オンプレミススーパーコンピュータとクラウドの連携

九州大学　情報基盤研究開発センター

南里　豪志

自己紹介

- 1992年
  - 島崎眞昭研@九州大学大型計算機センター

- 1997年
  - 九州大学大型計算機センター 助手
- 現在
  - 九州大学情報基盤研究開発センター 准教授

- 研究分野
  - 主に並列計算向けミドルウェア
    - 特に通信関係

今日のお話し

- オンプレミス vs クラウド
  - 性能、価格、使い勝手
- オンプレミス＋クラウド
  - 連携シナリオの例

オンプレミス vs クラウド

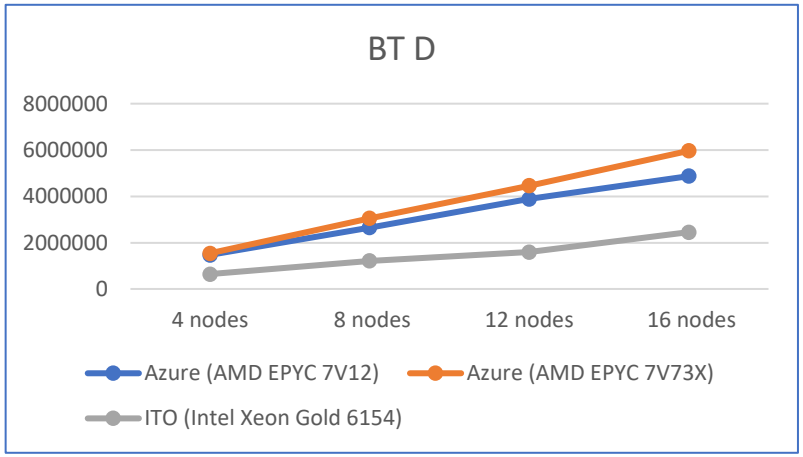
- クラウドコンピューティングサービスを
  オンプレミススーパーコンピュータ利用者の
  視点で評価
  - 日本Microsoft社より試験環境提供
    - 予めSlurmによるバッチ環境用意
  - 九大センター教員が「利用者」として利用
    - ４名（美添、大島、樋口、南里）
  - 性能と使い勝手を評価

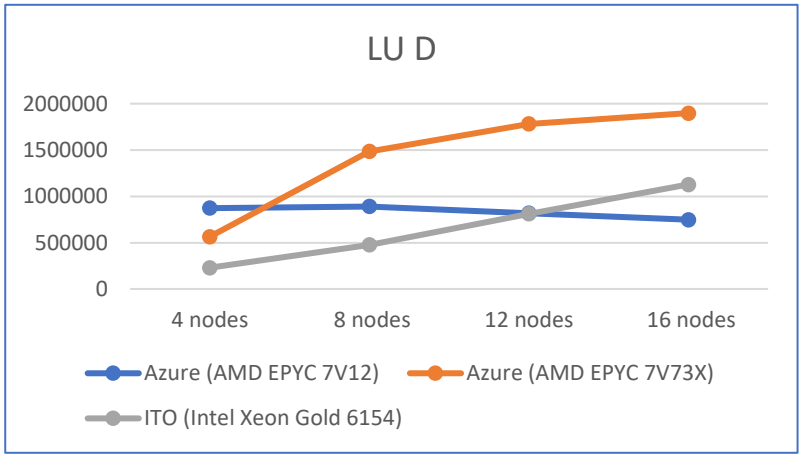
性能評価

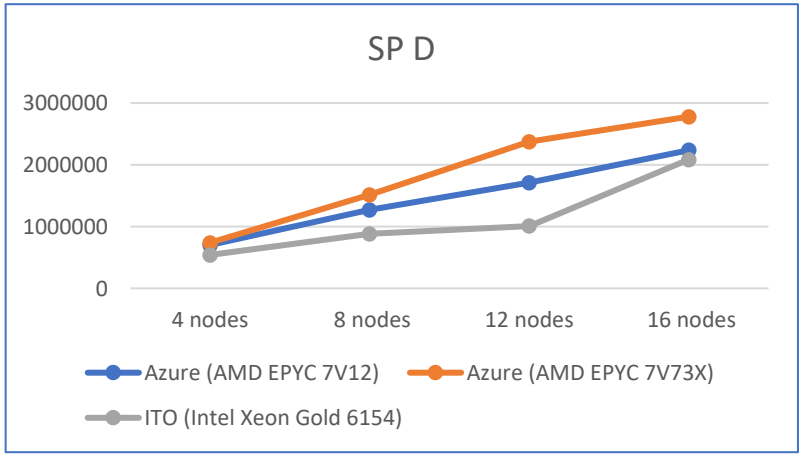
- 内容
  - 基本性能
    - NAS Parallel Benchmarks
  - 分子シミュレーション
    - 自作
    - GROMACS
    - LAMMPS
  - 機械学習
    - Graph Neural Networkで化合物の性質を予測
  - ファイル性能
    - FIO

