Kemari: Virtual Machine Synchronization for Fault Tolerance using DomT

Yoshi Tamura
NTT Cyber Space Labs.
tamura.yoshiaki@lab.ntt.co.jp

2008/6/24

Outline

- Our goal
- Design

} Brush up on Xen Summit 2007

- Architecture overview
- Implementation
- Evaluation
- Conclusion

What is Kemari?

蹴鞠

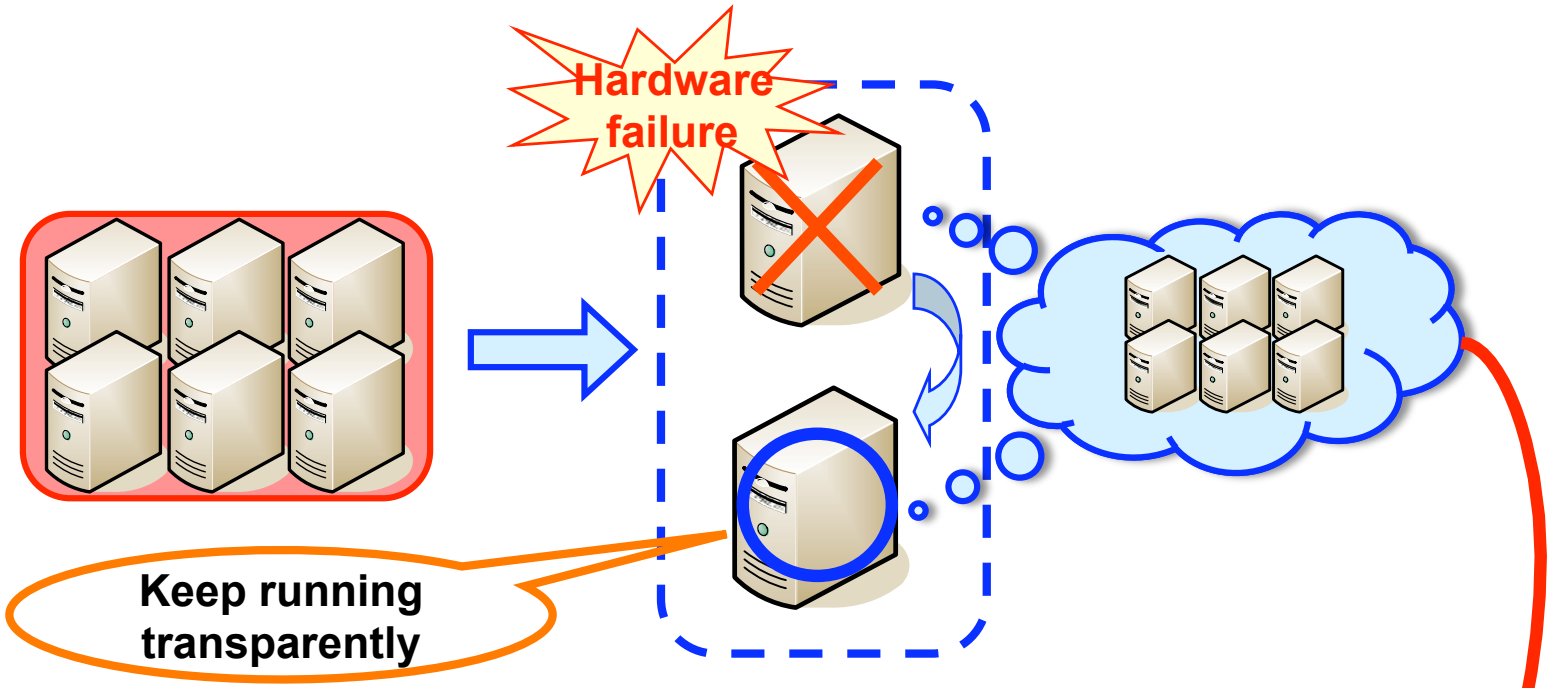
(Kemari)

- Kemari is a football game that players keep a ball in the air

Don’t drop the ball!

Our goal

Don’t drop the ball! Don’t drop the VMs!

