Low-Overhead Ring-Buffer of Kernel Tracing in a Virtualization System

Yoshihiro Yunomae

Linux Technology Center

Yokohama Research Lab.

Hitachi, Ltd.

HITACHI
Inspire the Next

Introducing

1. Purpose of a low-overhead ring-buffer in a virtualization system
2. “IVRing”, a low-overhead ring-buffer
3. IVRing VS general methods
4. How do we implement a ring-buffer?
5. Summary and future work

In this presentation…

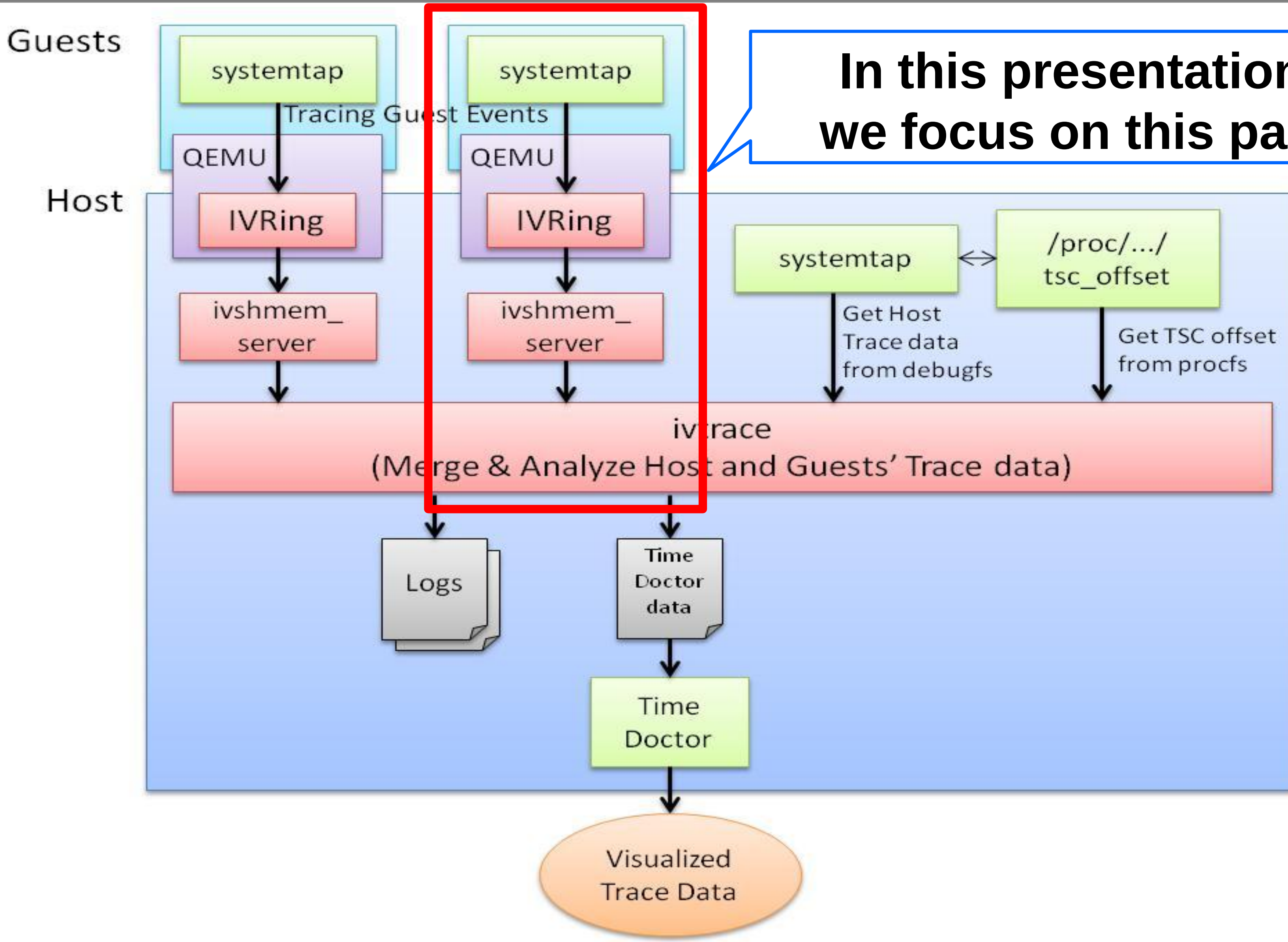
[1] To talk about new tracing buffer(1-3)

[2] To share problems of our implementation(4)

Introducing

1. Purpose of a low-overhead ring-buffer in a virtualization system
2. “IVRing”, a low-overhead ring-buffer
3. IVRing VS general methods
4. How do we implement a ring-buffer?
5. Summary and future work

Overview



In this presentation, we focus on this part.

Issue of Present Tracing in A Virtualization System

- Need to send trace data from guests to a host

⇒ One of methods is to use network I/O.

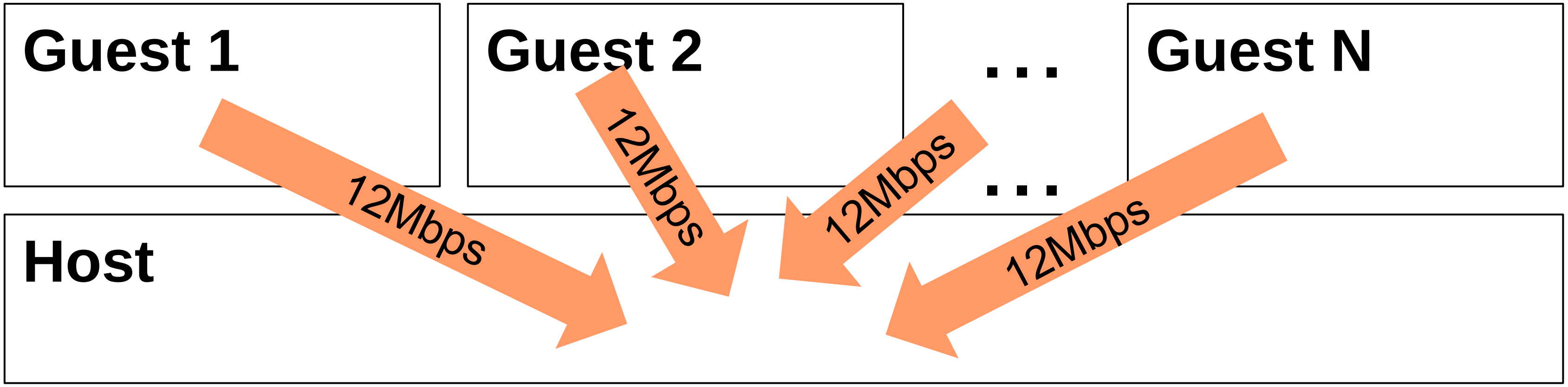
-To merge all trace data, a lot of data are sent.

