Accelerated graphics with real-time
How do they play together?

Theoretical part: Foundations and underlying principles of Linux real-time

Carsten Emde
Open Source Automation Development Lab (OSADL) eG

Main topics

- What means “real time”/“real-time”?
- How can we determine, whether a system is a “real-time” system?
- Typical real-time requirements
- A proposal of a better word for “real-time”
- Rationale for the OSADL QA Farm
- How to use the results of the OSADL QA Farm

What means “real time/real-time”?

- The term **“real time”** (the adjective “real” followed by the noun “time”) and the term **“real-time”** (a compound adjective of the components “real” and “time” that are separated by a dash) have nothing in common.

What means “real time/real-time”?

- The term **“real time”** (the adjective “real” followed by the noun “time”) and the term **“real-time”** (a compound adjective of the components “real” and “time” that are separated by a dash) have nothing in common.
- The term **“real time”** means the correct time of the day at a given moment in a given time zone on Earth.

What means “real time/real-time”?

- The term **“real time”** (the adjective “real” followed by the noun “time”) and the term **“real-time”** (a compound adjective of the components “real” and “time” that are separated by a dash) have nothing in common.
- The term **“real-time”**, e.g. in “real-time system”, means a deterministic response behavior: Under any circumstances will such a system be able to appropriately respond to an asynchronously arriving unpredictable external or internal event.

Why may this similarity be a problem?

- Because the only distinguishing feature between “real time” and “real-time” – space or dash – may be omitted under certain conditions:
  - In variable names of computer languages
  - In spoken language

Why may this similarity be a problem?

- Because the only distinguishing feature between “real time” and “real-time” – space or dash – may be omitted under certain conditions:
  - In variable names of computer languages
  - In spoken language
- Because there are environments where both meanings can be meaningful and confusion can lead to misbehavior.

Example of misbehavior in case of confusion

A number of POSIX calls such as

`clock_getres(clockid_t clk_id, struct timespec *res)`,
`clock_gettime(clockid_t clk_id, struct timespec *tp)` and
`clock_settime(clockid_t clk_id, const struct timespec *tp)`

expect the type of the clock (`clk_id`) as first argument.

There is **one clock** that delivers the correct and, thus, settable time of the day (the "real time"), but the space was omitted in the POSIX defined name `CLOCK_REALTIME`. This has tempted many developers of real-time systems to erroneously select this clock.

Example of misbehavior in case of confusion

A number of POSIX calls such as

`clock_getres(clockid_t clk_id, struct timespec *res)`,
`clock_gettime(clockid_t clk_id, struct timespec *tp)` and
`clock_settime(clockid_t clk_id, const struct timespec *tp)`

expect the type of the clock (`clk_id`) as first argument.

There is **another clock** that always must be used in a deterministic ("real-time") system, since it is not settable and will not be affected by any state change of the system, and this clock is called `CLOCK_MONOTONIC` or `CLOCK_MONOTONIC_RAW` in Linux.

Another important difference

Another important difference between “real time” and “real-time” is that

- “real time” of a system can be determined (e.g. using the POSIX call `clock_gettime()`)
- whether a system is “real-time” can hardly be determined, but must be designed. It can only be stated that a system is **not** “real-time” when the latency threshold is violated.

Another important difference

Another important difference between “real time” and “real-time” is that

- “real time” of a system can be determined (e.g. using the POSIX call `clock_gettime()`)
- whether a system is “real-time” can hardly be determined, but must be designed. It can only be stated that a system is **not** “real-time” when the latency threshold is violated.

But we need to know whether a system is “real-time”, before we can use it for a time-critical task.

How to measure “real-time”?

All methods to measure “real-time” have in common that either external signals are sent to the system or internal signals are generated within the system and the system is challenged to respond in time.

How to measure “real-time”?

All methods to measure “real-time” have in common that either external signals are sent to the system or internal signals are generated within the system and the system is challenged to respond in time. During this challenge, as many different other task as possible that might interfere with the real-time task are simultaneously executed in the hope that all possible system conditions are met and no other condition will happen later on in the field and prevent the system from being “real-time”.

We measure „conditional latency“, not „real-time“!

