Oclgrind: An Extensible OpenCL Device Simulator

James Price & Simon McIntosh-Smith
University of Bristol - High Performance Computing Group
http://uob-hpc.github.io

Funded in part by Imagination Technologies

Overview

- Simulates OpenCL kernels executing on a virtual OpenCL device

- Architecture-agnostic simulation

- Built on an interpreter for LLVM/SPIR 1.2

- Plugin interface delivers extensibility

Abstract Simulation

- Doesn’t model any specific architectural characteristics
- Simulates kernel execution with respect to the OpenCL execution and memory models
- Understands concepts such as work-items, work-groups, and the different address spaces

OpenCL Runtime API

- Provides a comprehensive implementation of the OpenCL 1.2 runtime API

- This allows existing OpenCL applications to target Oclgrind without the need for modifications

- Accepts OpenCL programs as either OpenCL C source or SPIR 1.2 binaries

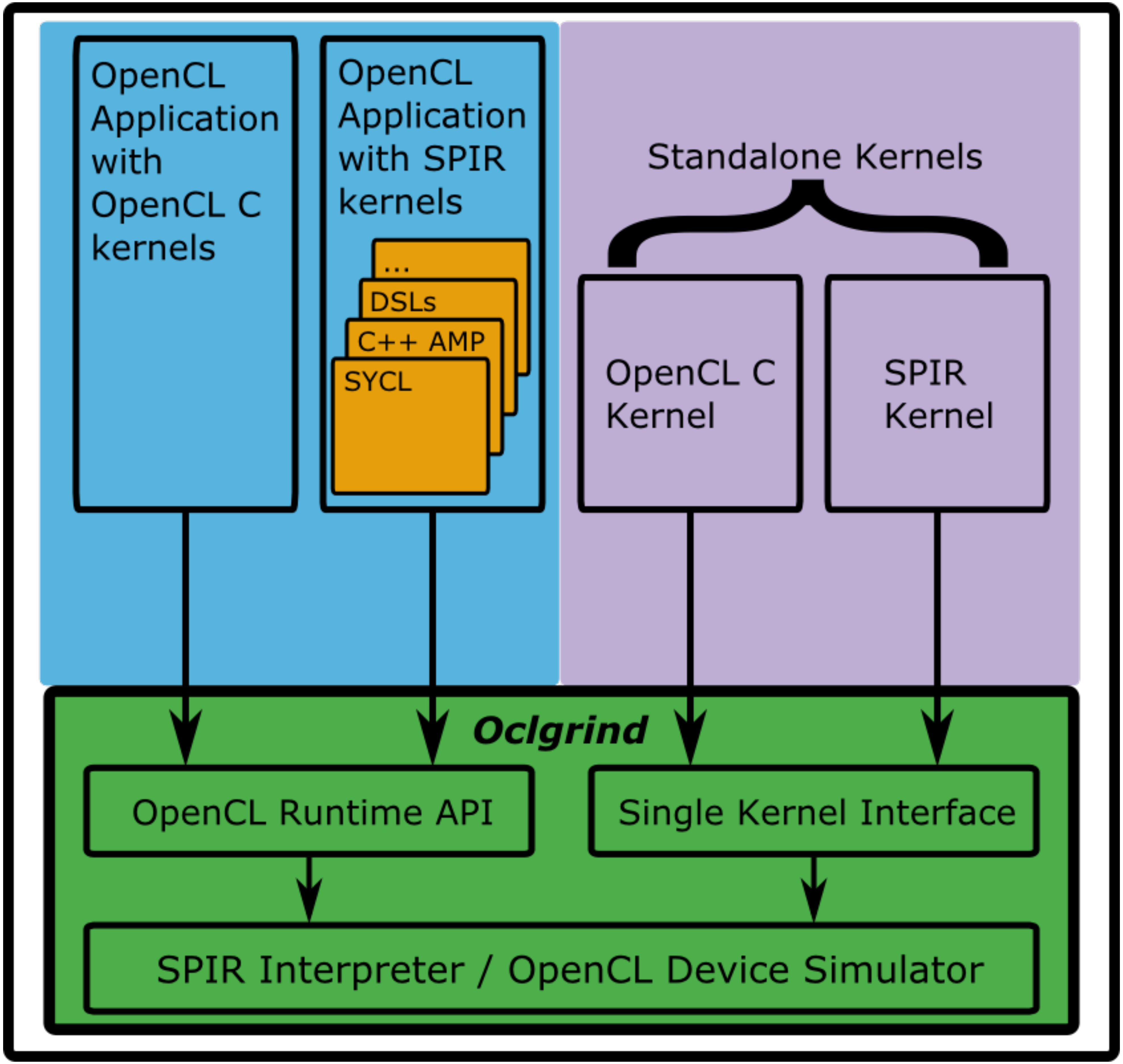
Single Kernel Interface

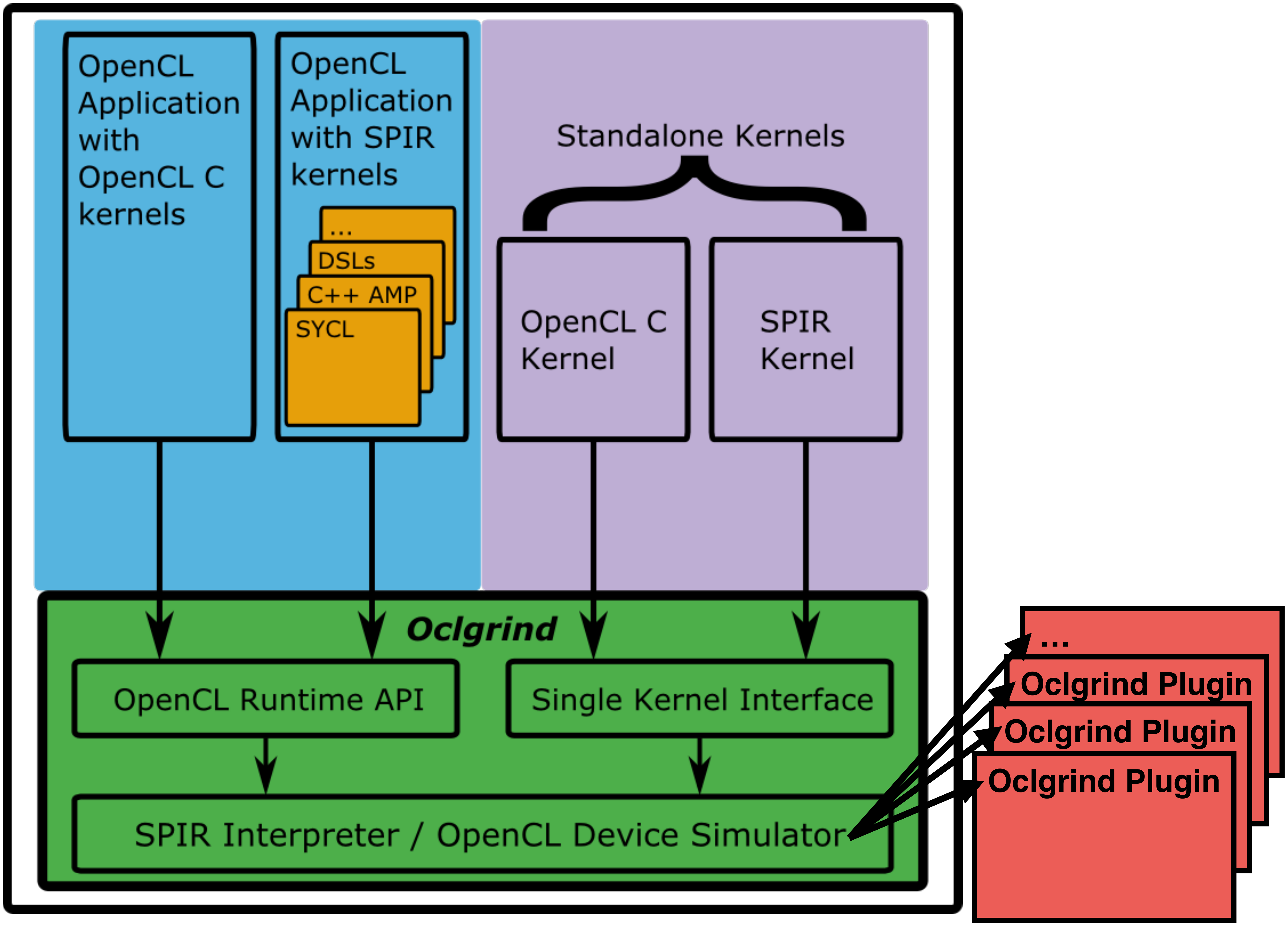
- Provides an interface to run individual kernels
- Simple configuration file describes kernel launch configuration and arguments
- Useful when analysing a specific kernel in a large application

