**IOS Assignment:**

**Summary1**

**Important keywords that need an explanation for better understanding of the paper-**

**Data parallel GPU programming model** emphasizes the distribution of data across multiple computing nodes and executing single instruction on these various distributed components. Current GPU programming models, such as OpenCL and CUDA, support data-parallel computations where all threads execute the same instruction stream on different arrays or matrices of data.

**Operational semantics** is a category of [formal programming language semantics](https://en.wikipedia.org/wiki/Semantics_(computer_science)) in which certain desired properties of a program, such as correctness, safety or security, are [verified](https://en.wikipedia.org/wiki/Formal_verification) by constructing proofs from logical statements about its execution and procedures, rather than by attaching mathematical meanings to its terms.

**Thread blocks:** In parallel computing many light weight processes can be combined together to get what are called as thread blocks. The threads in a given thread blocks can be either executed serially or in parallel. The maximum number of the threads that can be included in thread blocks was limited to 512 until June 2019, when the with Cuda 10 1024 threads could be included in one thread block. The threads in a block run on same stream processor and they communicate with each other via shared memory, barrier synchronization or atomic operations.

**Barrier synchronization** emphasizes that some threads in the parallel execution should be barred or stalled until all the threads executing in that thread block can reach the same point after which all the threads continue to run at that point.

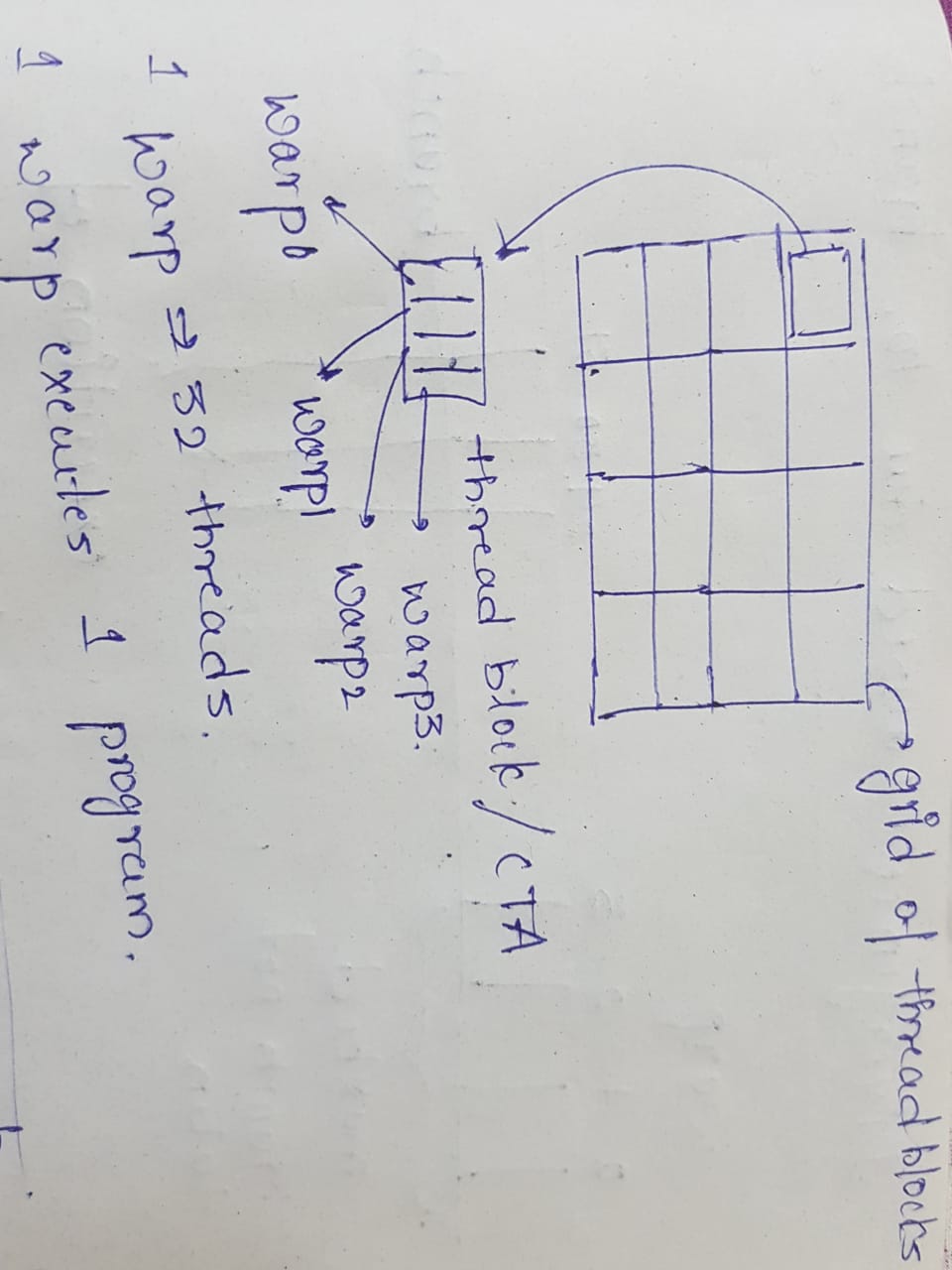
**References used for warp specialization:**

