

Chapter 5

Memory Management Simulator: MemSimRT

There are so many simulators available to simulate different test cases for scheduling in a real-time operating system [16] [20] [56] [73-76] like Litmus-RT, Mark3, rtsim, etc., but till date, no such simulator is available for simulating memory management algorithm for RTOS. Hence, MemSimRT has been designed to simulate various memory allocators for both SMP as well as NUMA architecture based RTOS. Its front end created in C# while back-end developed using python.

Download MemSimRT using this QRcode:

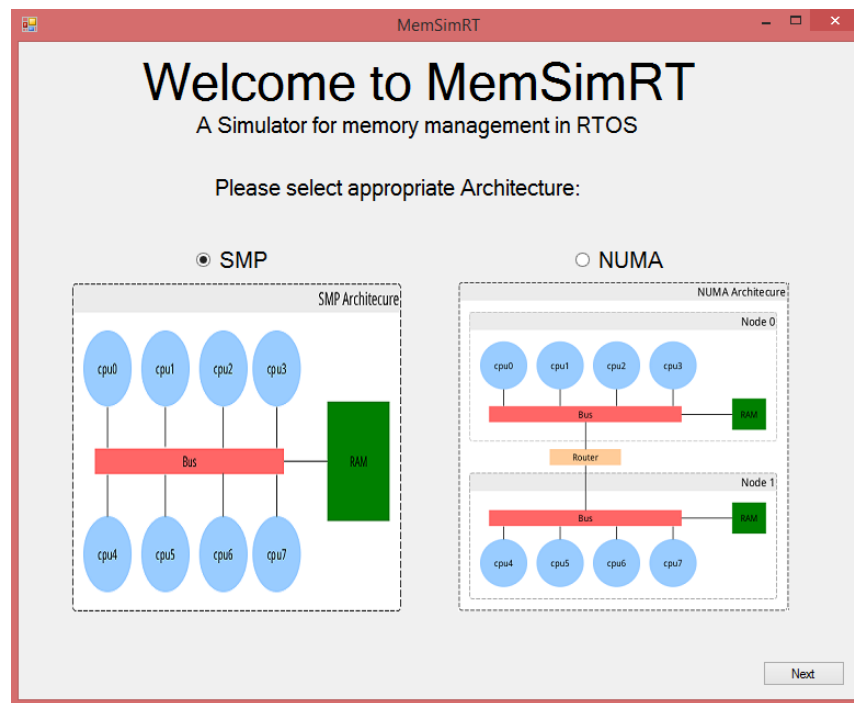


Figure 5.1: The Welcome screen of MemSimRT

Figure 5.1 shows the welcome screen of MemSimRT. So it is the Home screen of the simulator. As per our dynamic memory allocator, it has two alternatives. One is SMP, i.e., Symmetric multiprocessor and second is NUMA, i.e. Non-uniform memory access based architecture. Basically, in this simulator, NUMA is designed for eight processors, but it can be modified as per requirement by slightly changing the script.

