Extending L2 Cache in GPUs for Improved CUDA Performance

Huy Nguyen

1. Introduction

Modern high-performance computing (HPC), artificial intelligence (AI), and deep learning workloads place heavy demands on GPU memory hierarchies. The L2 cache is especially critical in balancing memory bandwidth utilization and reducing off-chip memory latency in CUDA-based applications. However, current GPU architecture often features relatively small L2 caches (on the order of a few megabytes), which can lead to cache contention, increased miss rates, and performance bottlenecks—particularly for workloads with large working sets.

This proposal explores methods to extend and optimize the L2 cache in GPUs running CUDA workloads. By investigating architectural modifications, software-driven cache management techniques, and hybrid memory approaches (e.g., SRAM + eDRAM), we aim to enhance cache hit rates, reduce off-chip traffic, and improve overall efficiency in a range of CUDA applications. In particular, we will evaluate whether increasing L2 cache size provides more benefits than improving associativity or prefetching mechanisms, while also analyzing the area vs. performance tradeoff to determine the most effective and practical solution.

2. Problem Motivation, Research Tasks & Key Objectives

2.1 Problem Motivation

1. Memory Bandwidth Bottlenecks: As CUDA workloads grow in complexity (e.g., large-scale neural networks, HPC simulations, and ML benchmarks), the frequency of DRAM accesses can become a major bottleneck, hampering overall performance.
2. Limited L2 Cache Capacity: Conventional L2 caches in modern GPUs (4–8 MB) can be overwhelmed by data-intensive CUDA applications, resulting in excessive cache thrashing and contention.
3. Latency and Energy Overheads: High miss rates increase off-chip memory access, thereby raising both latency and energy consumption.
4. Tradeoffs in Cache Extensions: Although extending L2 cache can reduce DRAM traffic and boost performance, it can also increase power usage, area overhead, and cache coherence complexity. We will investigate whether larger L2 cache is more effective than improving associativity or prefetching, and determine the best balance between performance and area efficiency.

2.2 Research Tasks & Key Objectives

1. Investigate Architectural Modifications: Explore viable methods to expand L2 cache capacity—such as stacking SRAM or employing hybrid SRAM-eDRAM designs—to improve cache hit rates without incurring prohibitive power/area overheads.

2. Optimize L2 Cache for CUDA Workloads: Identify classes of CUDA applications (e.g., ML/AI, HPC simulations) that gain the most from larger L2 caches, and propose targeted optimizations.
3. Analytical Modeling: Develop or extend existing models to predict performance gains from L2 cache expansion while capturing tradeoffs in power, area, and coherence.
4. Benchmarking & Validation: Use GPGPU-Sim as the primary simulator to evaluate different L2 cache setups (size, associativity, prefetching) on CUDA-based workloads (Rodinia, PARSEC, ML benchmarks) to quantify performance, energy efficiency, and cost benefits.

3. Summary of Related Work / Papers

I plan to read more papers about this for more relevant information about GPU Cache:

- Traditional GPU Caching Strategies: NVIDIA and AMD GPUs feature shared L2 cache, but its small size limits its effectiveness for large workloads.
- L2 Cache Extension in CPUs: Prior research LLCs cache in CPUs using 3D-stacked SRAM and eDRAM-based cache hierarchies suggests potential for GPU applications.
- High-Bandwidth Memory (HBM) and Cache Coherence: Studies indicate that increasing L2 cache size can reduce dependence on HBM, improving energy efficiency and throughput.
- Custom L2 Cache Management Techniques: Prior work has explored software-driven cache management techniques for optimizing CUDA kernel memory access patterns.

4. Research Plan and Timeline

Week 1-2: Complete a literature review focusing on GPU cache architectures and existing optimizations, also setting up GPU

Week 3-4: Conduct baseline benchmarking by running CUDA workloads to profile L2 cache performance.

Week 5-6: Implement the proposed design by simulating extended or hybrid L2 cache configurations.

Week 7-8: Perform a performance evaluation on other benchmarks to analyze the impact of the extended cache designs.

Week 9-10: Introduce optimization strategies, including software-driven cache management techniques, collecting performance metrics and tradeoff.

Week 11-12: Finalize the analysis and write-up, summarize findings and preparing the final report.

5. Specific Research Tasks and Milestones

Weeks 1–2

- **Milestone**: Complete the literature review on GPU cache architectures and existing optimizations.
- **Task 1 (Analyze Existing L2 Cache Performance):**
  - Use **GPGPU-Sim** to examine baseline cache behavior.
  - Identify workloads with high miss rates or intense memory demands.

Weeks 3–4

- **Milestone**: Conduct baseline benchmarking for performance comparison.
- **Task 1 (continued):**
  - Finalize baseline performance metrics.
  - Compile a list of candidate workloads (Rodinia, PARSEC, ML/AI benchmarks, etc.).

Weeks 5–6

- **Milestone**: Implement and simulate proposed L2 cache extensions.
- **Task 2 (Design & Simulate Extended L2 Cache Architectures):**
  - Prototype hybrid SRAM-eDRAM cache models in **GPGPU-Sim to extend L2 cache**.
  - Collect performance, area, and power data for each proposed configuration.

Weeks 7–8

- **Milestone**: Evaluate performance gains of the extended cache design.
- **Task 2 (continued):**
  - Compare extended cache results with baseline.
  - Analyze tradeoffs in throughput, latency, and energy consumption.

Weeks 9–10

- **Milestone**: Develop software-driven optimization strategies.
- **Task 3 (Implement Software-Driven Cache Optimizations):**
  - Propose and code cache-management techniques that prioritize critical memory accesses.
  - Test initial optimizations with synthetic workloads for controlled validation.

Weeks 11–12

- **Milestone**: Finalize analysis and documentation.
- **Task 4 (Evaluate Performance & Energy Efficiency):**

    - Run extended benchmarks (Rodinia, PARSEC, ML/AI) on optimized designs.
    - Synthesize results into a comprehensive report, highlighting performance, energy, and area tradeoffs.

6. Expected Contributions & Novelty

- Determine if larger L2 cache is better than increasing associativity or prefetching.
- Use GPGPU-Sim as the main simulation tool for evaluating L2 cache extensions.
- Compare different L2 cache configurations and assess performance vs. area tradeoff.
- Validate improvements with CUDA benchmarks to quantify real-world performance gains.

This research aims to improve L2 cache efficiency in GPUs running CUDA workloads by proposing architectural extensions and software-based optimizations while carefully evaluating power, area, and coherence tradeoffs. The findings will contribute to a better understanding of GPU memory hierarchies and future GPU cache design strategies.