Consolidate and Advance the GPU infrastructure in Clad

MidTerm Presentation

Vedant Goyal

Project Recap

Project Overview:

- This project aims to enhance clad by extending GPU support..
- By consolidating fragmented GPU support and adding missing features.
- Extend reverse-mode differentiation for GPU kernels.
- Benchmark against real-world HPC applications.
- Improve performance, scalability, and maintainability.

CUDA Shared Memory Support

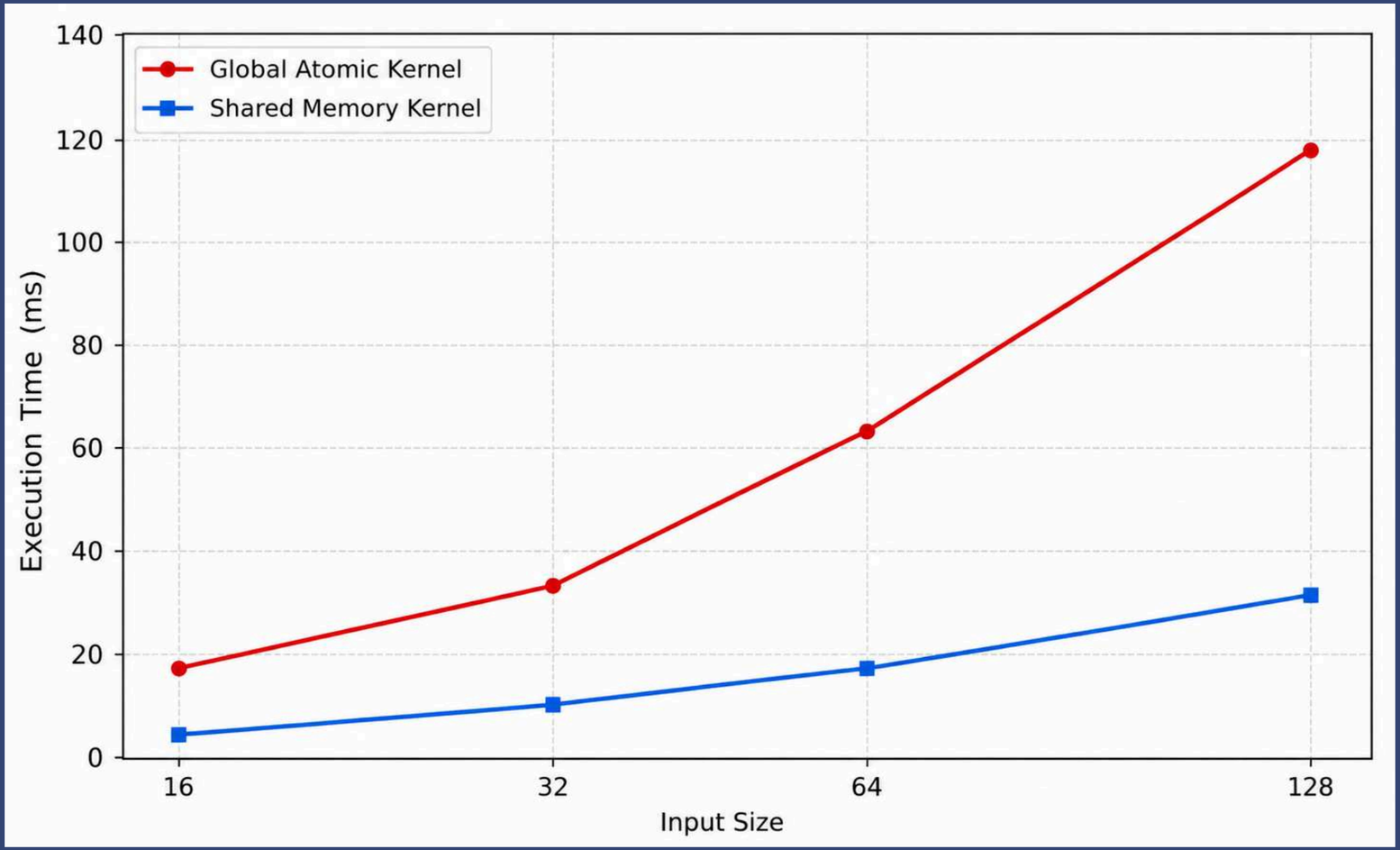
Added reverse-mode support for static shared memory (__shared__)

```
__global__ void sum_shared(int *input, int *output, int N)
    __shared__ int cache[256];

    int tid = threadIdx.x;
    int idx = blockIdx.x * blockDim.x + tid;
    int stride = blockDim.x * gridDim.x;

    int local_sum = 0;
    for (int i = idx; i < N; i += stride) {
        local_sum += input[i];
    }

    cache[tid] = local_sum;
    __syncthreads();

    for (int s = blockDim.x / 2; s > 0; s /= 2) {
        if (tid < s)
            cache[tid] += cache[tid + s];
        __syncthreads();
    }

    if (tid == 0)
        atomicAdd(address: output, val: cache[0]);
}
```