Single Kernel Interface

- Provi s
- Simp unch confi
- Usefu large appli

```
vecadd.cl # File containing OpenCL program
vecadd    # Name of kernel to run
1024 1 1  # NDRange
  16 1 1  # Work-group size

# First argument 'global int *a'
<size=4096 range=0:1:4095>

# Second argument 'global int *b'
<size=4096 range=4096:1:8191>

# Third argument 'global int *c'
<size=4096 fill=0 dump>

# Fourth argument 'int size'
<size=4>
1024
```


OpenCL Application with OpenCL C kernels
OpenCL Application with SPIR kernels
...
DSLs
C++ AMP
SYCL
Standalone Kernels
OpenCL C Kernel
SPIR Kernel
Oclgrind
OpenCL Runtime API
Single Kernel Interface
SPIR Interpreter / OpenCL Device Simulator


OpenCL Application with OpenCL C kernels
OpenCL Application with SPIR kernels
...
DSLs
C++ AMP
SYCL
Standalone Kernels
OpenCL C Kernel
SPIR Kernel
Oclgrind
OpenCL Runtime API
Single Kernel Interface
SPIR Interpreter / OpenCL Device Simulator
...
Oclgrind Plugin
Oclgrind Plugin
Oclgrind Plugin

Plugin Interface

- Delivers extensibility
- Plugins can be registered with Oclgrind to receive information about the simulation via callbacks
- Allows third-party developers to build tools on top of the simulator
- Plugins are passive

Plugin Callbacks

- Kernel begin/end
- Work-item/work-group begin/end
- Instruction executed
- Memory allocated/deallocated
- Memory load/store/atomic
- Work-group barrier

Plugin Callbacks

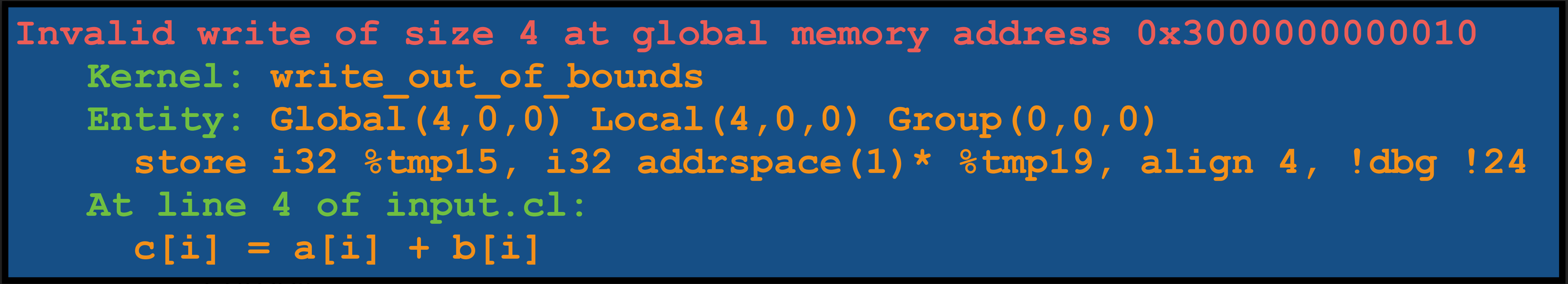
```
#include "oclgrind/Context.h"
#include "oclgrind/Plugin.h"
#include "oclgrind/WorkItem.h"

class InstPrinter : public oclgrind::Plugin
{
public:
  InstPrinter(const oclgrind::Context *context)
    : oclgrind::Plugin(context){};
  void instructionExecuted(const oclgrind::WorkItem *workItem,
                           const llvm::Instruction *instruction,
                           const oclgrind::TypedValue& result)
  {
    std::cout << "Work-Item " << workItem->getGlobalID() << ": ";
    oclgrind::dumpInstruction(std::cout, instruction);
    std::cout << std::endl;
  }
};
```

