NAME

nvidia−smi − NVIDIA System Management Interface program

SYNOPSIS

nvidia-smi [OPTION1 [ARG1]] [OPTION2 [ARG2]] ...

DESCRIPTION

nvidia-smi (also NVSMI) provides monitoring and management capabilities for each of NVIDIA's Tesla, Quadro, GRID and GeForce devices from Fermi and higher architecture families. GeForce Titan series devices are supported for most functions with very limited information provided for the remainder of the Geforce brand. NVSMI is a cross platform tool that supports all standard NVIDIA driver-supported Linux distros, as well as 64bit versions of Windows starting with Windows Server 2008 R2. Metrics can be consumed directly by users via stdout, or provided by file via CSV and XML formats for scripting purposes.

Note that much of the functionality of NVSMI is provided by the underlying NVML C-based library. See the NVIDIA developer website link below for more information about NVML. NVML-based python bindings are also available.

The output of NVSMI is not guaranteed to be backwards compatible. However, both NVML and the Python bindings are backwards compatible, and should be the first choice when writing any tools that must be maintained across NVIDIA driver releases.

NVML SDK: *http://developer.nvidia.com/nvidia-management-library-nvml/*

Python bindings: *http://pypi.python.org/pypi/nvidia-ml-py/*

OPTIONS

GENERAL OPTIONS

−h, −−help

Print usage information and exit.

SUMMARY OPTIONS

−L, −−list−gpus

List each of the NVIDIA GPUs in the system, along with their UUIDs.

QUERY OPTIONS

−q, −−query

Display GPU or Unit info. Displayed info includes all data listed in the (*GPU ATTRIBUTES*) or (*UNIT ATTRIBUTES*) sections of this document. Some devices and/or environments don't support all possible information. Any unsupported data is indicated by a "N/A" in the output. By default information for all available GPUs or Units is displayed. Use the **−i** option to restrict the output to a single GPU or Unit.

[plus optional]

−u, −−unit

Display Unit data instead of GPU data. Unit data is only available for NVIDIA S−class Tesla enclosures.

−i, −−id=ID

Display data for a single specified GPU or Unit. The specified id may be the GPU/Unit's 0−based index in the natural enumeration returned by the driver, the GPU's board serial number, the GPU's UUID, or the GPU's PCI bus ID (as domain:bus:device.function in hex). It is recommended that users desiring consistency use either UUID or PCI bus ID, since device enumeration ordering is not guaranteed to be consistent between reboots and board serial number might be shared between multiple GPUs on the same board.

−f FILE, −−filename=FILE
Redirect query output to the specified file in place of the default stdout. The specified file will be overwritten.

−x, −−xml−format
Produce XML output in place of the default human−readable format. Both GPU and Unit query outputs conform to corresponding DTDs. These are available via the **−−dtd** flag.

−−dtd
Use with **−x**. Embed the DTD in the XML output.

−−debug=FILE
Produces an encrypted debug log for use in submission of bugs back to NVIDIA.

−d TYPE, −−display=TYPE
Display only selected information: MEMORY, UTILIZATION, ECC, TEMPERATURE, POWER, CLOCK, COMPUTE, PIDS, PERFORMANCE, SUPPORTED_CLOCKS, PAGE_RETIREMENT, ACCOUNTING Flags can be combined with comma e.g. "MEMORY,ECC". Sampling data with max, min and avg is also returned for POWER, UTILIZATION and CLOCK display types. Doesn't work with -u/--unit or -x/--xml-format flags.

−l SEC, −−loop=SEC
Continuously report query data at the specified interval, rather than the default of just once. The application will sleep in−between queries. Note that on Linux ECC error or XID error events will print out during the sleep period if the *-x* flag was not specified. Pressing Ctrl+C at any time will abort the loop, which will otherwise run indefinitely. If no argument is specified for the **−l** form a default interval of 5 seconds is used.

SELECTIVE QUERY OPTIONS
Allows the caller to pass an explicit list of properties to query.

[one of]

−−query−gpu=
Information about GPU. Pass comma separated list of properties you want to query. e.g. −−query−gpu=pci.bus_id,persistence_mode. Call −−help−query−gpu for more info.

−−query−supported−clocks=
List of supported clocks. Call −−help−query−supported−clocks for more info.

−−query−compute−apps=
List of currently active compute processes. Call −−help−query−compute−apps for more info.

−−query−accounted−apps=
List of accounted compute processes. Call −−help−query−accounted−apps for more info.

−−query−retired−pages=
List of GPU device memory pages that have been retired. Call −−help−query−retired−pages for more info.

[mandatory]
−−format=
Comma separated list of format options:

- csv - comma separated values (MANDATORY)
- noheader - skip first line with column headers
- nounits - don't print units for numerical values

[plus any of]
−i, −−id=ID
Display data for a single specified GPU. The specified id may be the GPU's 0−based index in the natural enumeration returned by the driver, the GPU's board serial number, the GPU's UUID, or the GPU's PCI bus ID (as domain:bus:device.function in hex). It is recommended that users desiring consistency use either UUID or PCI bus ID, since device enumeration ordering is not guaranteed to be consistent between reboots and board serial number might be shared between multiple GPUs on the same board.

−f FILE, −−filename=FILE
Redirect query output to the specified file in place of the default stdout. The specified file will be overwritten.

−l SEC, −−loop=SEC
Continuously report query data at the specified interval, rather than the default of just once. The application will sleep in−between queries. Note that on Linux ECC error or XID error events will print out during the sleep period if the *-x* flag was not specified. Pressing Ctrl+C at any time will abort the loop, which will otherwise run indefinitely. If no argument is specified for the **−l** form a default interval of 5 seconds is used.

−lms ms, −−loop−ms=ms
Same as −l,−−loop but in milliseconds.

DEVICE MODIFICATION OPTIONS
[any one of]
−pm, −−persistence−mode=MODE
Set the persistence mode for the target GPUs. See the (*GPU ATTRIBUTES*) section for a description of persistence mode. Requires root. Will impact all GPUs unless a single GPU is specified using the *−i* argument. The effect of this operation is immediate. However, it does not persist across reboots. After each reboot persistence mode will default to "Disabled". Available on Linux only.

−e, −−ecc−config=CONFIG
Set the ECC mode for the target GPUs. See the (*GPU ATTRIBUTES*) section for a description of ECC mode. Requires root. Will impact all GPUs unless a single GPU is specified using the *−i* argument. This setting takes effect after the next reboot and is persistent.

−p, −−reset−ecc−errors=TYPE
Reset the ECC error counters for the target GPUs. See the (*GPU ATTRIBUTES*) section for a description of ECC error counter types. Available arguments are 0|VOLATILE or 1|AGGREGATE. Requires root. Will impact all GPUs unless a single GPU is specified using the *−i* argument. The effect of this operation is immediate.

−c, −−compute−mode=MODE
Set the compute mode for the target GPUs. See the (*GPU ATTRIBUTES*) section for a description of compute mode. Requires root. Will impact all GPUs unless a single GPU is specified using the *−i* argument. The effect of this operation is immediate. However, it does not persist across reboots. After each reboot compute mode will reset to "DEFAULT".

