GSoC 2025

Implement and improve an efficient, layered tape with prefetching capabilities

Mentors: Aaron Jomy, David Lange, Vassil Vassilev

Presented by
Aditi Milind Joshi

About Me

Academic Background

- Currently pursuing B.Tech in Computer Science and Engineering (AIML) at Manipal Institute of Technology, Manipal

---

Work Experience

- AI and Data Intern at Deloitte Touche Tohmatsu India LLP
- Samsung PRISM Virtual Intern
- AI Member at Project Manas

Project Overview

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

Reverse Mode Automatic Differentiation: In reverse-mode automatic differentiation (AD), we need to store intermediate results during the forward pass for use during the backward (gradient) pass.

Tape: The tape is a stack-like data structure that stores intermediate values for reverse mode AD.

Current Implementation

- Currently, Clad uses a monolithic memory buffer as the tape.
- While this approach is lightweight for small problems, it becomes inefficient and non-scalable for larger applications or parallel workloads.
- Frequent memory reallocations, lack of thread safety, and the absence of support for offloading make it a limiting factor in Clad’s usability in complex scenarios.

Implementation Plan

Replacing dynamic reallocation with a slab-based memory structure

Instead of reallocating memory, the propose structure uses a slab-based memory allocation strategy. This involves allocating connected memory chunks (slabs) and linking them dynamically as the tape grows, reducing unnecessary reallocations.

Introducing Small Buffer Optimization (SBO) for short-lived tapes

Additionally, to further optimize performance for small-scale or short-lived tapes, we introduce a small buffer optimization (SBO) as part of the design. When this buffer overflows does the system transition to heap-allocated slabs.

Enhancing Performance Benchmarks

Enhance the existing performance benchmarks for the tape to check and test the tape structure changes made previously.

Implementation Plan

Making the tape thread-safe

In multithreaded scenarios, the tape would act as a globally shared resource accessed by multiple threads simultaneously. To make the tape thread-safe, integrate a locking mechanism

Implementing multi-layer storage

- To scale AD beyond memory limits, we can offload older or less frequently accessed portions of the tape to disk, like how operating systems page memory.
- This leads to the idea of a multilayer tape where recent entries stay in memory and older entries are paged to the disk.
- The idea is to treat the tape like a hybrid memory buffer similar to LRU caching where slabs are evicted to disk when memory exceeds a threshold.

Implementation Plan

(Stretch Goal) Supporting CPU–GPU memory transfers

Once the slab-based tape is implemented, where the tape is made up of connected slabs of memory instead of relying on frequent reallocations, it becomes significantly easier to manage contiguous memory chunks that are friendly for memory transfer to the GPU.

(Stretch Goal) Introducing checkpointing

- Checkpointing in the context of reverse mode automatic differentiation (AD) refers to the strategic storage of intermediate program states during the forward pass, so that during the reverse pass, one can recompute parts of the program efficiently without storing all intermediate values.
- To implement this, in the forward pass store only certain intermediate values at the checkpoints and during the reverse pass recompute the intermediate values from the nearest checkpoint.

Thank You!