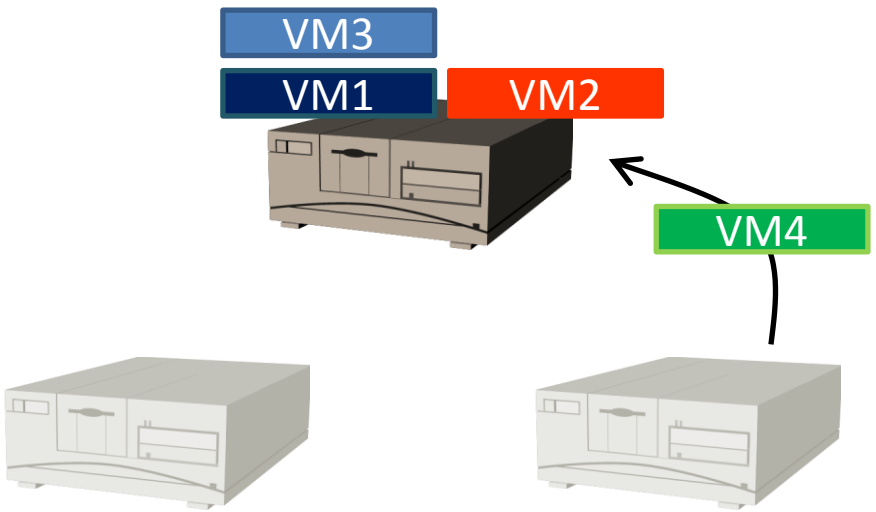
Acceleration of Virtual Machine Live Migration on QEMU/KVM by Reusing VM Memory

Soramichi Akiyama

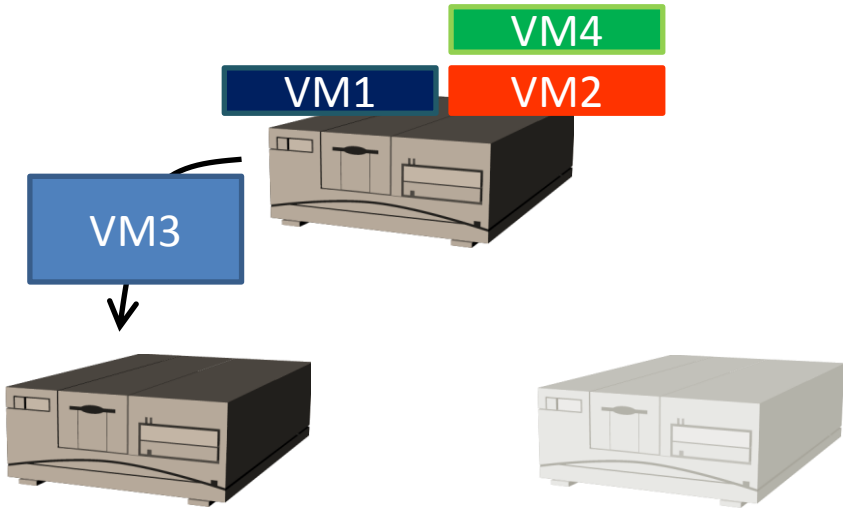

Department of Creative Informatics
Graduate School of Information Science and Technology
The University of Tokyo, Japan

