



# NSIGHT COMPUTE

v2019.5.1 | May 2020

**User Manual**



# TABLE OF CONTENTS

<b>Chapter 1. Introduction</b> .....	<b>1</b>
1.1. Overview.....	1
<b>Chapter 2. Quickstart</b> .....	<b>2</b>
2.1. Interactive Profile Activity.....	3
2.2. Non-Interactive Profile Activity.....	7
2.3. Navigate the Report.....	9
<b>Chapter 3. Connection Dialog</b> .....	<b>12</b>
3.1. Remote Connections.....	13
3.2. Interactive Profile Activity.....	16
3.3. Profile Activity.....	17
<b>Chapter 4. Main Menu and Toolbar</b> .....	<b>19</b>
4.1. Main Menu.....	19
4.2. Main Toolbar.....	21
<b>Chapter 5. Tool Windows</b> .....	<b>22</b>
5.1. API Statistics.....	22
5.2. API Stream.....	23
5.3. NVTX.....	24
5.4. Resources.....	25
5.5. Sections/Rules Info.....	25
<b>Chapter 6. Profiler Report</b> .....	<b>27</b>
6.1. Header.....	27
6.2. Report Pages.....	28
6.2.1. Session Page.....	28
6.2.2. Summary Page.....	28
6.2.3. Details Page.....	28
6.2.4. Source Page.....	29
6.2.5. Comments Page.....	31
6.2.6. NVTX Page.....	31
6.2.7. Raw Page.....	31
6.3. Metrics and Units.....	31
<b>Chapter 7. Baselines</b> .....	<b>33</b>
<b>Chapter 8. Options</b> .....	<b>35</b>
8.1. Profile.....	36
8.2. Environment.....	37
8.3. Connection.....	37
8.4. Source Lookup.....	38
8.5. Send Feedback.....	38
<b>Chapter 9. Projects</b> .....	<b>39</b>
9.1. Project Dialogs.....	39
9.2. Project Explorer.....	39

<b>Chapter 10. Visual Profiler Transition Guide.....</b>	<b>41</b>
10.1. Trace.....	41
10.2. Sessions.....	41
10.3. Timeline.....	42
10.4. Analysis.....	42
10.5. Command Line Arguments.....	43
<b>Chapter 11. Visual Studio Integration Guide.....</b>	<b>44</b>
11.1. Visual Studio Integration Overview.....	44
<b>Chapter 12. Library Support.....</b>	<b>45</b>
12.1. OptiX.....	45
<b>Appendix A. Statistical Sampler.....</b>	<b>46</b>
<b>Appendix B. Sections and Rules.....</b>	<b>49</b>

## LIST OF TABLES

Table 1	NVIDIA Nsight Compute Profile Options .....	36
Table 2	NVIDIA Nsight Compute Environment Options .....	37
Table 3	NVIDIA Nsight Compute Connection Options .....	37
Table 4	NVIDIA Nsight Compute Source Lookup Options .....	38
Table 5	NVIDIA Nsight Compute Send Feedback Options .....	38
Table 6	Warp Scheduler States .....	46
Table 7	Available Sections .....	49

# Chapter 1.

## INTRODUCTION

For users migrating from Visual Profiler to NVIDIA Nsight Compute, please see the [Visual Profiler Transition Guide](#) for comparison of features and workflows.

### 1.1. Overview

This document is a user guide to the next-generation NVIDIA Nsight Compute profiling tools. NVIDIA Nsight Compute is an interactive kernel profiler for CUDA applications. It provides detailed performance metrics and API debugging via a user interface and command line tool. In addition, its baseline feature allows users to compare results within the tool. NVIDIA Nsight Compute provides a customizable and data-driven user interface and metric collection and can be extended with analysis scripts for post-processing results.

#### Important Features

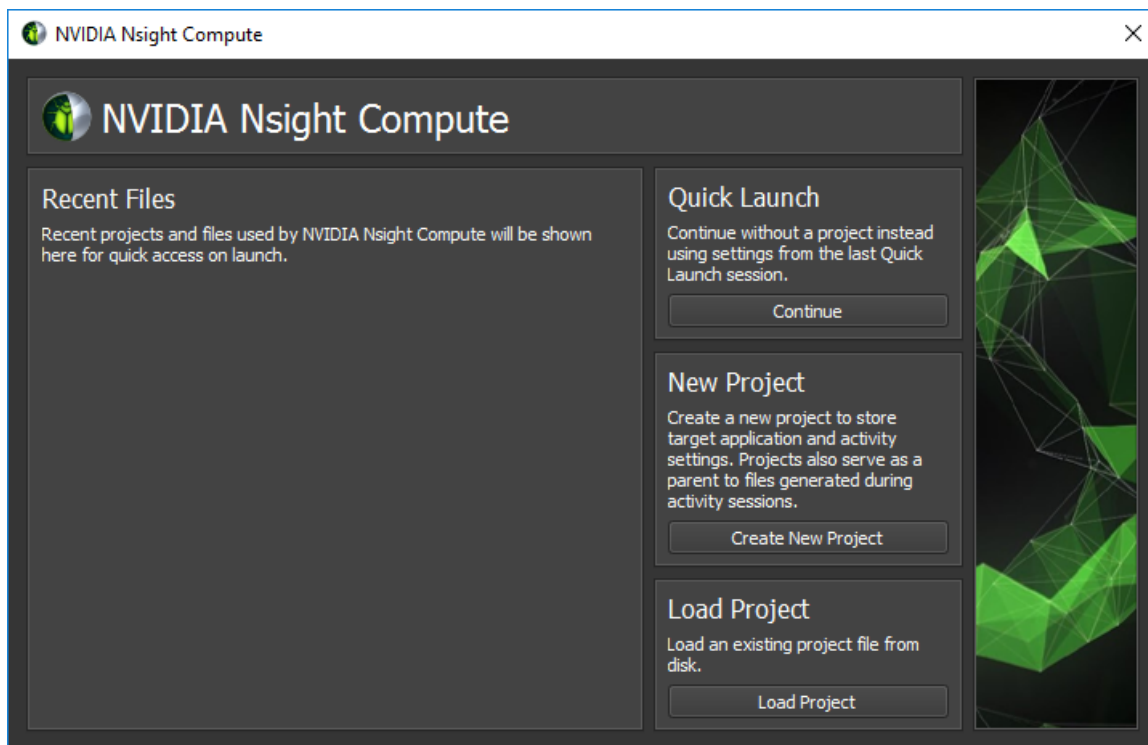
- ▶ Interactive kernel profiler and API debugger
- ▶ Graphical profile report
- ▶ Result comparison across one or multiple reports within the tool
- ▶ Fast Data Collection
- ▶ UI and Command Line interface
- ▶ Fully customizable reports and analysis rules

## Chapter 2. QUICKSTART

The following sections provide brief step-by-step guides of how to setup and run NVIDIA Nsight Compute to collect profile information. All directories are relative to the base directory of NVIDIA Nsight Compute, unless specified otherwise.

The UI executable is called `nv-nsight-cu`. A shortcut with this name is located in the base directory of the NVIDIA Nsight Compute installation. The actual executable is located in the folder `host\windows-desktop-win7-x64` on Windows or `host/linux-desktop-glibc_2_11_3-x64` on Linux. By default, when installing from a Linux `.run` file, NVIDIA Nsight Compute is located in `/usr/local/cuda-<cuda-version>/nsight-compute-<version>`. When installing from a `.deb` or `.rpm` package, it is located in `/opt/nvidia/nsight-compute/<version>` to be consistent with [Nsight Systems](#). In Windows, the default path is `C:\Program Files\NVIDIA Corporation\Nsight Compute <version>`.

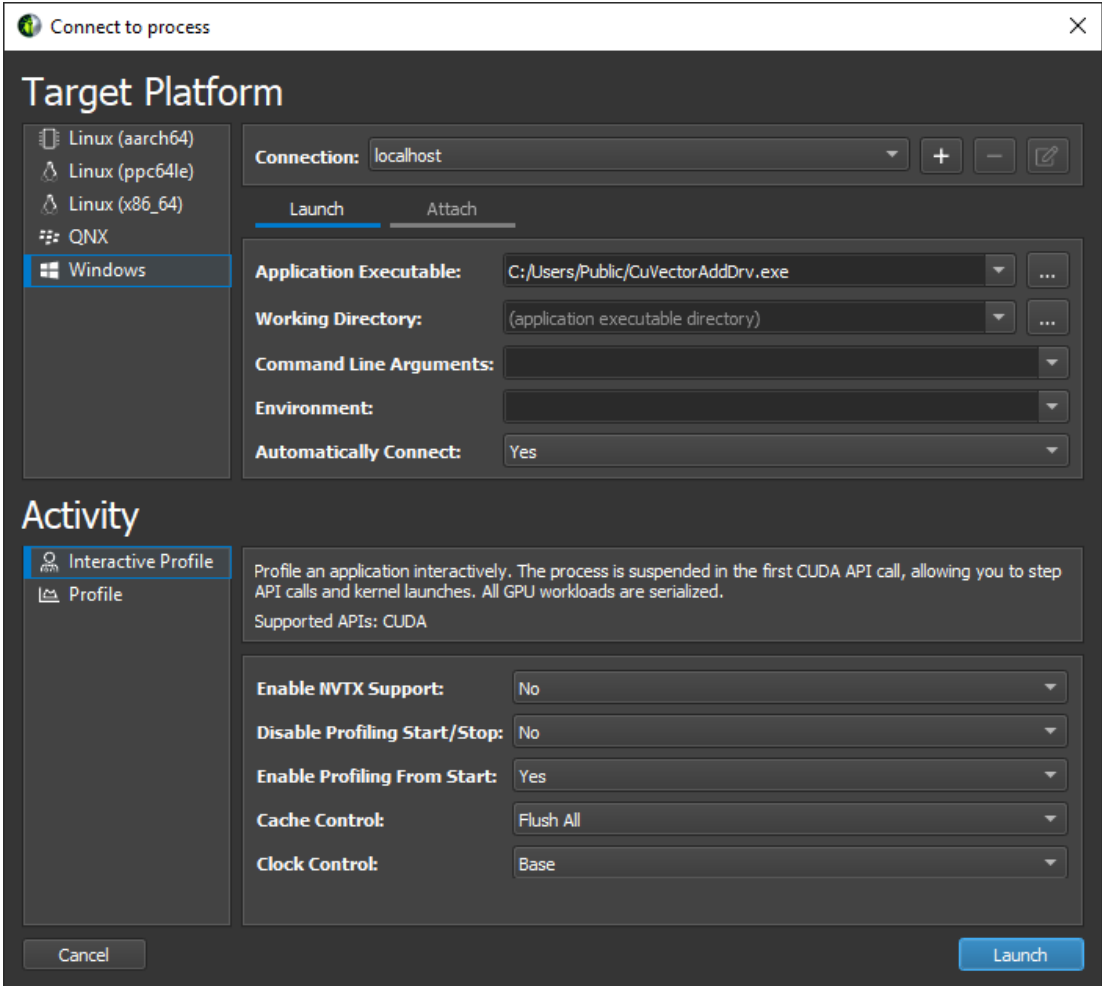
After starting NVIDIA Nsight Compute, by default the *Welcome Page* is opened. It provides links to recently opened reports and projects as well as quick access to the [Connection Dialog](#), and the [Projects](#) dialogs. To immediately start a profile run, select *Continue* under *Quick Launch*. See [Environment](#) on how to change the start-up action.



## 2.1. Interactive Profile Activity

### 1. Launch the target application from NVIDIA Nsight Compute

When starting NVIDIA Nsight Compute, the *Welcome Page* will appear. Click on *Quick Launch* to open the *Connection* dialog. If the *Connection* dialog doesn't appear, you can open it using the *Connect* button from the main toolbar, as long as you are not currently connected. Select your target platform on the left-hand side and your connection target (machine) from the *Connection* drop down. If you have your local target platform selected, `localhost` will become available as a connection. Use + to add a new connection target. Then, fill in the launch details and select *Launch*. In the *Activity* panel, select the Interactive Profile activity to initiate a session that allows controlling the execution of the target application and selecting the kernels of interest interactively. Press *Launch* to start the session.



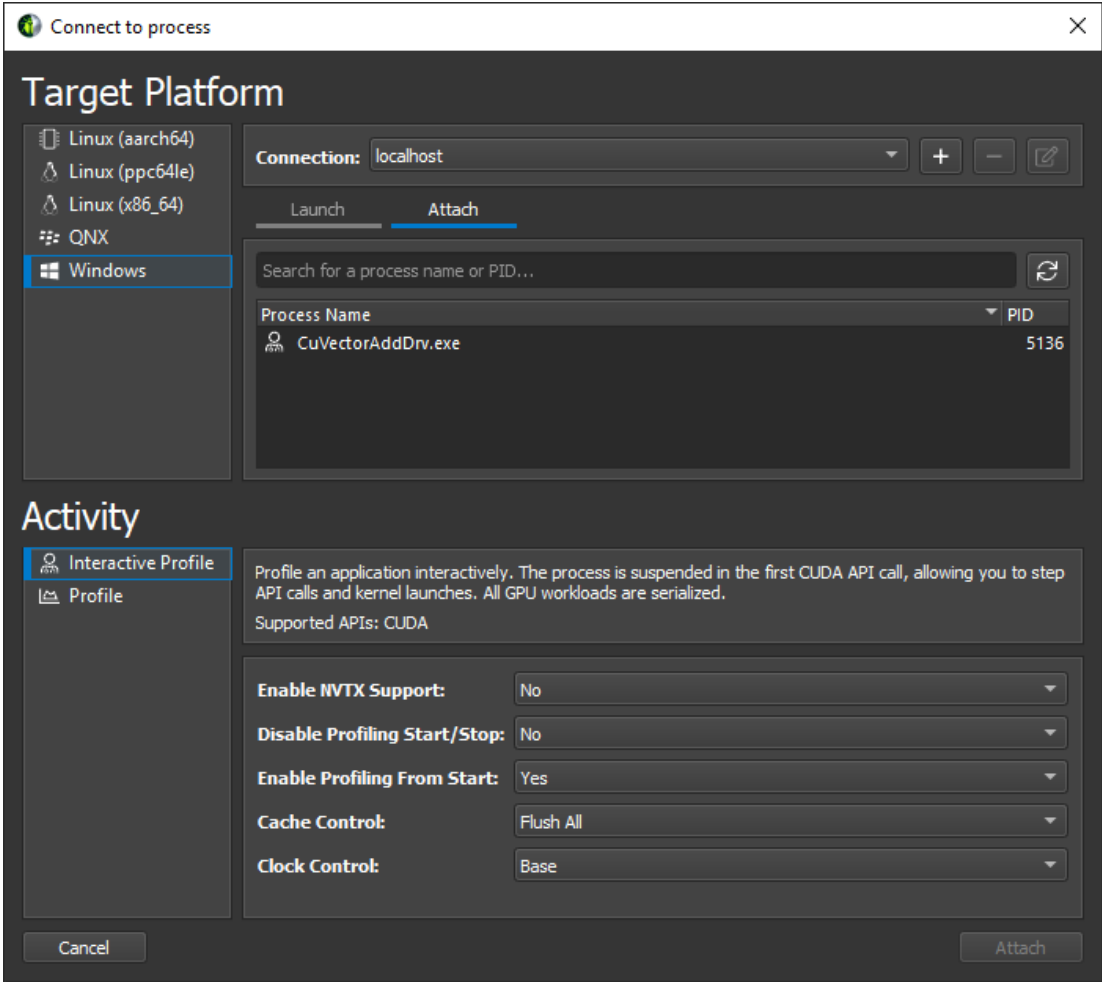
2. Launch the target application with tools instrumentation from the command line

The `nv-nsight-cu-cli` can act as a simple wrapper that forces the target application to load the necessary libraries for tools instrumentation. The parameter `--mode=launch` specifies that the target application should be launched and suspended before the first instrumented API call. That way the application waits until we connect with the UI.

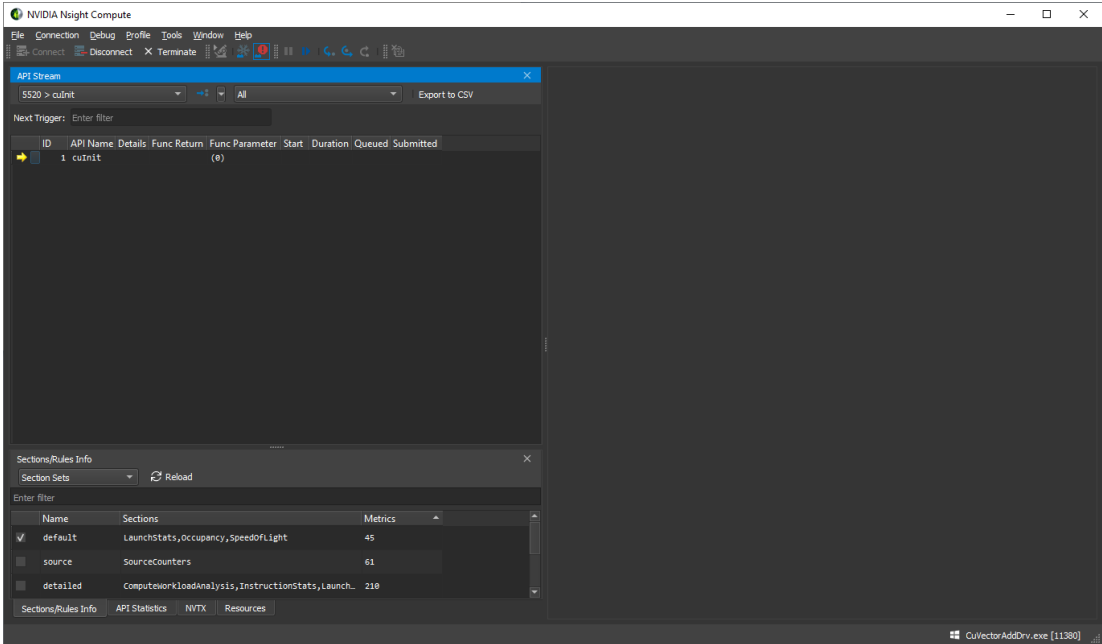
```
$ nv-nsight-cu-cli --mode=launch CuVectorAddDrv.exe
```

3. Launch NVIDIA Nsight Compute and connect to target application



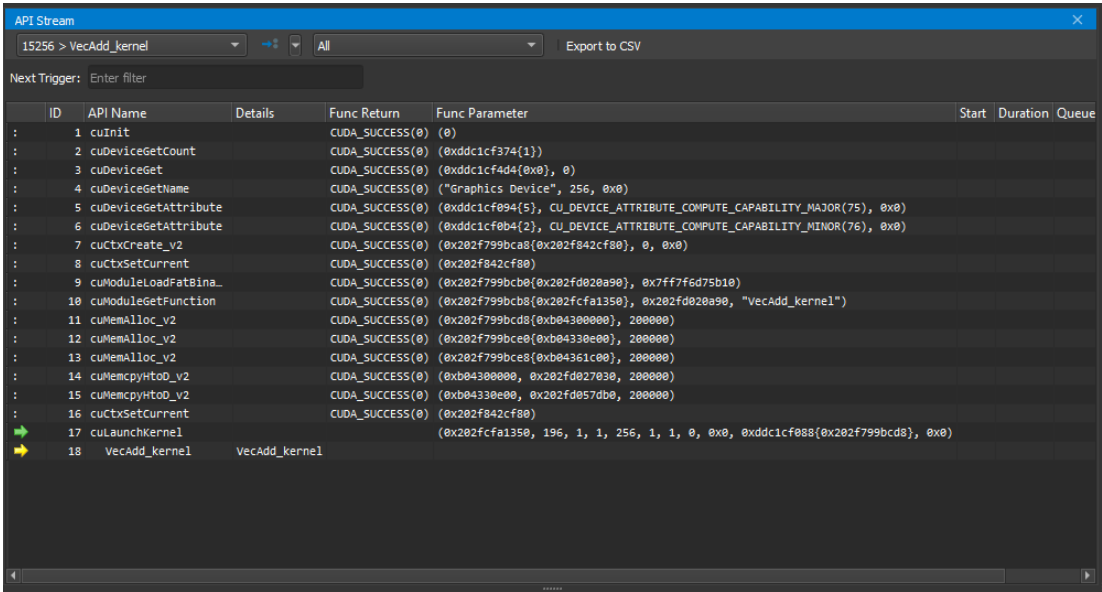


Select the target machine at the top of the dialog to connect and update the list of attachable applications. By default, *localhost* is pre-selected if the target matches your current local platform. Select the *Attach* tab and the target application of interest and press *Attach*. Once connected, the layout of NVIDIA Nsight Compute changes into stepping mode that allows you to control the execution of any calls into the instrumented API. When connected, the *API Stream* window indicates that the target application waits before the very first API call.



#### 4. Control application execution

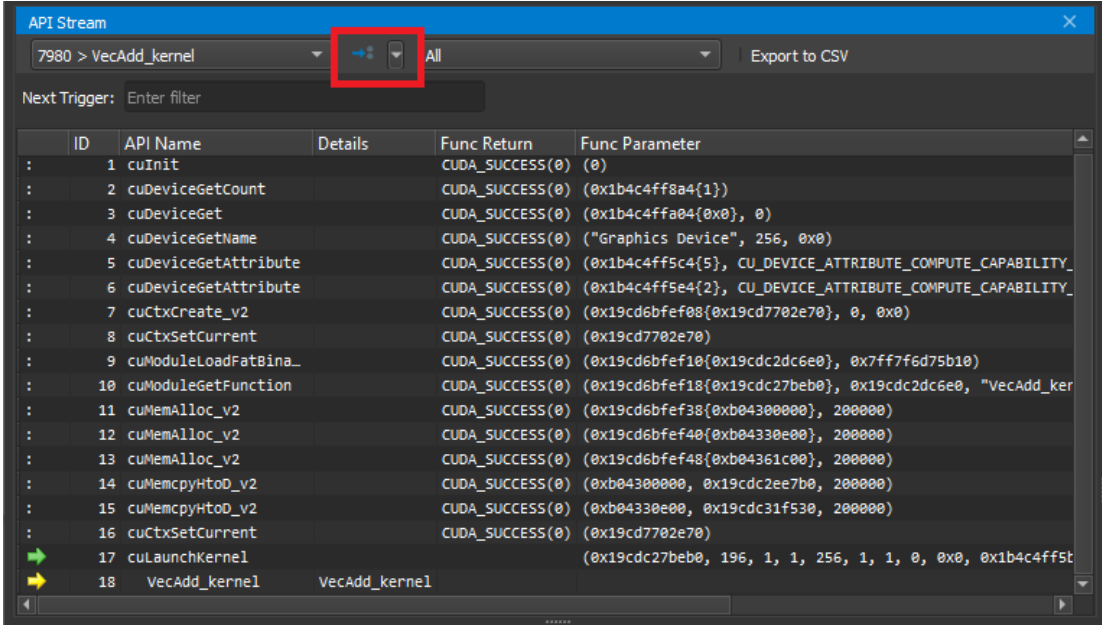
Use the *API Stream* window to step the calls into the instrumented API. The dropdown at the top allows switching between different CPU threads of the application. *Step In* (F11), *Step Over* (F10), and *Step Out* (Shift + F11) are available from the *Debug* menu or the corresponding toolbar buttons. While stepping, function return values and function parameters are captured.



Use *Resume* (F5) and *Pause* to allow the program to run freely. Freeze control is available to define the behavior of threads currently not in focus, i.e. selected in the thread drop down. By default, the *API Stream* stops on any API call that returns an error code. This can be toggled in the *Debug* menu by *Break On API Error*.

#### 5. Isolate a kernel launch

To quickly isolate a kernel launch for profiling, use the *Next API Launch* button in the toolbar of the *API Stream* window to jump to the next kernel launch. The execution will stop before the kernel launch is executed.



6. Profile a kernel launch

Once the execution of the target application is suspended at a kernel launch, additional actions become available in the UI. These actions are either available from the menu or from the toolbar. Please note that the actions are disabled, if the API stream is not at a qualifying state (not at a kernel launch or launching on an unsupported GPU). To profile, press *Profile* and wait until the result is shown in the *Profiler Report*. Profiling progress is reported in the lower right corner status bar.

Instead of manually selecting *Profile*, it is also possible to enable *Auto Profile* from the *Profile* menu. If enabled, each kernel matching the current kernel filter (if any) will be profiled using the current section configuration. This is especially useful if an application is to be profiled unattended, or the number of kernel launches to be profiled is very large. Sections can be enabled or disabled using the *Sections/Rules Info* tool window.

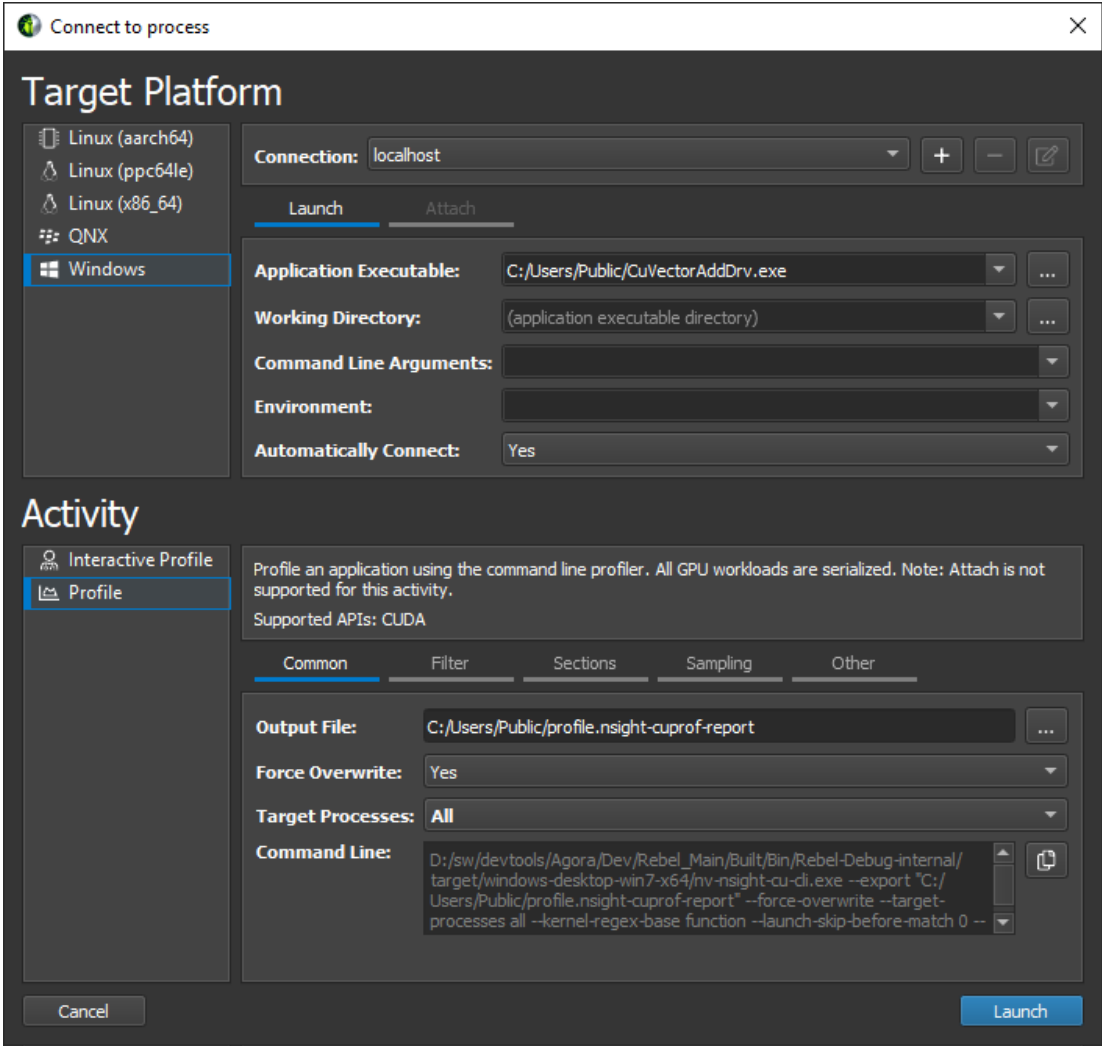
For a detailed description of the options available in this activity, see *Interactive Profile Activity*.

