COMPUTE SANITIZER

v2020.2.1 | October 2020

Release Notes

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1. RELEASE NOTES

1.1. Updates in 2020.2.1

- Fixed crash when loading cubins of size larger than 2 GiB.
- Fix error detection on systems with multiple GPUs.
- Fixed issue when using CUDA Virtual Memory Management API **`cuMemSetAccess`** to remove access to a subset of devices on a system with multiple GPUs.
- Added public API to translate between sanitizer and CUDA stream handles.

1.2. Updates in 2020.2

- Added support for CUDA graphs and CUDA memmap APIs.
- The memory access callback of the public API has been split into three distinct callbacks corresponding to global, shared and local memory accesses.

1.3. Updates in 2020.1.2

- Added sanitizer stream API. This fixes tool crashes when per-thread streams are being used.

1.4. Updates in 2020.1.1

- Support for Windows Hardware-accelerated GPU scheduling
- Support for tracking child processes spawned by the application launched under the tool via the --target-processes CLI option.

1.5. Updates in 2020.1

- Initial release of the Compute Sanitizer (with CUDA 11.0)

Updates to the Sanitizer API :

- Added support for per-thread streams
- Added APIs to retrieve the PC and size of a CUDA function or patch
- Added callback for cudaStreamAttachMemAsync
- Added direction to memcpy callback data
- Added stream to memcpy and memset callbacks data
- Added launch callback after syscall setup
- Added visibility field to allocation callback data
- Added PC argument to block entry callback
- Added incoming value to memory access callbacks
- Added threadCount to barrier callbacks
- Added cooperative group flags for barrier and function callbacks

1.6. Updates in 2019.1

- Initial release of the Compute Sanitizer API (with CUDA 10.1)

Chapter 2. KNOWN LIMITATIONS

- Applications run much slower under the Compute Sanitizer tools. This may cause some kernel launches to fail with a launch timeout error when running with the Compute Sanitizer enabled.
- Compute Sanitizer tools do not support device backtrace on Maxwell devices (SM 5.x).
- Compute Sanitizer tools do not support CUDA/Direct3D interop.
- Compute Sanitizer tools do not support CUDA/Vulkan interop.
- The memcheck tool does not support CUDA API error checking for API calls made on the GPU using dynamic parallelism.
- The racecheck, synccheck and initcheck tools do not support CUDA dynamic parallelism.
- CUDA dynamic parallelism is not supported when Windows Hardware-accelerated GPU scheduling is enabled.
- Compute Sanitizer tools do not support OptiX.
- Compute Sanitizer tools cannot interoperate with other CUDA developer tools. This includes CUDA coredumps which are automatically disabled by the Compute Sanitizer.

Chapter 3. KNOWN ISSUES

- On SM 7.0 and above, the racecheck tool does not fully support warp synchronization instructions with a partial thread mask. If such an instruction is encountered, it is handled as if the mask would have been full (i.e., 0xffffffff). As a result, checking can be too conservative at times and some potential intra-warp hazards will not be detected.
- With some versions of Windows Server 2016, programs built with some configurations might hang when used with the Compute Sanitizer. A workaround for this issue is to use the Computer Sanitizer with **`--show-backtrace device`** or **`--show-backtrace no`** options.

Chapter 4. SUPPORT

Information on supported platforms and GPUs.

4.1. Platform Support

Table 1 Platforms supported by Compute Sanitizer

Platform	Support
Windows	Yes
Linux (x86_64)	Yes
Linux (ppc64le)	Yes
Linux (aarch64sbsa)	Yes
Linux (aarch64)	No
QNX	No
MacOSX	No

4.2. GPU Support

Table 2 GPU architectures supported by Compute Sanitizer

Architecture	Support
Kepler	No
Maxwell	Yes
Pascal	Yes
Volta	Yes
Turing	Yes
Ampere	Yes

Notice

Trademarks

Copyright