⇒ High bandwidth, MAX12Mbps a guest, are required.

[15000(pb/s) * 100(byte/pb) * 8(bit/byte) ~ 12Mbps]

*pb: probes

- Using network I/O takes high overhead for application on guests.



Goal and Methods

<Goal>

To minimize effects for applications on guests

⇒ Decrease overhead caused by high-bandwidth tracing

<Methods>

(1) SSH & stdout ⇒ use network I/O

(2) NFS ⇒ use network I/O and disk I/O

(3) IVShmem

- Zero-copy communication between a guest and a host

⇒ We don't need to use network I/O and disk I/O.

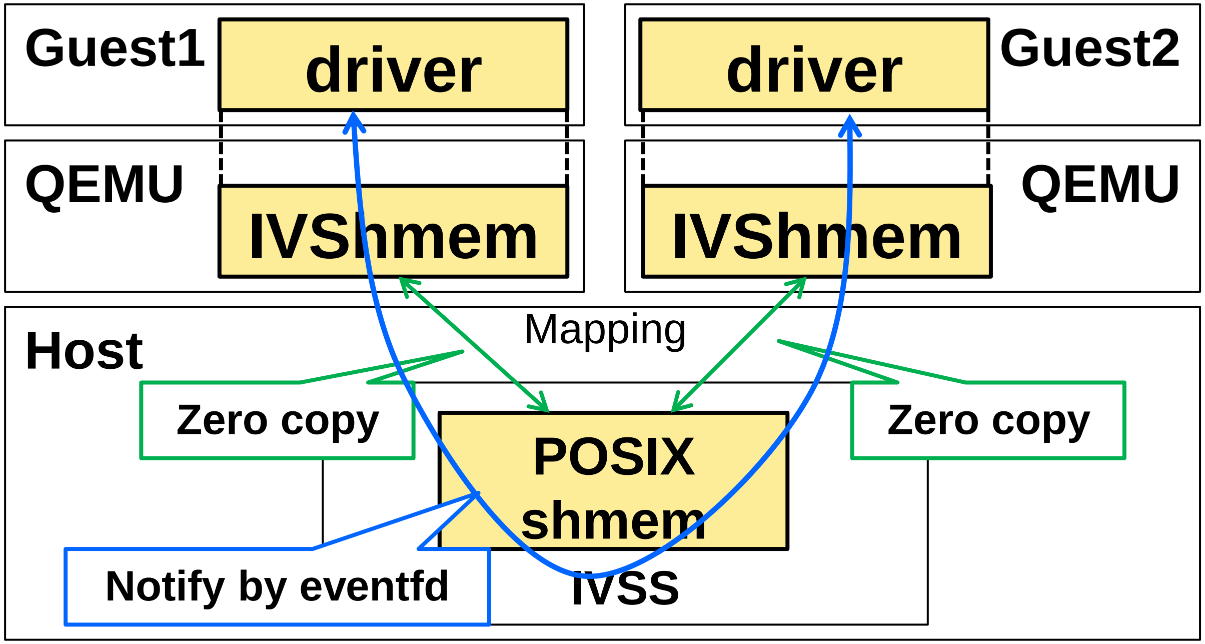
We adopted the IVShmem method.

Introducing

1. Purpose of a low-overhead ring-buffer in a virtualization system
2. “IVRing”, a low-overhead ring-buffer
3. IVRing VS general methods
4. How do we implement a ring-buffer?
5. Summary and future work

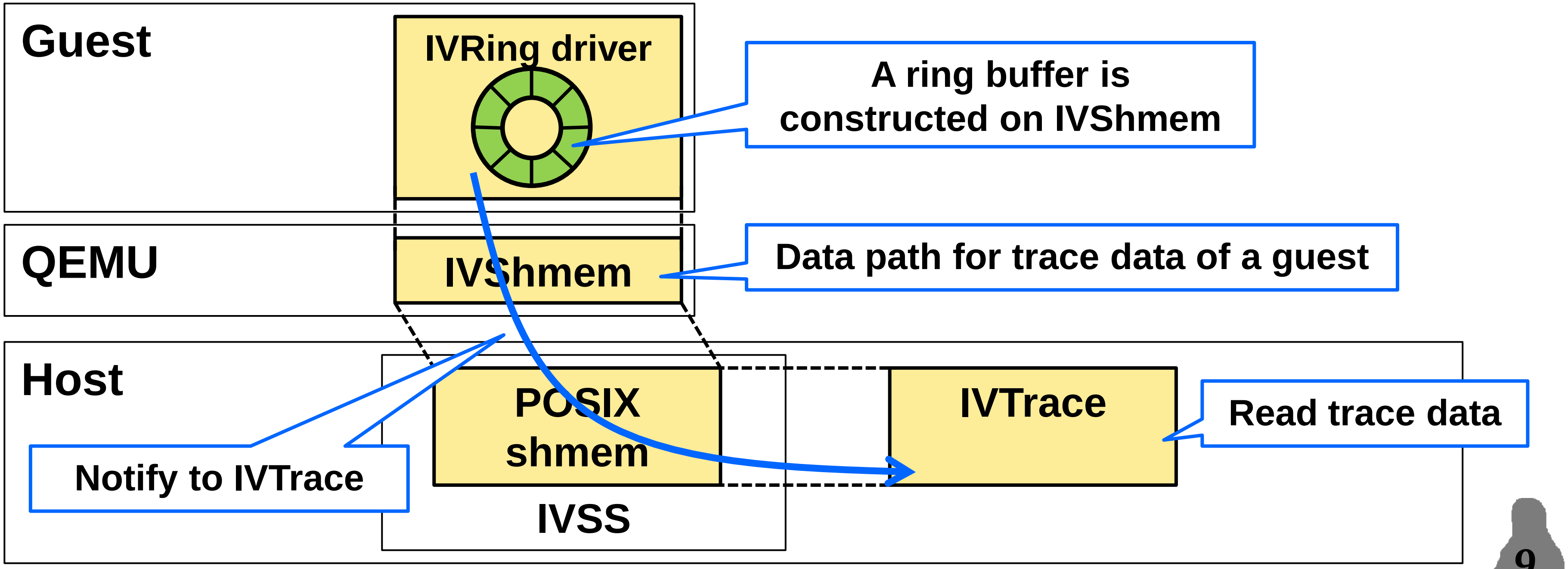
What is IVShmem?

