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

2019/09/20(金) 第171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. 研究の目的
3. 関連研究
4. 設計と実装
5. 実験と手法
6. 結果と考察
7. まとめと今後の課題

研究の背景

1. 研究の背景
2. 研究の目的
3. 関連研究
4. 設計と実装
5. 実験と手法
6. 結果と考察
7. まとめと今後の課題

研究の背景 - オブジェクトストレージの普及

- 科学と産業の両分野おける大規模なデータ処理
  - 価値のあるデータを発見しようとする試み
- オブジェクトストレージが活用
  - スケーラビリティが高い
  - 大容量のデータをより低コストで長期間保管

研究の背景 - 様々なオブジェクトストレージ

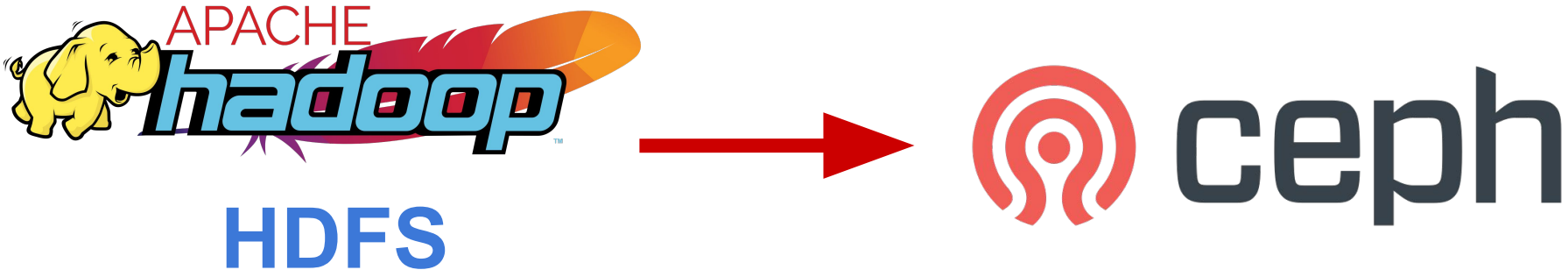
- 様々なオブジェクトストレージ
  - パブリッククラウド
    - AWS S3、Google Cloud Storage (GCS) など
  - オープンソース・ソフトウェア
    - **Ceph**、Swift、MinIO、DAOS (Intel) など
- POSIX 準拠ストレージの欠点を補うストレージとしてオブジェクトストレージが活用

研究の背景 - 分散オブジェクトストレージ Ceph

- **Ceph** (Sage A. Weil, 2006, OSDI '06)
  - Linux Foundation 傘下の Ceph Foundation で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
  - コア技術は RADOS オブジェクトストアと CRUSH アルゴリズム
    - MDS が不要、高いスケーラビリティを持つ
  - 利用例:
    - OpenStack 上での仮想ディスクの提供
    - CERN 等での大規模な科学実験データの格納

研究の背景 - 分散オブジェクトストレージ Ceph

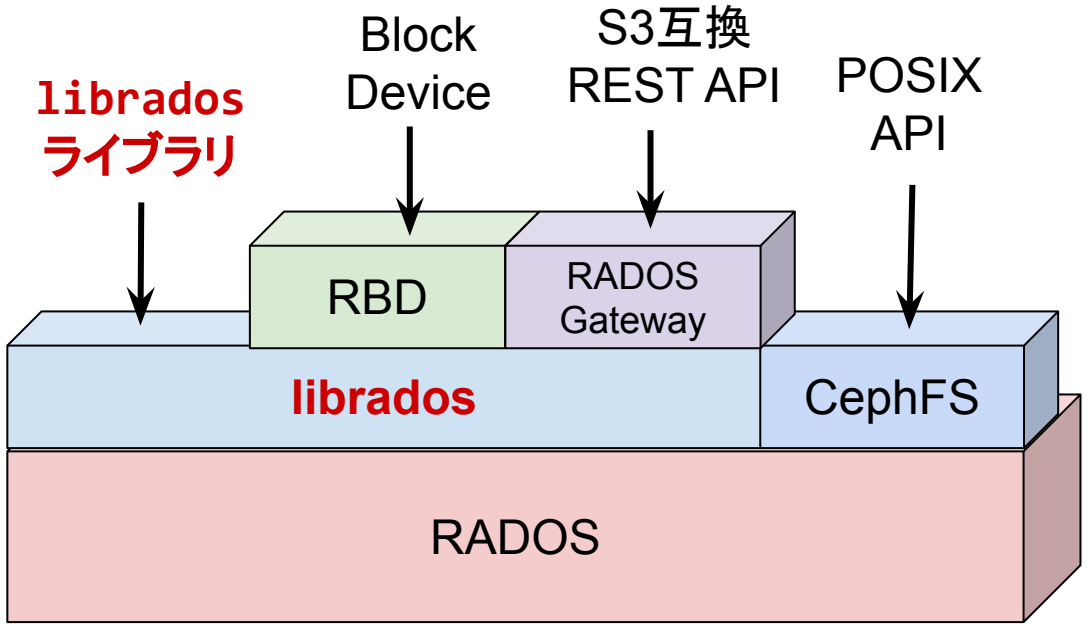
- Hadoop の HDFS にはスケーラビリティの問題が存在 [1]
- 大規模データの保管場所として HDFS の代わりにオブジェクトストレージ Ceph を活用する流れ[2]



[1] Konstantin, V. Shvachko. "HDFS scalability: The limits to growth." USENIX; login 35.2 (2010).
[2] Why Spark on Ceph?
https://www.redhat.com/ja/blog/why-spark-ceph-part-1-3

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)**



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

研究の背景 - 大規模データの処理基盤

- 大規模なデータセットの処理基盤
  - Apache Hadoop の MapReduce
  - **Apache Spark**

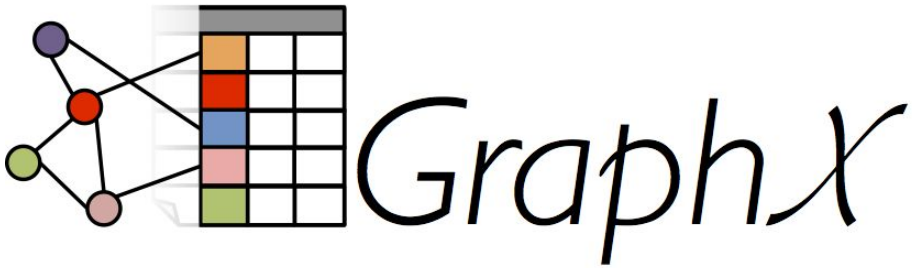
研究の背景 - Apache Spark

- **Apache Spark**
  (Zaharia et al., 2012, USENIX NSDI '12)
- MapReduce が苦手とする反復的な処理が高速
- データ解析の需要が高まってきていることにより広い分野で活用
- AWS EMR や Google Cloud DataProc など、主要クラウドプロバイダでも Apache Spark のマネージドサービスが普及

研究の背景 - Apache Spark

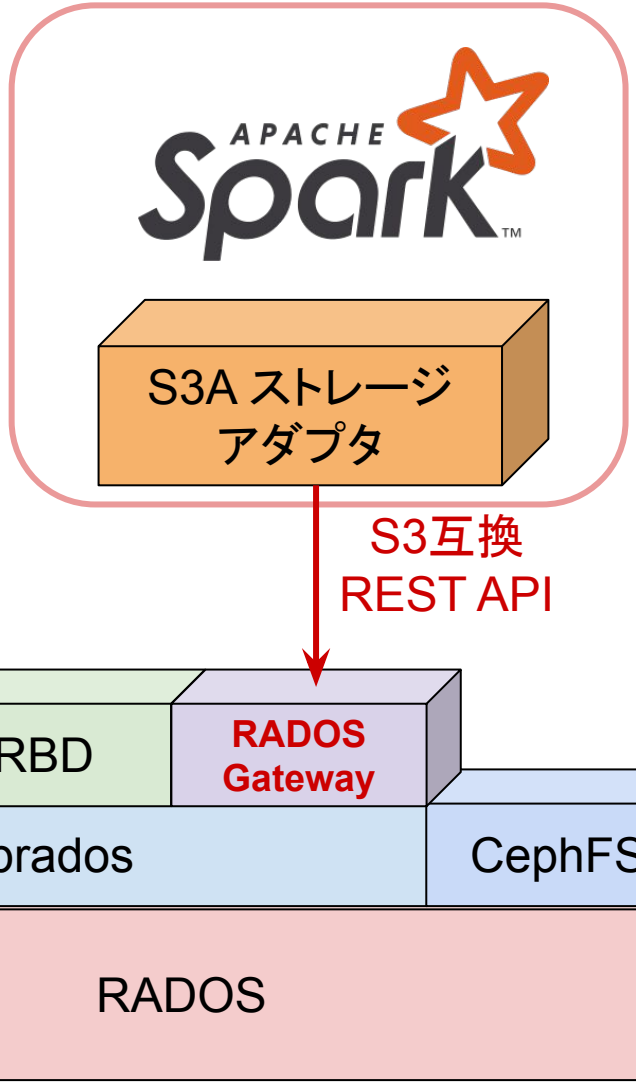
- 充実した標準ライブラリ
  - MLlib による機械学習
  - Spark Streaming によるリアルタイムデータストリーム処理
  - GraphX によるグラフデータ処理

