ivtrace: Tracing Across Host OS and Guest OSs

Jun 7th at LinuxCon Japan 2012

Akihiro Nagai
Linux Technology Center
Yokohama Research Lab
Hitachi Ltd,

HITACHI
Inspire the Next

Introduce myself

- I’m working at Linux Technology Center of Yokohama Research Lab in Hitachi Ltd.,

- I’m interested in
  - Tracing Technology
  - Automated software testing
  - Performance analysis
  - Development toolchains ...etc

Agenda

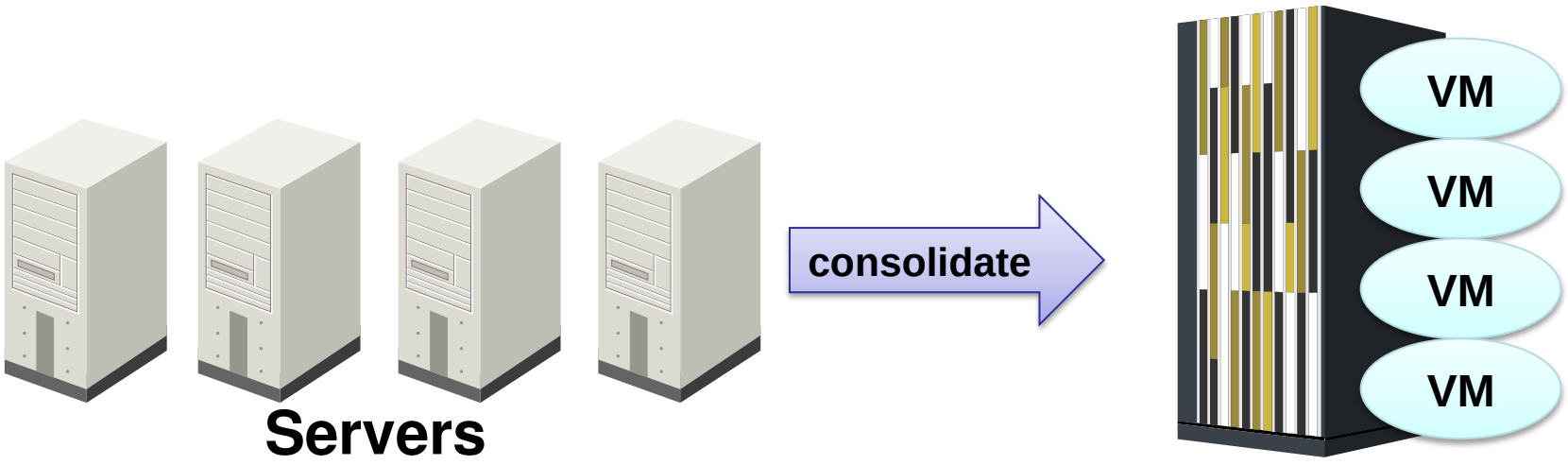
1. Background
2. Requirements of Tracing for Virtualized System
3. ivring System Overview
4. Evaluation
5. Conclusion

Agenda

1. Background
2. Requirements of Tracing for Virtualized System
3. ivring System Overview
4. Evaluation
5. Conclusion

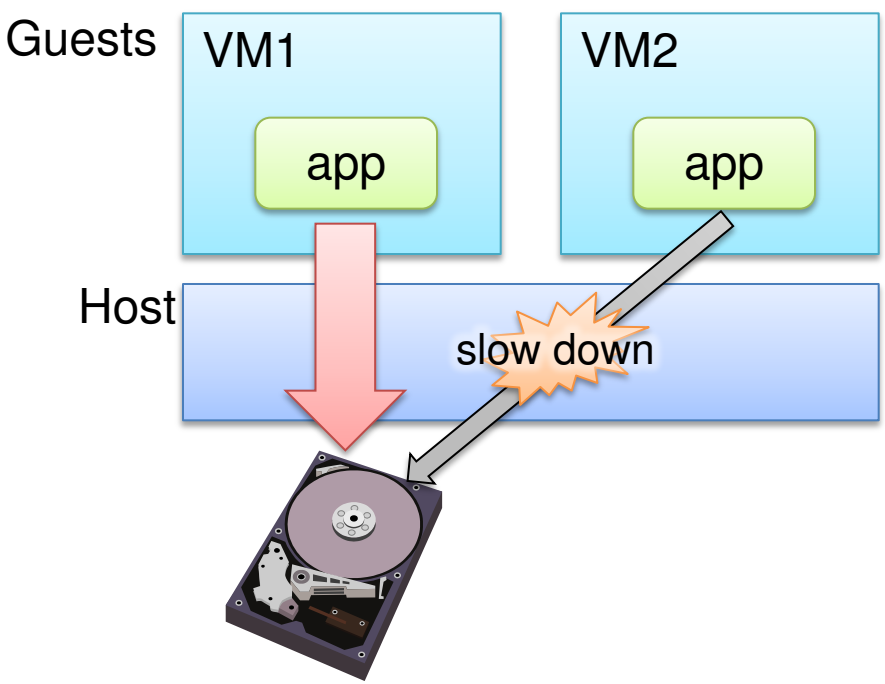
1 Background

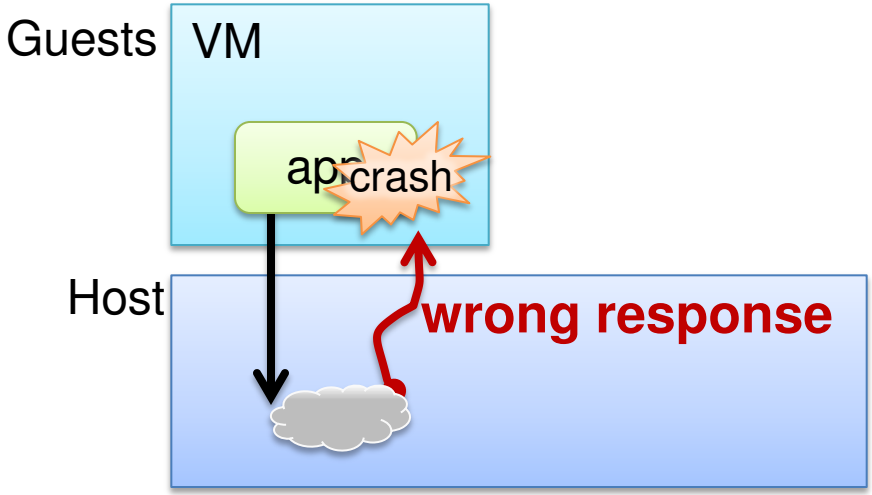
- Recently, enterprise systems are often built on cloud systems that use virtualization technology and aim for system consolidation.



1.1 Problems in Virtualized System

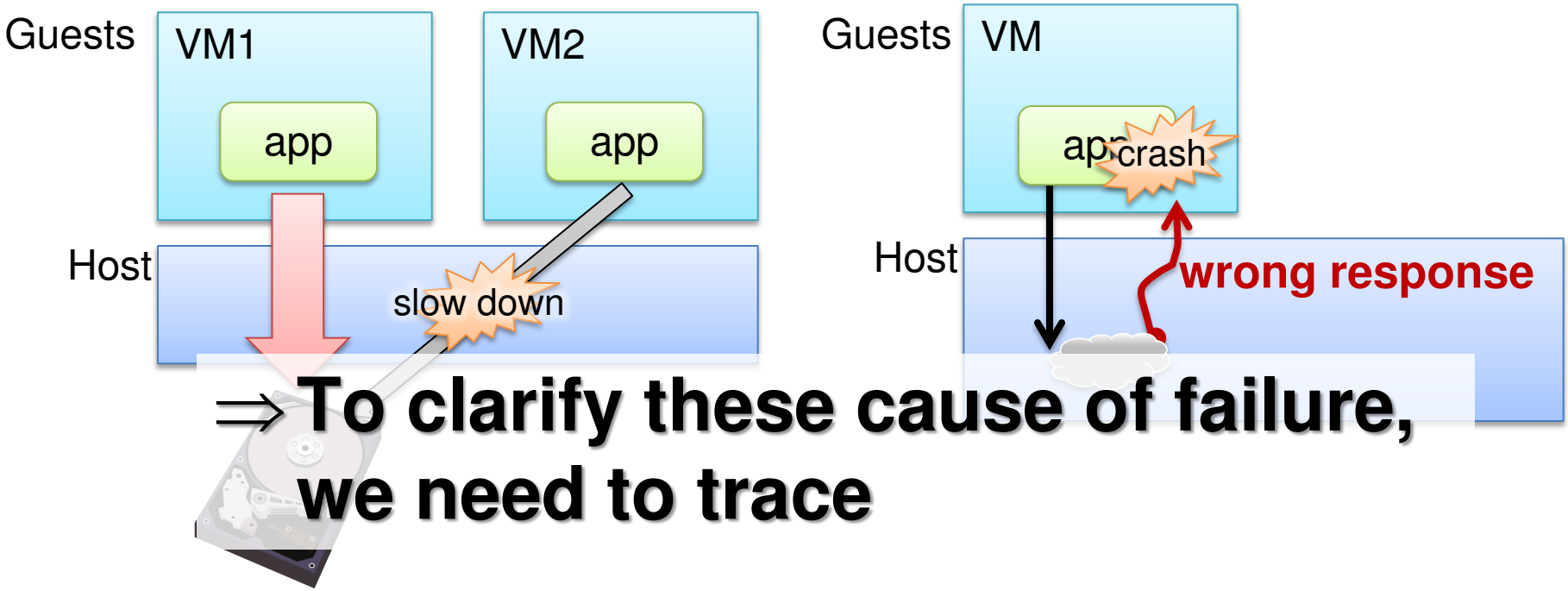
- A VM's behavior can affect other VMs' behavior
- A VM can die by host OS or hypervisor's bug





