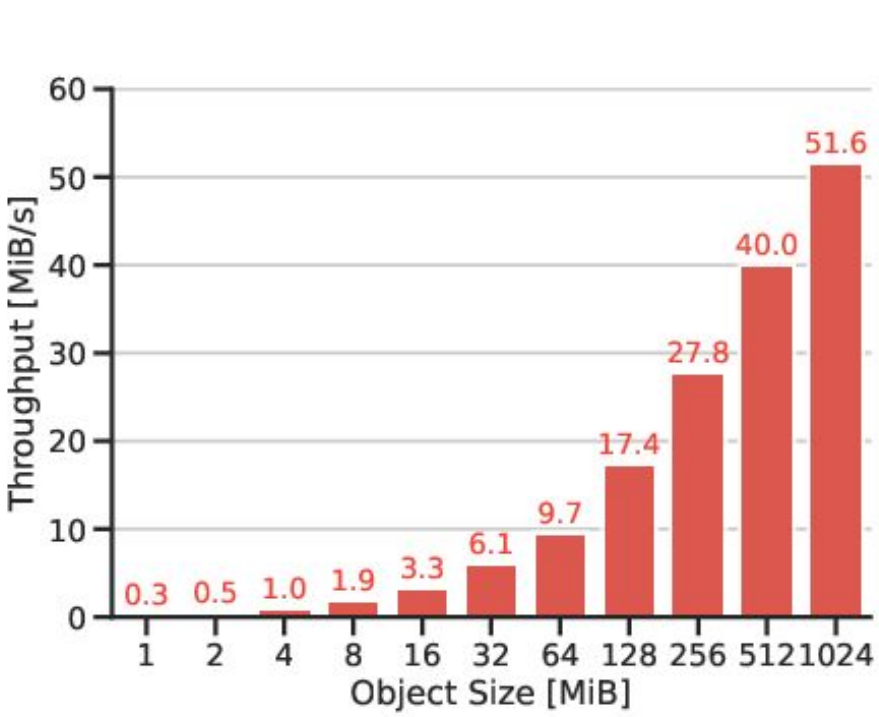


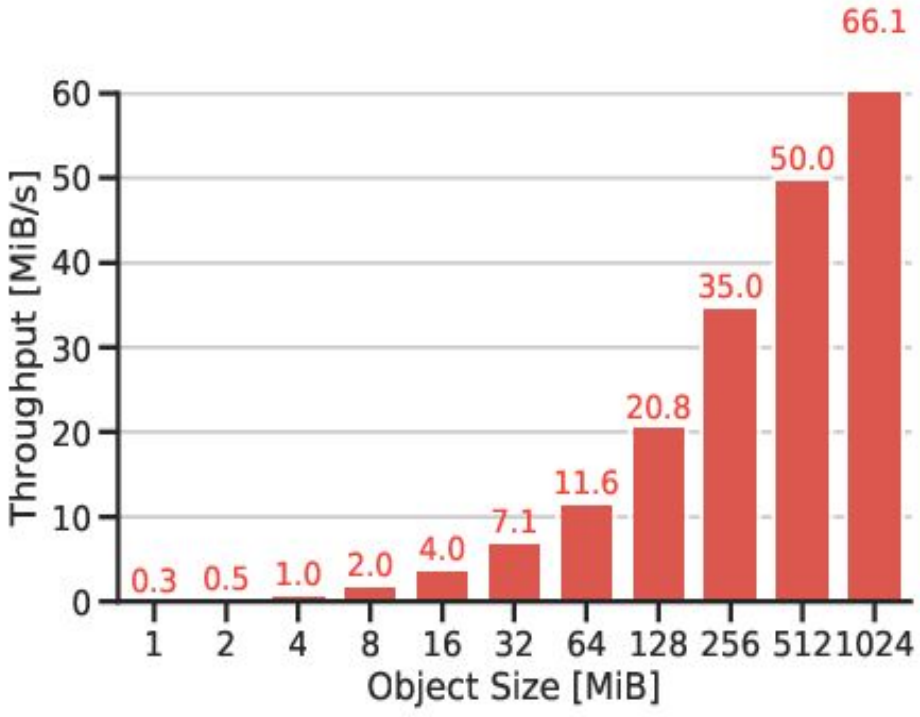


変更前 → 変更後

DataFrameを使用したシリアライズによる改善

- DataFrameのシリアライズ形式
  - Apache Parquet形式
  - ベース性能の約 110 倍の 66.1 MiB/s





変更前 → 変更後

まとめ

1. 研究の背景
2. 課題と研究の目的
3. 提案手法
4. 予備実験
5. 実験結果
6. まとめ
7. 関連研究
8. 今後の課題

まとめ

- ストレージコネクタ spark-ceph-connector は十分な書き込み性能が発揮できていなかった
- 本研究では
  - spark-ceph-connectorの書き込み性能が低い理由を分析
  - spark-ceph-connectorに対する改良を適用
  - 書き込み時の適切なシリアライズ方法を検討

