第5回 スーパーコンピュータ「不老」 ユーザ会

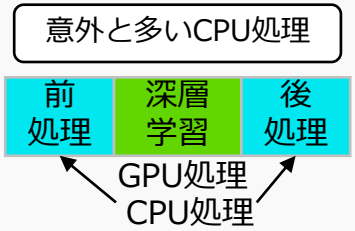
GPUを効率的に利用する技術：AI Computing Broker

富士通株式会社
コンピューティング研究所
大辻 弘貴

背景

- **GPUの効率的な利用**が求められている
  - 生成AIの発展による学習・チューニング・推論需要の増大
  - 需要に対して演算力の供給が追い付いていない
    →長いジョブ実行待ち時間・高止まりするコスト

- HPCシステムでは、**GPUは基本的に専有割り当て**が行われる
  - ジョブスケジューラがGPUジョブに対して必要数を割り当て
    - ジョブ間・プログラム間のGPU融通は基本的に行われない
  - 専有している一方で、本当のGPU利用率は必ずしも高くはない



処理内容にも踏み込む効率的なGPU割り当てが求められている

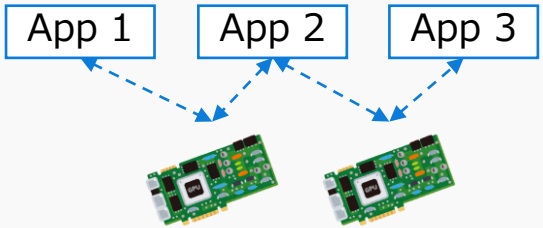
GPU割り当ての課題と解決の方向性

FUJITSU

- **アプリケーションに対して固定でGPUが割り当て**
  - 常にGPUが使用されているわけではない
    → CPU処理部分: e.g., データの前処理・後処理・探索・成形…
  - CPU処理の間にもGPUは割り当てられ続けている
    →GPUリソースが遊んでいる状態

“GPU pool” drawn by DALL・E3

- **AI Computing Broker (ACB)による動的なGPU割り当て**
  - ➔処理内容に踏み込み、プログラム間・ジョブ間で動的GPU割り当てを実現
  - ➔割当変更の際にメモリ領域の退避・復元を実施



AI Computing Broker (ACB)

FUJITSU

Realize more efficient use of GPU in AI learning process

Challenges in using GPU

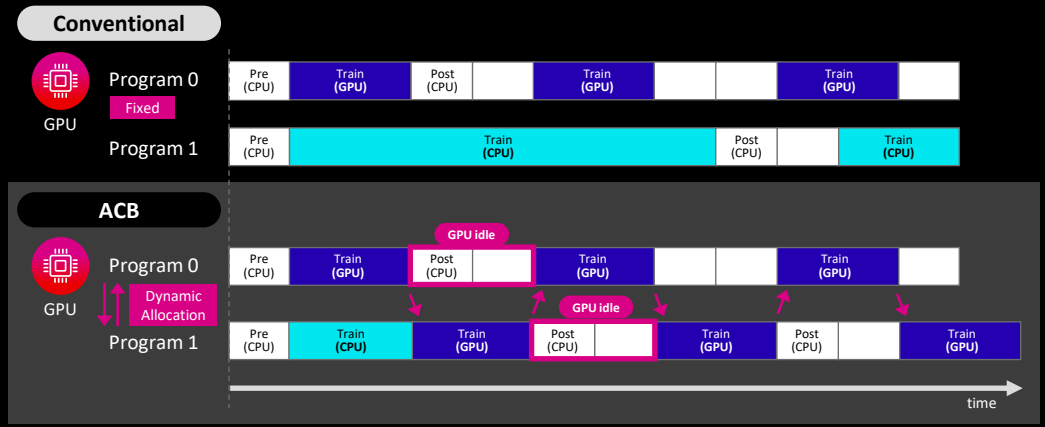
- Since the GPU is allocated on a per-program basis, even if the GPU idle time is large, other programs cannot use the GPU until the program ends.
- When virtualizing the GPU to share it among multiple programs, the GPU memory is divided for each program, reducing the capacity per program.

