BILKENT UNIVERSITY

CS342 OPERATING SYSTEMS

SPRING 2021

PROJECT #3 REPORT

CEMRE BİLTEKİN 21802369

MUSTAFA YAŞAR 21702808

SECTION 1

OVERVIEW

We have found out that buddy allocation algorithm may result in both external and internal fragmentation. However, compared to basic allocation algorithms, buddy allocation algorithm allows little external fragmentation as it serves a form of compaction [1]. When the requested size is smaller than the allocated block for the request, internal fragmentation occurs. When there are holes which cannot coarse together to allow more memory to be stored since they are not buddies; this results in external fragmentation. External fragmentation occurs when memory cannot be allocated although the total free memory permits it. External fragmentation is harder to measure compared to internal fragmentation since it requires a specific scenario (it does not happen all the time, and buddy system actually aims to give little external fragmentation). Thus, in the experiment, only internal fragmentation is regarded and calculated.

There are some observations worth noting before moving on to the experiment. The observations are the following:

- If all requests to memory fit exactly to a block (2^n sized), there is no internal fragmentation.
- Internal fragmentation is maximum when the request sizes are 2^n+1 (worst case).
- External fragmentation usually happens when merge/coarse cannot happen in deallocation due to the buddy system requirements. It indicates a scenario where buddies of free blocks are occupied, so those free blocks remain in little chunks and cannot be merged together due to unavailability of the buddies (because they can only merge with their buddies).

The experiment is designed using the uniform distribution properties. We know the minimum request and maximum request sizes which we can give to the uniform distribution. Then, according to uniform distribution, we generate a random variable that indicates the request size. This is how the request size is obtained for test allocations.

Experiment gives an understanding and measurement about internal fragmentation. It makes N = 1000 allocations (N is chosen as enough to make sufficient allocations; it is tested that 1000 allocations are enough to fill the memory sufficiently as it cannot allocate anymore after a while) with the generated request sizes per different segment sized memory and compares their results. The results for each trial is in the form of a ratio:

$$\frac{overallocated\ memory}{allocated\ memory}$$

Then, the results are plotted for each segment size trial. Overallocated memory means allocated memory - requested memory. Also, overallocated memory amount is plotted for each segment size.

EXPERIMENT RESULTS AND INTERPRETATIONS

For each different segment size, the experiment is performed 10 times (trials) and the results' average is computed to get more accurate data.

Trial	Overallocated Memory Size (bytes)	Allocated Memory Size (bytes)
1	8267	32768
2	9460	32768
3	10192	32768
4	7026	32768
5	7339	32768
6	7127	32768
7	9830	32768
8	10371	32768
9	6835	32768
10	9512	32768

Table 1. Trial table for 32 KB of segment size memory

From Table 1, the mean of the trial for overallocation gives: 8595.9 bytes that can be rounded off to 8596. This means that,

$$internal\ fragmentation\ percentage\colon\ \frac{8596}{32768} * \ 100 \ = 26.23\%$$

Trial	Overallocated Memory Size (bytes)	Allocated Memory Size (bytes)
1	18154	65536
2	14037	65536
3	14452	65536
4	12472	65536
5	19465	65536
6	18159	65536

7	17733	65536
8	17108	65536
9	18470	65536
10	14426	65536

Table 2. Trial table for 64 KB of segment size memory

From Table 2, the mean of the trial for overallocation gives: 16447.6 bytes that can be rounded off to 16448. This means that,

$$internal\ fragmentation\ percentage: \frac{16448}{65536} * 100 = 25.10\%$$

Trial	Overallocated Memory Size (bytes)	Allocated Memory Size (bytes)
1	27961	131072
2	28239	131072
3	28642	131072
4	33743	131072
5	26429	131072
6	31271	131072
7	33365	131072
8	37389	131072
9	30796	131072
10	39176	131072