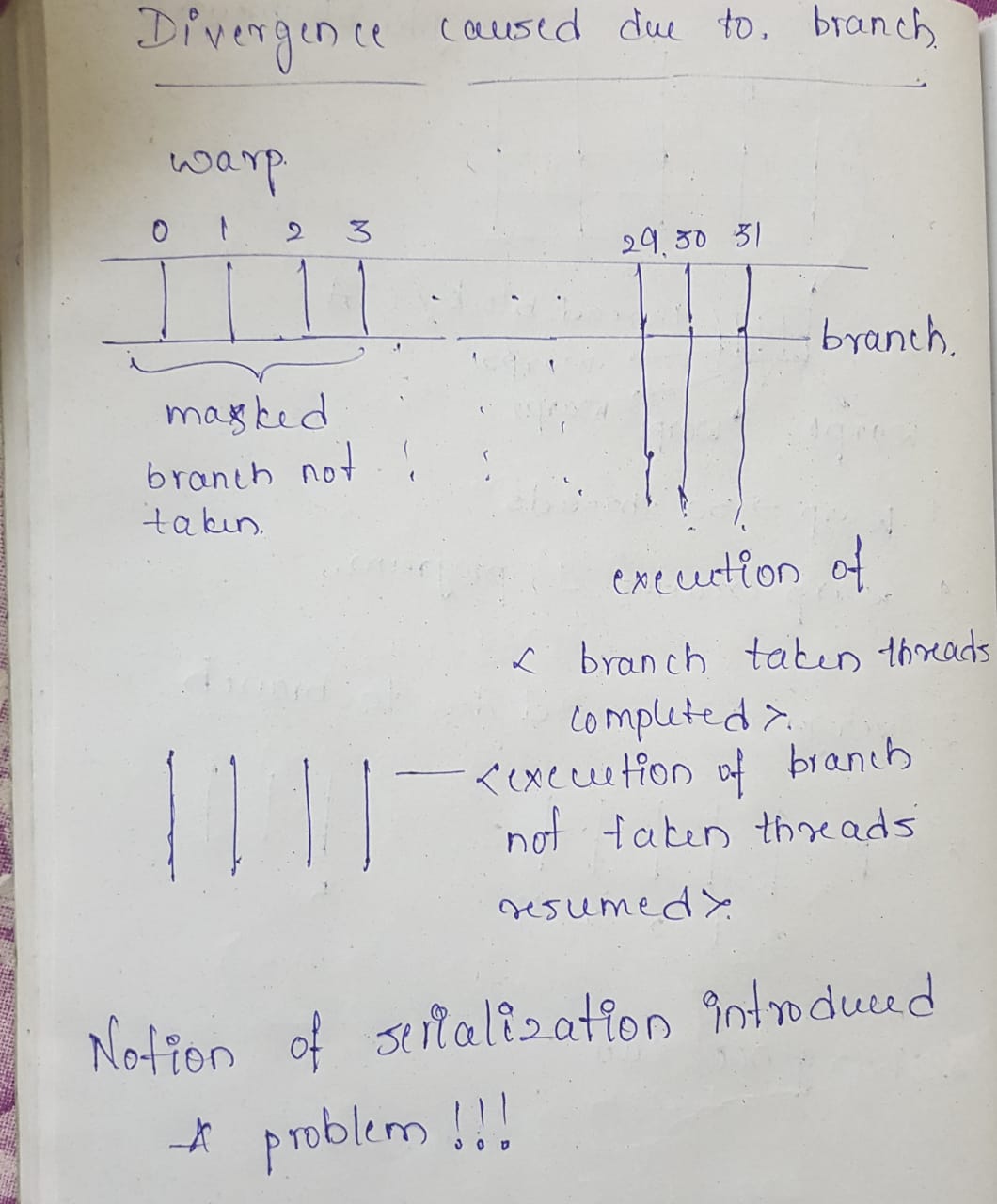
***Singe: Leveraging Warp Specialization for High Performance on GPUs***

