**KILO Transactional Memory for GPU**

Wilson W. L. Fung∗
wwlfung@ece.ubc.ca

Inderpeet Singh∗
isingh@ece.ubc.ca

Andrew Brownsworth
andrew@brownsword.ca

Tor M. Aamodt∗
aamodt@ece.ubc.ca

“University of British Columbia, Canada

---

**Transactional Memory on GPU?**

- **Motivation:** Explore irregular parallelism on GPUs
- **Enable more work-efficient algorithms + improve programmer productivity**

- **Coarse-Grained / No Synchronization**
  - **Functionality**
  - **Performance**

- **SW Development Time**
  - **300X Speedup**

- **Fine-Grained Locking**
  - **N-Body with 5M Bodies CUDA SDK:** O(n^2) – 1640 s (Barrier)

---

**Control Flow Divergence**

- Transaction aborts may cause a warp to diverge

**Scalable Conflict Detection:**

- 1000s of Concurrent Transactions
- 1000 x 1000 parallel address comparison – too expensive?
- No cache coherency protocol on GPU

**Version Management:**

- Storage Problem
- Checkpointing register file for 1000s of threads is not cheap
- No caches for write buffering

**Commit Bottleneck:**

- Potentially serializing all transaction
- Needs to allow non-conflicting transaction to commit in parallel

---

**Transaction-Aware SIMT Stack**

- **SIMT stack automatically serializes a warp @ CF divergence**
- **Special entries for TM to handle transaction aborts:**

- **@tx_begin:**
  - **Global Transaction:**
    - TM Version: atomic
    - Lock Version: Lock([A[s], A[t]];
      Unlock([A[s], A[t]]);

- **@tx_commit:**
  - **Thread 6 & 7 failed validation:**
    - @tx_commit, restart Tx for thread 6 & 7:

- **Branch Divergence within TX:**

---

**Performance:**

- **59% of FG-Locking**
- Modeled KILO TM on GPGPU-Sim v3.0
- Evaluated on TM-enhanced CUDA and OpenCL applications

---

**References**


See Our Full Paper for More Details:


---

**KILO Transactional Memory**

- Support 1000s of concurrent transactions on a GPU
- Support Unbounded Transaction ⊑ Limited by local memory size

**Key Ideas**

- **Only Detect Existence of Conflict (Not Identity)**
  - Check conflict with committed transactions

**Value-Based Conflict Detection** [2]: Committed TX in Memory

- Eliminate storage problem for 1000s of concurrent TX

**Self-Validation @ Commit**

- Value-Based Conflict Detection
- Invalidation @ Commit

- (Most CPU-Based HTM Systems)

---

**Other Design Choices**

- Minimalistic: Weak Isolation + Flattened Nested Transactions
- Software Register Rollback – Rarely Needed
- Linear Memory Write Logs in Local Memory – Rarely Searched

---

**Logical Commit Stages**

- TX Log Unit: Log Generation and Transmission
- Commit Unit: Parallel Validation and Commit Pipeline

- TX Log @tx_resume: Log transactions in Commit Order
- TX Log @tx_begin: Log transactions in Commit Order

---

**Commit Traffic**

- Read-Set + Write-Set
- Validated/Invalidated
- DRAM
- GDDR Memory

---

**Commit Unit**

- Younger Transactions
- RxLocal Outcome
- TX Rtx Outcome
- Communication with SIMT Cores

---

**References**


See Our Full Paper for More Details: