**Abstract**

**Memory Allocator**

**Objective-**

To extend the SLOB allocator and to compare the internal fragmentation that is resulted from various allocation strategies of the SLOB.

**Description-**

The Linux kernel is equipped with three memory allocators: SLAB and SLUB, and SLOB. These allocators are on a memory management layer that is logically on top of the system’s low level page allocator and are mutually exclusive (i.e. you can only have one of them compiled in your kernel). They are used when a kernel developer calls malloc() or a similar function. They can all be found in the mm directory. All of them follow, to various extends and by extending or simplifying, the traditional slab allocator design.

[ Slab allocators allocate prior to any request, for example at kernel boot time, large areas of virtual memory (called”slabs”). Each one of these slabs in then associated to a kernel structure of a specific type and size. Furthermore, each slab is divided into the appropriate number of slots for the size of the kernel structure it is associate with. ]

**SLAB:**

SLAB organizes physical memory frames in caches. Each cache is responsible for a specific kernel structure. Also, each cache holds slabs that consist of contiguous pages and these slabs are responsible for the actual storing of the kernel structure of the cache’s type. SLAB’s slab can have both allocated(in use) and deallocated (free) slots.

**SLUB:**

SLUB is currently the default allocator of the Linux kernel. It has introduced simplifications in respect to management overhead to achieve better performance. One of the main difference is that SLUB has no metadata at the begging of each slab like SLAB, but instead it has added it’s metadata variables in the Linux kernel’s page structure to track the allocator’s data on the physical pages.

**SLOB:**

SLOB is a stripped down kernel allocator implementation designed for systems with limited amounts of memory, for example embedded versions/ distributions of the Linux. In fact its design is closer to traditional userland memory allocators rather than the slab allocators SLAB and SLUB. SLOB places all objects/structures on pages arranged in three linked lists, for small, medium and large allocations.

The SLOB (Simple List of Blocks) allocator, located in the Linux kernel tree at mm/slob.c, is a piece of the kernel that, unlike the process scheduler, that can easily be extended.

SLOB basically uses 3 types of memory allocation techniques namely best-fit, first-fit and worst-fit techniques.

Internal fragmentation occurs when a memory block of size which is larger than required is allocated. The extra memory space gets wasted as it cannot be utilized in allocation of memory for some other block.

The best-fit technique finds the best block which is best suited according to the requirement and also where the internal fragmentation is minimum.

Whereas the first-fit technique finds the first memory block that can satisfy the requirements. It does not take into consideration the internal fragmentation and hence maximum wastage of memory takes place if this technique is used.

The worst-fit method, it is just the reverse of the best-fit method. It allocates the largest block available in storage list. The idea is to reduce the rate of production of small blocks.

We will write shell scripts that will make system calls to compute the total amount of memory on the free list as well as the total amount claimed by the SLOB allocator for allocations less than one page (i.e. memory that is either on the free list or has been allocated off the free list and not released).

The values returned by these functions will be used to compute a rough measure of internal fragmentation, and we will use these values to compare the amount of fragmentation that results from using different allocation strategies in the SLOB allocator.

Nishi Jain (121032)

Raj Jhala (121042)

Shailee Mehta (121048)

Shreeya Vachhani (121052)