## 2.2. Non-Interactive Profile Activity

1. Launch the target application from NVIDIA Nsight Compute

When starting NVIDIA Nsight Compute, the *Welcome Page* will appear. Click on *Quick Launch* to open the *Connection* dialog. If the *Connection* dialog doesn't appear, you can open it using the *Connect* button from the main toolbar, as long as you are not currently connected. Select your target platform on the left-hand side and your localhost from the *Connection* drop down. Then, fill in the launch details and select

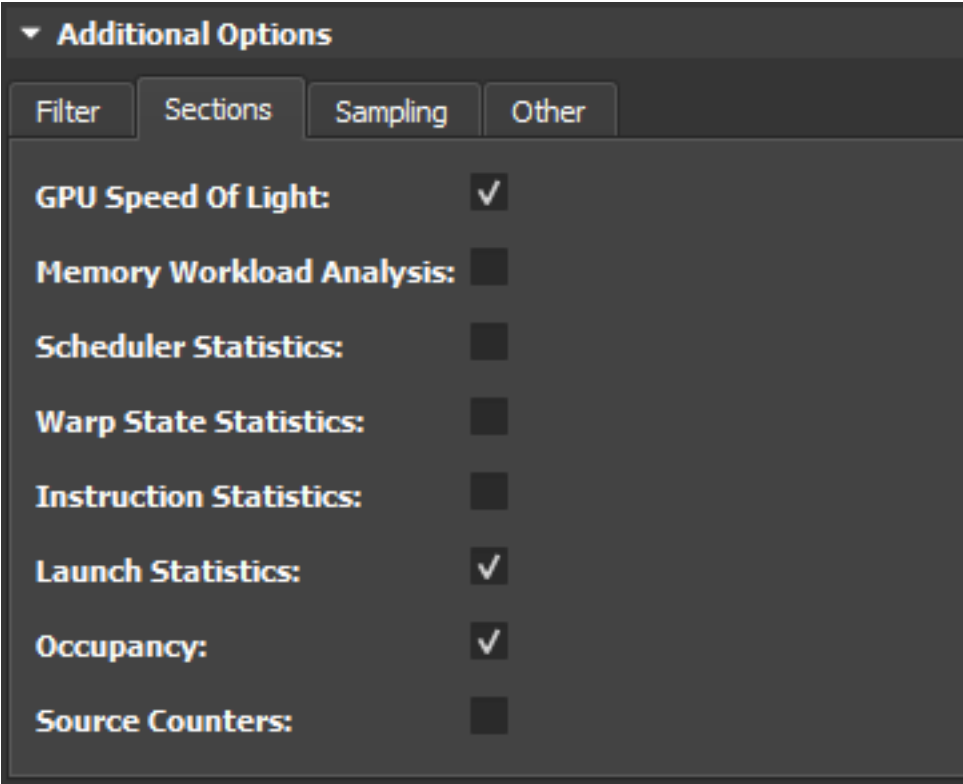
*Launch.* In the *Activity* panel, select the *Profile* activity to initiate a session that pre-configures the profile session and launches the command line profiler to collect the data. Provide the *Output File* name to enable starting the session with the *Launch* button.



2. Additional Launch Options

For more details on these options see the Command Line Options of the command line profiler. The options are grouped into tabs: The *Filter* tab exposes the options to specify which kernels should be profiled. Options include the kernel regex filter, the number of launches to skip, and the total number of launch to profile. The *Section* tab allows you to select which sections should be collected for each kernel launch. The *Sampling* tab allows you to configure sampling options for each kernel launch. The *Other* tab includes the option to collect NVTX information or custom metrics via the `--metrics` option.

The *Section* tab allows you to select which sections should be collected for each kernel launch. Hover over a section to see its description as a tool-tip. To change the sections that are enabled by default, use the [Sections/Rules Info](#) tool window.

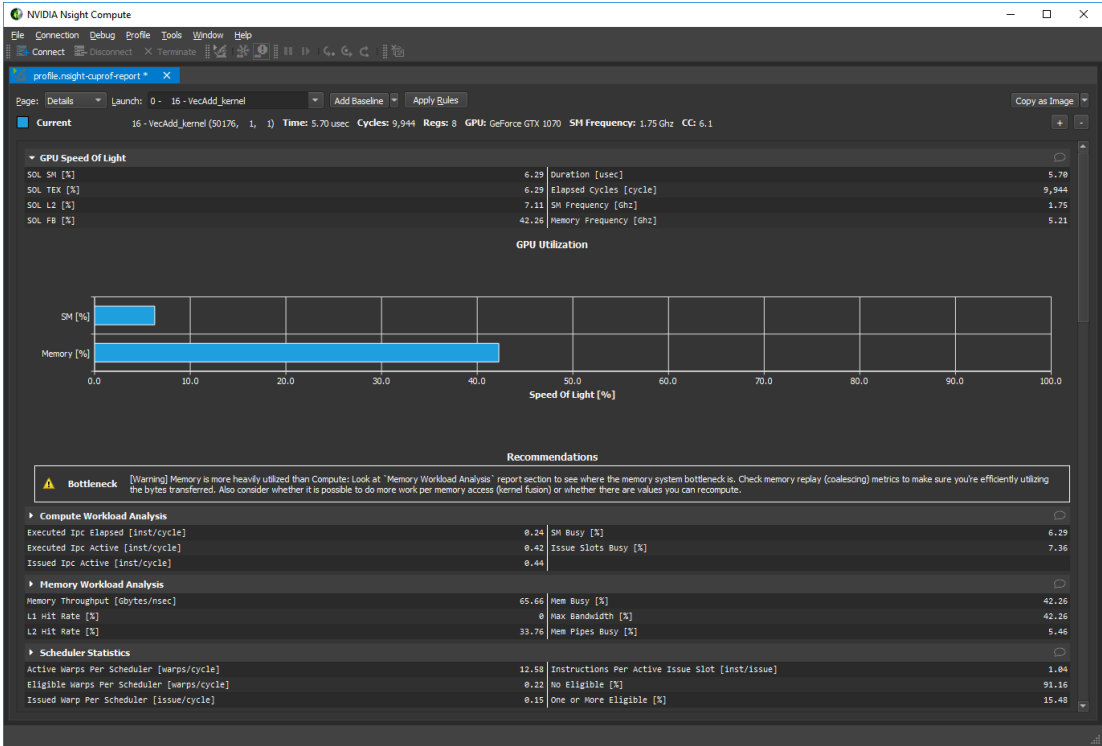


For a detailed description of the options available in this activity, see [Profile Activity](#).

## 2.3. Navigate the Report

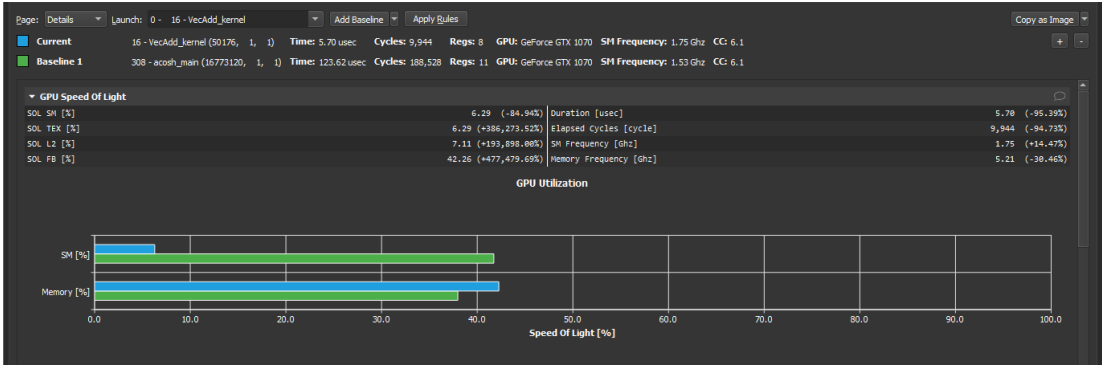
### 1. Navigate the report

The profile report comes up by default on the *Details* page. You can switch between different [Report Pages](#) of the report with the dropdown labeled Page on the top-left of the report. A report can contain any number of results from kernel launches. The *Launch* dropdown allows switching between the different results in a report.



### 2. Diffing multiple results

On the *Details* page, press the button *Add Baseline* to promote the current result in focus to become the baseline all other results from this report and any other report opened in the same instance of NVIDIA Nsight Compute gets compared to. If a baseline is set, every element on the Details page shows two values: The current value of the result in focus and the corresponding value of the baseline or the percentage of change from the corresponding baseline value.



Use the *Clear Baselines* entry from the dropdown button, the Profile menu or the corresponding toolbar button to remove all baselines. For more information see [Baselines](#).

### 3. Executing rules

On the *Details* page some sections may provide rules. Press the *Apply* button to execute an individual rule. The *Apply Rules* button on the top executes all available

rules for the current result in focus. Rules can be user-defined too. For more information see the *Customization Guide*.

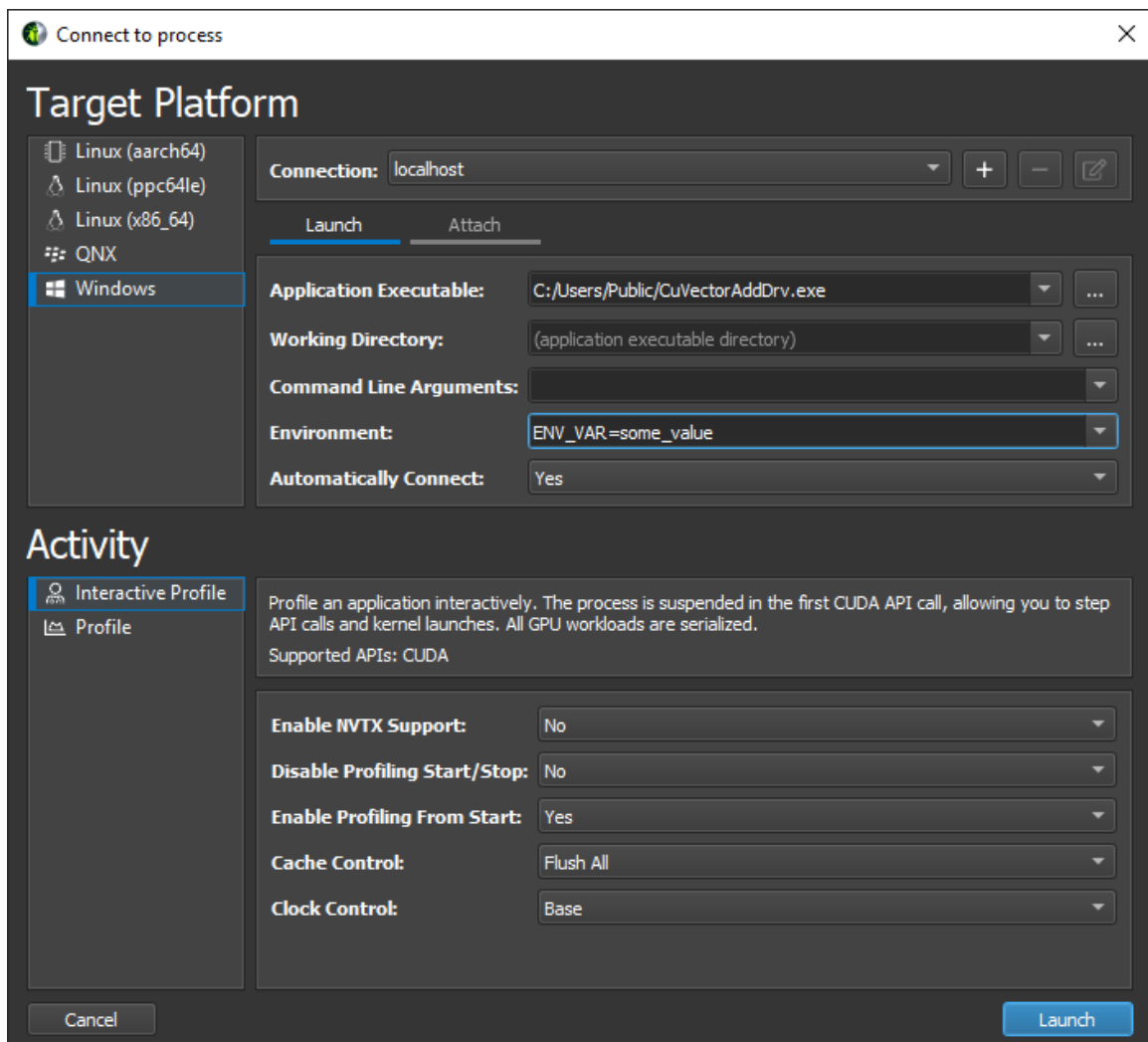
**Recommendations****Bottleneck**

[Warning] Memory is more heavily utilized than Compute: Look at 'Memory Workload Analysis' report section to see where the memory system bottleneck is. Check memory replay (coalescing) metrics to make sure you're efficiently utilizing the bytes transferred. Also consider whether it is possible to do more work per memory access (kernel fusion) or whether there are values you can recompute.

# Chapter 3.

## CONNECTION DIALOG

Use the *Connection Dialog* to launch and attach to applications on your local and remote platforms. Start by selecting the *Target Platform* for profiling. By default (and if supported) your local platform will be selected. Select the platform on which you would like to start the target application or connect to a running process.





When using a remote platform, you will be asked to select or create a *Connection* in the top drop down. To create a new connection, select + and enter your connection details. When using the local platform, *localhost* will be selected as the default and no further connection settings are required. You can still create or select a remote connection, if profiling will be on a remote system of the same platform.

Depending on your target platform, select either *Launch* or *Remote Launch* to launch an application for profiling on the target. Note that *Remote Launch* will only be available if supported on the target platform.

Fill in the following launch details for the application:

- ▶ **Application Executable:** Specifies the root application to launch. Note that this may not be the final application that you wish to profile. It can be a script or launcher that creates other processes.
- ▶ **Working Directory:** The directory in which the application will be launched.
- ▶ **Command Line Arguments:** Specify the arguments to pass to the application executable.
- ▶ **Environment:** The environment variables to set for the launched application.
- ▶ **Automatically Connect :** Specifies whether NVIDIA Nsight Compute should automatically connect to the launched application. If the launched application is a script or launcher that creates other processes that you ultimately wish to analyze, set this to 'No'.

Select *Attach* to attach the profiler to an application already running on the target platform. This application must have been started using another NVIDIA Nsight Compute CLI instance. The list will show all application processes running on the target system which can be attached. Select the refresh button to re-create this list.

Finally, select the *Activity* to be run on the target for the launched or attached application. Note that not all activities are necessarily compatible with all targets and connection options. Currently, the following activities exist:

- ▶ [Interactive Profile Activity](#)
- ▶ [Profile Activity](#)

## 3.1. Remote Connections

Remote devices that support SSH can also be configured as a target in the *Connection Dialog*. To configure a remote device, ensure an SSH-capable *Target Platform* is selected, then press the + button. The following configuration dialog will be presented.

SSH

**Authentication Mode:**  Password  Private Key

**IP/Host Name:** 192.168.1.1

**User Name:**

**Password:**

**Port:** 22

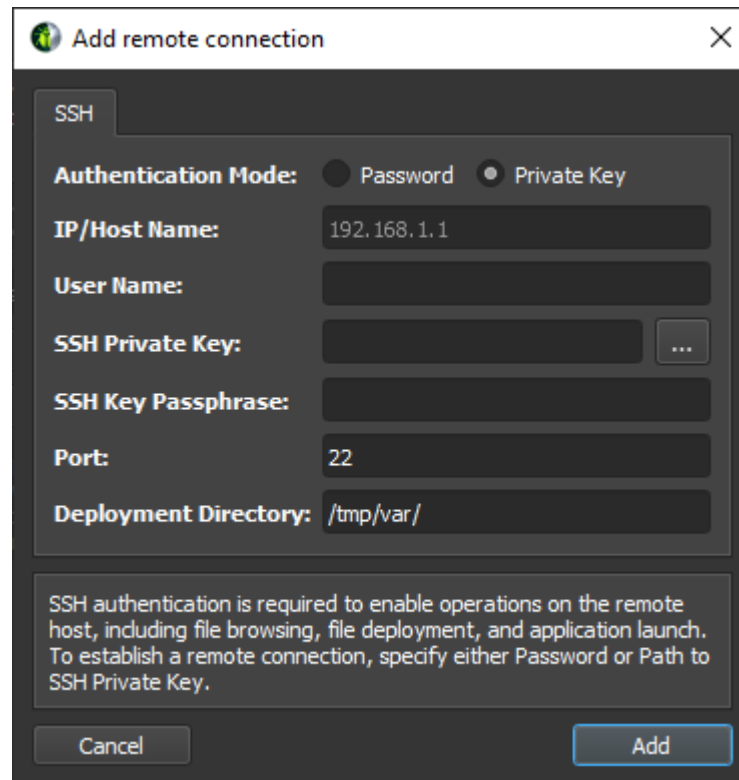
**Deployment Directory:** /tmp/var/

SSH authentication is required to enable operations on the remote host, including file browsing, file deployment, and application launch. To establish a remote connection, specify either Password or Path to SSH Private Key.

Cancel Add

NVIDIA Nsight Compute supports both password and private key authentication methods. In this dialog, select the authentication method and enter the following information:

- ▶ **Password**
  - ▶ **IP/Host Name:** The IP address or host name of the target device.
  - ▶ **User Name:** The user name to be used for the SSH connection.
  - ▶ **Password:** The user password to be used for the SSH connection.
  - ▶ **Port:** The port to be used for the SSH connection. (The default value is 22.)
  - ▶ **Deployment Directory:** The directory to use on the target device to deploy supporting files. The specified user must have write permissions to this location.
- ▶ **Private Key**



- ▶ **IP/Host Name:** The IP address or host name of the target device.
- ▶ **User Name:** The user name to be used for the SSH connection.
- ▶ **SSH Private Key:** The private key that is used to authenticate to SSH server.
- ▶ **SSH Key Passphrase:** The passphrase for your private key.
- ▶ **Deployment Directory:** The directory to use on the target device to deploy supporting files. The specified user must have write permissions to this location.

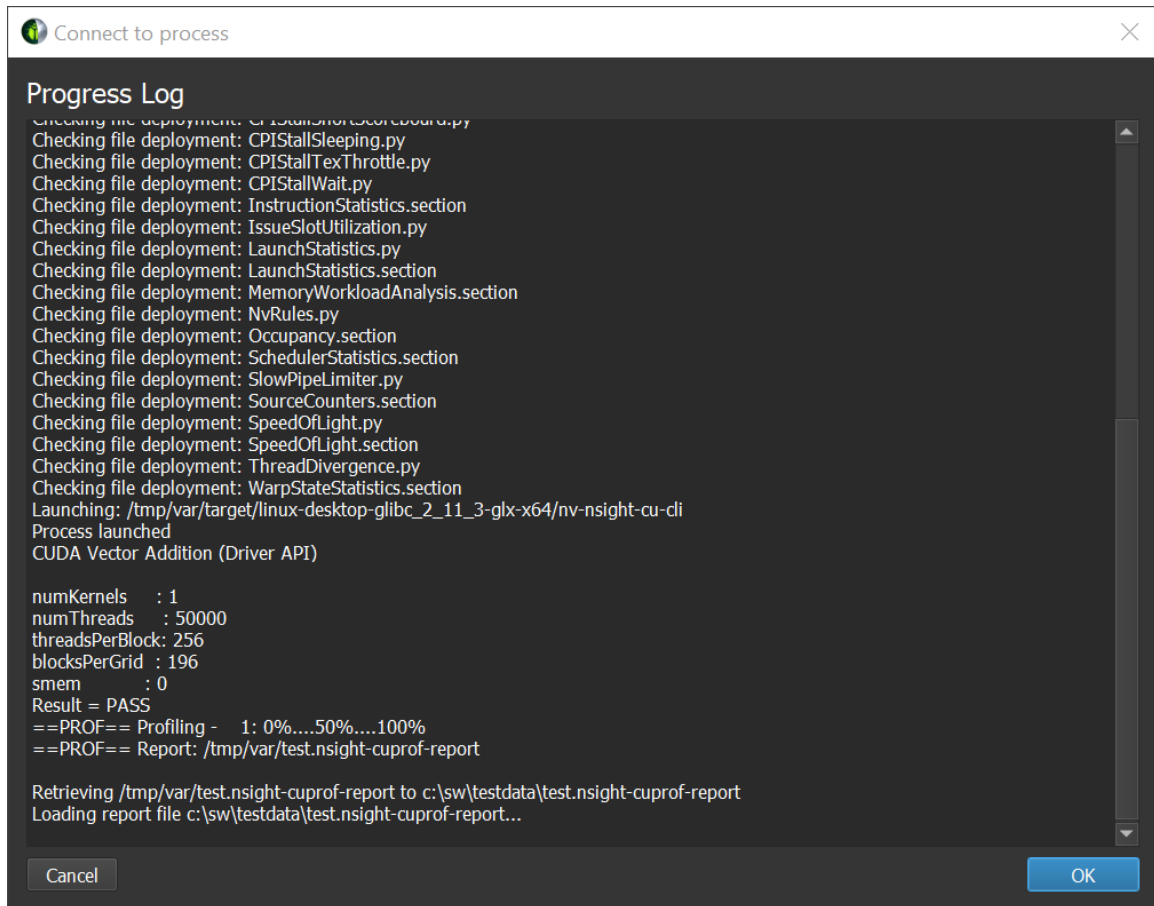
When all information is entered, click the *Add* button to make use of this new connection.

When a remote connection is selected in the *Connection Dialog*, the *Application Executable* file browser will browse the remote file system using the configured SSH connection, allowing the user to select the target application on the remote device.

When an activity is launched on a remote device, the following steps are taken:

1. The command line profiler and supporting files are copied into the *Deployment Directory* on the the remote device. (Only files that do not exist or are out of date are copied.)
2. The *Application Executable* is executed on the remote device.
  - ▶ For *Interactive Profile* activities, a connection is established to the remote application and the profiling session begins.
  - ▶ For *Non-Interactive Profile* activities, the remote application is executed under the command line profiler and the specified report file is generated.
3. For non-interactive profiling activities, the generated report file is copied back to the host, and opened.

The progress of each of these steps is presented in the *Progress Log*.



Note that once either activity type has been launched remotely, the tools necessary for further profiling sessions can be found in the *Deployment Directory* on the remote device.

## 3.2. Interactive Profile Activity

The *Interactive Profile* activity allows you to initiate a session that controls the execution of the target application, similar to a debugger. You can step API calls and workloads (CUDA kernels), pause and resume, and interactively select the kernels of interest and which metrics to collect.

This activity does currently not support profiling or attaching to child processes.

### ▶ Enable NVTX Support

Collect NVTX information provided by the application or its libraries. Required to support stepping to specific NVTX contexts.

### ▶ Disable Profiling Start/Stop

Ignore calls to **cu (da) ProfilerStart** or **cu (da) ProfilerStop** made by the application.

### ▶ Enable Profiling From Start

Enables profiling from the application start. Disabling this is useful if the application calls `cu(da)ProfilerStart` and kernels before the first call to this API should not be profiled. Note that disabling this does not prevent you from manually profiling kernels.

▶ **Cache Control**

Control the behavior of the GPU caches during profiling. Allowed values: For *Flush All*, all GPU caches are flushed before each kernel replay iteration during profiling. While metric values in the execution environment of the application might be slightly different without invalidating the caches, this mode offers the most reproducible metric results across the replay passes and also across multiple runs of the target application.

For *Flush None*, no GPU caches are flushed during profiling. This can improve performance and better replicates the application behavior if only a single kernel replay pass is necessary for metric collection. However, some metric results will vary depending on prior GPU work, and between replay iterations. This can lead to inconsistent and out-of-bounds metric values.

▶ **Clock Control**

Control the behavior of the GPU clocks during profiling. Allowed values: For *Base*, GPC and memory clocks are locked to their respective base frequency during profiling. This has no impact on thermal throttling. For *None*, no GPC or memory frequencies are changed during profiling.

### 3.3. Profile Activity

The *Profile* activity provides a traditional, pre-configurable profiler. After configuring which kernels to profile, which metrics to collect, etc, the application is run under the profiler without interactive control. The activity completes once the application terminates. For applications that normally do not terminate on their own, e.g. interactive user interfaces, you can cancel the activity once all expected kernels are profiled.

This activity does not support attaching to processes previously launched via NVIDIA Nsight Compute. These processes will be shown grayed out in the *Attach* tab.

▶ **Output File**

Path to report file where the collected profile should be stored. If not present, the report extension `.nsight-cuprof-report` is added automatically. The placeholder `%i` is supported for the filename component. It is replaced by a sequentially increasing number to create a unique filename. This maps to the `--export` command line option.

▶ **Force Overwrite**

If set, existing report file are overwritten. This maps to the `--force-overwrite` command line option.

▶ **Target Processes**

Select the processes you want to profile. In mode *Application Only*, only the root application process is profiled. In mode *all*, the root application process and all its

child processes are profiled. This maps to the `--target-processes` command line option.

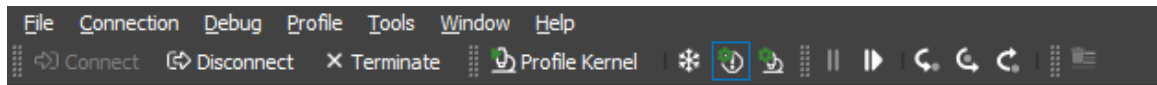
- ▶ **Additional Options**

All remaining options map to their command line profiler equivalents. See the *Command Line Options* section in the NVIDIA Nsight Compute CLI documentation for details.

# Chapter 4.

## MAIN MENU AND TOOLBAR

Information on the main menu and toolbar.



### 4.1. Main Menu

- ▶ File
  - ▶ **New Project** Create new profiling [Projects](#) with the [New Project Dialog](#)
  - ▶ **Open Project** Open an existing profiling project
  - ▶ **Recent Projects** Open an existing profiling project from the list of recently used projects
  - ▶ **Save Project** Save the current profiling project
  - ▶ **Save Project As** Save the current profiling project with a new filename
  - ▶ **Close Project** Close the current profiling project
- ▶ Connection
  - ▶ **Connect** Open the [Connection Dialog](#) to launch or attach to a target application. Disabled when already connected.
  - ▶ **Disconnect** Disconnect from the current target application, allows the application to continue normally and potentially re-attach.
  - ▶ **Terminate** Disconnect from and terminate the current target application immediately.
- ▶ Debug
  - ▶ **Pause** Pause the target application at the next intercepted API call or launch.
  - ▶ **Resume** Resume the target application.
  - ▶ **Step In** Step into the current API call or launch to the next nested call, if any, or the subsequent API call, otherwise.
  - ▶ **Step Over** Step over the current API call or launch and suspend at the next, non-nested API call or launch.

- ▶ **Step Out** Step out of the current nested API call or launch to the next, non-parent API call or launch one level above.
- ▶ **Next API Launch** See API Stream
- ▶ **Next API Trigger** See API Stream
- ▶ **Freeze API**

When disabled, all CPU threads are enabled and continue to run during stepping or resume, and all threads stop as soon as at least one thread arrives at the next API call or launch. This also means that during stepping or resume the currently selected thread might change as the old selected thread makes no forward progress and the API Stream automatically switches to the thread with a new API call or launch. When enabled, only the currently selected CPU thread is enabled. All other threads are disabled and blocked.

Stepping now completes if the current thread arrives at the next API call or launch. The selected thread never changes. However, if the selected thread does not call any further API calls or waits at a barrier for another thread to make progress, stepping may not complete and hang indefinitely. In this case, pause, select another thread, and continue stepping until the original thread is unblocked. In this mode, only the selected thread will ever make forward progress.

- ▶ **Break On API Error** When enabled, during resume or stepping, execution is suspended as soon as an API call returns an error code.
- ▶ **API Statistics** Opens the [API Statistics](#) tool window
- ▶ **API Stream** Opens the [API Stream](#) tool window
- ▶ **Resources** Opens the [Resources](#) tool window
- ▶ **NVTX** Opens the [NVTX](#) tool window
- ▶ Profile
  - ▶ **Clear Baselines** Clear all current baselines.
  - ▶ **Auto Profile** Enable or disable auto profiling. If enabled, each kernel matching the current kernel filter (if any) will be profiled using the current section configuration.
  - ▶ **Section/Rules Info** Opens the [Sections/Rules Info](#) tool window.
- ▶ Tools
  - ▶ **Profile Kernel** When suspended at a kernel launch, select the profile using the current configuration.
  - ▶ **Project Explorer** Opens the [Project Explorer](#) tool window.
  - ▶ **Output Messages** Opens the Output Messages tool window.
  - ▶ **Options** Opens the [Options](#) dialog.
- ▶ Window
  - ▶ **Show Welcome Page** Opens the [Welcome Page](#).
- ▶ Help
  - ▶ **Documentation** Opens the latest documentation for NVIDIA Nsight Compute online.



