Introduction to CUDA Parallel Programming
CUDA平行計算導論

https://ceiba.ntu.edu.tw/1092Phys8061_CUDA

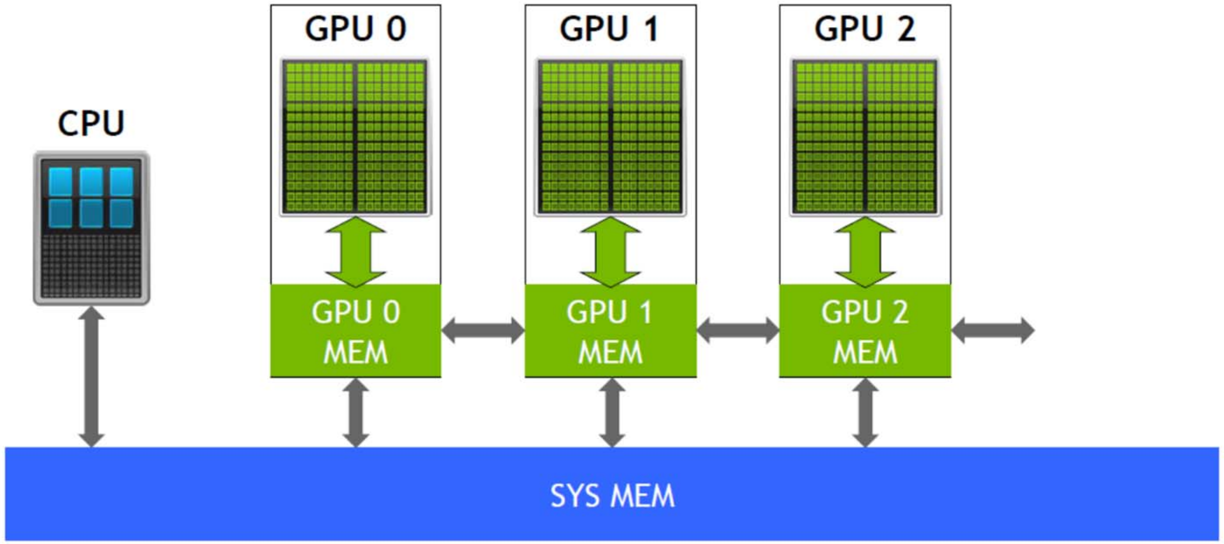
Professor Ting-Wai Chiu (趙挺偉)
Email: twchiu@phys.ntu.edu.tw
Physics Department
National Taiwan University

This lecture will cover:

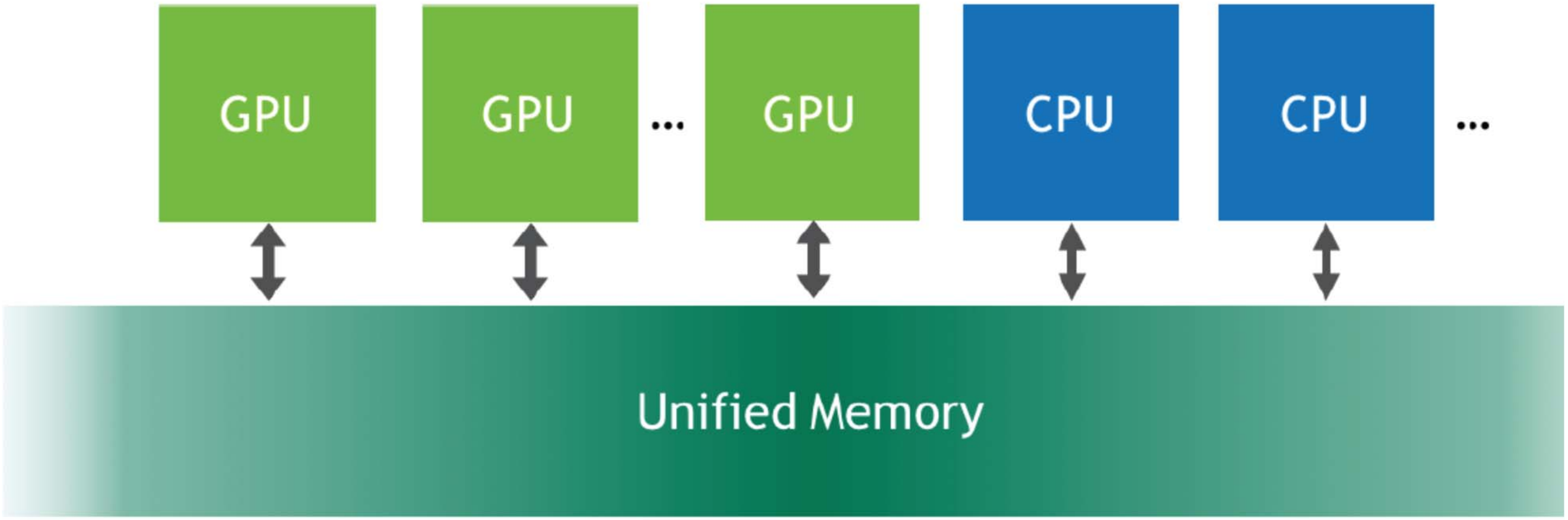
- Unified Memory in CUDA
- Vector Addition with Unified Memory

