



# Virtual Machine Synchronization for High Availability Clusters

---

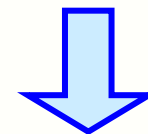
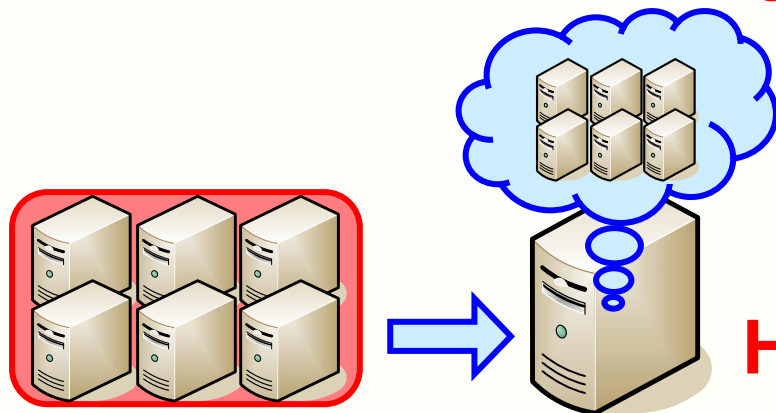
Yoshiaki Tamura, Koji Sato,  
Seiji Kihara, Satoshi Moriai  
NTT Cyber Space Labs.

2007/4/17

# Consolidating servers using VM

- Internet services are growing
- Costs for running servers aren't small
- Consolidating servers using VM is popular
- However...

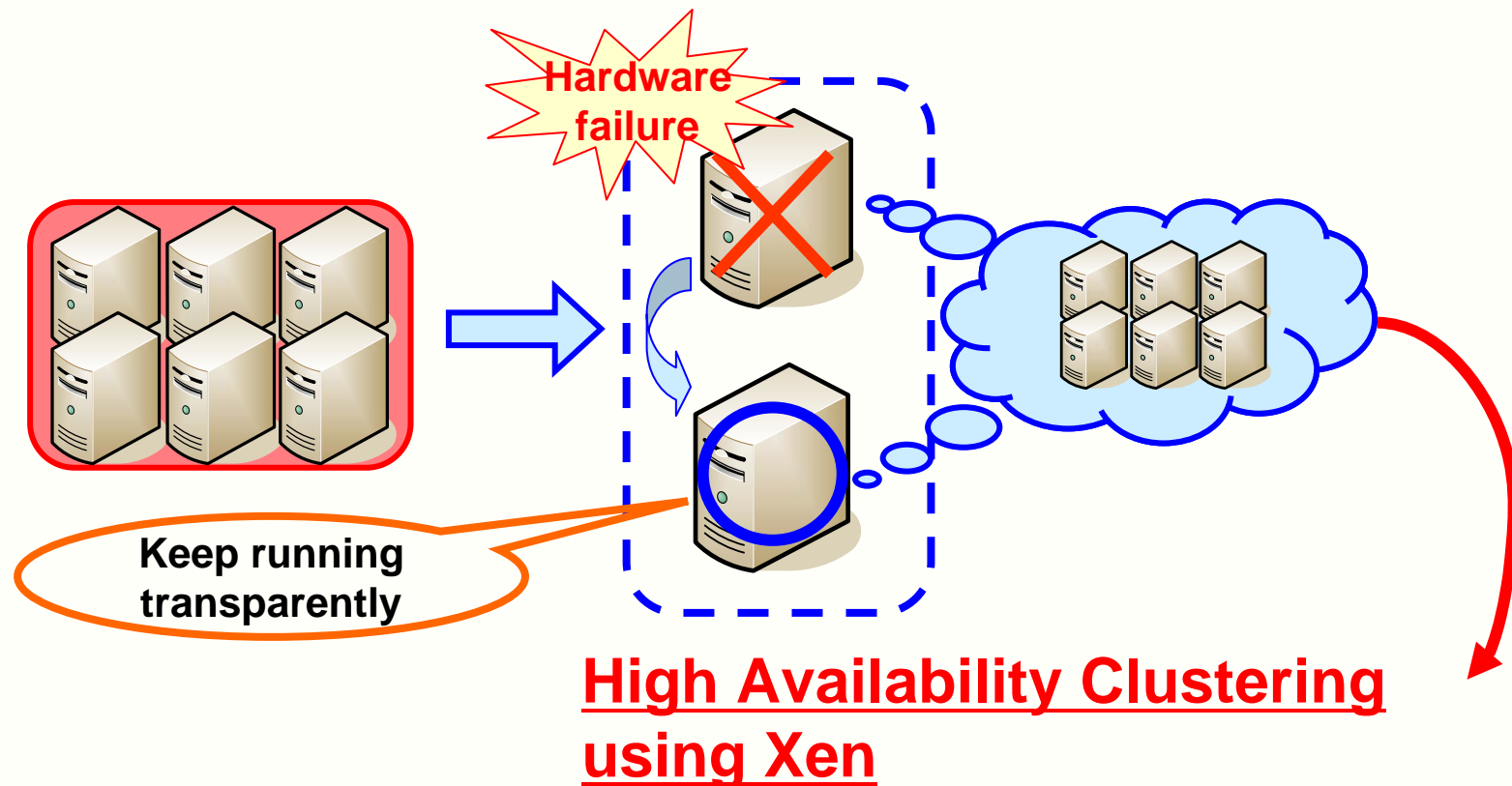
**More services may fail in a consolidated environment**