- ▶ **Documentation (local)** Opens the local HTML documentation for NVIDIA Nsight Compute that has shipped with the tool.
- ▶ **Check For Updates** Checks online if a newer version of NVIDIA Nsight Compute is available for download.
- ▶ **Reset Application Data** Reset all NVIDIA Nsight Compute configuration data saved on disk, including option settings, default paths, recent project references etc. This will not delete saved reports.
- ▶ **Send Feedback** Opens a dialog that allows you to send bug reports and suggestions for features. Optionally, the feedback includes basic system information, screenshots, or additional files (such as profile reports).
- ▶ **About** Opens the About dialog with information about the version of NVIDIA Nsight Compute.

## 4.2. Main Toolbar

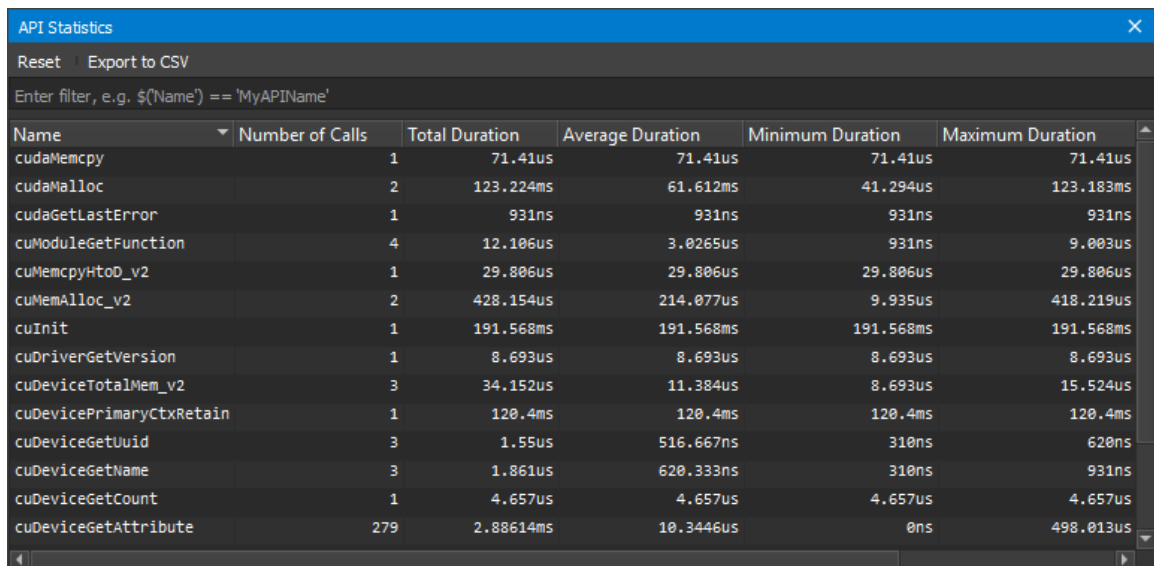
The main toolbar shows commonly used operations from the main menu. See [Main Menu](#) for their description.

# Chapter 5.

## TOOL WINDOWS

### 5.1. API Statistics

The *API Statistics* window is available when NVIDIA Nsight Compute is connected to a target application. It opens by default as soon as the connection is established. It can be re-opened using *Debug > API Statistics* from the main menu.



The screenshot shows the 'API Statistics' window with a table of API call statistics. The table has columns for Name, Number of Calls, Total Duration, Average Duration, Minimum Duration, and Maximum Duration. The data is as follows:

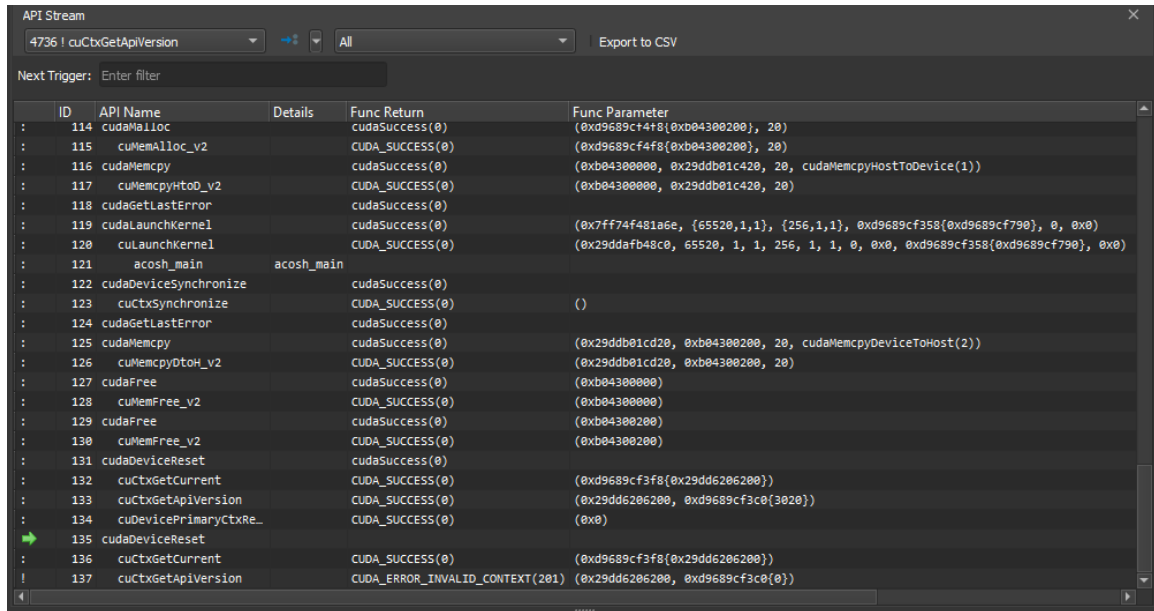
Name	Number of Calls	Total Duration	Average Duration	Minimum Duration	Maximum Duration
cudaMemcpy	1	71.41us	71.41us	71.41us	71.41us
cudaMalloc	2	123.224ms	61.612ms	41.294us	123.183ms
cudaGetLastError	1	931ns	931ns	931ns	931ns
cuModuleGetFunction	4	12.106us	3.0265us	931ns	9.003us
cuMemcpyHtoD_v2	1	29.806us	29.806us	29.806us	29.806us
cuMemAlloc_v2	2	428.154us	214.077us	9.935us	418.219us
cuInit	1	191.568ms	191.568ms	191.568ms	191.568ms
cuDriverGetVersion	1	8.693us	8.693us	8.693us	8.693us
cuDeviceTotalMem_v2	3	34.152us	11.384us	8.693us	15.524us
cuDevicePrimaryCtxRetain	1	120.4ms	120.4ms	120.4ms	120.4ms
cuDeviceGetUuid	3	1.55us	516.667ns	310ns	620ns
cuDeviceGetName	3	1.861us	620.333ns	310ns	931ns
cuDeviceGetCount	1	4.657us	4.657us	4.657us	4.657us
cuDeviceGetAttribute	279	2.88614ms	10.3446us	0ns	498.013us

Whenever the target application is suspended, it shows a summary of tracked API calls with some statistical information, such as the number of calls, their total, average, minimum and maximum duration. Note that this view cannot be used as a replacement for *Nsight Systems* when trying to optimize CPU performance of your application.

The *Reset* button deletes all statistics collected to the current point and starts a new collection. Use the *Export to CSV* button to export the current statistics to a CSV file.

## 5.2. API Stream

The *API Stream* window is available when NVIDIA Nsight Compute is connected to a target application. It opens by default as soon as the connection is established. It can be re-opened using *Debug > API Stream* from the main menu.



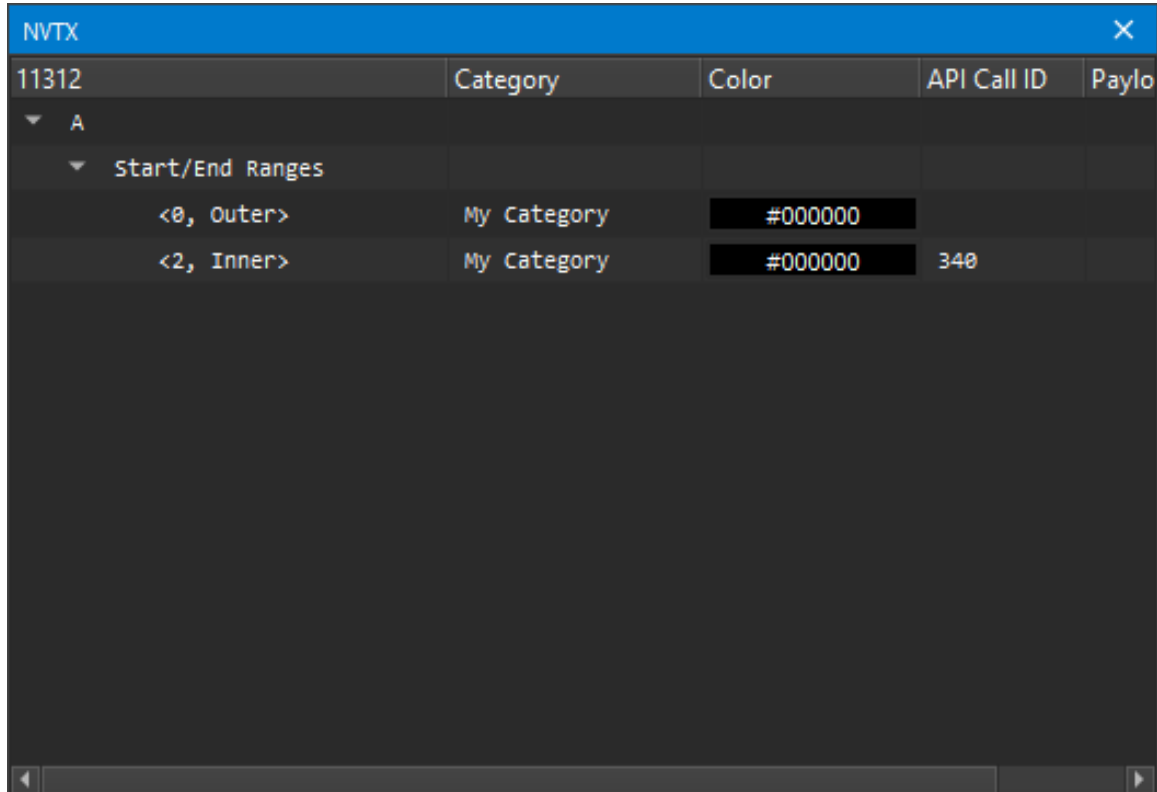
Whenever the target application is suspended, the window shows the history of API calls and traced kernel launches. The currently suspended API call or kernel launch (activity) is marked with a yellow arrow. If the suspension is at a subcall, the parent call is marked with a green arrow. The API call or kernel is suspended before being executed.

For each activity, further information is shown such as the kernel name or the function parameters (*Func Parameters*) and return value (*Func Return*). Note that the function return value will only become available once you step out or over the API call.

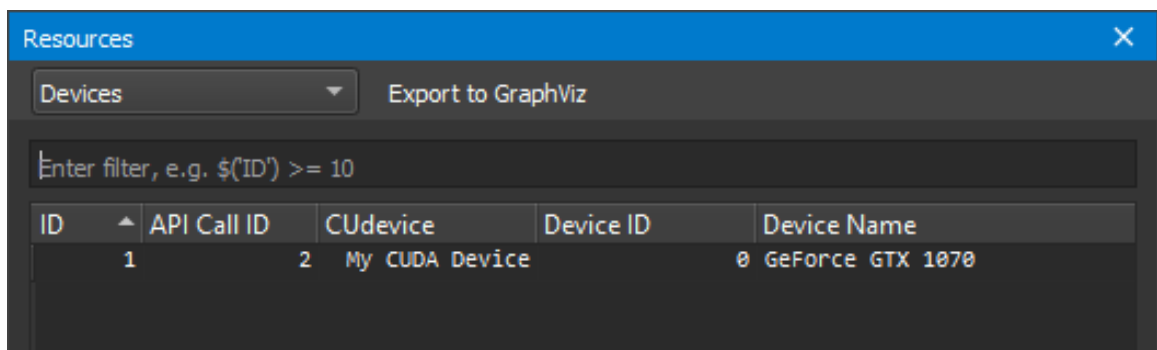
Use the *Current Thread* dropdown to switch between the active threads. The dropdown shows the thread ID followed by the current API name. One of several options can be chosen in the trigger dropdown, which are executed by the adjacent >> button. *Next Kernel Launch* resumes execution until the next kernel launch is found in any enabled thread. *Next API Call* resumes execution until the next API call matching *Next Trigger* is found in any enabled thread. *Next Range Start* resumes execution until the next start of an active profiler range is found. Profiler ranges are defined by using the **cu (da) ProfilerStart/Stop** API calls. *Next Range Stop* resumes execution until the next stop of an active profiler range is found. The *API Level* dropdown changes which API levels are shown in the stream. The *Export to CSV* button exports the currently visible stream to a CSV file.

## 5.3. NVTX

The *NVTX* window is available when NVIDIA Nsight Compute is connected to a target application. If closed, it can be re-opened using *Debug > NVTX* from the main menu. Whenever the target application is suspended, the window shows the state of all active NVTX domains and ranges in the currently selected thread. Note that *NVTX* information is only tracked if the launching command line profiler instance was started with `--nvtx` or NVTX was enabled in the NVIDIA Nsight Compute launch dialog.

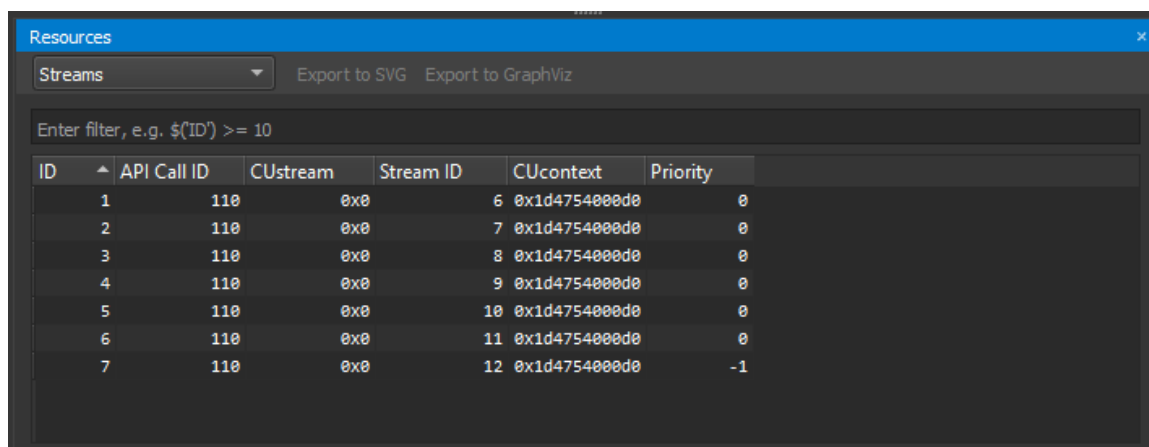


Use the *Current Thread* dropdown in the *API Stream* window to change the currently selected thread. NVIDIA Nsight Compute supports NVTX named resources, such as threads, CUDA devices, CUDA contexts, etc. If a resource is named using NVTX, the appropriate UI elements will be updated.