Heterogeneous Architectures with CPUs & GPUs

2 Nvidia GPU cards on the motherboard

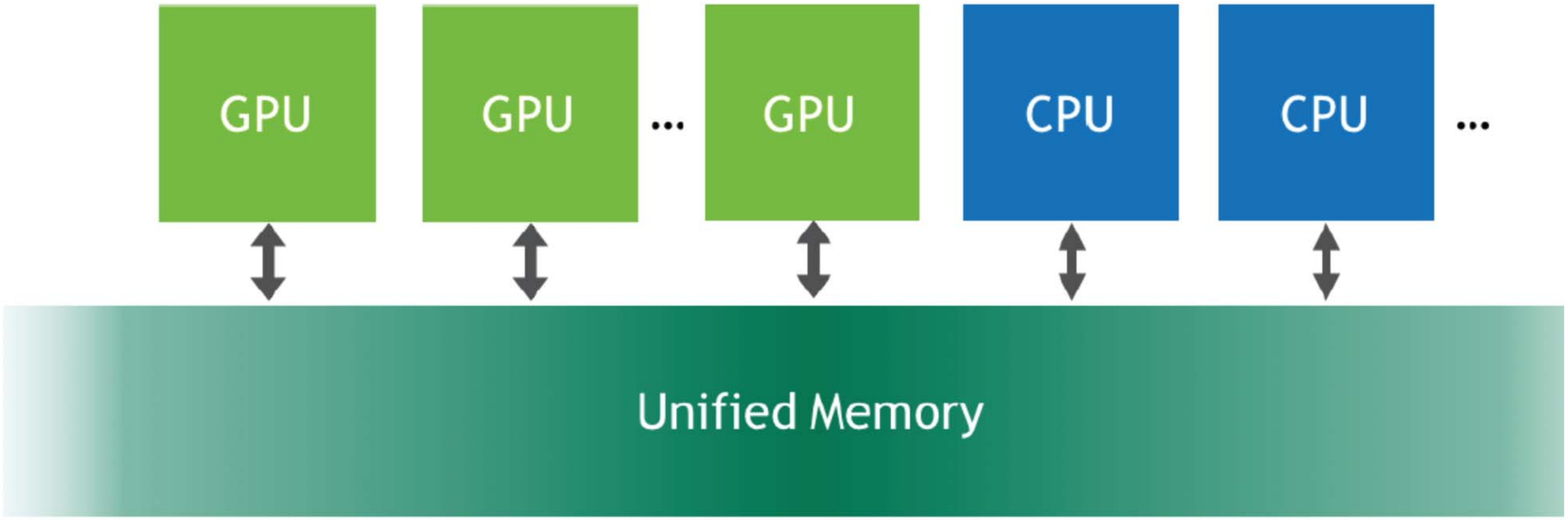


Unified Memory in CUDA



Unified Memory in CUDA is a single address space accessible from any processor in a system. This hardware/software technology allows applications to allocate data that can be read or written from code running on either CPUs or GPUs.