Advantage of opting for warp specialization kernels over traditional data parallel programming models is that the warp specialized kernels allow computations to be partitioned into sub computations which can then be assigned to the warps in the thread blocks (which are group of 32 threads). These warps can communication by changing the variable names rather than using the shared memory which is faster. The partitions allow warp specialized kernels to efficiently handle the irregular data access and computation. The partitions also facilitate fitting the large working data sets into the on chip memory.

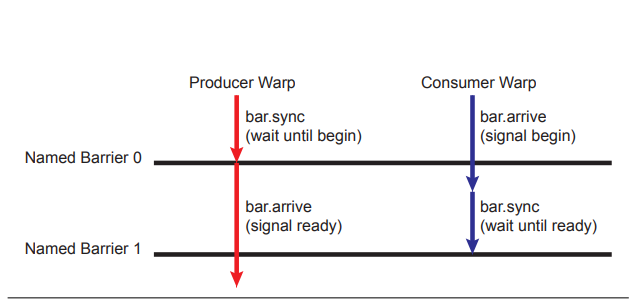
**Brief introduction about warp specialization model in CUDA:**

Cuda issues grids of thread blocks that can communicate within the grid, each block is made up of threads grouped into warps of 32 threads. The thread blocks are also known as Cooperative Thread Arrays. All the threads in a warp execute same program in parallel. Each warp executes on a single multiprocessor. When threads in a warp encounter a branch instruction, the SM executes the warp with all the threads that take the branch by masking the ones that don’t followed by executing the warp by masking off the threads that took the branch. This essentially brings serialization into picture degrading the Performance of the GPU as one set of the threads in the warp take one control path and other set takes other path after the first set has finished executing. The warp specialization requires a powerful synchronization mechanism to work efficiently and correctly. One of the options is to use named barriers. Two barriers, one non-blocking known as *arrive* barrier and other blocking known as *sync* barrier can be used.





Example of how the producer-consumer relationships can be encoded in the warp specialized programs is shown below.



The above diagram shows the use of named barriers along with the sync and arrive operation to synchronize movement of data between the consumer and producer warps. The producer warp waits for the consumer warp to signal that the buffer in the shared memory is ready to be filled at the named barrier 0. The consumer warp reaches the point of named barrier 0 and signals using a non-blocking arrive operation that the buffer is ready. As the consumer has used arrive operation it can perform other tasks. After a point consumer warp blocks itself by performing sync operation at named barrier 1 so that the buffer is full. The producer warp signals the consumer that the buffer is full by performing non-blocking arrive operation for the consumer to read the buffer.

**The main objective of the paper given to us is to define what it means for a warp specialized program to be correct and to measure the correctness of such programs. The paper presents the algorithms to verify the correctness. The authors of the paper also present the Weft algorithm to verify the correctness of the warp specialized code.**