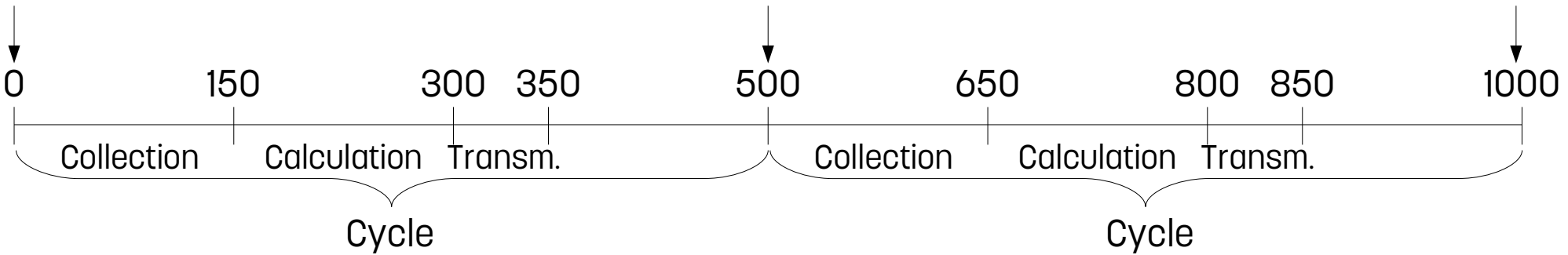
However, it must be made clear, that “real-time” – determinism – cannot be proven by a measurement. Only a certain probability can be established that the system will meet the defined **real-time condition**.

We measure „conditional latency“, not „real-time“!

However, it must be made clear, that “real-time” – determinism – cannot be proven by a measurement. Only a certain probability can be established that the system will meet the defined **real-time condition**. To establish such probability, the real-time condition must be known, but this depends on the use case. In consequence, a system never can be an absolute “real-time system”, but only a “system that meets the requirements of a given use case”.

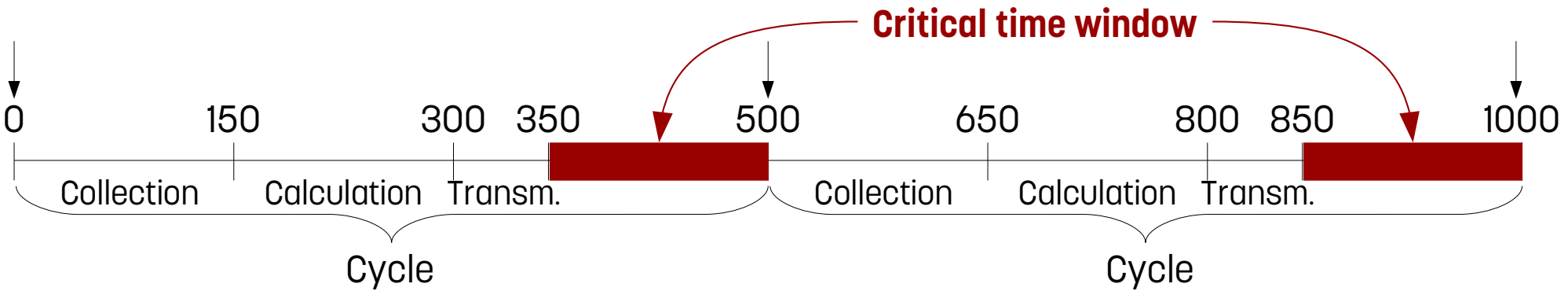
Typical real-time requirements

A typical real-time system may need to adjust the system every 500 µs, require 150 µs for data collection, another 150 µs for internal calculations, and 50 µs to transmit the data.



