Unified Memory Supercomputers in CCS

藤田 典久
筑波大学 計算科学研究センター 助教

筑波大学のスーパーコンピュータ

筑波大学
計算科学研究センター
Center for Computational Sciences

Retired

Sirius (PACS 12.0)

SIRIUS
PACS12.0

Cygnus
(Apr. 2019 - March 2025)
2.4 PFLOPS Peak

Node Configuration:
2x Intel Xeon Processors (Skylake)
4x NVIDIA V100 GPUs
4x Bittware 520N FPGA Boards
(Intel Stratix 10 FPGA)

Pegasus
(Apr. 2023 -)
8.1 PFLOPS Peak

Node Configuration:
1x Intel Xeon Processor (Sapphire Rapids)
1x NVIDIA H100 GPU (PCIe)
128GB DDR5 Memory + 2TB Intel Optane Persistent Memory per node

Miyabi
(Jan.2025 -)
80.1 PFLOPS Peak

GPU Node Configuration:
1x NVIDIA GH200 Superchip
(Grace CPU+Hopper GPU)
Co-operation with U. Tokyo (as JCAHPC)

Sirius (PACS 12.0)
(Planned: 2026Q1 -)
11.9 PFLOPS Peak

Node Configuration:
4x AMD MI300A APU
(Zen4 GCDs + CDNA3 XCDs)

(96x MI300A in total)

Unified Memory型スーパーコンピュータ

- 現在，2つのユニファイドメモリ型スーパーコンピュータを運用中
- Miyabi (NVIDIA)
  - NVIDIA GH200搭載スーパーコンピュータ
  - 筑波大学と東京大学が共同で運営
    （Joint Center for Advanced High Performance Computing: JCAHPC）
  - 80.1 PFLOPS 倍精度演算浮動小数点数演算ピーク

- Sirius PACS 12.0 (AMD)
  - CCSに導入予定のAMD MI300A搭載スーパーコンピュータ
  - 2026年Q1に運用開始予定
  - 11.9 PFLOPS 倍精度演算浮動小数点数演算ピーク
  - 本公演はこちらを主に扱う

SIRIUS
PACS12.0

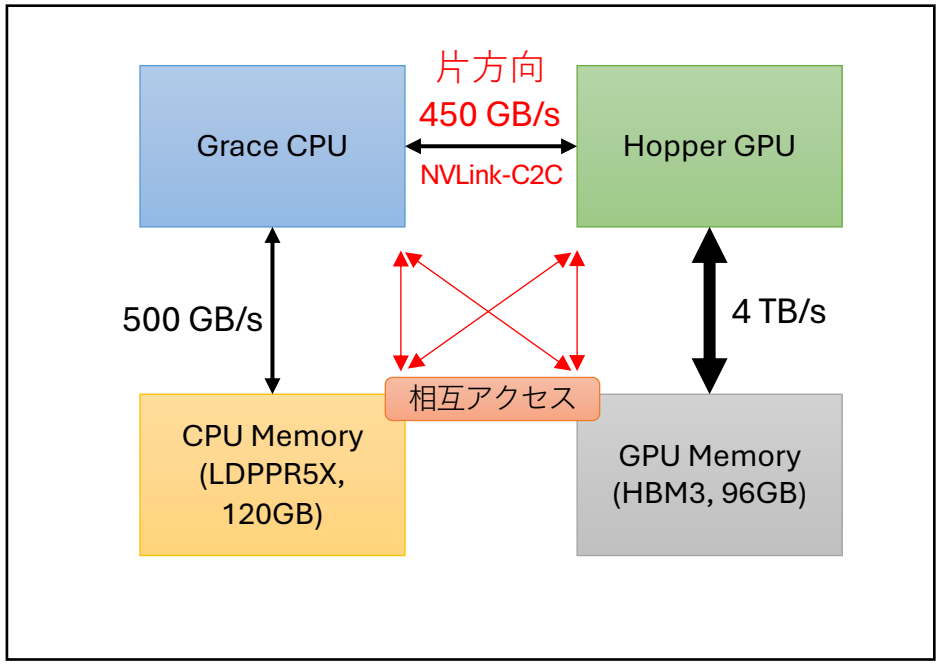
なぜUnified Memoryを採用するのか

- 課題：GPUプログラミングは難しい
  - CPUとGPUのメモリが別れている事がプログラミングを複雑化している
  - 計算処理のGPU化だけでも大変なのに，考慮しなければならない事柄が増えてしまう
    - GPUメモリの確保解放，CPU-GPU間データ転送，転送タイミング
- Unified Memoryがこの問題を軽減できると考えている

