Virtio-trace

- Towards the flexible fast interconnection between guest and host for tracing

Tracing Summit 2012

Masami Hiramatsu <masami.hiramatsu.pt@hitachi.com>

Yoshihiro YUNOMAE <yoshihiro.yunomae.ez@hitachi.com>

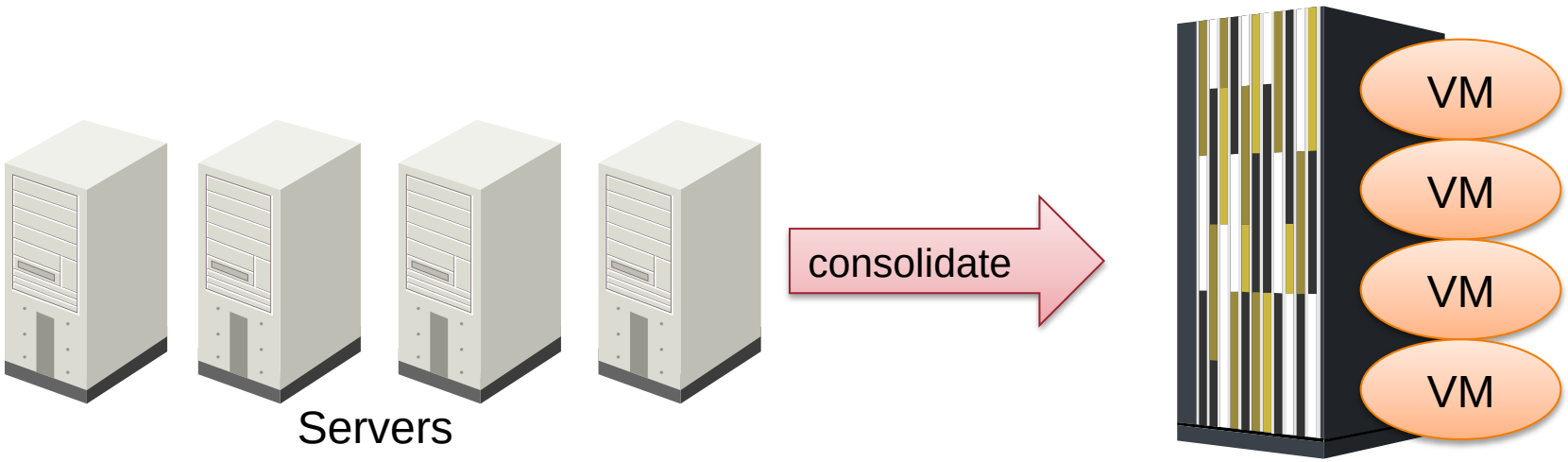
Linux Technology Center

Yokohama Research Lab., Hitachi, Ltd.

Background for Inter-VM Tracing

HITACHI
Inspire the Next

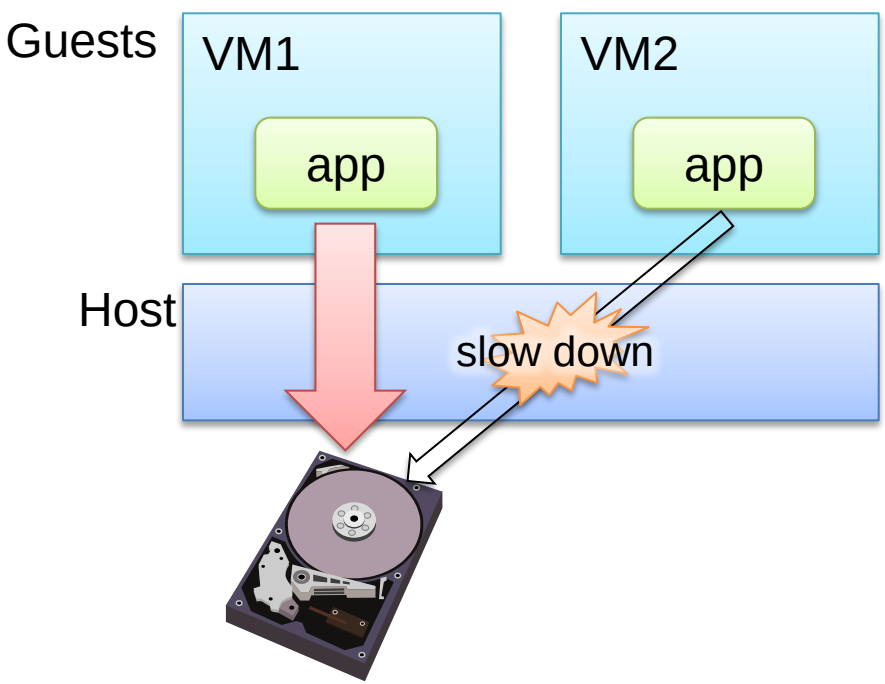
- Enterprise systems are moving on (private/public) cloud which uses virtualization technology and aim for system consolidation.
  - Multiple servers run on one physical system
  - This makes system trouble shooting harder

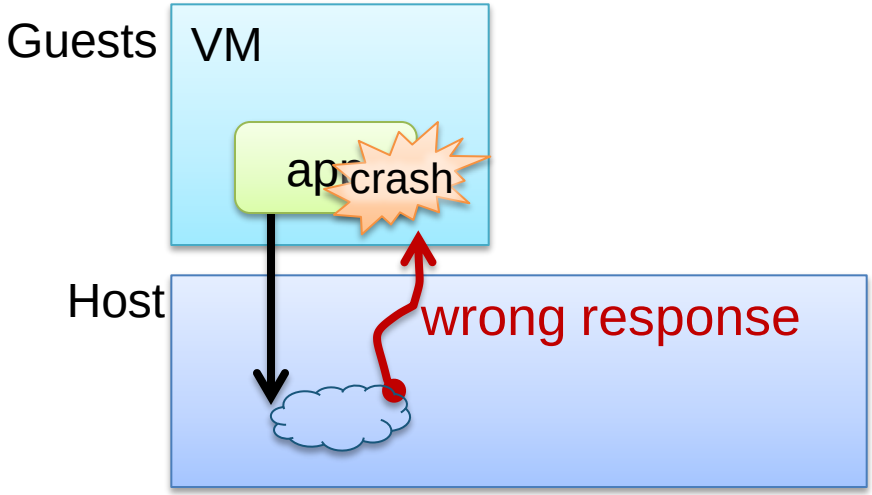


Troubles in Virtualized System

HITACHI
Inspire the Next

- A guest VM …
  - Can be affected by other VM operation
  - Can die by host OS or hypervisor's bug



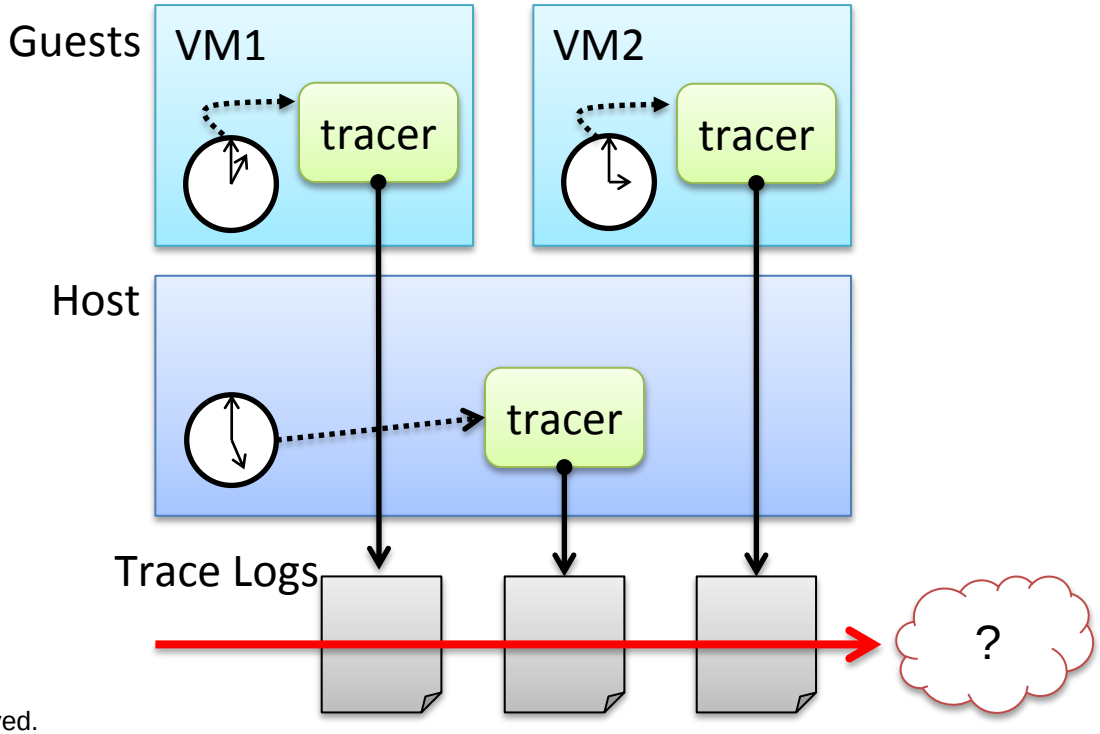


Host-Guest inter-VM tracing helps root cause analisys

Challenges of Inter-VM Tracing

HITACHI
Inspire the Next

- Tracing virtualized system has following challenges
  1. Synchronize time-stamp for each VM's log
     - Each VM has own clock
  2. Collect guests trace-log from host without overhead
     - Too huge trace logs (# of VM times of logs)

