Dynamic Page Retirement

Release r560

NVIDIA Corporation

Nov 13, 2024

Contents

Dynamic Page Retirement

This document describes retirement of framebuffer pages that contain bad memory cells.

Chapter 1. Overview

The NVIDIA® driver supports "retiring" framebuffer pages that contain bad memory cells. This is called "dynamic page retirement" and is done automatically for cells that are degrading in quality. This feature can improve the longevity of an otherwise good board and and is thus an important resiliency feature on supported products, especially in HPC and enterprise environments.

Chapter 2. Implementation

The marking of a page for exclusion is called "retiring", while the actual act of excluding that page from subsequent memory allocations is called "blacklisting". The NVIDIA driver will retire a page once it has experienced a single Double Bit ECC Error (DBE), or 2 Single Bit ECC Errors (SBE) on the same address. These addresses are stored in the InfoROM. When each GPU is attached and initialized the driver will retrieve these addresses from the InfoROM, then have the framebuffer manager set these pages aside, such that they cannot be used by the driver or user applications.

Note: Retiring of pages may only occur when ECC is enabled. However, once a page has been retired it will always be blacklisted by the driver, even if ECC is later disabled.

Ideally, the NVIDIA driver will catch weakening cells at the 2 SBE point and retire the page, before the cell degrades to the point of a DBE and disrupts an application.

2.1. Use Case 1: DBE Detected

1. The NVIDIA driver detects a DBE and reports that a DBE occurred.
2. Applications will receive a DBE event notification for graceful exit, and no further context will be created on the GPU until the DBE is mapped out.
3. The NVIDIA driver logs the DBE count and address in the InfoROM.

   Page retirement occurs and the `nvidia-smi` **Retired Pages 'Double Bit ECC'** field is incremented.

   The `nvidia-smi` 'Pending Page Blacklist' status becomes **'YES'**.
4. The NVIDIA driver logs, in a separate list, that the page containing the DBE is to be retired.
5. Upon the next reattachment of the GPU, the page is mapped out of usage.

   Page blacklisting occurs and `nvidia-smi` **'Pending Page Blacklist'** status becomes **'NO'**.

Note: The DBE addresses and counts are preserved across driver reloads.

2.2. Use Case 2: Two SBEs Detected at the Same Location

1. The NVIDIA driver detects an SBE and reports that an SBE occurred.
2. The NVIDIA driver logs the SBE count and address in the InfoROM.
3. If the SBE occurs more than once in a particular address, the driver logs, in a separate list, that the page containing that address is to be retired.

   Page retirement occurs and the `nvidia-smi` **Retired Pages 'Single Bit ECC'** field is incremented.

   The `nvidia-smi` **'Pending Page Blacklist'** status becomes **'YES'**.
4. Upon the next reattachment of the GPU, the page is mapped out of usage.

   Page blacklist occurs and the `nvidia-smi` **'Pending Page Blacklist'** status becomes **'NO'**.

Note: Unlike the DBE case, applications continue to run uninterrupted.

Note: The SBE addresses and counts are preserved across driver reloads.

Chapter 3. Blacklisting and ECC Error Recovery

Pages that have been previously retired are blacklisted for all future allocations of the framebuffer, provided that the target GPU has been properly reattached and initialized. This chapter presents a procedure for ensuring that retired pages are blacklisted and all GPUs have recovered from the ECC error.

Note: This procedure requires the termination of all clients on the target GPU. It is not possible to blacklist a new page while clients remain active.

Blacklisting: Procedure Overview:

1. Verify that there are pending retired pages.
2. Determine GPUs that are related and must be reattached together.
3. Stop applications and verify there are none left running.
4. Reinitialize the GPU, or reboot the system.
5. Verify that blacklisting has occurred.
6. Restart applications.

3.1. Verifying Retired Pages are Pending

When pages are retired but have not yet been blacklisted, the retired pages are marked as pending for that GPU. This can be seen through `nvidia-smi`:

```
$ nvidia-smi -i <target gpu> -q -d  PAGE_RETIREMENT
  ...
  Retired pages
  Single Bit ECC              : 2
  Double Bit ECC              : 0
  Pending Page Blacklist    : Yes
  ...
```

If Pending Page Blacklist shows “No”, then all retired pages have already been blacklisted.

If Pending Page Blacklist shows “Yes”, then at least one of the retired pages that are counted are not yet blacklisted. Note that the exact count of pending pages is not shown.