NAS Parallel Benchmark

- Class=D







- AMD EPYC 7V73XとITOのFLOPS比 1.3倍
- **クラウドとオンプレミスで、CPU、メモリ、ネットワークの性能に大きな相違は見られない**
- 名古屋大学の永井先生からの報告（2023年9月のHPC研究会）でも、同様の結果

分子シミュレーション

by 九州大学 樋口祐次 准教授

- 実行時間
  - 自作プログラム、GROMACS、LAMMPSとも、**オンプレミスサーバとクラウドで有意な差は見られなかった**
  - LAMMPS、GROMACSは、ファイルシステムによる性能差が若干見られた

- ファイル転送時間（768MB, 約80ファイル）

	アップロード(秒)	ダウンロード(秒)
Azure	82.2	174.1
Ito	8.5	12.1
物性研スパコン（千葉県柏市）	13.8	18.2

機械学習

by 九州大学 美添一樹 教授

- Graph Neural Networkによる化合物の性質の予測
  - 5つの特性を同時にMTL (Multi-Task Learning) で予測するように訓練
    - 教師データ
      - K. Terayamaらが Gaussian16で生成したデータを使用 [K. Terayama et al. Chemical Science. 2020]
      - ZINCデータベースの約100,000個の化合物に対して、5つの特性を計算 [John J. Irwin et al. Journal of Chemical Information and Modeling. 2012]
    - コード
      - https://github.com/masashitsubaki/molecularGNN_smiles
  - 同一ステップの学習時間
    - RTX3090 x 4: 19時間15分
    - Tesla V100 x 4: 11時間 7分
  - **ほぼGPUの性能に応じた学習時間と言える**

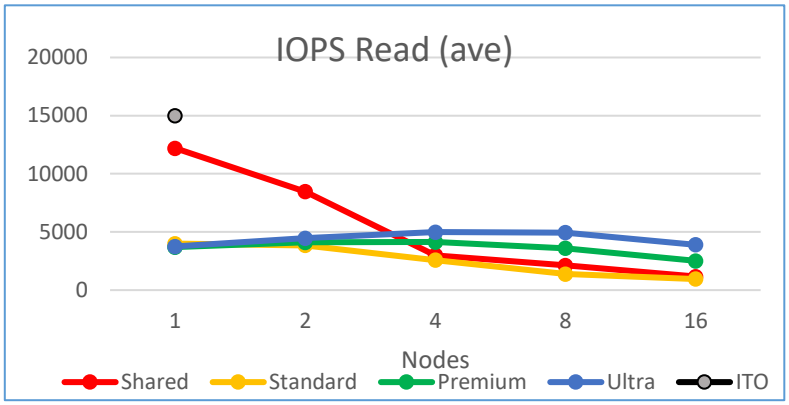
ファイル性能

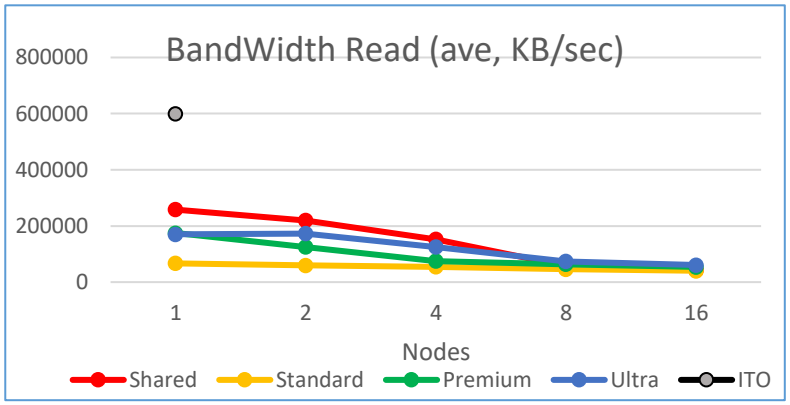
- 今回使用したファイルシステム

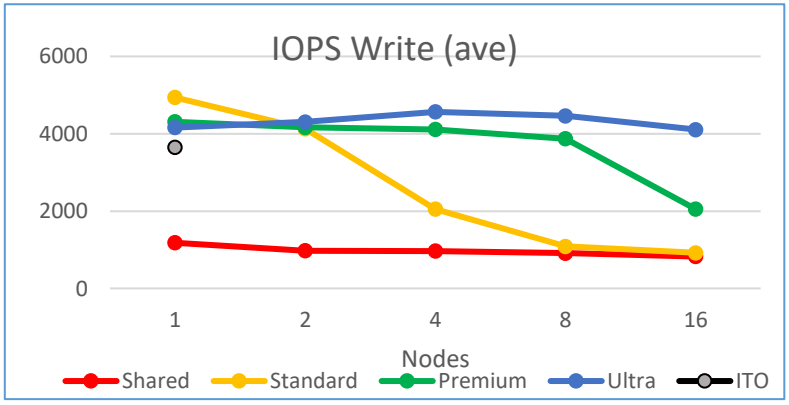
Azureのファイルシステム	スループット	IOPS	価格（$/GiB/月）
Azure Managed Disks (shared)	25MB/s	3,500 IOPS	$0.0075
Azure NetApp Files Standard	16MiB/s per TiB	1,000 IOPS/TiB	$0.14746
Azure NetApp Files Premium	64MiB/s per TiB	4,000 IOPS/TiB	$0.29419
Azure NetApp Files Ultra	128MiB/s per TiB	8,000 IOPS/TiB	$0.39274

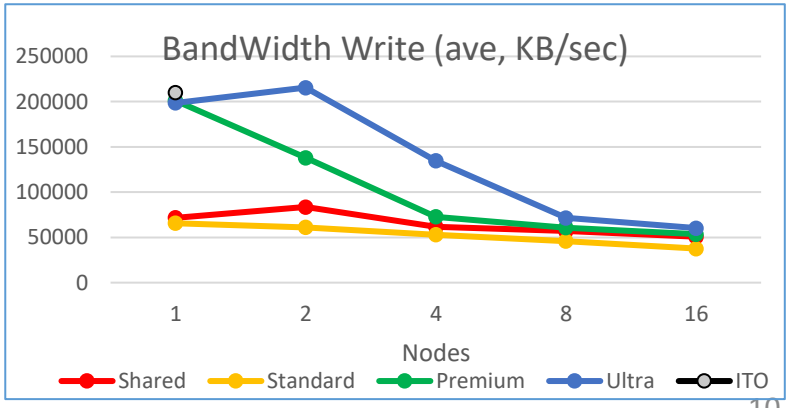
FIO

- 同時利用ノード数に応じた
  ノード当たり平均性能の変化









オンプレミスユーザから見たクラウドの使い勝手

- プログラムのコンパイル、実行に大きな問題はない
  - 同じプログラムがオンプレミスと同様に利用可能
  - 性能もほぼ想定通り

- 若干、工夫や慣れが必要な部分はあった
  - 外部との通信速度
    - 大規模データのアップロード、ダウンロードは速度面で不安が残
  - ファイルシステムによる性能差
    - アプリケーションのファイルIOの量や頻度によって適切な選択が必要
  - GPUの利用
    - MIGの利用、コンテナの利用については、オンプレミスとの環境の違いが大きい場合に、対応に時間を要することがあった
  - 予算管理
    - 特に複数利用者で金額を予算額内におさめるのが難しかった

オンプレミス vs クラウド

	オンプレミス	クラウド
資源単価	**比較的安価**	比較的高価
可用性	× • 定期保守、リプレースなどで停止 • 繁忙期の待ち時間	**〇**
資源の多様性	× • 基本的に５～６年間更新無し	**〇** • **最新アーキテクチャも選択可能**
予算管理の容易さ	**単純** • **トークン購入もしくは月額定額** • **予算を超過したら利用停止**	複雑 • 基本的に、使った分を後払い • 利用金額に対する制限の自動化には別ツール等が必要

オンプレミス + クラウド

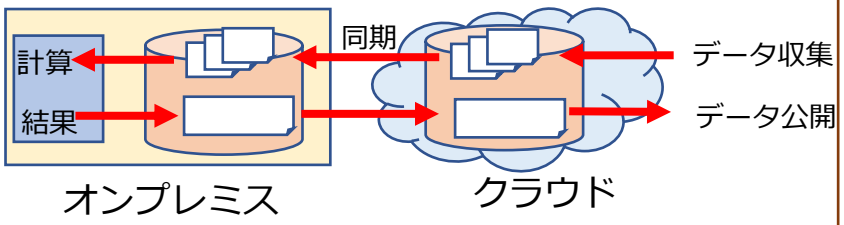
	オンプレミス	クラウド	オンプレミス＋クラウド
資源単価	**比較的安価**	比較的高価	**〇** • **状況に応じて選択可**
可用性	× • 定期保守、リプレースなどで停止 • 繁忙期の待ち時間	**〇**	**〇** • **状況に応じて選択可**
資源の多様性	× • 基本的に５～６年間更新無し	**〇** • **最新アーキテクチャも選択可能**	**〇** • **状況に応じて選択可**
予算管理の容易さ	**単純** • **トークン購入もしくは月額定額** • **予算を超過したら利用停止**	複雑 • 基本的に、使った分を後払い • 利用金額に対する制限の自動化には別ツール等が必要	**？**

