# Fast Live Migration for Data-intensive VMs by Exploiting Storage Area Network in Datacenter

LinuxCon Japan 2014, May 21st

Soramichi Akiyama

*The University of Tokyo*

# Myself

- Name
  - Soramichi Akiyama (穐山 空道)
- Contact
  - akiyama@nii.ac.jp
  - http://www.soramichi.jp/
- Education
  - March 2010 B.Eng. from Kyoto Univ
  - March 2012 M.Sc. from Univ of Tokyo
  - April 2012-now Ph.D. candidate at Univ of Tokyo
- Research
  - Cloud datacenter optimization, virtualization

# Background

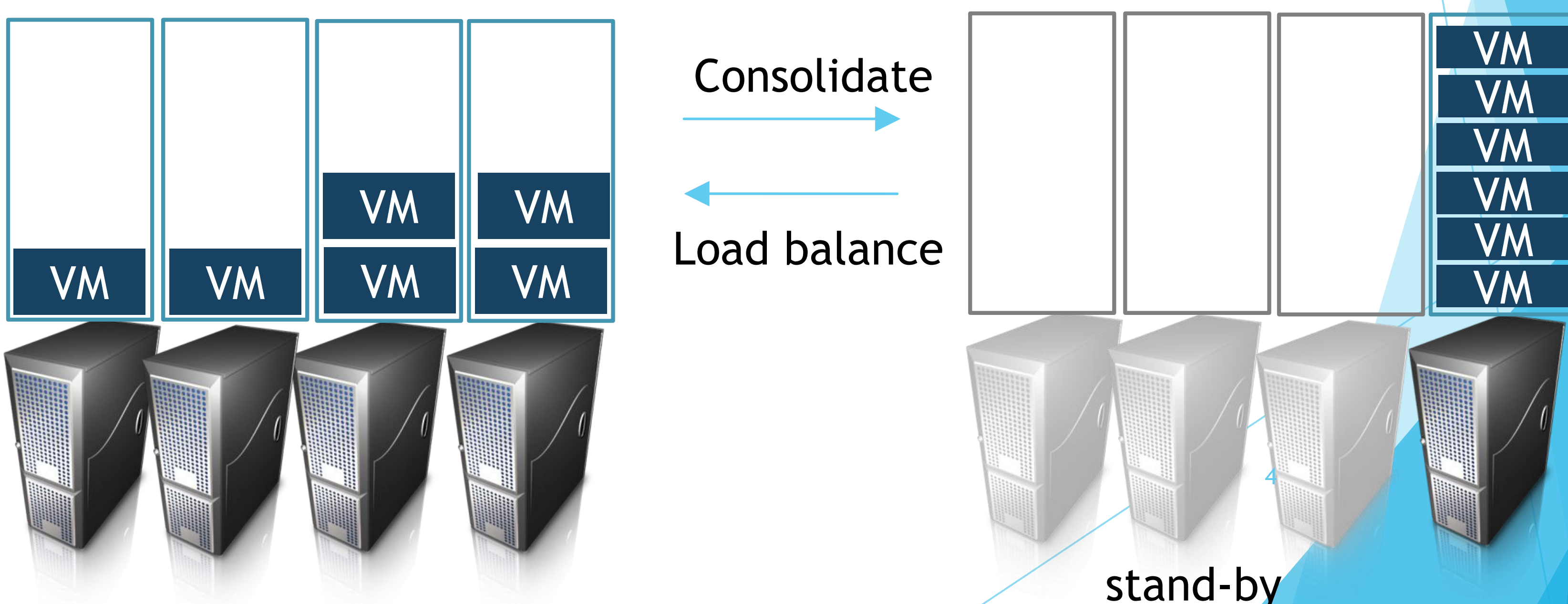
- Cloud computing has become common both for enterprise and consumers

amazon web services™ Microsoft Azure heroku

- Datacenters consume much energy
  - 1.5 % of the total energy consumed in the US was for datacenters in 2006 (“EPA Report on Server and Data Center Energy Efficiency”, 2007)
- Energy efficiency of datacenters is of great concern

# Dynamic VM consolidation

- Reduce datacenter energy consumption by consolidating idle VMs
- VMs are moved with *live migration* technique



# Live Migration

- Migrate a VM between servers without interrupting the services running on the VM
- Mandatory for dynamic VM consolidation

- What to share btw source & destination servers
  1. VM Memory
  2. Disk Image
  3. Device States