**High availability is needed !**

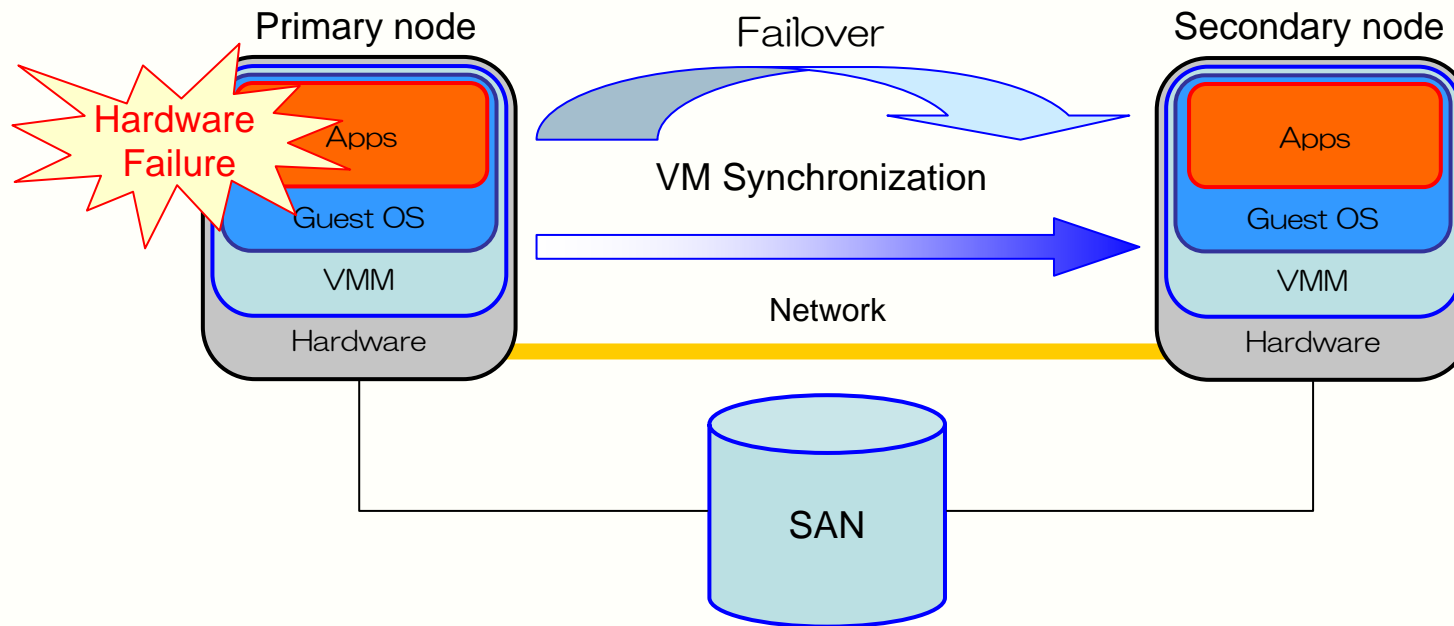
# Our goal

A new high availability clustering  
independent of applications or OS

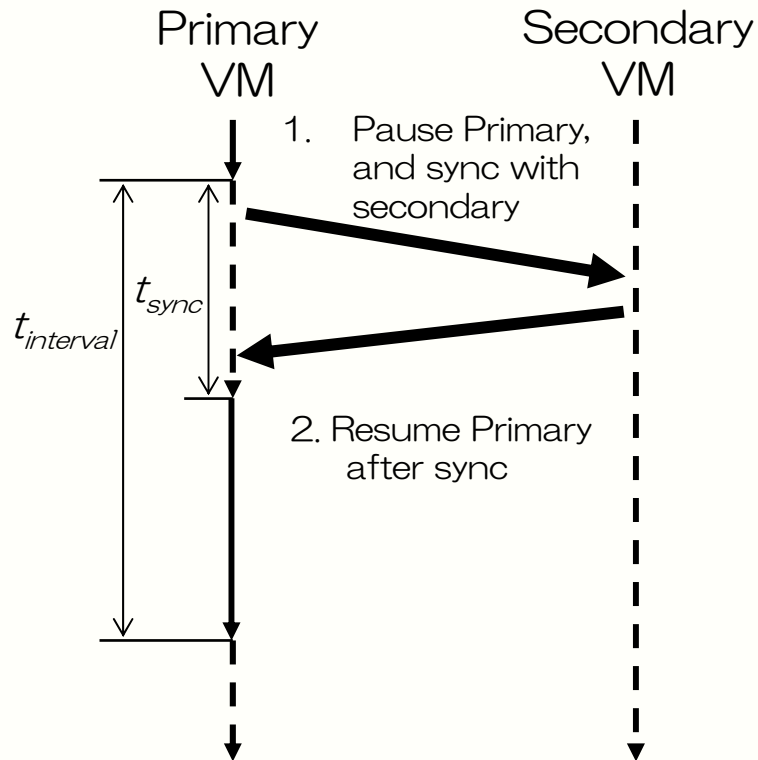


# What needs to be done?

- Virtual Machine Synchronization
    - ▶ Primary node and Secondary node must be identical
  - Detection of failure
  - Failover mechanism
- } **Extension of current techniques**



