HARI SHRAWGI
12/5/19

DSA ASSIGNMENT-5 REPORT

Synthetic Request Generation

Experiment Setup

The test programs “SimulateUniformBuddy” and “SimulateExponentialBuddy” were developed to generate synthetic allocation and de-allocation requests as per the assignment.

The first program generates both the size of allocation request and lifetime of the block uniformly. This is done entirely in C.

For the second program, the sizes are generated using a truncated exponential distribution. This generation is done with the help of Python. Then these exponentially distributed sizes are read in by the C program and coupled with uniform lifetimes to perform the experiment.

The experiment was performed for 1000 time-steps. This is a parameter that can be directly changed in the program.

Results

The results of the experiment qualitatively match the results in the referenced paper. The WeightedBuddy allocator performs better when the requests are exponential.

In the test, we got around 14% less internal fragmentation when the exponential distribution was used.

Comparison of SPHeap and OneBin

Experiment Setup

We have modified the Polynomial arithmetic program from Assignment-2 to compare different allocators. Along with these two, we also perform comparison with the standard library allocator.

To perform timing comparisons, we are timing only the memory management calls that are made as part of the polynomial programs. This allows us to compare the allocators directly without having to worry about external factors.

Now, for testing we created a simple polynomial test input, where the polynomial was of degree 100. This is present in “poly_input” file. Then we performed one polynomial arithmetic operation multiple times using this polynomial test case.

This whole test run was performed 10 times, so that we get a stable result.

HARI SHRAWGI
12/5/19

DSA ASSIGNMENT-5 REPORT

Results

The timing results we got as part of our testing are given in Table 1 below:

	Time taken (in secs) to execute the Polynomial test program		
	Standard Allocator	**OneBin Allocator**	**WeightedBuddy Allocator**
1st Run	2.996517	2.878446	36.108812
2nd Run	3.022364	2.989313	35.251436
3rd Run	2.989439	2.961816	39.178173
4th Run	2.975084	2.885776	37.775111
5th Run	2.986502	2.913438	38.669281
6th Run	2.995745	2.93972	40.663193
7th Run	2.983687	2.934193	39.1558
8th Run	2.999297	2.926194	38.651069
9th Run	2.969174	2.930631	41.328882
10th Run	2.979184	2.952567	38.502753
Average time	**2.9896993**	**2.9312094**	**38.528451**

Table 1: Timing comparison of the three allocators

The results clearly show that OneBin allocator is far superior to the WeightedBuddy allocator. And, this is expected as both the allocation and de-allocation operations of OneBin are order of O(1).

Also, when it comes to memory efficiency, OneBin is superior. The reason being that we have specialized OneBin so that each of the fixed size bins have exactly the size required by the Polynomial program.