CUDA DYNAMIC PARALLELISM WITH SCANSTACK

UNDERSTANDING CUDA DYNAMIC PARALLELISM

Leveraging dynamic parallelism for flexible programming

DYNAMIC KERNEL LAUNCHING

Dynamic parallelism allows kernels to launch other kernels for flexibility.

FLEXIBILITY IN WORKLOADS

This feature enables adaptable parallel programming models suitable for varying workloads.

REDUCED HOST OVERHEAD

It minimizes the need for host involvement, improving performance in certain applications.

ENHANCED PROGRAMMING PATTERNS

Supports more complex programming patterns that can be optimized for specific tasks.

APPLICATIONS IN REAL SCENARIOS

Particularly useful in scenarios with dynamic workload changes, like simulations.

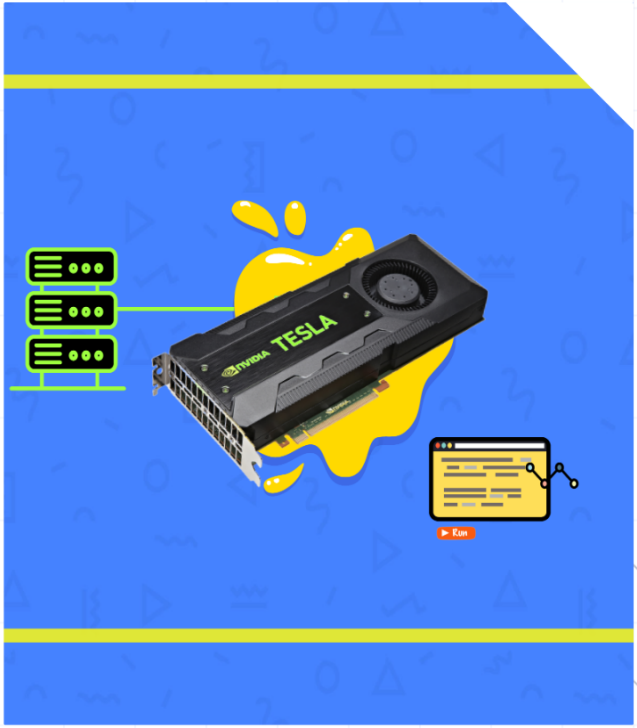
ESSENTIALS FOR CUDA DYNAMIC PARALLELISM

- **COMPUTE CAPABILITY REQUIREMENTS**

  Requires devices with Compute Capability 3.5+ (Kepler and later)

- **COMPILATION WITH DEVICE CODE SUPPORT – RELOCATABLE DEVICE CODE**

  Use -rdc=true to enable device code support for dynamic parallelism in the CUDA program.



UNDERSTANDING STACK CELL STATES IN CUDA

Key definitions for managing CUDA stack states

DEFINITION OF EMPTY STATE

The EMPTY state signifies an unoccupied cell in the stack, initialized to -1.

DEFINITION OF INVALID STATE

The INVALID state indicates a cell that has been invalidated or popped, marked as -2.

IMPORTANCE OF STATE DEFINITIONS

State definitions are crucial for managing stack operations effectively within CUDA.

USAGE IN DYNAMIC PARALLELISM

These states are essential for controlling flow in dynamic parallelism scenarios in CUDA applications.

UNDERSTANDING OPERATION REQUEST STRUCTURE

A look at the OpRequest structure

STRUCTURE OVERVIEW

The OpRequest structure facilitates stack operations in C++.

OPERATION TYPE

Type field indicates operation: 1 for push, -1 for pop.

VALUE FIELD

Holds the value for push operations; unused in pop.

RESULT FIELD

Stores operation result; INVALID for failures, actual value for success.

IMPLEMENTING STACK OPERATIONS IN CUDA

PUSH OPERATION OVERVIEW

The function aims to push a value onto the stack efficiently.

CONTENTION HANDLING

The lowest granularity addresses potential conflicts during simultaneous pushes.

POP OPERATION OVERVIEW

The function pops the top-most element of the stack.

