Reducing CPU usage of a Toro Appliance

Matias E. Vara Larsen

matiasevara@gmail.com

FOSDEM 2018

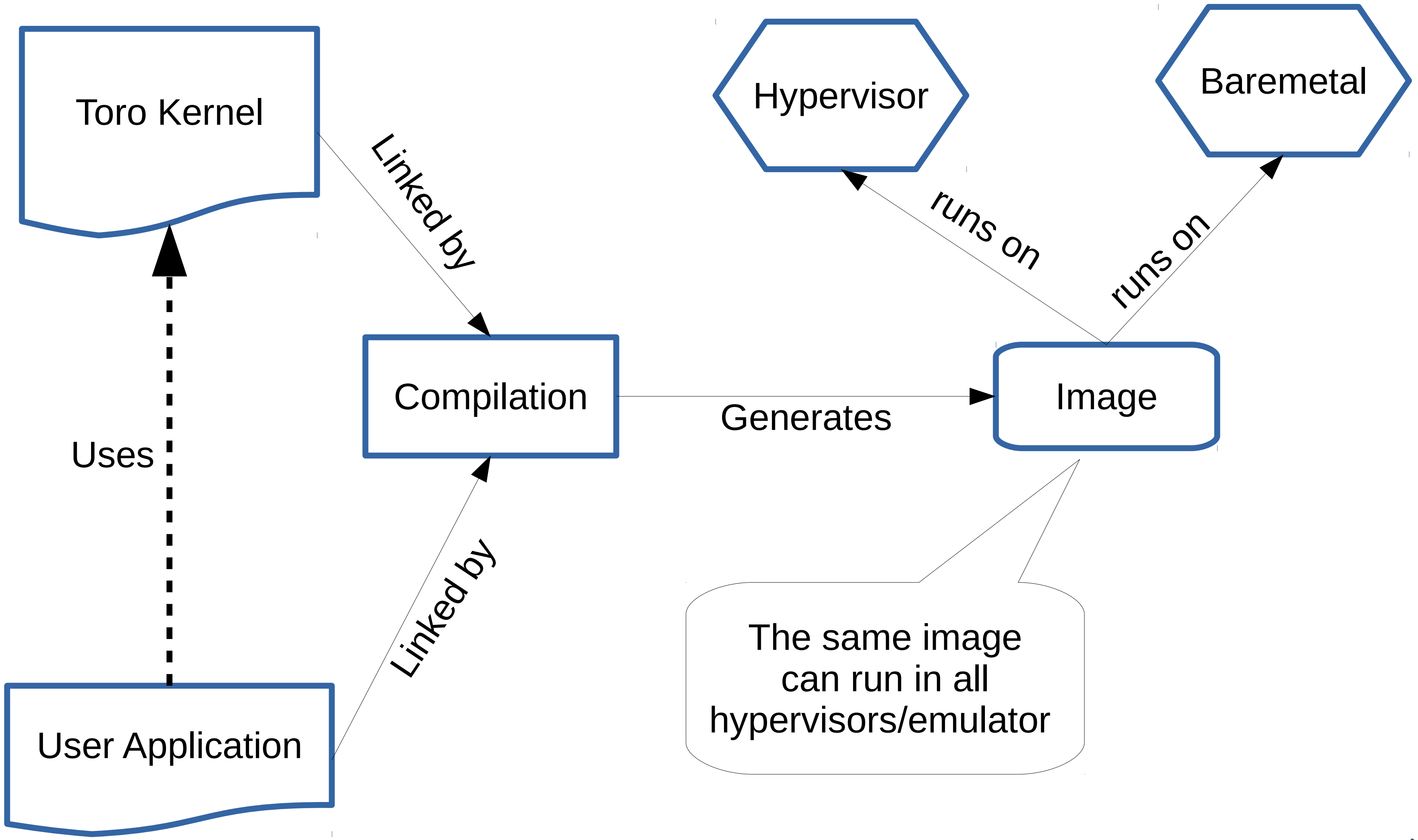
Who am I?

- Electronic Engineer from Universidad Nacional de La Plata, Argentina
- PhD in Computer Science, Universite Nice-Sophia Antipolis, Nice, France
- Citrix, Cambridge
- Silicon-Gears, Barcelona (Current job)
- During my free Time, I develop Toro Kernel

What is Toro?

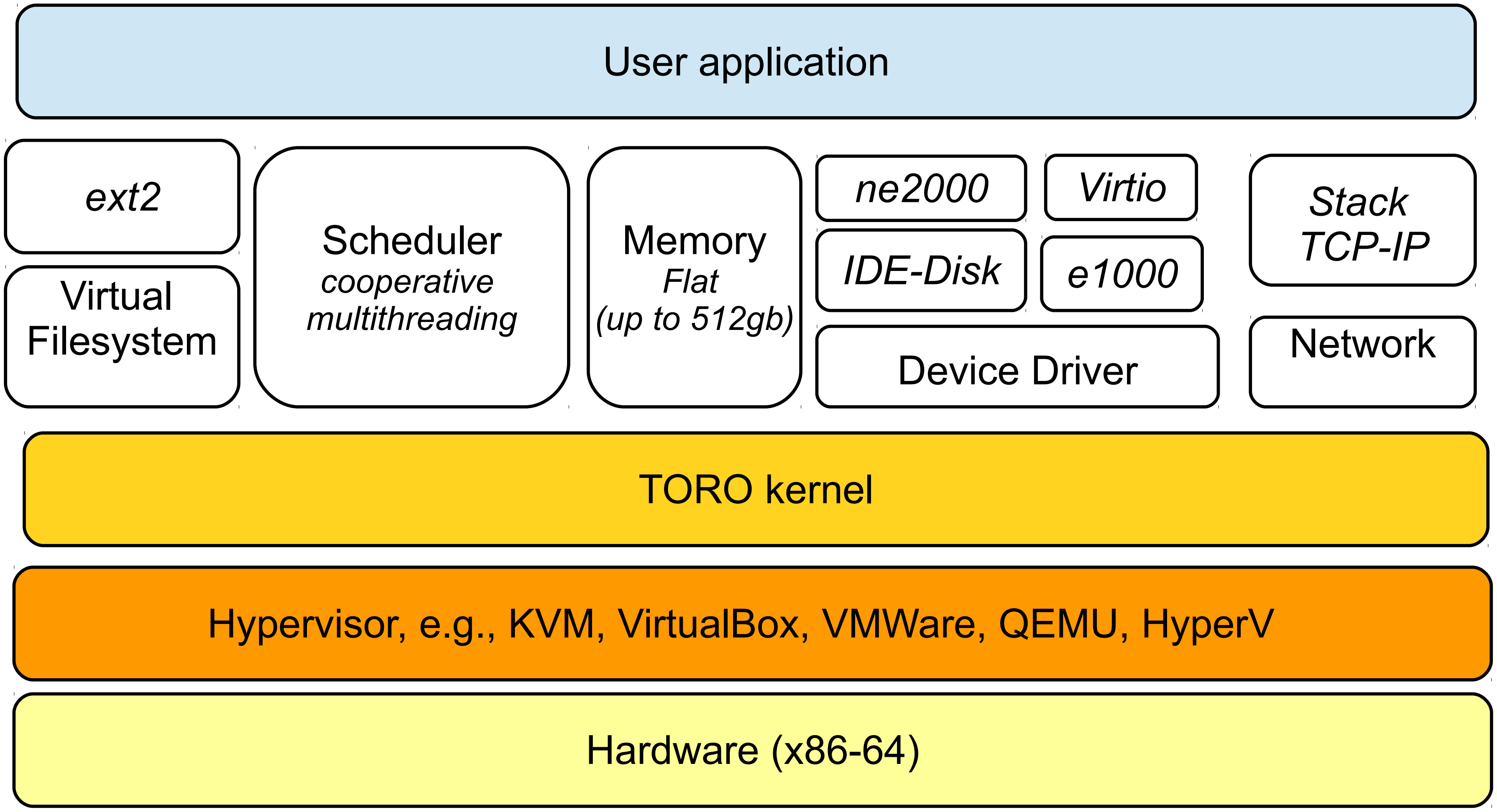
- It is a kernel based on Intel x86-64 architecture
- It is written in Freepascal, Yes, It is ...
- It provides a simple API for user application, i.e., **application-oriented**
- It is compiled within the user application thus resulting in a image, i.e., **library OS-like designing**
- It runs together with the user application at ring 0 and shares the memory space
- It can run either on baremetal or on top of an hypervisor/emulator, e.g., HyperV, KVM, QEMU, VirtualBox

What is Toro?



http://torokernel.io

Current state of Toro kernel



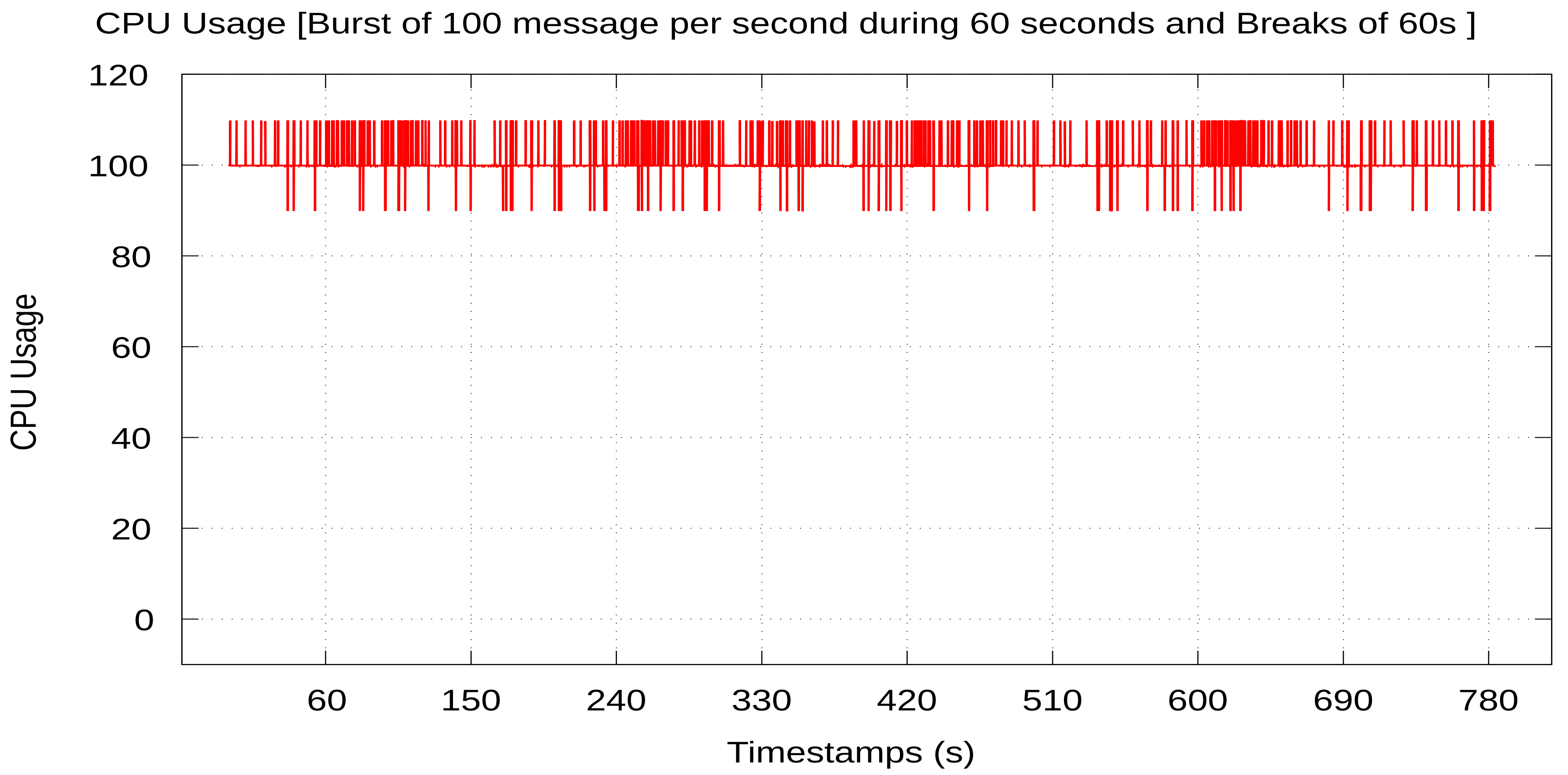
http://torokernel.io

What is this talk about?

- The CPU usage of a Toro guest is too high
- The CPU usage is high even when the guest is idle
- This is undesirable situation in production because CPU is shared resource
- “A high CPU usage value can lead to increased ready time and processor queuing of the virtual machines on the host” [1]

[1] VMware vSphere 5.1 Documentation Center, Solutions for Consistently High CPU Usage

What is this talk about?



- Top of a Qemu’s guest that runs a Http server in Toro

http://torokernel.io

Understanding the Problem

- The main issue is “idle loops” that consume a lot of CPU
  - i.e., a loop that keeps checking a condition
- I identified several cases in which idle loops were used:
  - Spin-locks
    - loop to get mutual exclusion to a shared resource in a multicore system
  - Scheduler
    - When there is no a thread in ready state, i.e., 2 cases
  - System Threads
    - Threads that keeps checking a condition when they are idle

Proposal

- Avoid the using of idle loops because is a bad programming habit

- Relax the CPU during idle loops [1,2,3]
  - 1) Spin-locks
  - 2) Scheduler
  - 3) Thread polling a variable

[1]“Benefiting Power and Performance Sleep Loops”
[2]”Idle Thread Talk”, G. Somlo
[3] Intel Manual

Partial Solution

- 1) Spin-locks:
  - Use “pause” instruction:
    - *“Essentially, the pause instruction delays the next instruction's execution for a finite period of time. By delaying the execution of the next instruction, the processor is not under demand, and parts of the pipeline are no longer being used, which in turn reduces the power consumed by the processor.*”

Partial Solution

- 2) Scheduler:
  - Use “halt” instruction when Scheduler is idle:
    - *“Stops instruction execution and places the processor in a HALT state. An enabled interrupt (including NMI and SMI), a debug exception, the BINIT# signal, the INIT# signal, or the RESET# signal will resume execution. If an interrupt (including NMI) is used to resume execution after a HLT instruction, the saved instruction pointer (CS:EIP) points to the instruction following the HLT instruction.” (It is a privileged instrucction).”*

Partial Solution

- 3) A System Thread that polls a variable:
  - It is a hard case because scheduler has to figured out when a thread is doing idle work
  - Proposal: To provide an API to enable the thread to tell the scheduler when it is doing idle work

Partial Solution

- **SysThreadSwitch(IdleFlag: Boolean)**
  - tells the scheduler when it can schedule a new thread
  - When **IdleFlag=true**, the scheduler knows that the thread is doing idle work, e.g., polling the variable.
  - The scheduler does the following steps:
    - 1. To check how much time the thread has been idle
    - 2. if it is more than some constant the thread’s state becomes idle
    - 3. To check if all threads in the system are idle and there is no thread in ready state, in this case the scheduler halts the core.
- **SysThreadActive()**
  - tells the scheduler that the thread has work to do,
  - the scheduler stops to count the idle time and the thread state become ts_ready.

Example: ProcessNetworkPackets()

```
...
while True do
begin
  Packet := SysNetworkRead;
  if Packet = nil then
  begin
    // thread tells scheduler that is idle
    SysThreadSwitch(True);
    Continue;
  end else
  begin
    // thread tells scheduler is not idle
    SysThreadActive;
  end;
  EthPacket := Packet.Data;
  ...
```

Executes when is idle

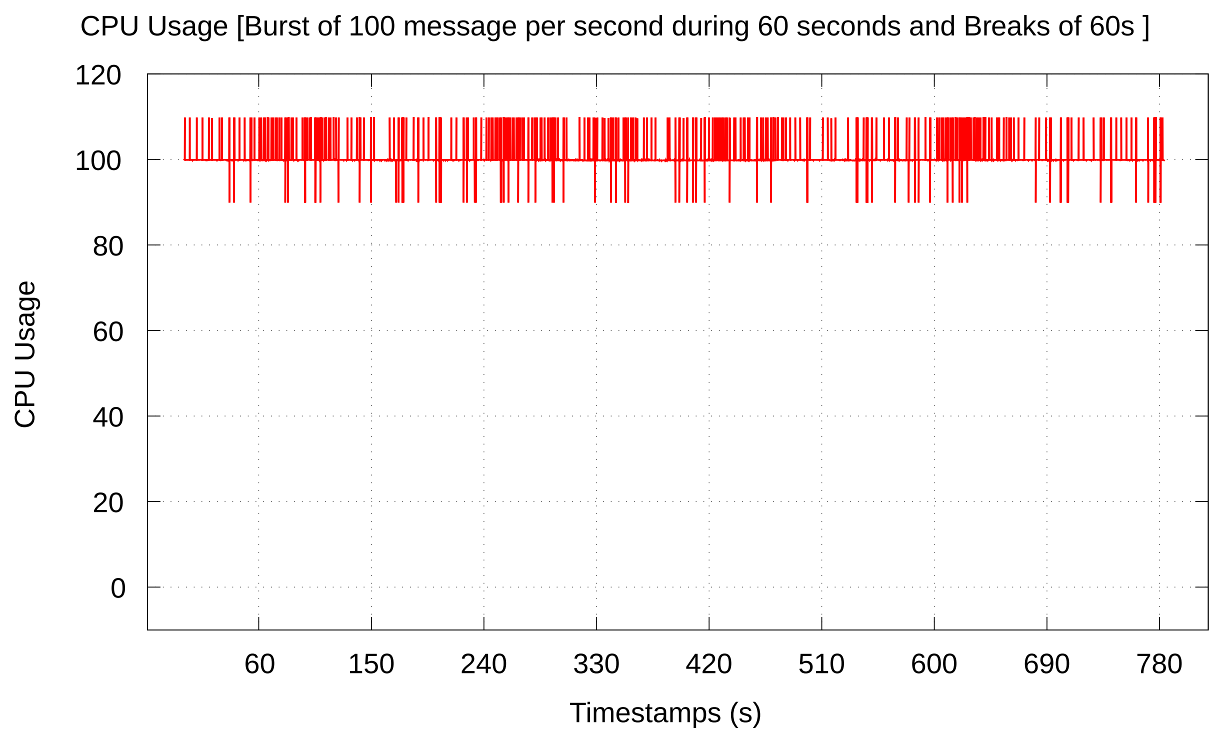
Executes when is not idle

Experimentation

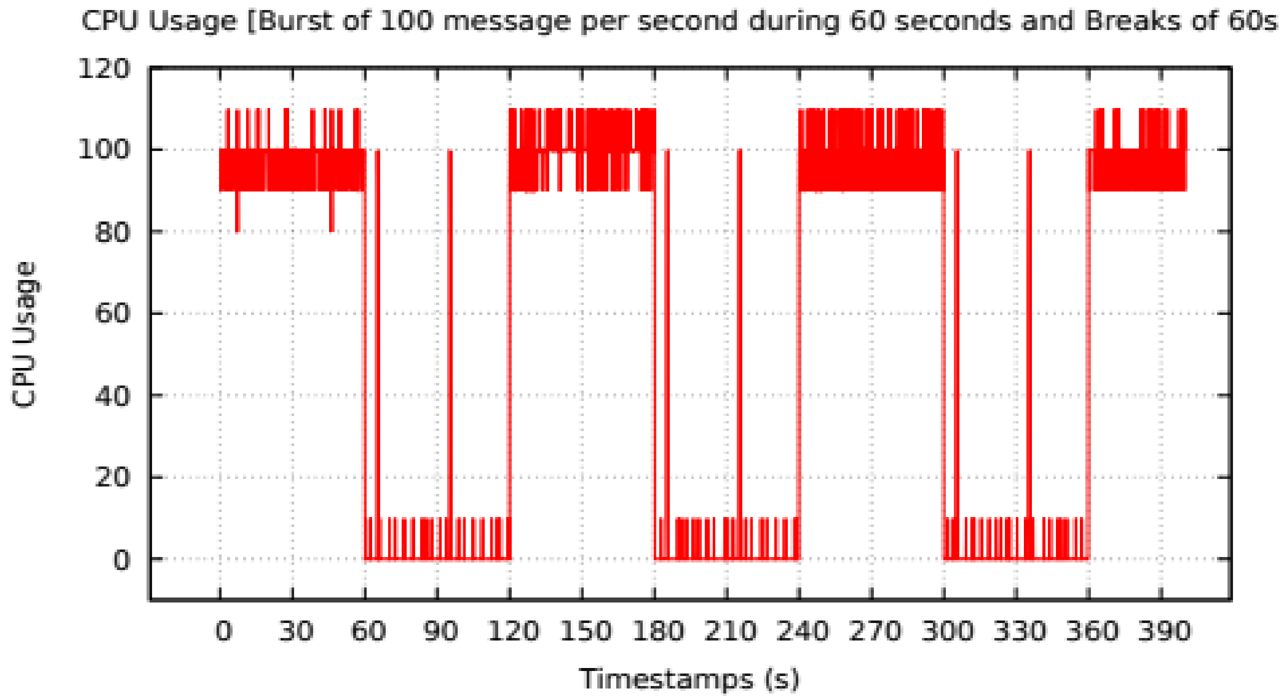
- Run a Web server in Toro as Qemu's guest
  - 2 cores but only one used
  - 512 MB, ~256 MB per core
- Generate N http requests and then stop, repeat it every X time
- Measure the CPU usage of the Qemu process in the host with "Top"
- Experiment and Compare the following cases:

1. Toro with and without the improvements
2. Toro and Apache running as Qemu's guest

- Toro guest without improvement[1]



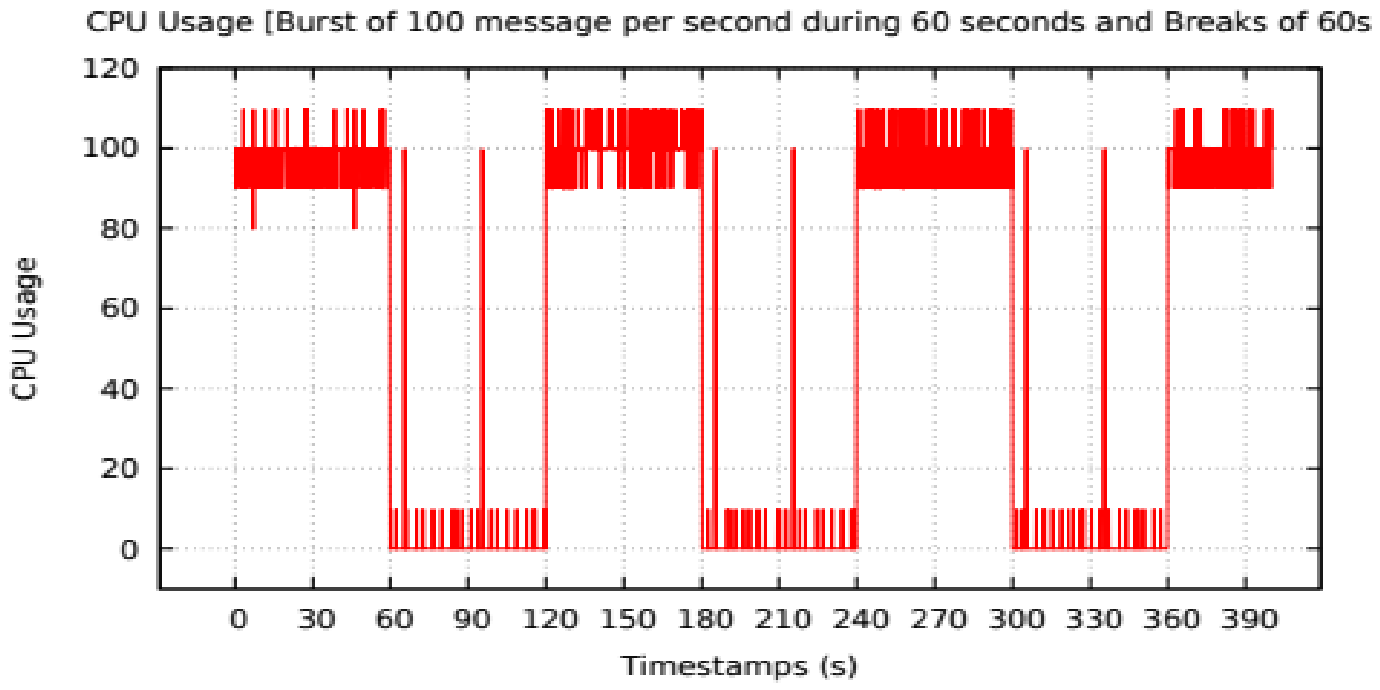
- Toro guest with improvement[1]



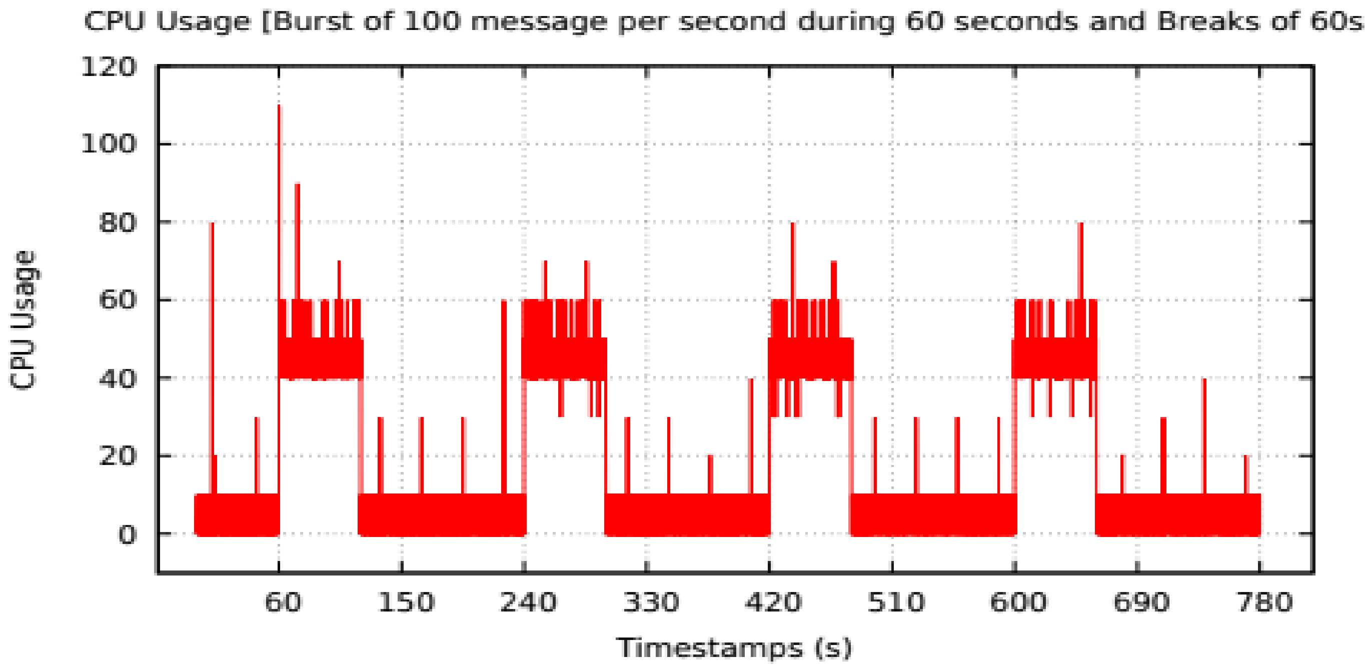
[1]Thanks to Cesar Bernardini!
mesarpe@gmail.com

http://torokernel.io

- Toro guest
  2 cores, 512MB[1]



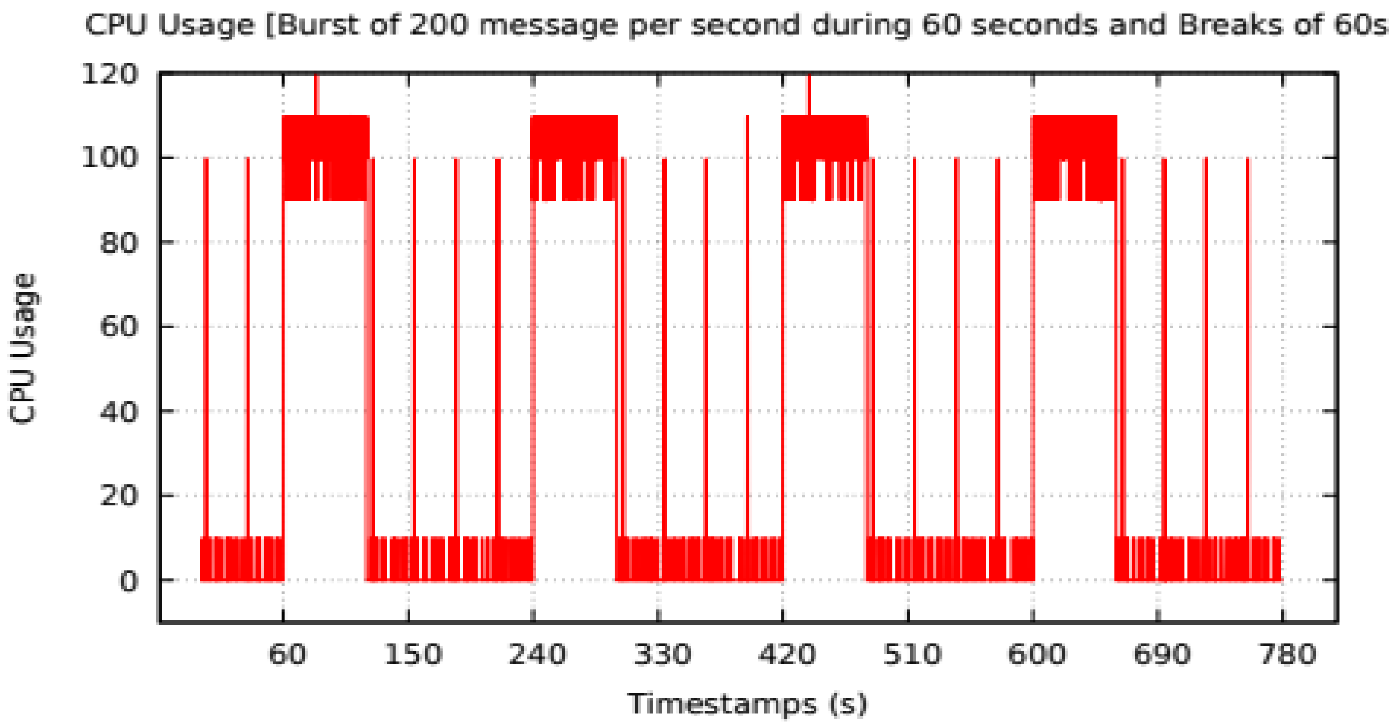
- Linux Ubuntu guest
  2 cores, 512MB[1]



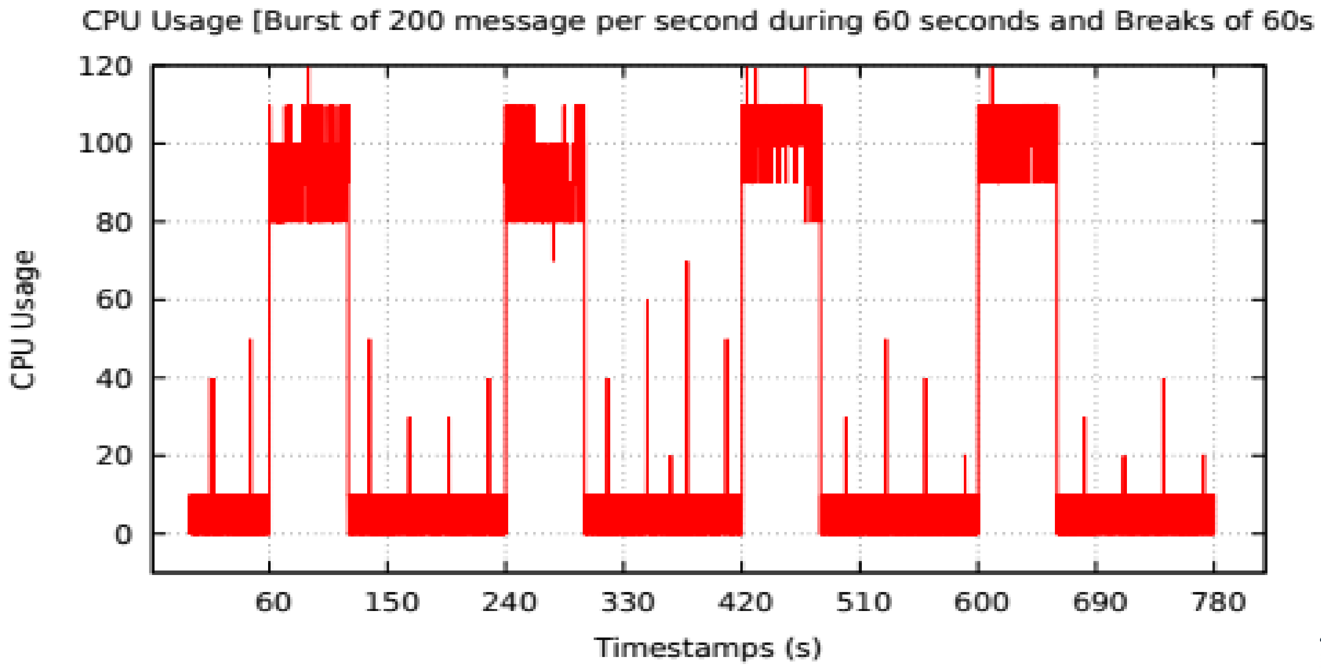
[1]Thanks to Cesar Bernardini!
mesarpe@gmail.com

http://torokernel.io

- Toro guest
  2 cores, 512MB[1]



- Linux Ubuntu guest
  2 cores, 512MB[1]



[1]Thanks to Cesar Bernardini!
mesarpe@gmail.com

http://torokernel.io

Take-away lessons

- CPU usage of VMs becomes very important in production because CPU is a shared resource

- The presented solution has reduced the CPU usage in a half, however a complete power management solution must also scale the CPU, i.e., processor in P-State

- Some solutions may depend on the hypervisor and it ability to emulate some instructions, e.g., mwat/mcontrol

Questions?

http://torokernel.io

Thanks!

www.torokernel.io

matiasevara@gmail.com