MLlib



研究の背景 - 現状の問題

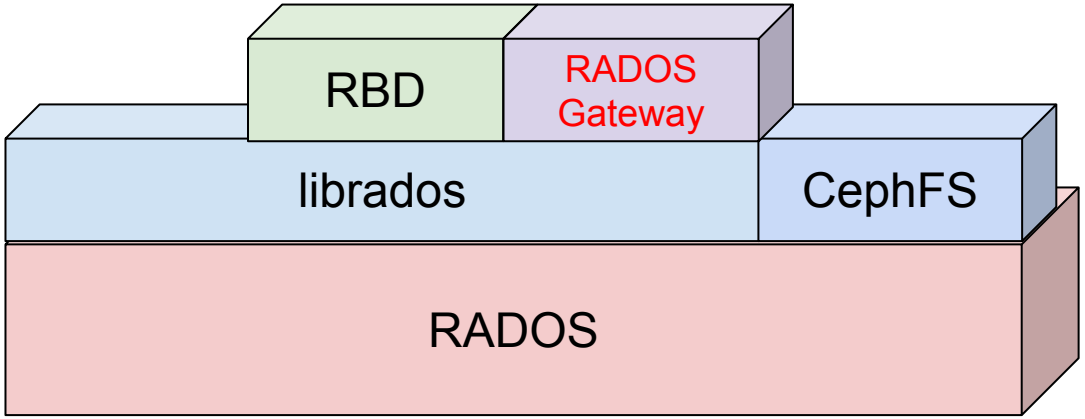
- AWS S3A アダプタ
  + Ceph RADOS Gateway
- 2重にオブジェクトの
  データ変換が行われることに
  よるオーバヘッドの問題
- S3Aの互換性によるユー
  ザービリティの問題



ceph

研究の背景 - 現状の問題

- RADOS Gateway のオーバーヘッドの存在 [3,4,5]
  - 特に、オブジェクト数が多くなった時に問題



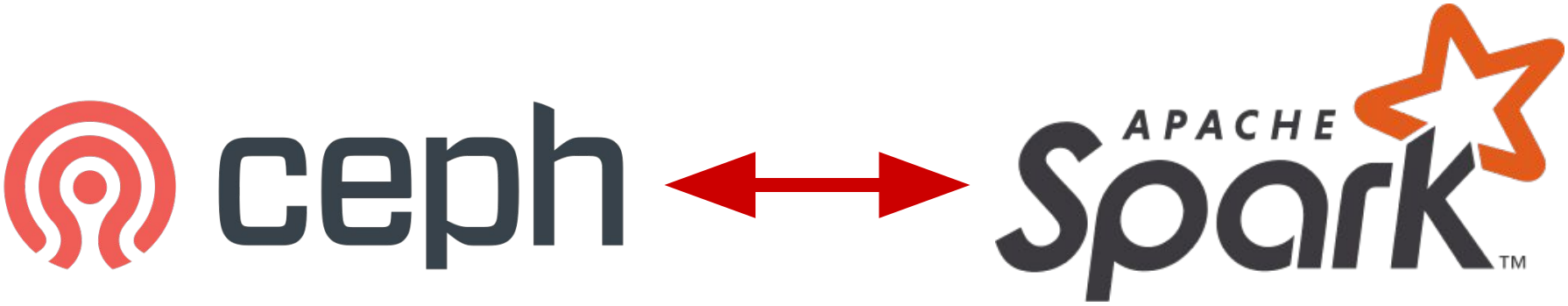
[3] [ceph-users] Luminous | PG split causing slow requests
http://lists.ceph.com/pipermail/ceph-users-ceph.com/2018-February/024984.html
[4] [ceph-users] All OSD fails after few requests to RGW
http://lists.ceph.com/pipermail/ceph-users-ceph.com/2017-May/017950.html
[5] [ceph-users] RadosGW performance degradation on the 18 millions objects stored.
http://lists.ceph.com/pipermail/ceph-users-ceph.com/2016-September/012983.html

研究の目的

1. 研究の背景
2. 研究の目的
3. 関連研究
4. 設計と実装
5. 実験と手法
6. 結果と考察
7. まとめと今後の課題

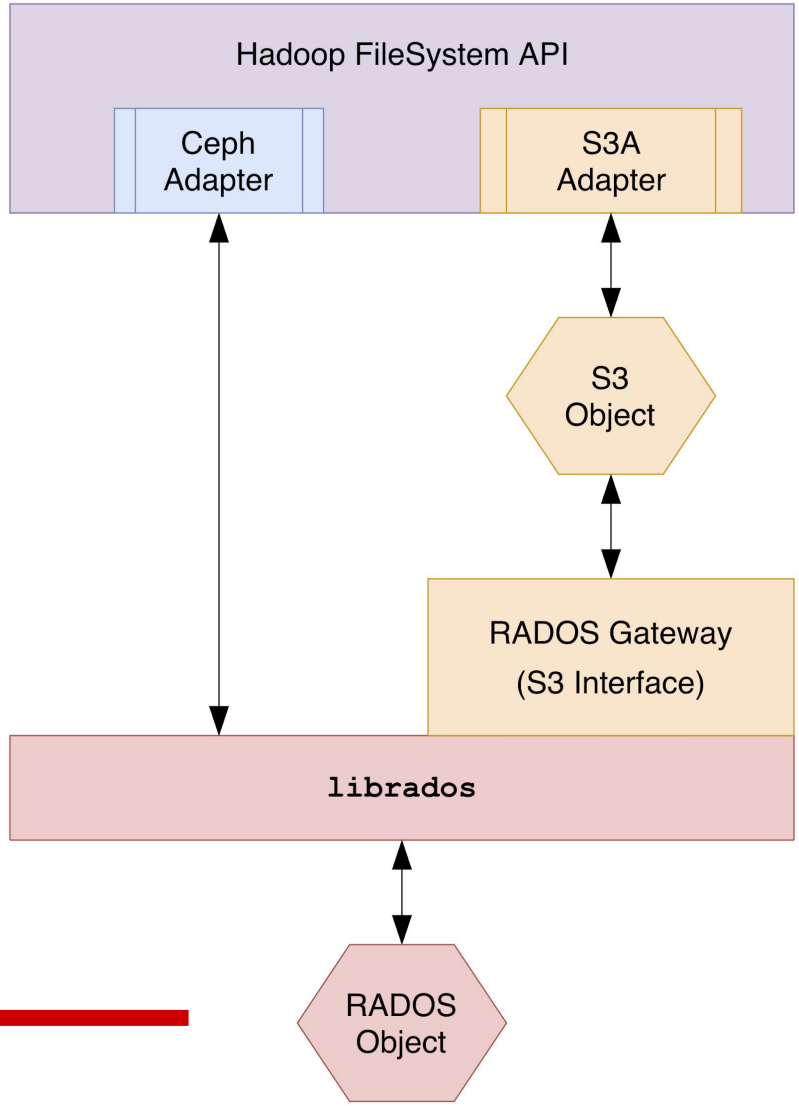
研究の目的 - 提案手法

- 目的: Apache Spark から Ceph の大規模データにより高速にアクセス・利用可能にする
  - Ceph の特性を活用できるストレージコネクタを設計・実装
  - 実装したコネクタの性能を測定し、その特性を明らかにする