−dm TYPE, −−driver−model=TYPE
−fdm TYPE, −−force−driver−model=TYPE
Enable or disable TCC driver model. For Windows only. Requires administrator privileges. *−dm* will fail if a display is attached, but *−fdm* will force the driver model to change. Will impact all GPUs unless a single GPU is specified using the *−i* argument. A reboot is required for the change to take place. See **Driver Model** for more information on Windows driver models.

−−gom=MODE
Set GPU Operation Mode: 0/ALL_ON, 1/COMPUTE, 2/LOW_DP Supported on GK110 M-class and X-class Tesla products from the Kepler family. Not supported on Quadro and Tesla C-class products. LOW_DP and ALL_ON are the only modes supported on GeForce Titan devices. Requires administrator privileges. See *GPU Operation Mode* for more information about GOM. GOM changes take effect after reboot. The reboot requirement might be removed in the future. Compute only GOMs don’t support WDDM (Windows Display Driver Model)

−r, −−gpu−reset
Trigger a reset of one or more GPUs. Can be used to clear GPU HW and SW state in situations that would otherwise require a machine reboot. Typically useful if a double bit ECC error has occurred. Optional *−i* switch can be used to target one or more specific devices. Without this option, all GPUs are reset. Requires root. There can't be any applications using these devices (e.g. CUDA application, graphics application like X server, monitoring application like other instance of nvidia-smi). There also can't be any compute applications running on any other GPU in the system.

Any GPUs with NVLink connections to a GPU being reset must also be reset in the same command. This can be done either by omitting the *−i* switch, or using the *−i* switch to specify the GPUs to be reset. If the *−i* option does not specify a complete set of NVLink GPUs to reset, this command will issue an error identifying the additional GPUs that must be included in the reset command.

GPU reset is not guaranteed to work in all cases. It is not recommended for production environments at this time. In some situations there may be HW components on the board that fail to revert back to an initial state following the reset request. This is more likely to be seen on Fermi-generation products vs. Kepler, and more likely to be seen if the reset is being performed on a hung GPU.

Following a reset, it is recommended that the health of each reset GPU be verified before further use. The nvidia-healthmon tool is a good choice for this test. If any GPU is not healthy a complete reset should be instigated by power cycling the node.

Visit *http://developer.nvidia.com/gpu-deployment-kit* to download the GDK and nvidia-healthmon.

−ac, −−applications−clocks=MEM_CLOCK,GRAPHICS_CLOCK
Specifies maximum <memory,graphics> clocks as a pair (e.g. 2000,800) that defines GPU’s speed while running applications on a GPU. Supported on Maxwell-based GeForce and from the Kepler+ family in Tesla/Quadro/Titan devices. Requires root unless restrictions are relaxed with the −acp command..

−rac, −−reset−applications−clocks
Resets the applications clocks to the default value. Supported on Maxwell-based GeForce and from the Kepler+ family in Tesla/Quadro/Titan devices. Requires root unless restrictions are relaxed with the −acp command.

−acp, −−applications−clocks−permission=MODE
Toggle whether applications clocks can be changed by all users or only by root. Available arguments are 0|UNRESTRICTED, 1|RESTRICTED. Supported on Maxwell-based GeForce and from the Kepler+ family in Tesla/Quadro/Titan devices. Requires root.

−pl, −−power−limit=POWER_LIMIT
Specifies maximum power limit in watts. Accepts integer and floating point numbers. Only on supported devices from Kepler family. Requires administrator privileges. Value needs to be between Min and Max Power Limit as reported by nvidia-smi.

−am, −−accounting−mode=MODE
Enables or disables GPU Accounting. With GPU Accounting one can keep track of usage of resources throughout lifespan of a single process. Only on supported devices from Kepler family. Requires administrator privileges. Available arguments are 0|DISABLED or 1|ENABLED.

−caa, −−clear−accounted−apps
Clears all processes accounted so far. Only on supported devices from Kepler family. Requires administrator privileges.

−−auto−boost−default=MODE
Set the default auto boost policy to 0/DISABLED or 1/ENABLED, enforcing the change only after the last boost client has exited. Only on certain Tesla devices from the Kepler+ family and Maxwell-based GeForce devices. Requires root.

−−auto−boost−default−force=MODE
Set the default auto boost policy to 0/DISABLED or 1/ENABLED, enforcing the change immediately. Only on certain Tesla devices from the Kepler+ family and Maxwell-based GeForce devices. Requires root.

−−auto−boost−permission=MODE
Allow non-admin/root control over auto boost mode. Available arguments are 0|UNRESTRICTED, 1|RESTRICTED. Only on certain Tesla devices from the Kepler+ family and Maxwell-based GeForce devices. Requires root.

[plus optional]
−i, −−id=ID
Modify a single specified GPU. The specified id may be the GPU/Unit's 0−based index in the natural enumeration returned by the driver, the GPU's board serial number, the GPU's UUID, or the GPU's PCI bus ID (as domain:bus:device.function in hex). It is recommended that users desiring consistency use either UUID or PCI bus ID, since device enumeration ordering is not guaranteed to be consistent between reboots and board serial number might be shared between multiple GPUs on the same board.

UNIT MODIFICATION OPTIONS

−t, −−toggle−led=STATE
Set the LED indicator state on the front and back of the unit to the specified color. See the (*UNIT ATTRIBUTES*) section for a description of the LED states. Allowed colors are 0|GREEN and 1|AMBER. Requires root.

[plus optional]
−i, −−id=ID
Modify a single specified Unit. The specified id is the Unit's 0-based index in the natural enumeration returned by the driver.

SHOW DTD OPTIONS
−−dtd
Display Device or Unit DTD.

[plus optional]
−f FILE, −−filename=FILE
Redirect query output to the specified file in place of the default stdout. The specified file will be overwritten.

−u, −−unit
Display Unit DTD instead of device DTD.

stats
Display statistics information about the GPU. Use "nvidia-smi stats -h" for more information. Linux only.

topo
Display topology information about the system. Use "nvidia-smi topo -h" for more information. Linux only. Shows all GPUs NVML is able to detect but CPU affinity information will only be shown for GPUs with Kepler or newer architectures. Note: GPU enumeration is the same as NVML.

drain
Display and modify the GPU drain states. Use "nvidia-smi drain -h" for more information. Linux only.

nvlink
Display nvlink information. Use "nvidia-smi nvlink -h" for more information.

clocks
Query and control clocking behavior. Currently, this only pertains to synchronized boost. Use "nvidia-smi clocks --help" for more information.

vgpu
Display information on GRID virtual GPUs. Use "nvidia-smi vgpu -h" for more information.

RETURN VALUE
Return code reflects whether the operation succeeded or failed and what was the reason of failure.

