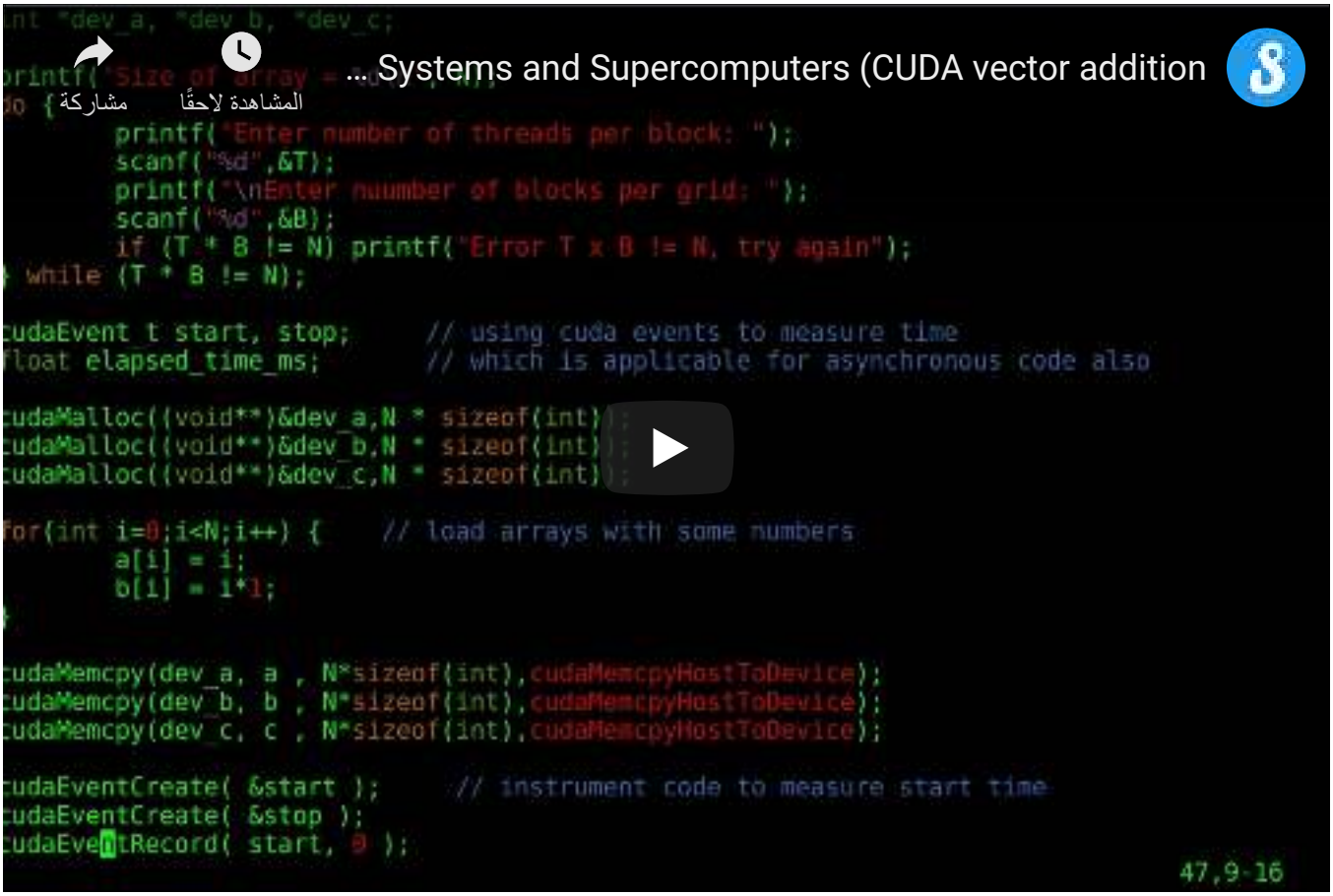
CUDA - vector additon demo

Let’s watch a demo on how to add two vectors in cuda



Code for the vector addition is given below (collected from source)

```
/**
 * Vector addition: w = u + v.
 *
 * This sample is a very basic sample that implements element by element
 * vector addition. It is the same as the sample illustrating Chapter 2
 * of the programming guide with some additions like error checking.
 */

#include <stdio.h>

// For the CUDA runtime routines (prefixed with "cuda_")
#include <cuda_runtime.h>

/**
 * CUDA Kernel Device code
```

```
 *
 * Computes the vector addition of u and v into w. The 3 vectors have the same
 * number of elements numElements.

 */
__global__ void
vectorAdd(const float *u, const float *v, float *w)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;

    w[i] = u[i] + v[i];
}

/**
 * Host main routine
 */
int
main(void)
{
    // Print the vector length to be used, and compute its size
    int numElements = 50000;
    size_t size = numElements * sizeof(float);
        // Observe that this program is ever so slightly busted,
        // for reasons that will become apparent later.
    printf("[Vector addition of %d elements]\n", numElements);

    // Allocate the host input vector u
    float *h_u = (float *)malloc(size);

    // Allocate the host input vector v
    float *h_v = (float *)malloc(size);

    // Allocate the host output vector w
    float *h_w = (float *)malloc(size);

    // Initialize the host input vectors
    for (int i = 0; i < numElements; ++i)
    {
        h_u[i] = rand()/(float)RAND_MAX;
        h_v[i] = rand()/(float)RAND_MAX;
    }

    // Allocate the device input vector u
    float *d_u = NULL;
    cudaMalloc((void **)&d_u, size);

    // Allocate the device input vector v
    float *d_v = NULL;
    cudaMalloc((void **)&d_v, size);

    // Allocate the device output vector w
    float *d_w = NULL;
    cudaMalloc((void **)&d_w, size);

    // Copy the host input vectors u and v in host memory to the
        // device input vectors in device memory
    printf("Copy input data from the host memory to the CUDA device\n");
    cudaMemcpy(d_u, h_u, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_v, h_v, size, cudaMemcpyHostToDevice);