まとめ

- これらの改善の結果:
- 128 MiBのオブジェクトサイズに対する書き込み性能
  - 約 0.6 MiB/s → 約 20.8 MiB/s
  - 約 x35 の性能向上
- 1,024 MiBのオブジェクトサイズに対する書き込み性能
  - 約 0.6 MiB/s → 最大で約 66.1 MiB/s
  - 約 x110 の大幅な改善
- 実用的な書き込み性能を発揮する方法が明らかになった

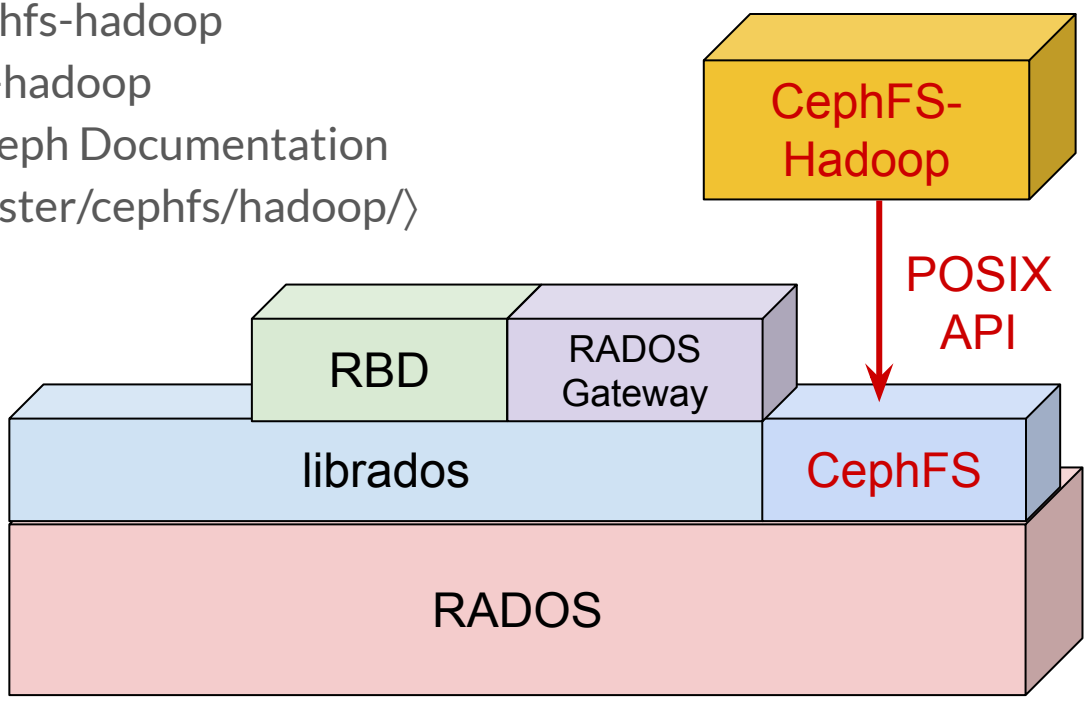
関連研究

1. 研究の背景
2. 課題と研究の目的
3. 提案手法
4. 予備実験
5. 実験結果
6. まとめ
7. 関連研究
8. 今後の課題

関連研究: CephFS-Hadoop プラグイン

- CephFS-Hadoop プラグイン
  - POSIX 準拠 CephFS インターフェイスを利用
  - Hadoop バージョン1.1系列が必要
  - CephFS のオーバーヘッドが多い

