

Pratik Dhimal

2407779

1. In this programmer has launched a kernel with kernel `<<<1,1>>>` which means 1 block and 1 thread. Also the dimension will $(1,1,1)$ for both the grid and block. Hence, **the number of threads will be 1**. And the output will be the configuration as shown below.

```
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$ ls
01.cu          02-results.txt  04.cu          05-results.txt  07.cu          exp01
01-results.txt  03.cu          04-results.txt  06.cu          07-results.txt  exp02
02.cu          03-results.txt  05.cu          06-results.txt  a.out         exp03
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$ ./one

gd = gridDim
bd = blockDim
bi = blockIdx
ti = threadIdx

gd( 1, 1, 1) bd( 1, 1, 1) bi( 0, 0, 0) ti( 0, 0, 0) 0
Kernel launch successful.
```

2. Similar to 1, but here we have used kernel `<<<2,3>>>`. Hence the block and thread all be in x direction. **So there will be 6 threads, 3 threads for each 2 blocks.**

```
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$ ./two

gd = gridDim
bd = blockDim
bi = blockIdx
ti = threadIdx

gd( 2, 1, 1) bd( 3, 1, 1) bi( 0, 0, 0) ti( 0, 0, 0) 0
gd( 2, 1, 1) bd( 3, 1, 1) bi( 0, 0, 0) ti( 1, 0, 0) 1
gd( 2, 1, 1) bd( 3, 1, 1) bi( 0, 0, 0) ti( 2, 0, 0) 2
gd( 2, 1, 1) bd( 3, 1, 1) bi( 1, 0, 0) ti( 0, 0, 0) 3
gd( 2, 1, 1) bd( 3, 1, 1) bi( 1, 0, 0) ti( 1, 0, 0) 4
gd( 2, 1, 1) bd( 3, 1, 1) bi( 1, 0, 0) ti( 2, 0, 0) 5
Kernel launch successful.
```

3. Here we have used dim3 bd(2,3) for block dim, and the grid dimension is default as (1,1,1). So we launch the kernel as kernel<<<1,bd>>>. Hence, we have **1 block in x direction** and because of the dimension of the block **we'll have 2 threads in x direction and 3 threads in y direction**. Hence totaling 6 threads in one block and as we have one block **we'll have 6 threads in total.**

```
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$ nvcc 03.cu -o three
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$ ./three

gd = blockDim
bd = blockDim
bi = blockIdx
ti = threadIdx

gd( 1, 1, 1) bd( 2, 3, 1) bi( 0, 0, 0) ti( 0, 0, 0) 0
gd( 1, 1, 1) bd( 2, 3, 1) bi( 0, 0, 0) ti( 1, 0, 0) 1
gd( 1, 1, 1) bd( 2, 3, 1) bi( 0, 0, 0) ti( 0, 1, 0) 2
gd( 1, 1, 1) bd( 2, 3, 1) bi( 0, 0, 0) ti( 1, 1, 0) 3
gd( 1, 1, 1) bd( 2, 3, 1) bi( 0, 0, 0) ti( 0, 2, 0) 4
gd( 1, 1, 1) bd( 2, 3, 1) bi( 0, 0, 0) ti( 1, 2, 0) 5
Kernel launch successful.
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$ █ I
```