オンプレミス＋クラウドの利用シナリオ例

例 1

ストレージ同期
- データ収集
- 研究データ公開

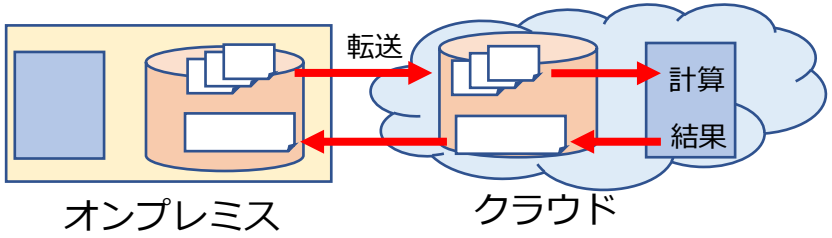
など



例 2

クラウドバースティング
- システム停止時の代替利用
- 混雑時の待ち時間低減
- 最新機器の利用

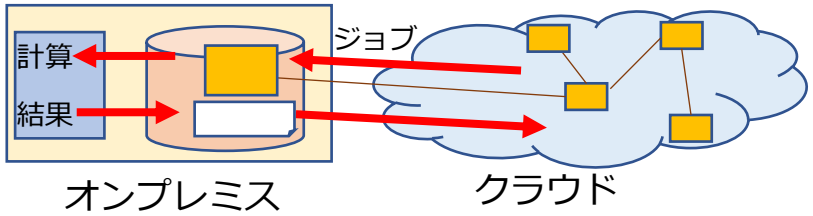
など



例 3

オンプレミスバースティング
- クラウドアプリ中の高負荷計算の価格低減

など



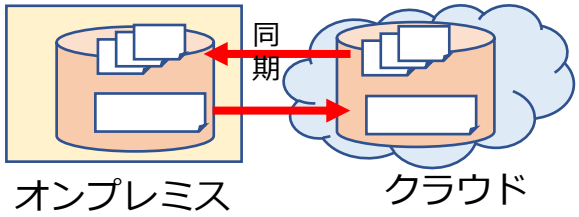
各シナリオ例の検証

- 前提
  - オンプレミス計算機は一般利用者権限で利用
    - 管理者権限不要
  - オンプレミス計算機の認証情報外部に持ち出さない
    - 多くの組織で認められていない
    - そのため、今回の検証では遠隔操作をすべてオンプレミス側から実行
  - 今回はAWSのクラウド計算環境を利用
    - 他のクラウド計算環境との連携は、今後検討

今回利用した主な機能

- AWSの機能
  - AWS CLI (Command Line Interface)
    - LinuxからAWSのクラウド計算資源を操作するコマンド群
  - Amazon S3 (Simple Storage Service)
    - AWSのクラウドストレージサービス
    - ファイル同期、ジョブ関連ファイルの転送などに利用
  - Amazon ParallelCluster
    - AWSのクラスタ管理インタフェース
    - クラウド環境のクラスタ構築、制御に利用
  - Amazon EventBridge
    - AWSのイベント通知機構
    - クラウドバースティング時のジョブ到着検知に利用
- Linuxの機能
  - cron
    - Linuxにおける定期的なコマンド実行機能
    - オンプレミスバースティング時のジョブ到着検知に利用

ファイル操作



- S3バケットの作成
  - オブジェクトの格納場所

```
aws s3 mb s3://test-bucket
```

- ファイルコピー
  - オンプレミスのファイル → S3のオブジェクト

```
aws s3 cp ./hello.c s3://test-bucket/
```

  - S3のオブジェクト → オンプレミスのファイル

```
aws s3 cp s3://test-bucket/result.dat .
```

- ディレクトリコピー
  - オンプレミスのディレクトリ → S3のプレフィクス

```
aws s3 sync ./work s3://test-bucket/work
```