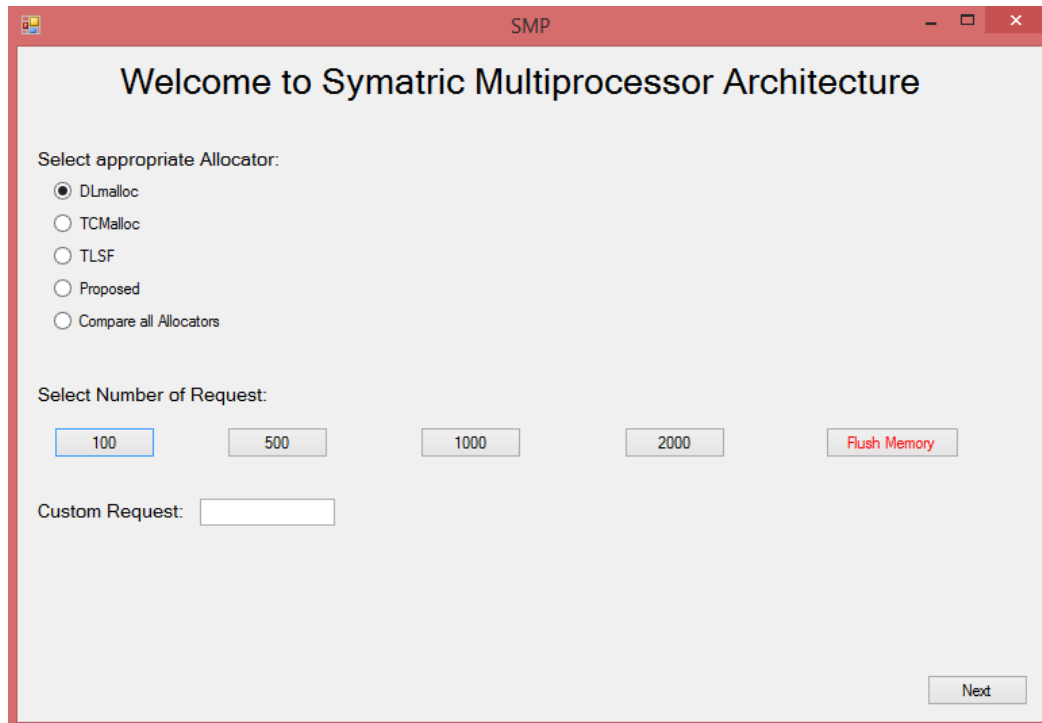


Figure 5.2: The Welcome screen of SMP

Figure 5.2 shows that, when SMP is selected, this screen will appear. One can select appropriate memory allocator for RTOS and select available number of the request. Also, memory block request other than 100, 500, 1000 and 2000 can be provided by defining the specific number in text box of the custom request and then press next button.

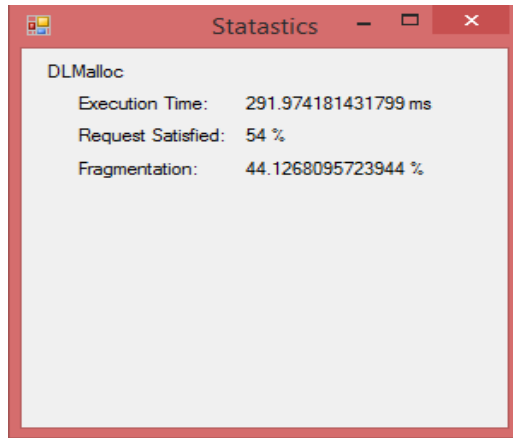


Figure 5.3 shows the result of the individual memory allocator. In result, it shows the execution time in ms (millisecond) taken by specific allocator, how many requests have been satisfied in % as well as Fragmentation generated by allocator in %.

Figure 5.3: Statistics of individual Memory Allocator



Figure 5.4: Memory block allocation as per request by DmRT

Figure 5.4 shows the exact memory allocation scenario by proposed memory allocator which is actually divided into three different categories as per algorithm for small blocks whose size less than 512 bytes, normal or medium blocks whose size is between 512 bytes to 2Mb and Large blocks whose size is beyond 2 Mb. The request denoted by asterisk (*) defines that block allocation is failed. At the end of each memory area, it shows a number of requests satisfied in %.

Also, any number of memory block request can be provided. Direct buttons are available for 100, 500, 1000 and 2000 memory block request. To provide memory block request other than 100, 500, 1000 and 2000, then define it in text box of the custom request and press next button.

Also, to allocate memory blocks in a specific area only like in small memory, medium memory or large memory can be done by selecting the specific checkbox. “*Flush memory*” button is used to flush the entire memory.

