Support usage of Thrust API in Clad

Author: Abdelrhman Elrawy
Mentors: Vassil Vassilev, Alexander Penev

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.

Midterm Progress Summary

Successfully implemented custom derivatives for `thrust::reduce`:

- This includes support for multiple binary operations like `plus`, `maximum`, `minimum`, and `multiplies`.
- Special care was taken to handle edge cases, such as the presence of zeros in the input for the `multiplies` operation.

Added support for `thrust::inner_product`:

- Pullbacks for both the 4-argument and 6-argument versions of the function have been implemented.
- This supports various binary operator combinations, including (`plus`, `multiplies`), (`plus`, `plus`), and (`plus`, `minus`).

Ongoing Work:

- Adding support for another reduction operations to further expand Clad's capabilities.

Detailed Progress: thrust::reduce

Pull Request #1472: Add custom derivative for `thrust::reduce`

Key Implementations:

- A new header, `ThrustDerivatives.h`, was created to contain the custom derivative logic.
- The `reduce_pullback` function was implemented to manage the gradient calculation for various reduction operations.
- Here is a snippet for the `thrust::plus<T>` operation:

```
if constexpr (::std::is_same_v<BinaryOp, ::thrust::plus<T>>) {
  if (d_init)
    *d_init += d_output;

  struct add_d_output {
    T d_output;
    add_d_output(T d) : d_output(d) {}
    CUDA_HOST_DEVICE void operator()(T& x) const { x += d_output; }
  };

  if (n > 0) {
    ::thrust::for_each(d_first_dev_ptr, d_first_dev_ptr + n,
                       add_d_output(d_output));
  }
```

Detailed Progress: thrust::inner_product

Pull Request #1480: Added support for `thrust::inner_product`

Key Implementations:

- I extended `ThrustDerivatives.h` with `inner_product_pullback`.
- This new function supports both the standard inner product and versions with custom binary operations.
- The following is the gradient calculation for the standard inner product:

```
struct grad_functor {
  T d_output;
  grad_functor(T d) : d_output(d) {}
  CUDA_HOST_DEVICE void
  operator()(::thrust::tuple<T&, T&, const T&, const T&> t) const {
    // d_x1 += d_y * x2
    ::thrust::get<0>(t) += d_output * ::thrust::get<3>(t);
    // d_x2 += d_y * x1
    ::thrust::get<1>(t) += d_output * ::thrust::get<2>(t);
  }
};
```

Challenges & Solutions

1. **Mismatched Function Signatures**
   - **Problem:** Silent failures occurred as Clad didn't initially warn about incorrect pullback function signatures.
   - **Solution:** Manually debugged to find the issue.

2. **GPU Memory Errors**
   - **Problem:** Tracing memory access violations within the CUDA/Thrust environment was complex.
   - **Solution:** Used `compute-sanitizer` and careful GPU pointer management to resolve memory errors.

3. **Mathematical Edge Cases**
   - **Problem:** The derivative for `thrust::reduce` with multiplication was incorrect when the input contained zeros.
   - **Solution:** Implemented logic to count zeros and correctly handle the gradient for single and multiple zero-value inputs.

Future Goals

- **Core Implementation**:
  - Finalize support for other complex reduction algorithms.
  - Implement custom derivatives for `thrust::transform`.
  - Address more complex algorithms like `thrust::transform_reduce` and `thrust::inclusive_scan`.

- **Testing Use case**:
  - Develop practical, real-world examples, such as in neural network training.

- **Documentation & Finalization**:
  - Create thorough documentation for the new Thrust support in Clad.
  - Prepare the final project report and presentation

Thanks!