A Compiler Framework for Optimizing Dynamic Parallelism on GPUs

Mhd Ghaith Olabi¹, Juan Gómez Luna², Onur Mutlu², Wen-mei Hwu³, ⁴, Izzat El Hajj¹

¹American University of Beirut  ²ETH Zurich  ³NVIDIA  ⁴University of Illinois at Urbana-Champaign
Dynamic Parallelism on GPUs

- **Dynamic parallelism** enables executing GPU threads to launch other grids of threads

  - Useful for implementing computations with **nested parallelism**
Dynamic Parallelism Overhead

• Using dynamic parallelism may cause many small grids to be launched

• Launching many small grids causes performance degradation due to:
  • Congestion
    • Limited number of grids can execute simultaneously (others need to wait)
  • Hardware underutilization
    • If grids are small, there may not be enough threads launched to fully utilize hardware resources

• Solution: launch fewer grids of larger sizes
Prior Work: Aggregation

• Aggregation is an optimization where:
  • Multiple child grids are consolidated into a single aggregated grid
  • One parent thread launches the aggregated grid on behalf of the rest

Prior Work: Aggregation

• Aggregation is an optimization where:
  • Multiple child grids are consolidated into a single aggregated grid
  • One parent thread launches the aggregated grid on behalf of the rest

+ Reduces congestion by reducing the number of launched grids
+ Improves utilization because aggregated child grids have more threads than original ones
Contributions

• **Thresholding** (as a compiler optimization)
  • Prior work relies on programmers to apply it manually

• **Coarsening** of child thread blocks
  • Prior work on compiler-based coarsening not specialized for dynamic parallelism

• **Aggregation** of child grids at multi-block granularity
  • Prior work only compiler-based aggregation only considers warp, block, and grid granularity

• One compiler framework that combined the three optimizations
Thresholding

• Thresholding is an optimization where:
  • A grid is launched dynamically only if the number of child threads exceeds a certain threshold
  • Otherwise, work is executed sequentially by the parent thread
Thresholding

• Thresholding is an optimization where:
  • A grid is launched dynamically only if the number of child threads exceeds a certain threshold
  • Otherwise, work is executed sequentially by the parent thread

+ Reduces congestion by reducing the number of launched grids
+ Improves utilization by only allowing grids with many threads to be launched
Coarsening

- Coarsening is a transformation where:
  - The work of multiple child blocks is assigned to a single child block
Coarsening

- Coarsening is a transformation where:
  - The work of multiple child blocks is assigned to a single child block

+ When applied before aggregation, amortizes the cost of disaggregation (incurred once per child blocks)
Multi-block Granularity Aggregation

- Multi-block granularity aggregation is an optimization where:
  - The child grids of multiple parent blocks are consolidated into a single aggregated grid
Multi-block Granularity Aggregation

• Multi-block granularity aggregation is an optimization where:
  • The child grids of multiple parent blocks are consolidated into a single aggregated grid

+ Compared to block granularity, launches fewer and larger grids
+ Compared to grid granularity, launches child grids more eagerly
We evaluate all combinations of optimizations for 7 benchmarks with 2 datasets each
Overall Speedup

We report speedup (higher is better) over the baseline that uses CUDA dynamic parallelism (CDP)
Observation #1: Not using CDP performs better than naïve CDP (same observation as prior work).
**Observation #2:** Aggregation improves performance of naïve CDP (same observation as prior work).

KLAP(CDP+A) is 12.1× faster than CDP on average (geomean).
Observation #3: Thresholding alone improves the performance over CDP. CDP+T is 13.4× faster than CDP on average (geomean).
Observation #4: Thresholding and Aggregation together improve the performance over CDP even more.

Despite both targeting the same source of inefficiency, one optimization does not obviate the other.
Observation #5: **Coarsening** alone does not improve performance substantially over **CDP**. **CDP+C** is $1.01\times$ faster than **CDP**.
**Overall Speedup**

- **Observation #6:** Coarsening does improve performance when combined with the other optimizations.
- Recall: main benefit was amortizing overhead of aggregation. \( \text{CDP+T+C+A} \) is \(1.22\times\) faster than \( \text{CDP+T+A} \).
Summary

• We present a **compiler framework** for optimizing the use of dynamic parallelism on GPUs in applications with nested parallelism

• The framework includes **three key optimizations**:
  • Thresholding
  • Coarsening
  • Aggregation

• Our evaluation shows that our compiler framework **substantially improves performance of applications with nested parallelism** that use dynamic parallelism
  • 43.0× faster than CDP.
  • 8.7× faster than No CDP
  • 3.6× faster than prior aggregation work (KLAP)
Thank you!

A Compiler Framework for Optimizing Dynamic Parallelism on GPUs

Mhd Ghaith Olabi\textsuperscript{1}, Juan Gómez Luna\textsuperscript{2}, Onur Mutlu\textsuperscript{2}, Wen-mei Hwu\textsuperscript{3,4}, Izzat El Hajj\textsuperscript{1}

\textsuperscript{1}American University of Beirut \quad \textsuperscript{2}ETH Zurich \quad \textsuperscript{3}NVIDIA \quad \textsuperscript{4}University of Illinois at Urbana-Champaign

Contact: moo02@mail.aub.edu