NVIDIA

INSTALLATION GUIDE

v2023.3.1 | August 2023

Nsight Systems Installation Guide

TABLE OF CONTENTS

Chapter 1. OVERVIEW

Nsight Systems is a statistical sampling profiler with tracing features. It is designed to work with devices and devkits based on NVIDIA Tegra SoCs (system-on-chip), Arm SBSA (server based system architecture) systems, IBM Power systems, and systems based on the x86_64 processor architecture that also include NVIDIA GPU(s).

Throughout this document we will refer to the device on which profiling happens as the **target**, and the computer on which the user works and controls the profiling session as the **host**. Note that for x86_64 based systems these may be on the same device, whereas with Tegra, Arm, or IBM Power based systems they will always be separate.

Furthermore, three different activities are distinguished as follows:

- **Profiling** — The process of collecting any performance data. A profiling session in Nsight Systems typically includes sampling and tracing.
- **Sampling** — The process of periodically stopping the *profilee* (the application under investigation during the profiling session), typically to collect backtraces (call stacks of active threads), which allows you to understand statistically how much time is spent in each function. Additionally, hardware counters can also be sampled. This process is inherently imprecise when a low number of samples have been collected.
- **Tracing** — The process of collecting precise information about various activities happening in the profilee or in the system. For example, profilee API execution may be traced providing the exact time and duration of a function call.

Nsight Systems supports multiple generations of Tegra SoCs, NVIDIA discrete GPUs, and various CPU architectures, as well as various target and host operating systems. This documentation describes the full set of features available in any version of Nsight Systems. In the event that a feature is not available in all versions, that will be noted in the text. In general, Nsight Systems Embedded Platforms Edition indicates the package that supports Tegra processors for the embedded and automotive market and Nsight Systems Workstation Edition supports x86_64, IBM Power, and Arm server (SBSA) processors for the workstation and cluster market.

Common features that are supported by Nsight Systems on most platforms include the following:

- Sampling of the profilee and collecting backtraces using multiple algorithms (such as frame pointers or DWARF data). Building top-down, bottom-up, and flat views

as appropriate. This information helps identify performance bottlenecks in CPU-intensive code.
- Sampling or tracing system power behaviors, such as CPU frequency.
- (Only on Nsight Systems Embedded Platforms Edition)Sampling counters from Arm PMU (Performance Monitoring Unit). Information such as cache misses gets statistically correlated with function execution.
- Support for multiple windows. Users with multiple monitors can see multiple reports simultaneously, or have multiple views into the same report file.

With Nsight Systems, a user could:

- Identify call paths that monopolize the CPU.
- Identify individual functions that monopolize the CPU (across different call paths).
- For Nsight Systems Embedded Platforms Edition, identify functions that have poor cache utilization.
- If platform supports CUDA, see visual representation of CUDA Runtime and Driver API calls, as well as CUDA GPU workload. Nsight Systems uses the CUDA Profiling Tools Interface (CUPTI), for more information, see: CUPTI documentation.
- If the user annotates with NVIDIA Tools Extension (NVTX), see visual representation of NVTX annotations: ranges, markers, and thread names.
- For Windows targets, see visual representation of D3D12: which API calls are being made on the CPU, graphic frames, stutter analysis, as well as GPU workloads (command lists and debug ranges).
- For x86_64 targets, see visual representation of Vulkan: which API calls are being made on the CPU, graphic frames, stutter analysis, as well as Vulkan GPU workloads (command buffers and debug ranges).

Chapter 2. SYSTEM REQUIREMENTS

Nsight Systems supports multiple platforms. For simplicity, think of these as Nsight Systems Embedded Platforms Edition and Nsight Systems Workstation Edition, where Nsight Systems Workstation Edition supports desktops, workstations, and clusters with x86_64, IBM Power, and Arm SBSA CPUs on Linux and Windows OSs, while Nsight Systems Embedded Platforms Edition supports NVIDIA Tegra products for the embedded and gaming space on Linux for Tegra and QNX OSs.