利点	欠点
GPUメモリ管理やデータ転送が根本的に不要となる	CPUとGPUの間で共有されるリソースが生まれ，性能低下やノイズが懸念される
プログラミングの容易化	どのような要素が性能に影響を与えるか明らかではない

NVIDIA GH200

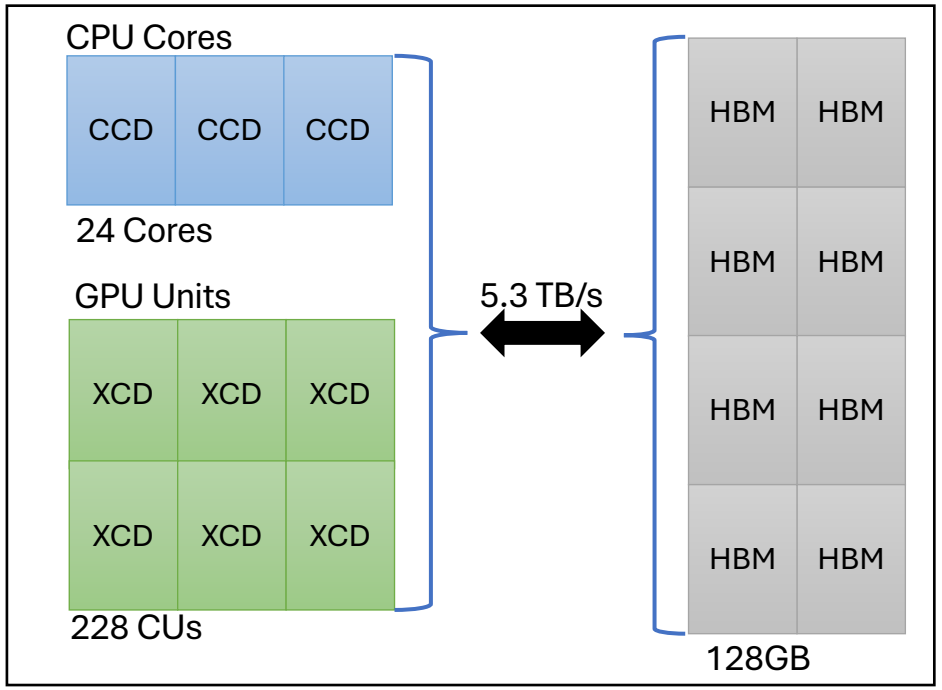
- Grace CPUとHopper GPUを1つのモジュールとして構築した製品
- CPUとGPUが別々のメモリを有するもののUnified Memoryを提供
  - NVLink-C2Cによる広帯域低レイテンシな相互結合
- CPU/GPUはCPU Memory/GPU Memoryに相互にメモリアクセスができる
  - アクセス頻度に応じたデータの自動移動（Migration）
  - High performance CUDA Managed Memory



GH200 module

AMD MI300A APU

- MI300 is an APU
  - CPUコアとGPUユニットが同じHBM3メモリを共有
  - キャッシュコヒーレントが保たれる
- MI300は9つのダイから構成される（MCM）
  - 3x Core Chiplet Dies (CCDs)
    - = CPU コア
    - 8x Zen4 cores / CCD; 24 Cores 合計
  - 6x Accelerated Compute Dies (XCDs)
    - = GPU ユニット
    - 38x Compute Units (CUs) / XCD; 228 CUs 合計
  - ただし，ソフトウエアからは，これらのダイはモノリシックに扱える
    - NUMAではない（分割設定も可能）
- HBM3メモリ
  - 128 GBの容量と5.3 TB/sの帯域



MI300Aの予備性能評価

- 性能評価
  - CPU / GPU メモリ帯域（BabelStream）
  - CPU / GPU 演算性能（SGEMM / DGEMM）
- 注：Siriusはインストール中のため，以降の性能評価には量子科学技術研究開発機構（QST）が運用するPlasma Simulator（PS）を用いる
  - SiriusとPSはほぼ同等のアーキテクチャであるため，同等の性能が得られると考えている
  - ただし，PSは水冷だが，Siriusは空冷（Power Limit設定は同じ）

MI300Aメモリバンド幅

- CPUとGPUは同じメモリを共有するが，実効メモリバンド幅は異なる
  - CPU BW = 190 GB/s, GPU BW = 3.8 TB/s
- GPU帯域は高い性能が得られるが，CPU帯域は制限されている
  - DDR5相当だが，PCIe経由のCPU-GPU間帯域よりは高速
  - APUあたり3 GCD (24コア) しか実装されていない
  - CPU L3 ⇔ Infinity Fabric の帯域が限られているように見える