Table 3. Trial table for 128 KB of segment size memory

From Table 3, the mean of the trial for overallocation gives: 31701.1 bytes that can be rounded off to 8596. This means that,

$$internal\ fragmentation\ percentage: \frac{31701}{131072} * 100 = 24.18\%$$

Trial	Overallocated Memory Size (bytes)	Allocated Memory Size (bytes)

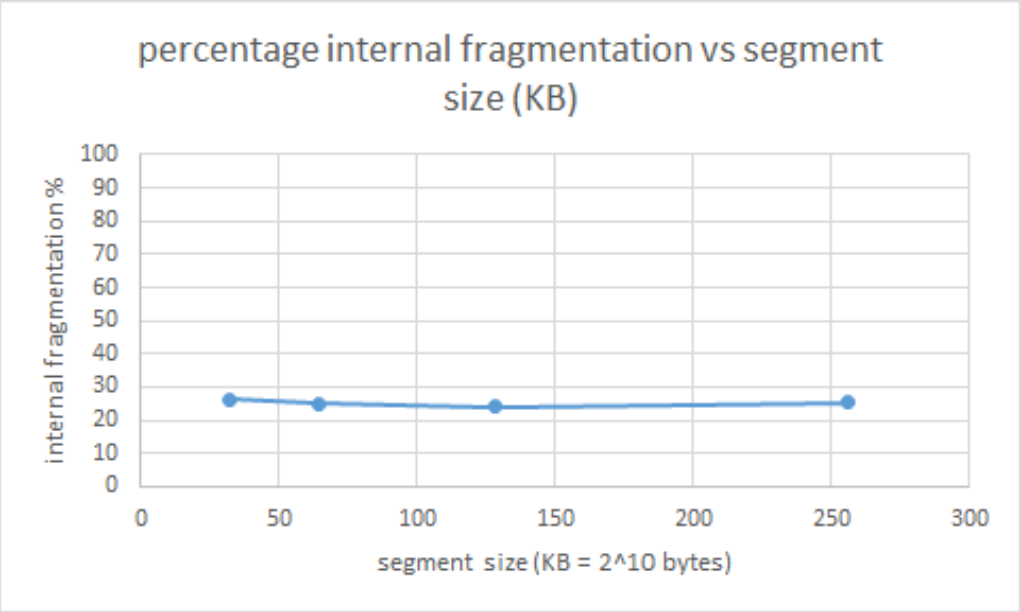
1	71811	262144
2	62962	262144
3	63732	262144
4	65447	262144
5	68358	262144
6	62551	262144
7	74785	262144
8	65347	262144
9	53365	262144
10	76615	262144

Table 4. Trial table for 256 KB of segment size memory

From Table 4, the mean of the trial for overallocation gives: 66497.3 bytes that can be rounded off to 8596. This means that,

$$internal\ fragmentation\ percentage: \frac{66497}{262144} * 100 = 25.37\%$$

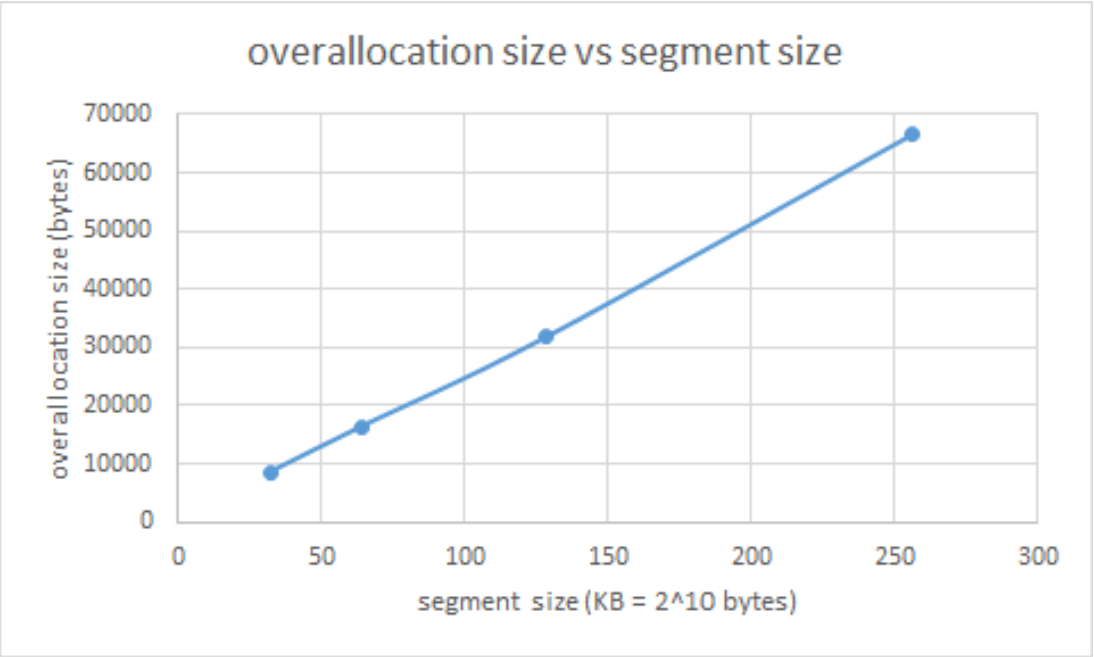
From each result we can deduce that internal fragmentation percentage for different segment sizes yield similar results. With these calculations, Graph 1 is drawn.



