

GPU・CPU一体型モジュール におけるUnified Memory 使用時の性能評価

情報科学類 情報システム専攻 202110949

吉田 智

指導教員 朴 泰祐, 藤田 典久

2025/2/13 (木)

GPUとUnified Memory

- GPU (Graphical Processing Unit)

- 高い演算性能と電力効率

- ➡ HPC分野での利用拡大

- ✗ プログラミングの生産性低下

- GPUメモリの管理、転送制御

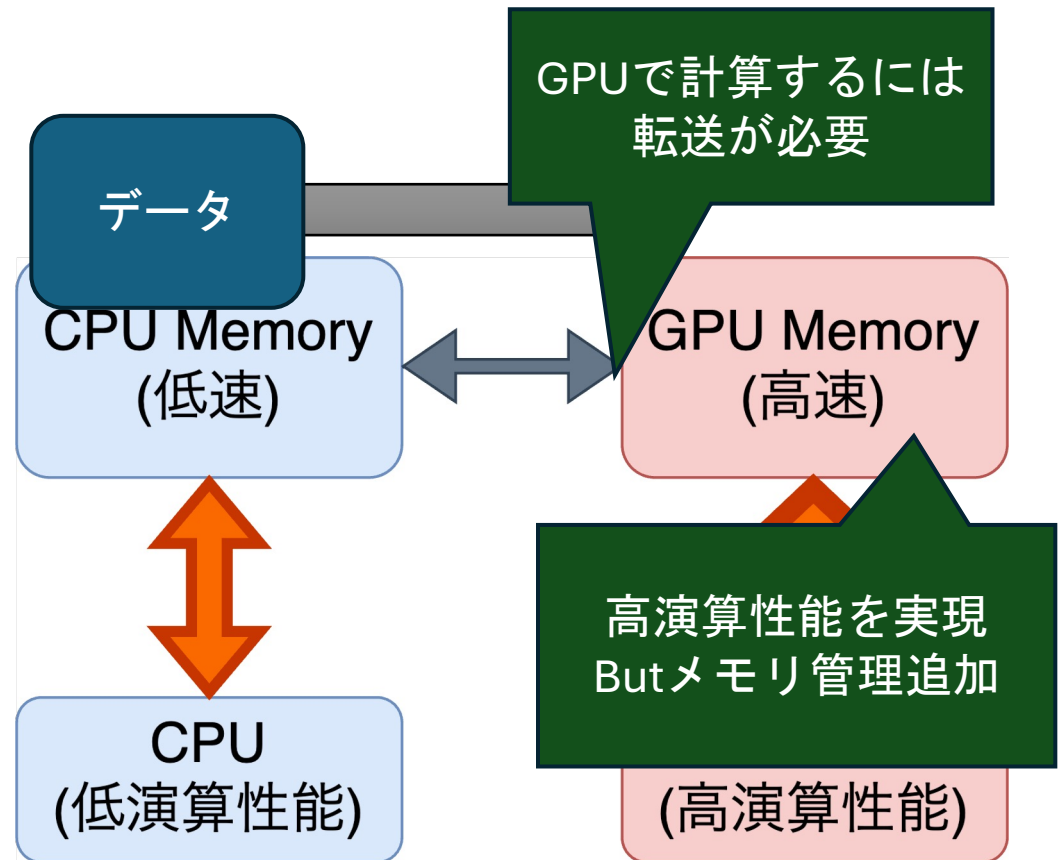
- CUDA

- NVIDIA製GPU向け開発環境

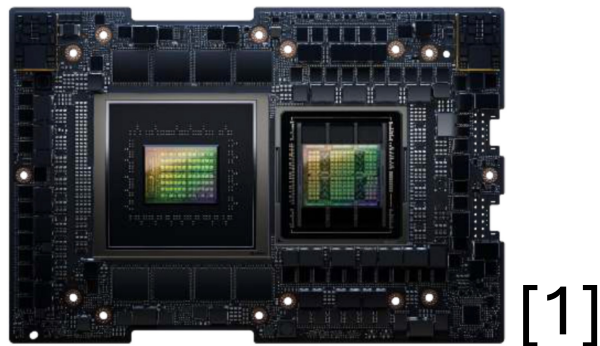
- Unified Memory (UM)