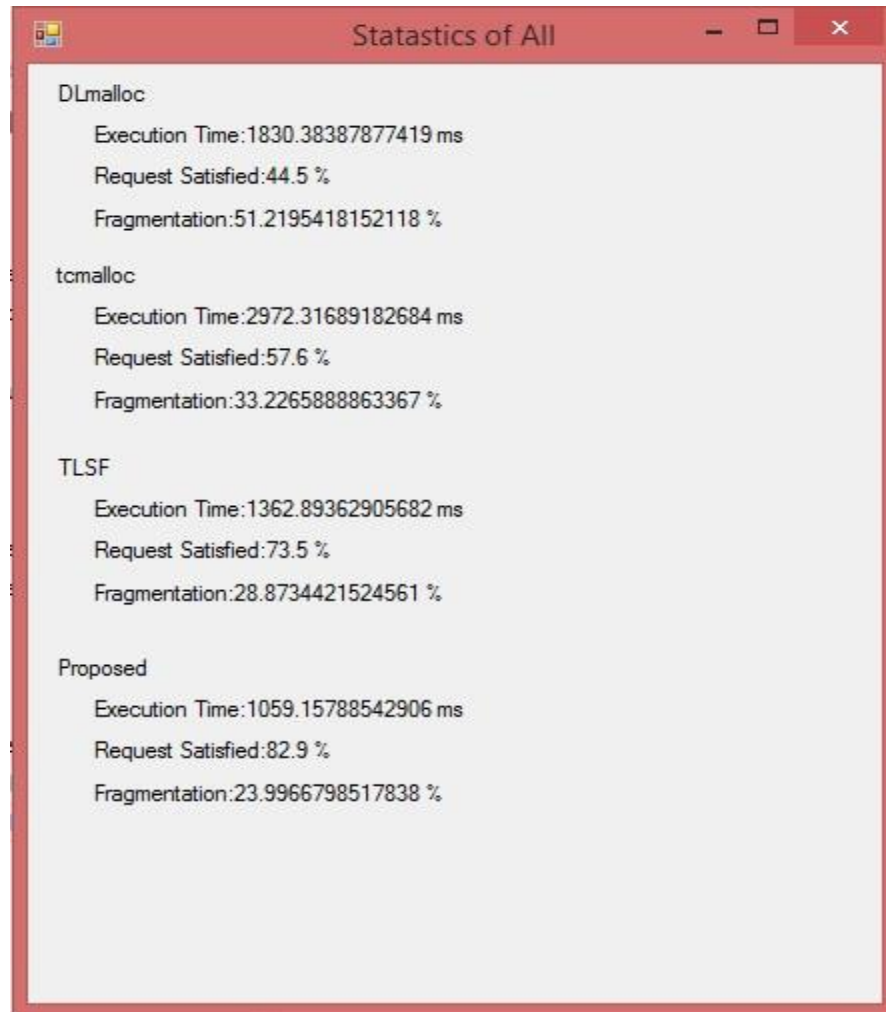


Figure 5.5: Statistics of all memory allocators

Figure 5.5 shows the statistics of all allocators in the same window. If “*compare all allocators*” is selected then result analysis for same block requests of all allocators will be shown in this window.

