Towards Shared Memory Consistency Models for GPUs

Tyler Sorensen, Ganesh Gopalakrishnan
University of Utah School of Computing

Vinod Grover
NVIDIA

Research Supported by NSF OCI 1148127

Abstract

With the widespread use of GPUs, it is important to ensure that programmers have a clear understanding of their shared memory consistency model i.e. what values can be read when issued concurrently with writes. GPUs present very different memory and concurrency systems from traditional CPUs and have not been the subject of any published study we know yet. We propose a collection of ***litmus tests*** that illustrate interesting visibility and ordering properties. We establish a model using intuitive data structures and implement our model in the Murphi modeling language. As a preliminary study, we restrict our model to Load (`Ld`), Store (`St`), Thread Fence (`TF`) and Thread Fence Block (`TFB`) instructions across global and shared memory.

Motivation

Lack of Clarity on Fences

Memory fences can have many different properties, including static (ordering within a thread) or dynamic (memory visibility to other threads)[2] properties. Fences can also be cumulative[3] which requires ensuring visibility of values the calling thread did not author. The current CUDA documentation does not mention any static or cumulative properties at all.

Performance Increase

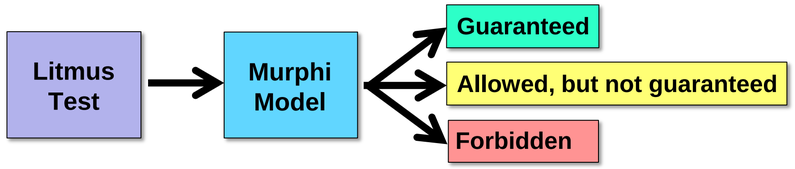
Well-synchronized code hides weak memory model issues from the programmer; however, synchronization operations can be expensive. Cutting edge algorithms bypass these expensive operations to obtain significant increases in performance[1] at the cost of exposing the memory model, which **must** be understood for correct implementation.

Looming Bugs

If developers write unsynchronized code assuming certain instruction orderings or memory visibilities which are not in line with the memory consistency model, then their code is buggy. We have found instances of such assumptions in un-synchronized real world code, instances in real-world linear algebra libraries, and graph traversal algorithms.

Murphi Implementation

We implemented our model in the Murphi modeling language. The model is available at:
http://www.cs.utah.edu/~tylers/CUMM.
It is easily modifiable to run custom litmus tests and verify assertions regarding the test. A flowchart of the process and possible outcomes is shown below:



Litmus Tests and Results

✗ **GPUs prohibit. Our model is faithful.**
✓ **GPUs allow. Our model is faithful.**
? **GPUs unknown. Our model is programmable.**

Relaxed Coherence ✓

```
T0 : St(a,1); ... ;
T1 : St(a,2); ... ;
T2 : Ld(a,1); TFB; Ld(a,2)
T3 : Ld(a,2); TFB; Ld(a,1)
```

Classical Coherence[3] ✗

```
T0 : St(a,1); St(a,2)
T1 : Ld(a,1); Ld(a,2)
T2 : Ld(a,2); Ld(a,1)
T3 : ... ;
```

Simple Global/Shared Visibility across threads

✗
```
T0 : St(a,1) ; St(b,2) ; ......... ;
T1 : Ld(b,2) ; Ld(a,1) ; ......... ;
```

✓
```
//T0 and T1 in the same block
T0 : St(a,1) ; TFB ; St(b,2) ; ... ;
T1 : Ld(b,2) ; TFB ; Ld(a,1) ; ... ;
```

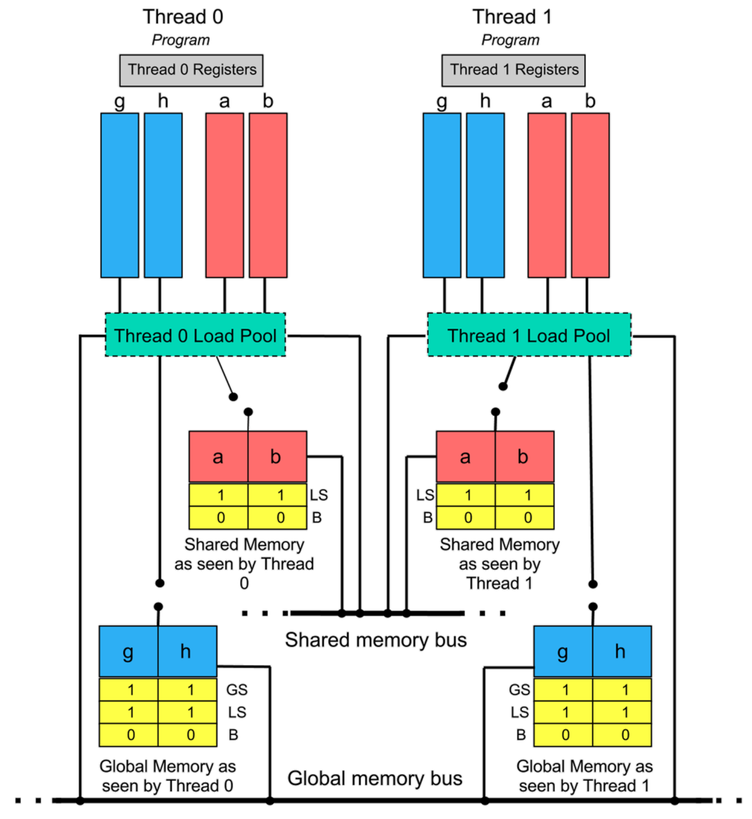
✗
```
//T0 and T1 not in the same block
T0 : St(a,1) ; TFB ; St(b,2) ; ... ;
T1 : Ld(b,2) ; TFB ; Ld(a,1) ; ... ;
```

✓
```
T0 : St(a,1) ; TF ; St(b,2) ; .... ;
T1 : Ld(b,2) ; TF ; Ld(a,1) ; .... ;
```

Write Atomicity Relaxation

?
```
T0 : St(a,1) ; ... ; ....... ;
T1 : St(b,2) ; ... ; ....... ;
T2 : Ld(a,1) ; TFB ; Ld(b,0) ;
T3 : Ld(b,2) ; TFB ; Ld(a,0) ;
```
?

Model



- **Each thread has its own view of registers, an instruction queue for each address and view of global and shared memory.**
- **Load pool allows relaxed coherence.**
- **Memory is shared and borrowed based on flags.**

Conclusions/References

This model is currently being reviewed by industry experts and is expected to grow and change based on feedback. Future work includes: a more complete treatment of PTX, a level-language model for CUDA, an axiomatic model with an equivalence proof and a contrast with observable behavior on GPUs.

1. Nhat Minh Lê, Antoniu Pop, Albert Cohen, and Francesco Zappa Nardelli. “Correct and efficient work-stealing for weak memory models.” In *Symp. on Principles and Practice of Parallel Programming (PPoPP)*, Shenzhen, China, February 2013.
2. Jade Alglave, Luc Maranget, Susmit Sarkar, and Peter Sewell. "Fences in weak memory models (extended version)." *Formal Methods in System Design*. 40.2 (2012): 170-205. Print.
3. D. J. Sorin, M. D. Hill, and D. A. Wood, A Primer on Memory Consistency and Cache Coherence, Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers, 2011.