- GPU, CPU両方からアクセスできるメモリ空間を提供

CPUメモリ LPDDR5X	GPUメモリ HBM3	バス Nvlink-C2C
512GB/s	4TB/s	450GB/s

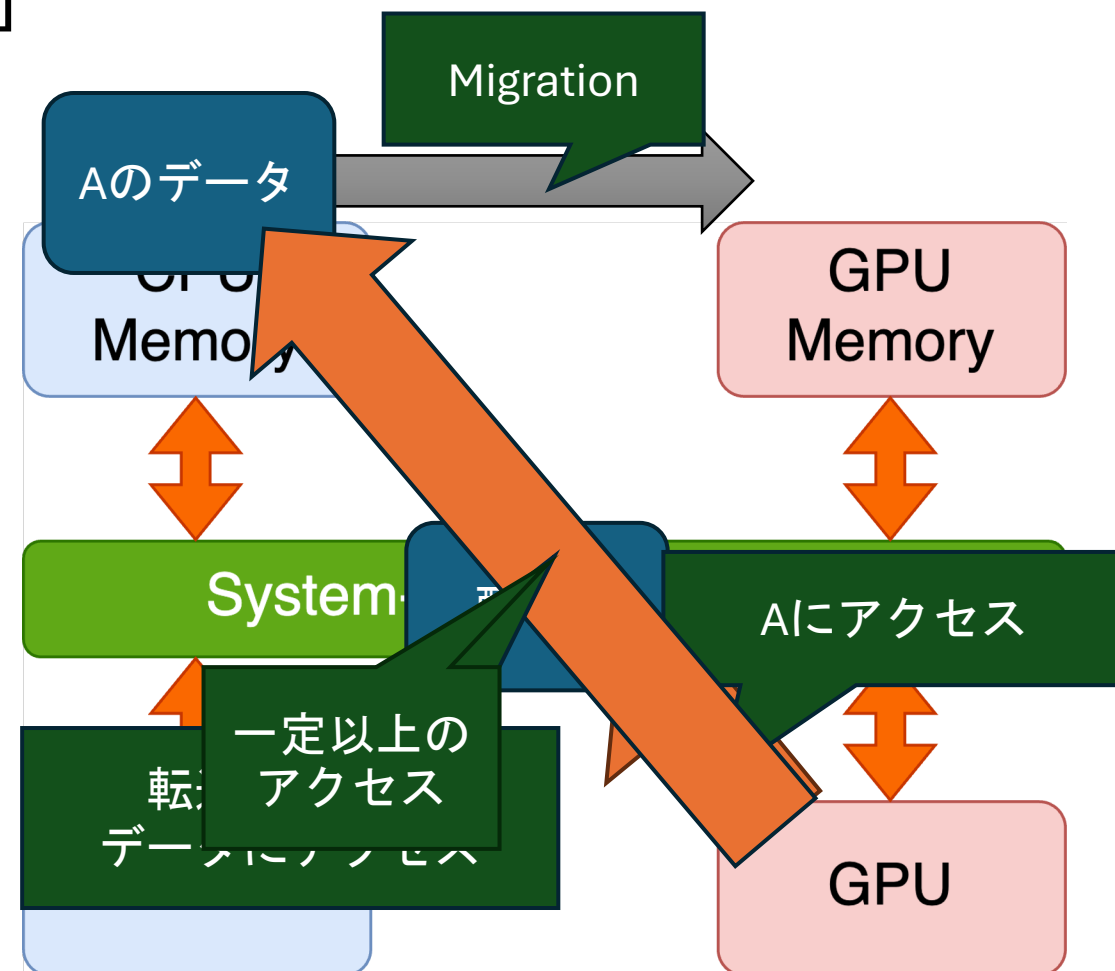


GH200



- GPU・CPU一体型モジュール
 - 一枚の基板上に実装
 - 高速なバスで2つを接続
- System-Allocated Memory (SAM)
 - GH200が提供する新しいUM
 - 転送処理なしにアクセス可
 - Migration
 - 一定以上のアクセスでデータが移動

