Path analysis vs. empirical determination of a system's real-time capabilities: The crucial role of latency tests

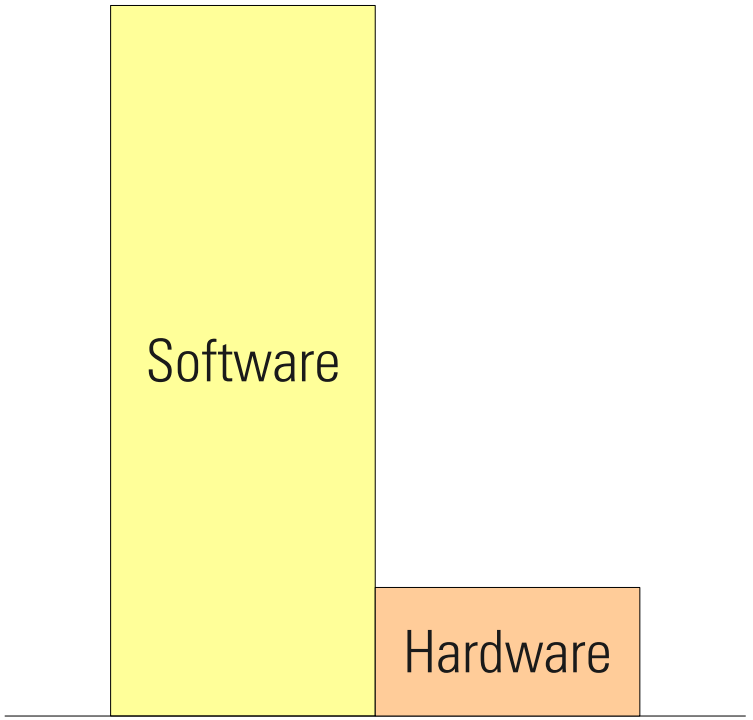
Carsten Emde

Open Source Automation Development Lab (OSADL) eG

Eleventh Real Time Linux Workshop
September 28 to 30, 2009, Dresden, Germany

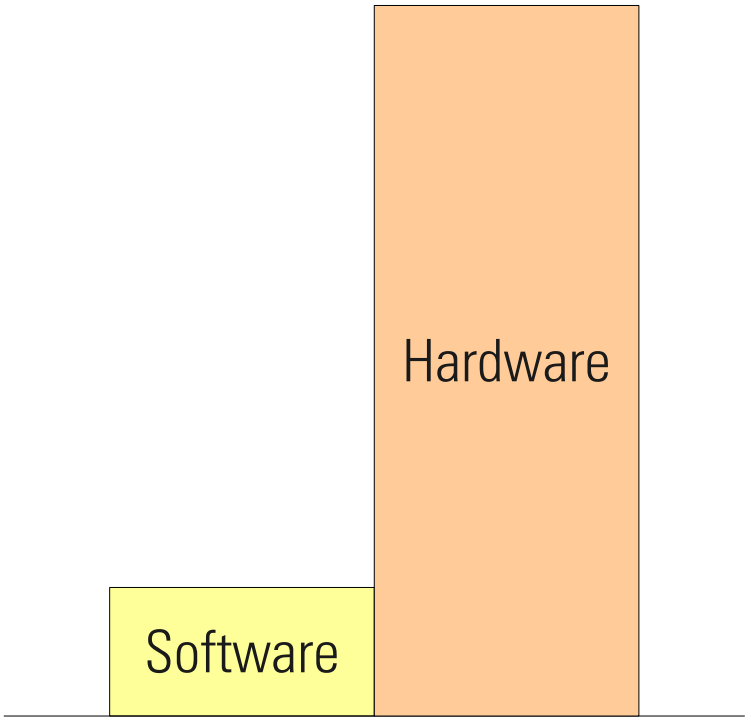
Open Source Automation Development Lab eG

Issues leading to system latency



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

x 20,000

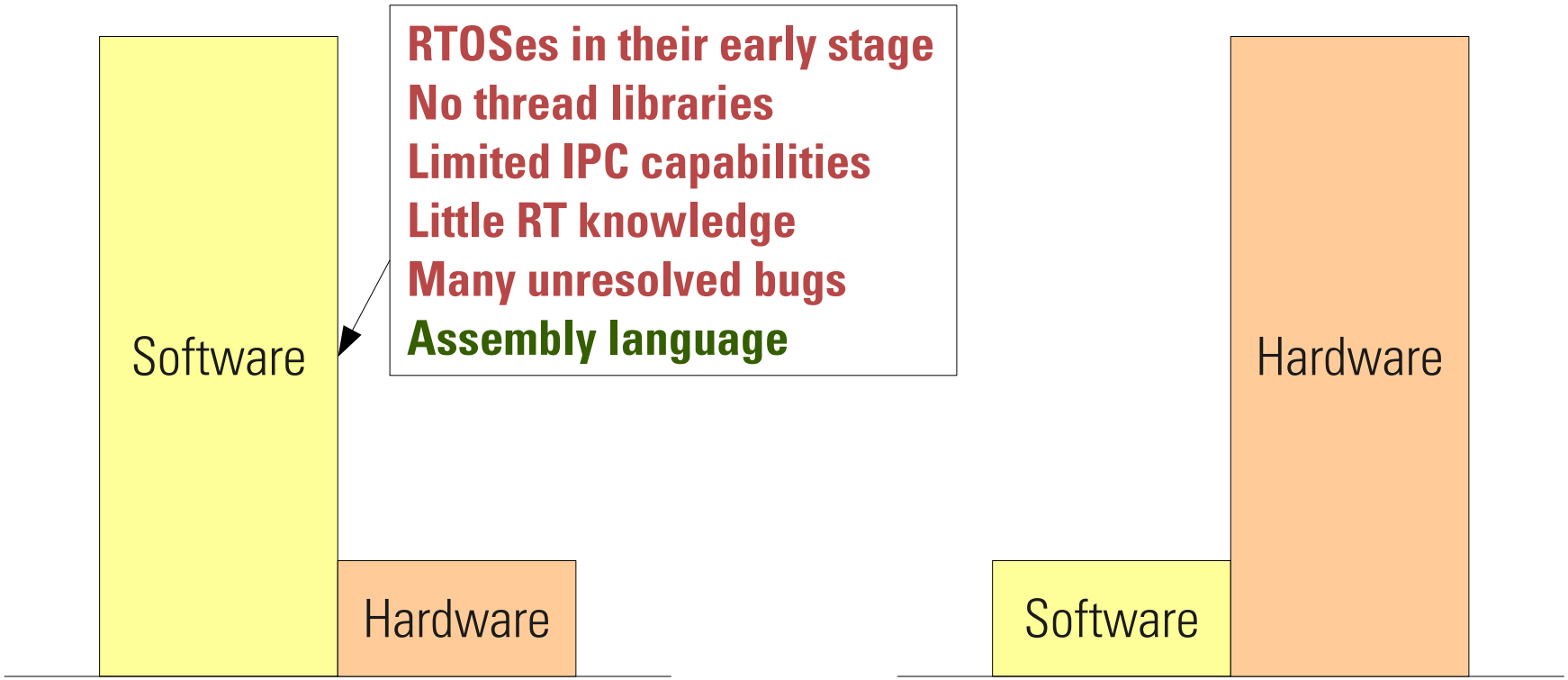


2009, e.g. Intel
Core 2 Duo @ 3 GHz
12,000,000 Dhrystones

Peak *vs.* worst-case performance

	1979	2009
Peak performance (e.g. Dhrystones)	600	12,000,000
Factor	1	20,000
Moore's Law [$2^{((2009-1979)/1.5)}$]	1	≈1,048,576
Worst-case performance (e.g. signal latency)	≈4,000 µs	20 µs
1/Factor	1	200

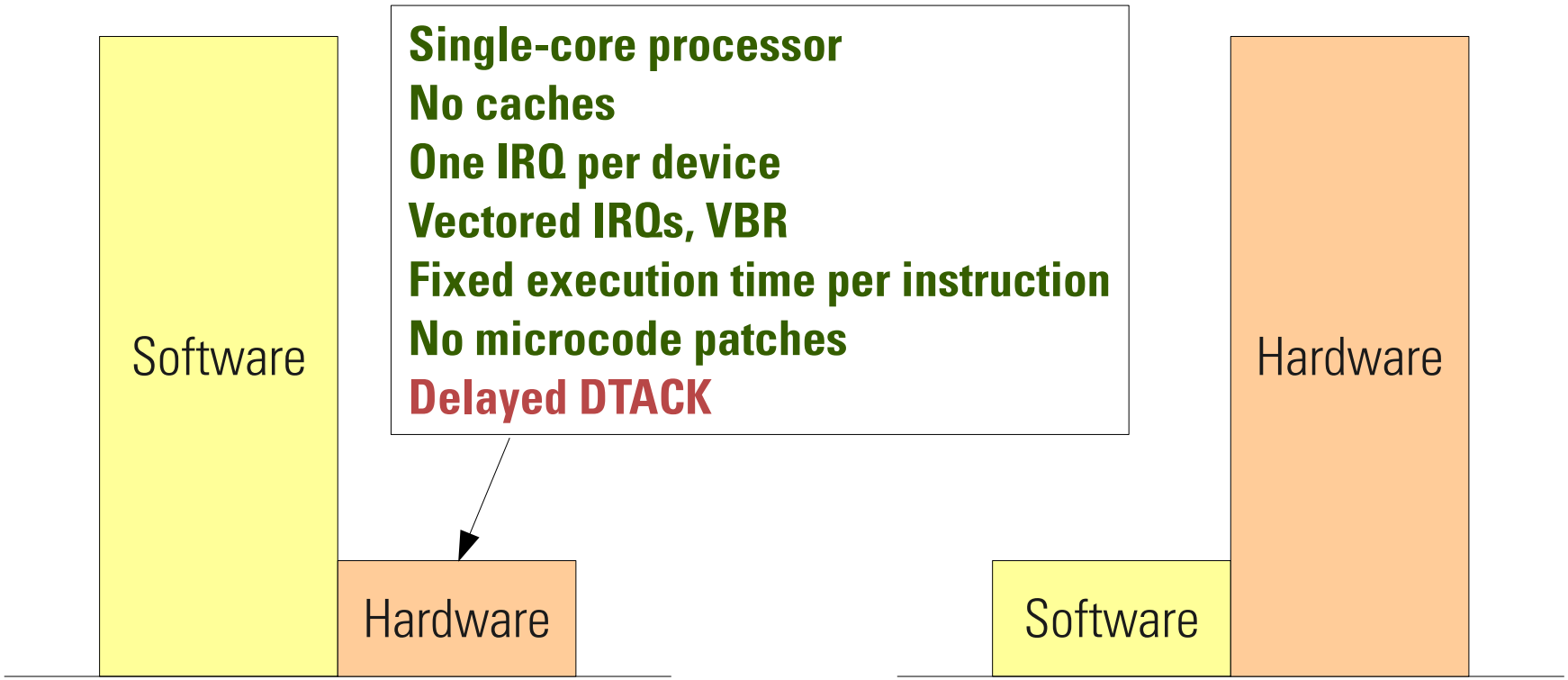
1979: Software issues related to system latency



