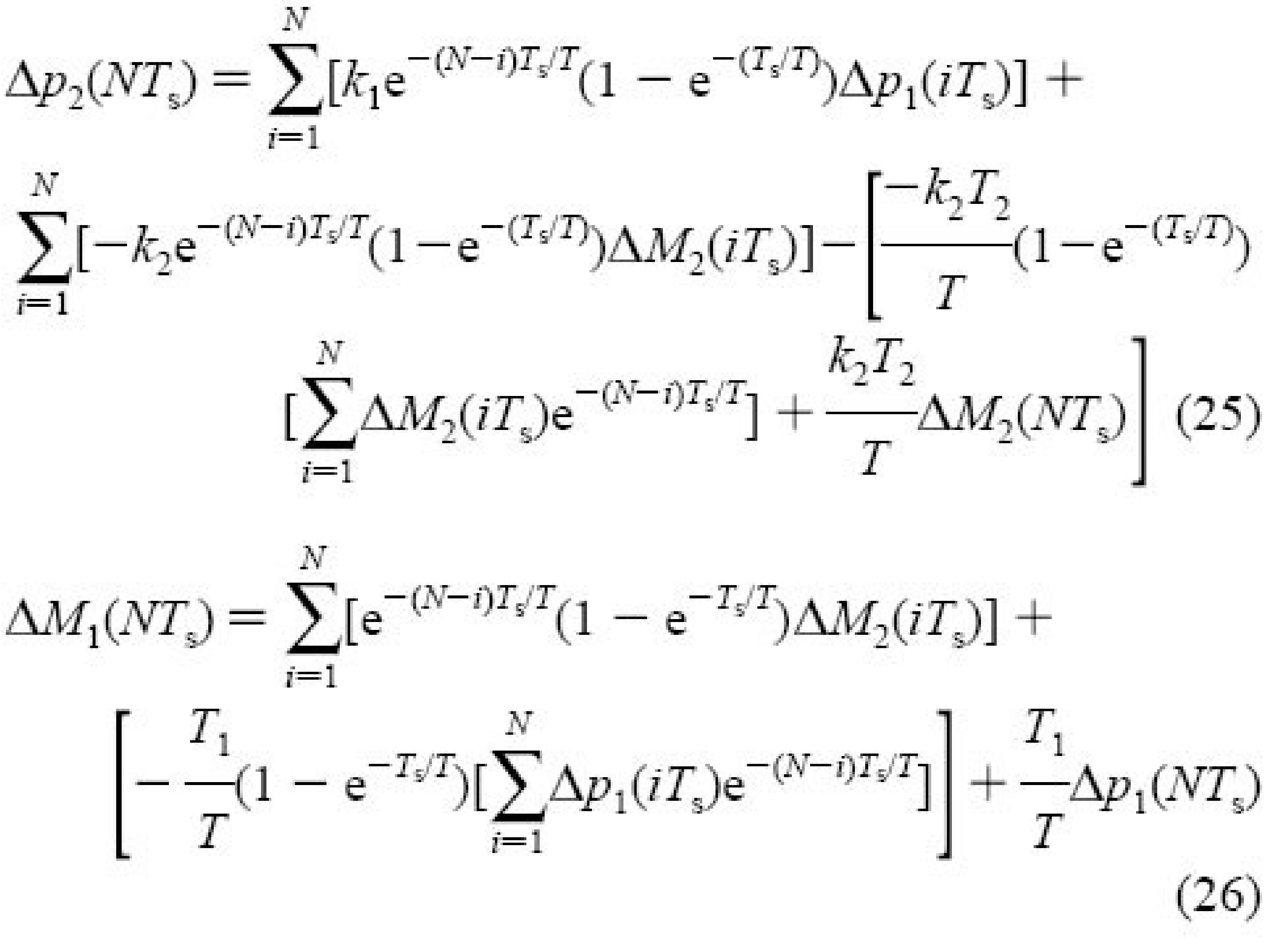
A Quest Against Time

- Why timekeeping is hard
- What we can do without guest help
- What we can do with guest help

TIME IS HARD

TIME IS HARD

- Not this hard...



$$
\begin{aligned}
\Delta p_2(NT_s) = {} & \sum_{i=1}^{N}[k_1 e^{-(N-i)T_s/T}(1 - e^{-(T_s/T)})\Delta p_1(iT_s)] + \\
& \sum_{i=1}^{N}[-k_2 e^{-(N-i)T_s/T}(1 - e^{-(T_s/T)})\Delta M_2(iT_s)] - \Bigg[\frac{-k_2 T_2}{T}(1 - e^{-(T_s/T)}) \\
& [\sum_{i=1}^{N}\Delta M_2(iT_s)e^{-(N-i)T_s/T}] + \frac{k_2 T_2}{T}\Delta M_2(NT_s)\Bigg]
\end{aligned}
\tag{25}
$$

$$
\begin{aligned}
\Delta M_1(NT_s) = {} & \sum_{i=1}^{N}[e^{-(N-i)T_s/T}(1 - e^{-T_s/T})\Delta M_2(iT_s)] + \\
& \Bigg[-\frac{T_1}{T}(1 - e^{-T_s/T})[\sum_{i=1}^{N}\Delta p_1(iT_s)e^{-(N-i)T_s/T}]\Bigg] + \frac{T_1}{T}\Delta p_1(NT_s)
\end{aligned}
\tag{26}
$$