Memory Access Checking

- Checks addresses used by load/store instructions
- Informs user when OpenCL kernels access invalid memory locations
- Also checks for violations of CL_MEM_READ_ONLY/WRITE_ONLY
- Finding bugs in real programs:
  - CloverLeaf
  - Parboil
  - ViennaCL

Memory Access Checking

- Checks addresses used by load/store instructions
- Informs user when OpenCL kernels access invalid memory locations



```
Invalid write of size 4 at global memory address 0x3000000000010
   Kernel: write_out_of_bounds
   Entity: Global(4,0,0) Local(4,0,0) Group(0,0,0)
     store i32 %tmp15, i32 addrspace(1)* %tmp19, align 4, !dbg !24
   At line 4 of input.cl:
     c[i] = a[i] + b[i]
```

  - Parboil
  - ViennaCL

- Also checks for violations of CL_MEM_READ_ONLY/WRITE_ONLY

Data-race Detection

- Keep track of when memory locations are read/written by work-items
- Handle synchronisation at work-group barriers
- Inform user when data-races are observed

Data-race Detection

```
Read-write data race at global memory address 0x1000000000004
   Kernel: global_read_write_race

   First entity:  Global(2,0,0) Local(0,0,0) Group(2,0,0)
     %tmp11 = load i32 addrspace(1)* %tmp10, align 4, !dbg !23
   At line 6 of input.cl:
     data[i] = data[i-1];

   Second entity: Global(1,0,0) Local(0,0,0) Group(1,0,0)
     store i32 %tmp11, i32 addrspace(1)* %tmp15, align 4, !dbg !23
   At line 6 of input.cl:
     data[i] = data[i-1];
```

