Shoot4U: Using VMM Assists to Optimize TLB Operations on Preempted vCPUs

Jiannan Ouyang, John Lange, Haoqiang Zheng
University of Pittsburgh, VMware Inc.

VEE’16
04/02/2016
CPU Consolidation in the Cloud

**CPU Consolidation**: multiple virtual CPUs (vCPUs) share the same physical CPU (pCPU).

**Motivation**: Improve datacenter utilization.

Figure 1. Average activity distribution of a typical shared Google clusters including Online Services, each containing over 20,000 servers, over a period of 3 months [Barroso 13].
Problems with Preempted vCPUs

Performance problems:

**Busy-waiting** based kernel synchronization operations

- Lock *Holder* Preemption problem
- Lock *Waiter* Preemption problem
- *TLB Shootdown* Preemption problem
Lock Holder Preemption

Lock holder preemption [Uhlig 04, Friebel 08]
- A preempted vCPU is holding a spinlock
- Causes dramatically longer lock waiting time
  - context switch latency + CPU shares allocated to other vCPUs

Scheduling Techniques
- co-scheduling, relaxed co-scheduling [VMware 10]
- Adaptive co-scheduling [Weng HPDC11]
- Balanced scheduling [Sukwong EuroSys11]
- Demand-based coordinated scheduling [Kim ASPLOS13]

Hardware Assisted Techniques
- Intel Pause-Loop Exiting (PLE) [Riel 11]
Lock Waiter Preemption [Ouyang VEE13]

Linux uses a FIFO order fair spinlock, named *ticket spinlock*

![Diagram of lock waiting queue with timeout values](image)

- **Timeout:** 0, 1, 2, 3, T

**Lock waiter preemption**
- A lock waiter is preempted, and blocks the queue
- \( P(\text{waiter preemption}) > P(\text{holder preemption}) \)

**Preemptable Ticket Spinlock**
- **Key idea: proportional timeout**
**TLB Shootdown Preemption**

KVM Paravirt Remote Flush TLB [kvmtlb 12]

- VMM maintains vCPU preemption states and shares with the guest.
- Use conventional approach if the remote vCPU is running.
- Defefer TLB flush if the remote vCPU is preempted.
- **Cons:** preemption state may change after checking.

TLB shootdown IPIs as scheduling heuristics [Kim ASPLOS13]

**Shoot4U**

- **Goal:** eliminate the problem
- **Key idea:** invalidate guest TLB entries from the VMM
Contributions

- An **analysis** of the impact that various low level synchronization operations have on system benchmark performance.

- **Shoot4U**: A novel virtualized TLB architecture that ensures consistently low latencies for synchronized TLB operations.

- An **evaluation** of the performance benefits achieved by Shoot4U over current state-of-art software and hardware assisted approaches.
Performance Analysis
Overhead of CPU Consolidation

PARSEC Runtime with co-located VM over running alone
(12-core VMs, measured on Linux/KVM, with PLE disabled)
CPU Usage Profiling (perf)

![Graph showing CPU usage profiling with categories such as k:lock, k:tlb, k:other, and u:* for 1VM and 2VM configurations. The graph displays the percentage (%) for various benchmarks like blackscholes, bodytrack, canneal, dedup, ferret, freqmine, raytrace, streamcluster, swaptions, vips, and x264.](image)
CDF of TLB Shootdown Latency (ktap)
How TLB Shootdown Works in VMs

- TLB (Translation Lookaside Buffer)
  - a per-core hardware cache for page table translation results
- TLB coherence is managed by the OS
  - TLB shootdown operations: IPI + invlpg
- Linux TLB shootdown is busy-waiting based

1. vIPIs
2. trap
3. pIPIs
4. inject virtual interrupts
5. vCPU is scheduled (TLB Shootdown Preemption)
6. invalidation & ACK
Shoot4U
Observation: modern hardware allows the VMM to invalidate guest TLB entries (e.g. Intel `invpid`)

