EECS 750 Mini Project #2

PALLOC on Raspberry Pi 3

In this mini-project, you will learn to patch your Linux kernel to replace the kernel's memory allocator with your own (PALLOC). Using PALLOC, you will enable color-aware memory allocation at the kernel and use it to control memory allocation of a user program. You should return **mini-proj2-1.txt, mini-proj2-2.txt** and **mini-proj2-3.txt**.

Part 1. Patch your kernel with PALLOC

Once you boot to the Pi 3, it's time to patch your kernel to support palloc. Below assumes that your linux kernel repository is located in 'linux' subdirectory of your current directory.

```
$ git clone https://github.com/heechul/palloc
$ cd linux
$ patch -p1 < ../palloc/palloc-4.4.patch
```

First, you should change the kernel configuration as follows.

```
$ make oldconfig
...
  Enable PALLOC (CGROUP_PALLOC) [N/y/?] (NEW) y
```

Alternatively, you can directly modify '.config' file to include "`CONFIG_CGROUP_PALLOC=y`" line. After changing the kernel configuration file, do the following to rebuild the kernel.

```
$ make -j4 zImage modules dtbs
```

Again, this will take about 1.5 hour. Note that it has to rebuild everything because the patch changes linux/mmzone.h linux/cgroup_subsys.h, which are included in many part of the kernel.

Once the complication is done, again, do the following to install everything.

```
$ sudo make modules_install
$ sudo cp arch/arm/boot/dts/*.dtb /boot/
$ sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
$ sudo cp arch/arm/boot/zImage /boot/kernel7.img
```

If everything went smoothly, then reboot the system and check if the kernel is the one you just compiled.

```
$ sync; sync; sudo reboot
```

```
...
$ uname -a
```

```
Linux raspberrypi 4.9.80-v7+ #1 SMP Mon **Feb 12 23:07:22 UTC 2018** armv7l
GNU/Linux
```

If your kernel was booted successfully, you would be able to see the following debugfs files.

```
$ sudo bash
# cd /sys/kernel/debug/palloc/
# ls -al
total 0
drwxr-xr-x  2 root root 0 Dec 31  1969 .
drwx------ 29 root root 0 Dec 31  1969 ..
-rw-------  1 root root 0 Feb 23 17:52 alloc_balance
-rw-------  1 root root 0 Feb 23 17:52 control
-rw-------  1 root root 0 Feb 23 17:52 debug_level
-rw-------  1 root root 0 Feb 23 17:52 palloc_mask
-rw-------  1 root root 0 Dec 31  1969 use_mc_xor
-rw-------  1 root root 0 Feb 23 17:52 use_palloc
```

Copy the output of ‘ls -al’ on the /sys/kernel/debug/palloc directory of your Pi3, save it as ‘mini-proj2-1.txt’ file. You should return the file as a proof.

Part 2. Configuring PALLOC

Next, let’s configure the palloc for Pi 3. You Pi 3 has 512KB L2 cache (16 way set associative) and 32KB L1 data cache (4way). We use physical address bit 14,15 for page coloring in PALLOC to partition the L2 cache using four different colors. Configure PALLOC bitmask as follows.

```
# echo 0xc000 > palloc_mask
```

Then, enable PALLOC as follows.

```
# echo 1 > use_palloc
```

Lastly, let’s check if everything is configured and enabled properlly.

```
# cat control
..
mask: 0xc000
```

```
weight: 2 (bins: 4)
bits: 14-15
XOR bits: disabled
Use PALLOC: enabled
#
```

Save the output of ‘cat /sys/kernel/debug/palloc/control’ on your Pi3 as ‘mini-proj2-2.txt’ file. You should return the file as a proof.

Part 3. Color-aware memory allocation using PALLOC

As in Mini Project #1, you will need IsolBench benchmark suite. You can skip the following, if you already downloaded it on your Pi3.

```
$ git clone https://github.com/CSL-KU/IsolBench
$ cd IsolBench/bench
$ make
```

As in the previous project, you will use the ‘latency’ and ‘bandwidth’ benchmarks. In addition, you need ‘pagetype’ program included in IsolBench suite..

```
$ sudo cp latency bandwidth /usr/local/bin/
$ sudo cp pagetype /usr/local/bin/
```

We are now ready to use PALLOC. First, create a ‘subject’ cgroup using PALLOC and configure it to use color 0 as follows.

```
# cgcreate –g palloc:subject
# echo 0 > /sys/fs/cgroup/palloc/subject/palloc.bins
```

As we use 2 physical address bits for coloring, there are 4 different colors (2^2 = 4). Now, let’s launch a bandwidth instance on the ‘subject’ cgroup as follows.

```
# cgexec -g palloc:subject bandwidth -t 1000
```

Now, check the bandwidth instance’s address space to see if the pages are allocated using color 0. Use pagetype program you installed earlier as follows.

```
# pagetype -L -p `pidof bandwidth`
voffset offset  flags
10      2a436   color=1 __RU_lA____M______________________
11      2a437   color=1 __RU_lA____M______________________
```

```
21      30302   color=0 ___U_lA____Ma_b___________________
22      21db1   color=0 ___U_lA____Ma_b___________________
1971    221e2   color=0 ___U__A____Ma_b___________________
76961   18eb0   color=0 ___U_lA____Ma_b___________________
76962   1a673   color=0 ___U_lA____Ma_b___________________
76963   185a1   color=0 ___U_lA____Ma_b___________________
...
...
             total            1313        5
             color[0]         1109        4
             color[1]           75        0
             color[2]           66        0
             color[3]           63        0
```

The results shows that 1109 pages (~4MB) are allocated using color 0.

Save the output as ‘mini-proj2-3.txt’ file. You should return the file as a proof.