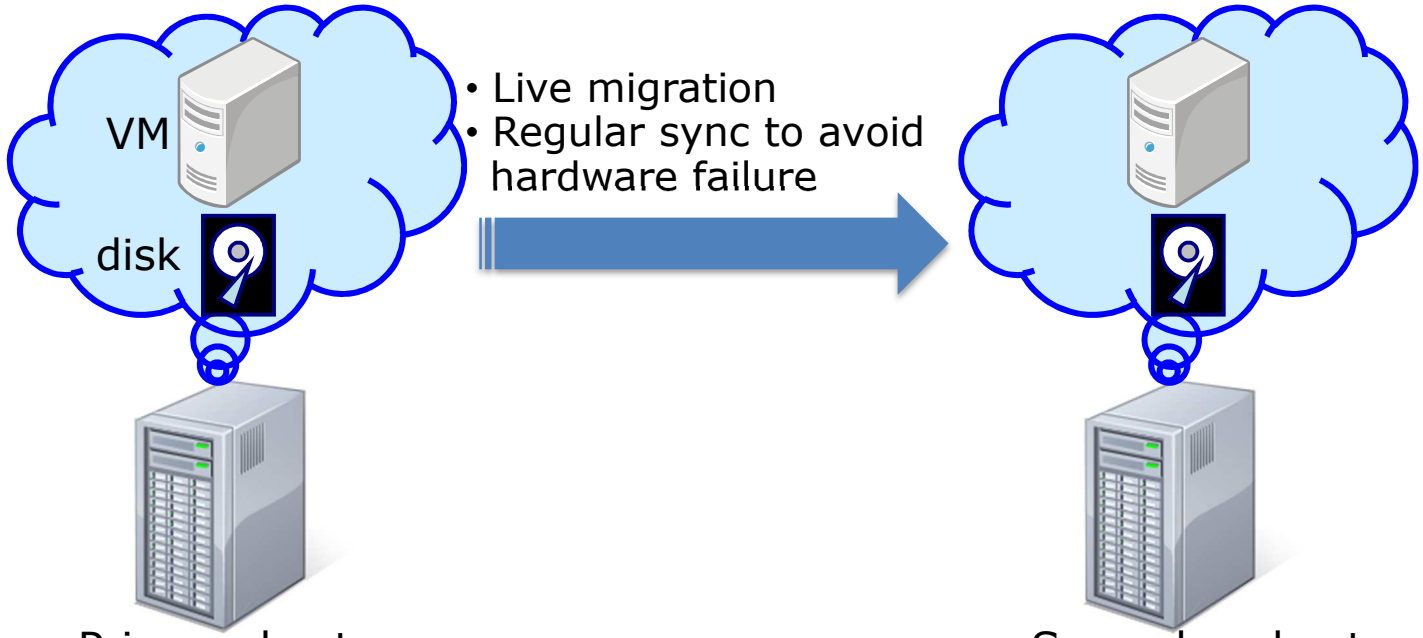
Rapid VM Synchronization with I/O Emulation Logging-Replay

Kei Ohmura
NTT Cyber Space Labs.
ohmura.kei@lab.ntt.co.jp

Aug 16, 2011

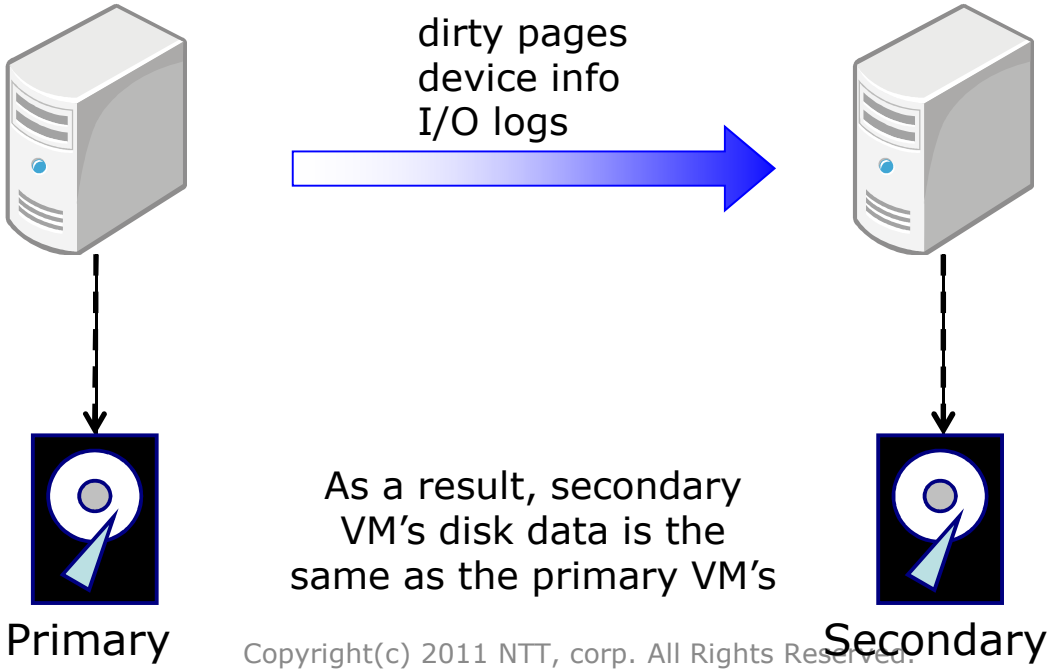
Motivation

- VM replication including backend storage is important
  - enables use of shared-nothing architecture
  - enhances VM mobility
- However, it heavily degrades VM performance

