CO 316 – COMPUTER ARCHITECTURE LAB
ASSIGNMENT 0 - REPORT
24-07-2018
Submitted by :
16CO212 – Bidyadhar Mohanty
16CO249 - Soham Patil

Q1. Write the device query code, compile and run it on your system. Query enough information to know all the details of your device. Example queries: GPU card's name, GPU computation capabilities, Maximum number of block dimensions, Maximum number of grid dimensions, Maximum size of GPU memory, Amount of constant and share memory, Warp size, etc. Answer the following questions in your report.

1. What is the architecture and compute capability of your GPU?
Answer 1) Architecture : Tesla K40m by NVIDIA based on the NVIDIA Kepler Architecture. NVIDIA ® Kepler TM architecture delivers 3X the performance of previous generations. Each thread block of a grid is split into warps, each gets executed by one multiprocessor (SM).A multiprocessor can execute multiple blocks concurrently.Shared memory and registers are partitioned among the threads of all concurrent blocks.So, decreasing shared memory usage (per block) and register usage (per thread) increases number of blocks that can run concurrently. Compute capability = major revision number . minor revision number = 3.5

2. What are the maximum block dimensions for your GPU?
Answer 2) Maximum dimension 0 of block: 1024
Maximum dimension 1 of block: 1024
Maximum dimension 2 of block: 64
Theoretically, 1024 threads per block can be used.

3. Suppose you are launching a one dimensional grid and block. If the hardware's maximum grid dimension is 65535 and the maximum block dimension is 512, what is the maximum number threads can be launched on the GPU?
Answer 3) 33553920 threads can be launched (65535 x 512)

4. Under what conditions might a programmer choose not want to launch the maximum number of threads?
Answer 4) A programmer will choose not to launch the max number of threads when the total number of threads is greater than the required number of parallel processes.

5. What can limit a program from launching the maximum number of threads on a GPU?
Answer 5) The factors involved in limiting the selection of maximum number of threads per GPU are :
● Threads per block should be a multiple of warp size to avoid wasting computation on under-populated warps and to facilitate coalescing.
● A minimum of 64 threads per block should be used, and only if there are multiple concurrent blocks per multiprocessor
● Between 128 and 256 threads per block is a better choice and a good initial range for experimentation with different block sizes.
● Use several (3 to 4) smaller thread blocks rather than one large thread block per multiprocessor if latency affects performance. This is particularly beneficial to kernels that frequently call __syncthreads().

6. What is shared memory? How much shared memory is on your GPU?
Ans 6) In CUDA, shared memory refers to the memory used for communication inside a single program, e.g. among its multiple threads, to provide communication among them or avoid redundant copies.Shared memory is allocated per thread block, so all threads in the block have access to the same shared memory. The visibility is limitedto only threads within the same block.
Total shared memory per block in GPU = 49152
Total shared memory = No. of blocks x shared memory per block
= (2147483647 x 65535 x 65535)x 49152

7. What is global memory? How much global memory is on your GPU?
Answer 7) In CUDA, global memory is the main allocated memory that lives until it is freed and used in order to store data on the gpu that can be communicated back to the host. Global memory is active until the application closes or is freed and it is visible to any thread and block that have a pointer to that memory region.
Total global memory = 3405643776

8. What is constant memory? How much constant memory is on your GPU?
Answer 8) In CUDA, constant memory is the memory used for data that will not change over the course of a kernel execution.The constant memory space is cached. A read from constant memory costs one memory read from device memory only on a cache miss; otherwise, it just costs one read from the constant cache.
Total constant memory = 65536

9. What does warp size signify on a GPU? What is your GPU's warp size?
Answer 9) Warp size refers to the number of threads in a warp, which is a subdivision of a block used in the hardware implementation to coalesce memory access and instruction dispatch.
GPU Warp Size = 32

10. Is double precision supported on your GPU?
Answer 10) Yes, double precision is supported on GPUs with compute capability above 1.3 and the compute capability of our GPU is 3.5.

Q2. Write a CUDA program to calculate the sum of the elements in an array. The array contains single precision floating point numbers. Generate your input array. For this question, do the following.

1. Allocate device memory
cudaMalloc((void **)&d_A, N * sizeof(float));
Allocates object in the device global memory 2 Parameters: Address of a pointer to the allocated object and size of allocated object in terms of bytes.

2. Copy host memory to device
cudaMemcpy(d_A, A, N * sizeof(float), cudaMemcpyHostToDevice);
memory data transfer 4 parameters: pointer to destination, pointer to source, bytes copied, type/direction of transfer.

3. Initialize thread block and kernel grid dimensions
<<<blocks, 1024>>>
where blocks = ceil(N/1024.00)

4. Invoke CUDA kernel
AddArray<<<blocks, 1024>>>(d_A, d_ans);

5. Copy results from device to host
cudaMemcpy(d_A, A, N * sizeof(float), cudaMemcpyHostToDevice);

6. Free device memory
cudaFree(d_A);

7. Write the CUDA kernel that computes the sum
```
__global__ void AddArray(float *A, float* ans)
{
unsigned int tid = threadIdx.x;
unsigned int i = blockDim.x * blockIdx.x + tid;
for(unsigned int s = blockDim.x / 2; s>0; s >>=1)
{
if(tid < s)
{
A[i] += A[i + s]
}
__syncthreads();
}
if(tid == 0)
{atomicAdd(ans, A[i]);
}
}
```
The code attached with this assignment generates an array of numbers from 1-1000 and returns the sum of all elements.