CGT620

Assign	
Status	

Lab3

Device Info

```
(dev)  zhiquan@zhiquan-ubuntu  ~/Git-Repositories/CGT620-Graphics-Processing-Unit-Computing/lab2  master ●  ./query

./query Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce 940MX"
  CUDA Driver Version / Runtime Version          10.2 / 10.1
  CUDA Capability Major/Minor version number:    5.0
  Total amount of global memory:                 984 MBytes (1031471104 bytes)
  ( 3) Multiprocessors, (128) CUDA Cores/MP:     384 CUDA Cores
  GPU Max Clock rate:                            993 MHz (0.99 GHz)
  Memory Clock rate:                             2505 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 1048576 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048 layers
  Total amount of constant memory:               65536 bytes
```

Kernel Size

- Block Dim : (32,1)
- Block Size : (128,1)

Vector Size

From 2^5 to 2^{20}

Key Implementation

Add Vector

```
// Kernel function to add the elements of two arrays
__global__ void add(int n, float *x, float *y) {
  uint index = blockIdx.x * blockDim.x + threadIdx.x;
  uint stride = blockDim.x * gridDim.x;
  for (int i = index; i < n; i += stride)
    y[i] = x[i] + y[i];
}
```

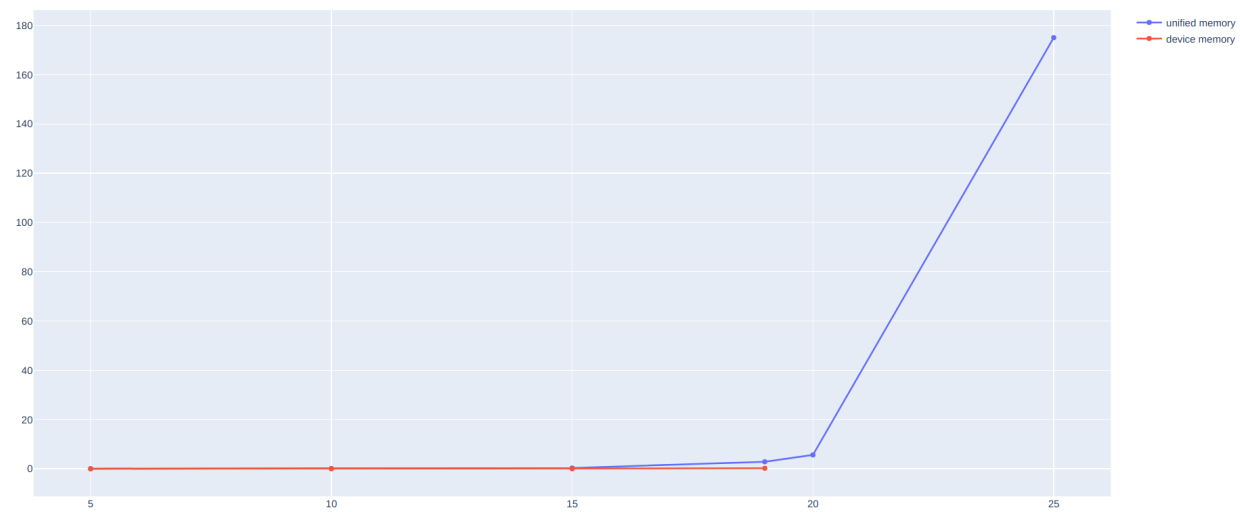
Unified Version

```
  int N = pow(2, 20);
  float *x, *y;
  // Allocate Unified Memory – accessible from CPU or GPU
  cudaMallocManaged(&x, N * sizeof(float));
  cudaMallocManaged(&y, N * sizeof(float));
  // initialize x and y arrays on the host
  for (int i = 0; i < N; i++) {
    x[i] = 1.0f;
    y[i] = 2.0f;
  }
```

Device Memory Version

```
  float *d_x;
  float *d_y;
  cudaError_t cudaStatus;
  // Allocate GPU buffers for three vectors (two input, one output)    .
  cudaStatus = cudaMalloc((void **)&d_x, N * sizeof(int));
  if (cudaStatus != cudaSuccess) {
    fprintf(stderr, "cudaMalloc failed!");
  }
  cudaStatus = cudaMemcpy(d_x, x, N * sizeof(int), cudaMemcpyHostToDevice);
  if (cudaStatus != cudaSuccess) {
    fprintf(stderr, "cudaMemcpy failed!");
  }
```

Performance (in ms)



Discussion

- Device memory version will be faster with the size of data increases
- Unified memory does not keep copies for both of the device and the host, as the device version program can not allocate enough space for the host array