# How to synchronize VMs?



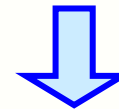
■ Need to make the overhead of sync smaller

▶ 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, timer, console

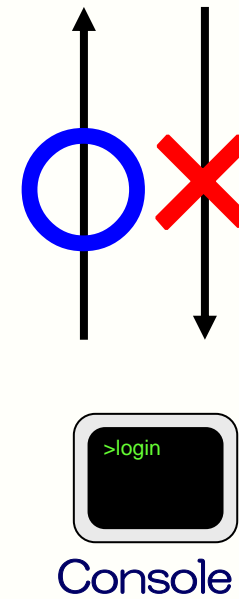
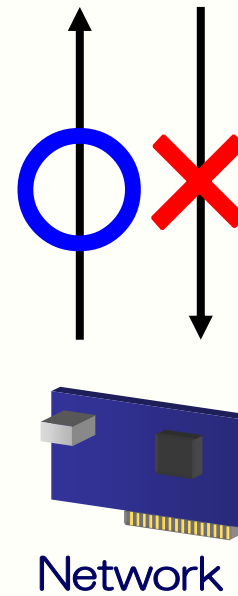
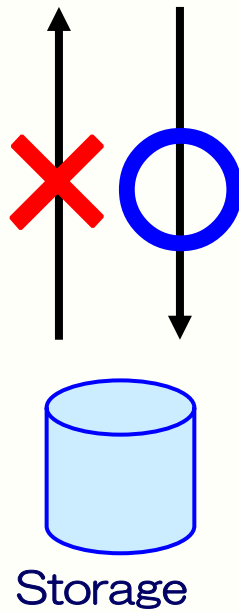
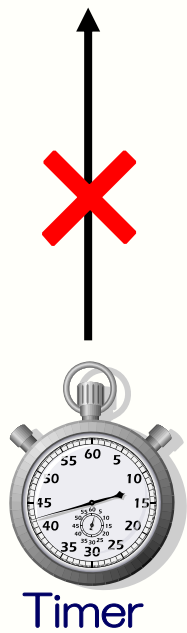
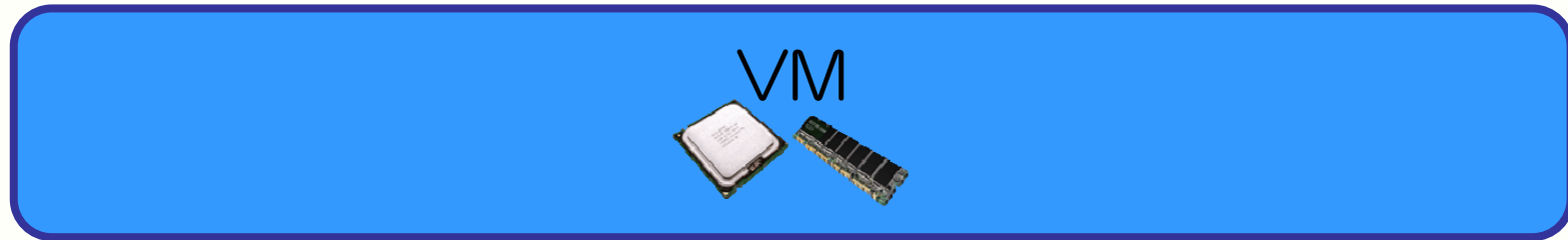
# Sync VMs on every event

