# Virtualization of a Real-Time Operating System for Robot Control with a Focus on Real-Time Compliance

Halil Pamuk, BSc.

Sebastian Rauh, MSc. Beng.

FH University of Applied Sciences

TECHNIKUM

WIEN

SIGMATEK

# Introduction

- Robots perform time-critical tasks

- Delays → catastrophic consequences

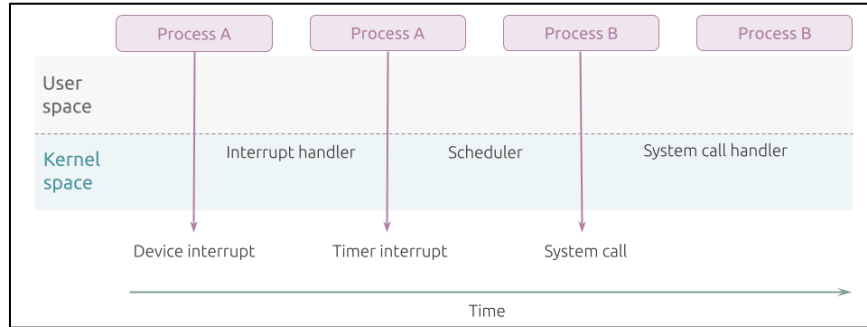- General-Purpose Operating System vs. Real-Time Operating System



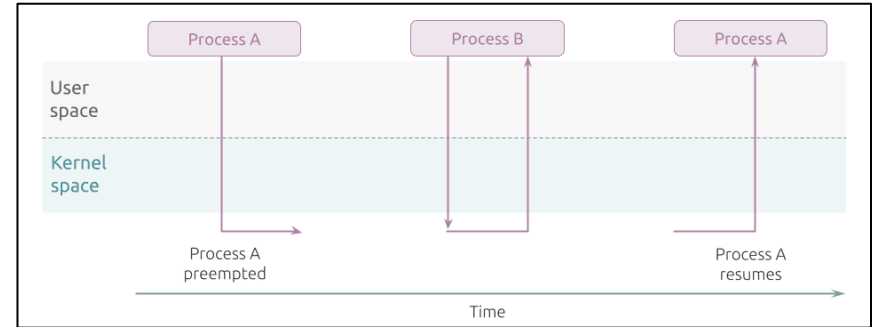Figure 1: Non-preemptible Kernel [1]



Figure 2: Preemptible Kernel [1]
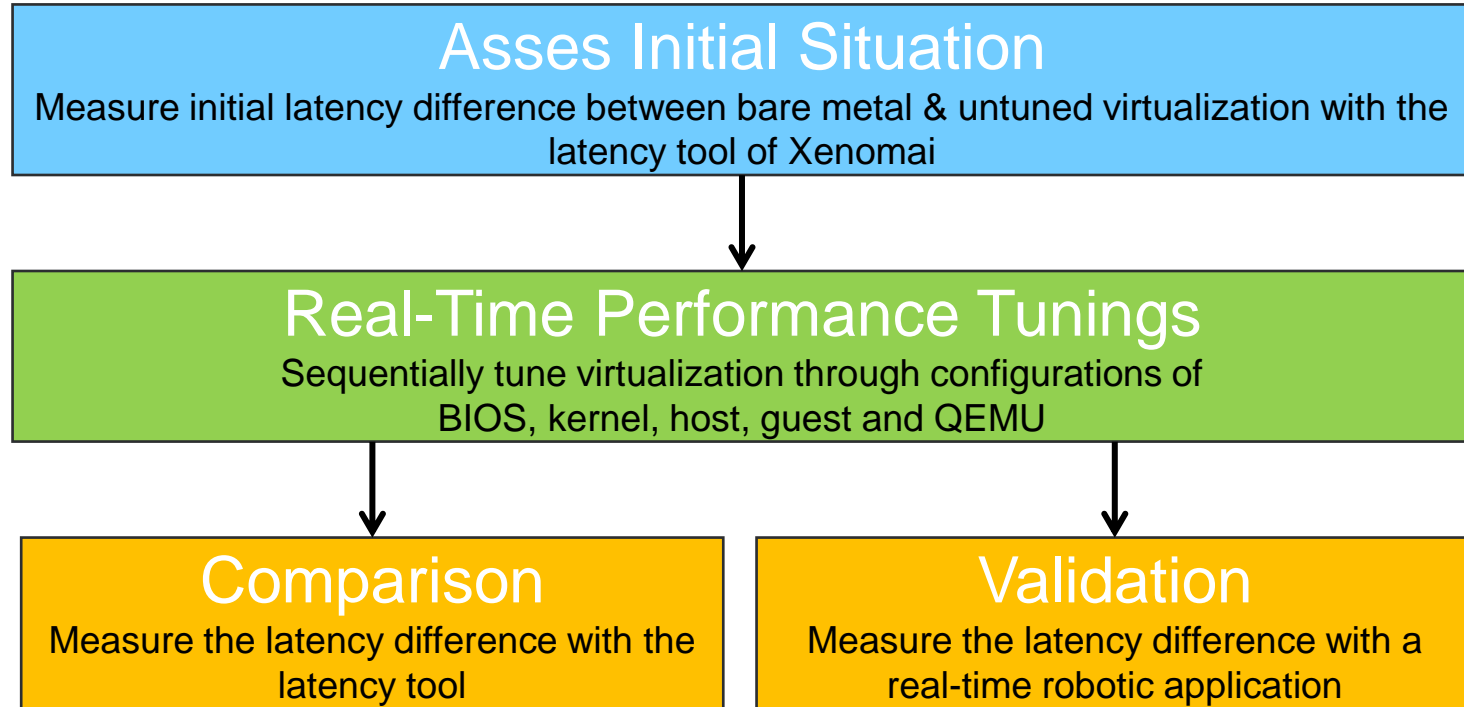
# Problem and Task Definition

| Virtualization of Real-Time Operating Systems | |
|---|---|
| **Advantages** | **Disadvantages** |
| Scalability & flexibility | Increased overhead and latency |
| Cheaper | Performance variability |
| Remote management | Complexity |

- **Research Question:** Is it possible, and if so, how can the latency of a real-time operating system virtualization be reduced using Yocto, Xenomai, and QEMU to a level that is closer to that of bare metal (below 50 µs)?

