Energy Efficient GPU Transactional Memory via Space-Time Optimizations

Wilson W. L. Fung
Tor M. Aamodt
Why TM for GPU?

- Simple Irregular Parallelism on GPU

<table>
<thead>
<tr>
<th></th>
<th>Regular</th>
<th>Irregular</th>
</tr>
</thead>
<tbody>
<tr>
<td>nBody</td>
<td>1640s</td>
<td>5.2s</td>
</tr>
<tr>
<td>5M Bodies</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Other Applications?
Why TM for GPU?

- Predictable Dev Time
- No Deadlock!
- Maintainable Code

TM on GPU
TM for GPU: Energy Overhead

- TM = Speculative Execution =

- **Kilo TM**: First Hardware TM for GPU
  - Simple Design for Scalability
  - 1000s of Concurrent Transactions
  - Scalar Transaction Management
  - Value-Based Conflict Detection
Space

Time

65% Speedup

2X $\rightarrow$ 1.3X Energy Usage
Background: Kilo TM

• 1000s of Concurrent Transactions
  – Value-based conflict detection: Global Metadata

• Special HW to boost validation and commit parallelism
Kilo TM Implementation

- SIMT Core:
  - TM-Aware SIMT Stack
  - L1 Data Cache
  - TX-Log Unit

- Memory Partition:
  - Commit Unit
  - L2 Cache
  - DRAM

Commit Protocol

Efficiency Concerns

128X
Speedup over CG-Locks

40%
FG-Locks Performance

2X
Energy Usage

• Scalar Transaction Management
  – Scalar Transaction fits SIMT Model
  – Simple Design
  – Poor Use of SIMD Memory Subsystem

• Rereading every memory location
  – Memory access takes energy
Inefficiency from Scalar Transaction Management

• Kilo TM ignores GPU thread hierarchy
  – Excessive Control Message Traffic

– Scalar Validation and Commit  
  → Poor L2 Bandwidth Utilization

• Simplify HW Design, but **Cost Energy**
Warp Level Transaction Management

• Key Idea:
  – Manage transactions within a warp as a whole

• Enables optimizations that exploit spatial locality:
  – Aggregate Control Messages
  – Validation and Commit Coalescing

• Challenge: Intra-Warp Conflicts
Warp Level Transaction Management: Aggregate Control Messages

Scalar Messages
12 Messages

Aggregated Messages
3 Messages

Contributes up to 40% of Interconnection Traffic

Wilson Fung

11
Warp Level Transaction Management: Validation and Commit Coalescing

Without Coalescing

Max Utility = 4/32 = 12.5%

Reduce 40% of Requests to L2 Cache

With Coalescing

Global Memory (L2 cache/DRAM)

Global Memory (L2 cache/DRAM)
Intra-Warp Conflict

• Potential existence of intra-warp conflict introduces complex corner cases:

- Read Set: X=9, Y=8, Z=7, W=6
- Write Set: Y=9, Z=8, W=7, X=6

@ Validation:
- Global Memory: X=9, Y=8, Z=7, W=6

All Committed (Wrong):
- Global Memory: X=6, Y=8, Z=8, W=6

Correct Outcomes:
- Global Memory: X=6, Y=9, Z=8, W=6
- Global Memory: X=9, Y=9, Z=7, W=7

Wilson Fung
Intra-Warp Conflict Resolution

- Kilo TM stores read-set and write-set in logs
  - Compact, fits in caches
  - Inefficient for search
- Naive, pair-wise resolution too slow
  - T threads/warp, R+W words/thread
  - \( O(T^2 \times (R+W)^2) \), \( T \geq 32 \)
Intra-Warp Conflict Resolution: 2-Phase Parallel Conflict Resolution

• Insight: Fixed priority for conflict resolution enables parallel resolution
• O(R+W)
• Two Phases
  – Ownership Table Construction
  – Parallel Match
Intra-Warp Conflict Resolution: 2-Phase Parallel Conflict Resolution

- Insight: Fixed priority for conflict resolution enables parallel resolution
**Intra-Warp Conflict Resolution:**

2-Phase Parallel Conflict Resolution

- **Insight:** Fixed priority for conflict resolution enables parallel resolution

---

### Ownership Table Construction

<table>
<thead>
<tr>
<th>TX1</th>
<th>TX2</th>
<th>TX3</th>
<th>TX4</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLog</td>
<td>WLog</td>
<td>WLog</td>
<td>WLog</td>
</tr>
</tbody>
</table>

**Ownership Table:** \( O(W) \)

### Parallel Match

**Read-Log:**

Owner ID < My ID \( \rightarrow \) Abort (E.g. Owner ID = 2 \( \rightarrow \) Abort)

