Towards Shared Memory Consistency Models for GPUs

Tyler Sorensen
University of Utah
Salt Lake City, UT
t.sorensen@utah.edu

Ganesh Gopalakrishnan
University of Utah
Salt Lake City, UT
ganesh@cs.utah.edu

Vinod Grover
NVIDIA
Santa Clara, CA
vgrover@nvidia.com

ABSTRACT

With the widespread use of graphical processing units (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. Compared to CPUs, GPUs present different shared memory behavior, and we know of no published formal consistency model for them. To fill this void, we establish a formal state transition model of GPU loads, stores, and fences in the language Murphi [2], and check properties – captured in litmus tests that pertain to ordering and visibility properties – over executions using the Murphi model checker.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Processor Architectures—*Multiple Data Stream Architectures (Multiprocessors)*; D.2.4 [**Software Engineering**]: Software/Program Verification—*model checking, reliability*

General Terms

Verification, Reliability

Keywords

GPU; Shared Memory Consistency; Memory Fences

1. MOTIVATION

We study GPU shared memory consistency with a focus on memory fences. Well-synchronized code can hide weak memory model issues from the programmer [4, p.64-65]; however, synchronization operations can be very expensive. Cutting-edge algorithms bypass these expensive operations to obtain significant speedup, at the cost of exposing the memory model [3], which must be understood for correct implementation. If developers write unsynchronized code assuming certain instruction orderings or memory visibilities which are not in line with the shared memory consistency model, then code can be buggy. We have found instances of such assumptions in real world code.

Memory fences can enforce different visibility semantics, including static (ordering within a thread) or dynamic (transitive visibility across threads) properties. Fences can also be cumulative which requires ensuring the visibility of values observed (but not written) by a given thread [1]. The current CUDA documentation only has scanty detail on these semantics.

ICS'13, June 10–14, 2013, Eugene, Oregon, USA.
ACM 978-1-4503-2130-3/13/06.

2. CONTRIBUTIONS

This work presents a variety of litmus tests i.e. short executions which are allowed, disallowed or guaranteed, that we believe illustrate the behavior of our subset of instructions. We provide tests that investigate classical properties such as coherence and write atomicity [4, p.11,69] as well as tests that take into account the unique memory and concurrency systems of GPUs. For example, different fences enforce different visibilities for inter and intra block threads. The results of the tests were determined by feedback from industry experts, the limited documentation available, and properties desired in modern multi-core systems. Some behaviors e.g. write atomicity, remain unresolved.

Using the results of the litmus tests, we developed an operational shared memory consistency model using intuitive data structures and simple operations. At a high level, each thread has: an instruction queue for each address, a view of global and shared memory and a set of unissued loads. Instructions are issued nondeterministically from instruction queues. Memory locations have flags and are shared between threads based on these flags.

Finally, we implemented our model in the Murphi [2] modeling language and validated our litmus tests through model checking. We provide our model at: `http://www.cs.utah.edu/~tylers/CUMM` along with instructions how to investigate custom tests which can aid developers.

Our GPU shared memory consistency model is currently being reviewed by industry experts and is expected to grow and change based on feedback. We have shown real world motivation for a GPU memory model and provided a reasonable model based on related work and industry feedback.

3. REFERENCES

[1] J. Alglave, L. Maranget, S. Sarkar, and P. Sewell. *Fences in weak memory models,* CAV 2010. Springer-Verlag.

[2] D. Dill. *The Murphi verification system,* CAV 1996. Springer.

[3] N. M. Lê, A. Pop, A. Cohen, and F. Zappa Nardelli. *Correct and efficient work-stealing for weak memory models,* PPoPP 2013. ACM.

[4] D. J. Sorin, M. D. Hill, and D. A. Wood. *A Primer on Memory Consistency and Cache Coherence,* 2011. Morgan & Claypool Publishers.