CPUメモリ LPDDR5X	GPUメモリ HBM3	バス Nvlink-C2C
512GB/s	4TB/s	450GB/s



研究目的

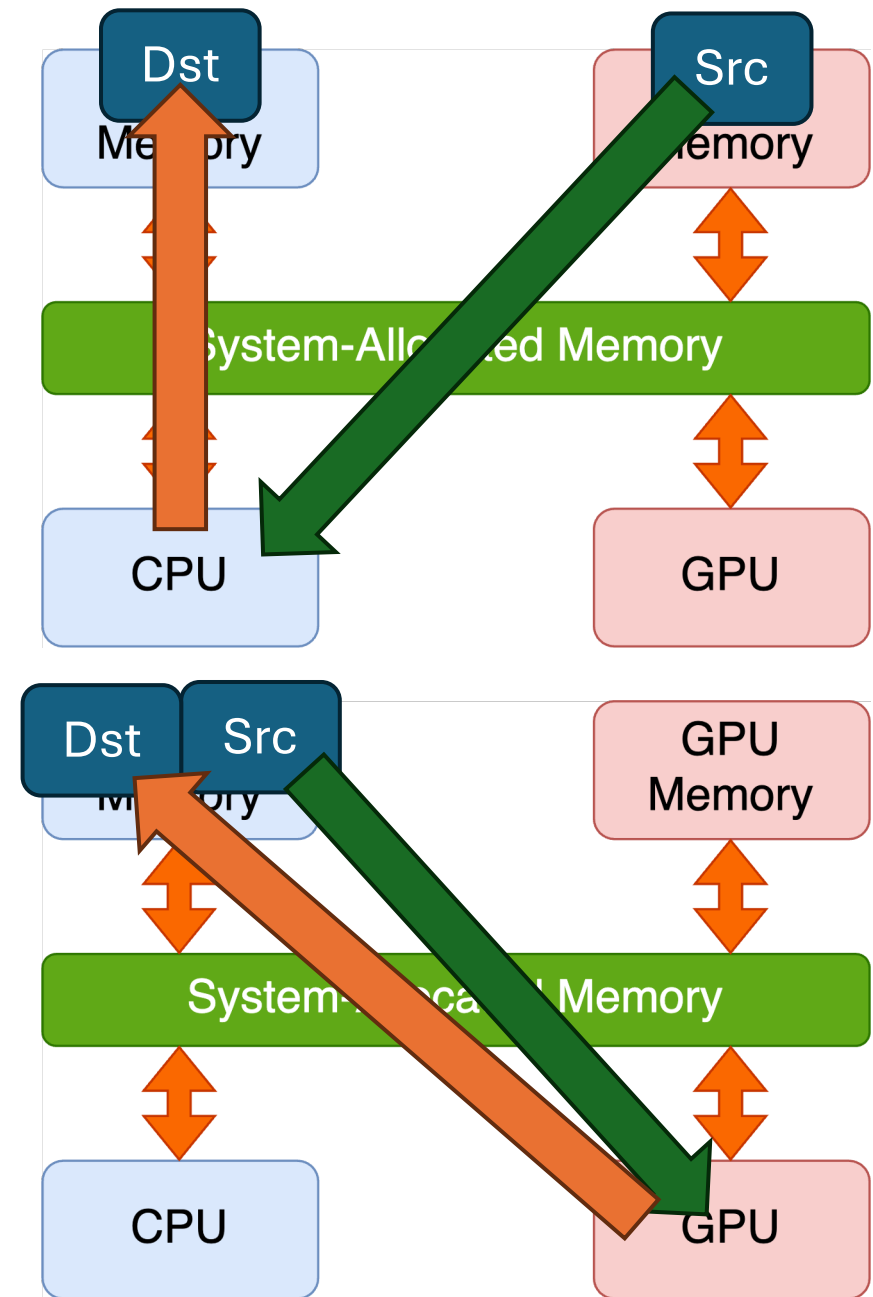
- GH200はほとんど使用経験がない
- GH200のメモリアクセスへの影響を理解
 - 様々なアクセスパターンにおける性能測定
 - Migrationの動作
 - どんなプログラムなら恩恵を受けられるか
- GH200の有効性の評価
 - 既存のシステムとの比較
 - 性能、プログラムの生産性はどうなるか

実験 (1/2)

