Virtual memory is a powerful and ubiquitous abstraction for managing memory. However, virtual memory suffers a performance penalty for these benefits, namely when translating program virtual addresses to system physical addresses. This overhead had been limited to 5-15% of system runtime by using a set of sophisticated hardware solutions, but has increased to 20-50% for many scenarios, including running workloads with large memory footprints and poor access locality or using deeper software stacks.

My thesis aims to solve this problem so that the memory systems can continue to scale without being hamstrung by the virtual memory system. We observe that while operating systems (OS) and hypervisors have a rich set of components in allocating memory, the hardware address translation unit only maintains a rigid and limited view of this ecosystem. Therefore, we seek for patterns inherently present in the memory allocation mechanisms to guide us in designing a more intelligent address translation unit.

First, we realize that OS memory allocators and program faulting sequence tend to produce contiguous or nearby mappings between virtual and physical pages. We propose Coalesced TLB and Clustered TLB designs to exploit these patterns accordingly. Once detected, the related mappings are stored in a single TLB entry to increase the TLB reach. Our designs help reduce TLB misses substantially and improve performance as a result.

Second, we see that there are often tradeoffs between reducing address translation overhead and improving resource consolidation in virtualized environments. For example, large pages are often used to mitigate the high cost of two-dimensional page walks, but hypervisors usually break large pages into small pages for easier sharing guests memory. When that happens, the majority of those small pages still remain aligned. Based on this observation, we propose a speculative TLB technique to regain almost all performance loss
caused by breaking large pages while running highly consolidated virtualized systems.