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



発表の構成

1. 研究の背景
2. 研究の目的
3. 関連研究
4. 設計と実装
5. 実験と手法
6. 結果と考察
7. まとめと今後の課題

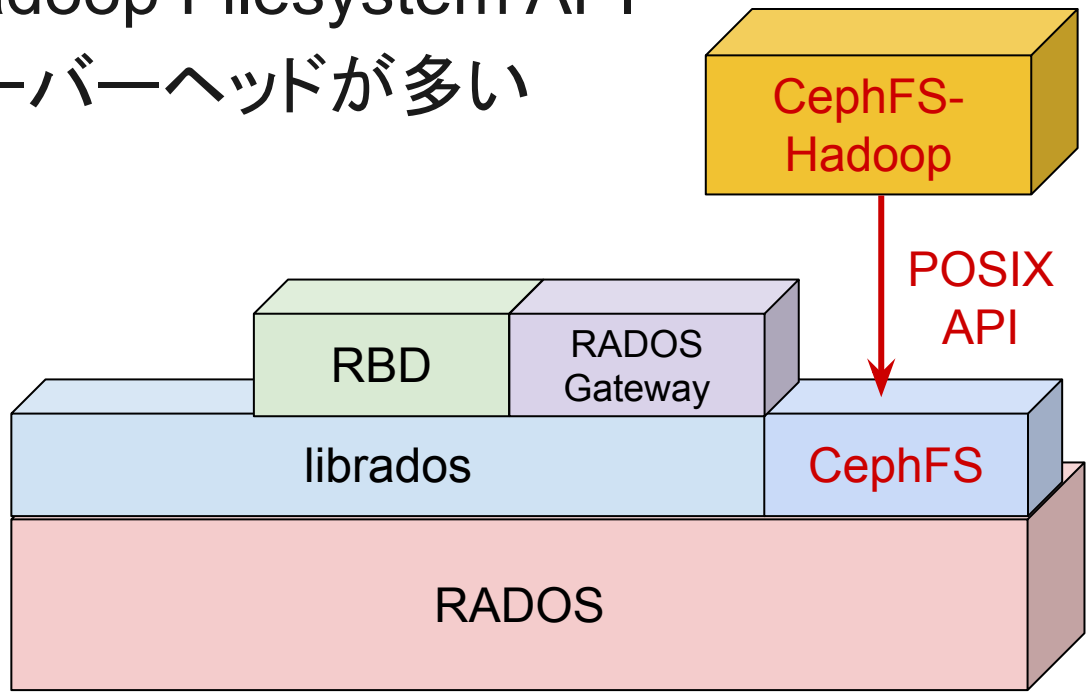
関連研究 - クラウドストレージのコネクタ

- Apache Spark からクラウドストレージ上の
  オブジェクトストレージへアクセスするコネクタ
  - Google Cloud Storage ストレージコネクタ [6,7]
  - AWS S3 のためのストレージコネクタ S3A [8]
    - Apache Spark で Ceph にアクセスする場合に
      利用されている
    - RADOS Gateway + S3A の2重の変換による
      オーバーヘッドが存在

[6] Google: bigdata-interop/gcs at master : ⟨https://github.com/GoogleCloudPlatform/bigdata-interop/tree/master/gcs⟩
[7] Google: Cloud Storage コネクタ | Cloud Dataproc | Google Cloud ⟨https://cloud.google.com/dataproc/docs/concepts/ connectors/cloud-storage?hl=ja⟩
[8] The Apache Software Foundation : Apache Hadoop Amazon Web Services support – Hadoop-AWS module: Integration with Amazon Web Services. ⟨http://hadoop.apache.org/docs/current/hadoop-aws/tools/hadoop-aws/index.html⟩

関連研究 - CephFS-Hadoop プラグイン

- CephFS-Hadoop プラグイン[9, 10]
  - POSIX 準拠 CephFS インターフェイスを利用
  - Hadoop Filesystem API を実装することにより Apache Spark/Hadoop から利用可能
  - CephFS に Hadoop Filesystem API には不要なオーバーヘッドが多い



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

ceph

関連研究 - 並列ファイルシステム向けプラグイン

- GlusterFS
  - glusterfs-hadoop [11]
- GPFS
  - GPFS-hadoop [12]
- Gfarm
  - Hadoop-Gfarm [13]

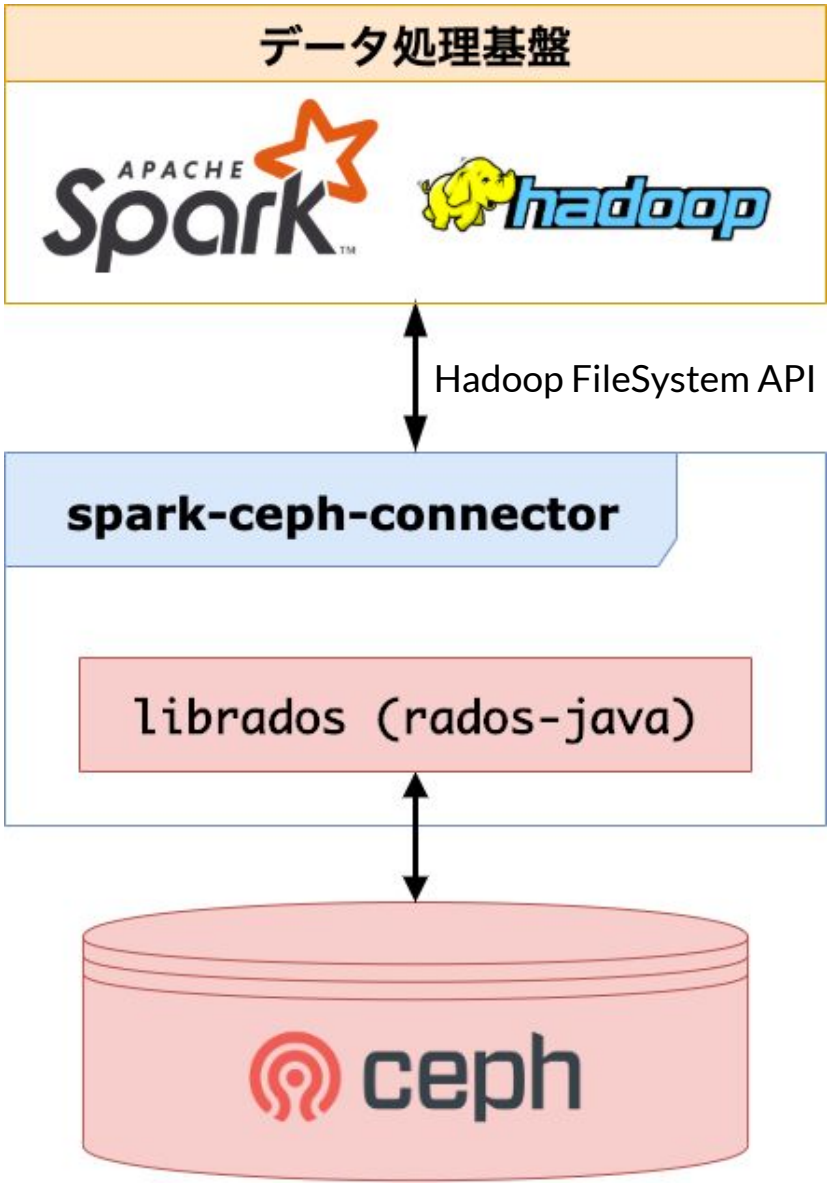
[11] gluster/glusterfs-hadoop: GlusterFS plugin for Hadoop HCFS
⟨https://github.com/gluster/glusterfs-hadoop⟩
[12] Raghavendra, R., Dewan, P. and Srivatsa, M.: Unifying HDFS and GPFS: Enabling Analytics on Software-Defined Storage, Proceedings of the 17th International Middleware Conference, Middleware ’16, New York, NY, USA, ACM, pp. 3:1–3:13
⟨https://dl.acm.org/citation.cfm?doid=2988336.2988339⟩
[13] Shunsuke Mikami ; Kazuki Ohta ; Osamu Tatebe: Using the Gfarm File System as a POSIX Compatible Storage Platform for Hadoop MapReduce Applications, 2011 IEEE/ACM 12th International Conference on Grid Computing, pp. 181-189
⟨https://doi.org/10.1109/Grid.2011.31⟩