- Types of events in a server environment
  - ▶ Timer, network, storage are majority
  - ▶ Network and File I/O performance may degrade
- Comparison of performance between no-sync and sync on every event (Netperf, IOzone)
  - ▶ Network 76% down
  - ▶ File I/O: O\_sync write 90% down, Bufferd write + fsync 69% down

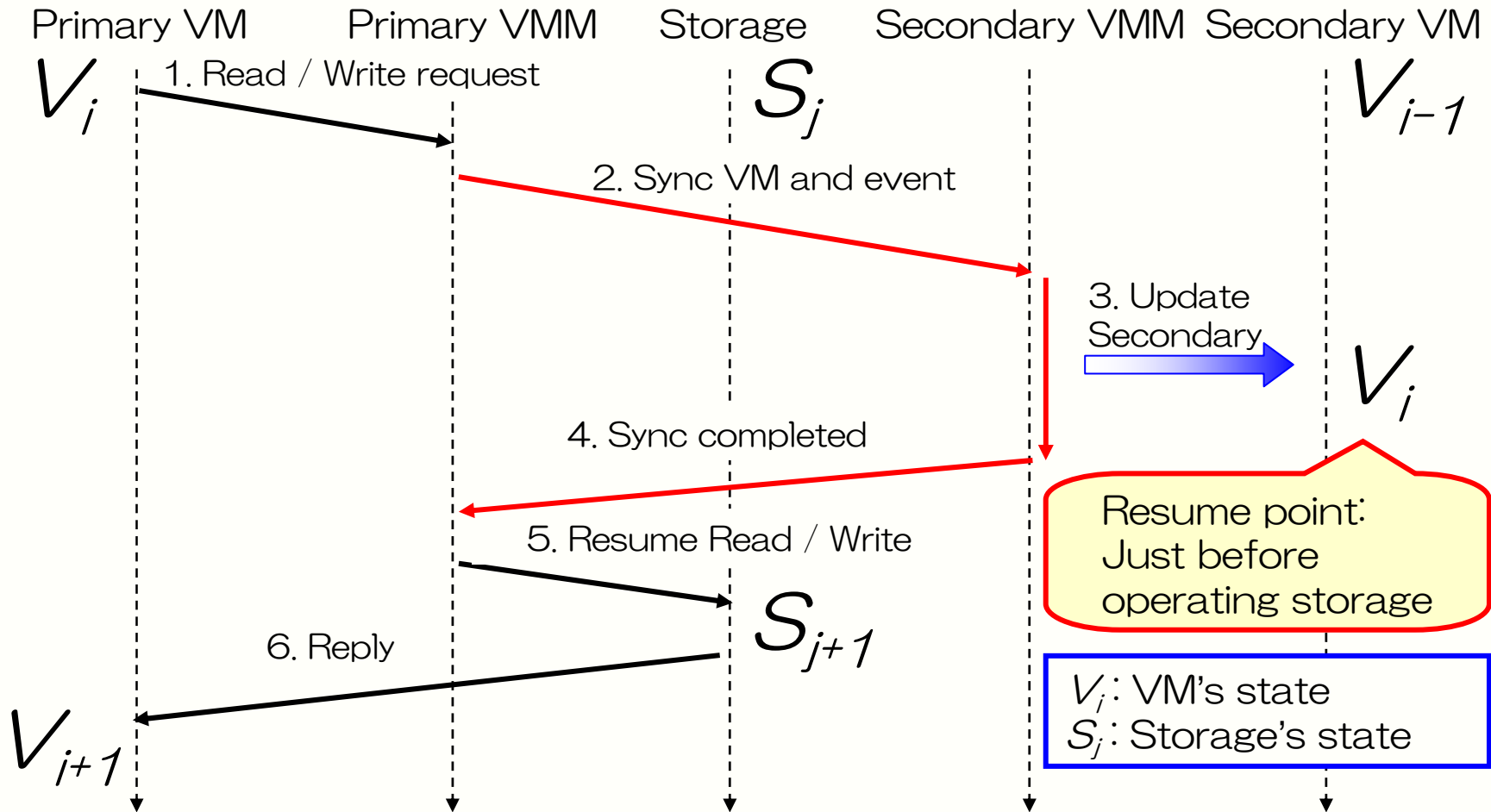
	Network [Mbps]	File I/O [MB/s]	
		O_SYNC	Buffered + fsync
No-sync	93.39	5.67	53.95
Sync on event	22.81	0.58	16.54

