Open Source in Industry:
Trouble shooting of real-time Linux

Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3

Determination of the real-time properties of a Linux system
Presentation of the OSADL QA Farm
Some information on today’s sessions

- Please provide feedback on Legal HOT using the online form
  - Use the quick link `osadl.org/FB` (FeedBack), same as `osadl.org/?id=3325`
- You may ask questions during the session to be answered online, if possible
  - The quick link URL is `osadl.org/AQ` (AskQuestion), same as `osadl.org/?id=3321`
- You may join an online discussion on all topics of today at 4 pm
  - The quick link URL is `osadl.org/OD` (OnlineDiscussion), same as `jitsi.osadl.org`
  - Meeting name `OSADLTechnicalHOT`
  - Username and password will be displayed here after the last presentation

(We will show this slide again at the end of this session)
Issues leading to system latency

1979, e.g. Motorola MC68000 @ 8 MHz
   600 Dhrystones

x 20,000

2009, e.g. Intel Core 2 Duo @ 3 GHz
   12,000,000 Dhrystones
## Peak vs. worst-case performance

<table>
<thead>
<tr>
<th></th>
<th>1979</th>
<th>2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak performance (e.g. Dhrystones)</td>
<td>600</td>
<td>12,000,000</td>
</tr>
<tr>
<td>Factor</td>
<td>1</td>
<td>20,000</td>
</tr>
<tr>
<td>Moore's Law [2^{((2009-1979)/1.5)}]</td>
<td>1</td>
<td>(~1.048.576)]</td>
</tr>
<tr>
<td>Worst-case performance (e.g. signal latency)</td>
<td>(~400) µs</td>
<td>20 µs</td>
</tr>
<tr>
<td>1/Factor</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>
1979: Software issues related to system latency

- RTOSes in their early stage
- No thread libraries
- Limited IPC capabilities
- Little RT knowledge
- Many unresolved bugs
- Assembly language

1979, e.g. Motorola
MC68000 @ 8 MHz
600 Dhrystones

2009, e.g. Intel
Core 2 Duo @ 3 GHz
12,000,000 Dhrystones
1979: Hardware issues related to system latency

- **Software**
  - Single-core processor
  - No caches
  - One IRQ per device
  - Vectored IRQs, VBR
  - Fixed execution time per instruction
  - No microcode patches
  - Delayed DTACK

- **Hardware**
  - 1979, e.g. Motorola
    - MC68000 @ 8 MHz
    - 600 Dhrystones
  - 2009, e.g. Intel
    - Core 2 Duo @ 3 GHz
    - 12,000,000 Dhrystones
2009: Software issues related to system latency

- Mature RTOS
- Mature IPC mechanisms
- Thread libraries
- Optimized kernel code
  - Kernel profiling
  - Kernel tracing
- Better RT knowledge
- Still unresolved bugs

**1979**, e.g. Motorola
MC68000 @ 8 MHz
600 Dhrystones

**2009**, e.g. Intel
Core 2 Duo @ 3 GHz
12,000,000 Dhrystones
2009: Hardware issues related to system latency

**1979**, e.g. Motorola
MC68000 @ 8 MHz
600 Dhrystones

**2009**, e.g. Intel
Core 2 Duo @ 3 GHz
12,000,000 Dhrystones

- Several levels of cache
- Mixed caches (D/I/S)
- Shared IRQs
- Arbitrated busses
- Multi-processing
- SMI interference
- Microcode patches
- Very high peak performance
A total of 18 requests

Software
N=1

Hardware
N=17
What is the impact of these findings on path analysis?