Supported Platforms

Depending on your OS, different GPUs are supported

L4T (Linux for Tegra)

- Jetson AGX Xavier
- Jetson TX2
- Jetson TX2i
- Jetson TX
- Jetson Nano
- Jetson Xavier NX

x86_64, IBM Power (from Power 9), or Arm SBSA

- NVIDIA GPU architectures starting with Pascal
- OS (64 bit only)
  - Ubuntu 18.04, 20.04, and 22.04
  - CentOS and RedHat Enterprise Linux 7.4+ with kernel version 3.10.0-693 or later.
  - Windows 10, 11, and Win Server 2022

CUDA Version

- Nsight Systems supports CUDA 10.0, 10.1, 10.2, and 11.X for most platforms

- Nsight Systems on Arm SBSA supports 10.2 and 11.X

Note that CUDA version and driver version must be compatible.

CUDA Version	Driver minimum version
11.0	450
10.2	440.30
10.1	418.39
10.0	410.48

From CUDA 11.X on, any driver from 450 on will be supported, although new features introduced in more recent drivers will not be available.

For information about which drivers were specifically released with each toolkit, see CUDA Toolkit Release Notes - Major Component Versions

Requirements for x86_64, Power, and Arm SBSA Targets on Linux

When attaching to x86_64, Power, or Arm SBSA Linux-based target from the GUI on the host, the connection is established through SSH.

Use of Linux Perf: To collect thread scheduling data and IP (instruction pointer) samples, the Linux operating system's perf_event_paranoid level must be 2 or less. Use the following command to check:

```
cat /proc/sys/kernel/perf_event_paranoid
```

If the output is >2, then do the following to temporarily adjust the paranoid level (note that this has to be done after each reboot):

```
sudo sh -c 'echo 2 >/proc/sys/kernel/
perf_event_paranoid'
```

To make the change permanent, use the following command:

```
sudo sh -c 'echo kernel.perf_event_paranoid=2
 > /etc/sysctl.d/local.conf'
```

Kernel version: To collect thread scheduling data and IP (instruction pointer) samples and backtraces, the kernel version must be:

- 3.10.0-693 or later for CentOS and RedHat Enterprise Linux 7.4+

- 4.3 or greater for all other distros including Ubuntu

To check the version number of the kernel on a target device, run the following command on the device:

```
uname -a
```

Note that only CentOS, RedHat, and Ubuntu distros are tested/confirmed to work correctly.

glibc version: To check the glibc version on a target device, run the following command:

```
ldd --version
```

Nsight Systems requires glibc 2.17 or more recent.

CUDA: See above for supported CUDA versions in this release. Use the deviceQuery command to determine the CUDA driver and runtime versions on the system. the deviceQuery command is available in the CUDA SDK. It is normally installed at:

```
/usr/local/cuda/samples/1_Utilities/
deviceQuery
```

Only pure 64-bit environments are supported. In other words, 32-bit systems or 32-bit processes running within a 64-bit environment are not supported.

Nsight Systems requires write permission to the **`/var/lock`** directory on the target system.

Docker: See Collecting Data within a Docker section of the User Guide for more information.

x86_64 Windows Target Device Requirements

DX12 Requires:

- Windows 10 with NVIDIA Driver 411.63 or higher for DX12 trace
- Windows 10 April 2018 Update (version 1803, AKA Redstone 4) with NVIDIA Driver 411.63 or higher for DirectX Ray Tracing, and tracing DX12 Copy command queues.

Host Application Requirements

The Nsight Systems host application runs on the following host platforms:

- Windows 10, Windows Server 2019. Only 64-bit versions are supported.
- Linux Ubuntu 14.04 and higher are known to work, running on other modern distributions should be possible as well. Only 64-bit versions are supported.
- OS X 10.10 "Yosemite" and higher.

Chapter 3. GETTING STARTED GUIDE

3.1. Finding the Right Package

Nsight Systems is available for multiple targets and multiple host OSs. To choose the right package, first consider the target system to be analyzed.

- For Tegra target systems, select Nsight Systems for Tegra available as part of NVIDIA JetPack SDK.
- For x86_64, IBM Power target systems, or Arm SBSA select from the target packages from Nsight Systems for Workstations, available from https://developer.nvidia.com/nsight-systems. This web release will always contain the latest and greatest Nsight Systems features.
- The x86_64, IBM Power, and Arm SBSA target versions of Nsight Systems are also available in the CUDA Toolkit.

Each package is limited to one architecture. For example, Tegra packages do not contain support for profiling x86 targets, and x86 packages do not contain support for profiling Tegra targets.

