Don't shoot down TLB shootdowns!

Nadav Amit, Amy Tai, Michael Wei

April 2020
Translation Lookaside Buffer (TLB)

TLB = cache for virtual to physical address translations
TLB Coherency

Hardware does not maintain TLBs coherent
The problem is left for software (OS)
TLB Shootdown (in Linux)

- TLB flush
- send IPI
- continue
- TLB flush
- ack

initiator

responder

time
Challenge

TLB shootdowns are expensive.

How can we further optimize them?

This work focus on:
  • Linux/x86 – common lessons
  • Userspace mappings – common case

Lessons are relevant to other environments
Existing Solutions

**Hardware based TLB invalidations**

- Not available on all architectures
- Does not coexist (yet) with software techniques:
  - No selective target cores for TLB invalidation

**Software solutions**

- Replicating page-tables [RadixVM, Clements’13]
  - Can increase overhead with low-latency IPIs
- Aggressive batching [LATR, Kumar’18]
  - Breaks POSIX semantics
TLB Flushes in Linux and FreeBSD

- Initiator
  - TLB flush
  - send IPI

- Responder
  - busy-wait
  - continue

- TLB flush
- ack

Time
Optimization 1: Concurrent Flushes (forgotten lesson)

RP3 TLB consistency algorithm [Rosenburg’89]
TLB Shootdown Responder

Entry

enter IRQ handler

SMP

read SMP request

read shootdown request

read kernel space

flush user space

TLB

return to user

ack SMP request

Page Table Isolation
Optimization 2: Cacheline Consolidation

Entry
- enter IRQ handler

SMP
- read SMP request
- SMP info
- TLB flush info

TLB
- read shutdown request
- flush kernel space
- flush user space

memory
- ack SMP request

return to user
Optimization 3: Early Acknowledgment

**Entry**
- enter IRQ handler

**SMP**
- read SMP request
- ack SMP request

**TLB**
- read shutdown request
- flush kernel table
- flush user table

**Return to user**
Optimization 3: Early Acknowledgment

Entry:
- enter IRQ handler
- read SMP request
- read shutdown request
- ack SMP request

Safe: flush will happen
Better: Initiator is faster

flush kernel table
flush user table

return to user
Optimization 4: In-Context Flushes

Entry
- enter IRQ handler

SMP
- read SMP request
- ack SMP request

TLB
- read shutdown request
- flush kernel table
- flush user table

return to user
Optimization 4: In-Context Flushes

1. Efficient
2. Better batching
In the Paper

Userspace-safe batching
  • Deferring TLB shootdowns while the kernel runs

Avoiding TLB flushes on Copy-on-Write
  • Special case we can optimize

TLB flushes in virtualization
  • The effect of page size mismatch

Many important and subtle details
Evaluation: Unmapping and Flushing 10 PTEs

madvise(MADV_DONTNEED)

Initiator:
- base
- concurrent
- cacheline
- early-ack
- in-context

- same core
- same socket
- diff socket

Concurrent:
- cycles: 16208, 14361, 16475
- same core: 16208, 14361, 16475
- same socket: 7985, 6247, 6929
- diff socket: 7313, 5879, 6929

Cacheline:
- cycles: 8411, 6785, 8039
- same core: 8411, 6785, 8039
- same socket: 7313, 5879, 6929
- diff socket: 7313, 5879, 6929

Early-ack:
- cycles: 8411, 6785, 8039
- same core: 8411, 6785, 8039
- same socket: 7313, 5879, 6929
- diff socket: 7313, 5879, 6929

In-context:
- cycles: 8411, 6785, 8039
- same core: 8411, 6785, 8039
- same socket: 7313, 5879, 6929
- diff socket: 7313, 5879, 6929

Responder:
- concurrent
- cacheline
- early-ack
- in-context

- same core
- same socket
- diff socket

Concurrent:
- cycles: 8411, 6785, 8039
- same core: 8411, 6785, 8039
- same socket: 7313, 5879, 6929
- diff socket: 7313, 5879, 6929

Cacheline:
- cycles: 8411, 6785, 8039
- same core: 8411, 6785, 8039
- same socket: 7313, 5879, 6929
- diff socket: 7313, 5879, 6929

Early-ack:
- cycles: 8411, 6785, 8039
- same core: 8411, 6785, 8039
- same socket: 7313, 5879, 6929
- diff socket: 7313, 5879, 6929

In-context:
- cycles: 8411, 6785, 8039
- same core: 8411, 6785, 8039
- same socket: 7313, 5879, 6929
- diff socket: 7313, 5879, 6929
Evaluation: SysBench – Random Writes

Random writes

Periodic flushes

Memory-mapped file

Emulated persistent memory, no write-cache

Graph showing speedup vs. threads for different settings:
Conclusions

TLB shootdown can be improved

Doing it well in software ➔ better hardware interfaces

We are working to push these enhancements to Linux