A total of 18 requests

N=17

Software

N=1

Hardware
Path analysis: 1979 vs. 2009

```
  i = dram[0];
i++;
dram[0] = i;
```

1979, e.g. Motorola
MC68000 @ 8 MHz
600 Dhrystones

2009, e.g. Intel
Core 2 Duo @ 3 GHz
12,000,000 Dhrystones
Path analysis: 1979 vs. 2009

1979

- `movea.l #dram,a0`
- `move.l (a0),d0`
- `add.l #1,d0`
- `move.l d0,(a0)`

Load instruction from memory and execute it.
Duration = 56 clock cycles

2009

- `move dram,eax`
- `mov eax,-4(ebp)`
- `addl $1,-4(ebp)`
- `mov -4(ebp),eax`
- `mov eax,dram`

1979, e.g. Motorola
MC68000 @ 8 MHz
600 Dhrystones

2009, e.g. Intel
Core 2 Duo @ 3 GHz
12,000,000 Dhrystones
Path analysis: 1979 vs. 2009

2009

```
movea.l #dram,a0
move.l   (a0),d0
add.l    #1,d0
move.l   d0,(a0)
```

Load instruction from cache and execute it. Duration = ?

- Instruction not in cache/no free cache lines
- Data not in cache/no free cache lines
- System Management Interrupt
- Instruction may be emulated (microcode patch)

2009, e.g. Intel
Core 2 Duo @ 3 GHz
12,000,000 Dhrystones

1979, e.g. Motorola
MC68000 @ 8 MHz
600 Dhrystones

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
Path analysis

Path analysis

- Generally accepted verification procedure
- Source code normally required
- Difficult to do in modern high-performance processors
- Required processor data often not disclosed
- Expensive procedure
- Normally not done by users
- Result of path analysis often not publicly available
- May need to be checked against empirical latency testing
Path analysis vs. latency testing

**Path analysis**
- Generally accepted verification procedure
- Source code normally required
- Difficult to do in modern high-performance processors
- Required processor data often not disclosed
- Expensive procedure
- Normally not done by users
- Result of path analysis often not publicly available
- May need to be checked against empirical latency testing

**Latency testing**
- Not considered a valid “verification”
- Source code not required
- System complexity irrelevant
- Easy procedure
- Can be done by everybody
## Path analysis vs. latency testing

<table>
<thead>
<tr>
<th>Path analysis</th>
<th>Latency testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generally accepted verification procedure</td>
<td>Not considered a valid “verification”</td>
</tr>
<tr>
<td>Source code normally required</td>
<td>Source code not required</td>
</tr>
<tr>
<td>Difficult to do in modern high-performance processors</td>
<td>System complexity irrelevant</td>
</tr>
<tr>
<td>Required processor data often not disclosed</td>
<td>Easy procedure</td>
</tr>
<tr>
<td>Expensive procedure</td>
<td>Can be done by everybody</td>
</tr>
<tr>
<td>Normally not done by users</td>
<td></td>
</tr>
<tr>
<td>Result of path analysis often not publicly available</td>
<td></td>
</tr>
<tr>
<td>May need to be checked against empirical latency testing</td>
<td>Let's do it!</td>
</tr>
</tbody>
</table>
Four levels of latency tests

External measurement with simulation
OSADL's “Latency-Box”

Internal latency recording
Built-in kernel latency histograms

Internal measurement with simulation
Cyclictest

Real-world internal measurement
Application

CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_SWITCHTIME_HIST=y

# cyclictest -a -t -n -p99

# <application>
Four levels of latency tests

External measurement with simulation
OSADL’s „Latency-Box“

Internal latency recording
Built-in kernel latency histograms

Internal measurement with simulation
Cyclicstest

Real-world internal measurement
Application

CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_SWITCHTIME_HIST=y

# cyclicstest -a -t -n -p99

# <application>
Signal path to be monitored

External event, e.g. from a light barrier

- Gate latency: 3
- CPU IRQ latency: 3
- Interrupt service routine: 9
- Total latency or preemption latency: 30
- Scheduling, context switch: 15

Wakeup application in user space
OSADL's „Latency Box“

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
OSADL's „Latency Box“ - Specification

PowerPC 750FX@600MHz
64 MB SDRAM on SODIMM, 16 MB Flash-EPROM
10/100 Mb/s Network
2 serial channels RS232 and RS485
2 TTL Outputs, 4 TTL Inputs
4 Status LEDs
On-board FPGA
OSADL's „Latency Box“ connected to a CPU board

PowerPC 750FX@600MHz
64 MB SDRAM on SODIMM, 16 MB Flash-EPROM
10/100 Mb/s Network
2 serial channels RS232 and RS485
2 TTL Outputs, 4 TTL Inputs
4 Status LEDs
On-board FPGA