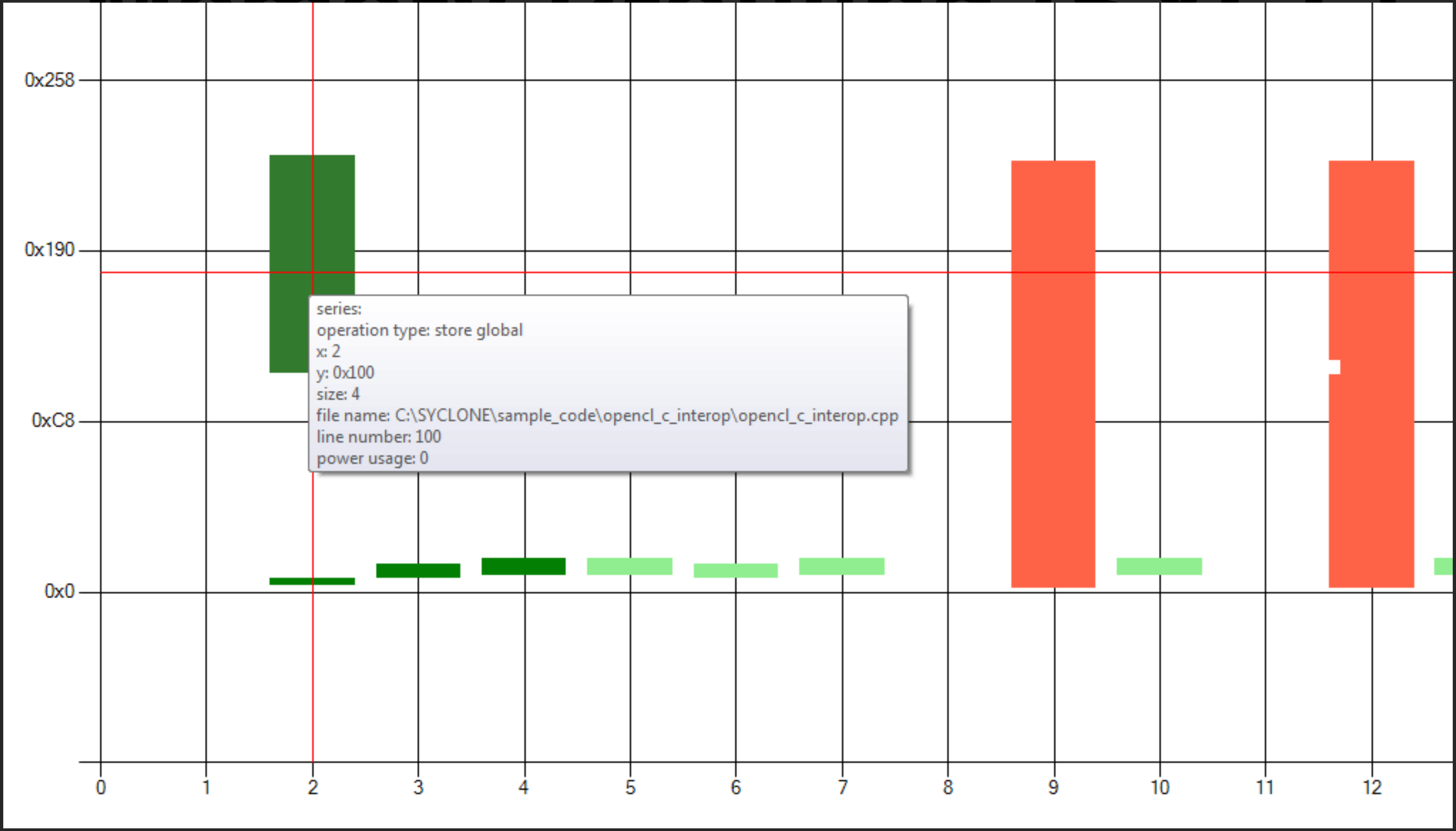
Interactive Debugging

- Provides a GDB-style interactive debugging interface
- Source line debugging of OpenCL C kernels
- Set breakpoints, inspect variables and memory, switch between work-items
- Automatically breaks when other plugins detect errors

Memory Profiling (SYCL)

- Implemented by Codeplay
- Uses Oclgrind to gather information about memory accesses within SYCL programs (via SPIR)
- Microsoft Visual Studio plugin to visualise these memory accesses, relating them back to the original source code

Memory Profiling (SYCL)



series:
operation type: store global
x: 2
y: 0x100
size: 4
file name: C:\SYCLONE\sample_code\opencl_c_interop\opencl_c_interop.cpp
line number: 100
power usage: 0

Other Features

- Detecting work-group divergence
- Detecting unaligned memory accesses
- Generating histograms of instructions executed
- Detecting other miscellaneous kernel errors
- Useful diagnostics for OpenCL runtime API errors

More Information

- Open source (GitHub)
- BSD license
- Compatible with Linux, Mac and Windows
- Feedback and contributions welcome (bug reports, pull requests, feature requests)

https://github.com/jrprice/Oclgrind/