# Consolidate and Advance the GPU infrastructure in Clad


## Mentors: Aaron Jomy, David Lange, Vassil Vassilev



---


## Presented By

### Vedant Goyal

# About Me

- **Academic Background**

  Currently pursuing B.E. in Electrical and Computer Engineering  at Thapar Institute of Engineering and Technology, Patiala

- **Work Experience**

  Undergraduate Researcher with experience in computer vision, machine learning, worked on YOLO-based satellite imagery and skin cancer classification systems.

# Project Overview

**Clad:** Clad is a Clang-based automatic differentiation tool that transforms C++ source code to compute derivatives efficiently.

```
double f_cubed_add1(double a, double b) {
 return a * a * a + b * b * b;
}
```

clad::differentiate

```
double f_cubed_add1_darg0(double a, double b) {
    double _d_a = 1;
    double _d_b = 0;
    double _t0 = a * a;
    double _t1 = b * b;
    return (_d_a * a + a * _d_a) * a + _t0 * _d_a + (_d_b *
b + b * _d_b) * b + _t1 * _d_b;
}
```

```
void foo(int *in, int *out){
   *out = 2 * *in;
}
```

clad::gradient

```
void foo_grad(int *in, int *out,
    clad::array_ref<int> _d_in) {
  *_d_out = 1;
  int _t0;
  _t0 = *out;
  *out = 2 * *in;

  *out = _t0;
  int _r_d0 = * _d_out;
  * _d_out -= _r_d0;
  * _d_in += 2 *  _r_d0;
}
```

**GPU support in Clad:** Clad can differentiate CUDA kernels and GPU-oriented C++ code at compile time.

# Current Implementation

- Clad currently supports differentiation of CUDA-based GPU code through Clang's compiler infrastructure and source transformation pipeline.
- Over the past few years, multiple contributors have implemented GPU-related features across several experimental branches and integrations.
- However, much of this work remains fragmented, inconsistently tested, and spread across older forks and development branches.
- Benchmark of workload validation is also limited, making it difficult to evaluate correctness, scalability, and performance consistently across GPU workloads.

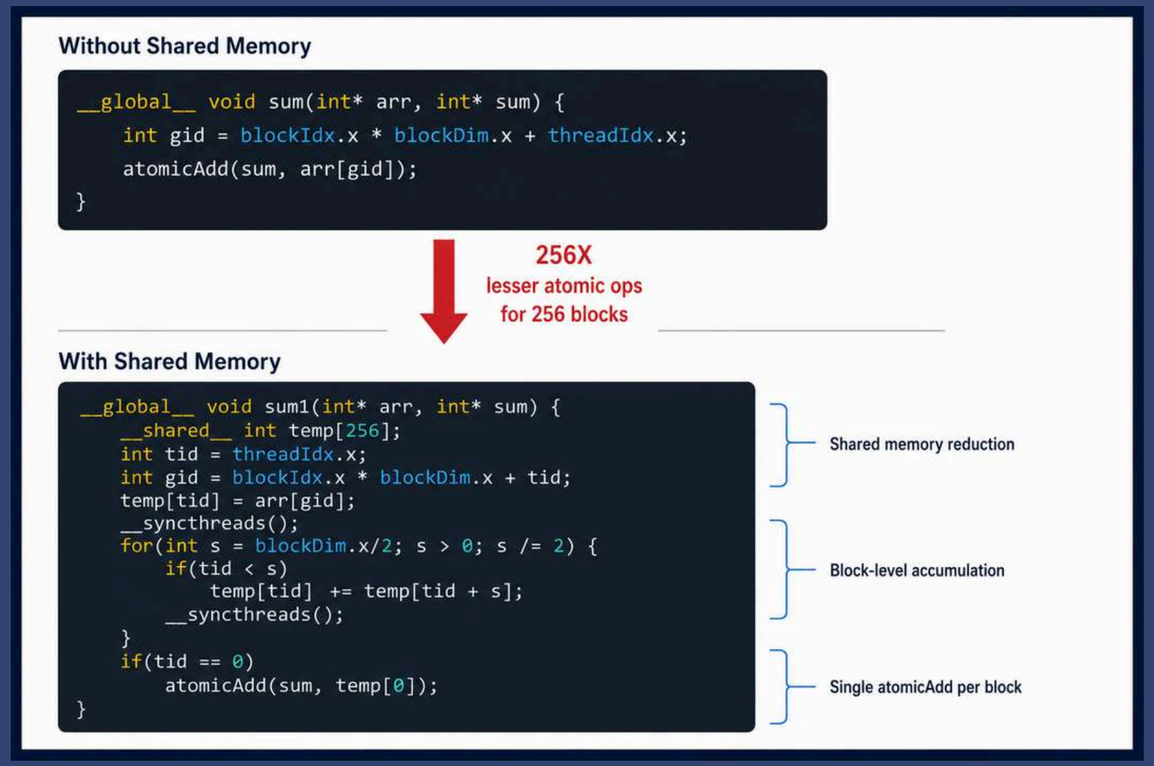
# Implementation Plan

## Reseach and Audit of work

- Investigate current GPU infrastructure and the existing GPU related branches to identify missing functionality and regressions

## Addressing the concurrency challenges

- Implementation of strategies like block level synchronization and other primitives like __syncthreads() will reduce the overdependance on expensive atomic operations.



HPC application integration

- Integration of HPC applications like LULESH, RSBench to current Clad infrastructure. Further integration of HPC test (like XSBench and LBM) and benchmarking them.

Testing and CI/CD integration

- Design and implement a CI pipeline that executes GPU-specific tests on capable runners, ensuring future commits do not break CUDA functionality.
- Develop practical examples like neural network training and optimization algorithms to demonstrate value and performance benefits.

Documentation and Finalization

- Create API documentation, mathematical background, usage examples, and performance guidelines for GPU support in Clad.

# THANK YOU!!