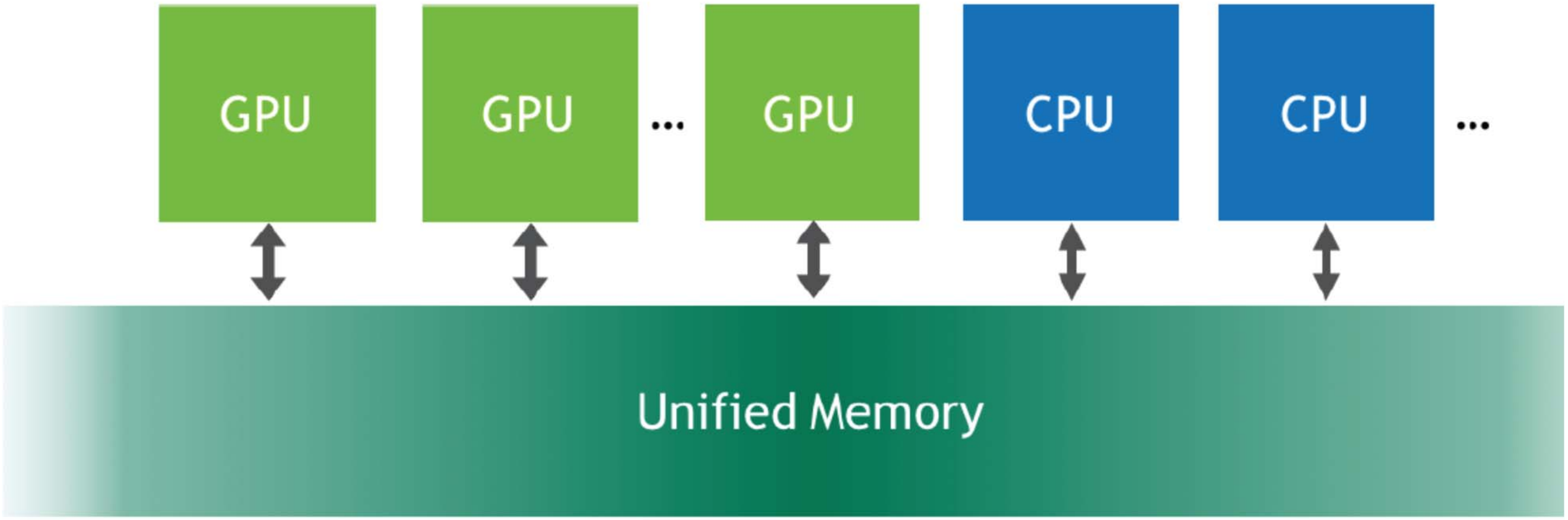
Unified Memory in CUDA



Unified Memory in CUDA is a single address space accessible from any processor in a system. This hardware/software technology allows applications to allocate data that can be read or written from code running on either CPUs or GPUs.

```
malloc( )
cudaMalloc( )
```