# Application Context and Conditions

- This work was written at SIGMATEK GmbH & Co KG

- **Host OS:** Ubuntu 22.04.4 LTS, PREEMPT-RT

- **Guest OS:** Salamander 4
  - Built with Yocto [2]
  - Virtualized through Quick Emulator (QEMU) [3]
  - Hard real-time with Xenomai 3 [4]

- Trace-cmd [5] and Kernelshark [6] for kernel tracing and visualization

# Methodology

**Asses Initial Situation**
Measure initial latency difference between bare metal & untuned virtualization with the latency tool of Xenomai

**Real-Time Performance Tunings**
Sequentially tune virtualization through configurations of BIOS, kernel, host, guest and QEMU

**Comparison**
Measure the latency difference with the latency tool

**Validation**
Measure the latency difference with a real-time robotic application

# Results – Latency Tool

- 10 minutes with a sampling period of 100 µs, priority of 99
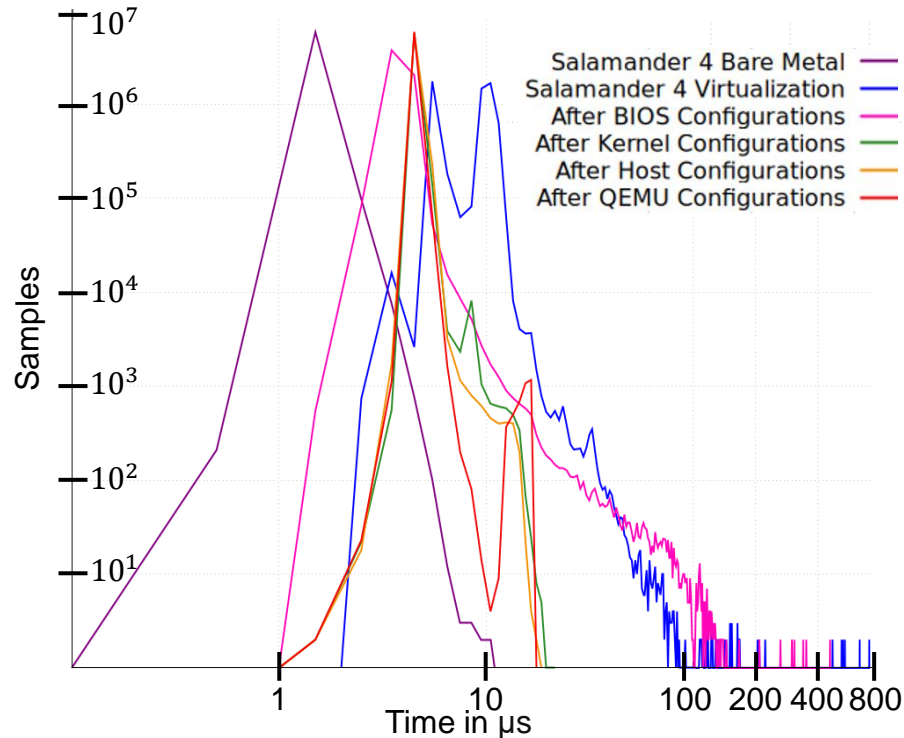


Figure 1: Comparison of Latency Distributions

| Version | Latency (µs) | | | Overruns |
|---|---|---|---|---|
| | Min | Avg | Max | |
| Bare Metal | 0.613 | 1.380 | 10.709 | 0 |
| Untuned Virtualization | 2.536 | 8.940 | 707.622 | 43 |
| After BIOS Configurations | 0.969 | 3.948 | 457.545 | 22 |
| After Kernel Configurations | 2.545 | 4.811 | 21.694 | 0 |
| After Host Configurations | 2.591 | 4.834 | 18.441 | 0 |
| After QEMU Configurations | 2.614 | 4.779 | 17.134 | 0 |

Table 1: Comparison of Latency Results

Virtualization of a Real-Time Operating System for Robot Control with a Focus on Real-Time Compliance | Halil Pamuk, BSc. | 10.10.2024

6

# Results – Robotic Application

- Difference between command issuance time and signal arrival at PWM
- 1,000 samples

| Version | Latency (ms) | | | Std Dev (ms) |
|---|---|---|---|---|
| | Min | Avg | Max | |
| Bare Metal | 1.211 | 1.347 | 1.49 | 0.082 |
| Untuned Virtualization | 3.1 | 24.603 | 129.46 | 13.876 |
| Tuned Virtualization | 1.219 | 2.62 | 3.988 | 0.812 |

Table 2: Comparison of Robotic Application Latency Results

# Discussion

✓ **Latency Tool**

• Worst latency decreased from 707.622 μs to 17.134 μs

• Close to bare metal's 10.709 μs

• No overruns

• Goal achieved

✓ **Robotic Application**

• Worst latency dropped from 129 ms to 3.988 ms

• Close to bare metal's 1.49 ms

• Tunings validated

# Outlook

- Additional configurations


- Other hypervisors and virtualization technologies


- More testing under workloads

# References

- [1] https://ubuntu.com/blog/what-is-real-time-linux-ii

- [2] https://docs.yoctoproject.org/

- [3] https://www.qemu.org/

- [4] https://xenomai.org/

- [5] https://trace-cmd.org/

- [6] https://kernelshark.org/

Virtualization of a Real-Time Operating System for Robot Control with a Focus on Real-Time Compliance | Halil Pamuk, BSc. | 10.10.2024

10

# Salamander 4 Bare Metal
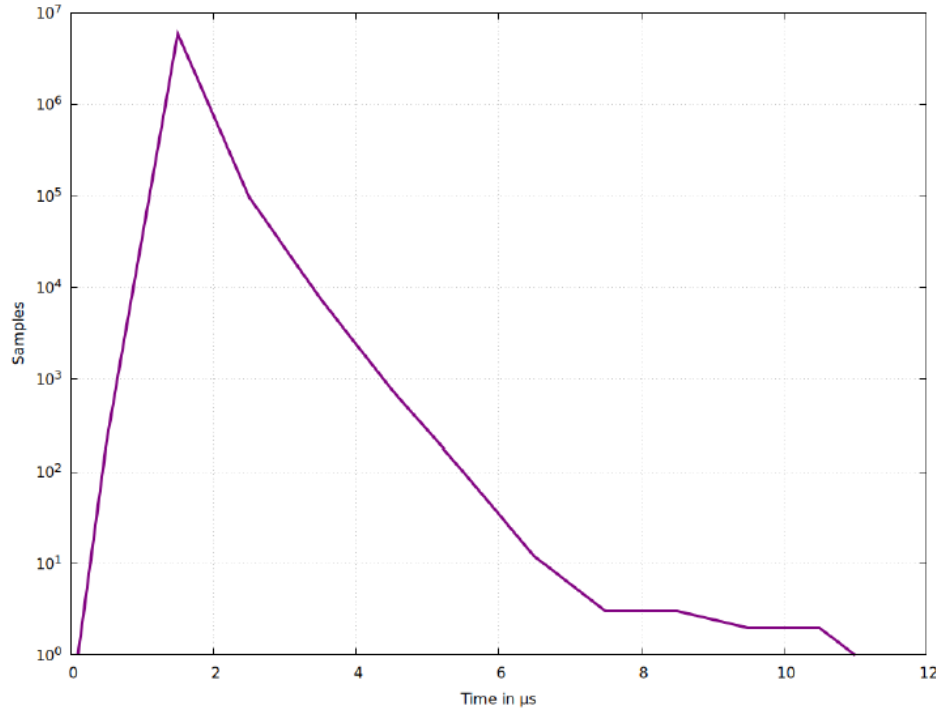


Figure 12: Latency Distribution of Salamander 4 Bare Metal

Table 2: Latency Parameters of Bare Metal

| Param | Samples | Average (µs) | Std Dev (µs) |
|-------|---------|--------------|--------------|
| min | 599 | 0.711 | 0.454 |
| avg | 5,999,988 | 1.019 | 0.150 |
| max | 599 | 3.528 | 0.895 |

