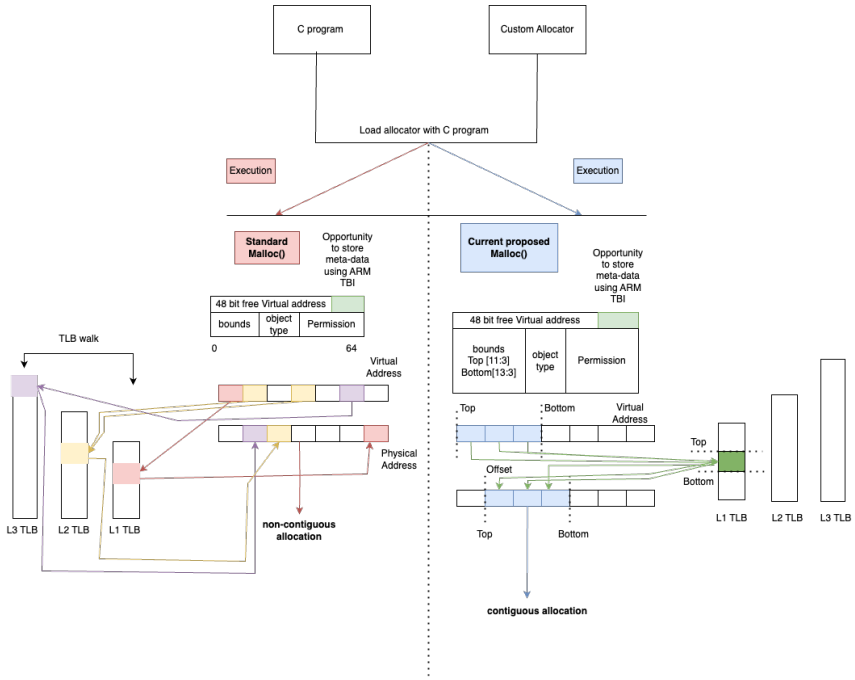
Contents

Future work

This documents is decision making to highlight potential paths to take for this PhD. We will initially talk about the current expirement which is a FAT pointer based memory allocator and will then expand into 2 potential paths:

- Cheri RISCV to prevent using the TLB.
- Allocator evaluation based on stripping instruction calls for larger allocators like Jemalloc.

1. Current expirement: FAT pointer based range addresses



The objective of this expirement was to ensure we can use the CHERI bounds as tracking mechanism of allocations instead of using multiple TLB entries. Using this approach we can use a single Huge page entry with bounds to ensure that the bounds (Which is the top and base address) can be extracted from the pointer using the Cheri compressed bounds mechanism.

We implemented a simple allocator which uses this technique with a basic malloc and free.

Objectives

- How does the utilization of bounds for tracking memory allocations, in addition to security purposes, affect the run times and Translation Lookaside Buffer (TLB) miss rates in modern computing systems ?
- How does the implementation of bounds for seeking through physically contiguous memory influence the complexity and efficiency of standard memory allocators, particularly those with advanced features such as transparent huge pages, and what are the implications for system performance in terms of execution speed, memory access latency, and resource utilization?

Hardware

- ARM morello (Huge page size 1GB used)

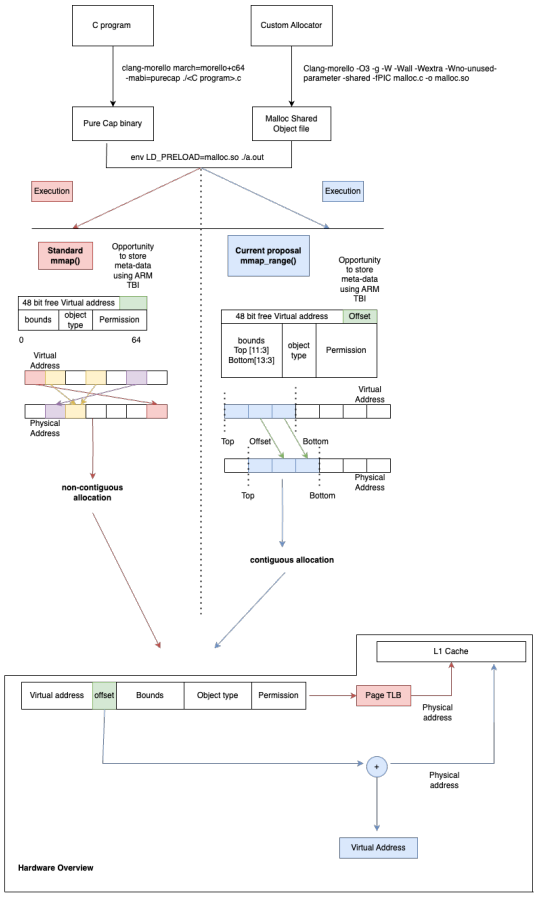
Evaluation

We conducted tests of the FAT Pointer-based range addresses against Jemalloc, the default memory allocator for CHERIBSD, to assess the performance improvements enabled by a CHERI-based huge page-aware alocator. Specifically, we evaluated the reduction in TLB misses and its impact on overall performance metrics, such as wall clock runtime. To comprehensively analyze the proposed allocator, we categorized benchmarks into two classes which are micro and macro benchmarks. Micro benchmarks comprise smaller C programs designed to target specific allocator patterns, enabling us to evaluate detailed aspects of the allocators behavior. Macro benchmarks, on the other hand, encompass larger, realworld C programs, allowing us to assess the allocators performance in more practical, real-world scenarios.

limitation

- Using Huge page still requires a TLB entry which could be mitigated (Refer to the FPGA work).
- ARMv8 only supports using to virtual addresses so it's required to bypass the TLB for address translation.

