Scudo Secondary Caching

Fernando Salas

frsalas513@gmail.com

Scudo Background

- LLVM Hardened Memory Allocator

Primary Allocator:

- Smaller Allocations (<65KB Android Config)

Secondary Allocator:

- Larger Allocations (>65KB Android Config)
- mmap() based
- Uses cache and Linked-List of In-Use memory blocks

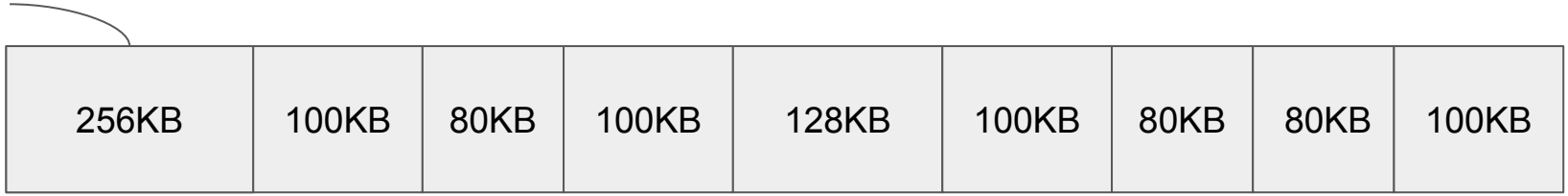
Optimizing Caching Scheme

- Secondary allocator called less frequently
- Returns a cached block or calls mmap() to provide memory to user/process
- Previous Scheme: First-Fit
  - Returns first available cached block (leading to potentially large amounts of fragmentation)
- Goal: Reduce fragmentation (Best-Fit)
  - Return the smallest block that satisfies the requested size
- Using ADB (Android Debug Bridge) tools and testing we were able to compare results and decide which direction to go

First-Fit

malloc(75KB);

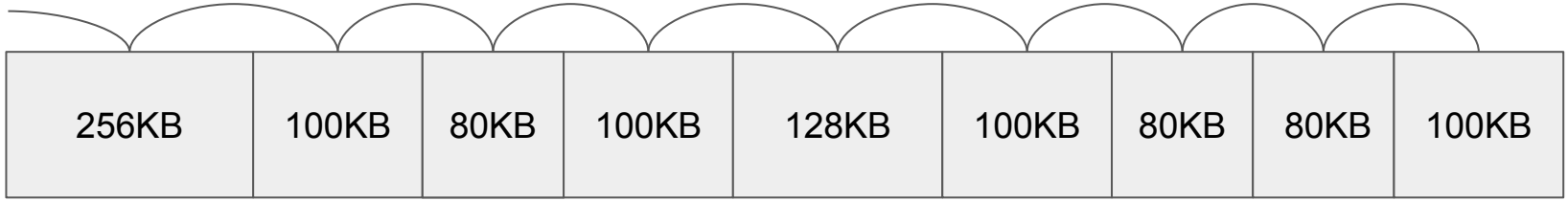
181KB Fragmented



Best-Fit

malloc(75KB);

5KB Fragmented



Best-Fit

- Caused a slowdown but clearly improved fragmentation

```
Stats: MapAllocator: allocated 10829 times (6003792K), freed 10829 times (6003792K), remains 0 (0K) max 36M
Stats: MapAllocatorCache: EntriesCount: 32, MaxEntriesCount: 32, MaxEntrySize: 2097152
Stats: RetrievalStats: Wasted: 42533K, SuccessRate: 10183/10441 (97.53%)
```

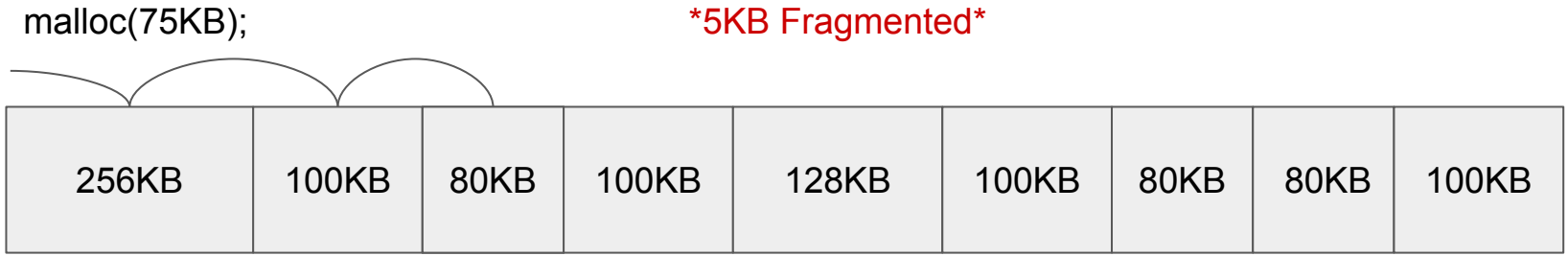
```
-- Average Operation Time -- -- Name (# of Calls) --
            83.8(ns)            Optimal-Fit Retrieve (10441)
```