Resource provisioning in datacenters with virtualization technologies

consolidation / distribution of Virtual Machines using Live Migration technique

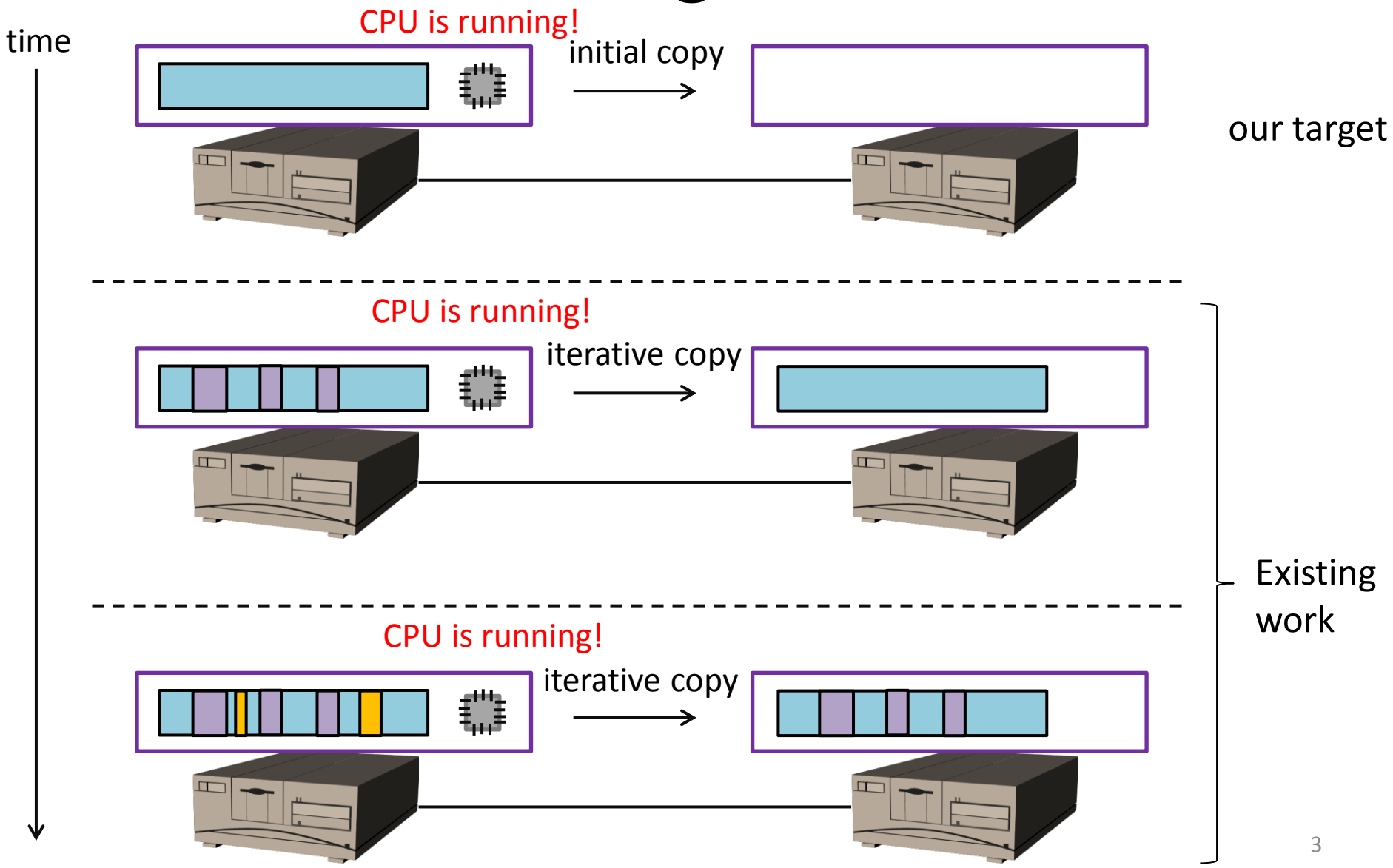


VM4 migrates for consolidation



VM3 migrates to guarantee performance

How Live Migration works

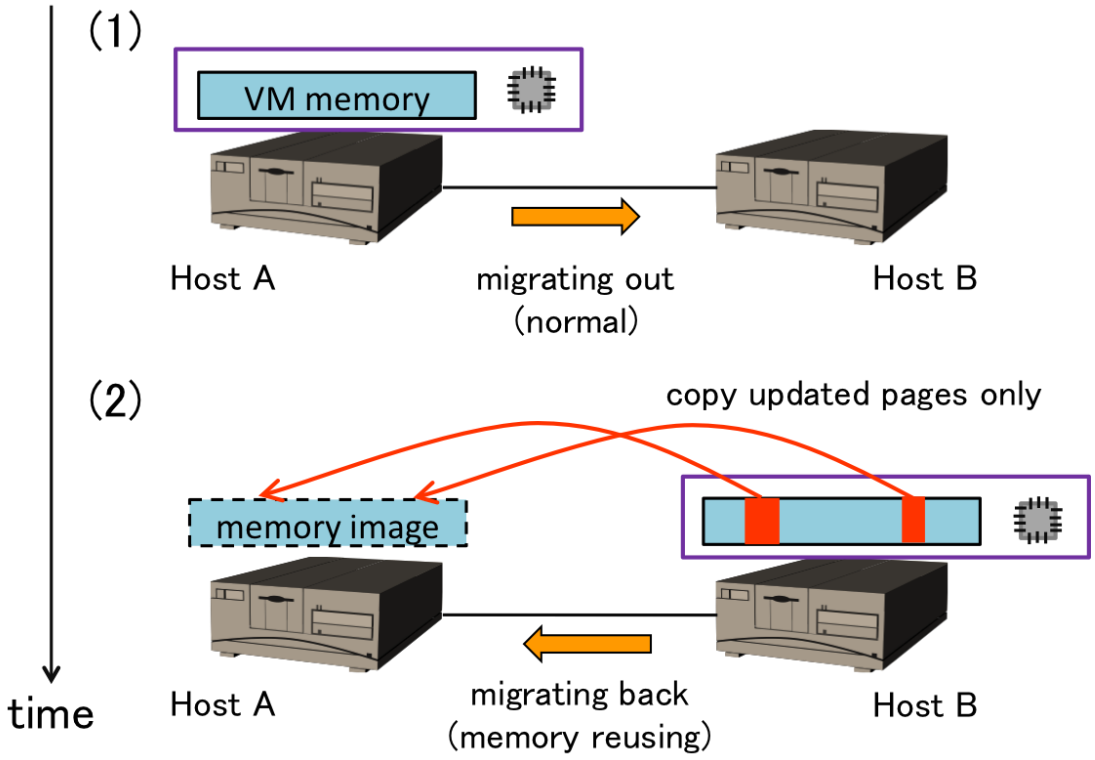


The idea: Memory Reusing

Dynamic consolidation

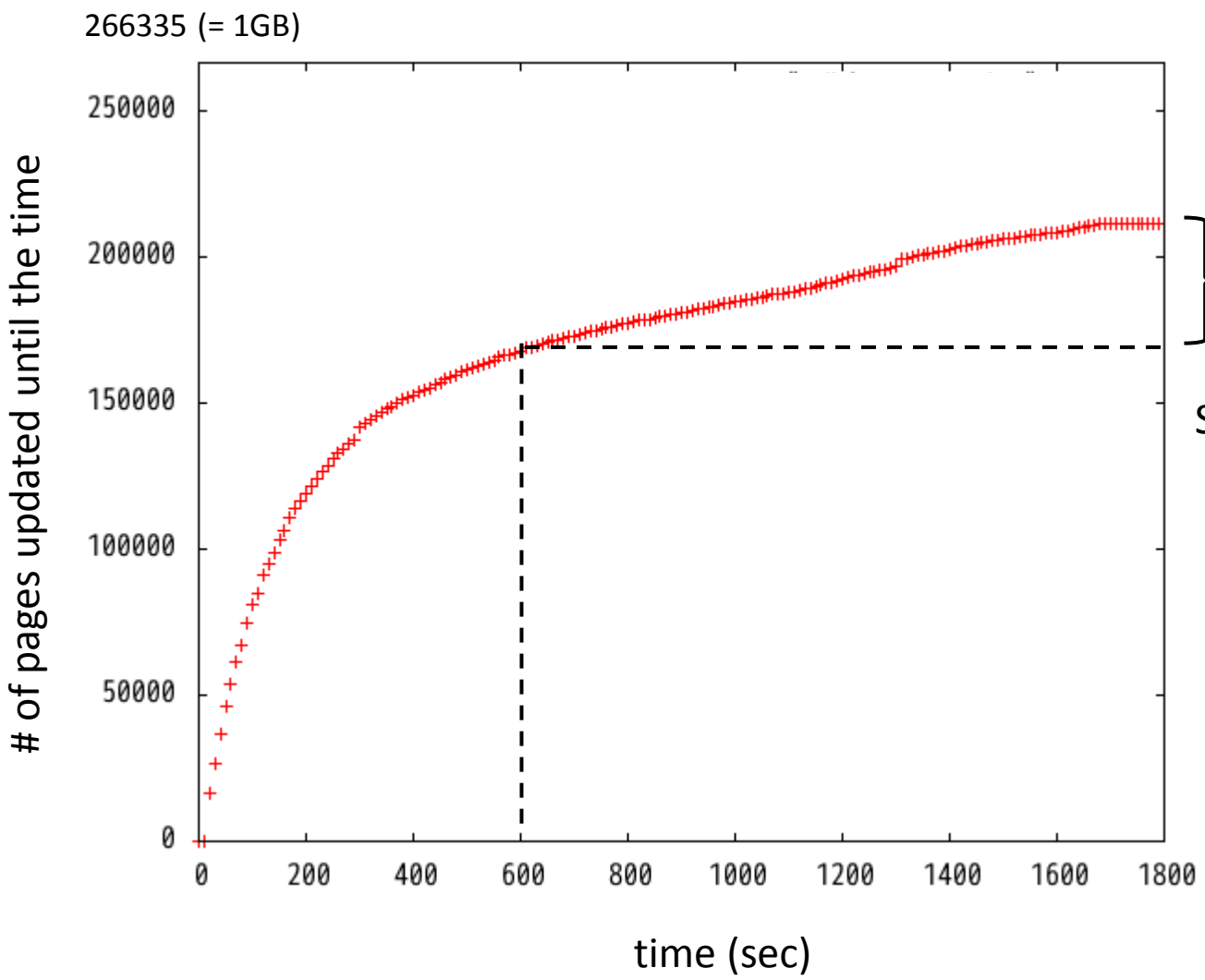
1. VMs migrate many times
2. a VM will come back later on

→ Keep the memory image on hosts and reuse it!