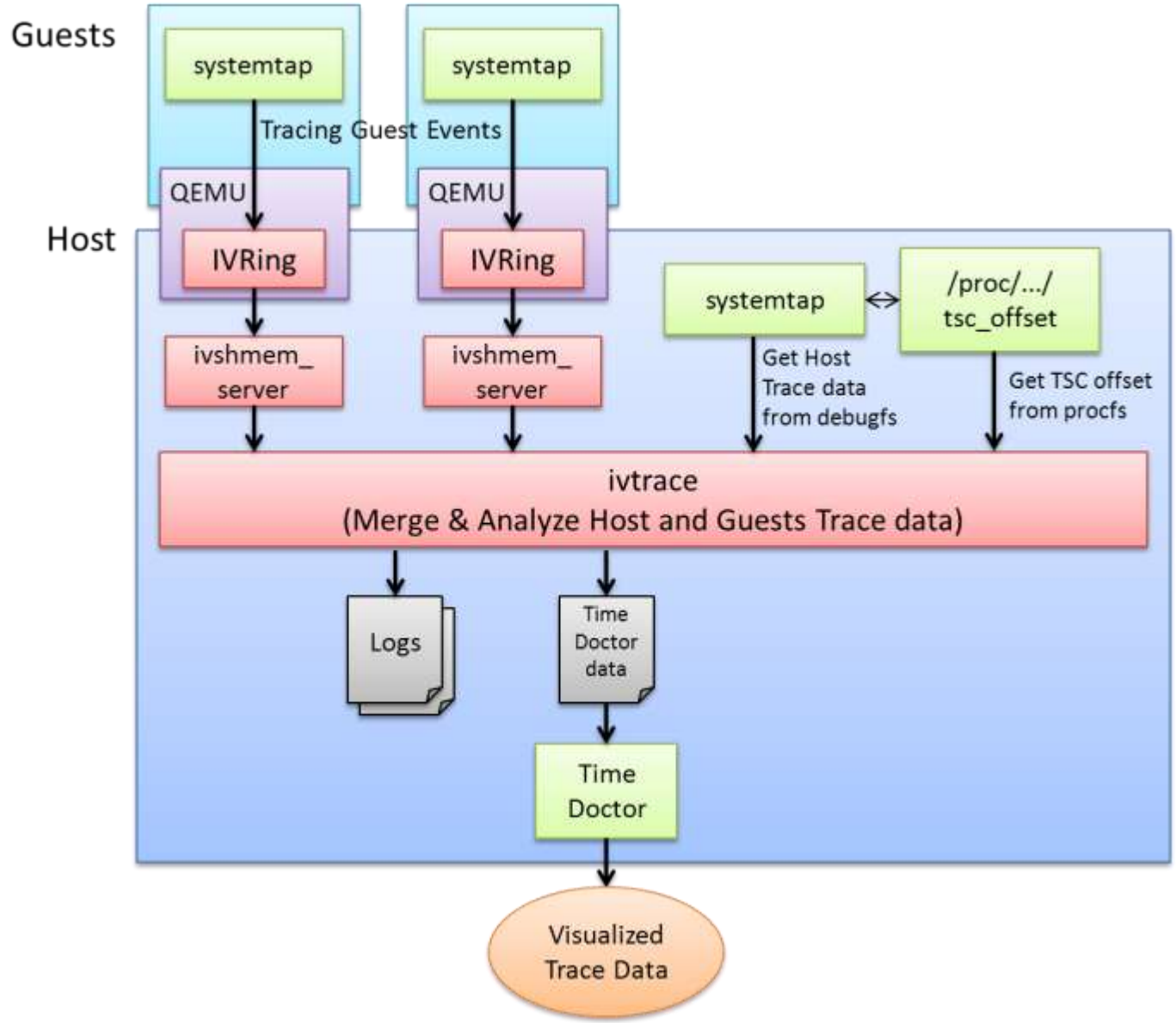


First Prototype (~May 2012)

HITACHI
Inspire the Next

- Use **systemtap** as a tracer
  - Trace both of kernel and user applications
- Use **tsc_offset** to adjust guests' time-stamp-counter(tsc)
  - This prototype use TSC for timestamp
  - Also, a VM is pinned on a CPU
- Use **IVRing** to pass the trace logs from guests to host
  - IVRing is an implementation of Inter-VM Ring buffer using IVShmem
- Use **TimeDoctor** to visualize trace data

