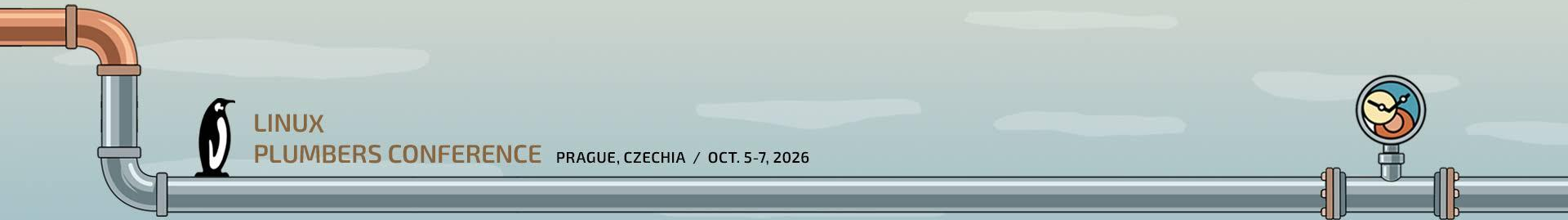


Why Collect VM-Exit Data?

1. Which exits fire under real workloads, and how often? And which exit reasons never fire at all?
2. Would the Caretaker handle each one, or would it park the vCPU?



Goal: Target Event Frequency (λ) Analysis

Mathematical Model

Key Definitions

- **X** = Guest Perceived Downtime Target
- **Y** = Kernel Reboot Time Target (Liveupdate)
- **T** = Downtime Risk Window (**Y** - **X**)
- **λ** = Max Event Frequency (VM_EXIT / sec)

Target Probabilities:

- P99 = 0.01 max risk threshold
- P90 = 0.10 max risk threshold

Probability Formula

$$P = 1 - e^{-\lambda T}$$

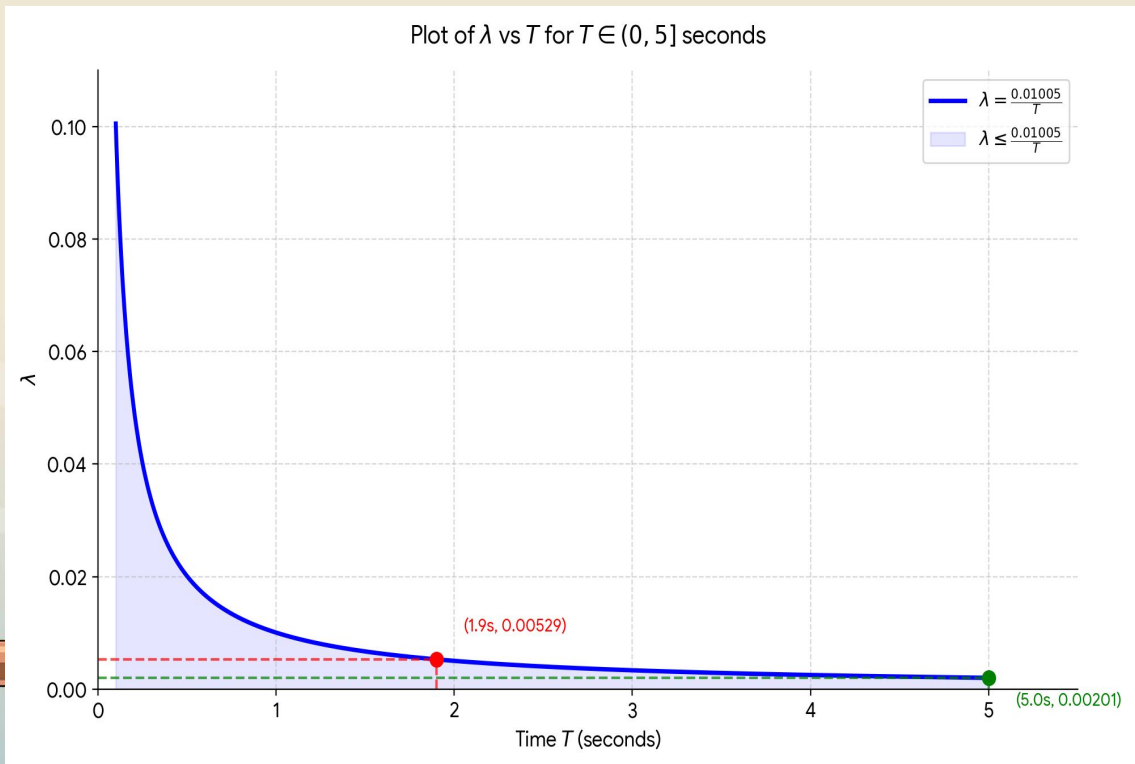
P99

Kernel Reboot (Y)	Downtime (X)	Max λ (VM_EXIT / s)
2 sec	100 ms	0.0052
2 sec	500 ms	0.0067
2 sec	1 sec	0.0100
5 sec	500 ms	0.0022
5 sec	1 sec	0.0025