## 5.4. Resources

The *Resources* window is available when NVIDIA Nsight Compute is connected to a target application. It shows information about the currently known resources, such as CUDA devices, CUDA streams or kernels. The window is updated every time the target application is suspended. If closed, it can be re-opened using *Debug > Resources* from the main menu.



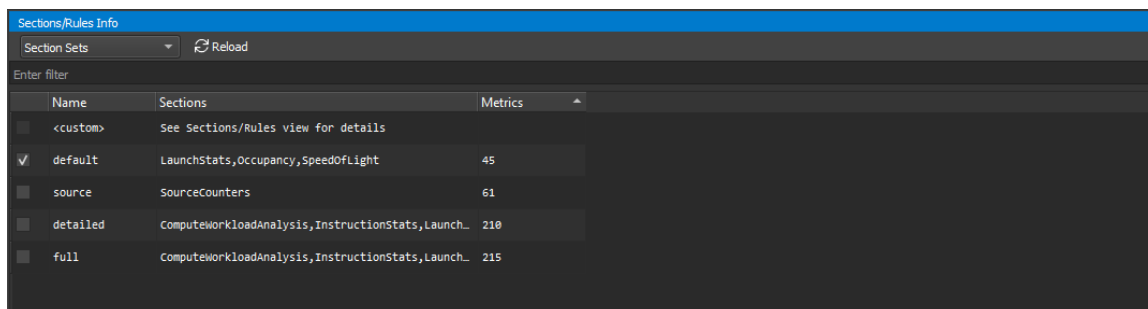
Using the dropdown on the top, different views can be selected, where each view is specific to one kind of resource (context, stream, kernel, ...). The *Filter* edit allows you to create filter expressions using the column headers of the currently selected resource.

The resource table shows all information for each resource instance. Each instance has a unique ID, the *API Call ID* when this resource was created, its handle, associated handles, and further parameters. When a resource is destroyed, it is removed from its table.

## 5.5. Sections/Rules Info

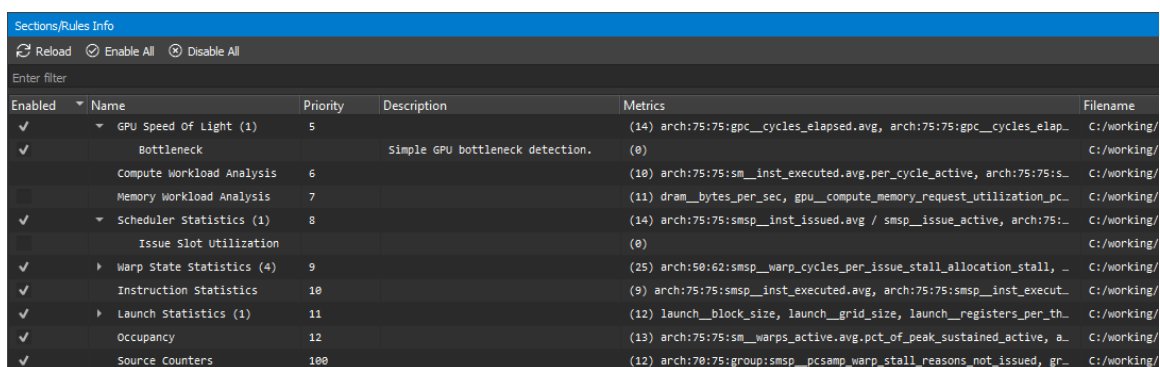
The *Sections/Rules Info* window can be opened from the main menu using *Profile > Sections/Rules Info*. It tracks all section sets, sections and rules currently loaded in NVIDIA Nsight Compute, independent from a specific connection or report. The directory to load those files from can be configured in the *Profile* options dialog. It is used to inspect available sets, sections and rules, as well as to configure which should be collected, and which rules should be applied. The window has two views, which can be selected using the dropdown in its header.

The **Section Sets** view shows all available section sets. Each set is associated with a number of sections. You can choose a set appropriate to the level of detail for which you want to collect performance metrics. Sets which collect more detailed information normally incur higher runtime overhead during profiling.



When enabling a set in this view, the associated sections are enabled in the *Sections/Rules* view. When disabling a set in this view, the associated sections in the *Sections/Rules* view are disabled. If no set is enabled, or if sections are manually enabled/disabled in the *Sections/Rules* view, the <custom> entry is marked active to represent that no section set is currently enabled. Note that the default set is enabled by default.

Whenever a kernel is profiled manually, or when auto-profiling is enabled, only sections enabled in the **Sections/Rules** view are collected. Similarly, whenever rules are applied, only rules enabled in this view are active.



The enabled states of sections and rules are persisted across NVIDIA Nsight Compute launches. The *Reload* button reloads all sections and rules from disk again. If a new section or rule is found, it will be enabled if possible. If any errors occur while loading a rule, they will be listed in an extra entry with a warning icon and a description of the error.

Use the *Enable All* and *Disable All* checkboxes to enable or disable all sections and rules at once. The Filter text box can be used to filter what is currently shown in the view. It does not alter activation of any entry.

The table shows sections and rules with their activation status, their relationship and further parameters, such as associated metrics or the original file on disk. Rules associated with a section are shown as children of their section entry. Rules independent of any section are shown under an additional *Independent Rules* entry.

Double-clicking an entry in the table's *Filename* column opens this file as a document. It can be edited and saved directly in NVIDIA Nsight Compute. After editing the file, *Reload* must be selected to apply those changes.

See [Sections and Rules](#) for the list of default sections for NVIDIA Nsight Compute.

# Chapter 6.

## PROFILER REPORT

The profiler report contains all the information collected during profiling for each kernel launch. In the user interface, it consists of a header with general information, as well as controls to switch between report pages or individual collected launches. By default, the report starts with the *Details* page selected.

### 6.1. Header

The *Page* dropdown can be used to switch between the available report pages, which are explained in detail in the [next section](#).



The *Process* dropdown filters which kernel launches are available in the *Launch* dropdown. Select a single process to see only launches from this process. Selecting *All* shows all instances.

The *Launch* dropdown can be used to switch between all collected kernel launches. The information displayed in each page commonly represents the selected launch instance. On some pages (e.g. *Raw*), information for all launches is shown and the selected instance is highlighted. You can type in this dropdown to quickly filter and find a kernel launch.

The *Add Baseline* button promotes the current result in focus to become the baseline of all other results from this report and any other report opened in the same instance of NVIDIA Nsight Compute. Select the arrow dropdown to access the *Clear Baselines* button, which removes all currently active baselines.

The *Apply Rules* button applies all rules available for this report. If rules had been applied previously, those results will be replaced. By default, rules are applied immediately once the kernel launch has been profiled. This can be changed in the options under *Tools > Options > Profile > Report UI > Apply Applicable Rules Automatically*.

A button on the right-hand side offers multiple operations that may be performed on the page. Available operations include:

- ▶ **Copy as Image** - Copies the contents of the page to the clipboard as an image.
- ▶ **Save as Image** - Saves the contents of the page to a file as an image.
- ▶ **Save as PDF** - Saves the contents of the page to a file as a PDF.
- ▶ **Export to CSV** - Exports the contents of page to CSV format.
- ▶ **Reset to Default** - Resets the page to a default state by removing any persisted settings.

Note that not all functions are available on all pages.

Information about the selected kernel is shown as *Current*. [+] and [-] buttons can be used to show or hide the section body content. The info toggle button *i* changes the section description's visibility.

## 6.2. Report Pages

Use the *Page* dropdown in the header to switch between the report pages.

### 6.2.1. Session Page

This *Session* page contains basic information about the report and the machine, as well as device attributes of all devices for which launches were profiled. When switching between launch instances, the respective device attributes are highlighted.

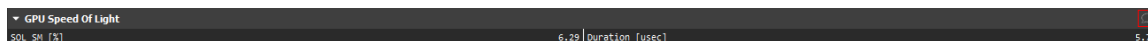
### 6.2.2. Summary Page

The *Summary* page shows a list of all collected results in this report, with selected important summary metrics. It gives you a quick comparison overview across all profiled kernel launches. You can transpose the table of kernels and metrics by using the *Transpose* button.

### 6.2.3. Details Page

The *Details* page is the main page for all metric data collected during a kernel launch. The page is split into individual sections. Each section consists of a header table and an optional body that can be expanded. The sections are completely user defined and can be changed easily by updating their respective files. For more information on customizing sections, see the *Customization Guide*. For a list of sections shipped with NVIDIA Nsight Compute, see [Sections and Rules](#).

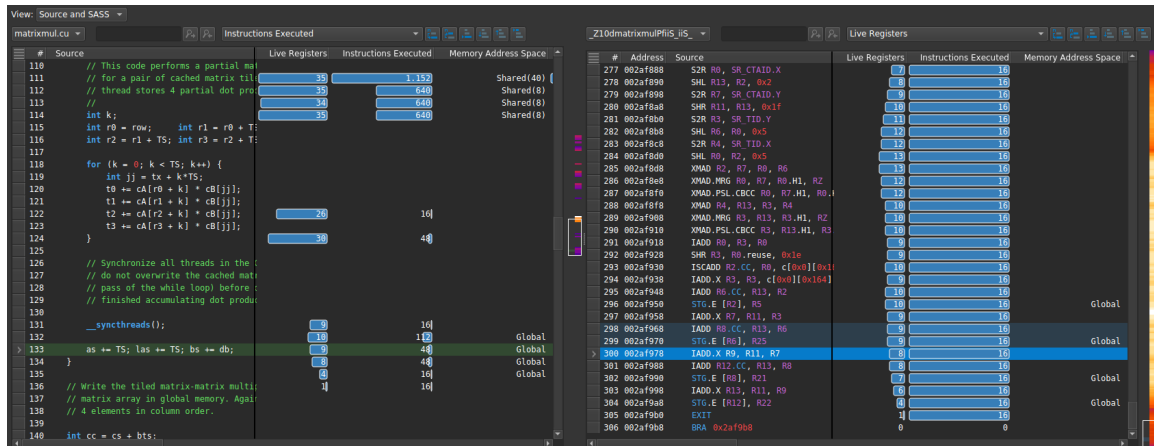
You can add or edit comments in each section of the *Details* view by clicking on the comment button (speech bubble). The comment icon will be highlighted in sections that contain a comment. Comments are persisted in the report and are summarized in the [Comments Page](#).





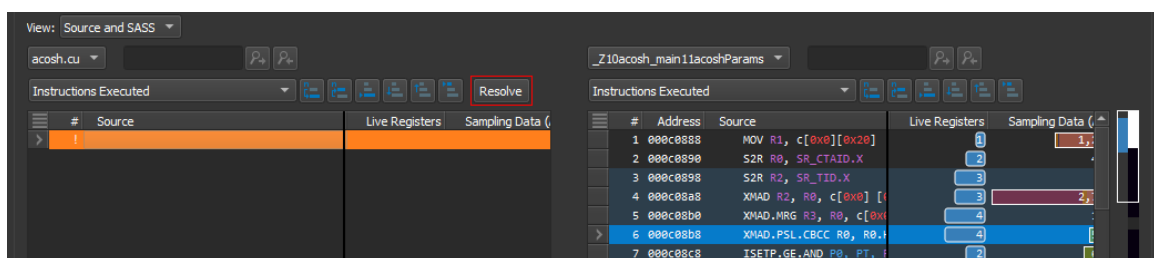
## 6.2.4. Source Page

The *Source* page correlates assembly (SASS) with high-level code and PTX. In addition, it displays metrics that can be correlated with source code. It is filtered to only show (SASS) functions that were executed in the kernel launch.



The *View* dropdown can be used to select different code (correlation) options. This includes SASS, PTX and Source (CUDA-C), as well as their combinations. Which options are available depends on the source information embedded into the executable.

If the application was built with the `-lineinfo` or `--generate-line-info` nvcc flag to correlate SASS and source, the CUDA-C view is available. If source files are not found locally in their original path, only their filenames are shown in the view. Select a filename and click the *Resolve* button above to select where this source can be found on the local filesystem. If the report was collected using remote profiling, and automatic resolution of remote files is enabled in the *Profile* options, NVIDIA Nsight Compute will attempt to load the source from the remote target. If the connection credentials are not yet available in the current NVIDIA Nsight Compute instance, they are prompted in a dialog. Loading from a remote target is currently only available for Linux x86\_64 targets and Linux and Windows hosts.



The heatmap on the right-hand side of each view can be used to quickly identify locations with high metric values of the currently selected metric in the dropdown. The heatmap uses a black-body radiation color scale where black denotes the lowest mapped value and white the highest, respectively. The current scale is shown when clicking and holding the heatmap with the right mouse button.

If a view contains multiple source files or functions, [+] and [-] buttons are shown. These can be used to expand or collapse the view, thereby showing or hiding the file or function content except for its header. If collapsed, all metrics are shown aggregated to provide a quick overview.



Views allow you to fix columns to not move out of view when scrolling horizontally. By default, the *Source* column is fixed to the left, enabling easy inspection of all metrics correlated to a source line. To change fixing of columns, select the triangle icon in the respective column header.



### Pre-Defined Source Metrics

#### ▶ **Live Registers**

Number of registers that need to be kept valid by the compiler. A high value indicates that many registers are required at this code location, potentially increasing the register pressure and the maximum number of register required by the kernel.

#### ▶ **Sampling Data (All)**

The number of samples from the [Statistical Sampler](#) at this program location.

#### ▶ **Sampling Data (No Issue)**

The number of samples from the [Statistical Sampler](#) at this program location on cycles the warp scheduler issued no instructions. Note that (*No Issue*) samples may be taken on a different profiling pass than (*All*) samples mentioned above, so their values do not strictly correlate.

This metric is only available on devices with compute capability 7.0 or higher.

#### ▶ **Instructions Executed**

Number of times the source (instruction) was executed by any warp.

