



Strategies to mitigate memory footprint on 16KB kernels

Juan Yescas <jyescas@google.com>

Kalesh Singh <kaleshsingh@google.com>

LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Kernel



**LINUX
PLUMBERS
CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026

Only load the kernel modules that are needed

Memory Increase reason in 4kb and 16kb devices

Loading kernel modules that are not needed, increase the memory footprint in 4kb and 16kb kernels.

What you should do

- Only load the system dkkm modules that are **NEEDED**.
- Only load the vendor dkkm modules that are **NEEDED**.



Memory Allocations

Memory allocations in the kernel

```
struct *page = alloc_pages(GFG_KERNEL, 0);
```

Memory Increase reason

- Drivers are allocating a page when in reality they need less memory.

What we are doing, and what you should do

- Use kmalloc() for allocations smaller than a PAGE
- Use an slab to alloc/free small, fixed-size data structures repeatedly



Use `PAGE_BLOCK_MAX_ORDER=7` to avoid cma mem increase

Memory Increase reason in 16kb devices

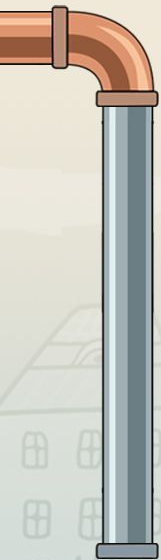
On large page size configurations (16KiB, 64KiB), the CMA alignment requirement (`CMA_MIN_ALIGNMENT_BYTES`) increases considerably, and this causes the CMA reservations to be larger than necessary.

What you should do

- Use `PAGE_BLOCK_MAX_ORDER=7` to have the same CMA alignment requirement that in 4 KiB kernels.
- For more details, see:

<https://lore.kernel.org/all/20250601054731.22245C4CEED@smtp.kernel.org/>





**LINUX
PLUMBERS
CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026

Userspace

Active(file) - File Backed memory mappings

Memory that is file backed using mmap

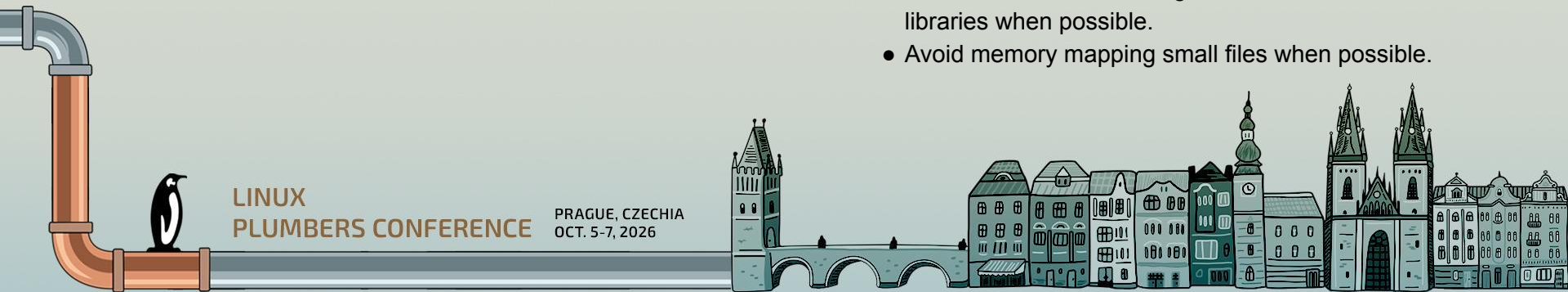
```
int fd = open(fileName, O_RDWR);  
..  
char *ptr = (char *)mmap(NULL, stats.st_size,  
                          PROT_READ | PROT_WRITE,  
                          MAP_SHARED, fd, 0);  
  
char *ptr = (char *)mmap(NULL, stats.st_size,  
                          PROT_READ | PROT_WRITE,  
                          MAP_SHARED_VALIDATE, fd, 0);
```

Memory Increase reason

- Shared libraries segments have to be memory mapped in page size granularity during program loading.
- Small files < 16KiB will use 1 page for the file backed mapping.

What we are doing, and what you should do

- Reduce the number of segments in the shared libraries when possible.
- Avoid memory mapping small files when possible.



Anon Pages - Anonymous Memory

Memory that is not file backed:

Stack memory

```
void do_task() {  
    char buffer[PAGE_SIZE];  
    char buffer[4096];  
    ...  
}
```

Heap memory

```
char *buffer = new char[PAGE_SIZE];  
char *buffer = new char[4096];
```

Private memory mappings

```
char *ptr = (char *)mmap(NULL, stats.st_size,  
    PROT_READ | PROT_WRITE,  
    MAP_ANON | MAP_PRIVATE, -1, 0);
```

Private file memory mappings

```
int fd = open(fileName, O_RDWR);  
char *ptr = (char *)mmap(NULL, stats.st_size,  
    PROT_READ | PROT_WRITE,  
    MAP_PRIVATE, fd, 0);
```

Note: Only memory writes in these mappings become anonymous.

Memory Increase reason

- Stack or heap allocations use more memory than is required:

```
char *buffer = new char[getpagesize()];
```

- The minimum size of the memory mapping is equal to the page size.

What we are doing, and what you should do

- For stack or heap allocations, allocate only the memory needed.
- Avoid creating small memory maps when possible.
- Combine multiple maps onto the same page, since there is more space now.



Shmem

Memory used for ram-backed files (tmpfs fs)

```
int size = getpagesize();  
int shmid = shmget(key, size, 0666);
```

Memory Increase reason

- There are hundreds of tmpfs files that are memory mapped in **tmpfs**. In 16KiB page size kernels, the minimum size of the memory mapped is 16 KiB, instead of 4KiB.

What we are doing, and what you should do

- Avoid memory mapping small files when possible.



Questions



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Thanks



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