- Return code 0 − Success
- Return code 2 − A supplied argument or flag is invalid

- Return code 3 − The requested operation is not available on target device
- Return code 4 − The current user does not have permission to access this device or perform this operation
- Return code 6 − A query to find an object was unsuccessful
- Return code 8 − A device's external power cables are not properly attached
- Return code 9 − NVIDIA driver is not loaded
- Return code 10 − NVIDIA Kernel detected an interrupt issue with a GPU
- Return code 12 − NVML Shared Library couldn't be found or loaded
- Return code 13 − Local version of NVML doesn't implement this function
- Return code 14 − infoROM is corrupted
- Return code 15 − The GPU has fallen off the bus or has otherwise become inaccessible
- Return code 255 − Other error or internal driver error occurred

GPU ATTRIBUTES

The following list describes all possible data returned by the **−q** device query option. Unless otherwise noted all numerical results are base 10 and unitless.

Timestamp

The current system timestamp at the time nvidia−smi was invoked. Format is "Day−of−week Month Day HH:MM:SS Year".

Driver Version

The version of the installed NVIDIA display driver. This is an alphanumeric string.

Attached GPUs

The number of NVIDIA GPUs in the system.

Product Name

The official product name of the GPU. This is an alphanumeric string. For all products.

Display Mode

A flag that indicates whether a physical display (e.g. monitor) is currently connected to any of the GPU's connectors. "Enabled" indicates an attached display. "Disabled" indicates otherwise.

Display Active

A flag that indicates whether a display is initialized on the GPU's (e.g. memory is allocated on the device for display). Display can be active even when no monitor is physically attached. "Enabled" indicates an active display. "Disabled" indicates otherwise.

Persistence Mode

A flag that indicates whether persistence mode is enabled for the GPU. Value is either "Enabled" or "Disabled". When persistence mode is enabled the NVIDIA driver remains loaded even when no active clients, such as X11 or nvidia-smi, exist. This minimizes the driver load latency associated with running dependent apps, such as CUDA programs. For all CUDA-capable products. Linux only.

Accounting Mode
A flag that indicates whether accounting mode is enabled for the GPU Value is either When accounting is enabled statistics are calculated for each compute process running on the GPU. Statistics can be queried during the lifetime or after termination of the process. The execution time of process is reported as 0 while the process is in running state and updated to actual execution time after the process has terminated. See --help-query-accounted-apps for more info.

Accounting Mode Buffer Size
Returns the size of the circular buffer that holds list of processes that can be queried for accounting stats. This is the maximum number of processes that accounting information will be stored for before information about oldest processes will get overwritten by information about new processes.

Driver Model
On Windows, the TCC and WDDM driver models are supported. The driver model can be changed with the (*−dm*) or (*−fdm*) flags. The TCC driver model is optimized for compute applications. I.E. kernel launch times will be quicker with TCC. The WDDM driver model is designed for graphics applications and is not recommended for compute applications. Linux does not support multiple driver models, and will always have the value of "N/A".

Current The driver model currently in use. Always "N/A" on Linux.

Pending The driver model that will be used on the next reboot. Always "N/A" on Linux.

Serial Number
This number matches the serial number physically printed on each board. It is a globally unique immutable alphanumeric value.

GPU UUID
This value is the globally unique immutable alphanumeric identifier of the GPU. It does not correspond to any physical label on the board.

Minor Number
The minor number for the device is such that the Nvidia device node file for each GPU will have the form /dev/nvidia[minor number]. Available only on Linux platform.

VBIOS Version
The BIOS of the GPU board.

MultiGPU Board
Whether or not this GPU is part of a multiGPU board.

Board ID
The unique board ID assigned by the driver. If two or more GPUs have the same board ID and the above "MultiGPU" field is true then the GPUs are on the same board.

Inforom Version
Version numbers for each object in the GPU board's inforom storage. The inforom is a small, persistent store of configuration and state data for the GPU. All inforom version fields are numerical. It can be useful to know these version numbers because some GPU features are only available with inforoms of a certain version or higher.

If any of the fields below return Unknown Error additional Inforom verification check is performed and appropriate warning message is displayed.

Image Version Global version of the infoROM image. Image version just like VBIOS version uniquely describes the exact version of the infoROM flashed on the board in contrast to infoROM object version which is only an indicator of supported features.

OEM Object Version for the OEM configuration data.

ECC Object Version for the ECC recording data.

Power Object Version for the power management data.

GPU Operation Mode

GOM allows to reduce power usage and optimize GPU throughput by disabling GPU features.

Each GOM is designed to meet specific user needs.

In "All On" mode everything is enabled and running at full speed.

The "Compute" mode is designed for running only compute tasks. Graphics operations are not allowed.

The "Low Double Precision" mode is designed for running graphics applications that don’t require high bandwidth double precision.

GOM can be changed with the (*−−gom*) flag.

Supported on GK110 M-class and X-class Tesla products from the Kepler family. Not supported on Quadro and Tesla C-class products. Low Double Precision and All On modes are the only modes available for supported GeForce Titan products.

Current The GOM currently in use.

Pending The GOM that will be used on the next reboot.

PCI

Basic PCI info for the device. Some of this information may change whenever cards are added/removed/moved in a system. For all products.

Bus PCI bus number, in hex

Device PCI device number, in hex

Domain PCI domain number, in hex

Device Id PCI vendor device id, in hex

Sub System Id PCI Sub System id, in hex

Bus Id PCI bus id as "domain:bus:device.function", in hex

GPU Link information
The PCIe link generation and bus width

Current The current link generation and width. These may be reduced when the GPU is not in use.

Maximum The maximum link generation and width possible with this GPU and system configuration. For example, if the GPU supports a higher PCIe generation than the system supports then this reports the system PCIe generation.

Bridge Chip
Information related to Bridge Chip on the device. The bridge chip firmware is only present on certain boards and may display "N/A" for some newer multiGPUs boards.

Type The type of bridge chip. Reported as N/A if doesn't exist.

Firmware Version
The firmware version of the bridge chip. Reported as N/A if doesn't exist.

Replay counter
This is the internal counter that records various errors on the PCIe bus.

Tx Throughput
The GPU-centric transmission throughput across the PCIe bus in MB/s over the past 20ms. Only supported on Maxwell architectures and newer.

Rx Throughput
The GPU-centric receive throughput across the PCIe bus in MB/s over the past 20ms. Only supported on Maxwell architectures and newer.

Fan Speed
The fan speed value is the percent of maximum speed that the device's fan is currently intended to run at. It ranges from 0 to 100%. Note: The reported speed is the intended fan speed. If the fan is physically blocked and unable to spin, this output will not match the actual fan speed. Many parts do not report fan speeds because they rely on cooling via fans in the surrounding enclosure. For all discrete products with dedicated fans.

Performance State
The current performance state for the GPU. States range from P0 (maximum performance) to P12 (minimum performance).

Clocks Throttle Reasons
Retrieves information about factors that are reducing the frequency of clocks.

If all throttle reasons are returned as "Not Active" it means that clocks are running as high as possible.

Idle Nothing is running on the GPU and the clocks are dropping to Idle state. This limiter may be removed in a later release.