設計と実装

1. 研究の背景
2. 研究の目的
3. 関連研究
4. 設計と実装
5. 実験と手法
6. 結果と考察
7. まとめと今後の課題

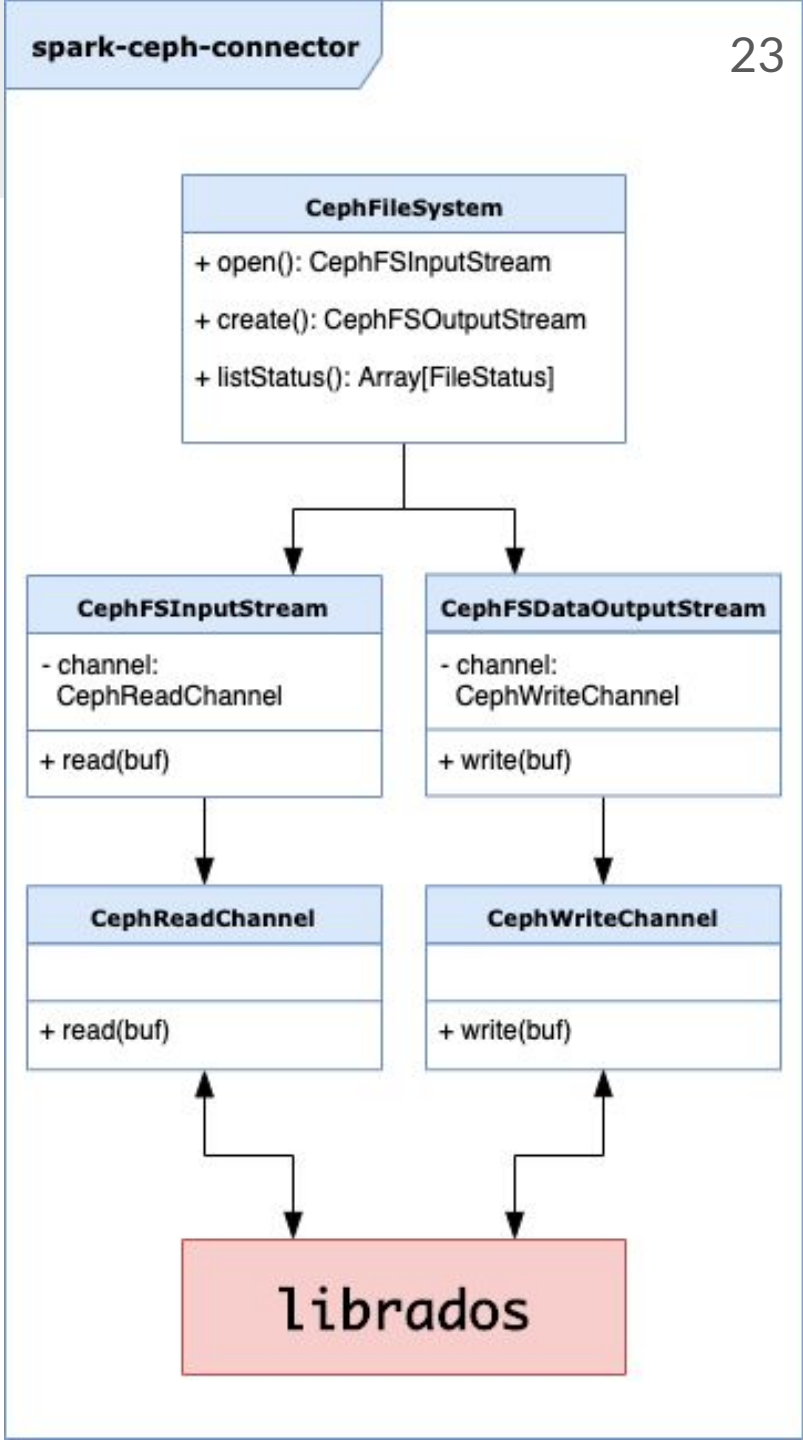
設計と実装 - librados を利用した実装

- コネクタ実装の方法
  - Apache Spark と互換性を持つ Hadoop FileSystem API を実装する
- ceph の librados ライブラリ rados-java を採用
  - JNA (Java Native Access) が利用されているためオーバーヘッドが小さい
- Scala を利用
  - 簡潔なコードで実装
  - ScalaTest



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



設計と実装 - librados との対応のための工夫

- **Ceph のフラットな名前空間の問題**
  - ディレクトリ階層が存在しない
- 仮想ディレクトリを必要とするメソッド
  - ディレクトリの移動: `rename()`
  - 再帰的な削除: `delete()`
  - ディレクトリ直下のファイルのみ表示: `listStatus()`
  - ディレクトリの再帰的な作成: `mkdirs()`
- 実装の方法
  - "/" で終わるサイズ0のオブジェクトを作成
  - 仮想的なディレクトリとして扱う

設計と実装 - librados との対応のための工夫

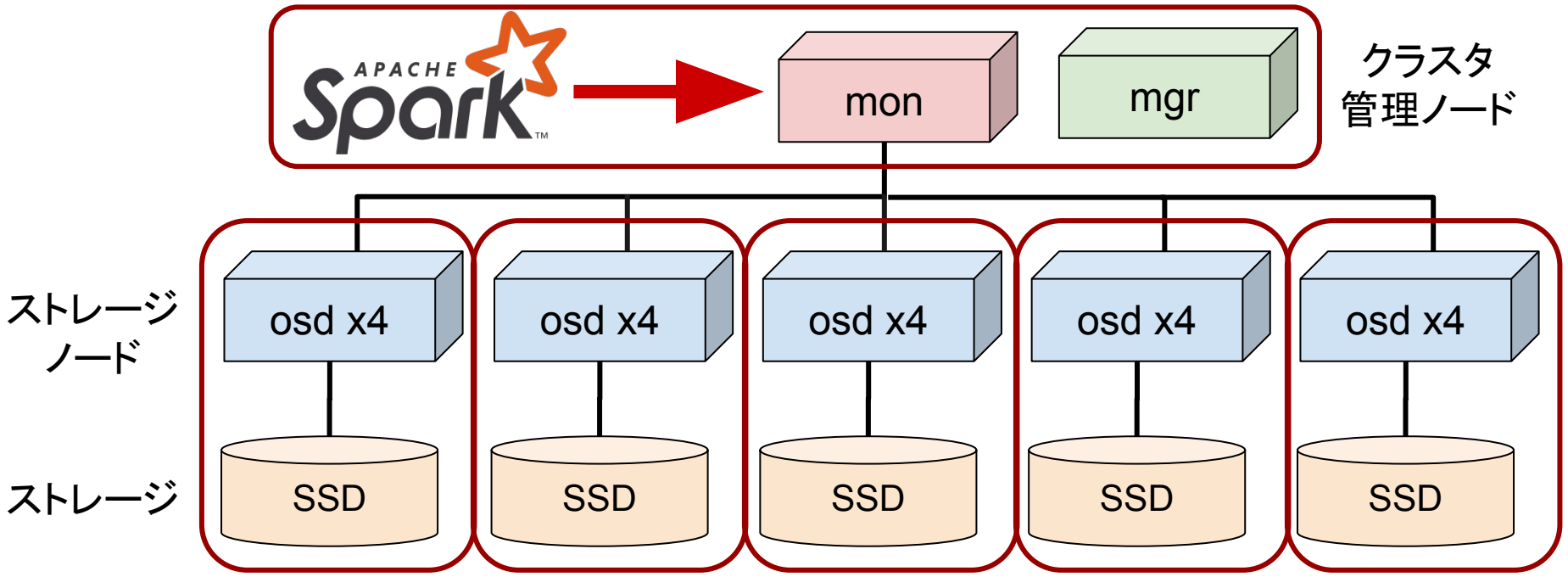
- rename() の問題
  - Ceph のオブジェクト名を変更
  - クラスタ内のデータの格納場所も変更
- 実装の方法
  - 1．新しい名前のオブジェクトをコピーして作成
  - 2．古い名前のオブジェクトを削除

実験の準備

1. 研究の背景
2. 研究の目的
3. 関連研究
4. 設計と実装
5. 実験と手法
6. 結果と考察
7. まとめと今後の課題

実験と手法 - Ceph クラスタの構成

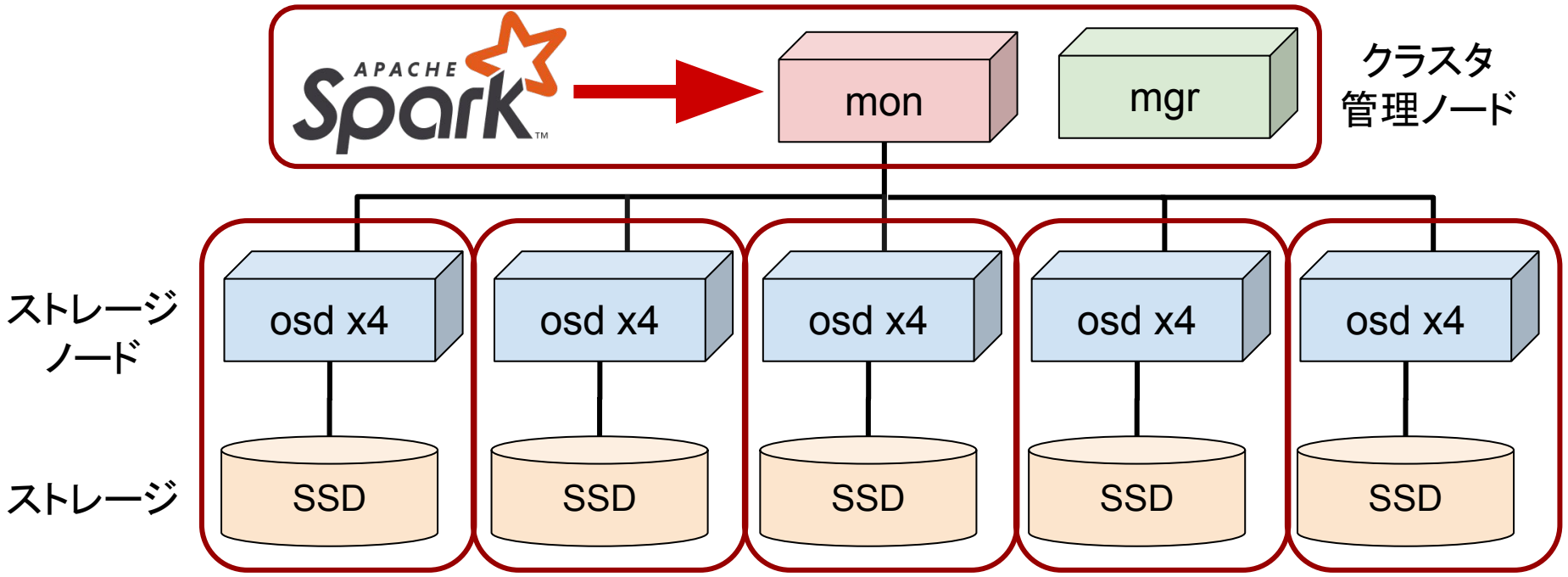
- Ceph クラスタを6ノードから構成
  - 管理ノード (1ノード): mon, mgr プロセスを実行
  - ストレージノード (5ノード): osd プロセスを実行