#### ▶ **Predicated-On Thread Instructions Executed**

Number of times the source (instruction) was executed by any active, predicated-on thread. For instructions that are executed unconditionally (i.e. without predicate), this is the number of active threads in the warp, multiplied with the respective *Instructions Executed* value.

#### ▶ **Information on Memory Operation**

This includes *Memory Address Space*, *Memory Access Operation*, *Memory Access Size*, *Memory L1 Transactions Shared*, *Memory L2 Transactions Global* and *Memory L2 Transactions Local*.

#### ▶ **Individual Sampling Data Metrics**

All *stall\_\** metrics show the information combined in *Sampling Data* individually. See [Statistical Sampler](#) for their description.

#### ▶ See the *Customization Guide* on how to add additional metrics for this view.

## 6.2.5. Comments Page

The *Comments* page aggregates all section comments in a single view and allows the user to edit those comments on any launch instance or section, as well as on the overall report. Comments are persisted with the report. If a section comment is added, the comment icon of the respective section in the [Details Page](#) will be highlighted.

## 6.2.6. NVTX Page

The *NVTX* page shows the NVTX context when the kernel was launched. All thread-specific information is with respect to the thread of the kernel's launch API call. Note that NVTX information is only collected if the profiler is started with NVTX support enabled, either in the [Connection Dialog](#) or using the NVIDIA Nsight Compute CLI command line parameter.

7967	Category	Color	API Call ID	Payload
▼	<default domain>			
▼	Push/Pop Stack			
	0: starting_cuda		105	
	1: starting_kernel_execution		122	

## 6.2.7. Raw Page

The *Raw* page shows a list of all collected metrics with their units per profiled kernel launch. It can be exported, for example, to CSV format for further analysis. The page features a filter edit to quickly find specific metrics. You can transpose the table of kernels and metrics by using the *Transpose* button.

## 6.3. Metrics and Units

Numeric metric values are shown in various places in the report, including the header and tables and charts on most pages. NVIDIA Nsight Compute supports various ways to display those metrics and their values.

When available and applicable to the UI component, metrics are shown along with their unit. This is to make it apparent if a metric represents cycles, threads, bytes/s, and so on. The unit will normally be shown in rectangular brackets, e.g. **Metric Name [bytes]** **128**.

By default, units are scaled automatically so that metric values are shown with a reasonable order of magnitude. Units are scaled using their SI-factors, i.e. byte-based units are scaled using a factor of 1000 and the prefixes K, M, G, etc. Time-based units are also scaled using a factor of 1000, with the prefixes n, u and m. This scaling can be disabled in the [Profile](#) options.

Metrics which could not be collected are shown as **n/a** and assigned a warning icon. If the metric floating point value is out of the regular range (i.e. **nan** (Not a number) or **inf**

(infinite)), they are also assigned a warning icon. The exception are metrics for which these values are expected and which are white-listed internally.

# Chapter 7. BASELINES

NVIDIA Nsight Compute supports diffing collected results across one or multiple reports using Baselines. Each result in any report can be promoted to a baseline. This causes metric values from all results in all reports to show the difference to the baseline. If multiple baselines are selected simultaneously, metric values are compared to the average across all current baselines. Note that currently, baselines are not stored with a report and are only available as long as the same NVIDIA Nsight Compute instance is open.

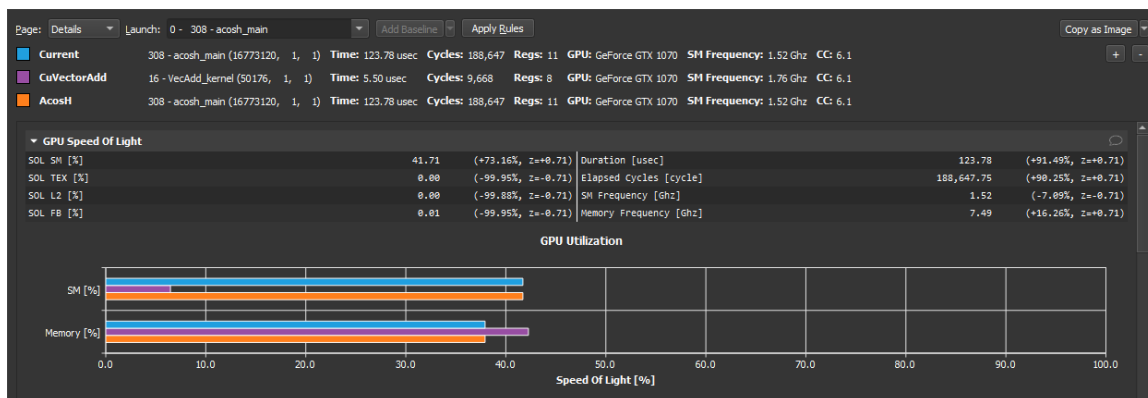


Select *Add Baseline* to promote the current result in focus to become a baseline. If a baseline is set, most metrics on the [Details Page](#), [Raw Page](#) and [Summary Page](#) show two values: the current value of the result in focus, and the corresponding value of the baseline or the percentage of change from the corresponding baseline value. (Note that an infinite percentage gain, *inf%*, may be displayed when the baseline value for the metric is zero, while the focus value is not.)

If multiple baselines are selected, each metric will show the following notation:

```
<focus value> (<difference to baselines average [%]>, z=<standard score>@<number of values>)
```

The standard score is the difference between the current value and the average across all baselines, normalized by the standard deviation. If the number of metric values contributing to the standard score equals the number of results (current and all baselines), the @<number of values> notation is omitted.



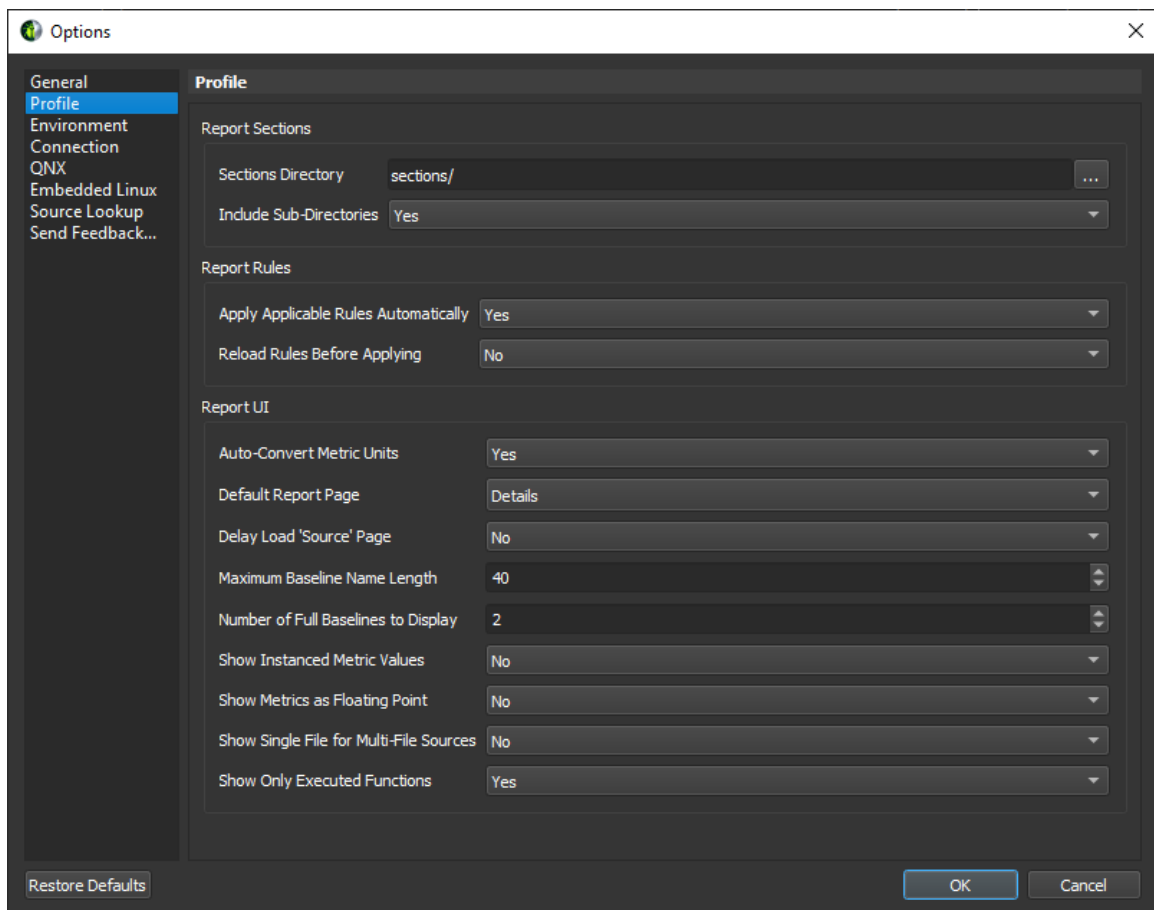
Hovering the mouse over a baseline name allows the user to edit the displayed name. Hovering over the baseline color icon allows the user to remove this specific baseline from the list.

Use the *Clear Baselines* entry from the dropdown button, the **Profile** menu, or the corresponding toolbar button to remove all baselines.

# Chapter 8.

## OPTIONS

NVIDIA Nsight Compute options can be accessed via the main menu under *Tools > Options*. All options are persisted on disk and available the next time NVIDIA Nsight Compute is launched. When an option is changed from its default setting, its label will become bold. You can use the *Restore Defaults* button to restore all options to their default values.



## 8.1. Profile

Table 1 NVIDIA Nsight Compute Profile Options

Name	Description	Values
Sections Directory	Directory from which to import section files and rules. Relative paths are with respect to the NVIDIA Nsight Compute installation directory.	
Include Sub-Directories	Recursively include section files and rules from sub-directories.	Yes (Default)/No
Apply Applicable Rules Automatically	Automatically apply active and applicable rules.	Yes (Default)/No
Reload Rules Before Applying	Force a rule reload before applying the rule to ensure changes in the rule script are recognized.	Yes/No (Default)
Auto-Convert Metric Units	Auto-adjust displayed metric units and values (e.g. Bytes to KBytes).	Yes (Default)/No
Default Report Page	The report page to show when a report is generated or opened.	<ul style="list-style-type: none"> <li>▶ Session</li> <li>▶ Summary</li> <li>▶ Details (Default)</li> <li>▶ Source</li> <li>▶ Comments</li> <li>▶ Raw</li> <li>▶ Nvtx</li> </ul>
Delay Load Source Page	Delays loading the content of the Source report page until the page becomes visible. Avoids processing costs and memory overhead until the page is opened.	Yes/No (Default)
Maximum Baseline Name Length	The maximum length of baseline names.	1..N
Number of Full Baselines to Display	Number of baselines to display in the report header with all details in addition to the current result.	0..N
Show Instanced Metric Values	Show the individual values of instanced metrics in tables.	Yes/No (Default)
Show Metrics As Floating Point	Show all numeric metrics as floating-point numbers.	Yes/No (Default)
Show Single File For Multi-File Sources	Shows a single file in each Source page view, even for multi-file sources.	Yes/No (Default)



Name	Description	Values
Show Only Executed Functions	Shows only executed functions in the source page views. Disabling this can impact performance.	Yes (Default)/No
Auto-Resolve Remote Source Files	Automatically try to resolve remote source files on the source page (e.g. via SSH) if the connection is still registered.	Yes/No (Default)

## 8.2. Environment

Table 2 NVIDIA Nsight Compute Environment Options

Name	Description	Values
Color Theme	The currently selected UI color theme.	<ul style="list-style-type: none"> <li>▶ Dark (Default)</li> <li>▶ Light</li> </ul>
Mixed DPI Scaling	Disable Mixed DPI Scaling if unwanted artifacts are detected when using monitors with different DPIs.	<ul style="list-style-type: none"> <li>▶ Auto (Default)</li> <li>▶ Off</li> </ul>
Default Document Folder	Directory where documents unassociated with a project will be saved.	
At Startup	What to do when NVIDIA Nsight Compute is launched.	<ul style="list-style-type: none"> <li>▶ Show welcome page (Default)</li> <li>▶ Show quick launch dialog</li> <li>▶ Load last project/Show empty environment</li> </ul>
Show version update notifications	Show notifications when a new version of this product is available.	<ul style="list-style-type: none"> <li>▶ Yes (Default)</li> <li>▶ No</li> </ul>

## 8.3. Connection

Table 3 NVIDIA Nsight Compute Connection Options

Name	Description	Values
Base Port	Base port used for connections to profiled target applications, both locally and remote.	1-65535
Maximum Ports	Maximum number of ports, starting from <i>Base Port</i> used for connecting to target applications.	2-65534

## 8.4. Source Lookup

Table 4 NVIDIA Nsight Compute Source Lookup Options

Name	Description	Values
Program Source Locations	Set program source search paths. These paths are used to resolve CUDA-C source files on the Source page if the respective file cannot be found in its original location.	

## 8.5. Send Feedback

Table 5 NVIDIA Nsight Compute Send Feedback Options

Name	Description	Values
Collect Usage and Platform Data	Choose whether or not you wish to allow NVIDIA Nsight Compute to collect usage and platform data.	<ul style="list-style-type: none"> <li>▶ Yes (Default)</li> <li>▶ No</li> </ul>

# Chapter 9.

## PROJECTS

NVIDIA Nsight Compute uses *Project Files* to group and organize profiling reports. At any given time, only one project can be open in NVIDIA Nsight Compute. Collected reports are automatically assigned to the current project. Reports stored on disk can be assigned to a project at any time. In addition to profiling reports, related files such as notes or source code can be associated with the project for future reference.

Note that only references to reports or other files are saved in the project file. Those references can become invalid, for example when associated files are deleted, removed or not available on the current system, in case the project file was moved itself.

NVIDIA Nsight Compute uses the **nsight-cuproj** file extension for project files.

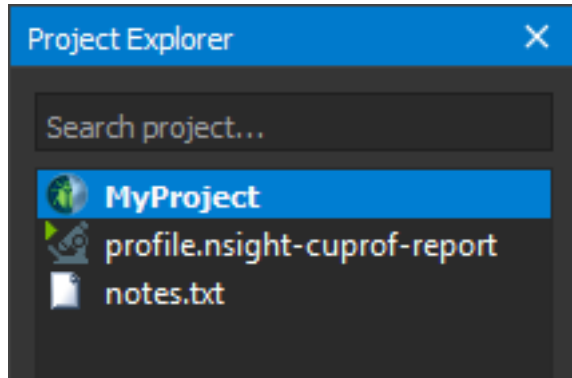
### 9.1. Project Dialogs

#### ▶ New Project

Creates a new project. The project must be given a name, which will also be used for the project file. You can select the location where the project file should be saved on disk. Select whether a new directory with the project name should be created in that location.

