Debugging

A Collaboration Between
David Kaeli, Northeastern University
Benedict R. Gaster, AMD

Instructor Notes

- GPU debugging is still immature, but being improved daily. You should definitely check to see the latest options available before giving this lecture.

Debugging Techniques

- Compiling for x86 CPU
  - Debugging with GDB
- GPU printf
- Live debuggers
  - Parallel Nsight
  - gDEBugger

CPU Debugging

- OpenCL allows the same code to run on different types of devices
  - Compiling to run on a CPU provides some extra facilities for debugging
  - Additional forms of IO (such as writing to disk) are still not available from the kernel
- AMD's OpenCL implementation recognizes any x86 processor as a target device
  - Simply select the CPU as the target device when executing the program
- NVIDIA's OpenCL implementation can support compiling to x86 CPUs if AMD's installable client driver is installed

CPU Debugging with GDB

- Setting up for GDB
  - Pass the compiler the “-g” flag
    - Pass “-g” to `clBuildProgram()`
    - Set an environment variable `CPU_COMPILER_OPTIONS="-g"`
  - Avoid non-deterministic execution by setting an environment variable `CPU_MAX_COMPUTE_UNITS=1`

CPU Debugging with GDB

- Run gdb with the OpenCL executable

```
> gdb a.out
```

- Breakpoints can be set by line number, function name, or kernel name
- To break at the kernel `hello` within gdb, enter:

```
(gdb) b __OpenCL_hello_kernel
```

  - The prefix and suffix are required for kernel names

- OpenCL kernel symbols are not known until the kernel is loaded, so setting a breakpoint at `clEnqueueNDRangeKernel()` is helpful

```
(gdb) b clEnqueueNDRangeKernel
```

CPU Debugging with GDB

- To break on a certain thread, introduce a conditional statement in the kernel and set the breakpoint inside the conditional body
  - Can use gdb commands to view thread state at this point

```
...
if(get_global_id(1) == 20 &&
   get_global_id(0) == 34) {
   ;  // Set breakpoint on this line
}
```

GPU Printf

- AMD GPUs support printing during execution using `printf()`
  - NVIDIA does not currently support printing for OpenCL kernels (though they do with CUDA/C)
- AMD requires the OpenCL extension *cl_amd_printf* to be enabled in the kernel
- `printf()` closely matches the definition found in the C99 standard

GPU Printf

- `printf()` can be used to print information about threads or check help track down bugs
- The following example prints information about threads trying to perform an improper memory access

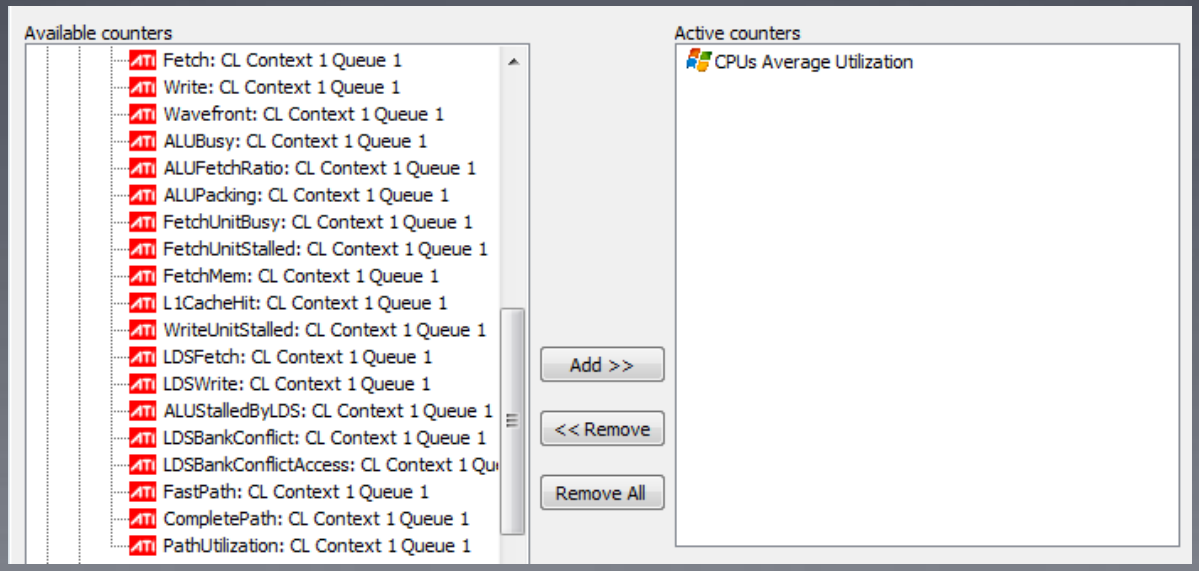
```
int myIdx = ... // index for addressing a matrix

if(myIdx < 0 || myIdx >= rows || myIdx >= cols) {

    printf("Thread %d,%d: bad index (%d)\n",

    get_global_id(1), get_global_id(0), myIdx));

}
```