Interrupt service routine
Scheduling, context switch
OSADL's „Latency Box“ data transfer

| Line #1 | 0 | (No latency recording below 1 µs duration) |
| Line #11 | 76 | (A total of 76 latency values between 10 and 11 µs duration) |
|         | 2238 |
|         | 8800 |
|         | 20027 | (Most frequently observer latency values between 13 and 14 µs duration) |
|         | 18433 |
|         | 430 |
|         | 25 |
|         | 14 |
|         | [...] |
| Line #1000 | 0 | (No overflow) |
OSADL's „Latency Box“ - data plot

OSADL Latency Box

Number of data points

Max. latency 35 µs

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
OSADL standard “latency plot” (RT system)
OSADL standard “latency plot” (slow RT system)
OSADL standard “latency plot” (non-RT system)
Test: „Potential latency“ vs. „Effective latency“

Find appropriate measurement parameters

# cyclictest -m -n -Sp90 -i100 -d0
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 10.43 6.56 3.38 2/1454 4126098

<table>
<thead>
<tr>
<th>T</th>
<th>P:99 I:100 C:5154828 Min:</th>
<th>3 Act:</th>
<th>4 Avg:</th>
<th>6 Max:</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(4122431)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(4122432)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(4122433)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(4122434)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(4122435)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(4122436)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(4122437)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>(4122438)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(4122439)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>(4122440)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>(4122441)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>(4122442)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
„Potential latency“ vs. „Effective latency“

Measurement

Trigger interval 200 µs

Latency ("Potential latency")

- Not detected
- Partially detected
- Completely detected

"Effective latency"
“Potential latency” vs. “Effective latency”

Measurement

Trigger interval 200 µs

Latency (“Potential latency”)

Not detected

Partially detected

Completely detected

But would have been registered!

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
Four levels of latency tests

External measurement with simulation
OSADL's „Latency-Box“

Internal latency recording
Built-in kernel latency histograms

Internal measurement with simulation
Cyclicstest

Real-world internal measurement
Application

CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_SWITCHTIME_HIST=y

# cyclicstest -a -t -n -p99
# <application>
"Potential latency" vs. "Effective latency"

Trigger interval 200 µs

Latency ("Potential latency")

Not detected
Partially detected
Completely detected

But would have been registered!
Internal recording of potential latencies

- Preemption off
- Interrupts off
- Preemption and interrupts off

Duration of critical section
Internal recording of effective latencies

- Wakeup time
- Context switch

Recording of execution time

Start Recording  Stop Recording  Start Recording  Stop Recording  Start Recording  Stop Recording
Internal recording of effective latencies, sections

Restarting a waiting application by timer expiration

Programmed  Effective  Insertion into the run queue  Context switch
Internal recording of effective latencies, variables

Restarting a waiting application by timer expiration

Programmed Effective Insertion into the run queue Context switch
missed_timer_offsets wakeup switchtime
timerandwakeup

timerwakeupswitch

Debug directory names
Four levels of latency tests

External measurement with simulation
OSADL's "Latency-Box"

Internal continuous recording
Built-in kernel latency histograms

Internal measurement with simulation
Cyclictest

Real-world internal measurement
Application

CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_SWITCHTIME_HIST=y

# cyclictest -a -t -n -p99

# <application>
CyclicTest - Principle

- cyclicTest
  - Master process
  - T1
  - T2
  - TN

- cyclicTest
  - Meas. Thread

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
Cyclicetest: Command line parameters

```bash
# cyclictest -a -t -n -p99 -i100 -d50
560.44 586.11 606.12 211/1160 3727
T: 0 (18617) P:99 I:100 C:1,011,846,111 Min: 2 Act: 4 Avg: 5 Max: 39
```

- **-a** **PROC** *Affinity*. Run all threads on processor number **PROC**. If **PROC** is not specified, run thread #N on processor #N.

- **-t** **NUM** *Threads*. Create **NUM** test threads (default is 1). If **NUM** is not specified, **NUM** is set to the number of available CPUs.

- **-n** *Nanosleep*. Run the tests with **clock_nanosleep()**. This is the standard and should always be used.