Application Clocks Setting
GPU clocks are limited by applications clocks setting. E.g. can be changed using nvidia−smi −−applications−clocks=

SW Power Cap SW Power Scaling algorithm is reducing the clocks below requested clocks because the GPU is consuming too much power. E.g. SW power cap limit can be changed with nvidia−smi −−power−limit=

HW Slowdown HW Slowdown (reducing the core clocks by a factor of 2 or more) is engaged. HW Thermal Slowdown and HW Power Brake will be displayed on Pascal+.

This is an indicator of:
* Temperature being too high (HW Thermal Slowdown)
* External Power Brake Assertion is triggered (e.g. by the system power supply) (HW Power Brake Slowdown)
* Power draw is too high and Fast Trigger protection is reducing the clocks

SW Thermal Slowdown
SW Thermal capping algorithm is reducing clocks below requested clocks because GPU temperature is higher than Max Operating Temp

FB Memory Usage
On-board frame buffer memory information. Reported total memory is affected by ECC state. If ECC is enabled the total available memory is decreased by several percent, due to the requisite parity bits. The driver may also reserve a small amount of memory for internal use, even without active work on the GPU. For all products.

Total Total size of FB memory.

Used Used size of FB memory.

Free Available size of FB memory.

BAR1 Memory Usage
BAR1 is used to map the FB (device memory) so that it can be directly accessed by the CPU or by 3rd party devices (peer-to-peer on the PCIe bus).

Total Total size of BAR1 memory.

Used Used size of BAR1 memory.

Free Available size of BAR1 memory.

Compute Mode
The compute mode flag indicates whether individual or multiple compute applications may run on the GPU.

"Default" means multiple contexts are allowed per device.

"Exclusive Process" means only one context is allowed per device, usable from multiple threads at a time.

"Prohibited" means no contexts are allowed per device (no compute apps).

"EXCLUSIVE_PROCESS" was added in CUDA 4.0. Prior CUDA releases supported only one exclusive mode, which is equivalent to "EXCLUSIVE_THREAD" in CUDA 4.0 and beyond.

For all CUDA-capable products.

Utilization

Utilization rates report how busy each GPU is over time, and can be used to determine how much an application is using the GPUs in the system.

Note: During driver initialization when ECC is enabled one can see high GPU and Memory Utilization readings. This is caused by ECC Memory Scrubbing mechanism that is performed during driver initialization.

GPU Percent of time over the past sample period during which one or more kernels was executing on the GPU. The sample period may be between 1 second and 1/6 second depending on the product.

Memory Percent of time over the past sample period during which global (device) memory was being read or written. The sample period may be between 1 second and 1/6 second depending on the product.

Encoder Percent of time over the past sample period during which the GPU's video encoder was being used. The sampling rate is variable and can be obtained directly via the nvmlDeviceGetEncoderUtilization() API

Decoder Percent of time over the past sample period during which the GPU's video decoder was being used. The sampling rate is variable and can be obtained directly via the nvmlDeviceGetDecoderUtilization() API

Ecc Mode

A flag that indicates whether ECC support is enabled. May be either "Enabled" or "Disabled". Changes to ECC mode require a reboot. Requires Inforom ECC object version 1.0 or higher.

Current The ECC mode that the GPU is currently operating under.

Pending The ECC mode that the GPU will operate under after the next reboot.

ECC Errors

NVIDIA GPUs can provide error counts for various types of ECC errors. Some ECC errors are either single or double bit, where single bit errors are corrected and double bit errors are uncorrectable. Texture memory errors may be correctable via resend or uncorrectable if the resend fails. These errors are available across two timescales (volatile and aggregate). Single bit ECC errors are automatically corrected by the HW and do not result in data corruption. Double bit errors are detected but not corrected. Please see the ECC documents on the web for information on compute application behavior when double bit errors occur. Volatile error counters track the number of errors detected since the last driver load. Aggregate error counts persist indefinitely and thus act as a lifetime counter.

A note about volatile counts: On Windows this is once per boot. On Linux this can be more frequent. On

Linux the driver unloads when no active clients exist. Hence, if persistence mode is enabled or there is always a driver client active (e.g. X11), then Linux also sees per-boot behavior. If not, volatile counts are reset each time a compute app is run.

Tesla and Quadro products from the Fermi and Kepler family can display total ECC error counts, as well as a breakdown of errors based on location on the chip. The locations are described below. Location−based data for aggregate error counts requires Inforom ECC object version 2.0. All other ECC counts require ECC object version 1.0.

Device Memory Errors detected in global device memory.

Register File Errors detected in register file memory.

L1 Cache Errors detected in the L1 cache.

L2 Cache Errors detected in the L2 cache.

Texture Memory
Parity errors detected in texture memory.

Total Total errors detected across entire chip. Sum of **Device Memory**, **Register File**, **L1 Cache**, **L2 Cache** and **Texture Memory**.

Page Retirement
NVIDIA GPUs can retire pages of GPU device memory when they become unreliable. This can happen when multiple single bit ECC errors occur for the same page, or on a double bit ECC error. When a page is retired, the NVIDIA driver will hide it such that no driver, or application memory allocations can access it.

Double Bit ECC The number of GPU device memory pages that have been retired due to a double bit ECC error.

Single Bit ECC The number of GPU device memory pages that have been retired due to multiple single bit ECC errors.

Pending Checks if any GPU device memory pages are pending retirement on the next reboot. Pages that are pending retirement can still be allocated, and may cause further reliability issues.

Temperature
Readings from temperature sensors on the board. All readings are in degrees C. Not all products support all reading types. In particular, products in module form factors that rely on case fans or passive cooling do not usually provide temperature readings. See below for restrictions.

GPU Core GPU temperature. For all discrete and S-class products.

Shutdown Temp The temperature at which a GPU will shutdown.

Slowdown Temp
The temperature at which a GPU will begin slowing itself down through HW, in order to cool.

Max Operating Temp
The temperature at which a GPU will begin slowing itself down through SW, in order to cool.

Power Readings
Power readings help to shed light on the current power usage of the GPU, and the factors that affect that usage. When power management is enabled the GPU limits power draw under load to fit within a predefined power envelope by manipulating the current performance state. See below for limits of availability. Please note that power readings are not applicable for Pascal and higher GPUs with BA sensor boards.

Power State Power State is deprecated and has been renamed to Performance State in 2.285. To maintain XML compatibility, in XML format Performance State is listed in both places.

Power Management
A flag that indicates whether power management is enabled. Either "Supported" or "N/A". Requires Inforom PWR object version 3.0 or higher or Kepler device.

Power Draw The last measured power draw for the entire board, in watts. Only available if power management is supported. This reading is accurate to within +/- 5 watts. Requires Inforom PWR object version 3.0 or higher or Kepler device. Please note that for boards without INA sensors, this refers to the power draw for the GPU and not for the entire board.

Power Limit The software power limit, in watts. Set by software such as nvidia-smi. Only available if power management is supported. Requires Inforom PWR object version 3.0 or higher or Kepler device. On Kepler devices Power Limit can be adjusted using −pl,−−power−limit= switches.