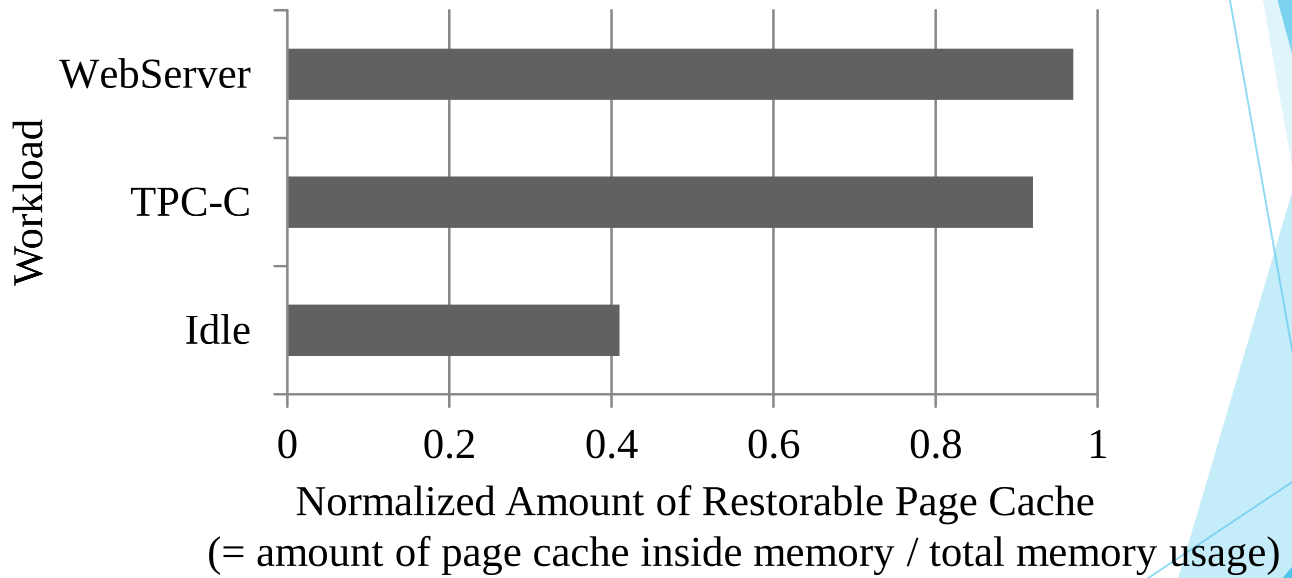
# Live Migration

- Migrate a VM between servers without interrupting the services running on the VM
- Mandatory for dynamic VM consolidation

- What to share btw source & destination servers
  1. VM Memory **Main focus**
  2. Disk Image No need to transfer when shared storage is applicable
  3. Device States Not a problem as they're pretty small

# Data-intensive VMs

- A data-intensive VM (e.g. web server VM, DB VM) has much page cache in its memory
  - *Page cache: on-memory cache of storage data*



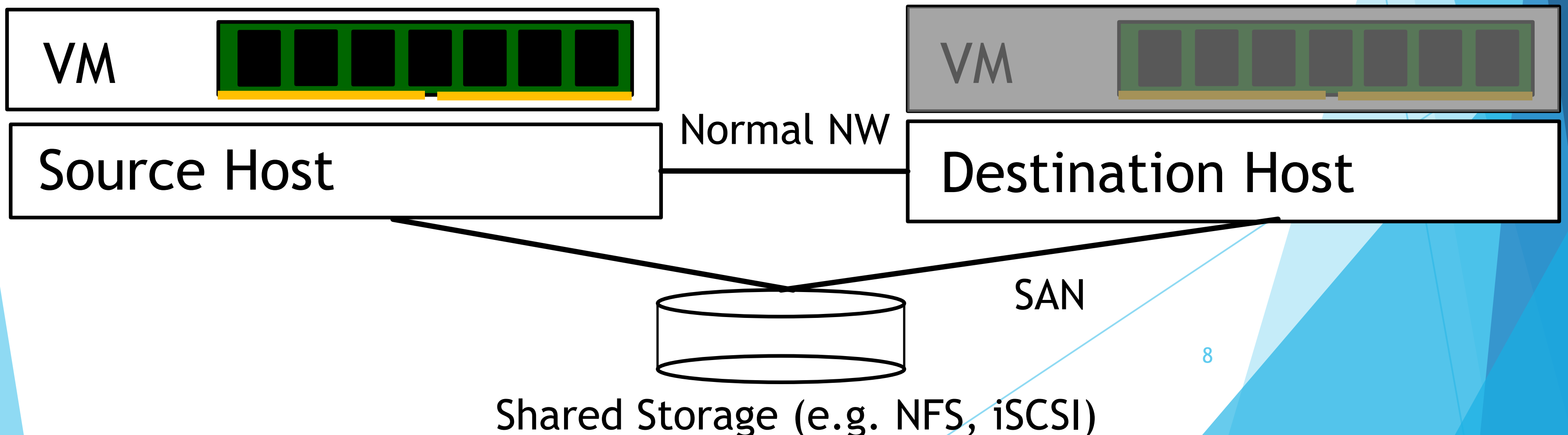

- This prevents quick migration/consolidation

# Page Cache Teleportation: Overview

- Base Idea

Restorable page cache can be transferred via SAN

- Datacenter networking is fully utilized for migration

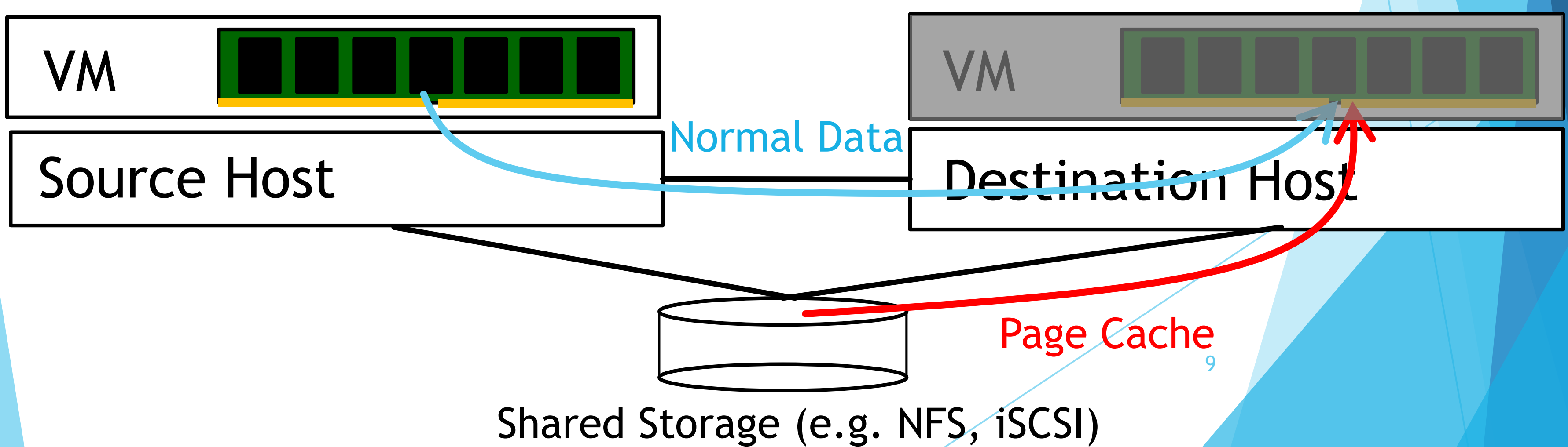


# Page Cache Teleportation: Overview

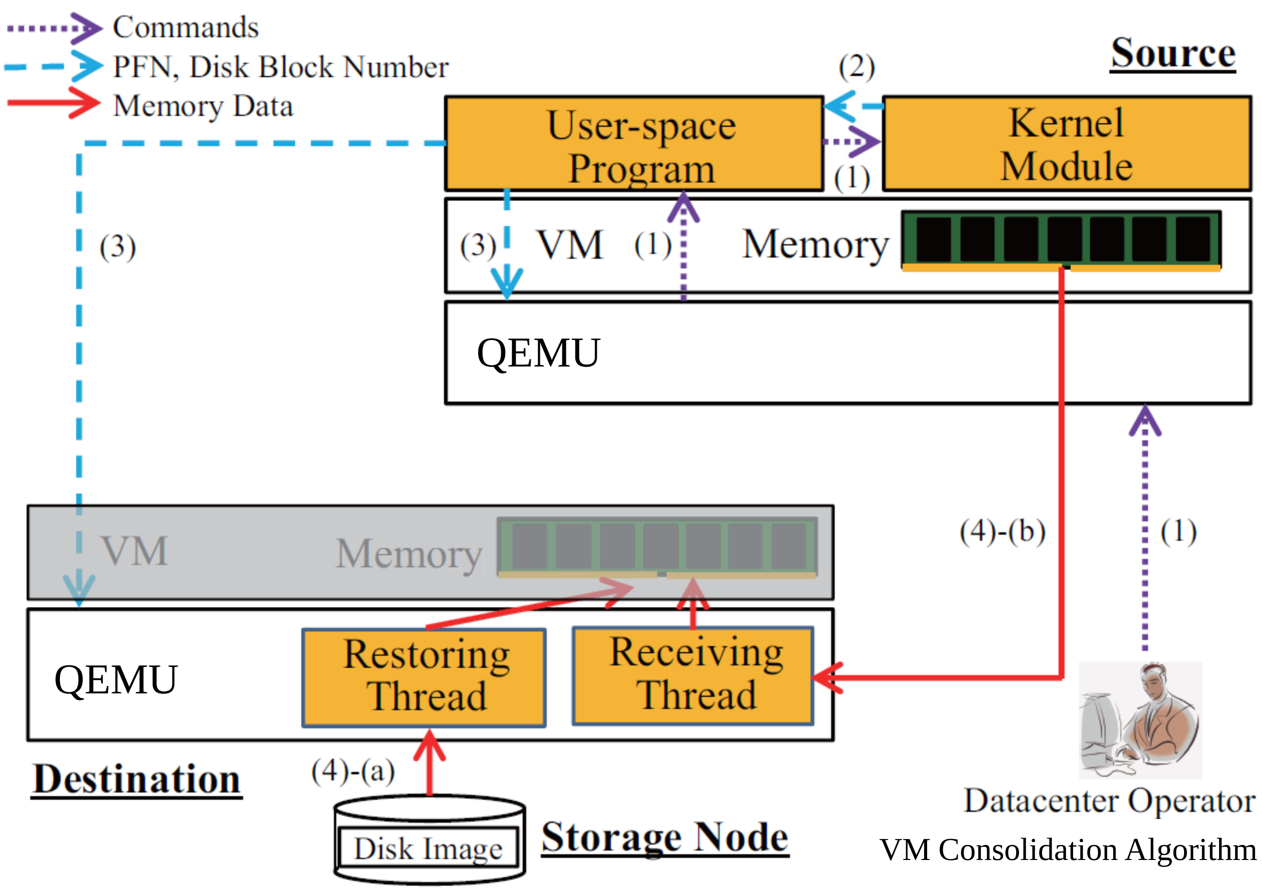
- Base Idea

Restorable page cache can be transferred via SAN

- Datacenter networking is fully utilized for migration



# How the System Works



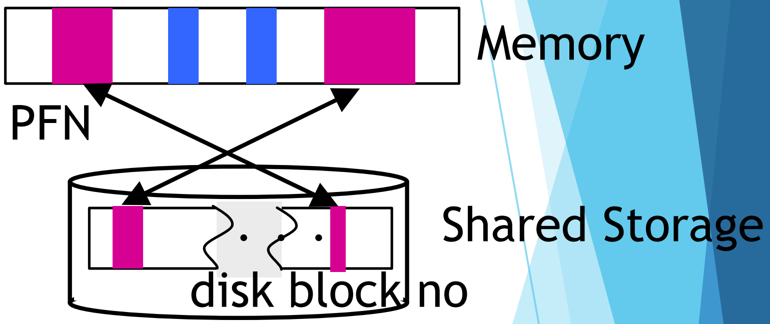

(1) Migration requested
(2) PFN-disk block mapping of restorable page cache detected
(3) The mapping informed to source & destination VMMs
(4) Restorable page cache (a) and normal data (b) transferred in parallel

# Detect Restorable Page Cache

- A new kernel module is installed into the guest to detect restorable page cache
  - page frame number (PFN)