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

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

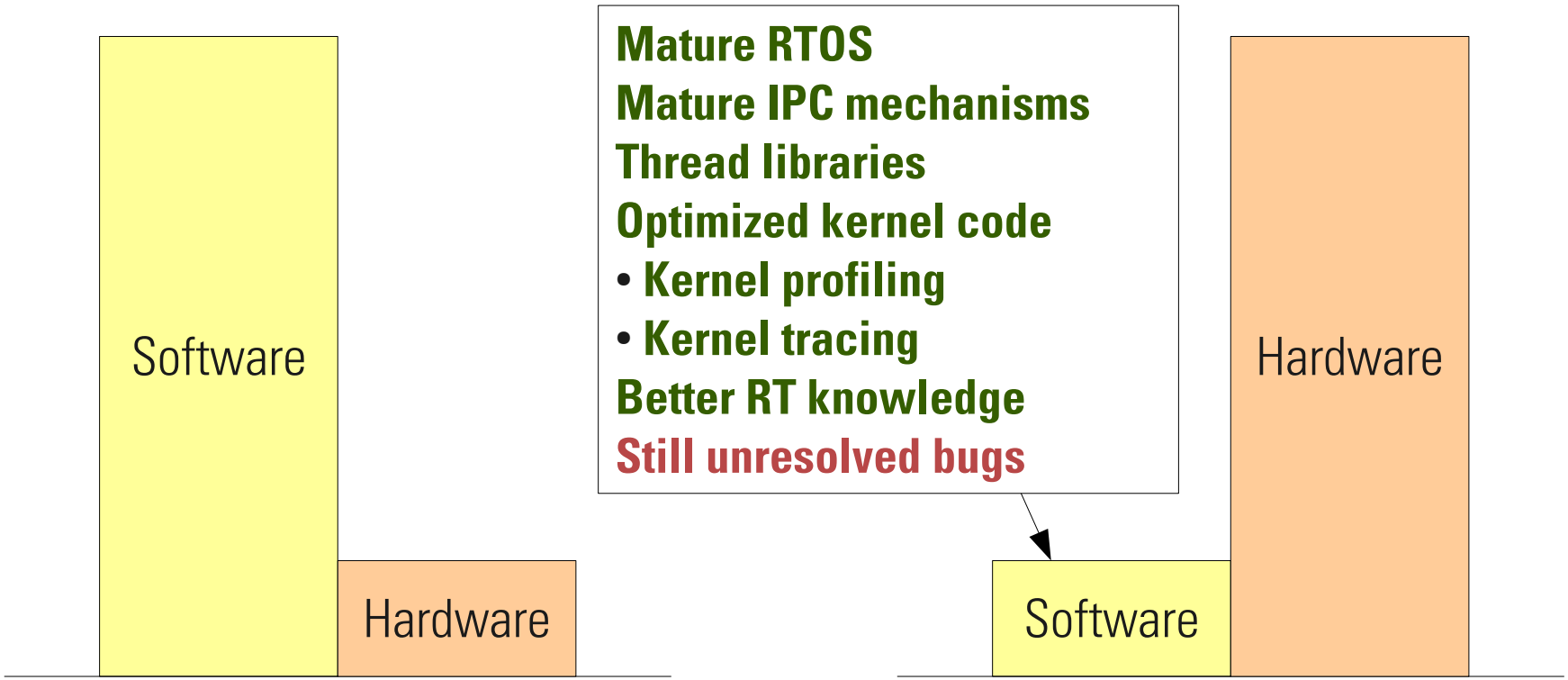
1979: Hardware issues related to system latency



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

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

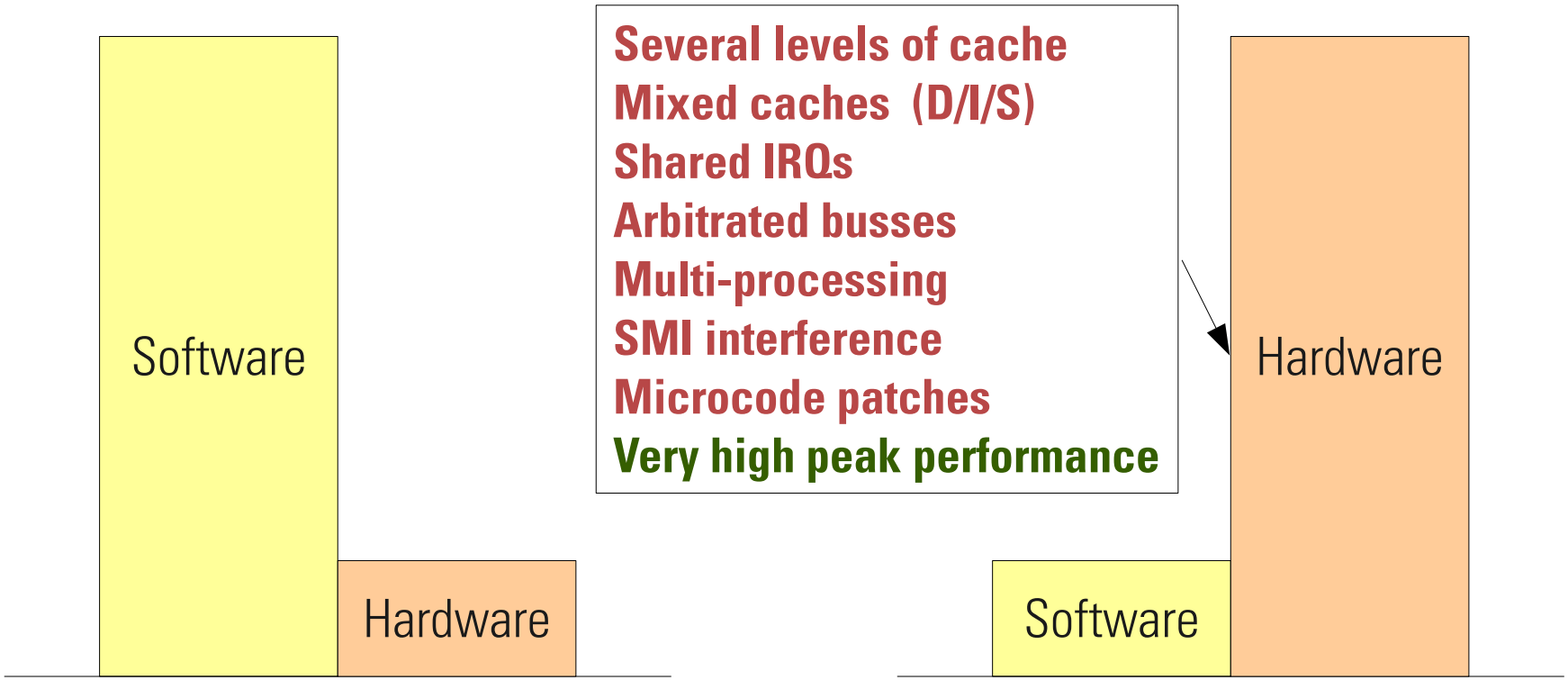
2009: Software issues related to system latency



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

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

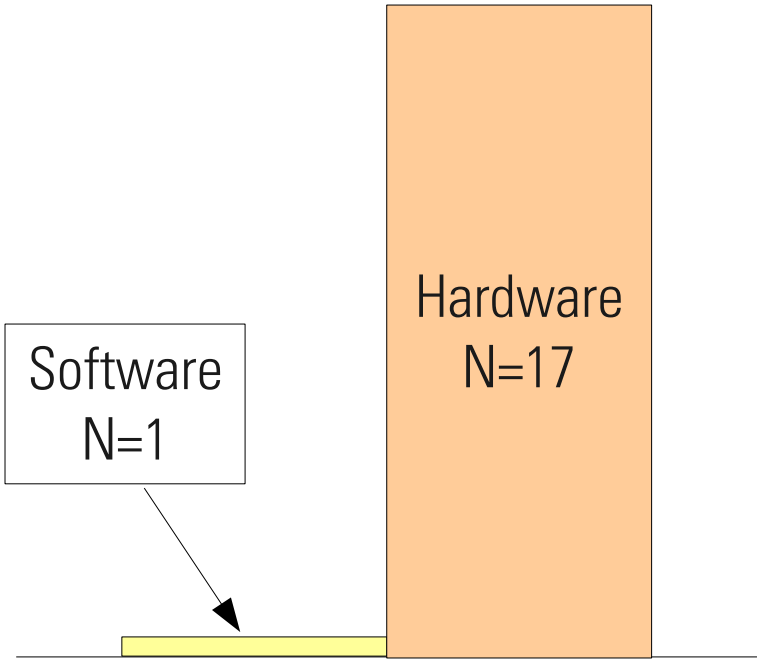
2009: Hardware issues related to system latency



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

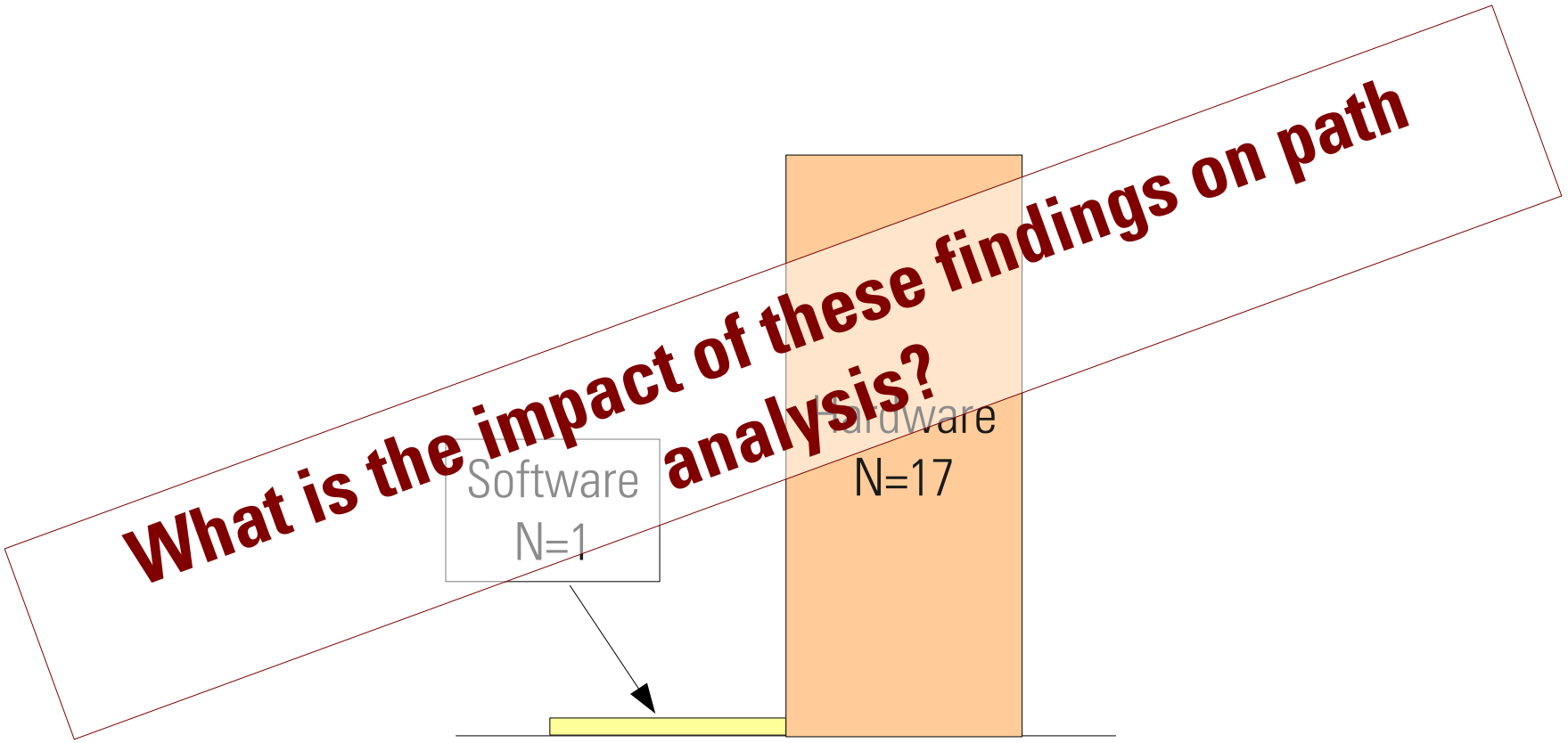
2009, e.g. Intel
Core 2 Duo @ 3 GHz
12,000,000 Dhrystones

