**Necessary points for understanding GPU:**

**> GPU's architecture and Programming**
**>Warp Specialization and Named Barriers**
**>Motivating Kernel(**which is a need for verification of code using named barriers**)**

we shall dive deep into above mentioned points.

**GPU's architecture and Programming :** As mentioned in the last summary CUDA that was called as proxy for the GPU programming model,is the only interface that supports named barriers(synchronization points) which is necessary for constructing warp-specialized kernels. There are so many different frameworks available for GPUs that provides variations on the same data-parallel programming model.

Streaming Multiprocessors : the part of GPU's that run CUDA kernels.A streaming multiprocessor consists of thousands of registers which can be shared by threads, multiple caches, warp schedulers and execution cores for integer and floating point operations.

Collection of thread blocks (CTAs) is created when a CUDA kernel is launched to execute the program.1 CTA is formed of 1024 threads. CTA's are dynamically assigned to one of the streaming multiprocessors(which is also lying inside GPU's) by some hardware internal part of the GPU's.

**Communication Mechanism among Threads:**

- Within the same CTA,CUDA provides an on-chip shared memory which will be there for managing software which will be visible to all the threads within the same CTA,to coordinate access to shared memory,CUDA supports syncthreads primitive that performs a CTA wide blocking barrier
- In CUDA we can only support synchronization between threads that lie within the same the CTA, that are executed concurrently.

Threads in logical CTA' represent flat collection of threads.Usually, they are the groups of 32 which is called as warps .

In the Producer-Consumer named barriers example, producer warp and consumer warp are there in sync and it continues with wait ,begin , ready and signal.

SM has hardware sheduler during warp allocation granurality, which is *a constraint in the hardware resource allocation*. Once the warp is scheduled all threads in the same CTA execute same instruction.But in case of branch instruction,not taken branch will be taken care and executed with the thread that took the branch masked off and only then taken branch will taken care of.

The common source of performance degradation is branch-divergence which is the resulting serialized execution of of different branches within a warp

for example not taken and taken branch.so the reason for using warp-specialized programs is that no penalty is threads in different warps diverge,There will be no performance degradation by making all threads to continue to execute the same instruction.

## Motivating kernels:

Two important properties of the warp specialized kernels:

1. The reuse of the named barriers in the same warp group execution: writers provide the execution sequence of the operations in the warp sets which are data parallel and warp wide. The most important condition to reuse the named barriers would be to check happens before condition for the warps participating in the named barriers by taking a look at operations flow char.
2. The assumption that the flow of the programs written is static that is can be determined at compile time: They ensure that the assumptions hold true for all the warp specialized functions and that its very common to have programs whose execution flow is deterministic.

## Warp-Specializd and Named barriers:

```
1 global void launch bounds (64,1) example deadlock(void) {
2 assert(warpSize == 32);
3 assert(blockDim.x == 64);
4 int warp id = threadIdx.x / 32;
5 if (warp id == 0) {
6 // bar.sync (barrier name), (participants);
7 asm volatile(”bar.sync 0, 64;”);
8 asm volatile(”bar.arrive 1, 64;”);
9 } else {
10 asm volatile(”bar.sync 1, 64;”);
11 asm volatile(”bar.arrive 0, 64;”);
12 }
```

This is the code for deadlock:named barriers do not cause a dead lock situation.

there are two warps that are running in parallel at different speeds. The warp with warp id 0 blocks itself at named barrier 0 with the sync operation waiting for the warp with warp id 1 to register itself at named barrier 0 by calling arrive but that never happens because the warp with warp id 1 itself blocks at named barrier 1 waiting for the warp with id 0 to signal by arriving at named barrier 1 which never happens!