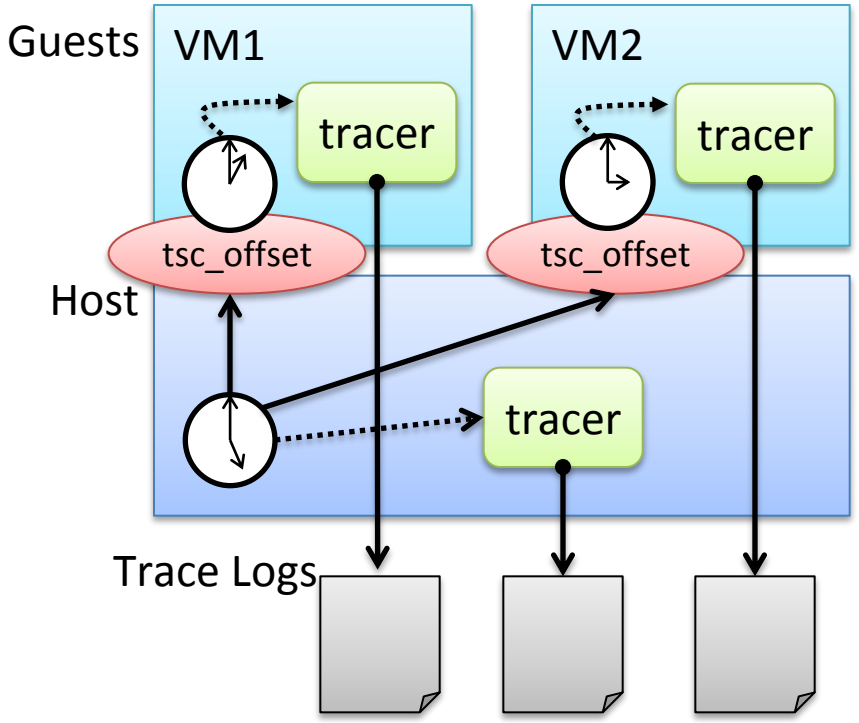
First Prototype Overview



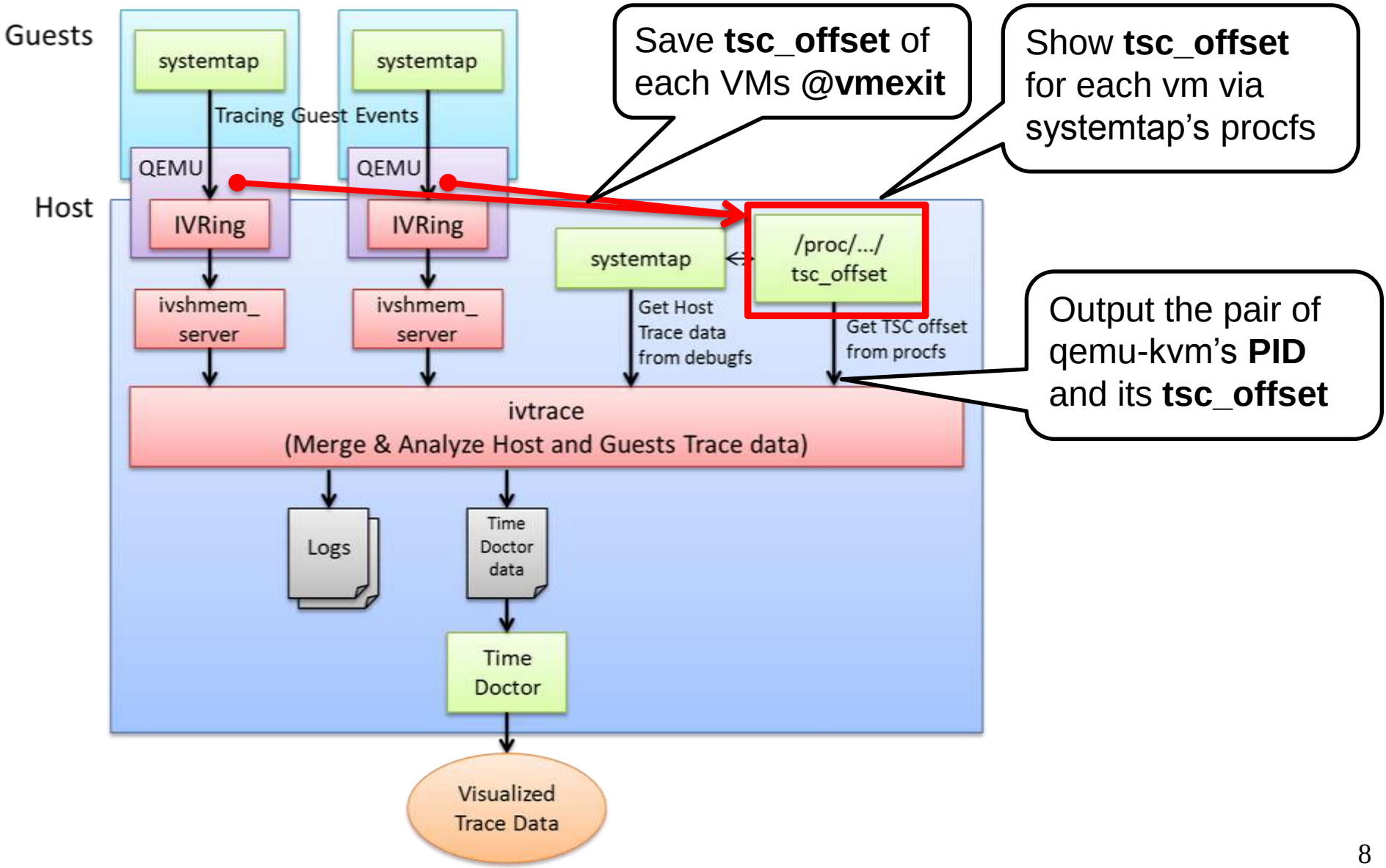
TSC on x86 Virtual Machine

HITACHI
Inspire the Next

- Each guests has virtualized independent tsc
  - The substitution of host's tsc and guests' one is called **tsc_offset**
  - Each tracers in guests uses own tsc to record time stamp, and we can adjusting this tsc using **tsc_offset** to merge logs.
  - Of course, this depends on "**constant_tsc**"