### 9.2. Project Explorer

The *Project Explorer* window allows you to inspect and manage the current project. It shows the project name as well as all *Items* (profile reports and other files) associated with it. Right-click on any entry to see further actions, such as adding, removing or grouping items. Type in the *Search project* toolbar at the top to filter the currently shown entries.



# Chapter 10.

## VISUAL PROFILER TRANSITION GUIDE

This guide provides tips for moving from Visual Profiler to NVIDIA Nsight Compute. NVIDIA Nsight Compute tries to provide as much parity as possible with Visual Profiler's kernel profiling features, but some functionality is now covered by different tools.

### 10.1. Trace

NVIDIA Nsight Compute does not support tracing GPU or API activities on an accurate timeline. This functionality is covered by [NVIDIA Nsight Systems](#). In the [Interactive Profile Activity](#), the [API Stream](#) tool window provides a stream of recent API calls on each thread. However, since all tracked API calls are serialized by default, it does not collect accurate timestamps.

### 10.2. Sessions

Instead of sessions, NVIDIA Nsight Compute uses [Projects](#) to launch and gather connection details and collected reports.

#### ► Executable and Import Sessions

Use the [Project Explorer](#) or the [Main Menu](#) to create a new project. Reports collected from the command line, i.e. using NVIDIA Nsight Compute CLI, can be opened directly using the main menu. In addition, you can use the Project Explorer to associate existing reports as well as any other artifacts such as executables, notes, etc., with the project. Note that those associations are only references; in other words, moving or deleting the project file on disk will not update its artifacts.

nvprof or command-line profiler output files, as well as Visual Profiler sessions, cannot be imported into NVIDIA Nsight Compute.

## 10.3. Timeline

Since trace analysis is now covered by Nsight Systems, NVIDIA Nsight Compute does not provide views of the application timeline. The [API Stream](#) tool window does show a per-thread stream of the last captured CUDA API calls. However, those are serialized and do not maintain runtime concurrency or provide accurate timing information.

## 10.4. Analysis

### ▶ **Guided Analysis**

All trace-based analysis is now covered by [NVIDIA Nsight Systems](#). This means that NVIDIA Nsight Compute does not include analysis regarding concurrent CUDA streams or (for example) UVM events. For per-kernel analysis, NVIDIA Nsight Compute provides recommendations based on collected performance data on the [Details Page](#). These rules currently require you to collect the required metrics via their sections up front, and do not support partial on-demand profiling.

To use the rule-based recommendations, enable the respective rules in the [Sections/Rules Info](#). Before profiling, enable *Apply Rules* in the [Profile Options](#), or click the *Apply Rules* button in the report afterward.

### ▶ **Unguided Analysis**

All trace-based analysis is now covered by Nsight Systems. For per-kernel analysis, Python-based rules provide analysis and recommendations. See *Guided Analysis* above for more details.

### ▶ **PC Sampling View**

Source-correlated PC sampling information can now be viewed in the [Source Page](#). Aggregated warp states are shown on the [Details Page](#) in the [Warp State Statistics](#) section.

### ▶ **Memory Statistics**

Memory Statistics are located on the [Details Page](#). Enable the [Memory Workload Analysis](#) sections to collect the respective information.

### ▶ **NVLink View**

NVIDIA Nsight Compute does not currently support NVLink metrics or topology information.

### ▶ **Source-Disassembly View**

Source correlated with PTX and SASS disassembly is shown on the [Source Page](#). Which information is available depends on your application's compilation/JIT flags.

### ▶ **GPU Details View**

NVIDIA Nsight Compute does not automatically collect data for each executed kernel, and it does not collect any data for device-side memory copies. Summary information for all profiled kernel launches is shown on the [Summary Page](#).

Comprehensive information on all collected metrics for all profiled kernel launches is shown on the [Raw Page](#).

▶ **CPU Details View**

CPU callstack sampling is now covered by [NVIDIA Nsight Systems](#).

▶ **OpenACC Details View**

OpenACC performance analysis is not supported by NVIDIA Nsight Compute. See the [NVIDIA Nsight Systems](#) release notes to check its latest support status.

▶ **OpenMP Details View**

OpenMP performance analysis is not supported by NVIDIA Nsight Compute. See the [NVIDIA Nsight Systems](#) release notes to check its latest support status.

▶ **Properties View**

NVIDIA Nsight Compute does not collect CUDA API and GPU activities and their properties. Performance data for profiled kernel launches is reported (for example) on the [Details Page](#).

▶ **Console View**

NVIDIA Nsight Compute does not currently collect stdout/stderr application output.

▶ **Settings View**

Application launch settings are specified in the [Connection Dialog](#). For reports collected from the UI, launch settings can be inspected on the [Session Page](#) after profiling.

▶ **CPU Source View**

Source for CPU-only APIs is not available. Source for profiled GPU kernel launches is shown on the [Source Page](#).

## 10.5. Command Line Arguments

Please execute `nv-nsight-cu` with the `-h` parameter within a shell window to see the currently supported command line arguments for the NVIDIA Nsight Compute UI.

To open a collected profile report with `nv-nsight-cu`, simply pass the path to the report file as a parameter to the shell command.

# Chapter 11.

## VISUAL STUDIO INTEGRATION GUIDE

This guide provides information on using NVIDIA Nsight Compute within Microsoft Visual Studio, using the [NVIDIA Nsight Integration](#) Visual Studio extension, allowing for a seamless development workflow.

### 11.1. Visual Studio Integration Overview

NVIDIA Nsight Integration is a Visual Studio extension that allows you to access the power of NVIDIA Nsight Compute from within Visual Studio.

When NVIDIA Nsight Compute is installed along with NVIDIA Nsight Integration, NVIDIA Nsight Compute activities will appear under the NVIDIA 'Nsight' menu in the Visual Studio menu bar. These activities launch NVIDIA Nsight Compute with the current project settings and executable.

For more information about using NVIDIA Nsight Compute from within Visual Studio, please visit

- ▶ [NVIDIA Nsight Integration Overview](#)
- ▶ [NVIDIA Nsight Integration User Guide](#)



# Chapter 12.

## LIBRARY SUPPORT

NVIDIA Nsight Compute can be used to profile CUDA applications, as well as applications that use CUDA via NVIDIA or third-party libraries. For most such libraries, the behavior is expected to be identical to applications using CUDA directly. However, for certain libraries, NVIDIA Nsight Compute has certain restrictions, alternate behavior, or requires non-default setup steps prior to profiling.

### 12.1. OptiX

NVIDIA Nsight Compute supports profiling of OptiX applications, but with certain restrictions.

- ▶ **Internal Kernels**

Kernels launched by OptiX that contain no user-defined code are given the generic name *NVIDIA internal*. These kernels show up on the API Stream in the NVIDIA Nsight Compute UI, and can be profiled in both the UI as well as the NVIDIA Nsight Compute CLI. However, no CUDA-C source, PTX or SASS is available for them.

- ▶ **User Kernels**

Kernels launched by OptiX can contain user-defined code. OptiX identifies these kernels in the API Stream with a custom name. This name starts with *raygen\_\_* (for "ray generation"). These kernels show up on the API Stream and can be profiled in the UI as well as the NVIDIA Nsight Compute CLI. The Source page displays CUDA-C source, PTX and SASS defined by the user. Certain parts of the kernel, including device functions that contain OptiX-internal code, will not be available in the Source page.

- ▶ **SASS**

When SASS information is available in the profile report, certain instructions might not be available in the Source page and shown as *N/A*.

# Appendix A.

## STATISTICAL SAMPLER

NVIDIA Nsight Compute supports periodic sampling of the warp program counter and warp scheduler state on desktop devices of compute capability 6.1 and above.

At a fixed interval of cycles, the sampler in each streaming multiprocessor selects an active warp and outputs the program counter and the warp scheduler state. The tool selects the minimum interval for the device. On small devices, this can be every 32 cycles. On larger chips with more multiprocessors, this may be 2048 cycles. The sampler selects a random active warp. On the same cycle the scheduler may select a different warp to issue.

Table 6 Warp Scheduler States

State	Hardware Support	Description
Allocation	5.2-6.1	Warp was stalled waiting for a branch to resolve, waiting for all memory operations to retire, or waiting to be allocated to the micro-scheduler.
Barrier	5.2+	Warp was stalled waiting for sibling warps at a CTA barrier. A high number of warps waiting at a barrier is commonly caused by diverging code paths before a barrier. This causes some warps to wait a long time until other warps reach the synchronization point. Whenever possible, try to divide up the work into blocks of uniform workloads. Also, try to identify which barrier instruction causes the most stalls, and optimize the code executed before that synchronization point first.
Dispatch	5.2+	Warp was stalled waiting on a dispatch stall. A warp stalled during dispatch has an instruction ready to issue, but the dispatcher holds back issuing the warp due to other conflicts or events.
Drain	5.2+	Warp was stalled after EXIT waiting for all memory instructions to complete so that warp resources can be freed. A high number of stalls due to draining warps typically occurs when a lot of data is written to memory towards the end of a kernel. Make sure the memory access patterns of these store operations are optimal for the target architecture and consider parallelized data reduction, if applicable.

State	Hardware Support	Description
IMC Miss	5.2+	Warp was stalled waiting for an immediate constant cache (IMC) miss. A read from constant memory costs one memory read from device memory only on a cache miss; otherwise, it just costs one read from the constant cache. Accesses to different addresses by threads within a warp are serialized, thus the cost scales linearly with the number of unique addresses read by all threads within a warp. As such, the constant cache is best when threads in the same warp access only a few distinct locations. If all threads of a warp access the same location, then constant memory can be as fast as a register access.
LG Throttle	7.0+	Warp was stalled waiting for the L1 instruction queue for local and global (LG) memory operations to be not full. Typically this stall occurs only when executing local or global memory instructions extremely frequently. If applicable, consider combining multiple lower-width memory operations into fewer wider memory operations and try interleaving memory operations and math instructions.
Long Scoreboard	5.2+	Warp was stalled waiting for a scoreboard dependency on a L1TEX (local, global, surface, tex) operation. To reduce the number of cycles waiting on L1TEX data accesses verify the memory access patterns are optimal for the target architecture, attempt to increase cache hit rates by increasing data locality, or by changing the cache configuration, and consider moving frequently used data to shared memory.
Math Pipe Throttle	5.2+	Warp was stalled waiting for the execution pipe to be available. This stall occurs when all active warps execute their next instruction on a specific, oversubscribed math pipeline. Try to increase the number of active warps to hide the existent latency or try changing the instruction mix to utilize all available pipelines in a more balanced way.
Membar	5.2+	Warp was stalled waiting on a memory barrier. Avoid executing any unnecessary memory barriers and assure that any outstanding memory operations are fully optimized for the target architecture.
MIO Throttle	5.2+	Warp was stalled waiting for the MIO (memory input/output) instruction queue to be not full. This stall reason is high in cases of extreme utilization of the MIO pipelines, which include special math instructions, dynamic branches, as well as shared memory instructions.
Misc	5.2+	Warp was stalled for a miscellaneous hardware reason.
No Instructions	5.2+	Warp was stalled waiting to be selected to fetch an instruction or waiting on an instruction cache miss. A high number of warps not having an instruction fetched is typical for very short kernels with less than one full wave of work in the grid. Excessively jumping across large blocks of assembly code can also lead to more warps stalled for this reason.
Not Selected	5.2+	Warp was stalled waiting for the microscheduler to select the warp to issue. Not selected warps are eligible warps that were not picked by the scheduler to issue that cycle as

State	Hardware Support	Description
		another warp was selected. A high number of not selected warps typically means you have sufficient warps to cover warp latencies and you may consider reducing the number of active warps to possibly increase cache coherence and data locality.
Selected	5.2+	Warp was selected by the microscheduler and issued an instruction.
Short Scoreboard	5.2+	Warp was stalled waiting for a scoreboard dependency on a MIO (memory input/output) operation (not to L1TEX). The primary reason for a high number of stalls due to short scoreboards is typically memory operations to shared memory. Other reasons include frequent execution of special math instructions (e.g. MUFU) or dynamic branching (e.g. BRX, JMX). Verify if there are shared memory operations and reduce bank conflicts, if applicable.
Sleeping	7.0+	Warp was stalled due to all threads in the warp being in the blocked, yielded, or sleep state. Reduce the number of executed NANOSLEEP instructions, lower the specified time delay, and attempt to group threads in a way that multiple threads in a warp sleep at the same time.
Tex Throttle	5.2+	Warp was stalled waiting for the L1 instruction queue for texture operations to be not full. This stall reason is high in cases of extreme utilization of the L1TEX pipeline. If applicable, consider combining multiple lower-width memory operations into fewer wider memory operations and try interleaving memory operations and math instructions.
Wait	5.2+	Warp was stalled waiting on a fixed latency execution dependency. Typically, this stall reason should be very low and only shows up as a top contributor in already highly optimized kernels. If possible, try to further increase the number of active warps to hide the corresponding instruction latencies.

# Appendix B.

## SECTIONS AND RULES

Table 7 Available Sections

Identifier	Filename	Description
ComputeWorkloadAnalysis	ComputeWorkloadAnalysis	Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.
InstructionStats	InstructionStatistics.section	Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution.
LaunchStats	LaunchStatistics.section	Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.
MemoryWorkloadAnalysis	MemoryWorkloadAnalysis	Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Depending on the limiting factor, the memory chart and tables allow to identify the exact bottleneck in the memory system.
Occupancy	Occupancy.section	Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical

Identifier	Filename	Description
		and the achieved occupancy during execution typically indicates highly imbalanced workloads.
SchedulerStats	SchedulerStatistics.sections	Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.
SourceCounters	SourceCounters.sections	Source metrics, including PC sampling information.
SpeedOfLight	SpeedOfLight.sections	High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. On Volta+ GPUs, it reports the breakdown of <i>SOL SM</i> and <i>SOL Memory</i> to each individual sub-metric to clearly identify the highest contributor.
WarpStateStats	WarpStateStatistics.sections	Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle.

## **Notice**

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## **Trademarks**

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## **Copyright**

© 2018-2020 NVIDIA Corporation. All rights reserved.

This product includes software developed by the Syncro Soft SRL (<http://www.sync.ro/>).