Design Virtual Heap

NING CAO

April 9, 2023

1 How would modify your code so that pages are saved to disk when memory capacity is exhausted?

To modify the pm-heap code to save pages to disk when memory capacity is exhausted, we can use a technique called swapping or paging. Swapping involves temporarily moving pages from memory to disk to free up space in memory. When a page is needed again, it can be loaded back into memory from disk. For more detail, when the pm-heap runs out of free pages, we can save the contents of the used pages to a file on disk, free the pages in memory, and then load new free pages from the file as needed.

2 How would you load previously saved pages back in?

To load previously saved pages back into the heap, we can add a function that reads the contents of the heap from a file on disk, and modify functions. Create a function can read the contents of the file into a buffer in memory, and then copy the contents into the `pm_heap` array. Once the contents of the heap have been loaded from disk, the `pm_free_list` pointer can be initialized to point to the first block of free memory.

3 Does your design work for a virtual heap?

The most part of the design is working. But we need to add swapping or paging. To implement swapping in the pm-heap, we need to modify the `pm_malloc()` function to detect when the heap is exhausted and swap out one or more pages to disk. We also need to modify the `pm_free()` function to detect when a page is no longer needed and swap it back into memory if it is not already in memory.

4 Do you need to change your design?

Yes, my design of virtual heap is paged but without page table. To make my pm-heap work for a virtual heap, we need to add swapping or paging and implement page table. This involves dividing the virtual address space into pages, which can be allocated and deallocated independently. When a page is accessed for the first time, it is loaded into memory from disk, and when a page is no longer needed, it can be swapped out to disk. This is a standard technique used by operating systems to manage memory.

5 What kind of file access would you use? Which C functions would you need?

To implement swapping, we need to use file I/O to read and write pages to disk. We can use the standard C functions fopen, fwrite, and fread functions to open, write to, and read from the disk file. We can also use the fseek function to move the file position to a specific page in the file. We also need to keep track of which pages are in memory and which pages are on disk, so that we can swap them in and out as needed.

6 How would you detect when the heap is exhausted?

To detect when the heap is exhausted, we can keep track of the total amount of memory used and compare it to the total amount of memory available. When the used memory reaches a certain threshold, we can start swapping out pages to disk to free up space.

In detail, we can detect when the heap is exhausted in the `pm_malloc()` by checking whether the `pm_free_list` is empty or not. eg:

```c
void* pm_malloc(size_t size)
{
    // Round up the size to a multiple of the page size
    size_t num_pages = (size + PM_PAGE_SIZE - 1) / PM_PAGE_SIZE;
    size = num_pages * PM_PAGE_SIZE;

    // Find a free block of the required size
    pm_block_header* prev_block = NULL;
    pm_block_header* block = pm_free_list;
    while (block != NULL) {
        // original code for finding free block and splitting for larger size
    }
    // heap is exhausted
    return NULL;
}
```

Also, we can set a waterline, for example 80%, when the page table run to the waterline. We can send part of the pages to the disk.

7 How would you know that you needed to reload a page?

To determine when to reload a page, we can use a flag(D bit) in each page that indicates whether it has been modified since it was last loaded from disk. When the `pm_free` function is called, it can check this flag for each page in the heap, and if any pages have been modified, it can call the `write_heap_to_disk` function to save the modified pages to disk. Similarly, when `pm_malloc` is called and there is not enough free space in the heap, it can check the flag for each page in the heap to determine if any pages need to be reloaded from disk.

8 What kind of page replacement strategy would you implement and how?

For page replacement, we can use a simple least recently used (LRU) strategy, where the page that has been unused for the longest time is swapped out. This can be implemented using a timestamp or a counter that is updated every time a page is accessed.

The page replacement strategy, such as LRU or FIFO, determines which pages are evicted from RAM when memory is exhausted. For example, LRU would evict the least recently used page from RAM, while FIFO would evict the oldest page. The choice of page replacement strategy can depend on the specific requirements of the application and the characteristics of the data being stored.

As for my design, I will choose LRU as our replacement strategy.