Testing GPU Memory Models

Daniel Poetzl

Joint work with
Jade Alglave (UCL), Mark Batty (Cambridge), Alastair Donaldson (Imperial), Ganesh Gopalakrishnan (Utah), Tyler Sorensen (Utah), John Wickerson (Imperial)
Graphics Processing Units (GPUs)

- GPUs have traditionally been designed to accelerate graphics applications
- General-purpose computing on GPUs (GPGPU) is becoming increasingly widespread (CUDA, OpenCL)
- GPUs have found their way into many systems:
  - Desktops and Laptops
  - Supercomputers
  - iPhone 5S
  - Audi self-driving car
    - Video processing
    - Safety-critical (!)
GPU Architectures
Nvidia's Maxwell (2014) - GTX 750 Ti

- Streaming Multiprocessor (SM)
  - Processing Block
    - Warp Scheduler
    - $PE_1, PE_2, \ldots, PE_n$
  - L1 Cache
  - Processing Block
    - Warp Scheduler
    - $PE_1, PE_2, \ldots, PE_n$
  - L1 Cache
  - Shared Memory
  - x 5

- L2 Cache
- DRAM
Programming Model

CUDA

- CUDA is Nvidia’s framework for general-purpose computing on GPUs
- Threads are hierarchically organized:

```
          Kernel
        /     \
       /       \
Block-----Block
     /         /
Warp-----Warp
       |       |
  T_0 ... T_{31}   T_{32} ... T_{63}   T_{64} ... T_{95}   T_{96} ... T_{127}
```

- Different memory spaces: global, shared, local, constant, texture, parameter
Weak Memory Models

- Which behaviours can be observed on a GPU when threads concurrently access the memory?
  - Consult the manual: prose, ambiguous, little detail, sometimes plain wrong
  - We want a formal memory model!
Weak Memory Models

- Which behaviours can be observed on a GPU when threads concurrently access the memory?
  - Consult the manual: prose, ambiguous, little detail, sometimes plain wrong
  - We want a formal memory model!

- Formal memory model based on:
  - Vendor documentation
  - Testing
  - Discussion with industry contacts
Test Framework

- We extended the diy and litmus tools to generate and run GPU litmus tests

- diy to generate tests
  - Short assembly code snippets called litmus tests
  - Test generation based on an axiomatic modeling framework

- litmus to run tests
  - Runs tests produced by diy many times
  - Adds additional code to create noise ("incantations") to make weak behaviors appear
GPU Litmus Tests

P0
mov.s32 r0,1
st.wb.s32 [x],r0
mov.s32 r2,1
st.wb.s32 [y],r2

P1
ld.ca.s32 r0,[y]
ld.ca.s32 r2,[x]

(device (kernel (block (warp P0) (warp P1))))

x: global, y: global

exists
(1:r0=1 \ 1:r2=0)
Read-Read-Coherenece

Consider the following test, with $P_0$ and $P_1$ in different blocks, $x$ in global memory, and initially $x = 0$:

<table>
<thead>
<tr>
<th></th>
<th>$P_0$</th>
<th>$P_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r1 = x$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r2 = x$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r1 == 1 \land r2 == 0$</td>
</tr>
</tbody>
</table>
Consider the following test, with $P_0$ and $P_1$ in different blocks, $x$ in global memory, and initially $x = 0$:

<table>
<thead>
<tr>
<th>$P_0$</th>
<th>$P_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = 1$</td>
<td>$r1 = x$</td>
</tr>
<tr>
<td></td>
<td>$r2 = x$</td>
</tr>
<tr>
<td></td>
<td>$r1 == 1 \land r2 == 0$</td>
</tr>
</tbody>
</table>

Running this test 100,000 times with litmus on the GeForce GTX 660 yields the following histogram:

<table>
<thead>
<tr>
<th>TEST</th>
<th>CORR</th>
<th>ALLOWED</th>
</tr>
</thead>
<tbody>
<tr>
<td>59875</td>
<td>$&gt;1: r0=0; 1: r2=0;$</td>
<td></td>
</tr>
<tr>
<td>828</td>
<td>$*&gt;1: r0=1; 1: r2=0;$</td>
<td></td>
</tr>
<tr>
<td>2422</td>
<td>$&gt;1: r0=0; 1: r2=1;$</td>
<td></td>
</tr>
<tr>
<td>36875</td>
<td>$&gt;1: r0=1; 1: r2=1;$</td>
<td></td>
</tr>
</tbody>
</table>
Read-Read-Coherence

Consider the following test, with $P_0$ and $P_1$ in different blocks, $x$ in global memory, and initially $x = 0$:

<table>
<thead>
<tr>
<th>$P_0$</th>
<th>$P_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = 1$</td>
<td>$r1 = x$</td>
</tr>
<tr>
<td></td>
<td>$r2 = x$</td>
</tr>
<tr>
<td></td>
<td>$r1 == 1 \land r2 == 0$</td>
</tr>
</tbody>
</table>

Behaviour is considered a bug:

- Does not guarantee what is typically required by programming language standards (OpenCL, C++11)
- OpenCL and C++11 require that there is a total order on all writes to a memory location (coherence order)
- No thread shall read values that contradict this order

Bug occurred in all Nvidia chips of the Fermi and Kepler generations we tested

(Appears to be) fixed in the new Maxwell architecture
Read-Read-Coherence

- Consider the following test, with $P_0$ and $P_1$ in different blocks, $x$ in global memory, and initially $x = 0$:

<table>
<thead>
<tr>
<th>$P_0$</th>
<th>$P_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = 1$</td>
<td>$r_1 = x$</td>
</tr>
<tr>
<td></td>
<td>$r_2 = x$</td>
</tr>
<tr>
<td>$r_1 == 1 \land r_2 == 0$</td>
<td></td>
</tr>
</tbody>
</table>

- GPUs are fairly deterministic (compared to CPUs)

- By fixing the random test parameters, we can make the bug deterministically show up (on Fermi and Kepler GPUs):

Test CoRR Allowed
Histogram (1 state)
100000 $\ast>1:r0=1; 1:r2=0;$
Summary

- DIY to generate GPU litmus tests
- litmus to run GPU litmus tests
- Testing the hardware is a necessary first step towards building a formal memory model:
  - Documentation is insufficient
  - Side effect: We find bugs in the hardware
  - Test results serve as a basis for communication with industry contacts

Thank you!
Summary

- diy to generate GPU litmus tests
- litmus to run GPU litmus tests
- Testing the hardware is a necessary first step towards building a formal memory model:
  - Documentation is insufficient
  - Side effect: We find bugs in the hardware
  - Test results serve as a basis for communication with industry contacts

Thank you!