Why You Should Use Scudo

Christopher Ferris, Chia-Hung Duan

Outline

- History
- The design of Scudo
- Security features
- Performance and memory footprint
- Future works
- How to start getting involved

Scudo History

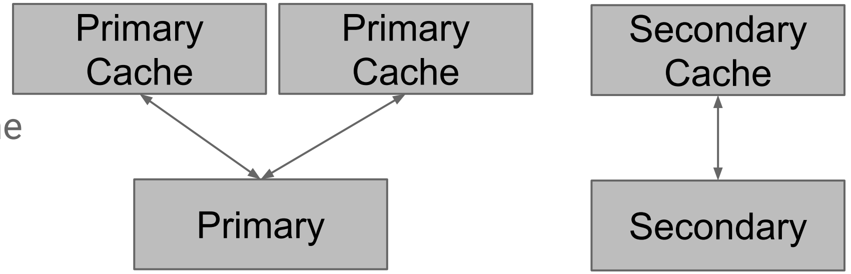
- Hardened Memory Allocator
  - Named after the Italian word for shield (escudo)
- Based on LLVM Sanitizer Allocator

Scudo History Cont.

- Refactored into a standalone version
- Adopted by Android
  - First introduced in Android 11 (released 2020) for Google Pixel devices
- Recent investments in performance/memory consumption improvements

Scudo Design

- Primary allocator is a size-class table based allocator
  - Region manages blocks with the same size
    - Implemented as bump-pointer allocator
- Secondary Allocator is a mmap-based allocator
  - Each allocation is fulfilled by system calls if there's nothing available in the cache
- Cache System
  - Exclusive Cache - per thread cache
  - Shared Cache
    - A cache pool shared between threads
  - Secondary only supports single central cache



Google

Scudo Design Cont.

- Proactive page releasing
  - Instead of calling mallopt with M_PURGE, Scudo supports releasing the pages with heuristics
- Supporting mallopt with M_LOG_STATS to dump allocator status
  - Memory usage in each region, the fragmentation info, cache hit rate.etc

Security Features

- Pointer Randomization
  - Unlikely to have two consecutive pointers with adjacent addresses
- Pointer Quarantine
  - A pointer won’t be reused in a short period of time
- Double-Free detection
- Out-of-Bound access detection
  - In primary, write over the header will cause the checksum corruption on free()
  - In secondary, touching the guard pages will trigger the segmentation fault
- Memory Tagging Extension (MTE) support
  - Armv9 introduced the Arm Memory Tagging Extension (MTE), a hardware extension that allows you to catch several kinds of memory bugs in your native code

Performance and Memory Footprint

- On Android, Geekbench score difference is almost in par with jemalloc
  - < 1% difference
- Due to the pointer randomization, it was consuming more memory at the beginning. Now the memory footprint has been greatly reduced and it uses less memory than jemalloc in many cases
  - E.g., A top third party game consumes ~3.2% less memory in average

Future Works

- Support more memory usage data
    - Page utilization
- Support better size-class table recommendation system
    - Move from default configuration to your custom one

Steps to Get Involved

- Developing on Android
  - `dumpsys meminfo --logstats` will help you understand the memory usage of your program
  - Pixel 8 & 9 series support MTE in developer mode
- Use while developing or testing
  - To hint some common memory bugs earlier
  - On Android, it’s default memory allocator in bionic libc
  - On non-Android platform, If you’re using llvm libc, then you got it
- Start your own Scudo configuration
  - See “allocator_config.def” for all the allocator parameters
  - Create your “custom_scudo_config.h” and build it with your Scudo source

Thanks

- Questions and feedbacks are welcomed
  - cferris@google.com, github.com/cferris1000
  - chiahungduan@google.com, github.com/ChiaHungDuan

Google