Table 3: Minimum, Average, and Maximum Latency with Overrun Counts of Bare Metal

| Lat Min (µs) | Lat Avg (µs) | Lat Max (µs) | Overruns |
|--------------|--------------|--------------|----------|
| 0.613 | 1.380 | 10.709 | 0 |

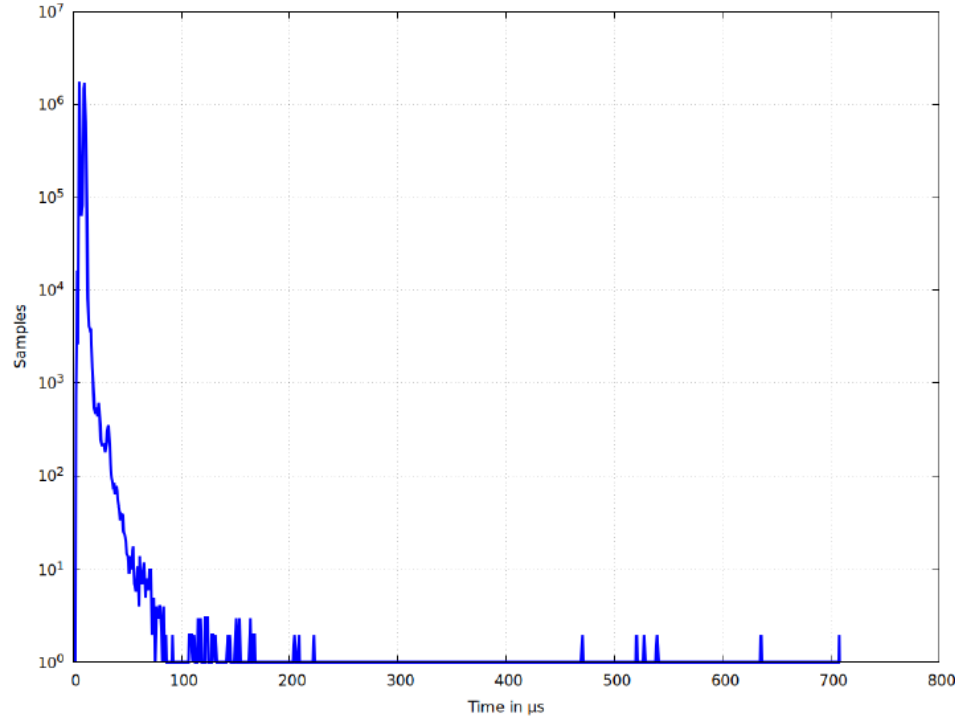# Salamander 4 Untuned Virtualization



Figure 13: Latency Distribution of Salamander 4 Untuned Virtualization

Table 4: Latency Parameters of Untuned Virtualization

| Param | Samples | Average (µs) | Std Dev (µs) |
|-------|---------|--------------|--------------|
| min | 599 | 3.713 | 1.355 |
| avg | 5,999,922 | 8.247 | 2.521 |
| max | 599 | 45.705 | 52.196 |

Table 5: Minimum, Average, and Maximum Latency with Overrun Counts of Untuned Virtualization

| Lat Min (µs) | Lat Avg (µs) | Lat Max (µs) | Overruns |
|--------------|--------------|--------------|----------|
| 2.536 | 8.940 | 707.622 | 43 |

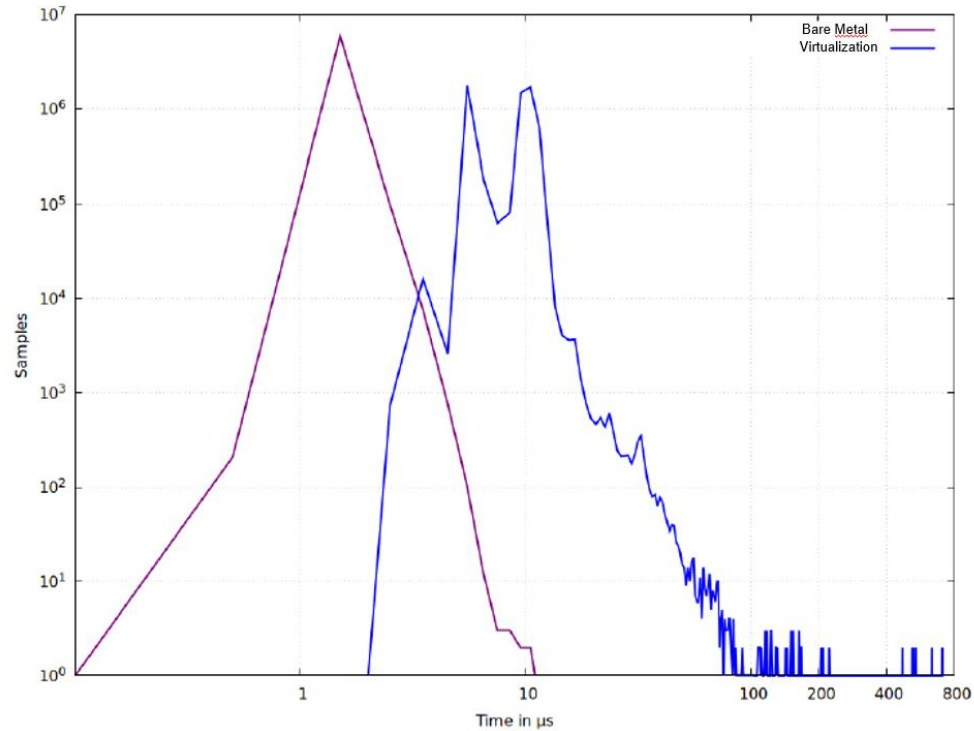# Salamander 4 Bare Metal vs. Virtualization



Figure 14: Initial Comparison of Latency Distribution between Bare Metal and Virtualization