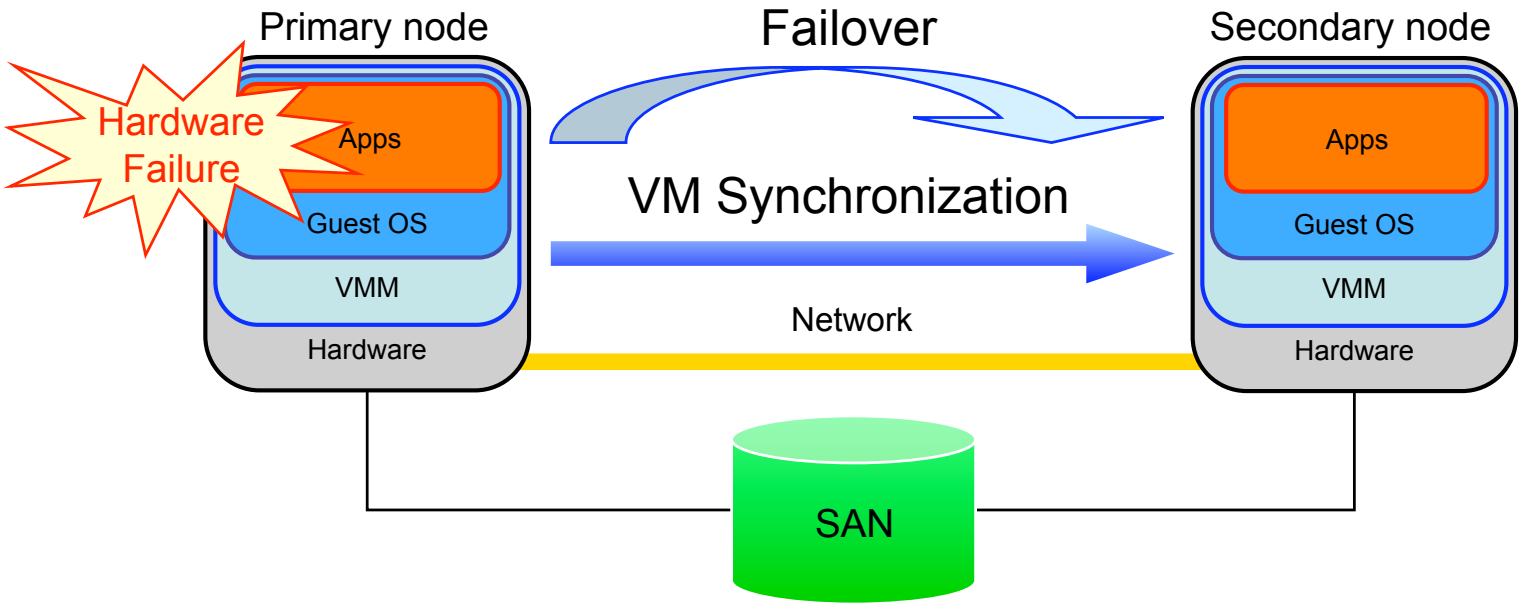


Kemari: Virtual Machine Synchronization

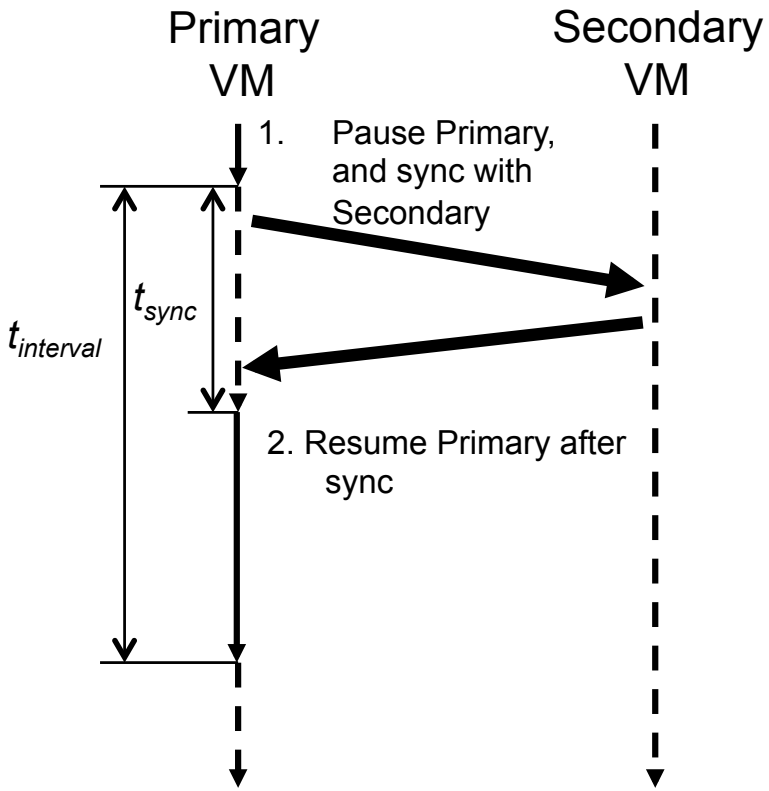
What needs to be done?

- Virtual Machine Synchronization
  - Primary VM and Secondary VM must be identical
- Detection of failure
- Failover mechanism

Extension of existing techniques



How to synchronize VMs?

