LAB 3

COP4610 FALL 2017

Rolf Kinder Gilet 5734407
Carlos Larrauri 6055949

1. Introduction

Lab3 have to do with switching the kernel memory allocation from SLAB allocator to SLOB allocator and then change SLOB from the first fit algorithm implementation to the best-fit algorithm implementation so we can reduce SLOB's internal fragmentation rate. We also need to implement two system calls which we will use to generate statistical data over a sample of 100 entries for the amount of memory claimed, and the amount of memory free. Using the results from our 100 entire sample, we will be able to compare first fit and best fit and show that we successfully reduced the internal fragmentation of SLOB allocator. To realize this project, we must understand how SLOB work and most importantly how the first fit algorithm is implemented in order for us to be able to modify it to best fit. We will be using the provided resources which is chapter 8 which really provide us with the much-needed knowledge about memory management and the first fit, best fit and worst fit algorithm. We also need to read SLOB.c to understand the actual implementation of SLOB allocator and finally use our knowledge of system calls from lab 1 and 2 to implement the system calls that will return the statistical data of SLOB internal fragmentation. In all, This Lab really exposes us to the complexity of the memory system and the direct implementation of its components more specifically SLOB allocator.

2. Problem Statement

We must switch the kernel memory allocation from SLAB allocator to SLOB allocator, change SLOB from the first fit algorithm implementation to the best fit, and implement two system calls which will return statistical data over a sample of 100 entries for the amount of memory claimed, and the amount of memory free.

3. Methodology

Change from SLAB to SLOB:

- Inside of linux-2.6.36-dev directory, enter the command ‘make menuconfig‘ in the terminal.
- Press enter on “*General setup”*
- Press N on “*Configure standard kernel features (for small systems)’ to enable it*
- Then create the new RAM if necessary
- then *press enter on SLAB allocator*
- Press enter on SLOB (Simple Allocator).
- Once this is done, press escape (esc) twice to get out of the menu config
- Save when prompted to do so
- lastly compile your kernel that uses SLOB instead of SLAB.

System Call:

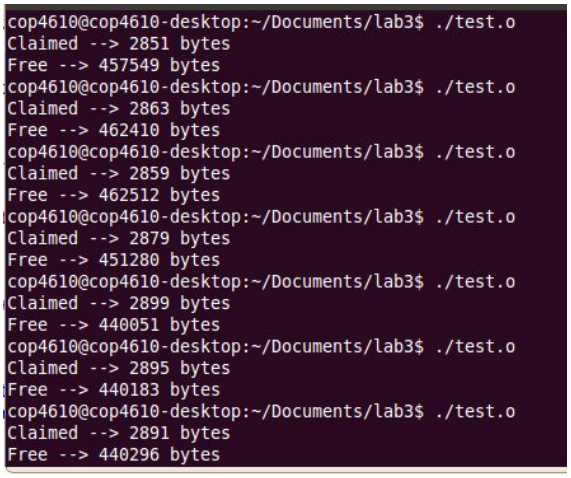
- Go to linux-2.6.36-dev/arch/x86/kernel/syscall_table_32.S and add ".long sys_get_slob_amt_claimed(void) " and ".long sys_get_slob_amt_free(void) " at the end of the list.
- Go to linux-2.6.36-dev/arch/x86/include/asm/unistd_32.h and add "#define __NR_get_slob_amt_claimed <Last_System_Call_Num + 1>" "#define __NR_get_slob_amt_free <Last_System_Call_Num + 1>" at the end of the list
- Go to linux-2.6.36-dev/include/linux/syscalls.h and add "asmlinkage long sys_get_slob_amt_free (void);" "asmlinkage long sys_get_slob_amt_claimed (void);" at the end of the file
- Add #include <linux/linkage.h> in the beginning of SLOB.c
- Write the sys call at the end of SLOB.c (they simply do total of bytes in our sample and divide by sample size.)
- Recompile and install the kernel, reboot

Best Fit implementation:

- Go to linux-2.6.36-dev/mm and open slob.c with your favorite editor
- Add #include <linux/linkage.h> in the beginning of SLOB.c if you haven't done so already
- Modify static void *slob_alloc(size_t size, gfp_t gfp, int align, int node) and static void *slob_page_alloc(struct slob_page *sp, size_t size, int align) by adding if statement inside of them To compare the current memory slot we have with what we are temporarily holding as our best fit. If the current is smaller and is enough to hold the memory allocation request, we update the offset of the temp best fit to the offset of our current memory slot. Similar to iterating a link list looking for smallest. Another thing is if we are at the end of the memory page and we didn't find anything better than what we have on hand for the best fit, we return our best fit and let SLOB do his work of allocating that memory slot. Similarly, if we find a memory slot that is the exact amount we are looking for, we return it immediately since there is nothing better than that.

