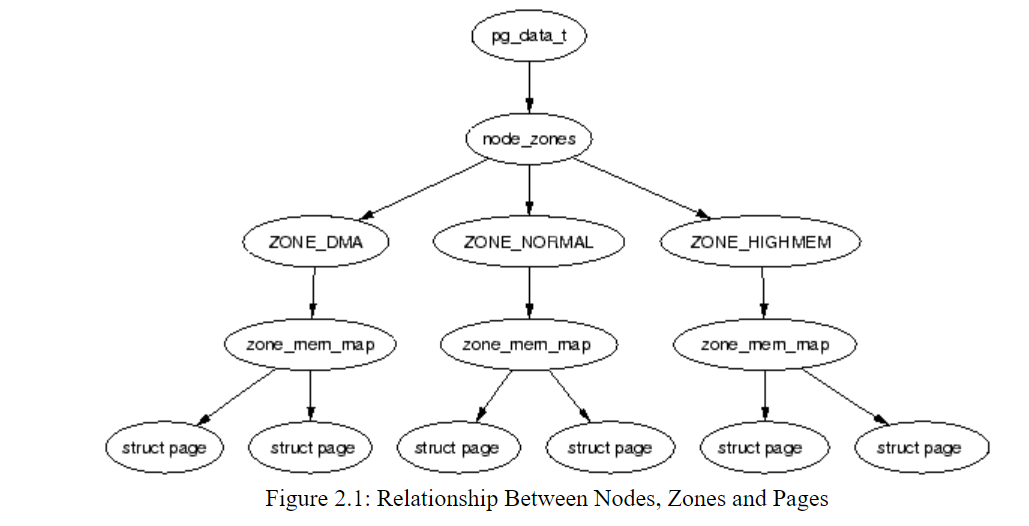
# **Kernel Memory**

**Section 1: Memory organization**



Nodes:

Data structure: pg\_data\_t  => typedef for a struct pglist\_data.

Zones:

Data structure zone\_t => typedef for zone\_struct

|  |  |
| --- | --- |
| **ZONE\_DMA** | First 16MiB of memory |
| **ZONE\_NORMAL** | 16MiB - 896MiB (Most of kernel operations) |
| **ZONE\_HIGHMEM** | 896 MiB - End |

**Process Address Space:**

* Described by the mm\_struct struct
* only one exists for each process and is shared between userspace threads.
* Each process’s task\_struct structure has a pointer to mm\_struct.

<https://linux-kernel-labs.github.io/refs/heads/master/labs/memory_mapping.html>

**Structures used for memory mapping**

### struct page: **struct page** is used to embed information about all physical pages in the system. The kernel has a **struct page** structure for all pages in the system.

### struct vm\_area\_struct: **struct vm\_area\_struct** holds information about a contiguous virtual memory area. Created at each **mmap()** call issued from user space.

### struct mm\_struct

**struct mm\_struct** encompasses all memory areas associated with a process.

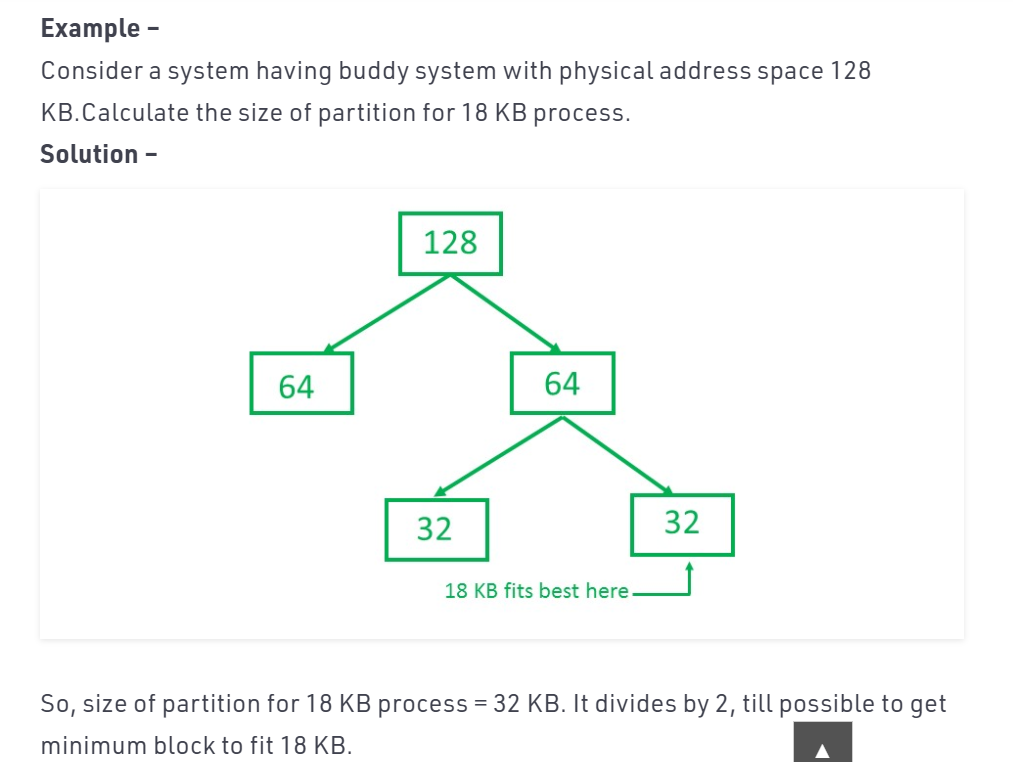
root@qemux86:~# cat /proc/1/maps

**Section 2: Allocation Schemes:**

* Buddy Allocation:
* Slab allocation

**Buddy Allocation**: The **buddy system** is a memory allocation and management algorithm that manages memory in **power of two increments**. Assume the memory size is 2U, suppose a size of S is required.