  - S3のプレフィクス → オンプレミスのディレクトリ

```
aws s3 sync s3://test-bucket/work ./work
```

備考）
aws s3 syncコマンドによるファイル転送

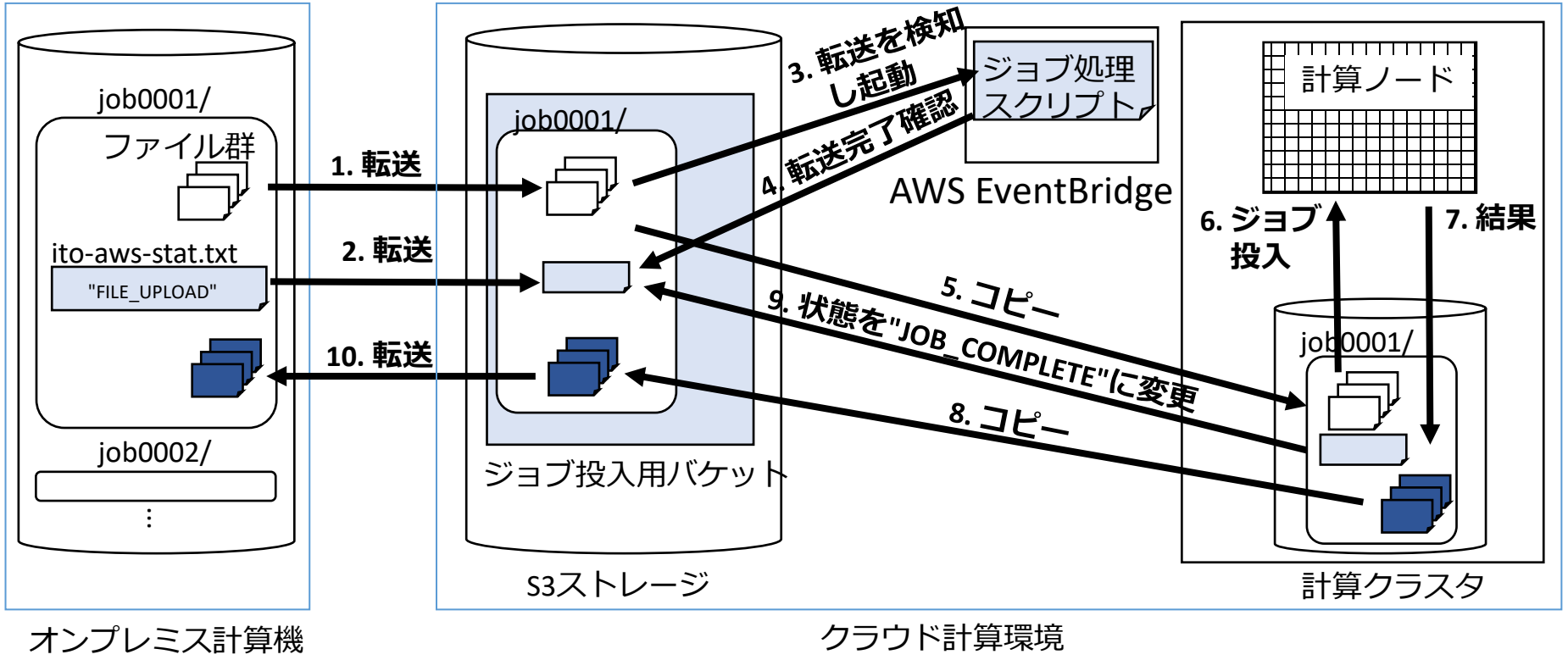
- オンプレミス → S3
  - S3バケットに同名のオブジェクトが存在しないファイル
  - S3バケットの同名のオブジェクトとサイズが違うファイル
  - S3バケットの同名のオブジェクトより**更新時刻が新しい**ファイル

- S3 → オンプレミス
  - ディレクトリに同名のファイルが存在しないオブジェクト
  - ディレクトリの同名のファイルとサイズが違うオブジェクト
  - ディレクトリの同名のファイルより**更新時刻が古い**オブジェクト

