Improving *page_owner* for profiling and monitoring memory usage per allocation stack trace

Mauricio Faria de Oliveira

mfo (at) igalia.com

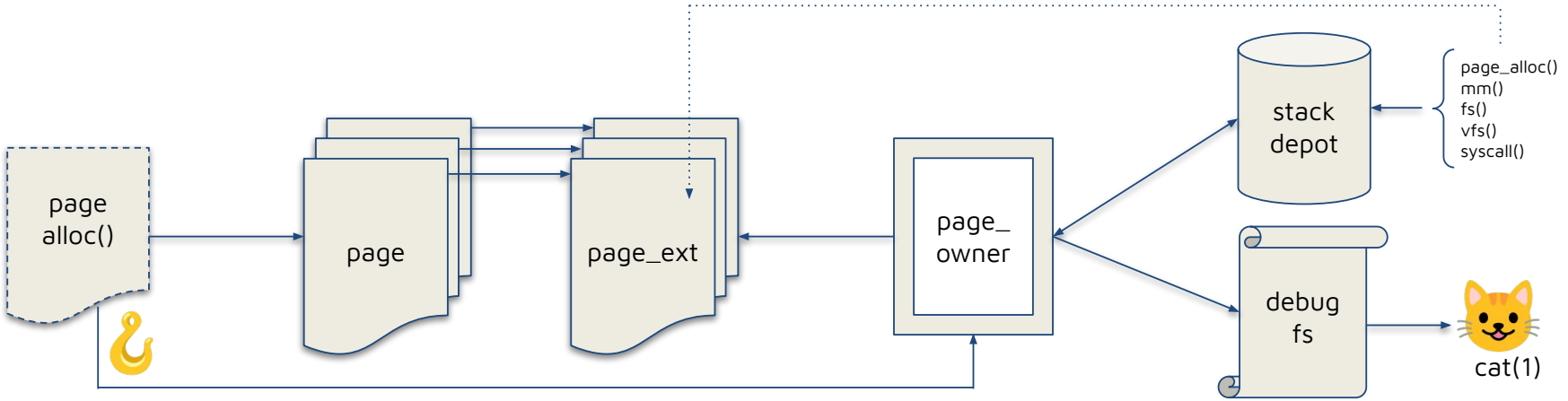
Linux Plumbers Conference 2025

Summary

- **What is *page_owner?***
  - Usage
  - Examples
- **Profiling and monitoring with *page_owner***
  - Definitions
  - Issues
- **Improving *page_owner***
  - Output format
  - Swap space
- **Discussion**

What is *page_owner?*

- It is a debug feature for memory allocation (or *usage*).
- It stores the stack trace (and more) of memory allocation for *every page*.
- It combines page alloc hooks, page_ext, stack_depot, and debugfs.



What is *page_owner?*

- It provides a system-wide snapshot of memory allocations
  - Per page
  - Per stack trace
- It can answer the questions:
  - What stack trace allocated this page? (its "owner")
  - How many pages are allocated by this stack trace?
- Use case example:
  - Profile system-wide memory usage over time, with swapping.
  - Monitor the top memory consumers.

Usage

1. Build with **CONFIG_PAGE_OWNER=y** (mm/Kconfig.debug)
2. Boot with **page_owner=on** (kernel cmdline)
3. cat /sys/kernel/debug/**page_owner** *# (stacks per page)*
4. cat /sys/kernel/debug/**page_owner_stacks/show_stacks** *# (pages per stack)*

Examples (1/2)

("What stack trace allocated this page?")

```
# dd if=/dev/zero of=/tmpfs/file status=none bs=1M count=256 # (Write a 256 MiB file to tmpfs)

# cat /sys/kernel/debug/page_owner # (stacks per page)
...

Page allocated via order 0, mask 0x...(GFP_HIGHUSER_MOVABLE|__GFP_COMP), pid 228, tgid 228 (dd), ts 8716052197 ns
PFN 0xccf9 type Movable Block 102 type Movable Flags 0x...(referenced|uptodate|dirty|lru|swapbacked|node=0|zone=1)
 get_page_from_freelist+0x13b9/0x1530
 ...
 shmem_alloc_folio+0x97/0xb0
 ...
 shmem_file_write_iter+0x81/0x90
 vfs_write+0x28d/0x450
 ...
 entry_SYSCALL_64_after_hwframe+0x77/0x7f
Charged to memcg session-1.scope

# cat /sys/kernel/debug/page_owner | grep -c '^Page allocated via order 0, .* (dd)' # (pages allocated by dd)
65553

# echo $((65553*4096/1024**2)) # (number of 4 KiB pages in MiB; approximately 256 MiB)
256
```