Added reverse-mode support for dynamic shared memory (extern __shared__)

```
__global__ void sum_extern_shared(int *input, int *output, int N) {
    extern __shared__ int cache[];

    int tid = threadIdx.x;
    int idx = blockIdx.x * blockDim.x + tid;
    int stride = blockDim.x * gridDim.x;

    int local_sum = 0;
    for (int i = idx; i < N; i += stride) {
        local_sum += input[i];
    }

    cache[tid] = local_sum;
    __syncthreads();

    for (int s = blockDim.x / 2; s > 0; s /= 2) {
        if (tid < s)
            cache[tid] += cache[tid + s];
        __syncthreads();
    }

    if (tid == 0)
        atomicAdd(address: output, val: cache[0]);
}
```

Enhanced Performance



Using Shared memory leads to nearly 3.6x lower execution time and improves memory bandwidth utilization

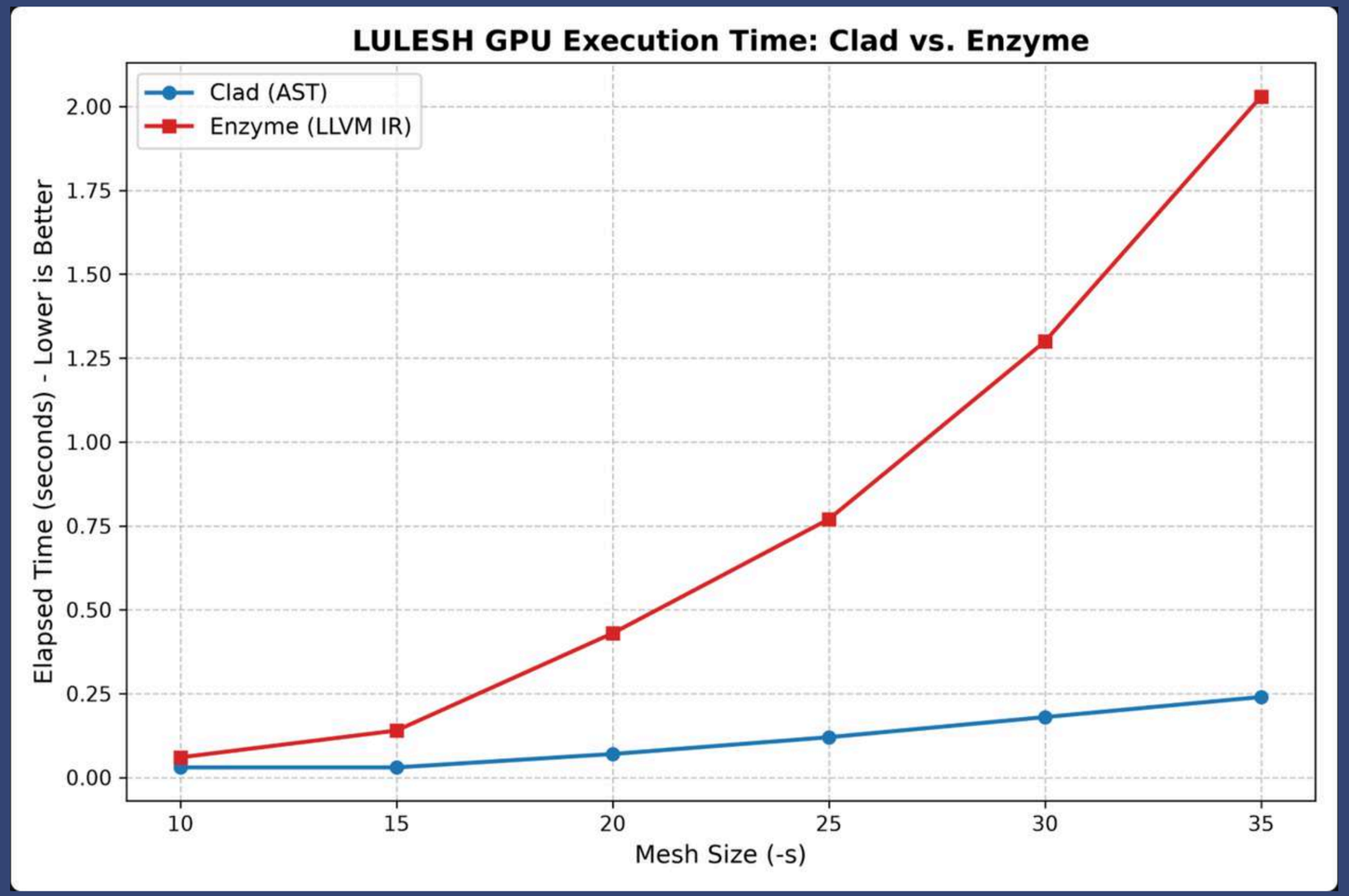
GPU Support for Restore Tracker

- Restore Tracker stores primal values during the forward sweep for use in the reverse sweep.
- Existing implementation relied on STL containers (std::map, std::vector), which are unsupported in CUDA device code.