2. Cheri RISCV to prevent using the TLB

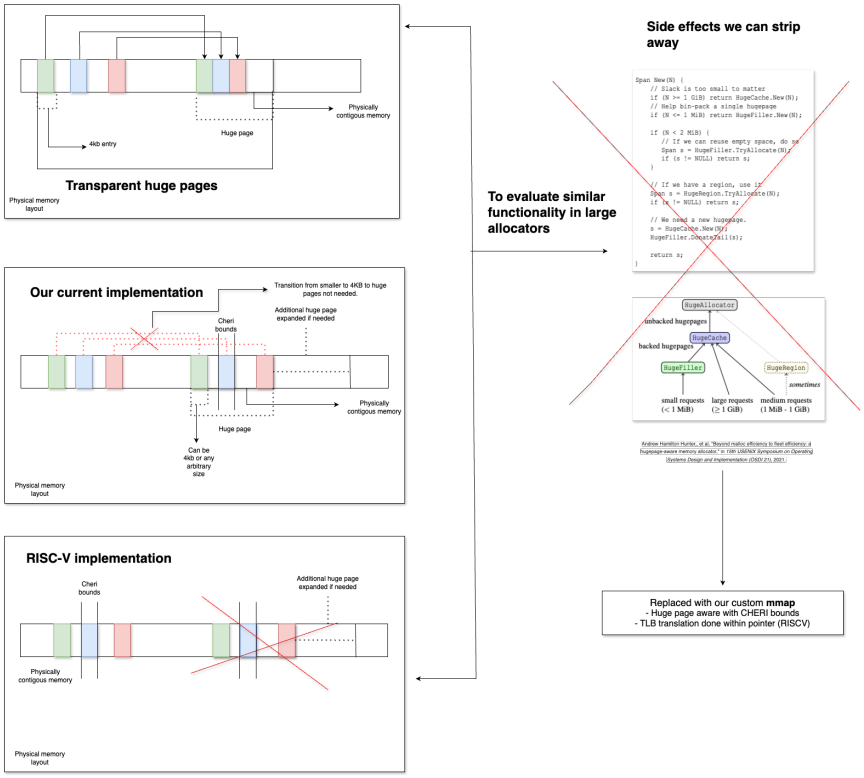


In the current ARM Morello setup, address translations rely on the TLB. The future approach on RISC-V Tooba involves storing the offset directly within the pointer. This is possible due to CHERIs capability model, which supports fine-grained memory protection and can encode bounds within pointers. Utilizing Bounds in CHERI for Block-Based Allocation: CHERI capabilities allow pointers to carry metadata about memory bounds, providing hardware-enforced memory safety. By encoding the offset and bounds within the pointer, the system can directly access memory without needing intermediate translations via the TLB. This enables the implementation of a block-based allocator that can efficiently manage memory allocations and deallocations within defined bounds. Bypassing the TLB in RISC-V Tooba.

Hardware modifications

The Bluespec design of the RISC-V processor will be modified to allow certain memory operations to bypass the TLB. This means that when a pointer with encoded offset and bounds is used, the system can directly compute the physical address from the capability information. This modification reduces the dependency on the TLB, decreasing latency. and improving performance, especially for frequent memory operations.

3. Allocator evaluation based on stripping instruction calls for larger allocators



Box 1

The diagram above mentions 3 particular implementations. The first box which is the standard THP(Transparent huge pages) utilised by modern allocators. THP initially emphasises on doing smalled allocations and as the number of allocations grows uses a technique which groups all smaller

allocations together and when done converts them into a large page of size 4mb in allocators such as jemalloc.

This approach does incur addtional operations such as grouping smaller allocations chaging the TLB entries (Adding more oppurtunity for TLB misses). Only once the huge page is created the TLB misses are reduced.

Box 2

Box 2 which refers to our current implementation always pre-allocates huge pages and untilises CHERI bounds to track each allocation inside the huge page. Allowing a single entry with the combination of bounds to provide block based behavoir in physically contigous memory while ensuring a pointer can only access a regoin within it's defined bounds.

Another aspect to note is that the bounds can be of a dynamic size when defined. This is in contrast to defining multiple page entries which need to be fixed sizes which means they always incur multiple entries. In the current approach when the huge page size is hit a new one is created. The limitaton of this is appraoch being we are limited to the huge page set by the processor implementation (In our case the CHERI ARM v8.1).

Box 3

The 3rd box specifies an alternate appraoch by not using huge pages and required memory is not required to be physically contigous. In this approach the pointer stores all the metadata to the translation from virtual to physical addresses.

Building up from the work of Box 2 and Box 3

Box 2 and 3 from a high overview there is only minor difference which can be noted which is 1 uses huge pages and other does not. Both approaches can strip down the number intructions needed in modern allocators (Stripping away the need transitioning from smaller to larger pages). This document is yet to give an exact breakdown.

As seen to the right of the diagram is a sample snippet of TC malloc from the paper (Beyond malloc efficiency to fleet allocators). This whole span function would not be required in our approach. The other benefit being easier get the approach by getting mmap embedded inside the allocator.

Evaluation:

- Amount of instructions that can be stripped away from the page aware memory allocator.
- Comparing memory allocator with wall clock run time with the modified mmap and without the modified mmap.
- CHERI purecap does incur additional instruction such as bound checks. Does this approach as a whole reduce the number of instructions as whole (Comparing CHERIpurecap instructions with memory allocator emitted vs regular ARMv8 clang program with the same allocator).