hpc-practical-vector-addition

April 9, 2024

[1]:
```
!python --version
!nvcc --version
!pip install nvcc4jupyter
%load_ext nvcc4jupyter
```

```
Python 3.10.12
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Tue_Aug_15_22:02:13_PDT_2023
Cuda compilation tools, release 12.2, V12.2.140
Build cuda_12.2.r12.2/compiler.33191640_0
Collecting nvcc4jupyter
  Downloading nvcc4jupyter-1.2.1-py3-none-any.whl (10 kB)
Installing collected packages: nvcc4jupyter
Successfully installed nvcc4jupyter-1.2.1
Detected platform "Colab". Running its setup…
Source files will be saved in "/tmp/tmpxzovgo81".
```

[14]:
```
%%cuda
#include <iostream>
#include <cuda_runtime.h>

__global__ void addVectors(int* A, int* B, int* C, int n)
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < n)
    {
        C[i] = A[i] + B[i];
    }
}

int main()
{
    std::cout<<"This is Atharva Pingale's code";
    std::cout<<"\nPractical 4 : Vector Addition\n";
    int n = 1000000;
    int* A, * B, * C;
    int size = n * sizeof(int);
```

```
// Allocate memory on the host
cudaMallocHost(&A, size);
cudaMallocHost(&B, size);
cudaMallocHost(&C, size);

// Initialize the vectors
for (int i = 0; i < n; i++)
{
    A[i] = i * 2 * 10;
    B[i] = i * 3 * 12;
}


// Printing the vectors
std::cout<<"\nPrinting Vector A ( first 20 elements ): ";
std::cout<<"\n";
for(int i=0;i<20;i++){
    std::cout<<A[i]<<" ";
}
std::cout<<"\n";

std::cout<<"\nPrinting Vector B ( first 20 elements ) : ";
std::cout<<"\n";
for(int i=0;i<20;i++){
    std::cout<<B[i]<<" ";
}
std::cout<<"\n";


// Allocate memory on the device
int* dev_A, * dev_B, * dev_C;
cudaMalloc(&dev_A, size);
cudaMalloc(&dev_B, size);
cudaMalloc(&dev_C, size);

// Copy data from host to device
cudaMemcpy(dev_A, A, size, cudaMemcpyHostToDevice);
cudaMemcpy(dev_B, B, size, cudaMemcpyHostToDevice);

// Launch the kernel
int blockSize = 256;
int numBlocks = (n + blockSize - 1) / blockSize;
addVectors<<<numBlocks, blockSize>>>(dev_A, dev_B, dev_C, n);

// Copy data from device to host
cudaMemcpy(C, dev_C, size, cudaMemcpyDeviceToHost);
```

```
    // Print the results
    std::cout<<"\nPrinting Result Vector C ( first 20 elements ): ";
    std::cout<<"\n";
    for (int i = 0; i < 20; i++)
    {
        std::cout << C[i] << " ";
    }
    std::cout<<"\n";

    // Free memory
    cudaFree(dev_A);
    cudaFree(dev_B);
    cudaFree(dev_C);
    cudaFreeHost(A);
    cudaFreeHost(B);
    cudaFreeHost(C);

    return 0;
}
```

```
This is Atharva Pingale's code
Practical 4 : Vector Addition

Printing Vector A ( first 20 elements ):
0 20 40 60 80 100 120 140 160 180 200 220 240 260 280 300 320 340 360 380

Printing Vector B ( first 20 elements ) :
0 36 72 108 144 180 216 252 288 324 360 396 432 468 504 540 576 612 648 684

Printing Result Vector C ( first 20 elements ):
0 56 112 168 224 280 336 392 448 504 560 616 672 728 784 840 896 952 1008 1064
```

[ ]: