Asst2: Spooky Searching

testplan.pdf

Students: Aditya Oka (aao96, 01), Chris DeAngelis (cjd237)

Workload: The key is equal to a random number between 0 and (MAX - 1). For 150 iterations, search the same key and swap the values in the key's index with a different random index's value in the array after every search.

1. The general trend of time vs size of the array to search with Processes and Threads
    a. To find this aspect from our multi-processed and multi-threaded search algorithms, we decided to take the average elapsed time from 100 workload runs every time we run our "searchtest.c" driver. On top of this, we decided to run our driver 5 times for each size we inputted and get the average from the 5 average elapsed times. From here, we would plot the averages we find for each size in a graph where the y-axis represents time (in milliseconds) and the x-axis represents the size of the array. The sizes we decided to test are: 15, 50, 100, 150, 250, 500, 750, 1000, 1500, 2500, 5000, 7500, 10000, 15000, 20000, and 25000. We think this will properly determine the general trend of time vs size of the array to search with both processes and threads.
2. The tradeoff point for Processes vs Threads
    a. To find this aspect from our multi-processed and multi-threaded search algorithms, we will use our graphs and test results from the previous part to

determine a tradeoff point. The way to find this out is to select an array size for processes (we selected 250), take the average time it took to complete the workload, and find a point in the thread run data that reaches a similar average time completion as the process average time at its respective array size.

3. The tradeoff point for parallelism for Processes and Threads
    a. To find this aspect from our multi-processed and multi-threaded search algorithms, we decided to choose a constant array size (we selected 250) and manually change the partition sizes in the code so that it would generate more/less processes/threads for searching. For the multi-threaded testing, we made the partition sizes equal to 250, 125, 50, and 25, and for multi-processed testing, we made the partition sizes equal to 250 and 125. These tests will show the point where more threads and processes lead to increasingly inefficient behavior.