Automatic Test Case Reduction for OpenCL

Moritz Pflanzer
Alastair F. Donaldson
Andrei Lascu
Reporting a compiler bug

Program

```cpp
color = compute_red();
set_bg(color);
```

- Compiler ♥
- Compiler B
- Compiler C
Reporting a compiler bug

Program

```cpp
color = compute_red()
set_bg(color);
```

```cpp
// camera and ray
vec3 cPos = vec3(0.0, 0.0, time);
vec3 cUp = normalize(vec3(0.1, 0.4, 0.0));
vec3 cDir = cross(cUp, vec3(-1.0, 0.0, 0.0));
vec3 cSide = cross(cDir, cUp);
vec3 ray = normalize(cSide * p.x +
                   cUp * p.y + cDir * targetDepth);

// direction light
vec3 lightDir = normalize(vec3(1, 1, -2));

// marching loop
float dist;
float depth = 0.0;
vec3 dPos = cPos;
for(int i = 0; i < 64; i++){
    dPos = cPos + depth * ray;
    if (abs(dist) < EPS) break;
}
[...]
```

Code excerpt taken from http://glisandbox.com/e#29059
Reporting a compiler bug
Excerpt from a CLsmith*-generated kernel

```
for (p_2718->g_197 = 0; (p_2718->g_197 < 51); p_2718->g_197 =
    safe_add_func_int64_t_s_s(p_2718->g_197, 4))
{
    /* block id: 242 */
    int16_t *l_445 = (void*)0;
    int16_t ***l_443 = &l_444[2][2];
    int i, j;
    (*l_437) = p_60;
    p_2718->g_446 = ((*l_443) = (void*)0);
}

(*l_467) ^= (0x68097A9AF58B0784L | ((safe_div_func_uint64_t_u_u((l_433 ,
(safe_unary_minus_func_int16_t_s((safe_sub_func_int32_t_s_s((p_61 ==
((safe_mod_func_uint8_t_u_u(l_422 != (*l_457) = &l_314[3]),
((safe_lshift_func_int16_t_s_u(((((*l_460)--) && (safe_lshift_func_uint8_t_u_s((p_61 , p_2718->g_167[9][3]), ((safe_rshift_func_uint16_t_u_u(0xA1B0L, (((void*)0 != &p_2718->g_446) < 1UL)) |
p_2718->g_216)))) , l_375) , p_61), 10)) ^ p_2718->g_234.f2.f1)) >= p_2718->g_216), l_319)))))) , 1L))
    if (((*p_60))
        break;
}
```

* Many-core Compiler Fuzzing. Lidbury et al., PLDI'15.
Overview

- Creduce
- Lifting Creduce to OpenCL
- ShadowKeeper
- Experimental evaluation
- Future work
- Closing remarks
int foo (int x) {
    int y = 3;
    int a[20];
    for (int i = 0; i <= 20; i++)
        a[i] = i;
    if (x+y < 20)
        return a[x+y];
    return a[19];
}
int foo (int x) {
    int y = 3;
    int a[20];
    for (int i = 0; i <= 20; i++)
        a[i] = i;
    if (x+y < 20)
        return a[x+y];
    return a[19];
}
Creduce
(Test-case reduction for C compiler bugs. Regehr et. al, PLDI'12)

```c
int foo (int x) {
    int y = 3;
    int a[20];
    for (int i = 0; i <= 20; i++)
        a[i] = i;
    if (x+y < 20)
        return a[x+y];
    return a[19];
}
```
int foo (int x) {
    int y = 3;
    int a[20];
    for (int i = 0; i <= 20; i++)
        a[i] = i;
    if (x+y < 20)
        return a[x+y];
    return a[19];
}
Creduce

*(Test-case reduction for C compiler bugs. Regehr et. al, PLDI'12)*

```c
int foo (int x) {
    int y = 3;
    int a[20];
    for (int i = 0; i <= 20; i++)
        a[i] = i;
    if (x+y < 20)
        return a[x+y];
    return a[19];
}
```
Creduce
(Test-case reduction for C compiler bugs. Regehr et al., PLDI'12)
Creduce

(*Test-case reduction for C compiler bugs. Regehr et al., PLDI'12*)
Lifting Creduce to OpenCL

Clang \rightarrow Clang static analyser \rightarrow Oclgrind

OpenCL kernel \rightarrow Apply reduction

Custom tailored based on CLsmith*-generated kernels

* Many-core Compiler Fuzzing. Lidbury et al., PLDI'15.
Lifting Creduce to OpenCL

• New bugs
  – data races
  – barrier divergence
• New tools
  – Oclgrind
Lifting Creduce to OpenCL

- Modular OpenCL emulator
- Checks for multiple types of errors in kernels
  - data races
  - uninitialized variables
Oclgrind's former uninitialized value detection plugin was insufficient

Conceptually based on Valgrind's Memcheck and Clang's MemorySanitizer

Strives for bit level accuracy, except when it would greatly impact performance
ShadowKeeper
Technical details

• Real operations simulated in shadow memory
• Shadow memory contains definedness of each bit of real memory
• Propagate definedness along shadow operations
Experimental evaluation

Setup

- 5 OpenCL devices
- 127 kernels yielding different results between optimised and not optimised compilation
- 272 automatic reductions
Experimental evaluation
CDF of reduction times
Experimental evaluation
Comparison of average times for multiple parallel tests

![Bar chart showing reduction time for sequential and parallel tests](chart.png)
Experimental evaluation
ShadowKeeper versus Old plugin
Future work

- Usability improvements
- Improve reduction time
- Bug analysis
Closing remarks

- Lifted the Creduce framework to OpenCL
- Using a small amount of human time, can find reasonably small kernels presenting compiler bugs
- Improved Oclgrind, including developing a new uninitialized value detection plugin
Thank you!

Questions?

al2510@imperial.ac.uk
multicore.doc.ic.ac.uk