Note: The retired pages count increments immediately when a page is retired and not on the next driver reload when the page is blacklisted.

3.2. Stopping GPU Clients

Before the the NVIDIA driver can reattach the GPUs, clients that are using those GPUs must first be stopped and the GPU must be unused.

All applications that are using the GPUs should first be stopped. Use `nvidia-smi` to list processes that are actively using the GPUs. In the example below, a tensorflow python program is using both GPUs 0 and 1. Both will need to be stopped.

```
$ nvidia-smi
...
+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      8962      C   python                                     15465MiB |
|    1      8963      C   python                                     15467MiB |
+-----------------------------------------------------------------------------+
```

Once all applications are stopped, `nvidia-smi` should show no processes found:

```
$ nvidia-smi
...
+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

On Linux systems, additional software infrastructure can hold the GPU open and prevent the GPU from being detached by the driver. These include the `nvidia-persistenced`, and version 1 of nvidia-docker. Nvidia-docker version 2 does not need to be stopped.

A list of open proesses using the driver can be verified on Linux with the `lsof` command:

```
$ sudo lsof /dev/nvidia*
COMMAND   PID                USER   FD   TYPE  DEVICE NODE NAME
nvidia-pe 941 nvidia-persistenced    2u   CHR 195,255  453 /dev/nvidiactl
nvidia-pe 941 nvidia-persistenced    3u   CHR   195,0  454 /dev/nvidia0
nvidia-pe 941 nvidia-persistenced    4u   CHR 195,254  607 /dev/nvidia-modeset
nvidia-pe 941 nvidia-persistenced    5u   CHR   195,1  584 /dev/nvidia1
nvidia-pe 941 nvidia-persistenced    6u   CHR 195,254  607 /dev/nvidia-modeset
```

Once all clients of the GPU are stopped, lsof should return no entries:

```
$ sudo service nvidia-persistenced stop
$ sudo lsof /dev/nvidia*
$
```

3.3. Reattaching the GPU

Reattaching the GPU, to blacklist pending retired pages, can be done in several ways. In order of cost, from low to high:

- Re-attach the GPUs (persistence mode disabled only)
- Reset the GPUs
- Reload the kernel module (nvidia.ko)
- Reboot the machine (or VM)

Reattaching the GPU is the least invasive solution. The detachment process occurs automatically a few seconds after the last client terminates on the GPU, as long as persistence mode is not enabled. The next client that targets the GPU will trigger the driver to reattach and blacklist all marked pages.

If persistence mode is enabled, the preferred solution is to reset the GPU using `nvidia-smi`. To reset an individual GPU:

```
$ nvidia-smi -i < target GPU> -r
```

Or to reset all GPUs together:

```
$ nvidia-smi -r
```

These operations reattach the GPU as a step in the larger process of resetting all GPU SW and HW state.

Reloading the NVIDIA kernel module triggers reattachment of all GPUs on the machine, and thus requires the termination of all clients on all GPUs.

Finally, rebooting the machine will effectively reattach the GPUs as the driver is reloaded and reinitialized during reboot. While rebooting isn't required and is highly invasive, it might simplify the recovery action in some operating environments.

Chapter 4. Availability

Dynamic page retirement is supported on the following products and environments:

- Drivers: R319 and newer
- OSes: All standard driver-supported Linux and Windows TCC platforms
- GPUs:
  - K20 and newer Tesla products, including the Tesla V100 and T4
  - Quadro GV100 and newer products
  - Quadro Virtual Data Center Workstation (Quadro vDWS) and NVIDIA vComputeServer (starting with NVIDIA Virtual GPU Software v9.0)
  - No GeForce products are currently supported

Chapter 5. Visibility

Three main mechanisms provide visibility into page retirement: XID errors in system logs, the NVML API and the nvidia-smi command line tool.

5.1. XIDs

XID errors are driver errors that are logged to the system error log. Please see the XID Whitepaper for general info on XIDs. There are three main XIDs related to dynamic page retirement:

- XID 48: A DBE has occurred.
- XID 63: A page has successfully been retired.
- XID 64: A page has failed retirement due to an error.

In the system log these XIDs show up in the following forms:

- XID 48: "XID 48 An uncorrectable double bit error (DBE) has been detected on GPU (<id>)"
- XID 63: "XID 63 Dynamic Page Retirement: New retired page, reload the driver to activate. (<address>)"
- XID 64: "XID 64 Dynamic Page Retirement: Fatal error, unable to retire page (<address>)"

5.2. NVML

The NVIDIA Management Library (NVML) is a public C-based library for GPU monitoring and management. It includes APIs that report the status and count of retired pages. Refer to the NVML API doc for general info on the library.

The set of currently retired pages, and their addresses, can be retrieved using:

```
nvmlReturn_t nvmlDeviceGetRetiredPages (nvmlDevice_t device, nvmlPageRetirementCause_
↪t cause, unsigned int* pageCount, unsigned long long* addresses)
```

Driver versions 410.72 or later provide a newer API that also returns page retirement timestamp:

```
nvmlReturn_t nvmlDeviceGetRetiredPages_v2 (nvmlDevice_t device,
↪nvmlPageRetirementCause_t cause, unsigned int* pageCount, unsigned long long*
↪addresses, unsigned long long* timestamps)
```

For both APIs, the `nvmlPageRetirementCause_t` passed is one of:

- `NVML_PAGE_RETIREMENT_CAUSE_MULTIPLE_SINGLE_BIT_ECC_ERRORS`
- `NVML_PAGE_RETIREMENT_CAUSE_DOUBLE_BIT_ECC_ERROR`

The current state of the driver (whether any pages are pending retirement) can be retrieved using:

```
nvmlReturn_t nvmlDeviceGetRetiredPagesPendingStatus (nvmlDevice_t device,
↪nvmlEnableState_t* isPending)
```

5.3. nvidia-smi

`nvidia-smi` is a public command line interface for GPU monitoring and management. It implements most of the NVML APIs and supports reporting the status and count of retired pages. Please see the `nvidia-smi` man page for general info on the tool.

The `nvidia-smi` tool provides:

- the ability to list the number of retired pages–sorted by cause of retirement–and indicate whether any pages are pending retirement, and
- the ability to list all retired page addresses.

To view the number of retired pages and the page retirement state of the driver in human readable form:

```
$ nvidia-smi -i <target gpu> -q -d  PAGE_RETIREMENT
     ...
     Retired pages
            Single Bit ECC    : 2
            Double Bit ECC    : 0
            Pending Page Blacklist   : No
     ...
```

The `nvidia-smi` **"Pending Page Blacklist"** field indicates whether a page has been recently retired and will be blacklisted on the next system reboot or driver load.

Note: The retired page count increments immediately when a page is retired and not on the next driver reload when the page is blacklisted.

If pages have been retired, the affected addresses can be viewed through `nvidia-smi`'s scriptable outputs, in either XML format:

```
$ nvidia-smi -i <target gpu> -q -x
...
 <retired_pages>
     <multiple_single_bit_retirement>
     <retired_count>2</retired_count>
          <retired_page_addresses>
               <retired_page_addresse>0xABC123</retired_page_addresse>
               <retired_page_addresse>0xDEF456</retired_page_addresse>
          </retired_page_addresses>
     </multiple_single_bit_retirement>
     <double_bit_retirement>
```

(continues on next page)

(continued from previous page)

```
        <retired_count>0</retired_count>
        <retired_page_addresses></retired_page_addresses>
    </double_bit_retirement>
    <pending_retirement>No</pending_retirement>
 </retired_pages>
...
```

or CSV format:

```
$ nvidia-smi -i <target gpu> --query-retired-pages=
gpu_uuid,retired_pages.address,retired_pages.cause --format=csv
...
gpu_uuid, retired_pages.address, retired_pages.cause
GPU-d73c8888-9482-7d65-c95c-4b58c7d9eb4c, 0xABC123, Double Bit ECC
GPU-d73c8888-9482-7d65-c95c-4b58c7d9eb4c, 0xDEF456, Double Bit ECC
GPU-d73c8888-9482-7d65-c95c-4b58c7d9eb4c, 0x123ABC, Single Bit ECC
```

Chapter 6. Caveats

There exists a race condition between logging errors to the InfoROM and ending a CUDA™ job while in persistence mode. This race condition is most often hit when shutting down in response to a DBE. The effect of this condition is that a page may fail to retire in certain corner cases.

Temporarily exiting persistence mode before rebooting the system will forcibly flush any pending writes to the InfoROM. If XID 48 is seen and XID 63 is not seen, it is recommended to exit persistence mode via the command:

```
% nvidia-smi -i <target GPU> -pm 0
```

At this point, the XID 63 should be seen and the NVML query can be used to verify the page was written to the InfoROM.

There are no current plans to fix the race condition in persistence mode, as persistence mode is replaced by the persistence daemon. The persistence daemon is not susceptible to this race condition. To let you know that your platform is susceptible to this race condition, the kernel driver will print out a warning in the dmesg log files to indicate a persistence daemon is not being used.

Chapter 7. FAQ

7.1. RMA Eligibility

How many pages can be mapped out before the GPU should be returned for repair?

The Tesla board will continue to retire pages up until the page retirement table is full, at 64 dynamically retired memory pages. However, a board that generates 60 or more retired pages is eligible for an RMA. A Tesla card with 64 pages retired will fail the NVIDIA Field Diagnostic tool.

Additionally, if a board is found to exhibit 15 or more retired pages and continues to retire memory pages at a rate of one or more newly retired pages per week, it can be evaluated for an RMA before the 60-page RMA threshold has been reached. Please track the page retirement rate and provide that information with the returned board..

7.2. Memory Impact

Is available memory reduced by retired pages?

Each retired page decreases the total memory available to applications. However, the total maximum size of memory retired in this way is only on the order of 4 MiB. This is insignificant relative to other factors, such as natural fluctuations in memory allocated internally by the NVIDIA driver during normal operation.

What is the size of each retired page?

64KiB

How many pages can be retired?

A combined total of 64 pages can be mapped out, or retired. This can be any combination of DBE and SBE pages.

How many addresses can be stored in the InfoROM?

The InfoROM can store at least 192 different addresses. Different GPU models and InfoROM formats may extend this beyond 192 addresses up to a maximum of 600.

7.3. Configuration

Is ECC enabled on my GPU?

nvidia-smi can be used to show if the GPU currently has ECC enabled, and what mode is pending to be activated following a reboot.

```
$ nvidia-smi -q
...
   Ecc Mode
     Current                      : Disabled
     Pending                      : Enabled
```

Can page retirement be disabled?

No, page retirement is an important reliability feature and cannot be disabled for either SBEs or DBEs. Any pages already marked as retired will continue to be excluded in all future allocations. Note though that if ECC is disabled no new memory errors will be detected and thus no new pages will be blacklisted for future retirement.

ECC can be disabled through nvidia-smi:

```
$ nvidia-smi -e 0
Disabled ECC support for GPU 00000000:06:00.0.
All done.
Reboot required.
```

Or re-enabled, similarly:

```
sudo nvidia-smi -e 1
Enabled ECC support for GPU 00000000:06:00.0.
All done.
Reboot required.
```

Is the SBE recurrence threshold for triggering retirement be configurable?

No

Can I disable DMA protected range (DPR) for SBEs or DBEs?

No, NVIDIA does not support disabling DPR, either entirely or for just SBEs or DBEs.

7.4. App Behavior

Is application behavior affected?

No, applications behave the same. Since pages are retired only after the driver has been restarted the act of retiring a page occurs outside the lifetime of any GPU process or application. An application running on a GPU with pages scheduled for retirement (blacklisted) will continue to see those pages in its memory space, though any page retired due to a double bit error (DBE) will necessarily cause an application to terminate. This is true even without page retirement.

7.5. App Performance

Is application performance affected?

No, application performance is unaffected by either the retirement of pages or their subsequent blacklisting. Retirement is the only act taken during application execution, while the actual blacklisting event happens only after the application has terminated. As noted in the first FAQ question above, the memory impact of retired pages is also negligible.

Is memory fragmentation due to page retirement expected to impact app performance?

Fragmentation is not expected to affect performance.

7.6. Driver Behavior

Must multiple SBEs be located at the same address to trigger retirement?

Yes, multiple SBEs must be located at the exact same location (address). Multiple SBEs at different locations within the same page will not trigger page retirement.

Are "stuck" bits rewritten by the driver, or corrected on each read?

On Kepler-class GPUs and later, the driver rewrites the data to avoid stuck bits.

Are all SBE and DBE addresses tracked indefinitely?

SBE and DBE addresses are tracked indefinitely, up to the maximum number of addresses that can be stored (See *Memory Impact*). Additional addresses beyond the maximum are dropped.

Chapter 8. Notices

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

8.1. Trademarks

Copyright