# Salamander 4 Virtualization BIOS Configurations

Table 6: BIOS Configurations for Real-Time Performance

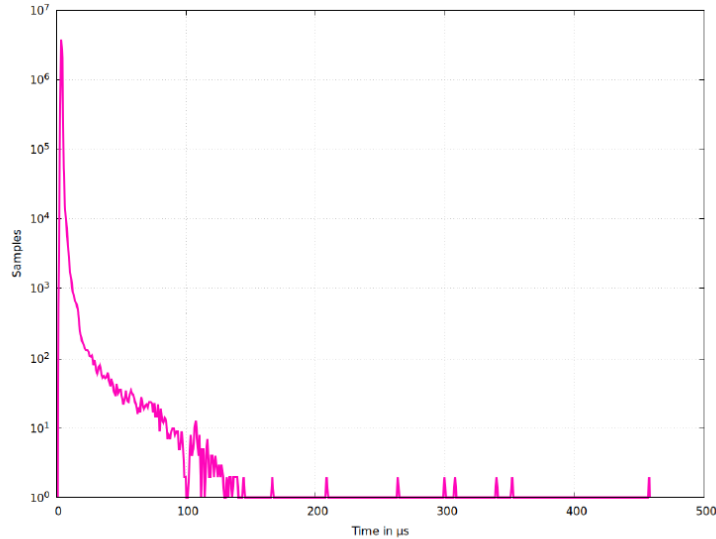| Option | Status |
|---|---|
| Hyper Threading | Disabled |
| Intel SpeedStep® | Disabled |
| Intel® Speed Shift Technology | Disabled |
| C-States | Disabled |
| VT-d | Enabled |

# Salamander 4 Virtualization after BIOS Configurations



Figure 15: Latency Distribution of Salamander 4 Virtualization after tuning BIOS Configurations

Table 7: Latency Parameters after BIOS Configurations

| Param | Samples | Average (µs) | Std Dev (µs) |
|-------|---------|--------------|--------------|
| min | 599 | 1.419 | 0.507 |
| avg | 5,999,885 | 3.398 | 1.251 |
| max | 599 | 74.015 | 26.590 |

Table 8: Minimum, Average, and Maximum Latency with Overrun Counts after BIOS Configurations

| Lat Min (µs) | Lat Avg (µs) | Lat Max (µs) | Overruns |
|--------------|--------------|--------------|----------|
| 0.969 | 3.948 | 457.545 | 22 |

# Salamander 4 Virtualization Kernel Configurations

```
1    GRUB_CMDLINE_LINUX="isolcpus=4 rcu_nocbs=4 rcu_nocb_poll nohz_full=4
        default_hugepagesz=1G hugepagesz=1G hugepages=8 intel_iommu=on
        rdt=l3cat nmi_watchdog=0 idle=poll clocksource=tsc tsc=reliable audit=0
        skew_tick=1 intel_pstate=disable intel.max_cstate=0
        intel_idle.max_cstate=0 processor.max_cstate=0
        processor_idle.max_cstate=0 nosoftlockup no_timer_check nospectre_v2
        spectre_v2_user=off kvm.kvmclock_periodic_sync=N kvm_intel.ple_gap=0
        irqaffinity=0"
```

Code 5: Kernel Configurations for Real–Time Performance

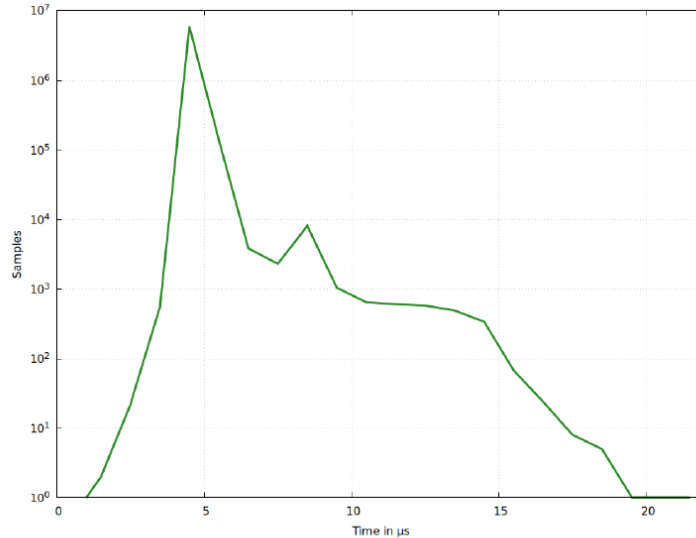# Salamander 4 Virtualization after Kernel Configurations



Figure 16: Latency Distribution of Salamander 4 Virtualization after tuning Kernel Configurations

Table 9: Latency Parameters after Kernel Configurations

| Param | Samples | Average (µs) | Std Dev (µs) |
|-------|---------|--------------|--------------|
| min | 599 | 3.356 | 0.551 |
| avg | 5,999,972 | 4.036 | 0.290 |
| max | 599 | 13.484 | 1.454 |

Table 10: Minimum, Average, and Maximum Latency with Overrun Counts after Kernel Configurations

| Lat Min (µs) | Lat Avg (µs) | Lat Max (µs) | Overruns |
|--------------|--------------|--------------|----------|
| 2.545 | 4.811 | 21.694 | 0 |

# Salamander 4 Virtualization Host Configurations

CPU Affinity and Isolation

Interrupt Affinity

RT-Priority

Disable RT Throttling

Disable Timer Migration

Set Device Driver Work Queue

Disable RCU CPU Stall Warnings

Stop Services

Disable Machine Check

Boot into Text-Based Environment
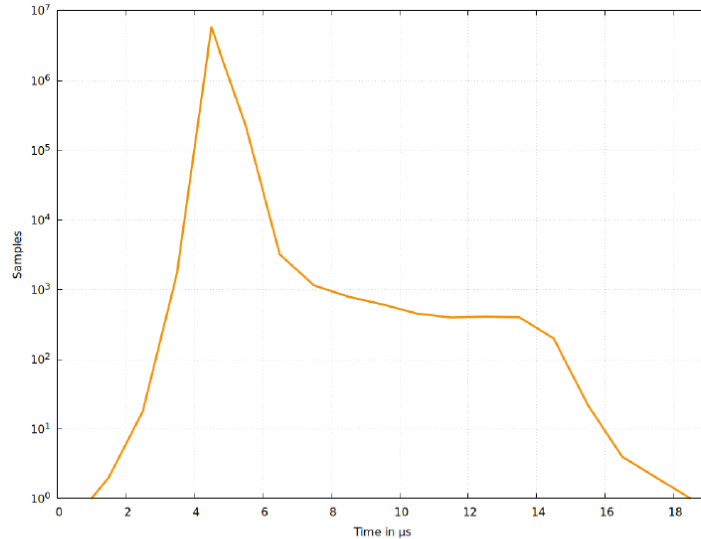
# Salamander 4 Virtualization after Host Configurations



Figure 17: Latency Distribution of Salamander 4 Virtualization after tuning Host Configurations

