Dynamic Thread Block Launch:  
A Lightweight Execution Mechanism to Support Irregular Applications on GPUs  
Jin Wang†, Norm Rubin*, Albert Sidelnik* and Sudhakar Yalamanchili†  
†Georgia Institute of Technology, *NVIDIA Research

Motivation  
- GPUs are effective for structured applications  
  - Rigid 1/2/3D data structure
- However, for unstructured applications  
  - Poor workload balance -> control flow divergence  
  - Un-coalesced memory access -> memory divergence  
  - Low Compute Utilization

Architecture Extensions and SMX Scheduling  
- Introduce new microarchitecture data structures to manage the aggregated groups and TBs  
- The SMX Scheduler is extended for efficient scheduling of aggregated TBs  
- Simulated using GPGPUSim²

DTBL: Dynamic Thread Block Launch  
- Extend the current GPU execution model with DTBL  
- Thread blocks can be dynamically launched from a GPU thread  
- Use light-weight thread blocks rather than heavy-weight device kernel for DFP

Potential Benefits  
- Reduce control divergence  
- Increase coalesced memory accesses

Primary Kernel  
- Native Kernel: original launched by host or device  
- Native TB: thread blocks in a native kernel  
- Aggregated TB: dynamically launched thread blocks  
- Aggregated Group: a group of aggregated TBs organized in three dimensions that are coalesced with a native kernel  
- Aggregated Kernel: kernel with aggregated TBs  
- New aggregated TBs can be coalesced with the same kernel (a) or to a different kernel (b)

Thread Hierarchy, Memory Model and Synchronizations  
- Same as the original CUDA programming model  
- Three-dimension hierarchy  
- Local memory is private to thread, shared memory is private to TB  
- No synchronizations between TBs

Programming Interface  
- Utilize current CDP device runtime APIs  
- Add a new device runtime API call: cudaLaunchAggGroup

References  