1.1 Problems in Virtualized System

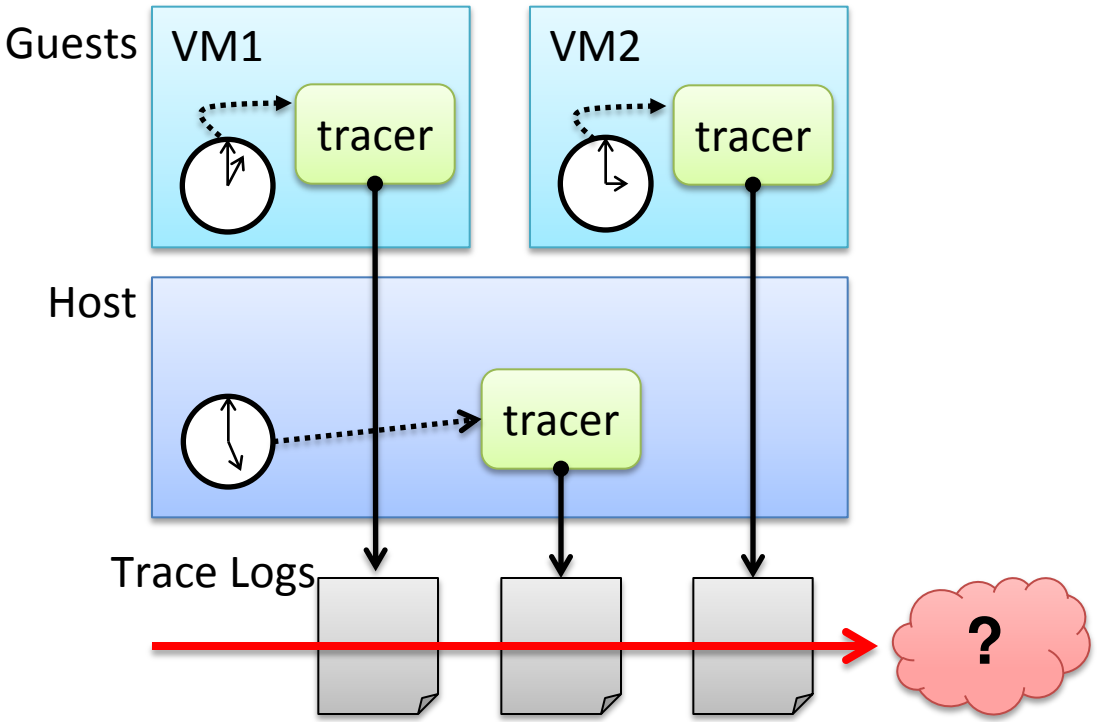
- A VM's behavior can affect other VMs' behavior
- A VM can die by host OS or hypervisor's bug



⇒ **To clarify these cause of failure, we need to trace**

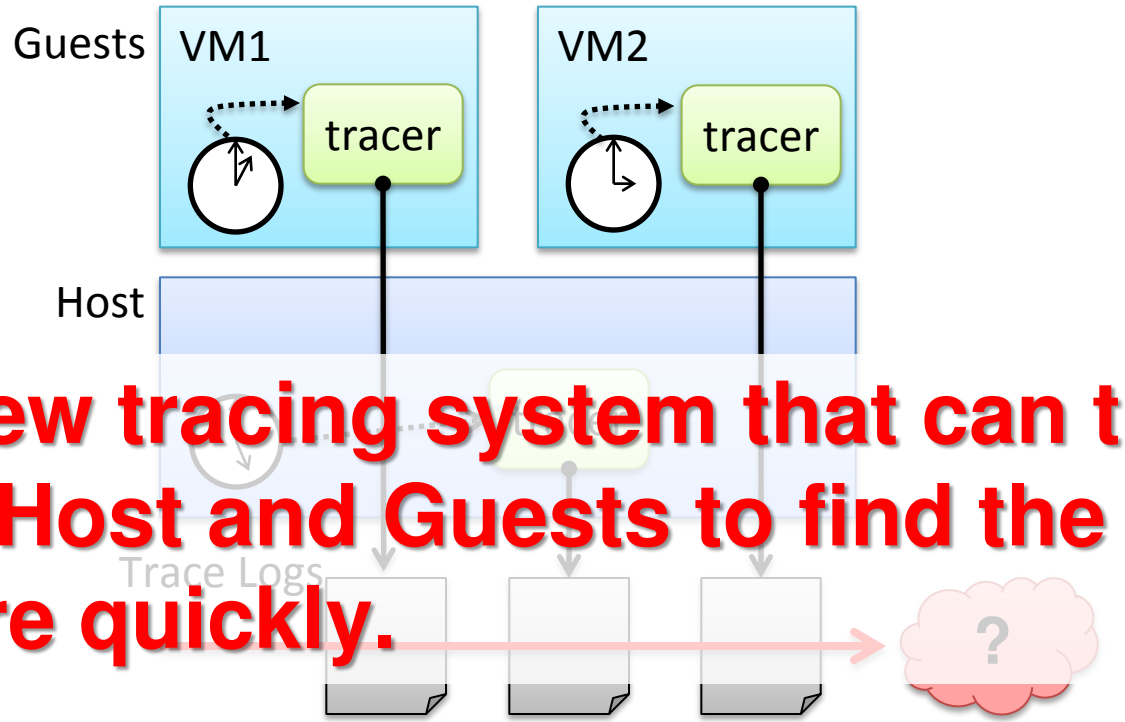
1.2 Problems in Tracing

- Get trace logs to find out the cause of failures, it has following problems
  - Each VM has own time-stamp-counter
  - Too huge trace logs (x VMs) to analyze them



1.2 Problems in Tracing

- Get trace logs to find out the cause of failures, it has following problems
  - Each VM has own time-stamp-counter
  - Too huge trace logs (x VMs) to analyze them



Need new tracing system that can trace across Host and Guests to find the cause of failure quickly.

Agenda

1. Background
2. Requirements of Tracing for Virtualized System
3. Tracing System Overview
4. Evaluation
5. Conclusion

2 Requirements

- Adjusting host OS's time-stamp-counter and Guests' one
  - Need it to merge trace logs
- Visualization
  - Find out the cause of failure from huge trace logs quickly

