Abstract

Since its inception, virtual memory has become a powerful and ubiquitous abstraction for allocating and managing memory with a flexible and clean programming model. Typically, the systems community has been comfortable paying a performance tax for these programmability benefits. Unfortunately, emerging software with large data requirements and deeper stacks (e.g., large graphs, key value stores, virtualization), and emerging hardware accelerators requiring manual data orchestration by the CPU are increasing this performance tax drastically, while also conceding various programmability benefits of virtual memory.

In this talk, I discuss techniques to reclaim this lost performance and programmability by enriching existing address translation hardware to more elasticity adapt to memory allocation aspects of the operating system. Specifically, I show how hardware support that detects patterns in page table allocation can be used to design low-overhead, high performance address translation hardware. In addition, I discuss how to design memory management units for accelerators in support of unified address spaces. Overall, these techniques are broadly applicable across both server and client systems.