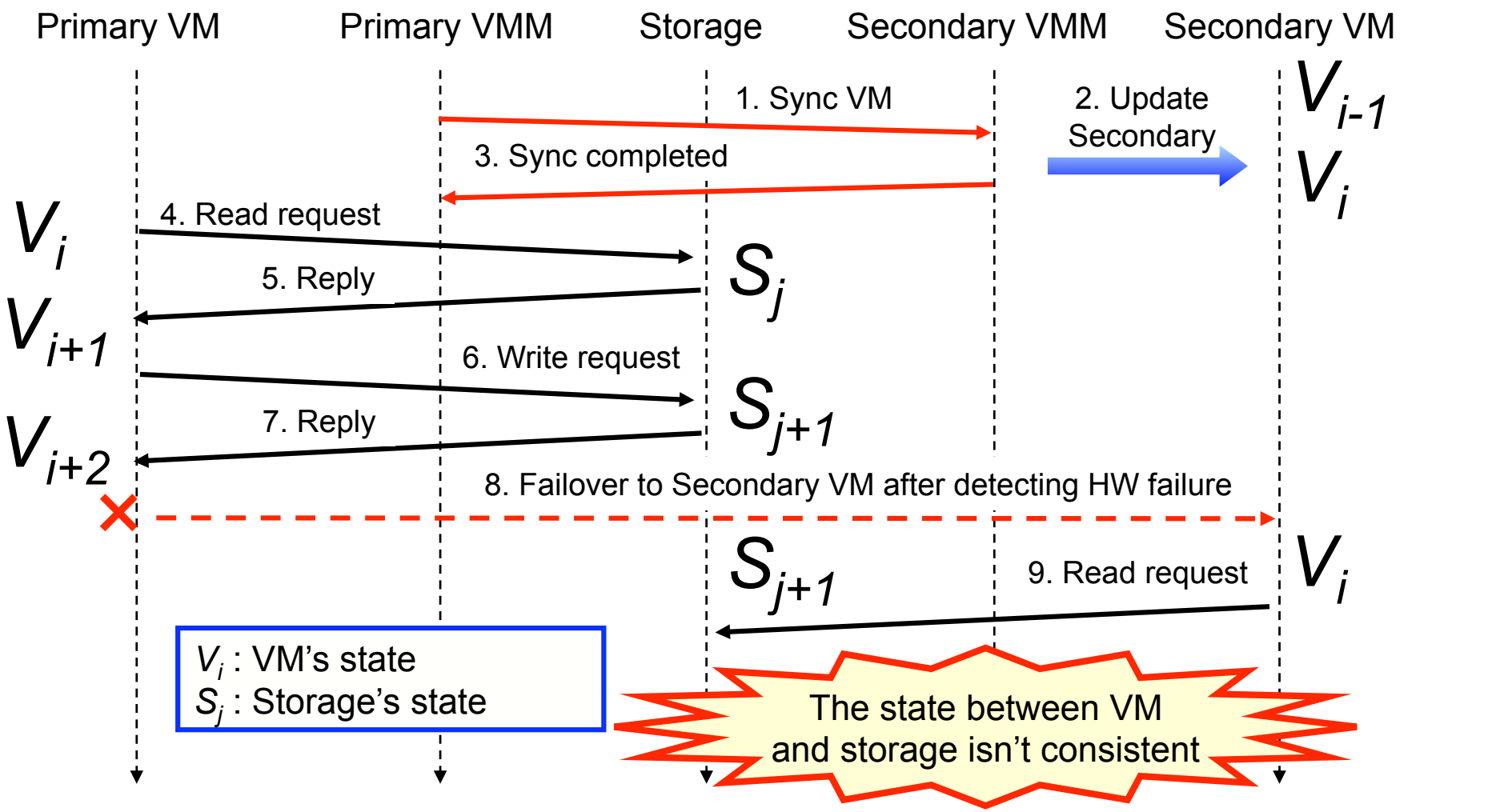


- Need to make the overhead of sync smaller
  - Make sync time shorter
    - ⇨ Only transfer updated data
  - Sync VMs less often
    - ⇨ Secondary must be able to continue transparently

⇩

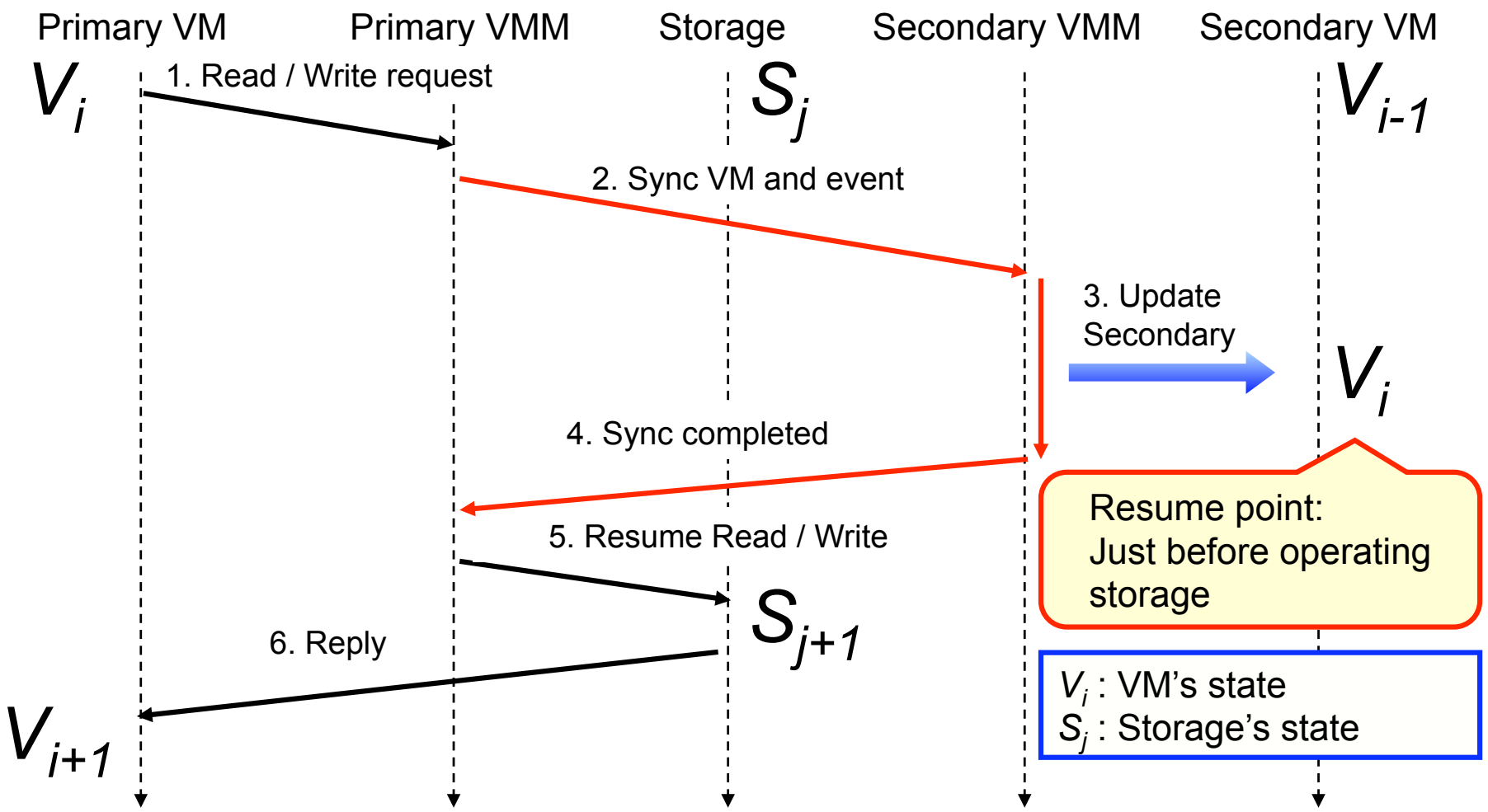
- Sync VMs before sending or receiving Events
  - Events: Storage, network, console