Ceph : ceph/cephfs-hadoop: cephfs-hadoop
https://github.com/ceph/cephfs-hadoop
Using Hadoop with CephFS — Ceph Documentation
⟨https://docs.ceph.com/docs/master/cephfs/hadoop/⟩



ceph

関連研究: zero-rename committer

- AWS S3オブジェクトストレージ向けのS3Aストレージコネクタのために開発されたCommitter
- 名前の通り、rename操作を行わないように改良されたオブジェクトストレージ向けのCommitter

Steve Loughran: A Zero-Rename Committer: Object-storage as a destination for Apache Hadoop and Spark , (online), available from ⟨https://github.com/steveloughran/zero-rename-committer (2020-08-20) ⟩

関連研究: ストレージコネクタStocator

- zero-rename committerと同様の改善を独自のアプローチで実装したストレージコネクタ
- Amazon S3やSwiftでの利用は考慮されているが、Ceph向けには開発されておらず、Stocator用のストレージプラグインを追加で開発する必要がある

Vernik, G., Factor, M., Kolodner, E. K., Michiardi, P., Ofer, E. and Pace, F.: Stocator: Providing High Performance and Fault Tolerance for Apache Spark Over Object Storage, 2018 18th IEEE / ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), IEEE, (online), DOI: 0.1109/ccgrid.2018.00073 (2018).

今後の課題

1. 研究の背景
2. 課題と研究の目的
3. 提案手法
4. 予備実験
5. 実験結果
6. まとめ
7. 関連研究
8. 今後の課題

今後の課題

- これまで: 読み込みと書き込みの基本的な性能を評価するためのマイクロベンチマークのみ
  - →より実用的なアプリケーションに近いワークロードを用いたベンチマークを実施し、ストレージコネクタの実用性を評価する必要がある
- Committer の改善
  - rename 処理に改善の余地がある
  - →S3Aストレージコネクターのzero rename committerと同等の機能を実装

Appendix

- ストレージコネクタは、GitHub上でApacheライセンスで公開
- shuuji3/spark-ceph-connector: 🌟Spark Ceph Connector: Implementation of Hadoop Filesystem API for Ceph
  https://github.com/shuuji3/spark-ceph-connector