Features of AI Computing Broker

- GPU scheduling specialized for deep learning
- Control of GPU allocation at the processing unit level within the program
- Improves GPU utilization rate, reducing overall processing time
- More processes can be executed with the same GPU resources
- Unlike sharing of GPUs through virtualization technology, the program can fully utilize the GPU-equipped memory

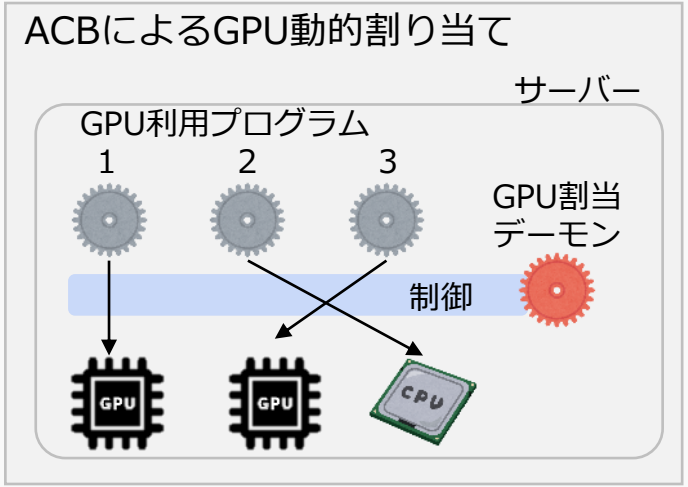
Technology of AI Computing Broker

Analyze AI processing content, dynamically allocate GPU to processes that require GPU



ACBの利用イメージ

- 要素
  - GPU割り当てデーモン（GPU Assigner）
  - 対象プログラム
    - Pytorch, TensorFlowなどを利用したアプリケーション(Python)
- 適用方法
  - 手動適用
    - コード内においてGPU利用前後に機能呼び出し
  - 自動適用
    - GPU関連の関数呼び出しを自動検出＆適用
- 実行方法
  - 専用のランチャー経由でプログラムを実行
  - **ユーザ権限で動作可能**



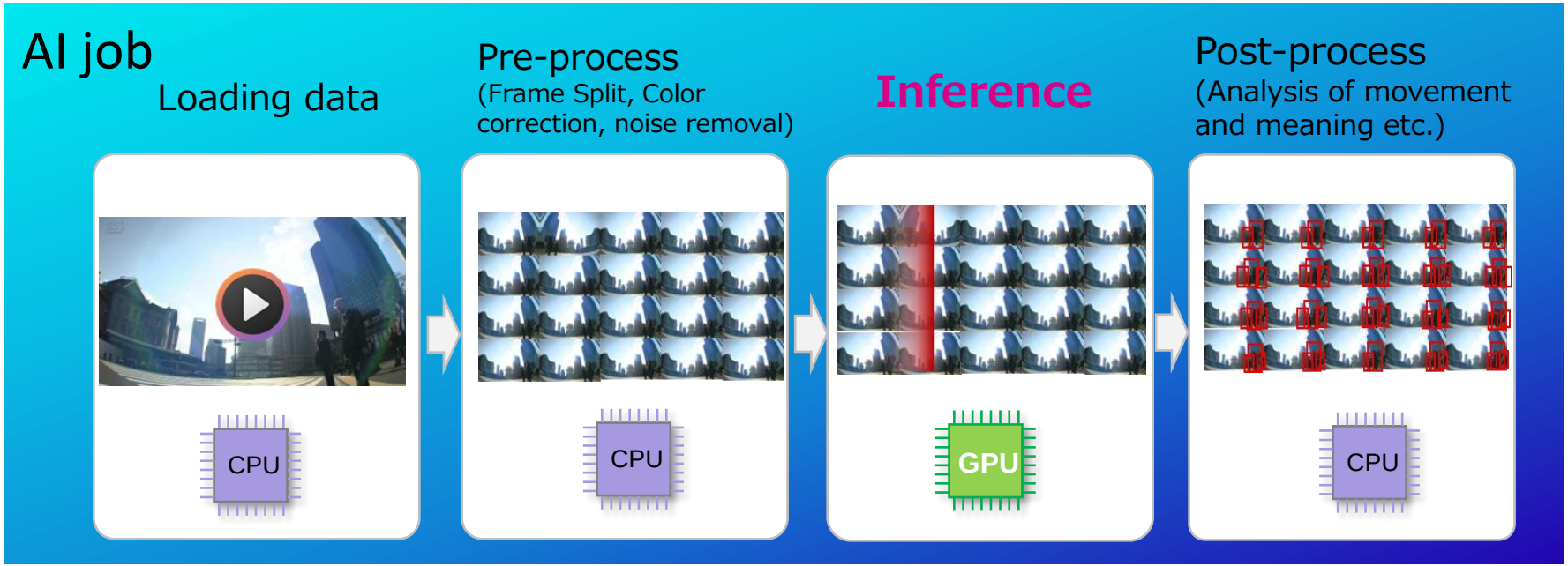
適用事例(1)