Memory bandwidth of CPU and GPU
BabelStream* v5.0, Array Size = 8.0 GB x 3

	Copy	Mul	Add	Triad	Dot
CPU (OpenMP)	186 GB/s	186 GB/s	195 GB/s	195 GB/s	197 GB/s
GPU (HIP)	3827 GB/s	3877 GB/s	3772 GB/s	3780 GB/s	3206 GB/s

* github.com/UoB-HPC/BabelStream

Note: Measured on Plasma Simulator at QST, Japan. One of four APU is used with ROCm 6.4.3. GB=2^{30}.

MI300A GEMM演算性能

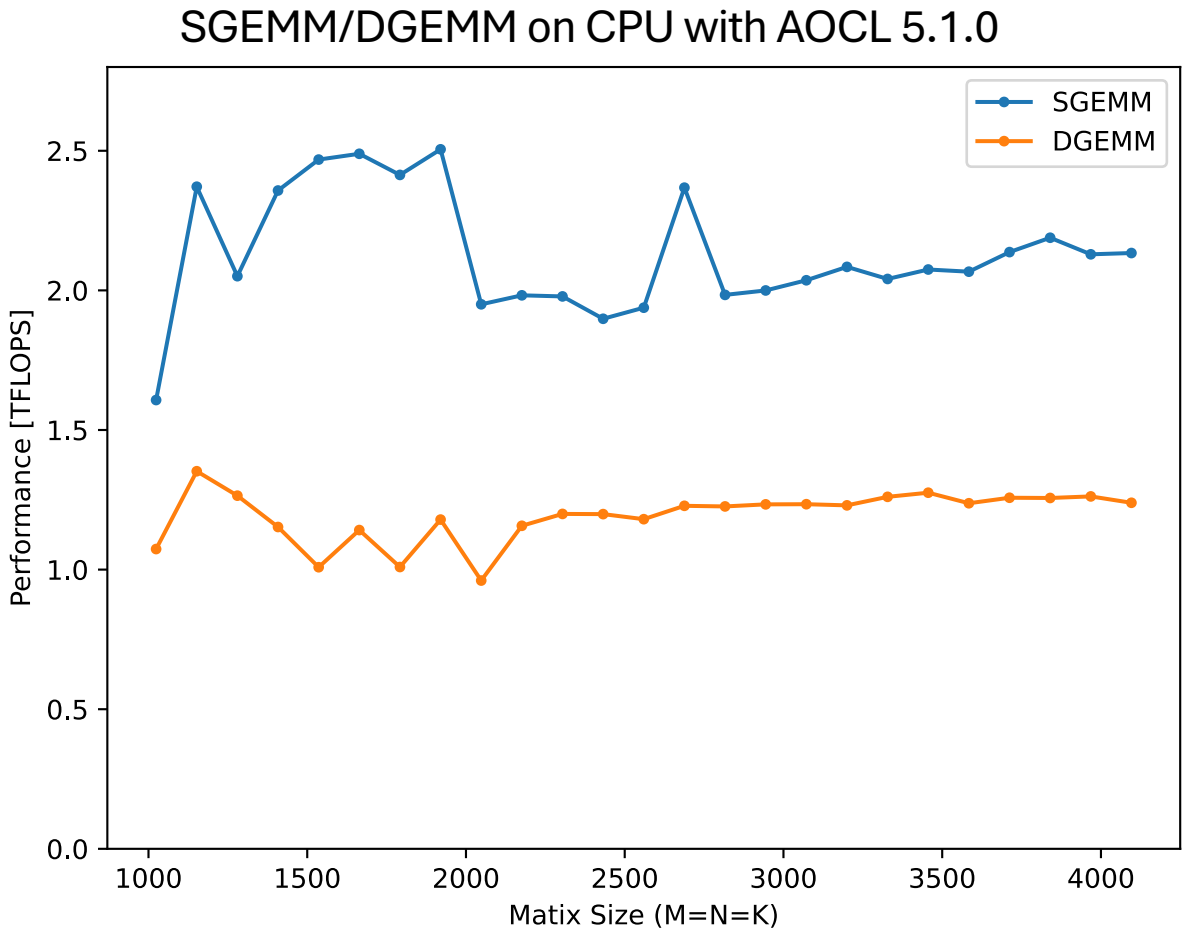
- 演算理論ピーク性能
  - CPU (24x Zen4 Cores): 2.8 TFLOPS (SP), 1.4 TFLOPS (DP)
  - GPU (228x CDNA3 CUs): 122.6 TFLOPS (SP / SP Dense Matrix),
    61.3 TFLOPS (DP), 122.6 TFLOPS (DP Dense Matrix)
    - SP:DP = 2:1 だが matrix fused-multiply-add (MFMA) 命令が DP Dense Matrix演算を加速する
- ほとんどの演算性能はGPUによって得られる
  - CPU:GPU = 1:44 for SP/DP scalar or 1:88 for DP matrix
  - 実測でも 40x (SGEMM), 58x (DGEMM) の性能差がある

