**INTRODUCTION SUMMARY**

GPU is Graphical processing unit which is a class by itself which goes far beyond the basic graphic functions , and is programmable and powerful computational device. GPU's are useful for applications that require computational resources and bandwidth.

Different writing tools for verifying the correctness of GPU kernels assumed a data-parllel GPU programming model by CUDA and Open CL. In this model kernels execute:

1. Load data on chip

2. Perform computation

3. Write data off chip

A thread block is a programming abstraction that represents a group of threads that can be executed serially or in parallel. Threads in the same block can communicate with each other via shared memory, barrier synchronization or other synchronization primitives such as atomic operations. To synchronize all three phases, a barrier is required. In data-parllel GPU programming model a thread block wide barrier is sufficient because all threads in the kernel perform same computation.

Warp Specialized kernels assign different computations to warps (group of 32 threads within the same thread block) to achieve performance goals and memory bandwidth. To synchronize different warps, warp specialized kernels use producer-consumer named barriers. Named barriers are implemented directly in hardware.

Producer do not block on named barrier. 3 properties to check for warp specialized kernels:

1) Deadlock Freedom: Use of named barriers should not result in deadlock

2) Safe barrier recycling: Named barriers are limited , make sure to check they are properly used

3) Race freedom: shared memory access synchronization of named barriers are race free.

Weft is a verifier for warp specialized code.

Sound - Developer doesn't need to test the above three properties.

completeness –Guarantees that all viloations reported by WEFT are actual bugs , there are no false positives

Efficient – WEFT is enough to verify the real codes

**GPU ARCHITECTURE , WARP SPECILIZATION**

CUDA - supports named barrier primitives

When CUDA kernel is launched on GPU, CTAs is created (1024 threads-group of 32 called warps). CUDA has software for shared memory to communicate between threads within CTA.CUDA supports syncthread primitive, which performs a CTA-wide blocking barrier to provide synchronization.

Warp Specialization and Named Barriers

CudaDMA specilizes warps i.e compute and DMA warps to optimize the loading of data in shared memory. Singe Compiler-handles mapping of static data flow graph onto CTA.

Warp specilizations kernels communicate through shared memory and is assymetric in nature.

DMA warps -producer, compute warps -consumer

To handle producer-consumer synchronization , GPU’s support named barriers in PTX

PTX has two instructions –

sync - causes threads in warp arriving at the barrier to block until barrier completes.

arrive- allows a warp to register arrival at the barrier and execute without blocking.

**RESEARCH**

WARP SPECILIZATION:

In the data parallel model, a collection of threads within a thread block all execute the same program over independent elements from arrays of input data. On the hardware, however, a thread block is broken into warps consisting of (typically) 32 threads which serve as the unit of scheduling. Warp specialization exploits the division of a thread block into warps to partition computations into sub computations such that each sub-computation is executed by a different warp within a thread block

Cuda DMA:

* Declare CudaDMA object to manage shared buffer
* Separate DMA and compute warps
* Provide synchronization primitives
* Perform repeated transfer operations
* Declare CudaDMA object to manage buffer
* Split DMA warps from compute warps
* Process buffer using compute warps

EXECUTION MODEL:

* Use PTX Named barriers
  + bar sync
  + bar arrive