- Replaced STL containers with fixed-size metadata and byte buffers.
- Buffer sizes are configurable with sensible default values.

```
#ifdef __CUDACC__
  …
  struct MetaData {
    char* addr;
    size_t size;
    size_t off;
  };
  MetaData m_meta[Max_Records];
  uint8_t m_buf[Max_Bytes];
  size_t m_cnt = 0, m_off = 0;
public:
  __host__ __device__ restore_tracker() = default;

  template <typename T> __host__ __device__ void store(const T& val) {
    for (size_t i = 0; i < m_cnt; ++i)
      if (m_meta[i].addr == (char*)&val)
        return;

    if (m_cnt >= Max_Records || m_off + sizeof(T) > Max_Bytes) {
      // Clad restore_tracker GPU capacity exceeded. Try again with larger value
      return;
    }
    m_meta[m_cnt] = {(char*)&val, sizeof(T), m_off};
    std::memcpy(m_buf + m_off, &val, sizeof(T));
    m_off += sizeof(T);
    m_cnt++;
  }
  __host__ __device__ void restore() {
    for (size_t i = 0; i < m_cnt; ++i)
      std::memcpy(m_meta[i].addr, m_buf + m_meta[i].off, m_meta[i].size);
    m_cnt = m_off = 0;
  }
```