ceph

実験と手法 - Ceph クラスタの構成

- ストレージ: SSD モジュールを使用
- レプリケーションファクタ: Ceph のデフォルト 3 に設定



ceph

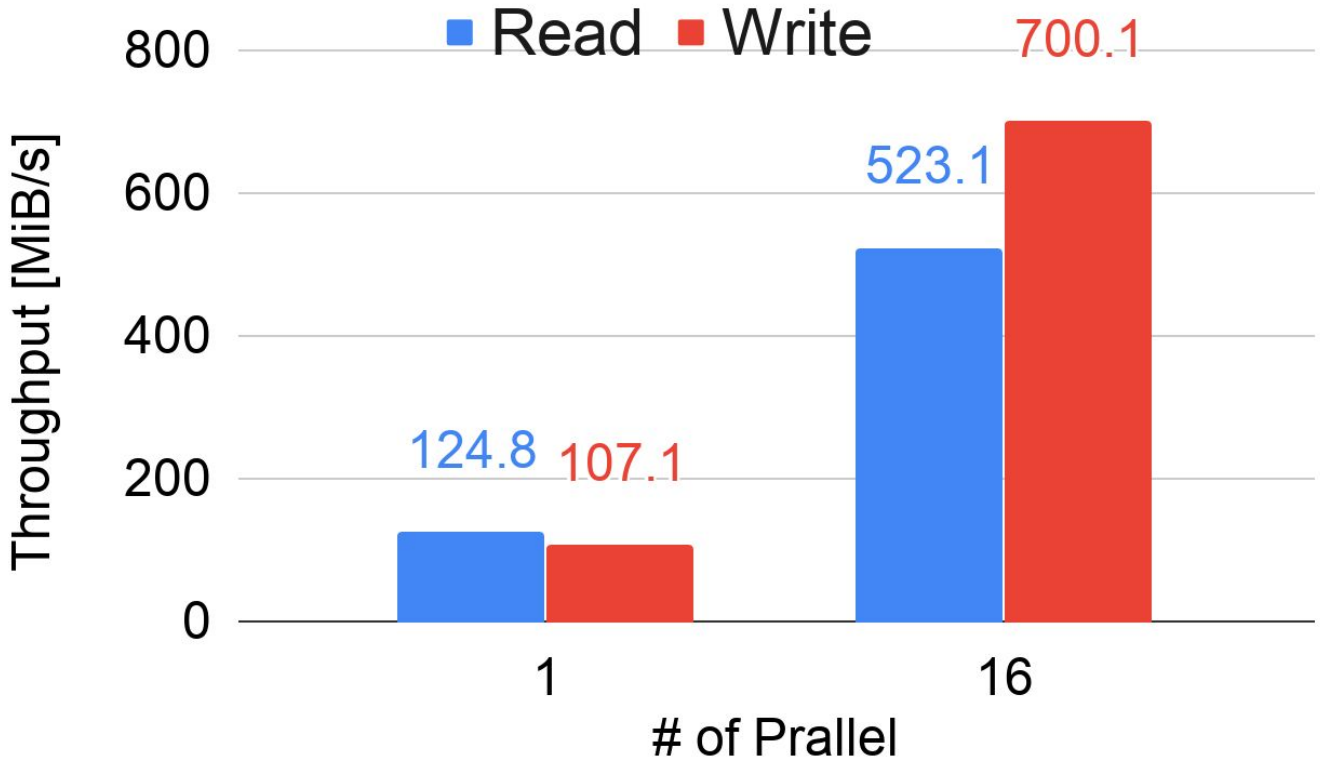
実験と手法 - 実験環境

- Ceph クラスタを構成する各ノードの環境

CPU	Intel Xeon CPU E5410 @ 2.33GHz x2
メモリ	DDR3 FB-DIMM 667 MHz 4GB x 8 (Total 32 GB)
ネットワーク	10 Gbit Ethernet
ストレージ	RevoDrive 3 X2 PCI-Express SSD / 240 GB
OS	CentOS 7.6.1810
Linux Kernel	3.10.0-957.21.3.el7.x86 64
Ceph	v14.2.2 Nautilus (latest stable)

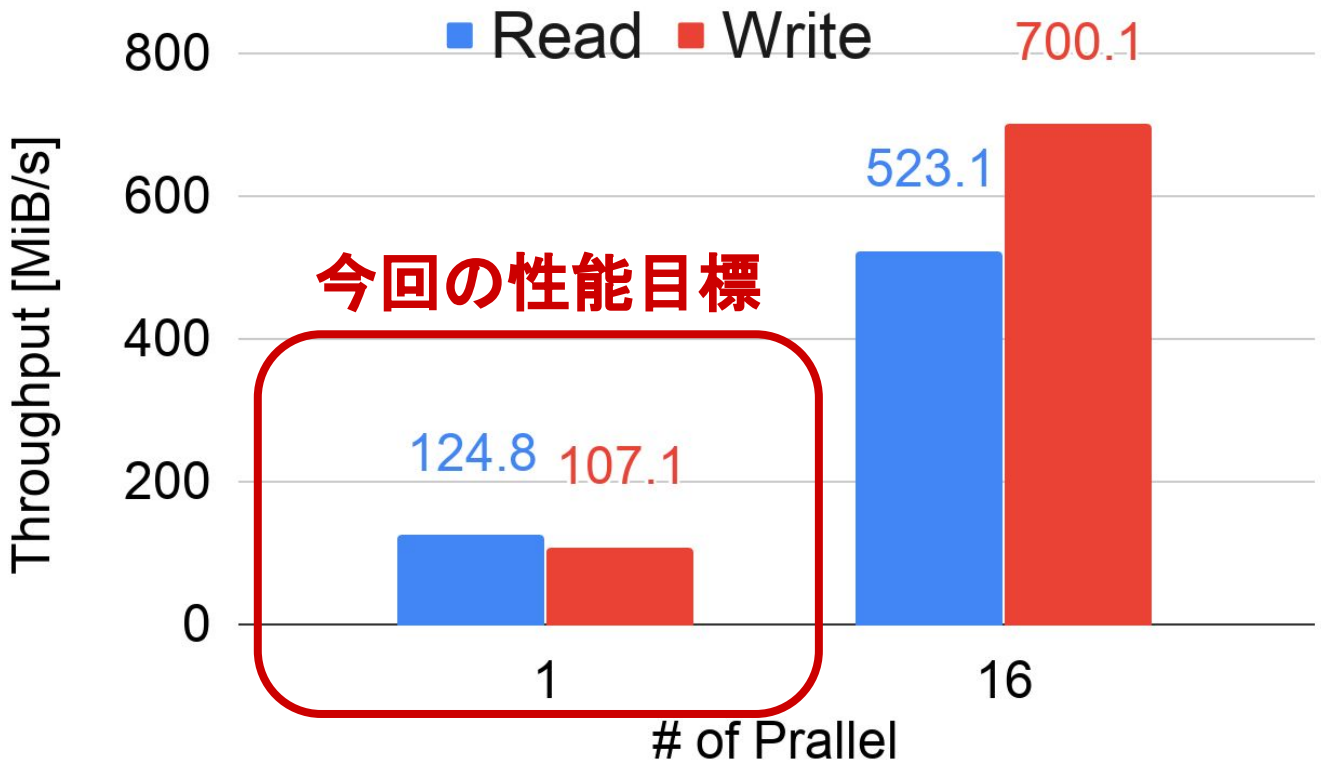
実験と手法 - RADOS Bench によるピーク性能の測定

- Ceph に付属するベンチマークツール RADOS Bench を使用
- 実験環境の Ceph クラスタのピーク性能として シーケンシャルの Read/Write 性能を測定