Memory Access Pattern of TPC-C

simulated DB access of a net-shop

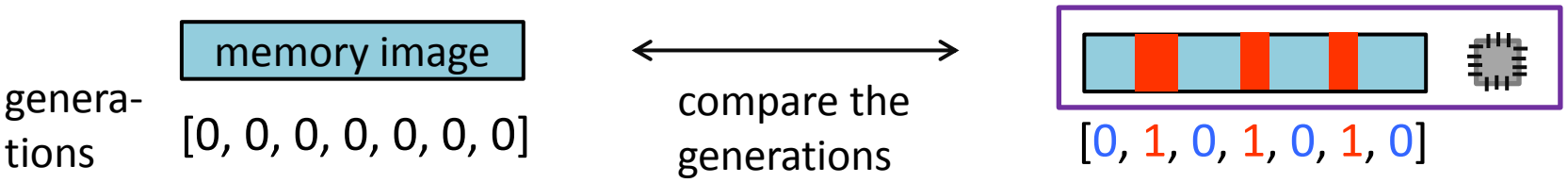


Suppose:
migrate out at 0 sec and
migrate back at 600 sec

→ 17% of pages can be reused!

The algorithm

1. Each memory page has a *generation*
   - 0 in the boot-time
   - Migration: +1 generations of updated pages
2. Keep the memory image of the VM on the source host in a migration
3. Compare latest generations and ones of the kept memory image when migrating back
   - Pages with the same generation: reused

