CS 342 Operating Systems

Project 4 Report

Mohamad Barham

Erdogan Ege Tokdemir

December 12, 2015

Mohamad Barham 21503965
Erdogan Ege Tokdemir 21200488
December 12, 2015
CS 342 Project 4 Report

Buddy Algorithm Approach

- There exists a list for every chunk size (256 bytes, 512 bytes, … , 32MB)
- Plus, one list for free nodes
- Number of lists = 17 + 1 = 18.

Case-by-case Analysis

- When the user asks to allocate memory of certain size, the list of that size is searched for a free chunk (node)
    - 1. If empty chunk, set to full and return relative internal pointer + base.
    - 2. If no empty chunk, check one level above. Count++. Repeat 2.
        - If empty node found, divide chunk *count* many times. Go to step 1.
        - If no empty chunk, return null.
- When user asks to free memory, search for node in all lists until the node that describes that chunk is found.
    - 1. If no node found, return.
    - 2. If node found, free from memory.
        - If its buddy is free too, combine both into one chunk and add to upper level. Repeat.
- When printing, each list sorted in ascending order. The first node of each list is inserted (using insertion sort) into an array of 18.
    - 1. Print first element in array and remove it.
    - 2. Insert element from the same list where printed element was from. Repeat 1.

Timing Experiments

Experiments have been performed on an Intel i7 8-core machine with 6GB RAM, to measure the amount of time it takes to allocate the entire chunk of 32MB in smaller chunks, ranging from 256B to 64KB. The results can be summarized with the following table:

size	time(s)
256 B	22.94813
512 B	5.577251
1 KB	1.432806
2 KB	0.377052
4 KB	0.11852
8 KB	0.041106
16 KB	0.014981
32 KB	0.005737
64 KB	0.00357

Figure 1: Size of individual chunks in a 32MB chunk vs. time taken for allocation

Mohamad Barham 21503965
Erdogan Ege Tokdemir 21200488
December 12, 2015
CS 342 Project 4 Report

The following graph captures the time taken to allocate as a function of chunk size:

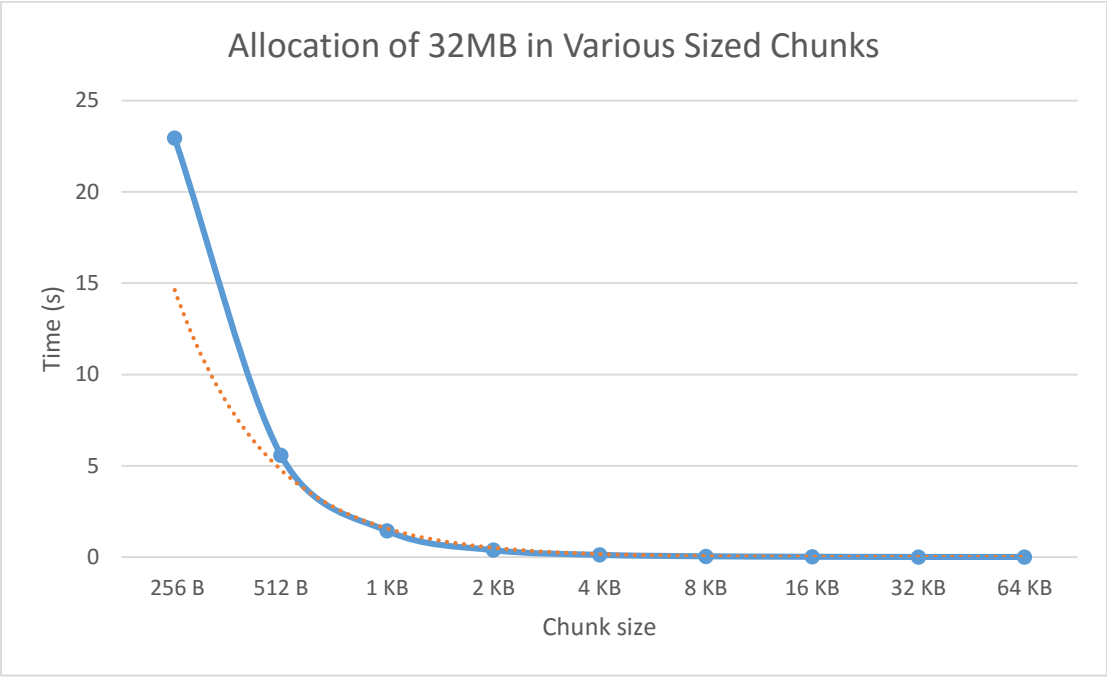

Figure 2: Line graph of figure 1, where blue line represents actual results and dashed orange line is the approximate exponential trend line