What happens if synced on specific intervals?



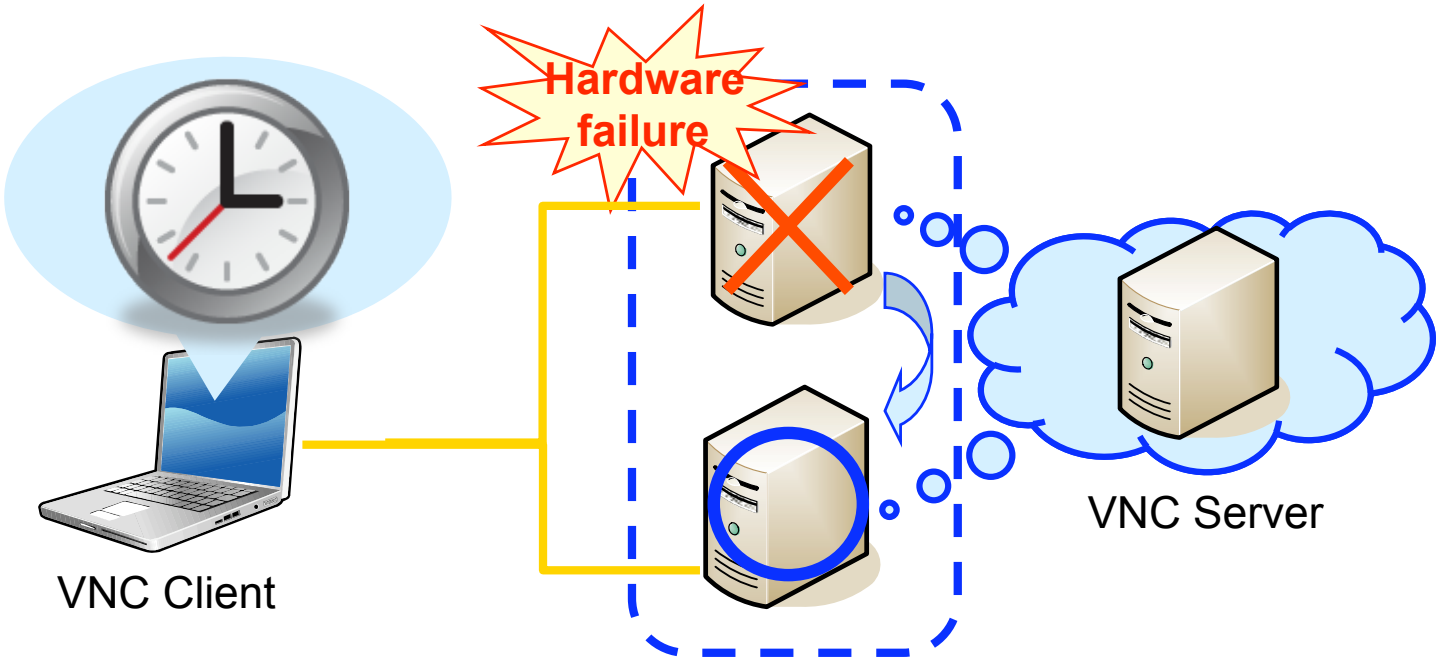
- Secondary VM won't be able to continue transparently