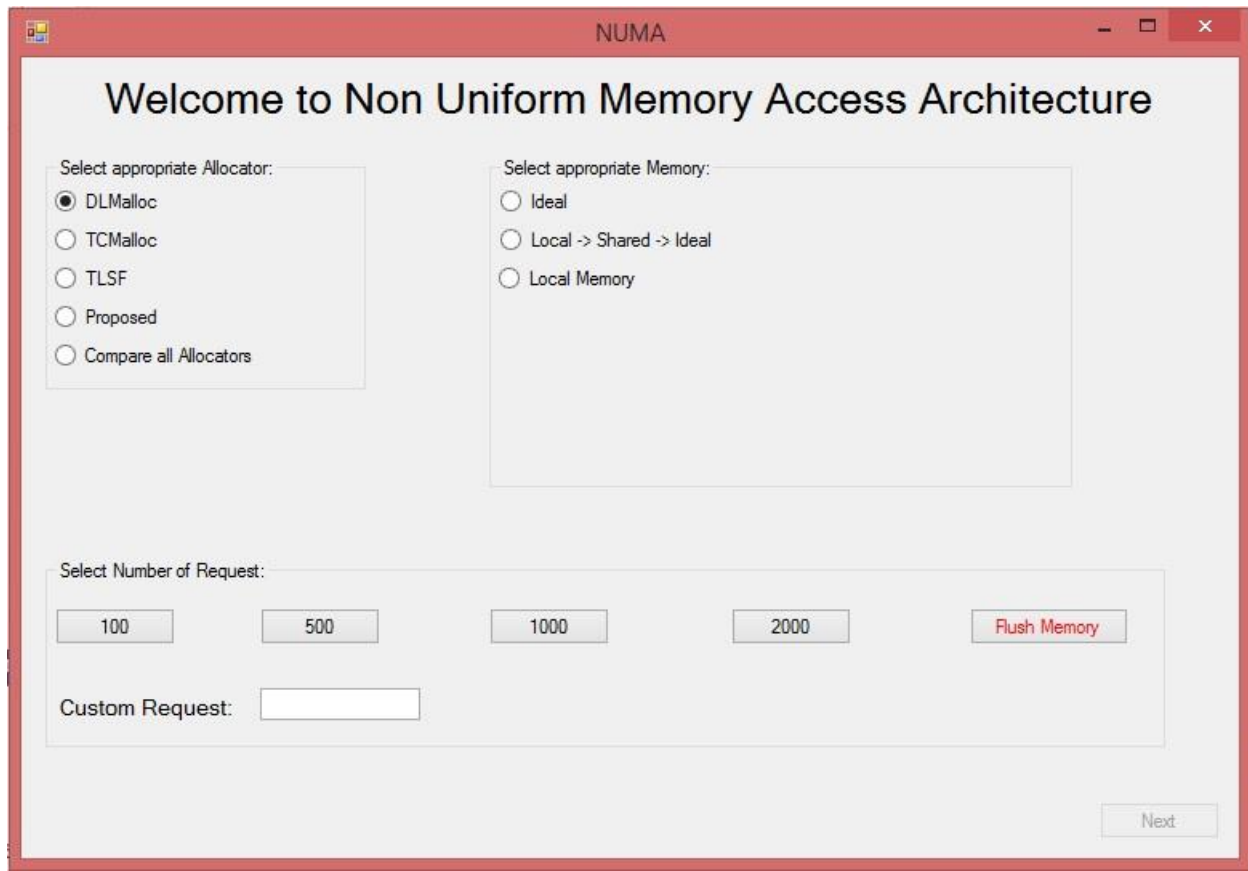


Figure 5.6: The Welcome screen of NUMA

As shown in Figure 5.6, it is the first screen for NUMA, and one can select a specific allocator, appropriate memory, and number of request from it. Memory can be *Local*, *Ideal* or *Local -> Shared -> Ideal*, i.e. it first tries to allocate a memory block from Local Memory if it fails then Shared memory and still if it fails then it will search from the Ideal memory. If Ideal memory is selected then first thing is to find processor which can have ideal memory that is shown in figure 5.7.