- SAM上のメモリ性能の評価
 - 8パターンのメモリ性能を測定
 - この発表では2パターン取り上げる
 - 配列Src, Dstを一方に格納
 - $\text{Dst}[i] = \text{Src}[i]$ をGPU or CPUで実行
 - 連続で200回性能測定
 - これを10回行う。

上 (1) Src: GPUM Dst: CPUM 処理: CPU

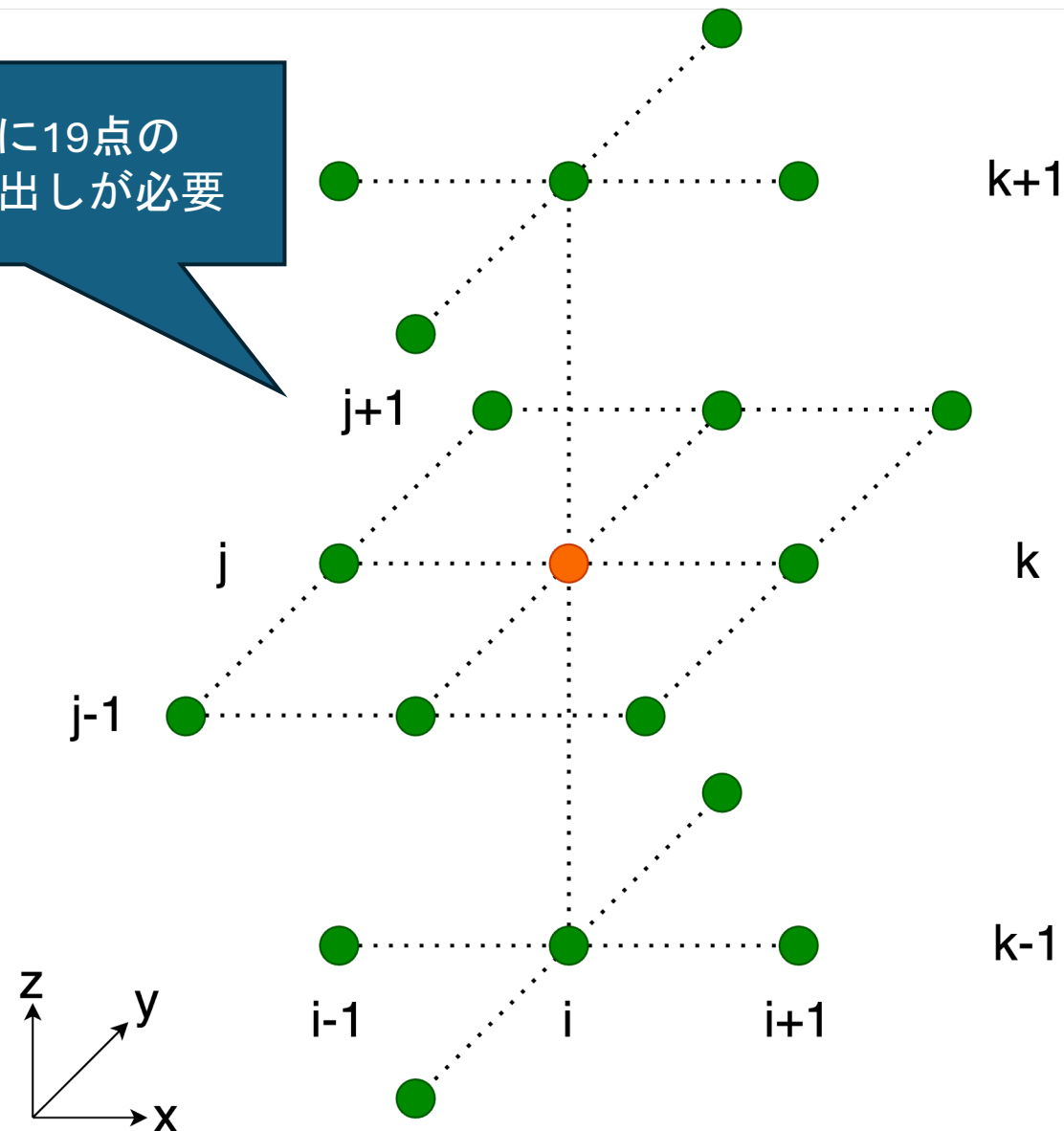
下 (2) Src: CPUM Dst: CPUM 処理: GPU



実験 (2/2)

