Support usage of Thrust API in Clad

Author: Abdelrhman Elrawy
Mentors: Vassil Vassilev, Alexander Penev

About Me

- **Academic Background**
  - **Master in Applied Computing** (Machine Learning & Parallel Programming)
    - Wilfrid Laurier University, Waterloo, ON (2024-2026)
  - **Bachelor of Science in Computer Science**
    - Helwan University, Cairo, Egypt (2020-2024)

- **Work Experience:**
  - **Machine Learning Engineer** (Simli AS, Norway)
    - Worked on creating animatable avatars. (June 2023, Feb 2025)

Project Context

Clad: A source-transformation automatic differentiation (AD) library in Clang.

Thrust: NVIDIA's powerful GPU-parallel algorithms and data structures library.

The Challenge:

- This project aims to enhance Clad by adding support for NVIDIA's Thrust library.
- By enabling differentiation of Thrust's GPU-parallel algorithms, Clad users will gain the ability to automatically generate gradients for CUDA-accelerated code.
- This work will bridge the gap between high-performance GPU computing and AD, potentially accelerating gradient-based optimization tasks by orders of magnitude.

Project Overview

Goal: To integrate Thrust function support into Clad, allowing automatic differentiation of GPU-accelerated code.

Approach:

- Extend Clad's source-to-source transformation engine to recognize Thrust primitives (e.g., transform, reduce).
- Implement custom derivatives for these Thrust operations.
- Validate performance through real-world use cases and benchmarks.

Personal Motivation

Combining Expertise & Passion

- This project aligns with my academic background and professional experience in **Machine Learning** and **Parallel Programming**.
- I am deeply interested in **automatic differentiation** and its potential to accelerate scientific computing and machine learning workloads.

Impact & Contribution

- The project has the potential to bridge a critical gap between high-performance GPU computing and AD, making advanced optimization techniques more accessible and efficient.
- Contributing to an open-source project like Clad offers an opportunity to make a real impact on the scientific community.

Project Implementation Overview

Phase 1: Research & Proof-of-Concept

- **Clad Differentiation Architecture Analysis:** Investigate Clad architecture and existing CUDA support.
- **Thrust API Analysis and Prioritization:** Identify and prioritize Thrust functions most valuable for AD (e.g., transform, reduce, scan).
- **Proof-of-Concept:** Manually implement and test forward and hand-written derivatives for key Thrust primitives (e.g., thrust::transform with a square function) in standalone programs.
- **Milestones:**
  - **Manually implement forward computations and hand-written derivatives for:**
    - `thrust::transform` (for element-wise vector scaling or squaring).
    - `thrust::reduce` (a simple sum).

Project Implementation Overview

Phase 2: Core Implementation

- **Thrust Function Recognition in Clad:** Extend Clad to recognize and process Thrust function calls
- **Basic Thrust Algorithms Support:** Implement custom derivative handlers for fundamental operations like thrust::transform and thrust::reduce (sum).
- **Advanced Thrust Algorithm Support:** Tackle more complex algorithms such as thrust::transform_reduce and thrust::inclusive_scan, addressing unique dependency patterns.
- **Milestones:**
  - Implement custom derivative handlers for foundational Thrust algorithms:
    - `thrust::transform` , `thrust::reduce`
  - Implement custom derivative handlers for more complex and composite operations:
    - `thrust::transform_reduce` (e.g., for dot products or L2 norms)

Project Implementation Overview

Phase 3: Testing & Integration

- **Real-world Integration Examples:** Develop practical examples like neural network training and optimization algorithms using Thrust and Clad to demonstrate value and performance benefits.
- **CI/CD Integration:** Integrate Thrust support with Clad's continuous integration/continuous deployment pipeline.

Phase 4: Documentation & Finalization

- **Comprehensive Documentation:** Create API documentation, mathematical background, usage examples, and performance guidelines for Thrust support in Clad.
- **Final Report and Presentation:** Prepare a detailed technical report, presentation materials, and future work recommendations.

Goals

- **Code Contributions:** Fully integrated Thrust function support within Clad.
- **Real-world Examples:** Development of practical examples like neural network training.
- **Performance Benchmarks:** Quantitative comparative analysis demonstrating speedup from GPU-accelerated differentiation.
- **Documentation:** Comprehensive API reference and user guide with practical code examples and tutorials.
- **Final Report & Presentation:** A detailed technical report outlining design decisions, challenges, and results, along with a presentation of key findings.
- **Future Work:** Explore support for other parallel computing frameworks like MPI, building on the experience gained from Thrust integration.

Thanks!