動画に対する物体検知（推論）処理の効率化

動画に対する推論処理におけるGPU利用率の低さ

FUJITSU

常に専有GPUを必要としているわけではない



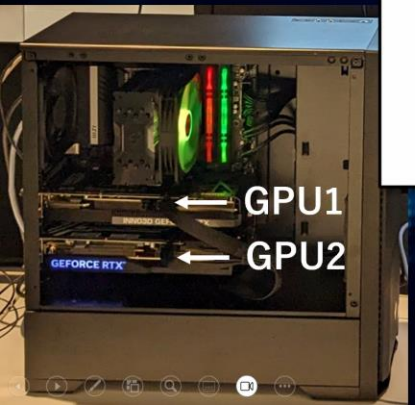
ACBはGPU利用処理に対して動的にGPUを割り当て

実行内容

FUJITSU

- 2つの動画ストリームに対する物体検出処理
  - フレームごとに物体検知（推論）を実施して、boundary boxを描画
- GPUが1枚のサーバで2つのストリームを同時に処理
  - ACBにより期待される効果：**スムーズ**な描画および**高いフレームレート**
- 比較内容
  - ACBを用いずに2つの動画を処理
  - ACBを適用した環境で2つの動画を同時に処理

2 GPU



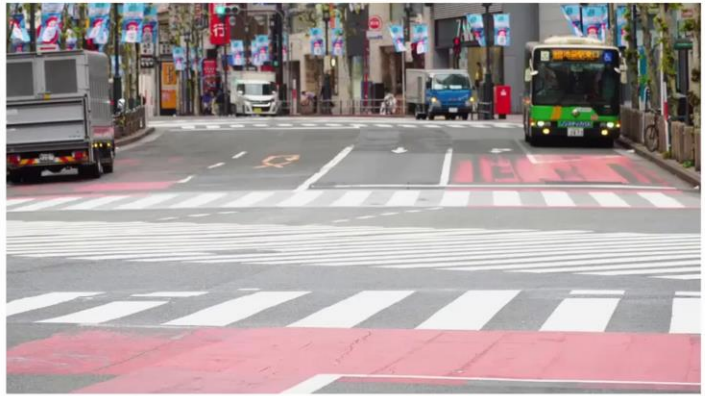
streamlit-m · Streamlit

localhost:8502

Conventional 2GPU AlphaFold2 Demo Dashboards

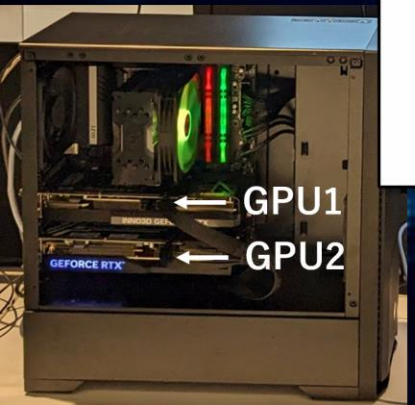
Deploy

2GPU