Table 13: Latency Parameters for Host Configurations

| Param | Samples | Average (µs) | Std Dev (µs) |
|-------|---------|--------------|--------------|
| min | 599 | 2.998 | 0.242 |
| avg | 5,999,972 | 4.043 | 0.255 |
| max | 599 | 12.124 | 1.828 |

Table 14: Minimum, Average, and Maximum Latency with Overrun Counts after Host Configurations

| Lat Min (µs) | Lat Avg (µs) | Lat Max (µs) | Overruns |
|--------------|--------------|--------------|----------|
| 2.591 | 4.834 | 18.441 | 0 |

# Salamander 4 Virtualization QEMU Configurations

```
1      #!/bin/sh
2
3      if  [ ! -d drive-c/ ]; then
4          echo "Filling drive-c/"
5          mkdir drive-c/
6          tar -C drive-c/ -xf stek-drive-c-image-sigmatek-core2.tar.gz
7      fi
8
9      exec taskset -c 4 chrt -f 99 qemu-system-x86_64 -M pc,accel=kvm -kernel
           ./bzImage \
10     -m 2048 -drive
           file=salamander-image-sigmatek-core2.ext4,format=raw,media=disk \
11     -append "console=ttyS0 console=tty1 root=/dev/sda rw panic=1
           sigmatek_lrt.QEMU=1 ip=dhcp rootfstype=ext4 schedstats=enable nohlt
           idle=poll quiet xeno_hal.smi=1 xenomai.smi=1 threadirqs" \
12     -net nic,model=e1000,netdev=e1000 -netdev bridge,id=e1000,br=nm-bridge \
13     -fsdev local,security_model=none,id=fsdev0,path=drive-c -device
           virtio-9p-pci,id=fs0,fsdev=fsdev0,mount_tag=/mnt/drive-C \
14     -device vhost-vsock-pci,guest-cid=3,id=vsock0 \
15     -drive if=pflash,format=qcow2,file=ovmf.code.qcow2 \
16     -object memory-backend-ram,id=ram0,size=4G,prealloc=on \
17     -mem-prealloc -mem-path /dev/hugepages \
18     -device vfio-pci,host=03:00.0 \
19     -no-reboot -nographic
```

Code 10: Final QEMU Script for Salamander 4 Virtualization with PCI Configuration

Tune LAPIC Timer Advance

Set QEMU Options for Real-Time VM

# Priorities for Different Scheduling Policies

Table 11: Minimum and Maximum Priorities for Different Scheduling Policies

| Scheduling Policy | Min Priority | Max Priority |
|---|---|---|
| SCHED_OTHER | 0 | 0 |
| SCHED_FIFO | 1 | 99 |
| SCHED_RR | 1 | 99 |
| SCHED_BATCH | 0 | 0 |
| SCHED_IDLE | 0 | 0 |
| SCHED_DEADLINE | 0 | 0 |

# Salamander 4 Virtualization after QEMU Configurations
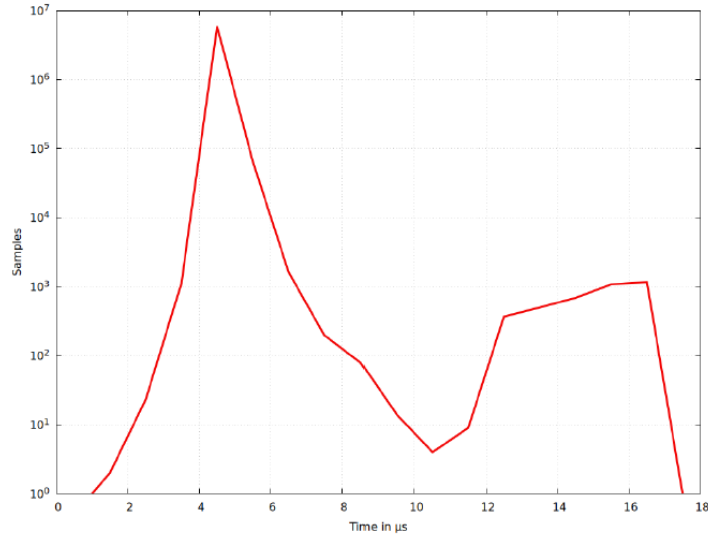


Figure 18: Latency Distribution of Salamander 4 Virtualization after tuning QEMU Configurations

Table 16: Latency Parameters after QEMU Configurations

| Param | Samples | Average (µs) | Std Dev (µs) |
|-------|---------|--------------|--------------|
| min | 599 | 3.125 | 0.432 |
| avg | 5,999,973 | 4.018 | 0.291 |
| max | 599 | 15.883 | 0.351 |

Table 17: Minimum, Average, and Maximum Latency with Overrun Counts after QEMU Configurations

| Lat Min (µs) | Lat Avg (µs) | Lat Max (µs) | Overruns |
|--------------|--------------|--------------|----------|
| 2.614 | 4.779 | 17.134 | 0 |

# Robot



Figure 19: Mini-Robot of the Experiment [60]



| Wire Color | Description |
|---|---|
| Brown | Ground wire connected to the ground of system |
| Red | Powers the motor typically +5V is used |
| Orange | PWM signal is given in through this wire to drive the motor |

Figure 20: MG996R Servo Motor [62]
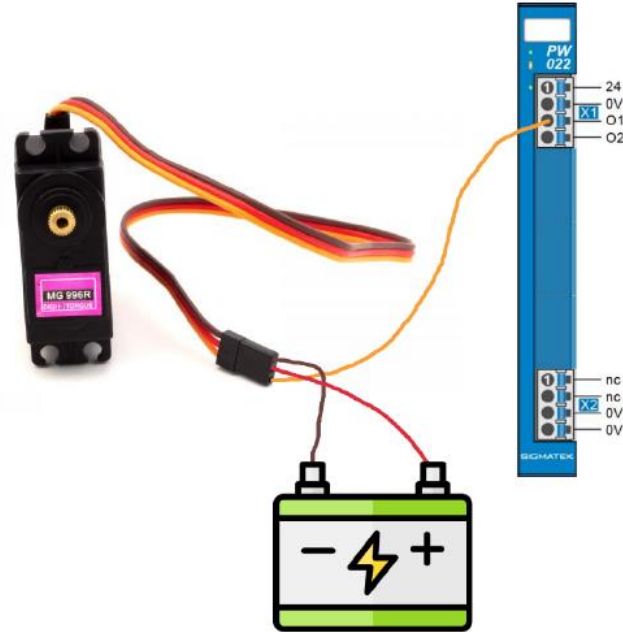
# Servo Motor with PWM module



