addition

May 8, 2025

[ ]:
```
!pip install git+https://github.com/afnan47/cuda.git
```

```
Collecting git+https://github.com/afnan47/cuda.git
  Cloning https://github.com/afnan47/cuda.git to /tmp/pip-req-build-h0vksgpm
  Running command git clone --filter=blob:none --quiet
https://github.com/afnan47/cuda.git /tmp/pip-req-build-h0vksgpm
  Resolved https://github.com/afnan47/cuda.git to commit
aac710a35f52bb78ab34d2e52517237941399eff
  Preparing metadata (setup.py) … done
Building wheels for collected packages: NVCCPlugin
  Building wheel for NVCCPlugin (setup.py) … done
  Created wheel for NVCCPlugin: filename=NVCCPlugin-0.0.2-py3-none-any.whl
size=4290
sha256=2c7999dd3ee29d11e5210f2727ab451b0d63256b3ee4f1f07e011bf70402cc23
  Stored in directory: /tmp/pip-ephem-wheel-cache-r0lj2seq/wheels/bc/4e/e0/2d86b
d15f671dbeb32144013f1159dba09757fde36dc51a963
Successfully built NVCCPlugin
Installing collected packages: NVCCPlugin
Successfully installed NVCCPlugin-0.0.2
```

[ ]:
```
%load_ext nvcc_plugin
```

```
created output directory at /content/src
Out bin /content/result.out
```

[ ]:
```
%%writefile vector_add.cu
#include <iostream>
using namespace std;
__global__
void add(int* A, int* B, int* C, int size) {
int tid = blockIdx.x * blockDim.x + threadIdx.x;
if (tid < size) {
C[tid] = A[tid] + B[tid];
}
}
void initialize(int* vector, int size) {
for (int i = 0; i < size; i++) {
vector[i] = rand() % 10;
}
}
void print(int* vector, int size) {
for (int i = 0; i < size; i++) {
cout << vector[i] << " ";
}
cout << endl;
}
int main() {
int N = 4;
int* A = new int[N];
int* B = new int[N];
int* C = new int[N];
initialize(A, N);
initialize(B, N);
cout << "Vector A: ";
print(A, N);
cout << "Vector B: ";
print(B, N);
int *d_A, *d_B, *d_C;
size_t bytes = N * sizeof(int);
cudaMalloc(&d_A, bytes);
cudaMalloc(&d_B, bytes);
cudaMalloc(&d_C, bytes);
cudaMemcpy(d_A, A, bytes, cudaMemcpyHostToDevice);
cudaMemcpy(d_B, B, bytes, cudaMemcpyHostToDevice);
int threadsPerBlock = 256;
int blocksPerGrid = (N + threadsPerBlock - 1) / threadsPerBlock;
add<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
cudaDeviceSynchronize();
cudaMemcpy(C, d_C, bytes, cudaMemcpyDeviceToHost);
cudaError_t err = cudaGetLastError();
if (err != cudaSuccess) {
cout << "CUDA Error: " << cudaGetErrorString(err) << endl;
return 1;
}
cout << "Addition: ";
print(C, N);
delete[] A;
delete[] B;
delete[] C;
cudaFree(d_A);
cudaFree(d_B);
cudaFree(d_C);
return 0;
}
```

```
Overwriting vector_add.cu
```

[ ]:
```
!nvcc -arch=sm_75 vector_add.cu -o vector_add
```

[ ]:
```
!./vector_add
```

```
Vector A: 3 6 7 5
Vector B: 3 5 6 2
Addition: 6 11 13 7
```

[ ]:
```
```