Start

2 GPU



streamlit-m · Streamlit

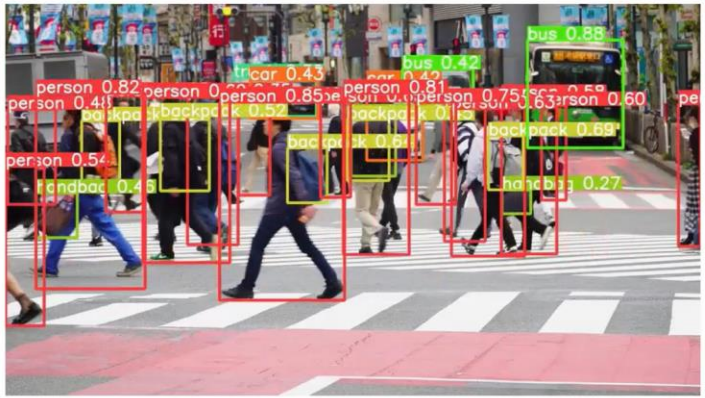
localhost:8502

Conventional 2GPU AlphaFold2 Demo Dashboards

RUNNING... Stop Deploy

2GPU

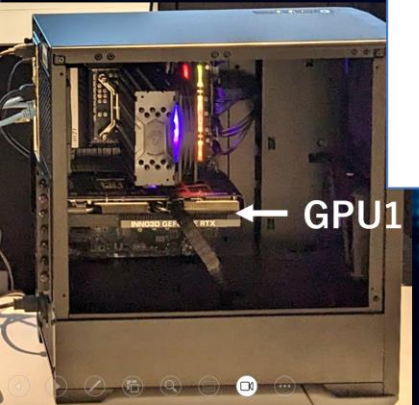
20.33 (fps)



20.35 (fps)

Start

1 GPU
(w/o ACB)



streamlit-m · Streamlit × | streamlit-m · Streamlit ×

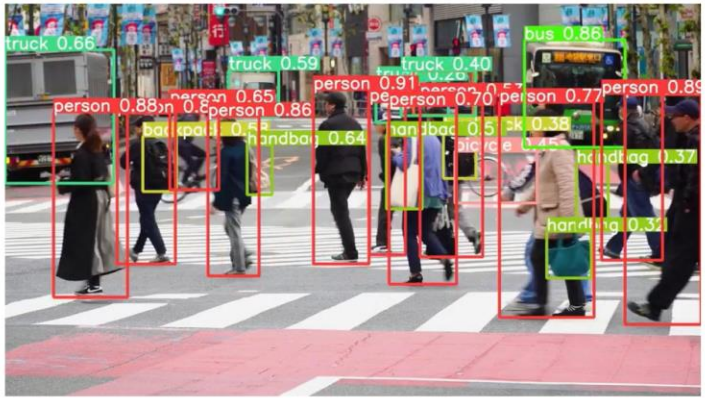
localhost:8501 100%

Conventional 1GPU | AI Computing Broke... | AlphaFold2 Demo | Dashboards

RUNNING... Stop Deploy

1GPU

20.10 (fps)

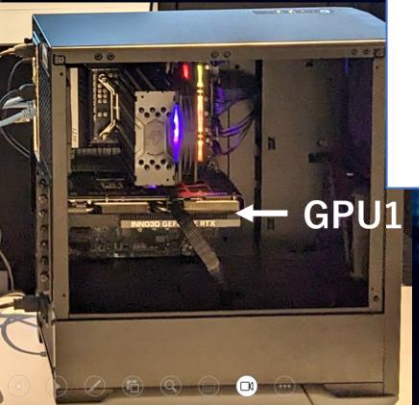


7.56 (fps)

Start

1 GPU + ACB

with ACB



streamlit-m · Streamlit × | streamlit-m · Streamlit ×

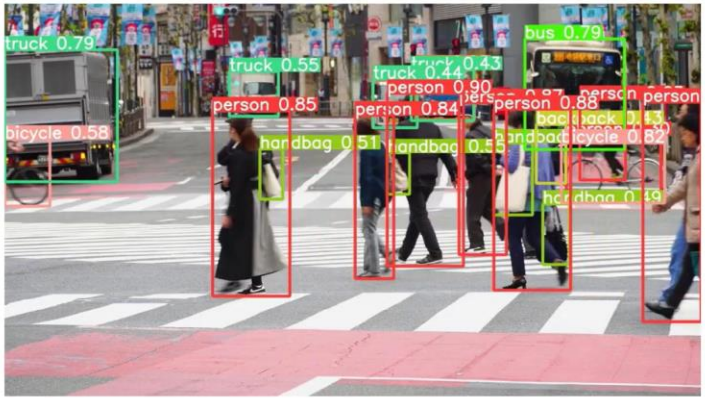
localhost:8502 100%

Conventional 1GPU | AI Computing Broke... | AlphaFold2 Demo | Dashboards

RUNNING... Stop Deploy

1GPU + Adaptive GPU Allocator

20.17 (fps)



19.97 (fps)

Start

1 GPU + ACB

with ACB

streamlit-m · Streamlit

localhost:8502

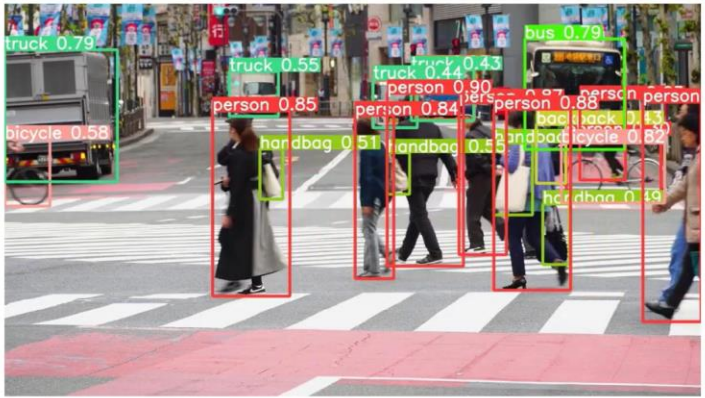
Conventional 1GPU | AI Computing Broke... | AlphaFold2 Demo | Dashboards

RUNNING... Stop Deploy

1GPU + Adaptive GPU Allocator

20.17 (fps)

19.97 (fps)



Start

ACBにより1つのGPUでスムーズに2つの動画を同時に処理

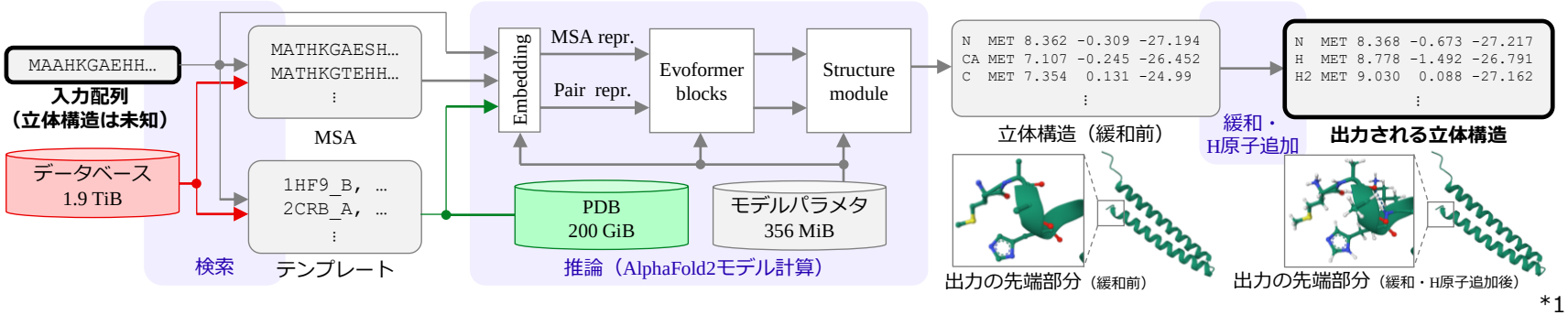
FUJITSU

適用事例(2)