- **-p99** *Priority*. Set the priority of the first thread. The given priority is assigned to the first test thread. Each further thread receives the priority reduced by the number of the thread.

- **-i100** *Interval*. Repetition interval of the first thread in μs (default is 1000 μs).

- **-d50** *Delay of additional threads*. Set the distance of thread intervals in μs (default is 500 μs). When cyclicetest is called with the -t option and more than a single thread is created, then this distance value is added to the interval of the threads.
Why are we testing computer boards and systems?

- Use as release testing for OSADL's „Latest Stable“ Linux real-time kernel
- Provide selection criteria for automation hardware
- Generate availability and stability data of individual systems
- „Freeze and grow“
- Generate reliable data for certification purpose (e.g. real-time)
„Freeze and grow“

Freeze release for production purpose

Grow under test conditions

Possible upgrade problem (documented or fixed)

2.6.31.12-rt21

3.0.0-rt1

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
OSADL QA Farm osadl.org/QA (1)

OSADL Test Rack

- Eight individual tablets
- Power supply 220 V, Ethernet, RS232
- 10/100/1000 Mb/s Switch with port mirroring
- Power distribution unit with power monitoring for every tablet
- Remote power switch for every tablet
- Serial network adapter for every tablet
- KVM switch (optional) for every tablet
- One central server per rack
OSADL QA Farm osadl.org/QA (2)

Mounting the individual systems on specially designed removable tablets
OSADL QA Farm osadl.org/QA (3)

Cloud-based communication between test systems, data collectors and admin systems

Test center #1

Test center #2

Data collector

Management, evaluation

VPN Channels
OSADL QA Farm osadl.org/QA (4)

Exhaustive and transparent documentation of every system

- Vendor, board
- BIOS version
- Distribution
- Kernel
- Kernel command line
- Command to generate latency plot histogram data
- CPU, interrupts, scaling governor, timer, RT features
- RAM, DIMMs
- PCI components
- BIOS analysis
- Kernel configuration, off-tree patches, script to reproduce kernel source tree
### Processor families/processors under test (selection)

<table>
<thead>
<tr>
<th>ARM</th>
<th>MIPS</th>
<th>PowerPC</th>
<th>Intel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcom</td>
<td>ICT</td>
<td>Freescale</td>
<td>Pentium @133 MHz, 32 bit</td>
</tr>
<tr>
<td>BCM2708 @700 MHz, 32 bit</td>
<td>Loongson 2F @800 MHz, 64 bit</td>
<td>i.MX27 @400 MHz, 32 bit</td>
<td>Atom D510 @1667 MHz, 64 bit</td>
</tr>
<tr>
<td>Freescale</td>
<td></td>
<td>i.MX35 @532 MHz, 32 bit</td>
<td>Atom N270 @1600 MHz, 32 bit</td>
</tr>
<tr>
<td>Broadcom</td>
<td></td>
<td>i.MX53 @886 MHz, 32 bit</td>
<td>Atom D2700 @2133 MHz, 64 bit</td>
</tr>
<tr>
<td>i.MX6 X4 @996 MHz, 32 bit</td>
<td>Loongson 2F @800 MHz, 64 bit</td>
<td>i.MX35 @400 MHz, 32 bit</td>
<td>Celeron M @1500 MHz, 32 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.MX35 @532 MHz, 32 bit</td>
<td>Pentium M @2300 MHz, 32 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.MX53 @886 MHz, 32 bit</td>
<td>Xeon @2000 MHz, 32 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.MX6 X4 @996 MHz, 32 bit</td>
<td>Core 2 Duo @2400 MHz, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Core 2 Quad @2400 MHz, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Opteron X32 @2100 MHz, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Phenom II X6 @3200 MHz, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AMD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>K6 3D, @333 MHz, 32 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LX-800 @500 MHz, 32 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Athlon XP 2000+, 32 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Athlon 64 2800+, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>G-Series T56N @1400 MHz, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Phenom II X6 @3200 MHz, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Opteron X32 @2100 MHz, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kaveri A10 7850k @3700 MHz, 64 bit</td>
</tr>
<tr>
<td>Texas Instruments</td>
<td></td>
<td>Freescale</td>
<td>C3 Samuel 2 @533 MHz, 32 bit</td>
</tr>
<tr>
<td>AM3517 @600 MHz, 32 bit</td>
<td>SheevaPlug @1200 MHz, 32 bit</td>
<td>i.MX27 @400 MHz, 32 bit</td>
<td>C7 @1000 MHz, 32 bit</td>
</tr>
<tr>
<td>OMAP3525 @720 MHz, 32 bit</td>
<td>SheevaPlug @1200 MHz, 32 bit</td>
<td>i.MX35 @532 MHz, 32 bit</td>
<td>Nano X2 L4050 @1400 MHz, 64 bit</td>
</tr>
<tr>
<td>OMAP4430 X2 @1008 MHz, 32 bit</td>
<td>SheevaPlug @1200 MHz, 32 bit</td>
<td>OMAP53 @532 MHz, 32 bit</td>
<td></td>
</tr>
<tr>
<td>OMAP4460 X2 @1200 MHz, 32 bit</td>
<td>SheevaPlug @1200 MHz, 32 bit</td>
<td>OMAP53 @532 MHz, 32 bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.MX53 @886 MHz, 32 bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.MX6 X4 @996 MHz, 32 bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Continuously determined variables (1)