latency-fighters@osadl.org



A total of 18 requests

latency-fighters@osadl.org



Path analysis: 1979 *vs.* 2009

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

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

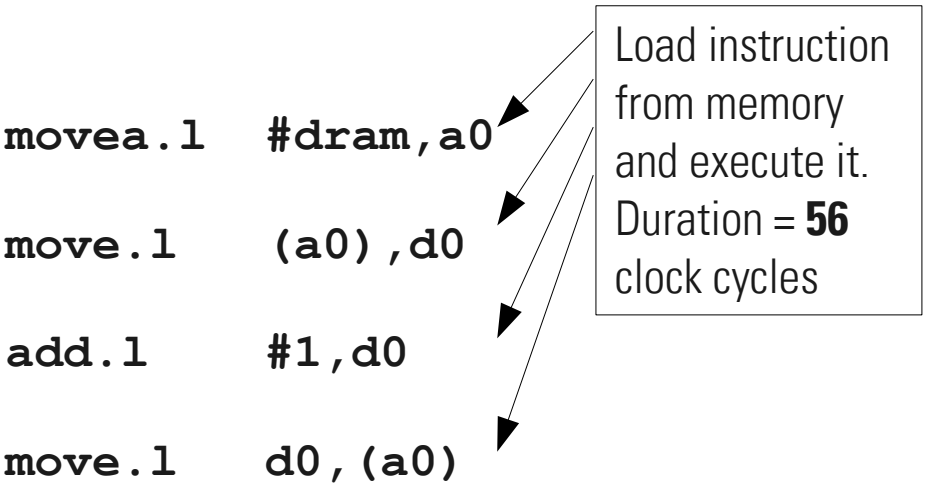
```
mov    dram,eax
mov    eax,-4(ebp)
addl   $1,-4(ebp)
mov    -4(ebp),eax
mov    eax,dram
```

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

1979



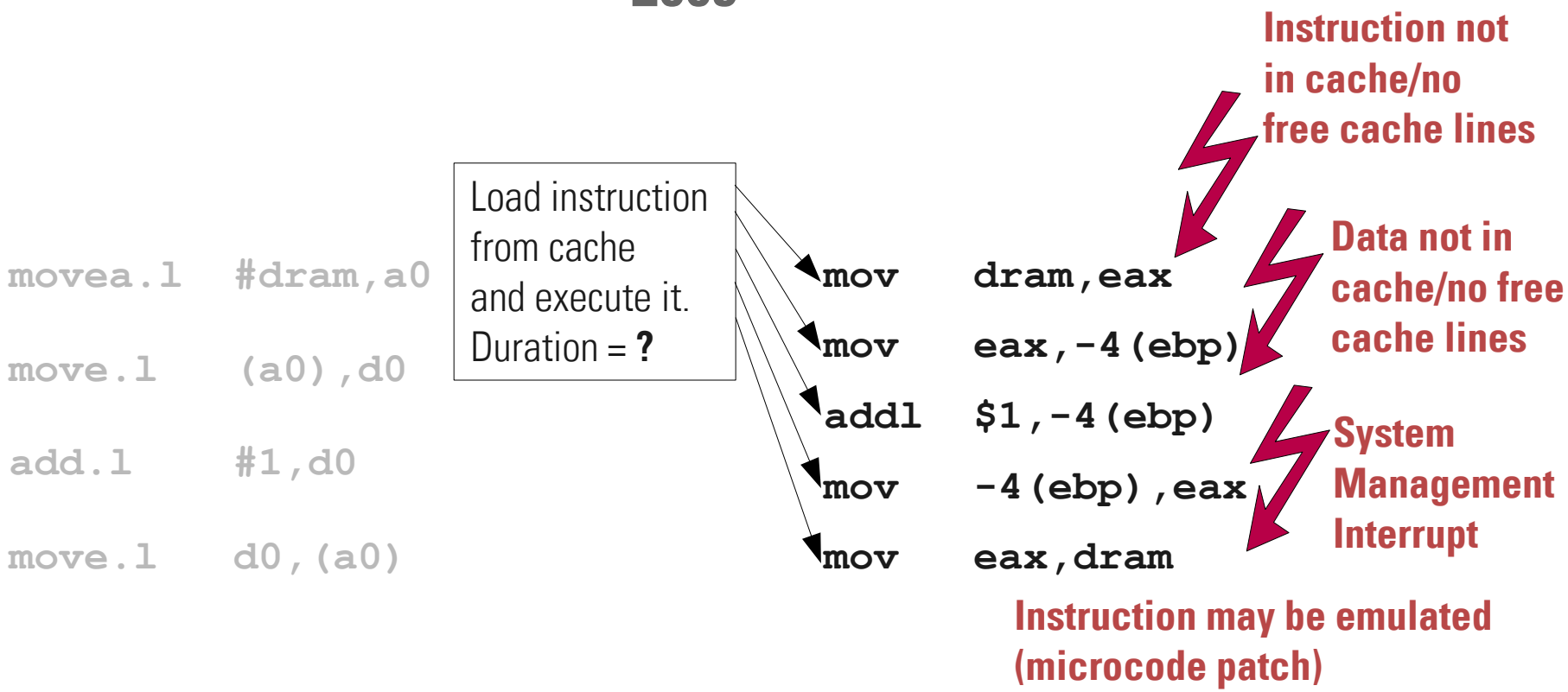

```
mov    dram,eax
mov    eax,-4(ebp)
addl   $1,-4(ebp)
mov    -4(ebp),eax
mov    eax,dram
```

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



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

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

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

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

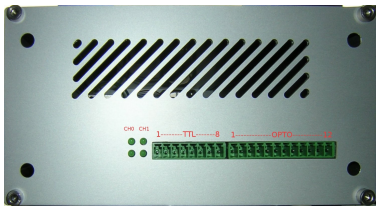
- Not considered a valid “verification”
- Source code not required
- System complexity irrelevant
- Easy procedure
- Can be done by everybody

Let's do it!

Four levels of latency tests

External measurement with simulation

OSADL's „Latency-Box“



Internal latency recording

Built-in kernel latency histograms

```
CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_PREEMPT_OFF_HIST=y
```

Internal measurement with simulation

Cyclictest

```
# cyclictest -a -t -n -p99
```

Real-world internal measurement

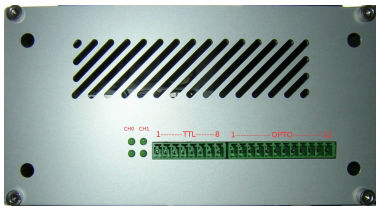
Application

```
# <application>
```

Four levels of latency tests

External measurement with simulation

OSADL's „Latency-Box“



Internal latency recording

Built-in kernel latency histograms

```
CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_PREEMPT_OFF_HIST=y
```