    -(pfn_to_page)-> struct page

    -(bmap)-> disk block number

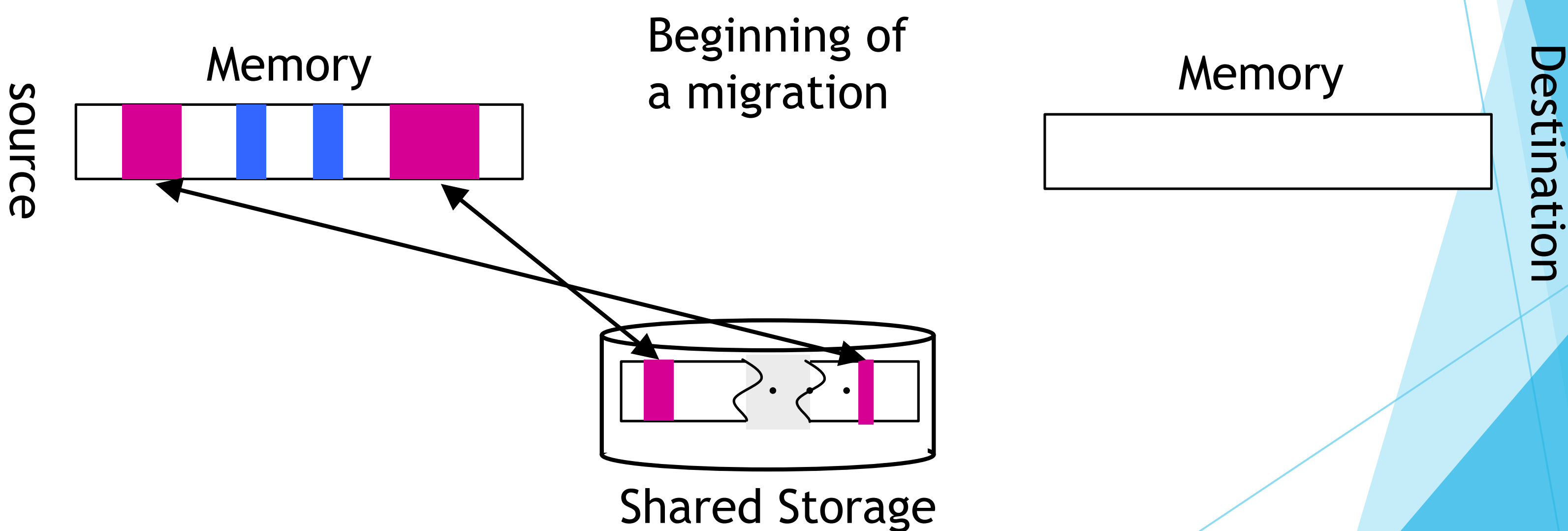



| | Kernel Module (our proposal) | Introspection | Disk IO-Monitoring |
|---|---|---|---|
| Implementation | Easy (<200 loc) | Hard | Middle |
| Runtime overhead | None | None | Disk write hooked |
| Migration overhead | Small (<1 sec) | Big (binary scan) | Small |
| Guest modification | Yes | No | No |

Comparison between other possible impl methods

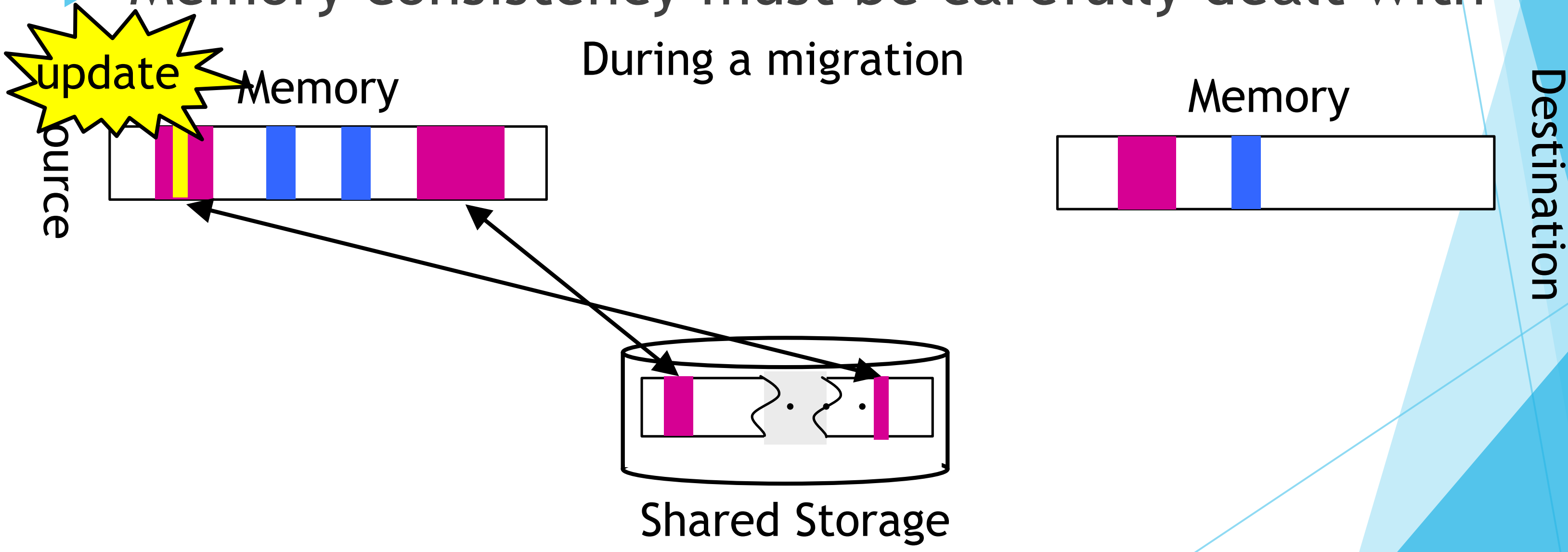
# Memory Consistency (1/4)