Enforced Power Limit
The power management algorithm's power ceiling, in watts. Total board power draw is manipulated by the power management algorithm such that it stays under this value. This limit is the minimum of various limits such as the software limit listed above. Only available if power management is supported. Requires a Kepler device. Please note that for boards without INA sensors, it is the GPU power draw that is being manipulated.

Default Power Limit
The default power management algorithm's power ceiling, in watts. Power Limit will be set back to Default Power Limit after driver unload. Only on supported devices from Kepler family.

Min Power Limit
The minimum value in watts that power limit can be set to. Only on supported devices from Kepler family.

Max Power Limit
The maximum value in watts that power limit can be set to. Only on supported devices from Kepler family.

Clocks
Current frequency at which parts of the GPU are running. All readings are in MHz.

Graphics Current frequency of graphics (shader) clock.

SM Current frequency of SM (Streaming Multiprocessor) clock.

Memory Current frequency of memory clock.

Video Current frequency of video (encoder + decoder) clocks.

Applications Clocks
User specified frequency at which applications will be running at. Can be changed with [−ac | −−applications−clocks] switches.

Graphics User specified frequency of graphics (shader) clock.

Memory User specified frequency of memory clock.

Default Applications Clocks
Default frequency at which applications will be running at. Application clocks can be changed with [−ac | −−applications−clocks] switches. Application clocks can be set to default using [−rac | −−reset−applications−clocks] switches.

Graphics Default frequency of applications graphics (shader) clock.

Memory Default frequency of applications memory clock.

Max Clocks
Maximum frequency at which parts of the GPU are design to run. All readings are in MHz.

On GPUs from Fermi family current P0 clocks (reported in Clocks section) can differ from max clocks by few MHz.

Graphics Maximum frequency of graphics (shader) clock.

SM Maximum frequency of SM (Streaming Multiprocessor) clock.

Memory Maximum frequency of memory clock.

Video Maximum frequency of video (encoder + decoder) clock.

Clock Policy
User-specified settings for automated clocking changes such as auto boost.

Auto Boost Indicates whether auto boost mode is currently enabled for this GPU (On) or disabled for this GPU (Off). Shows (N/A) if boost is not supported. Auto boost allows dynamic GPU clocking based on power, thermal and utilization. When auto boost is disabled the GPU

will attempt to maintain clocks at precisely the Current Application Clocks settings (whenever a CUDA context is active). With auto boost enabled the GPU will still attempt to maintain this floor, but will opportunistically boost to higher clocks when power, thermal and utilization headroom allow. This setting persists for the life of the CUDA context for which it was requested. Apps can request a particular mode either via an NVML call (see NVML SDK) or by setting the CUDA environment variable CUDA_AUTO_BOOST.

Auto Boost Default

Indicates the default setting for auto boost mode, either enabled (On) or disabled (Off). Shows (N/A) if boost is not supported. Apps will run in the default mode if they have not explicitly requested a particular mode. Note: Auto Boost settings can only be modified if "Persistence Mode" is enabled, which is NOT by default.

Supported clocks

List of possible memory and graphics clocks combinations that the GPU can operate on (not taking into account HW brake reduced clocks). These are the only clock combinations that can be passed to −−applications−clocks flag. Supported Clocks are listed only when −q −d SUPPORTED_CLOCKS switches are provided or in XML format.

Processes

List of processes having Compute or Graphics Context on the device. Compute processes are reported on all the fully supported products. Reporting for Graphics processes is limited to the supported products starting with Kepler architecture.

Each Entry is of format "<GPU Index> <PID> <Type> <Process Name> <GPU Memory Usage>"

GPU Index Represents NVML Index of the device.

PID Represents Process ID corresponding to the active Compute or Graphics context.

Type Displayed as "C" for Compute Process, "G" for Graphics Process, and "C+G" for the process having both Compute and Graphics contexts.

Process Name Represents process name for the Compute or Graphics process.

GPU Memory Usage

Amount of memory used on the device by the context. Not available on Windows when running in WDDM mode because Windows KMD manages all the memory not NVIDIA driver.

Stats (EXPERIMENTAL)

List GPU statistics such as power samples, utilization samples, xid events, clock change events and violation counters.

Supported on Tesla, GRID and Quadro based products under Linux.

Limited to Kepler or newer GPUs.

Displays statistics in CSV format as follows:

<GPU device index>, <metric name>, <CPU Timestamp in us>, <value for metric>

The metrics to display with their units are as follows:

Power samples in Watts.

GPU Temperature samples in degrees Celsius.

GPU, Memory, Encoder and Decoder utilization samples in Percentage.

Xid error events reported with Xid error code. The error code is 999 for unknown xid error.

Processor and Memory clock changes in MHz.

Violation due to Power capping with violation time in ns. (Tesla Only)

Violation due to Thermal capping with violation boolean flag (1/0). (Tesla Only)

Notes:

Any statistic preceded by "#" is a comment.

Non supported device is displayed as "#<device Index>, Device not supported".

Non supported metric is displayed as "<device index>, <metric name>, N/A, N/A".

Violation due to Thermal/Power supported only for Tesla based products. Thermal Violations are limited to Tesla K20 and higher.

Device Monitoring

The "nvidia-smi dmon" command-line is used to monitor one or more GPUs (up to 4 devices) plugged into the system. This tool allows the user to see one line of monitoring data per monitoring cycle. The output is in concise format and easy to interpret in interactive mode. The output data per line is limited by the terminal size. It is supported on Tesla, GRID, Quadro and limited GeForce products for Kepler or newer GPUs under bare metal 64 bits Linux. By default, the monitoring data includes Power Usage, Temperature, SM clocks, Memory clocks and Utilization values for SM, Memory, Encoder and Decoder. It can also be configured to report other metrics such as frame buffer memory usage, bar1 memory usage, power/thermal violations and aggregate single/double bit ecc errors. If any of the metric is not supported on the device or any other error in fetching the metric is reported as "-" in the output data. The user can also configure monitoring frequency and the number of monitoring iterations for each run. There is also an option to include date and time at each line. All the supported options are exclusive and can be used together in any order.

Usage:

1) Default with no arguments

nvidia-smi dmon

Monitors default metrics for up to 4 supported devices under natural enumeration (starting with GPU index 0) at a frequency of 1 sec. Runs until terminated with ˆC.

2) Select one or more devices

nvidia-smi dmon -i <device1,device2, .. , deviceN>

Reports default metrics for the devices selected by comma separated device list. The tool picks up to 4 supported devices from the list under natural enumeration (starting with GPU index 0).

3) Select metrics to be displayed

nvidia-smi dmon -s <metric_group>

<metric_group> can be one or more from the following:

p - Power Usage (in Watts) and Temperature (in C)

u - Utilization (SM, Memory, Encoder and Decoder Utilization in %)

c - Proc and Mem Clocks (in MHz)

v - Power Violations (in %) and Thermal Violations (as a boolean flag)

m - Frame Buffer and Bar1 memory usage (in MB)

e - ECC (Number of aggregated single bit, double bit ecc errors) and PCIe Replay errors

t - PCIe Rx and Tx Throughput in MB/s (Maxwell and above)

4) Configure monitoring iterations

nvidia-smi dmon -c <number of samples>

Displays data for specified number of samples and exit.