**Key idea: invalidate guest TLB entries from the VMM**
- Tell the VMM what TLB entries and vCPUs to invalidate (hypervall)
- The VMM invalidates and returns, no interrupt injection and waiting

1. hypercall
   `<vcpu set, addr range>`

2. pPIPs

3. invalidation and ACK
Implementation

Shoot4U API

kvm_hypercall3(unsigned long KVM_HC_SHOOT4U, unsigned long vcpu_bitmap, unsigned long start_addr, unsigned long end_addr);

KVM/Linux 3.16, ~200 LOC (~50 LOC guest side)
- https://github.com/ouyangjn/shoot4u

Guest
- use hypercall for TLB shootdowns

VMM
- hypercall handler: vCPU set => pCPU set, and send IPIs
- IPI handler: invalidate guest TLB entries with invpid
Evaluation

Dual-socket Dell R450 server
- 6-core Intel “Ivy-Bridge” Xeon processors with hyperthreading
- 24 GB RAM split across two NUMA domains.
- CentOS 7 (Linux 3.16)

Virtual Machines
- 12 vCPUs, 4G RAM on the same socket
- Fedora 19 (Linux 4.0)
- VM1: PARSEC Benchmark Suite, VM2 sysbench CPU test

Schemes
- baseline: unmodified Linux kernel
- kvmtlb [kvmtlb 12]
- Shoot4U
- Pause-Loop Exiting (PLE) [Riel 11]
- Preemptable Ticket Spinlock (PMT) [Ouyang VEE ’13]
## TLB Shootdown Latency (Cycles)

<table>
<thead>
<tr>
<th></th>
<th>baseline</th>
<th>kvmtlb</th>
<th>shoot4u</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1VM</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>166</td>
<td>122</td>
<td>28</td>
</tr>
<tr>
<td>Max</td>
<td>24,428</td>
<td>9,953</td>
<td>453</td>
</tr>
<tr>
<td><strong>2VM</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>9,048</td>
<td>5,401</td>
<td>22</td>
</tr>
<tr>
<td>Max</td>
<td>194,108</td>
<td>126,923</td>
<td>15,034</td>
</tr>
</tbody>
</table>

*Order of magnitude lower latency*
TLB Shootdown Latency (CDF)
Parsec Performance (2-VMs)

Normalized Execution Time

- baseline
- ple
- pmt
- pmt+kvmtlb
- pmt+shoot4u
- ple+pmt+shoot4u

Applications:
- blackscholes
- bodytrack
- canneal
- dedup
- ferret
- fregmine
- raytrace
- streamcluster
- swaptions
- vips
- x264
Revisiting Performance Slowdown

Slowdown

Baseline

dedup 70.6

bodytrack

canneal

dedup

eret

freqmine

raytrace

streamcluster

swaptions

vips

x264
Revisiting CPU Usage Profiling

![Bar graph showing the percentage of CPU usage for different benchmarks and configurations. The graph compares baseline 2VM and ple+pmt+shoot4u 2VM scenarios. The benchmarks includes blackscholes, bodytrack, canneal, dedup, ferret, freqmine, raytrace, streamcluster, swaptions, vips, and x264. The categories include k:lock, k:tlb, k:other, and u:*.]
Conclusions

- We conducted a set of experiments in order to provide a breakdown of overheads caused by preempted virtual CPU cores, showing that TLB operations can have a significant impact on performance with certain workloads.

- We Shoot4U, an optimization for TLB shootdown operations that internalizes TLB shootdowns in the VMM and so no longer requires the involvement of a guest’s vCPUs.

- Our evaluation demonstrates the effectiveness of our approach, and illustrates how under certain workloads our approach is dramatically better than state-of-the-art techniques.
https://github.com/ouyangjn/shoot4u
Jiannan Ouyang
Ph.D. Candidate
University of Pittsburgh
ouyang@cs.pitt.edu
http://www.cs.pitt.edu/~ouyang/

The Prognostic Lab
University of Pittsburgh
http://www.prognosticlab.org

Q & A

Pisces Co-Kernel

Kitten Lightweight Kernel

Palacios VMM
References