Figure 23: Connection between PW 022 and Mini-Robot [65]
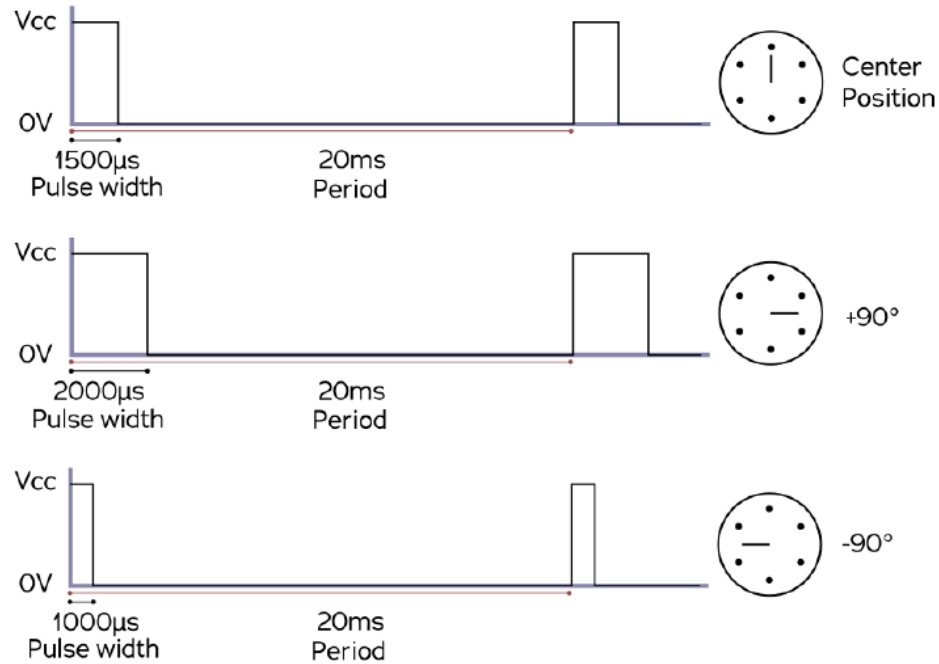
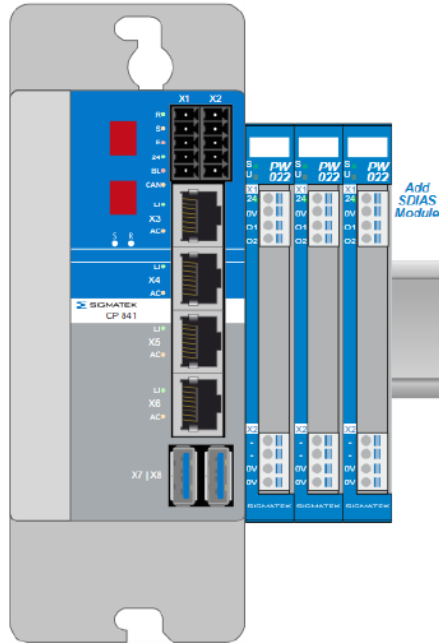# Servo Motor Function



Figure 21: Controlling the MG996R Servo Motor [63]

# Robotic Application Setups
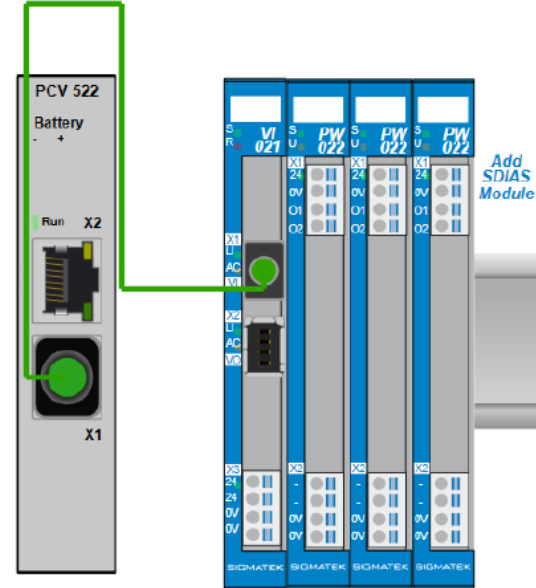


Figure 24: Setup of Salamander 4 Bare Metal



Figure 25: Setup of Salamander 4 Virtualization
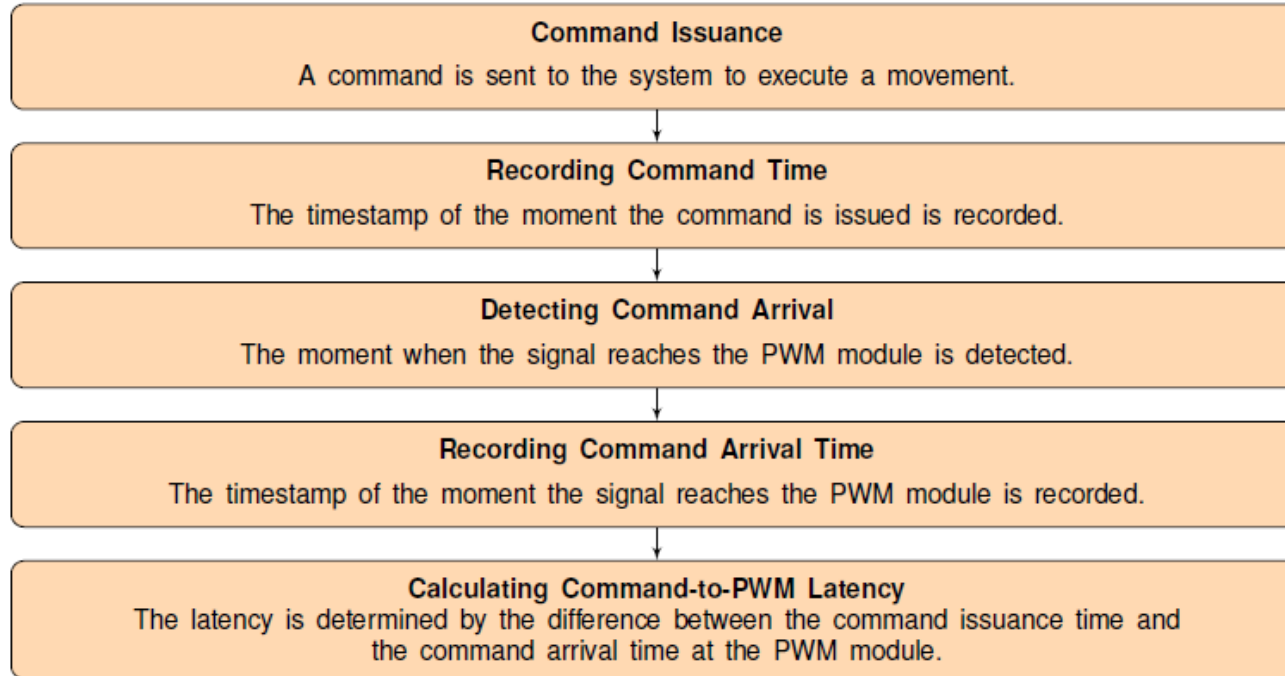
# Robotic Application Flowchart