Unified Memory in CUDA

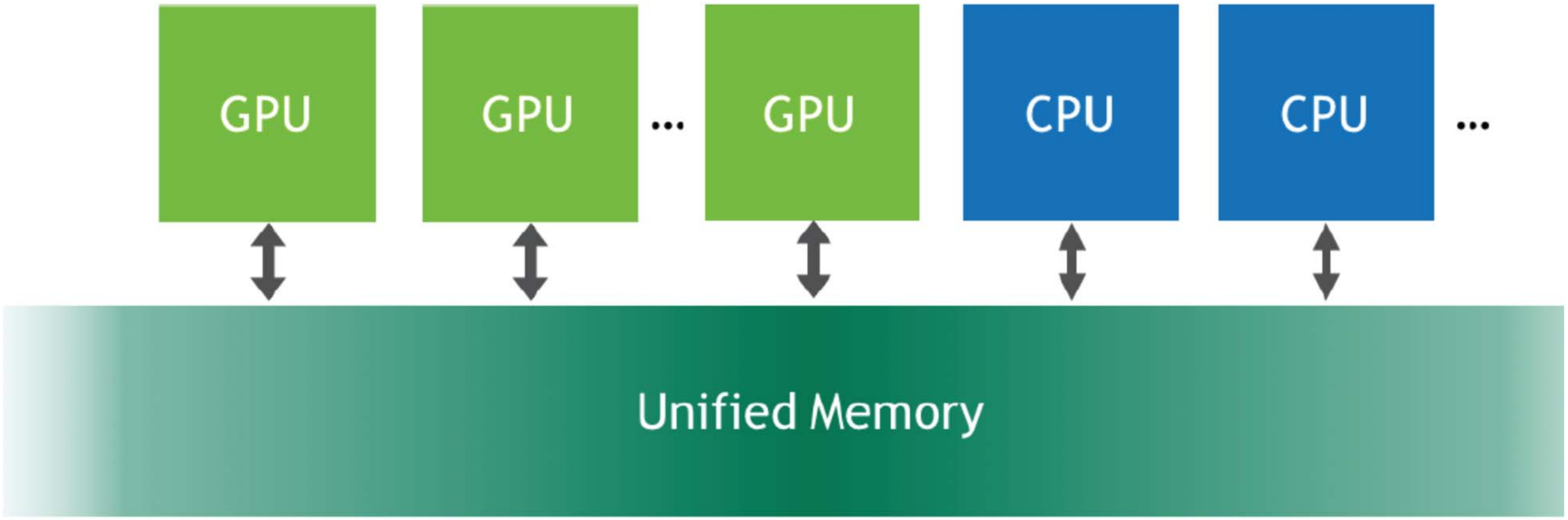


Unified Memory in CUDA is a single address space accessible from any processor in a system. This hardware/software technology allows applications to allocate data that can be read or written from code running on either CPUs or GPUs.

```
malloc( )
cudaMalloc( )
```

```
cudaMallocManaged( )
```

Unified Memory in CUDA



Unified Memory in CUDA is a single address space accessible from any processor in a system. This hardware/software technology allows applications to allocate data that can be read or written from code running on either CPUs or GPUs.

```
cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice)
cudaMemcpy(h_A, d_A, size, cudaMemcpyDeviceToHost)
```

Vector Addition with Multi-GPUs

```
__global__ void VecAdd(const float* A, const float* B, float* C, int N)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if (i<N) C[i]=A[i]+B[i];
    __syncthreads();
}

int main(void) {
...
 omp_set_num_threads(NGPU);
 #pragma omp parallel private(cpu_thread_id)
 {
   cpu_thread_id = omp_get_thread_num();
   cudaSetDevice(cpu_thread_id);
   cudaMalloc((void**)&d_A, size/NGPU);
   cudaMalloc((void**)&d_B, size/NGPU);
   cudaMalloc((void**)&d_C, size/NGPU);
   cudaMemcpy(d_A, h_A+N/NGPU*cpu_thread_id, size/NGPU, cudaMemcpyHostToDevice);
   cudaMemcpy(d_B, h_B+N/NGPU*cpu_thread_id, size/NGPU, cudaMemcpyHostToDevice);
   VecAdd<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N/NGPU);
   cudaDeviceSynchronize();
   cudaMemcpy(h_C+N/NGPU*cpu_thread_id, d_C, size/NGPU, cudaMemcpyDeviceToHost);
 ...
 }
...
}
```

Vector Addition with Multi-GPUs (using Unified Memory)

```
__global__ void VecAdd(float* A,  float* B,  float* C,  int N,  int NGPU,
                       int  cpu_thread_id)
{
    int offset = N/NGPU*cpu_thread_id;
    int i = blockDim.x * blockIdx.x + threadIdx.x + offset;
    if (i<N)  C[i] = A[i] + B[i];
    __syncthreads();
}

int main(void) {
...
 cudaMallocManaged((void**)&A, size); //  Allocate the Unified Memory for CPU/GPU
 cudaMallocManaged((void**)&B, size);
 cudaMallocManaged((void**)&C, size);

 omp_set_num_threads(NGPU);
 #pragma omp parallel private(cpu_thread_id)
 {
   cpu_thread_id = omp_get_thread_num();
   cudaSetDevice(cpu_thread_id);
   VecAdd<<<blocksPerGrid, threadsPerBlock>>>(A, B, C, N, NGPU, cpu_thread_id);
   cudaDeviceSynchronize();
 }
...
}   // see the complete code in twqcd80:/home/cuda_lecture_2021/vecAdd_NGPU_umem
```

Unified Memory in CUDA

- What is the underlying mechanism of the unified memory ?
  How does it work ?
- Under what circumstances the unified memory can enhance
  the performance (maximally) ?
- The efficiency of unified memory depends on the hardware (the GPU)
  as well as the version of CUDA
- For further readings

  Beyond GPU Memory Limits with Unified Memory on Pascal
https://devblogs.nvidia.com/beyond-gpu-memory-limits-unified-memory-pascal/
  Maximizing Unified Memory Performance in CUDA
https://devblogs.nvidia.com/maximizing-unified-memory-performance-cuda/
  Unified Memory for CUDA Beginners
https://devblogs.nvidia.com/maximizing-unified-memory-performance-cuda/