- 姫野ベンチマーク[2]
 - ポアソン方程式を解く処理を実行
 - 格子の全ての点で計算
 - メモリ性能に依存
- 3バージョンのGPU化
 - 通常のGPU化
 - UMを使用
 - SAMを使用
- 既存システム、GH200システムで実行し評価

1 点の計算に19点のデータの読み出しが必要



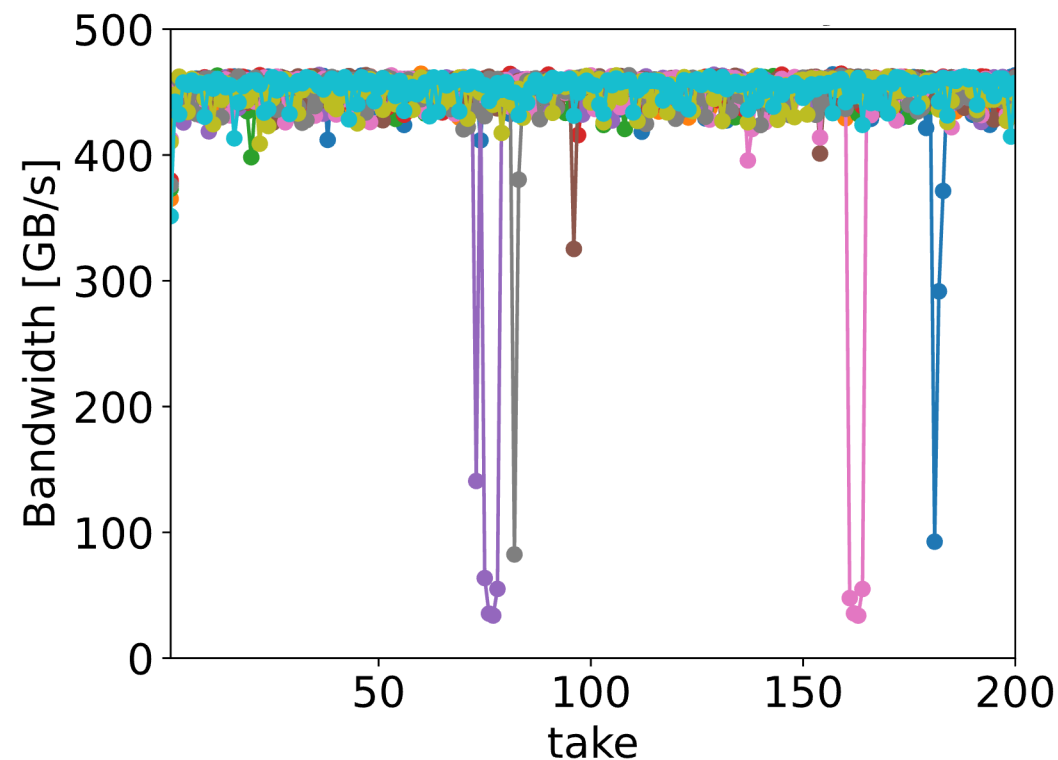
実験環境

- Miyabi [3]
 - 最先端共同HPC基盤施設が運用
 - 2025年1月14日より稼働開始
 - GH200を搭載した国内唯一のスパコン
- Pegasus [4]
 - 筑波大学計算科学研究センターが運用
 - 従来型のGPU・CPUシステム
 - GPUとしてH100を搭載
- 両方とも1ノードを使用