- Sync protocol needs to be improved
  - ▶ Keep the degradation to 50% at most

# Types of events to sync

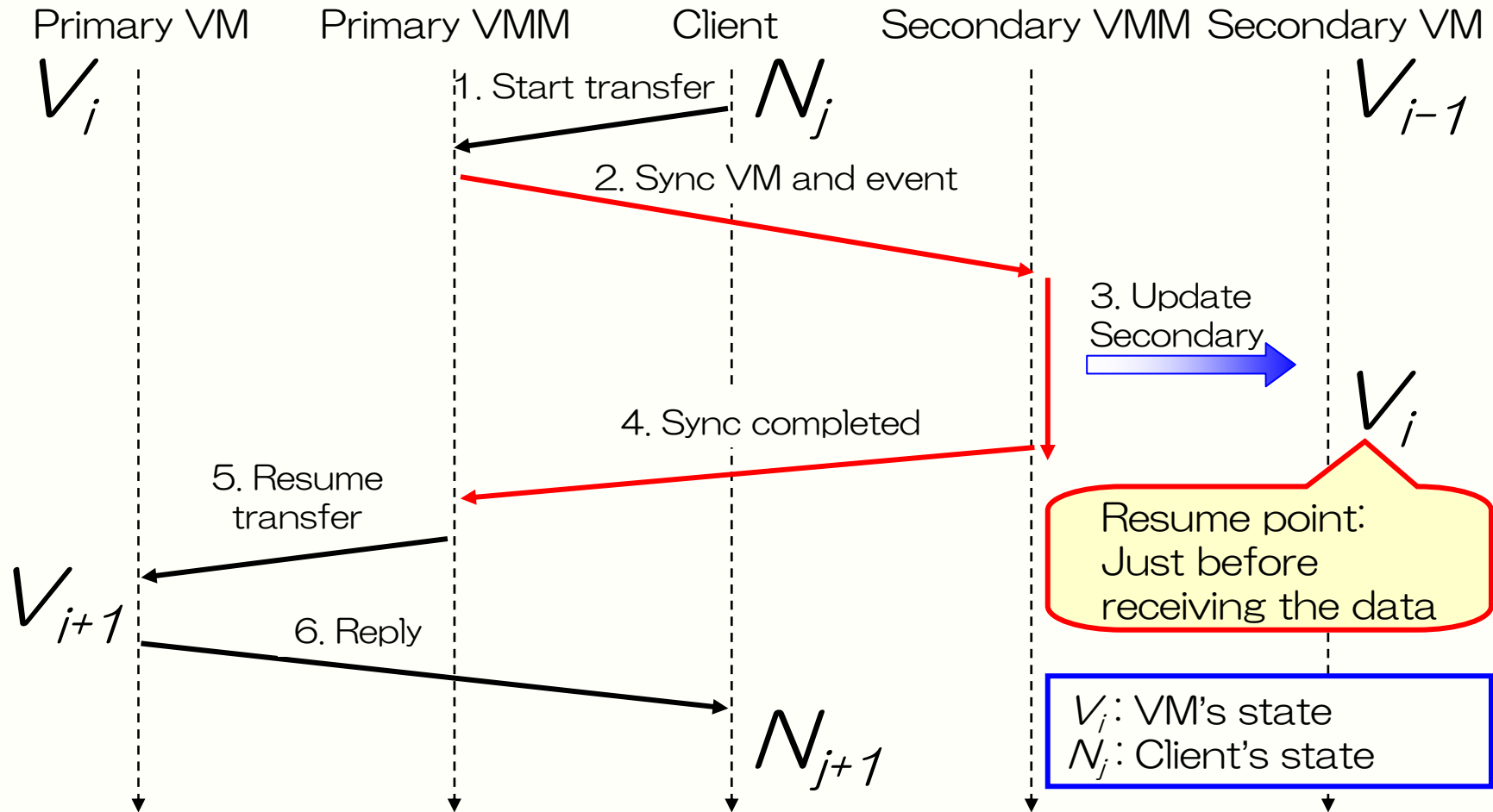


# Sync on events from VM to storage



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

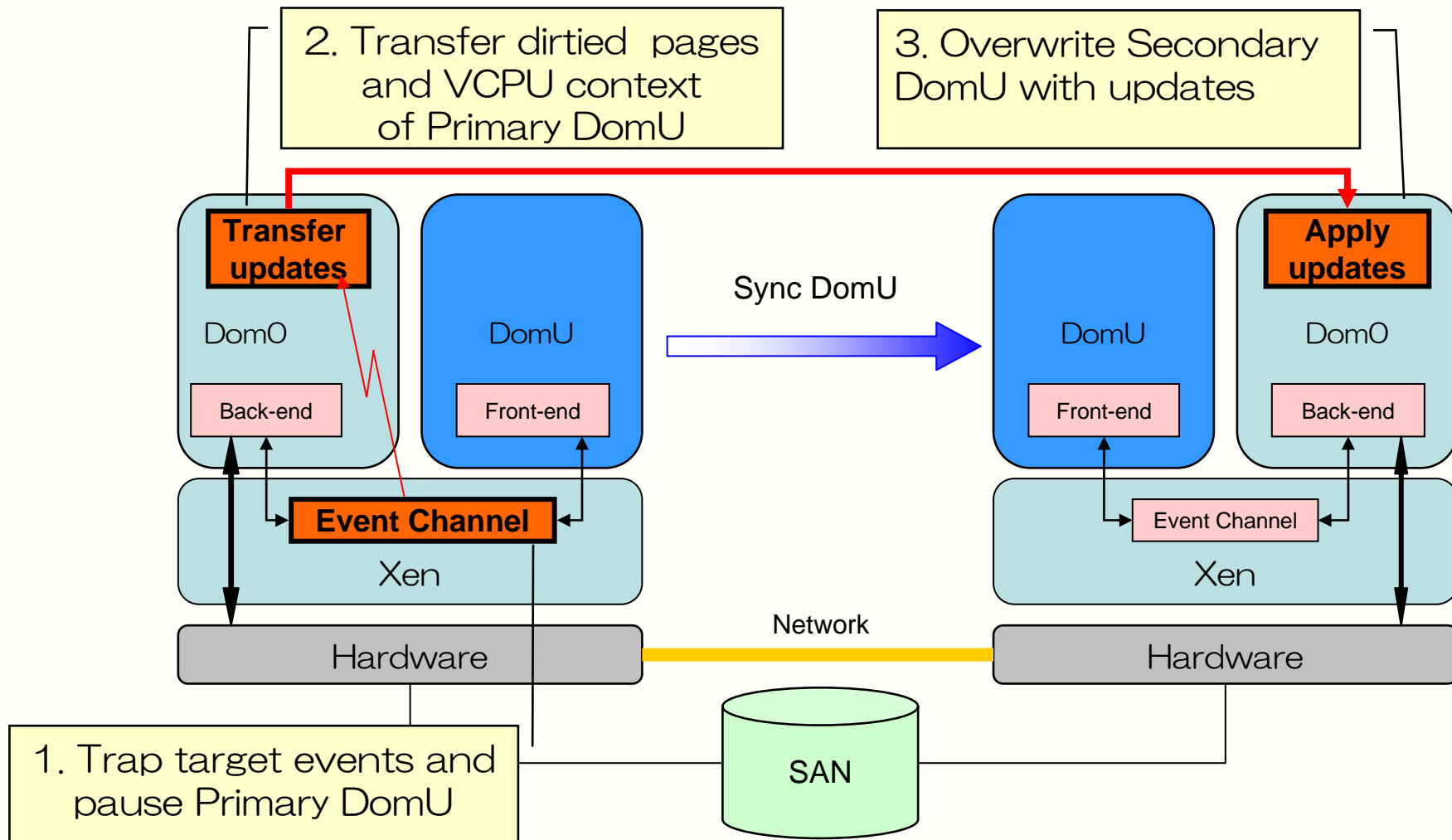
# Sync on events from network to VM



- Secondary may resend the same data Primary has sent...
  - ▶ Reliable protocols (ex: TCP) can handle by itself
  - ▶ Apps should handle in case of unreliable protocols (ex: UDP)

# Prototype

- Implemented the protocol using Xen's live migration (2006/12)



# Implementation

- `xen/common/event_channel.c`
  - ▶ Hooks at `evtchn_send()`
  - ▶ Device → Domain
    - Calls `domain_pause_by_systemcontroller(rd)`
    - Notifies the user land process to start transfer via `VIRQ`
    - `evtch_set_pending`
  - ▶ Domain → Device
    - Sets the port number of the user land process's event channel
    - Calls `domain_pause_for_debugger()`
    - User land process sends event to the device after transfer
- `tools/libxc/xc_linux_save.c`
  - ▶ Waits for a signal from the user process
  - ▶ Sends dirtied pages and vcpu context to Secondary
  - ▶ Continuously wait and stop until receiving the signal to finalize
- `tools/libxc/xc_linux_restore.c`
  - ▶ Receives dirtied pages and vcpu context, and updates the domain
  - ▶ Continuously receives data and updates the domain
  - ▶ Starts building domains after receiving notification from Primary
  - ▶ Fails pinning page tables

