MP 4 – Implementing Best-Fit Memory Allocation Scheme in Linux

CS 423 – Spring 2011
Revision 1.0

Assigned March 16, 2011
Due April 6, 11:59 PM
Extension 48 hours (penalty 20% of total points possible)

1 Change Log

1.0 Initial Release.

1.1 Added another sample output 6 and updated 4.3

2 Objectives and Background

The purpose of this MP is to help the students:

- Understand the functioning of the most simple memory allocation subsystem provided by Linux - SLOB (Simple List Of Blocks)
- Modify the currently implemented First-Fit in SLOB to a better scheme of Best-Fit.

3 Overview

In this MP, you'll implement a Best-Fit allocation scheme, as discussed in class. Linux kernel provides various complex memory allocation subsystems such as SLAB and SLUB. But in this MP, we'll work with the most primitive SLOB allocator, which is very similar to the memory allocation scheme discussed in the class. SLOB is particularly useful in small embedded Linux systems, mainly due to its small size and quick & efficient allocation framework. However, the current implementation of SLOB is based on the First-Fit allocation scheme, which reduces the average running time of an allocation request, but at the cost of increased fragmentation. In this MP, we'll explore the Best-Fit scheme, which will increase the average running time of an allocation request, but will (hopefully) reduce the degree of external fragmentation.

4 Description

4.1 Enabling SLOB

Your Linux kernel is by default configured to use the more complex SLAB allocator. We first need to switch to the SLOB allocator by performing following steps:

1. Search for `CONFIG_EMBEDDED` in the `.config` file present in the topmost level of your kernel directory. Replace it by `CONFIG_EMBEDDED=y`
2. Now issue following command in your kernel directory: `make menuconfig`
3. Next navigate to `General Setup - Choose SLAB allocator`. Choose SLOB as the new allocator.

4.2 Modifying the existing SLOB allocator

The implementation of the SLOB allocator is in the file `mm/slob.c`. Familiarize yourself with the functions `slob_alloc()` and `slob_page_alloc()`. The `slob_alloc()` function implements a next-fit scheme to find a page that has enough space for an incomming request. The `slob_page_alloc()` function implements a first-fit scheme to allocate space within a candidate page. In particular, you only have to change the first-fit scheme of `slob_page_alloc()` to a best-fit scheme.

4.3 Kernel Logs

You'll report some statistics related to the best-fit allocation. Use `printk()` to print out these logs, which can be accessed by using **`dmesg`** command from the userspace. Prepend all the `printk`s with **CS423:**, so that it is easy to filter out your logs.

Generally, if we print the stats for every allocation request, it will amount to huge amount of log messages. To counter this, you can print the stats for every n*th* (for exapmle n = 5000) invocation of slob_alloc(). For this, you can define a `static unsigned long counter` in slob.c and increment it every time `slob_alloc()` is called. The print statements in `slob_page_alloc()` shall be called when `(counter % n)` is false.

5 Requirements

1. (25 points) In this assignment you should:
   1. Implement a best-fit scheme by modifying `slob_page_alloc()`.
   2. In `slob_page_alloc()`, print out the amount of space requested, the entire list of holes scanned and the best-fit hole. Note that the effective space of a hole, after alignment adjustment, is given by **`avail - delta`**.

6 Sample Output

```
CS423: Request: 24
CS423: Options: 40, 40, 4, 12, 4, 4, 8, 16, 36, 40, 4, 12, Best-fit: 36

CS423: Request: 92
CS423: Options: 236, 92, 24, 24, 184, 128, 92, 120, 24, 256, 28, Best-fit: 92
```

In the above case, following is also an acceptable output (i.e you can break as soon as an 'exact' fit is found.

```
CS423: Options: 236, 92, Best-fit: 92

CS423: Request: 1024
CS423: Options: 236, 24, 24, 184, 128, 92, 120, 24, 256, 28, Best-fit: None
```

7 Deliverables

You should place in a directory named `mp4`

1. Your modified `slob.c` file.
2. A `README` file, explaining all the modifications made and some logs collected.

You should `handin` a tarball of this directory, named `mp4.tgz`, made by `tar czf mp4.tgz mp4`