Sync on events from VM to storage



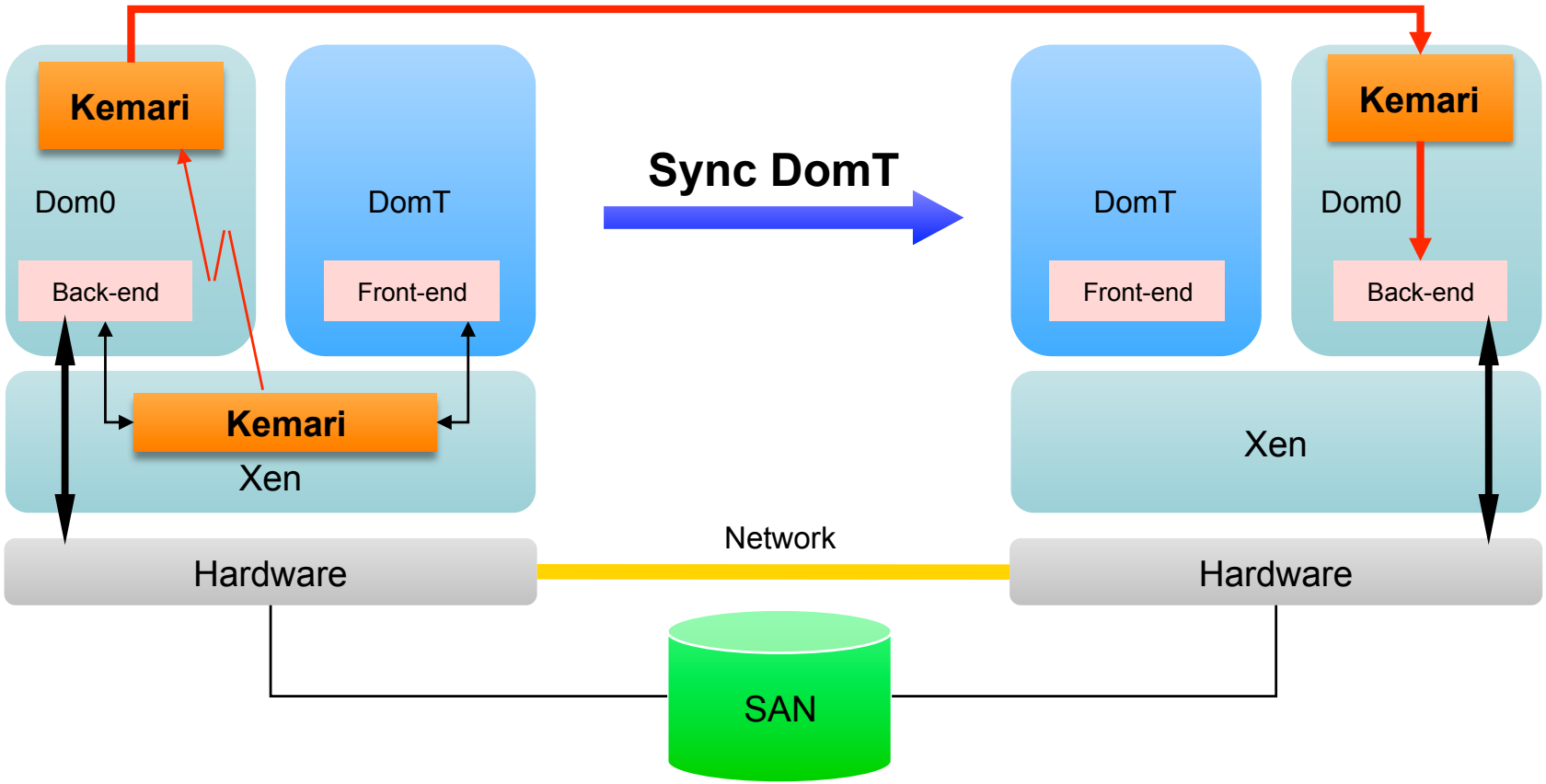
- Secondary will redo the same operation as Primary
  - Secondary will receive the same reply as Primary

Demo…



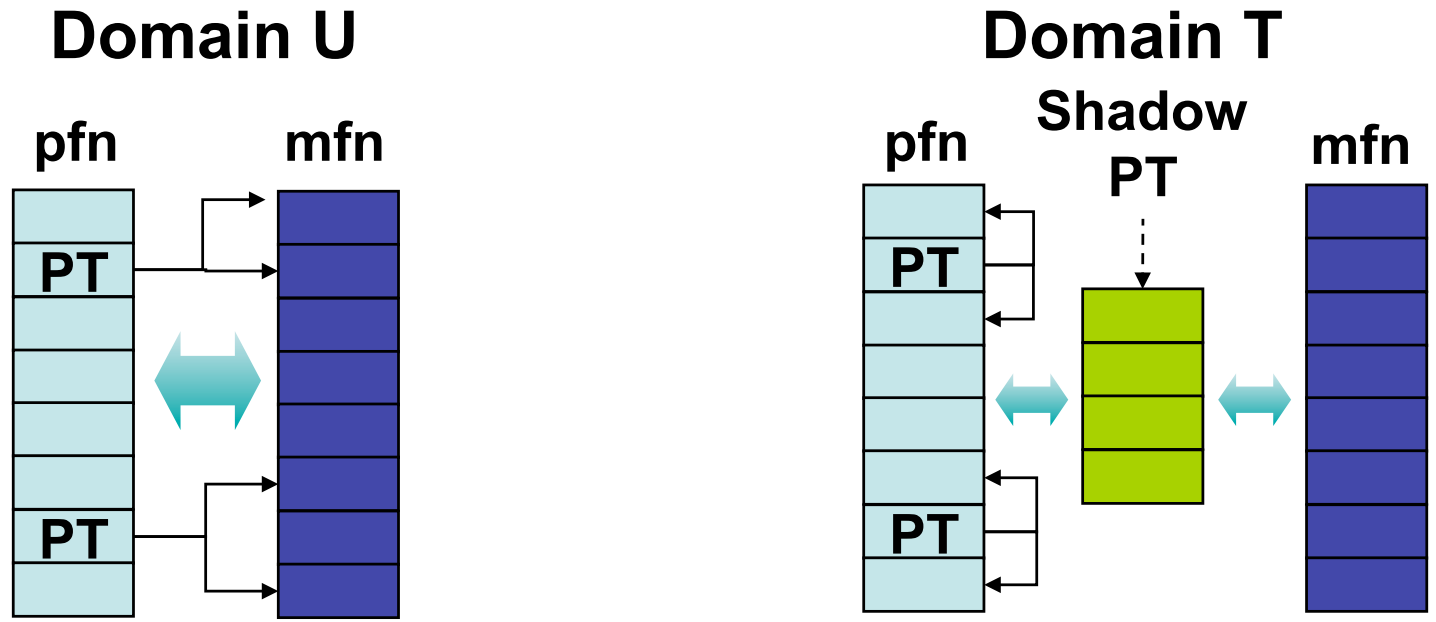
- Guest VM is running on Kemari
- Guest is running VNC server, and the client accesses via VNC client
- xclock is launched from the client
- See what happens to the clock when the primary physical server is shut downed from HP iLO2 management console