Internal measurement with simulation

Cyclictest

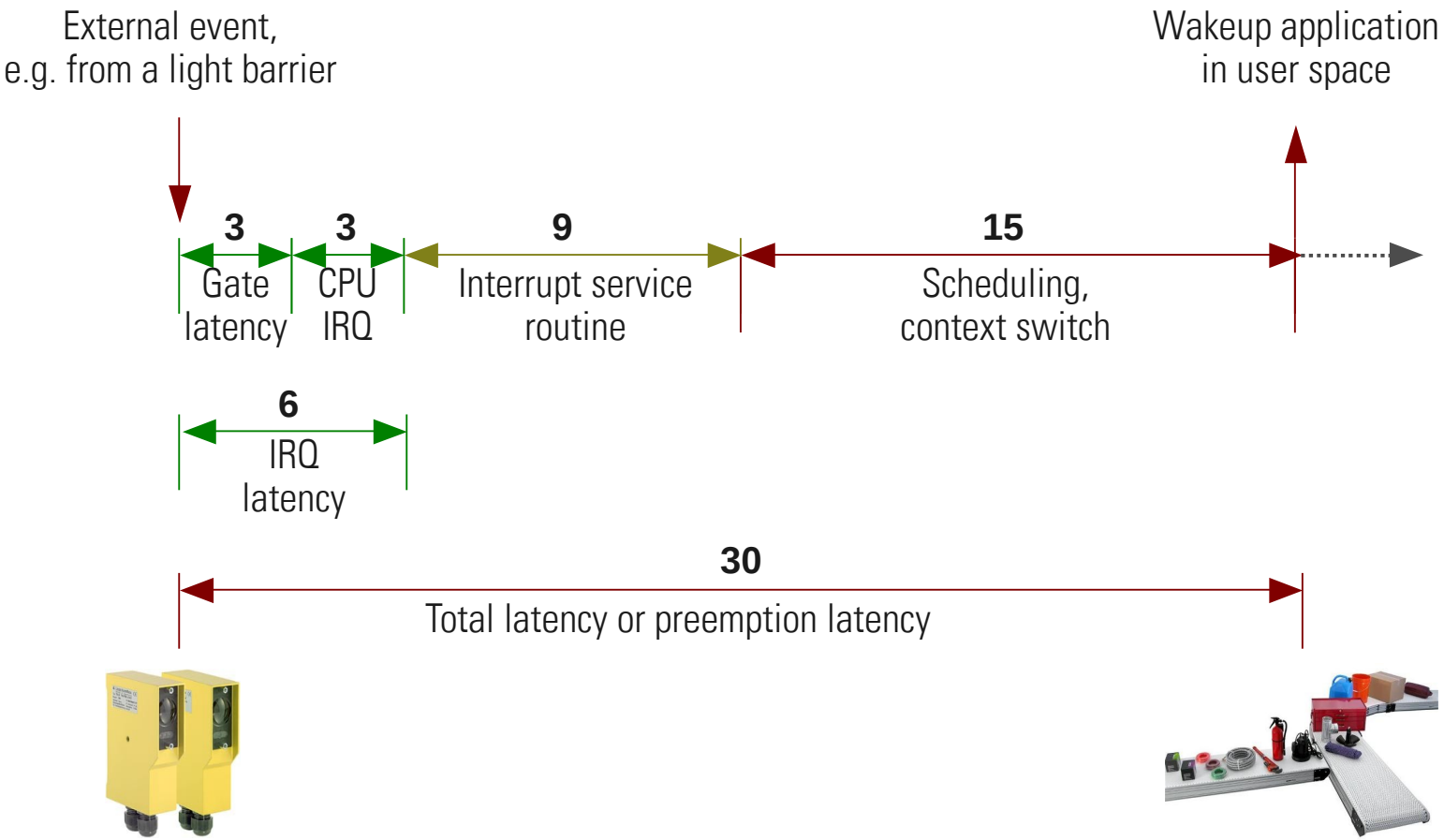
```
# cyclictest -a -t -n -p99
```

Real-world internal measurement

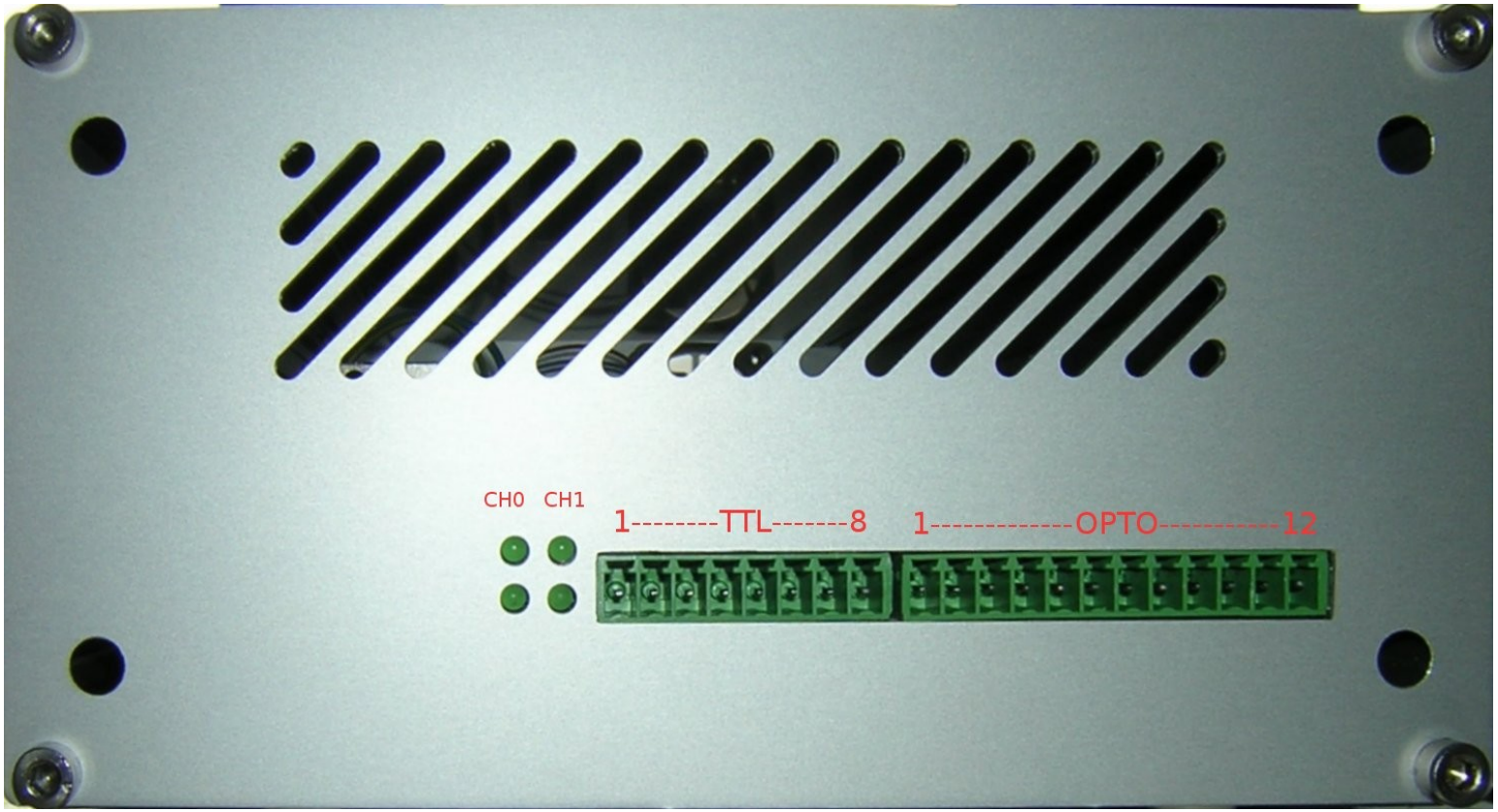
Application

```
# <application>
```

Signal path to be monitored

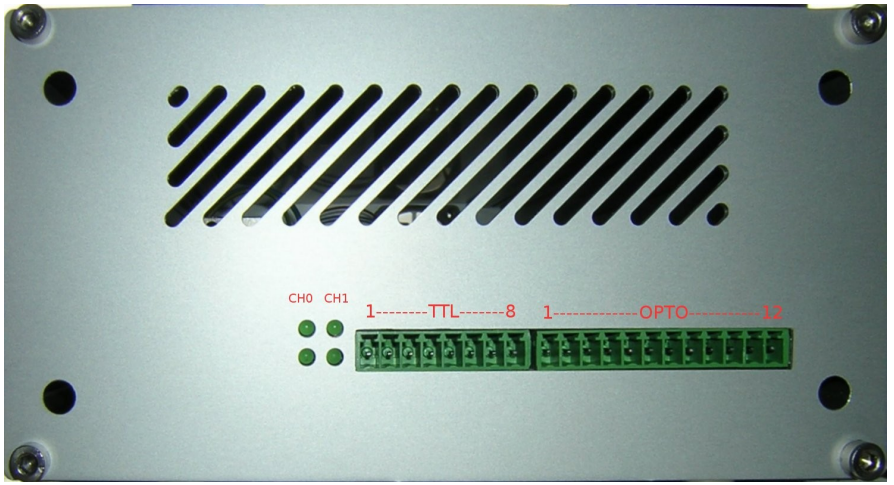


OSADL's „Latency Box“



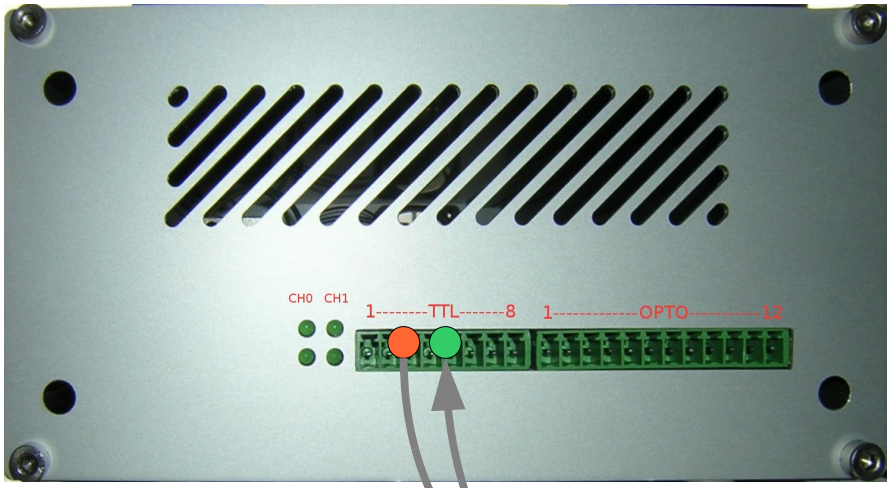
ELTEC systems

OSADL's „Latency Box“ - Specification

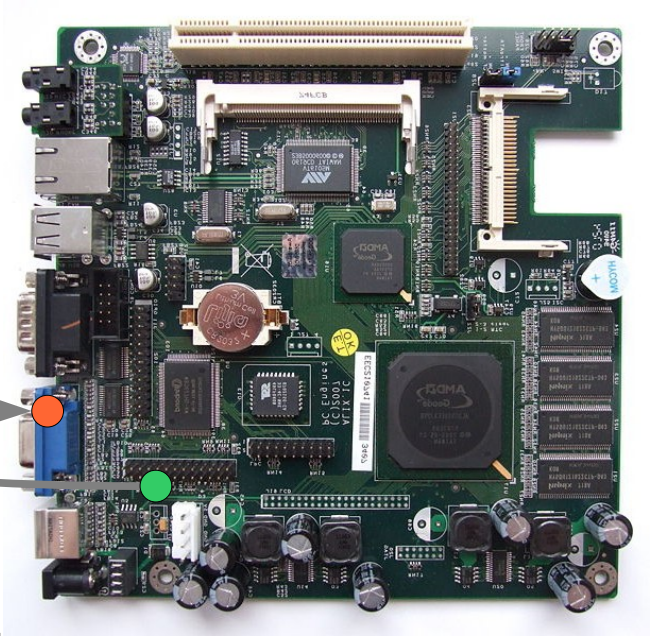


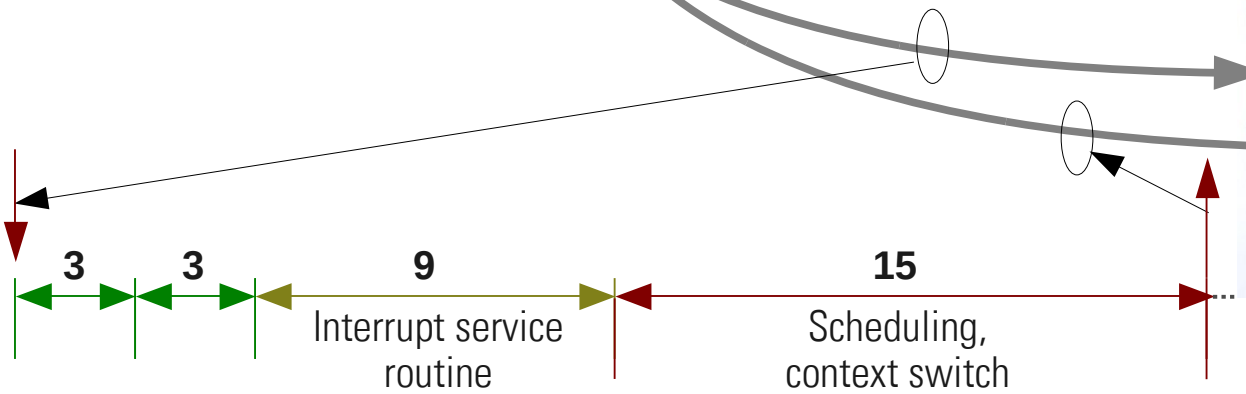
PowerPC 750FX@600MHz
64 MB SDRAM on SODIMM, 16 MB Flash-EPROM
10/100 Mb/s Network
2 serial channels RS232 and RS485
2 TTL Outputs, 4 TTL Inputs
4 Status LEDs
On-board FPGA

OSADL's „Latency Box“ connected to a CPU board



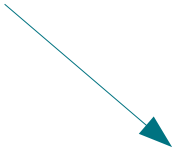
PowerPC 750FX@600MHz
64 MB SDRAM on SODIMM, 16 MB Flash-EPROM
10/100 Mb/s Network
2 serial channels RS232 and RS485
2 TTL Outputs, 4 TTL Inputs
4 Status LEDs
On-board FPGA





