Exploring Dynamic Parallelism for Irregular Applications on GPUs

Jin Wang¹, Norm Rubin*, Albert Sidelnik* and Sudhakar Yalamanchili†
¹Georgia Institute of Technology, *NVIDIA Research

Irregular Applications and DFP
• 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

- Child kernels in CDP or child thread blocks in DTBL are
  assigned to TBs
- Assign priority information to TBs

Dynamically Formed Parallelism (DFP)
- Pockets of Dynamically Formed Structured Parallelism
- Locally uniform control flow
- Locally coalesced memory access

Locality-Aware Kernel/Thread Block Scheduler
• Utilize the spatial locality and temporal locality between the parent and the child
• Child kernels in CDP or child thread blocks in DTBL are
  scheduled closer to parents
• Spatial: children are scheduled immediate after they are invoked
• Temporal: children are scheduled the same SMX as parent if possible
• Load balancing: allow SMXs to pick children of
  other parents if necessary

Microarchitecture Implementation
• Assign priority information to TBs
• Assign SMX binding information to TBs
• Extended SMX scheduler to manage
  prioritized TBs

Implementing Irregular Applications with CDP[1]
• Launch a child kernel for detected DFP
• Launch only when sufficient parallelism is available

Parent Kernel
Launch by T1
Launch by T3
Launch by T5
Launch by T7
Child Kernels
Launch by T1
Launch by T3
Launch by T5
Launch by T7

Potential Benefits
• Reduce control divergence
• Increase coalesced memory accesses

Issues
• Large number of kernels
• Kernel launch overhead
• Memory footprint

Overall Performance: Speedup over flat implementations
CDP-ideal: excluding launch overhead, average 1.43x speedup
CDP-actual: including launch overhead, average 1.16x slowdown

DTBL: Dynamic Thread Block Launch[2]
• Extend the current GPU execution model with DTBL
• Thread blocks can be dynamically launched from a GPU thread

Possible Benefits
• Reduce launching latency
• Increase SMX execution efficiency
• Another 1.27x speedup over DTBL

Locality-Aware Scheduler
• Increase cache hit rate and memory efficiency

References