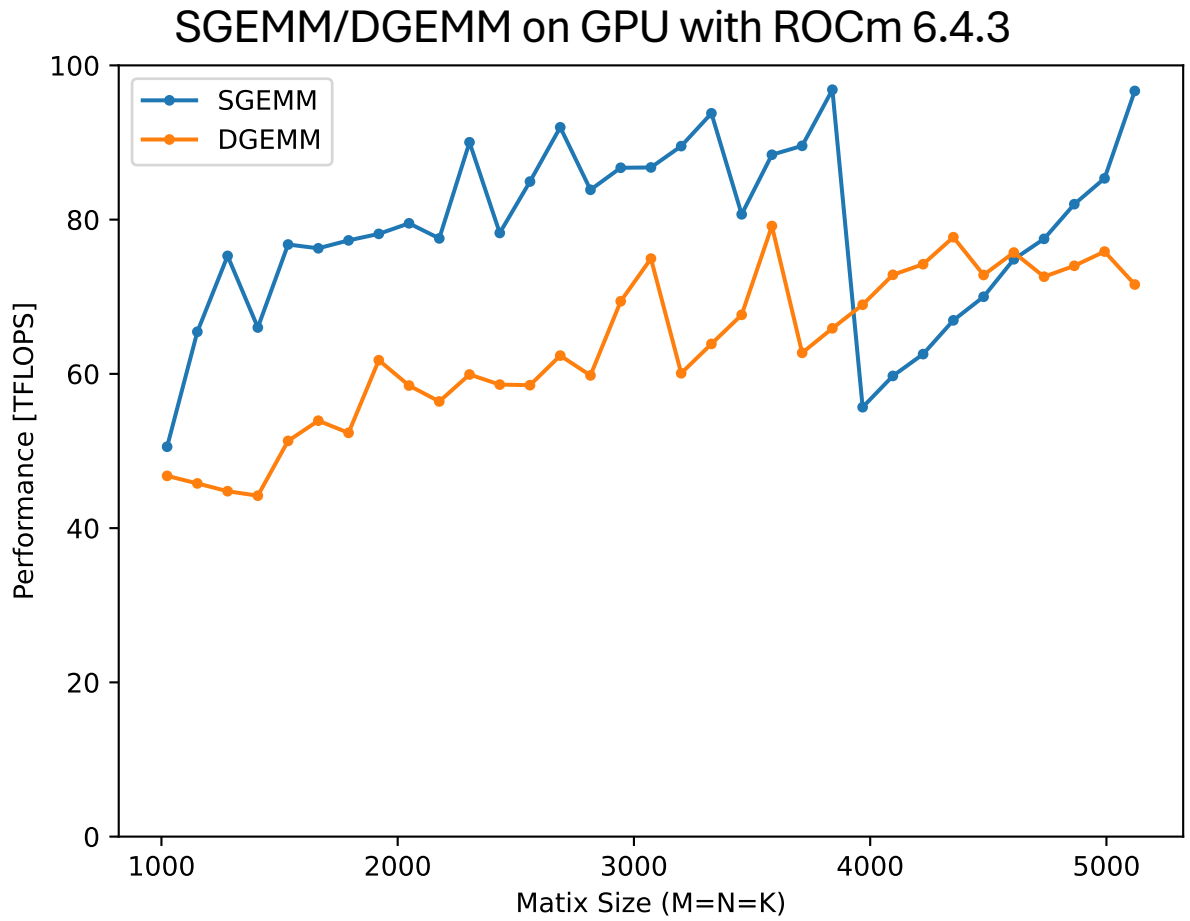
SGEMM and DGEMM performance of CPU and GPU
(M=N=K; NoTrans; AOCL BLIS-mt for CPU, and hipBLAS for GPU)

	SGEMM	DGEMM
CPU	2.50 TFLOPS (N=1920)	1.35 TFLOPS (N=1152)
GPU	98.6 TFLOPS (N=3840)	79.1 TFLOPS (N=3584)

Note: Measured on Plasma Simulator at QST, Japan. One of four APU is used with AOCL 5.1.0 and ROCm 6.4.3.