P90

Kernel Reboot (Y)	Downtime (X)	Max λ (VM_EXIT / s)
2 sec	100 ms	0.0520
2 sec	500 ms	0.0670
2 sec	1 sec	0.1
5 sec	500 ms	0.0230
5 sec	1 sec	0.0210

Goal: Target Event Frequency (λ) Analysis



Categorization

1. Caretaker Can handle
2. Caretaker Can't handle, Make sure $\lambda \leq$ Max Target freq

Note: VM_EXIT can be handled by adding support in caretaker,

Need for handling by caretaker $\propto \lambda / Z$;
Z = Engineering complexity



What Data We Collected

Simulated Workloads

Environment

Mainline kernel/qemu + Ubuntu 24 LTS VM

Workload Profile

Memory, IO, Network intensive workload

Arch

Intel, AMD, ARM64

Fleet Telemetry

Environment

Production kernel/VMM

Workload Profile

Aggregated Varied real life workload

Arch

Intel, AMD, ARM64



How the Data Was Collected

1. kvm:kvm_exit tracepoint

already fires on every VM exit



2. tracefs hist trigger

counts per exit reason in the kernel

Each run: warm-up (not counted) → counter = 0 → full run (300 s / 60 s) → read counter



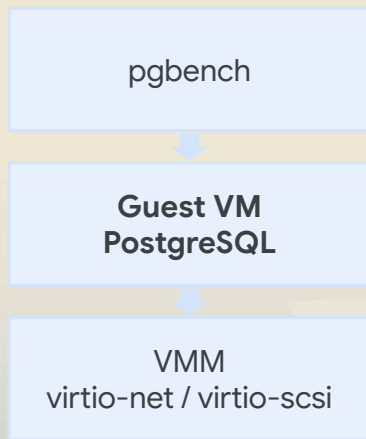
LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



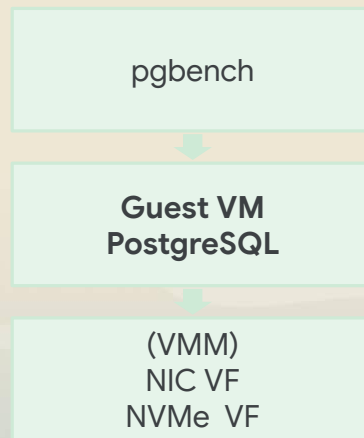
Simulated Workload Setup

Virtio + PostgreSQL



117 GB DB; 300 s

Passthrough + PostgreSQL



117 GB DB; 300 s

Guest VM: 16 vCPUs, 64 GB RAM

VM Exits per Run — Virtio + PostgreSQL (Disk) (Prefaulted)

INTEL

VM exit reason (Intel)	Exits per vCPUs per Sec
HLT	3,248.32
EPT_MISCONFIG	1,969.46
MSR_WRITE	980.47
PREEMPTION_TIMER	95.53
EXTERNAL_INTERRUPT	91.22
PAUSE_INSTRUCTION	5.83
MSR_READ / CPUID	1.00 / 0.02
VMCALL / APIC_WRITE	0.06 / 0.004
EXCEPTION_NMI	0.01
EPT_VIOLATION IO_INSTRUCTION ... so on	None

ARM64

VM exit reason (ARM64)	Exits per vCPUs per Sec
TRAP_WF _x	3,272.33
TRAP_DABT_LOW	2,014.51
IRQ (Host Physical IRQ EL2)	1,916.92
TRAP_SYS64	985.92
TRAP_HVC64	0.01
TRAP_IABT_LOW TRAP_SMC64 MMIO_EXIT_USER ... so on	None

AMD

VM exit reason (AMD)	Exits per vCPUs per Sec
IDLE_HLT	3,569.07
MSR	3,399.76
INTR	3,232.94
NPF	2,544.58
VINTR	1,837.90
VMMCALL	0.06
CPUID	0.02
NMI	0.02
IOIO AVIC_INCOMP_IPI PAUSE ... so on	None