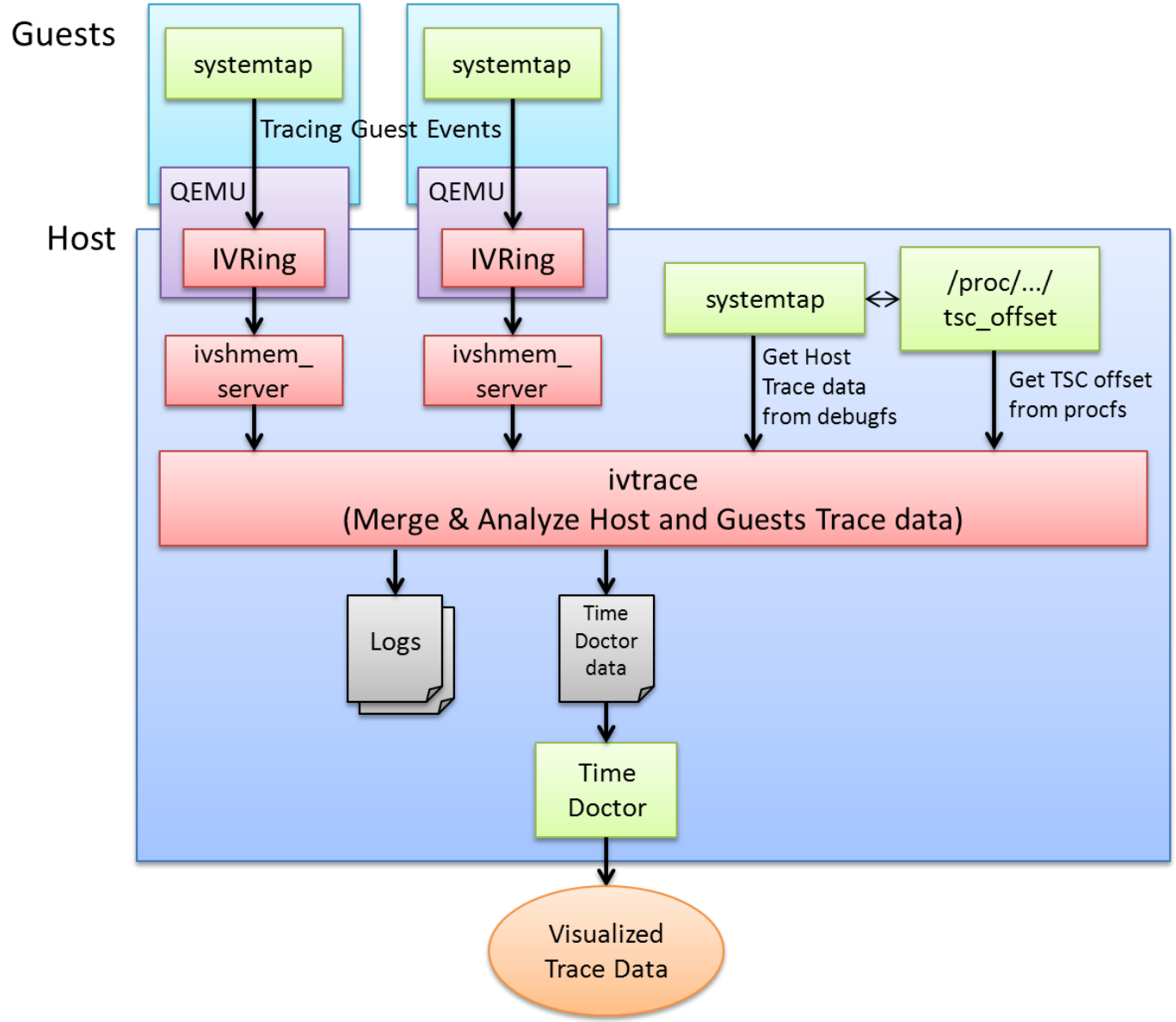
2.1 What we use

- Use **systemtap** as a tracer
  - Trace both of kernel and user applications
- Adjusting guests' time-stamp-counter(tsc) in host OS using **tsc_offset**
- 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