古いオブジェクトで新しいファイルが上書きされるので注意

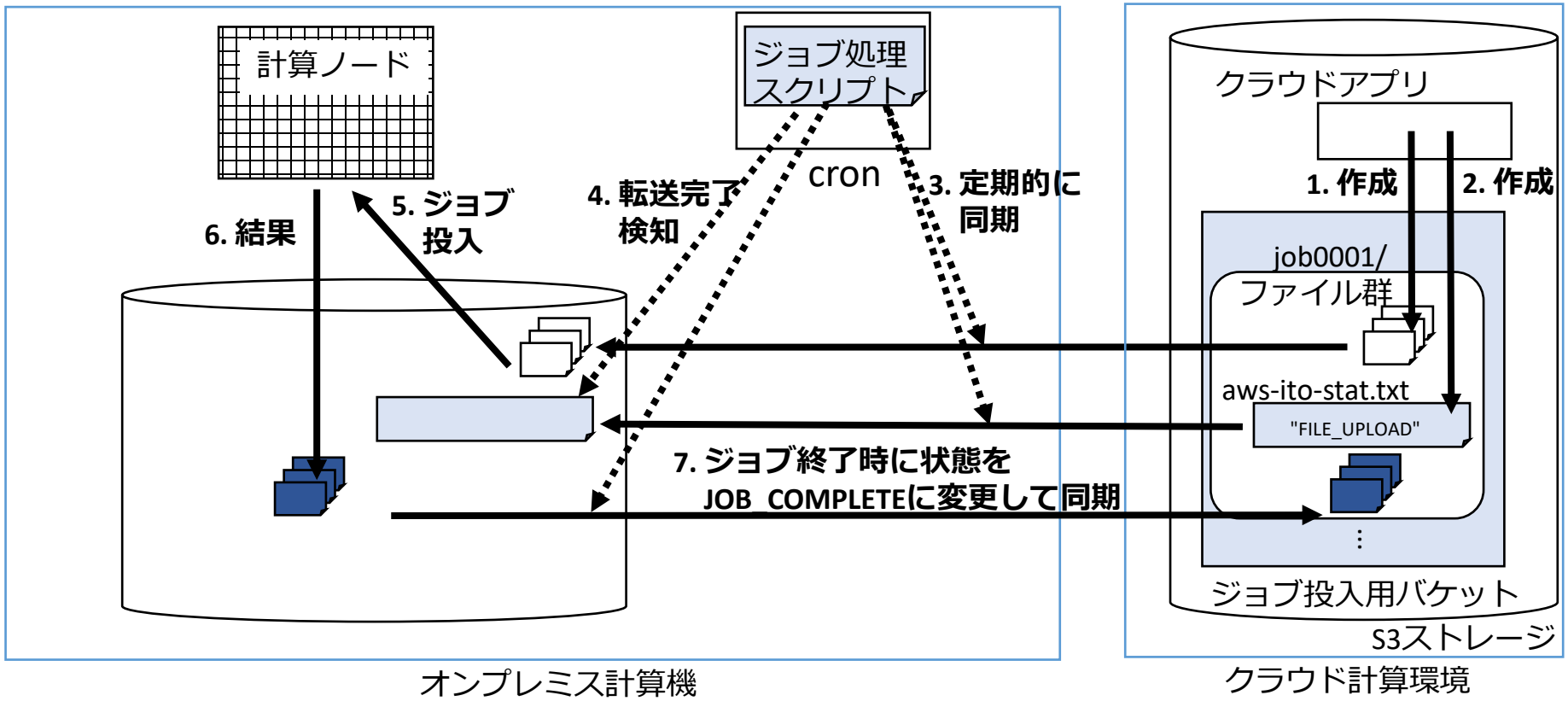
クラウドバースティング

- AWS EventBridgeによるイベント検知



オンプレミスバースティング

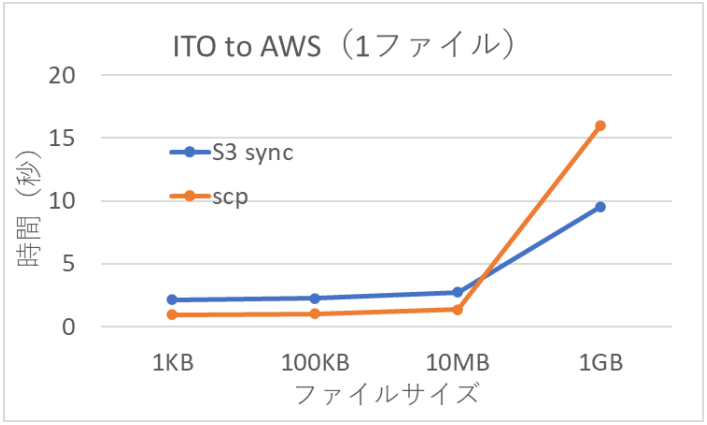
- cronによるイベント検知

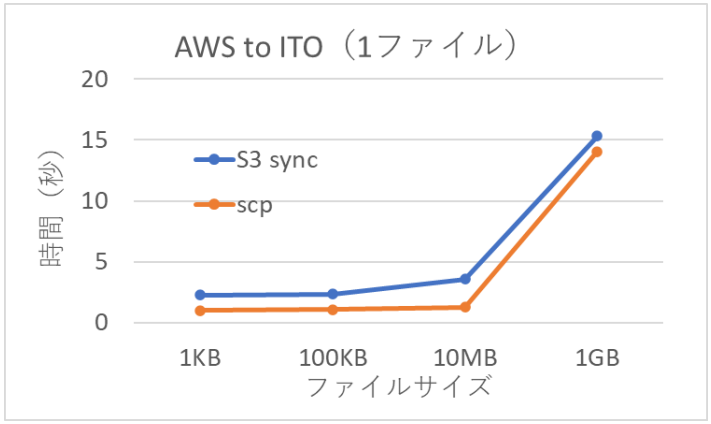


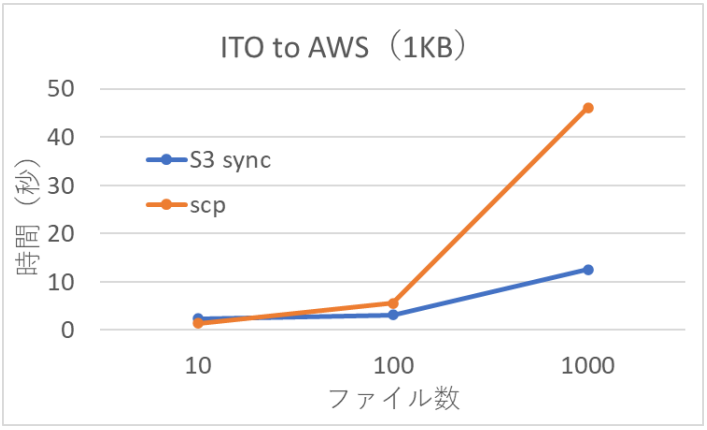
性能計測

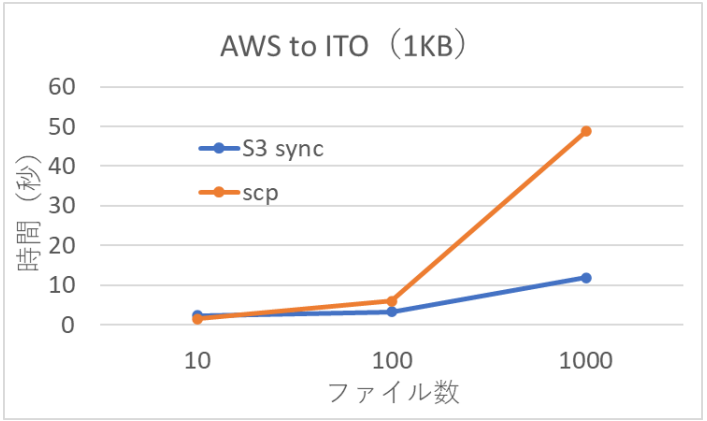
- 計測環境
  - オンプレミス
    - 九州大学情報基盤研究開発センター、スーパーコンピュータ ITO ログインノード
      - Intel Xeon Gold 6140 x 2, 384GB RAM
      - 10 Gb Ethernet x1
      - RedHat Enterprise Linux 7
      - AWS CLI 2.2.35
      - AWS ParallelCluster 3.3.0
  - クラウド
    - AWS EC2
      - Head Node: t3a.small
      - Compute Node: c4.large
      - Region: ap-northeast-1

