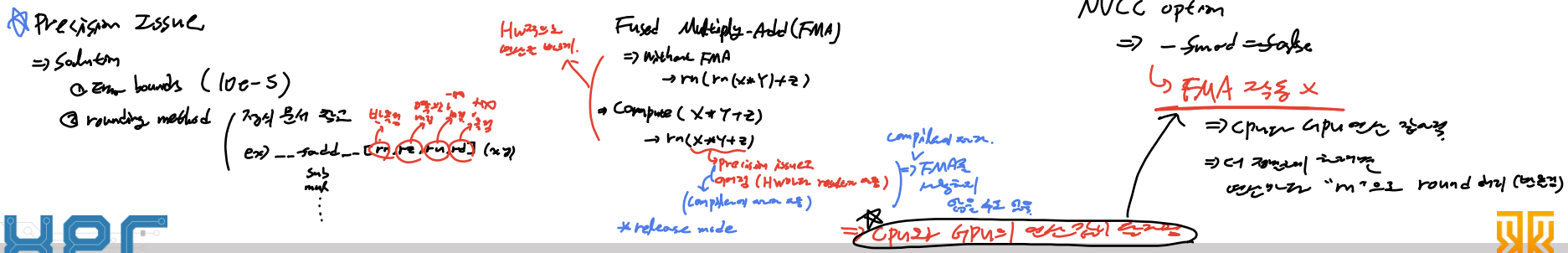
Lecture 12-1

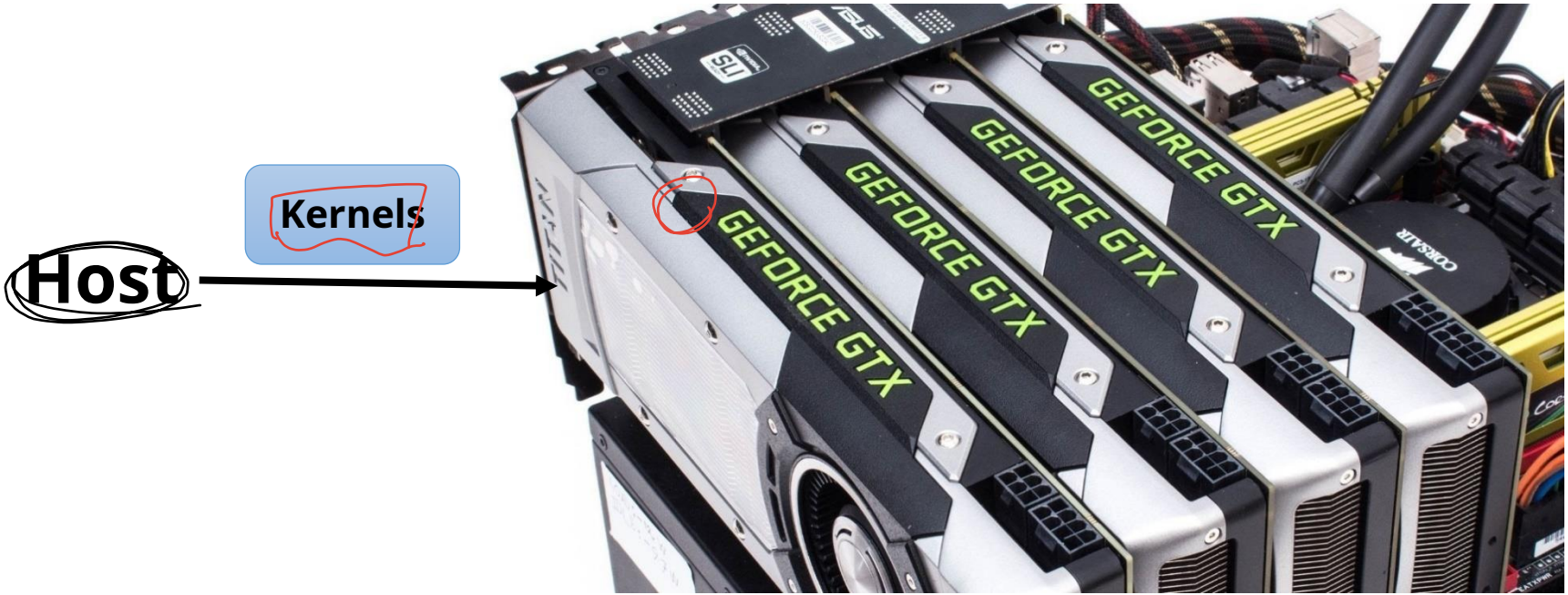
Multi-GPUs

Getting more power!



How Many GPUs Do You Have?

How Many GPUs Can You Use?