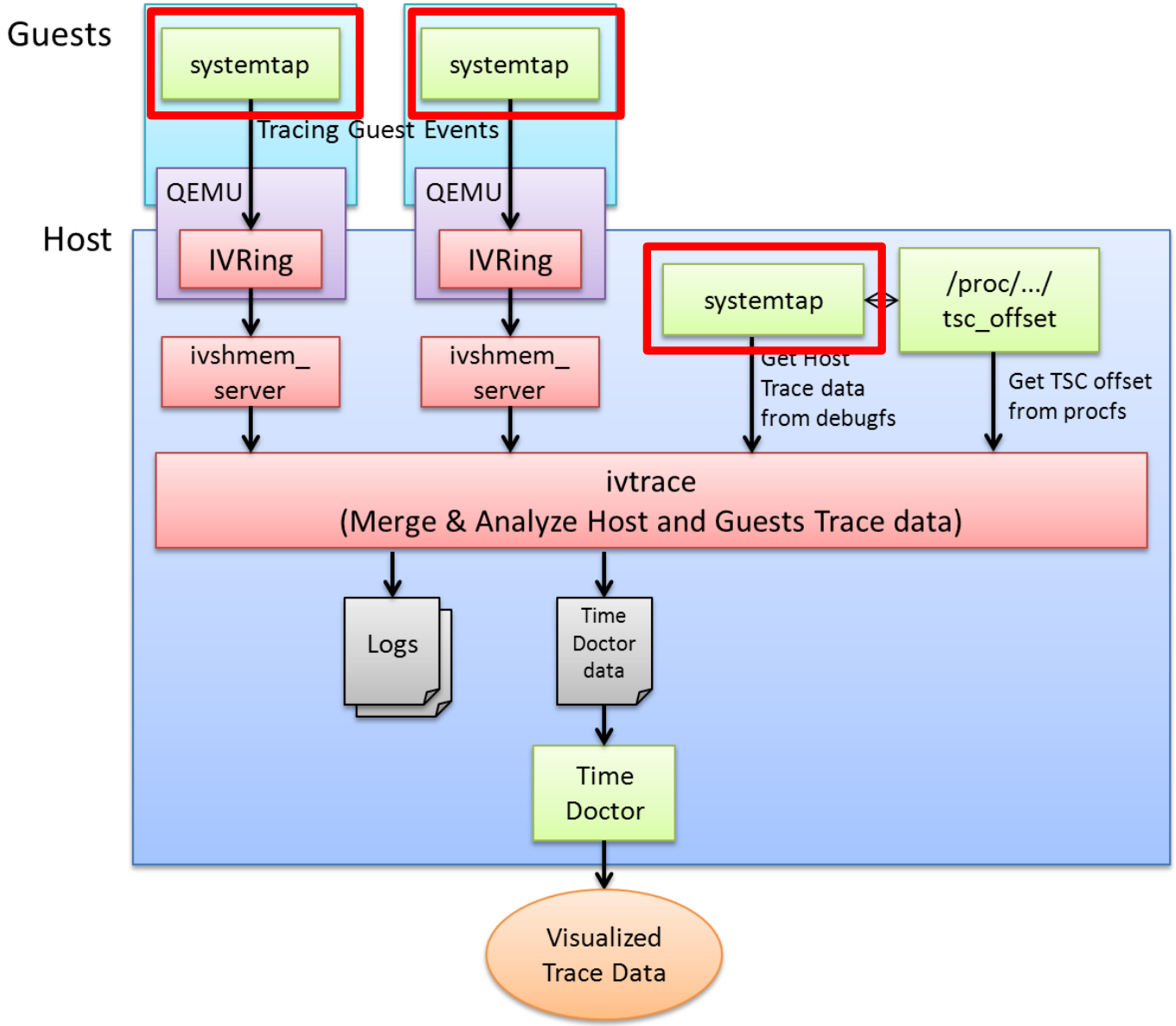
Agenda

1. Background
2. Requirements of Tracing for Virtualized System
3. ivring System Overview
4. Evaluation
5. Conclusion

3 System Overview



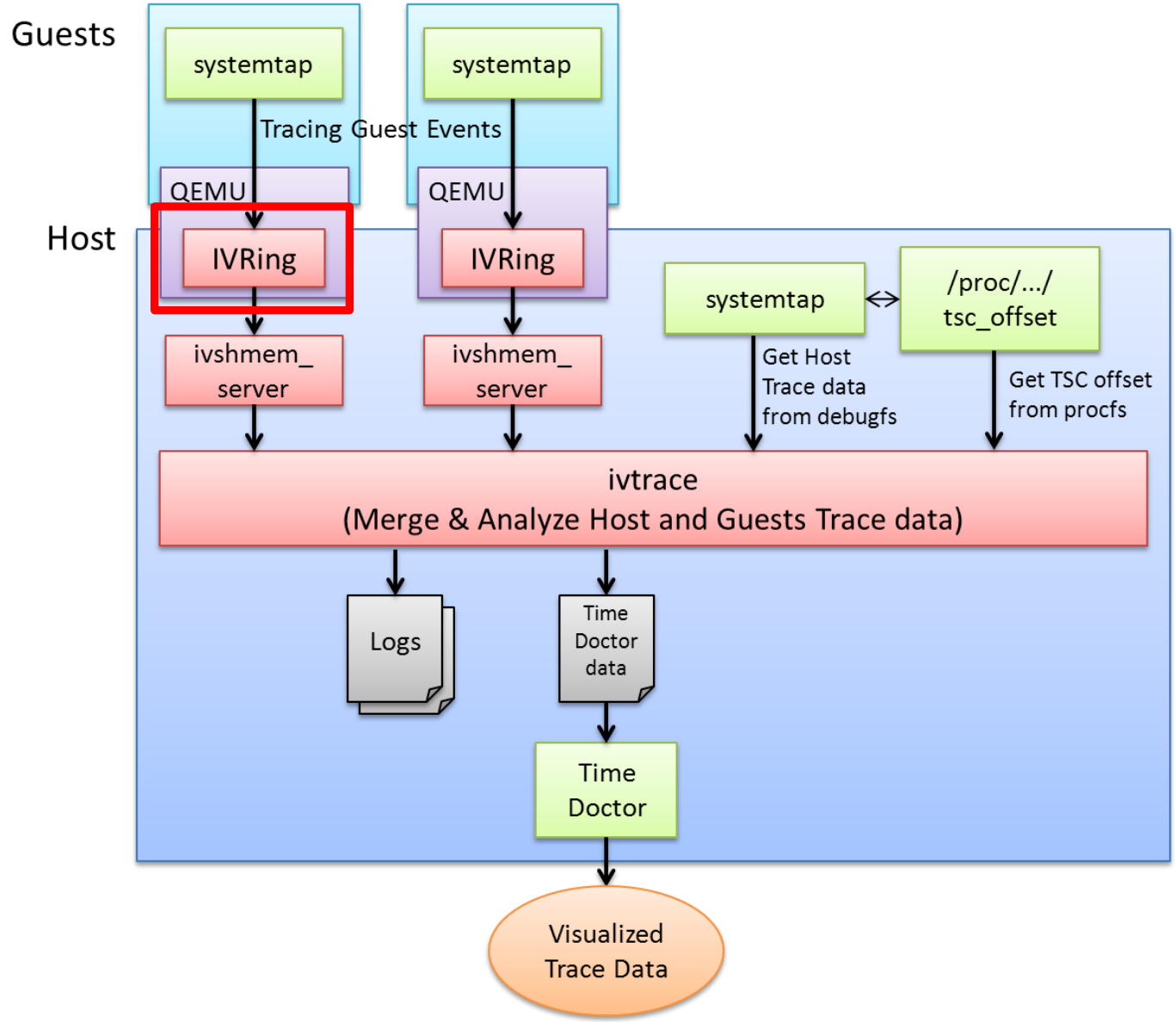
3.1 Systemtap



3.1 Systemtap

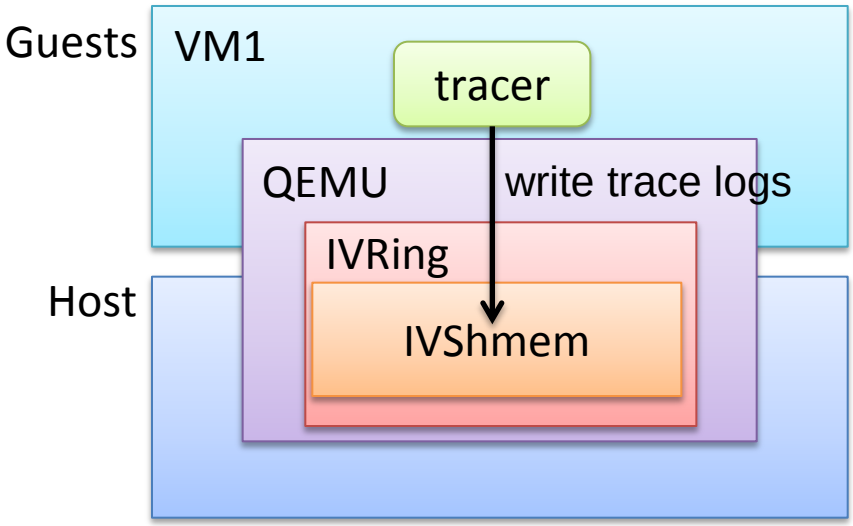
- Systemtap is a Tracer
- Possible to trace both of kernel and user-space applications
- Programmable tracing by stap-script
- Dynamic probing
- Easy to install in Fedora/Cent/RHEL

3.2 IVRing



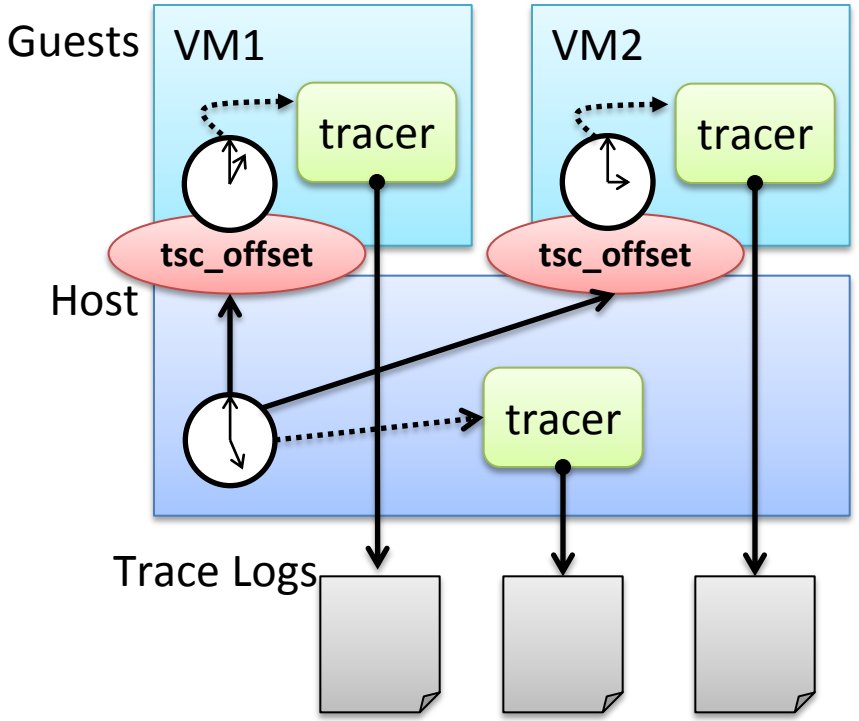
3.2 IVRing

- IVRing is a Ring Buffer based on QEMU's IVShmem (Inter-VM Shared memory)
- Copyless & Low overhead data communication between VMs and host
- ivtrace uses it to pass the trace logs from guests to host
- Implementation details are presented in the next session



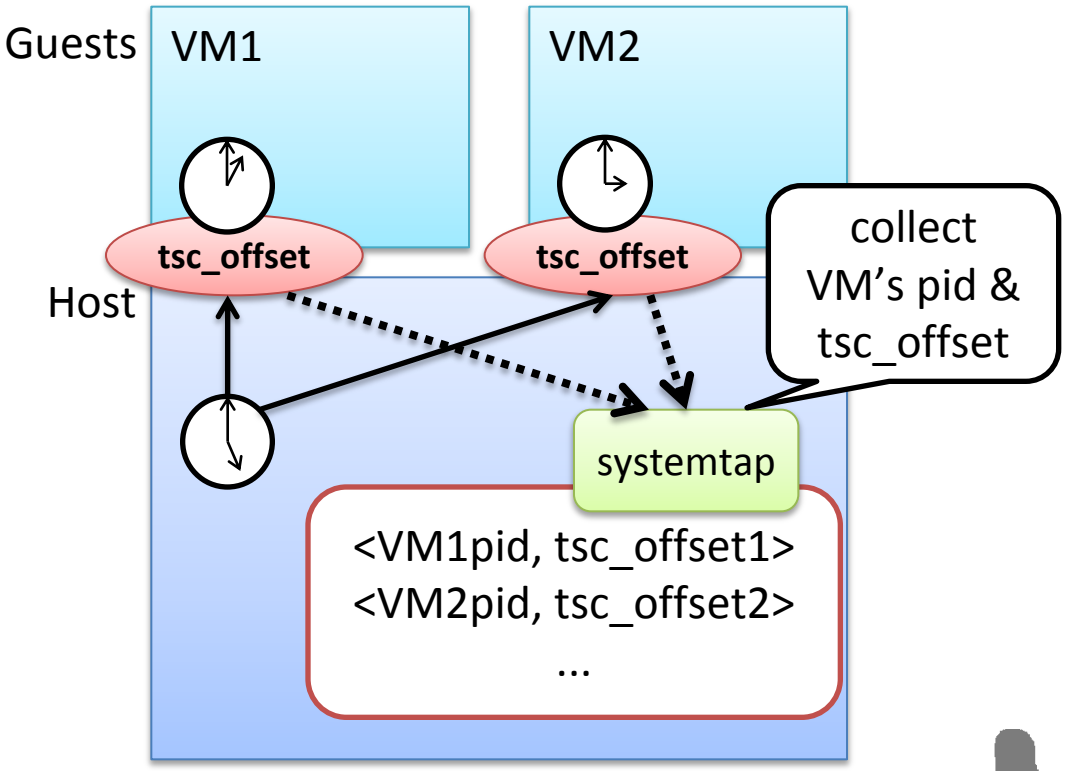
3.3 Adjusting tsc (1/3)

- 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, so we need to fix it using **tsc_offset** to merge logs.