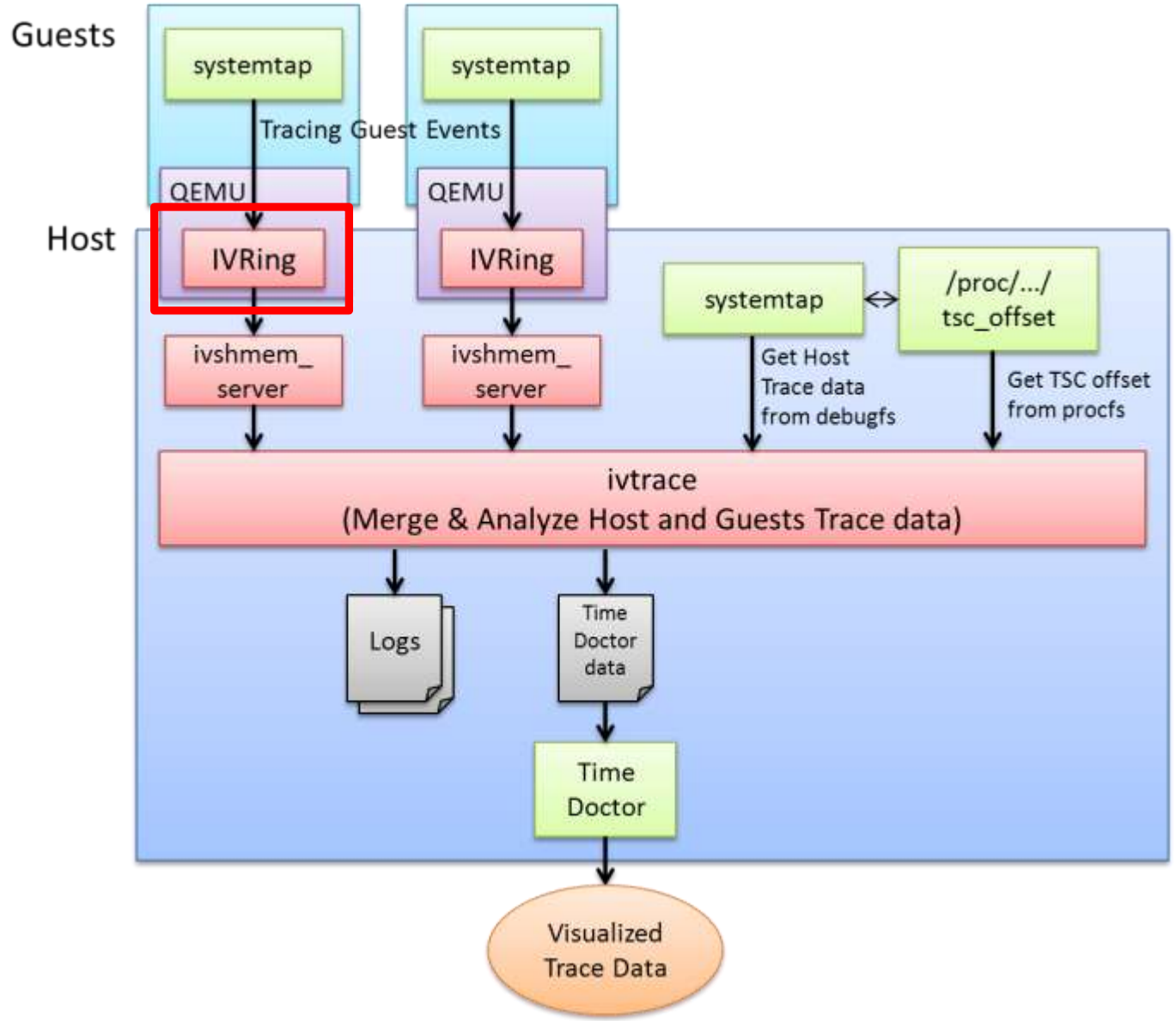


Adjusting TSC on Prototype



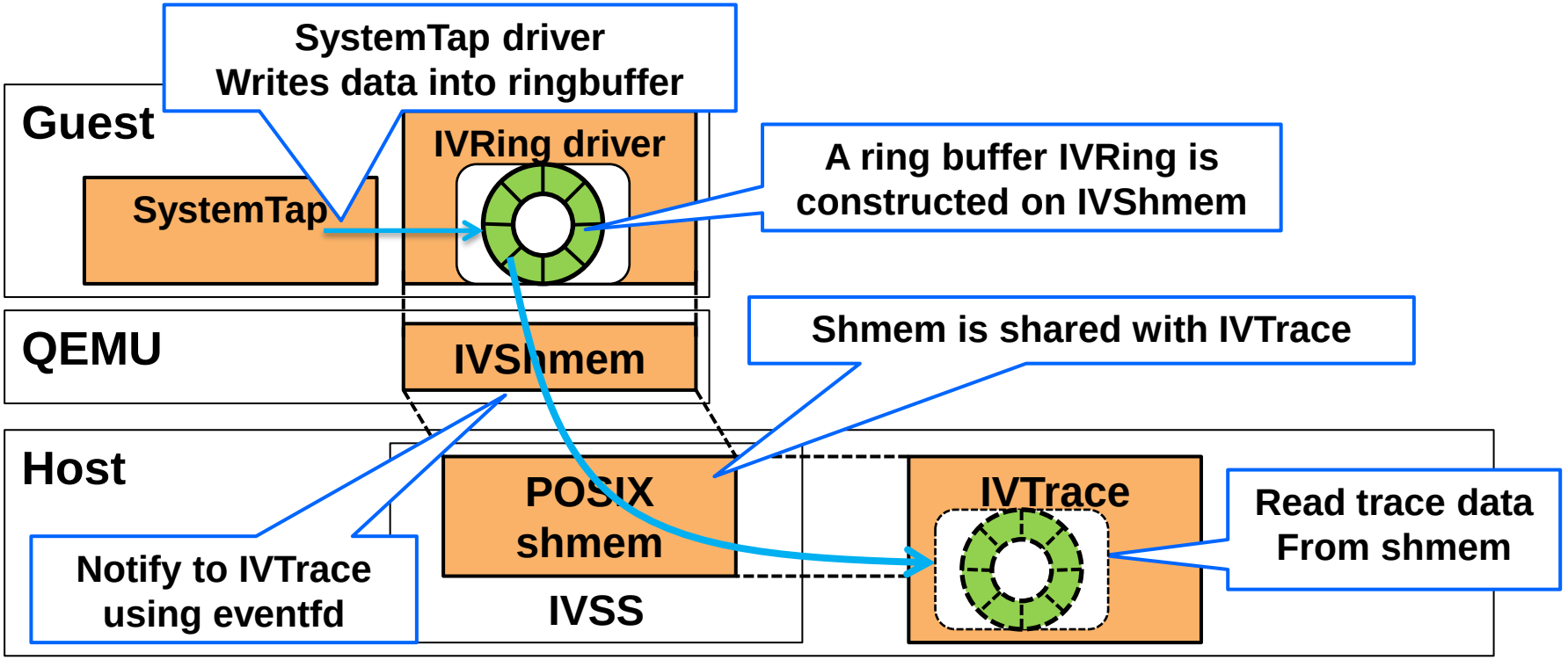
IVRing - Fast Interconnect for Tracing

HITACHI
Inspire the Next



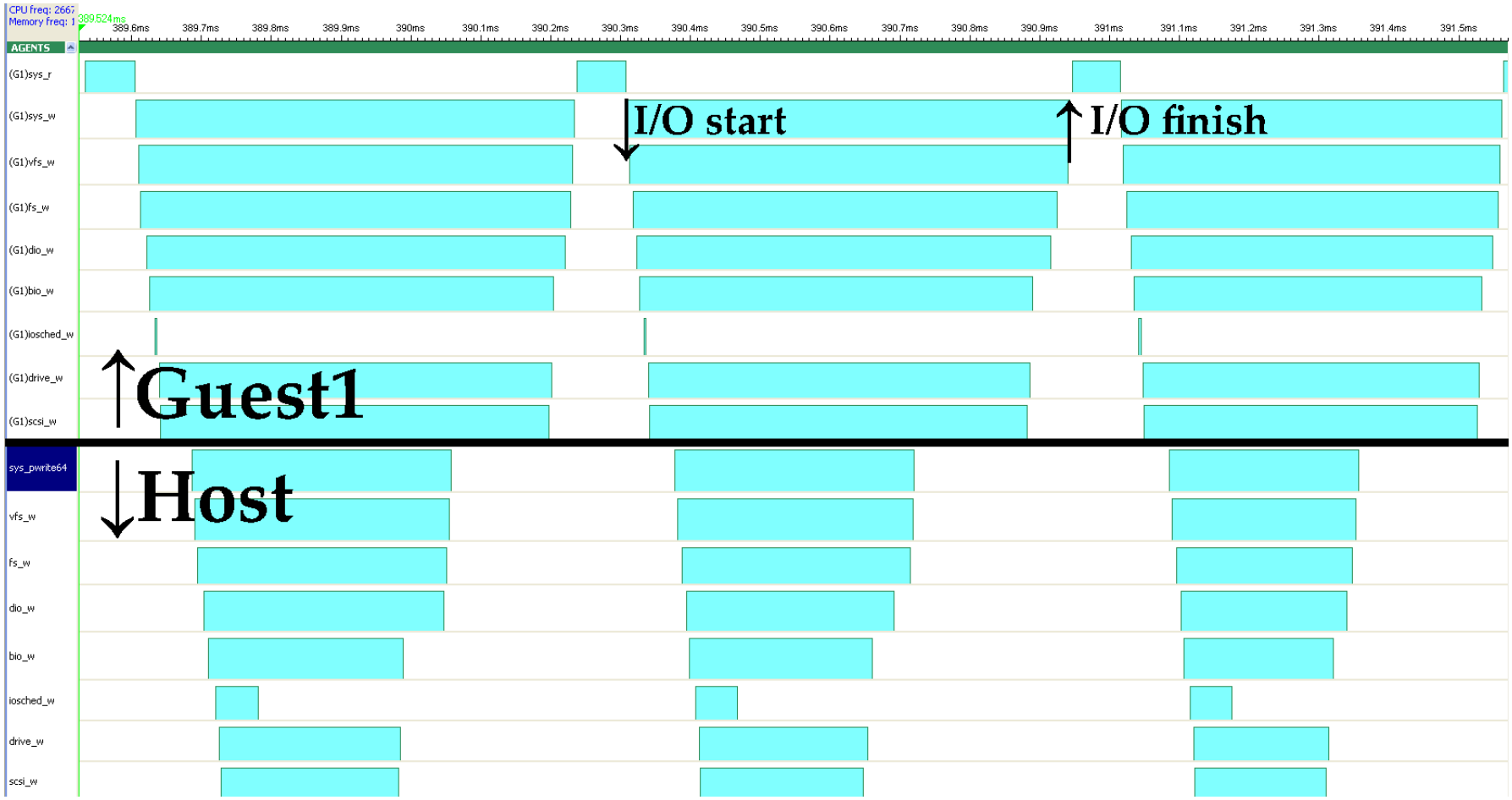
IVRing on IVShmem

- A ring-buffer IVRing is constructed on IVShmem as a data path for trace data of a guest.
- IVShmem is a memory-PCI device
  - Backend memory is a posix shmem.
- IVTrace can read the data without memory copying.