Getting Information of Your System

- **The number of CUDA-enable devices**
  - cudaError_t **cudaGetDeviceCount** (int *count);

- **Querying properties of each device**
  - cudaError_t **cudaGetDeviceProperties** (cudaDeviceProp* prop, int deviceID)
  - cudaDeviceProp structure
    - See the programming guide [link]

```
struct cudaDeviceProp {
    char name[256];
    cudaUUID_t uuid;
    size_t totalGlobalMem;
    size_t sharedMemPerBlock;
    int regsPerBlock;
    int warpSize;
    size_t memPitch;
    int maxThreadsPerBlock;
    int maxThreadsDim[3];
    int maxGridSize[3];
    int clockRate;
    size_t totalConstMem;
    int major;
    int minor;
    size_t textureAlignment;
    size_t texturePitchAlignment;
    int deviceOverlap;
    int multiProcessorCount;
    ...
```

Query GPU Information

typing ①

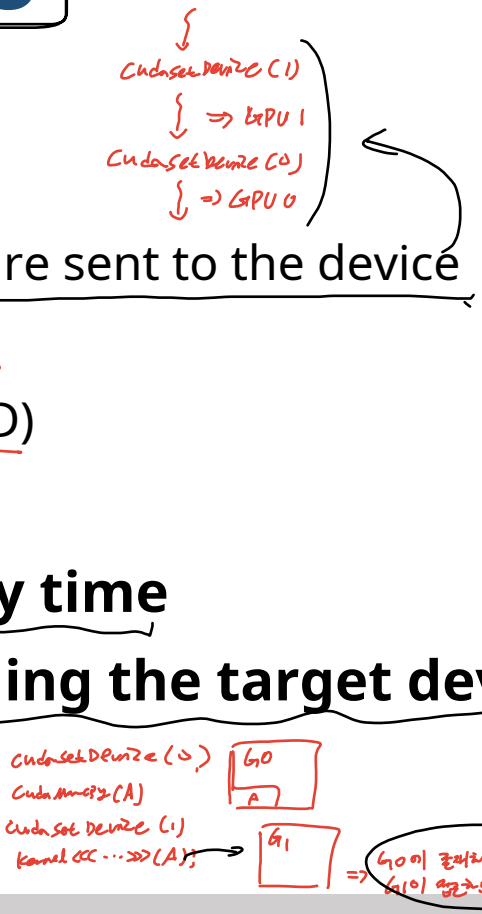
```
void main(void) {
    int ngpus;
    cudaGetDeviceCount(&ngpus);

    for (int i = 0; i < ngpus; i++) {
        cudaDeviceProp devProp;

        cudaGetDeviceProperties(&devProp, i);
        printf("Device[%d](%s) compute capability : %d.%d.\n"
               , i, devProp.name, devProp.major, devProp.minor);
    }
}
```

```
Device[0](GeForce RTX 2080 Ti) compute capability : 7.5.
Device[1](GeForce GTX 1080) compute capability : 6.1.
```

Selecting the target GPU

- **Set the target device**
  - cudaError_t **cudaSetDevice** (int deviceID)
  - After calling this function, all operations are sent to the device
- **Get the current device ID**
  - cudaError_t **cudaGetDevice** (int *deviceID)

- **We can change the target device any time**
- **However, we should carefully changing the target device**
  - Which device having the target data?



Lecture 12-2

Heterogeneous Parallel Computing

Fully utilize your computing resources

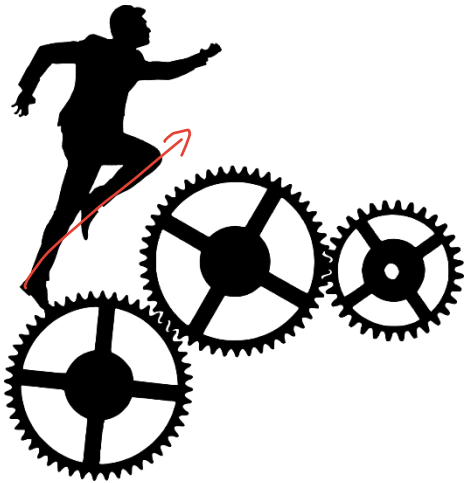
Heterogeneous Architecture

- **A heterogeneous architecture consisting of more than one type of computing resources**
- **Examples**
  - A desktop PC having both multi-core CPUs and GPUs
  - A multi-GPU system consisting of different types of GPUs

Heterogeneous Computing

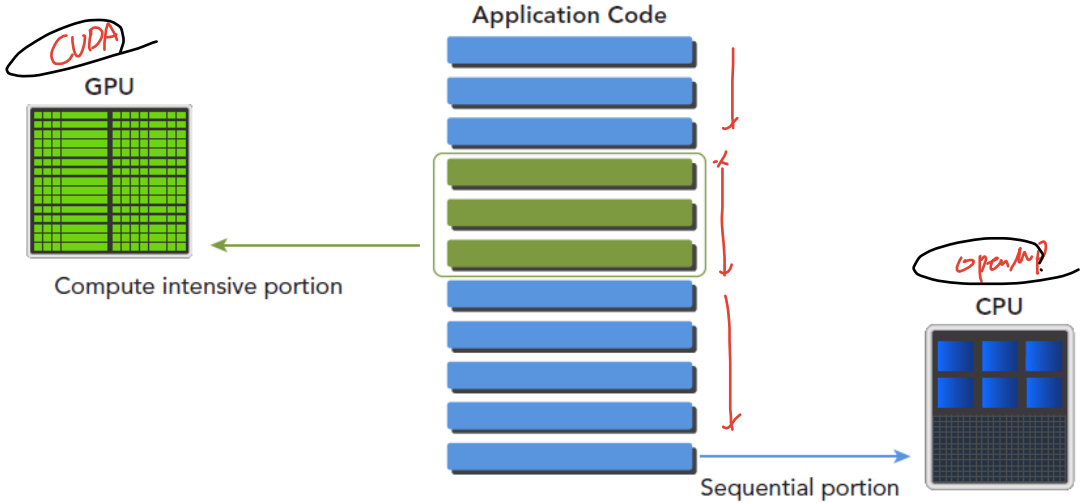
- **Use multiple heterogeneous computing resources at once for solving a problem**
  - ←→ Homogeneous computing