**Write-Log:**

Owner ID != My ID \( \rightarrow \) Abort (E.g. Owner ID = 3 \( \rightarrow \) Pass)

**Ownership Table:** \( O(R+W) \)
Warp Level Transaction Management Made Practical

• Enables optimizations that exploit spatial locality:
  – Aggregate Control Messages
  – Validation and Commit Coalescing

• **Challenge:** Intra-Warp Conflicts
Temporal Conflict Detection

• Motivation:
  Skip value-based conflict detection for conflict-free read-only transactions

\[ \text{TX1} \]
if (C == 0)
\[ \text{B = B + 1;} \]

\[ \text{TX2} \]
int K;
\[ K = X + Y; \]

– 40% and 85% of the transactions in two of our workloads.
Temporal Conflict Detection

- If \( \text{LastWrittenTime}(X) < \text{StartTime} \), Pass
- Otherwise, Conflict Detected
Temporal Conflict Detection

TX1
LD [A];
LD [B];

ST [A]       TX1 starts
Life Time of [A] loaded by TX1

ST [A]       TX1 LD [A]

ST [B]       TX1 LD [B]
Life Time of [B] loaded by TX1

Effective instantaneous execution time for TX1 w.r.t. other threads
Temporal Conflict Detection

Value loaded by LD [A] and value loaded by LD [B] cannot coexist at any point of time – a detected conflict.
Temporal Conflict Detection Implementation

16kB Recency Bloom Filter
- Approximate but Conservative
- Aliasing two very old store is OK
Evaluation

• GPGPU-Sim 3.2.1
  – Detailed: IPC Correlation of 0.90 vs. Fermi GPU

• Model Energy Overhead of Kilo TM
  – Extra Hardware
    • CACTI for access energy of major SRAM arrays
  – Extra Activity via GPUWattch
  – Increased Execution Time (More Leakage)

• GPU TM Applications

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HT-[H/M/L]</td>
<td>Hash Table Construction</td>
<td>ATM – Bank Transactions</td>
</tr>
<tr>
<td>BH-[H/L]</td>
<td>Barnes Huts (N-Body)</td>
<td>CL/CLto – Cloth Simulation</td>
</tr>
<tr>
<td>CC</td>
<td>Maxflow/Mincut Graph</td>
<td>AP – Data Mining</td>
</tr>
</tbody>
</table>
40% → 66% FG-Lock Performance

2X → 1.3X Energy Usage

Low Contention Workload: Kilo TM w/ SW Optimizations on par with FG Lock
Summary

• Two Enhancements for Kilo TM
  – Warp Level Transaction Management
    • Exploit Spatial Locality in Thread Hierarchy
  – Temporal Conflict Detection
    • Silent Commit of Read-Only Transaction

• Reduce Performance and Energy Overhead of Kilo TM

• Low Contention Workload: Kilo TM w/ Optimizations on par with FG Lock
BACKUP SLIDES
Normalized Performance

40% → 66%

FG-Locks Performance

Low Contention Workload:
Kilo TM w/ SW Optimizations on par with FG Lock
Normalized Energy Usage

Energy Usage Normalized to FGLock

- Core
- L1Cache
- SMem
- NOC
- L2Cache
- DRAM
- KiloTM
- Idle
- Leakage

Intra-Warp Conflict Resolution: 2-Phase Parallel Conflict Resolution
2PCR vs. SCR

- KiloTM-Base
- WarpTM+2PCR
- WarpTM+2PCR(NoOverhead)
- WarpTM+SCR
- WarpTM+SCR(NoOverhead)

Exec. Time Normalized to FGLock

- HT-H
- HT-M
- HT-L
- ATM
- CL
- CLto
- BH-H
- BH-L
- CC
- AP
Spatial Locality among Transactions

![Spatial Locality among Transactions](image)

Wilson Fung


32
ABA Problem?

• Classic Example: Linked List Based Stack

```
while (true) {
    t = top;
    Next = t->Next;
    // thread 2: pop A, pop B, push A
    if (atomicCAS(&top, t, next) == t) break;  // succeeds!
}
```

Wilson Fung
ABA Problem?

• atomicCAS protects only a single word
  – Only part of the data structure

```
while (true) {
    t = top;
    Next = t->Next;
    if (atomicCAS(&top, t, next) == t) break;  // succeeds!
}
```

• Value-based conflict detection protects all relevant parts of the data structure
ABA Problem?

• If every memory input value is identical, the transaction code should generate the same output.
  – No point to re-execute transaction for ABA event.

![Transaction Code Diagram]