- A virtual PCI RAM device originally for communication between two guests
  - ivshmem_server(IVSS) maps IVShmem POSIX shmem on a host.
  - Eventfd is available. ⇒notify to another guest



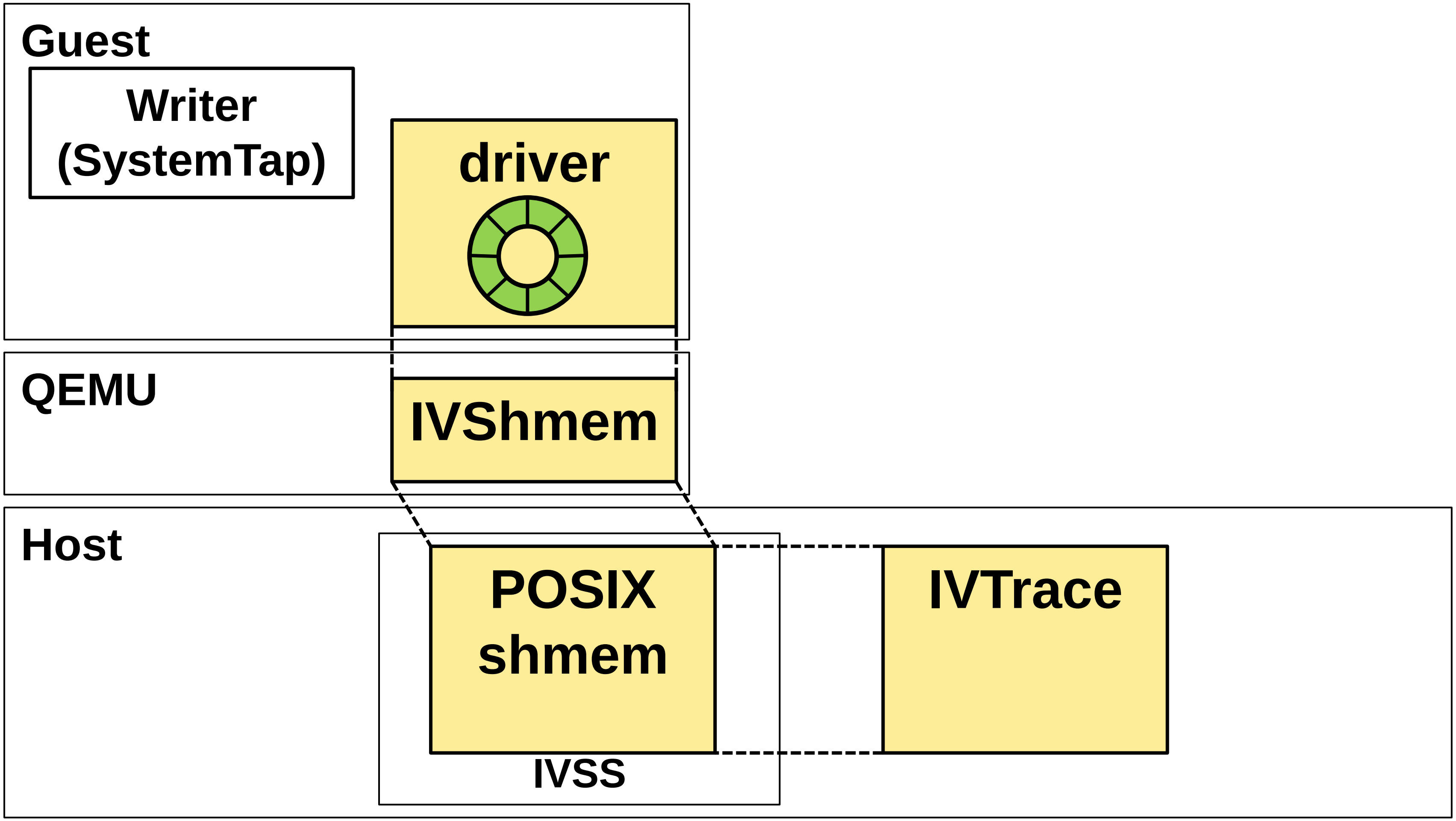
A Ring-Buffer on IVShmem

- A ring-buffer is constructed on IVShmem as a data path for trace data of a guest.
- IVTrace can read the data without memory copying.
  - A driver notifies to IVTrace, and IVTrace statrs to read trace data.
    - We use eventfd to notify to IVTrace.