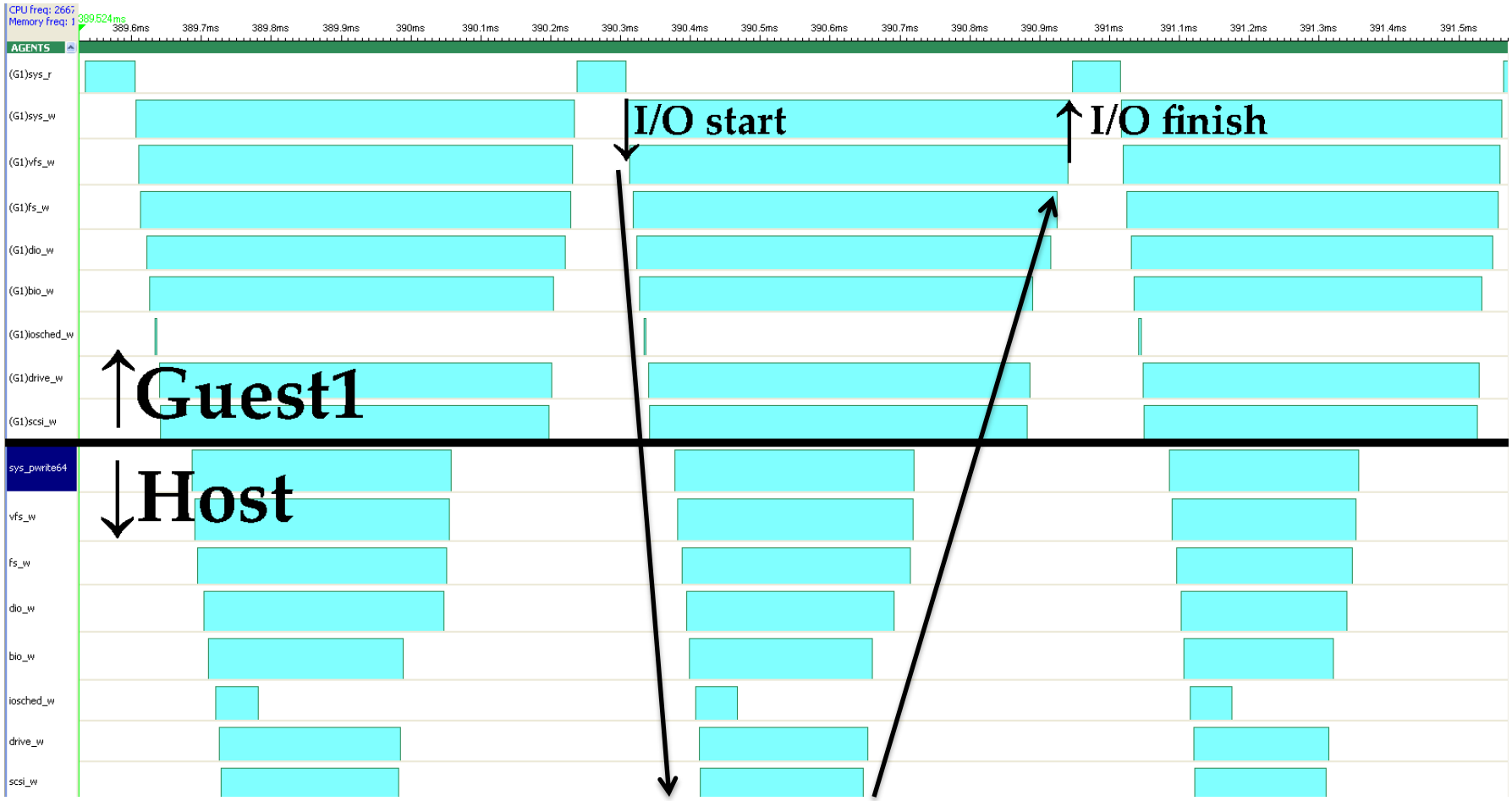
Result

- The case of no contentions



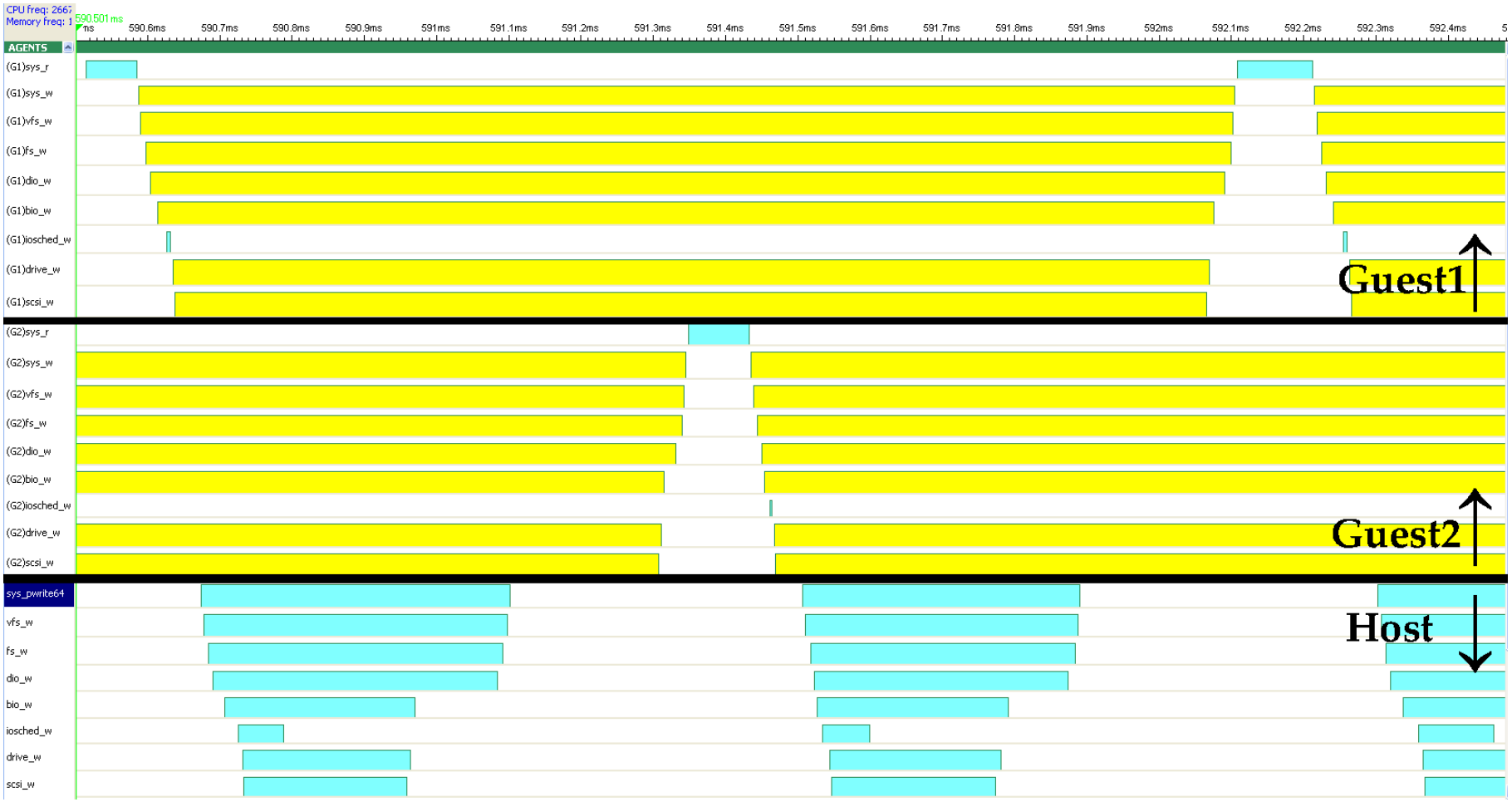
Result

- The case of no contentions



Result2

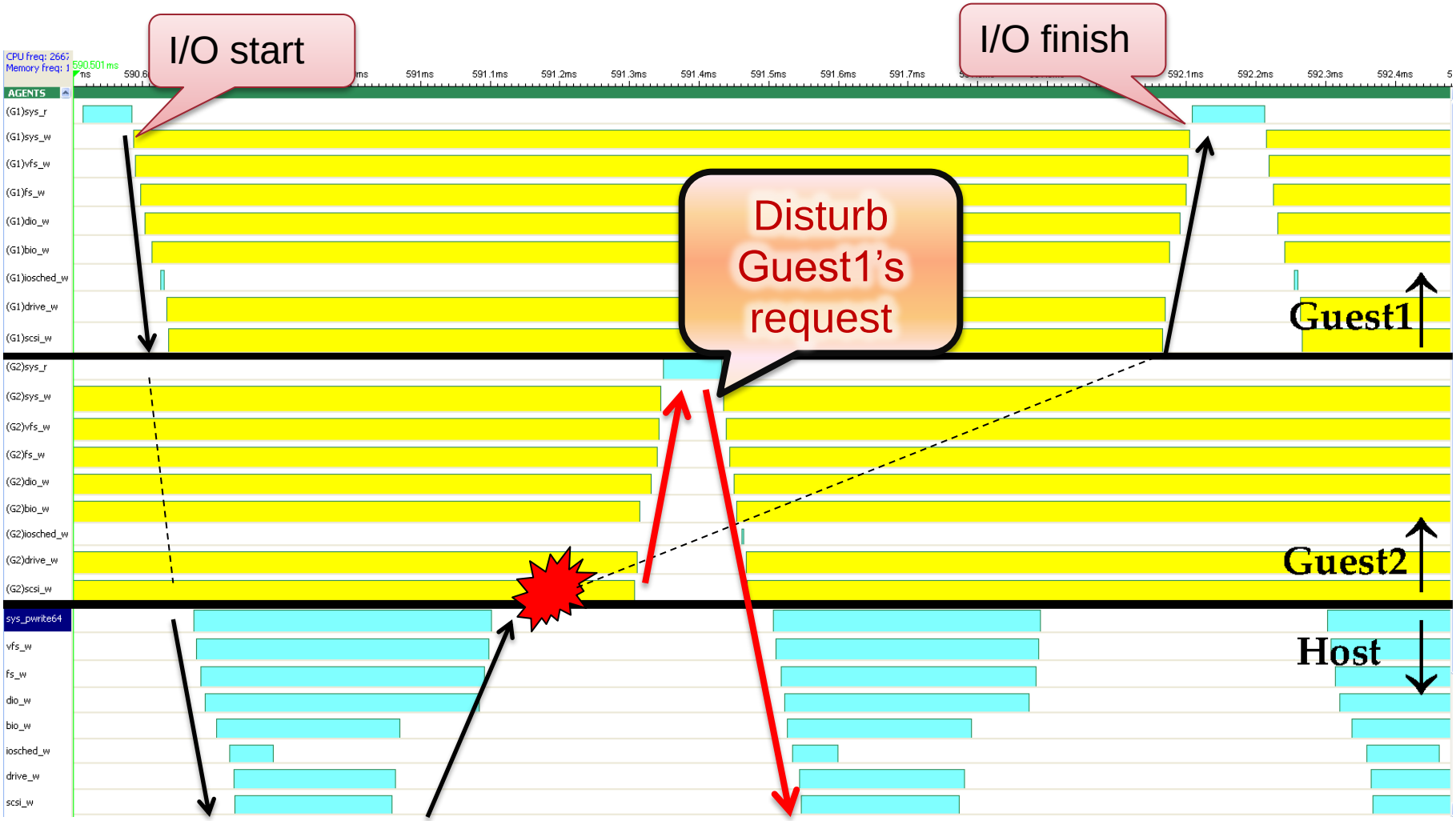
- The case of contentions. I/O slowdown



Result2

HITACHI
Inspire the Next

- The case of contentions. I/O slowdown



Proposal of IVRing on LKML

HITACHI
Inspire the Next

- Maintainers Don't like it

> (2012/06/06 8:22), **Greg Kroah-Hartman** wrote:
> > On Wed, Jun 06, 2012 at 07:03:06AM +0800, **Anthony Liguori** wrote:
> >> On 06/05/2012 09:10 PM, **Borislav Petkov** wrote:
> >>>
> **>>> Yet another ring buffer?**
> >>>
> >>> We already have an ftrace and perf ring buffer, can't you use one of those?
> >>
> **>> Not to mention virtio :-)**
> >>
> >> Why not just make a virtio device for this kind of thing?
> >
> **> Yeah, that's exactly what I was thinking, why reinvent things again?**