複数のAlphaFold2プロセスに対する動的GPU割り当てによる高速化

「不老」Type II上における評価結果

AlphaFold2について

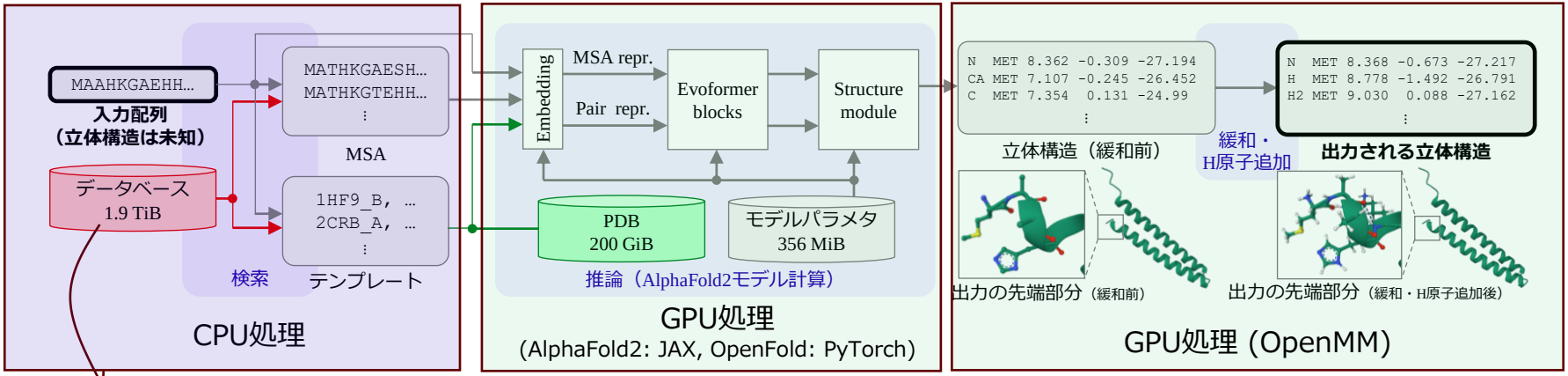


*1

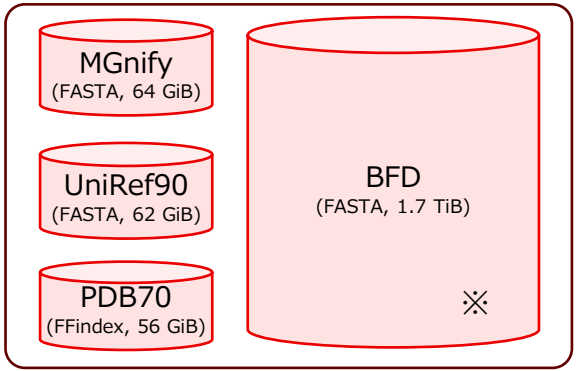
- 入力：タンパク質のアミノ酸配列
- 出力：予測されたタンパク質の構造 (原子座標)
- 処理
  - 検索：入力をもとに既知のタンパク質データベースを検索　CPU処理
  - モデル計算：入力配列と検索結果からモデル計算を行う　GPU処理
  - 緩和：モデル計算の出力構造に対してエネルギー最小化を行う　GPU処理

*1 Oyama et al, Accelerating AlphaFold2 Inference of Protein Three-Dimensional Structure on the Supercomputer Fugaku, FlexScience'23