Figure 26: Flowchart of Robotic Application

Virtualization of a Real-Time Operating System for Robot Control with a Focus on Real-Time Compliance | Halil Pamuk, BSc. | 10.10.2024

27

# Results of Robotic Application
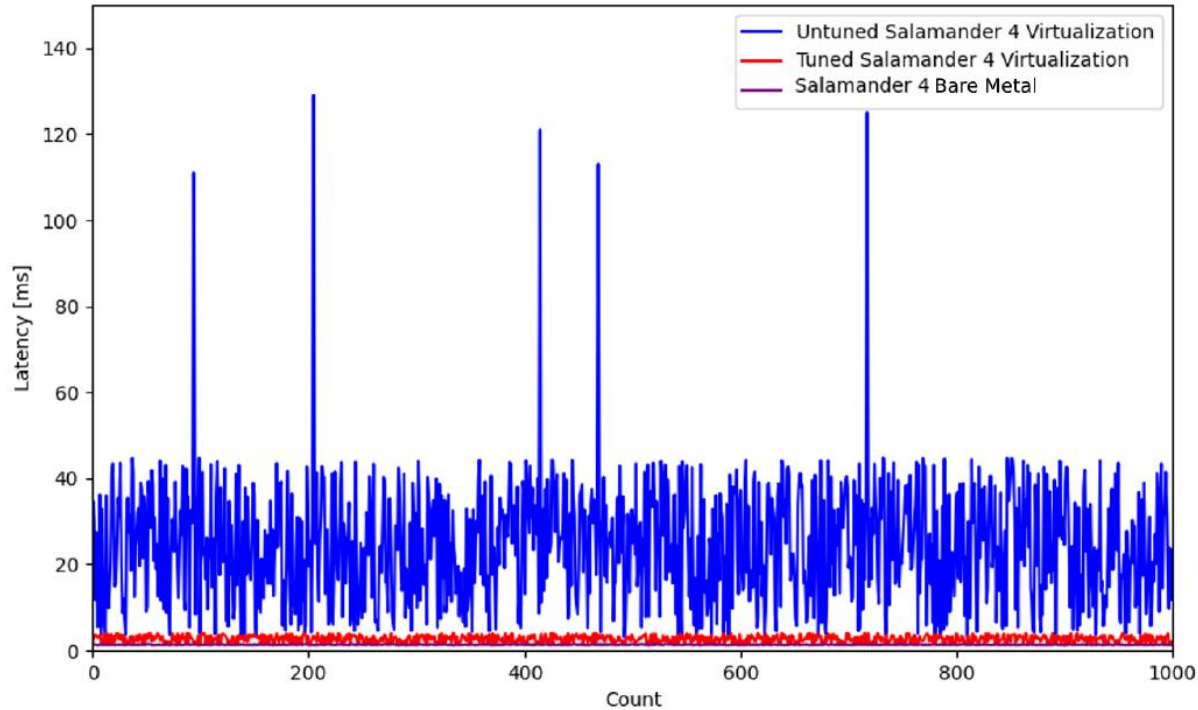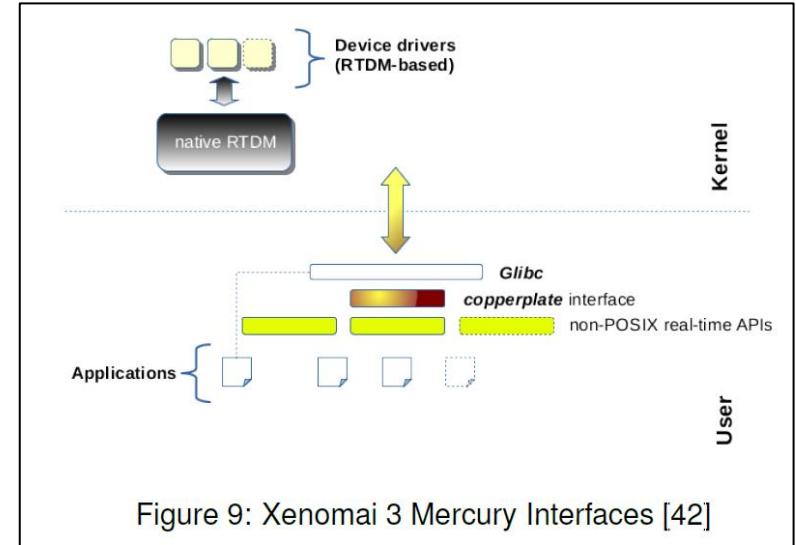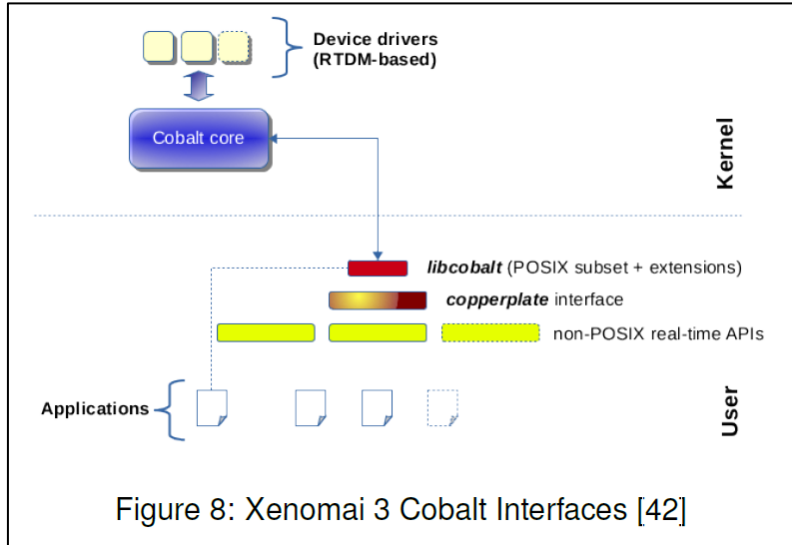


Figure 31: Comparison of Robotic Application Latency after Configurations

# Host Operating System

Table 1: Host Operating System Configuration

| | |
|---|---|
| **CPU** | Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 6 cores |
| **Memory** | $4 \times$ 8GB DDR4-2666/2400 MHz, 32GB |
| **GPU** | Intel UHD Graphics 630 GPU |
| **Storage** | 500GB NVMe SSD |
| **BIOS** | Dell Version 1.29.0 |
| **OS** | Ubuntu 22.04.4 LTS |
| **Kernel Version** | 6.8.0-40-generic |

# Xenomai Approaches



Figure 8: Xenomai 3 Cobalt Interfaces [42]



