The OSADL QA Farm and how real-time Ethernet is implemented

Basic lecture: What is latency monitoring and what is it good for?

Carsten Emde
Open Source Automation Development Lab (OSADL) eG

Agenda

How does the OSADL QA Farm work and what does it provide?

1. Quick description of rationale, tools and methods
2. Standard schedule to measure performance, stability and real-time capabilities
3. Individual latency monitoring programs of single systems or of groups of systems
   - Effect of energy-saving on real-time
   - Effect of virtualization on real-time of the host system
   - Performance of real-time Ethernet
     - Peer-to-peer duplex UDP without and with VLAN
     - Ethercat
     - Powerlink
     - **OPC UA PubSub over TSN**

Opinions (in about 2000)

Established experts' opinion	Linux kernel developers' opinion
"It is impossible to refactor a general-purpose kernel of 10 million lines of code and to convert it to a real-time kernel."	

Opinions (in about 2000)

Established experts' opinion	Linux kernel developers' opinion
"It is impossible to refactor a general-purpose kernel of 10 million lines of code and to convert it to a real-time kernel."	"Nothing is impossible."

Milestone (in 2005)

- The first patch set was released to convert the general-purpose Linux kernel version 2.6.11 into a real-time kernel.
- Short-term tests showed an impressive low latency.
- Were the experts wrong?

The real-time dilemma

- A “real-time” system is expected to **always** react within a predefined amount of time – even a single failure to do so within the system’s entire life time is not acceptable.
- BTW: A “real-time” system has nothing to do with the “real” “time”; instead of “real-time” it would better be called “deterministic”.

The real-time dilemma

- A “real-time” system is expected to **always** react within a predefined amount of time – even a single failure to do so within the system’s entire life time is not acceptable.
- BTW: A “real-time” system has nothing to do with the “real” “time”; instead of “real-time” it would better be called “deterministic”.
- Determinism cannot be confirmed by a measurement, a measurement can only confirm that a system is not deterministic.
- How can we decide whether the experts were wrong?

Is path analysis a solution?

- What is path analysis?
  - Find the longest code path while the system is not responsive (e.g. since interrupts are disabled), and calculate the duration of that path from the duration of the contained instructions (available in the manual).
  - This is the silver bullet of a system's worst-case latency determination.

Is path analysis a solution?

- What is path analysis?
  - Find the longest code path while the system is not responsive (e.g. since interrupts are disabled), and calculate the duration of that path from the duration of the contained instructions (available in the manual).
  - This is the silver bullet of a system's worst-case latency determination.
- Why can't we do path analysis any longer?
  - The duration of individual instructions is not constant in modern processors and, thus, no longer available in the manual.

So we need to measure worst-case latency

So we need to measure worst-case latency, although it is impossible

So we need to measure worst-case latency, although it is impossible

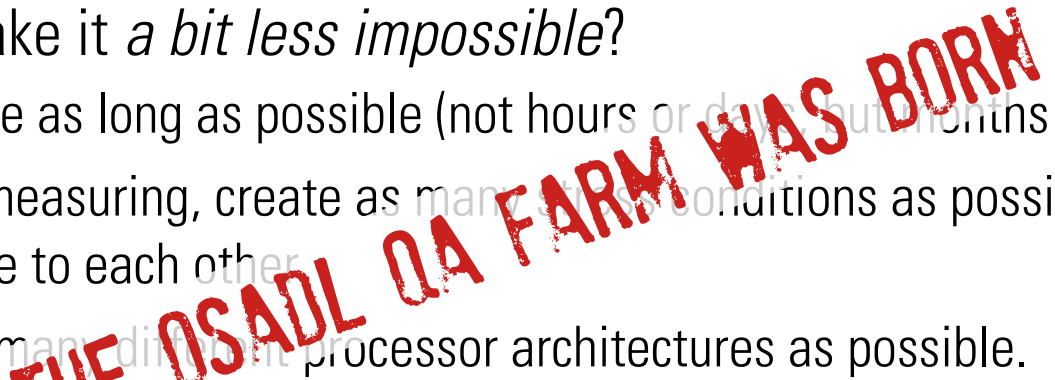
- How to make it *a bit less impossible*?
  - Measure as long as possible (not hours or days, but months or even years).
  - While measuring, create as many stress conditions as possible that randomly interfere to each other.
  - Use as many different processor architectures as possible.
  - Use as many different platforms as possible

So we need to measure worst-case latency, although it is impossible

- How to make it *a bit less impossible*?
  - Measure as long as possible (not hours or days, but months or even years).
  - While measuring, create as many stress conditions as possible that randomly interfere to each other.
  - Use as many different processor architectures as possible.
  - Use as many different platforms as possible.
- But keep in mind that this only estimates the worst-case latency with a certain uncertainty, it is no mathematical proof.

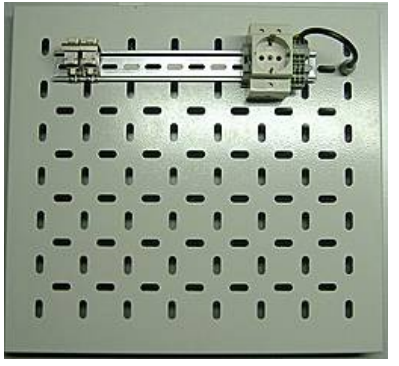
So we need to measure worst-case latency, although it is impossible

- How to make it *a bit less impossible*?
  - Measure as long as possible (not hours or days, but months or even years).
  - While measuring, create as many stress conditions as possible that randomly interfere to each other.
  - Use as many different processor architectures as possible.
  - Use as many different platforms as possible.
- But keep in mind that this only estimates the worst-case latency with a certain uncertainty, it is no mathematical proof.

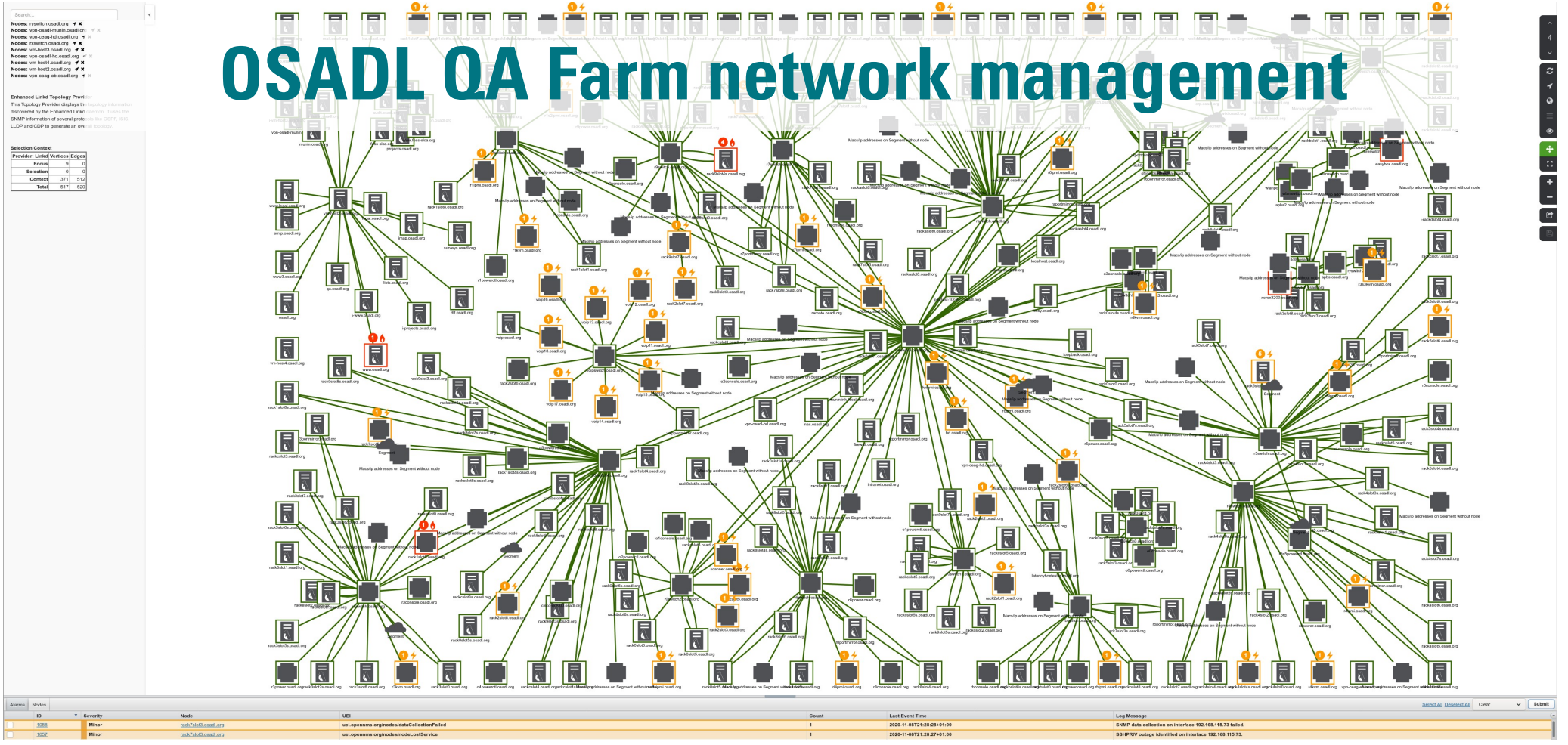


What is the OSADL QA Farm?

