mc-imperial / **gvki**

GPUVerify OpenCL Kernel Interceptor

111 commits | **3** branches | **0** releases | **2** contributors | BSD-3-Clause

Branch: **master** | **New pull request** | **Find file** | **Clone or download**

Peng Peng Merge branch 'master' of https://github.com/mc-imperial/gvki — Latest commit `5d7d6e6` on 6 Jul 2015

cmake/findopencl	Try to fix configure on OSX.	3 years ago
include	Added a feature (on by default) to make just one invocation of a kern…	2 years ago
lib	Merge branch 'master' of https://github.com/mc-imperial/gvki	2 years ago
tests	Change JSON interface for unconstrained local_size. Now we just don't	3 years ago
.gitattributes	Added .gitattributes file to try and make sure code is always checked	3 years ago
.gitignore	Ignore ctags file.	3 years ago
.travis.yml	Added travis file for TravisCI for Linux and OSX. OSX won't work	3 years ago
CMakeLists.txt	Updated CMake to avoid absolute path to inttypes for Windows. Added c…	2 years ago
LICENSE	Add Jeroen to list of authors in LICENSE file.	3 years ago
README.md	Updated CMake to avoid absolute path to inttypes for Windows. Added c…	2 years ago
TODO.md	Add stuff to TODO	3 years ago

README.md

GPUVerify OpenCL Kernel Interceptor

build failing

This is a simple wrapper library to intercept OpenCL host code calls to collect information needed by GPUVerify so it can verify executed kernels.

It provides two ways of intercepting calls to OpenCL host functions

- A preloadable library. A shared library is built which can used with `LD_PRELOAD` (Linux) or `DYLD_INSERT_LIBRARIES` (OSX) to intercept calls of binary applications.

```
# Linux
$ LD_PRELOAD=/path/to/gvki/lib/libGVKI_preload.so ./your_program

# OSX
```

```
$ DYLD_FORCE_FLAT_NAMESPACE=1 DYLD_INSERT_LIBRARIES=/path/to/gvki/lib/libGVKI_preload.dylib
./your_program
```

- "Macro library". For systems that do not support pre loadable libraries we also provide a header file that can be included in your application to rewrite all relevant calls to OpenCL host functions to calls into our interceptor library (`lib/libGVKI_macro.a`) which you then must link with afterwards.

Add `#include "gvki_macro_header.h"` into your source files just after your include of the OpenCL header e.g.

```
#include <CL/OpenCL.h>
#include "gvki/gvki_macro_header.h"

int main(int argc, char** argv)
{
...
}
```

Then compile your application linking the interceptor library

```
$ gcc -I/path/to/gvki_src/include/ your_application.c lib/libGVKI_macro.a -o your_application
```

Building

Requirements

- CMake >= 2.8.7
- OpenCL header files
- OpenCL library (needed for testing only)
- Python >= 2.7 (needed for testing only)

Linux/OSX

```
$ mkdir gvki
$ git clone git://github.com/mc-imperial/gvki.git src
$ mkdir build
$ cd build
$ cmake -DENABLE_TESTING:BOOL=ON ../src/
$ make
```

Note if you don't have a working OpenCL implementation on your system set `ENABLE_TESTING` to `OFF` .

Windows

1. Download msinttypes and put it somewhere on your machine: https://code.google.com/p/msinttypes/
2. Clone the gvki repository
3. Now run `cmake-gui` and set the source directory to the git repository and the build directory to anywhere you want (preferrably not the git repository)
4. Set MSINTTYPES_DIR to be the directory of your msinttypes download that contains inttypes.h.
5. Click the Configure button and select the generator you want to use (e.g. `Visual Studio 12 2013`)
6. CMake will try to configure. It is likely that CMake will not be able to find the OpenCL header files and libraries. If it does not you should manually set `OPENCL_INCLUDE_DIRS` (required) and `OPENCL_LIBRARIES` (only needed if you want to do testing) in the `cmake-gui` interface. Once you've done that press the configure button again until it succeeds.

7. Press the Generate button.
8. You can now build the project using the system relevant to the generator you chose. If you chose Visual Studio there will be a `.sln` file in the build directory you can open.

Testing

Run the test target (`ENABLE_TESTING` must be set to true when configuring with cmake)

```
$ make check
```

Output produced

When intercepting a `gvki-<N>` directory is created where `<N>` is the next available integer. The location of this directories can be controlled using `GVKI_ROOT` . If `GVKI_NO_NUM_DIRS` si specified then numbered directories are not created and instead everything is logged into `GVKI_ROOT` which must not already exist.

The directory contains the following

- `log.json` file should which contains information about logged executions.
- `<entry_point>.<M>.cl` files which are the logged OpenCL kernels where `<entry_point>` is the name of kernel and `<M>` is the next available integer.

An example invocation of GPUVerify on the logged kernels is

```
$ gpuverify --json gvki-0/log.json
```

Special environment variables

Setting various environment variables changes its behaviour

- `GVKI_DEBUG` Setting this causes debug information to be sent to stderr during interception.
- `GVKI_ROOT` is the directory that `gvki-*` directories are created in. If not set the current working directory is used.
- `GVKI_LOG_FILE` Setting this to a valid file path will cause logging messages to be written to a file in addition to the normal stderr output.
- `GVKI_NO_NUM_DIRS` Setting this causes `GVKI_ROOT` to be used as the directory for logging files instead of using `gvki-*` .