### Benchmarks
- GL benchmark gltestperf
- UnixBench (multi-core)
- UnixBench (single-core)
- UnixBench 2D graphics performance

### Disk
- Disk IOs per device
- Disk latency per device
- Disk throughput per device
- Disk usage in percent
- Disk utilization per device
- File system mount-scheduled checks
- File system time-scheduled checks
- Filesystem usage (in bytes)
- Inode usage in percent
- IO Service time
- IOstat
- S.M.A.R.T values of every drive

### Network
- eth0 errors
- eth0 traffic
- Firewall Throughput
- HTTP loadtime of a page
- Netstat

### Real-time system
- 5-min max. timer and wakeup latency
- 5-min max. timer offsets
- 5-min max. wakeup latency
- RT Features

### Email
- Sendmail email traffic
- Sendmail email volumes
- Sendmail queued mails

### NFS
- NFS Client
- NFSv4 Client

### Processes
- Fork rate
- Number of threads
- Processes
- Processes priority
- VMstat

### Sensors
- Fans
- HDD temperature
- **Power consumption**
- Temperatures
Continuously determined variables (2)

**System**
- Available entropy
- C states
- CPU frequency
- CPU usage
- File table usage
- Individual interrupts
- Inode table usage
- Interrupts and context switches
- Kernel version
- Load average
- Logged in users
- Memory usage
- Split memory usage
- Application memory usage
- Swap in/out
- Uptime

**Virtual systems**
- Virtual domain block device I/O
- Virtual domain CPU time
- Virtual domain memory usage
- Virtual domain network I/O

**Time synchronization**
- NTP kernel PLL estimated error (secs)
- NTP kernel PLL frequency (ppm + 0)
- NTP kernel PLL offset (secs)
- NTP states
- NTP timing statistics for system peer
### CPU and graphics benchmarks

<table>
<thead>
<tr>
<th></th>
<th>Fastest (greenest)</th>
<th>Slowest (reddest)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r5s0</td>
<td>51053</td>
<td>2404.18:12</td>
</tr>
<tr>
<td>r9s1</td>
<td>53541</td>
<td>2404.18:11</td>
</tr>
<tr>
<td>r0s0</td>
<td>62655</td>
<td>2404.18:12</td>
</tr>
<tr>
<td>r8s2</td>
<td>62708</td>
<td>2404.18:10</td>
</tr>
<tr>
<td>r3s8</td>
<td>69034</td>
<td>2404.18:13</td>
</tr>
<tr>
<td>r7s2</td>
<td>89680</td>
<td>2404.06:14</td>
</tr>
<tr>
<td>r5s1</td>
<td>102987</td>
<td>2404.18:12</td>
</tr>
<tr>
<td>r0s2</td>
<td>105523</td>
<td>2404.18:11</td>
</tr>
<tr>
<td>r8s8</td>
<td>124787</td>
<td>2404.06:11</td>
</tr>
<tr>
<td>r0s3</td>
<td>149833</td>
<td>2404.18:11</td>
</tr>
<tr>
<td>r8s3</td>
<td>171306</td>
<td>2404.18:11</td>
</tr>
<tr>
<td>r4s6</td>
<td>180687</td>
<td>2404.18:11</td>
</tr>
<tr>
<td>r0s8</td>
<td>194089</td>
<td>2404.18:16</td>
</tr>
</tbody>
</table>

