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].

1. Current expirement: FAT pointer based range addresses

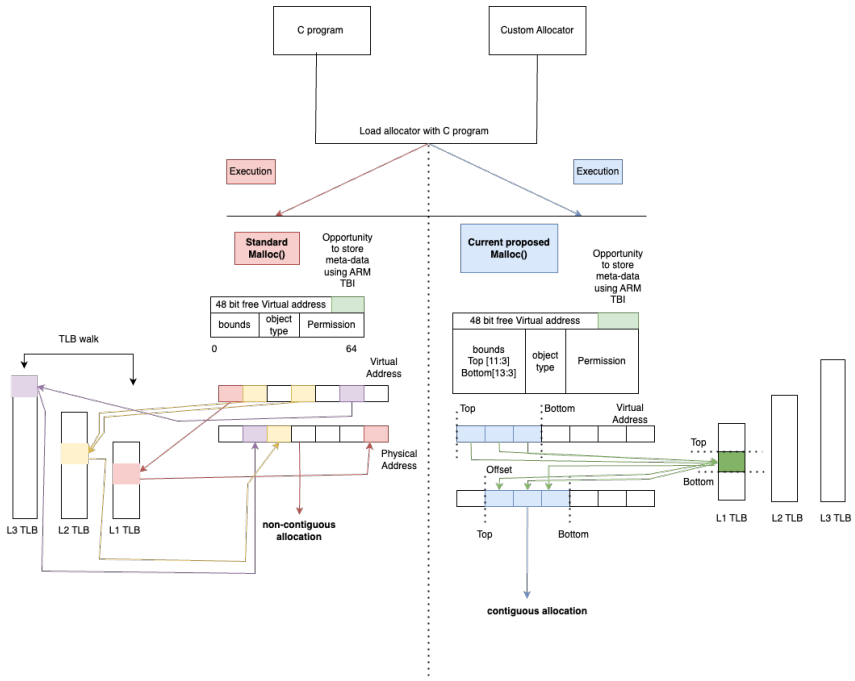

Figure 1: FAT pointer implementation with Huge pages against a standard malloc allocator.

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[2] 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[3] 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[4] (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

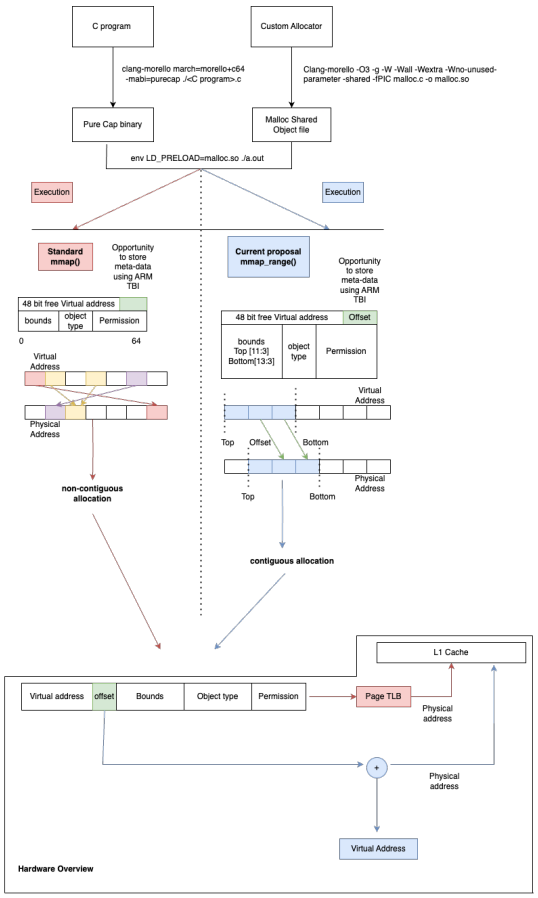

Figure 2: FAT pointer implementation with RISCV CHERI Toooba to strip the requirement of requiring a TLB.

