Open in Colab

Vector Add

In this example, a step by step vector addition on GPU will be shown. This kind of operation is known as SAXPY (Single-precision A*X Plus Y).

In [1]:
```
%%file vector_add.cu
#include <stdio.h>
#include <assert.h>


//cudaMemcpy (void *dst, const void *src, size t count, enum cudaMemcpyKind kind)
#define MAX_THREADS_IN_BLOCK 1024

#define MAX_ERR 1e-6

using namespace std;

void cpu_vector_add(float *h_out, float *h_a, float *h_b, int n) {
    int tid = 0; // this is CPU zero, so we start at zero
    while (tid < n)
    {
        h_out[tid] = h_a[tid] + h_b[tid];
        tid += 1;  // we have one CPU, so we increment by one
    }

    // same, using the for loop
    // for(int i = 0; i < n; i++){
    //     h_out[i] = h_a[i] + h_b[i];
    // }
}

__global__ void gpu_vector_add(float *out, float *a, float *b, int n) {
    // built-in variable blockDim.x describes amount threads per block
    int tid = blockIdx.x * blockDim.x + threadIdx.x;

    // check if still inside array
    if (tid < n)
        out[tid] = a[tid] + b[tid];

    // more advanced version - handling arbitrary vector/kernel size
    // Consider case when gridDim*blockDim < vector size
    // int step = gridDim.x * blockDim.x;
    // while (tid < n)
    // {
    //     out[tid] = a[tid] + b[tid];
    //     tid += step;
    // }

    // same, using the for loop
    // for(; tid < n; tid += step){
    //     out[tid] = a[tid] + b[tid];
    // }
}

void CPU_version_wrapper(const int N)
{
    float *h_a, *h_b, *h_out;

    // Allocate host memory (RAM for CPU)
    h_a   = (float*)malloc(sizeof(float) * N);
    h_b   = (float*)malloc(sizeof(float) * N);
    h_out = (float*)malloc(sizeof(float) * N);

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

    // Main function
    cpu_vector_add(h_out, h_a, h_b, N);

    for(int i = 0; i < N; i++){
        assert(fabs(h_out[i] - h_a[i] - h_b[i]) < MAX_ERR);
    }
    printf("CPU assertion PASSED\n");
    printf("CPU Last element in the array: out[%d] = %.2f\n\n",N-1,  h_out[N-1]);

    // Cleanup host memory
    free(h_a); free(h_b); free(h_out);
}

void GPU_version_wrapper(const int N)
{

    // Allocate CPU memory
    float *h_a, *h_b, *h_out;
    h_a   = (float*)malloc(sizeof(float) * N);
    h_b   = (float*)malloc(sizeof(float) * N);
    h_out = (float*)malloc(sizeof(float) * N);

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

    // Allocate device memory for d_a
    float *d_a, *d_b, *d_out;
    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  (global) memory
    cudaMemcpy(d_a, h_a, sizeof(float) * N, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, h_b, sizeof(float) * N, cudaMemcpyHostToDevice);

    // Main function: Call the kernel
    gpu_vector_add<<<1,MAX_THREADS_IN_BLOCK>>>(d_out, d_a, d_b, N);//  <<<blocks, threads_per_block

    // implement a kernel for which gridDim*blockDim < vector size
    // gpu_vector_add<<<2,64>>>(d_out, d_a, d_b, N);//  <<<blocks, threads_per_block>>>

    // if N is a friendly multiplier of THREADS_PER_BLOCK
    // gpu_vector_add<<<N/MAX_THREADS_IN_BLOCK,MAX_THREADS_IN_BLOCK>>>(d_out, d_a, d_b, N);

    // if N is not a friendly multiplier of THREADS_PER_BLOCK
    // gpu_vector_add<<<(N + MAX_THREADS_IN_BLOCK-1) / MAX_THREADS_IN_BLOCK, MAX_THREADS_IN_BLOCK>>

    // Transfer data from device (global) memory to host
    cudaMemcpy(h_out, d_out, sizeof(float) * N, cudaMemcpyDeviceToHost);
    // cudaMemcpy() Blocks the CPU until the copy is complete
    // Copy begins when all preceding CUDA calls have completed

    // Verification
    printf("GPU Last element in the array: out[%d] = %.2f\n",N-1,  h_out[N-1]);
    for(int i = 0; i < N; i++){
        assert(fabs(h_out[i] - h_a[i] - h_b[i]) < MAX_ERR);
    }

    printf("GPU assertion PASSED\n\n");

    // Cleanup memory after kernel execution
    cudaFree(d_a);cudaFree(d_b);cudaFree(d_out);
    free(h_a);free(h_b);free(h_out);
}



int main(){
    const int N = 1024;
    CPU_version_wrapper(N);
    GPU_version_wrapper(N);

    return 0;
}
```

```
Writing vector_add.cu
```

In [2]:
```
!echo "Check your GPU version"
!nvidia-smi
```

```
Check your GPU version
Sun Apr 16 17:39:57 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.108.03   Driver Version: 510.108.03   CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:01:00.0  On |                  N/A |
|  0%   58C    P5    31W / 250W |   1441MiB /  8192MiB |      4%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1703      G   /usr/lib/xorg/Xorg                 96MiB |
|    0   N/A  N/A      2624      G   /usr/lib/xorg/Xorg                684MiB |
|    0   N/A  N/A      2819      G   /usr/bin/gnome-shell               66MiB |
|    0   N/A  N/A      3712      G   ...RendererForSitePerProcess        3MiB |
|    0   N/A  N/A      8175      G   ...features=BackForwardCache       10MiB |
|    0   N/A  N/A      8543      G   ...957248867340528764,131072      205MiB |
|    0   N/A  N/A     13016      G   ...AAAAAAAAA= --shared-files       35MiB |
|    0   N/A  N/A     14339      G   ...b/thunderbird/thunderbird      121MiB |
|    0   N/A  N/A     15643      G   ...RendererForSitePerProcess      169MiB |
+-----------------------------------------------------------------------------+
```

In [3]:
```
%%bash

CUDA_SUFF=70
nvcc -gencode arch=compute_${CUDA_SUFF},code=sm_${CUDA_SUFF} ./vector_add.cu -o vector_add
./vector_add
```

```
CPU assertion PASSED
CPU Last element in the array: out[1023] = 3.00

GPU Last element in the array: out[1023] = 3.00
GPU assertion PASSED
```

In [4]:
```
!ls
```

```
10_intro_setup.ipynb                    a.cpp          pdfy
20_vector_add.ipynb                     a.out          README.md
30_element_wise_matrix_add.ipynb        experimental   requirements.txt
30_matrix_matrix_multiplication.ipynb   gpu_colab      solutions
40_parallel_reduction.ipynb             hello          src
50_thrust.ipynb                         hello.cpp      to_pdf_1by1.sh
60_python_cuda_intro.ipynb              hello_cuda     vector_add
70_heat_diffusion2D.ipynb               hello_cuda.cu  vector_add.cu
```