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
Four different time resolutions (e.g. temperatures)

Day

Week

Month

Year

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
Alert colors of warnings and alarms (Munin)

- rack1slot2.osadl.org [ benchmarks disk network nfs processes sendmail sensors system time ]
- rack1slot3.osadl.org [ benchmarks disk network nfs processes sendmail sensors system time ]
- rack1slot4.osadl.org [ benchmarks disk network nfs processes sendmail sensors system time ]
- rack1slot6.osadl.org [ benchmarks disk network processes sendmail system time ]
- rack1slot8.osadl.org [ benchmarks disk network nfs processes sendmail sensors system time ]
- rack2slot0.osadl.org [ benchmarks disk network nfs postfix processes sendmail sensors system time ]
- rack2slot2.osadl.org [ benchmarks disk network nfs processes system time ]
- rack2slot3.osadl.org [ benchmarks disk network nfs processes system time ]
- rack2slot5.osadl.org [ benchmarks disk network nfs processes system time ]
- rack2slot6.osadl.org [ benchmarks disk network nfs processes sendmail sensors system time ]
- rack2slot8.osadl.org [ benchmarks disk network nfs processes system time ]
- rack3slot0.osadl.org [ benchmarks disk network nfs processes sendmail sensors system time ]
- rack3slot1.osadl.org [ benchmarks disk network nfs processes sendmail sensors system time ]
- rack3slot2.osadl.org [ benchmarks disk network nfs processes sendmail sensors system time ]
- rack3slot3.osadl.org [ benchmarks disk memory network nfs processes sendmail sensors system time ]

---

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
# Event recording with Nagios (1)

## Service Overview For All Host Groups

### OSADL Test Racks (osadl-test-racks)

<table>
<thead>
<tr>
<th>Host</th>
<th>Status</th>
<th>Services</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ou-int.osadl.org</td>
<td>UP</td>
<td>1 OK</td>
<td></td>
</tr>
<tr>
<td>ou.osadl.org</td>
<td>UP</td>
<td>1 OK</td>
<td></td>
</tr>
<tr>
<td>rack0slot0.osadl.org</td>
<td>UP</td>
<td>10 OK</td>
<td></td>
</tr>
<tr>
<td>rack0slot1.osadl.org</td>
<td>UP</td>
<td>6 OK</td>
<td>3 WARNING</td>
</tr>
<tr>
<td>rack0slot2.osadl.org</td>
<td>UP</td>
<td>10 OK</td>
<td></td>
</tr>
<tr>
<td>rack0slot3.osadl.org</td>
<td>UP</td>
<td>9 OK</td>
<td>1 PENDING</td>
</tr>
<tr>
<td>rack0slot4.osadl.org</td>
<td>UP</td>
<td>9 OK</td>
<td></td>
</tr>
<tr>
<td>rack0slot5.osadl.org</td>
<td>UP</td>
<td>4 OK</td>
<td>1 CRITICAL</td>
</tr>
</tbody>
</table>

### Web-Alm Servers (web-alm-servers)

<table>
<thead>
<tr>
<th>Host</th>
<th>Status</th>
<th>Services</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>dns.web-alm.net</td>
<td>UP</td>
<td>1 OK</td>
<td></td>
</tr>
<tr>
<td>mail.web-alm.net</td>
<td>UP</td>
<td>1 OK</td>
<td></td>
</tr>
<tr>
<td>swiss.web-alm.net</td>
<td>UP</td>
<td>6 OK</td>
<td></td>
</tr>
<tr>
<td>toro.web-alm.net</td>
<td>UP</td>
<td>4 OK</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.osadl.org">www.osadl.org</a></td>
<td>UP</td>
<td>5 OK</td>
<td>1 PENDING</td>
</tr>
</tbody>
</table>