    // Launch the Vector Add CUDA Kernel
    int threadsPerBlock = 256;
    int blocksPerGrid =(numElements + threadsPerBlock - 1) / threadsPerBlock;
```

```
    printf("CUDA kernel launch with %d blocks of %d threads\n", blocksPerGrid, threadsPerBloc
    vectorAdd<<<blocksPerGrid, threadsPerBlock>>>(d_u, d_v, d_w);

    // Copy the device result vector in device memory to the host result vector
    // in host memory.
    printf("Copy output data from the CUDA device to the host memory\n");
    cudaMemcpy(h_w, d_w, size, cudaMemcpyDeviceToHost);

    // Verify that the result vector is correct
    for (int i = 0; i < numElements; ++i)
    {
        if (fabs(h_u[i] + h_v[i] - h_w[i]) > 1e-5)
        {
            fprintf(stderr, "Result verification failed at element %d!\n", i);
            exit(EXIT_FAILURE);
        }
    }
    printf("Test PASSED\n");

    // Free device global memory
    cudaFree(d_u);
    cudaFree(d_v);
    cudaFree(d_w);

    // Free host memory
    free(h_u);
    free(h_v);
    free(h_w);

    // Reset the device and exit
    cudaDeviceReset();

    printf("Done\n");
    return 0;
}
```

While the code was running, I ran the `nvidia-smi` command that will show the GPU usage stat (watch the GPU#3, being used at 3%):

```
nvidia-smi
Tue Jun  6 12:09:28 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.20                 Driver Version: 375.20                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-SXM2...  Off  | 0000:04:00.0     Off |                    0 |
| N/A   37C    P0    41W / 300W |      0MiB / 16308MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla P100-SXM2...  Off  | 0000:06:00.0     Off |                    0 |
| N/A   37C    P0    43W / 300W |      0MiB / 16308MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla P100-SXM2...  Off  | 0000:07:00.0     Off |                    0 |
| N/A   36C    P0    40W / 300W |      0MiB / 16308MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla P100-SXM2...  Off  | 0000:08:00.0     Off |                    0 |
| N/A   35C    P0    34W / 300W |      0MiB / 16308MiB |      3%      Default |
```

```
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```