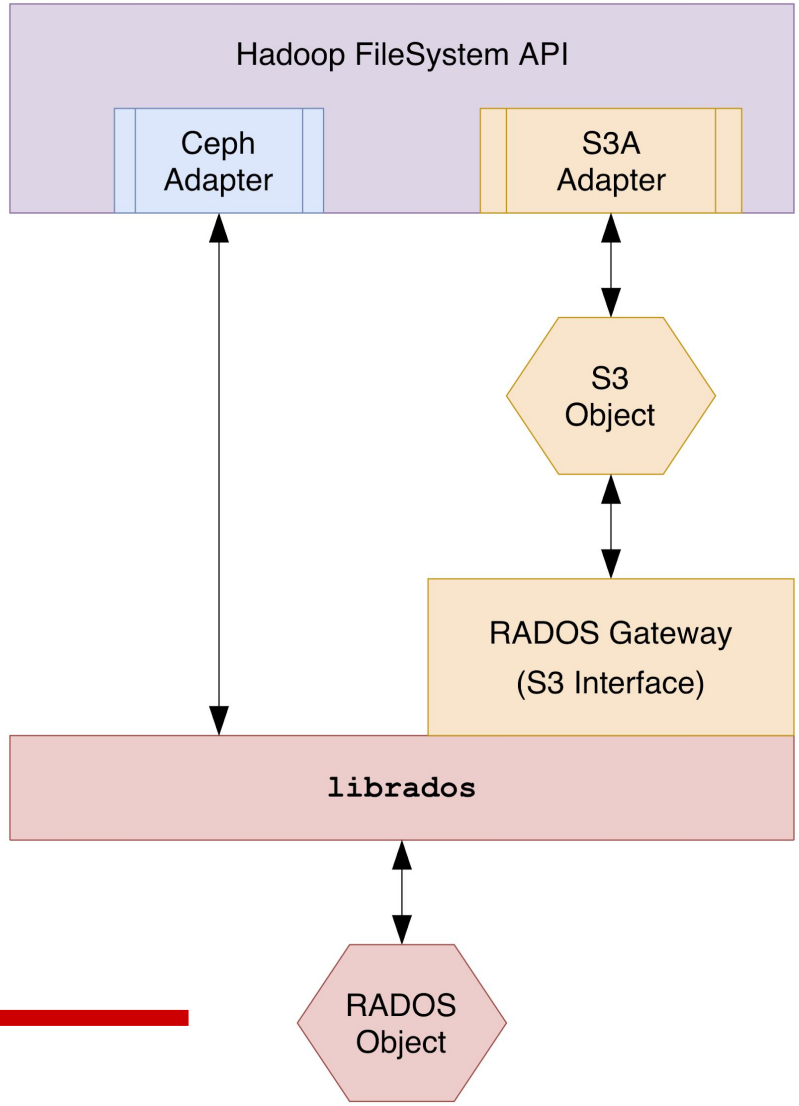
謝辞

本研究の一部は、筑波大学計算科学研究センターの学際共同利用プログラム（Cygnus）、国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）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。

Appendix

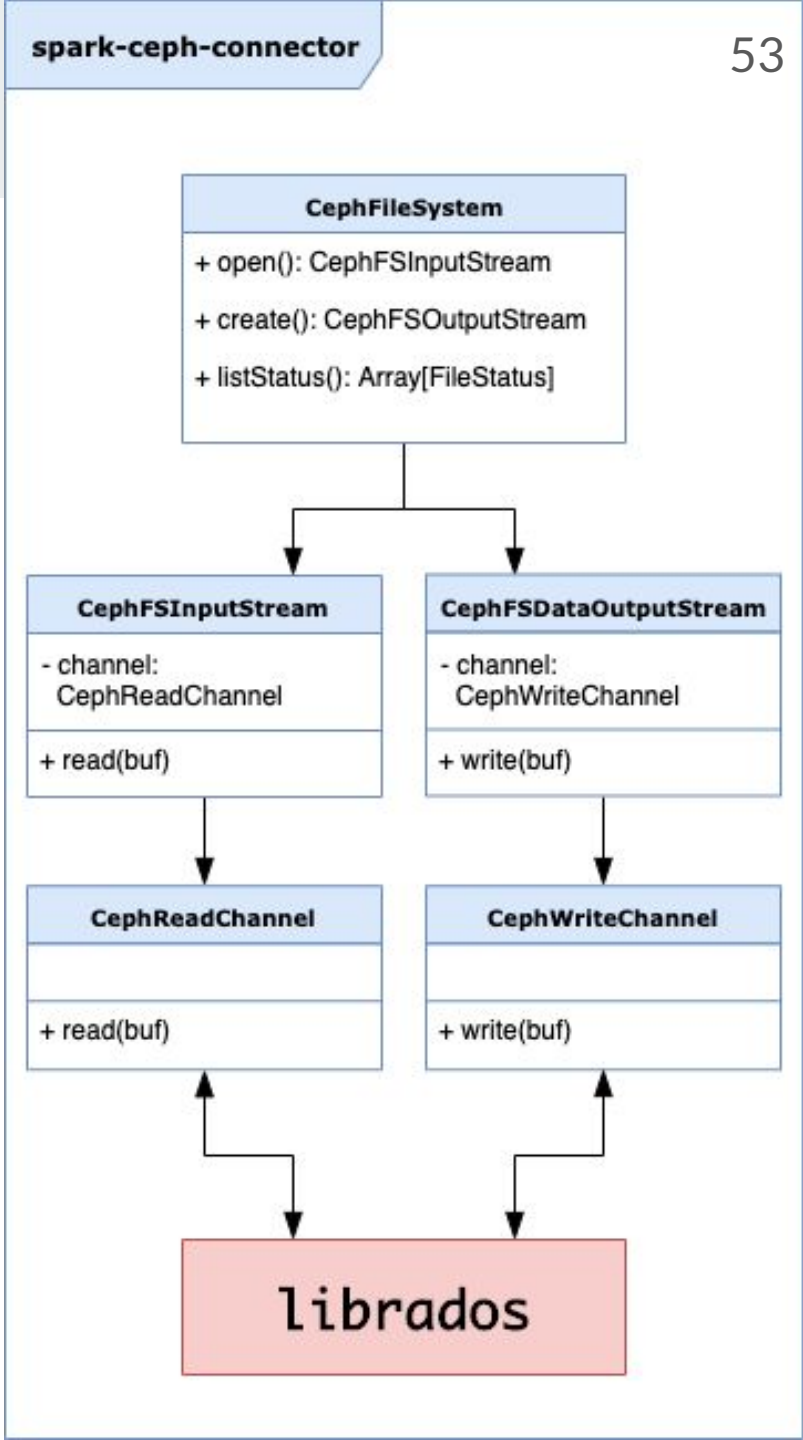
研究の目的 - 既存手法との比較

- 現状では S3A を利用 (右)
  - S3A は AWS のために開発
  - RADOS Gateway との 2重のデータ変換によるオーバヘッド
- 本研究が提案する Ceph コネクタ (左)
  - `librados` を利用
  - Ceph ネイティブのオブジェクトへのアクセスを行う



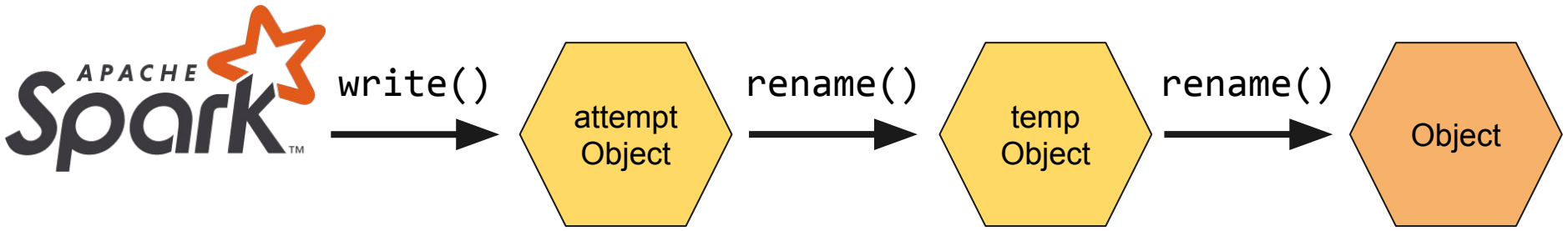
設計と実装 - コネクタ実装クラス

- hadoop.fs.FileSystem を継承した CephFileSystem
- Read:
  - CephInputStream
  - CephReadChannel
- Write:
  - CephOutputDataStream
  - CephWriteChannel
- 実装内部では Java の New I/O ライブラリを活用



実験と手法 - 性能劣化の原因 (1)

- 原因1: Apache Spark によるデータ保存時の書き込み方法と Ceph の CRUSH アルゴリズムの相性の悪さ
- Spark のデータ書き込みプロセス
  - (1) 分散タスクが `attempt` というファイルを書き込む
  - (2) データの書き込みの成功後、当該タスクごとにファイルシステム上の一時ファイルに `rename` する
  - (3) すべてのタスクが成功したら、一時ファイルを最終的な書き込み場所に移動するために `rename` を実行



実験と手法 - 性能劣化の原因 (1)

- 1 MiB のオブジェクト Write 時の Ceph クラスタ内の I/O の内訳 (計測範囲: 0s - 100s)
- Write だけでなく約 1/4〜1/3 もの Read が含まれる