Virtual Machine Boot-up Analysis

Abderrahmane Benbachir
May 5, 2017

École Polytechnique de Montréal
Laboratoire DORSAL
Agenda

Introduction
Tracing Through Para-virtualization Layers
Investigating Boot-up Issues
Questions
Introduction

CLOUD BENEFITS

- Easily upgraded
- Productivity anywhere
- Off site data storage
- Lower cost of ownership
- No IT maintenance costs
- Always up
- Disaster assistance
Boot-up process

- BIOS + MBR + Boot loader
  - Execute /sbin/init, initramfs root
  - CPU, Memory, Bus, IRQ, ...
  - Load built-in Modules
  - Run threads
  - Initializations
  - Execute /sbin/init, initramfs root

- No tracer, a few printk

- Ftrace handle this part

- LTTng, Ftrace, Perf, ...

- Init process

POLYTECHNIQUE MONTREAL – Abderrahmane Benbachir (Abder)
Tracing through para-virtualization layers
Hypercall

Direct Execution of Guest Requests

VM#1
Paravirtualized Guest OS

Qemu

Host Kernel
KVM.KO

Hardware

Layer not involved
Micro-Benchmarks

310 ns: VMCS switch + kvm_exit + kvm_entry

Hypercall Overhead

Mean
Median
Std deviation

340 ns
310 ns
833 ns

Hardware: Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz (x86_64)
Implementation

- ftrace
- function
- function_graph
- hypergraph
- nop
- wakeup
- sched_switch
- Irqsoff
Implementation

Tracing through Hypercalls

• Hook to Ftrace “function graph” probe for entry & exit
• Send function address
• Tracing only the hypervisor
• Use only host Timestamp
  -> No synchronization is required
• Resolve addresses using guest kernel symbols
Ftrace: Mcount and hypercall

1. Instrumented function
2. Change return entry to hook
3. Function entry tracer
4. Return hook
5. Function exit tracer
6. hypercall

Caller

hypercall

POLYTECHNIQUE MONTREAL – Abderrahmane Benbachir (Abder)
Depth Overhead Analysis

Function entry

hypercall

Context Switch Overhead

kvm_exit  
kvm_entry

Hypervisor Overhead

Hypervisor / Host

A

B

C

D

E

Hypercall

Function exit

kvm_exit

kvm_entry

Hypervisor / Host

kvm_exit

kvm_entry
Investigating Boot-up issues

Kernel Crash/Panic
Latencies
Boot-up issues
Boot-up issues

Received sequence: A-B-C-D-E-F
Boot-up issues

Kernel crash: Oops, panic, ... etc.

Received sequence: A-B-C-D-E-F
Boot-up issues

→ Reduce by half the trace events
→ Better for Performance Analysis

Trace only exits + duration
Boot-up issues

Received sequence: B-E-F-D-C-A

Hypercall overhead
Boot-up issues

Kernel crash: Oops, panic, ... etc.

Received sequence: B-E

Hypercall overhead

→ Not good for debugging kernel crash
Visualization
Call graph

TraceCompass: Callstack XML analysis

Graph of Guest Kernel space

Thanks to Geneviève
Call graph

```plaintext
tracer: hypergraph

<table>
<thead>
<tr>
<th>CPU</th>
<th>TASK/PID</th>
<th>DURATION</th>
<th>Depth</th>
<th>FUNCTION CALLS</th>
</tr>
</thead>
</table>
| 5   | <->-1    | 0.912 us  | d=0   | mutex_unlock() {
| 5   | <->-1    | 0.679 us  | d=1   | -initcall_blacklisted() {
| 5   | <->-1    | 1.452 us  | d=0   | -do_one_initcall() {
| 5   | <->-1    | 0.674 us  | d=1   | -initcall_blacklisted() {
| 5   | <->-1    | 0.733 us  | d=2   | --register_module_notifier() {
| 5   | <->-1    | 1.388 us  | d=1   | --blocking_notifier_chain_register() {
| 5   | <->-1    | 2.05 us   | d=0   | -do_one_initcall() {
| 5   | <->-1    | 0.675 us  | d=1   | -initcall_blacklisted() {
| 5   | <->-1    | 0.68 us   | d=3   | --alloc_workqueue_key() {
| 5   | <->-1    | 0.665 us  | d=3   | --kmalloc() {
| 5   | <->-1    | 1.393 us  | d=3   | --kmem_slab() {
| 5   | <->-1    | 0.673 us  | d=2   | --cond_resched() {
| 5   | <->-1    | 0.677 us  | d=2   | --mutex_init() {
| 5   | <->-1    | 0.677 us  | d=2   | --alloc_percpu() {
| 5   | <->-1    | 0.667 us  | d=3   | --pcpu_alloc() {
| 5   | <->-1    | 0.667 us  | d=5   | --mutex_lock() {
| 5   | <->-1    | 0.667 us  | d=5   | --cond_resched() {
```
Questions

abderrahmane.benbachir@polymtl.ca
https://github.com/abbenbachir
Hypercall Implementation: https://gist.github.com/abenbachir/344822b5ba9fc5ac384cdec3f087e018

QEMU Hypertrace Patches: http://patchwork.ozlabs.org/project/qemu-devel/list/?state=&q=Hypertrace&archive


Callstack xml analysis: https://gist.github.com/abenbachir/e813790f183945b6f74dc74ecce57c75