NSIGHT SYSTEMS

v2020.5.1 | November 2020

Release Notes

TABLE OF CONTENTS

Chapter 1. WHAT'S NEW

- Thread state information collection without turning on full sampling on Linux.
- Better integration with NVIDIA Nsight Compute.
- Fortran support for MPI tracing.
- UVM transfers colored to indicate benign and blocking transfers.
- CUDA memory graph trace.
- Initial release of CLI for Windows targets.
  - DX11, DX12, Vulkan Trace.
  - Analysis start/stop control with API and hotkeys.
- Various bug fixes and performance enhancements.

Chapter 2. KNOWN ISSUES

2.1. General Issues

- Nsight Systems 2020.4 introduces collection of thread scheduling information without full sampling. While this allows system information at a lower cost, it does add overhead. To turn off thread schedule information collection add --cpuctxsw=none to your command line or turn off in the GUI
- Profiling greater than 5 minutes is not officially supported at this time. Profiling high activity applications, on high performance machines, over a long analysis time can create large result files that may take a very long time to load, run out of memory, or lock up the system. If you have a complex application, we recommend starting with a short profiling session duration of no more than 5 minutes for your initial profile. If your application has a natural repeating pattern, often referred to as a frame, you may typically only need a few of these. This suggested limit will increase in future releases.
- Attaching or re-attaching to a process from the GUI is not supported with the x86_64 Linux or IBM Power target. Equivalent results can be obtained by using the interactive CLI to launch the process and then starting and stopping analysis at multiple points.
- To reduce overhead,

  Nsight Systems traces a subset of API calls likely to impact performance when tracing APIs rather than all possible calls. There is currently no way to change the subset being traced when using the CLI. See respective library portion of this documentation for a list of calls traced by default. The CLI limitation will be removed in a future version of the product.
- There is an upper bound on the default size used by the tool to record trace events during the collection. If you see the following diagnostic error, then Nsight Systems hit the upper limit.

  ```
  Reached the size limit on recording trace events for this process.
         Try reducing the profiling duration or reduce the number of features
         traced.
  ```
- When profiling a framework or application that uses CUPTI, like some versions of TensorFlow(tm), Nsight Systems will not be able to trace CUDA usage due to

limitations in CUPTI. These limitations will be corrected in a future version of CUPTI. Consider turning off the application's use of CUPTI if CUDA tracing is required.

As an example, in the TensorFlow **`mnist_with_summaries.py`** tutorial, you will be able to use Nsight Systems to perform CUDA trace if you remove usage of **`RunOptions.FULL_TRACE`** from the code. For more information, see RunOptions documentation.

- Tracing an application that uses a memory allocator that is not thread-safe is not supported.
- Tracing OS Runtime libraries in an application that preloads glibc symbols is unsupported and can lead to undefined behavior.
- Nsight Systems cannot profile applications launched through a virtual window manager like GNU Screen.
- Using Nsight Systems MPI trace functionality with the Darshan runtime module can lead to segfaults. To resolve the issue, unload the module.

```
module unload darshan-runtime
```

- Profiling MPI Fortran APIs with MPI_Status as an argument, e.g. MPI_Recv, MPI_Test[all], MPI_Wait[all], can potentially cause memory corruption for MPICH versions 3.0.x. The reason is that the MPI_Status structure in MPICH 3.0.x has a different memory layout than in other MPICH versions (2.1.x and >=3.1.x have been tested) and the version (3.3.2) we used to compile the Nsight Systems MPI interception library.

2.2. Docker Issues

- In a Docker, when a system's host utilizes a kernel older than v4.3, it is not possible for Nsight Systems to collect sampling data unless both the host and Docker are running a RHEL or CentOS operating system utilizing kernel version 3.10.1-693 or newer. A user override for this will be made available in a future version.
- When **`docker exec`** is called on a running container and stdout is kept open from a command invoked inside that shell, the exec shell hangs until the command exits. You can avoid this issue by running with **`docker exec --tty`**. See the bug reports at:
  - https://github.com/moby/moby/issues/33039
  - https://github.com/drud/ddev/issues/732

2.3. CUDA Trace Issues

- Nsight Systems does not support tracing codes using CDP (CUDA Dynamic Parallelism).
- On Tegra platforms, CUDA trace requires root privileges. Use the **Launch as root** checkbox in project settings to make the profiled application run as root.

- If the target application uses multiple streams from multiple threads, CUDA event buffers may not be released properly. In this case, you will see the following diagnostic error:

  ```
  Couldn't allocate CUPTI bufer x times. Some CUPTI events may
         be missing.
  ```

  Please contact the Nsight Systems team.
- In this version of Nsight Systems, if you are starting and stopping profiling inside your application using the interactive CLI, the CUDA memory allocation graph generation is only guaranteed to be correct in the first profiling range. This limitation will be removed in a future version of the product.
- CUDA GPU trace collection requires a fraction of GPU memory. If your application utilizes all available GPU memory, CUDA trace might not work or can break your application. As an example cuDNN application can crash with **`CUDNN_STATUS_INTERNAL_ERROR`** error if GPU memory allocation fails.
- For older Linux kernels, prior to 4.4, when profiling very short-lived applications (~1 second) that exit in the middle of the profiling session, it is possible that Nsight Systems will not show the CUDA events on the timeline.
- When more than 64k serialized CUDA kernels and memory copies are executed in the application, you may encounter the following exception during profiling:

  ```
  InvalidArgumentException: "Wrong event order detected"
  ```

  Please upgrade to the CUDA 9.2 driver at minimum to avoid this problem. If you cannot upgrade, you can get a partial analysis, missing potentially a large fraction of CUDA events, by using the CLI.
- On Vibrante, when running a profiling session with multiple targets that are guest VMs in a CCC configuration behind a NAT, you may encounter an error with the following text during profiling:

  ```
  Failed to sync time on device.
  ```

  Please edit the group connection settings, select **Targets on the same SoC** checkbox there and try again.