Towards Speculative Offset-Based Address Translation

Chloe Alverti, Vasileios Karakostas
National Technical University of Athens

Research problem. Virtual Memory (VM) is a fundamental abstraction of computing systems. Architectural support primarily relies on Translation Lookaside Buffers (TLBs) to hide the address translation overhead. In modern systems, though, TLBs often fail to cover the large working sets of data-intensive workloads, exposing VM as a potential performance bottleneck.

Motivation. Prior research [3] introduced range translations to address this challenge, by extending traditional pages with ranges – unlimited contiguous virtual-to-physical mappings of unified access rights. Memory preallocation (eager paging) was proposed to increase contiguity, and range TLB was introduced to accelerate translation. However, eager paging alone is insufficient to extract the available physical contiguity and is sensitive to fragmentation. In addition, memory management techniques [4] can gradually break ranges. Such scenarios may lead to thrashing range TLB performance.

Our contributions. (i) We propose a simple memory management extension to boost contiguous memory allocation and relax the need for preallocation, and (ii) we explore the potential of speculative address translation based on offsets to sustain high translation performance.

Contiguity-aware Paging. Linux's core buddy allocator maintains lists of free blocks, where a list of order-*i* contains blocks of 2^i contiguous pages, and attempts to coalesce aligned buddy blocks to keep higher-order lists populated. In [3], physical memory is preallocated at request time (eager paging), instead of access time (demand paging), and contiguity is extracted by allocating blocks from the highest possible order.

We propose enhancing the system allocator with contiguity awareness (CA). When a physical block is allocated to map a virtual range, the system tracks and stores the *offset* of the mapping (v_{addr}-p_{addr}). In a future allocation request, the allocator tries to use a physical block that complies with the tracked offset in a best-effort basis, instead of picking a random one. Counter-intuitively, CA has minimal performance overhead, because the availability status of a block is easily retrieved by indexing the VMM's mem_map array structure with the block's physical address. The proposed memory management extension: (i) enables range construction across allocation requests, (ii) is more robust when fragmentation and high order allocation failures occur, (iii) overcomes alignment restrictions, and (iv) is orthogonal and independent to preallocation.

We implement CA in Linux 3.15.5 and our results show that: (i) CA with eager paging boosts range creation and sustains RMM [3] performance to 99% (TLB miss reduction) when external fragmentation exists, and (ii) CA with demand paging maintains similar performance (10% more TLB misses compared to the previous configuration), while eliminating the inefficiencies of preallocation, such as memory bloat.

Speculative Address Translation. Still, discontinuities in ranges may appear, either at creation time due to external fragmentation, or later due to memory management techniques, such as copy-on-write, working set sampling, and memory deduplication (in virtualized systems) [4]. These scenarios may lead to multiple ranges, that are non-contiguous but do share the same offset. As the number of ranges increases, a larger range TLB might be necessary to achieve high hit ratio. However, simply increasing the range TLB size is not a sustainable option due to its inherent logic that requires (i) fully-associative lookups, and (ii) range comparisons.

We explore the idea of speculating the address translation, while performing verification page table walks in the background. Our proposal is based on tracking *offsets*, decoupled from virtual range boundaries, and speculatively selecting one to perform virtual-to-physical translation. Since only a few program instructions are usually responsible for most TLB misses, we index the speculation structures by PC. Previous work has considered speculative translation but at 2MB page granularity [1, 4] or in the presence of identity mappings (v_{addr}==p_{addr}) [2], while recent work [5] proposed a similar PC-based mechanism but for partial translation. We envision speculation as a lightweight auxiliary translation mechanism, placed in the miss-path of a range TLB in RMM [3]. To examine the idea's potential, we evaluate the challenging scenario of replacing the range TLB with speculation. Interestingly, our preliminary results indicate that a simple speculation mechanism, tracking one offset per PC, predicts translations successfully for 75%-99% of the TLB misses at optimal memory management conditions (Spec2006 and PARSEC).

References

[1] Thomas Barr, Alan Cox, and Scott Rixner. SpecTLB: A mechanism for speculative address translation. In *ISCA 2011*.
[2] Swapnil Haria, Mark D. Hill, and Michael M. Swift. Devirtualizing memory in heterogeneous systems. In *ASPLOS 2018*.
[3] Vasileios Karakostas, Jayneel Gandhi, and et al. Redundant memory mappings for fast access to large memories. In *ISCA 2015*.
[4] Binh Pham and et al. Large pages and lightweight memory management in virtualized environments: Can you have it both ways? In *MICRO 2015*.
[5] Tianhao Zheng, Haishan Zhu, and Mattan Erez. SIPT: Speculatively Indexed, Physically Tagged Caches. In *HPCA 2018*.