- **Advantage**
  - Fully utilize all available computing resources
  - Achieve high performance



[Figures from Pixabay]

Heterogeneous Computing



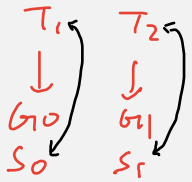
[Figures from Professional CUDA C Programming, Wrox]

Host-Device Concurrent Execution

[Image from PublicDomainPixtures, Pixabay]

Example – OpenMP + CUDA

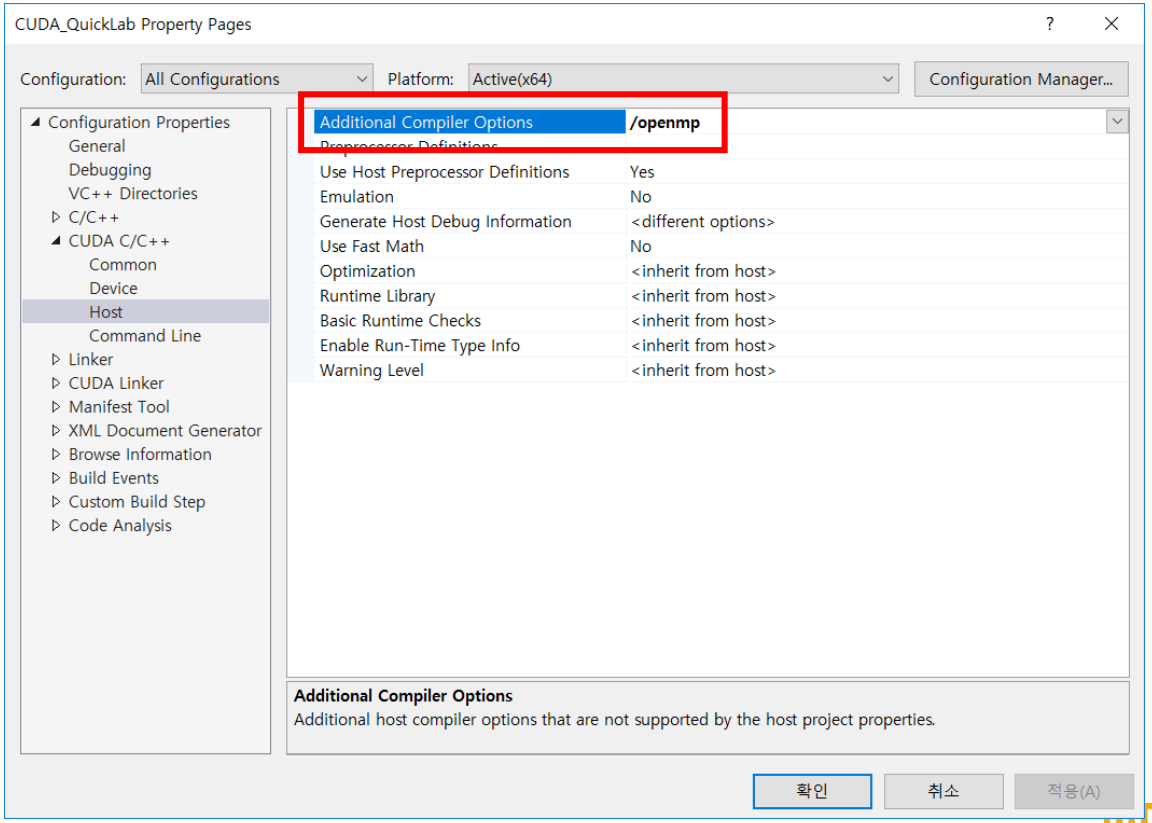
```
#pragma omp parallel for
LOOP_I(NUM_STREAMS)
{
    // Do work of device
    cudaSetDevice(device[i]);
    myKernel <<<NUM_BLOCK / NUM_STREAMS, NUM_T_IN_B, 0, stream[i]>>> (...);

    // Do work of host
    for(…) {
    }

    // Synchronization
    cudaStreamSynchronize(stream[i]);
}
```



Using Both OpenMP and CUDA

- **NVCC compiler option**
  - Using OpenMP on CUDA code (.cu)

↳ compile option Add



Using Both OpenMP and CUDA

- **Decouple source file for host code and device code**
  - See the CUDA_MatrixAdd Project on the course git repository [link]