5) Configure monitoring frequency

nvidia-smi dmon -d <time in secs>

Collects and displays data at every specified monitoring interval until terminated with ˆC.

6) Display date

nvidia-smi dmon -o D

Prepends monitoring data with date in YYYYMMDD format.

7) Display time

nvidia-smi dmon -o T

Prepends monitoring data with time in HH:MM:SS format.

8) Help Information

nvidia-smi dmon -h

Displays help information for using the command line.

Daemon (EXPERIMENTAL)

The "nvidia-smi daemon" starts a background process to monitor one or more GPUs plugged in to the system. It monitors the requested GPUs every monitoring cycle and logs the file in compressed format at the user provided path or the default location at /var/log/nvstats/. The log file is created with system's date appended to it and of the format nvstats-YYYYMMDD. The flush operation to the log file is done every alternate monitoring cycle. Daemon also logs it's own PID at /var/run/nvsmi.pid. By default, the monitoring data to persist includes Power Usage, Temperature, SM clocks, Memory clocks and Utilization values for SM, Memory, Encoder and Decoder. The daemon tools can also be configured to record other metrics such as frame buffer memory usage, bar1 memory usage, power/thermal violations and aggregate single/double bit ecc errors.The default monitoring cycle is set to 10 secs and can be configured via command-line. It is supported on Tesla, GRID, Quadro and GeForce products for Kepler or newer GPUs under bare metal 64 bits Linux. The daemon requires root privileges to run, and only supports running a single instance on the system. All of the supported options are exclusive and can be used together in any order.

Usage:

1) Default with no arguments

nvidia-smi daemon

Runs in the background to monitor default metrics for up to 4 supported devices under natural enumeration (starting with GPU index 0) at a frequency of 10 sec. The date stamped log file is created at /var/log/nvstats/.

2) Select one or more devices

nvidia-smi daemon -i <device1,device2, .. , deviceN>

Runs in the background to monitor default metrics for the devices selected by comma separated device list. The tool picks up to 4 supported devices from the list under natural enumeration (starting with GPU index 0).

3) Select metrics to be monitored

nvidia-smi daemon -s <metric_group>

<metric_group> can be one or more from the following:

p - Power Usage (in Watts) and Temperature (in C)

u - Utilization (SM, Memory, Encoder and Decoder Utilization in %)

c - Proc and Mem Clocks (in MHz)

v - Power Violations (in %) and Thermal Violations (as a boolean flag)

m - Frame Buffer and Bar1 memory usage (in MB)

e - ECC (Number of aggregated single bit, double bit ecc errors) and PCIe Replay errors

t - PCIe Rx and Tx Throughput in MB/s (Maxwell and above)

4) Configure monitoring frequency

nvidia-smi daemon -d <time in secs>

Collects data at every specified monitoring interval until terminated.

5) Configure log directory

nvidia-smi daemon -p <path of directory>

The log files are created at the specified directory.

6) Configure log file name

nvidia-smi daemon -j <string to append log file name>

The command-line is used to append the log file name with the user provided string.

7) Terminate the daemon

nvidia-smi daemon -t

This command-line uses the stored PID (at /var/run/nvsmi.pid) to terminate the daemon. It makes the best effort to stop the daemon and offers no guarantees for it's termination. In case the daemon is not terminated, then the user can manually terminate by sending kill signal to the daemon. Performing a GPU reset operation (via nvidia-smi) requires all GPU processes to be exited, including the daemon. Users who have the daemon open will see an error to the effect that the GPU is busy.

8) Help Information

nvidia-smi daemon -h

Displays help information for using the command line.

Replay Mode (EXPERIMENTAL)

The "nvidia-smi replay" command-line is used to extract/replay all or parts of log file generated by the daemon. By default, the tool tries to pull the metrics such as Power Usage, Temperature, SM clocks, Memory clocks and Utilization values for SM, Memory, Encoder and Decoder. The replay tool can also fetch other metrics such as frame buffer memory usage, bar1 memory usage, power/thermal violations and aggregate single/double bit ecc errors. There is an option to select a set of metrics to replay, If any of the requested metric is not maintained or logged as not-supported then it's shown as "-" in the output. The format of data produced by this mode is such that the user is running the device monitoring utility interactively. The command line requires mandatory option "-f" to specify complete path of the log filename, all the other supported options are exclusive and can be used together in any order.

Usage:

1) Specify log file to be replayed

nvidia-smi replay -f <log file name>

Fetches monitoring data from the compressed log file and allows the user to see one line of monitoring data (default metrics with time-stamp) for each monitoring iteration stored in the log file. A new line of monitoring data is replayed every other second irrespective of the actual monitoring frequency maintained at the time of collection. It is displayed till the end of file or until terminated by ^C.

2) Filter metrics to be replayed

nvidia-smi replay -f <path to log file> -s <metric_group>

<metric_group> can be one or more from the following:

p - Power Usage (in Watts) and Temperature (in C)

u - Utilization (SM, Memory, Encoder and Decoder Utilization in %)

c - Proc and Mem Clocks (in MHz)

v - Power Violations (in %) and Thermal Violations (as a boolean flag)

m - Frame Buffer and Bar1 memory usage (in MB)

e - ECC (Number of aggregated single bit, double bit ecc errors) and PCIe Replay errors

t - PCIe Rx and Tx Throughput in MB/s (Maxwell and above)

3) Limit replay to one or more devices

nvidia-smi replay -f <log file> -i <device1,device2, .. , deviceN>

Limits reporting of the metrics to the set of devices selected by comma separated device list. The tool skips any of the devices not maintained in the log file.

4) Restrict the time frame between which data is reported

nvidia-smi replay -f <log file> -b <start time in HH:MM:SS format> -e <end time in HH:MM:SS format>

This option allows the data to be limited between the specified time range. Specifying time as 0 with -b or -e option implies start or end file respectively.

5) Redirect replay information to a log file

nvidia-smi replay -f <log file> -r <output file name>

This option takes log file as an input and extracts the information related to default metrics in the specified output file.

6) Help Information

nvidia-smi replay -h

Displays help information for using the command line.

Process Monitoring

The "nvidia-smi pmon" command-line is used to monitor compute and graphics processes running on one or more GPUs (up to 4 devices) plugged into the system. This tool allows the user to see the statistics for all the running processes on each device at every monitoring cycle. The output is in concise format and easy to interpret in interactive mode. The output data per line is limited by the terminal size. It is supported on Tesla, GRID, Quadro and limited GeForce products for Kepler or newer GPUs under bare metal 64 bits Linux. By default, the monitoring data for each process includes the pid, command name and average utilization values for SM, Memory, Encoder and Decoder since the last monitoring cycle. It can also be configured to report frame buffer memory usage for each process. If there is no process running for the device, then all the metrics are reported as "-" for the device. If any of the metric is not supported on the device or any other error in fetching the metric is also reported as "-" in the output data. The user can also configure monitoring frequency and the number of monitoring iterations for each run. There is also an option to include date and time at each line. All the supported options are exclusive and can be used together in any order.