- About 180 systems in open racks each of eight test systems and one control server
- Remote power distribution and control
- Access via serial console, network and graphics
- Continuous condition monitoring and reporting
- 24-hour stress program



OSADL QA Farm network management



COOL – Compact OSADL Online Lectures, November 18, 2020

Basic lecture: What is latency monitoring and what is it good for?

Open Source Automation Development Lab (OSADL) eG, Heidelberg

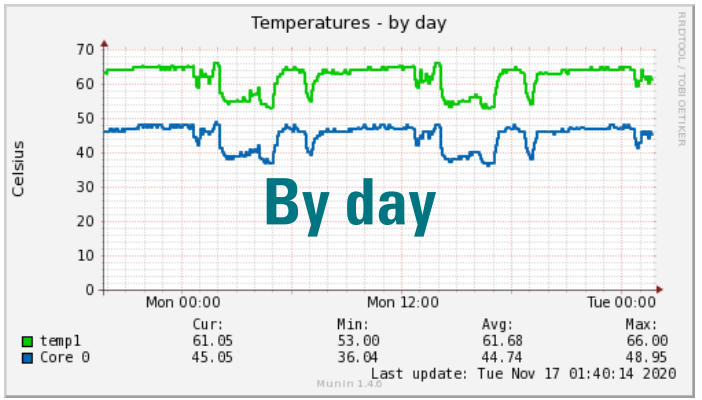
COMPACT OSADL ONLINE LECTURES

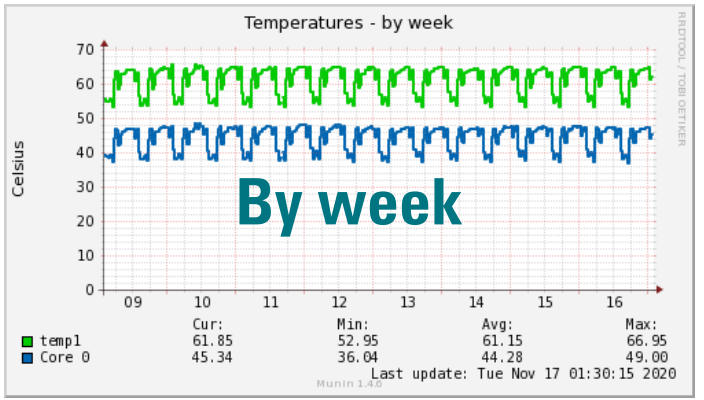
Open Source Automation Development Lab eG

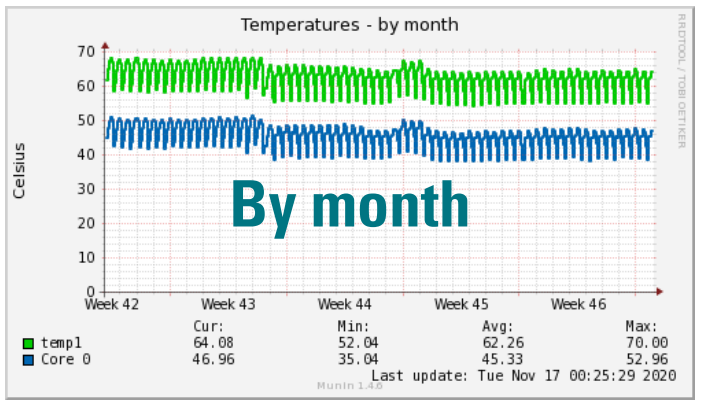
OSADL QA Farm monitoring (Munin)

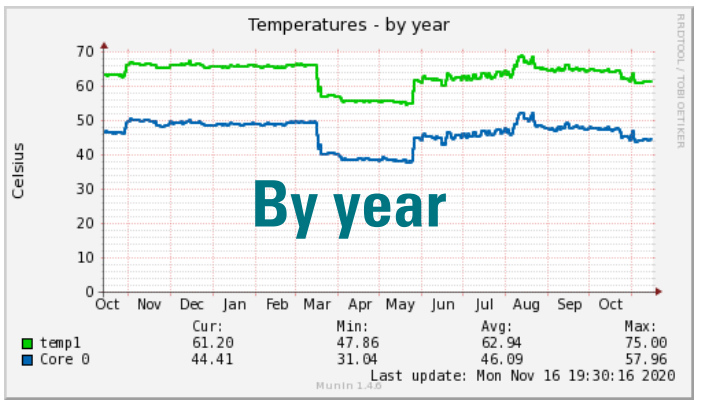
- **rack1slot6.osadl.org** [benchmarks disk network nfs processes sendmail sensors system time]
- **rack1slot6s.osadl.org** [benchmarks disk network nfs processes sendmail sensors system time]
- **rack1slot8.osadl.org** [benchmarks disk network nfs processes sendmail sensors system time]
- **rack1slot8s.osadl.org** [benchmarks disk network processes sendmail sensors system time]
- **rack2slot0.osadl.org** [benchmarks disk network nfs 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]
- **rack2slot6s.osadl.org** [disk network nfs processes sensors system time]
- **rack2slot7.osadl.org** [benchmarks disk network nfs processes sendmail sensors system time]
- **rack2slot8.osadl.org** [benchmarks disk network nfs processes sensors 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]
- **rack3slot2s.osadl.org** [benchmarks disk network nfs processes sendmail sensors system time]
- **rack3slot3.osadl.org** [benchmarks disk network nfs processes sendmail sensors system time]
- **rack3slot5.osadl.org** [disk network nfs processes sendmail system time]
- **rack3slot5s.osadl.org** [benchmarks disk network nfs processes sendmail system time]
- **rack3slot6.osadl.org** [benchmarks disk network nfs processes sensors system time]
- **rack3slot6s.osadl.org** [benchmarks disk network nfs processes sendmail sensors system time]
- **rack3slot7.osadl.org** [benchmarks disk network nfs processes sendmail sensors system time]

OSADL QA Farm monitoring (Munin)