In the current ARM Morello setup, address translations rely on the TLB. The future approach on RISC-V Tooba[5] 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, provid-

ing 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 (Transparent huge pages)

The diagram 3 highlights three specific implementations, the first of which is the standard Transparent Huge Pages (THP) mechanism employed by modern memory allocators. THP initially focuses on handling smaller memory allocations. As the volume of allocations increases, it employs a strategy that consolidates these smaller allocations into contiguous memory regions. Once aggregated, these regions are subsequently converted into larger memory pages, typically of size 4MB, as seen in allocators like jemalloc. This approach optimizes memory management by reducing fragmentation and improving performance through the use of larger page sizes.

This approach, however, introduces additional overhead, including the operations required to consolidate smaller allocations and modify Translation Lookaside Buffer (TLB) entries. These modifications can initially increase the likelihood of TLB misses, as the process of grouping and reorganizing memory allocations temporarily disrupts the efficiency of TLB utilization. It is only after the successful creation of the huge page that the benefits materialize, leading to a reduction in TLB misses due to the improved alignment of memory access patterns with the larger page size.

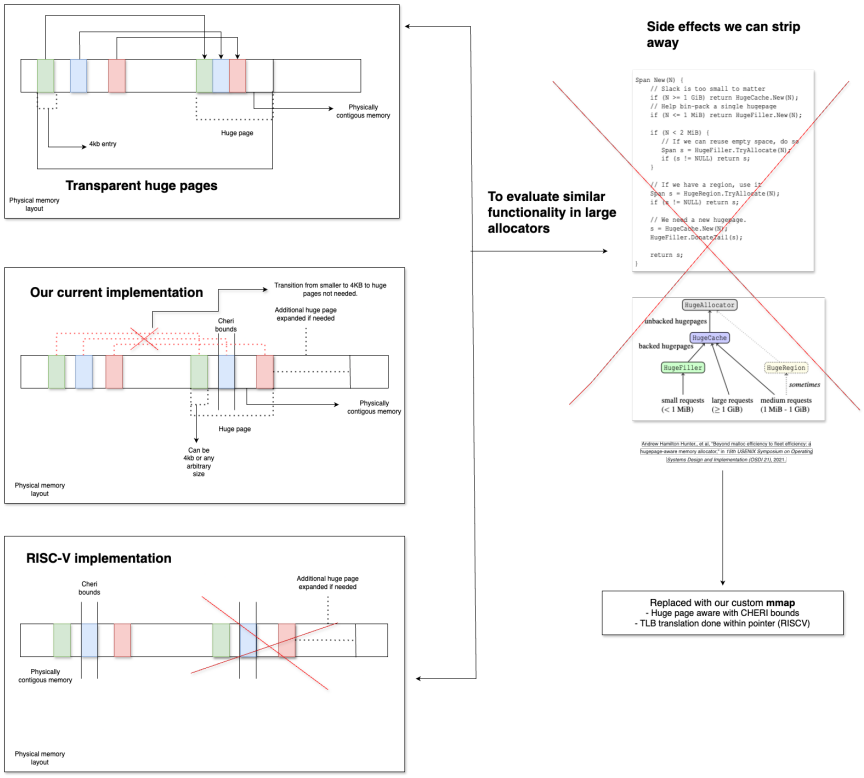

Figure 3: Deprecating the use of THP with CHERI bound aware embedded mmap.

Box 2 (Our current implementation)

Box 2 outlines the current implementation, which involves the pre-allocation of huge pages and leverages CHERI (Capability Hardware Enhanced RISC Instructions) bounds to meticulously track each allocation within these pages. This approach enables a single TLB entry, combined with the precise bounds defined by CHERI capabilities, to facilitate block-based memory management within physically contiguous regions. By enforcing strict bounds on pointers, the system ensures that each pointer can only access memory within its explicitly defined region.