- Points
  - NO "yet another ring buffer" in linux kernel
    - Use ftrace and perf ring buffer for guest recording
  - Use virtio instead of ivshmem

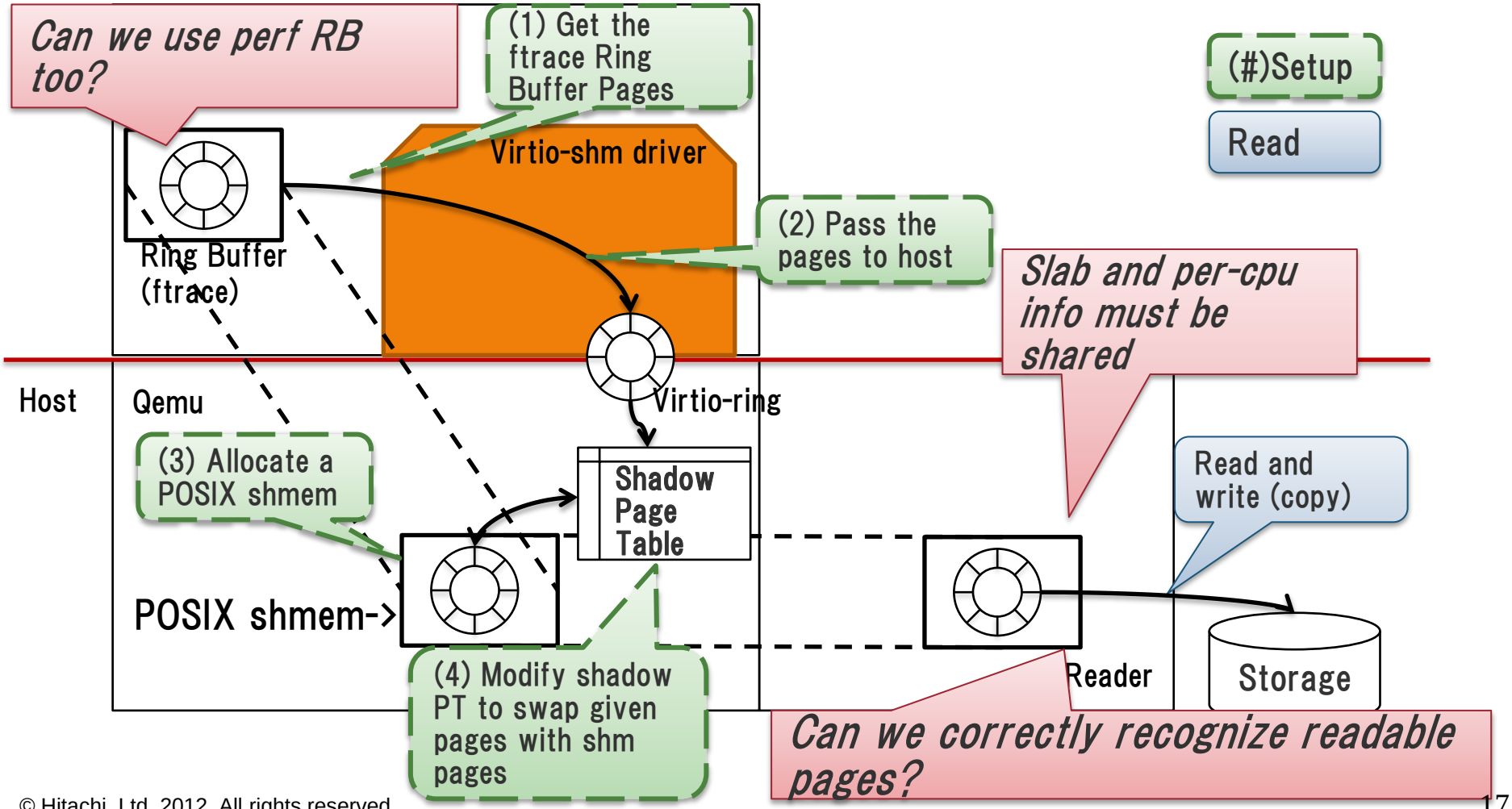
Virtio Candidates

HITACHI
Inspire the Next

- Virtio-shmem (new device)
  - Virtio device which provides APIs for assigning the guest pages to shared memory in host
    - Guest can assign any page to host's shmem
    - Qemu remaps original pages with the pages on shared memory
    - Similar to the ivshmem, but no big PCI address space required
- Virtio-serial with splice (enhancement)
  - Virtio device which provides chardev interface for guest
    - Guest can "splice" its data into the char device
    - Qemu copies data page to host-side pipe
    - Vhost can offload the copying process

Virtio-shmem Overview

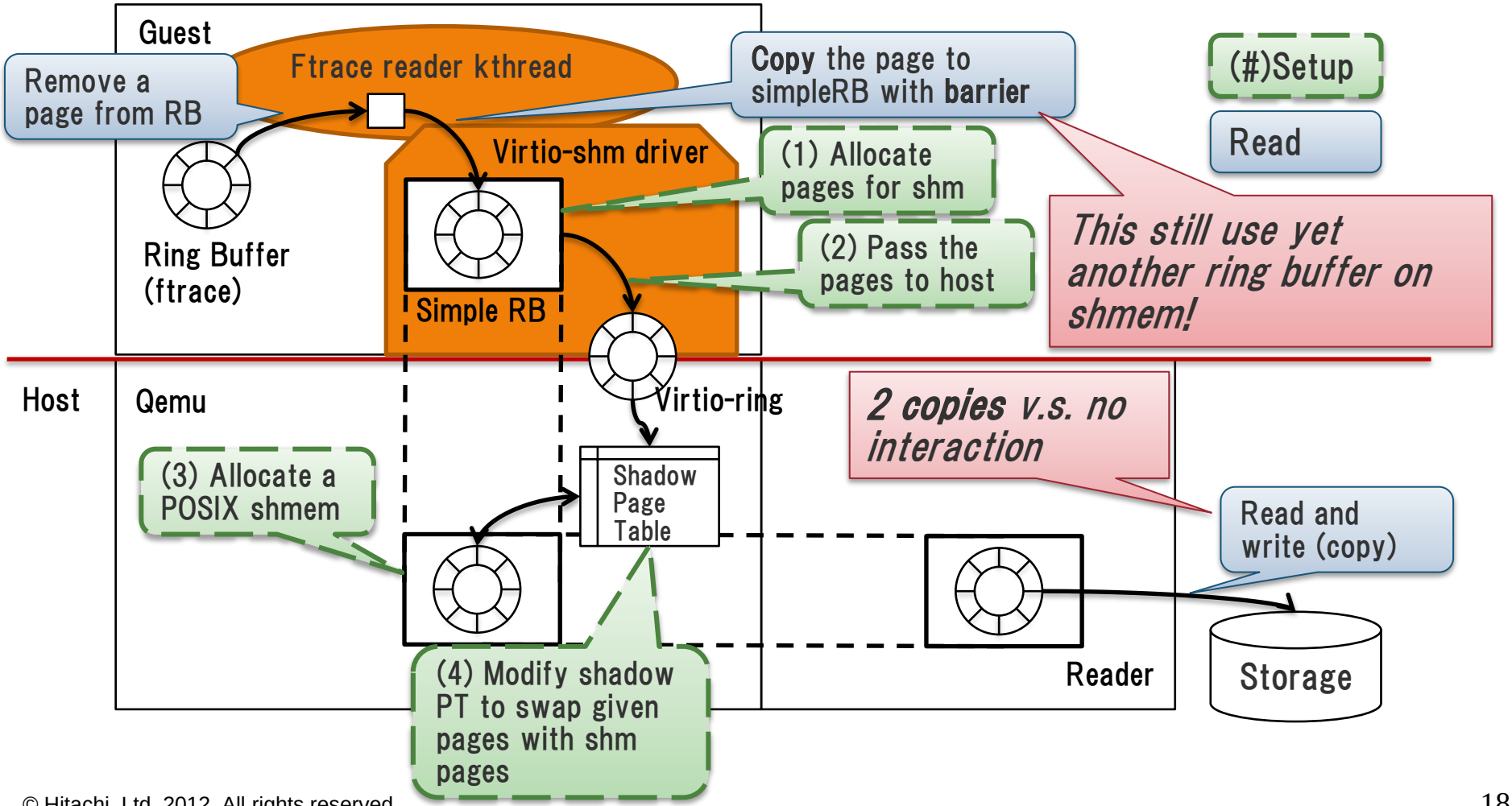
- Export Ftrace ring-buffer pages directly to host
  - How we can export per-cpu kmalloc object?