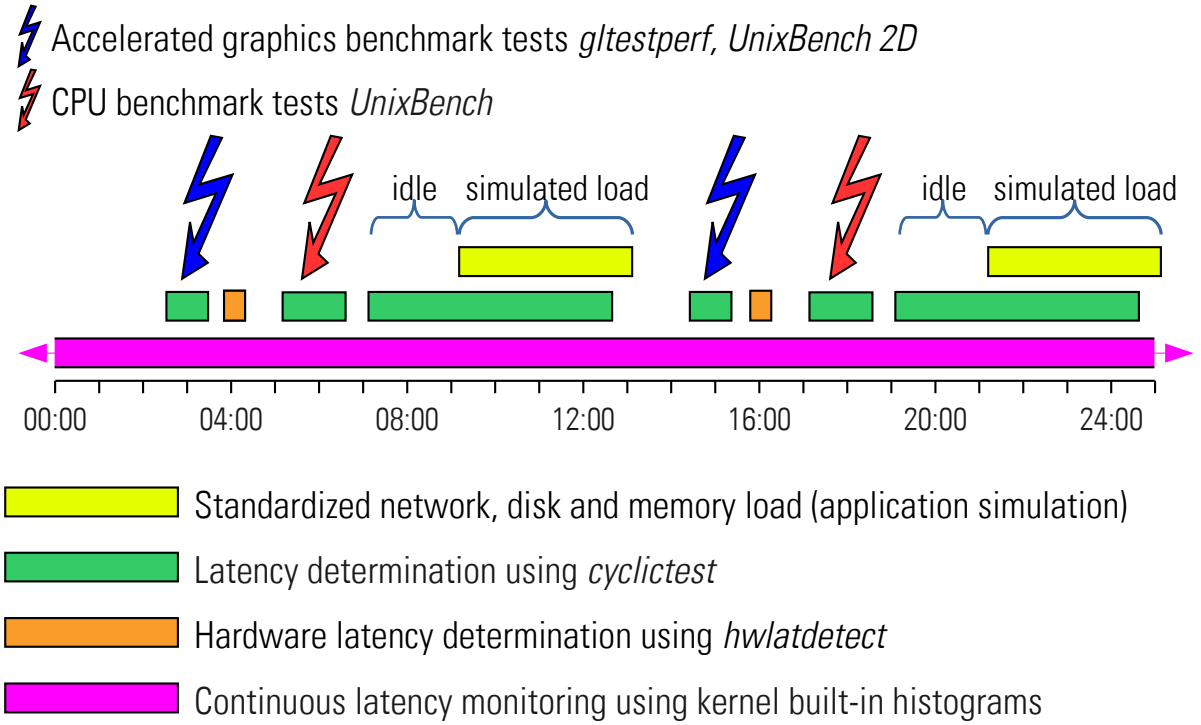




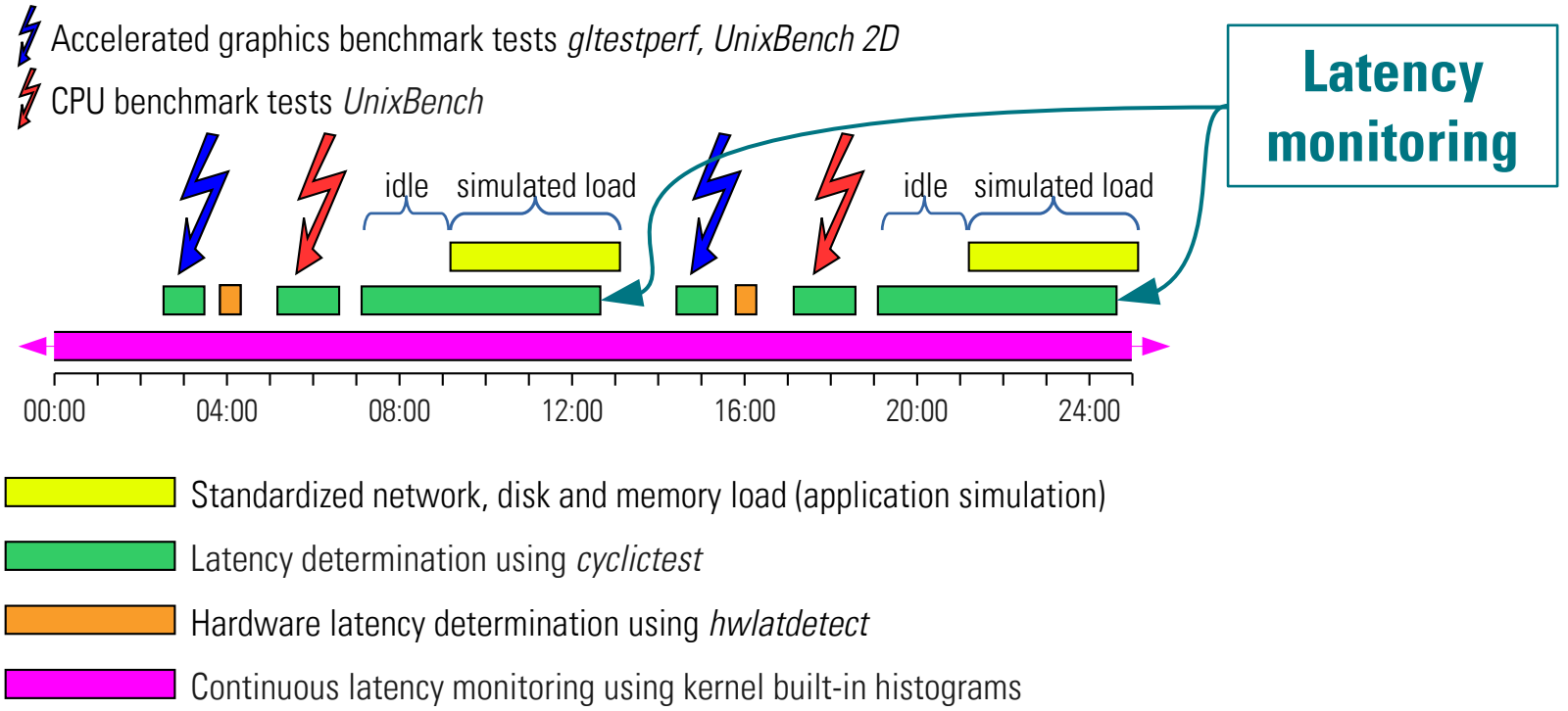




OSADL QA Farm stress program



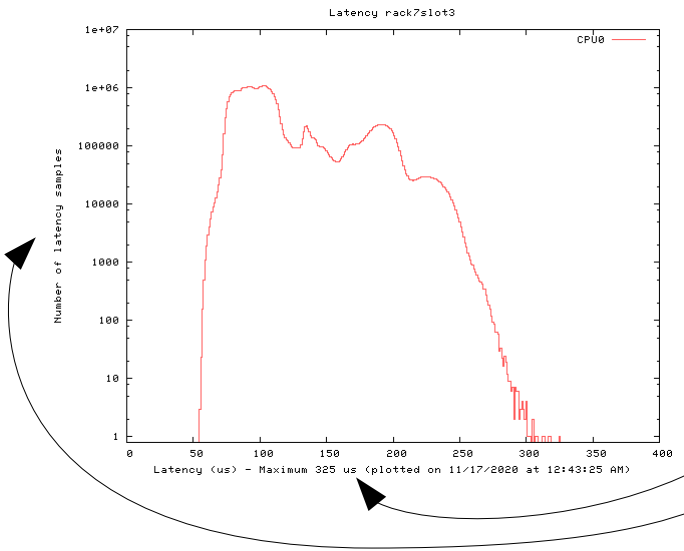
OSADL QA Farm stress program



OSADL QA Farm single latency plot

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare



Number of triggers:	100 million
Cycle interval:	200 µs
Resulting duration:	5 h, 33 min

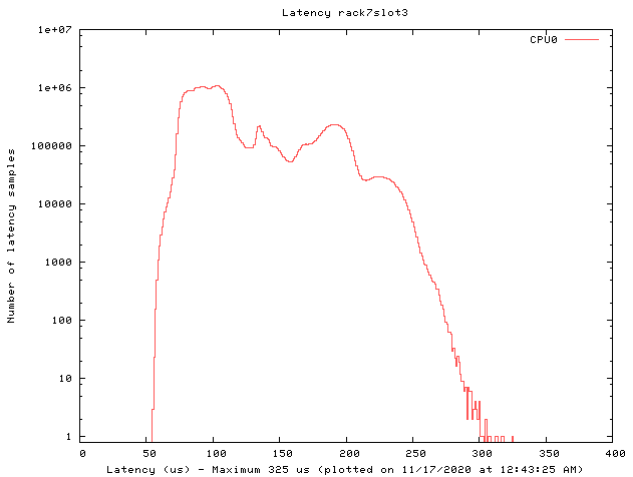
Linear x-scale

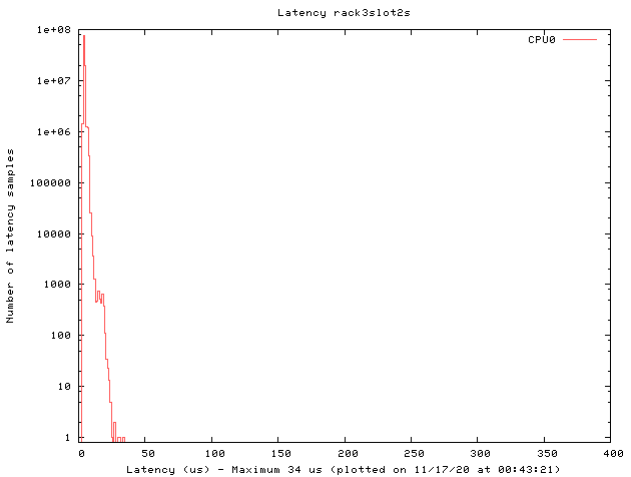
Logarithmic y-scale

OSADL QA Farm single latency plot

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare

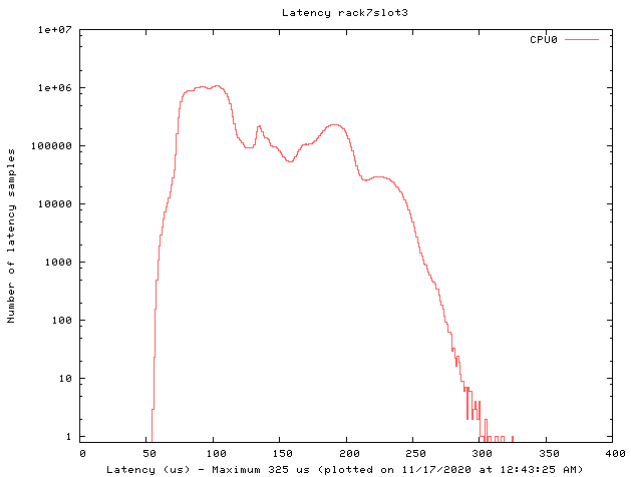


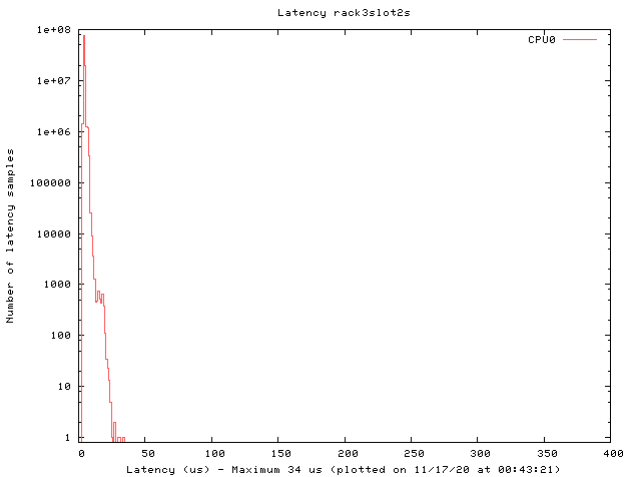


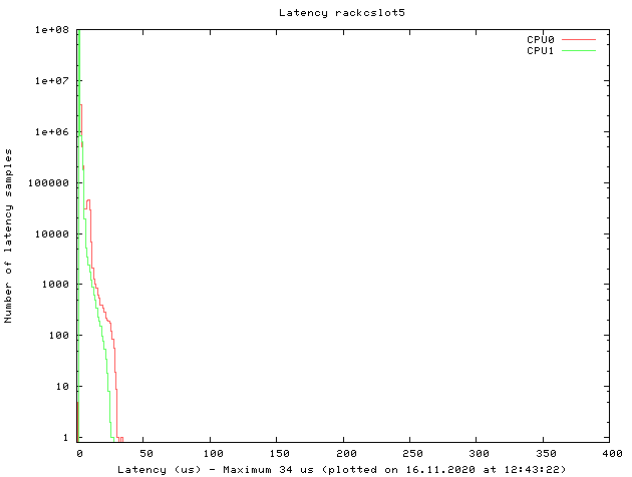
OSADL QA Farm single latency plot

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare



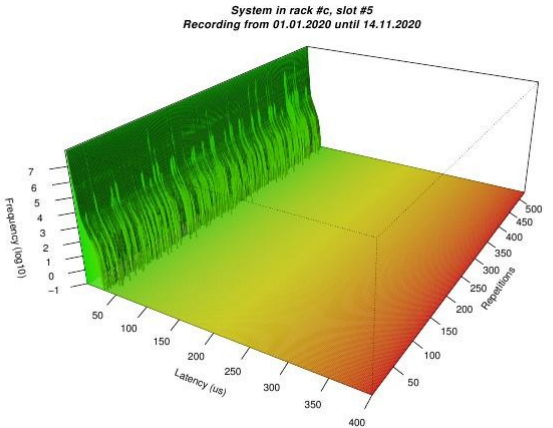


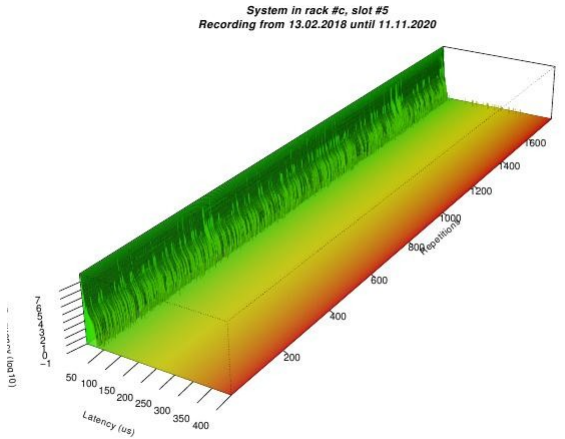


OSADL QA Farm single latency plot

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare

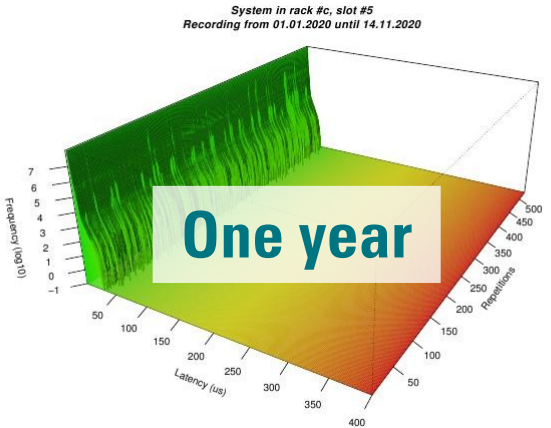


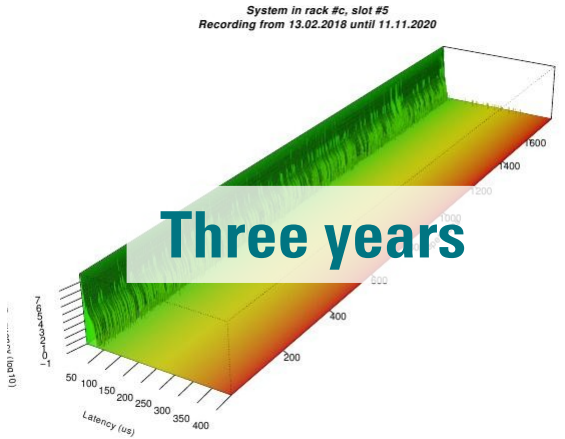


OSADL QA Farm single latency plot

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare

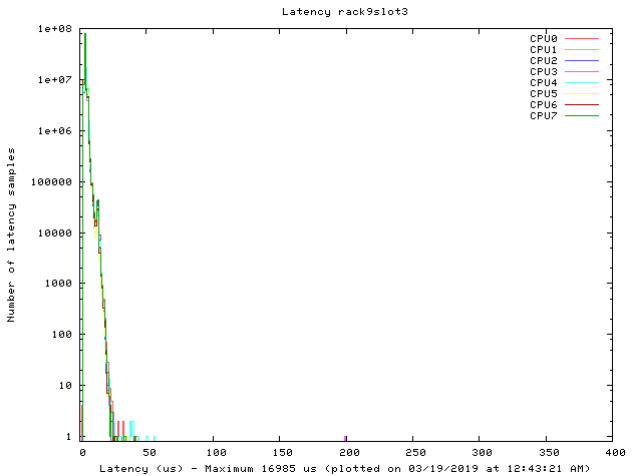




Latency outlier in a short-term and long-term plot

OSADL QA Farm on Real-time of Mainline Linux

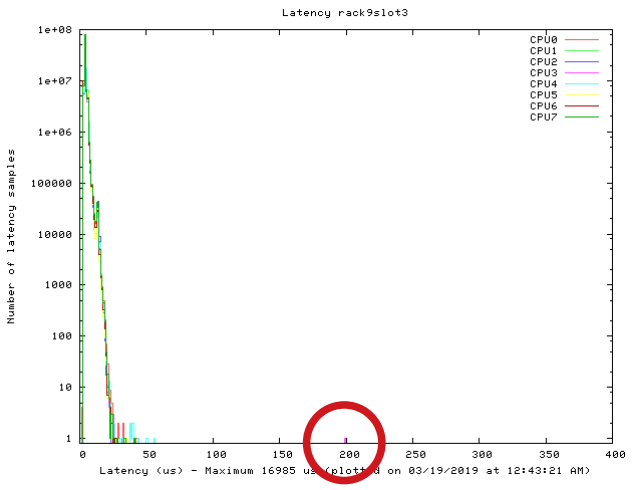
About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare

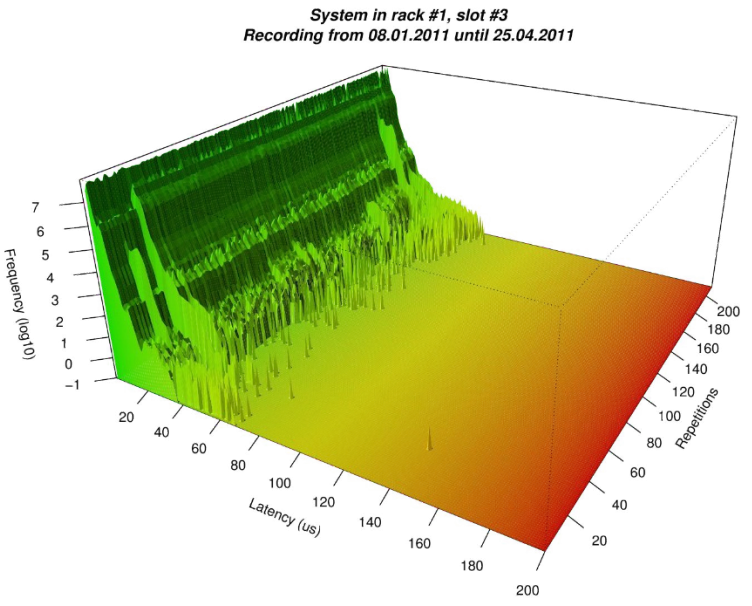


Latency outlier in a short-term and long-term plot

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare

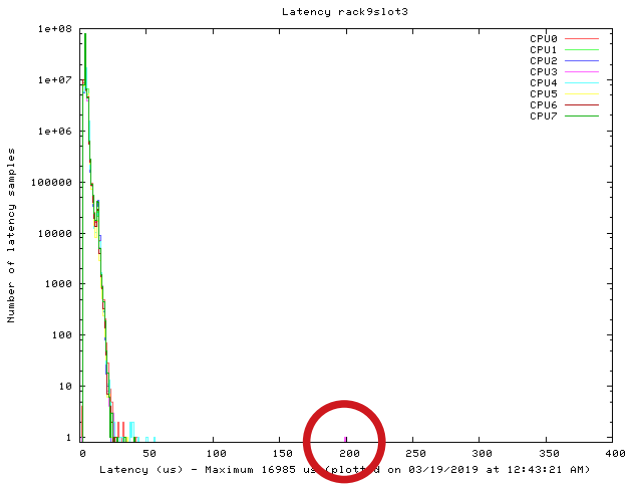


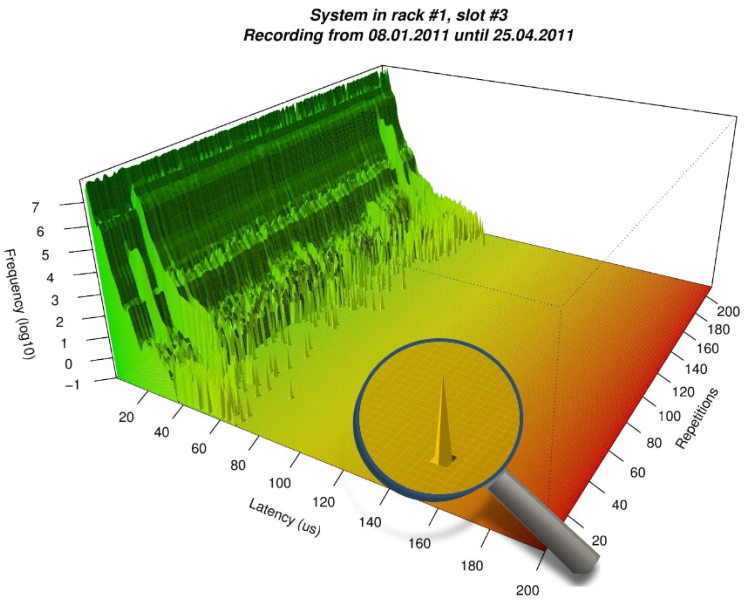


Latency outlier in a short-term and long-term plot

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare





Advantages and disadvantages of latency plots

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare

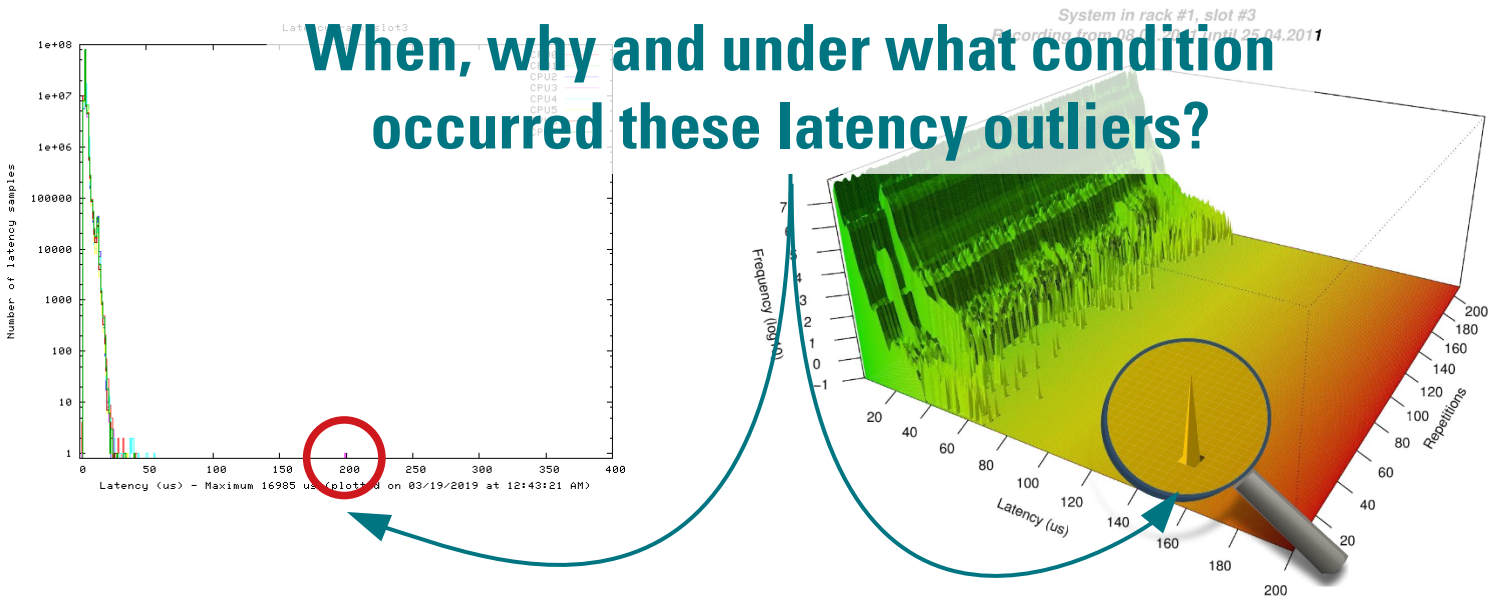
Advantages	Disadvantage
Easy to do	Time information lost
Clear evidence in case of a latency outlier	

Time information is lost

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - Latency monitoring - **Latency plots** - System data - Profiles - Compare

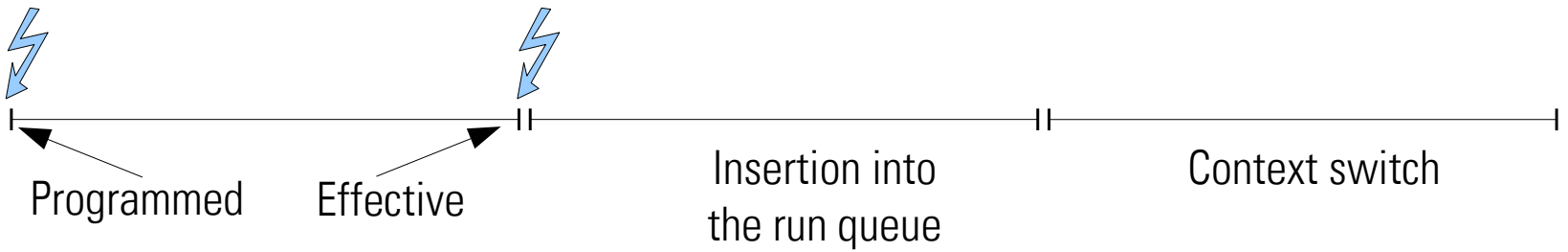
When, why and under what condition occurred these latency outliers?



OSADL QA Farm latency monitoring

OSADL QA Farm on Real-time of Mainline Linux

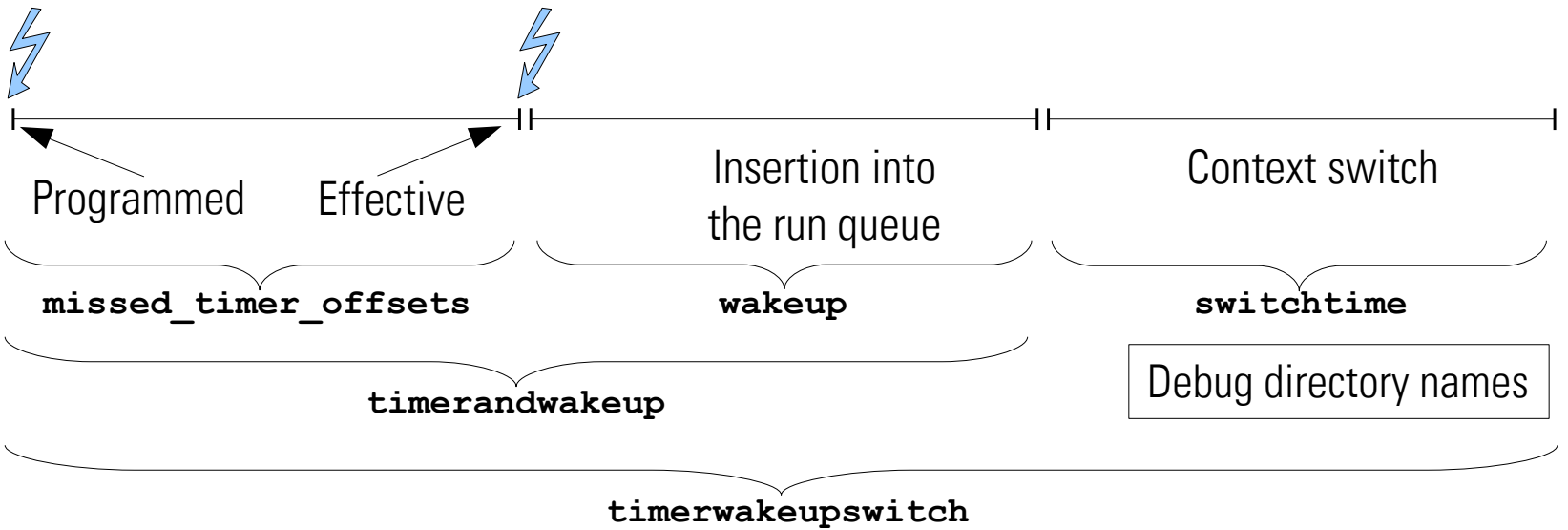
About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - **Latency monitoring** - Latency plots - System data - Profiles - Compare



OSADL QA Farm latency monitoring

OSADL QA Farm on Real-time of Mainline Linux

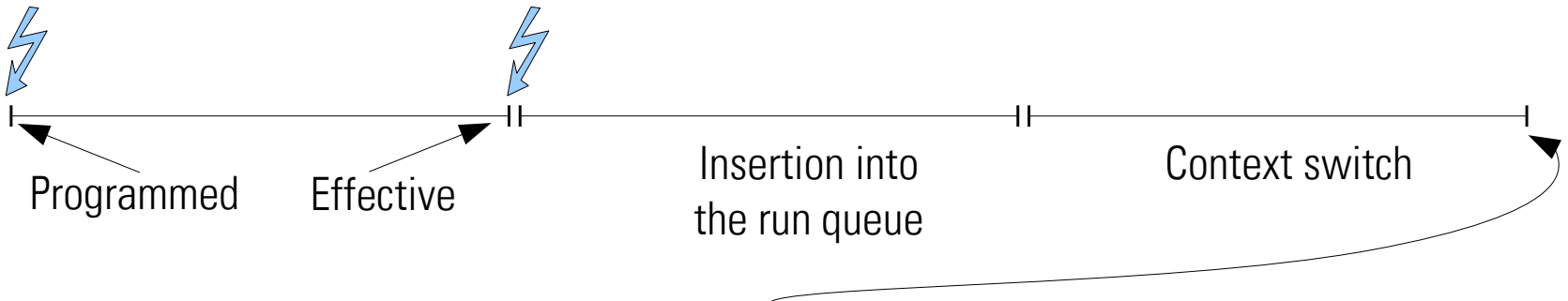
About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - **Latency monitoring** - Latency plots - System data - Profiles - Compare



OSADL QA Farm latency monitoring

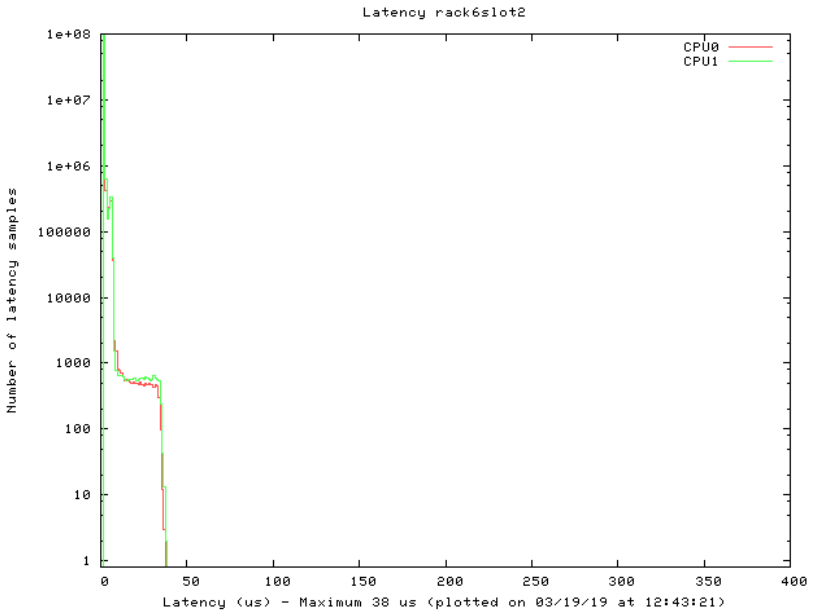
OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - **Latency monitoring** - Latency plots - System data - Profiles - Compare

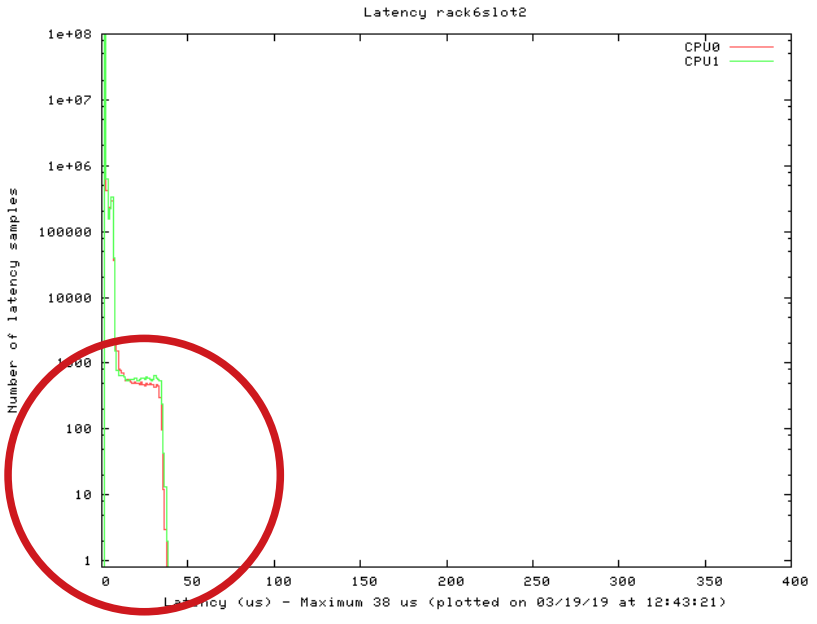


If the current latency exceeds the most recent latency so far (for example per 5-min interval, characteristic data of the task switch are stored in a so-called "culprit/victim table".

Using the culprit/victim table for latency fighting



Using the culprit/victim table for latency fighting



Culprit/victim table

Characteristics of the 20 highest latencies: **System rack6slot2.osadl.org (updated Mon Mar 18, 2019 00:43:27)**									
Delayed (victim)					**Switcher (culprit)**			**Timestamp**	**CPU**
PID	**Prio**	**Total latency (µs)**	**T*(,W**) latency (µs)**	**Cmd**	**PID**	**Prio**	**Cmd**		
5528	2	114	0,1	sleep1	21	-21	rcuc/1	19:31:47	1
18185	2	61	0,0	sleep1	0	-21	swapper/1	21:32:02	1
28116	2	43	8,8	sleep1	0	-21	swapper/1	19:05:18	1
28316	99	39	32,6	cyclictest	2408	50	irq/16-enp2s0f0	20:36:47	1
28316	99	38	33,4	cyclictest	2408	50	irq/16-enp2s0f0	20:41:45	1
28316	99	38	32,5	cyclictest	2408	50	irq/16-enp2s0f0	00:11:50	1
28316	99	38	31,6	cyclictest	2408	50	irq/16-enp2s0f0	22:54:36	1
28316	99	37	34,2	cyclictest	2408	50	irq/16-enp2s0f0	20:13:03	1
28316	99	37	32,4	cyclictest	2408	50	irq/16-enp2s0f0	23:43:28	1
28316	99	37	32,4	cyclictest	2408	50	irq/16-enp2s0f0	21:06:47	1
28316	99	37	32,4	cyclictest	2408	50	irq/16-enp2s0f0	20:56:53	1
28316	99	37	32,4	cyclictest	2408	50	irq/16-enp2s0f0	19:26:50	1
28316	99	37	32,4	cyclictest	2408	50	irq/16-enp2s0f0	19:12:46	1
28316	99	37	31,5	cyclictest	2408	50	irq/16-enp2s0f0	21:52:05	1
28316	99	36	33,2	cyclictest	2408	50	irq/16-enp2s0f0	22:20:44	1
28316	99	36	32,3	cyclictest	2408	50	irq/16-enp2s0f0	23:41:50	1
28316	99	36	32,3	cyclictest	2408	50	irq/16-enp2s0f0	23:26:47	1
28316	99	36	32,3	cyclictest	2408	50	irq/16-enp2s0f0	21:45:53	1
28316	99	36	32,3	cyclictest	2408	50	irq/16-enp2s0f0	21:35:53	1
28316	99	36	32,3	cyclictest	2408	50	irq/16-enp2s0f0	20:03:54	1
*Timer **Wakeup (latency=timer+wakeup+contextswitch)									

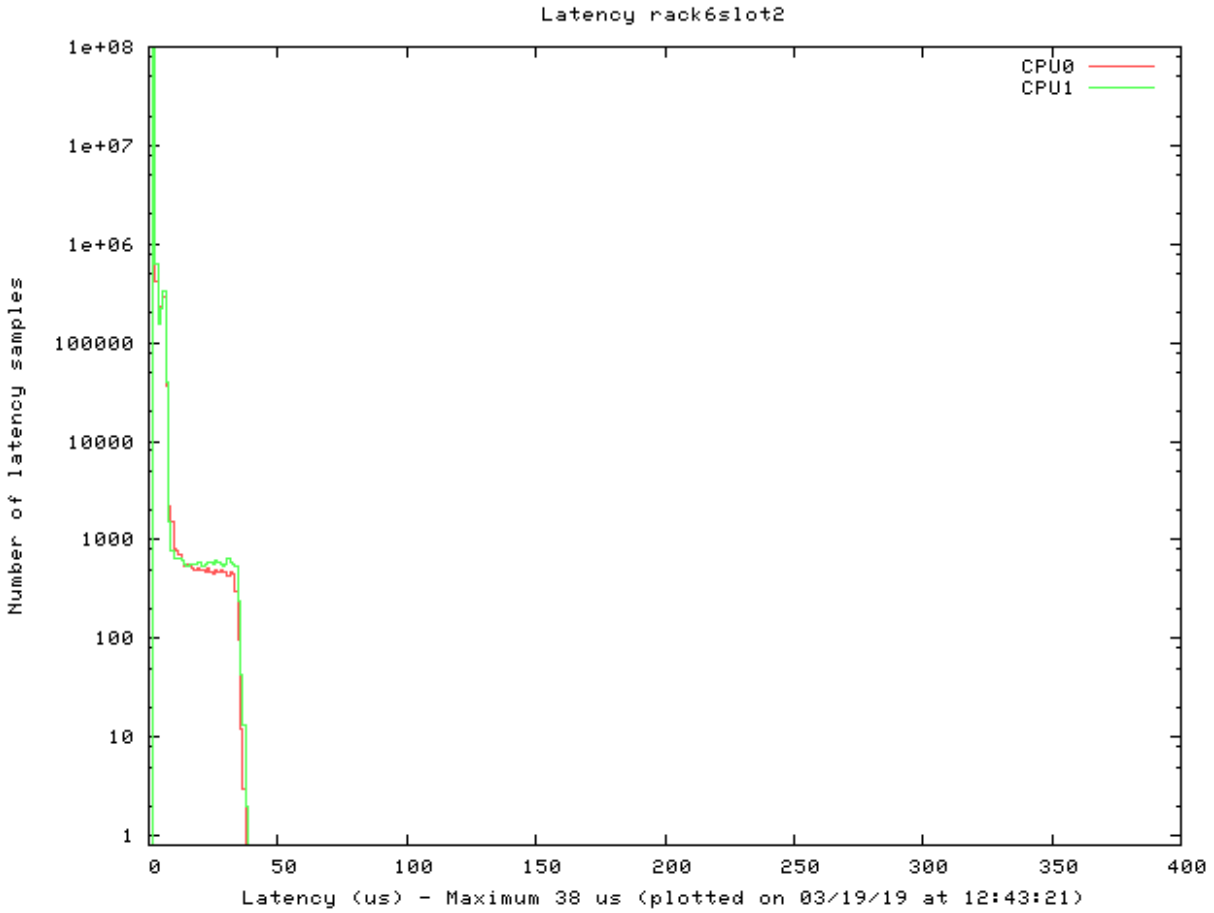
Culprit/victim table

Characteristics of the 20 highest latencies: **System rack6slot2.osadl.org (updated Mon Mar 18, 2019 00:43:27)**									
Delayed (victim)					**Switcher (culprit)**			**Timestamp**	**CPU**
PID	**Prio**	**Total latency (µs)**	**T*(,W**) latency (µs)**	**Cmd**	**PID**	**Prio**	**Cmd**		
5528	2	114	0,1	sleep1	21	-21	rcuc/1	19:31:47	1
18185	2	61	0,0	sleep1	0	-21	swapper/1	21:32:02	1
28116	2	43	8,8	sleep1	0	-21	swapper/1	19:05:18	1
28316	99	***39***	32,6	***cyclictest***	2408	50	***irq/16-enp2s0f0***	20:36:47	1
28316	99	***38***	33,4	***cyclictest***	2408	50	***irq/16-enp2s0f0***	20:41:45	1
28316	99	***38***	32,5	***cyclictest***	2408	50	***irq/16-enp2s0f0***	00:11:50	1
28316	99	***38***	31,6	***cyclictest***	2408	50	***irq/16-enp2s0f0***	22:54:36	1
28316	99	***37***	34,2	***cyclictest***	2408	50	***irq/16-enp2s0f0***	20:13:03	1
28316	99	***37***	32,4	***cyclictest***	2408	50	***irq/16-enp2s0f0***	23:43:28	1
28316	99	***37***	32,4	***cyclictest***	2408	50	***irq/16-enp2s0f0***	21:06:47	1
28316	99	***37***	32,4	***cyclictest***	2408	50	***irq/16-enp2s0f0***	20:56:53	1
28316	99	***37***	32,4	***cyclictest***	2408	50	***irq/16-enp2s0f0***	19:26:50	1
28316	99	***37***	32,4	***cyclictest***	2408	50	***irq/16-enp2s0f0***	19:12:46	1
28316	99	***37***	31,5	***cyclictest***	2408	50	***irq/16-enp2s0f0***	21:52:05	1
28316	99	***36***	33,2	***cyclictest***	2408	50	***irq/16-enp2s0f0***	22:20:44	1
28316	99	***36***	32,3	***cyclictest***	2408	50	***irq/16-enp2s0f0***	23:41:50	1
28316	99	***36***	32,3	***cyclictest***	2408	50	***irq/16-enp2s0f0***	23:26:47	1
28316	99	***36***	32,3	***cyclictest***	2408	50	***irq/16-enp2s0f0***	21:45:53	1
28316	99	***36***	32,3	***cyclictest***	2408	50	***irq/16-enp2s0f0***	21:35:53	1
28316	99	***36***	32,3	***cyclictest***	2408	50	***irq/16-enp2s0f0***	20:03:54	1
*Timer **Wakeup (latency=timer+wakeup+contextswitch)									

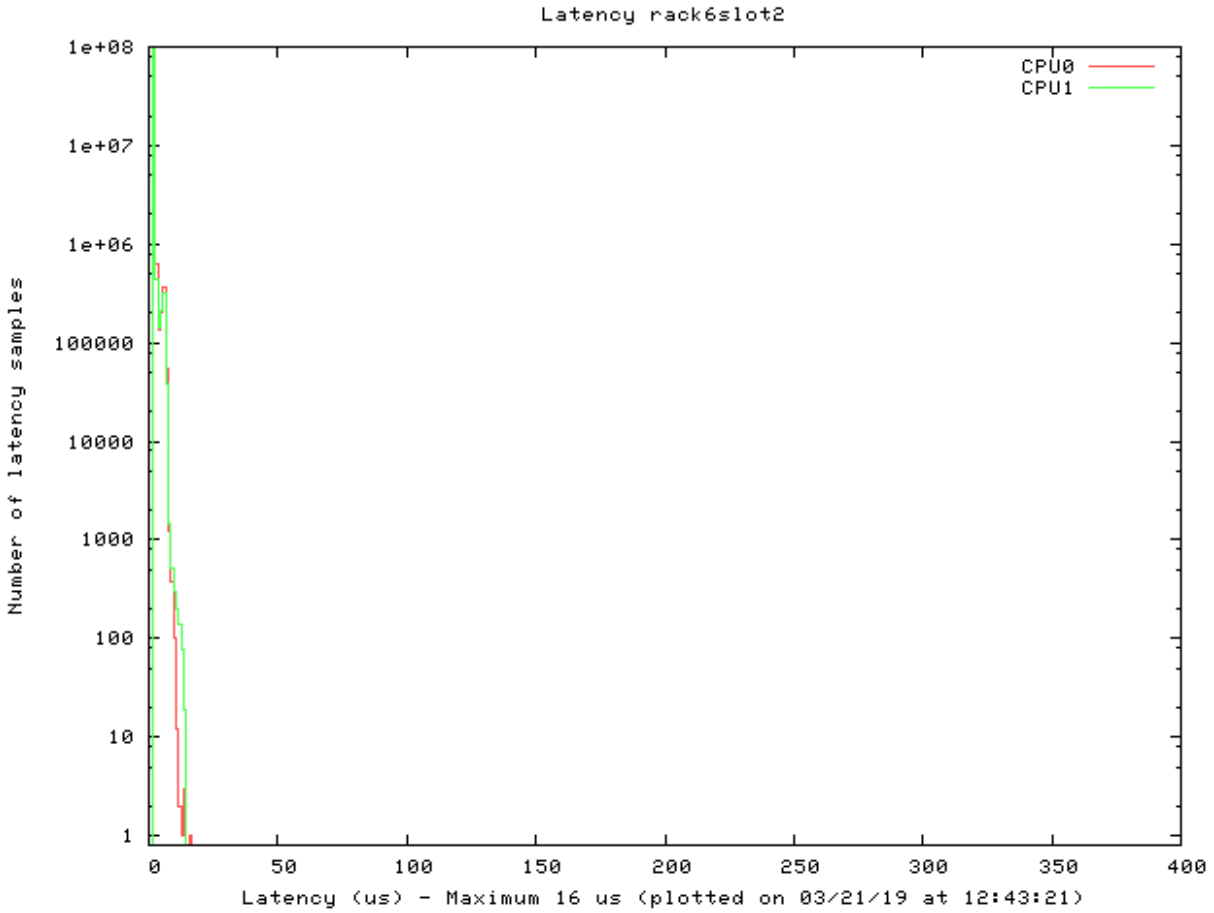
Countermeasures

- Quick fix
  - Uninstall Broadcom Limited NetLink BCM57785 Gigabit Ethernet PCIe network device and unload driver *tg3*.
  - Install another, for example USB, Ethernet controller and reconfigure the system to no longer use the network device *enp2s0f0*.

Before


Latency rack6slot2
CPU0
CPU1
Number of latency samples
1e+08
1e+07
1e+06
100000
10000
1000
100
10
1
0
50
100
150
200
250
300
350
400
Latency (us) - Maximum 38 us (plotted on 03/19/19 at 12:43:21)

After

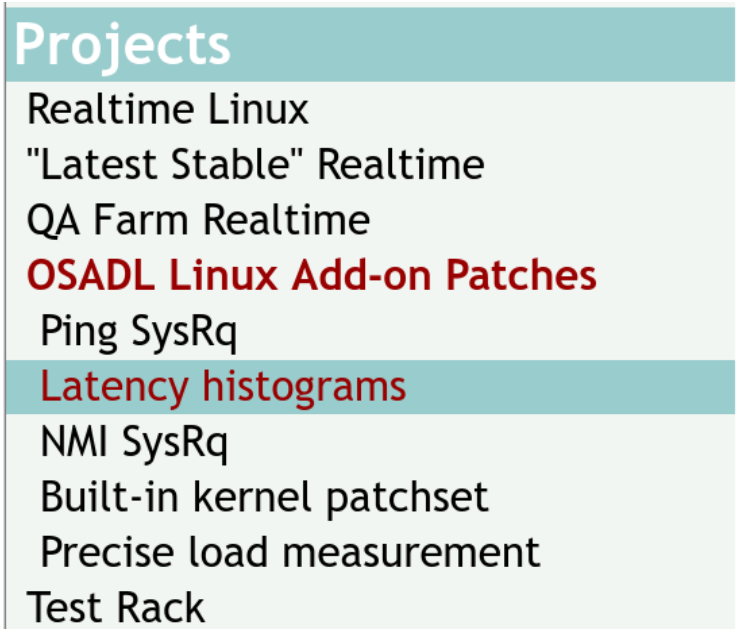

Latency rack6slot2
CPU0
CPU1
Number of latency samples
1e+08
1e+07
1e+06
100000
10000
1000
100
10
1
0
50
100
150
200
250
300
350
400
Latency (us) - Maximum 16 us (plotted on 03/21/19 at 12:43:21)

How to install the culprit/victim latency monitoring?

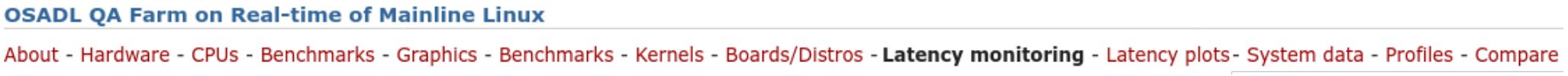
- Originally, the functionality of kernel built-in latency histograms was part of the PREEMPT_RT real-time patches.
- Later on, culprit/victim tables were added by OSADL
- In order to speed up the mainline merge process, it was decided to remove the latency histograms along with the culprit/victim tables from the real-time patches. They are now maintained separately by OSADL, and adapted versions are made available for every real-time patch version.

How to install the culprit/victim latency monitoring?

- Originally, the fur y histograms was part of the PREEMPT_
- Later on, culprit/v DL
- In order to speed was decided to remove the latency histog tables from the real-time patches. The by OSADL, and adapted versions are made ch version.



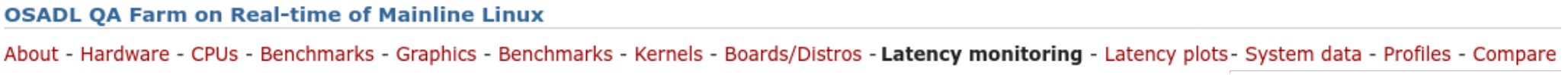
OSADL QA Farm more latency monitoring



A number of questions and problems with respect to various aspects of a system's real-time behavior cannot be answered and solved using stand-alone systems.

Therefore, several additional test scenarios have been added to the OSADL QA Farm under the heading of "Latency monitoring".

OSADL QA Farm more latency monitoring



Among other

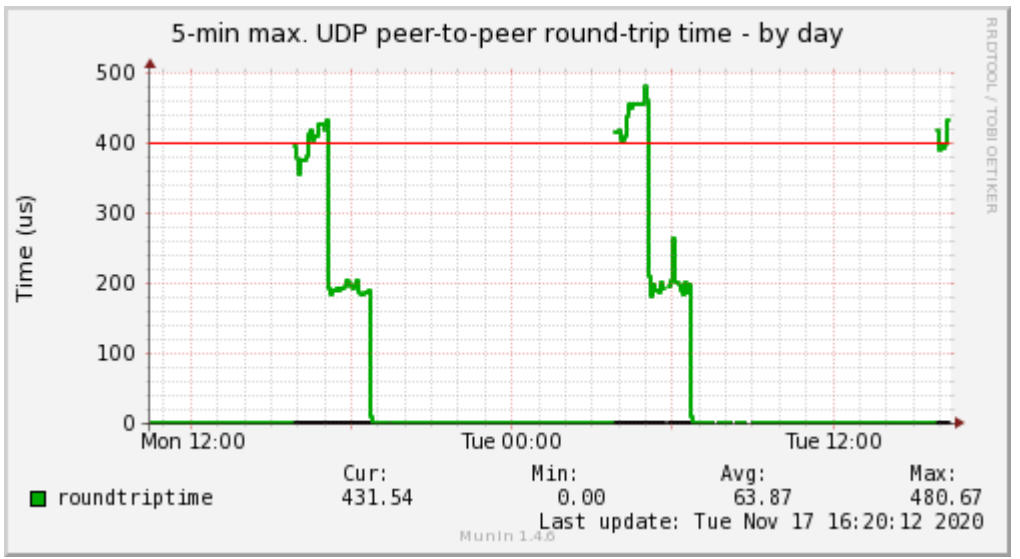
- Peer-to-peer UDP duplex link
- **OPC UA PubSub over TSN**
- Powerlink
- Ethercat
- Network load
- KVM

OSADL QA Farm latency monitoring example

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - **Latency monitoring** - Latency plots - System data - Profiles - Compare

Example:
Peer-to-peer UDP duplex link

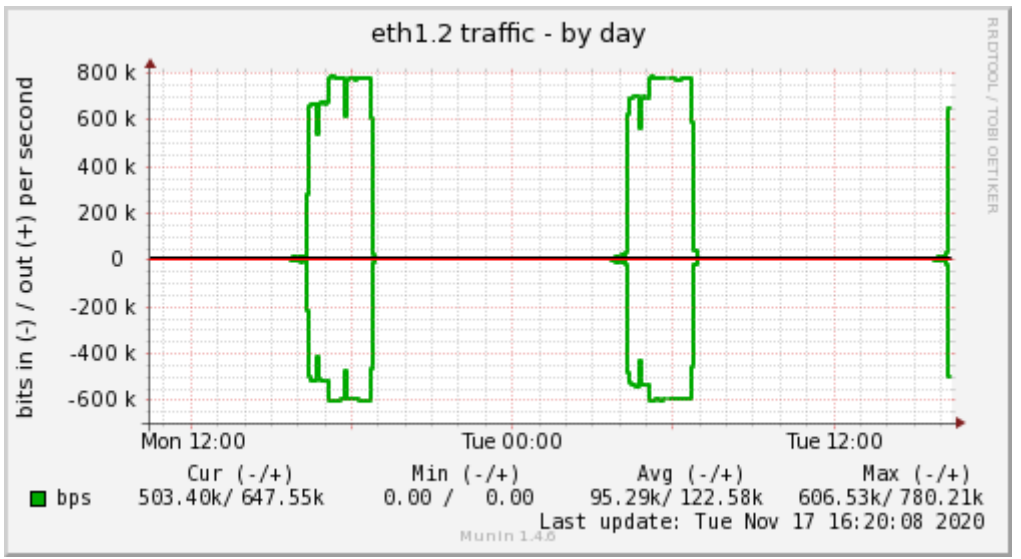


OSADL QA Farm latency monitoring example

OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - **Latency monitoring** - Latency plots - System data - Profiles - Compare

Example:
Peer-to-peer UDP duplex link

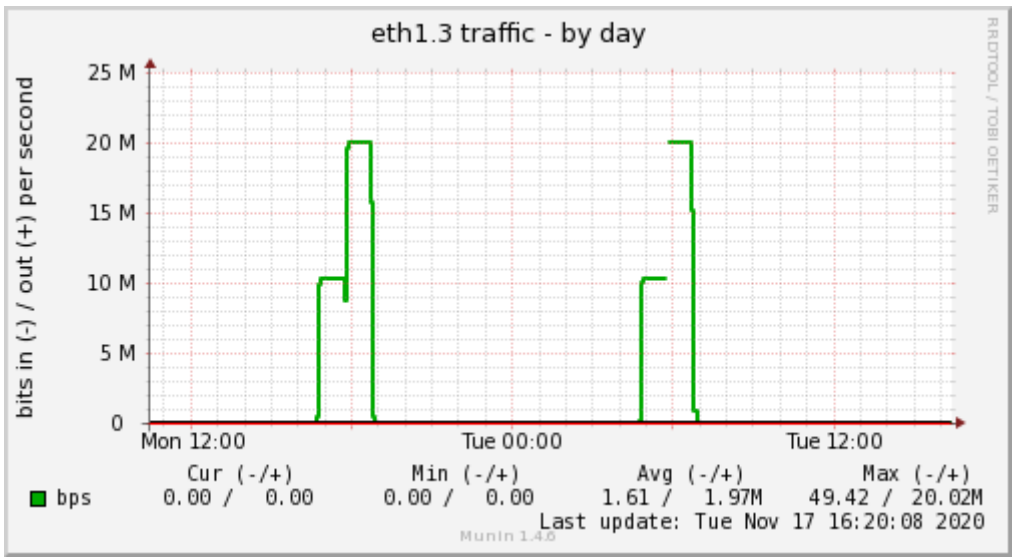


OSADL QA Farm latency monitoring example

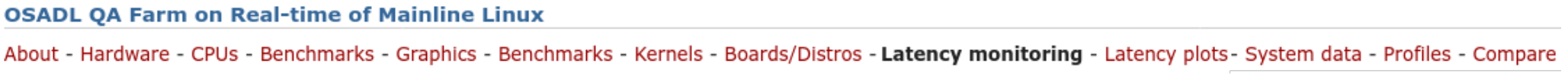
OSADL QA Farm on Real-time of Mainline Linux

About - Hardware - CPUs - Benchmarks - Graphics - Benchmarks - Kernels - Boards/Distros - **Latency monitoring** - Latency plots - System data - Profiles - Compare

Example:
Peer-to-peer UDP duplex link



Conclusion – latency monitoring

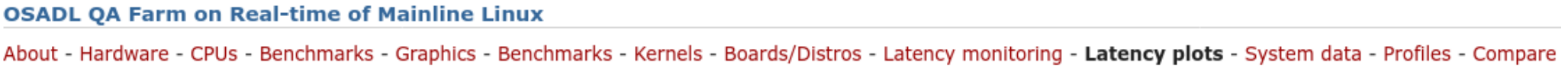


Latency monitoring is used to

- study individual scenarios in more detail
- analyze behavior of groups of systems

The setup may change frequently as new test strategies may be added at any time.

Conclusion – latency plots



Latency plots are used to

- gain a more general overview about the real-time behavior of systems
- select a system that is best suited for a particular purpose
- maintain systems during their life cycle
- detect performance regressions

The setup is rather constant and based on general test strategies.