Project 2 Final Write-up

November 15, 2018

Team `arpl-yiannig`: Áron Ricardo Perez-Lopez, Yianni Giannaris

Design

We changed our design significantly from the beta. The major difference is that, in the beta, we had fixed size blocks (whose size was a power of two). In our final submission, we relaxed this constraint. Each bin now holds blocks that have sizes within a specific range of values. In addition to modifying the structure of our binned free list, modified our header to just be the size of available space for the user and added a footer. Both the header and the footer were 8 bytes in size.

The new allocator operates with a modified version of binned free lists. It reserves blocks of the requested size. Upon freeing, they get put into their respective bin (rounded up to the closest power of two).

We finished (reimplemented) coalescing. Our coalescing feature works by coalescing with blocks immediately to the left and right of the block we are trying to free. The footer came in handy during coalescing. We could easily determine the size of the previous block by observing the footer, which was immediately to the left of the current block's header. If the footer was nonzero, the block was free and had a size indicated by the value stored in the footer. If the footer had a value of zero, the block was in use and could not be coalesced. Coalescing allowed us to combine adjacent free blocks into larger blocks. We also use the very last free block as a "user-space heap" (separately from the bins), so if we can't find enough space, we only request as much memory from the system as needed and merge it into our free block at the end of the heap.

Our `realloc` has some more tricks: if there is an adjacent free block that is big enough, the current block will "expand" into it instead of allocating a new, larger block. Similarly, if the block is at the end of the heap, the program will just request more heap space from the system instead of using a larger block, thereby saving the time it takes to copy the data from the old location to the new one.

Performance

Our beta submission achieved a score of 53.6 on our own machines. After implementing the new allocation strategy, we got 74.9. Afterwards, we implemented a faster logarithm (to decide which bin the free blocks should go into). Our original logarithm function had utilized a while loop, which was a major bottleneck. We found a better implementation of a logarithm function online and improved our performance index by 8 points. With this, we reached 93.1 as a final score.

Testing

We used the check function for testing, although we had to fix a few bugs there as well, reflecting the changes in our design, to make sure that the space occupied by the header and the footer were both accounted for.

Bugs

Most often we forgot to initialize some variables, such as setting newly declared pointers to null, so we were trying to read garbage from those locations. It also took a lot of time to get all of the coalescing and reallocation edge cases right, but it helped a lot that we modularized our code from the beta that contained very long functions.