Figure 9: Xenomai 3 Mercury Interfaces [42]

# QEMU Script

```sh
1   #!/bin/sh
2
3   if  [ ! -d drive-c/ ]; then
4         echo "Filling drive-c/"
5         mkdir drive-c/
6         tar -C drive-c/ -xf stek-drive-c-image-sigmatek-core2.tar.gz
7   fi
8
9   exec qemu-system-x86_64 -M pc,accel=kvm -kernel ./bzImage \
10  -m 2048 -drive
        file=salamander-image-sigmatek-core2.ext4,format=raw,media=disk \
11  -append "console=ttyS0 console=tty1 root=/dev/sda rw panic=1
        sigmatek_lrt.QEMU=1 ip=dhcp rootfstype=ext4" \
12  -net nic,model=e1000,netdev=e1000 -netdev bridge,id=e1000,br=nm-bridge \
13  -fsdev local,security_model=none,id=fsdev0,path=drive-c -device
        virtio-9p-pci,id=fs0,fsdev=fsdev0,mount_tag=/mnt/drive-C \
14  -device vhost-vsock-pci,guest-cid=3,id=vsock0 \
15  -drive if=pflash,format=qcow2,file=ovmf.code.qcow2 \
16  -no-reboot -nographic
```

Code 3: QEMU Script for starting Salamander 4 Virtualization

# Trace-cmd

```
1     # Vsockets settings
2     CONFIG_VSOCKETS=m
3     CONFIG_VHOST_VSOCK=m
4     CONFIG_VIRTIO_VSOCKETS=m
5     CONFIG_VIRTIO_VSOCKETS_COMMON=m
6     CONFIG_VSOCKETS_DIAG=m
7     CONFIG_VSOCKETS_LOOPBACK=m
8     # Tracing settings
9     CONFIG_TRACING=y
10    CONFIG_FTRACE=y
11    CONFIG_FUNCTION_TRACER=y
12    CONFIG_FUNCTION_GRAPH_TRACER=y
13    CONFIG_DYNAMIC_FTRACE=y
14    CONFIG_DYNAMIC_FTRACE_WITH_REGS=y
15    CONFIG_DYNAMIC_FTRACE_WITH_DIRECT_CALLS=y
16    CONFIG_DYNAMIC_FTRACE_WITH_ARGS=y
17    CONFIG_SCHED_TRACER=y
18    CONFIG_FTRACE_SYSCALLS=y
19    CONFIG_TRACER_SNAPSHOT=y
20    CONFIG_KPROBE_EVENTS=y
21    CONFIG_UPROBE_EVENTS=y
22    CONFIG_BPF_EVENTS=y
23    CONFIG_DYNAMIC_EVENTS=y
24    CONFIG_PROBE_EVENTS=y
25    CONFIG_SYNTH_EVENTS=y
26    CONFIG_HIST_TRIGGERS=y
```

Code 4: Kernel Flags for Vsocks and Tracing

# Trace-cmd

- sudo trace-cmd record -e all -A @3:823 -name Salamander4 -e all

- sudo trace-cmd record -e kvm:kvm_entry -e kvm:kvm_exit -A @3:823 -name Salamander4 -e all

- sudo trace-cmd record -e kvm -e sched -e irq -e -A @3:823 -name Salamander4 -e all
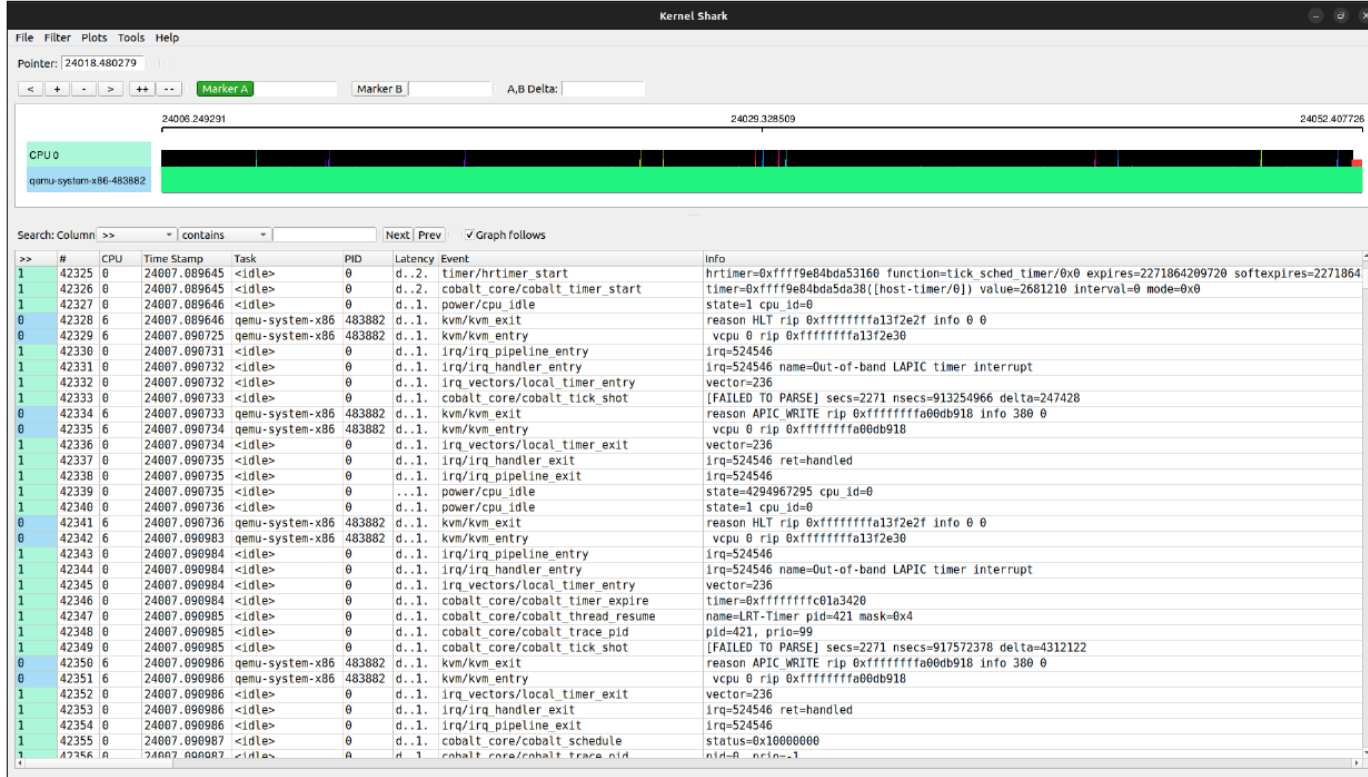
# Kernelshark



Figure 10: Guest vCPUs plotted on top of Host Threads in KernelShark