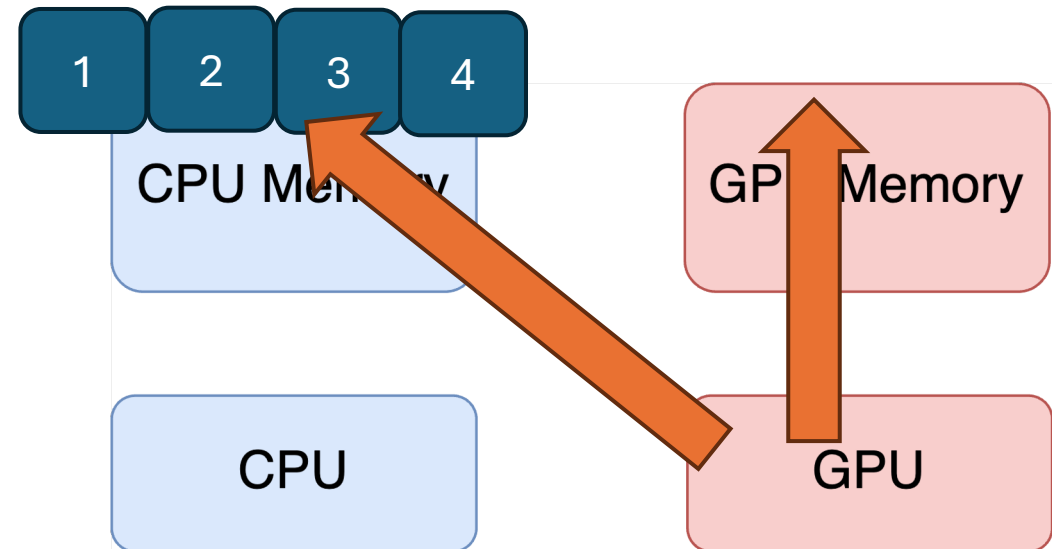
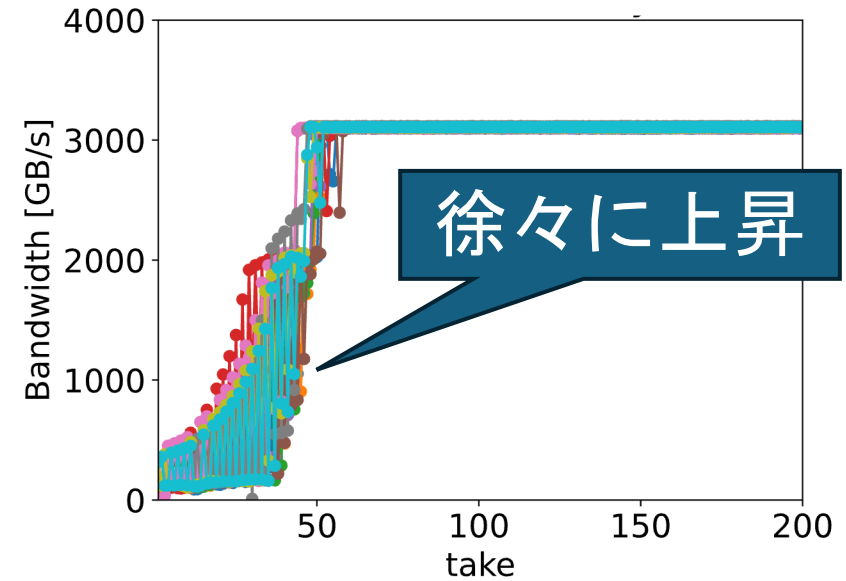
性能評価 (1/4)

- SAM上でのメモリ性能
 - 配列サイズ: 4GB
 - 右: (1)の結果
 - GPU-CPU間バスの理論性能
450GB/s
 - それと同等の性能
 - 性能が落ちるスパイクが発生
- GPU-CPU間のデータ転送の多い
プログラムに恩恵