4. Here we have used dim3 bd(2,3,4) for block dim, and the grid dimension is default as (1,1,1). So we launch the kernel as kernel<<<1,bd>>>. Hence, we have 1 block in x direction and because of the dimension of the block we'll have **2 threads in x direction, 3 threads in y direction, and 4 threads in z direction**. Hence totaling **24 threads in one block** and as we have one block **we'll have 24 threads in total**.

```
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$ nvcc 04.cu -o four
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$ ./four

gd = gridDim
bd = blockDim
bi = blockIdx
ti = threadIdx

gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 0, 0) 0
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 0, 0) 1
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 1, 0) 2
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 1, 0) 3
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 2, 0) 4
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 2, 0) 5
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 0, 1) 6
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 0, 1) 7
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 1, 1) 8
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 1, 1) 9
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 2, 1) 10
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 2, 1) 11
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 0, 2) 12
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 0, 2) 13
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 1, 2) 14
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 1, 2) 15
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 2, 2) 16
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 2, 2) 17
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 0, 3) 18
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 0, 3) 19
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 1, 3) 20
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 1, 3) 21
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 0, 2, 3) 22
gd( 1, 1, 1) bd( 2, 3, 4) bi( 0, 0, 0) ti( 1, 2, 3) 23
Kernel launch successful.
pratik@herald-OptiPlex-3050:~/workshop/kernel_dimensions$
```

5. Here we have used dim3 bd(2,3,4) for block dim, and the grid dimension is set to (5,1,1). So we launch the kernel as kernel<<<5,bd>>>. Now, we have **5 blocks in x direction** and because of the dimension of the block **we'll have 2 threads in x direction, 3 threads in y direction, and 4 threads in z direction.** Hence totaling **24 threads in one block** and as we have one **5 blocks we'll have 120 threads in total.**

```
pratikgheald-OptiPlex-3050:~/workshop/kernel_dimensions$ nvcc 05.cu -o five
pratikgheald-OptiPlex-3050:~/workshop/kernel_dimensions$ ./five

gd = gridDim
bd = blockDim
bl = blockIdx
tl = threadIdx

gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 0, 0) 24
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 0, 0) 25
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 1, 0) 26
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 1, 0) 27
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 2, 0) 28
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 2, 0) 29
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 0, 1) 30
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 0, 1) 31
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 1, 1) 32
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 1, 1) 33
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 2, 1) 34
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 2, 1) 35
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 0, 2) 36
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 0, 2) 37
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 1, 2) 38
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 1, 2) 39
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 2, 2) 40
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 2, 2) 41
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 0, 3) 42
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 0, 3) 43
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 1, 3) 44
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 1, 3) 45
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 0, 2, 3) 46
gd( 5, 1, 1) bd( 2, 3, 4) bl( 1, 0, 0) tl( 1, 2, 3) 47
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 0, 0) 48
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 0, 0) 49
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 1, 0) 50
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 1, 0) 51
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 2, 0) 52
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 2, 0) 53
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 0, 1) 54
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 0, 1) 55
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 1, 1) 56
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 1, 1) 57
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 2, 1) 58
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 2, 1) 59
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 0, 2) 60
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 0, 2) 61
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 1, 2) 62
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 1, 2) 63
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 2, 2) 64
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 2, 2) 65
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 0, 0, 3) 66
gd( 5, 1, 1) bd( 2, 3, 4) bl( 2, 0, 0) tl( 1, 0, 3) 67
...
```

7. Here are the understanding i got after testing various dimensions:

- When we use print statements in cuda it cannot directly print the result in the host terminal so it has a fixed sized print buffer to hold any print statement which is later printed in the terminal after the device is synchronized.
So, printing a lot of information can lead to buffer overflow
- Threads execute in warps of 32, so for better efficiency we need to make **total threads per block multiple of 32**.
- More blocks offers better SM utilization for better load balancing.
- For example, for a total of 4096 threads: 16 blocks with 256 threads each will give better performance than a setup of 8 blocks with 512 threads. Although, in both the cases the threads are exactly multiple of 32 but the number of blocks take the weight here for performance.
- Making 2D 3D thread will not give better performance, we need to follow the above steps for better performance no matter if we use 1-D,2-D, or even 3-D dimensions. (**NOT 100% sure**)

Hence, among the various configurations of the kernel launch. The one with bd(5, 3, 4) gd(2, 3, 4) is better as it has 60 threads per block which **gives better warps per block**. Also it has more blocks which promotes better SM utilization.