AlphaFold2の処理



*1



<u>CPUとGPUを用いる処理が混在</u>している

CPU処理：
　4つのDBに対する検索処理
GPU処理：
　モデル計算（推論）および緩和・水素原子追加

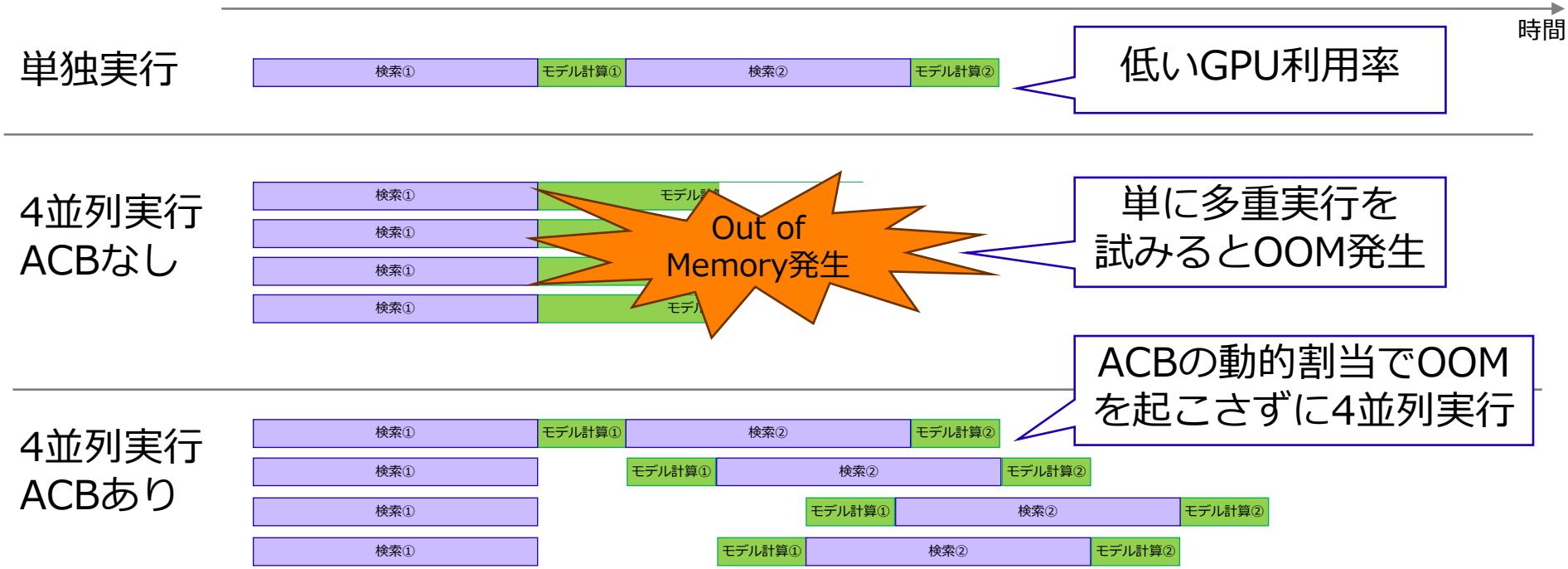
※今回の評価ではsmall_bfd (FASTA, 17 GiB)を利用

AlphaFold2に対するACBの適用と評価

FUJITSU

- 適用方法
  - ACBの自動適用機能を利用（対象プログラムに対するコード変更なし）
- 評価内容
  - ACBによるAlphaFold2 (OpenFold)実行スループット向上
- 評価環境
  - 「不老」Type IIシステム (V100 GPU x4)
- 評価方法および項目
  - ACBおよび対象プログラムをジョブとして投入して実行**（システム変更なし）**
    - ACBを用いた多重実行によるスループット向上効果の評価