Examples (2/2)

("How many pages are allocated by this stack trace?")

```
# dd if=/dev/zero of=/tmpfs/file status=none bs=1M count=256 # (Write a 256 MiB file to tmpfs)

# cat /sys/kernel/debug/page_owner_stacks/show_stacks # (pages per stack)
...

 get_page_from_freelist+0x13b9/0x1530
 ...
 shmem_alloc_folio+0x97/0xb0
 ...
 shmem_file_write_iter+0x81/0x90
 vfs_write+0x28d/0x450
 ...
 entry_SYSCALL_64_after_hwframe+0x77/0x7f
nr_base_pages: 65649
...

# echo $((65649*4096/1024**2)) # (number of 4 KiB pages in MiB; approximately 256 MiB)
256
```

Profiling and monitoring with *page_owner*

- **Sampling**:
  - System-wide snapshots of memory allocations (pages) per stack trace.
- **Profiling**:
  - How many pages are used by each memory allocation stack trace?
  - Examples: analyze memory usage; identify top memory consumers.
- **Monitoring**:
  - How many pages are used by some memory allocation stack trace(s)?
  - Examples: testing code changes; verify memory-constrained cases.

- OK. Now, the implementation...

Profiling and monitoring with *page_owner*

- **Sampling**:
  - cat /sys/kernel/debug/page_owner_stacks/**show_stacks**
- **Profiling**. For every sample:
  - Calculate an unique identifier (key, hash) for every stack trace.
  - Store the number of pages used by each stack trace, and total.
  - Calculate percentages for each stack trace.
- **Monitoring**. For every sample:
  - Calculate an unique identifier (key, hash) for every stack trace.
  - Filter some stack trace(s) among those.
  - Check number of pages used.

Profiling and monitoring with *page_owner*

- **Sampling**:
  - cat /sys/kernel/debug/page_owner_stacks/**show_stacks**
- **Profiling**. For **every** sample:
  - Calculate an unique identifier (key, hash) for every stack trace. **(expensive)**
  - Store the number of pages used by each stack trace, and total. **(cheap)**
  - Calculate percentages for each stack trace. **(cheap)**
- **Monitoring**. For **every** sample:
  - Calculate an unique identifier (key, hash) for every stack trace. **(expensive)**
  - Filter some stack trace(s) among those. **(cheap)**
  - Check number of pages used. **(cheap)**

Improving *page_owner (1/2)*

- **Output format**: /sys/kernel/debug/page_owner_stacks/**show_stacks**:
  - Stack trace
  - Number of base pages
- **Issue**:
  - Profiling and monitoring have to **calculate** an unique identifier (key, hash) for **every** stack trace (hundreds/thousands?) on **every** sample (5 secs; less?).
  - This processes **repeats** (wastes) the **same** calculation/result on every sample.
  - (It's **worse** for monitoring: must calculate **all** unique identifiers, but use a **few**.)
- **Idea**: unique identifiers for stack traces provided (not calculated) by the **kernel**.
  - If that is cheap, it all becomes cheap.

Improving *page_owner (1/2)*

- **Output format**: /sys/kernel/debug/page_owner_stacks/**show_handles**:
  - Stack trace's handle number
  - Number of base pages
- **Implementation**:
  - **stack_depot** already has unique identifiers per stack trace (*handle number*)
  - Different debugfs file (*show_**stacks** -> show_**handles***)

```
# cat /sys/kernel/debug/page_owner_stacks/show_handles \
   | grep -B1 '^nr_base_pages: 65649$' # (dd example)
handle: 23199752
nr_base_pages: 65649
```

- **Question:** *Wait! Now, where's the stack trace?!*

Improving *page_owner (1/2)*

- **Output format**: /sys/kernel/debug/page_owner_stacks/**show_stacks_handles**:
  - Stack trace's handle number
  - Stack trace
- **Implementation**:
  - Another debugfs file (show_stacks_handles).
  -

```
# /sys/kernel/debug/page_owner_stacks/show_stacks_handles \
    | sed -n '/^$/h; /^ /H; /^handle: 23199752$/ {H;x;p}'

 get_page_from_freelist+0x13b9/0x1530
 ...
 entry_SYSCALL_64_after_hwframe+0x77/0x7f
handle: 23199752
```

- **Result**: efficient sampling (show_handles); lazy-resolve handles to stacks (at end), instead of repeatedly calculating identifiers.

