OpenCUDA-Lab

Project Title: OpenCUDA-Lab: Predictive and Fragmentation-Aware GPU Memory Runtime with DevOps Observability

Executive Summary

The **OpenCUDA-Lab** project addresses GPU memory (VRAM) inefficiency, unpredicted Out-Of-Memory (OOM) failures, and severe fragmentation in dynamic CUDA workloads. Existing solutions are bifurcated: deep learning frameworks contain complex internal allocators, while system tools provide hardware-level metrics without application-level predictive control.

This research proposes a lightweight, application-level C/CUDA runtime that unifies **peak memory prediction**, a **fragmentation-aware pool allocator with dynamic resizing**, and **DevOps-style observability**. The system targets resource-constrained educational and research environments, providing predictable memory execution and actionable batch-size guidance.

1. Problem Statement & Research Gap

1. **Unpredictable OOM Crashes:** Standard CUDA execution offers no pre-execution validation for peak VRAM demand under dynamic input shapes or changing batch sizes.
2. **Fragmentation in Dynamic Workloads:** Resizing runtime buffers via standard cudaMalloc/cudaFree causes high allocation overhead and severe memory fragmentation over multi-kernel pipelines.
3. **Disconnected Telemetry:** Infrastructure-level monitoring tools (e.g., Datadog, Grafana) track GPU usage globally at the driver layer, but cannot correlate memory spikes with application-level kernels or provide preemptive batch-size recommendations.

Domain / Prior Work	Focus	Limitations relative to OpenCUDA-Lab
Kernel Auto-Tuning *(e.g., Kernel Tuner)*	Execution time & launch configurations	Ignores VRAM consumption and fragmentation.

Framework Allocators *(e.g., PyTorch, GMLake)*	Internal tensor memory inside heavy AI engines	C/C++ CUDA apps cannot easily leverage them.
DevOps Observability *(e.g., Grafana, Azure AKS)*	System-level GPU utilization	Post-hoc metrics; lacks application control or pre-execution warnings.

2. Technical Architecture & Methodology

OpenCUDA-Lab will be implemented as a reusable C/CUDA shared library (.so) replacing standard runtime calls with ocl_malloc, ocl_realloc, and ocl_free.

System Modules

1. **Predictive Memory Estimator:**
    - Analyzes input shapes, grid/block dimensions, and kernel memory requirements statically/analytically prior to execution.
    - Generates early warnings if estimated peak usage exceeds available hardware VRAM limits.
2. **Fragmentation-Aware Pool Allocator:**
    - Pre-allocates and manages contiguous memory blocks on the GPU to bypass expensive driver-level calls.
    - Implements custom memory reuse policies and dynamic buffer resizing (ocl_realloc) to minimize external fragmentation.
3. **DevOps Observability Layer:**
    - Exports application telemetry in structured formats (JSON/CSV).
    - Exposes metrics including peak memory prediction vs. actual usage, fragmentation ratio, reallocation counts, and recommended batch sizes.

3. Workload Benchmark Suite

Evaluation will be conducted using two categories of CUDA workloads:

- **HPC System Kernels:** Vector operations, Dense Matrix Multiplication (GEMM), and parallel sorting algorithms.
- **Small AI Workloads:** Fundamental deep learning layers (Convolution, Multi-Head Attention blocks) operating under dynamic batch sizes.

4. Evaluation Metrics & Hypotheses

Metric	Target / Benchmark	Hypothesis
Prediction Accuracy	Mean Absolute Percentage Error (MAPE)	**H1:** MAPE < 15% between estimated and actual peak memory.
Effective Capacity	Maximum runnable batch size before OOM	**H2:** Pooled allocation enables larger input/batch sizes without OOM vs. cudaMalloc.
Runtime Overhead	Execution latency difference	**H3:** Memory management overhead < 5-10% of total execution time.
Operational Utility	Reduction in failed job runs	**H4:** Telemetry enables proactive batch sizing, avoiding manual trial-and-error OOM crashes.

5. Potential Research Output & Publication Plan

This project targets a complete system implementation and empirical evaluation suitable for publication in peer-reviewed HPC and systems venues (e.g., IEEE ICPADS, IEEE TPC, or Elsevier High Performance Computing proceedings).

- **Primary System Deliverables:** OpenCUDA-Lab C/CUDA Library (libopencuda.so), Automated Benchmarking Suite, and Telemetry Dashboard Exporter.
- **Core Paper Contribution:** An integrated, lightweight framework proving that application-level memory estimation combined with pooled dynamic reallocations significantly improves GPU operational reliability in resource-constrained lab environments.