GPU Printf

- `printf()` works by buffering output until the end of execution and transferring the output back to the host
  - It is important that a kernel completes in order to retrieve printed information
  - Commenting out code following `printf()` is a good technique if the kernel is crashing

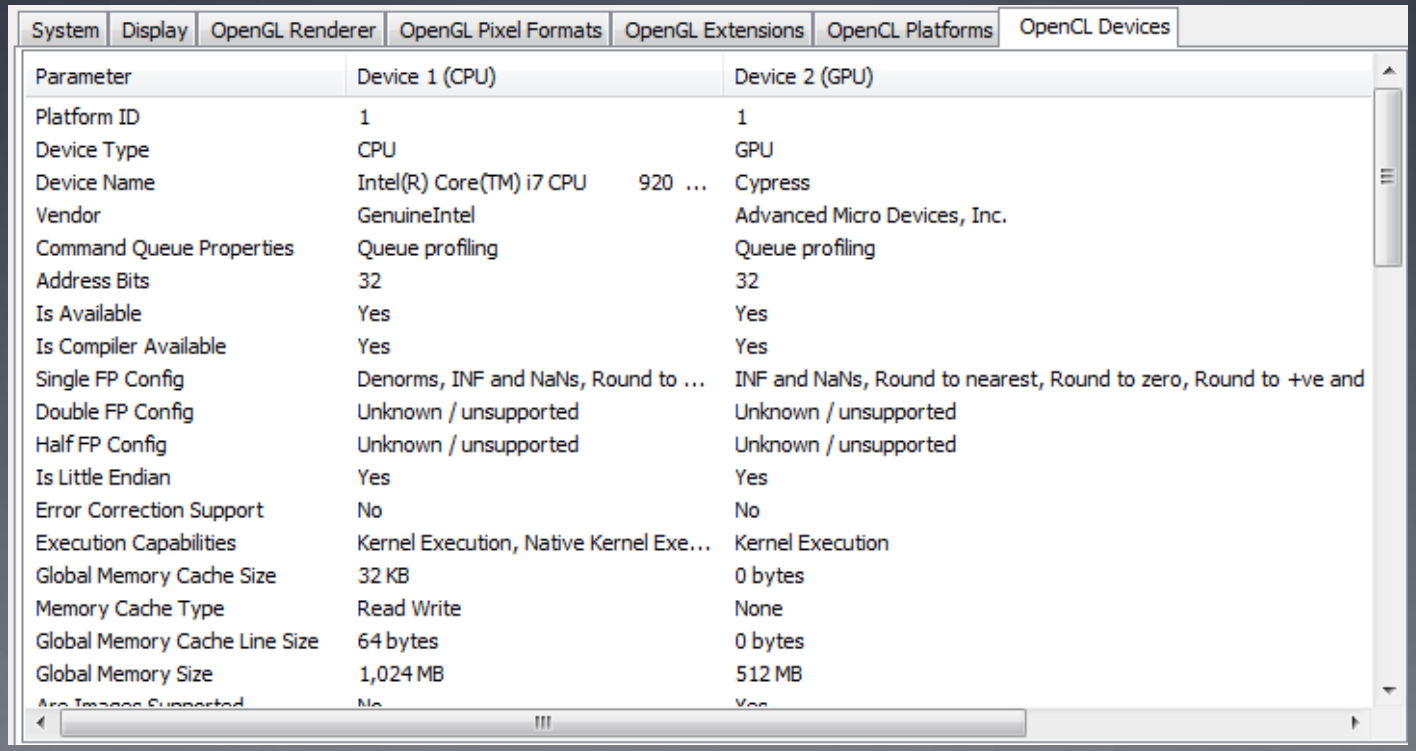
gDEBugger

- Developed by Graphic Remedy
  - Cost: not free
- Debugger, profiler, memory analyzer
- Integrated with AMD/ATI and NVIDIA performance counters



gDEBugger

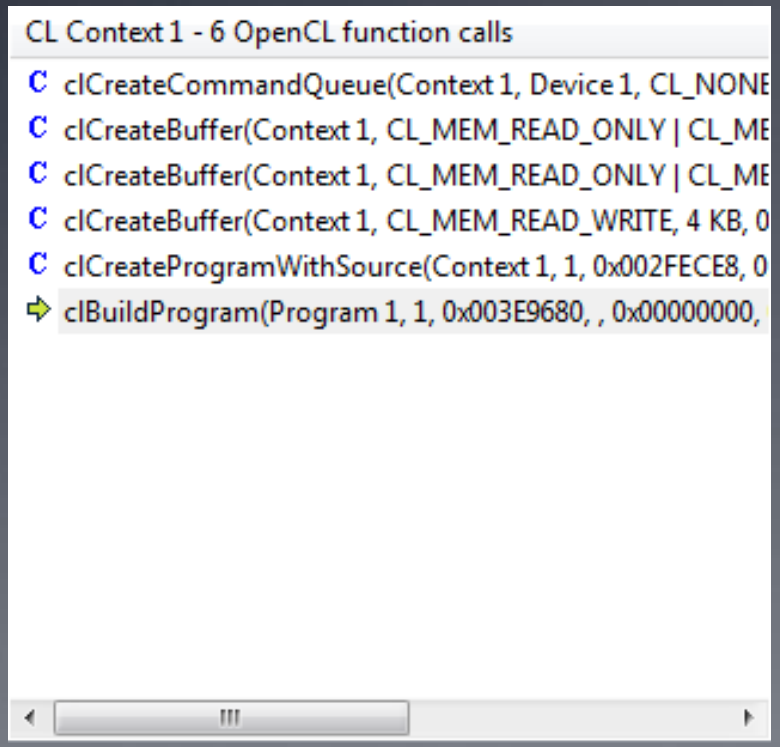
- Displays information about OpenCL platforms and devices present in the system