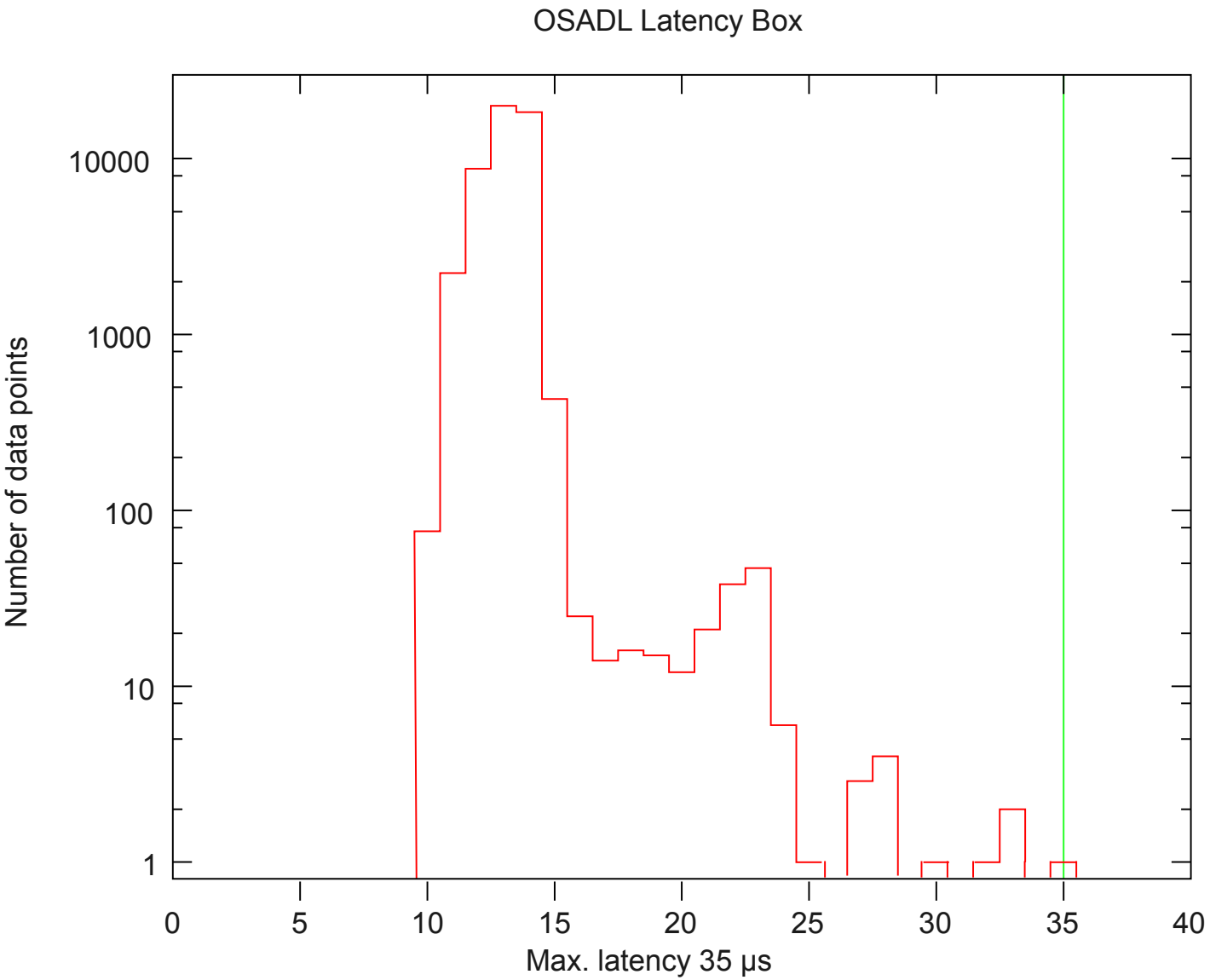
OSADL's „Latency Box“ data transfer

Histogram data

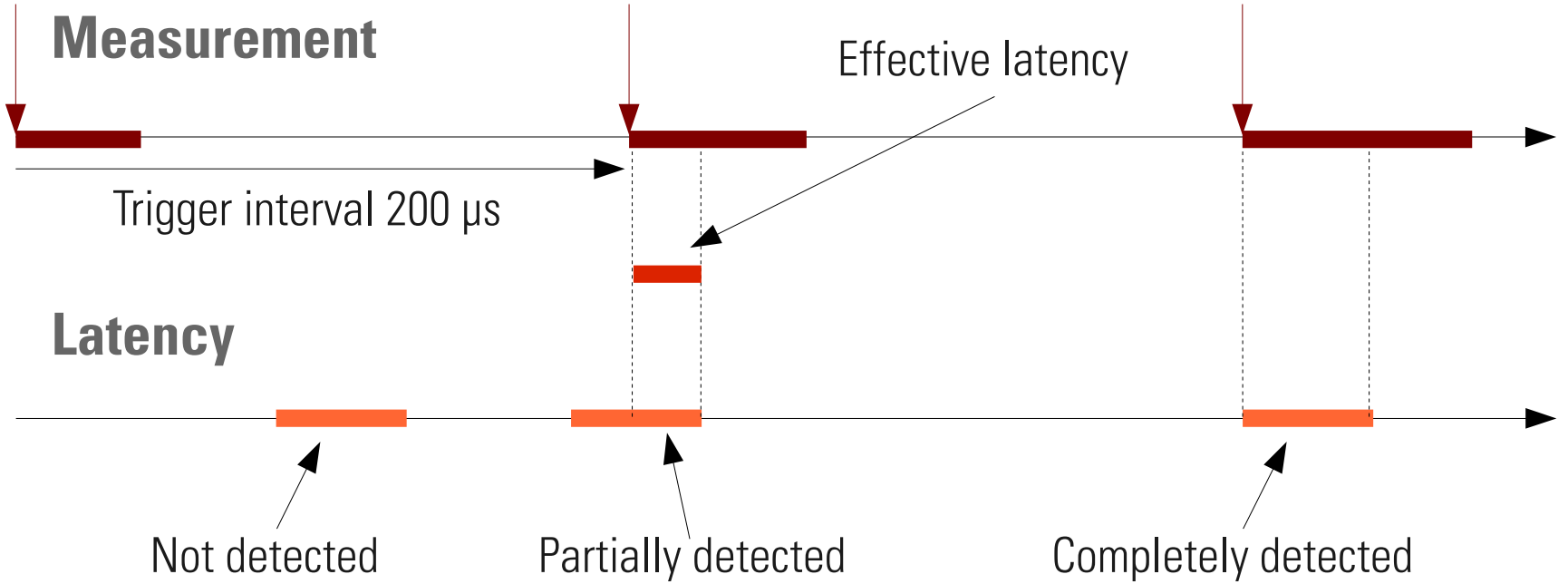


```
Line #1     0      (No latency recording below 1 µs duration)
            0
            0
            0
            0
            0
            0
            0
            0
            0
Line #11    76     (A total of 76 latency values between 10 and 11 µs duration)
            2238
            8800
            20027  (Most frequently observer latency values between 13 and 14 µs duration)
            18433
            430
            25
            14
            [..]
Line #1000  0      (No overflow)
```

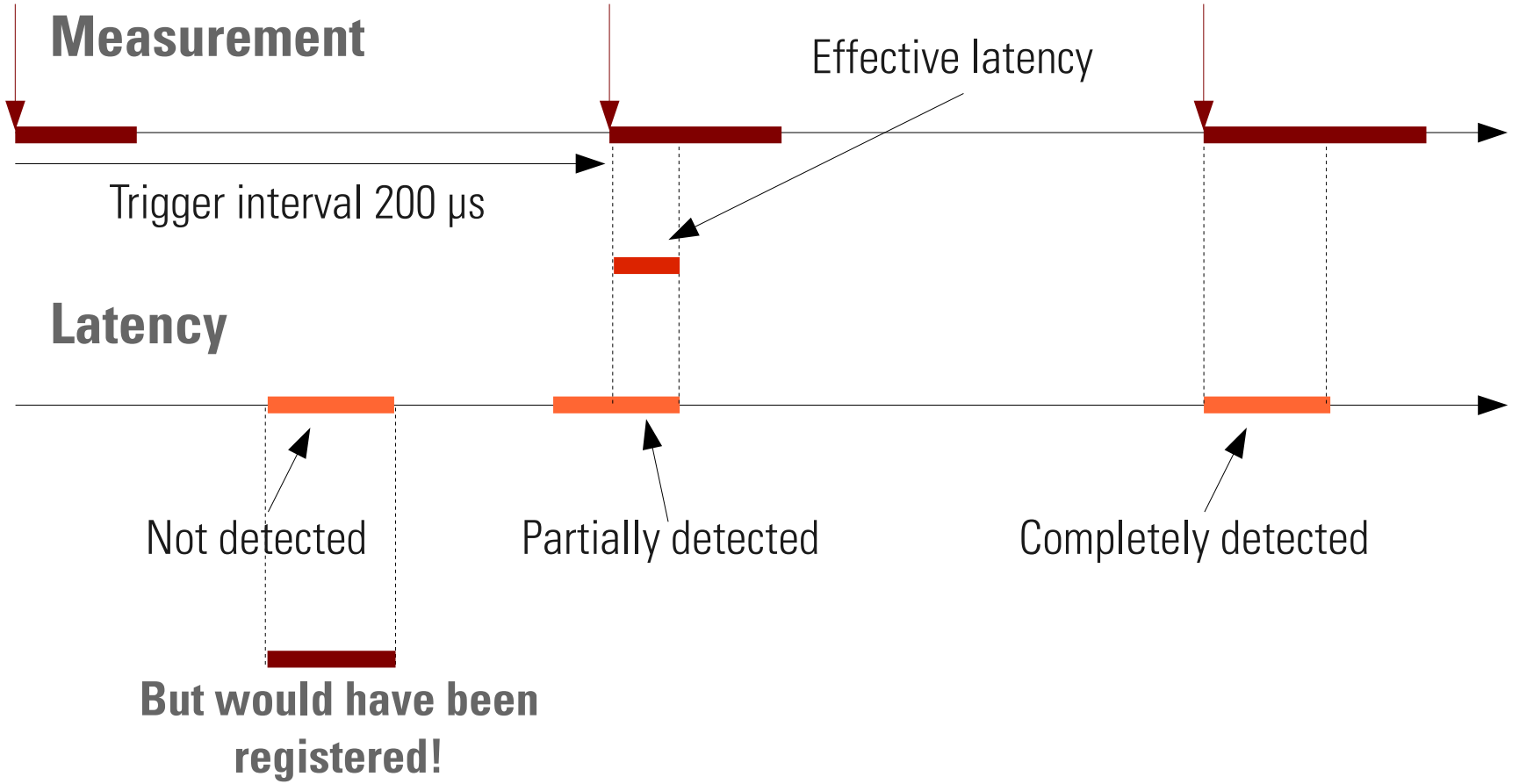
OSADL's „Latency Box“ - data plot



„Potential latency“ *vs.* „Effective latency“



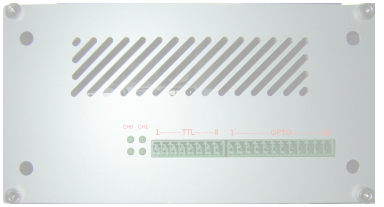
„Potential latency“ *vs.* „Effective latency“



Four levels of latency tests

External measurement with simulation

OSADL's „Latency-Box“



Internal latency recording

Built-in kernel latency histograms

```
CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_PREEMPT_OFF_HIST=y
```

