UCLA Samueli School of Engineering

Only Buffer When You Need To: Reducing On-chip GPU Traffic with Reconfigurable Local Atomic Buffers

Original Authors:

Preyesh Dalmia, et al.
UW Madison

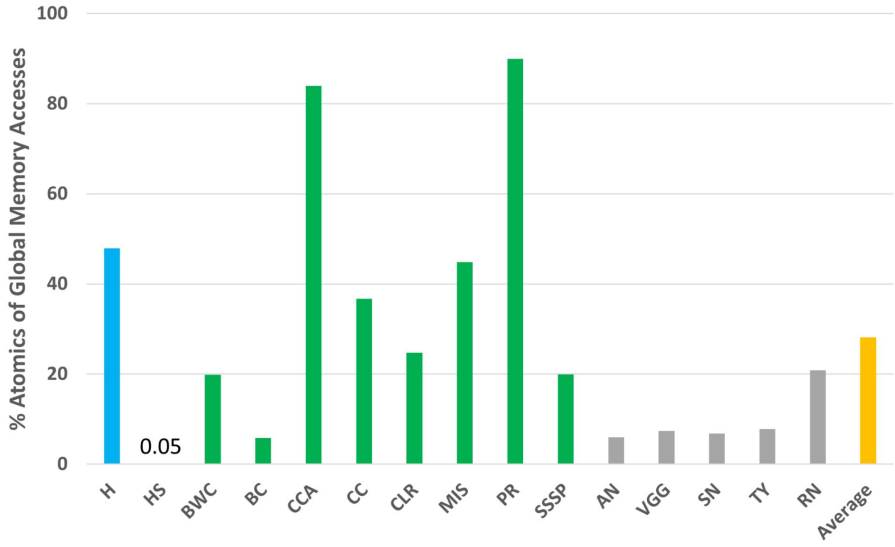
Presented By:

Parangat Mittal
UCLA

Introduction

Problem: Shared atomic updates on GPU are a memory bottleneck.

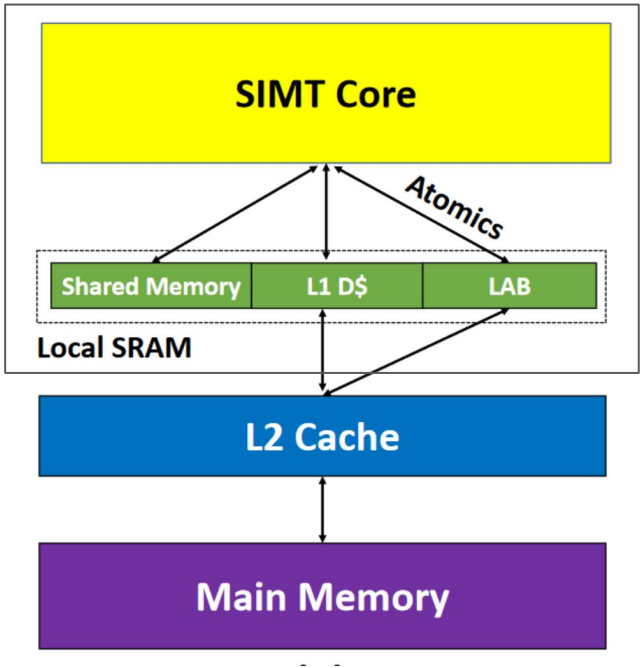
- GPUs are widely used as accelerators for memory-hungry ML Training and Graph Analytics.
- ML algorithms like **Stochastic Gradient Descent** (SGD) regularly update shared weights in the memory, using atomic operations for shared variables (30% of total memory accesses).
- Prior work optimized ML training by reducing memory accesses or keeping access limited to register files, but certain workloads still have atomic accesses as the overhead.