Example

Output format: 3 files, for **stacks**, **handles**, and **number of pages**

```
# dd if=/dev/zero of=/tmpfs/file status=none bs=1M count=256 # (Write a 256 MiB file to tmpfs)

# cat /sys/kernel/debug/page_owner_stacks/show_stacks # (pages per stack)
...
 get_page_from_freelist+0x13b9/0x1530
 ...
 entry_SYSCALL_64_after_hwframe+0x77/0x7f
nr_base_pages: 65649
...

# cat /sys/kernel/debug/page_owner_stacks/show_handles | grep -B1 '^nr_base_pages: 65649$'
handle: 23199752
nr_base_pages: 65649

# /sys/kernel/debug/page_owner_stacks/show_stacks_handles | sed -n '/^$/h; /^ /H; /^handle: 23199752$/ {H;x;p}'

 get_page_from_freelist+0x13b9/0x1530
 ...
 entry_SYSCALL_64_after_hwframe+0x77/0x7f
handle: 23199752
```

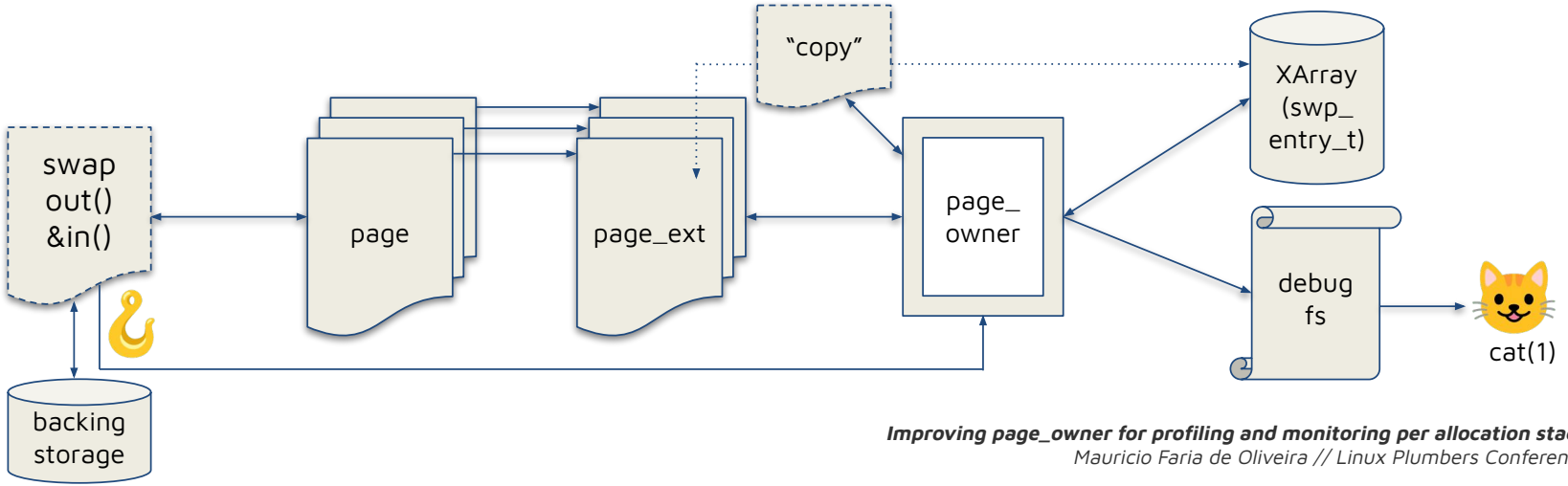
(Note: merged for **v6.19**.)

Improving *page_owner (2/2)*

- **Swap space:**
  - *page_owner* does not support pages in swap space, *currently*.
  - Profiling and monitoring are **incomplete** and **broken** with swapping.
- **Issue**:
  - Swap-out: the page is freed, and another memory allocation takes it.
    - This **overrides** the stack trace of the initial memory allocation of a page.
  - Swap-in: another page is allocated.
    - This is a **different** stack trace than the initial memory allocation's.
- **Idea**: **store/load** the stack trace of the **initial** memory allocation at **swap-out/in**.

Improving *page_owner (2/2)*

- **Swap hooks** for ARM64's Memory Tagging Extension (CONFIG_ARM64_MTE).
- **Implementation** (based on it):
  - Swap-out hook: **store a copy** of the allocation stack trace (and more).
  - Swap-in hook: **load the copy back** (overrides the allocation during swap-in).