# Evaluation



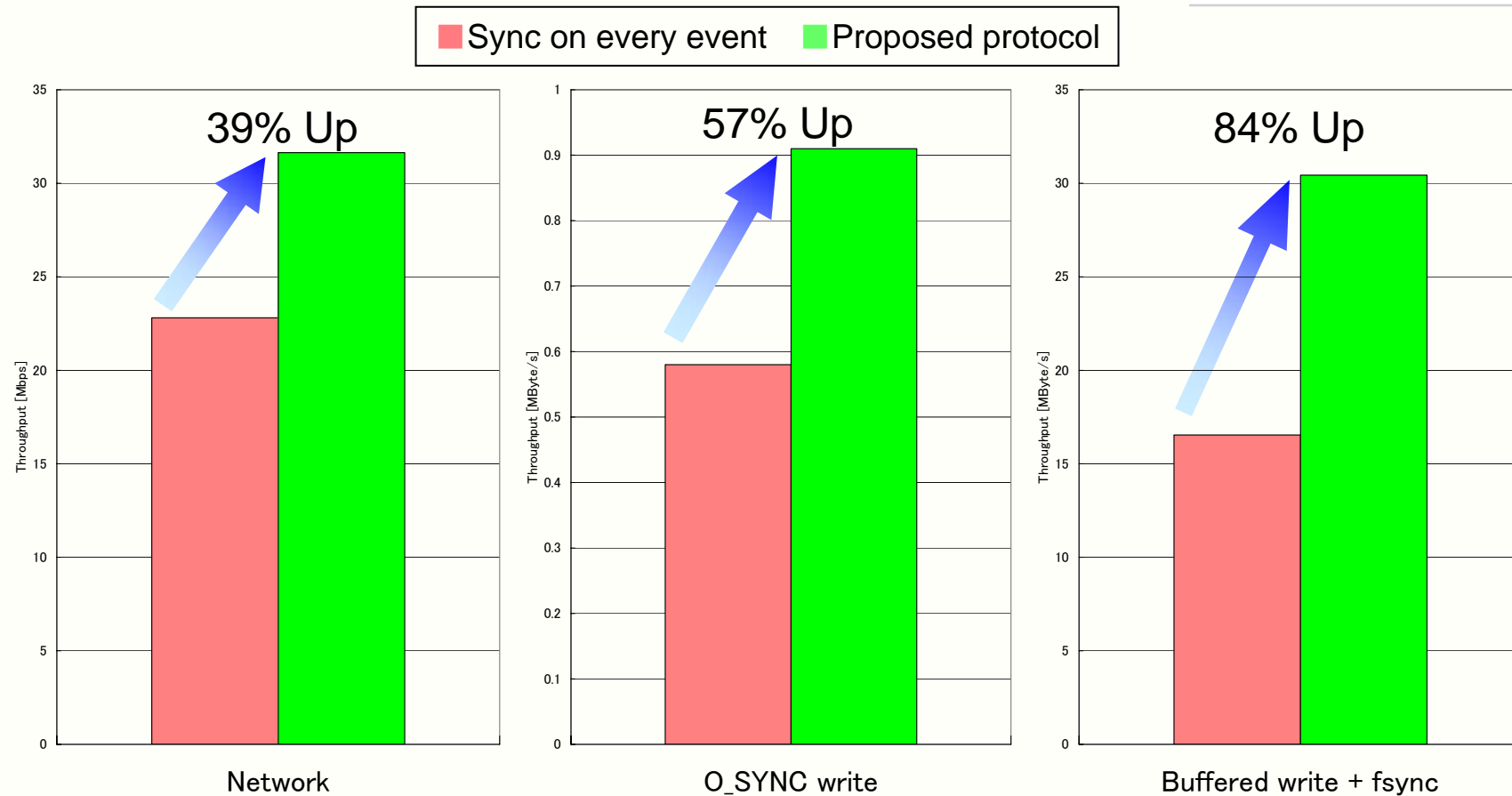
## ■ Evaluation items

- ▶ Performance of the Primary VM (Network and File I/O)
- ▶ Number of events to sync
- ▶ Average time to transfer dirtied page

