LESSON 2

Measuring the Execution Time

TOPICS

- How to measure the runtimes of your routines in the GPU.

- ***Key Words***: *cudaEvent_t, cudaEventCreate(), cudaEventRecord(), cudaEventSynchronize(), cudaEventElapsedTime()*.

Events

- The way to calculate the execution time on the GPU is asking him to stamp a point —event— in the time while our kernel is executed; to do this, we will use the CUDA event API.
- It's important to take into account that the time measured is taken based on the GPU clock.
- To measure the time of our GPU application, we need to create two events, one at the start and one at the end.
- An event has completed when all tasks – or optionally, all commands in a given stream – preceding the event have completed.

- The steps to create the events are:

- [1] Create the events. To do this we are going to use the cudaEvent_t, a data type used by CUDA Runtime. Then, the cudaEventCreate() function is used to create an event.

```
cudaError_t  cudaEventCreate(cudaEvent_t * event )

where:
event - Newly created event
```

```
cudaEvent_t start, stop;
cudaEventCreate(&start);
cudaEventCreate(&stop);
```

- [2] Record the initial event.

```
cudaError_t  cudaEventRecord(cudaEvent_t event, cudaStream_t stream = 0)

where:
event     - Event to record
stream    - Stream in which to record event
```

```
cudaEventRecord(start, 0);
.....
// work on the GPU
.....
```

- [3] Record the final event.

```
cudaEventRecord(stop, 0);
```

- [4] Wait until the all the instructions on the GPU have reached the stop event. It is important because we want to be sure that our kernel finished its duty before calculating the time.

```
cudaEventSynchronize(stop);
```

- [5] Now, we need to calculate the elapsed time between our events. The function used to do this is *cudaEventElapsedTime()*. It calculates the time in milliseconds.

```
cudaError_t  cudaEventElapsedTime(float * ms, cudaEvent_t   start,
cudaEvent_t  end)
Where:
ms          - Time between start and end in ms
start       - Starting event
end         - Ending event
```

```
float time;
cudaEventElapsedTime(&time, start, stop);
```

- [6] Finally, we need to destroy the events that we have created.

```
cudaEventDestroy(start);
cudaEventDestroy(stop);
```

- It is important to notice that the events are **unsuitable to calculate more than kernel executions and memory copies in the device.**

Example

- 01matrix_transpose_events: This example calculates the execution time used to calculate the transposed matrix.

Practice

- 01matrix_multiplication: This code must execute the multiplication between two matrices and implement a header called GpuTimer.h that contains the functions to implement events in 01matrix_multiplication.cu, and Matrix.h that contains the structure of a matrix.