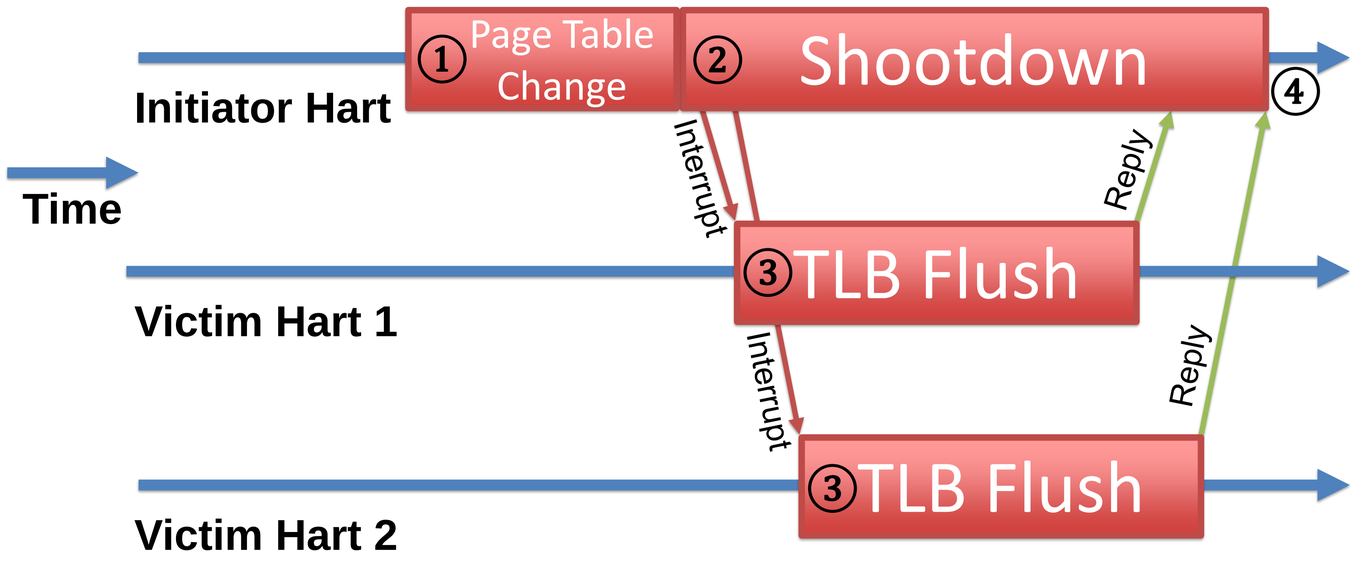
Improving RISC-V TLB Shootdown Performance

John Henry Deppe (deppe@ece.ubc.ca) and Guy Lemieux (lemieux@ece.ubc.ca)

What Is TLB Shootdown?

Today's address translation caches, also known as "translation lookaside buffers" or TLBs, are not coherent like CPU data caches are. When page table entries (PTEs) containing active translations are changed to reduce access, **TLBs must be invalidated by the operating system** to remove cached and potentially dangerous stale translations.



Above is an example of the shootdown process. An initiator hart ① changes the page table. Then the initiator ② sends inter-processor interrupts to all harts that may be caching the now-stale PTEs. The interrupted harts ③ flush their own stale cached translations and reply. After the initiator hart ④ collects all replies, it can continue with other tasks.

TLB Shootdown Works. Why Fix It?

Interrupt-based TLB shootdown flushes pipelines, pollutes caches, and disturbs application execution. **TLB invalidation is a scaling problem** as hart count increases in shared virtual memory systems.

Other architectures have adopted non-interrupting methods for remote TLB invalidation to improve performance. AMD's INVLPGB instruction broadcasts a non-interrupting TLB invalidation to all harts and is now used in Linux. ARM's similar TLBI instruction had performance problems in important early implementations, causing slow Linux adoption. Intel's Remote Action Request (RAR) approach is flexible, but its complex interface is not yet used in upstream Linux.

Even non-interrupting TLB invalidation broadcasts still disturb execution with synchronization, interconnect traffic, and cache probes. In Linux, some ARM platforms perform better with interrupting shootdowns because the kernel can filter interrupts but cannot filter broadcast invalidations.

What Should RISC-V Do?

Today's RISC-V does not have remote TLB invalidation in the instruction set; OpenSBI's remote TLB invalidation uses interrupts in its implementation. The Linux adoption pains of ARM and Intel's non-interrupting remote TLB invalidations show that uneven performance and complex interfaces will both confuse maintenance and slow adoption.

- Should RISC-V include remote TLB invalidation instructions?
  - Would such instructions broadcast to all harts or allow filtering?
  - Change address space identifiers from hart-private to system-wide?
- How do RISC-V remote TLB invalidation instructions collect replies?
  - Poll memory like Intel's RAR?
  - Synchronization instruction like RISC-V's Svinval extension or ARM?
- Could hardware page table walkers efficiently collect and expose translation sharing to enhance shootdown filtering in many-hart systems?

Our PTE Sharing Study

We modify Linux to track which harts actually cache a page table entry. This information will improve TLB shootdown filtering.

Software-Loaded TLB Tracks Shared PTEs

The T-Head XuanTie C920's TLBs can be manipulated with control and status registers (CSRs). We mark Linux page table entries as invalid and software-load correct translations with CSRs upon page fault. These faults let us maintain bitmasks that track which harts cache each translation.

Filter TLB Shootdowns With Tracked Sharing

Most architectures, including RISC-V, don't expose fine-grained information about translation caching. A syscall like `mremap()` requires shootdown interrupts for all harts currently running that process and deferred invalidation for harts that may still cache that process's translations.

Today, Linux filters TLB shootdowns with information from process scheduling; a stale translation won't be found in a hart that didn't access that page table. Some invalidations safely avoid interrupts by deferring until context-switch. Our PTE-tracking bitmasks enable additional interrupt avoidance: harts currently running the shootdown's process but not caching a stale translation.

Limit Overhead By Tracking Marked Areas

Extra page faults are performance overhead and extra bitmasks are memory overhead. To ameliorate these overheads, we track only memory designated by an `madvise()` syscall.

Future Work

Our shootdown filtering may provide net performance benefit to multi-threaded applications that suffer from TLB shootdown overhead, such as databases using `mmap()`.

We will also use collected translation sharing data in designing future translation coherence mechanisms.

THE UNIVERSITY OF BRITISH COLUMBIA

This work received funding from Andes Technology. We thank Steven Yeung for his help and support.