MI300A GEMM Performance in Details





Strange performance drop around N=4096.

NVIDIAからAMDへの移植

- CUDAコード
  - AMD公式変換ツール hipify-perl / hipify-clang
  - CUDAコードをHIPコードに自動的に変換できる
  - cuBLAS, cuFFT等も対応できる
  - 相互に共通の機能がない場合は対応できない
  - コードベース変換であり，最適化などは行われない
- OpenACCコード
  - OpenACCコードをそのままAMD GPUで動かすのは現状難しい
  - OpenMPに変換することが次善策か
- OpenMPコード（Target）
  - AMD, NVIDIAともにコンパイラが提供されている
  - 性能の可搬性については今後の検証が必要
- その他 言語・ライブラリ
  - Kokkos, Julia等

NVIDIA Platform		AMD Platform
CUDA	Hipify	HIP
OpenACC	変換?	OpenMP
OpenMP Target	As-is?	OpenMP Target

Portability Layer
Kokkos (C++ Library)
Raja (C++ Library)
Julia
SYCL (C++ based)
CuPy (Pyhton Library)
…等

Unified Memory Programming with OpenMP

Data mapがCPU-GPU間でデータ転送をするために必要

```
#include <cstdio>


int main() {
    float* a = new float[10];
    float* b = new float[10];
    float* c = new float[10];

    for (int i = 0; i < 10; i++) {
        a[i] = i; b[i] = i; c[i] = 0; }

#pragma omp target data map(to:a[:10],b[:10]) map(from:c[:10])
#pragma omp target teams distribute parallel for
    for (int i = 0; i < 10; i++) {
        c[i] = 2.0f * a[i] + b[i];
    }

    for (int i = 0; i < 10; i++) {
        printf("c[%d] = %f¥n", i, c[i]); }
    return 0;
}
```

GPUで実行される領域

Traditional (Separated) Memory Model

Unified shared memory (USM)が必須と宣言

Data mapは必要ない

```
#include <cstdio>
#pragma omp requires unified_shared_memory

int main() {
    float* a = new float[10];
    float* b = new float[10];
    float* c = new float[10];

    for (int i = 0; i < 10; i++) {
        a[i] = i; b[i] = i; c[i] = 0;


#pragma omp target teams distribute parallel for
    for (int i = 0; i < 10; i++) {
        c[i] = 2.0f * a[i] + b[i];
    }

    for (int i = 0; i < 10; i++) {
        printf("c[%d] = %f¥n", i, c[i]); }
    return 0;
}
```

Unified Shared Memory Model

実行

コンパイルオプションは同一

Traditional (Separated) Memory Model

```
$ amdclang++ sample.cpp -O3 -fopenmp --offload-arch=gfx942
$ ./a.out
c[0] = 0.000000
c[1] = 3.000000
c[2] = 6.000000
c[3] = 9.000000
c[4] = 12.000000
c[5] = 15.000000
c[6] = 18.000000
c[7] = 21.000000
c[8] = 24.000000
c[9] = 27.000000
```

HSA_XNACK=1でUSMを有効化する

Unified Shared Memory Model

```
$ amdclang++ sample.cpp -O3 -fopenmp --offload-arch=gfx942
$ HSA_XNACK=1 ./a.out
c[0] = 0.000000
c[1] = 3.000000
c[2] = 6.000000
c[3] = 9.000000
c[4] = 12.000000
c[5] = 15.000000
c[6] = 18.000000
c[7] = 21.000000
c[8] = 24.000000
c[9] = 27.000000
```

まとめ

筑波大学
計算科学研究センター
Center for Computational Sciences

SIRIUS
PACS12.0

- 筑波大学 計算科学研究センターでは2つのUnified Memory型スーパーコンピュータを運用中
  - Miyabi (NVIDIA GH200)
  - Sirius PACS 12.0 (AMD MI300A)
    - 現在設置中であり運用開始は2026年Q1を予定
  - GH200とMI300Aは似て非なるアーキテクチャとなっている
- AMD MI300A APU
  - CPUとGPUが融合されているが，大半の演算性能はGPUがもたらす
    - GPUの利用が必須であり，CPUのみを使うべきではない
  - Unified Shared MemoryはGPUプログラミングを簡素化する
    - メモリの管理やデータ転送は不要となる
    - OpenMP GPU (target) は，CPU OpenMPプログラミングと同等の複雑度でGPUプログラミングを実現できる
- 今後の課題として，実アプリでの性能評価や生産性の評価がある