GPU Atomic Performance Modeling with Microbenchmarks

Devon McKee, UC Santa Cruz Concurrency and Heterogeneous Programming Lab
CPU ATOMIC RMWS

thread 0
atomic_fetch_add(a, ...)

Thread 0 attempts atomic operation on a and locks cache line

(locked)
Thread 1 attempts atomic operation on a and has to wait
CPU ATOMIC RMWS

thread 0

atomic_fetch_add(a, ...)

Thread 0 finishes atomic operation and releases lock

thread 1

atomic_fetch_add(a, ...)

[Diagram showing the flow of atomic operations and the release of a lock]
CPU ATOMIC RMWS

<table>
<thead>
<tr>
<th>thread 0</th>
<th>thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic_fetch_add(a, ...)</td>
<td>atomic_fetch_add(a, ...)</td>
</tr>
</tbody>
</table>

Thread 1 performs atomic operation on location a
CPU ATOMIC RMWS

• CPU atomic read-modify-write instructions (RMWs) have a well-defined performance profile
CPU ATOMIC RMWS

• CPU atomic read-modify-write instructions (RMWs) have a well-defined performance profile
• Threads needing access prefer to lock cache lines
CPU ATOMIC RMWS

- CPU atomic read-modify-write instructions (RMWs) have a well-defined performance profile.
- Threads needing access prefer to lock cache lines.
- RMW throughput is hurt by threads contending for same location.

Every 4 threads accessing each location, no padding:
40.89 atomic ops/microsecond

Contention: 4
Padding: 1
CPU ATOMIC RMWS

- CPU atomic read-modify-write instructions (RMWs) have a well-defined performance profile
- Threads needing access prefer to lock cache lines
- RMW throughput is hurt by threads contending for same location
- RMW throughput is helped by locations padded out to cache line size
CPU ATOMIC RMWS

- CPU atomic read-modify-write instructions (RMWs) have a well-defined performance profile.
- Threads needing access prefer to lock cache lines.
- RMW throughput is hurt by threads contending for same location.
- RMW throughput is helped by locations padded out to cache line size.

Each thread accessing one atomic location, no padding:
73.14 atomic ops/microsecond

Every 4 threads accessing each location, no padding:
40.89 atomic ops/microsecond

Each thread accessing one atomic location, padded to cache line size:
387.74 atomic ops/microsecond

Live demo of CPU atomics: devon.engineering/epiphron-web
CPU ATOMIC RMWS

Live demo of CPU atomics: devon.engineering/epiphron-web
CPU ATOMIC RMWS

Live demo of CPU atomics: devon.engineering/epiphron-web
CPU ATOMIC RMWS

Live demo of CPU atomics: devon.engineering/epiphron-web
## CPU ATOMIC RMWS

### Diagram

A heat map showing atomic operations per microsecond with varying contention and padding levels.

### Table

<table>
<thead>
<tr>
<th>Contention</th>
<th>Padding</th>
<th>Operations per Microsecond</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>65.56</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>43.69</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>41.82</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>108.45</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>41.82</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>40.49</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>41.82</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>42.77</td>
</tr>
</tbody>
</table>

### Live Demo

Live demo of CPU atomics: [devon.engineering/epiphron-web](http://devon.engineering/epiphron-web)
GPU SYNCHRONIZATION

• GPU compute becoming increasingly critical
• GPUs provide a high degree of concurrency
• Heavily concurrent algorithms require synchronization
• GPU synchronization comes in the form of two major primitives:
  • Atomic instructions
  • Barriers
• GPU atomic performance varies greatly across devices
• GPU atomics sensitive to several performance characteristics
  • Contention
  • Padding
  • Thread access patterns
VULKAN MICROBENCHMARKING SUITE

- Contiguous access: chunking threads within warps
- Cross-warp: striding threads across warps
- Random access: distributing threads to atomic locations randomly*
Microbenchmarks currently targeting atomic read-modify-write (RMW) instructions

Written in Vulkan for cross-platform testing

C++ application, currently being rewritten for Android testing

Uses OpenCL kernels, transpiled to SPIR-V using clspv
VULKAN MICROBENCHMARKING SUITE

- Benchmarks sweep values for contention and padding
- Separate benchmarks for each access pattern
- Heatmaps are produced to show trends in throughput
VULKAN MICROBENCHMARKING SUITE

GPU

CPU

GPU ATOMIC PERFORMANCE MODELING WITH MICROBENCHMARKS
Intel discrete card results (contiguous access):

- Peak throughput at contention = 1, padding = 1
- Throughput degrades down to less than 5% of peak throughput in some scenarios
AMD discrete card results (contiguous access):

- Peak throughput at contention = 4, padding = 1
- Throughput degrades away all the way down to ~1% of peak
MICROBENCHMARK RESULTS

NVIDIA discrete card results (contiguous access):
• Peak throughput past contention = 32
• Slower region throughput is ~4% of peak region
MICROBENCHMARK RESULTS

2024

GPU ATOMIC PERFORMANCE MODELING WITH MICROBENCHMARKS
Random access microbenchmark:

- Distributes GPU threads randomly across array of atomic integers
- Varies size of array to distribute threads more
MICROBENCHMARK RESULTS

AMD discrete card results (random access):
• Throughput reaches local peak at 16 atomics, increases past 64 atomics
• With lower number of atomics needed, bit packing until 16 atomics used greatly improves throughput
MICROBENCHMARK RESULTS

NVIDIA discrete card results (random access):

• Throughput falls immediately upon number of atomics increasing

• If application can reduce number of atomic locations used to 1, throughput is greatly improved

![Graph showing throughput vs. number of atomics](image-url)
How do we characterize these results?

Can atomic performance profiles be reduced to conditions to give greater throughput?
• Working on creating performance model to describe atomic behavior
• Basic model involves some access pattern and conditions for coalescing
• Complexities of some results make it hard to identify conditions
Contiguous distribution of threads with chunks of 64 threads accessing same atomic location

At least 32 threads contending on the same location

Atoms are executed at below peak throughput

Atoms are executed at near peak throughput
ATOMIC PERFORMANCE MODEL

Warp strided distribution of threads across a number of atomic locations

Conditions for atomic coalescing/speedup

Some function of peak throughput

AMD Radeon RX 7900 XT
workgroup size: 1024
workgroups: 108

cross_warp: atomic_fa_relaxed

Atomic Operations per Microsecond

2024

GPU ATOMIC PERFORMANCE MODELING WITH MICROBENCHMARKS
SUMMARY OF CONTRIBUTIONS

- Study of relative atomic performance across GPUs
- Microbenchmark suite to be executed on any device supporting Vulkan
- General performance model to capture complexities of GPU atomic performance
CONCLUSION

How do these results help developers optimize applications?

Performance model allows developers to structure atomic RMWs as a device-specific optimization.

Looking for opportunities to include these results as device-specific information in Vulkan specification.
FUTURE DIRECTIONS

• Modification of atomic implementation in GPGPU simulator to examine effect on applications
• Expanded testing on new devices, including mobile platforms and Apple devices (through MoltenVK)
• Application of methodology to other synchronization primitives (namely, barriers)
THANK YOU

Devon McKee
dlmckee@ucsc.edu

Tyler Sorensen, Primary Investigator

Ishita Chaturvedi, Project Collaborator

Gurpreet Dhillon, Project Collaborator

Sean Siddens, Project Collaborator