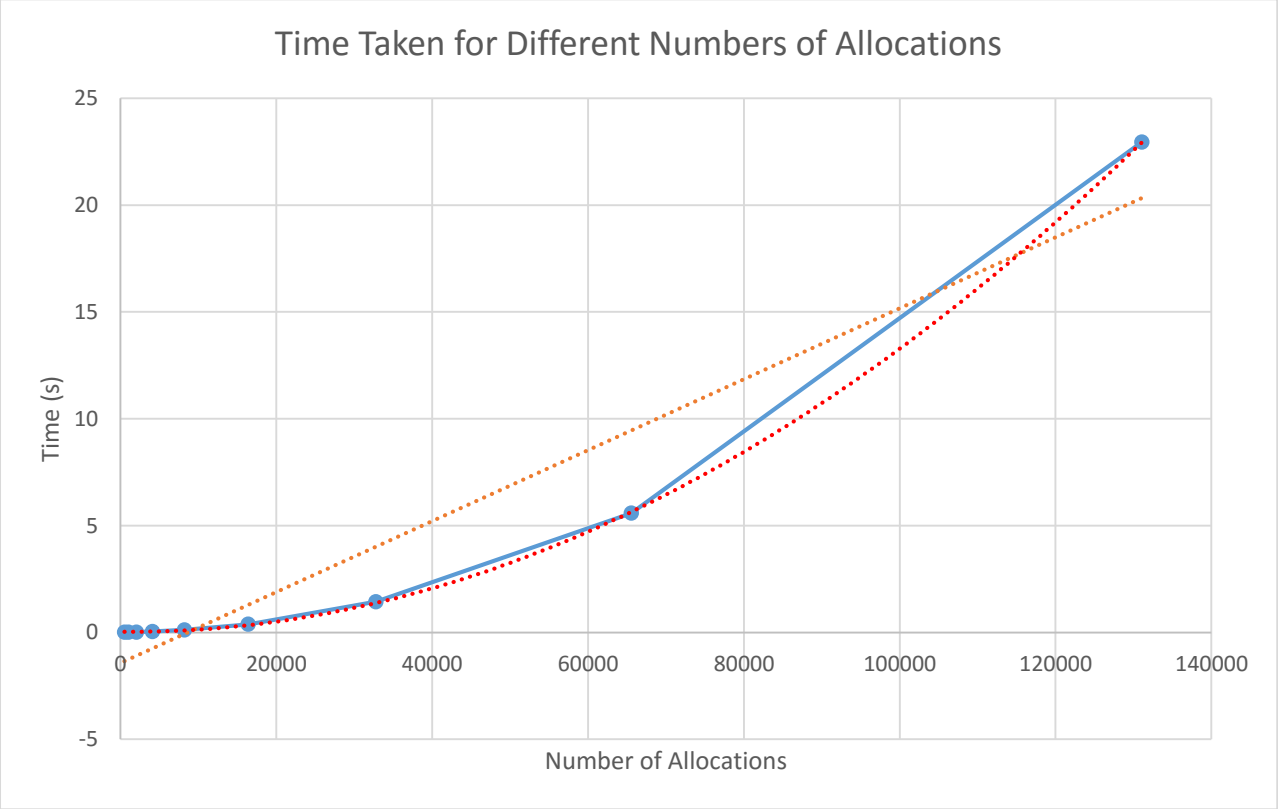

Figure 3: Scatter plot with two trend lines. Polynomial shown in red, linear shown in orange. Blue line shows original data

Mohamad Barham 21503965
Erdogan Ege Tokdemir 21200488
December 12, 2015
CS 342 Project 4 Report

Interpretation:

Figure 2 accurately demonstrates that as the size of the individual chunks increases, and therefore the number of chunks decreases, allocation time decreases significantly. Figure 3 shows that the increasing number of allocations, increases the allocation time significantly. As suggested by the behavior of the graph in figure 2, a linear relationship between the number of allocations and the time taken to make them appears feasible. This is due to the exponential (2^x) scale of the number of allocations, and the time curve appearing to follow this exponential scale almost perfectly. Upon closer inspection (figure 3), however, this relationship appears more quadratic than linear.

Figure 3 manages to scatter the independent variable (number of allocations) on the horizontal axis on a linear scale. As the spreadsheet software calculates a trend line, it appears that a polynomial of order 2, in other words a quadratic is better suited than a linear trend (shown in orange). This means that the time taken to allocate a number of chunks is a factor of the number of chunks squared. The factor appears to be extremely small in this case.

The relation between number of allocations and time taken seems to be more intricate than a linear function; a steady increase in number of allocations will cause a higher increase in the time required. This implies that an operating system is required to consider this relationship and utilize the buddy algorithm accordingly. There should be a lower and an upper threshold of allocation size, so that a steady number of allocation can be made. If an allocation takes time in the order of seconds instead of milliseconds and microseconds, then the number of allocations must be decreased. The operating system is responsible for arranging user data into larger, defragmented chunks in order to save CPU time.