TIME IS HARD

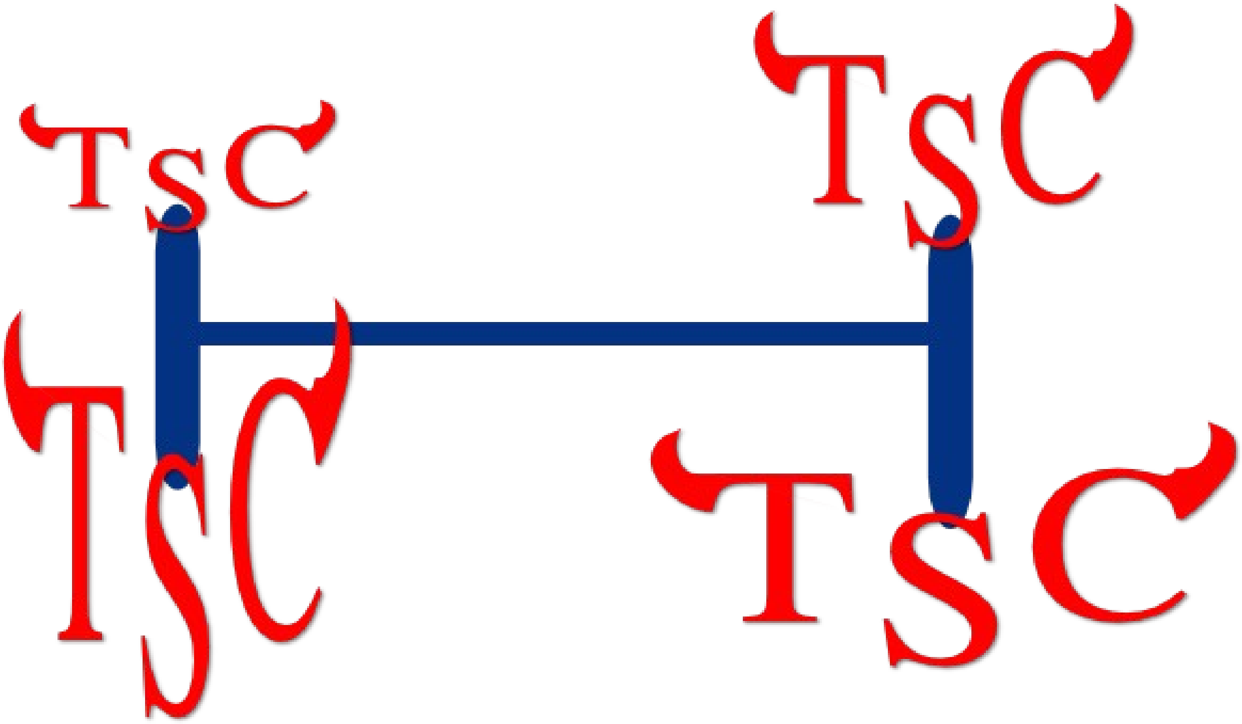
- Not this hard...
- It's worse

Already hard on bare metal

Highest resolution clock is very problematic

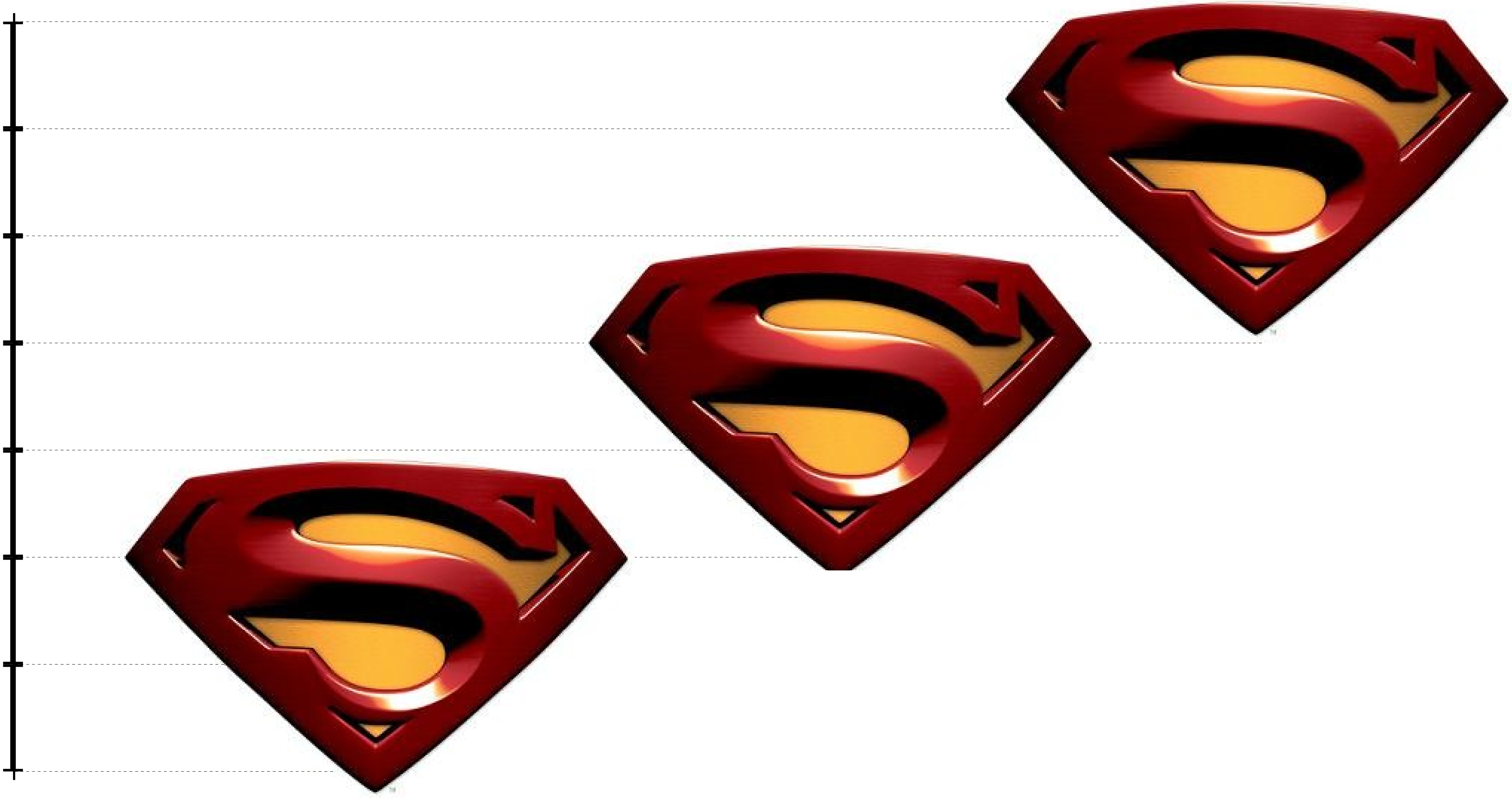
Reaching agreement is hard (inter-cpu drift)

Reaching agreement is hard (inter-socket drift)



Reaching agreement is hard (thermal effects)

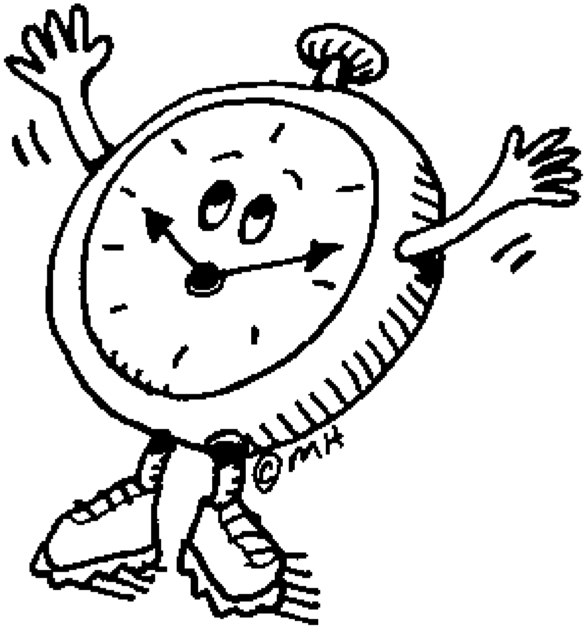
Reaching agreement is hard (super-scalar execution)



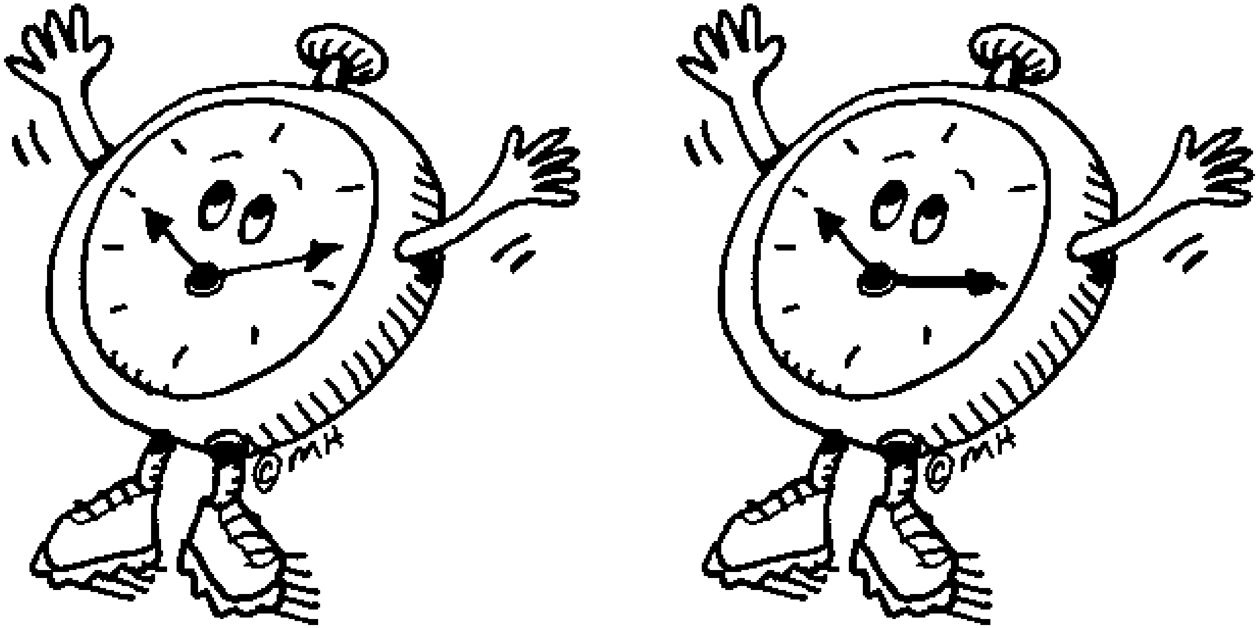
Reaching agreement is hard (hotplug CPUs)

Under virtualization, basic assumptions can break

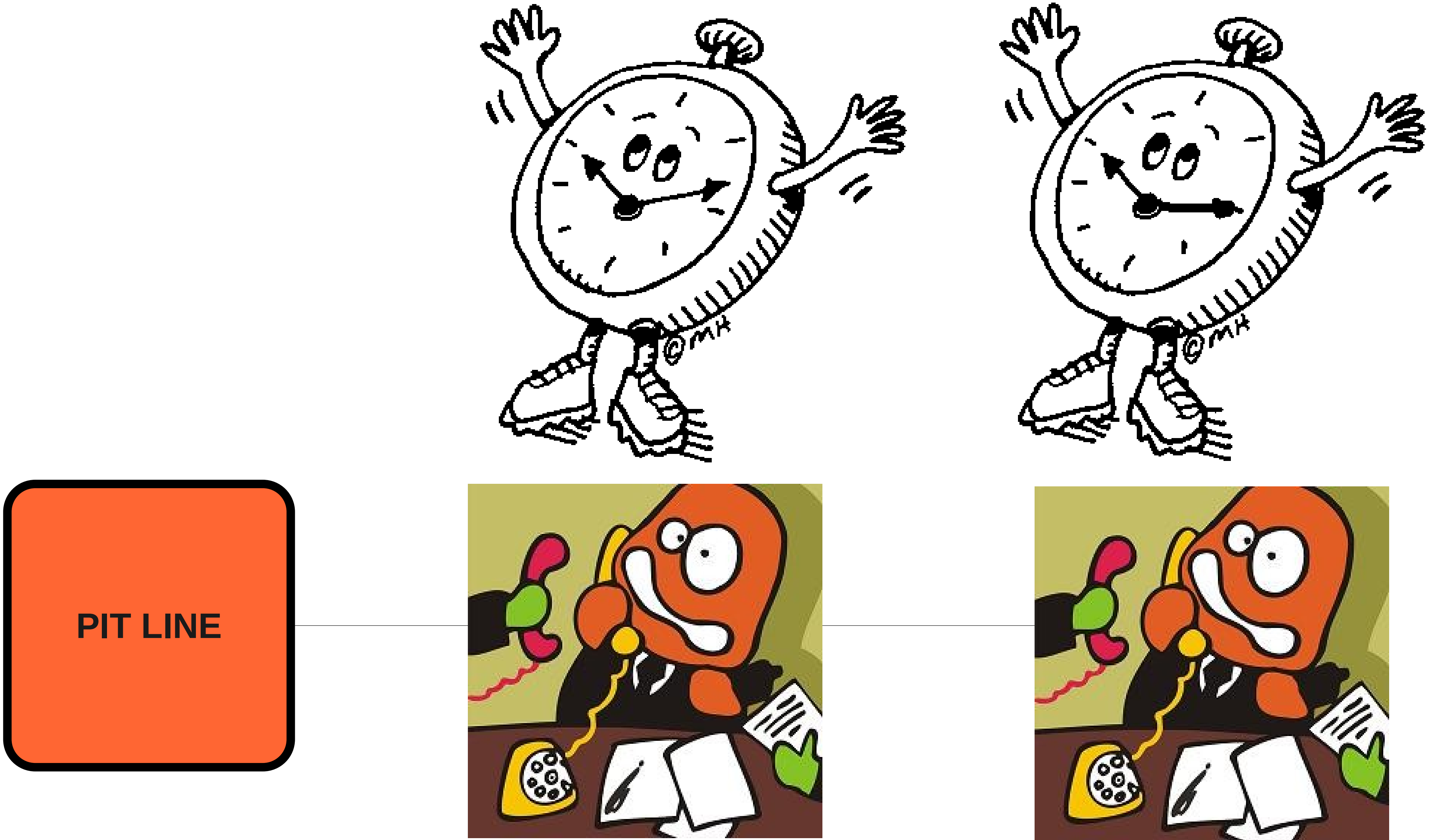
Every measurement is an observation...



And every observation must be consistent....

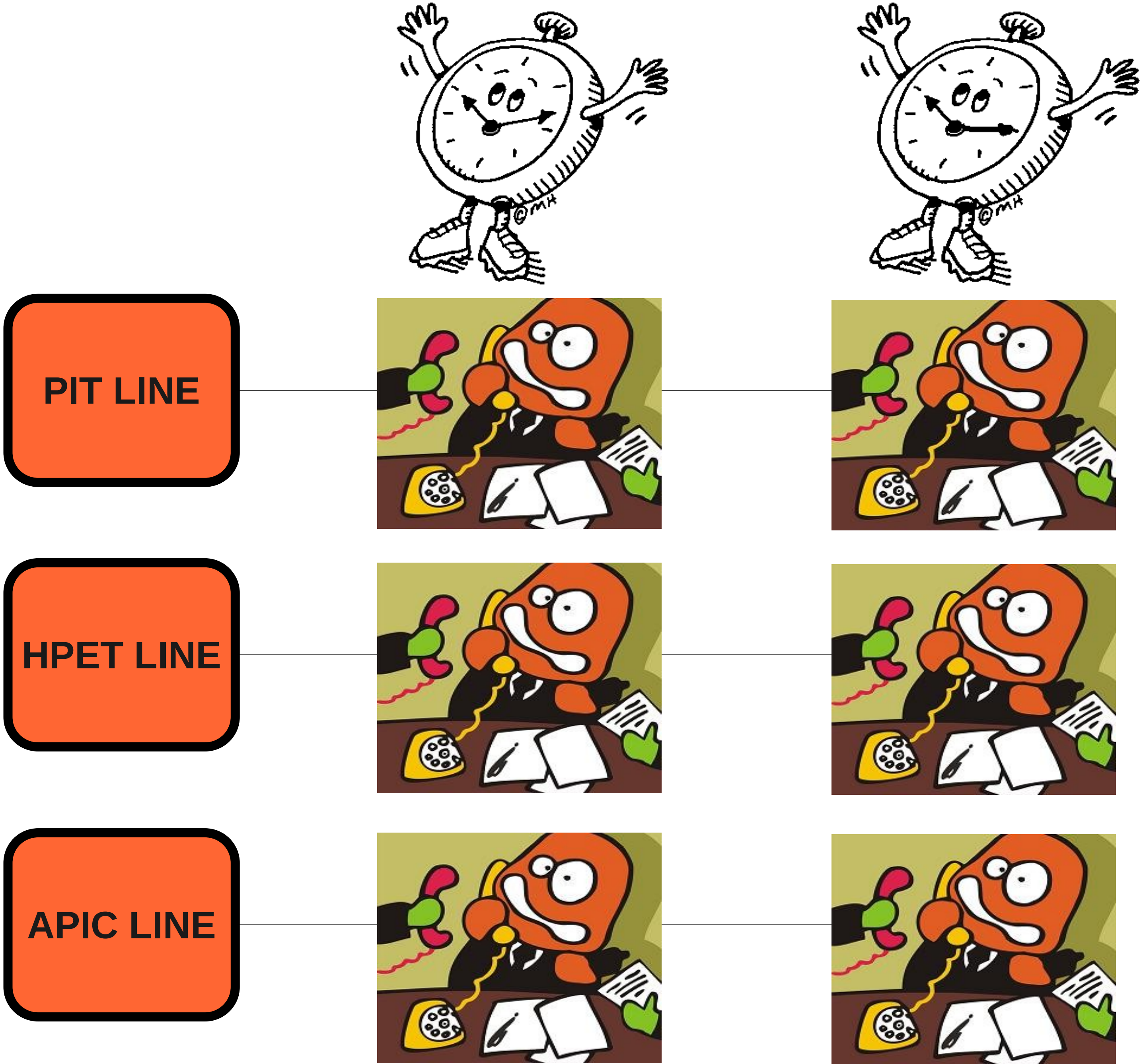


Not just with itself, but with other clock interrupts



PIT LINE

And there are many of these



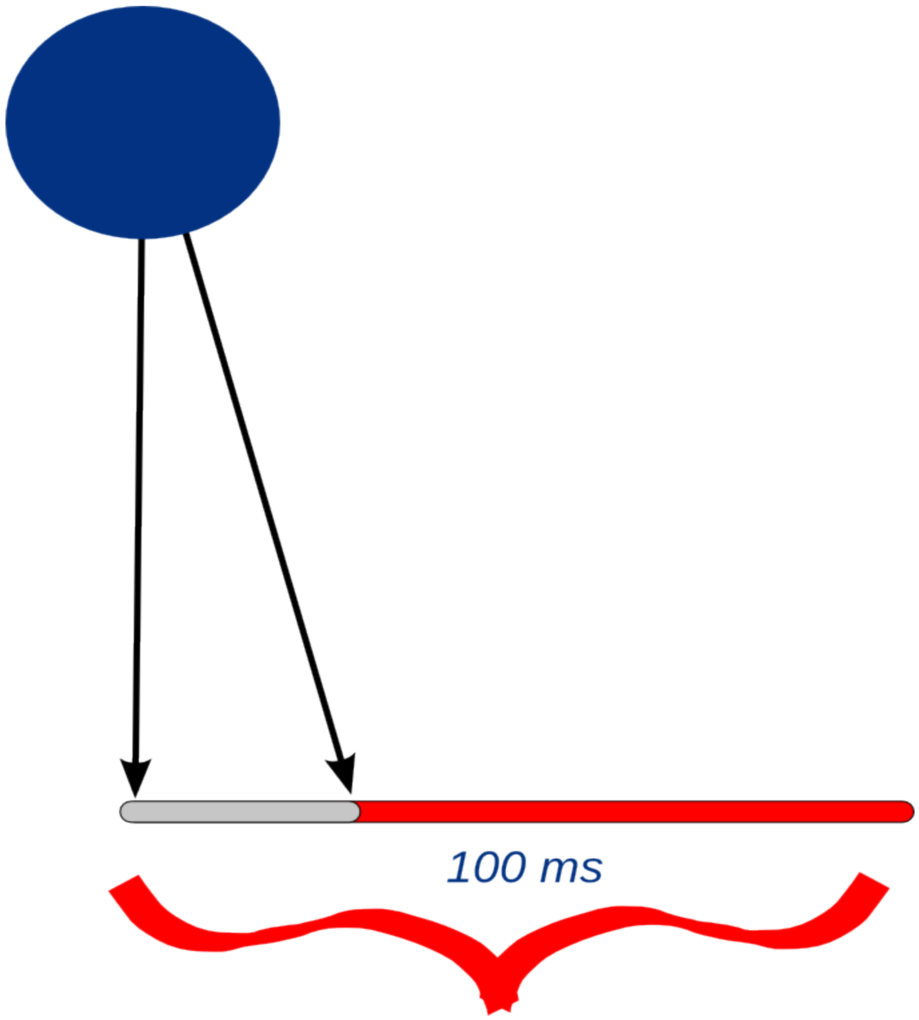
PIT LINE

HPET LINE

APIC LINE

Interrupts delivered, guest is out

Delay to resuming guest

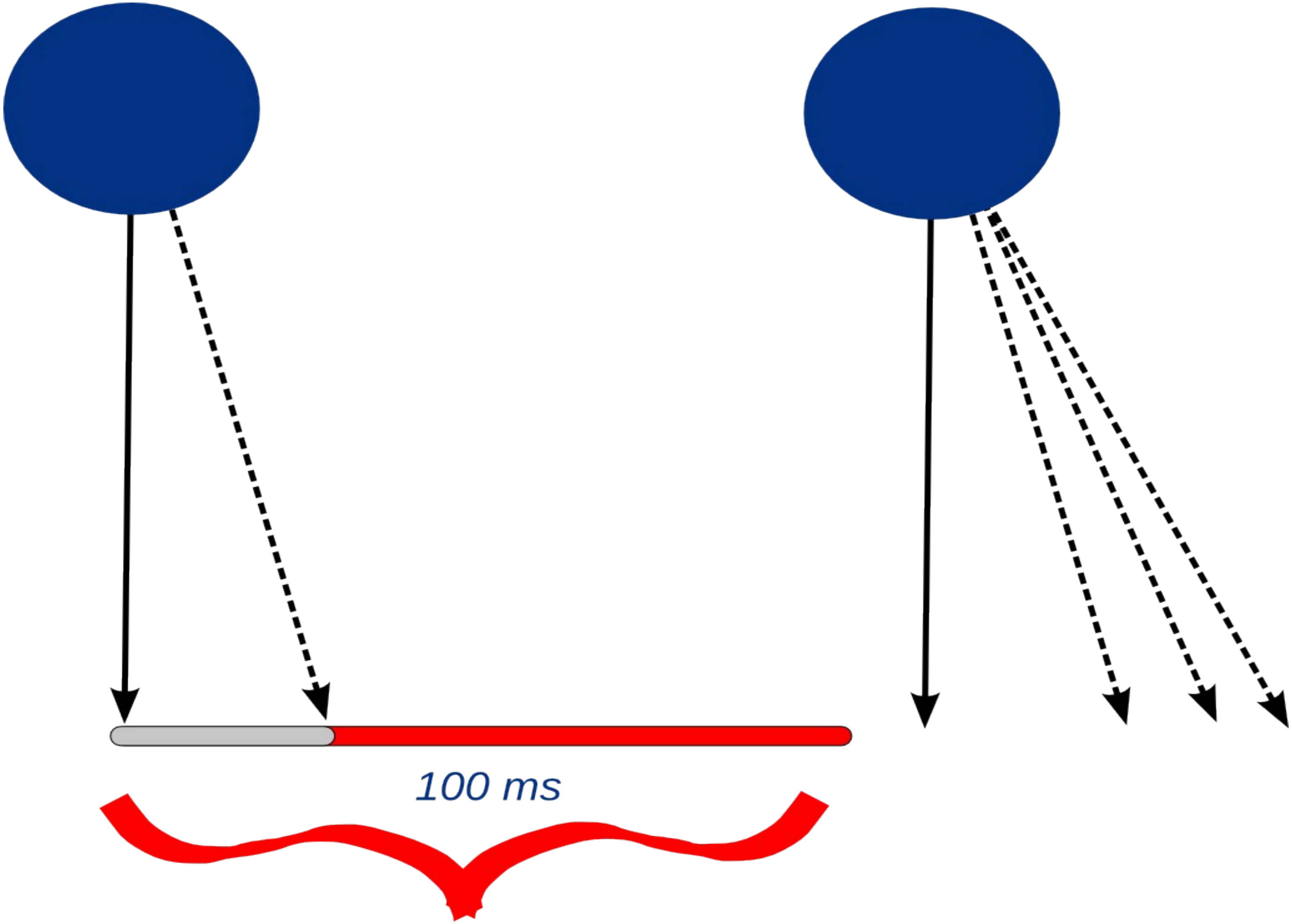


On-time delivery is a hard target to hit, especially with multiple guests

How will guest deal with lateness?

How will guest deal with lateness?

Interrupt Re-injection



Requires a lot of CPU

Ideally, not rely on interrupts

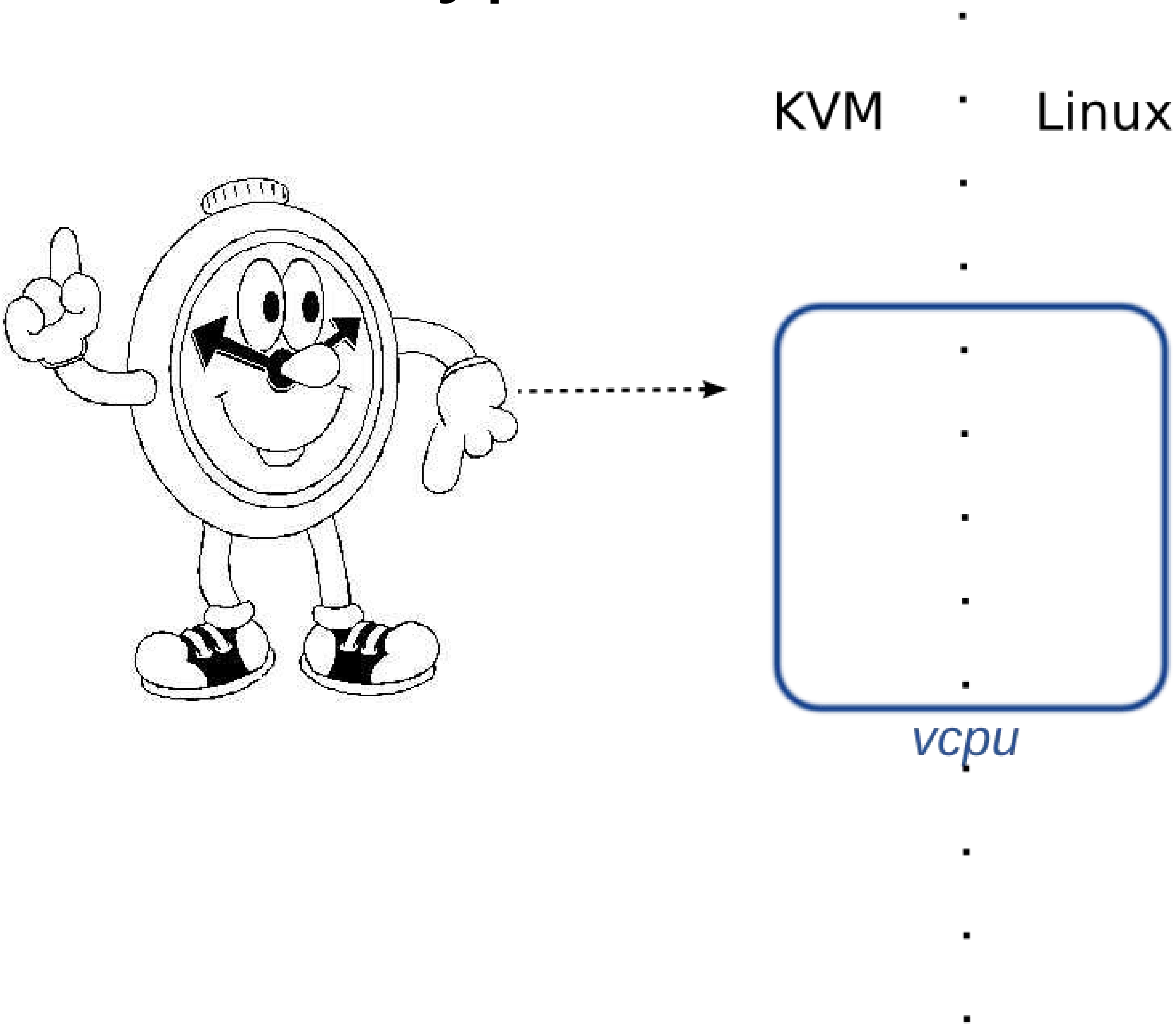
- Read clock timestamp directly (modern linux clocksources)

Guest Based Compensation

- Read clock timestamp directly (modern linux clocksources) => and then figure out how many ticks we should account.
- Requires accurate TSC

KVM

Hypervisor tells time

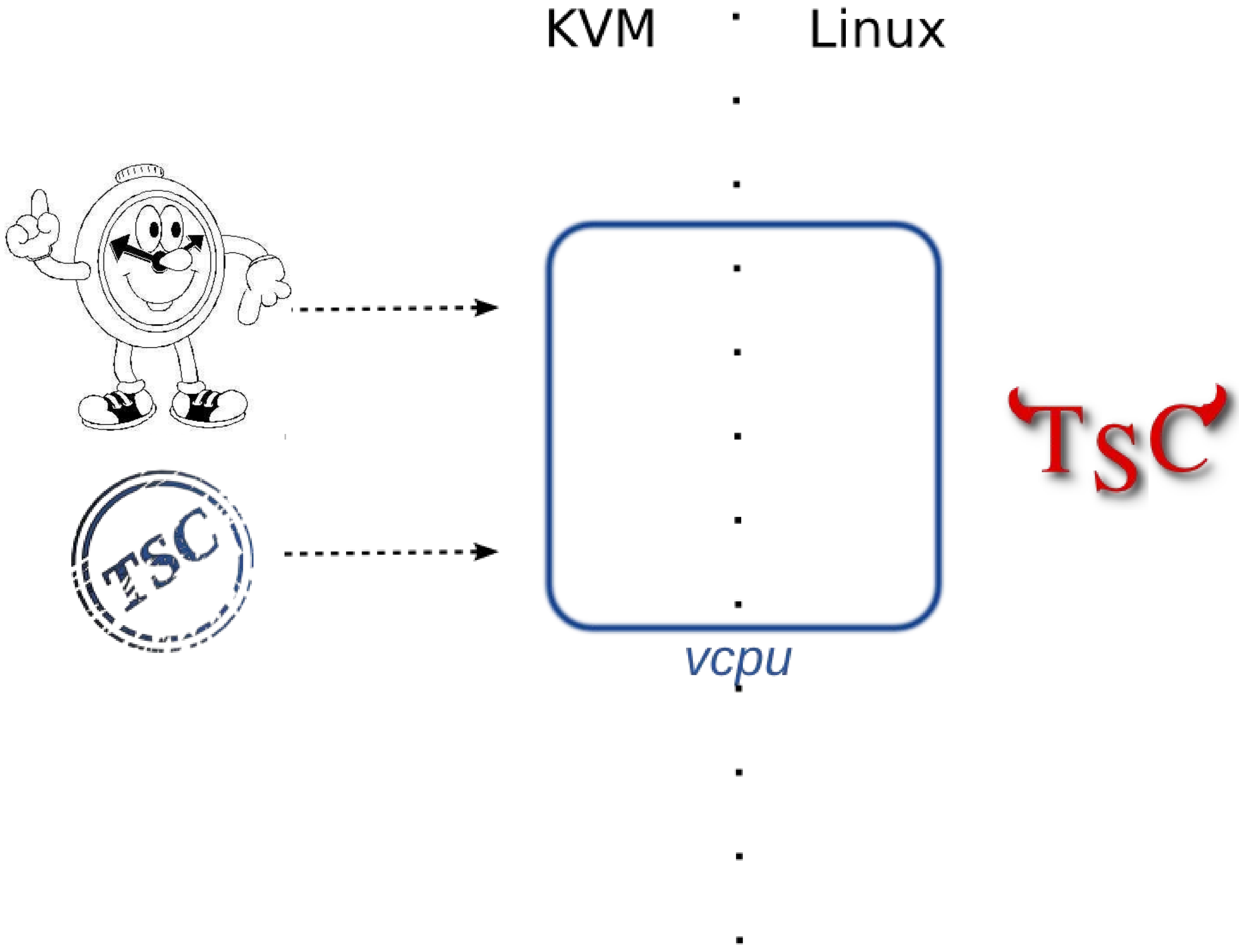


KVM

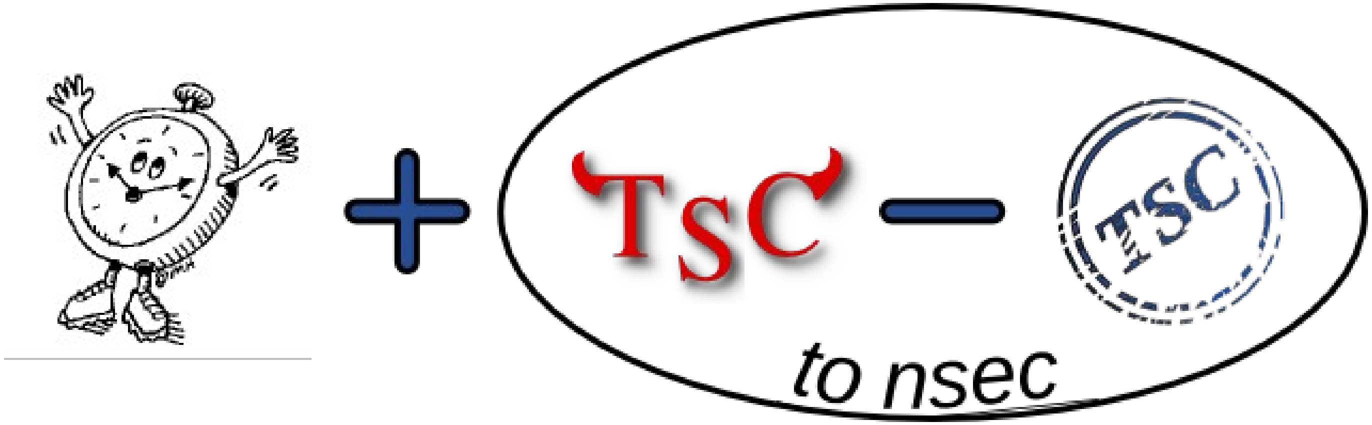
Linux

vcpu

Adjust locally with tsc

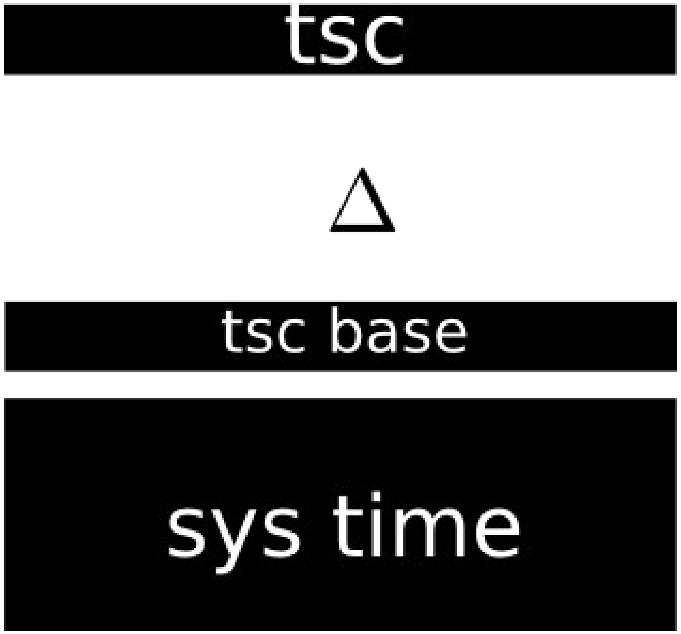


Adjust locally with tsc



to nsec

The picture



tsc

Δ

tsc base

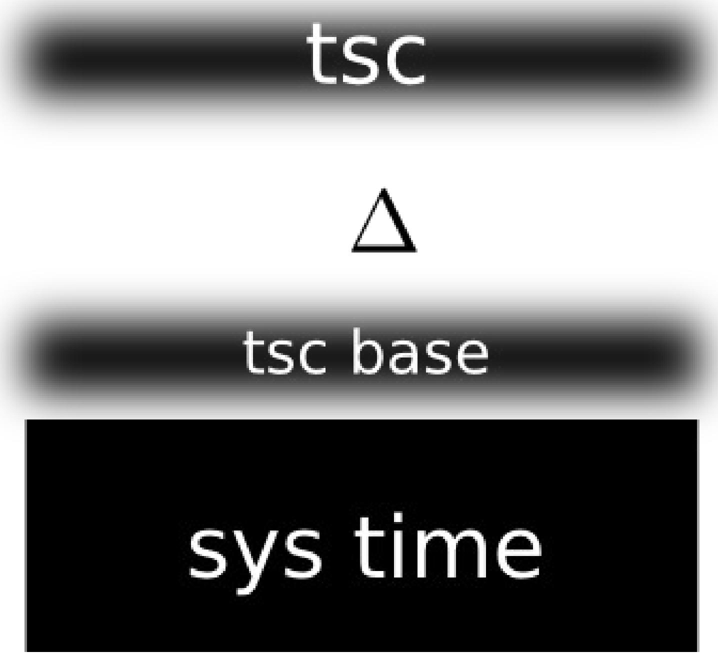
sys time

Must be done carefully

TSC and host clock may run at different resolutions

TSC has issues

Even if everything works ok



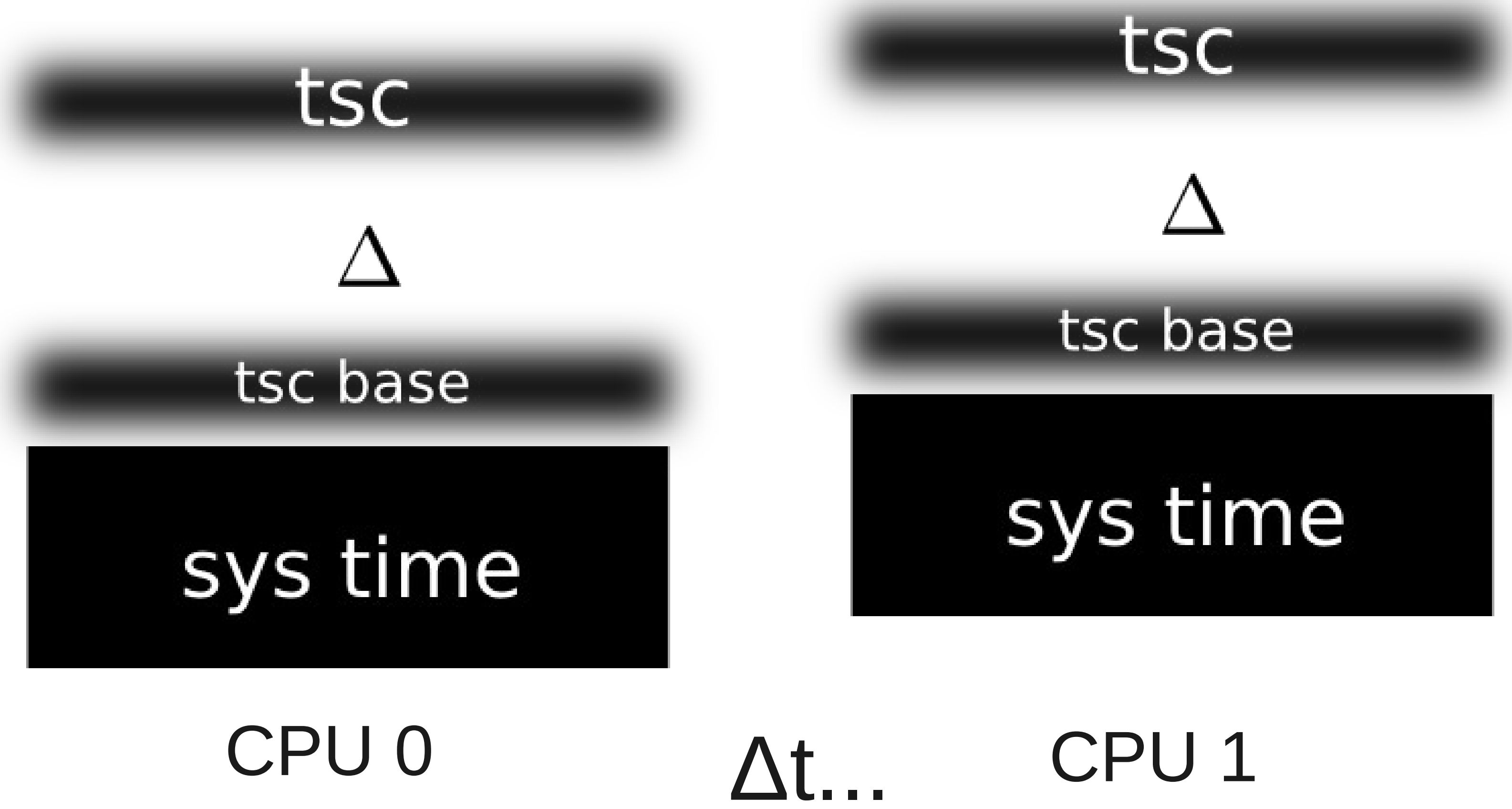
tsc

Δ

tsc base

sys time

Recalibration has serious issues



tsc

Δ

tsc base

sys time

CPU 0

Δt...

tsc

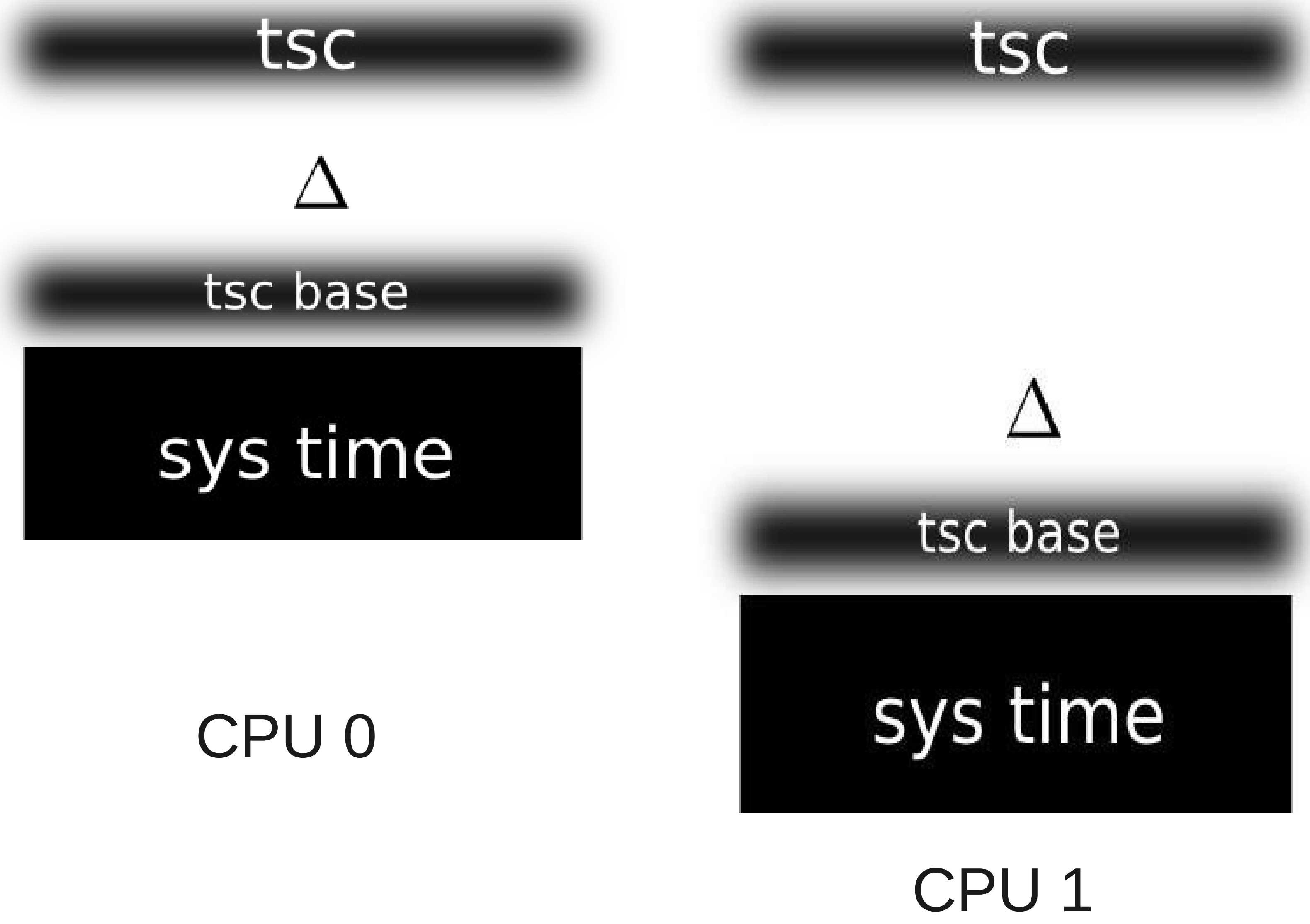
Δ

tsc base

sys time

CPU 1

As does SMP



tsc

Δ

tsc base

sys time

CPU 0

tsc

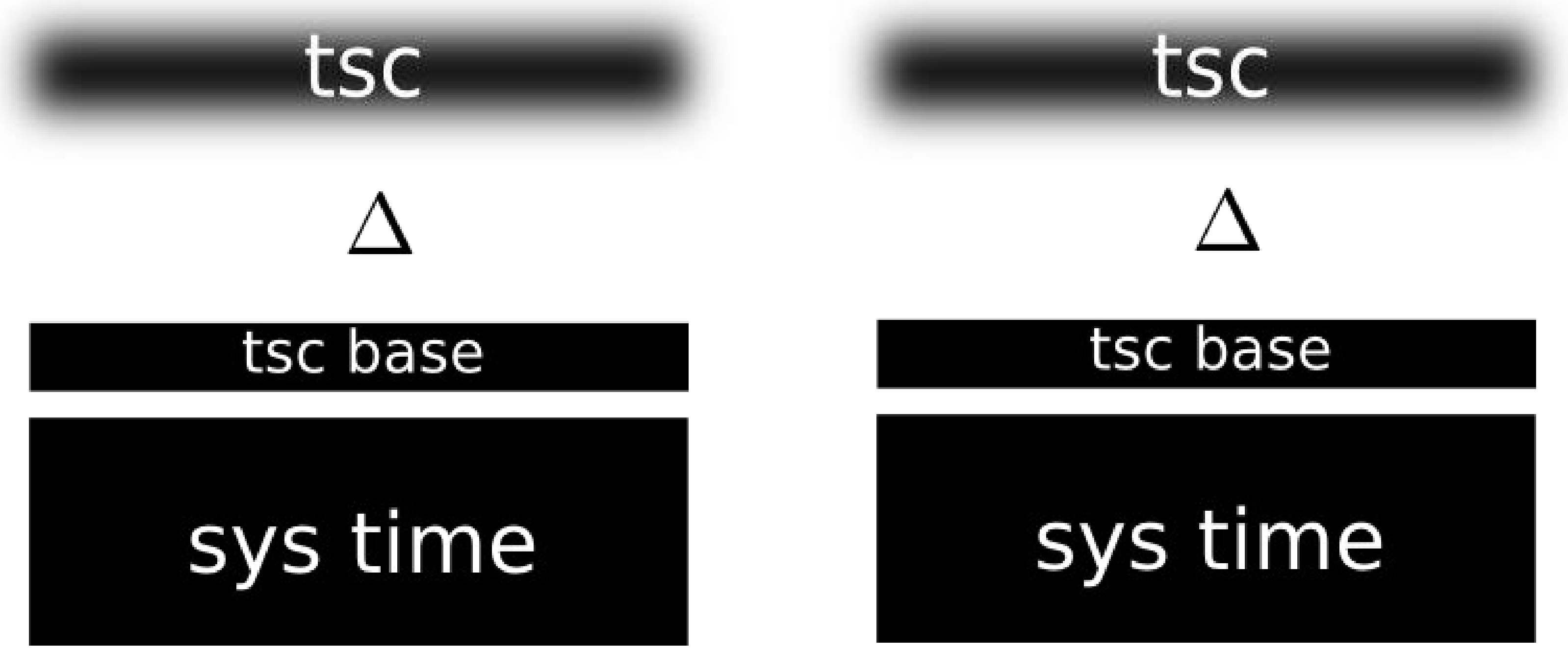
Δ

tsc base

sys time

CPU 1

Perfect synchronization still has issues



tsc

Δ

tsc base

sys time

tsc

Δ

tsc base

sys time

CPU 0

CPU 1

Summary

- Time is a hard problem
- Interrupt based timekeeping doesn't scale
- Perfect synchronization is rare
- Backwards jumps can arise in numerous ways

TSC / PIT / RTC clock

- Use re-injection for RTC (Windows)
- Use guest compensation for PIT (Older Linux)
- Use TSC stabilization techniques
- TSC frequency compensation
- TSC trapping for SMP (unstable)

KVM clock

- No interrupt re-injection
- Try for perfect synchronization where possible
- Use TSC stabilization techniques
- No frequency compensation
- No TSC trapping (userspace TSC imperfect)
- RDTSCP

Questions