COMPUTE SANITIZER

v2023.1.1 | April 2023

Release Notes

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1. RELEASE NOTES

1.1. Updates in 2023.1.1

- Fixed bug where memcheck would report out-of-bound accesses when loading user parameter values using a ternary operator.
- Fixed potential crash when using leakcheck with applications using CUBLAS.
- Fixed potential false positives when using synccheck or racecheck with applications using CUDA barriers.

1.2. Updates in 2023.1

- Added racecheck support for distributed shared memory.
- Extended stream-ordered race detection to **`cudaMemcpy`** APIs.
- Added memcheck, synccheck and patching API support for warpgroup operations.
- Added **`--coredump-name`** CLI option to set the coredump file name.
- Added support for Unicode file paths.
- Added support for OptiX 7.7.

1.3. Updates in 2022.4.1

- Fixed bug where synccheck would incorrectly report illegal instructions for code using **`cluster.sync()`** and compiled with **`--device-debug`**
- Fixed incorrect address reports in SanitizerCallbackMemcpyAsync in some specific cases, leading to potential invalid results in memcheck and racecheck.
- Fixed potential hangs and invalid results with racecheck on OptiX applications.
- Fixed potential crash or invalid results when using CUDA Lazy Module Loading with memcheck or initcheck if **`--check-device-heap`** is enabled. Lazy Module Loading will be automatically disabled in these cases.

1.4. Updates in 2022.4

- Added support for **`__nv_aligned_device_malloc`**.
- Added support for **`ldmatrix`** and **`stmatrix`** instructions.
- Added support for cache control operations when using the **`--check-cache-control`** command-line option.
- Added new command-line option **`--unused-memory-threshold`** to control the threshold for unused memory reports.
- Improved support for CUDA pipeline memcpy-async related hazards in racecheck.

1.5. Updates in 2022.3

- Added support for the NVIDIA GH100/SM 9.x GPU architecture.
- Added support for the NVIDIA AD10x/SM 8.9 GPU architecture.
- Added support for lazy kernel loading.
- Added memcheck support for distributed shared memory.
- Added new options **`--num-callers-device`** and **`--num-callers-host`** to control the number of callers to print in stack traces.
- Added support for OptiX 7.6 applications.
- Fix bug on Linux ppc64le where the host stack trace was incomplete.

1.6. Updates in 2022.2.1

- Fixed incorrect device backtrace for applications compiled with **`-lineinfo`**.

1.7. Updates in 2022.2

- Added memcheck support for use-before-alloc and use-after-free race detection. See the stream-ordered race detection documentation for more information.
- Added leakcheck support for asynchronous allocations, OptiX resources and CUDA memmap (on Linux only for the latter).
- Added option to ignore **`CUDA_ERROR_NOT_FOUND`** error codes returned by the **`cuGetProcAddress`** API.
- Added new sanitizer API functions to allocate and free page-locked host memory.
- Added sanitizer API callbacks for the event management API.

1.8. Updates in 2022.1.1

- Fixed initcheck issue where the tool would incorrectly abort a CUDA kernel launch after reporting an uninitialized access on Windows with hardware scheduling enabled.

1.9. Updates in 2022.1

- Added support for generating coredumps.
- Improved support for stack overflow detection.
- Added new option **`--target-processes-filter`** to filter the processes being tracked by name.
- Added initcheck support for asynchronous allocations. Requires CUDA driver version 510 or newer.
- Added initcheck support for accesses on peer devices. Requires CUDA driver version 510 or newer.
- Added support for OptiX 7 applications.
- Added support for tracking the child processes of 32-bit processes in multi-process applications on Linux and Windows x86_64.

1.10. Updates in 2021.3.1

- Fixed intermittent issue on vGPU where synccheck would incorrectly detect divergent threads.
- Fixed potential hang when tracking several graph launches.