実験と手法 - RADOS Bench によるピーク性能の測定

- 1並列
  - **Read 124.8 MiB/s**
  - **Write 107.1 MiB/s**
- 16並列 (参考値)
  - Read 523.1 MiB/s
  - Write 700.1 MiB/s



実験と手法 - Cephコネクタの性能評価

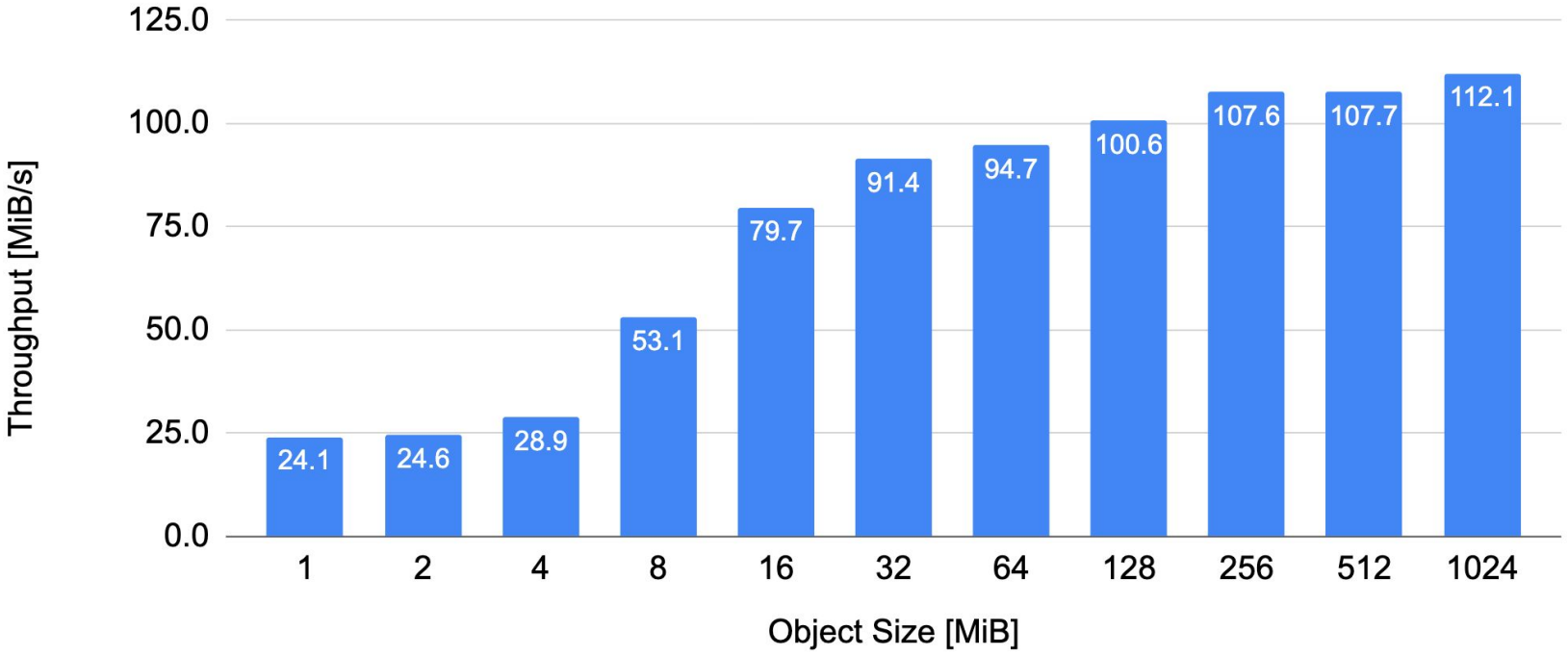
- Ceph クラスタと Apache Spark との間のオブジェクトの Read と Write の性能を測定
- **Read:**
  - テキストデータのオブジェクトを 1GiB 分生成
  - Ceph クラスタに保存
  - Spark から Ceph コネクタ経由で読み込み
- **Write:**
  - Spark 上に書き込みデータをキャッシュ
    - Spark のキャッシュ機能を利用
  - キャッシュしたデータを Ceph コネクタ経由で Ceph クラスタへ書き込み
- **生成したテキストデータのオブジェクトサイズ:**
  - 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1,024 MiB

結果と考察

1. 研究の背景
2. 研究の目的
3. 関連研究
4. 設計と実装
5. 実験と手法
6. 結果と考察
7. まとめと今後の課題

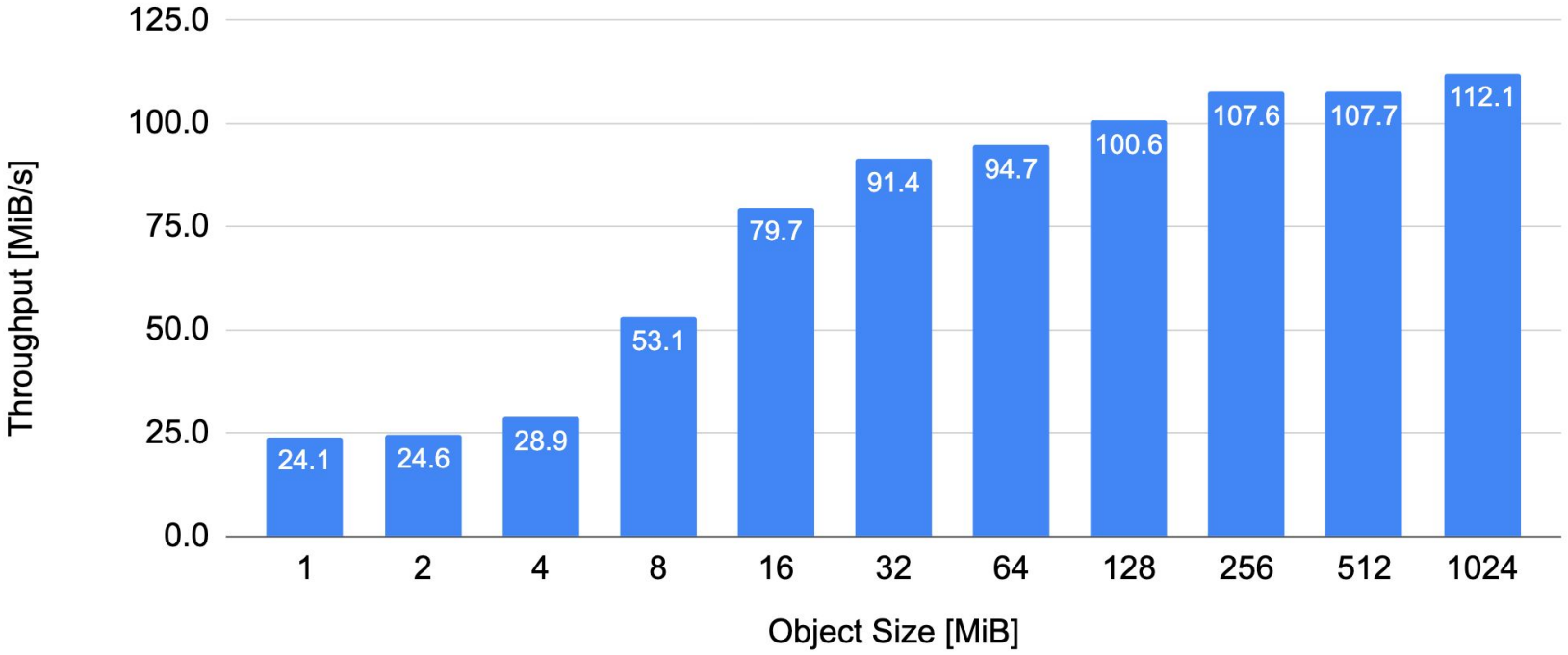
実験と手法 - Read 性能

- オブジェクトサイズが大きくなると Read 性能も向上、32 MiB あたりから徐々に飽和
- 最も大きなオブジェクトサイズ 1024 MiB のとき、Read 性能が最大の 112.1 MiB/s