Contributions

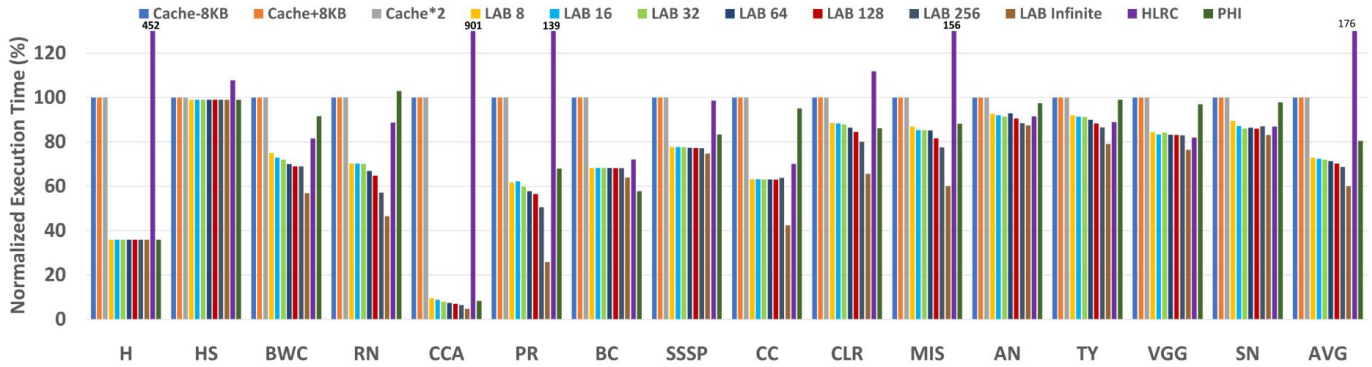
Solution: Cache partial commutative atomic updates.

- Updates to the memory are performed atomically to ensure correctness, but their order does not matter since the updated memory is accessed by the program only after all updates are completed or in subsequent layers.
- Hardware-software co-design approach to address this inefficiency:
  - **Hardware**: Buffer partial device-level atomic updates locally at each SM in a local atomic buffer (LAB), and send coalesced updates to shared L2 LLC later.
  - **Software**: Distinguish commutative atomic updates which can be buffered locally, through consistency model extension and memory ordering annotation.

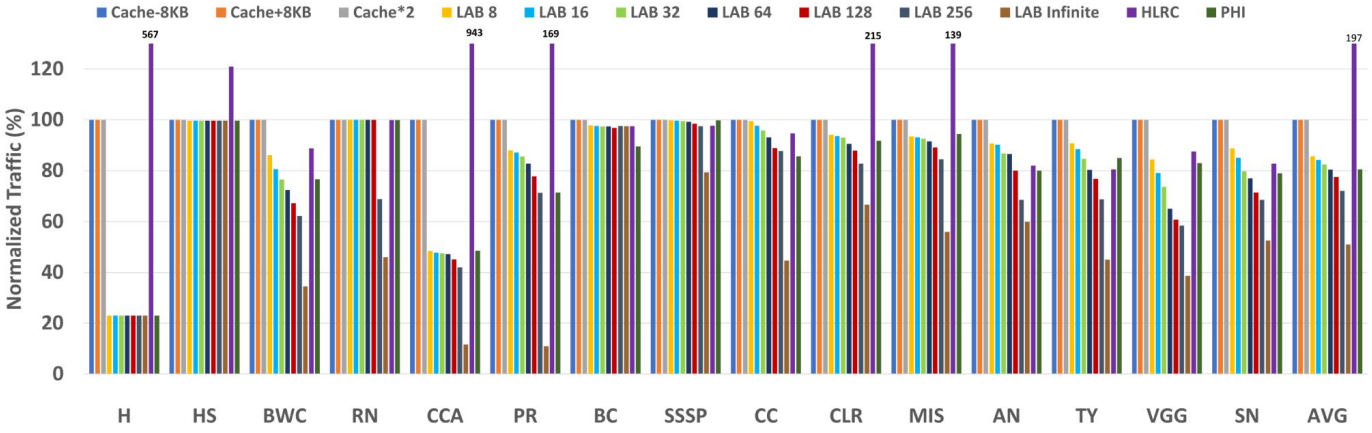


Results

LAB improves performance by **28%** on average across the evaluated graph analytics and ML training workloads.