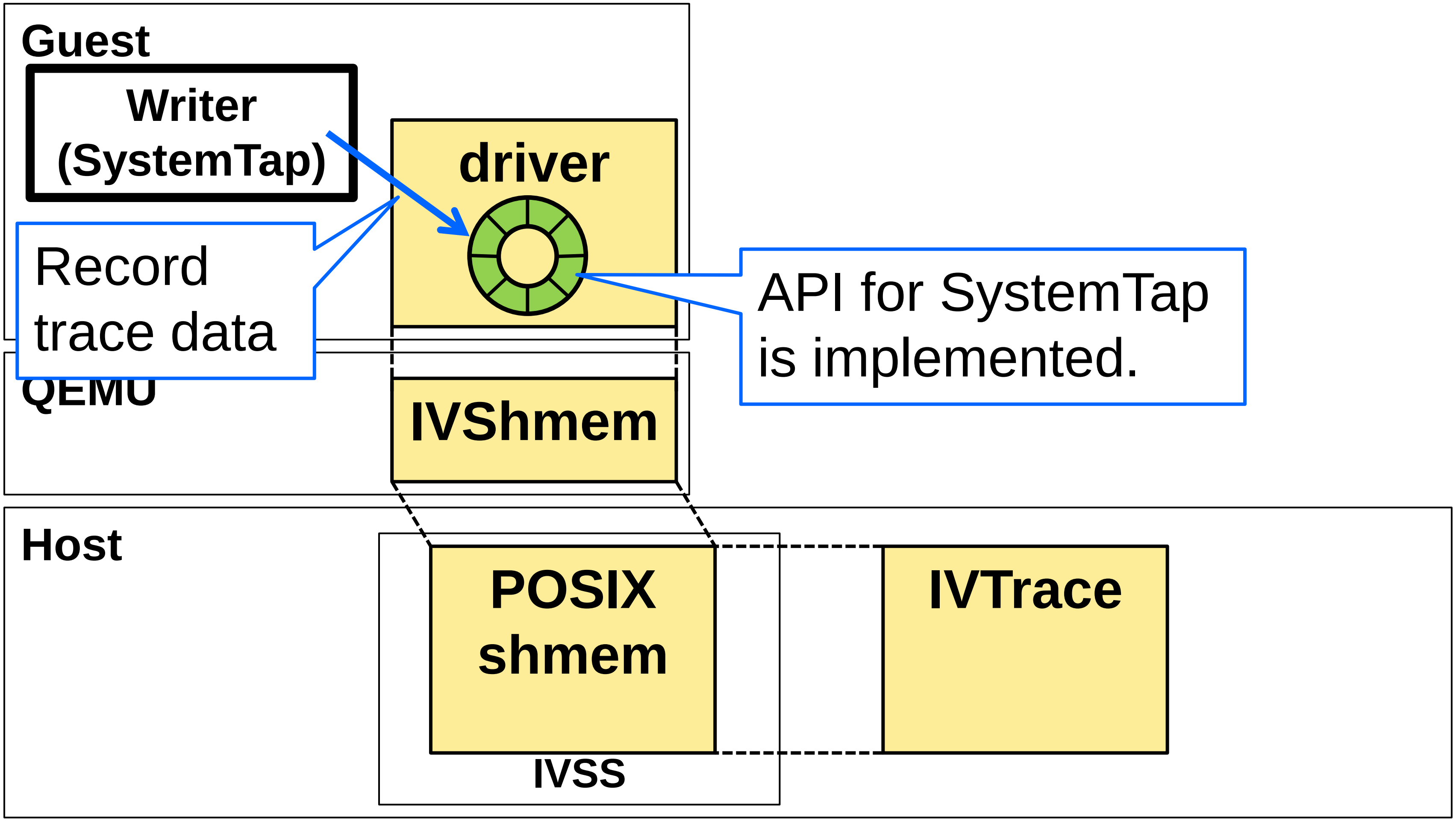
Copyright (c) 2012 Hitachi LTD., Yokohama Research Laboratory. All right reserved

Tne compornents of IVRing



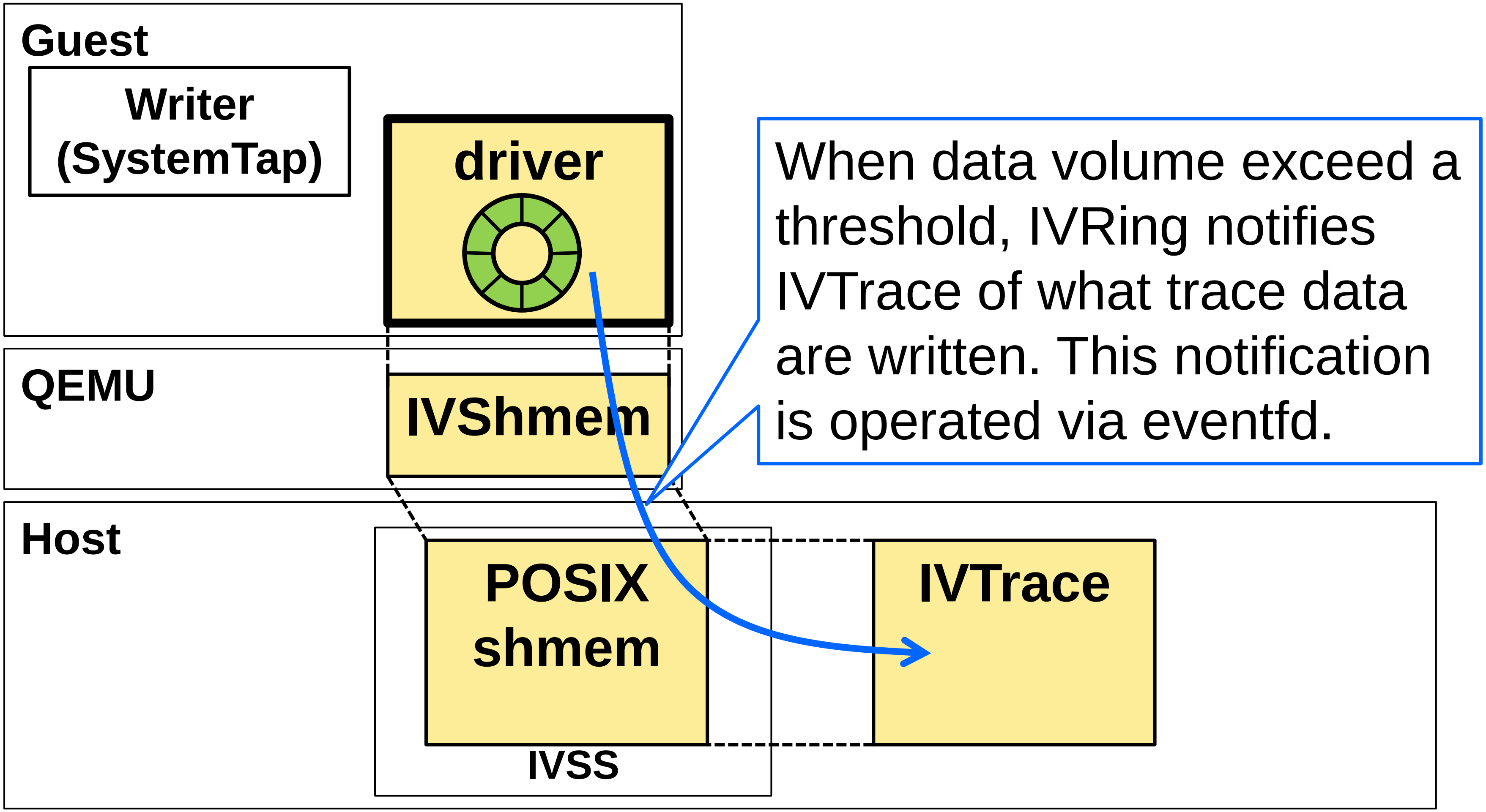
IVRing

API for SystemTap is implemented on a ring-buffer.



