CS 411 Operating Systems II Project 4

SLOB best-fit alogorithm

Li Li

1 What do you think the main point of this assignment is?

This work was not supposed to be very hard, but our team began in a wrong direction and then almost sweep anything and roll back to the begining. So let me talk about my thoughts to this over 30 hours work. This project mainly need us to do:

- *understand the code* Even SLOB is the simpliest allocator among the three implementations in linux kernel, it is a 626 lines c program. To understand it is much important before trying to hack it. It includes this structs, lists and methods to manipulate it.
- *best-fit implementation* Ways that how we really do it. And some kinds of optimization.
- *testing* This include how kernel space interact with user space, writing a system call. At last, we even built a module to sovle this problem.

2 How did you personally approach the problem? Design decisions, algorithm, etc.

My attampt is: first use a # if BEST_FIT # endif method to make our new approach. Set BEST_FIT to 1 if we want a best-fit solution, otherwise a default first-fit method. And than make our own struct best_slob includes struct slob_page *page; slob_t *prev; slob_t *slob; slob_t *next; slobidx_t size;

Then we constructed two functions static int best_fist(). Basicly what this best_fit function does is traversing a giving slob_list do the alignment and find the best size that fit the request size. The loop would break when we found an exact segment or hit the last element in the list. Finally, after the loop we check if we find the best-fit unit or not, if so return 1, or we make a new page.

After several tests, we sovled all kernel panics. But it was so slow and do some time-out like reject in booting the system. Our way to sovle this problem is to make more slob_list. The original three slob_list turns into 11 lists. We break it into 4-8, 8-16,16-32,32-64,64-128,128-256,256-512,512-1024,1024-2048 and the rest. Finally, although is it really slow, we can make it into the login interface in 5 minutes.

3 How did you ensure your solution was correct? Testing details, for instance.

How we do the testing is a little hard. In slob.c we made a fragmentation function to get the bigest free fragment and total free memory. In the beginning, we make two syscall to pass parameters from kernel space to user space, but we had encounter many unusual large numbers or negtive numbers which seem to be overflow and underflow. It is a dead end, after 3 hours debugging.

Finally, we made a module to call the function in slob.c that we made. It cast some time to fix problem. And finally, we make it works to output positive resluts.

4 What did you learn?

It is hard to build a brand new struct, methods and etc in linux. Printk have so many limits, not easy to run floating points, and it has a limit to print the message. So do not use it too much in debugging. Another thing is explaining your codes to someone else can fix many bugs in the process.