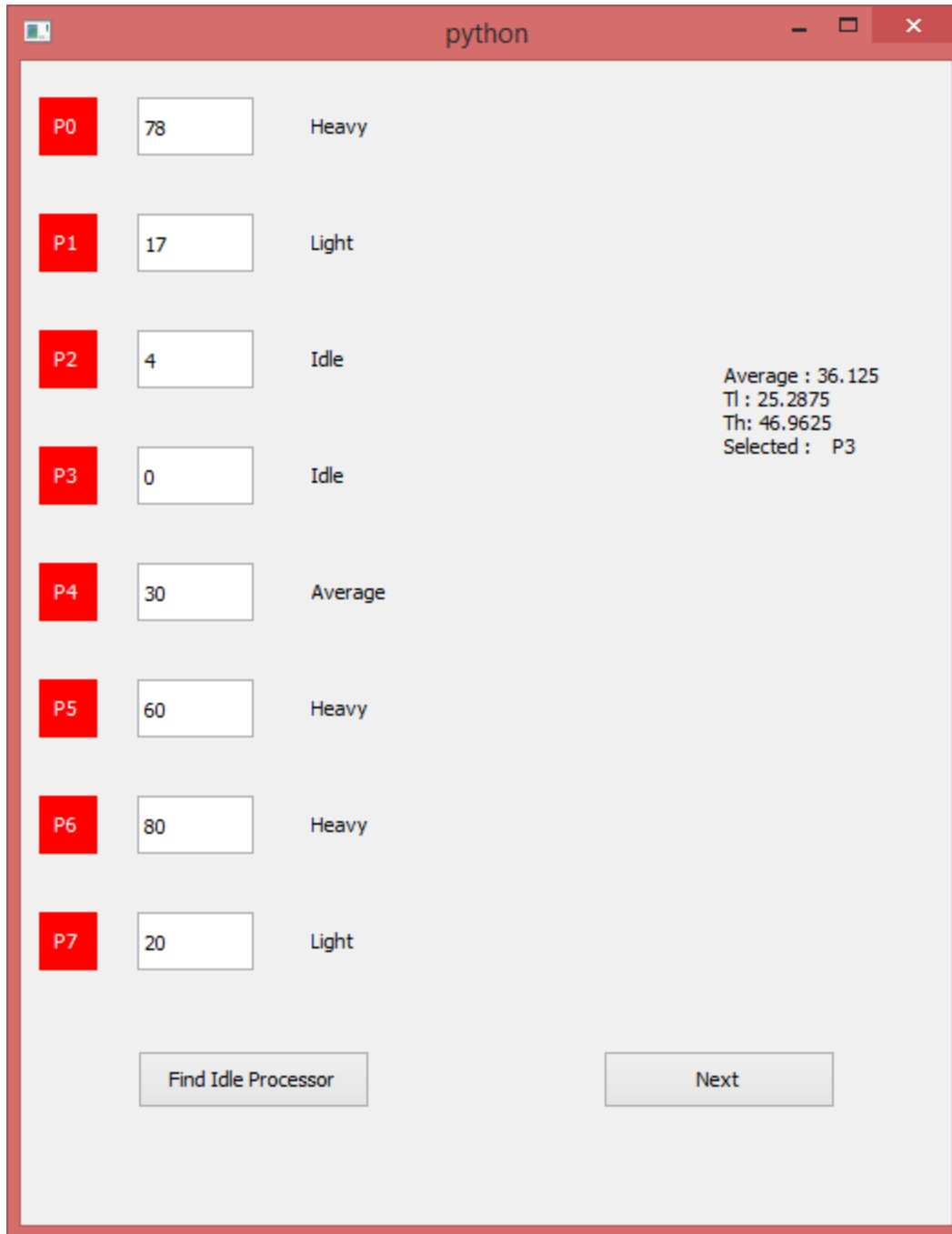
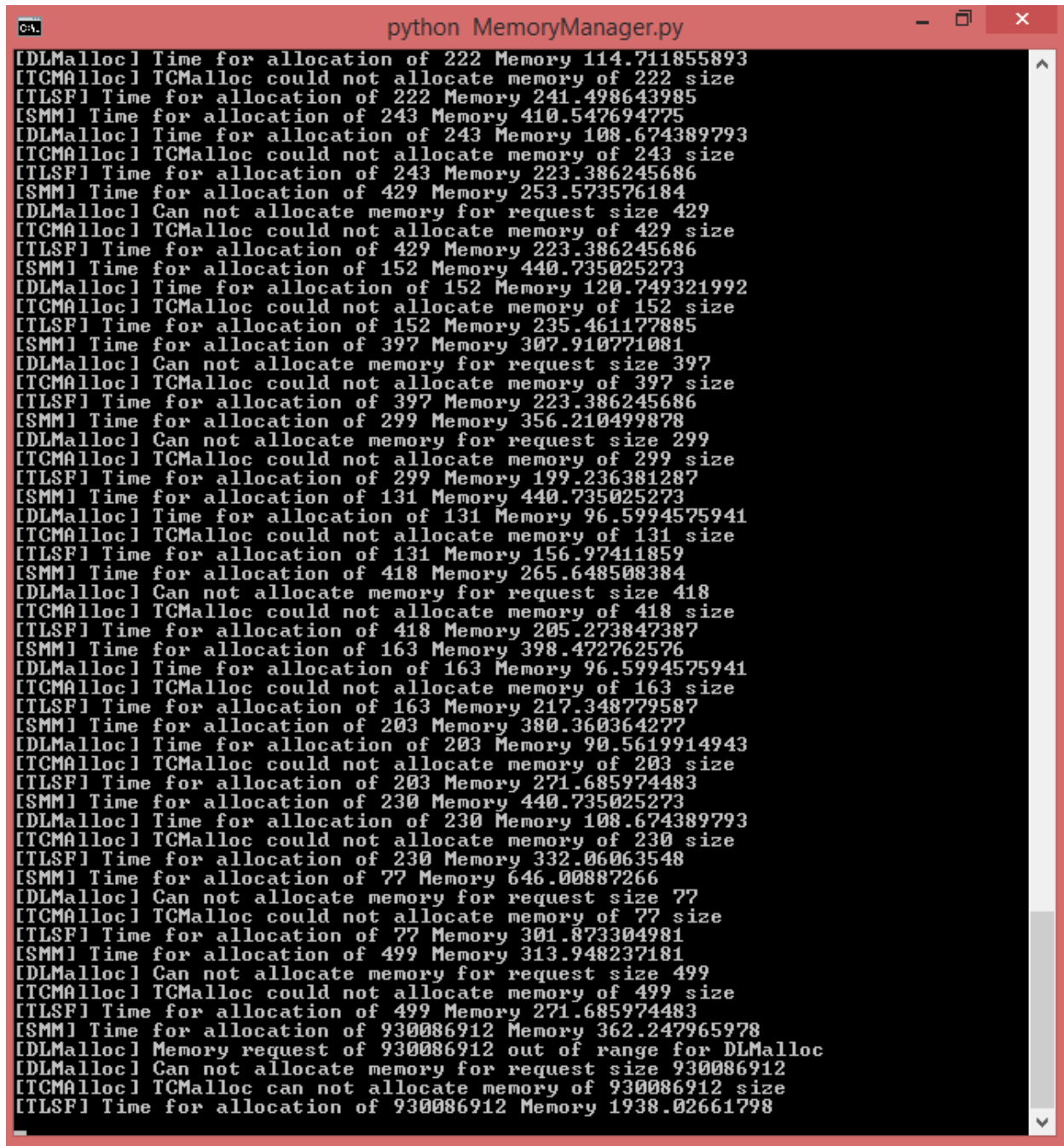