## ■ Test machines

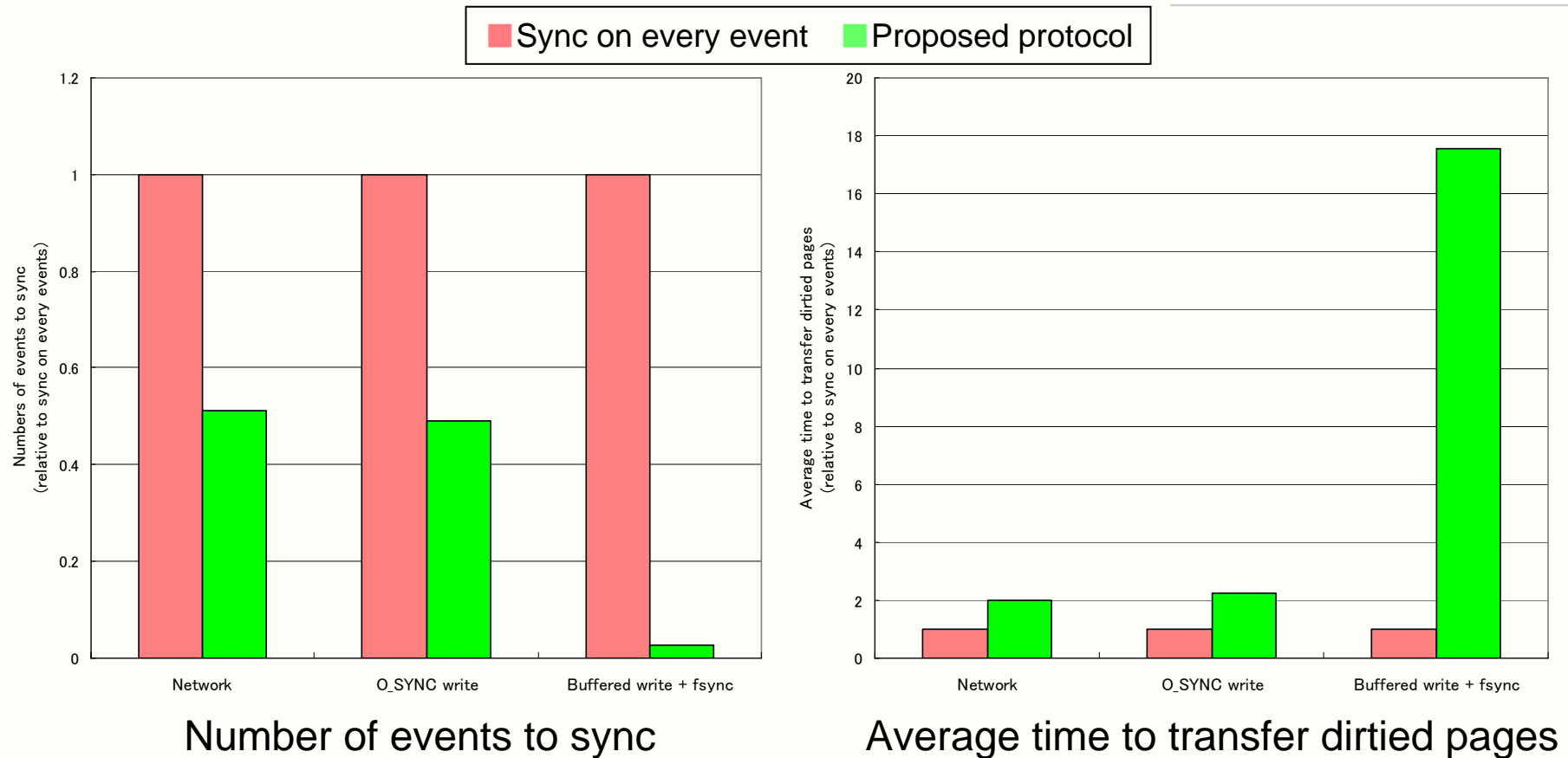
- ▶ Hardware spec
  - CPU: Intel Xeon 3GHz X 2
  - Memory: 4GB
  - Network: Gigabit Ethernet
  - SAN: FC Disk
- ▶ VM spec
  - VMM: Xen unstable (2006/12)
  - Guest OS: Debian Etch Testing
  - Memory: 1GB

# Performance of Primary VM



- Performance improved by applying proposed protocol
  - ▶ Degradation of buffered write + fsync compared to no-sync was 44%

# Num of events, Average time to transfer



- Number of events to sync reduced 49% - 96%
- Average time to transfer dirtied pages multiplied by 2 -18
  - ▶ Optimize transfer function, compress pages ?

# Related Work



- Hypervisor-based fault tolerance
  - ▶ T. C. Bressoud et al. ACM TOCS '96
  - ▶ Protocols to implement a fault tolerant system using VMM
- SecondSite
  - ▶ B. Cully et al. HotDep '06
  - ▶ Continuously checkpoints a domain to a persistent storage
- ExtraVirt
  - ▶ D. Lucchetti et al. SOSPP '05
  - ▶ Runs multiple replicas on a single machine to recover from processor faults

# Conclusion



- VM synchronization for high availability clusters
  - ▶ Applications and OS can continue without modifications
  - ▶ Proposed protocol makes overhead smaller
- Implemented the prototype using Xen
- Showed the effectiveness of the protocol
  - ▶ Performance improved 39% - 84%
  - ▶ Number of events to sync reduced 49% - 96%

# Future work



- Function for Xen to stop domains instantly and resume on Secondary
  - ▶ Currently using pause to stop domains
- Improve the performance of Primary VM
  - ▶ Optimize transfer function
- Functions to implement for practical use
  - ▶ Detection of failure
  - ▶ Failover mechanism