Another critical aspect of this implementation is the ability to define bounds of dynamic sizes, which stands in contrast to traditional approaches that rely on fixed-size page entries. Fixed-size entries inherently require multiple TLB entries, regardless of the actual memory usage, leading to inefficiencies. In the current approach, when the allocated memory within a huge page reaches its capacity, a new huge page is allocated. However, a notable limitation of this method is its dependence on the maximum huge page size supported by the underlying processor architecture. In this case, the system is constrained by the huge page size defined by the CHERI-enhanced ARM v8.1[6] architecture. While this approach offers flexibility in memory allocation and reduces the need for multiple TLB entries, it is ultimately bounded by the hardware's architectural specifications.

Box 3 (RISC-V implementation)

The third approach, as outlined in Box 3, deviates from the use of huge pages and does not require memory to be physically contiguous. In this model, each pointer is designed to store comprehensive metadata at the pointer necessary for the translation from virtual to physical addresses. This metadata enables the system to manage memory allocations without the constraints of physical contiguity, thereby offering greater flexibility in memory utilization. By embedding translation information directly within the pointers, this approach eliminates the need for large, contiguous memory regions and allows for more granular and dynamic memory management.

Building up from the work of Box 2 and Box 3 (Side effects we can strip away)

From a high-level perspective, the primary distinction between Box 2 and Box 3 lies in the use of huge pages in the former and their absence in the

latter. Both approaches share the common advantage of reducing the number of instructions required in modern memory allocators by eliminating the need for transitioning between smaller and larger pages. This simplification streamlines memory management processes. However, this document has not yet provided a detailed breakdown or quantitative analysis of the specific performance implications or trade-offs.

As illustrated to the right of the diagram, a sample snippet of TC malloc from the paper Beyond malloc Efficiency to Fleet Efficiency is provided. In the proposed approach, the entire span function, which is essential in TC malloc, would become unnecessary. Additionally, the approach offers the advantage of simplifying memory management by integrating mmap directly within the allocator. This integration eliminates the need for separate mechanisms to handle memory mapping.

Evaluation:

- The number of instructions that can be eliminated from a page-aware memory allocator by adopting the proposed approach.
- A comparative analysis of the memory allocator's performance using wall-clock runtime measurements, both with and without the modified mmap implementation.
- While CHERI Purecap introduces additional instructions, such as bounds checks, the overall approach aims to determine whether it reduces the total number of instructions when compared to a traditional ARMv8 Clang program using the same allocator. This involves evaluating the trade-offs between the overhead of CHERI-specific instructions and the potential reductions in allocator-emitted instructions.

Current tasks on hand

1. To port over all chapters written to a EuroSys Conferenece document and send over the to the supervisory team for a review.
    - This will then get send over to Jeremy for an external review.
    - Expands to a talk in the University of Glasgow seminar.
    - Concurrent suggestions of the Eurosys submission till may.
2. Work on allocator evaluation based on stripping instruction calls for larger allocators (Expirement agreed in the meeting).

3. Write to Cambridge a proposal for using the RISC-V expirement (2 weeks).

References

[1] "JEMALLOC." [Online]. Available: https://jemalloc.net/jemalloc.3.html

[2] J. Navarro, "Practical, transparent operating system support for superpages."

[3] J. Woodruff, A. Joannou, H. Xia, A. Fox, R. M. Norton, D. Chisnall, B. Davis, K. Gudka, N. W. Filardo, A. T. Markettos, M. Roe, P. G. Neumann, R. N. M. Watson, and S. W. Moore, "CHERI Concentrate: Practical Compressed Capabilities," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1455–1469, Oct. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8703061/

[4] "Department of Computer Science and Technology CHERI: The Arm Morello Board." [Online]. Available: https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/cheri-morello.html

[5] "CTSRD-CHERI/Toooba: RISC-V Core; superscalar, out-of-order, multi-core capable; based on RISCY-OOO from MIT." [Online]. Available: https://github.com/CTSRD-CHERI/Toooba

[6] "Arm Architecture Reference Manual for A-profile architecture." [Online]. Available: https://developer.arm.com/documentation/ddi0487/latest