IVRing

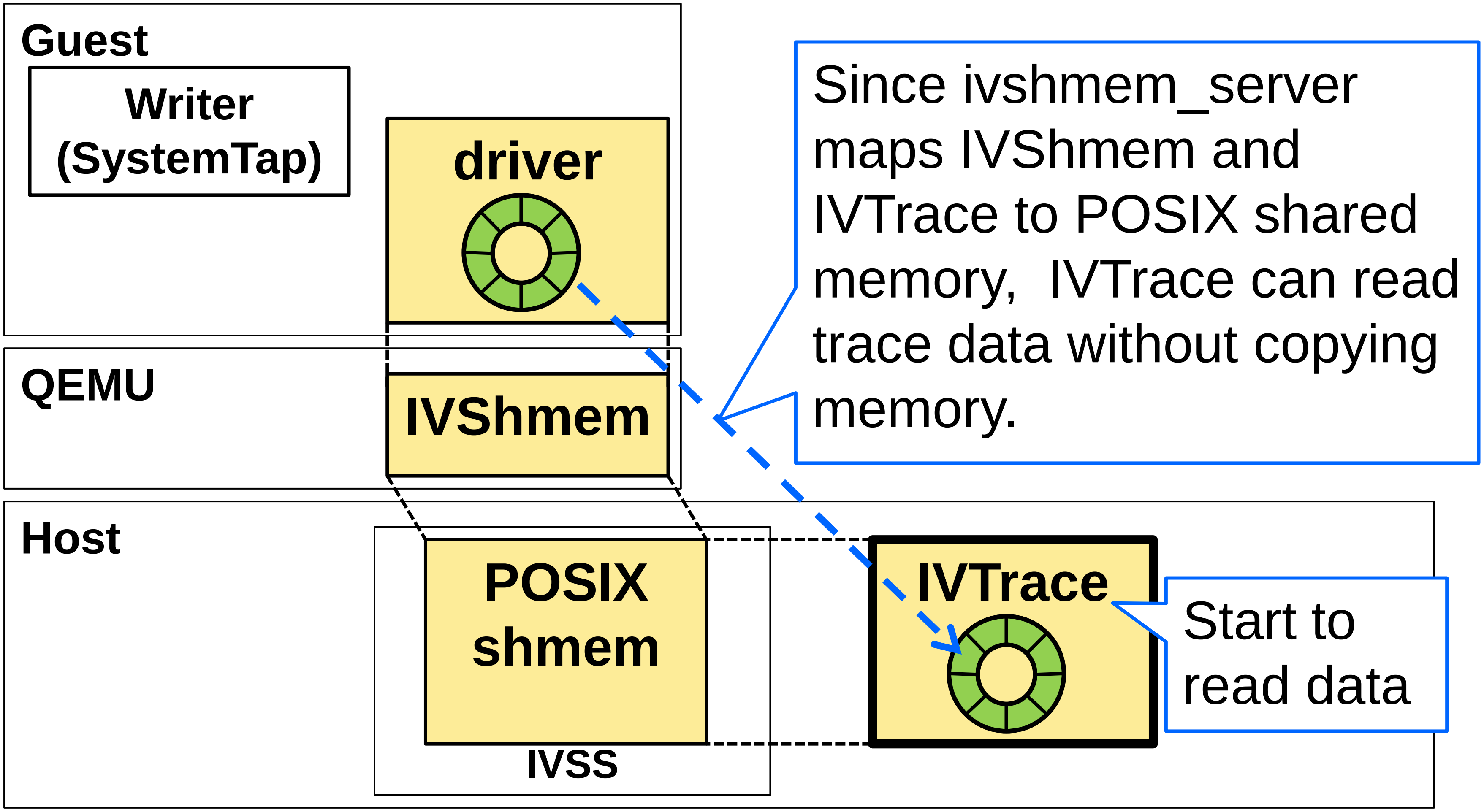
Notification using eventfd makes IVTrace operate.



When data volume exceed a threshold, IVRing notifies IVTrace of what trace data are written. This notification is operated via eventfd.

IVRing

IVTrace reads a ring buffer without copying memory.