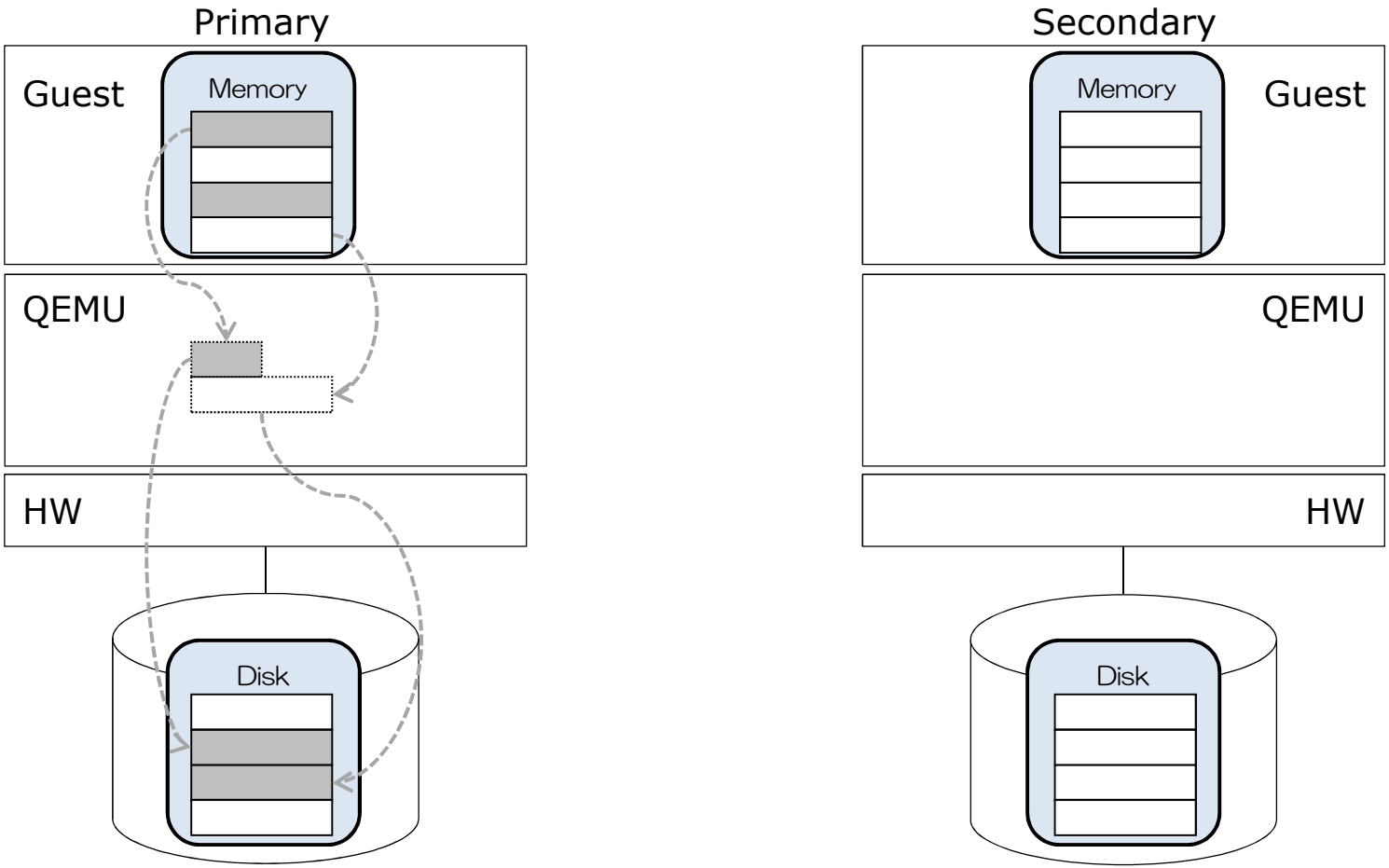


Our goal and approach

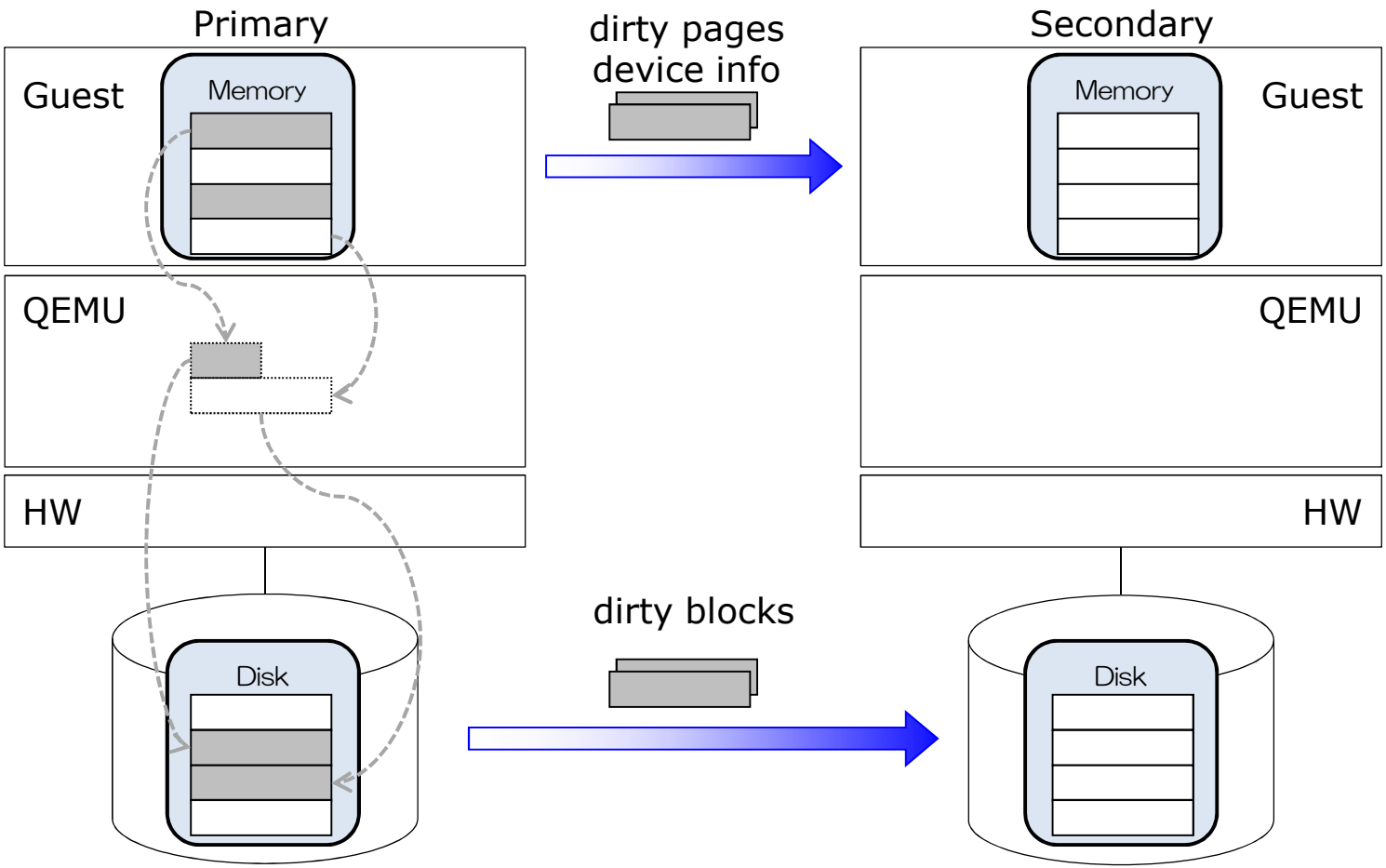
- Goal
  - Developing a rapid VM (including backend storage) synchronization mechanism
- Approach
  - logging-replay: The secondary VM replays the disk I/O events produced by the primary VM.



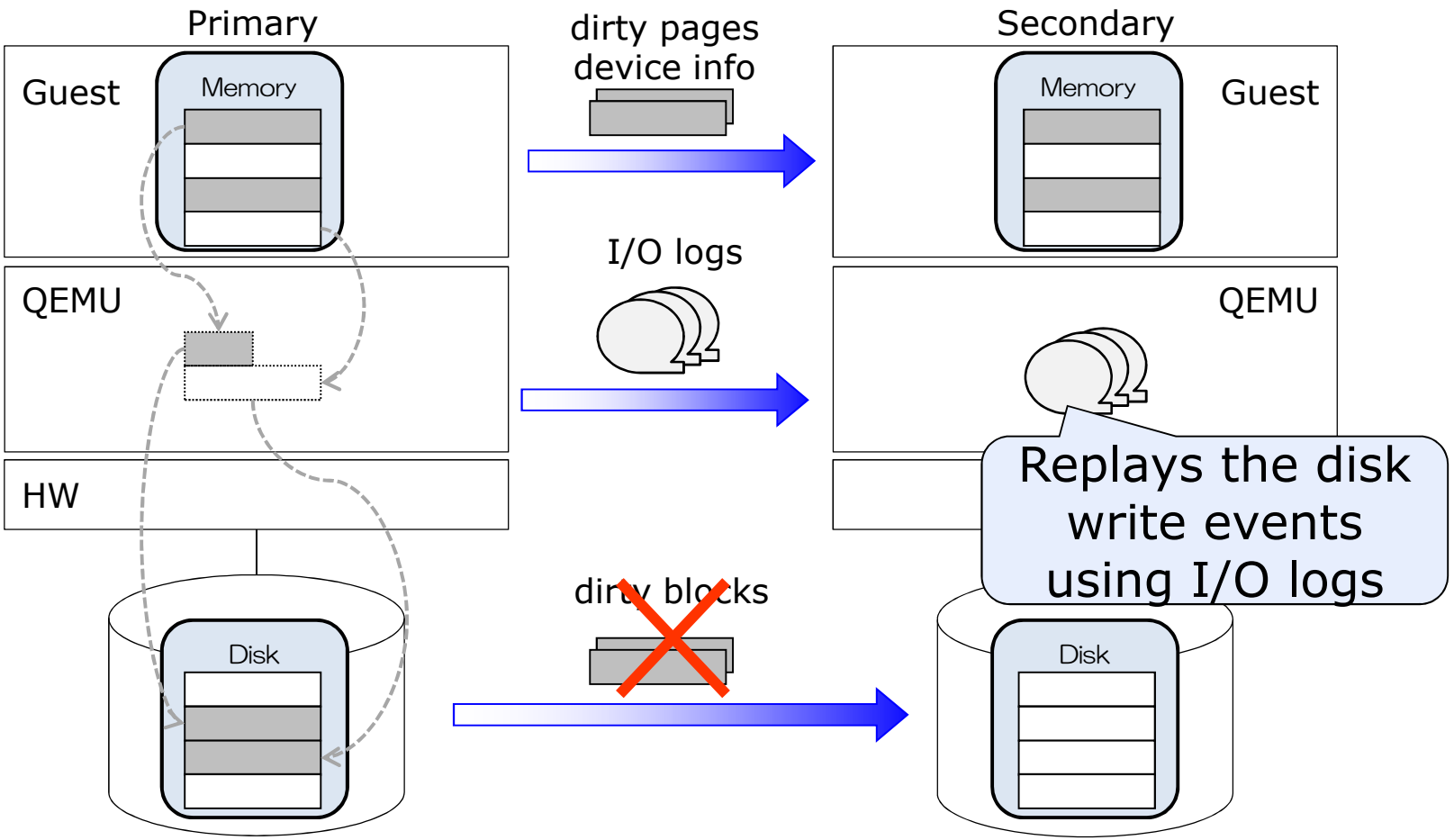
How to replicate VM and disk data?



How to replicate VM and disk data?



How to replicate VM and disk data?



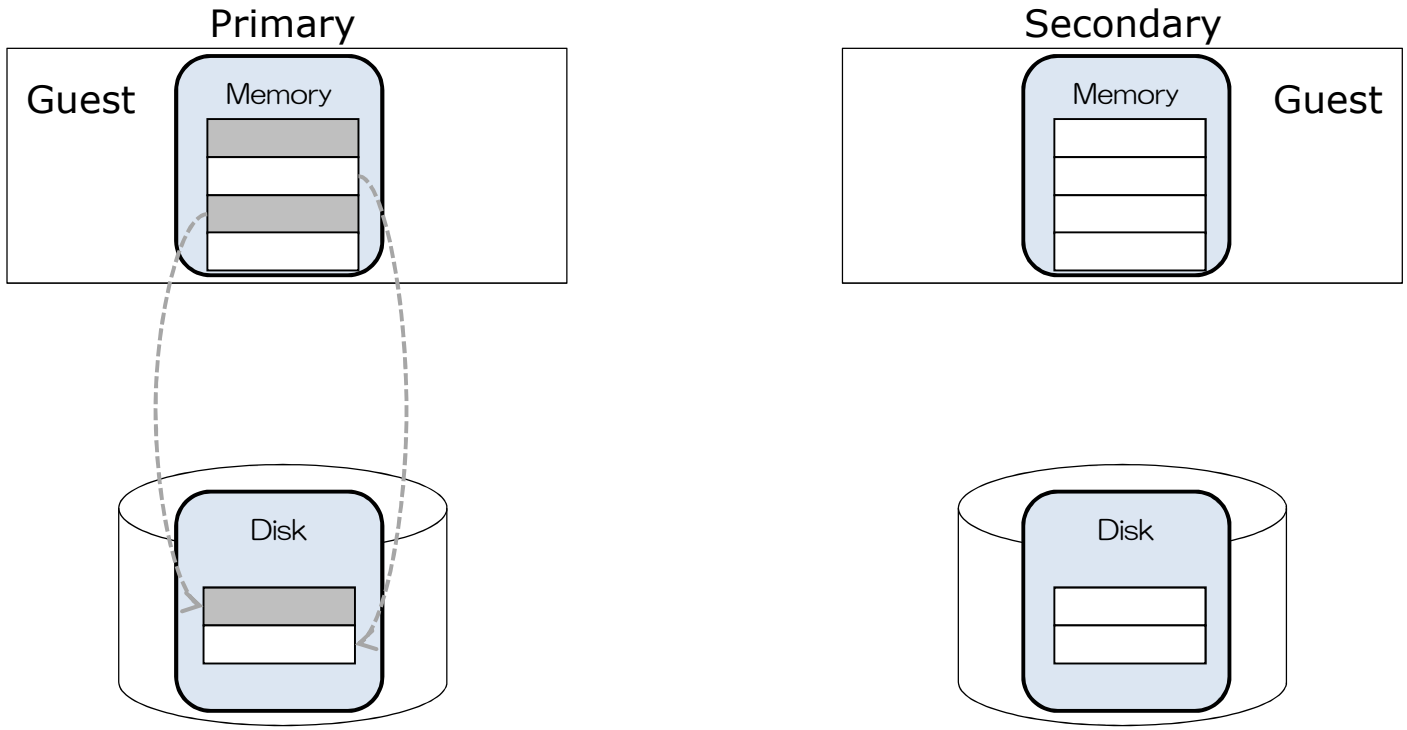
- Transfer I/O logs instead of dirty blocks to the secondary VM
  - Transfer VM info(dirty pages and device info) to the secondary VM as usual
- The secondary VM only replays disk write events using I/O logs

Disk consistency

- Needs to be the same VM memory as the primary VM's when secondary VM replays disk write events
  - Secondary VM can't make the same VM memory as the primary VM's
- Should we sync VMs when primary VM runs disk write events?

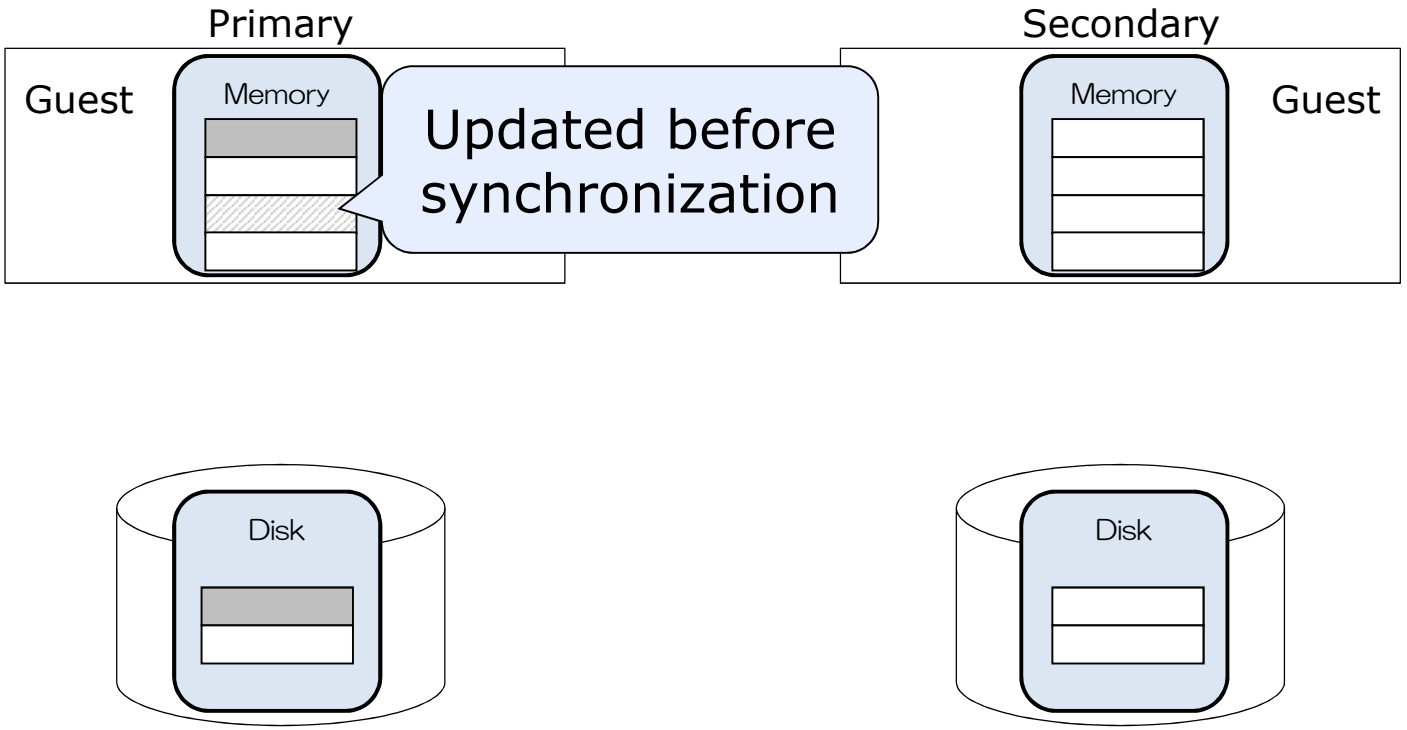
Specific event-driven VM synchronization

- Disk write using two kinds of data
  - dirty pages and non-dirty pages



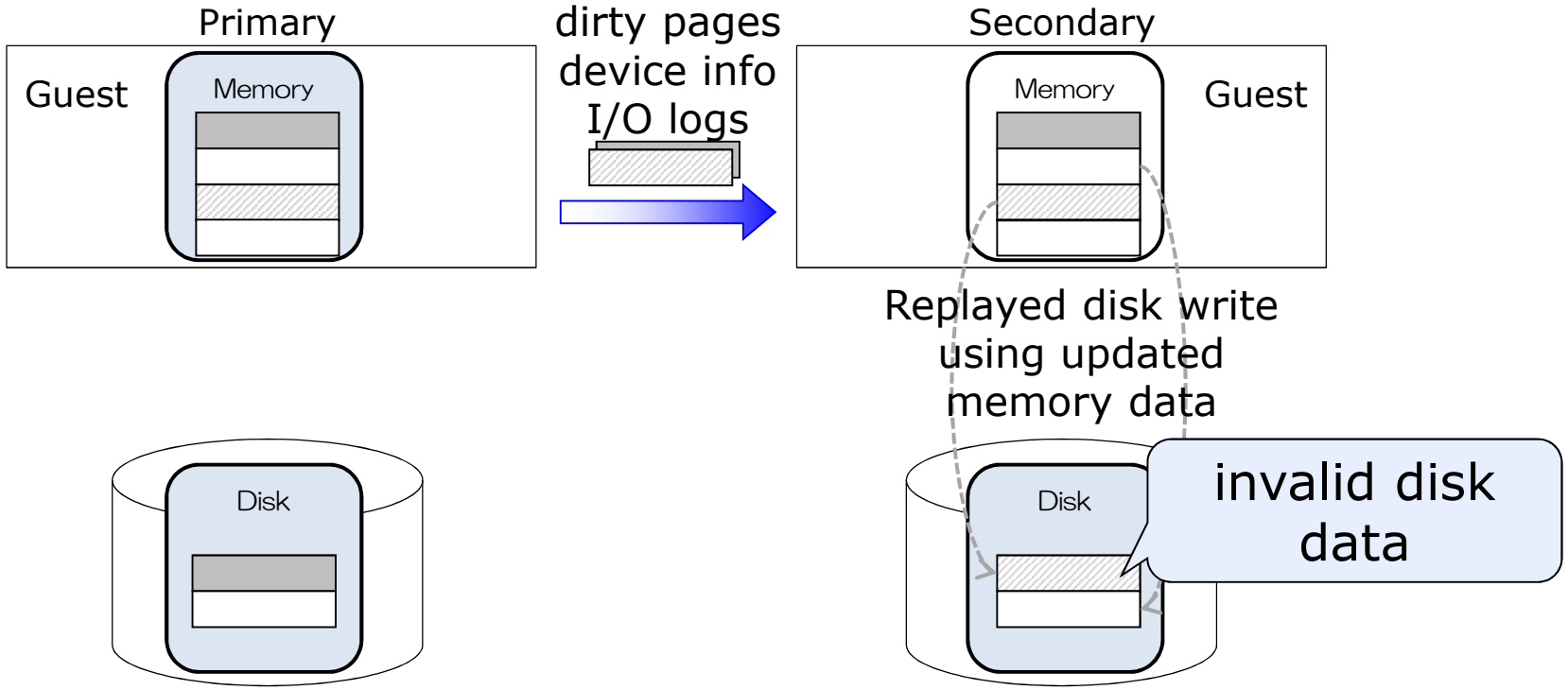
Specific event-driven VM synchronization

- Disk write using two kinds of data
  - dirty pages and non-dirty pages



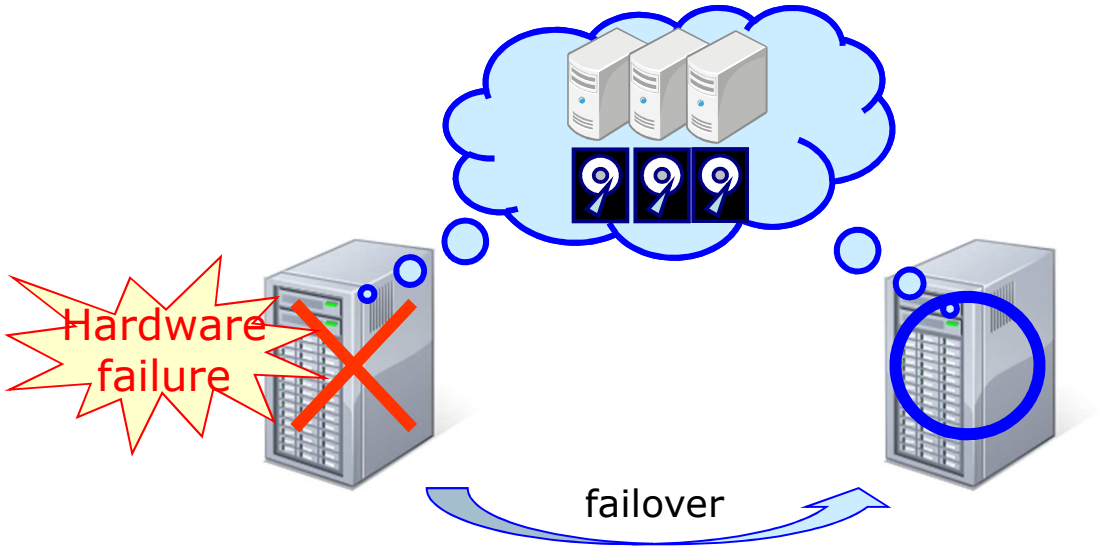
Specific event-driven VM synchronization

- Disk write using two kinds of data
  - dirty pages and non-dirty pages

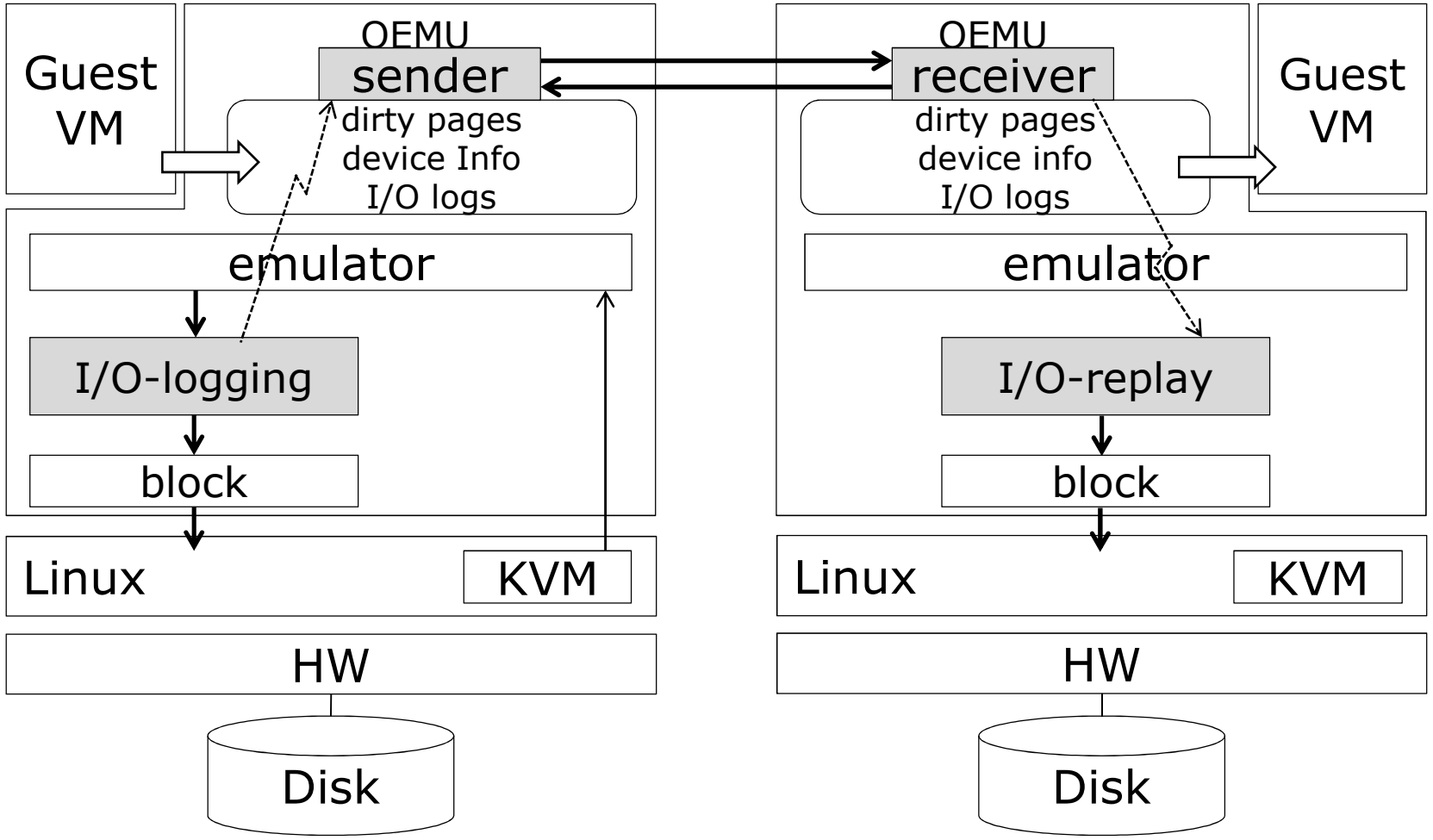


Implementation

- Kemari: VM synchronization mechanism for achieving fault tolerance
  - Needs shared disks
- Kemari with logging-replay can become a shared-nothing architecture

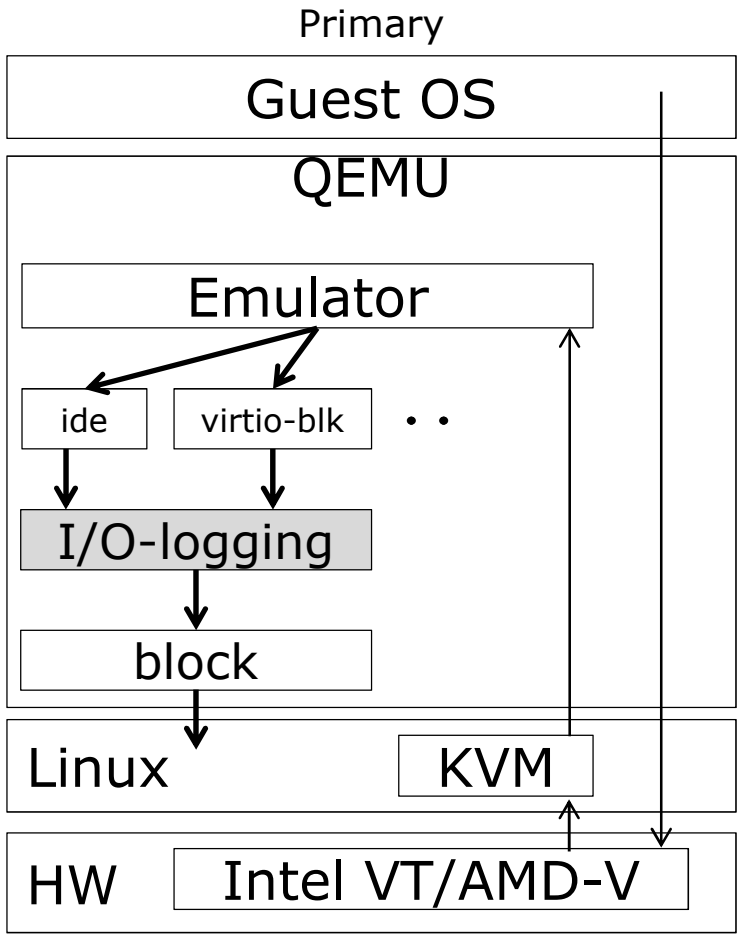


Architecture based on QEMU/KVM



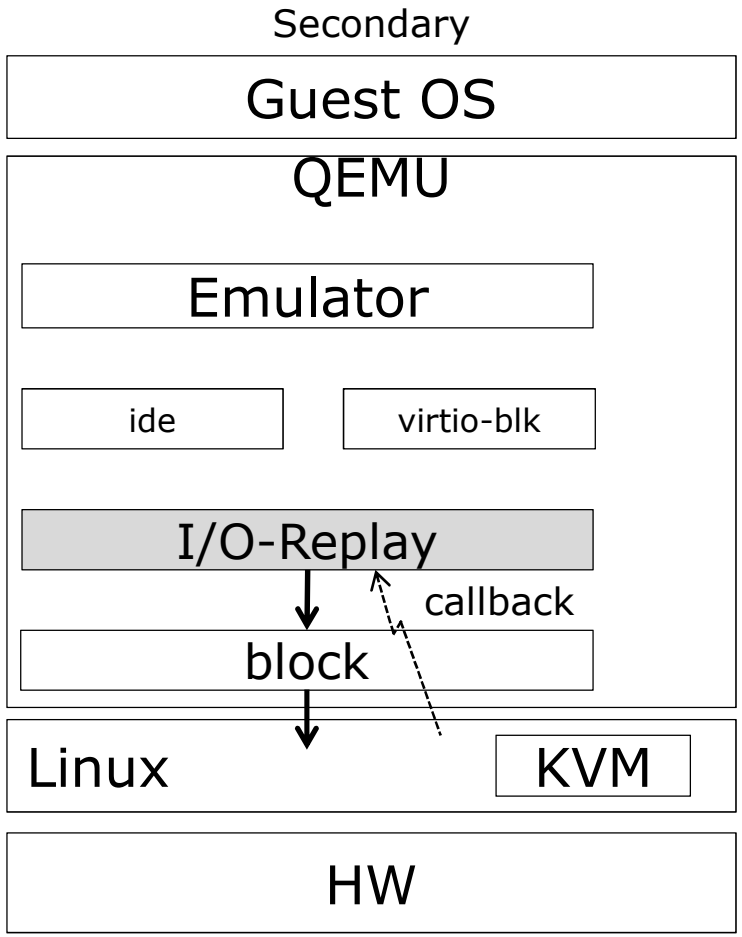
- I/O-logging: saves the disk write events' logs
- I/O-replay: replays the disk write events using I/O logs

I/O-logging: saves the disk write events' logs



- Saves I/O logs at block layer in QEMU
  - applicable to many device models
- Need to save the following information to replay disk I/O events:
  - device name
    ⇒ to get BlockDriverState structure
  - memory page address
    ⇒ to get disk write data
    （memory data transferred already）
  - num and location of disk sectors
    ⇒ to write data

I/O-Replay: replays the disk write events using I/O logs

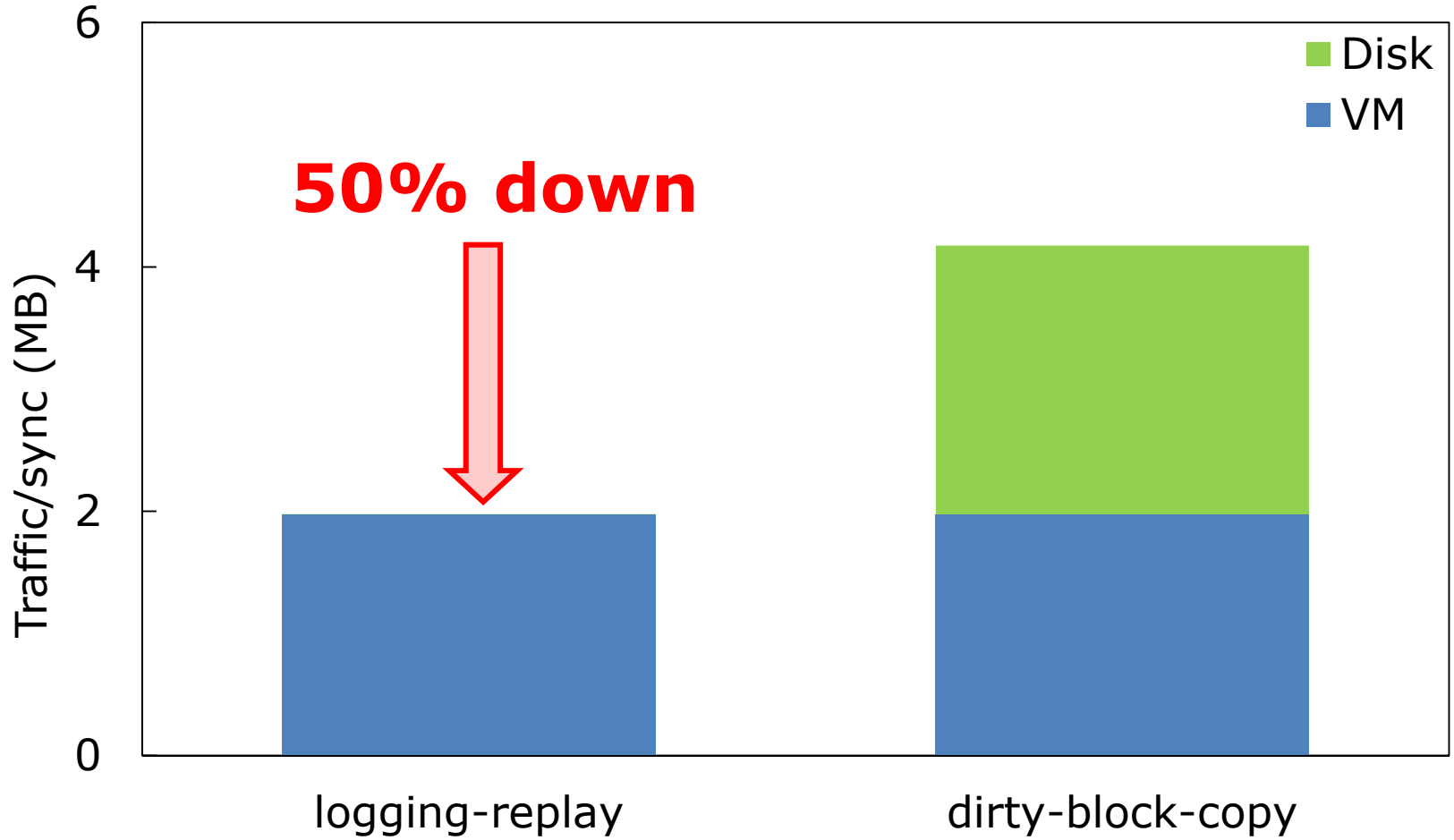


- Secondary VM requests I/O events using I/O logs from block-layer
- I/O-Replay's callback function is invoked when disk write is finished
- I/O-replay runs at the block-layer
  - device info is not modified

Experimentation

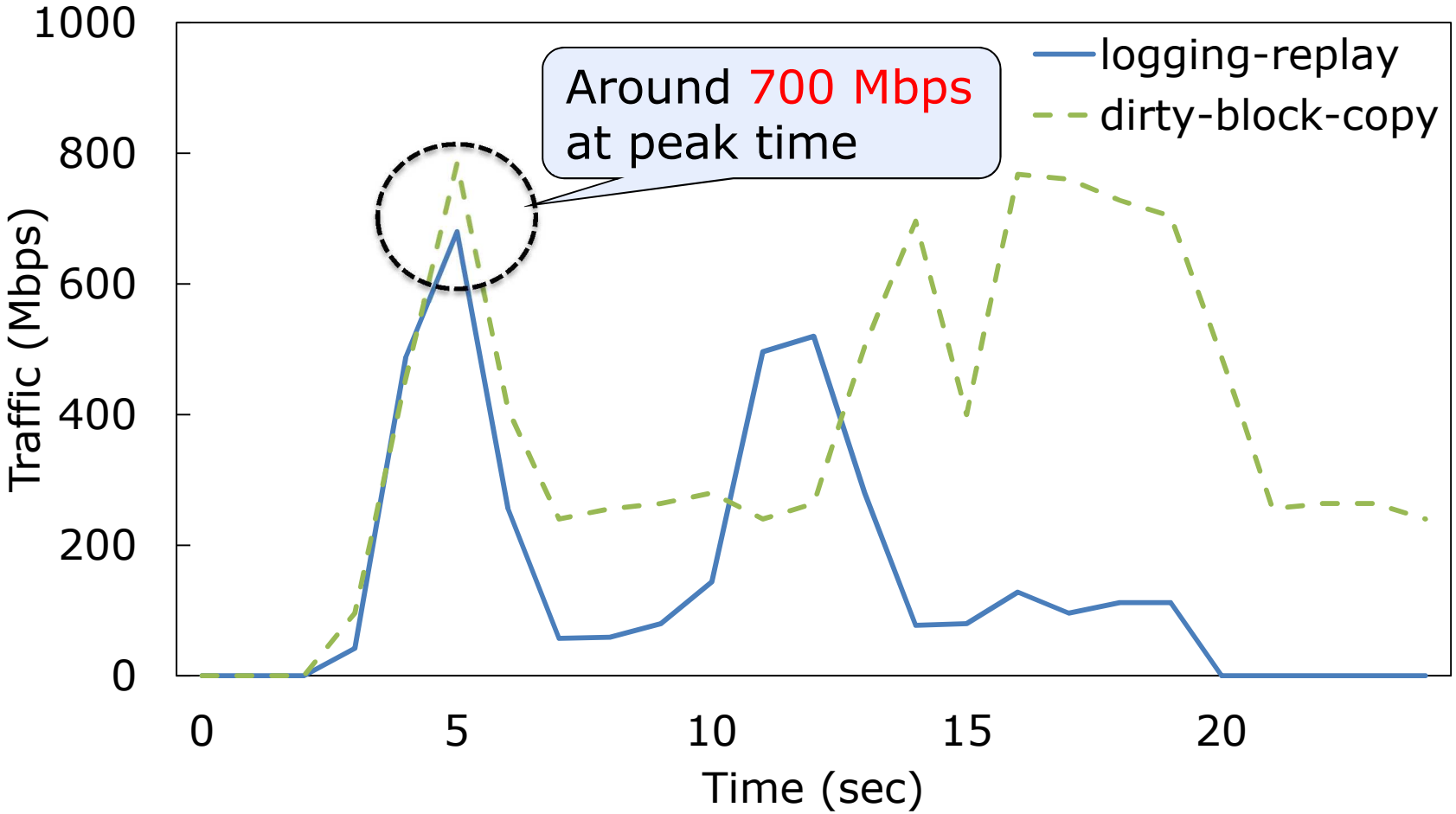
- Experiment items
  - Traffic needed to synchronize VMs and disks
  - Performance of the primary VM (File I/O) using IOzone
- Techniques implemented in Kemari:
  - logging-replay
  - dirty-block-copy
    - Transferring dirty blocks to secondary VM
- Experimental environment
  - Hardware specs:
    - CPU: Quad-core Intel Xeon 2.6GHz X 2
    - Network: 1 GbE
  - VM specs:
    - KVM: Linux 2.6.33
    - QEMU: qemu-0.14.0
    - Guest OS: Debian lenny w/ virtio-blk
    - Memory: 1 GB