Architecture overview



- The core of the synchronization mechanism resides in hypervisor to synchronize virtual machines efficiently
- LOC ≅ 3000 (hypervisor: 1000, Dom0+Tools: 2000)

What is DomT?



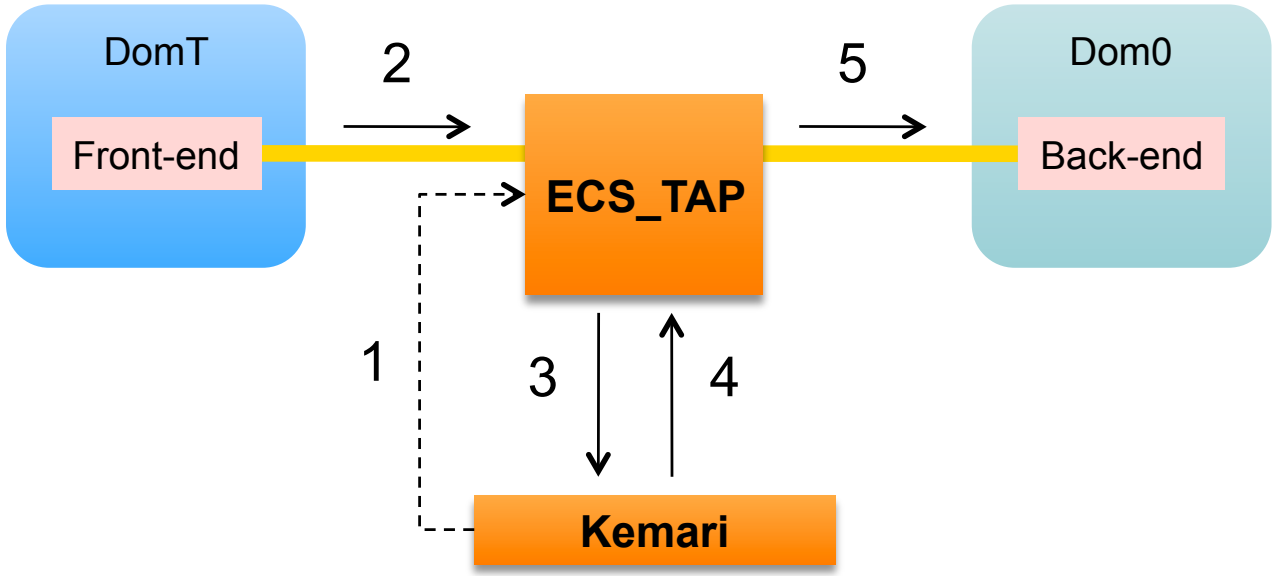
- Para-virtualized domain which uses shadow page table (auto-translated-mode)
- Don't have to translate the page tables on transferring
- DomT patch set for xen-3.0.4 was written by Michael A Fetterman from University of Cambridge

Implementation of Kemari

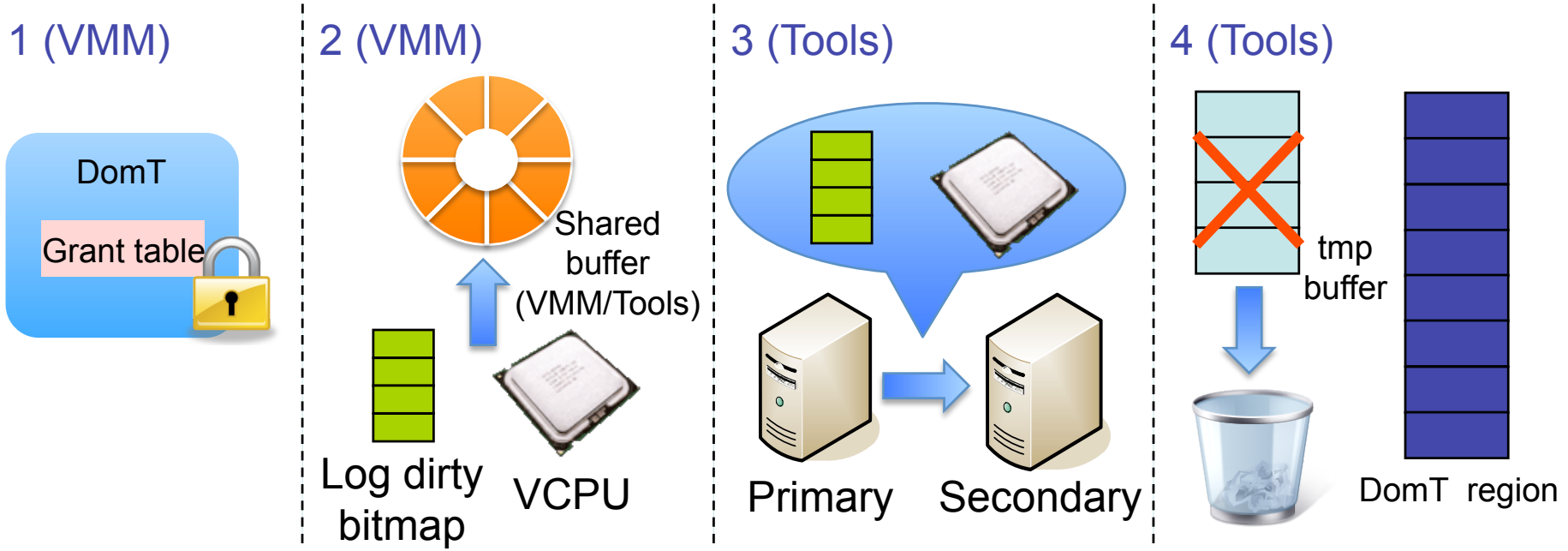
- Event Channel tapping
- Transferring DomT
- Restoring para-virtualized devices