Internal measurement with simulation

Cyclictest

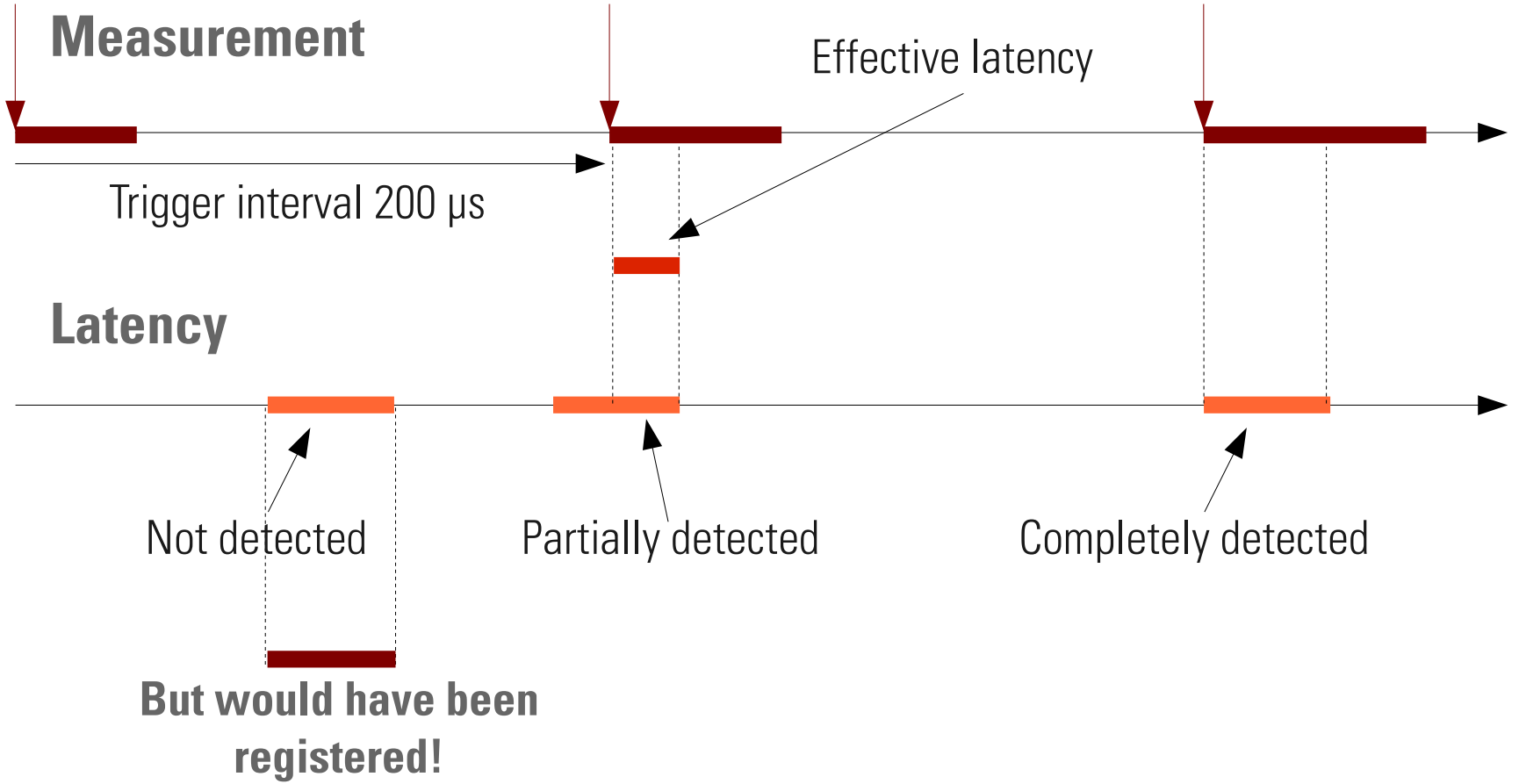
```
# cyclictest -a -t -n -p99
```

Real-world internal measurement

Application

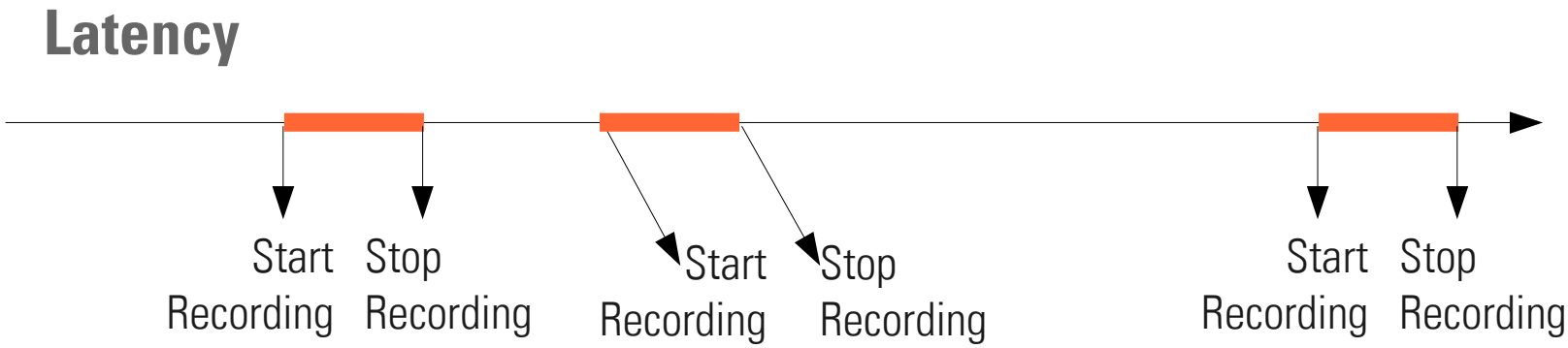
```
# <application>
```

„Potential latency“ *vs.* „Effective latency“



Internal recording of potential latencies

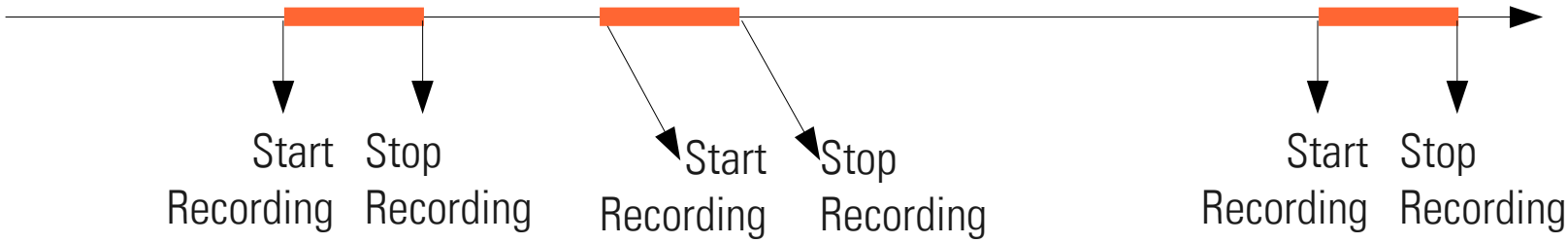
- Preemption off
- Interrupts off
- Preemption and interrupts off



Internal recording of effective latencies

- Wakeup time

Delay between wakeup and context switch



Internal latency recording

Kernel configuration

```
CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_PREEMPT_OFF_HIST=y
```

Access via debug file system

Command

```
mount -t debugfs nodev /sys/kernel/debug
```

Entry in /etc/fstab

```
nodev /sys/kernel/debug debugfs defaults 0 0
```

Directories

```
/sys/kernel/debug/tracing/latency_hist/enable

/sys/kernel/debug/tracing/latency_hist/irqsoff
/sys/kernel/debug/tracing/latency_hist/preemptirqsoff
/sys/kernel/debug/tracing/latency_hist/preemptoff

/sys/kernel/debug/tracing/latency_hist/wakeup
```

Internal latency recording - Files

Files

Enable latency recording

```
echo 1 >/sys/kernel/debug/tracing/latency_hist/enable/preemptirqsoff
echo 1 >/sys/kernel/debug/tracing/latency_hist/enable/wakeup
```

Latency histogram data

```
/sys/kernel/debug/tracing/latency_hist/irqsoff/CPU?
echo 1 >/sys/kernel/debug/tracing/latency_hist/irqsoff/reset

/sys/kernel/debug/tracing/latency_hist/preemptirqsoff/CPU?
echo 1 >/sys/kernel/debug/tracing/latency_hist/preemptirqsoff/reset

/sys/kernel/debug/tracing/latency_hist/preemptoff/CPU?
echo 1 >/sys/kernel/debug/tracing/latency_hist/preemptoff/reset

/sys/kernel/debug/tracing/latency_hist/wakeup/CPU?
/sys/kernel/debug/tracing/latency_hist/wakeup/max_latency-CPU?
echo $pid >/sys/kernel/debug/tracing/latency_hist/wakeup/pid
echo 1 >/sys/kernel/debug/tracing/latency_hist/wakeup/reset
```

Handle histograms - Reset

Reset

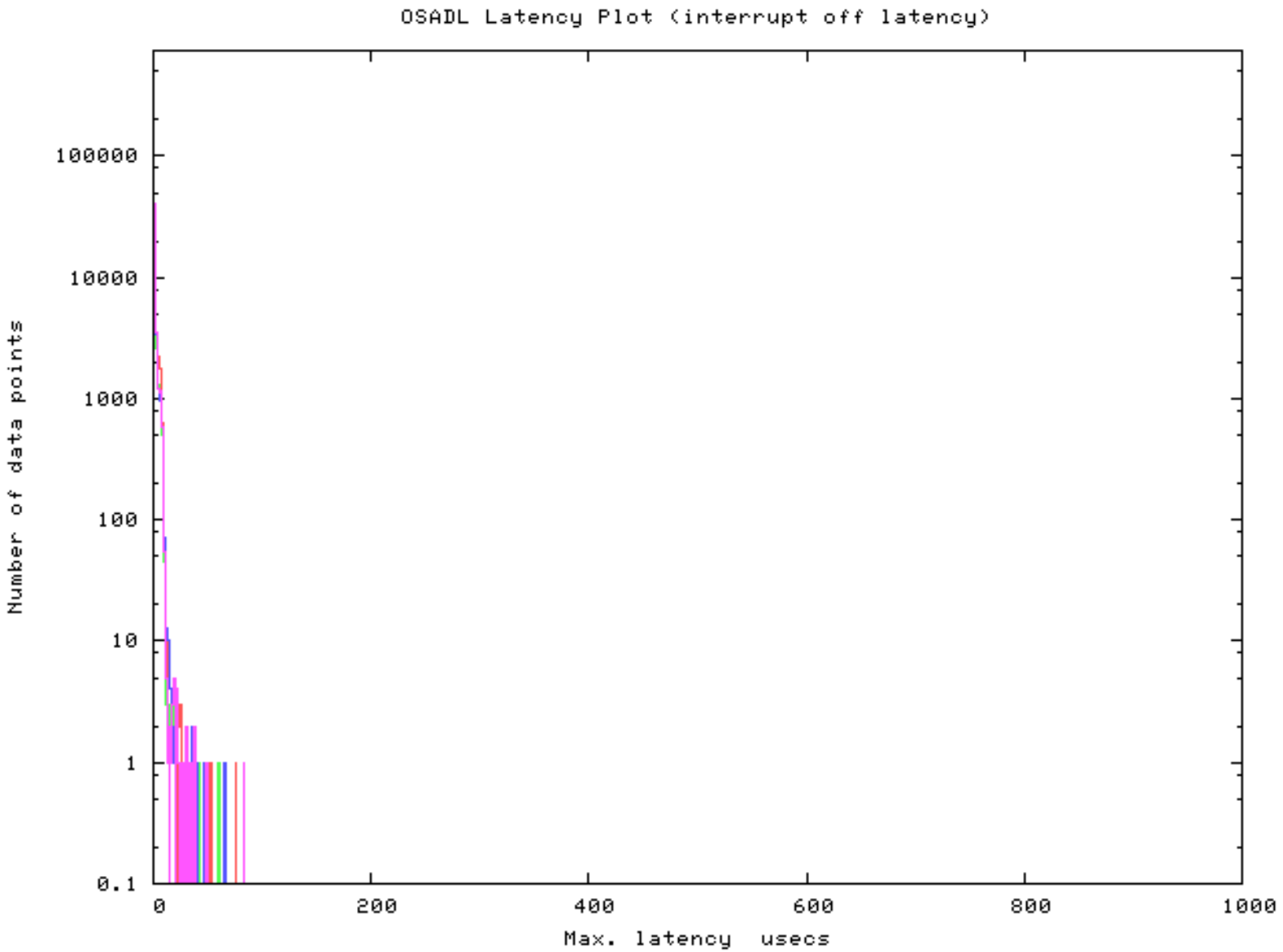
```
#!/bin/bash

HISTDIR=/sys/kernel/debug/tracing/latency_hist
if test -d $HISTDIR
then
  cd $HISTDIR
  for i in */reset
  do
    echo 1 >$i
  done
fi
```