---

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
Event recording with Nagios (2)

Service alert histogram,
e.g. hour-of-the-day analysis of latency peaks in current year

Event History For Service ‘5-min max. timer and wakeup latency’ On Host ‘rack3slot7.osadl.org’
Tue Jan 1 00:00:00 2013 to Sun May 12 15:28:34 2013

<table>
<thead>
<tr>
<th>EVENT TYPE</th>
<th>MIN</th>
<th>MAX</th>
<th>SUM</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recovery (OK):</td>
<td>0</td>
<td>4</td>
<td>62</td>
<td>0.65</td>
</tr>
<tr>
<td>Warning:</td>
<td>0</td>
<td>1</td>
<td>12</td>
<td>0.12</td>
</tr>
<tr>
<td>Unknown:</td>
<td>0</td>
<td>3</td>
<td>44</td>
<td>0.46</td>
</tr>
<tr>
<td>Critical:</td>
<td>0</td>
<td>2</td>
<td>9</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Hour of the Day (15 minute increments)
Monitoring and benchmark schedule

- Accelerated graphics benchmark *gltestperf*
- CPU benchmark *UnixBench*

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00</td>
<td>Continuous latency monitoring using kernel built-in histograms</td>
</tr>
<tr>
<td>04:00</td>
<td>Latency determination using <em>cyclictest</em></td>
</tr>
<tr>
<td>08:00</td>
<td>Standardized network, disk and memory load</td>
</tr>
</tbody>
</table>
Example 1a: Memory leak diagnosis

Normal (no leak)
Example 1b: Memory leak diagnosis

- **System leak**
- **Application leak**

Troubleshooting of real-time Linux

Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3

Open Source Automation Development Lab (OSADL), Heidelberg
Example 2: Stable vs. instable system
Example 3a: Power management
Example 3b: Power management

CPU frequency scaling - by day

Electricity (power) at r0s8 - by day

Troubleshooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
Example 3c: Power management
Example 3d: Power management
Example 4a: Determinism

Latency plot with linear x scale and logarithmic y scale
Example 4b: Determinism

Standard OSADL plot (very low maximum latency)
Example 4c: Determinism

Standard OSADL plot (relatively high maximum latency)
Repetitive latency plots each of 100 million cycles (1)

Consecutive latency plots in a single combined display

System in rack #1, slot #3
Recording from 08.01.2011 until 25.04.2011
Repetitive latency plots each of 100 million cycles (2)

Consecutive latency plots in a single combined display

System in rack #1, slot #3
Recording from 08.01.2011 until 25.04.2011

Trouble shooting of real-time Linux
Technical Heidelberg OSADL Talks, April 29, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg
Example 5a: Real-time optimization

System in rack #1, slot #1
Recording from 08.01.2011 until 04.05.2011

Periods with prolonged maximum latency
Example 5b: Real-time optimization

System in rack #1, slot #1
Recording from 08.01.2011 until 04.05.2011

Periods with prolonged maximum latency: Enabled sleep states
Example 6: Real-time optimization

System in rack #4, slot #2
Recording from 08.01.2011 until 04.05.2011

Determinism (no outlier in more than 22 billion cycle)
Example 7: Long-term latency plot

Very short latency
Example 8: Long-term latency plot

Very short latency
No outlier in more than 130 billion cycles
Example 9: Long-term latency plot

System in rack #3, slot #7
Recording from 08.01.2011 until 03.07.2011

Sporadic outliers due to a DMA problem of the Ethernet controller
Example 10: Long-term latency plot

System in rack #2, slot #6
Recording from 08.01.2011 until 04.05.2011

Erroneous use of a non-real-time kernel
Example 11: Long-term latency plot

System in rack #2, slot #3
Recording from 08.01.2011 until 04.05.2011

Highest system priority assigned more than once per core
Four levels of latency tests

External measurement with simulation
OSADL's „Latency-Box“

Internal continuous recording
Built-in kernel latency histograms

Internal measurement with simulation
Cyclictest

Real-world internal measurement
Application

CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_PREEMPT_OFF_HIST=y

# cyclictest -a -t -n -p99
# <application>