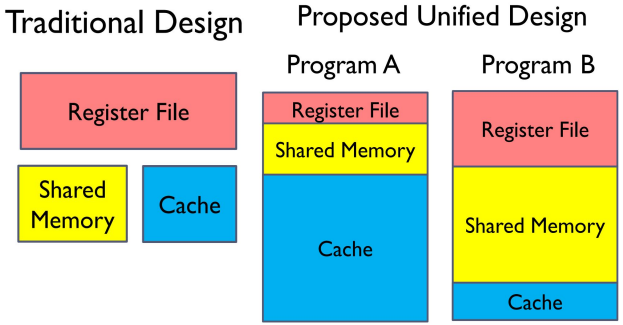
This reduces energy by a mean of **19%**, and reduces interconnect traffic by **19%** across the diverse workloads tested.

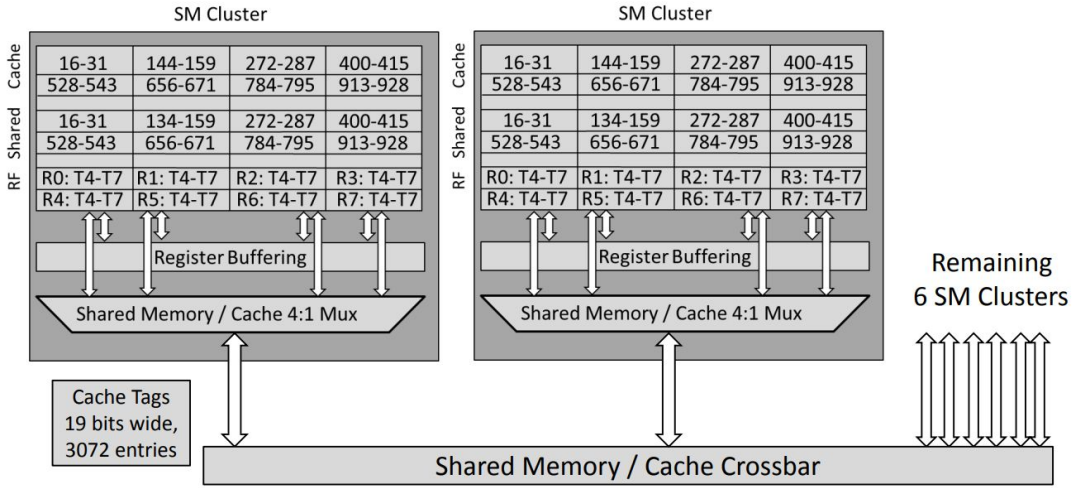


Unified Memory Architecture

Gebhart M., et al. 2012. Unifying Primary Cache, Scratch, and Register File Memories in a Throughput Processor. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-45)*.

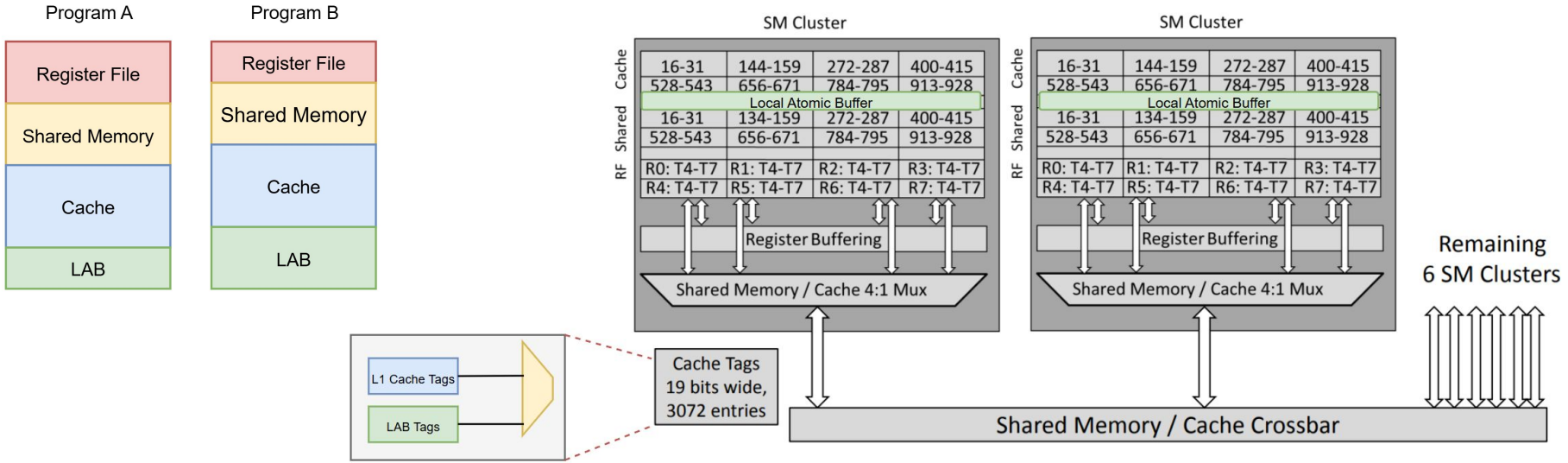
- This work extends the reconfigurable unified local memory proposed by Mark Gebhart, et al.* in MICRO-45 (2012).
  - Local SRAM on each SM cluster is dynamically partitioned into Register File, Shared Memory and L1 Cache, eliminating separate off-cluster fixed size shared memory and L1 cache.
  - All three memories access the crossbar connected across all SM clusters with a 4:1 MUX.