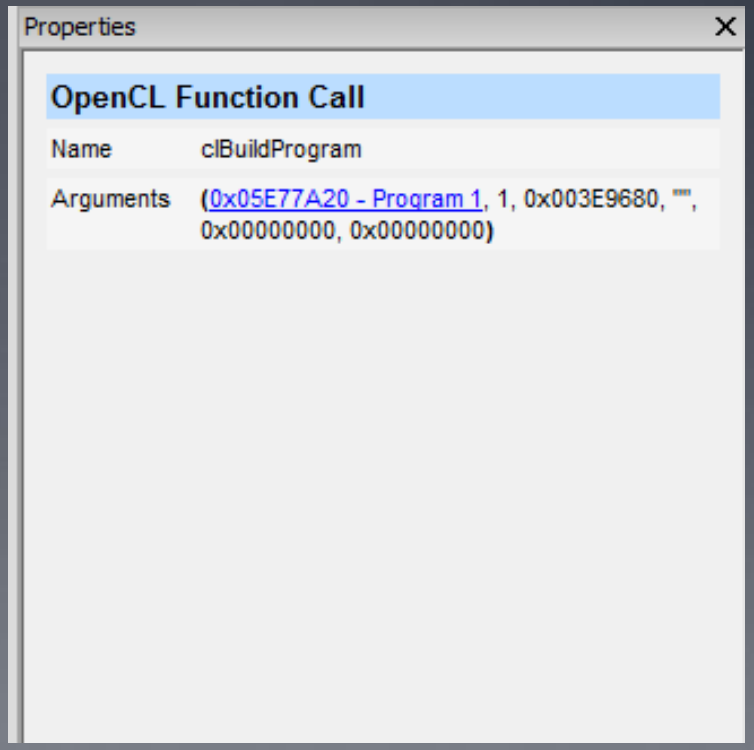


Parameter	Device 1 (CPU)	Device 2 (GPU)
Platform ID	1	1
Device Type	CPU	GPU
Device Name	Intel(R) Core(TM) i7 CPU 920 ...	Cypress
Vendor	GenuineIntel	Advanced Micro Devices, Inc.
Command Queue Properties	Queue profiling	Queue profiling
Address Bits	32	32
Is Available	Yes	Yes
Is Compiler Available	Yes	Yes
Single FP Config	Denorms, INF and NaNs, Round to ...	INF and NaNs, Round to nearest, Round to zero, Round to +ve and
Double FP Config	Unknown / unsupported	Unknown / unsupported
Half FP Config	Unknown / unsupported	Unknown / unsupported
Is Little Endian	Yes	Yes
Error Correction Support	No	No
Execution Capabilities	Kernel Execution, Native Kernel Exe...	Kernel Execution
Global Memory Cache Size	32 KB	0 bytes
Memory Cache Type	Read Write	None
Global Memory Cache Line Size	64 bytes	0 bytes
Global Memory Size	1,024 MB	512 MB
Are Images Supported	No	Yes

gDEBugger

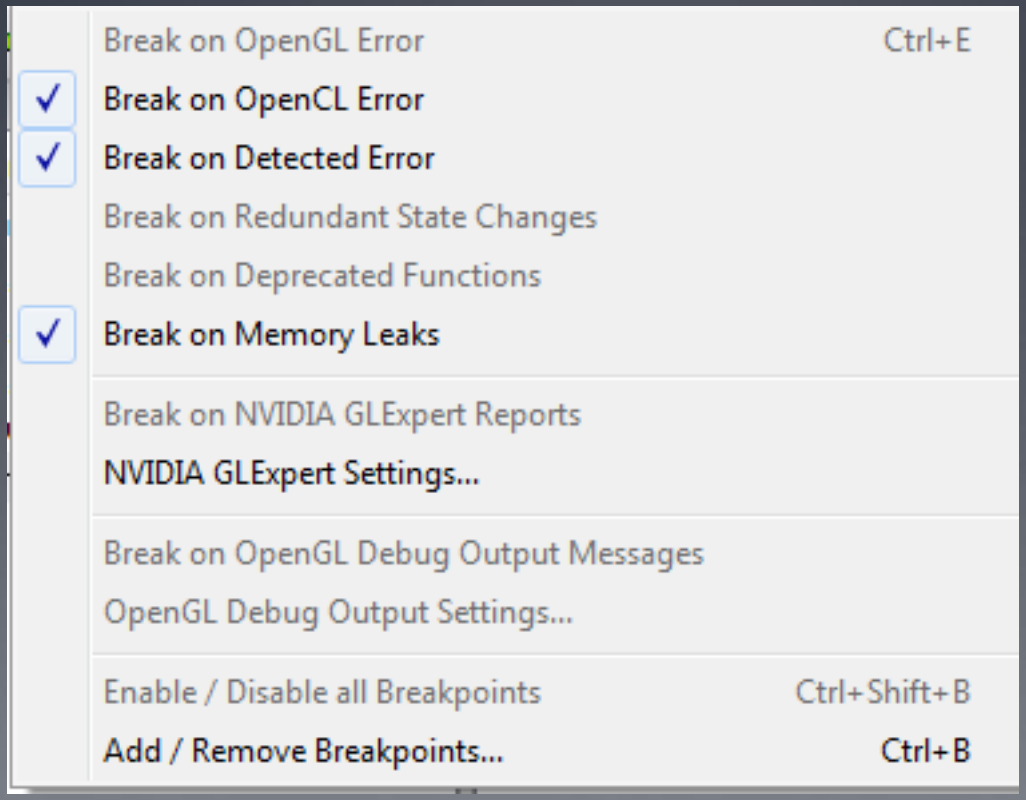
- Can step through OpenCL calls, and view arguments
  - Links to programs, kernels, etc. when possible in the function call view