Event Channel tapping

- Simple but the key component of Kemari
- Monitors IN/OUT or Both
- Registered function is called on specific events
- Dynamically attachable
  - May be useful for measurements



Transferring DomT



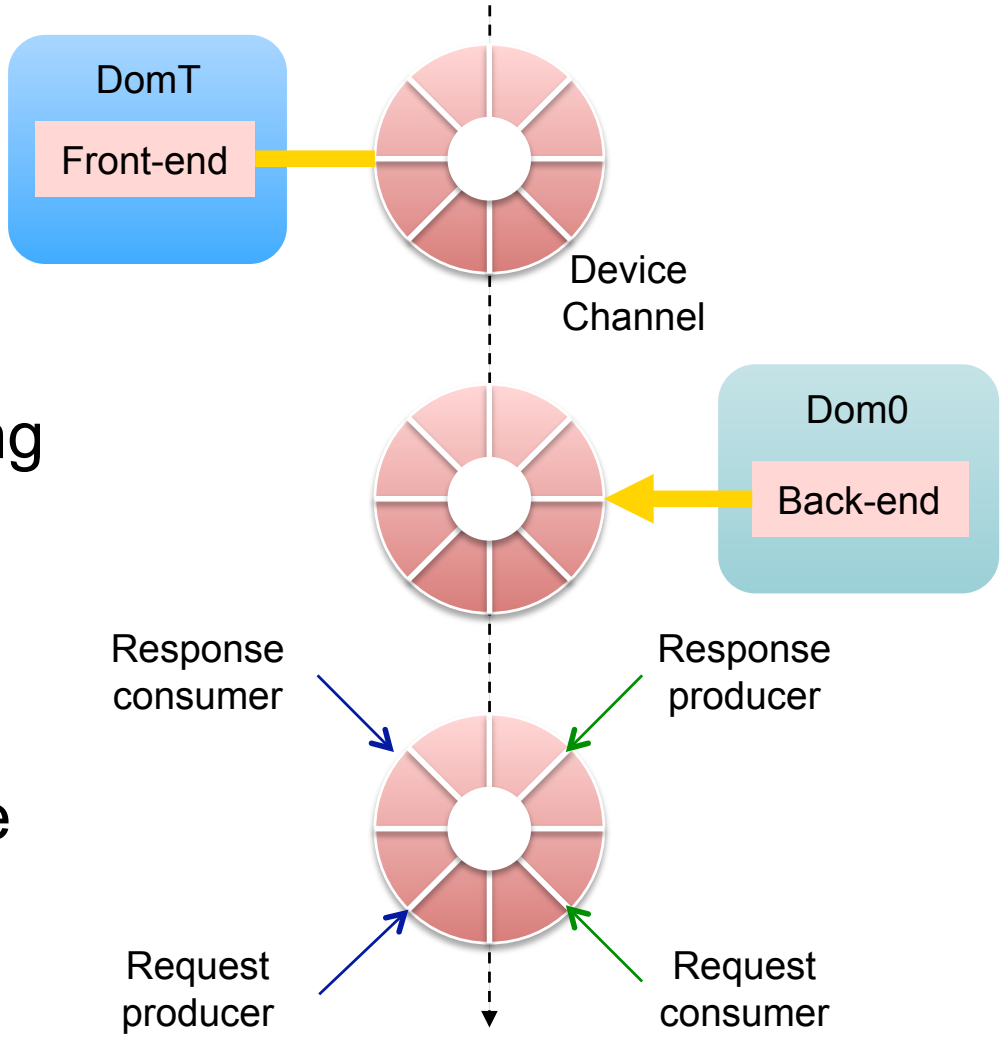
1. Pauses DomT and locks the grant tables. No need to suspend!
   - Grant tables are mapped at the last 4 pages of DomT region
2. Extracts dirtied pfn from the bitmap, copies pfns and the vcpu to the shared buffer, and notifies Tools via event channel
3. Maps dirtied pages, transfers pages and vcpu to the secondary
4. Secondary prepares temp buffers to rollback when failure is detected during transfer