Traffic



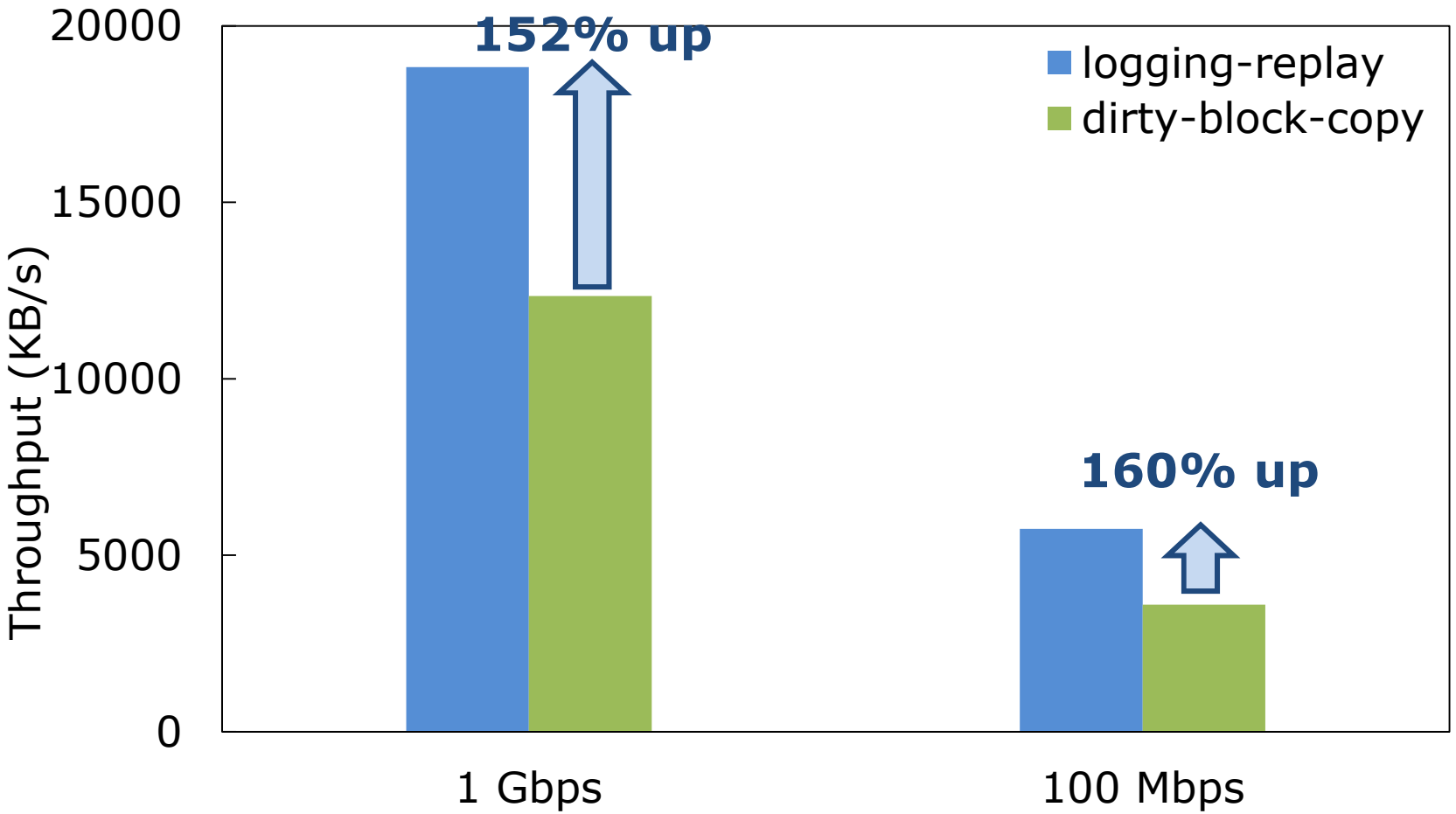
- Logging-replay reduces the traffic needed to synchronize only disks by 99%

Synchronized NW traffic (1 Gbps)



- Logging-replay reduces average traffic by 50%
- Traffic is as high as around 700 Mbps
  - Feature of event-driven synchronization

File I/O for both bandwidths



- Good performance obtained at both bandwidths, especially at low bandwidth

Summary

- A rapid VM synchronization mechanism is being developed.

- Logging-replay was implemented in Kemari.
  - Traffic/sync reduced by 50%
  - Throughput increased by 160%

Future work

- Addressing the problem of disk failure propagation
  - Disk read errors propagate to secondary VM when primary VM's disk crashes.
  - In other failure cases, we can verify disk consistency.
- Implementing memory replication with logging-replay
  - Enabling secondary VM to replay disk read events produced by the primary VM
- Implementing live migration with logging-replay to improve performance