Virtio-shmem w/ RB Overview

HITACHI
Inspire the Next

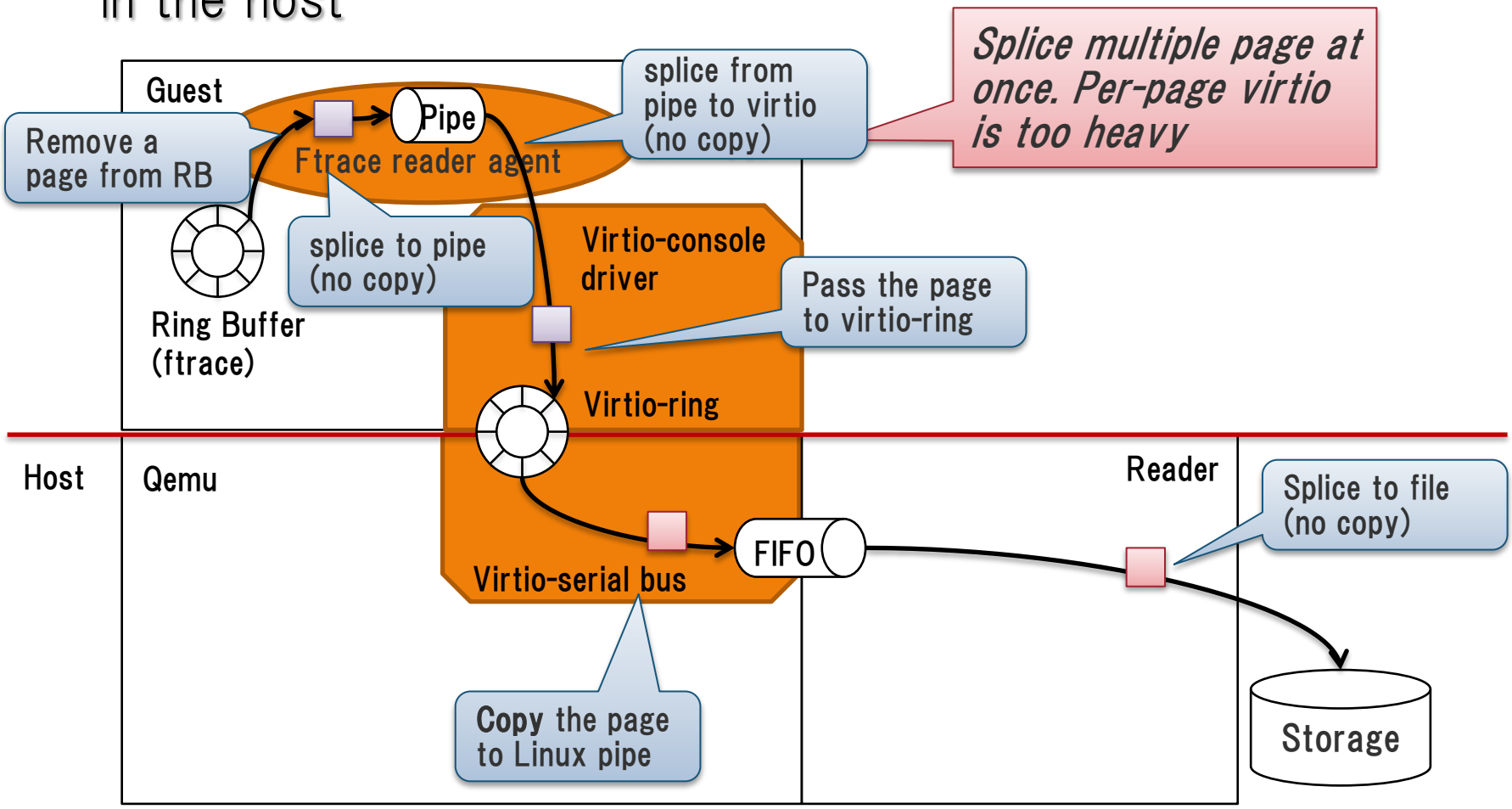
- To generalize interface and guarantee memory coherency
  - At first, we have simple ring-buffer for memory coherency



Virtio-serial w/ splice overview

HITACHI
Inspire the Next

- Virtio-serial opens chardev in the guest and FIFO(named pipe) in the host



Comparison

	IVRing	Virtio-serial w/ splice	Virtio-shmem w/ simpleRB	Virtio-shmem
Qemu-Reader	Shmem	Pipe (1way)	Shmem	Shmem
Guest-Qemu	Shmem(PCI)	virtio	Shmem(EPT)	Shmem(EPT)
# of Copies	1 (reader to file)	1 (virtio to pipe)	2 (in Guest, reader to file)	1 (reader to file)
Guest-host interaction	No	Once per I/O (16pages or more)	No	No
Supported tracer	SystemTap	Ftrace, user tools	Ftrace, and others?	Ftrace only
No another RB	IVRing	No	Simple RB	No
Use virtio	No	Yes	Hmm...	Hmm...
Buffer Resize	No	Support	Support	Support
SMP scaling	RB w/ lock	Per-cpu pipes	Per-cpu shmem	Per-cpu shmem
VCPU hot-add	Support (w/lock)	Add channels	Add shmem	Add shmem
Live migration	No	Possible	Possible	Possible
Expectation	Not scalable, not acceptable	Upstream Acceptable	Need Discussion	Need Discussion

Comparison

HITACHI
Inspire the Next

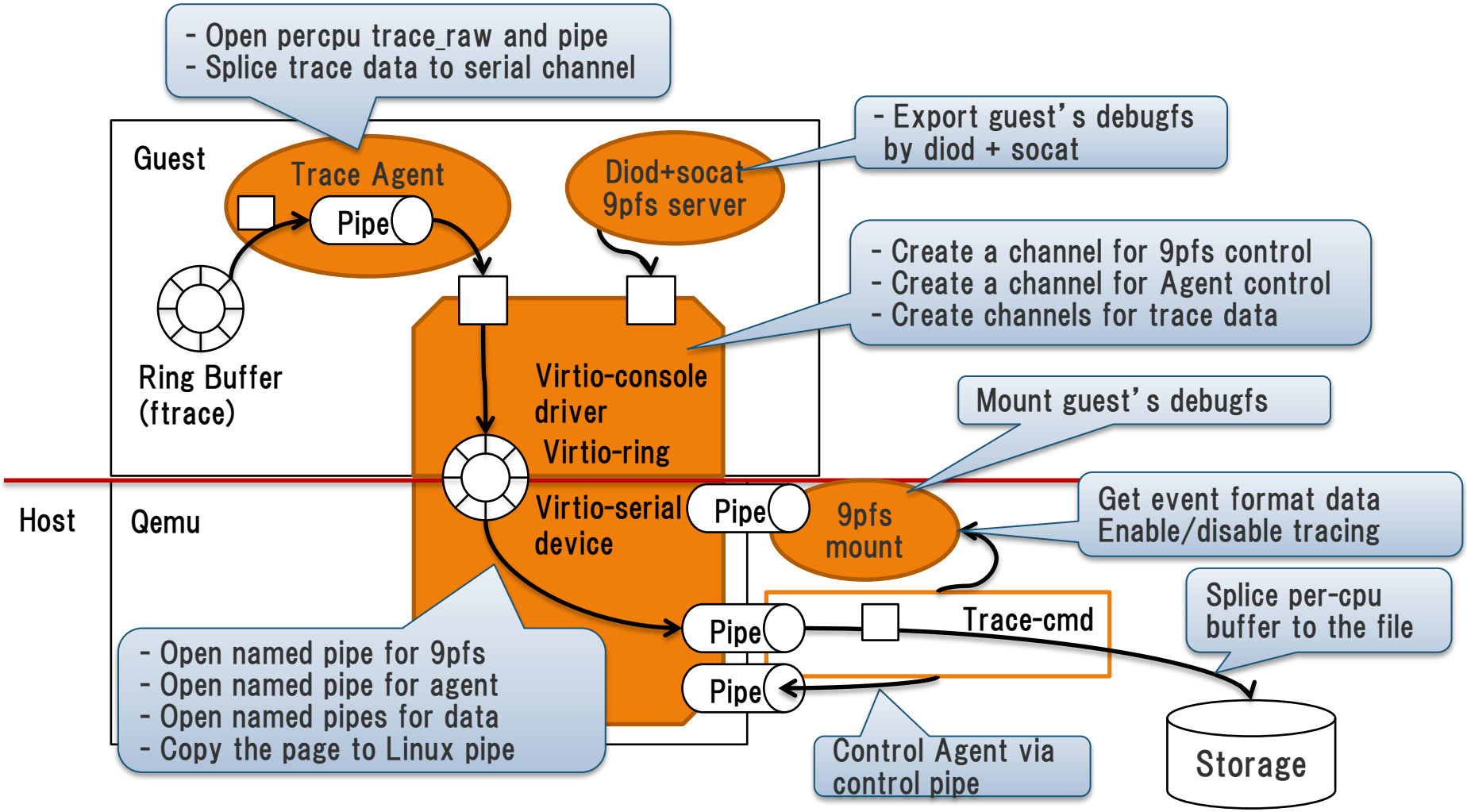
Chosen method
- Simple implementation
- No claiming points
- Fit to virtio framework

