Blue Waters Petascale Semester Curriculum v1.0

Unit 7: CUDA

Lesson 11: Branching and GPGPU Efficiency (profiling and debugging)

Developed by David A. Joiner

for the Shodor Education Foundation, Inc.

Browse and search the full curriculum at
http://shodor.org/petascale/materials/semester-curriculum

We welcome your improvements! You can submit your proposed changes to this material and the rest of the curriculum in our GitHub repository at https://github.com/shodor-education/petascale-semester-curriculum

We want to hear from you! Please let us know your experiences using this material by sending email to petascale@shodor.org

Branching and GPU Efficiency

Many-Core, but not independent

- GPUs are many core, but the structure of memory in the GPU and how threads are organized to access the memory is strictly tied together.
- Threads are bundled in “warps” which have access to the same register and have a bundled set of instructions

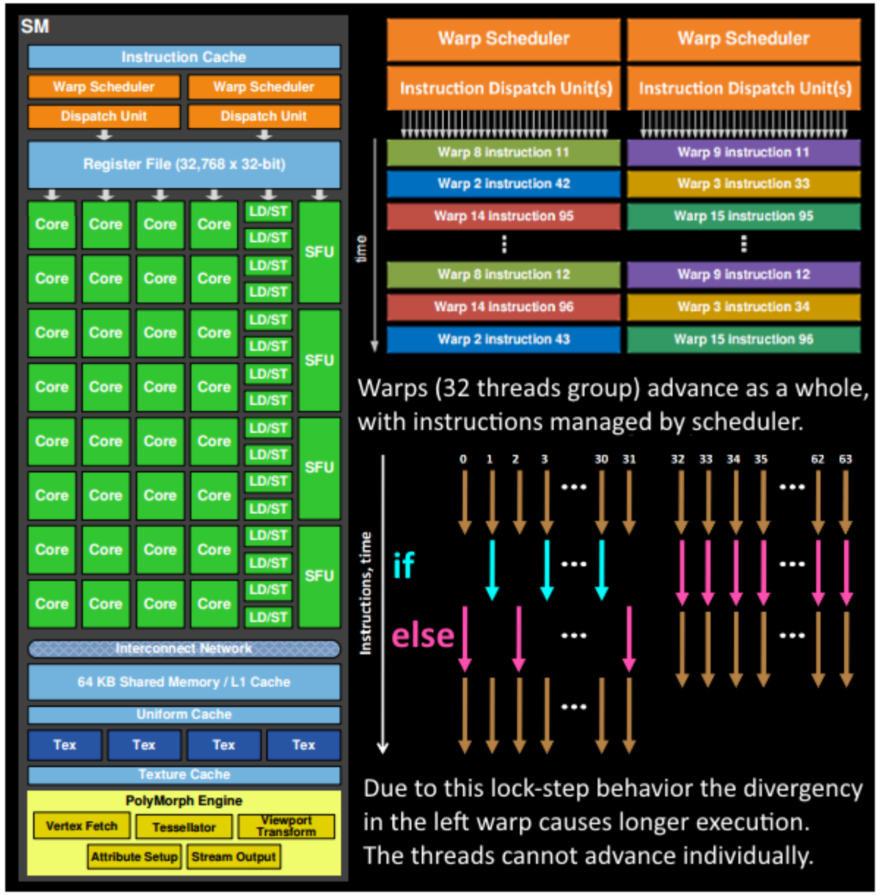
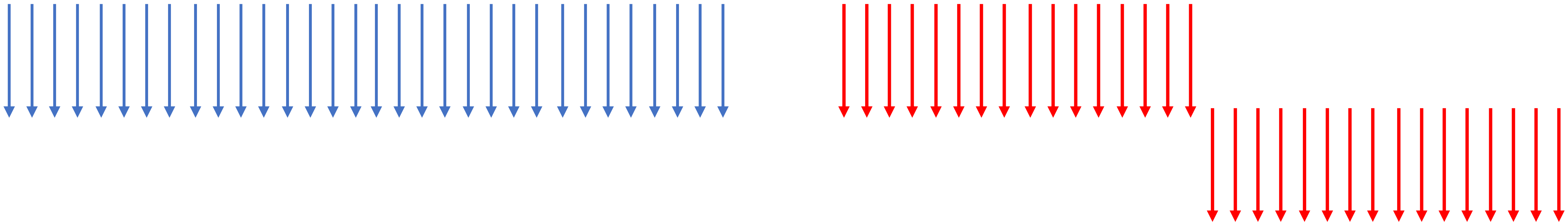


Image From: https://developer.nvidia.com/content/life-triangle-nvidias-logical-pipeline

Branching can reduce efficiency

- Any time one of the (typically) 32 threads in a warp does something different from the others, all of the others have to wait for the thread to complete it’s branched activity



What branching looks like in your kernel

```
__global__ void branching(int n)
{

 int threadMod = threadIdx.x%threadsPerBlock;

     if ( threadMod < n) {
         ...
     } else if(threadMod<2*n){
         ...
     } else if(threadMod<3*n){
         ...
     } else if(threadMod<4*n){
         ...
     }
}
```