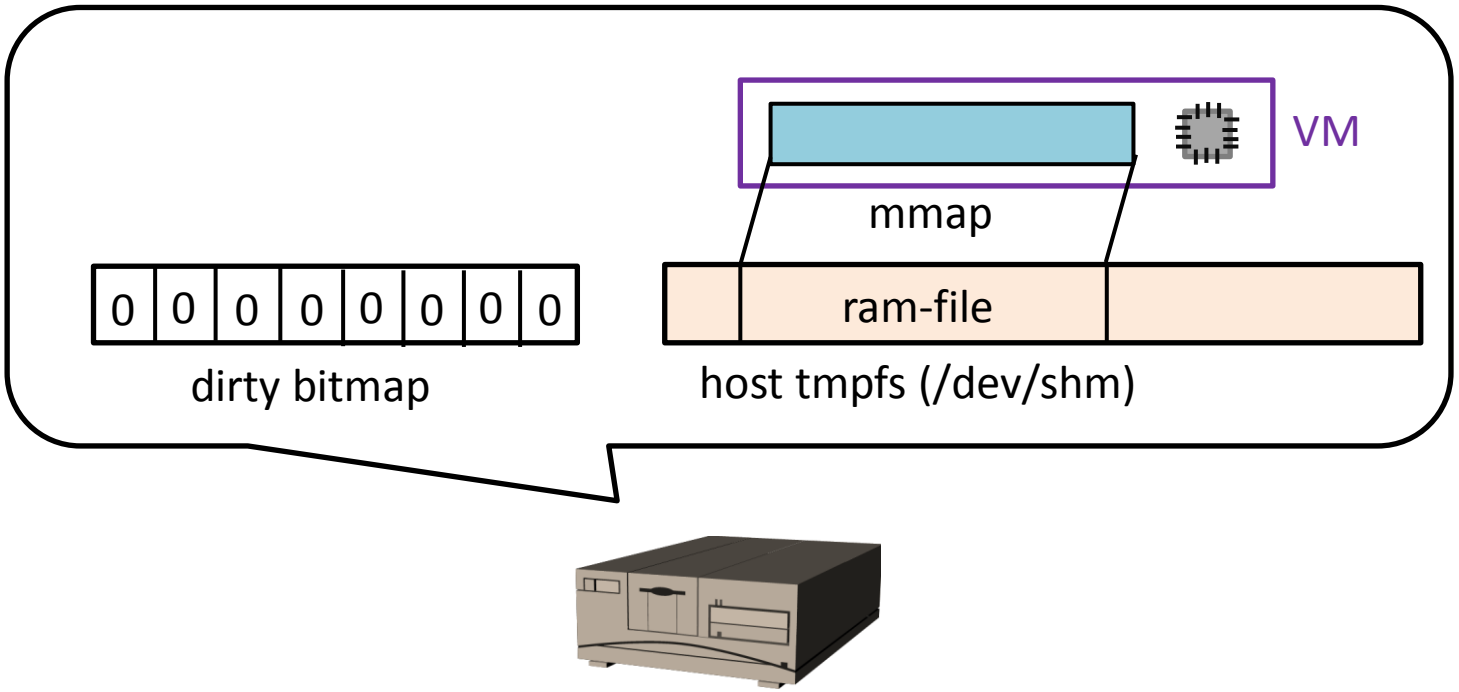


Implementation

- QEMU 0.13.0 + KVM on Linux
  - original: transfer the whole memory
  - modified: transfer updated pages only
- A central server manages the generations
  - gathered generations can be used to optimize VM placements

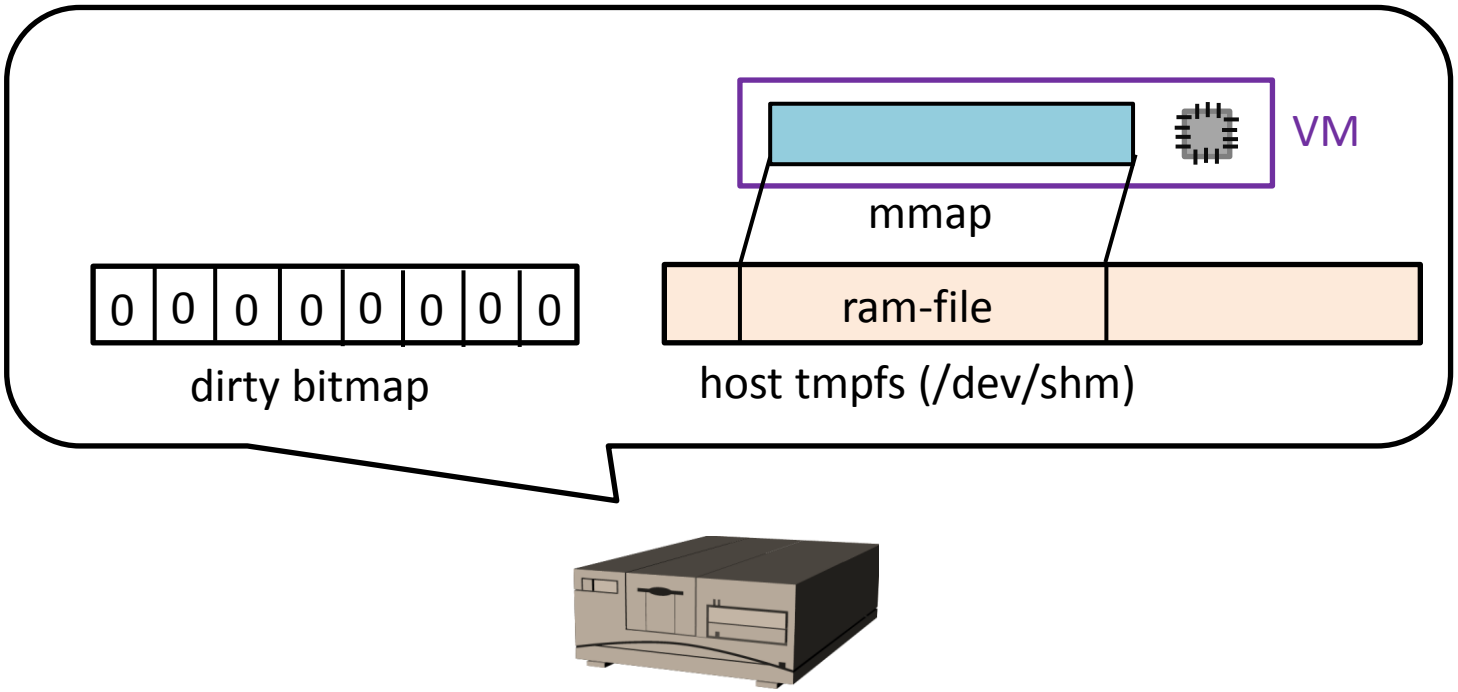
Modification to QEMU

1. mmap the VM memory with a ram-file in /dev/shm of the host (for later reuse)



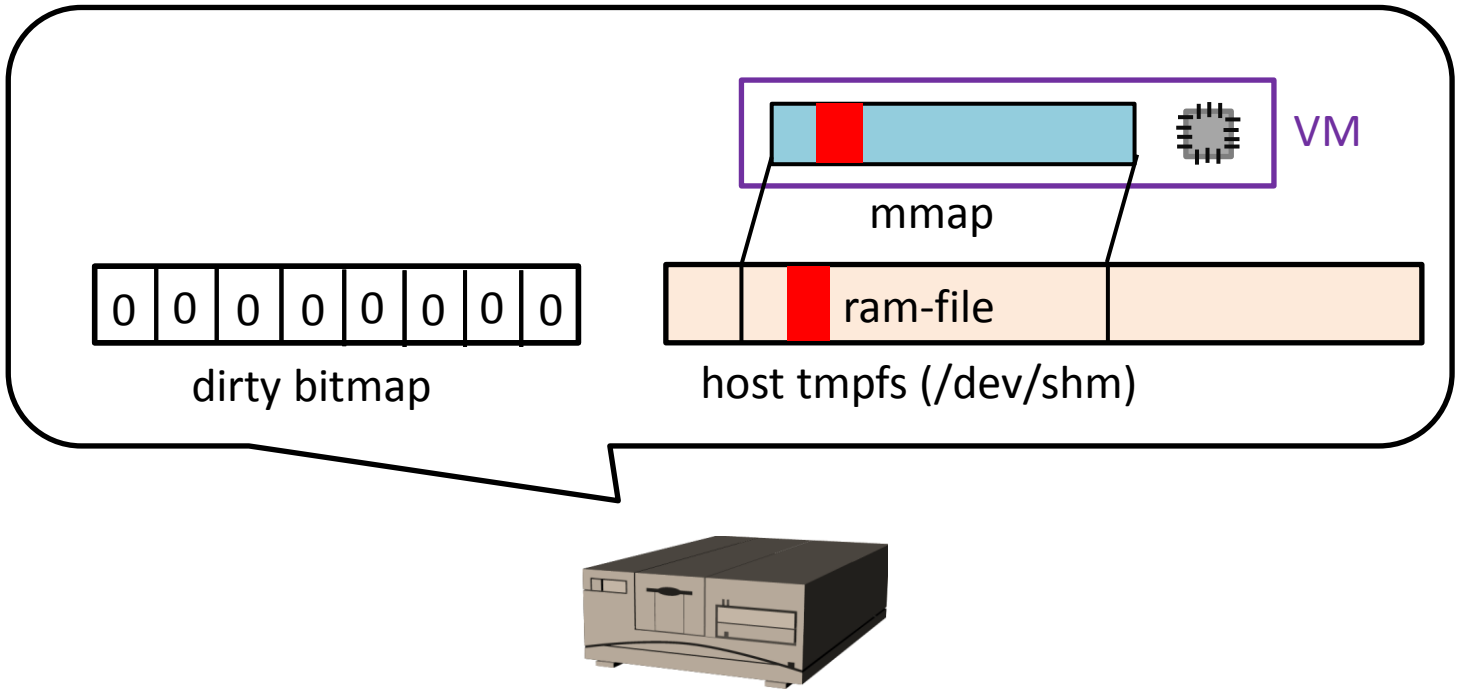
Modification to QEMU

2. use “dirty page tracking” to detect updates



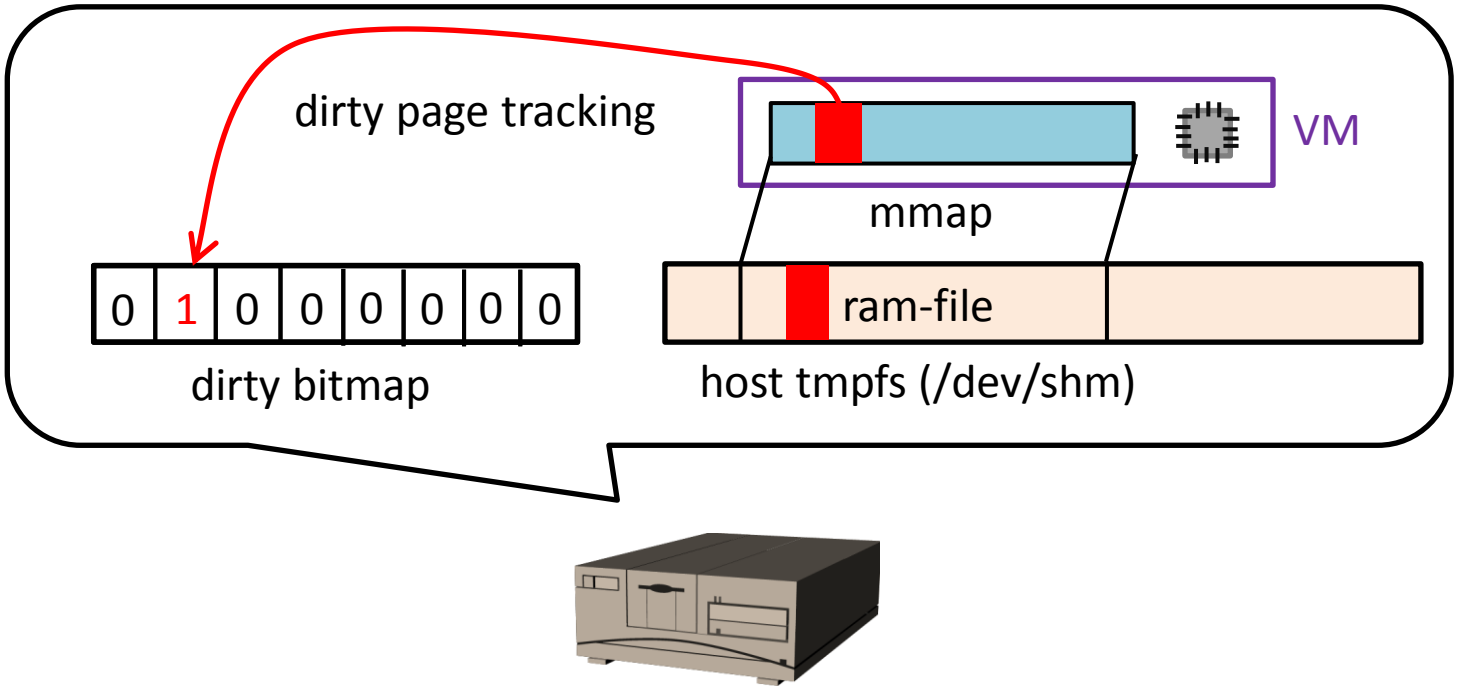