ファイル転送時間


ITO to AWS（1ファイル）
20
15
10
5
0
時間（秒）
S3 sync
scp
1KB
100KB
10MB
1GB
ファイルサイズ


AWS to ITO（1ファイル）
20
15
10
5
0
時間（秒）
S3 sync
scp
1KB
100KB
10MB
1GB
ファイルサイズ


ITO to AWS（1KB）
50
40
30
20
10
0
時間（秒）
S3 sync
scp
10
100
1000
ファイル数


AWS to ITO（1KB）
60
50
40
30
20
10
0
時間（秒）
S3 sync
scp
10
100
1000
ファイル数

クラウドバースティング

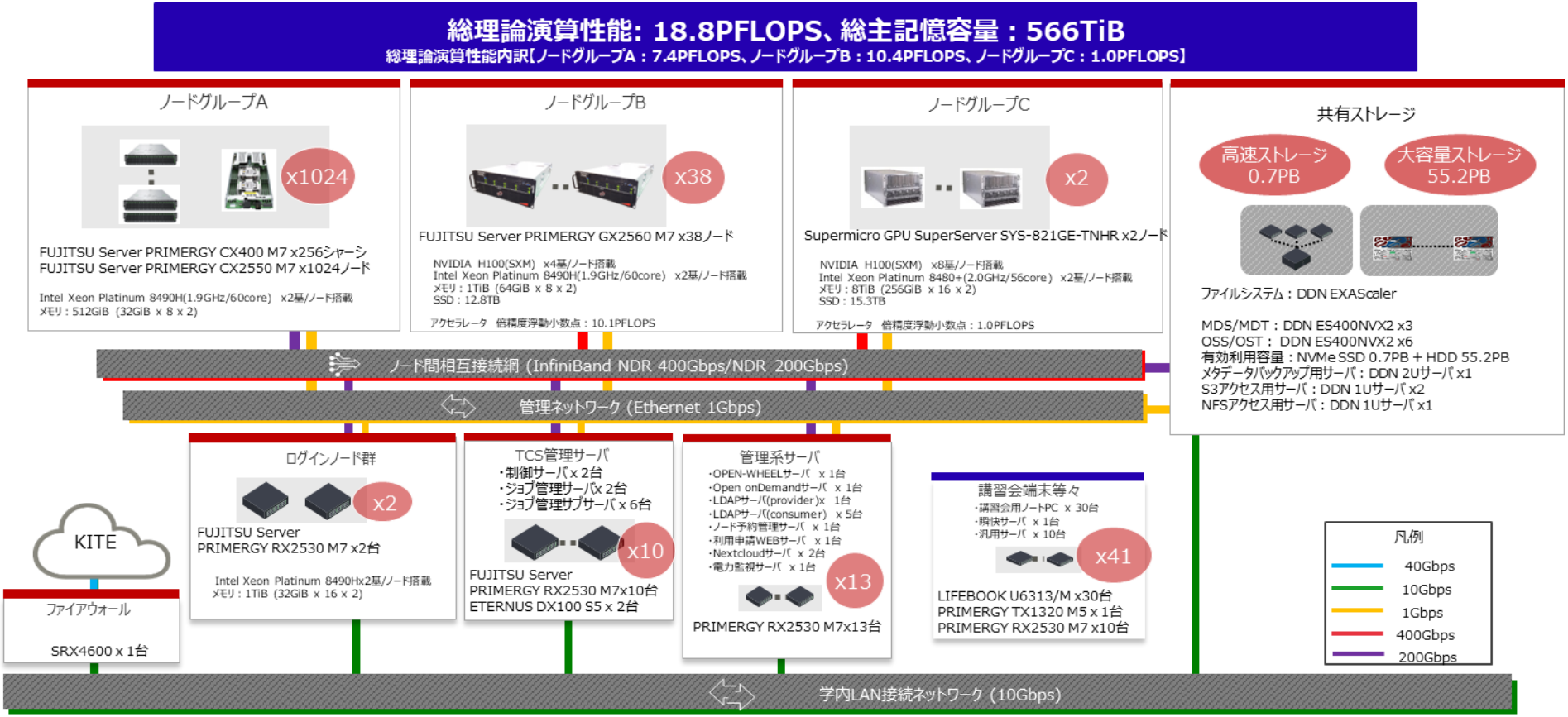
操作	所要時間（秒）
バケット生成	2.8
バケット削除	2.3
クラスタ生成	340
クラウドバースティング： 最初のジョブ実行（ノード起動時間含む）	235
クラウドバースティング： 2回目以降のジョブ実行	12

オンプレミスバースティングについては未計測
（今回の実装ではcronやバッチジョブの状況で
結果が大きく変わるため）

オンプレミス＋クラウド連携に向けて

- それぞれの特性に応じた、「いいとこどり」の連携をしたい
  - 例えば
    - 24時間365日の可用性が必要な部分はクラウド
    - 高負荷で、かつ、少し待っても良い部分はオンプレミス
    - オンプレミスが混雑している場合や、オンプレミスにない資源を利用したい場合はクラウド
- 技術的にはまだまだ検討が必要
  - 相互認証手段
  - 連携環境構築
- HPCクラウド部会は、議論に最適の場

九州大学の次期システム紹介



- クラウド連携ツール群も導入予定
  - AWS CLI, AWS ParallelCluster, Azure CLI, Azure CycleCloud, Oracle OCI, Google Gcloud
- システムの詳細は AXIES2023で

謝辞

- 本資料の作成に当たり以下の支援をいただきました
  - クラウド計算資源提供
    - 日本マイクロソフト株式会社
    - アマゾンウェブサービスジャパン合同株式会社
  - 技術支援
    - 日本マイクロソフト株式会社
      - 五十木秀一氏
    - アマゾンウェブサービスジャパン合同会社
      - 宇津井峻氏、櫻田武嗣氏、佐々木啓氏
    - 株式会社Fusic
      - 新川拓也氏、早崎司氏、槇原竜之輔氏、室井慎太郎氏、藤村直美氏