1.11. Updates in 2021.3

- Improved Linux host backtrace.
- Removed requirement to call **`cudaDeviceReset()`** for accurate reporting of memory leaks and unused memory features.
- Fixed synccheck potential hang when calling **`__syncthreads`** in divergent code paths on Volta GPUs or newer.
- Added print of nearest allocation information for memcheck precise errors in global memory.
- Added warning when calling device-side **`malloc`** with an empty size.
- Added separate sanitizer API device callback for **`cuda::memcpy_async`**.
- Added new command-line option **`--num-cuda-barriers`** to override the expected number of **`cuda::barrier`** used by the target application.
- Added new command-line options **`--print-session-details`** to print session information and **`--save-session-details`** to save it to the output file.
- Added support for WSL2.

1.12. Updates in 2021.2.2

- Enabled stack canaries with random canary values for L4T builds.

1.13. Updates in 2021.2.1

- Added device backtrace for malloc/free errors in CUDA kernels.
- Improved racecheck host memory footprint.

1.14. Updates in 2021.2

- Added racecheck and synccheck support for **`cuda::barrier`** on Ampere GPUs or newer.
- Added racecheck support for **`__syncwarp`** with partial mask.
- Added **`--launch-count`** and **`--launch-skip`** filtering options. See the Command Line Options documentation for more information.
- **`--filter`** and **`--exclude`** options have been respectively renamed to **`--kernel-regex`** and **`--kernel-regex-exclude`**.
- Added support for QNX and Linux aarch64 platforms.
- Added support for CUDA graphs memory nodes.

1.15. Updates in 2021.1.1

- Fixed an issue where incorrect line numbers could be shown in errors reports.

1.16. Updates in 2021.1

- Added support for allocation padding via the **`--padding`** option.
- Added experimental support for NVTX memory API using option **`--nvtx yes`**. Please refer to NVTX API for Compute Sanitizer Reference Manual for more information.

1.17. Updates in 2020.3.1

- Fixed issue when launching a CUDA graph multiple times.
- Fixed false positives when using cooperative groups synchronization primitives with initcheck and synccheck.

1.18. Updates in 2020.3

- Added support for CUDA memory pools and CUDA API reduced serialization.
- Added host backtrace for unused memory reports.

1.19. Updates in 2020.2.1

- Fixed crash when loading cubins of size larger than 2 GiB.
- Fixed 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 sanitizer API to translate between sanitizer and CUDA stream handles.

1.20. Updates in 2020.2

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

1.21. Updates in 2020.1.2

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

1.22. Updates in 2020.1.1

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

1.23. 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

1.24. 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 device backtrace on Windows Server 2016 for devices in WDDM mode.
- Compute Sanitizer tools do not support device backtrace and coredumps on WSL2.
- 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 cannot interoperate with other CUDA developer tools. This includes CUDA coredumps which are automatically disabled by the Compute Sanitizer. They can be enabled instead by using the **`--generate-coredump`** option.
- Compute Sanitizer tools do not support IPC memory pools. Using it will result in false positives.
- Compute Sanitizer tools are not supported when SLI is enabled.
- The racecheck and synccheck tools do not support distributed shared memory.
- Compute Sanitizer tools do not support CUDA device graph launches.

Chapter 3. KNOWN ISSUES

- The racecheck tool may print incorrect data for "Current value" when reporting a hazard on a shared memory location where the last access was an atomic operation. This can also impact the severity of this hazard.
- 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.
- On QNX, when using the **`--target-processes all`** option, analyzing shell scripts may hang after the script has completed. End the application using *Ctrl-C* on the command line in that case.
- The initcheck tool might report false positives for device-to-host cudaMemcpy operations on padded structs that were initialized by a CUDA kernel. The **`#pragma pack`** directive can be used to disable the padding as a workaround.
- When a hardware exception occur during a kernel launch that was skipped due to the usage of the **`kernel-regex`**, **`kernel-regex-exclude`**, **`launch-count`** or **`launch-skip`** options, the memcheck tool will not be able to report additional details as an imprecise error. To get imprecise information about a hardware exception without running all the check, the option **`--binary-patching no`** can be used as a workaround.
- The memcheck and initcheck tools don't support CUDA lazy module loading when device heap checking is enabled. Lazy module loading will be automatically disabled when using these tools if the option **`--check-device-heap no`** was not specified.

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)	Yes
QNX	Yes
MacOSX	No

4.2. GPU Support

The compute-sanitizer tools are supported on all CUDA capable GPUs with SM versions 5.0 and above.

Notice

Trademarks

Copyright