Example output of adb logcat for a camera memory-trace

- Nano-seconds is still plenty fast for most of scudo’s purposes
- Best-Fit has a large chance to traverse through all the cached-blocks for each allocation

Compromise: Optimal-Fit

- Slight slowdown from returning the first available block
- Reduced Fragmentation from setting an upper-bound on allowed Fragmentation
- Upper-Bound set to: **10%** of requested size
- Immediate return if Fragmented Bytes <= the upper-bound
- Upper-Bound value could potentially be easily changed to accommodate for different circumstances/configurations



Conclusion

- We are now reducing Fragmentation and have some insight about how the environment of the Secondary allocator is behaving.
  - Less calls of mmap()
- Testing revealed the variation of processes and devices memory needs
- Configurable Upper-Bound fragmentation would accommodate different needs
- Using stats dumping would also help users realize their own memory needs

Example output of getStats():

```
scudo   : Stats: MapAllocator: allocated 2142 times (313084K), freed 2078 times (298100K), remains 64 (14984K) max 9M, Fragmented 512K
scudo   : Stats: MapAllocatorCache: EntriesCount: 21, MaxEntriesCount: 32, MaxEntrySize: 2097152
scudo   : Stats: CacheRetrievalStats: SuccessRate: 2002/2123 (94.30%)
scudo   : StartBlockAddress: 0x7f6a67c000, EndBlockAddress: 0x7f6a68d000, BlockSize: 69632
scudo   : StartBlockAddress: 0x7ac47d2000, EndBlockAddress: 0x7ac48d3000, BlockSize: 1052672 [R]
scudo   : StartBlockAddress: 0x7ca75e9000, EndBlockAddress: 0x7ca75ff000, BlockSize: 90112
scudo   : StartBlockAddress: 0x7ac4ce1000, EndBlockAddress: 0x7ac4de2000, BlockSize: 1052672 [R]
scudo   : StartBlockAddress: 0x7ac49d8000, EndBlockAddress: 0x7ac4ad9000, BlockSize: 1052672 [R]
scudo   : StartBlockAddress: 0x7ac4adb000, EndBlockAddress: 0x7ac4bdc000, BlockSize: 1052672 [R]
scudo   : StartBlockAddress: 0x7c944df000, EndBlockAddress: 0x7c9452e000, BlockSize: 323584 [R]
scudo   : StartBlockAddress: 0x7cad1ab000, EndBlockAddress: 0x7cad1e7000, BlockSize: 245760 [R]
scudo   : StartBlockAddress: 0x7ac4bde000, EndBlockAddress: 0x7ac4cdf000, BlockSize: 1052672 [R]
scudo   : StartBlockAddress: 0x7ac48d5000, EndBlockAddress: 0x7ac49d6000, BlockSize: 1052672 [R]
scudo   : StartBlockAddress: 0x7ac45cc000, EndBlockAddress: 0x7ac46cd000, BlockSize: 1052672 [R]
scudo   : StartBlockAddress: 0x7ac46cf000, EndBlockAddress: 0x7ac47d0000, BlockSize: 1052672 [R]
scudo   : StartBlockAddress: 0x7bb0591000, EndBlockAddress: 0x7bb05ff000, BlockSize: 450560 [R]
scudo   : StartBlockAddress: 0x7f6daa5000, EndBlockAddress: 0x7f6dac4000, BlockSize: 126976 [R]
scudo   : StartBlockAddress: 0x7ca4026000, EndBlockAddress: 0x7ca4054000, BlockSize: 188416 [R]
scudo   : StartBlockAddress: 0x7cad3ea000, EndBlockAddress: 0x7cad402000, BlockSize: 98304
scudo   : StartBlockAddress: 0x7b7c308000, EndBlockAddress: 0x7b7c396000, BlockSize: 581632 [R]
scudo   : StartBlockAddress: 0x7c9cc06000, EndBlockAddress: 0x7c9cc4b000, BlockSize: 282624 [R]
scudo   : StartBlockAddress: 0x7f6dbfc000, EndBlockAddress: 0x7f6dc1b000, BlockSize: 126976 [R]
scudo   : StartBlockAddress: 0x7c96461000, EndBlockAddress: 0x7c964a2000, BlockSize: 266240 [R]
scudo   : StartBlockAddress: 0x7c9fe0c000, EndBlockAddress: 0x7c9fe23000, BlockSize: 94208 [R]
scudo   : Stats: Quarantine: batches: 0; bytes: 0 (user: 0); chunks: 0 (capacity: 0); 0% chunks used; 0% memory overhead
scudo   : Quarantine limits: global: 0K; thread local: 0K
```

Thank You!

Acknowledgements:

Chia-Hung Duan
Christopher Ferris

References:
https://reviews.llvm.org/D157155
https://android-review.googlesource.com/c/platform/external/scudo/+/2687661
https://source.android.com/docs/security/test/scudo
https://llvm.org/docs/ScudoHardenedAllocator.html

frsalas513@gmail.com