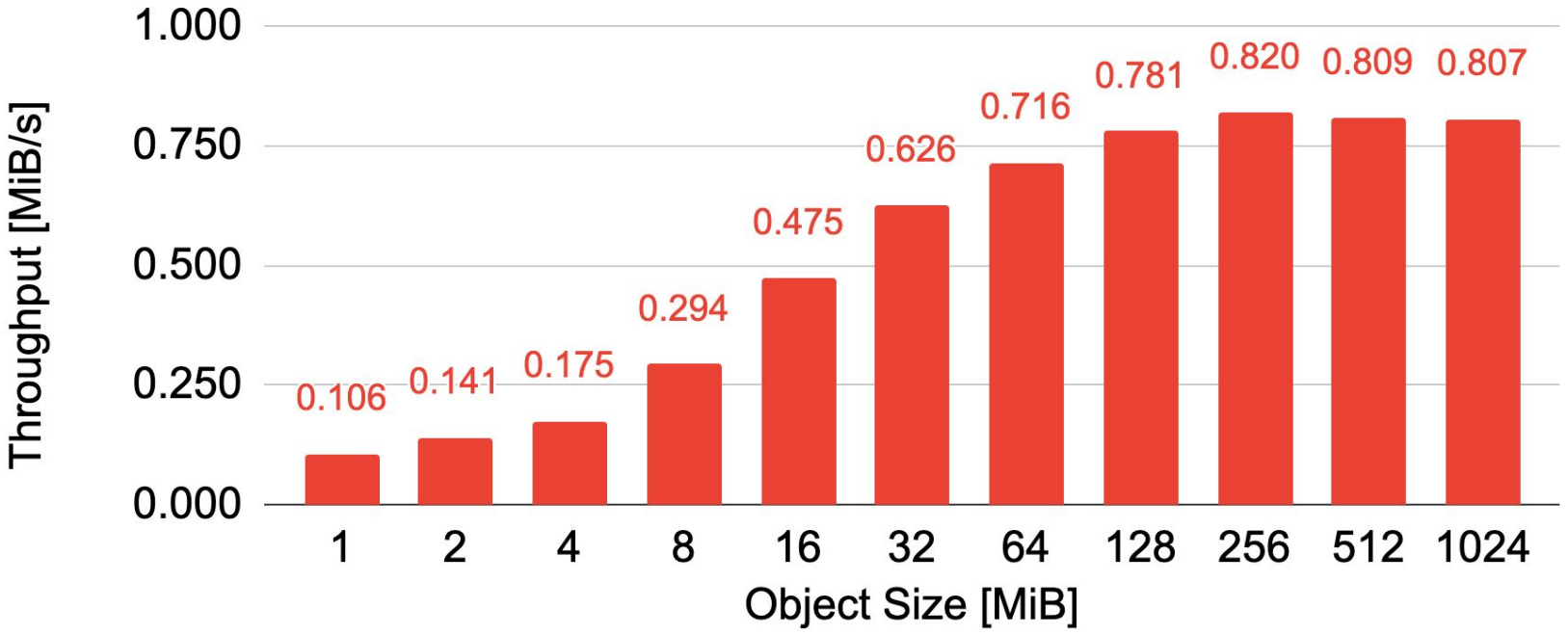
実験と手法 - Read 性能

- 今回の実験環境における Ceph クラスタのピーク性能の約 90 % の高い性能を発揮
- HDFS のデフォルトブロックサイズ 128 MiB の場合でも、約 81 % の 100.6MiB を発揮



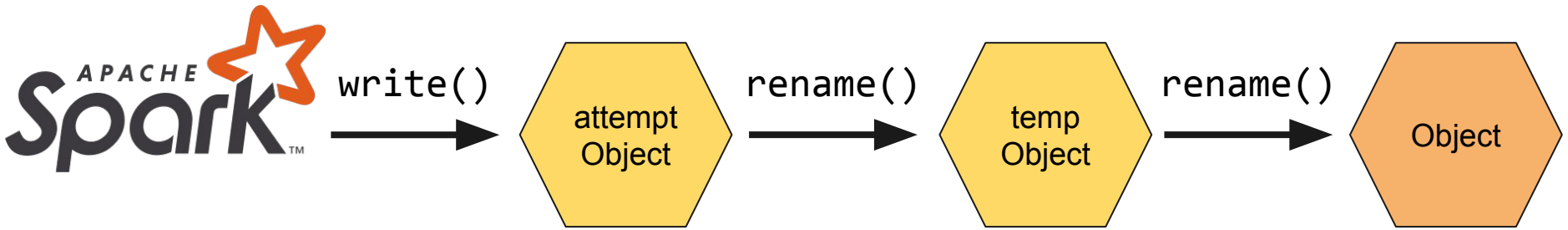
実験と手法 - Write 性能

- オブジェクトサイズ 32 MiB 付近から飽和する傾向は Read と同様
- 一方、Write 性能は Read に比べると著しく低く、約 1/200 ～ 1/100 しか発揮されていない
- 最大でも 0.820 MiB/s という 1 MiB/s を下回る結果



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

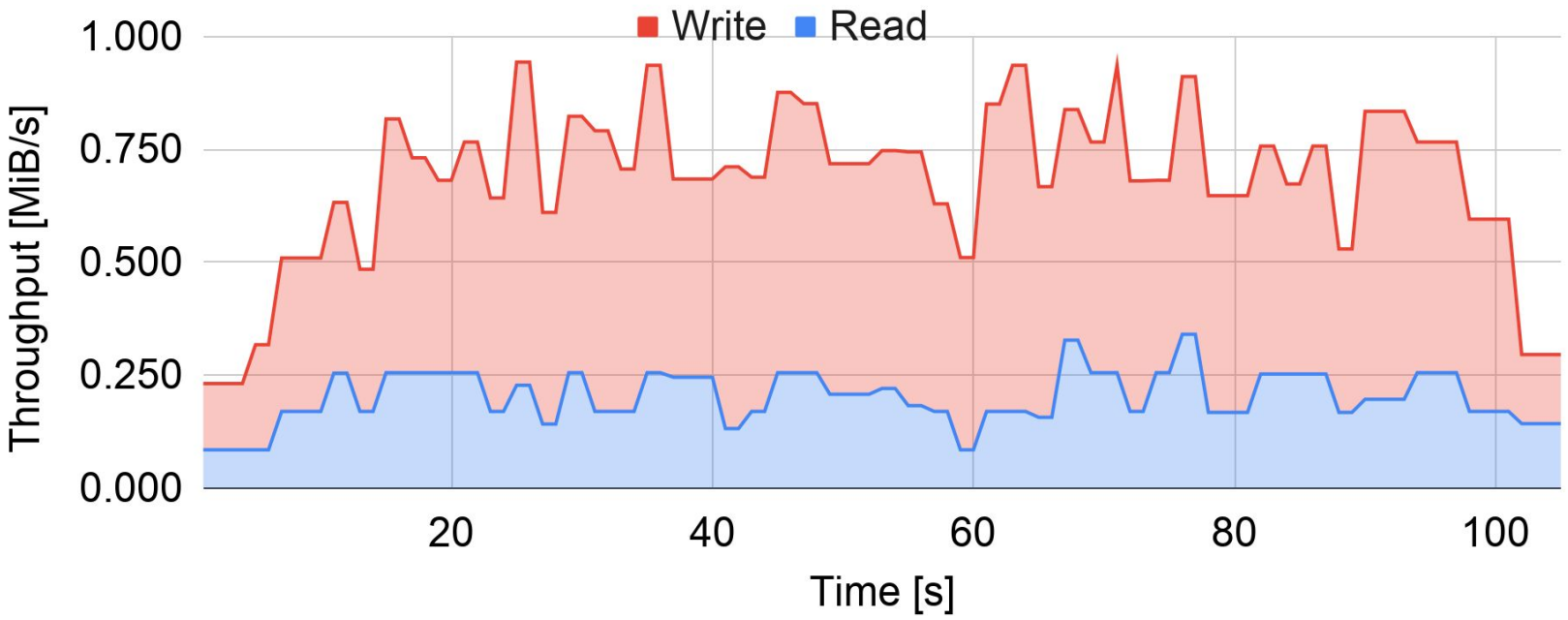


実験と手法 - 性能劣化の原因 (1)

- Ceph では MDS を排除するために、オブジェクト名とクラスタマップを元に CRUSH アルゴリズムにより保存先の OSD を確定的に決定できる
  - Spark による rename のたびに RADOS オブジェクトの読み込み・書き込みが必要になってしまう
- Ceph クラスタ内の I/O について解析する