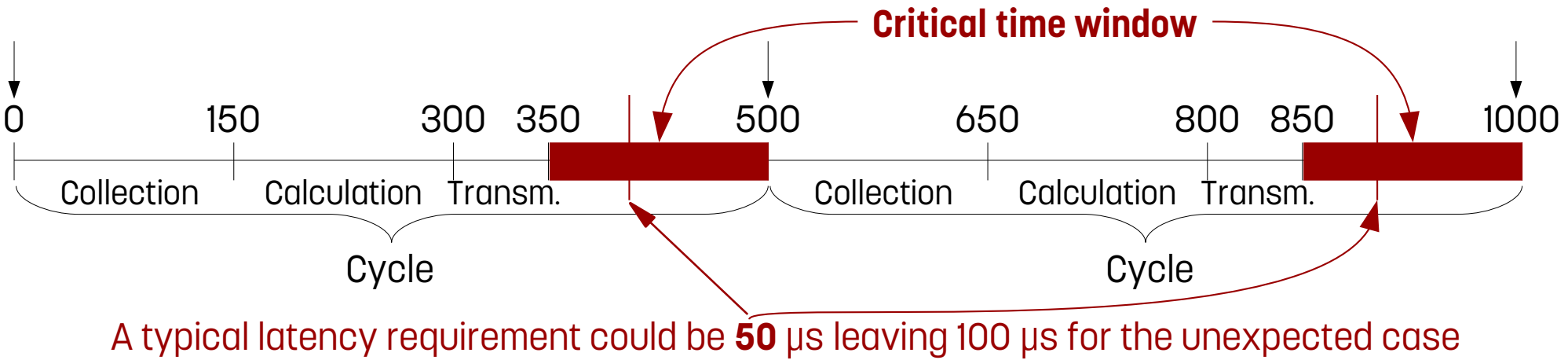
Typical real-time requirements

A typical real-time system may need to adjust the system every 500 µs, require 150 µs for data collection, another 150 µs for internal calculations, and 50 µs to transmit the data.



Typical real-time requirements

A typical real-time system may need to adjust the system every 500 µs, require 150 µs for data collection, another 150 µs for internal calculations, and 50 µs to transmit the data.



A typical latency requirement could be **50** µs leaving 100 µs for the unexpected case

A proposal of a better word for „real-time“

Since 50 µs maximum acceptable latency may be “real-time” for one system, but not for another system, the term “real-time” should always include the numerical value of the maximum acceptable latency.

A proposal of a better word for „real-time“

Since 50 µs maximum acceptable latency may be “real-time” for one system, but not for another system, the term “real-time” should always include the numerical value of the maximum acceptable latency.

It, therefore, is proposed to better call a “real-time” system a “deterministic system”, to abbreviate it “D system” and to add a subscript number that gives the maximum acceptable latency in microseconds, thus D_{50} in our example.

What maximum latency is achievable?

Experience from many years of studying system latency of real-time systems with a large variety of CPU architectures and system components has taught us the following "rule of thumb" of system latency:

$$\frac{1}{clock\,frequency} \times 10^5$$

What maximum latency is achievable?

Experience from many years of studying system latency of real-time systems with a large variety of CPU architectures and system components has taught us the following "rule of thumb" of system latency:

$$\frac{1}{100 \times 10^6} \times 10^5 = 0.01 \times 10^{-1} s = 1ms$$

Example 100 MHz

What maximum latency is achievable?

Experience from many years of studying system latency of real-time systems with a large variety of CPU architectures and system components has taught us the following "rule of thumb" of system latency:

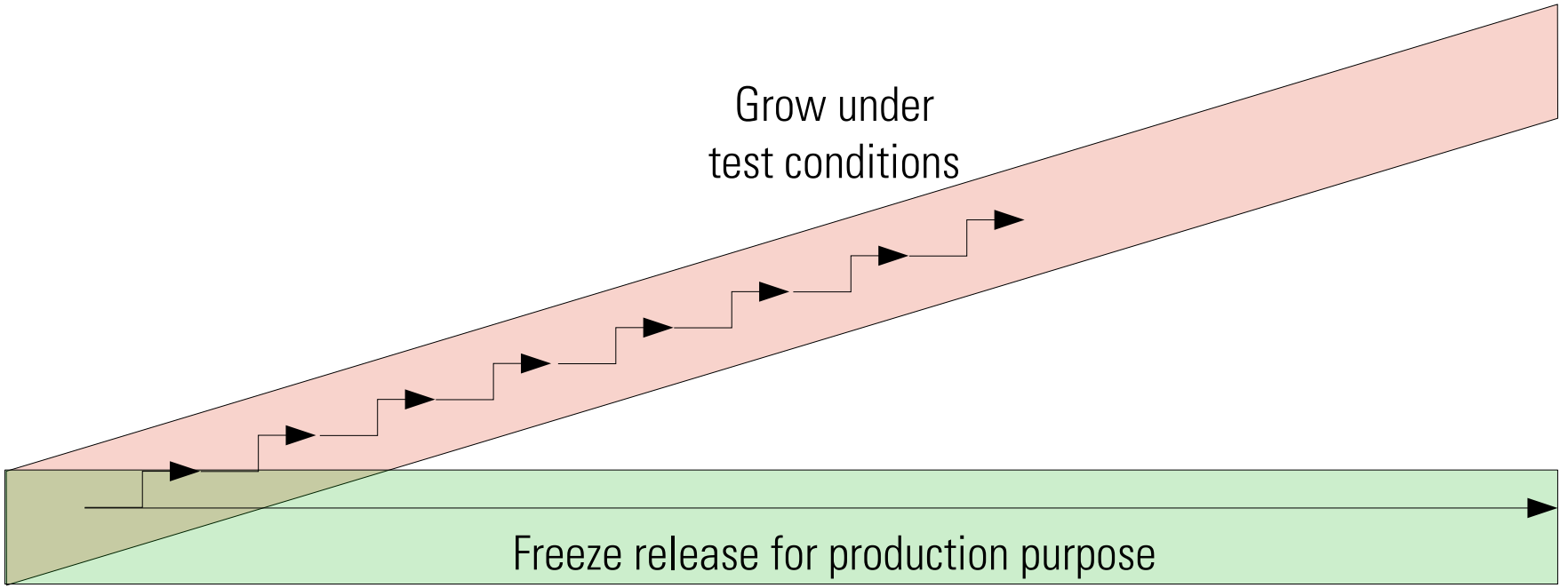
$$\frac{1}{2.5 \times 10^9} \times 10^5 = 0.4 \times 10^{-4} s = 40 \mu s$$

Example 2.5 GHz

Rationale for the OSADL QA Farm

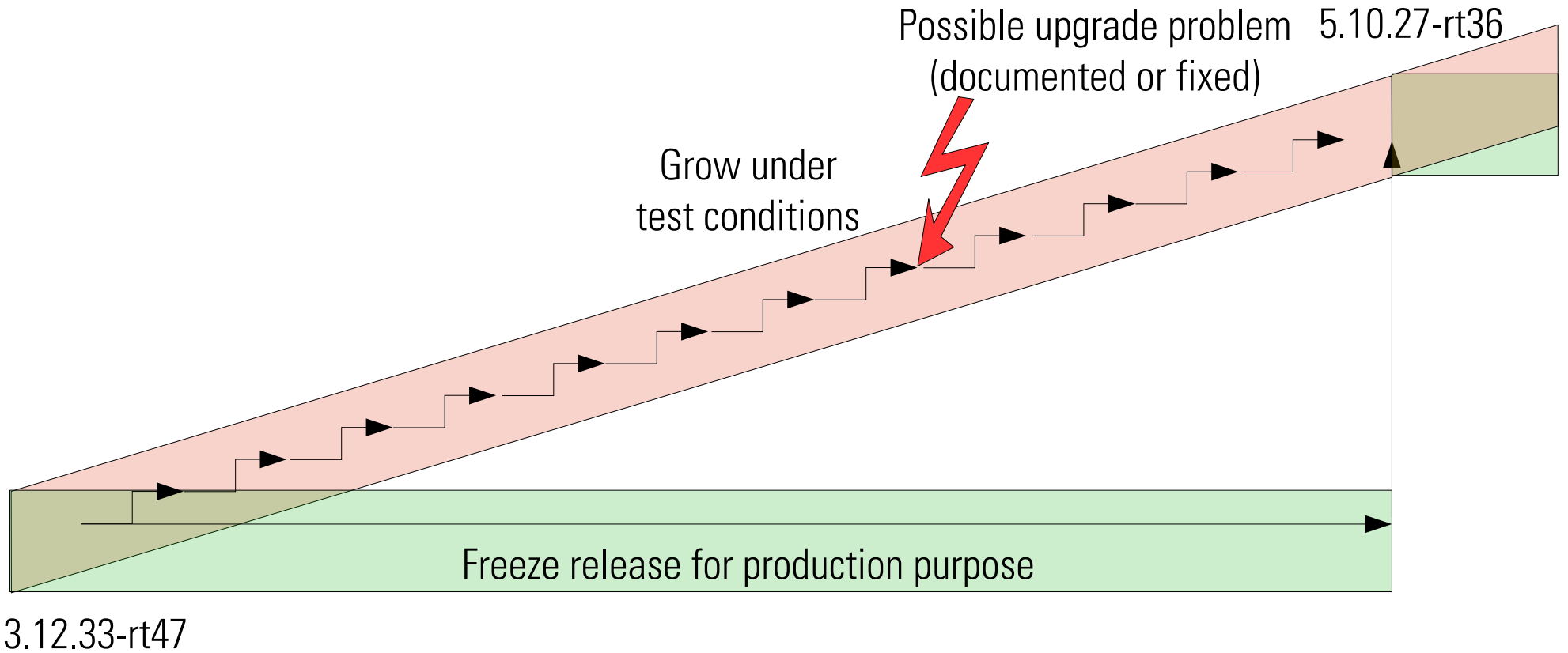
- Although we know that we cannot prove determinism, we still can measure long enough and under as many different conditions as possible to decrease the **probability** that a given system will once fail to meet the latency requirement.
- This was the reason to create the OSADL QA Farm.
- In addition, we wanted to provide a solution for the problem that a system configuration must be frozen for production, but be upgradable for maintenance.

"Freeze'n grow"



3.12.33-rt47

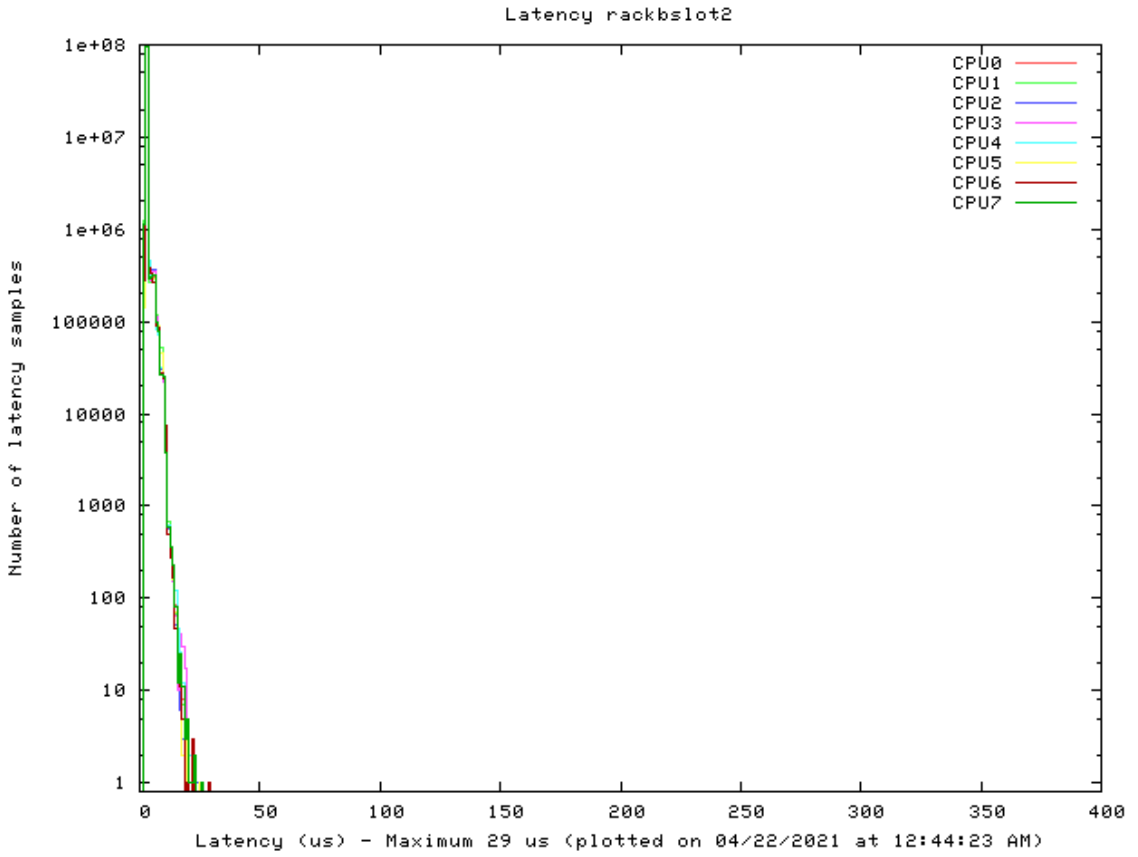
"Freeze'n grow"



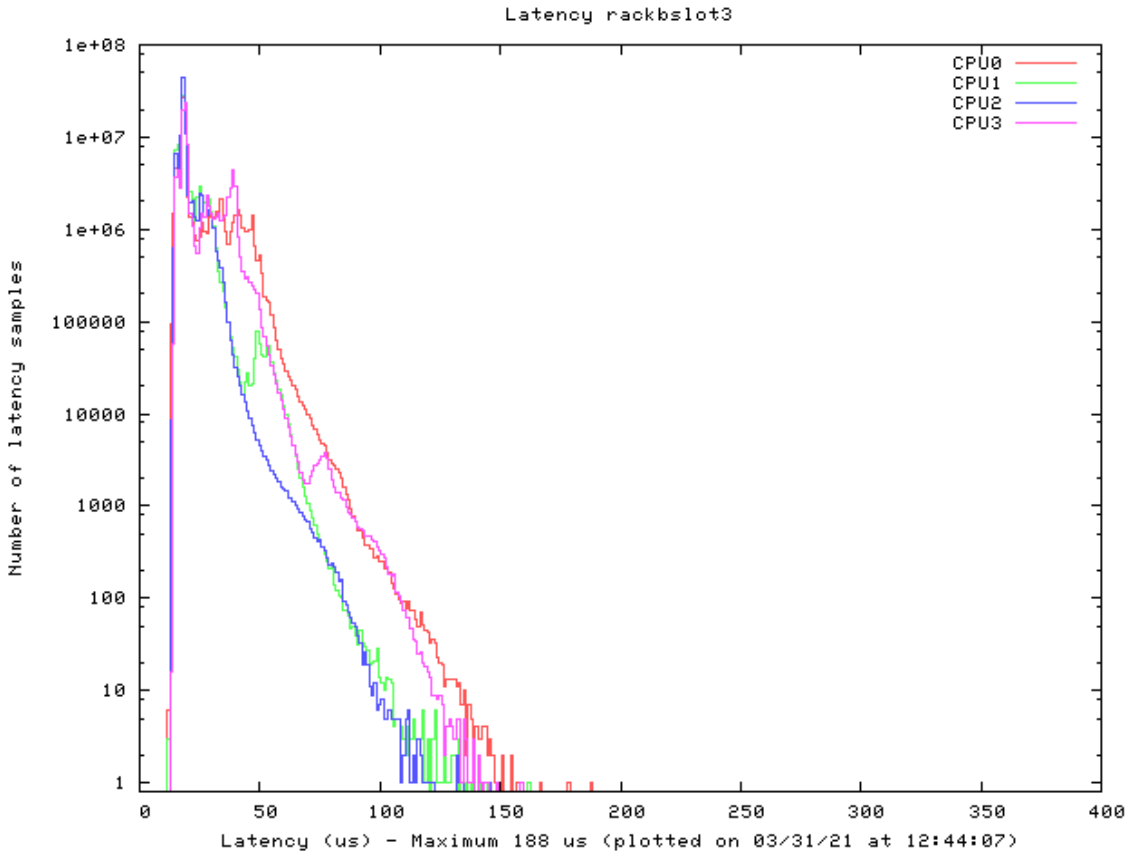
How to use the results of the OSADL QA Farm

- The most important measurements of the OSADL QA Farm focus on latency. Whenever possible, we trigger twice the day with 100 millions cycles with an interval of 200 µs. This takes about 5 hours and 33 minutes per measurement.
- The results are displayed in form of “latency plots”.
- A “latency plot” is kind of a frequency plot (aka histogram) with linear latency classes (scaled in microseconds) in the x and logarithmic frequency values is the y scale.