CUDA DYNAMIC PARALLELISM

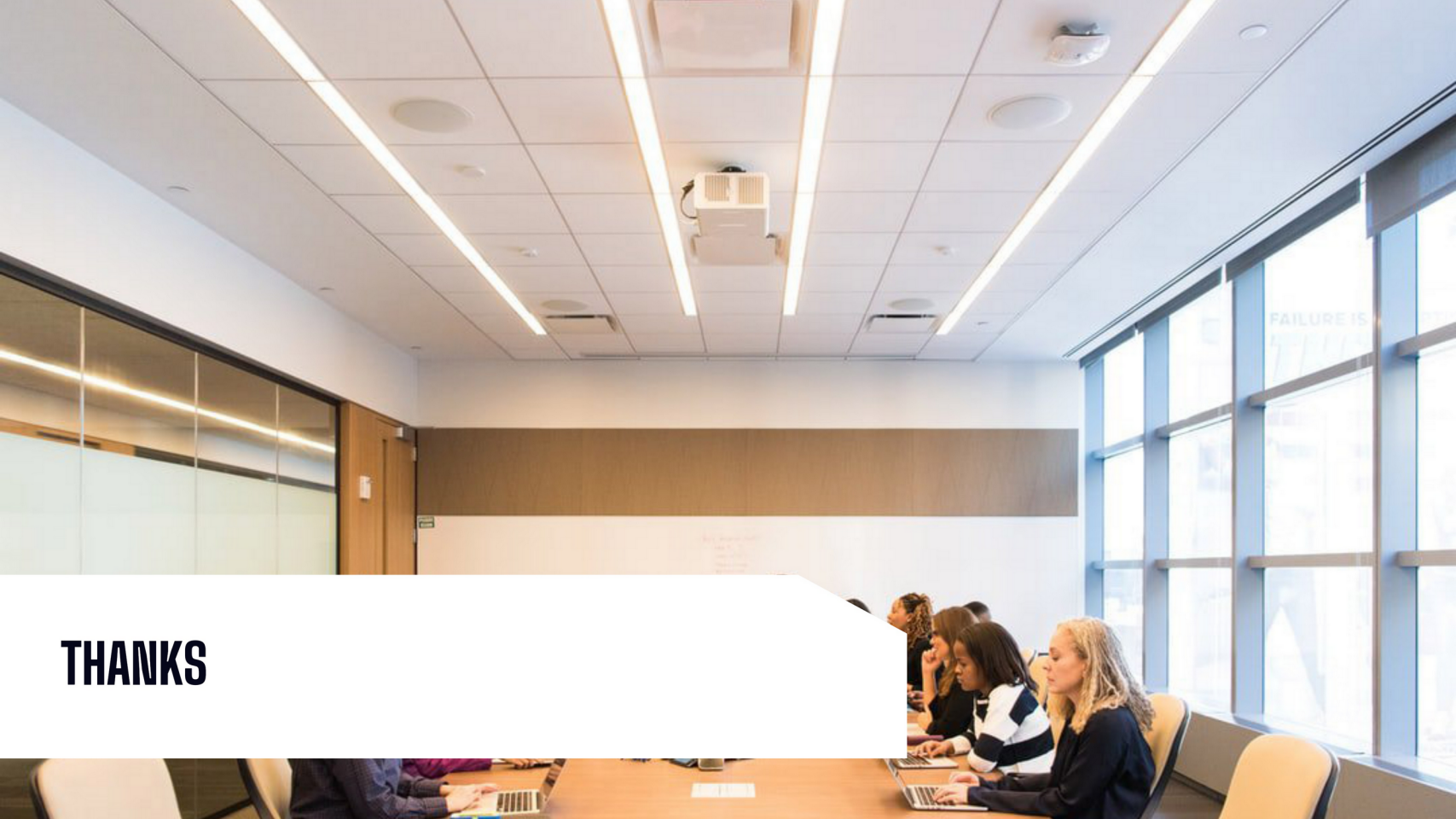
Utilizes CUDA's features for dynamic parallel execution of device functions.

FUTURE WORK

- **SCALABILITY**
  Extend the implementation to support higher number of operations.
- **OPTIMIZING CONTENTION HANDLING**
  Focus on improving how contention among threads is managed during operations.
- **EXPLORING ADDITIONAL FUNCTIONALITIES**
  Further developments could include new features to enhance stack capabilities and usability.

IDIA®

Cuda
Progr
Lang



THANKS