Figure 5.7: All Processors with its memory utilization in (%)

As shown in Figure 5.7, it is a NUMA architecture of eight processors (P0 to P7) and its local memory utilization is defined with it. Also, it has found the category of memory according to the threshold policy. Explicit memory utilization can also be defined by mentioning in the textfield.



```
python MemoryManager.py
[DLMalloc] Time for allocation of 222 Memory 114.711855893
[ICMalloc] ICMalloc could not allocate memory of 222 size
[TLSP] Time for allocation of 222 Memory 241.498643985
[SMM] Time for allocation of 243 Memory 410.547694775
[DLMalloc] Time for allocation of 243 Memory 108.674389793
[ICMalloc] ICMalloc could not allocate memory of 243 size
[TLSP] Time for allocation of 243 Memory 223.386245686
[SMM] Time for allocation of 429 Memory 253.573576184
[DLMalloc] Can not allocate memory for request size 429
[ICMalloc] ICMalloc could not allocate memory of 429 size
[TLSP] Time for allocation of 429 Memory 223.386245686
[SMM] Time for allocation of 152 Memory 440.735025273
[DLMalloc] Time for allocation of 152 Memory 120.749321992
[ICMalloc] ICMalloc could not allocate memory of 152 size
[TLSP] Time for allocation of 152 Memory 235.461177885
[SMM] Time for allocation of 397 Memory 307.910771081
[DLMalloc] Can not allocate memory for request size 397
[ICMalloc] ICMalloc could not allocate memory of 397 size
[TLSP] Time for allocation of 397 Memory 223.386245686
[SMM] Time for allocation of 299 Memory 356.210499878
[DLMalloc] Can not allocate memory for request size 299
[ICMalloc] ICMalloc could not allocate memory of 299 size
[TLSP] Time for allocation of 299 Memory 199.236381287
[SMM] Time for allocation of 131 Memory 440.735025273
[DLMalloc] Time for allocation of 131 Memory 96.5994575941
[ICMalloc] ICMalloc could not allocate memory of 131 size
[TLSP] Time for allocation of 131 Memory 156.97411859
[SMM] Time for allocation of 418 Memory 265.648508384
[DLMalloc] Can not allocate memory for request size 418
[ICMalloc] ICMalloc could not allocate memory of 418 size
[TLSP] Time for allocation of 418 Memory 205.273847387
[SMM] Time for allocation of 163 Memory 398.472762576
[DLMalloc] Time for allocation of 163 Memory 96.5994575941
[ICMalloc] ICMalloc could not allocate memory of 163 size
[TLSP] Time for allocation of 163 Memory 217.348779587
[SMM] Time for allocation of 203 Memory 380.360364277
[DLMalloc] Time for allocation of 203 Memory 90.5619914943
[ICMalloc] ICMalloc could not allocate memory of 203 size
[TLSP] Time for allocation of 203 Memory 271.685974483
[SMM] Time for allocation of 230 Memory 440.735025273
[DLMalloc] Time for allocation of 230 Memory 108.674389793
[ICMalloc] ICMalloc could not allocate memory of 230 size
[TLSP] Time for allocation of 230 Memory 332.06063548
[SMM] Time for allocation of 77 Memory 646.00887266
[DLMalloc] Can not allocate memory for request size 77
[ICMalloc] ICMalloc could not allocate memory of 77 size
[TLSP] Time for allocation of 77 Memory 301.873304981
[SMM] Time for allocation of 499 Memory 313.948237181
[DLMalloc] Can not allocate memory for request size 499
[ICMalloc] ICMalloc could not allocate memory of 499 size
[TLSP] Time for allocation of 499 Memory 271.685974483
[SMM] Time for allocation of 930086912 Memory 362.247965978
[DLMalloc] Memory request of 930086912 out of range for DLMalloc
[DLMalloc] Can not allocate memory for request size 930086912
[ICMalloc] ICMalloc can not allocate memory of 930086912 size
[TLSP] Time for allocation of 930086912 Memory 1938.02661798
```

Figure 5.8: Memory block allocation Log

As shown in Figure 5.8, its memory allocation log. It is the status of requested memory block of the specific allocator. Also, it generates the CSV file for the same.