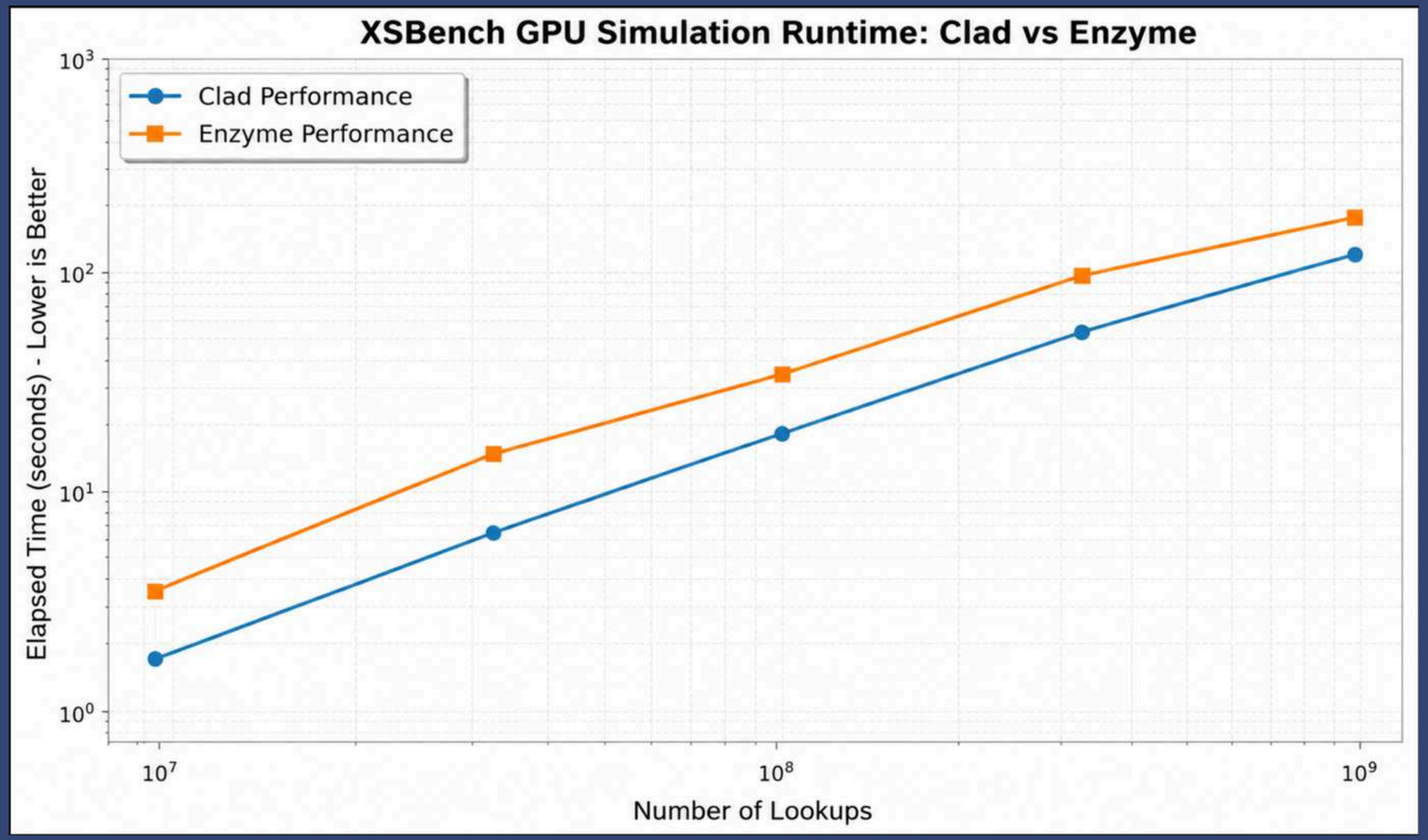
HPC Benchmarks

LULESH Benchmark: It is a proxy application used in HPC community to evaluate the performance of compilers, programming models, and hardware architectures.



- For grid size = 25: clad is 6.4x faster than enzyme
- For grid size = 35: clad is 8.6x faster than enzyme

XSBench Benchmark: It is another proxy application used in Monte Carlo Neutorns simulations



Clad is nearly 2x faster than Enzyme

Current Progress:

- Add LULESH Benchmark (PR #1841) - Merged
- Fixes bug in zero initialization of custom structs on GPU (PR #1862) - Merged
- Add XSBench Benchmark (PR #1878) - Merged
- Enable restore_tracker usage in device kernels (PR #1893) - Merged
- Add support for SourceLocExpr (PR #1929) - Merged
- Add reverse-mode support for CUDA shared memory attributes (PR #1826) - Open
- Optimizes RestoreTracker (PR #1821) - Open

Future Work:

- Add support for the remaining missing Thrust primitives.
- Implement activity analysis for CUDA device code.
- Integrate additional HPC benchmarks, including LBM and RSBench.
- Extend reverse-mode support for CUDA built-in functions and intrinsics.

THANK YOU!!

Any questions?