* **If 2U-1<S<=2U:** Allocate the whole block
* **Else:** Recursively divide the block equally and test the condition at each time, when it satisfies, allocate the block and get out the loop.



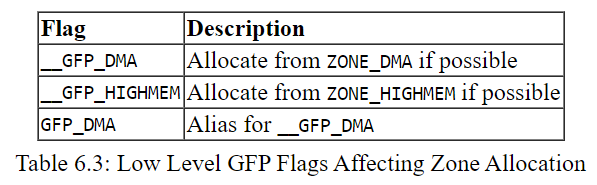
**Slab Allocation**: Slab allocator caches commonly used objects kept in an initialised state available for use by the kernel. The slab allocator aims to cache the freed object so that the basic structure is preserved between uses.

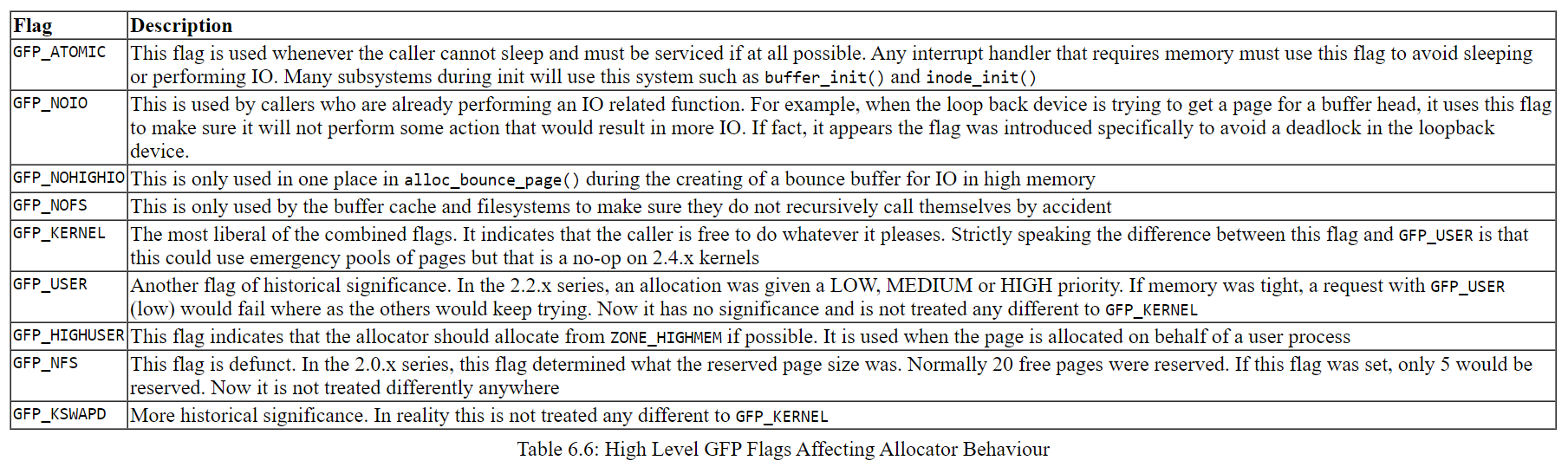
The slab allocator has three principle aims:

* The allocation of small blocks of memory to help eliminate internal fragmentation that would be otherwise caused by the buddy system;
* The caching of commonly used objects so that the system does not waste time allocating, initializing and destroying objects.
* The better utilization of hardware cache by aligning objects to the L1 or L2 caches.

**Section 3: Getting free pages:**

## Get Free Page (GFP) Flags



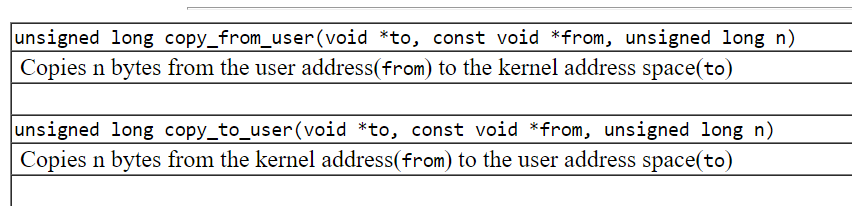


<https://www.kernel.org/doc/gorman/html/understand/understand009.html#htoc42>

The kmalloc() & vmalloc() functions are a simple interface for obtaining kernel memory in byte-sized chunks.

1. The kmalloc() function guarantees that the pages are physically contiguous (and virtually contiguous).
2. The vmalloc() function works in a similar fashion to kmalloc(), except it allocates memory that is only virtually contiguous and not necessarily physically contiguous

**Section 4: Copying data between user and kernel space**



References:

<https://www.kernel.org/doc/gorman/html/understand/understand001.html>

<https://www.kernel.org/doc/html/latest/admin-guide/mm/concepts.html>

<https://linux-kernel-labs.github.io/refs/heads/master/labs/memory_mapping.html>