4. Results

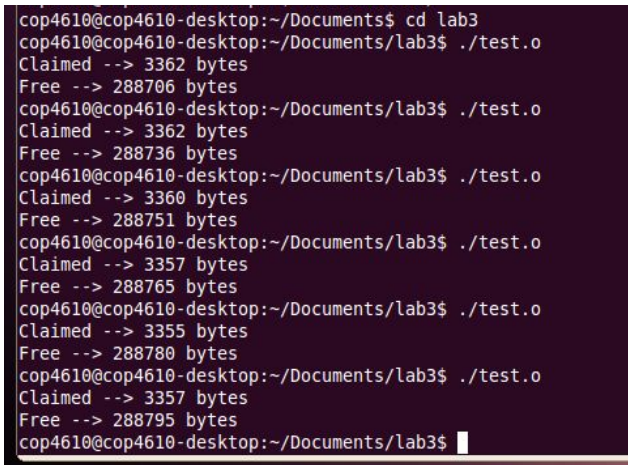
First fit:



Fig(1)

Observing the data we can see that there is a lot of fragmentation, also the memory claimed and memory free are interdependent. As the memory claimed goes up, the memory free (fragmented) goes down.

Best fit



Fig(2)

Observing the data, we can see an improvement in the fragmentation. From 440296 bytes, we dropped to 288751 bytes. We also have the same consistency from Fig(1) in the data.

5. Analysis

SLOB using the first-fit algorithm has a high fragmentation rate, using the data gathered from our system calls, we can observe how the amount of memory claimed, generate a large amount of free memory. This free memory is not usable and as the system make those claims, we can imagine how bad the situation can become. For example we may not be able to satisfy a memory request why we have a lot of free memory (fragmented free memory). With Best fit, we expect to reduce the internal fragmentation of SLOB and make it more efficient(memory wise). Thanks to best fit, we can roughly see that the internal fragmentation of SLOB drop by 40% - 45%. The reason behind that drop of internal fragmentation is the fact that best fit doesn't allocate the first memory slot that is big enough to support the memory request like first fit does. Instead best fit iterate through the entire memory page and find the smallest available memory space that is big enough to satisfy the memory request. We must also point out the slow down that arise from the implementation of Best fit as it iterates through the entire memory page in order to find the best fit, it burns more CPU cycle and takes longer to do the allocation. At the end, we can see why best fit is more efficient memory-wise than the first fit and how first fit is more efficient time-wise compare to best fit. The way we implement SLOB allocator is then tie to our need, for example if we don't mind the extra time, best fit is what we would go with.

6. Member Roles

Me and Nick, Irma and Carlos, exchanged ideas during lab time. Multiple time we were faced with conceptual problems which we would try to tackle together by looking for hypothesis we could test later. We would also share resources found on the internet that we think the others may need for the completion of the lab.

7. Conclusion

This lab introduced us to the Linux memory allocation more precisely to SLOB allocator. We learned how SLOB works and was successfully able to change SLOB from the first fit to best-fit algorithm. We also used two system calls which return statistical data from memory claimed. We used those data to compare first fit and best fit. After the completion of this lab, I fully understand SLOB allocator and the benefits and drawback that may result from the implementation of first fit or best fit in the SLOB.

References

- https://moodle.cis.fiu.edu/v3.1/pluginfile.php/107882/mod_resource/content/1/ch8_ulk.pdf
- https://events.linuxfoundation.org/sites/events/files/slides/slaballocators.pdf
- https://lwn.net/Articles/157944/
- https://github.com/spotify/linux
- https://github.com/fusion2004/cop4610/tree/master/lab3