After choosing an appropriate target version, select the package corresponding to the host OS, the OS on the system where results will be viewed. These packages are in the form of common installer types: .msi for Windows; .run, .rpm, and .deb for x86 Linux; .deb and .rpm for Linux on IBM Power; and .dmg for the macOS installer.

Note: the IBM Power and Arm SBSA packages do not have a GUI for visualization of the result. If you wish to visualize your result, please download and install the GUI available for macOS, x86_64 Linux, or Windows systems.

Tegra packages

- Windows host – Install .msi on Windows machine. Enables remote access to Tegra device for profiling.
- Linux host – Install .run on Linux system. Enables remote access to Tegra device for profiling.
- macOS host – Install .dmg on macOS machine. Enables remote access to Tegra device for profiling.

x86_64 packages

- Windows host – Install .msi on Windows machine. Enables remote access to Linux x86_64 or Windows devices for profiling as well as running on local system.
- Linux host – Install .run, .rpm, or .deb on Linux system. Enables remote access to Linux x86_64 or Windows devices for profiling or running collection on localhost.
- Linux CLI only – The Linux CLI is shipped in all x86 packages, but if you just want the CLI, we have a package for that. Install .deb on Linux system. Enables only CLI collection, report can be imported or opened in x86_64 host.
- macOS host – Install .dmg on macOS machine. Enables remote access to Linux x86_64 device for profiling.

IBM Power packages

- Power CLI only - The IBM Power support does not include a host GUI. Install .deb or .rpm on your Power system. Enables only CLI collection, report can be imported or opened in GUI on any supported host platform.

Arm SBSA packages

- Arm SBSA CLI only - Arm SBSA support does not include a host GUI. Install .deb or .rpm on your Arm SBSA system. Enables only CLI collection, report can be imported or opened in GUI on any supported host platform.

3.2. Installing GUI on the Host System

Copy the appropriate file to your host system in a directory where you have write and execute permissions. Run the install file, accept the EULA, and Nsight Systems will install on your system.

On Linux, there are special options to enable automated installation. Running the installer with the **`--accept`** flag will automatically accept the EULA, running with the **`--accept`** flag and the **`--quiet`** flag will automatically accept the EULA without printing to stdout. Running with **`--quiet`** without **`--accept`** will display an error.

The installation will create a Host directory for this host and a Target directory for each target this Nsight Systems package supports.

All binaries needed to collect data on a target device will be installed on the target by the host on first connection to the device. There is no need to install the package on the target device.

If installing from the CUDA Toolkit, see the CUDA Toolkit documentation.

3.3. Optional: Setting up the CLI

All Nsight Systems targets can be profiled using the CLI. IBM Power and Arm SBSA targets can only be profiled using the CLI. The CLI is especially helpful when scripts are used to run unattended collections or when access to the target system via ssh is not possible. In particular, this can be used to enable collection in a Docker container.

The CLI can be found in the Target directory of the Nsight Systems installation. Users who want to install the CLI as a standalone tool can do so by copying the files within the Target directory to the location of their choice.

If you wish to run the CLI without root (recommended mode) you will want to install in a directory where you have full access.

Once you have the CLI set up, you can use the **`nsys status -e`** command to check your environment.

```
~$ nsys status -e

Sampling Environment Check
Linux Kernel Paranoid Level = 1: OK
Linux Distribution = Ubuntu
Linux Kernel Version = 4.15.0-109-generic: OK
Linux perf_event_open syscall available: OK
Sampling trigger event available: OK
Intel(c) Last Branch Record support: Available
Sampling Environment: OK
```

This status check allows you to ensure that the system requirements for CPU sampling using Nsight Systems are met in your local environment. If the Sampling Environment is not OK, you will still be able to run various trace operations.

Intel(c) Last Branch Record allows tools, including Nsight Systems to use hardware to quickly get limited stack information. Nsight Systems will use this method for stack resolution by default if available.

For information about changing these environment settings, see System Requirements section in the Installation Guide. For information about changing the backtrace method, see Profiling from the CLI in the User Guide.

To get started using the CLI, run **`nsys --help`** for a list of options or see Profiling Applications from the CLI in the User Guide for full documentation.

3.4. Launching the GUI

Depending on your OS, Nsight Systems will have installed an icon on your host desktop that you can use to launch the GUI. To launch the GUI directly, run the **`nsight-sys`** executable in the Host sub-directory of your installation.