Handle histograms – Evaluate data

Data

```
# grep -v " 0$" /sys/kernel/debug/tracing/latency_hist/irqsoff/CPU0
#Minimum latency: 0 microseconds.
#Average latency: 0 microseconds.
#Maximum latency: 63 microseconds.
#Total samples: 2622976567
#There are 0 samples greater or equal than 10240 microseconds
#usecs             samples
    0          2174555930
    1           251129896
    2           108221353
    3            22726693
    4            17853433
    5            20486535
    6            13811530
    7             6996682
    8             3464499
    9             2084766
   10              832247
   11              366531
   12              158594
   13               67561
   14               40456
   15               28985
   16               21873
   17               16504
```

Interrupt-off latency histogram



Calibration of latency recording

“Bad” driver (blocksys.ko)

```
local_irq_disable();
while (nops--)
  asm("nop");
local_irq_enable();
```

Using the “bad” driver (mklatency)

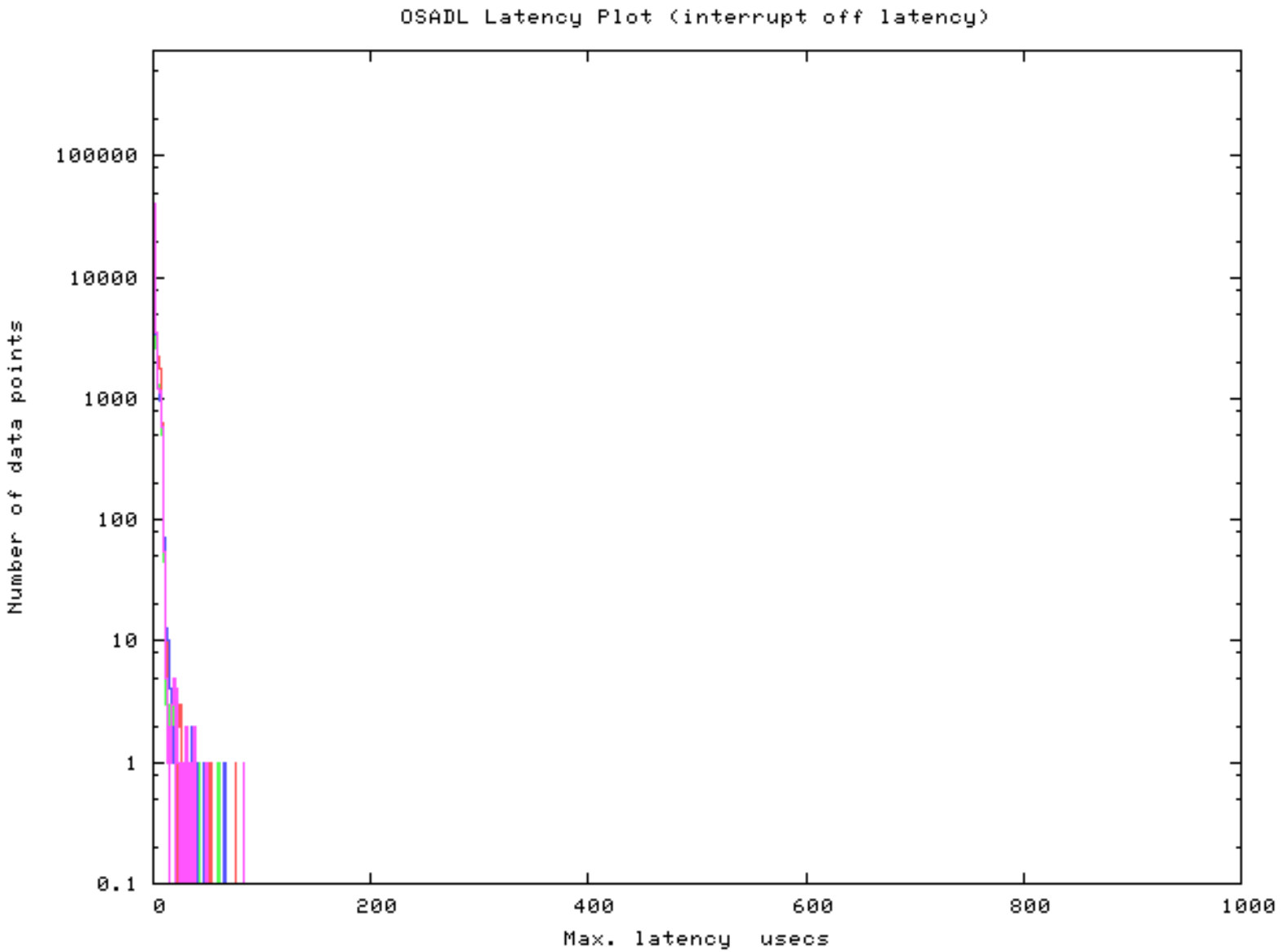
Command

```
mklatency
```

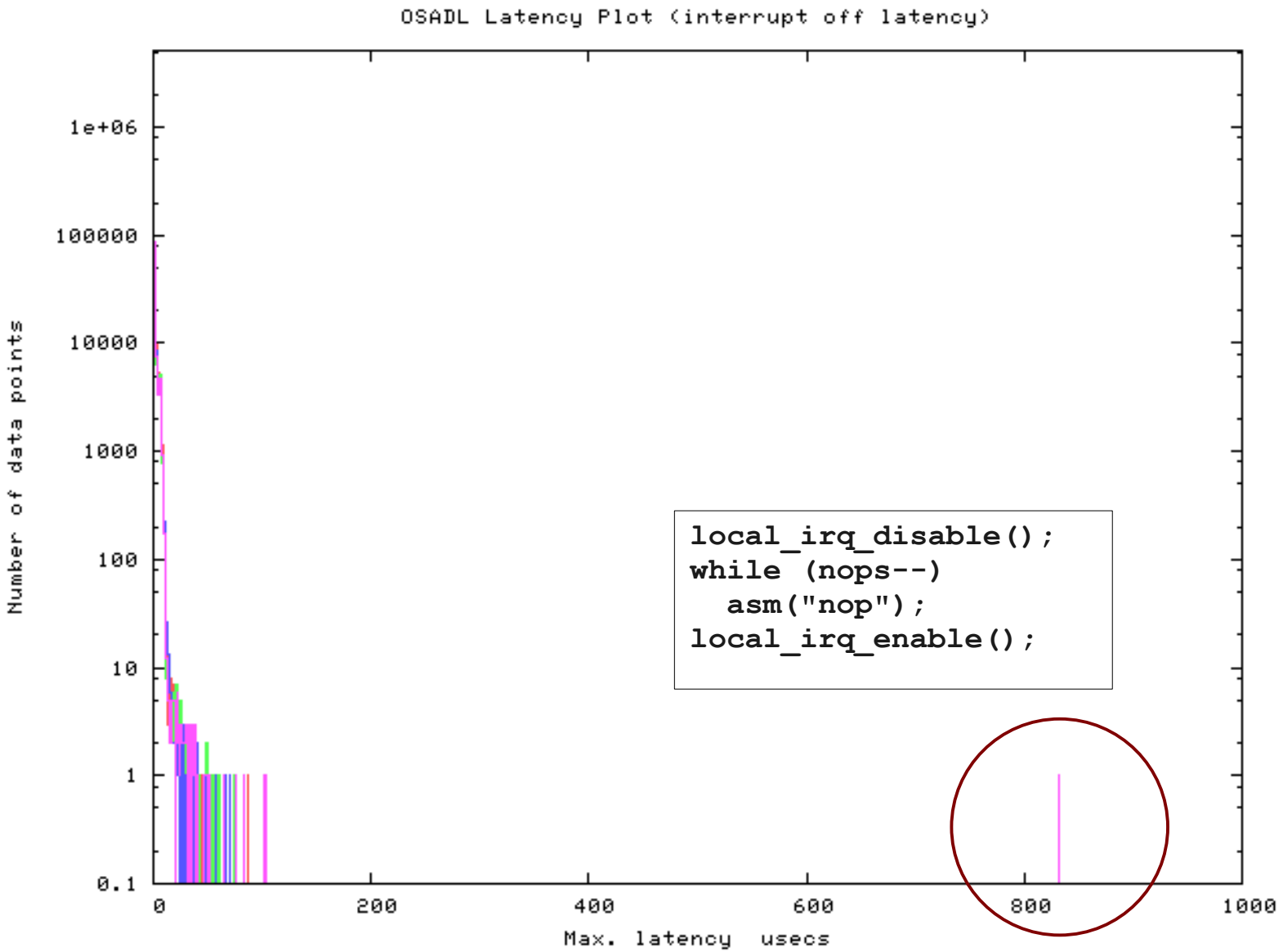
Kernel log

```
[..] kernel: blocksys: CPU #0 will be blocked for 2000000 nops
[..] kernel: blocksys: CPU #0 blocked about 835 us
```

Interrupt-off latency histogram (before)



Interrupt-off latency histogram (after)



Penalty of latency recording

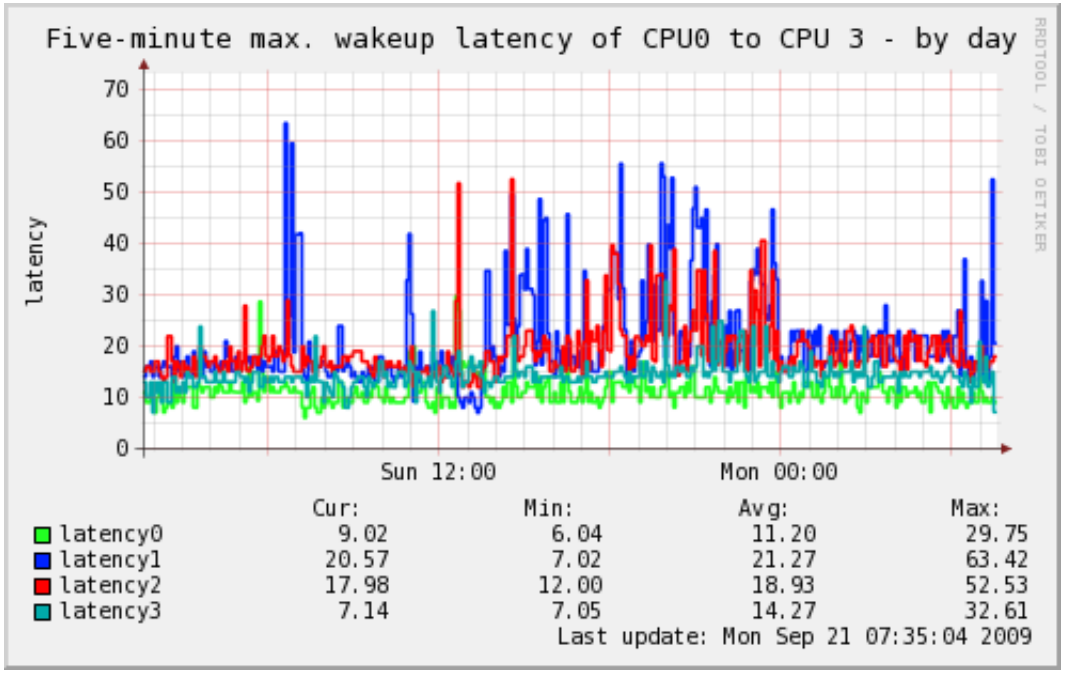
Latency recording of potential latencies (interrupt off etc.)

has a measurable effect on the system latency in the range of 5%.

