Efficient GPU Hardware Transactional Memory through Early Conflict Resolution

Sui Chen and Lu Peng
Division of Electrical and Computer Engineering, Louisiana State University

Accepted & presented at the 22th IEEE Symposium on High Performance Computer Architecture (HPCA), Barcelona, Spain

Motivation and Goal

Goal: Conflict ⇓ ⇒ Performance ⇑ Energy ⇓

Parallel programs with critical sections

Thread 1 Read/Write | Thread 2 Read/Write | ... GPU: massively parallel

Coarse-grained locks

+ easy to use
- slow (serializes everything)

Fine-grained locks

+ fast
- hard to program
Example: RBTree
- error-prone

Transactional Memory

atomic

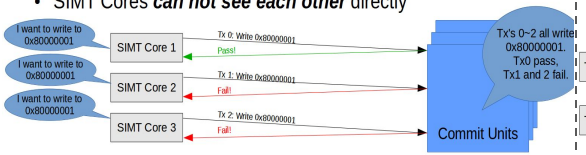
+ easy to use
+ good performance

TM: Get the best of both coarse- and fine- locks

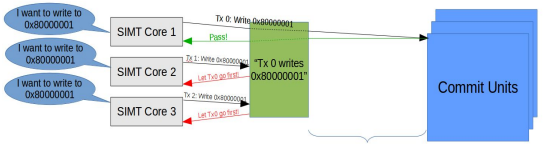
Opportunity for Speedup

Early abort

- SIMT Cores ***can not see each other*** directly



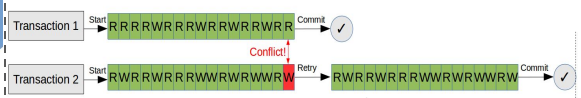
- What if they can?



- Detect Spatial 2 3 conflicts in SIMT Cores
- Reduces contention in Commit Units

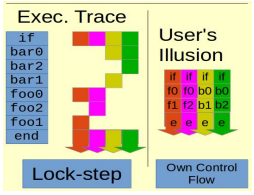
Pause-and-Go

- Aborting and retrying may be expensive
  - One small conflict wastes all transactional work

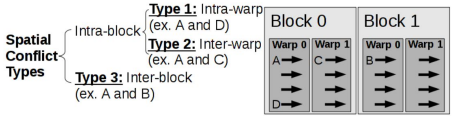


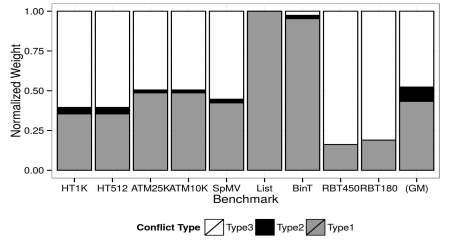
- If conflict can be avoided ...

GPU-Specific Architecture & Conflict Types



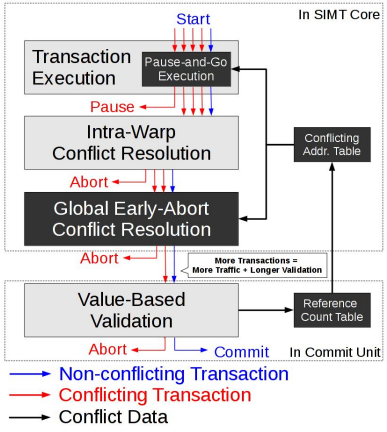
SIMT execution gives 3 spatial types of conflicts

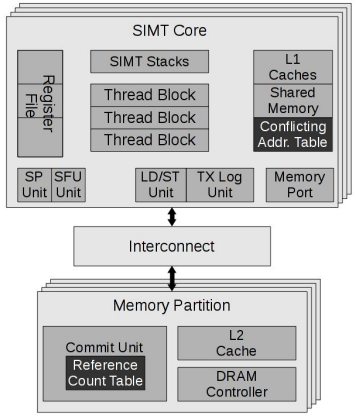




Spatial types 1 and 3 frequent in workloads

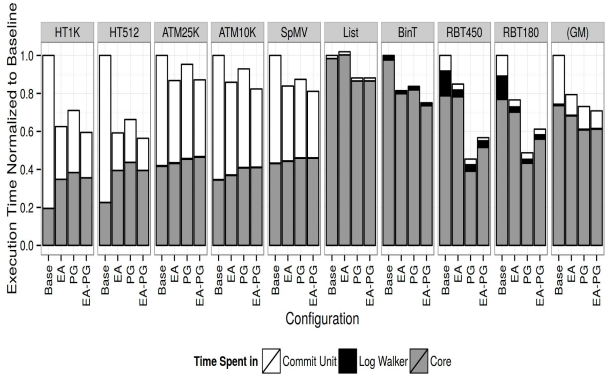
Interaction with Hardware





- One set of hardware change facilitates both approaches
- Modified transaction execution flow

Results



- 1.41x speedup
- 0.8x energy consumption
- Table size chosen using sensitivity study
- 5 workloads are CU-heavy; the rest are SIMT Core-heavy