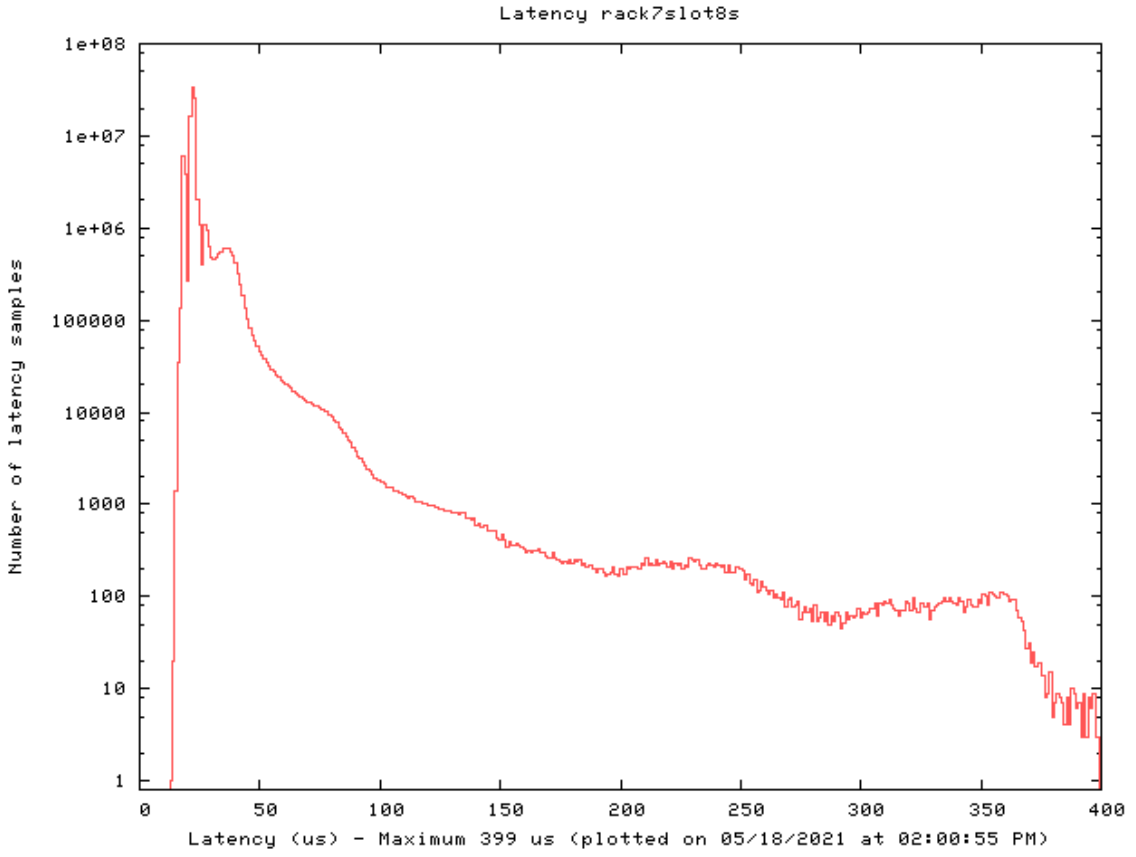
A typical „good“ latency plot



A typical „less good" latency plot



A typical „bad“ latency plot



Conclusion

- “Real time” and “real-time” are two completely different terms. Care must be taken to not confuse them.
- Determinism cannot be proven, but worst-case latency can be determined with a certain probability.
- The OSADL QA Farm serves to select, test and confirm computer systems intended to be used under real-time conditions.

Conclusion

- “Real time” and “real-time” are two completely different terms. Care must be taken to not confuse them.
- Determinism cannot be proven, but worst-case latency can be determined with a certain probability.
- The OSADL QA Farm serves to select, test and confirm computer systems intended to be used under real-time conditions.

Let’s have a closer look!

Copyright © 2021 Open Source Automation Development Lab (OSADL) eG