Kernel Interceptor

Ethel Bardsley, Alastair Donaldson, John Wickerson

GPUVerify

- A tool from Multicore at Imperial College London, with Microsoft Research
- Tries to prove absence of data races and barrier divergence — another piece in the trust puzzle
- Found various real world bugs

Technique overview

- Two arbitrary threads, *i* and *j* where *i != j*
- Thread *i* logs
- Thread *j* checks
- Barriers clear the log

Example

```
kernel void foo (local int* A) {

    int tid = get_local_id(0);

    int t = A[tid+1];

    A[tid] = t;

}
```

```
kernel void foo (local int* A) {

}
```

Example

```
kernel void foo (local int* A) {

    int tid = get_local_id(0);

    int t = A[tid+1];

    A[tid] = t;

}
```

```
kernel void foo (local int* A) {

    int tid1, tid2;

}
```

Example

```
kernel void foo (local int* A) {

    int tid = get_local_id(0);

    int t = A[tid+1];

    A[tid] = t;

}
```

```
kernel void foo (local int* A) {

    int tid1, tid2;

    reads_A += {tid1+1};
    assert (tid2+1) ∉ writes_A;
    havoc t1, t2;


}
```

Example

```
kernel void foo (local int* A) {

    int tid = get_local_id(0);

    int t = A[tid+1];

    A[tid] = t;

}
```

```
kernel void foo (local int* A) {

    int tid1, tid2;

    reads_A += {tid1+1};
    assert (tid2+1) ∉ writes_A;
    havoc t1, t2;

    writes_A += {tid1};
    assert tid1 ∉ writes_A U reads_A;

}
```

Example

```
kernel void foo (local int* A) {

    int tid = get_local_id(0);

    int t = A[tid+1];

    barrier(CLK_LOCAL_MEM_FENCE);

    A[tid] = t;

}
```

```
kernel void foo (local int* A) {

    int tid1, tid2;

    reads_A += {tid1+1};
    assert (tid2+1) ∉ writes_A;
    havoc t1, t2;

    reads_A = {};
    writes_A = {};

    writes_A += {tid1};
    assert tid1 ∉ writes_A ∪ reads_A;

}
```

Proven success

- Large number of GPGPU features supported
- High degree of automation, e.g., invariant inference

- However, not quite there…

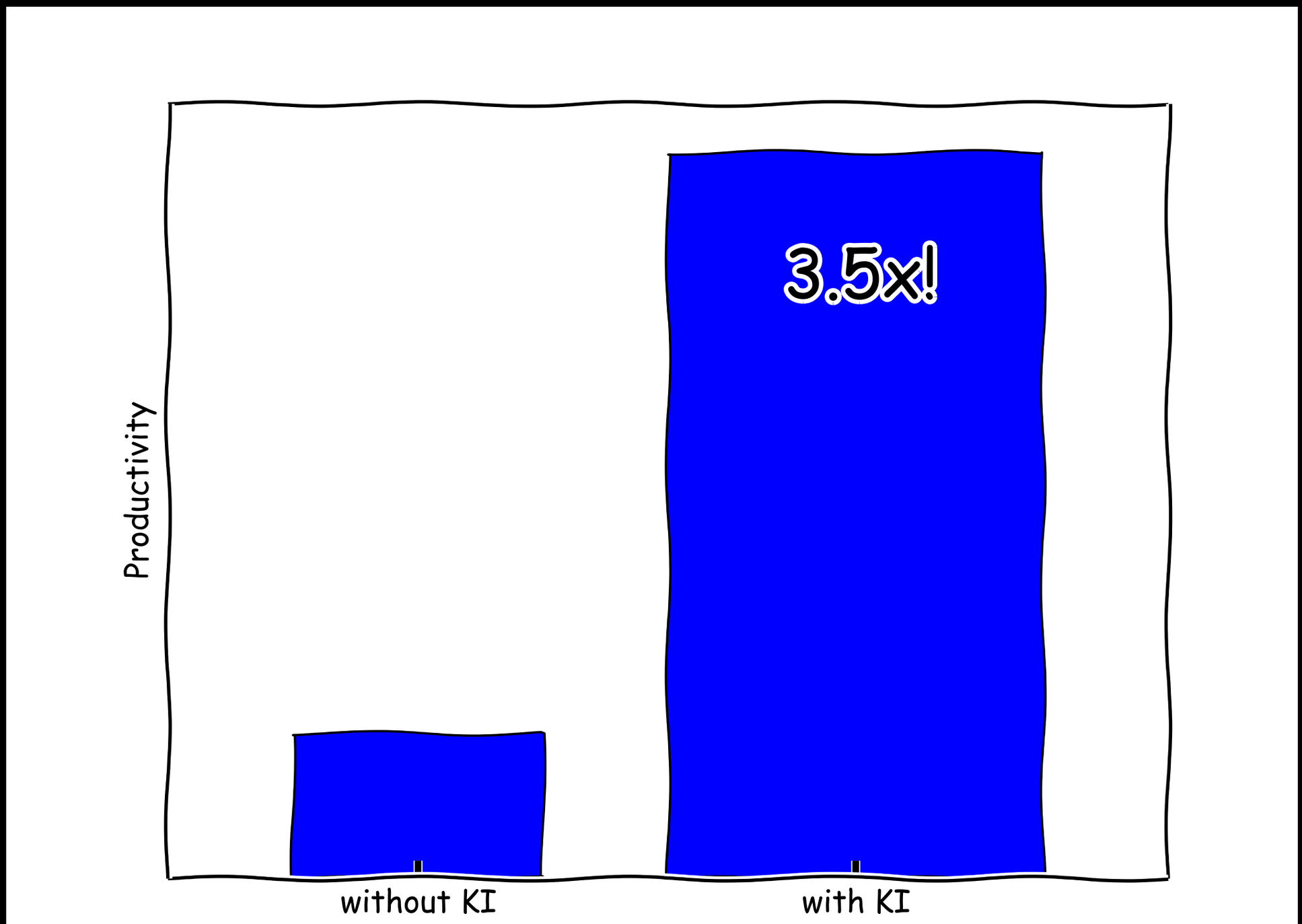
Kernel Interceptor

- Another level of automation
- Small portable C++ “library”
- Unobtrusive, suitable for debug build

In a nutshell

- Log calls to the OpenCL runtime
- At kernel launch, dump that instance
- Offline, run analysis on acquired instances

Graph slide!



Further work

- Inferring data-dependent invariants
- Finer grained parameter abstraction

- Please offer suggestions!

Summary

- Use GPUVerify
- Ease the pain with Kernel Interceptor

QUESTIONS?

multicore.doc.ic.ac.uk/tools/GPUVerify/IWOCL2014

gpuverify.codeplex.com

rise4fun.com/GPUVerify-OpenCL