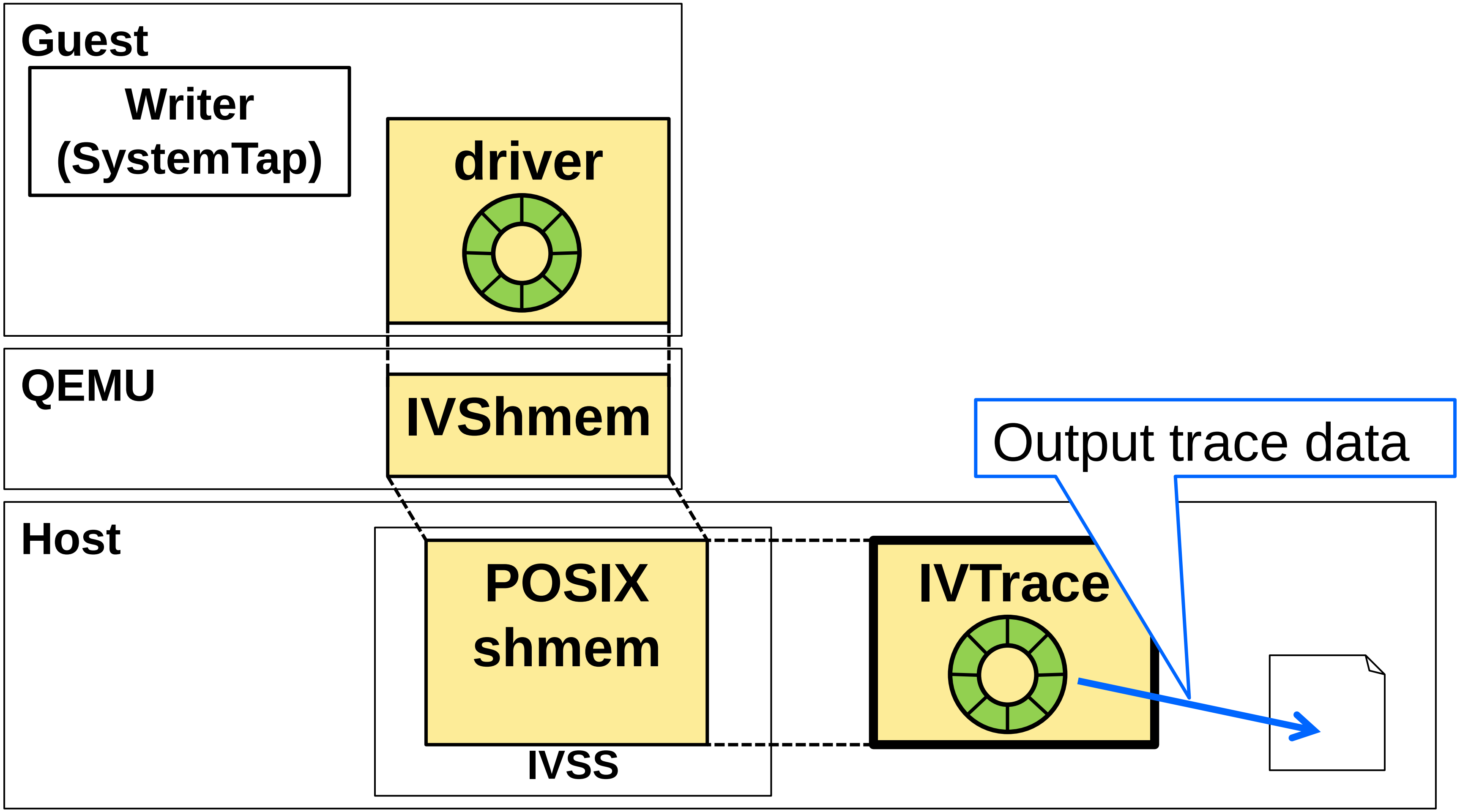


Since ivshmem_server maps IVShmem and IVTrace to POSIX shared memory, IVTrace can read trace data without copying memory.

Start to read data

IVRing

IVTrace outputs trace data of a guest.



Introducing

1. Purpose of a low-overhead ring-buffer in a virtualization system
2. “IVRing”, a low-overhead ring-buffer
3. IVRing VS general methods
4. How do we implement a ring-buffer?
5. Summary and future work

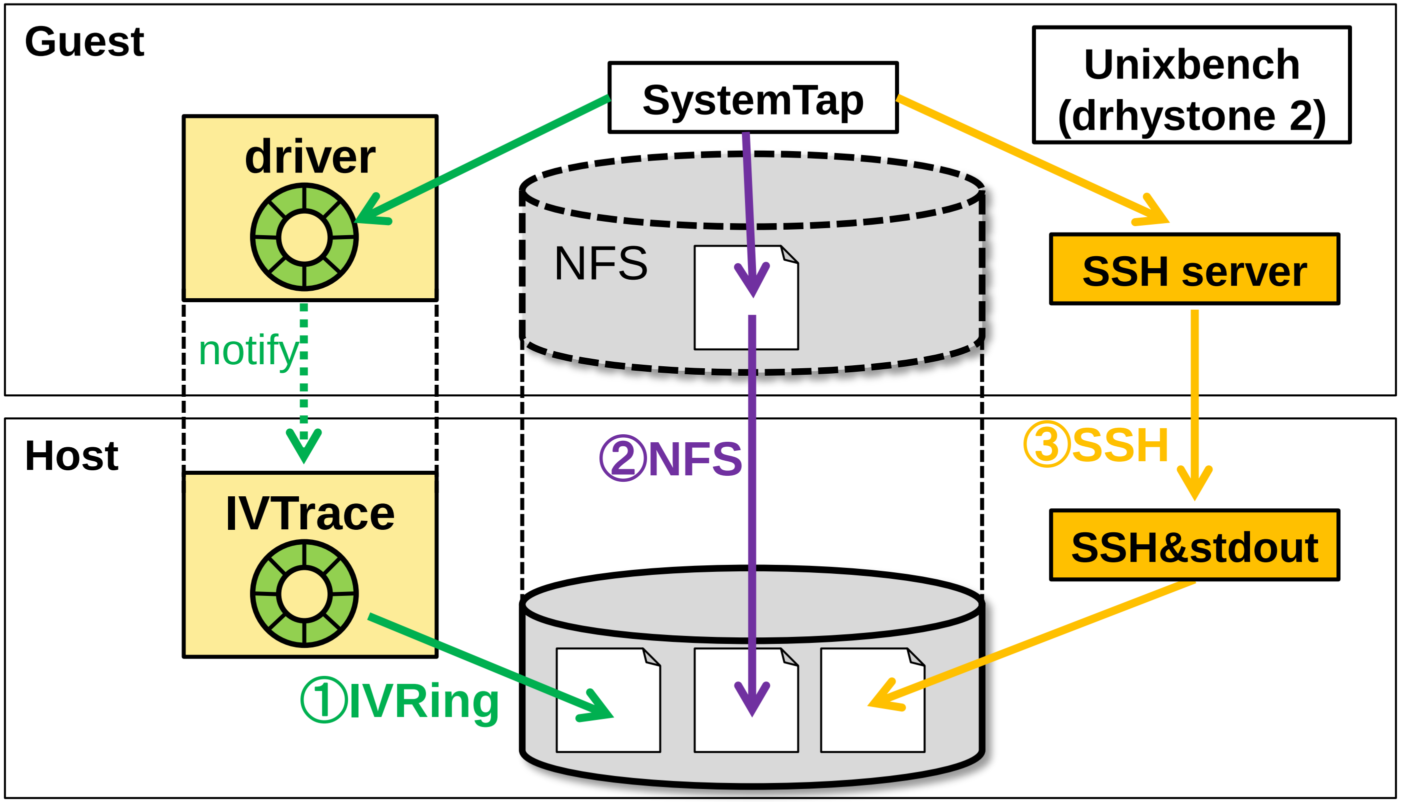
Evaluation

We compared the performance of each method.

