Contents

1 Introduction

In computing, achieving high performance is an ongoing challenge, especially as applications handle increasingly complex workloads. Memory management is a key factor in performance, where efficient use of resources is essential. Translation Lookaside Buffers (TLBs) are crucial in this context, speeding up memory access by caching recent memory address translations. A TLB, a specialised cache in the memory management unit (MMU), reduces the time required to convert virtual addresses to physical ones. When a program accesses data in memory, the MMU first checks the TLB for a matching entry, avoiding the slower process of consulting page tables. However, as applications grow larger and more complex, the fixed size of TLBs often cannot keep up, leading to more TLB misses and performance slowdowns[1]. To tackle this issue, researchers have explored new solutions, including the use of huge pages[2].

Huge pages, also known as large pages, allow for the allocation of memory in significantly larger chunks compared to traditional small pages. By reducing the number of TLB entries needed to access a given amount of memory, Huge pages offer a potential avenue for optimising TLB utilisation by reducing the number of entries needed to map large memory regions. This not only decreases the frequency of TLB misses but also lowers the overhead associated with address translation. By minimising these bottlenecks, huge pages can improve system performance in several ways, such as speeding up memory-intensive applications, reducing latency in data access, and enhancing throughput for workloads that rely heavily on large datasets.

Simultaneously, advancements in hardware-level security, such as the Capability Hardware Enhanced RISC Instructions (CHERI) [3] architecture, present additional opportunities for performance enhancement. CHERI's capability-based addressing approach not only strengthens system security by tightly controlling memory access but also opens avenues for optimising memory management operations. By integrating CHERIs compressed[4] encoded bounds with the use of huge pages, it becomes possible to track and manage large, physically contiguous memory blocks more efficiently. This combination reduces TLB pressure by minimising the number of entries required to map extensive memory regions, thereby decreasing TLB misses

and improving address translation performance. Furthermore, it accelerates memory-intensive tasks by reducing the overhead associated with managing fragmented or non-contiguous memory allocations. The contributions for the following paper are as follows:

- **Fat-pointer Based Range Addresses**: Introduces fat-pointers that include memory bounds, allowing efficient tracking and management of physically contiguous memory regions.
- **Custom Memory Allocation with Huge Pages**: Proposes a custom ‘mmap‘ function and kernel module for allocating huge pages of physically contiguous memory, reducing the need for traditional TLB entries and improving efficiency.
- **Novel Memory Allocation Algorithms**: Provides new algorithms for allocating and freeing physically contiguous memory, integrating huge pages with CHERIs capability-based bounds for enhanced memory management.
- **CHERIs Capability-based Optimization**: Demonstrates how CHERI's architecture can be used to optimize memory allocation by encoding memory bounds directly within pointers, reducing TLB reliance.

Through comprehensive evaluation, including micro and macro benchmarks, we demonstrate the allocators ability to reduce TLB misses by up to 90%, yielding significant improvements in wall clock runtimes for memory-intensive applications. While its impact on larger, computation-heavy workloads is less pronounced, the proposed allocator shows strong potential for advancing memory management in scenarios requiring high memory throughput and low translation overhead. The following below are research questions we are addressing:

1. 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?
2. 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?

References

[1] S. Mittal, "A survey of techniques for architecting TLBs," vol. 29, no. 10, p. e4061, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4061. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4061

[2] A. Panwar, S. Bansal, and K. Gopinath, "HawkEye: Efficient fine-grained OS support for huge pages," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, pp. 347–360. [Online]. Available: https://dl.acm.org/doi/10.1145/3297858.3304064

[3] J. Woodruff, R. N. Watson, D. Chisnall, S. W. Moore, J. Anderson, B. Davis, B. Laurie, P. G. Neumann, R. Norton, and M. Roe, "The CHERI capability model: revisiting RISC in an age of risk," vol. 42, no. 3, pp. 457–468. [Online]. Available: https://doi.org/10.1145/2678373.2665740

[4] 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," vol. 68, no. 10, pp. 1455–1469. [Online]. Available: https://ieeexplore.ieee.org/document/8703061/