Local Atomic Buffer

- The local SRAM is partitioned into Register File, Shared Memory, L1 Data Cache and Local Atomic Buffer (LAB).
  - The 4th input of 4:1 MUX in each SM is now the LAB.
- Data array holds the partial values for a given address. LAB utilizes a portion of the cache tag array to manage metadata such as address, atomic function information.



LAB Operation

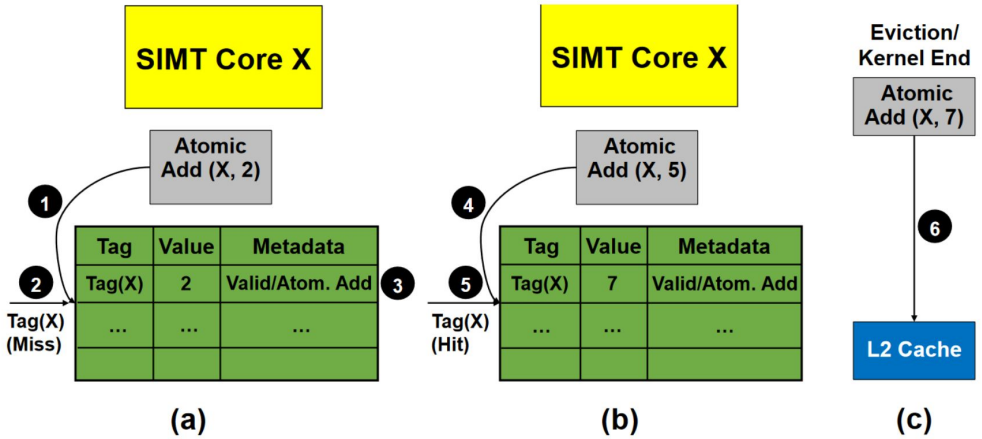
Entry: GPU Coalescer combines the requests from threads within the same wrap before sending it to LAB.

Eviction: When LAB is full, entries are evicted as per LRU, and evicted atomic request is executed, i.e. the respective memory location is updated.

Divergence: When uncoalesced memory accesses occur where each thread access unique cache line, each request is treated as unique instead of combining it into one.

Example:

1. AtomicAdd(X,2) is issued.
2. Tag lookup occurs, cache miss happens.
3. New entry is allocated in cache
4. AtomicAdd(X,5) to same address issued.
5. Tag lookup occurs, entry matches.
6. Partial value updates from 2 to 7.
7. Atomic update sent out, memory updated.



LAB Operation

Aspect	Baseline GPU	LAB-enabled GPU
Location of Atomic Updates	Centralized at L2 cache	Decentralized at SM level (LAB)
Local Memory Usage	Unified local SRAM partitioned between L1 cache and shared memory	Unified local SRAM partitioned between L1 cache, shared memory, and LAB
Atomic Serialization	High serialization at L2 cache	Reduced serialization due to local buffering
Traffic to L2 Cache	High: Every atomic update sent to L2	Lower: Only coalesced updates sent to L2
Energy Efficiency	Lower: Frequent interconnect traffic and queuing delays	Higher: Reduced traffic and decentralized updates
Hardware Overhead	None	Minimal: Reuse of local SRAM and additional metadata
Scalability	Limited by L2 bandwidth and contention	Scalable due to localized processing at each SM

Q&A