Green: Caretaker can handle
Red: parks the vCPU

VM Exits per Run — Passthrough + PostgreSQL (Disk)

Prefaulted

INTEL

VM exit reason	Exits per vCPUs per Sec
HLT	3,480.95
MSR_WRITE	1,013.48
EXTERNAL_INTERRUPT	94.87
PREEMPTION_TIMER	90.37
MSR_READ	1.11
CPUID	0.02
EXCEPTION_NMI*	0.01
APIC_WRITE EPT_VIOLATION EPT_MISCONFIG IO_INSTRUCTION ... so on	Below 0.0021

ARM64

VM exit reason (AMD)	Exits per vCPUs per Sec
TRAP_WF _x	3,723.94
IRQ (Host Physical IRQ EL2)	3,000.19
TRAP_SYS64	2,083.12
TRAP_HVC64	0.0041
TRAP_DABT_LOW TRAP_IABT_LOW MMIO_EXIT_USER MMIO_EXIT_KERNEL...so on	Below 0.0021

AMD

VM exit reason (ARM)	Exits per vCPUs per Sec
IDLE_HLT	3,219.78
MSR	1,059.68
AVIC_INCOMP_IPI	570.37
INTR	149.51
CPUID	0.04
NMI	0.01
NPF IOIO VINTR (100)...so on	Below 0.0021

Green: Caretaker can handle

Red: parks the vCPU



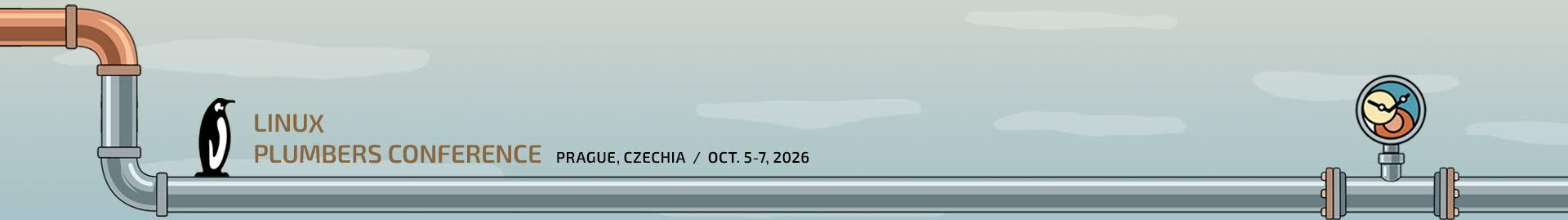
LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Fleet telemetry

- Gives insight into what real workloads do at scale
- Collected across multiple modern VM families
- Aggregated over a 24 hour window



Fleet telemetry: Intel

Exit Reason
MSR_WRITE
EXTERNAL_INTERRUPT
HLT
VMRESUME
MSR_READ
EPT_VIOLATION
CPUID
XSETBV
PREEMPTION_TIMER
IO_INSTRUCTION
VMWRITE
EPT_MISCONFIG
VMPTRLD

Exit Reason
APIC_WRITE
VMREAD
INVEPT
EOI_INDUCED
EXCEPTION_NMI
DR_ACCESS
VMCLEAR
VMLAUNCH
VMCALL
RDTSC
CR_ACCESS

Green: Caretaker can handle

Red: parks the vCPU

Gray: Nested VMs



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Fleet telemetry: AMD

Exit Reason
MSR
INTR
VINTR
AVIC_UNACCELERATED_ACCESS
CPUID
AVIC_INCOMPLETE_IPI
NPF
READ_DR6
IOIO

Exit Reason
HLT
WRITE_CR8
RDPMC
WRITE_CR4
READ_CR4
VMRUN
WRITE_DR7
VMMCALL

Green: Caretaker can handle

Red: parks the vCPU

Gray: Nested VMs



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Fleet telemetry: ARM64

Exit Reason
IRQ
SYS64
WFI
WFE
DABT_FAULT
IABT_LOW
HVC64
IABT_LOW_S1PTW

Green: Caretaker can handle

Red: parks the vCPU

Gray: Nested VMs



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Fleet telemetry: key takeaways

- MMIO exits are the biggest blocker
 - Things like console, virtio, TPM, etc.
- Virtualization is naturally trying to reduce host involvement
 - Passthrough devices, posted interrupts, nested page tables, etc.
- Slice-of-hardware VMs are easier to handle
- Host interrupts are frequent
 - But during live update, they should be masked
- A lot of exits need to be handled by the Caretaker
 - We should try to re-use as much existing exit handling logic as we can