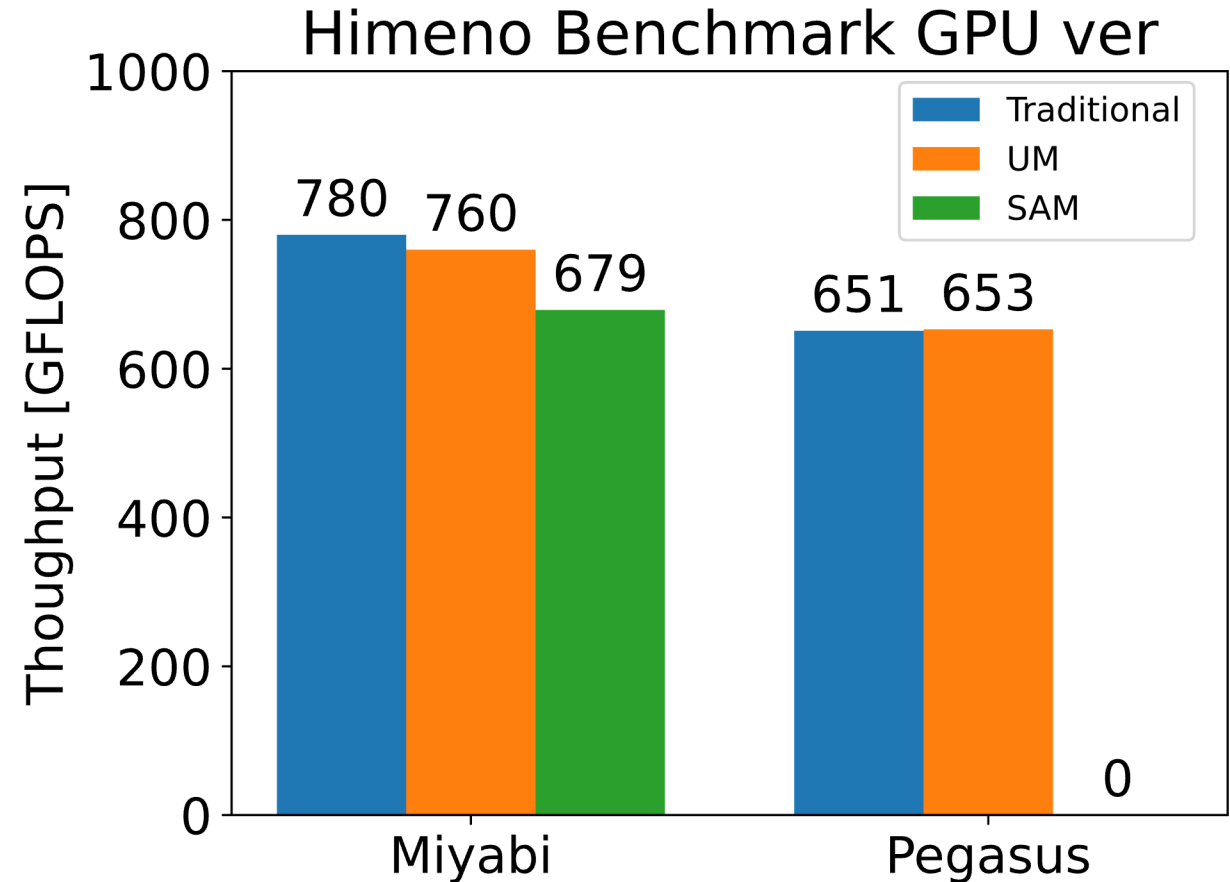
性能評価 (2/4)

- SAM上のメモリ性能
 - 配列サイズ: 4GB
 - (2)の結果
 - GPUメモリの理論性能4TB/s
- 徐々に性能が上昇
 - Migrationによるもの
 - 少しずつデータを移動させている



性能評価 (3/4)

- 姫野ベンチマーク
 - サイズ512*512*1024
 - 3000回反復
 - 100GFLOPS低下
 - SAMの性能が若干低
 - 転送が最初だけ
 - Migration前が遅い



性能評価 (4/4)

- 姫野ベンチマーク

- 生産性の改善

- UMでもデータ転送制御が必要
 - SAMではそれも不要

通常のコード例

```
struct Matrix {  
    ....  
    float *m  
}
```

Matrix A, B, C, ...

// GPU化コード

```
Matrix *dA, *dB, *dC..  
float *dAm, *dBm, *dCm  
cudaMalloc()
```

...

```
cudaMalloc()  
cudaMemcpy()
```

...

```
cudaMemcpy()
```

...

```
cudaMemcpy()
```

...

```
for 3000 {compute( )}
```

UMのコード例

```
struct Matrix {  
    ....  
    float *m  
}
```

Matrix A, B, C, ...

// GPU化コード

```
Matrix *dA, *dB,*dC..  
cudaMalloc()
```

...

```
cudaMemcpy()
```

...

```
for 3000 {compute( )}
```

SAMのコード例

```
struct Matrix {  
    ....  
    float *m  
}
```

Matrix A, B

// GPU化コード

```
for 3000 {compute()}
```

まとめ

- SAM上のメモリアクセスの評価
 - データ転送の多いプログラムに有効
- 姫野ベンチマーク
 - UMよりも生産性を改善
- GH200は有効であると言える。
- 今後の課題
 - より多くのベンチマークでの性能評価
 - マルチノードでのSAMの影響の評価