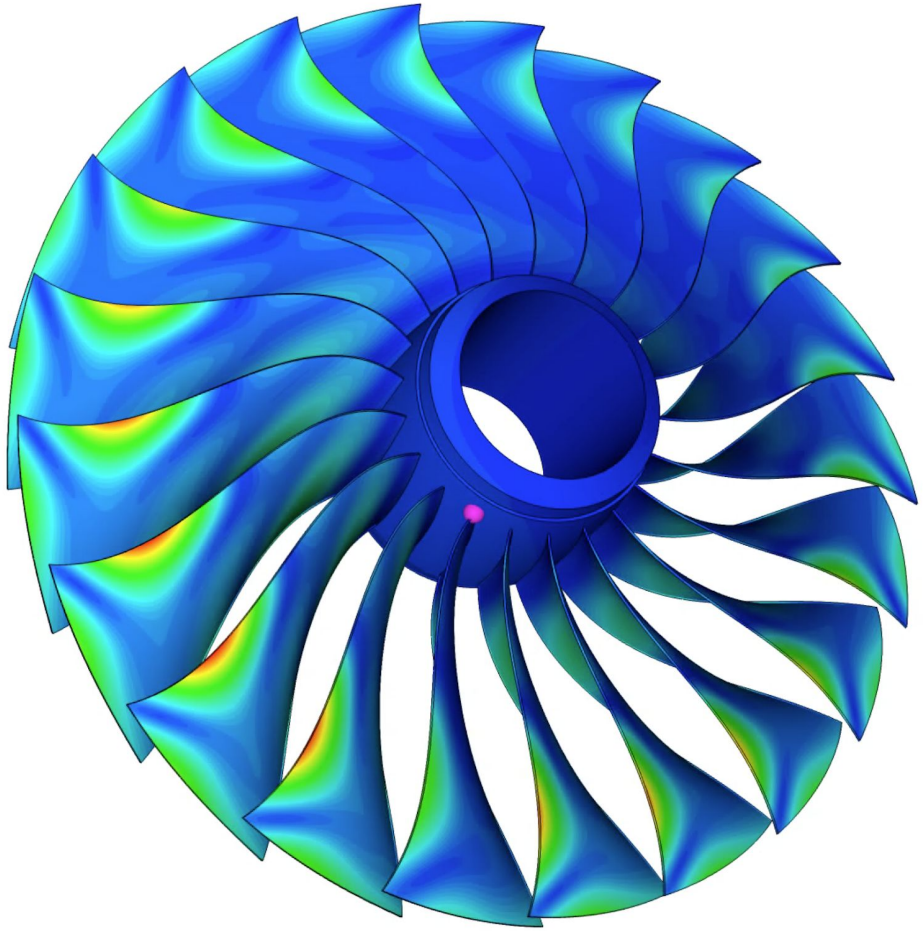
GPU accelerated computing for Finite Element Method



Introduction to GPU

Heterogenous program

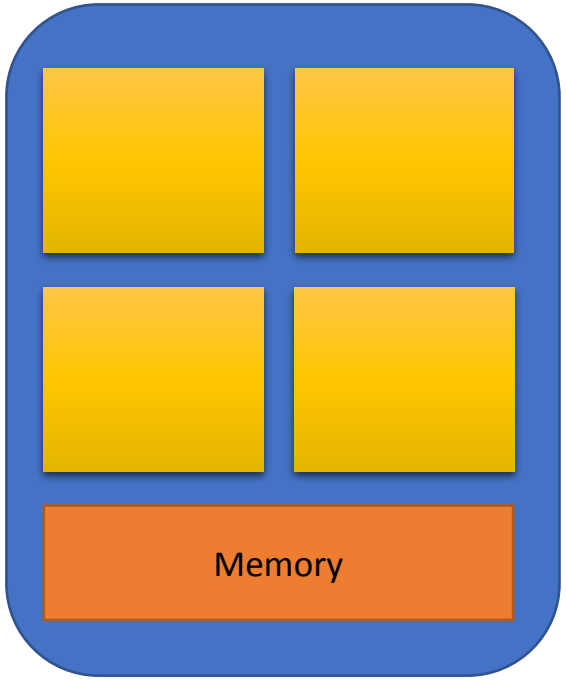
Objectives

- To learn the basics of CUDA code
  - Host and Device
  - Memory allocation
  - Data transfer

