KLAP: Kernel Launch Aggregation and Promotion for Optimizing Dynamic Parallelism

Izzat El Hajj (Illinois), Juan Gómez-Luna (Córdoba), Cheng Li (Illinois), Li-Wen Chang (Illinois), Dejan Milojicic (HPE), Wen-mei Hwu (Illinois)

elhajj2@illinois.edu, elgoluj@uco.es

Kernel Launch Aggregation

**Problem:** Launching too many fine-grain kernels
- Large kernel count incurs too much launch overhead
- Fine granularity of kernels underutilizes GPU resources

**Solution:** One thread launches a kernel on behalf of a group

![Warp-Granularity Kernel Launch Aggregation](Image)

![Block-Granularity Kernel Launch Aggregation](Image)

**Code Transformation:**
```c
// Original Kernel Call
allocate arrays for args, g0, and b0 store in g0 array, and b0 array new g0 = g0 array across warp/block new m0 = m0 array across warp/block threadIdx = threadIdx in warp/block

// Transformed Kernel Call (block-granularity aggregation example)
allocate postponed arg buffers, threadIdx = threadIdx in warp/block
```

**Results:**

- **Kepler:** Increasing aggregation granularity improves performance (geometric speedup of 6.3× for K-aggregation on Kepler)
- **Maxwell:** Performance improvement comes from reduced launch overhead and better resource utilization

Kernel Launch Promotion

**Problem:** Deep call stacks
- Launch overhead dominates the critical path
- Call stack has limited depth

**Solution:** One thread launches a kernel on behalf of a group

![Original Kernel](Image)

![Promotion](Image)

**Code Transformation:**
```
#include <cuda_runtime.h>

__global__ void kernel(params args) {
    // Kernel body
}

// Transformed Kernel Call (called from host)
#include <cuda_runtime.h>

__global__ void kernel_(params args) {
    // Transformed kernel body
}
```

**Results:**

![Achieved Occupancy](Image)

- **Kepler:** Achieved Occupancy: Aggregation improves occupancy due to fewer coarser grain kernels
- **Maxwell:** Instructions per second: Overlap improves instructions per second due to more work parallel work available