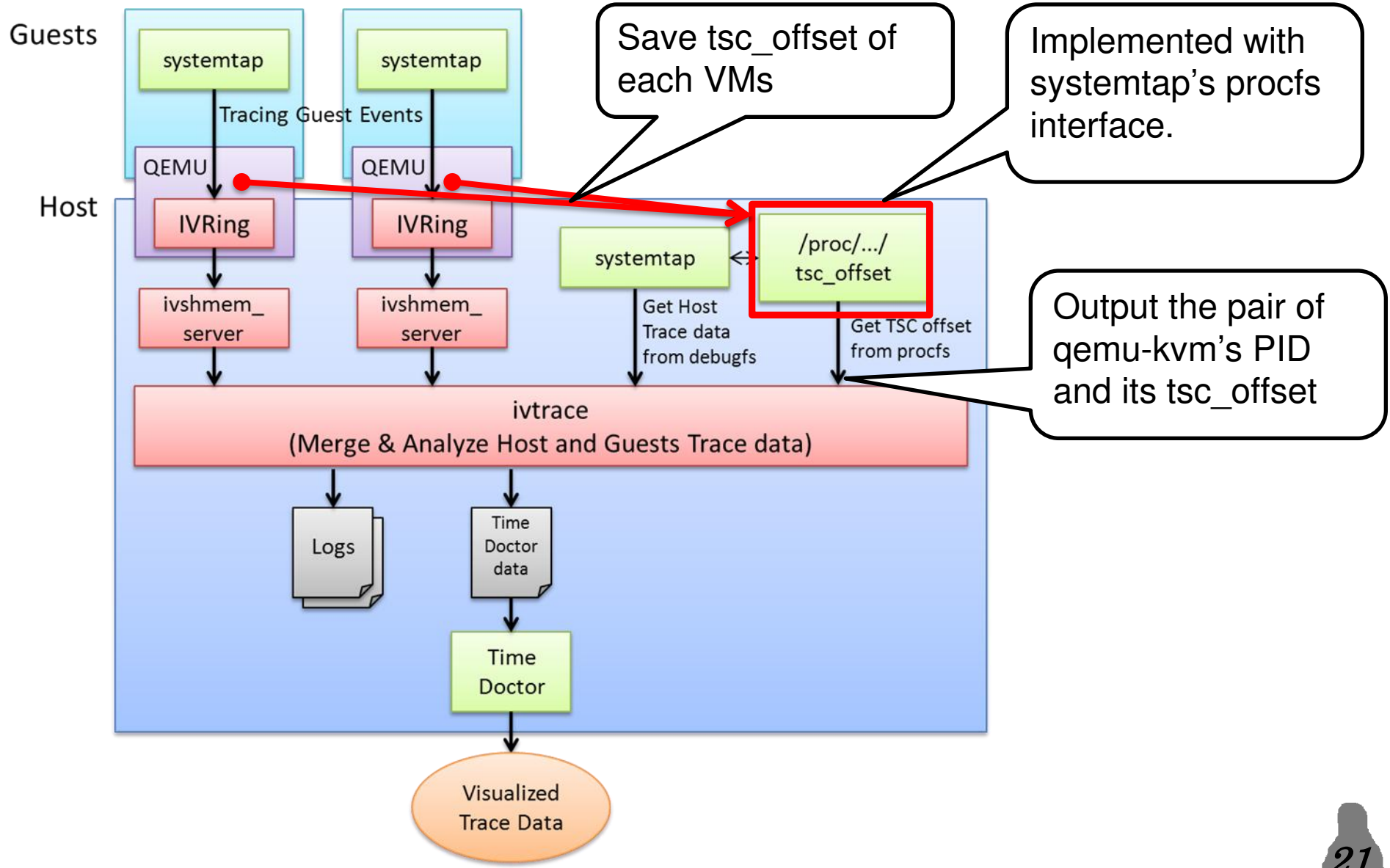


3.3 Adjusting tsc (2/3)

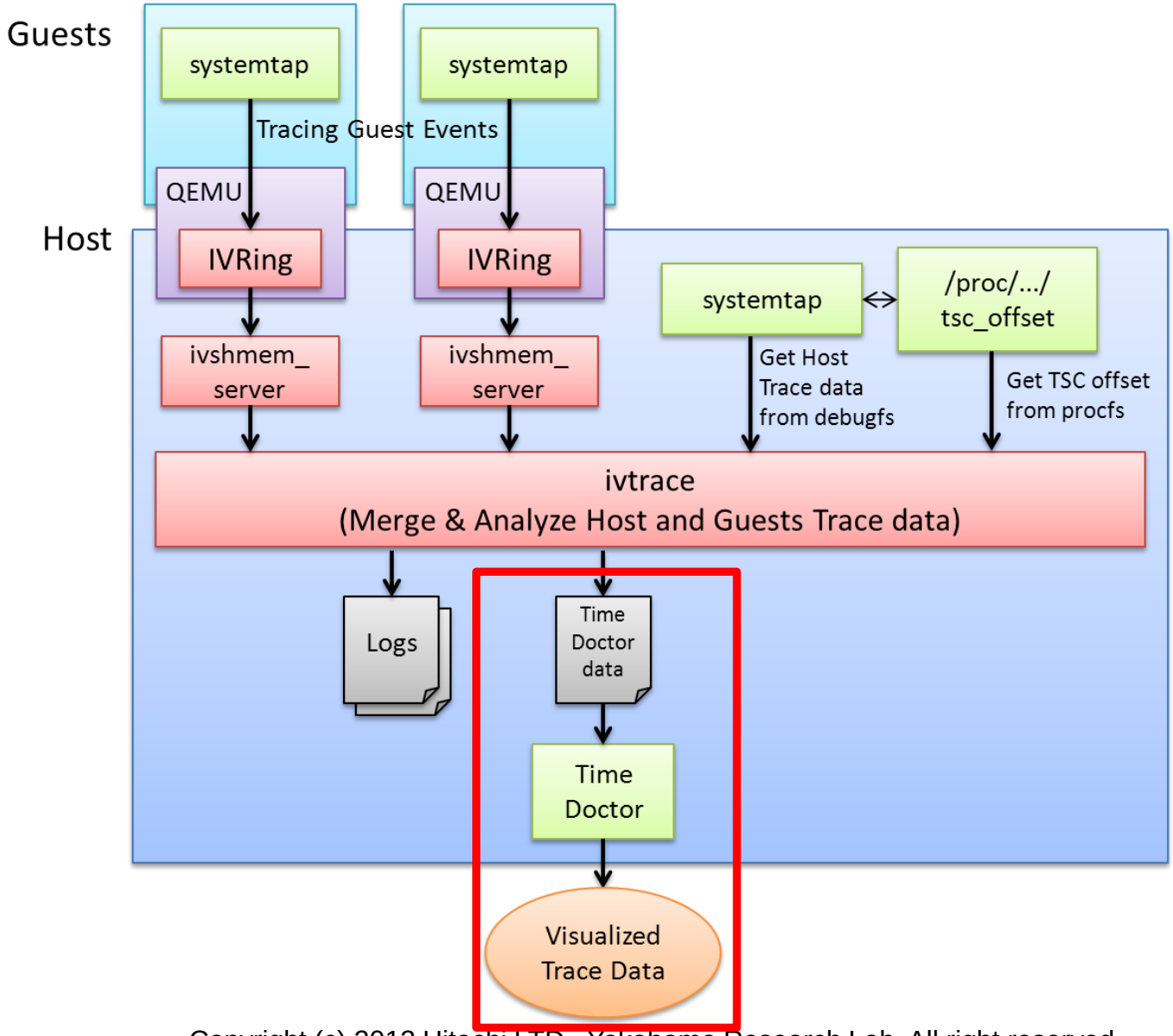
- **tsc_offset** are available with kernel function **`vmcs_read()`** in host.
- ivtrace records the tsc_offset as the pair of <qemuPID, tsc_offset>
  - Each qemu-kvm has one tsc_offset
- At vm_exit, ivtrace gets tsc_offset using systemtap probing.



