**NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**IT301 : Parallel Computing Lab**

**PC Lab 9 [Total marks = 10]**

**Date 26 October 2021**

**CUDA Programs in Google Colab**

Goto -> https://colab.research.google.com/notebooks/intro.ipynb

Open New Notebook.

Change Run time type to GPU

Setup the environment for running CUDA program as given in following link.

<https://www.geeksforgeeks.org/how-to-run-cuda-c-c-on-jupyter-notebook-in-google-colaboratory/>

After setting up the notebook for running CUDA execute the following programs.

**Program 1: To know details of the device. Run the program and Explain the result.**

**[2 marks]**

%%cu

#include<stdio.h>

int main()

{

int devcount;

cudaGetDeviceCount(&devcount);

printf("Device count:%d\n",devcount);

for (int i = 0; i < devcount; ++i)

{

// Get device properties

printf("\nCUDA Device #%d\n", i);

cudaDeviceProp devProp;

cudaGetDeviceProperties(&devProp, i);

printf("Name:%s\n", devProp.name);

printf("Compute capability: %d.%d\n",devProp.major ,devProp.minor);

printf("Warp Size %d\n",devProp.warpSize);

printf("Total global memory:%u bytes\n",devProp.totalGlobalMem);

printf("Total shared memory per block: %u bytes\n", devProp.sharedMemPerBlock);

printf("Total registers per block : %d\n",devProp.regsPerBlock);

printf("Clock rate: %d khz\n",devProp.clockRate);

printf("Maximum threads per block:%d\n", devProp.maxThreadsPerBlock);

for (int i = 0; i < 3; ++i)

printf("Maximum dimension %d of block: %d\n", i, devProp.maxThreadsDim[i]);

for (int i = 0; i <= 2; ++i)

printf("Maximum dimension %d of grid: %d\n", i, devProp.maxGridSize[i]);

printf("Number of multiprocessors:%d\n", devProp.multiProcessorCount);

}

return 0;

}

**Program 2: Hello world program. Record the result and write the observation.**

**[2 marks]**

%%cu

#include<stdio.h>

#include<cuda.h>

\_\_global\_\_ void helloworld(void)

{

printf("Hello World from GPU\n");

}

int main() {

helloworld<<<1,10>>>();

printf("Hello World\n");

return 0;

}

-------------------------------------------------------------------------------------------------------

**Program 3: Progrma to perfrom c[i] =a[i]+b[i]; Here, c[i] is calcualted for all i. But results are displayed only for few c[i]. Explain your observation. [2 x 3 = 6 Marks]**

Run the program for following and note down the time.

a) vecAdd**<<<1,100>>>**(d\_a, d\_b, d\_c, n);

b)vecAdd**<<<1,50>>>**(d\_a, d\_b, d\_c, n);

c)vecAdd**<<<2,50>>>(**d\_a, d\_b, d\_c, n);

%%cu

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <sys/time.h>

\_\_global\_\_ void vecAdd(double \*a, double \*b, double \*c, int n)

{

// Get global thread

int id = blockIdx.x\*blockDim.x+threadIdx.x;

// Do not go out of bounds

if (id < n)

c[id] = a[id] + b[id];

}

int main( int argc, char\* argv[] )

{

// Size of vectors

int n = 100;

//time varibales

struct timeval t1, t2;

// Host input vectors

double \*h\_a, \*h\_b;

//Host output vector

double \*h\_c;

// Device input vectors

double \*d\_a, \*d\_b;

//Device output vector

double \*d\_c;

// Size, in bytes, of each vector

size\_t bytes = n\*sizeof(double);

// Allocate memory for each vector on host

h\_a = (double\*)malloc(bytes);

h\_b = (double\*)malloc(bytes);

h\_c = (double\*)malloc(bytes);

// Allocate memory for each vector on GPU

cudaMalloc(&d\_a, bytes);

cudaMalloc(&d\_b, bytes);

cudaMalloc(&d\_c, bytes);

int i;

// Initialize vectors on host

for( i = 0; i < n; i++ ) {

h\_a[i] = i+1;

h\_b[i] = i+1;

}

// Copy host vectors to device

cudaMemcpy( d\_a, h\_a, bytes, cudaMemcpyHostToDevice);

cudaMemcpy( d\_b, h\_b, bytes, cudaMemcpyHostToDevice);

gettimeofday(&t1, 0);

// Execute the kernel

vecAdd<<<1,100>>>(d\_a, d\_b, d\_c, n);

cudaDeviceSynchronize();

gettimeofday(&t2, 0);

// Copy array back to host

cudaMemcpy( h\_c, d\_c, bytes, cudaMemcpyDeviceToHost );

for(i=0; i<n; i=i+10)

printf("c[%d]=%f\n",i,h\_c[i]);

double time = (1000000.0\*(t2.tv\_sec-t1.tv\_sec) + t2.tv\_usec-t1.tv\_usec)/1000.0;

printf("Time to generate: %3.10f ms \n", time);

// Release device memory

cudaFree(d\_a);

cudaFree(d\_b);

cudaFree(d\_c);

// Release host memory

free(h\_a);

free(h\_b);

free(h\_c);

return 0;

}