	IVRing	Virtio-serial w/ splice	Vi sir	w/o
Qemu-Reader	Shmem	Pipe (1way)	Sh	
Guest-Qemu	Shmem(PCI)	virtio	Shmem(EPT)	Shmem(EPT)
# of Copies	1 (reader to file)	1 (virtio to pipe)	2 (in Guest, reader to file)	1 (reader to file)
Guest-host interaction	No	Once per I/O (16pages or more)	No	No
Supported tracer	SystemTap	Ftrace, user tools	Ftrace, and others?	Ftrace only
No another RB	IVRing	No	Simple RB	No
Use virtio	No	Yes	Hmm...	Hmm...
Buffer Resize	No	Support	Support	Support
SMP scaling	RB w/ lock	Per-cpu pipes	Per-cpu shmem	Per-cpu shmem
VCPU hot-add	Support (w/lock)	Add channels	Add shmem	Add shmem
Live migration	No	Possible	Possible	Possible
Expectation	Not scalable, not acceptable	Upstream Acceptable	Need Discussion	Need Discussion

Virtio-trace(serial w/ splice) Prototype

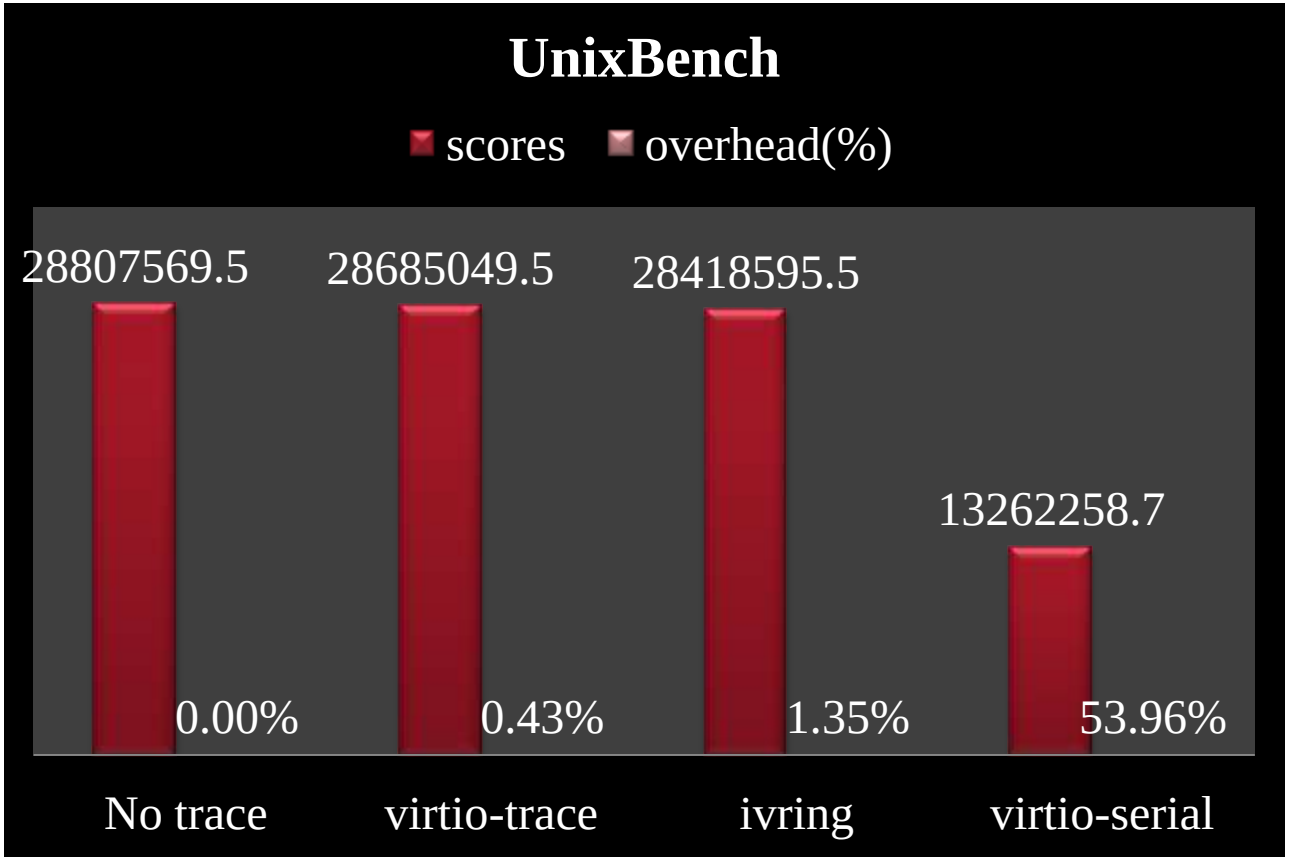
HITACHI
Inspire the Next

- Prototype consists of trace-agent, virtio-console driver, virtio-pipe device, and trace-cmd



Performance Measurement

- Compared with IVRing, virtio-trace is really fast?
  - Running UnixBench with tracing on guest VM



Host:
Xeon x5660
2.8Ghz/48GB

Guest:
Single-Core VM
1GB

Tracing:
Scheduler and
Syscall events

Compared with ivring, virtio-trace has lower overhead!

Virtio-trace on LKML

HITACHI
Inspire the Next

- Maintainers accept it ☺

> On Thu, 09 Aug 2012 21:30:29 +0900, Yoshihiro YUNOMAE <yoshihiro.yunomae.ez@hitachi.com> wrote:
> > Hi All,
> >
> > The following patch set provides a low-overhead system for collecting kernel
> > tracing data of guests by a host in a virtualization environment.
>
> Thankyou!
>
> I've applied this, and it will head into linux-next in the next few
> days.
>
> Cheers,
> Rusty.

- Points
  - NO "yet another ring buffer" in linux kernel
    - Use virtio ring for passing data
  - More generic feature (not only for tracing)
    - Maybe useful for other use, like SPICE

TODOs

HITACHI
Inspire the Next

- So, what is the next step?
  - Synchronize time-stamp between guest and host
    - Tsc-based trace_clock is an option (for constant_tsc machine)
    - Or agent gives the guest's trace_clock offset (but how?)
      - In generic, vmexit pattern matching can give us a hint.
  - Consolidate 9pfs server to trace-agent
    - For simplicity and ease of use (reducing steps of setup)
    - We can prepare setup script for guest tracing
  - Fix some issues on Qemu's chardev(serial backend)
    - Hotplug issue
    - Guest blocking issue
  - Live migration support

Conclusion

HITACHI
Inspire the Next

- IVRing is not acceptable for upstream
- Improved virtio-serial to support splice is accepted
  - Build the prototype and measure the performance
- Still under development...
  - Make setup easier
  - Time synchronizing
  - Clarify “chardev” issues on Qemu
  - Fix Qemu chardev for CPU-hotplug and non-blocking

Previous Talks

HITACHI
Inspire the Next

- See, IVTrace slide @ LinuxCon Japan 2012
  - Low-Overhead Ring-Buffer of Kernel Tracing
    http://events.linuxfoundation.org/images/stories/pdf/lcjp2012_yunomae.pdf
  - Tracing Across Host OS and Guest OS
    http://events.linuxfoundation.org/images/stories/pdf/lcjp2012_nagai.pdf

Legal Statements

HITACHI
Inspire the Next

- Linux is a registered trademark of Linus Torvalds.
- UNIX is a registered trademark of The Open Group.
- All other trademarks and copyrights are the property of their respective owners.