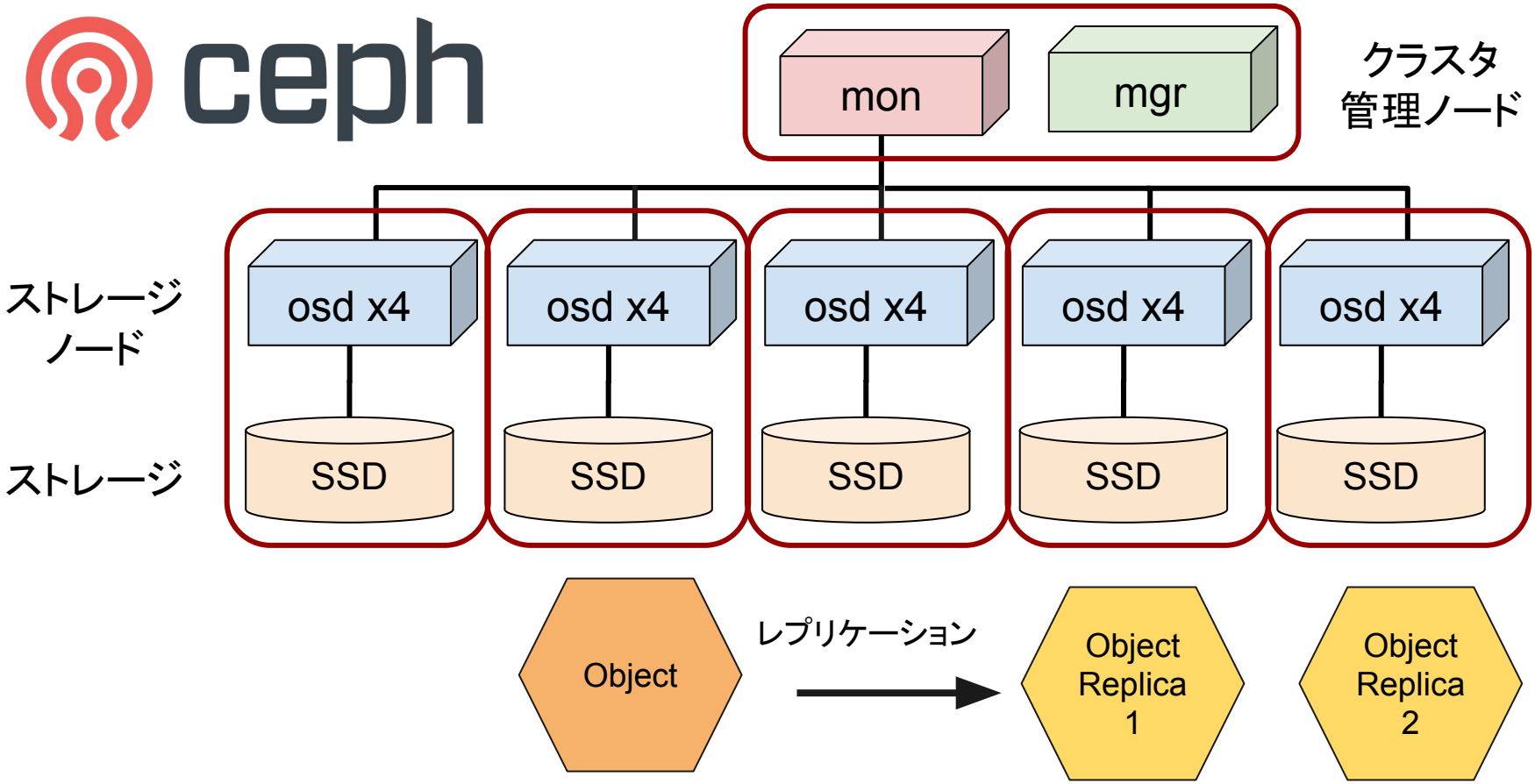
実験と手法 - 性能劣化の原因 (1)

- 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



実験と手法 - 性能劣化の原因 (2)

- 原因2: Ceph のレプリケーションの影響
  - 全書き込みで自動的にレプリケーションが作成

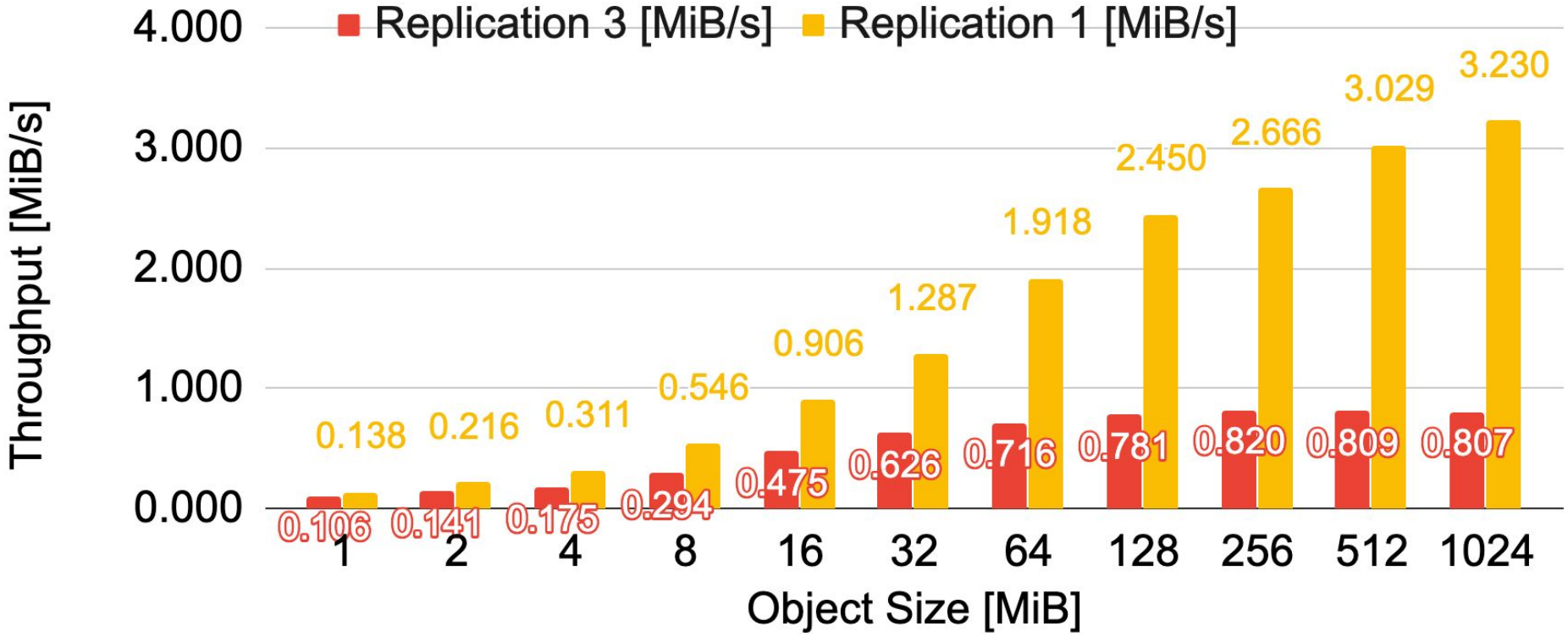


実験と手法 - 性能劣化の原因 (2)

- 原因2: Ceph のレプリケーションの影響
  - Spark による前述の rename の複数書き込みに加えて、Ceph クラスタ内部で中間ファイルを含むすべてのオブジェクトデータが3つの OSD にレプリケーションされる
  - レプリケーション数を 3 から 1 に変更し、レプリケーション数の影響を確認

実験と手法 - 性能劣化の原因 (2)

- オブジェクトサイズが 1,024 MiB のとき、最大で約 4 倍の 3.23 MiB/s にまで性能が向上
- Read と比較すると依然として性能が低い



まとめと今後の課題

1. 研究の背景
2. 研究の目的
3. 関連研究
4. 設計と実装
5. 実験と手法
6. 結果と考察
7. まとめと今後の課題

まとめと今後の課題 - まとめ

- 分散オブジェクトストレージ Ceph に格納された大規模データをビッグデータ処理基盤である Apache Spark などから高効率で利用可能にすることを目的として、Hadoop FileSystem API を実装した Ceph のストレージコネクタの設計と実装を行った
- これにより、Ceph の RADOS ネイティブオブジェクトを直接活用したストレージコネクタが実際に利用できることを示すことができた

まとめと今後の課題 - まとめ

- 性能測定の結果、最大 112.1 MiB/s の良好な Read 性能が発揮できた
- 一方で、Write 時には最大でも 0.820 MiB/s、レプリケーションファクタを 1 にしても Read の約 1/30 の 3.23 MiB/s の性能しか出ず、オブジェクトストレージ特有の問題が存在することが明らかになった

まとめと今後の課題 - 今後の課題

- 今回の実験で行うことができなかった、Ceph RADOS Gateway + S3A コネクタとの比較実験を行う
- Write 性能の向上のために、`librados` の非同期 I/O API の活用や、ノードローカルストレージを活用し書き込み回数の削減を行うことが考えられる
- 今回の実験ではシングルノードからの I/O 性能のみを測定したが、Spark クラスタを多数ノードで構成した場合のストレージアクセスのスケーラビリティの解析も明らかにしたい