Graph 1. Internal fragmentation percentages against segment size (KB) from 32 KB to 256 KB

Graph 1 shows a constant function graph; for all segment sizes, the internal fragmentation results in approximately 25% fragmentation of the total segment size.

Internal fragmentation amount trend is investigated in Graph 2 with the calculated means of each segment experiment against the segment sizes.



Graph 2. Overallocation size (bytes) vs segment size (KB) from 32 KB to 256 KB

In Graph 2, there is an increasing linear trend with overallocation size against segment size. This suggests that as segment size grows, overallocation size grows linearly. Thus, it is noted that Graph 2's slope gives percentage internal fragmentation (considering the whole data) which we can observe in Graph 1 for each segment size.

CONCLUSION

Graph 1 and the tables imply that internal fragmentation percentage is the same for all different segment sizes given the same minimum and maximum request size. Thus, no matter what the segment size is, with the buddy allocation algorithm, the fragmented part occupies about 25% of the total memory size. Additionally, Graph 2 shows overallocated memory size grows as segment size grows in linear fashion. Thus, Graph 2's slope can also give internal fragmentation percentage for generalization. Since as segment size gets larger, it can allow more allocations which results in more internal fragmentation compared to smaller segment size. This confirms the trend in Graph 2. It is shown with the experiment that the buddy allocation algorithm allows 25% fragmentation when all of the memory is allocated.

REFERENCES

[1] “Buddy memory allocation,” *Wikipedia*, 20-Dec-2019. [Online]. Available: https://en.wikipedia.org/wiki/Buddy_memory_allocation. [Accessed: 21-Apr-2021].

APPENDIX - EXPERIMENT PROGRAM

The experiment program test.c:

```
#include <unistd.h>
#include <stdlib.h>
#include <time.h>

#include "sbmem.h"

// Generates random variable according to uniform distribution
int unif()
{
   int num = (rand() % (4096 - 128 + 1)) + 128;
     return num;
}

int main()
{
   int i, ret;

   ret = sbmem_open();
   if (ret == -1)
      exit (1);

   //Experiment here
   srand(time(0));

   for (int i = 0; i<1000; i++){
      //Do allocation with uniform distribution
      sbmem_alloc(unif());
   }

   sbmem_close();

   return (0);
}
```

The below code is a small code piece that was implemented in sbmem_alloc function in sbmemlib.c for experiment purposes before successful returns to print allocation info (overallocated_mem and allocated_mem are integers):

```
overallocated_mem = overallocated_mem + ((temp.finish_addr - temp.start_addr + 1)-size);
allocated_mem = allocated_mem + (temp.finish_addr - temp.start_addr + 1);
printf("Overallocated is: %d, Allocated is: %d\n", overallocated_mem, allocated_mem);
```