Restoring para-virtualized devices

1. Device Channel is stored in DomT region

2. Attach the Back-end to the Device Channel using *BACK_RING_ATTACH* macro

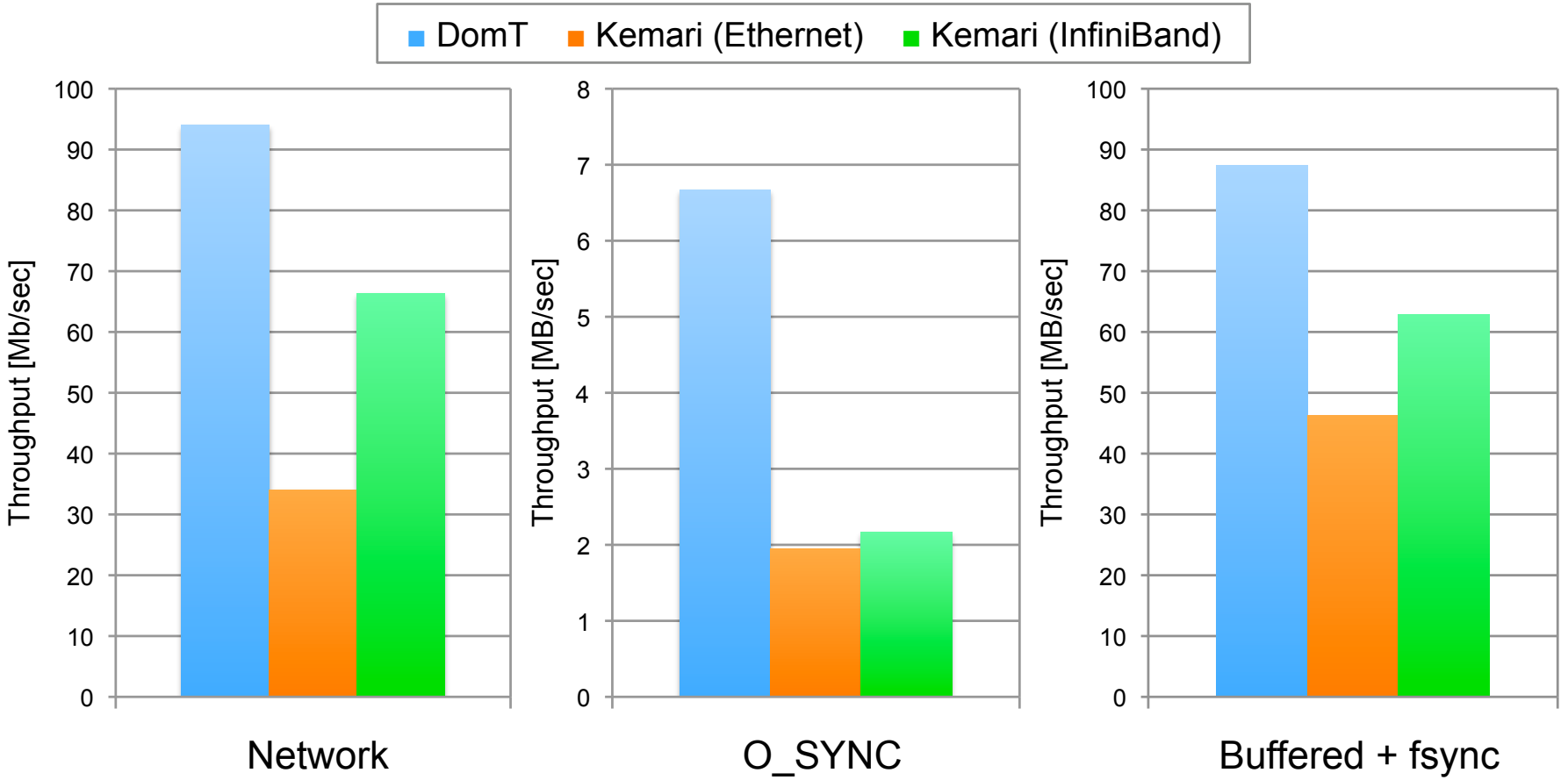
3. Adjust producer and consumer indexes of the Back-end appropriately



Evaluation

- Evaluation items
  - Performance of the Primary VM (Network and File I/O) using netperf and iozone
- Test machines
  - Hardware spec
    - CPU: Intel Xeon 3GHz X 2
    - Memory: 4GB
    - Network: Gigabit Ethernet, InfiniBand
    - SAN: FC Disk Array
  - VM spec
    - VMM: Xen 3.0.4 with DomT support
    - Guest OS: Debian Etch
    - Memory: 512MB

Performance of Primary VM



- InfiniBand boosted the performance of Network and Buffered + fsync, both of which dirties many pages
- All benchmarks continued transparently when the primary server was shut downed from the HP iLO 2 management console

Conclusion

- Kemari is a Virtual Machine synchronization mechanism to achieve Fault Tolerance
- Don't drop the ball! Don't drop the VMs!
- Implemented Kemari using Xen and DomT
  - Thanks to Michael from University of Cambridge
- Demonstrated Kemari achieved acceptable performance

Future work

- Demonstrate the range of applications Kemari can manage to run transparently
- Improve the performance of I/O intensive applications that send numbers of events
- Hosting HVM domains with PV drivers
- Hosting multiple domains simultaneously
- Functions to implement for practical use such as detection of HW failure and failover mechanism