- Page cache can be updated during a migration and no guarantee to be flushed
- Memory consistency must be carefully dealt with



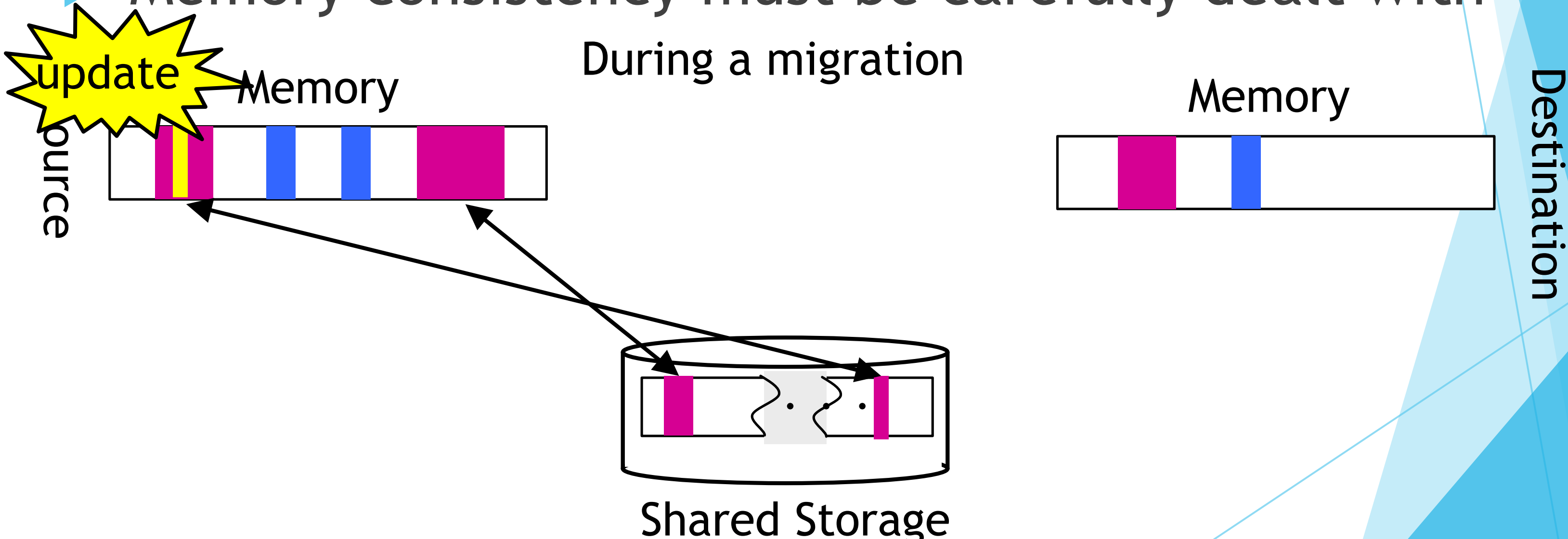
# Memory Consistency (2/4)

- Page cache can be updated during a migration and no guarantee to be flushed
- Memory consistency must be carefully dealt with



# Memory Consistency (3/4)

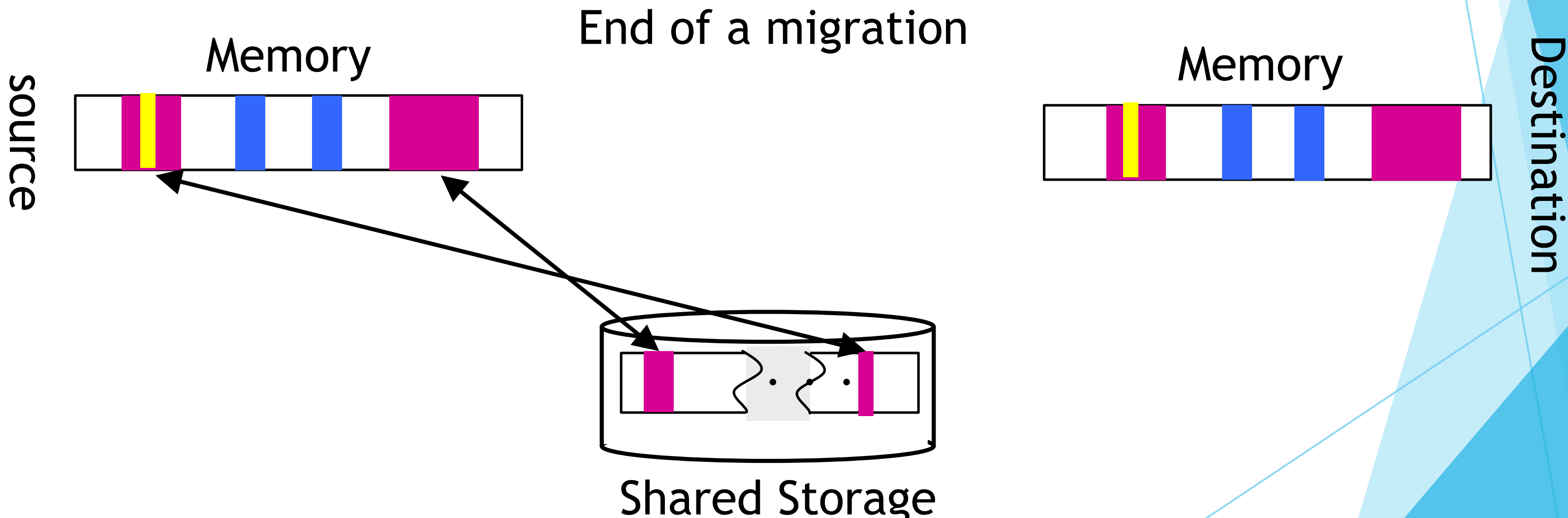
- Page cache can be updated during a migration and no guarantee to be flushed
- Memory consistency must be carefully dealt with