Usage:

1) Default with no arguments

nvidia-smi pmon

Monitors all the processes running on each device for up to 4 supported devices under natural enumeration (starting with GPU index 0) at a frequency of 1 sec. Runs until terminated with ˆC.

2) Select one or more devices

nvidia-smi pmon -i <device1,device2, .. , deviceN>

Reports statistics for all the processes running on the devices selected by comma separated device list. The tool picks up to 4 supported devices from the list under natural enumeration (starting with GPU index 0).

3) Select metrics to be displayed

nvidia-smi pmon -s <metric_group>

<metric_group> can be one or more from the following:

u - Utilization (SM, Memory, Encoder and Decoder Utilization for the process in %). Reports average utilization since last monitoring cycle.

m - Frame Buffer usage (in MB). Reports instantaneous value for memory usage.

4) Configure monitoring iterations

nvidia-smi pmon -c <number of samples>

Displays data for specified number of samples and exit.

5) Configure monitoring frequency

nvidia-smi pmon -d <time in secs>

Collects and displays data at every specified monitoring interval until terminated with ^C. The monitoring frequency must be between 1 to 10 secs.

6) Display date

nvidia-smi pmon -o D

Prepends monitoring data with date in YYYYMMDD format.

7) Display time

nvidia-smi pmon -o T

Prepends monitoring data with time in HH:MM:SS format.

8) Help Information

nvidia-smi pmon -h

Displays help information for using the command line.

Topology (EXPERIMENTAL)

List topology information about the system's GPUs, how they connect to each other as well as qualified NICs capable of RDMA

Displays a matrix of available GPUs with the following legend:

Legend:

X = Self
SYS = Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UPI)
NODE = Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA node
PHB = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
PXB = Connection traversing multiple PCIe switches (without traversing the PCIe Host Bridge)
PIX = Connection traversing a single PCIe switch
NV# = Connection traversing a bonded set of # NVLinks

vGPU Management

The "nvidia-smi vgpu" command reports on GRID vGPUs executing on supported GPUs and hypervisors (refer to driver release notes for supported platforms). Summary reporting provides basic information about vGPUs currently executing on the system. Additional options provide detailed reporting of vGPU properties, per-vGPU reporting of SM, Memory, Encoder, and Decoder utilization, and per-GPU reporting of supported and creatable vGPUs. Periodic reports can be automatically generated by specifying a configurable loop frequency to any command.

Usage:

1) Default with no arguments

nvidia-smi vgpu

Reports summary of all the vGPUs currently active on each device.

2) Display detailed info on currently active vGPUs

nvidia-smi vgpu -q

Collects and displays information on currently active vGPUs on each device, including driver version, utilization, and other information.

3) Select one or more devices

nvidia-smi vgpu -i <device1,device2, .. , deviceN>

Reports summary for all the vGPUs currently active on the devices selected by comma-separated device list.

4) Display supported vGPUs

nvidia-smi vgpu -s

Displays vGPU types supported on each device. Use the -v / --verbose option to show detailed info on each vGPU type.

5) Display creatable vGPUs

nvidia-smi vgpu -c

Displays vGPU types creatable on each device. This varies dynamically, depending on the vGPUs already active on the device. Use the -v / --verbose option to show detailed info on each vGPU type.

6) Report utilization for currently active vGPUs.

nvidia-smi vgpu -u

Reports average utilization (SM, Memory, Encoder and Decoder) for each active vGPU since last monitoring cycle. The default cycle time is 1 second, and the command runs until terminated with ˆC. If a device has no active vGPUs, its metrics are reported as "-".

7) Configure loop frequency

nvidia-smi vgpu [-s -c -q -u] -l <time in secs>

Collects and displays data at a specified loop interval until terminated with ˆC. The loop frequency must be between 1 and 10 secs. When no time is specified, the loop frequency defaults to 5 secs.

8) Help Information

nvidia-smi vgpu -h

Displays help information for using the command line.

UNIT ATTRIBUTES

The following list describes all possible data returned by the **−q -u** unit query option. Unless otherwise noted all numerical results are base 10 and unitless.

Timestamp
The current system timestamp at the time nvidia−smi was invoked. Format is "Day−of−week Month Day HH:MM:SS Year".

Driver Version
The version of the installed NVIDIA display driver. Format is "Major−Number.Minor−Number".

HIC Info
Information about any Host Interface Cards (HIC) that are installed in the system.

Firmware Version
The version of the firmware running on the HIC.

Attached Units
The number of attached Units in the system.

Product Name
The official product name of the unit. This is an alphanumeric value. For all S-class products.

Product Id
The product identifier for the unit. This is an alphanumeric value of the form "part1-part2-part3". For all S-class products.

Product Serial
The immutable globally unique identifier for the unit. This is an alphanumeric value. For all S-class products.

Firmware Version
The version of the firmware running on the unit. Format is "Major−Number.Minor−Number". For all S-class products.

LED State
The LED indicator is used to flag systems with potential problems. An LED color of AMBER indicates an issue. For all S-class products.

Color The color of the LED indicator. Either "GREEN" or "AMBER".

Cause The reason for the current LED color. The cause may be listed as any combination of "Unknown", "Set to AMBER by host system", "Thermal sensor failure", "Fan failure" and "Temperature exceeds critical limit".

Temperature
Temperature readings for important components of the Unit. All readings are in degrees C. Not all readings may be available. For all S-class products.

Intake Air temperature at the unit intake.

Exhaust Air temperature at the unit exhaust point.

Board Air temperature across the unit board.

PSU
Readings for the unit power supply. For all S-class products.

State Operating state of the PSU. The power supply state can be any of the following: "Normal", "Abnormal", "High voltage", "Fan failure", "Heatsink temperature", "Current limit", "Voltage below UV alarm threshold", "Low−voltage", "I2C remote off command", "MOD_DISABLE input" or "Short pin transition".

Voltage PSU voltage setting, in volts.

Current PSU current draw, in amps.

Fan Info
Fan readings for the unit. A reading is provided for each fan, of which there can be many. For all S-class products.

State The state of the fan, either "NORMAL" or "FAILED".

Speed For a healthy fan, the fan's speed in RPM.

Attached GPUs
A list of PCI bus ids that correspond to each of the GPUs attached to the unit. The bus ids have the form "domain:bus:device.function", in hex. For all S-class products.

NOTES

On Linux, NVIDIA device files may be modified by nvidia-smi if run as root. Please see the relevant section of the driver README file.

The **−a** and **−g** arguments are now deprecated in favor of **−q** and **−i**, respectively. However, the old arguments still work for this release.

EXAMPLES

nvidia−smi -q
Query attributes for all GPUs once, and display in plain text to stdout.

nvidia−smi −−format=csv,noheader −−query−gpu=uuid,persistence_mode
Query UUID and persistence mode of all GPUs in the system.

nvidia−smi -q -d ECC,POWER -i 0 -l 10 -f out.log
Query ECC errors and power consumption for GPU 0 at a frequency of 10 seconds, indefinitely, and record to the file out.log.