Modification to QEMU

2. use “dirty page tracking” to detect updates

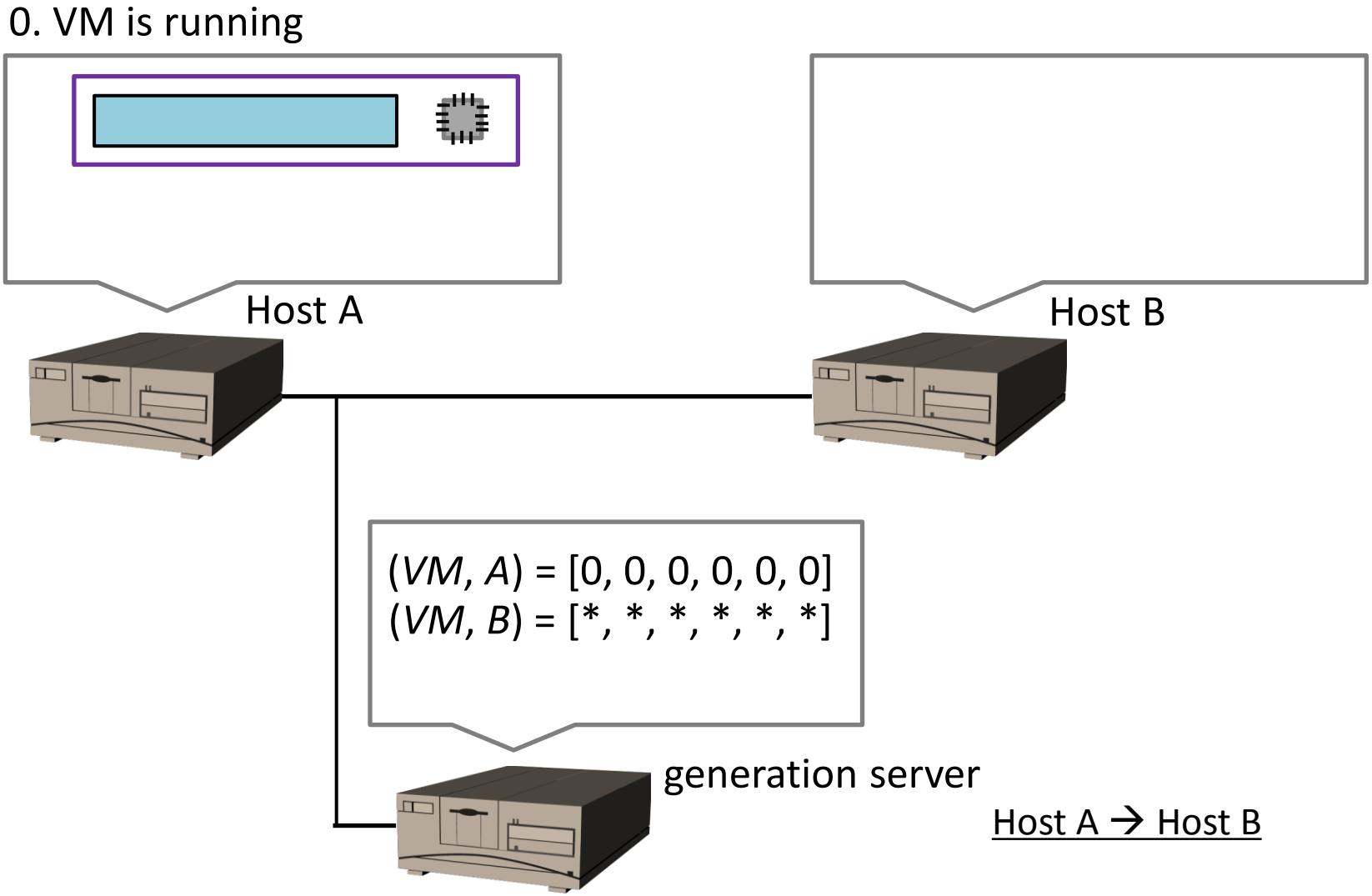


Modification to QEMU