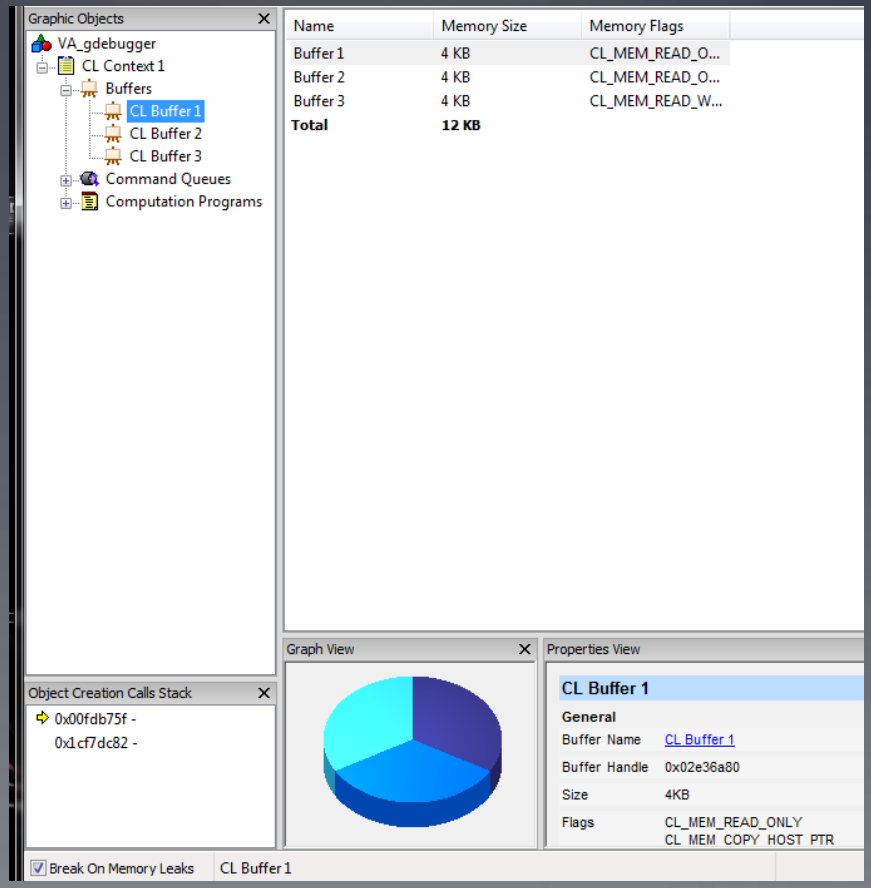
gDEBugger

- Automatically detects OpenCL errors and memory leaks



gDEBugger

- Displays contents of buffers and images present on OpenCL devices
  - View live
  - Export to disk



Summary

- GPU debugging is still immature
  - NVIDIA has a live debugger for Windows only
  - AMD and NVIDIA allow restrictive printing from the GPU
  - AMD allows code to be compiled and run with gdb on the CPU
  - Graphic Remedy (gDEBugger) provides online memory analysis and is integrated with performance counters, but cannot debug on a thread-by-thread basis