Usage

0. [PATCH **RFC** 0/9] mm/page_owner: add support for pages in swap space

https://lore.kernel.org/linux-mm/20251205231721.104505-1-mfo@igalia.com/

1. Build with **CONFIG_SWAP_PAGE_OWNER=y** (mm/Kconfig.debug)
2. Boot with **page_owner=on*,swap*** (kernel cmdline)
3. cat /sys/kernel/debug/**page_owner** | grep '^Page .* **(swapped)**$' # (reporting)
4. cat /sys/kernel/debug/**swap_page_owner** # (new file: pages in swap space)

Example 1/2

New file: **swap_page_owner** (pages in swap space)

```
Allocate 256 MiB with dd, push it (swap out). The new file shows ~256 MiB in swap, allocated by dd.

# dd if=/dev/zero of=/tmpfs/file status=none bs=1M count=256

# ln -sf $(which dd) ./DD
# ./DD if=/dev/zero of=/tmpfs-noswap/file status=none bs=1M count=384

# cat /sys/kernel/debug/swap_page_owner # (pages in swap space)
...
Page allocated via pid 231, tgid 231 (dd), ts 13255313898 ns
SWP entry 0x2e
 get_page_from_freelist+0x13b9/0x1530
 ...
 entry_SYSCALL_64_after_hwframe+0x77/0x7f

# cat /sys/kernel/debug/swap_page_owner | grep -c '^Page allocated .* (dd)' # (pages in swap space, allocated by dd)
65276

# echo $((65276*4096/1024**2)) # (number of 4 KiB pages in MiB; approximately 256 MiB)
254
```

Example 2/2 (before)

Before: initial allocation stack/info is **lost** during swap-out/swap-in.

```
Allocate 256 MiB with dd, push it (swap out), read it (swap in) with cat. – Before: The 256 MiB becomes owned by cat.

# dd if=/dev/zero of=/tmpfs/file status=none bs=1M count=256

# COMM=dd; cat /sys/kernel/debug/page_owner \
   | awk -F '[ ,]' '/^Page allocated .* \('${COMM}'\)/ { PAGES+=2^$5 } END { print PAGES*4096/1024^2 " MiB" }'
        256.559 MiB

# ./DD if=/dev/zero of=/tmpfs-noswap/file status=none bs=1M count=384

# rm /tmpfs-noswap/file

# cat /tmpfs/file >/dev/null

# COMM=dd; cat /sys/kernel/debug/page_owner \
   | awk -F '[ ,]' '/^Page allocated .* \('${COMM}'\)/ { PAGES+=2^$5 } END { print PAGES*4096/1024^2 " MiB" }'
        1.66016 MiB

# COMM=cat; cat /sys/kernel/debug/page_owner \
   | awk -F '[ ,]' '/^Page allocated .* \('${COMM}'\)/ { PAGES+=2^$5 } END { print PAGES*4096/1024^2 " MiB" }'
        258.309 MiB
```

Example 2/2 (after)

After: initial allocation stack/info is **maintained** during swap-out/swap-in.

```
Allocate 256 MiB with dd, push it (swap out), read it (swap in) with cat. – After: The 256 MiB remains owned by dd.

# dd if=/dev/zero of=/tmpfs/file status=none bs=1M count=256

# COMM=dd; cat /sys/kernel/debug/page_owner \
   | awk -F '[ ,]' '/^Page allocated .* \('${COMM}'\)/ { PAGES+=2^$5 } END { print PAGES*4096/1024^2 " MiB" }'
        256.504 MiB

# ./DD if=/dev/zero of=/tmpfs-noswap/file status=none bs=1M count=384

# rm /tmpfs-noswap/file

# cat /tmpfs/file >/dev/null

# COMM=dd; cat /sys/kernel/debug/page_owner \
   | awk -F '[ ,]' '/^Page allocated .* \('${COMM}'\)/ { PAGES+=2^$5 } END { print PAGES*4096/1024^2 " MiB" }'
        256.445 MiB

# COMM=cat; cat /sys/kernel/debug/page_owner \
   | awk -F '[ ,]' '/^Page allocated .* \('${COMM}'\)/ { PAGES+=2^$5 } END { print PAGES*4096/1024^2 " MiB" }'
        7.80859 MiB
```

Discussion

- **Questions**
  - *How did you like the colors?* (j/k; but hope it helps!)
  - Do you think this work may be useful to you?
    - System-wide profiling & monitoring per stack trace.
    - Swap space support for page_owner.
  - Do you see any issues or potential problems?
    - Design, approach, integration, code, etc.
  - Do you have any suggestions or enhancements?
  - Questions from the audience?
- **RFC:** Please comment, if you have a chance.
  - [PATCH RFC 0/9] mm/page_owner: add support for pages in swap space
    https://lore.kernel.org/linux-mm/20251205231721.104505-1-mfo@igalia.com/
- **Thanks!**

Join us!

https://www.igalia.com/jobs