3.3 Adjusting tsc (3/3)

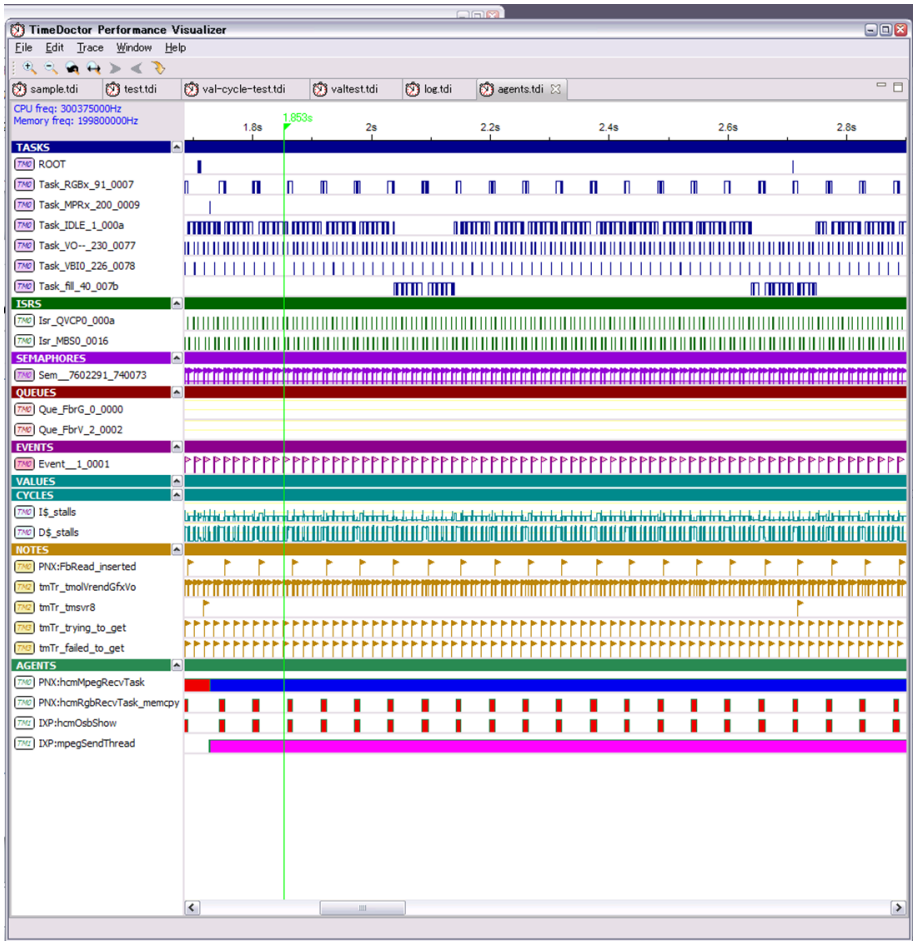


3.4 Visualization



3.4 Visualization

- Use TimeDoctor
  - Visualizing tool specialized for trace data
- Input text file with simple syntax
- Multi Platform
  - Eclipse plugin

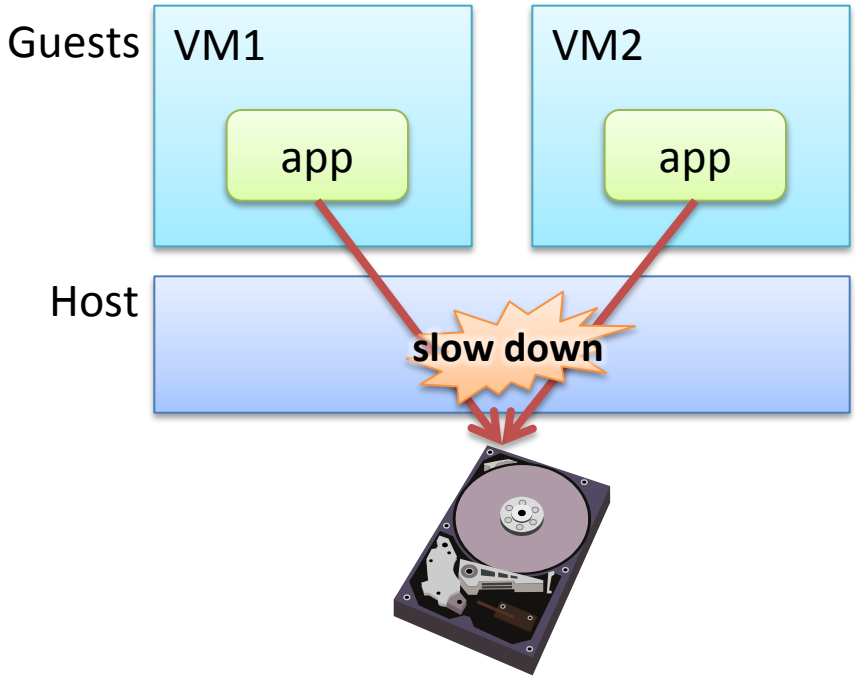


Agenda

1. Background
2. Requirements of Tracing for Virtualized System
3. ivring System Overview
4. Evaluation
5. Conclusion

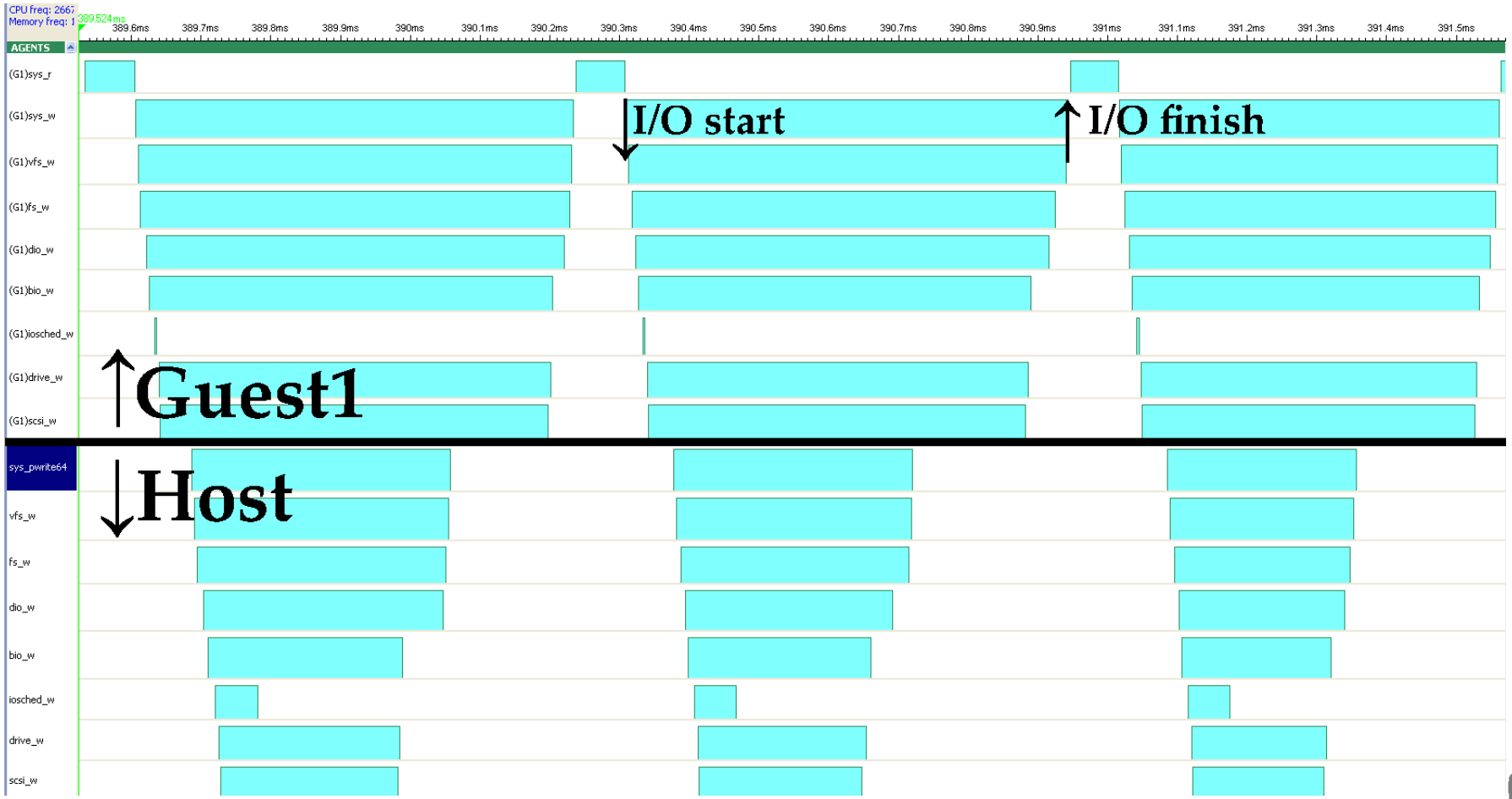
4 Evaluation

- Clarify the cause of performance degradation by disk access contentions using ivtrace



