Contents

1 Design

The general idea is that I use an explicit lock for the locking version, and used thread-local storage for the no-lock version. Specifically, since the two versions use different global head/tail, I split all the malloc/free helper functions in two.

```
// lock version
void updateBlock_lock(block_t *cur, size_t size);
void addBlock_lock(block_t *ptr);
void mergeBlocks_lock(block_t *lhs, block_t *rhs);

// no-lock version
void updateBlock_nolock(block_t *cur, size_t size);
void addBlock_nolock(block_t *ptr);
void mergeBlocks_nolock(block_t *lhs, block_t *rhs);
```

1.1 lock version

I used an explicit mutex lock in the lock version. The implementation abstraction is:

```
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;

pthread_mutex_lock(&lock);
// malloc & free function body here
pthread_mutex_unlock(&lock);
```

Basically, I made all the function body part of malloc and free as critical code section. I use a mutex lock at the beginning of malloc and free, and unlock before every return inside the function body. It is worth noting that we should not wrap a second lock around `sbrk` in the lock version, else it would end up with dead-lock and the execution never ends.

1.2 no-lock version

I used thread-local storage for the no-lock version. The idea is that every thread uses its own linked list, so I created thread-specific new head and new tail as:

```
__thread block_t *nolockhead = NULL;
__thread block_t *nolocktail = NULL;
```

Then I copy and paste all the malloc/free helper functions from project1, modified the head and tail used inside the functions, and wrapped a lock around the `sbrk` function since it's not thread-safe.

2 Performance Analysis

I ran the tests with `-ggdb3` debug flag on a 2-processor, 4 GB base memory, Ubuntu 20 virtual machine on Duke VM server.

2.1 Results

To be more accurate, I conducted the measurements 7 times for lock and no-lock respectively. The results are in Figure 1 and 2.

```
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test
No overlapping allocated regions found!
Test passed
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_malloc_free
No overlapping allocated regions found!
Test passed
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_malloc_free_change_thread
No overlapping allocated regions found!
Test passed
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.089617 seconds
Data Segment Size = 43701568 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.150699 seconds
Data Segment Size = 43683368 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.120207 seconds
Data Segment Size = 44277552 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.133242 seconds
Data Segment Size = 43661960 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.135312 seconds
Data Segment Size = 42916360 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.113517 seconds
Data Segment Size = 42202664 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.146367 seconds
Data Segment Size = 44305792 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$
```

Figure 1: Lock version Results

2.2 Analysis

2.2.1 Execution time

I calculate the average execution time of the lock version to be 0.12957s, while the average execution time of the no-lock version is 0.09686s. Using the lock version as a baseline, the no-lock version shows a 25.24% improvement in execution time. I account this improvement to the removal of overhead of mutex locks around the whole function

```
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test
No overlapping allocated regions found!
Test passed
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_malloc_free
No overlapping allocated regions found!
Test passed
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_malloc_free_change_thread
No overlapping allocated regions found!
Test passed
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.094361 seconds
Data Segment Size = 43067712 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.100493 seconds
Data Segment Size = 43094840 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.092210 seconds
Data Segment Size = 42147184 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.087526 seconds
Data Segment Size = 43521760 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.111381 seconds
Data Segment Size = 42844592 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.106344 seconds
Data Segment Size = 42374368 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.085705 seconds
Data Segment Size = 42259464 bytes
sl846@vcm-38454:~/ece650/project2/thread_tests$
```

Figure 2: No-lock version Results

body of malloc and free. Since the no-lock version has thread-local linked list storage, there is no need to take care of the global list race conditions, thus the parallelism can be more flexible, resulting in a shorter execution time.

2.2.2 Data segment size

I calculate the average data segment size of the lock version to be 43,637,274 Bytes, while that of the no-lock version is 42,901,507 Bytes. They are roughly the same because they are both using the best-fit allocation policy. The slight decrease in the no-lock version might be due to the fact that memory is more fragmented and used in better efficiency with separate thread-local linked lists.