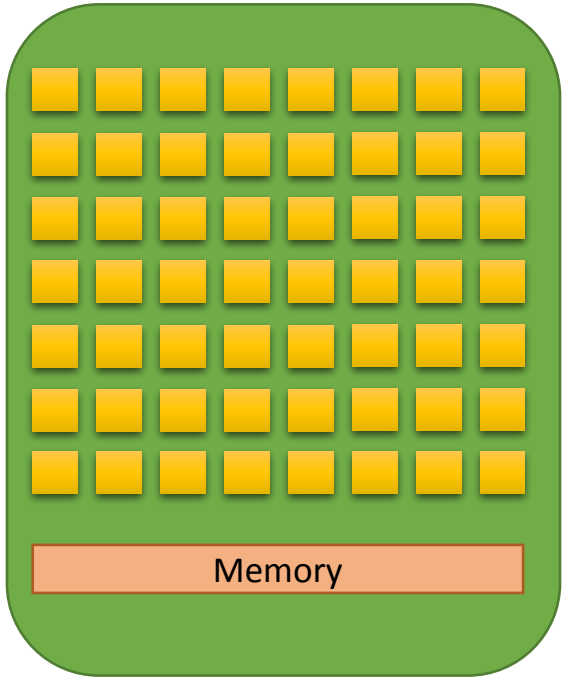
Introduction to GPU

Heterogenous program

CPU - Host



GPU – Device



Introduction to GPU

Heterogenous program

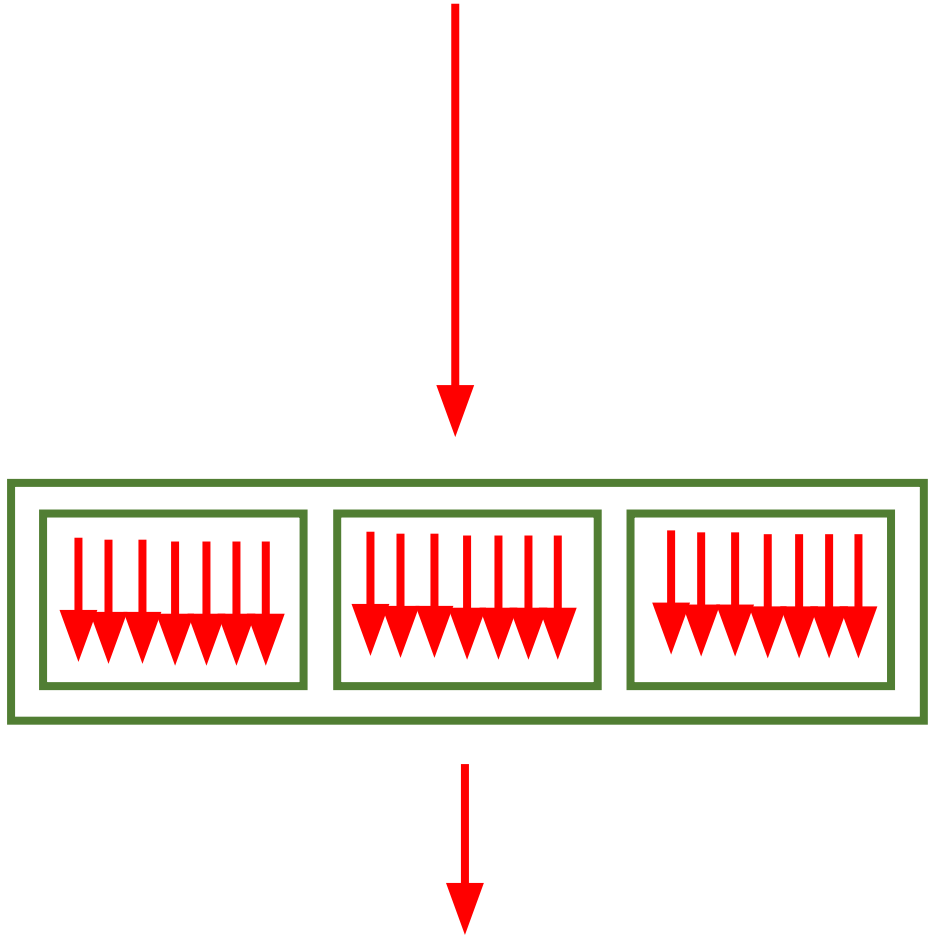
CUDA programming Model – Heterogenous Computing

```cpp
int main(){
    float *a, *b, *out;
    float *d_a, *d_b, *d_out;

    a   = (float*)malloc(sizeof(float) * N);
    b   = (float*)malloc(sizeof(float) * N);
    out = (float*)malloc(sizeof(float) * N);

    // Initialize array
    for(int i = 0; i < N; i++){
        a[i] = 1.0f;
        b[i] = 2.0f;
    }

    // Allocate device memore for a
    cudaMalloc((void**)&d_a,sizeof(float)*N);
    cudaMalloc((void**)&d_b,sizeof(float)*N);
    cudaMalloc((void**)&d_out,sizeof(float)*N);

    // Transfer data from host to device memory
    cudaMemcpy(d_a,a, sizeof(float)*N, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b,b, sizeof(float)*N, cudaMemcpyHostToDevice);

    // Main function
    int block_size = 256;
    int grid_size  = (N+block_size)/block_size;
    vector_add<<<grid_size,block_size>>>(d_out, d_a, d_b, N);

    cudaMemcpy(out, d_out, sizeof(float)*N, cudaMemcpyDeviceToHost);

    // Deallocate device memory
    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_out);

    // Deallocate host memory
    free(a);
    free(b);
    free(out);

}
```

Serial Code

Parallel Code

Serial Code



Serial Code in host

Parallel Code in device

Serial Code in host

Introduction to GPU

Heterogenous program

CUDA programming Model – Heterogenous Computing

Host – CPU, Device - GPU

```cpp
int main(){
    float *a, *b, *out;
    float *d_a, *d_b, *d_out;

    a   = (float*)malloc(sizeof(float) * N);
    b   = (float*)malloc(sizeof(float) * N);
    out = (float*)malloc(sizeof(float) * N);

    // Initialize array
    for(int i = 0; i < N; i++){
        a[i] = 1.0f;
        b[i] = 2.0f;
    }

    // Allocate device memore for a
    cudaMalloc((void**)&d_a,sizeof(float)*N);
    cudaMalloc((void**)&d_b,sizeof(float)*N);
    cudaMalloc((void**)&d_out,sizeof(float)*N);

    // Transfer data from host to device memory
    cudaMemcpy(d_a,a, sizeof(float)*N, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b,b, sizeof(float)*N, cudaMemcpyHostToDevice);

    // Main function
    int block_size = 256;
    int grid_size  = (N+block_size)/block_size;
    vector_add<<<grid_size,block_size>>>(d_out, d_a, d_b, N);

    cudaMemcpy(out, d_out, sizeof(float)*N, cudaMemcpyDeviceToHost);

    // Deallocate device memory
    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_out);

    // Deallocate host memory
    free(a);
    free(b);
    free(out);

}
```