Write access is monitored by dirty page tracking
→ Updated page cache is transferred via normal NW

# Memory Consistency (4/4)

- Page cache can be updated during a migration and no guarantee to be flushed
- Memory consistency must be carefully dealt with

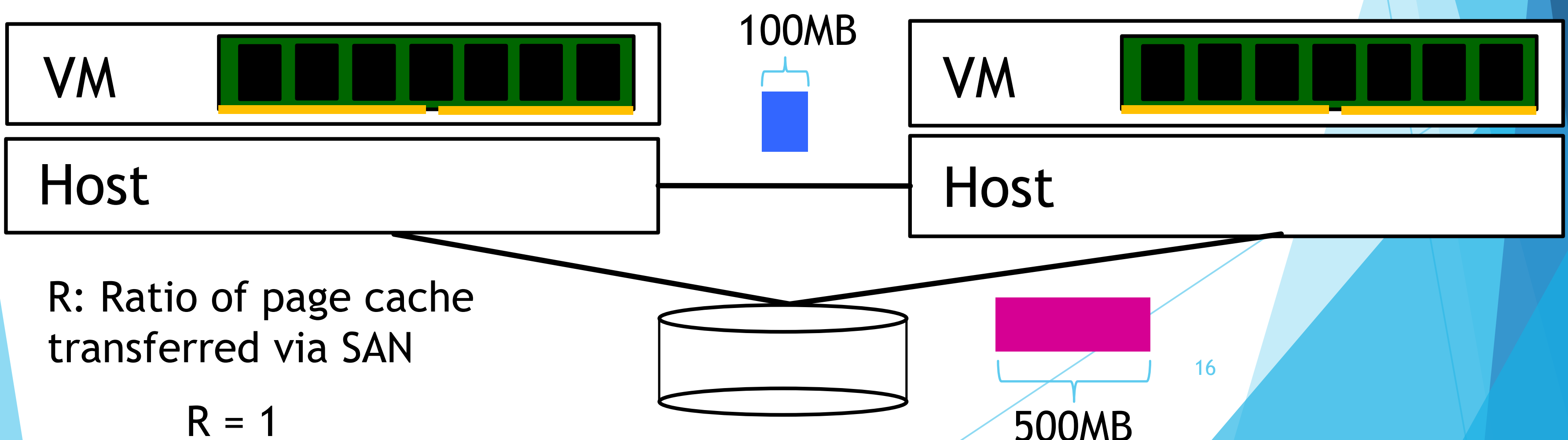



Write access is monitored by dirty page tracking
→ Updated page cache is transferred via normal NW

# Dividing Page Cache (1/2)

- Naïve transfer cannot fully utilize DC networking
- Example:
  - Page cache 500MB, Normal data 100MB

→ Transferring page cache takes 5X than transferring normal data

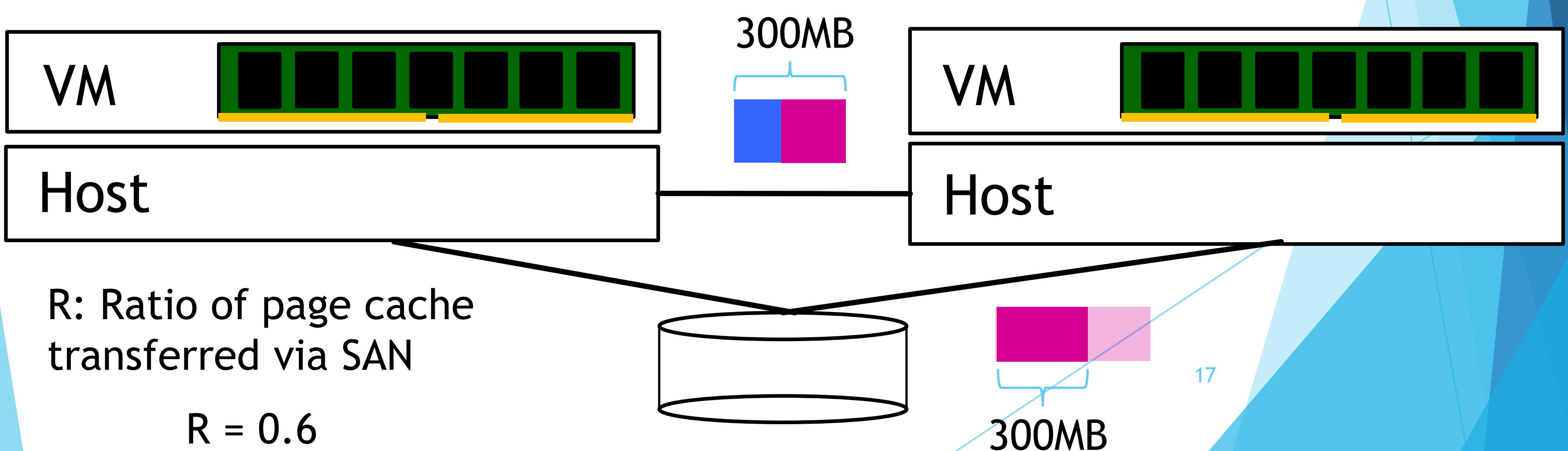



R: Ratio of page cache transferred via SAN

R = 1

# Dividing Page Cache (2/2)

- A portion of page cache must be transferred via normal network
- "Fully utilize" means: Two networks are used during the same length of time



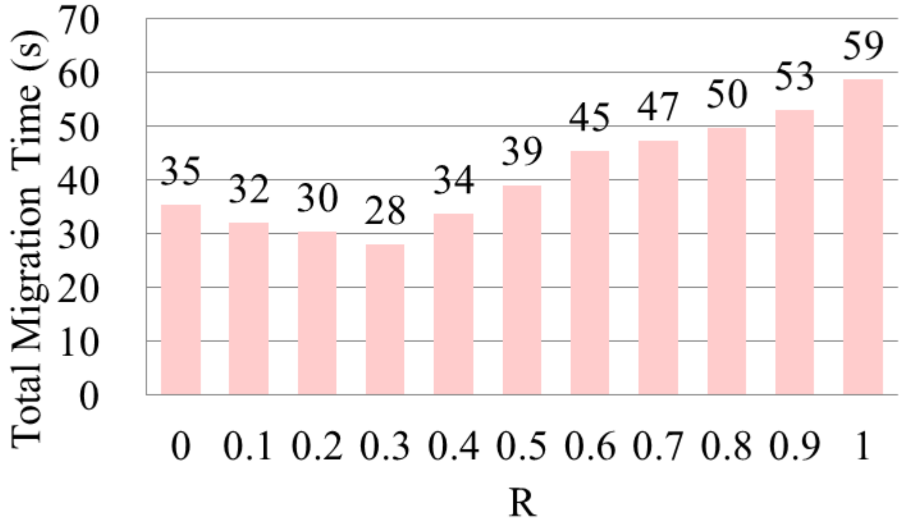

R: Ratio of page cache transferred via SAN

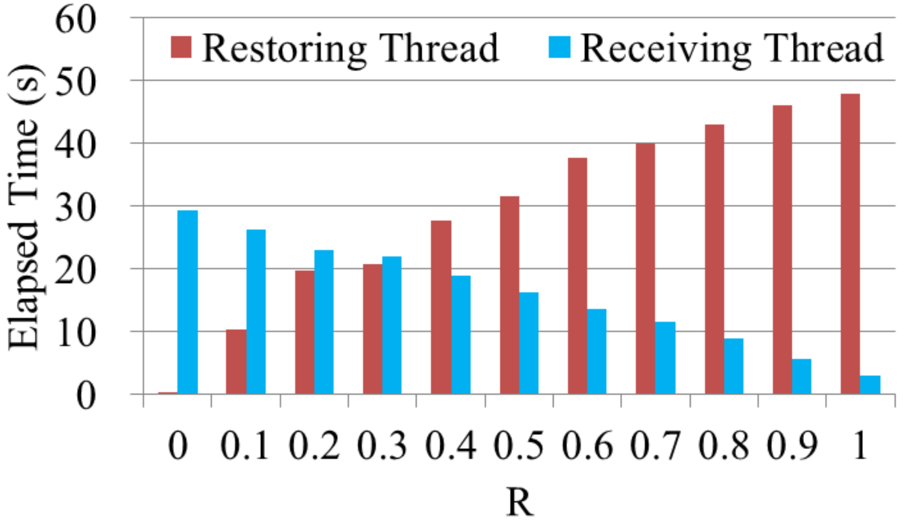
R = 0.6

# Experimental Settings

- Total migration time with Page Cache Teleportation v.s. non-optimized migration
- Workloads
  1. ***Web server***: VM hosts Apache web server with static web pages, accessed with 120 Mbps
  2. ***Database***: VM hosts MySQL database, accessed by a TPC-C load generator (typical web shopping queries)
- Machines
  - ***Host***: Debian Squeeze (kernel 2.6.32), Xeon X5460, 8 GB Mem, QEMU/KVM 0.13.0 (not 1.3.0 ☹)
  - ***Guest***: Debian Squeeze, 1 vCPU, 4 GB Mem

# Migrating Web-server VM

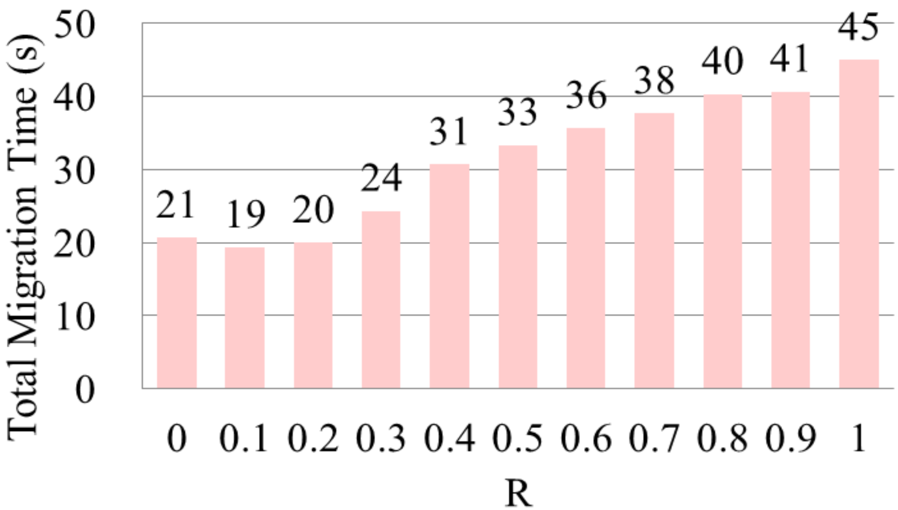


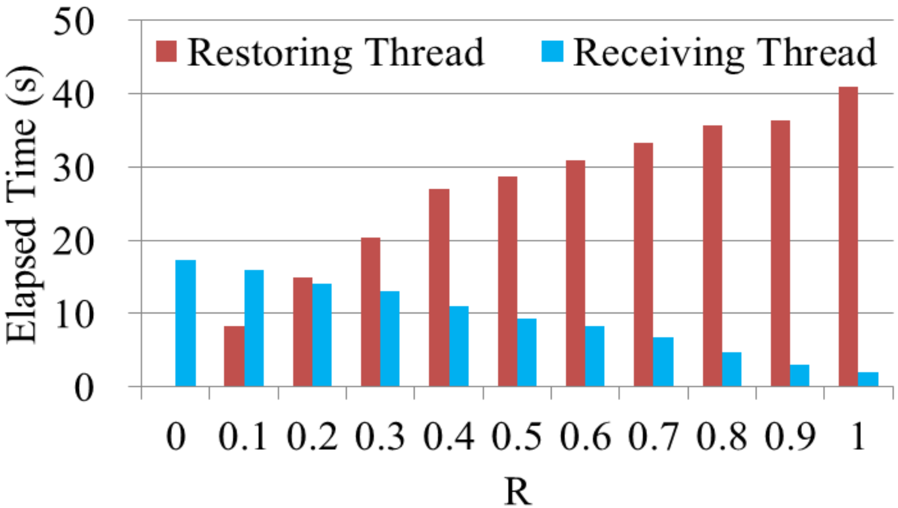




R = 0 means SAN is not used
(equiv to non-optimized migration)

- Web Data: 3GB
- 1Gbps Networks
- 35 sec → 28 sec
- Shortest migration time got when R=0.3 (both networks are fully utilized)

# Migrating Database VM



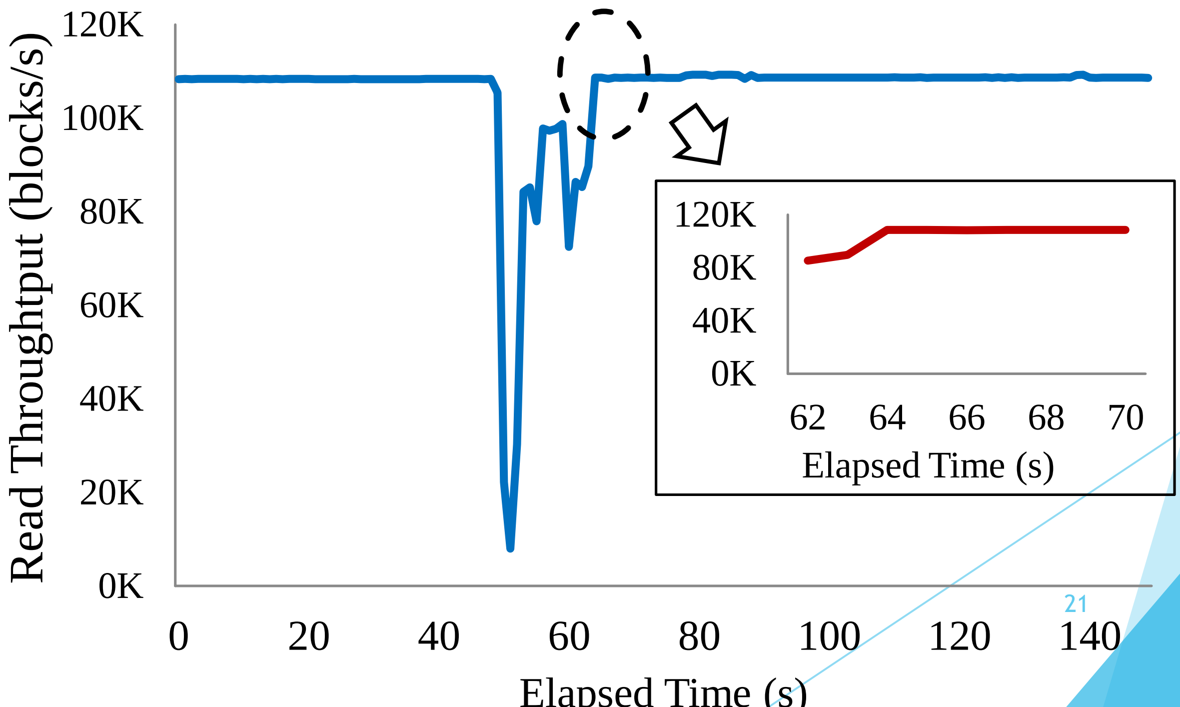




R = 0 means SAN is not used
(equiv to non-optimized migration)

- DB Data: 1.9GB
- 500Mbps Networks
- 21 sec → 19 sec
- Page cache is scattered across wide range of disk
  → Copying it out is really slow even with SSD

# IO Performance Penalty

- VM reading a large file stored in page cache is migrated with our method during 40<t<62
- Throughput recovers immediately after migration finishes (t=62)



# Related Work

- Some work **skip** transferring page cache to accelerate migration
  - Hines *et al.*, VEE'09
  - Koto *et al.*, Apsys'12
  - Page cache loss makes workload slow down
- Jo et al. propose to use the SAN like our method, but they do not divide page cache
  - C. Jo *et al.*, VEE'13
  - C. Jo et al., CloudCom'13

# Conclusion

- Summary
  - Dynamic VM consolidation is a key to improve datacenter energy efficiency
  - Live migration of data-intensive VMs are accelerated by exploiting SAN to transfer page cache efficiently
- Future Work
  - Omit parameter R (bad Rs yield worse results than non-optimized) → almost done, to be evaluated thoroughly
  - Evaluate total energy reduction by our proposal

    → ongoing using modified SimGrid