Thread-Safe Malloc

Xueqian Hu | Netid:xh110

ECE650:HW2

1 Thread Safe Malloc Implementation Detail

1.1 Lock version Malloc

The idea of implementing lock version malloc is pretty straight forward: I will lock the whole malloc/free before the function called, and unlock it after malloc/free complete.

1.2 No-lock version Malloc

How to implement no-lock version malloc, first we have to figure out the reason that race competition happens. The most straight forward one is that the sbrk function. When multiple threads need to expand the heap through sbrk function, they might first use sbrk(0) to get the same current address then move the top of the head according to the current address they allocated, so we need to add a lock before using the sbrk to allocate new space. But sbrk not the only possibility of race competition. Each time when we manipulate the free block list, including splitting, allocating and merging, we might cause race competition. So, the idea is to make sure each thread can use its own list.

2 Result and Analysis

My result is shown below:

Version	Execution time	Data Segment Size
Lock	1.67	42900344
Unlock	0.37	42574400

Figure 1: result

We can see from the result, the unlock version is much faster than the lock version. That is because the lock version actually serializes everything. However, the unlock version take some advantages of multiple threading.