2. use “dirty page tracking” to detect updates

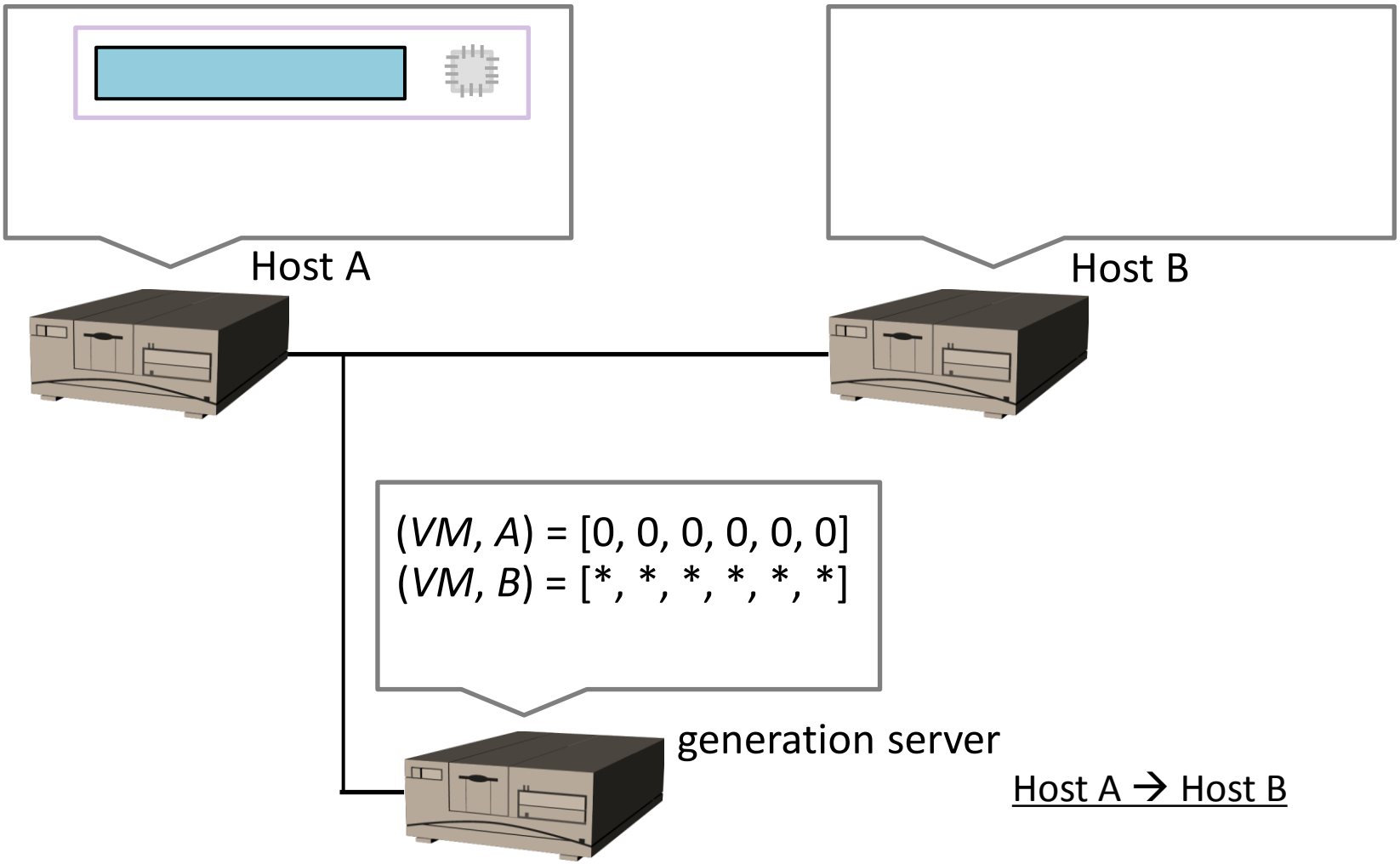


How the system works: the first migration