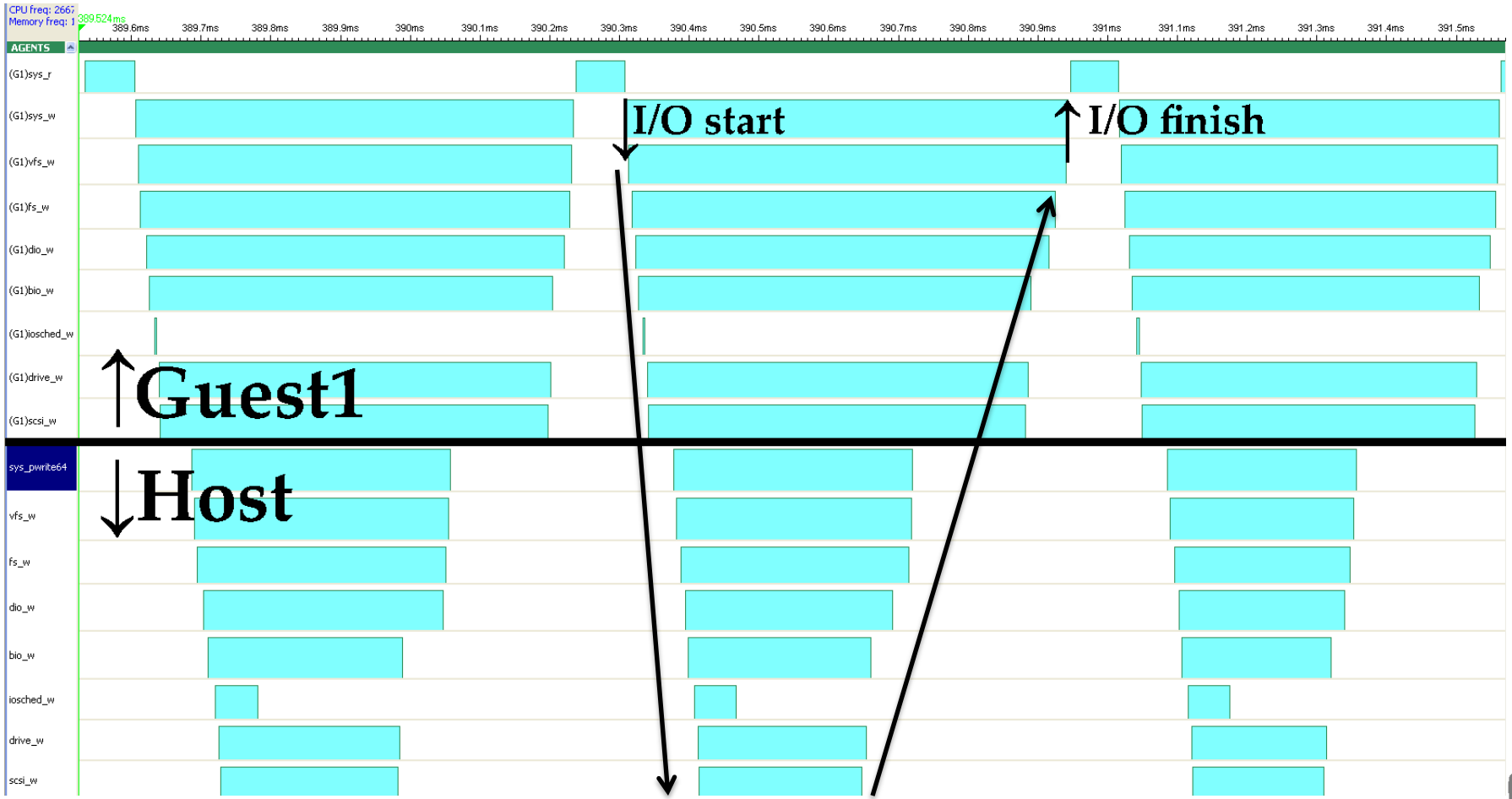
4.1 Result

- The case of no contentions



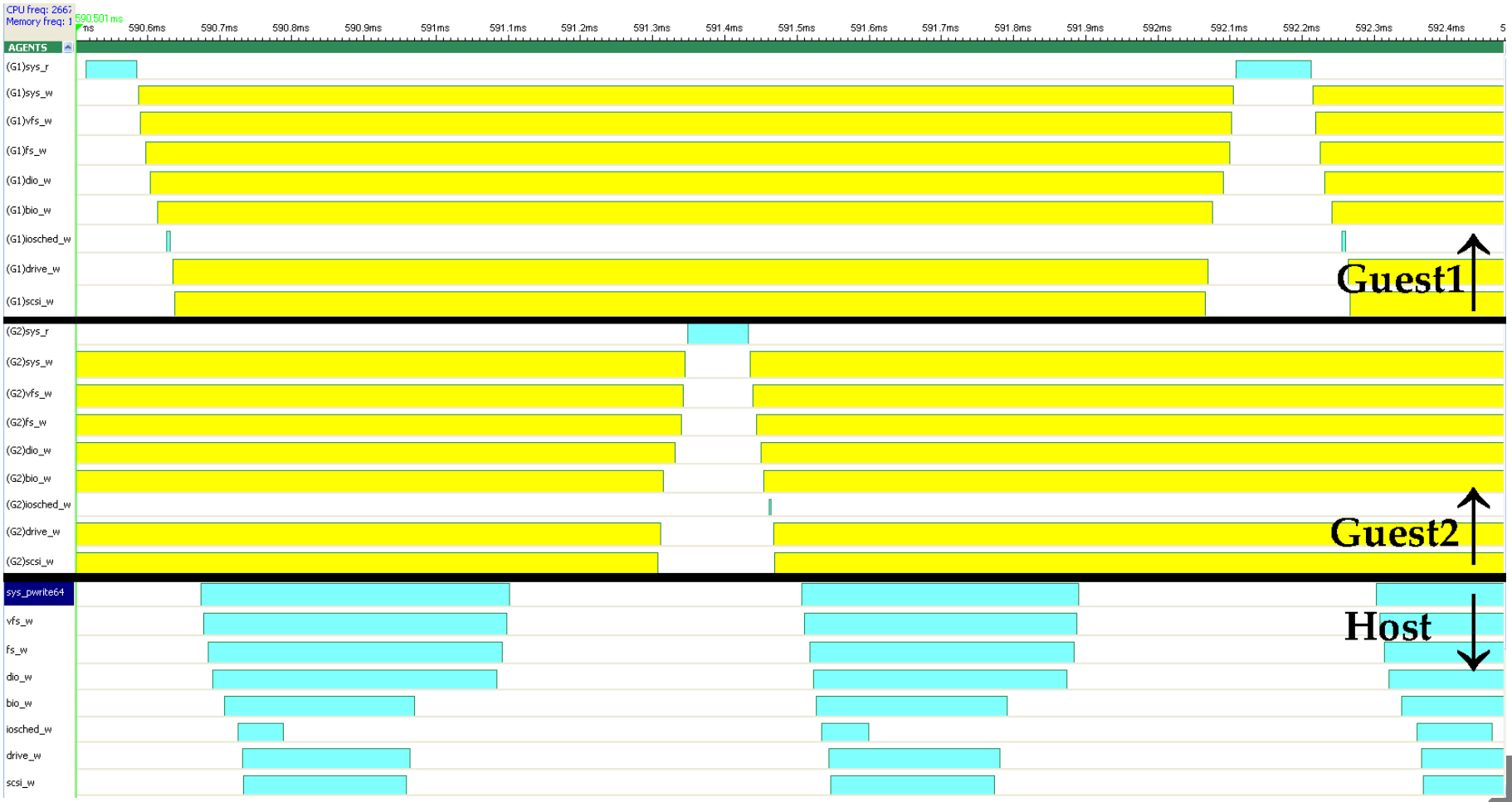
4.1 Result

- The case of no contentions



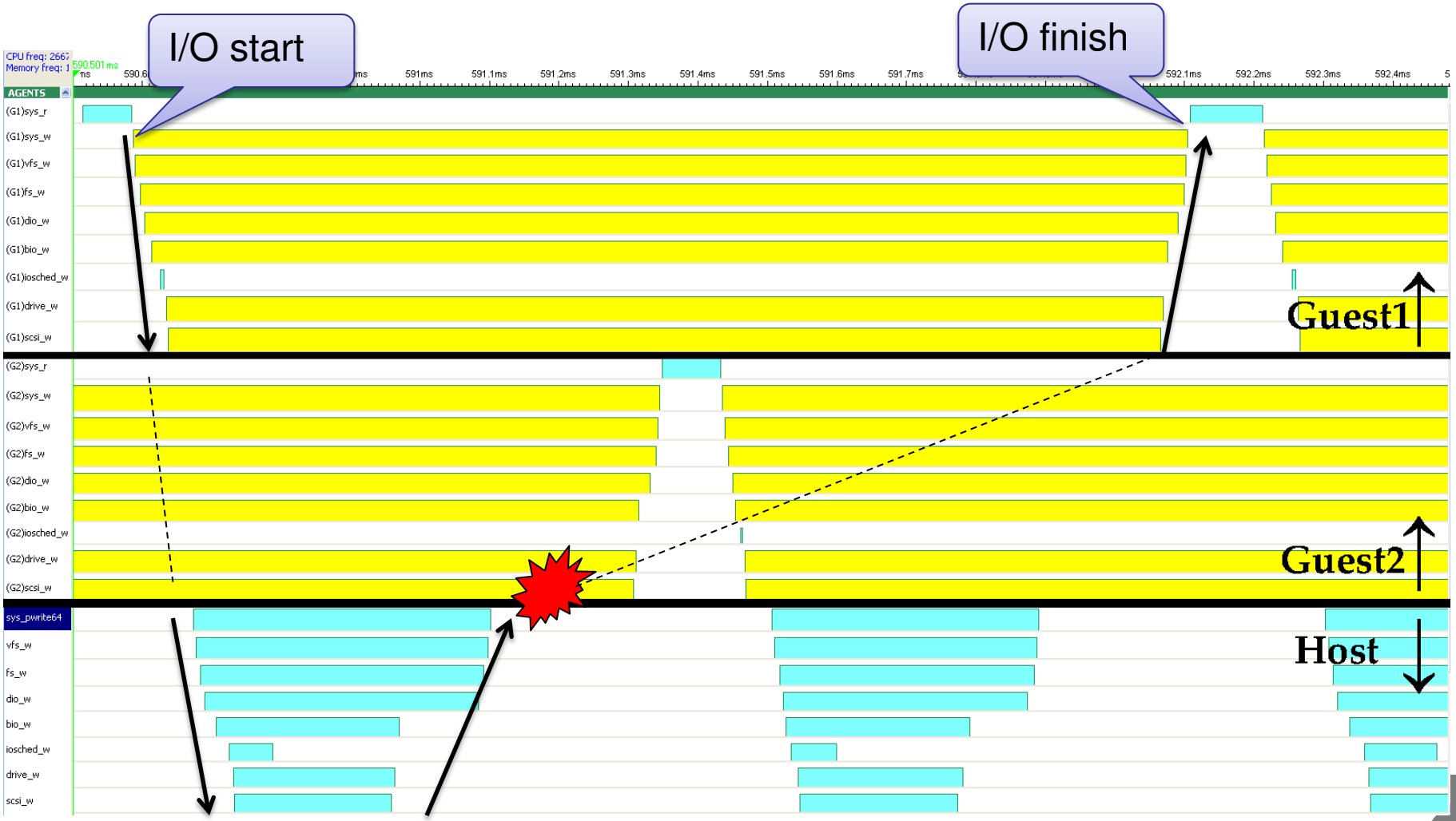
4.2 Result2

- The case of contentions. I/O slowdown



4.2 Result2

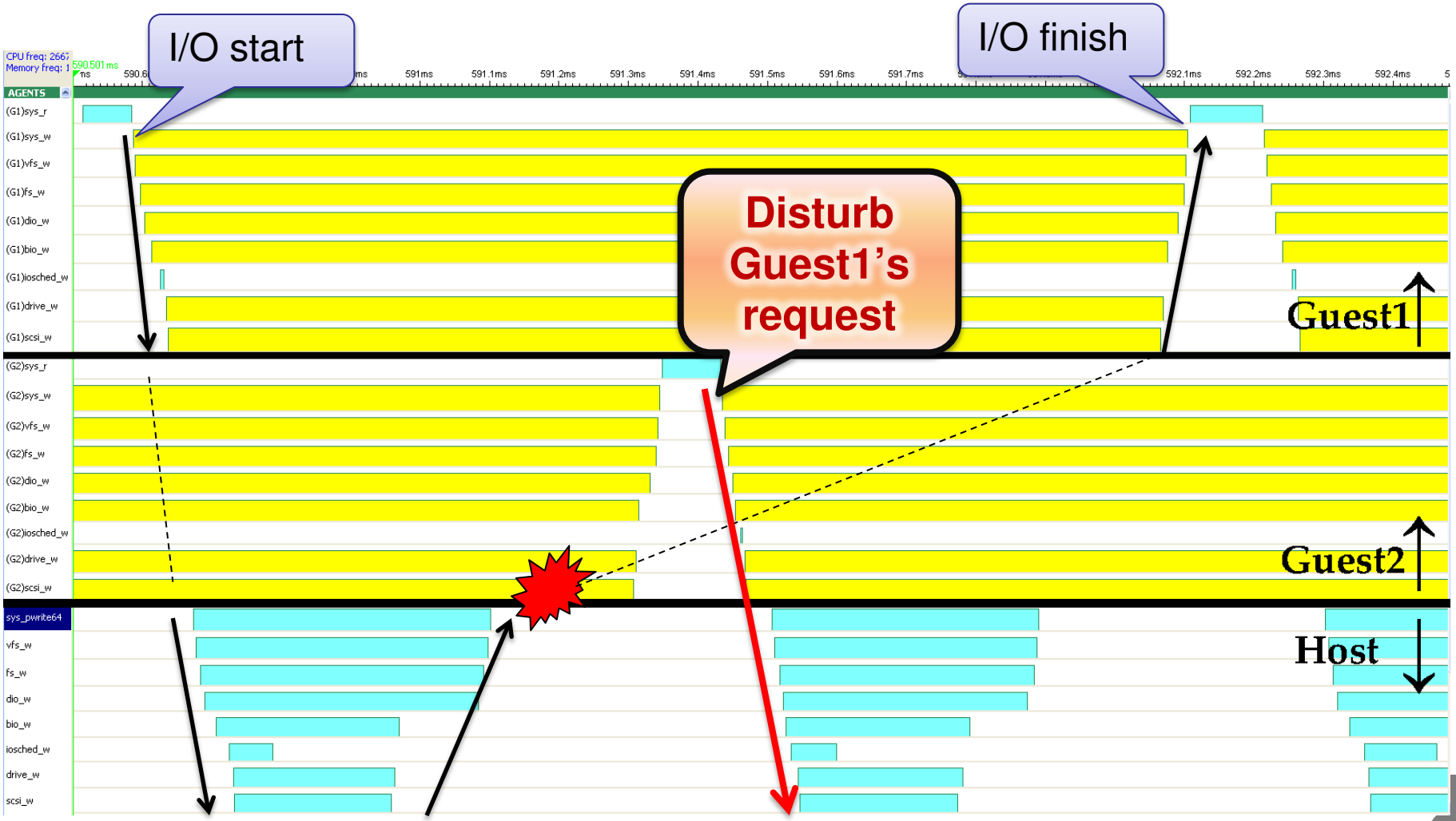
- The case of contentions. I/O slowdown



4.2 Result2

Linux Technology Center

- The case of contentions. I/O slowdown



Agenda

1. Background
2. Requirements of Tracing for Virtualized System
3. ivring System Overview
4. Evaluation
5. Conclusion

5 Conclusion

- The cause of system failure are getting difficult to find by using virtualization technology. e.g. cloud system.
- Need new tracing system which is possible to trace across host and guests.
- This session suggests new tracing system “ivtrace”: using systemtap, IVRing and TimeDoctor
- “ivtrace” can clarify the delay of disk I/O requests

5.1 Future Works

- Support other failure patterns
  - Need the framework to add new patterns
- Support live migration
  - Need to apply to real cloud systems
- Support CPU hot-plug

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.