① IVRing: record trace data in IVRing

② NFS: output trace data on a NFS

③ SSH: output trace data using stdout via SSH

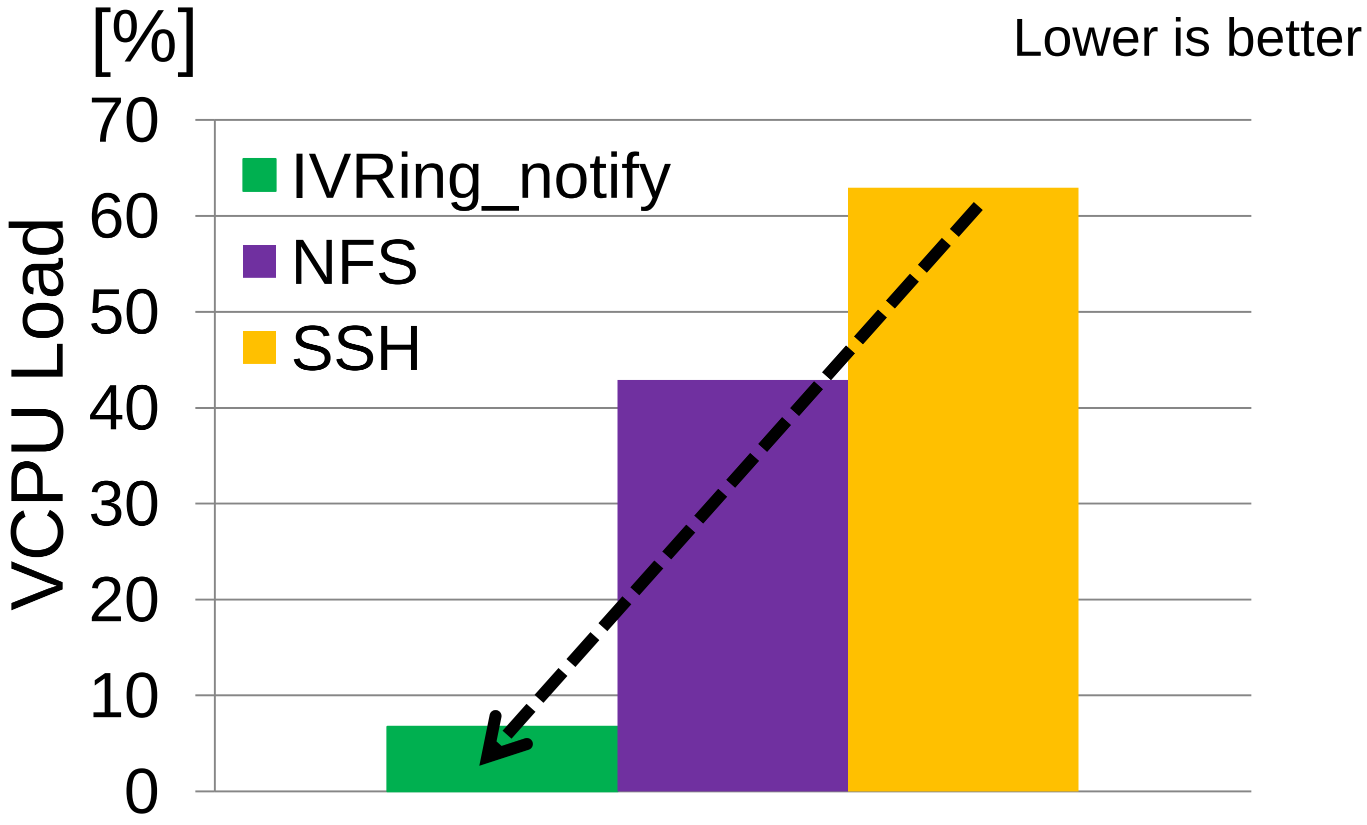


kernel:3.3.4-3,
1VCPU,
Memory:2GB

Kernel:3.3.1-5,
Intel Xeon x5660@
2.80GHz(6core),
Memory:50GB

Performance Comparison Result1

We compared 3 pattern based on the bare environment.
IVRing is much smaller load than NFS and SSH.



Copyright (c) 2012 Hitachi LTD., Yokohama Research Laboratory. All right reserved

Introducing

1. Purpose of a low-overhead ring-buffer in a virtualization system
2. “IVRing”, a low-overhead ring-buffer
3. IVRing VS general methods
4. How do we implement a ring-buffer?
5. Summary and future work

Present Problems – SMP

We implemented IVRing as a prototype,

so IVRing has the problem of scalability.

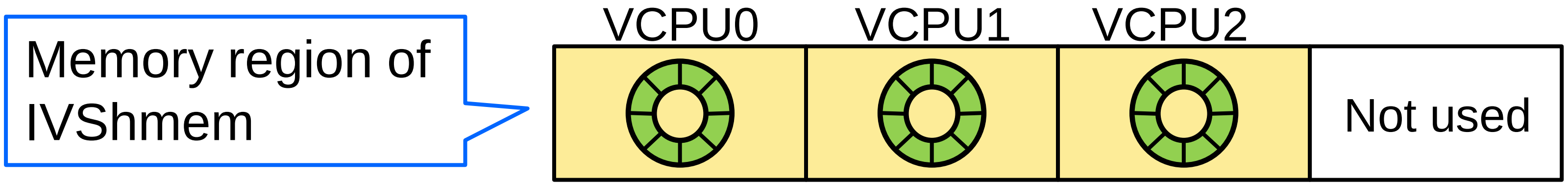
1. Multiple VCPU Support

- Spinlock ring-buffer is implemented to avoid competition.
- For scalability, a lockless ring-buffer is needed.

  ⇒ One VCPU requires one ring-buffer.
- Since IVShmem emulates a PCI device, the memory size is limited to power of two.

  ⇒ Unusable memory region remains on IVShmem.

  c.f. 3VCPUs are assigned to a guest.



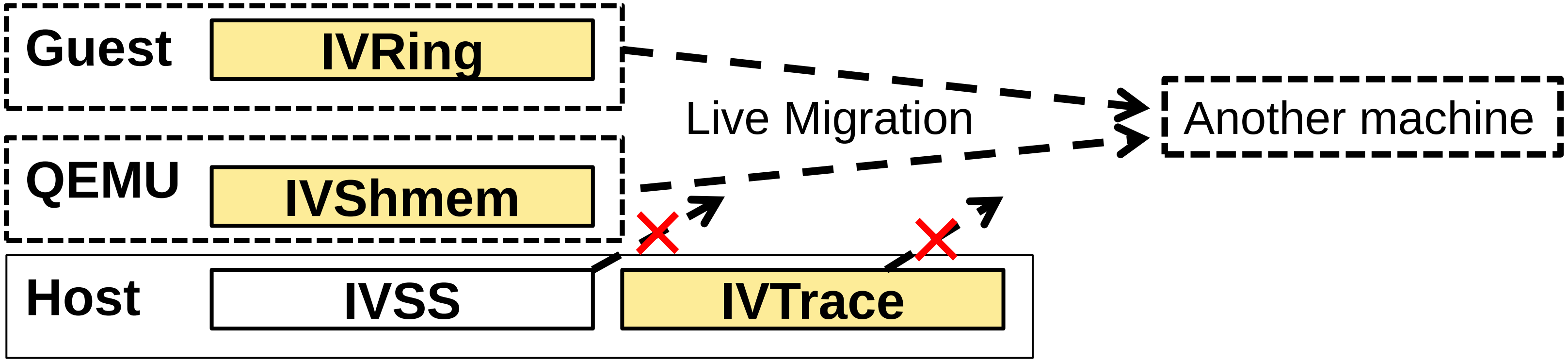
Present Problems – Live Migration

We implemented IVRing as a prototype,

so IVRing has the problem of scalability.

2. Live Migration Support

- Functions of IVSS related to eventfd
- I/F for Live Migration
- Operation of IVTrace in Live Migration
- Assigning of shared memory … etc



Summary And Future Work

<Summary>

- We implemented IVRing, a low-overhead ring-buffer, as a driver of IVShmem, and a reader of IVRing
- IVRing implemented as a prototype has some work to do.

<Future Work>

- To be useable in tracing system existing in-kernel
- To be useable in SMP environment
- To design for Live Migration
- To implement a new virtual device for tracing

A1. How to Use IVRing

1. Run IVShmem_server on a host

assign an UNIX socket path(PATH), a shmem object, and shmem size(SIZE)

2. Boot a QEMU and a guest with following options

- device ivshmem,size=<SIZE>,chardev=ivshmem

- chardev socket,path=<PATH>,id=ivshmem

3. Run reader on the host

assign file name, file size, log#, and PATH

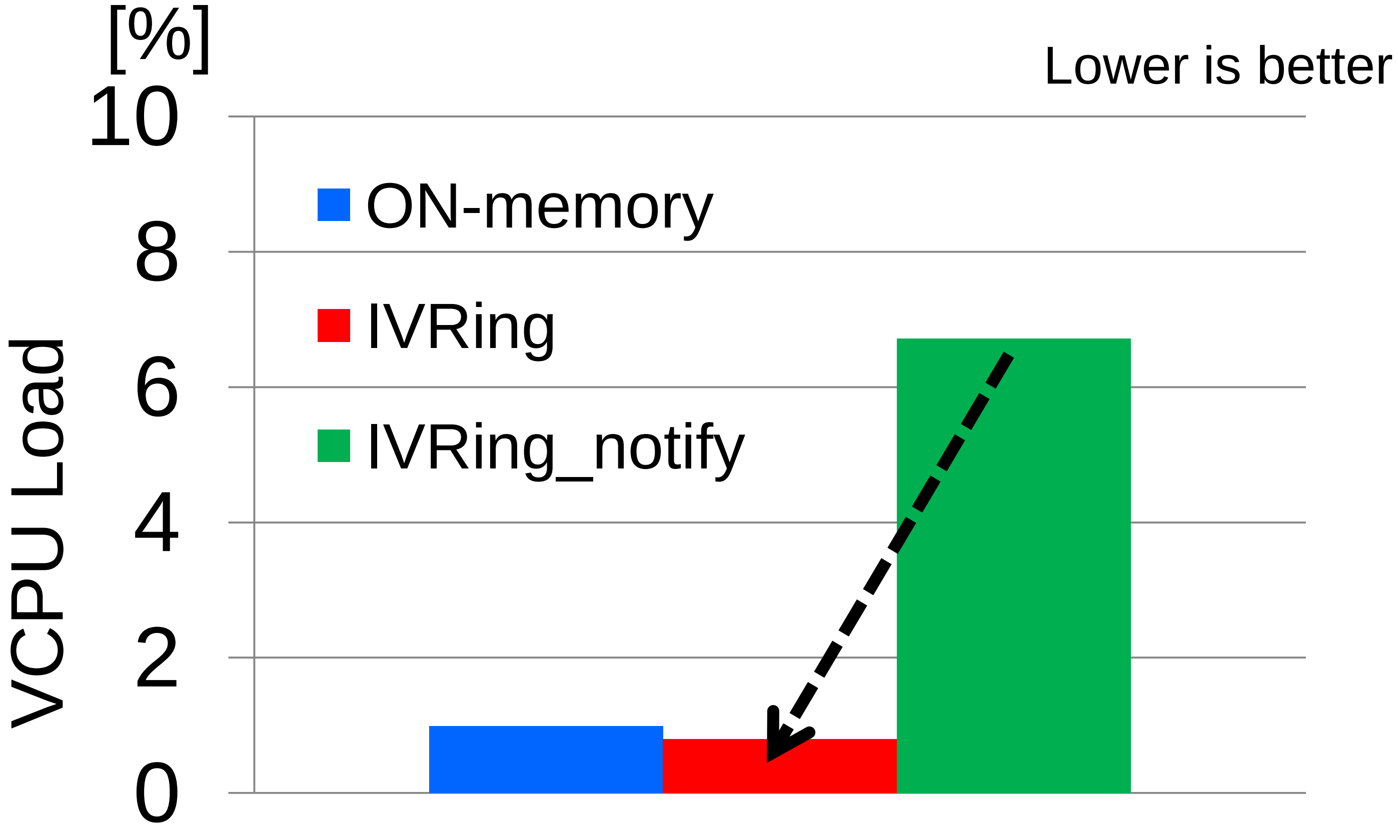
4. Load writer module on the guest

5. Run a SystemTap script on the guest

use ivring_write(), which is an API of IVRing

A2. Performance Comparison Result

By stopping notification, which causes VM-EXIT, load of IVRing gets close to that of ON-memory.
⇒ Need to decide notification times as a future work.



HITACHI

Inspire the Next

Legal statements

- 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.