3.5. Installing Multi Report Analysis System

PREVIEW FEATURE

The Nsight Systems multi-report analysis system can be located in the **`<install-dir>/target-linux-x64/python/packages`** directory. For this initial preview release, multi-node analysis is only available to run recipes on Linux targets, and only available to visualize on Linux or Windows hosts.

Recipe Dependencies

The system is written in Python and depends on a set of Python packages. The prerequisites are Python 3.6 or newer with pip and venv. If you don't have Python, you can install it from python.org or your Linux package manager.

Pip/venv on Ubuntu

If pip/venv were not installed with Python, run:

```
        $ sudo apt-get install python3-pip
        $ sudo apt-get install python3-venv
```

On a fresh Ubuntu install, we will need to run the following before the above commands:

```
        $ sudo apt-get update
```

The dependent packages can either be installed automatically by an automated script or manually.

Automated script

The **`<install-dir>/target-linux-x64/python/packages/nsys_recipe/install.py`** script automates the installation of the recipe dependencies. You must select either the **`--current`** or **`--venv PATH`** option when you run the script.

Options:

- -h: Display help
- --current: Install packages in the current environment. If a venv is active, packages will be installed there. Otherwise, packages will be\ installed in the system site-packages directory. It enables usage of **`nsys recipe`** without having to source a virtual environment. However, new packages risk colliding with existing ones if different versions are required.
- --venv PATH: Install packages in a virtual environment. If it doesn't already exist, it is created. It prevents risk of package collision in the current environment but requires the virtual environment to be activated before running **`nsys recipe`**.
- --tar: download wheel packages online and tar them
- --untar: untar the wheel packages and install
- --python: change the python executable (default is python3)
- --no-jupyter: do not install requirements for the jupyter notebook
- --no-dask: do not install requirements for Dask

If --tar or --untar option wasn't specified, the script will directly download the pip packages from the internet.

Manual steps

If you would rather install the dependencies manually, please follow the following steps:

- Create a virtual environment

  We recommend creating a virtual environment to avoid installing packages directly into your system Python. The commands create the virtual environment in the current working directory.

See venv - python doc

To create a venv named recipe_env:

```
            $ python3 -m venv recipe_env
            $ source recipe_env/bin/activate
```

- List of dependencies

  We have three files located in **`<install-dir>/target-linux-x64/python/packages/nsys_recipe/requirements`** for the dependencies:

  - Common.txt (required): dependencies needed by all recipes
  - Dask.txt (optional): dependencies needed by the Dask mode
  - Jupyter.txt (optional): dependencies needed to open the Jupyter notebook

One-step installation

The following command will install all dependencies for CLI and GUI. Please note that you will want to activate your venv first as described above, otherwise the modules will not be available in the venv.

```
        $ python3 -m pip install -r nsys_recipe/requirements/dask.txt -r
nsys_recipe/requirements/common.txt -r nsys_recipe/requirements/jupyter.txt
```

Two-step installation (for machines without internet)

If you wish to download the dependencies on a machine without internet, you can download the wheel packages on a machine with internet, transfer them to the target machine and install the packages there.

On the machine with internet:

```
        $  python3 -m pip download -r nsys_recipe/requirements/dask.txt -r
nsys_recipe/requirements/common.txt -r nsys_recipe/requirements/jupyter.txt -d
“recipe-deps”
        $ tar -cvfz recipe-deps.tar.gz recipe-deps
```

On the machine with no internet:

```
        $ tar -xvfz recipe-deps.tar.gz
        $ python3 -m pip install recipe-deps/* --no-index
```

Jupyter Notebook

The Nsight Systems UI has the ability to internally load a Jupyter notebook. It uses the Jupyter notebook installation associated with the Python on your $PATH, which is expected to be the Python installed into the virtual environment created in the earlier steps of this guide.

If Jupyter is installed in a different location, you can add a third variable to the config.ini file that will override the default path to Jupyter:

```
        JupyterPythonExe=“/path/to/recipe_env/bin/python”
```

This config.ini file should be placed in **`<install_dir>/host-linux-x64`**

Note that on Windows, the path should use Windows slashes and they must be double slashes:

```
        JupyterPythonExe="c:\\path\\to\\recipe_env\\bin\\python.exe"
```