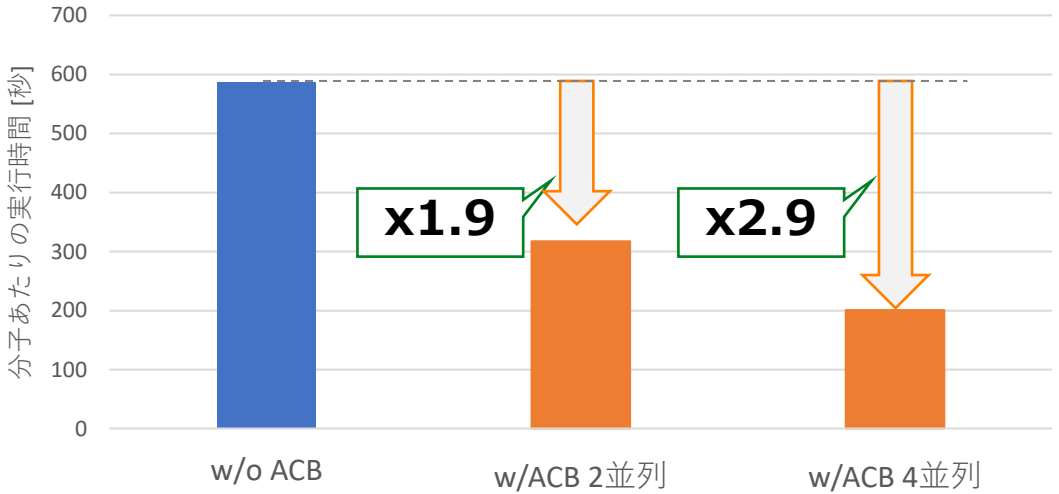
評価シナリオ



⇒ACBにより4つの処理を並列に行い、CPU・GPU処理を
交互に進めることでスループット向上を狙う

AlphaFold2に対するACBの適用と効果

- ACB無しとACBにより並列実行を行った場合を比較
  - 分子あたりの実行時間で比較



アミノ酸シーケンス長：400
使用GPU数: 1
※OpenFoldのメモリ動的管理機能を無効化

ACBの適用によりCPU・GPU混在プログラムを並列実行
GPU利用率が向上し、4並列時に2.9倍のスループット向上

FUJITSU

おわりに

ご利用（試行）イメージ

- FEFS共有領域に配置してあるACBパッケージ群をpipにて個人環境にインストールする
  - ご利用にあたっては、下記の問い合わせ先に確認ください
- TypeⅡサブシステム向けのリソースグループに対してジョブ（ACB有効化を含む）を投入する
  - 次頁にバッチジョブ記述例を掲載しているため準じて修正してください
- 詳しい動作条件などはお問い合わせください
  - 問い合わせ先：名古屋大学情報環境部 Q&A SYSTEM

ご利用（試行）イメージ

- バッチジョブ記述例（アプリケーションalphafold2の使用例）

```
#!/bin/sh
#PJM -L rscgrp=cx-single
#PJM -L node=1
#PJM -L elapse=24:00:00
#PJM -j

GPU=1
PARALLEL=2

module purge
module load gcc/11.3.0 cuda/11.8.0

. "/home/center/[userid]/miniforge3/etc/profile.d/conda.sh"

cd /home/center/[userid]/alphafold2
conda activate openfold_env

if [ ${GPU} -eq 1 ]
then
    gpu_assigner start --gpu-list 0
else
    gpu_assigner start
fi
time ./run_seq400_aga.sh $((${PARALLEL} * ${GPU}))
pkill gpu_assigner
```

不老で提供されている CUDA-11.8.0 を利用する場合

miniconda環境下で実行の場合

pip でインストールしたACB環境（gpu_assinger）を起動
※ 左記では GPU数:1 or 4 の分岐にしているが、特にGPU数を指定する必要性が無い場合は分岐文は不要

まとめ

FUJITSU

- GPUの効率的な利用が求められている
- アプリケーションまたぎのGPU割り当てにより効率化が可能
- ACB技術によりGPUの動的な割り当てとメモリ管理を実現
- GPU・CPU処理の並列実行によりアプリスループットを向上

Thank you