Latency recording of effective latencies (wakeup latency)

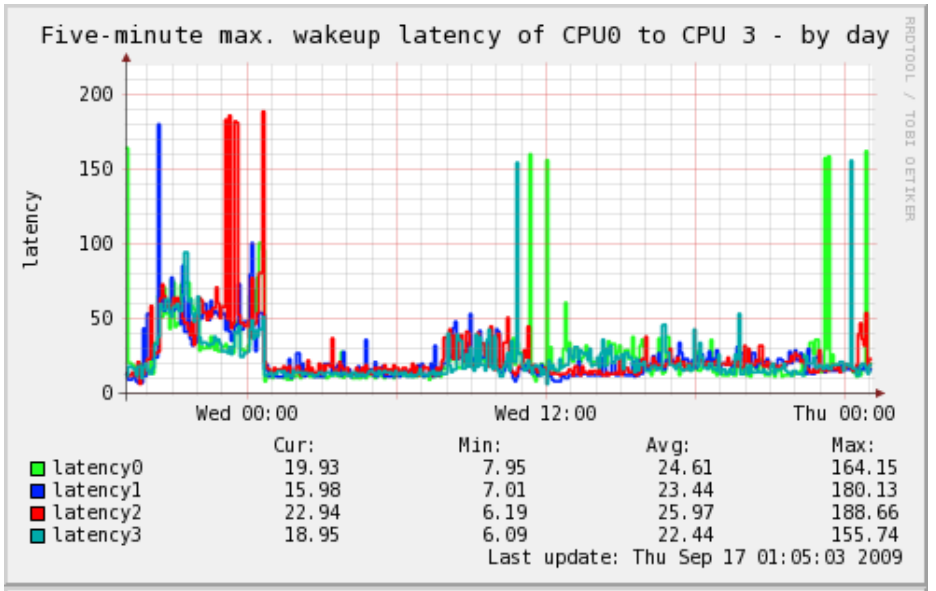
has a negligible effect on the system latency in the range of <1%. This makes it possible to continuously monitor the wakeup latency in a production system (even during its entire life cycle).

Continuous recording of the wakeup latency (1)

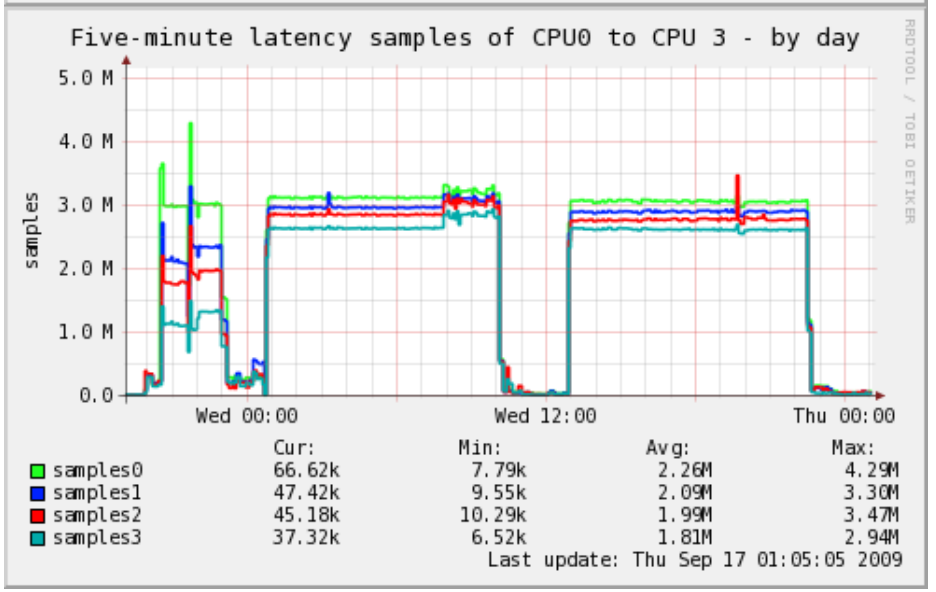


(using the Munin monitoring tool)

Continuous recording of the wakeup latency (2)



Latencies

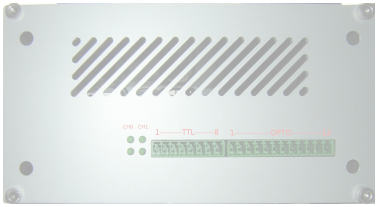


Number of samples

Four levels of latency tests

External measurement with simulation

OSADL's „Latency-Box“



Internal continuous recording

Built-in kernel latency histograms

```
CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_PREEMPT_OFF_HIST=y
```

Internal measurement with simulation

Cyclictest

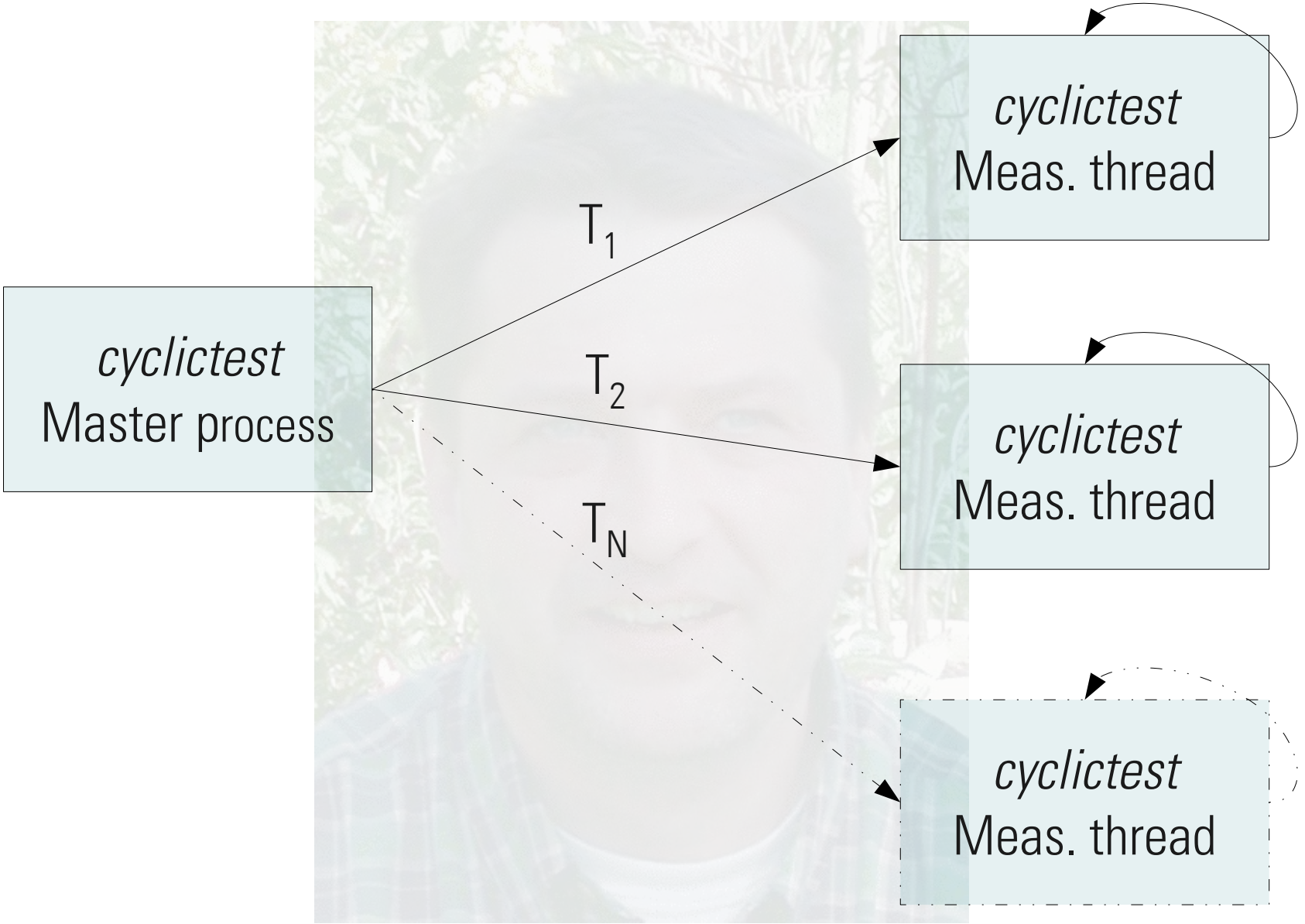
```
# cyclictest -a -t -n -p99
```

Real-world internal measurement

Application

```
# <application>
```

Cyclictest - Principle



Cyclictest: Command line parameters

```
# 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
T: 1 (18618) P:98 I:150 C:  708,641,019 Min:  2 Act:  5 Avg: 11 Max: 57
```

-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 specifed, **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.

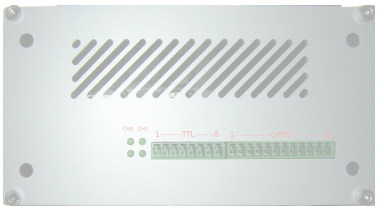
-p99 *Priority*. Set the priority of the first thread. The given priority is assigned to the first test thread. Each further thread receives the priority reduced by the number of the thread.

-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 cyclictest 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.

Four levels of latency tests

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



Internal continuous recording
Built-in kernel latency histograms

```
CONFIG_WAKEUP_LATENCY_HIST=y
CONFIG_INTERRUPT_OFF_HIST=y
CONFIG_PREEMPT_OFF_HIST=y
```

Internal measurement with simulation
Cyclictest

```
# cyclictest -a -t -n -p99
```

Real-world internal measurement
Application

```
# <application>
```

Conclusions (1)

Latency tests must be done

- long enough (recommended at least 10^9 measurements or continuously)
- frequently enough (interval between triggers no more than twice the expected worst-case latency)
- under appropriate load (every OS has a low wakeup latency when idle)
- after calibration (make sure that latencies are, in fact, recorded)

Conclusions (2)

Path analysis is the “gold standard” - it is the best way to determine the worst-case latency of a system. Use it whenever possible.

On modern high-performance processors, path analysis may no longer be feasible. The empirical determination of the worst-case latency may be used instead. When done correctly, it may provide a level of confidence that is similar to path analysis.