VM kernel tracing

with ftrace, trace-cmd and Kernel Shark

Tzvetomir Stoyanov

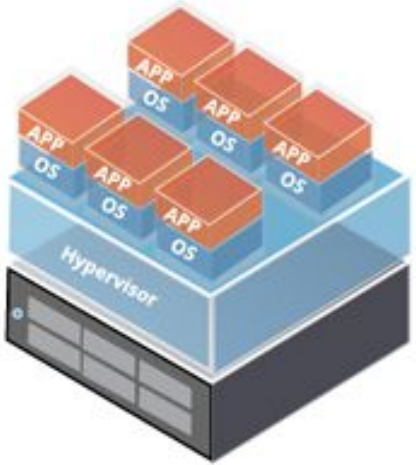
VMware Open Source Technology Center

vmware

Agenda

- ➔ The problem
- ➔ The general idea
- ➔ The challenges
- ➔ The solution

The problem



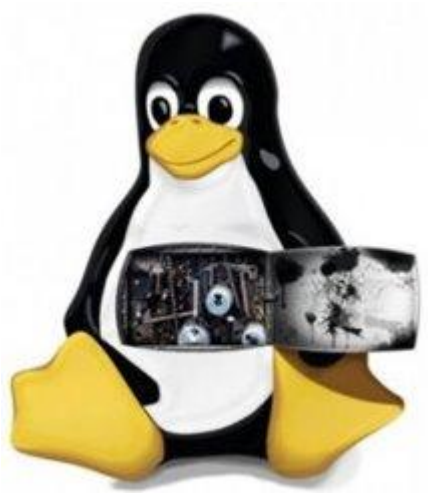
+ scaling
+ redundancy
+ hardware utilization

- nightmare for developers and administrators

The problem

According to the Linux foundation Linux OS runs **90%** of the public cloud workload

The problem



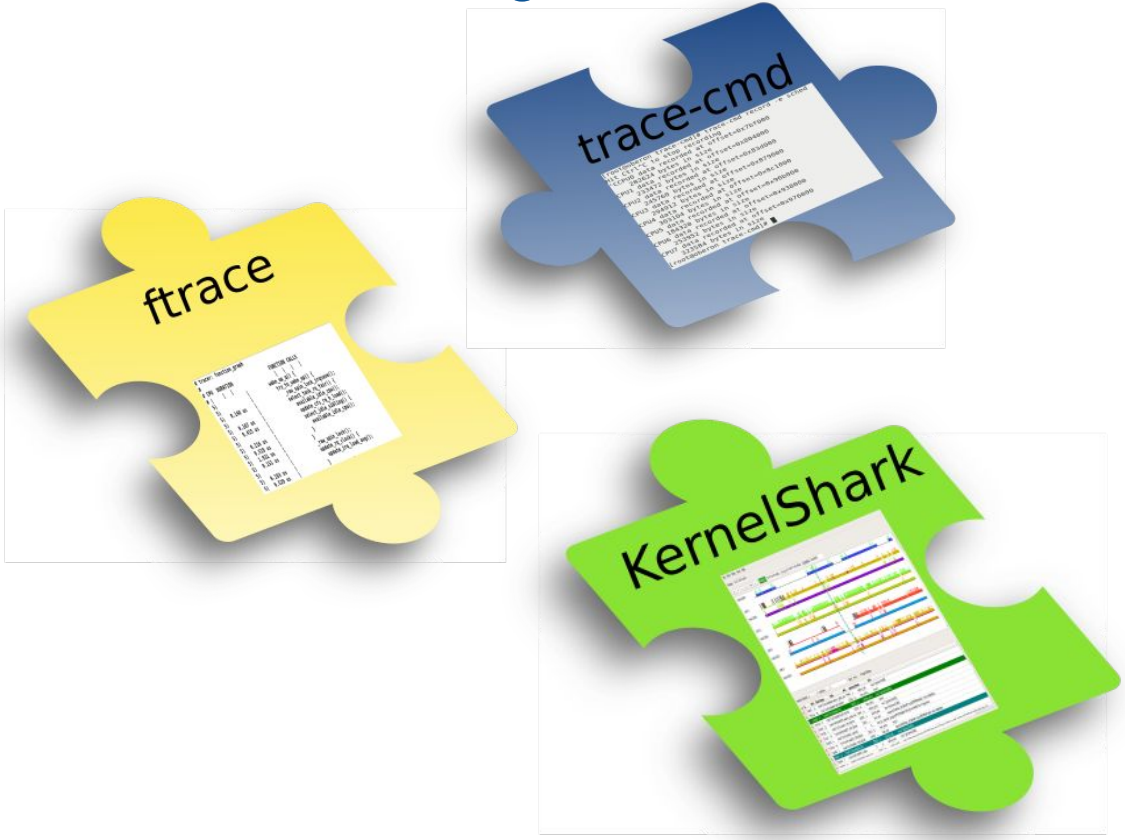
In majority of use cases Linux is used in both sides - as host and guest OS

ftrace

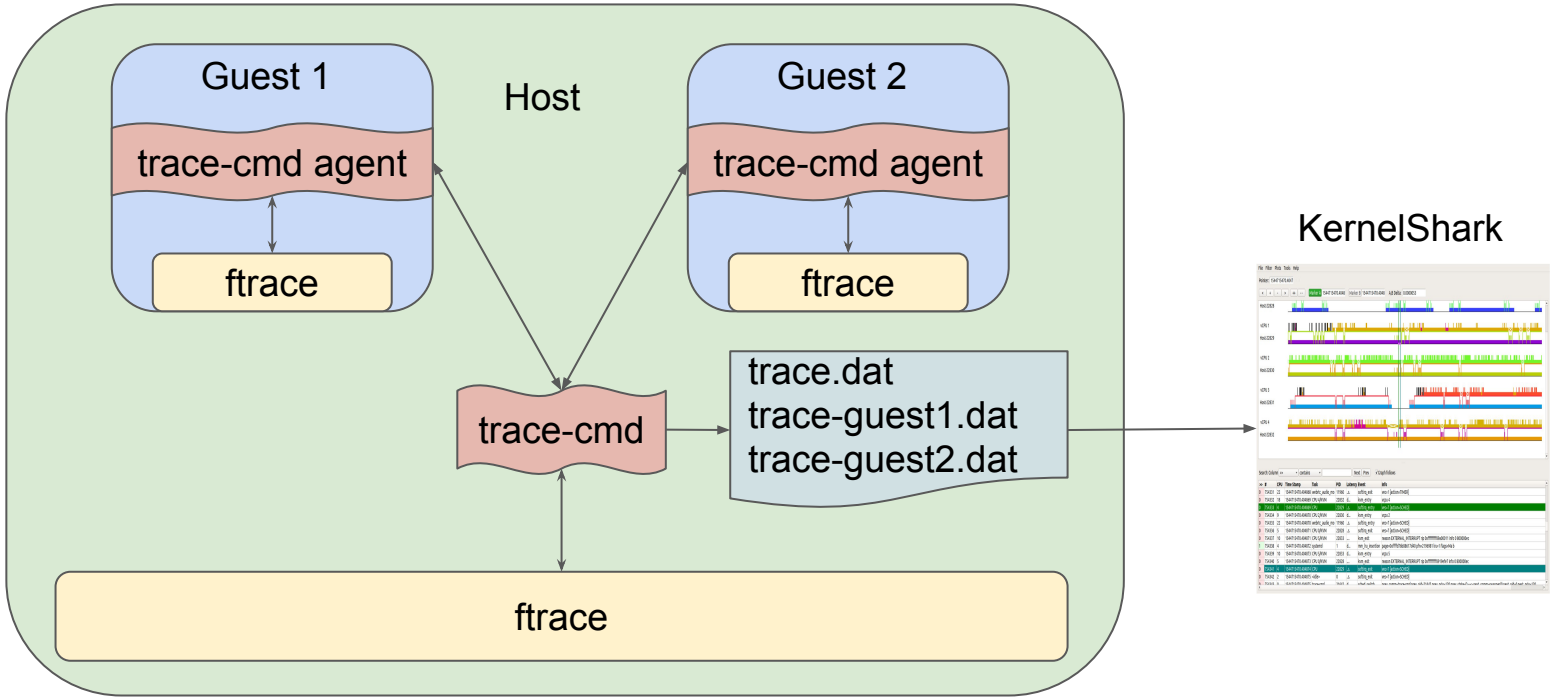
```
# tracer: function_graph
#
# CPU  DURATION                  FUNCTION CALLS
# |     |   |                     |   |   |   |
 5)               |          wake_up_q() {
 5)               |            try_to_wake_up() {
 5)   0.148 us    |              _raw_spin_lock_irqsave();
 5)               |              select_task_rq_fair() {
 5)   0.167 us    |                available_idle_cpu();
 5)   0.415 us    |                update_cfs_rq_h_load();
 5)               |                select_idle_sibling() {
 5)   0.216 us    |                  available_idle_cpu();
 5)   0.519 us    |                }
 5)   1.931 us    |              }
 5)   0.151 us    |              _raw_spin_lock();
 5)               |              update_rq_clock() {
 5)   0.193 us    |                update_irq_load_avg();
 5)   0.529 us    |              }
```

- The official tracer of the Linux kernel
- Developed by Steven Rostded more than 10 years ago
- Part of the kernel, compiled by default in most popular Linux distros.
- Allows you to look inside every corner of a live running kernel.

The general idea



The general idea



Challenges

- ➔ Fast transfer of huge tracing data between guest and host
- ➔ Time stamps synchronisation
- ➔ User friendly visualisation of huge host and guest trace data

VM Data Transfer: TCP/IP sockets

- **Pros**
  - Hypervisor independant
  - Allows bi-directional communication
  - Can be used for nested virtualisation tracing
- **Cons**
  - Slow, very slow
  - Pass through the whole kernel's network stack
  - Communication can happen between any host or VM which is undesirable

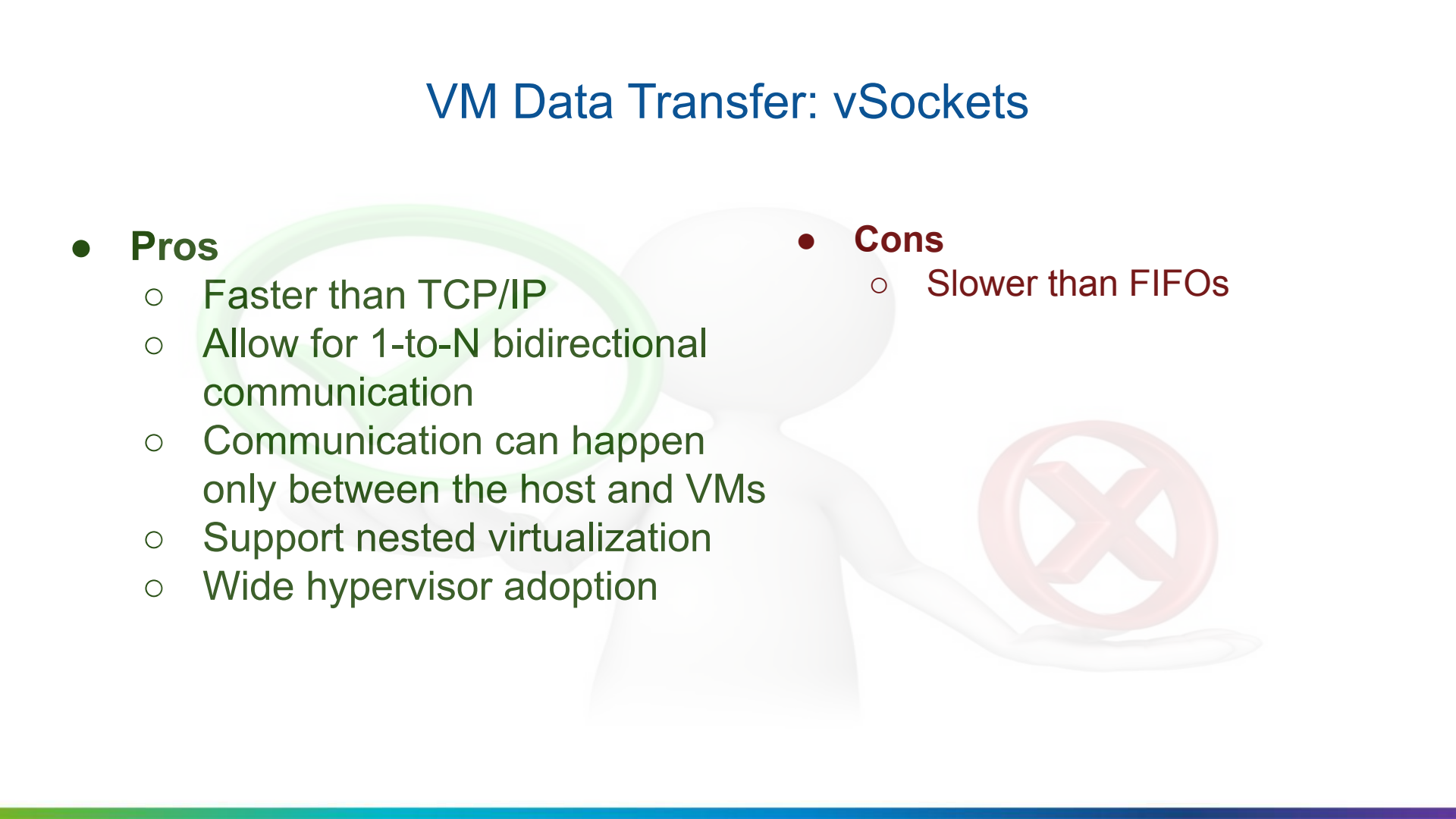
VM Data Transfer: FIFOs

- **Pros**
  - Fastest
  - Has no security concerns, as the TCP / IP solution
- **Cons**
  - works only for KVM
  - 1to1 unidirectional communication only
  - a FIFO for each direction is needed
  - Cannot be used for nested virtualization use case

VM Data Transfer: vSockets

- **Pros**
  - Faster than TCP/IP
  - Allow for 1-to-N bidirectional communication
  - Communication can happen only between the host and VMs
  - Support nested virtualization
  - Wide hypervisor adoption

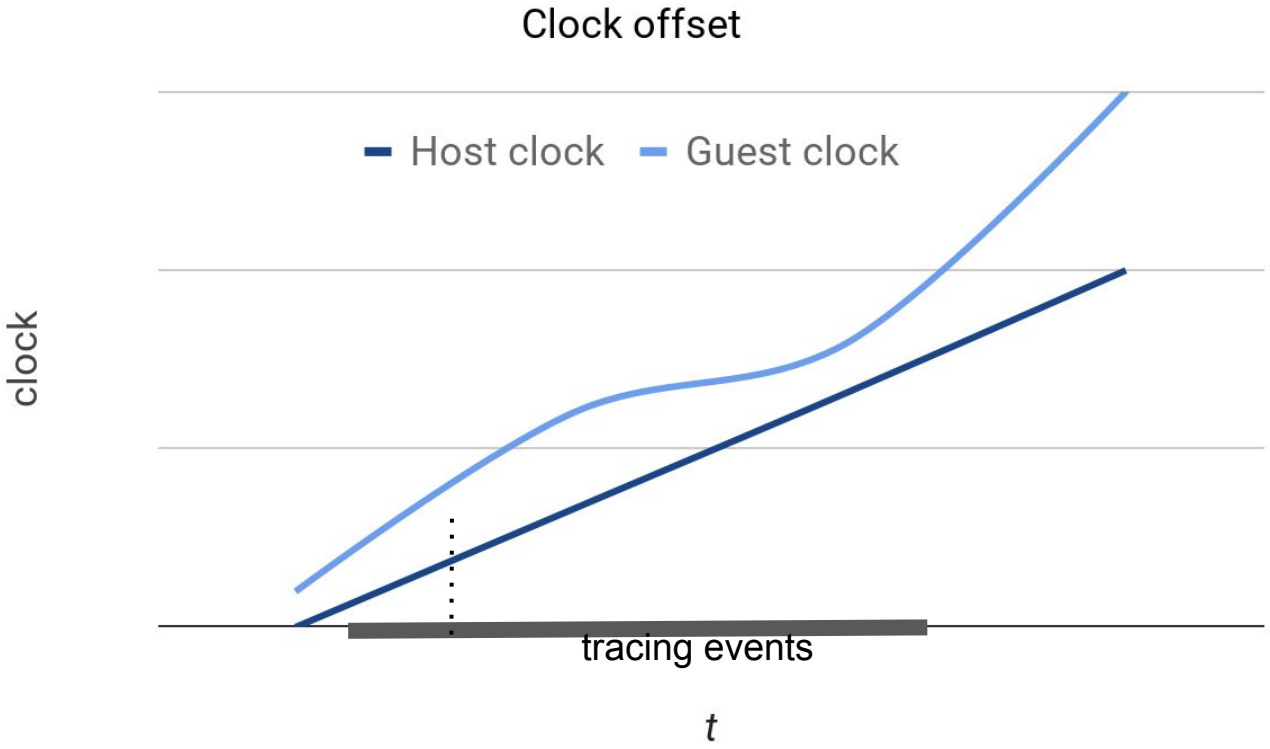
- **Cons**
  - Slower than FIFOs



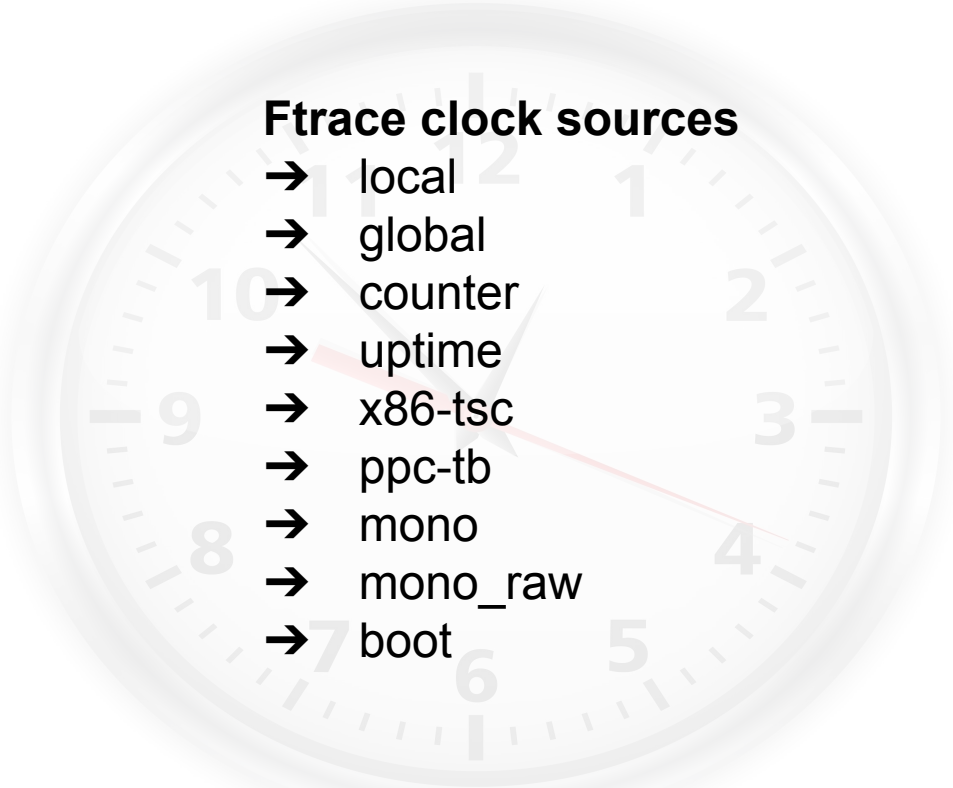
KVM Channels Throughput

Channel	Throughput
FIFOs	1000 MB/s
vsockets	900 MB/s
TCP/IP sockets	275 MB/s

Synchronisation of time stamps



Synchronisation of time stamps

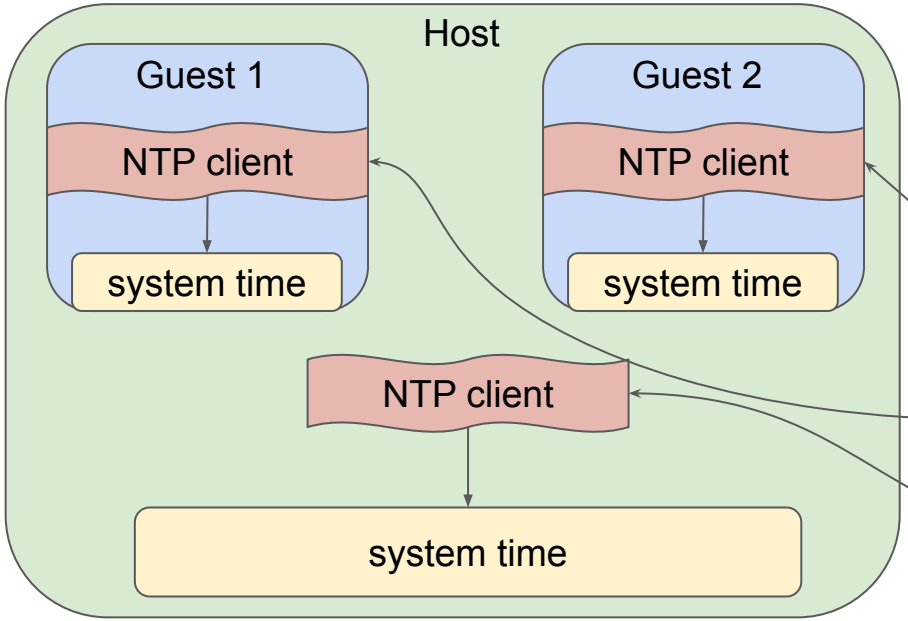


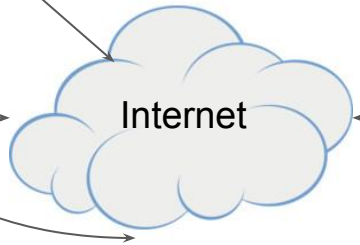
Ftrace clock sources

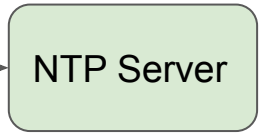
- ➔ local
- ➔ global
- ➔ counter
- ➔ uptime
- ➔ x86-tsc
- ➔ ppc-tb
- ➔ mono
- ➔ mono_raw
- ➔ boot

Synchronisation of time stamps

NTP approach



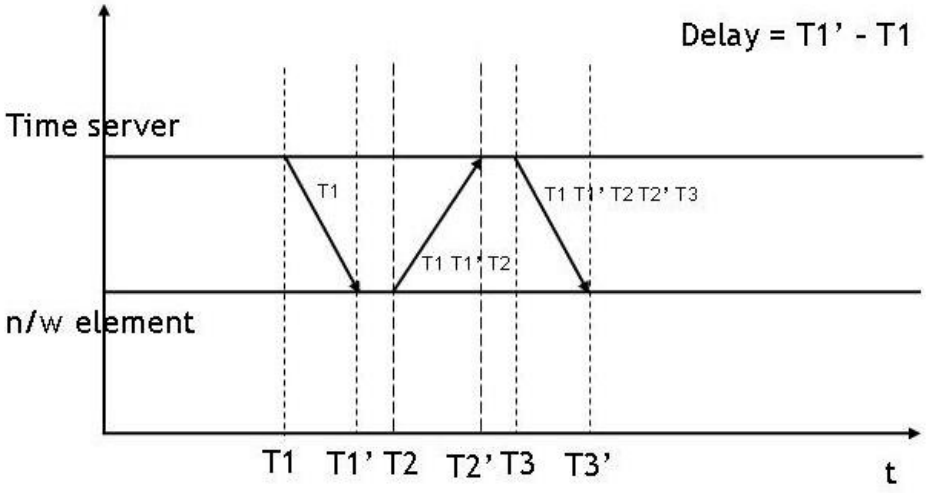




- tens of milliseconds accuracy
- clocks must be synchronized in advanced, before running the trace

Synchronisation of time stamps

PTP approach

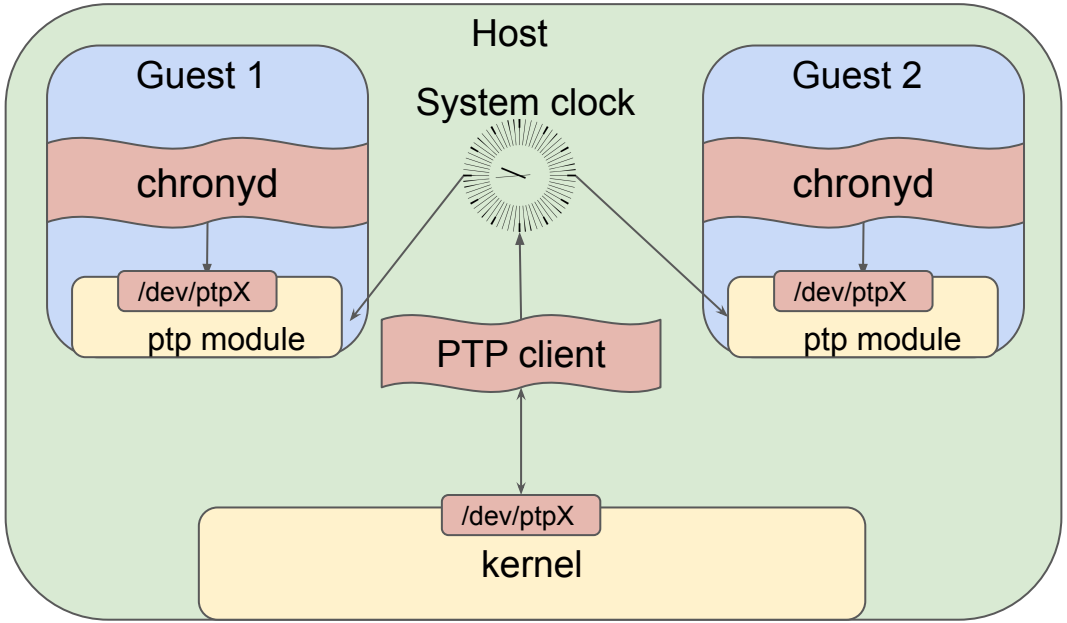


Clock offset

(T1' - T1 - T2' + T2) / 2

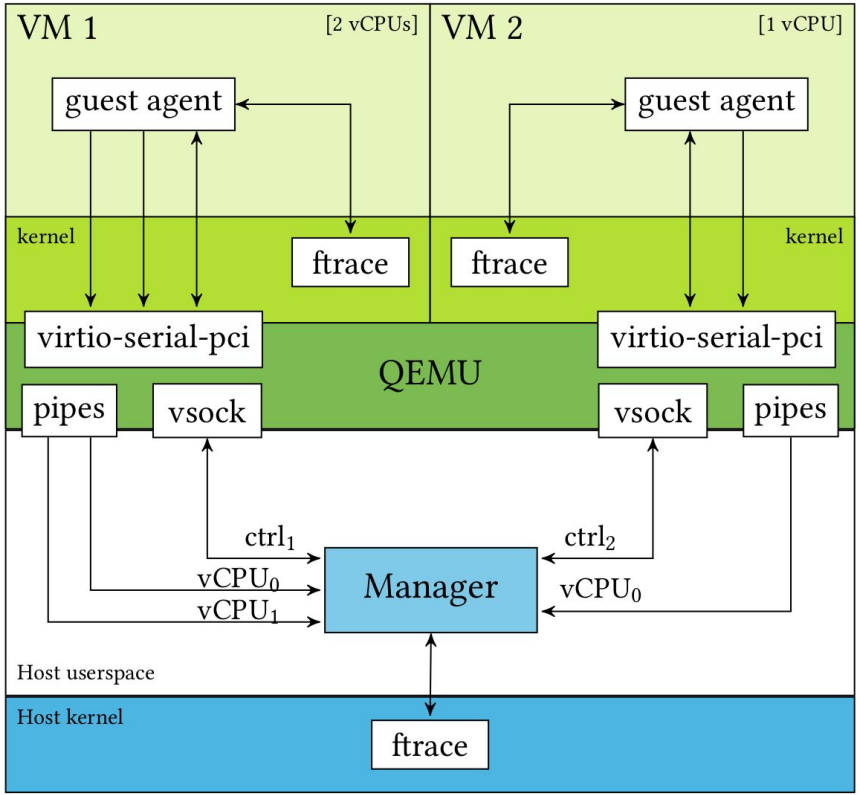
Synchronisation of time stamps

PTP in VM environment



- sub-microsecond accuracy
- clocks must be synchronized in advanced, before running the trace

VM Tracing Overview



Trace data transfer

- ➔ FIFOs, in case of KVM hypervisor
- ➔ vsockets
- ➔ splice()

Timestamps synchronisation

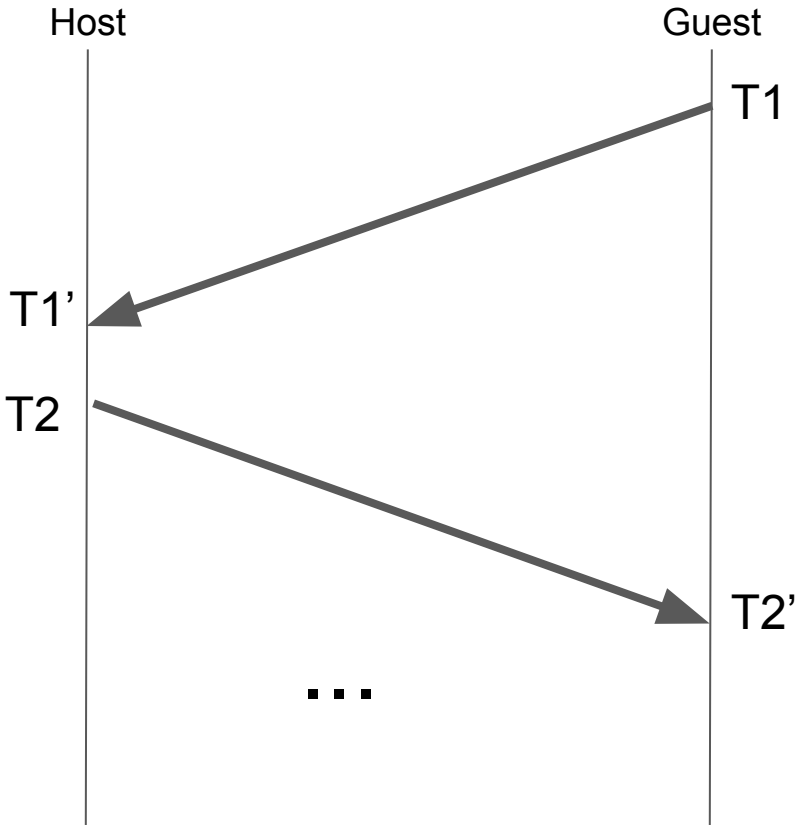
using system time

trace-cmd record --date …

- converts trace timestamps to system time
- all clocks must be synchronized in advanced
- accuracy depends on clock synchronization

Timestamps synchronisation

using PTP-like algorithm



Clock offset

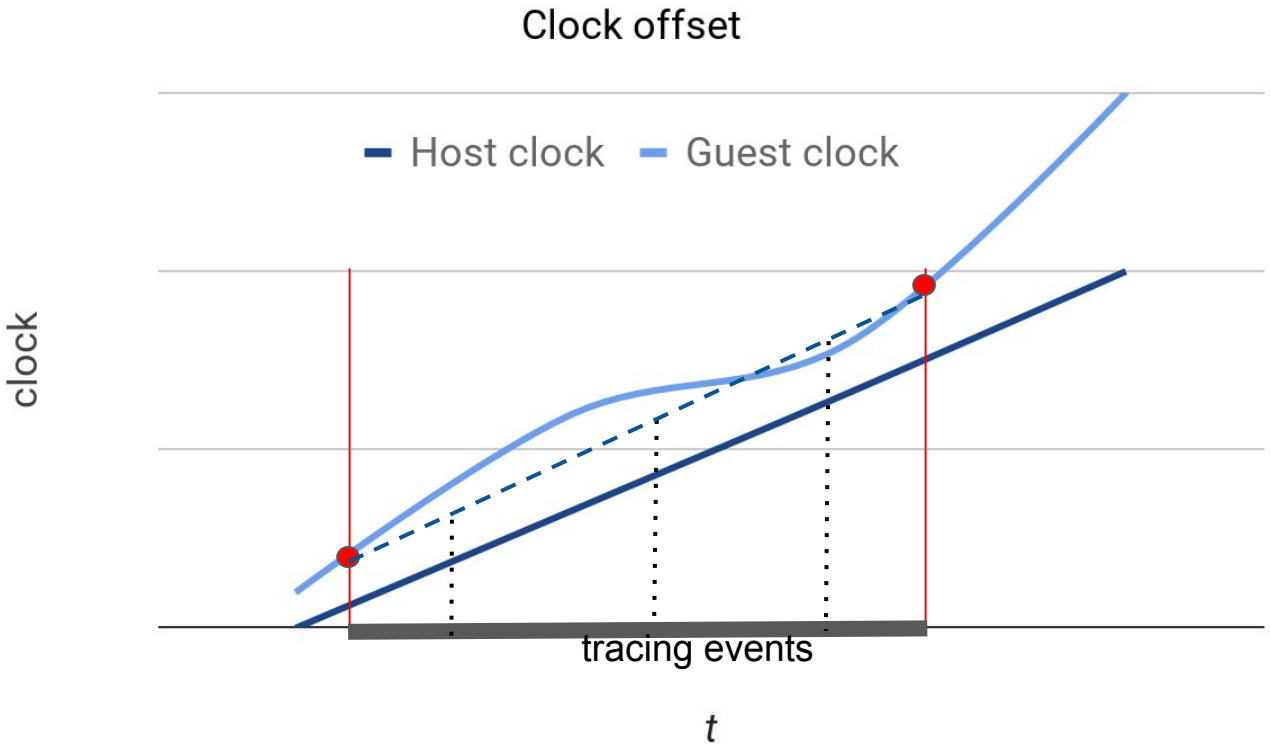
(T1' - T1 - T2' + T2) / 2

- round trip time is not symmetric
- no hardware timestamping

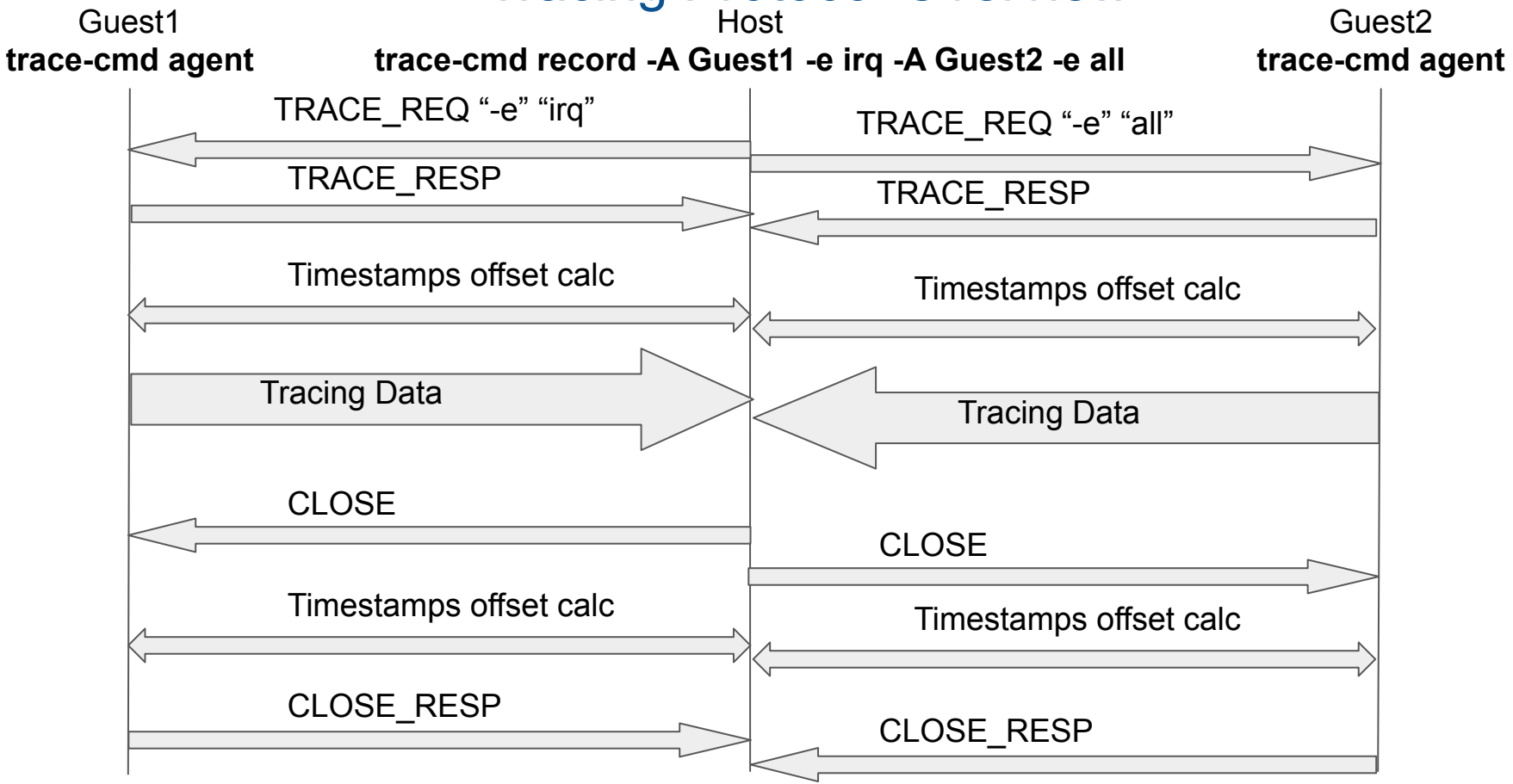
- Up to few hundred packets are exchanged in one clock offset measurement
- ftrace is used to get the packet times

Timestamps synchronisation

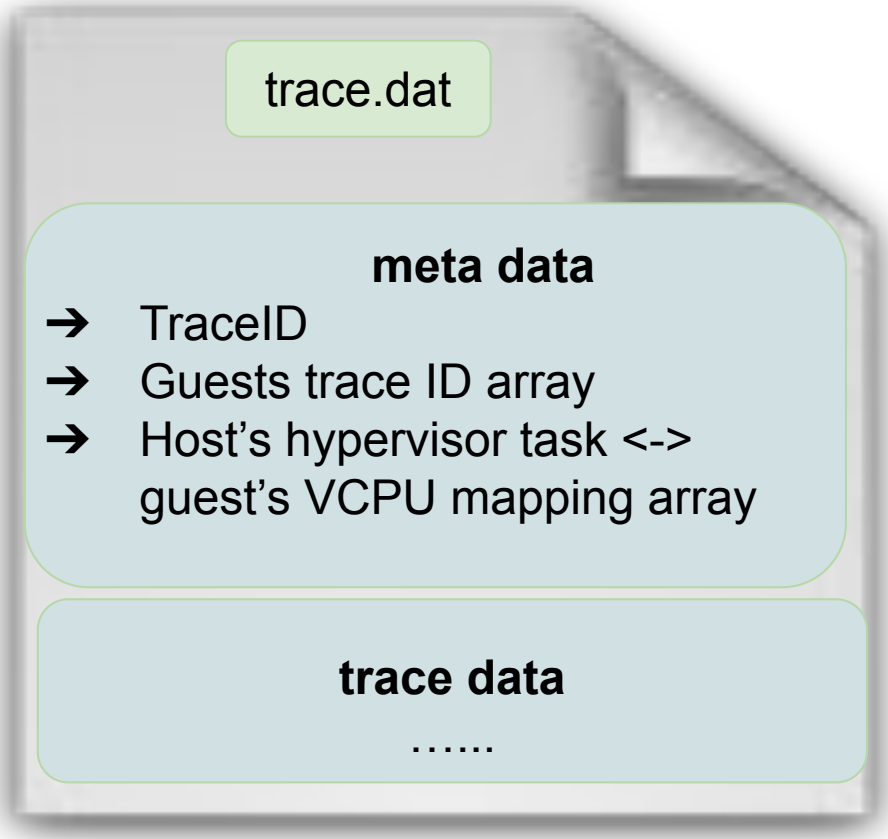
using PTP-like algorithm

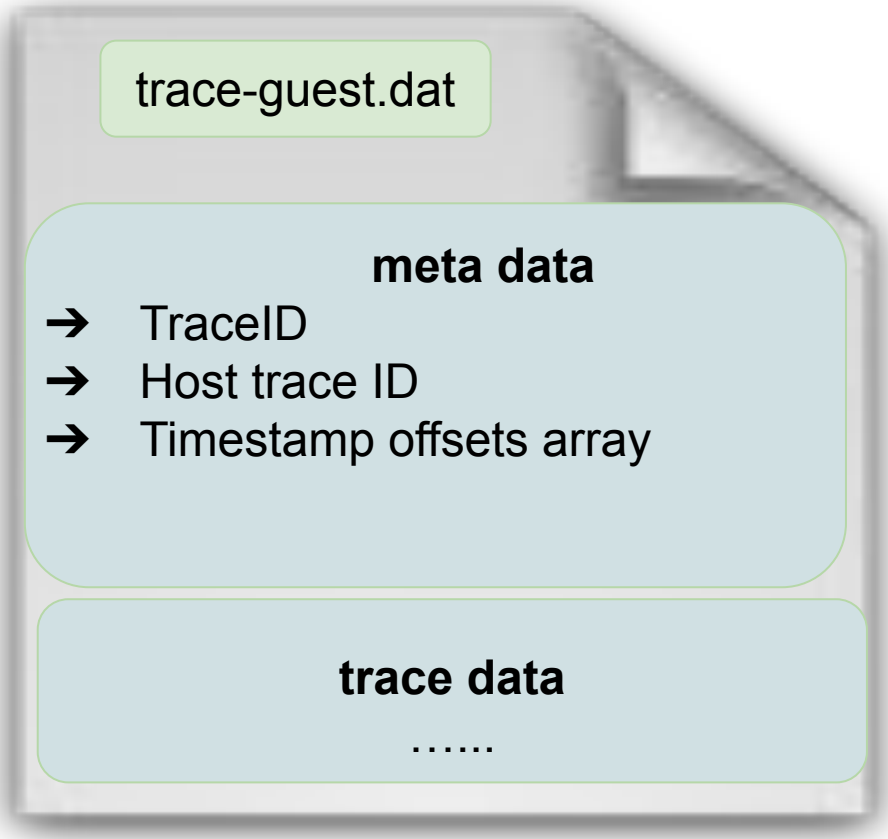


Tracing Protocol Overview

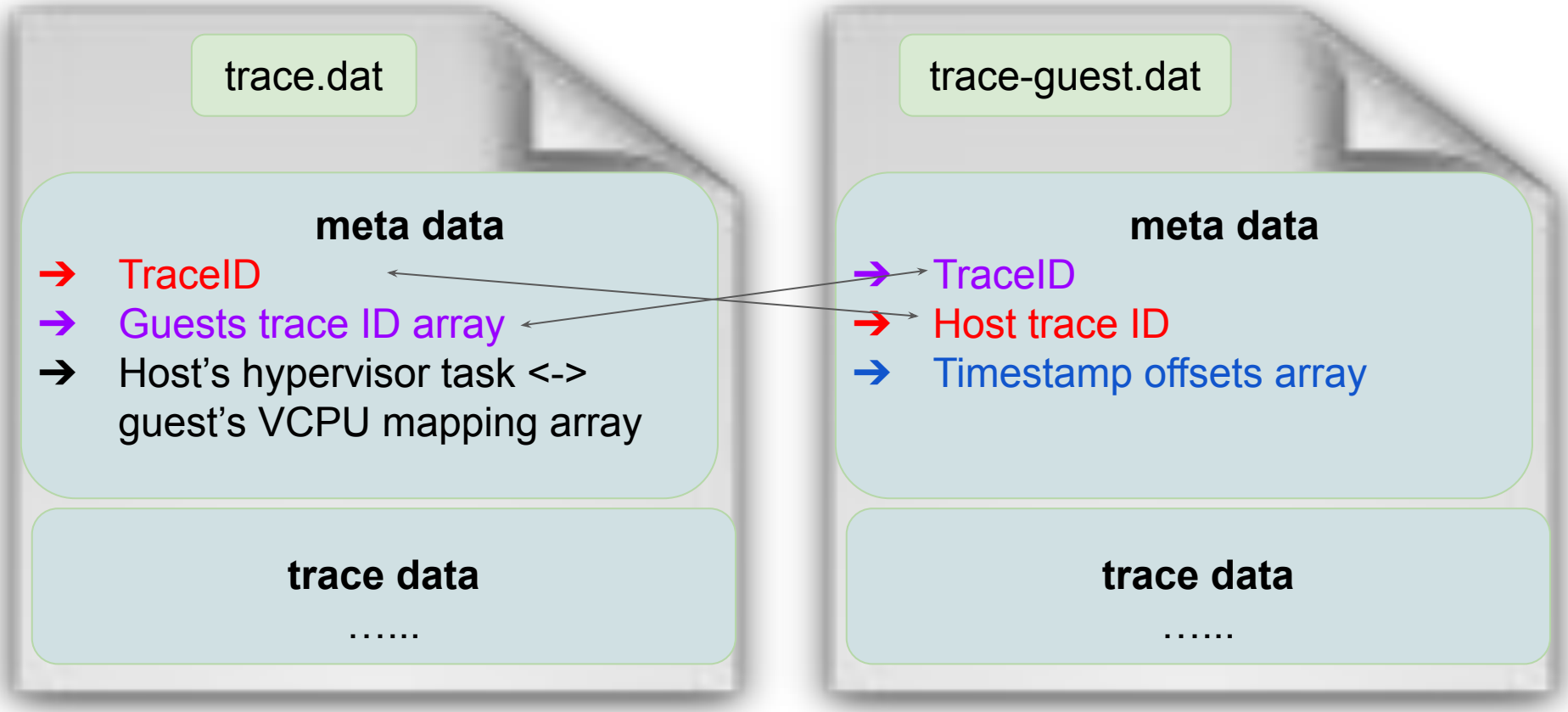


trace files

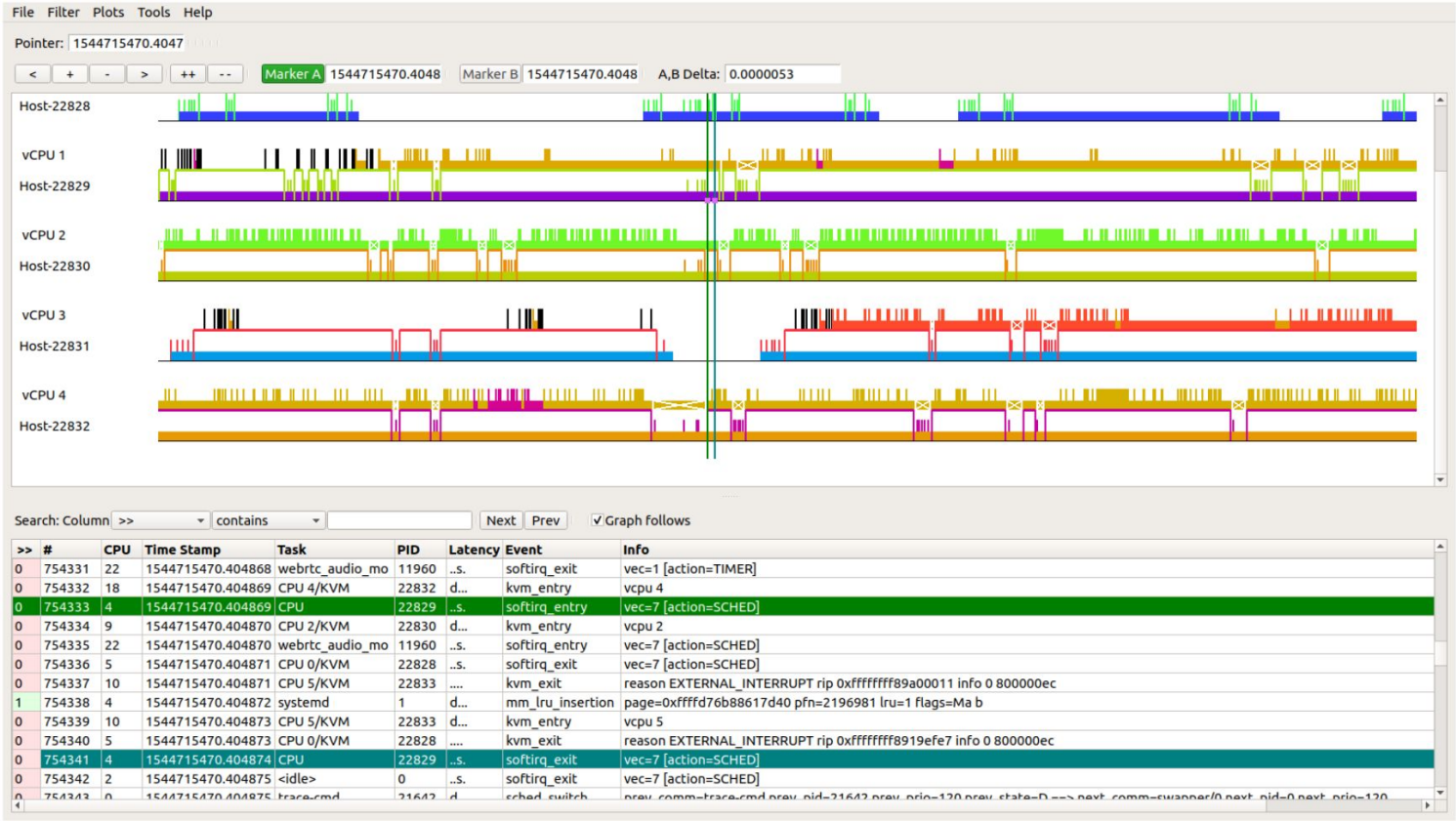




trace files



KernelShark



>>	#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	754331	22	1544715470.404868	webrtc_audio_mo	11960	..s.	softirq_exit	vec=1 [action=TIMER]
0	754332	18	1544715470.404869	CPU 4/KVM	22832	d...	kvm_entry	vcpu 4
0	754333	4	1544715470.404869	CPU	22829	..s.	softirq_entry	vec=7 [action=SCHED]
0	754334	9	1544715470.404870	CPU 2/KVM	22830	d...	kvm_entry	vcpu 2
0	754335	22	1544715470.404870	webrtc_audio_mo	11960	..s.	softirq_entry	vec=7 [action=SCHED]
0	754336	5	1544715470.404871	CPU 0/KVM	22828	..s.	softirq_exit	vec=7 [action=SCHED]
0	754337	10	1544715470.404871	CPU 5/KVM	22833	....	kvm_exit	reason EXTERNAL_INTERRUPT rip 0xffffffff89a00011 info 0 800000ec
1	754338	4	1544715470.404872	systemd	1	d...	mm_lru_insertion	page=0xffffd76b88617d40 pfn=2196981 lru=1 flags=Ma b
0	754339	10	1544715470.404873	CPU 5/KVM	22833	d...	kvm_entry	vcpu 5
0	754340	5	1544715470.404873	CPU 0/KVM	22828	....	kvm_exit	reason EXTERNAL_INTERRUPT rip 0xffffffff8919efe7 info 0 800000ec
0	754341	4	1544715470.404874	CPU	22829	..s.	softirq_exit	vec=7 [action=SCHED]
0	754342	2	1544715470.404875	<idle>	0	..s.	softirq_exit	vec=7 [action=SCHED]
0	754343	0	1544715470.404875	trace-cmd	21642	d...	sched_switch	prev_comm=trace-cmd prev_pid=21642 prev_prio=120 prev_state=D ==> next_comm=swapper/0 next_pid=0 next_prio=120

Final words

- Ftrace, trace-cmd and KernelShark are originally developed by Steven Rostedt
- VM tracing implementation was started by Yoshihiro Yunomae
- Redesigned by VMware's Open Source team last year
- Newly released trace-cmd 2.9 has support of VM tracing
- Next major release of KernelShark will inlude support for VM tracing visualisation

Resources

ftrace (part of the Linux kernel)
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git

trace-cmd
https://trace-cmd.org/

KernelShark
https://www.kernelshark.org

Bugzilla
https://bugzilla.kernel.org/buglist.cgi?component=Trace-cmd%2FKernelshark

Thank You !

tstoyanov@vmware.com

@VMWopensource
blogs.vmware.com/opensource