"nvidia−smi -c 1 -i GPU−b2f5f1b745e3d23d−65a3a26d-097db358−7303e0b6−149642ff3d219f8587cde3a8"
Set the compute mode to "PROHIBITED" for GPU with UUID "GPU−b2f5f1b745e3d23d−65a3a26d-097db358−7303e0b6−149642ff3d219f8587cde3a8".

nvidia−smi -q -u -x --dtd
Query attributes for all Units once, and display in XML format with embedded DTD to stdout.

nvidia−smi --dtd -u -f nvsmi_unit.dtd
Write the Unit DTD to nvsmi_unit.dtd.

nvidia−smi -q -d SUPPORTED_CLOCKS
Display supported clocks of all GPUs.

nvidia−smi -i 0 --applications-clocks 2500,745
Set applications clocks to 2500 MHz memory, and 745 MHz graphics.

CHANGE LOG

=== Known Issues ===

* On Linux GPU Reset can’t be triggered when there is pending GOM change.

* On Linux GPU Reset may not successfully change pending ECC mode. A full reboot may be required to enable the mode change.

=== Changes between nvidia-smi v346 Update and v352 ===

* Added topo support to display affinities per GPU

* Added topo support to display neighboring GPUs for a given level

* Added topo support to show pathway between two given GPUs

* Added "nvidia-smi pmon" command-line for process monitoring in scrolling format

* Added "--debug" option to produce an encrypted debug log for use in submission of bugs back to NVIDIA

* Fixed reporting of Used/Free memory under Windows WDDM mode

* The accounting stats is updated to include both running and terminated processes. The execution time of running process is reported as 0 and updated to actual value when the process is terminated.

=== Changes between nvidia-smi v340 Update and v346 ===

* Added reporting of PCIe replay counters

* Added support for reporting Graphics processes via nvidia-smi

* Added reporting of PCIe utilization

* Added dmon command-line for device monitoring in scrolling format

* Added daemon command-line to run in background and monitor devices as a daemon process. Generates dated log files at /var/log/nvstats/

* Added replay command-line to replay/extract the stat files generated by the daemon tool

=== Changes between nvidia-smi v331 Update and v340 ===

* Added reporting of temperature threshold information.

* Added reporting of brand information (e.g. Tesla, Quadro, etc.)

* Added support for K40d and K80.

* Added reporting of max, min and avg for samples (power, utilization, clock changes). Example commandline: nvidia-smi -q -d power,utilization, clock

* Added nvidia-smi stats interface to collect statistics such as power, utilization, clock changes, xid events and perf capping counters with a notion of time attached to each sample. Example commandline: nvidia-smi stats

* Added support for collectively reporting metrics on more than one GPU. Used with comma separated with "-i" option. Example: nvidia-smi -i 0,1,2

* Added support for displaying the GPU encoder and decoder utilizations

* Added nvidia-smi topo interface to display the GPUDirect communication matrix (EXPERIMENTAL)

* Added support for displayed the GPU board ID and whether or not it is a multiGPU board

* Removed user-defined throttle reason from XML output

=== Changes between nvidia-smi v5.319 Update and v331 ===

* Added reporting of minor number.

* Added reporting BAR1 memory size.

* Added reporting of bridge chip firmware.

=== Changes between nvidia-smi v4.319 Production and v4.319 Update ===

* Added new --applications-clocks-permission switch to change permission requirements for setting and resetting applications clocks.

=== Changes between nvidia-smi v4.304 and v4.319 Production ===

* Added reporting of Display Active state and updated documentation to clarify how it differs from Display Mode and Display Active state

* For consistency on multi-GPU boards nvidia-smi -L always displays UUID instead of serial number

* Added machine readable selective reporting. See SELECTIVE QUERY OPTIONS section of nvidia-smi -h

* Added queries for page retirement information. See --help-query-retired-pages and -d PAGE_RETIREMENT

* Renamed Clock Throttle Reason User Defined Clocks to Applications Clocks Setting

* On error, return codes have distinct non zero values for each error class. See RETURN VALUE section

* nvidia-smi -i can now query information from healthy GPU when there is a problem with other GPU in the system

* All messages that point to a problem with a GPU print pci bus id of a GPU at fault

* New flag --loop-ms for querying information at higher rates than once a second (can have negative impact on system performance)

* Added queries for accounting procsses. See --help-query-accounted-apps and -d ACCOUNTING

* Added the enforced power limit to the query output

=== Changes between nvidia-smi v4.304 RC and v4.304 Production ===

* Added reporting of GPU Operation Mode (GOM)

* Added new --gom switch to set GPU Operation Mode

=== Changes between nvidia-smi v3.295 and v4.304 RC ===

* Reformatted non-verbose output due to user feedback. Removed pending information from table.

* Print out helpful message if initialization fails due to kernel module not receiving interrupts

* Better error handling when NVML shared library is not present in the system

* Added new --applications-clocks switch

* Added new filter to --display switch. Run with -d SUPPORTED_CLOCKS to list possible clocks on a GPU

* When reporting free memory, calculate it from the rounded total and used memory so that values add up

* Added reporting of power management limit constraints and default limit

* Added new --power-limit switch

* Added reporting of texture memory ECC errors

* Added reporting of Clock Throttle Reasons

=== Changes between nvidia-smi v2.285 and v3.295 ===

* Clearer error reporting for running commands (like changing compute mode)

* When running commands on multiple GPUs at once N/A errors are treated as warnings.

* nvidia-smi -i now also supports UUID

* UUID format changed to match UUID standard and will report a different value.

=== Changes between nvidia-smi v2.0 and v2.285 ===

* Report VBIOS version.

* Added -d/--display flag to filter parts of data

* Added reporting of PCI Sub System ID

* Updated docs to indicate we support M2075 and C2075

* Report HIC HWBC firmware version with -u switch

* Report max(P0) clocks next to current clocks

* Added --dtd flag to print the device or unit DTD

* Added message when NVIDIA driver is not running

* Added reporting of PCIe link generation (max and current), and link width (max and current).

* Getting pending driver model works on non-admin

* Added support for running nvidia-smi on Windows Guest accounts

* Running nvidia-smi without -q command will output non verbose version of -q instead of help

* Fixed parsing of -l/--loop= argument (default value, 0, to big value)

* Changed format of pciBusId (to XXXX:XX:XX.X - this change was visible in 280)

* Parsing of busId for -i command is less restrictive. You can pass 0:2:0.0 or 0000:02:00 and other variations

* Changed versioning scheme to also include "driver version"

* XML format always conforms to DTD, even when error conditions occur

* Added support for single and double bit ECC events and XID errors (enabled by default with -l flag disabled for -x flag)

* Added device reset -r --gpu-reset flags

* Added listing of compute running processes

* Renamed power state to performance state. Deprecated support exists in XML output only.

* Updated DTD version number to 2.0 to match the updated XML output

SEE ALSO

On Linux, the driver README is installed as /usr/share/doc/NVIDIA_GLX-1.0/README.txt

AUTHOR

NVIDIA Corporation

COPYRIGHT

Copyright 2011-2017 NVIDIA Corporation.