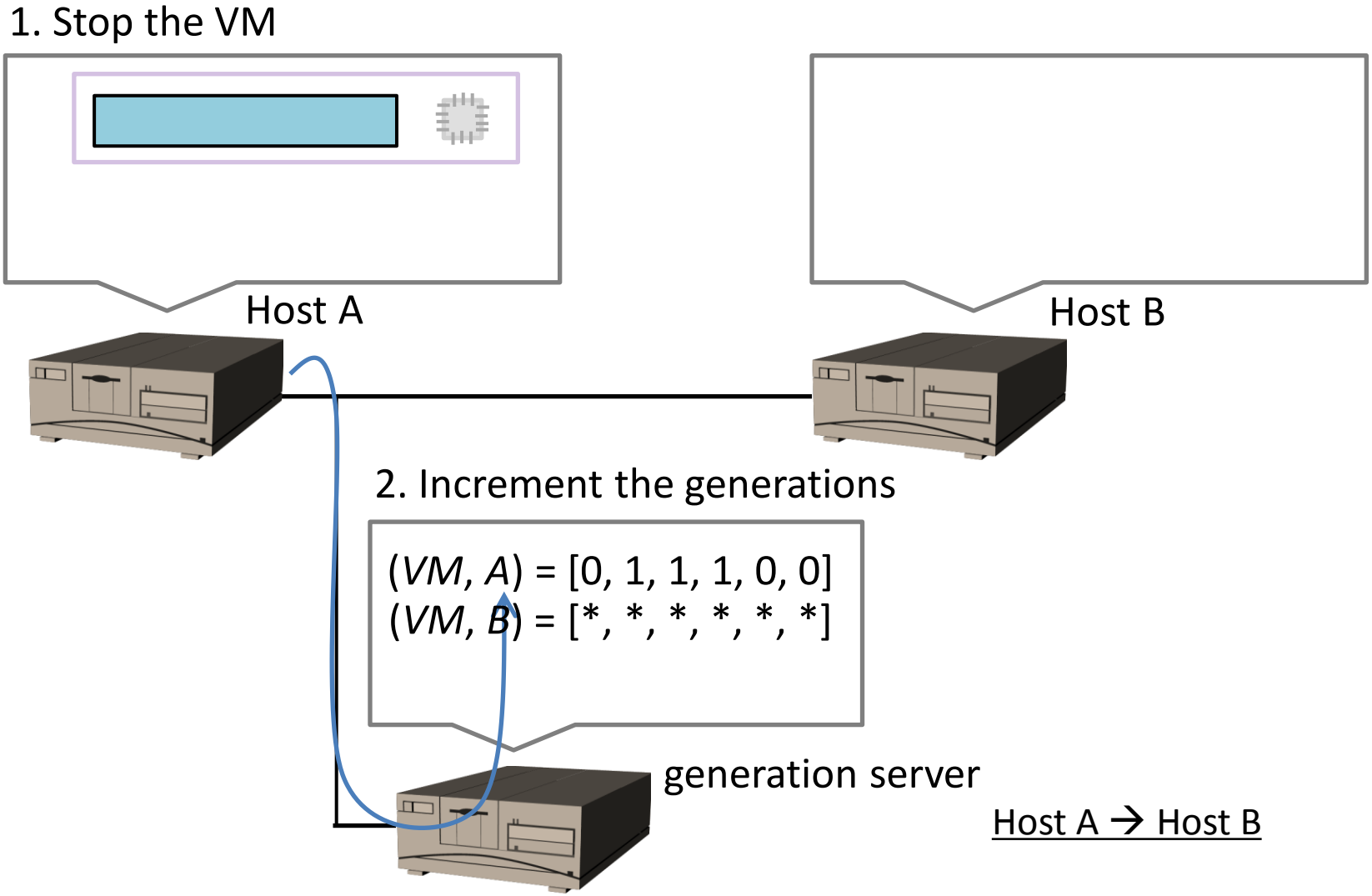


How the system works: the first migration

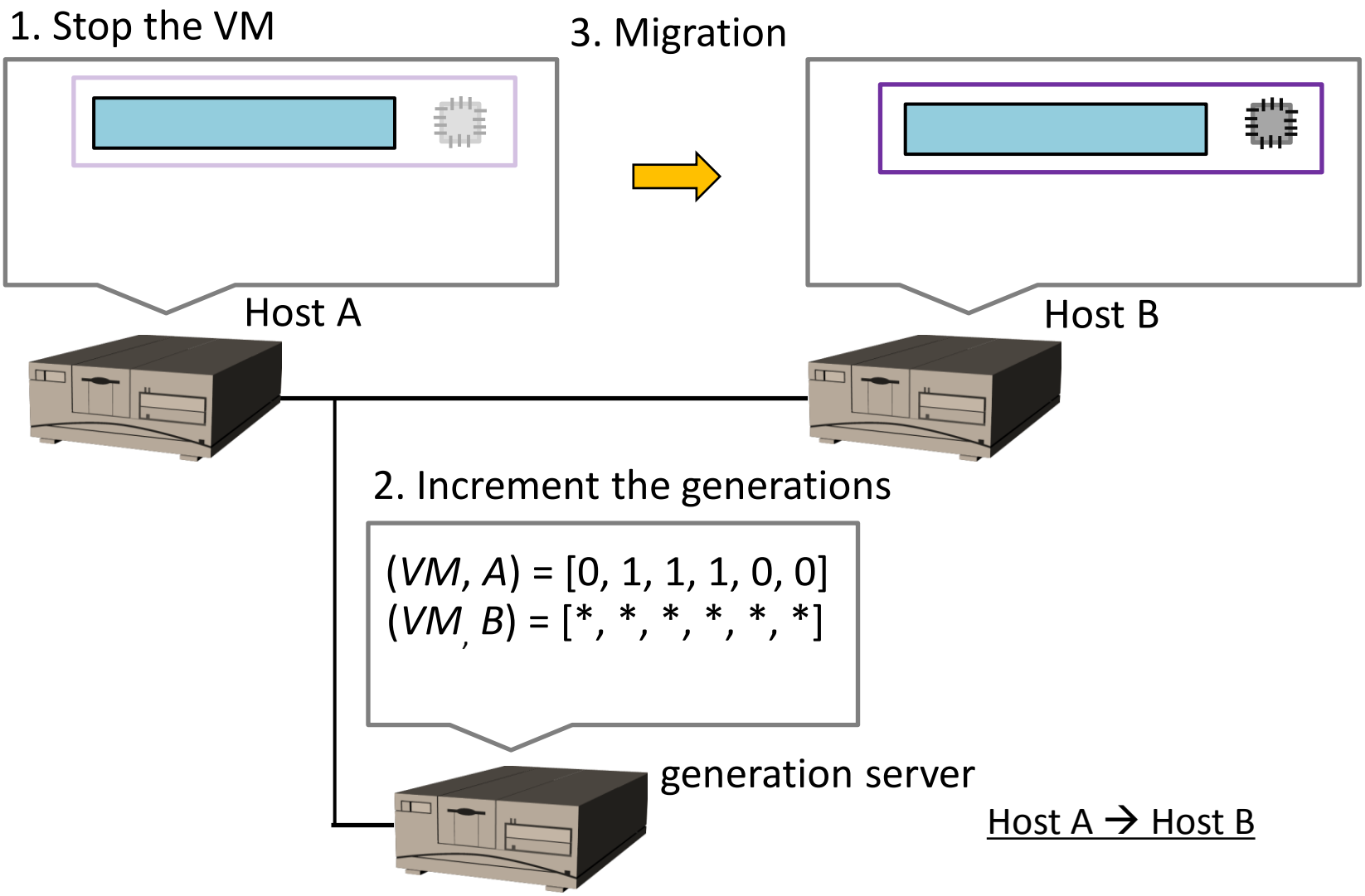
1. Stop the VM



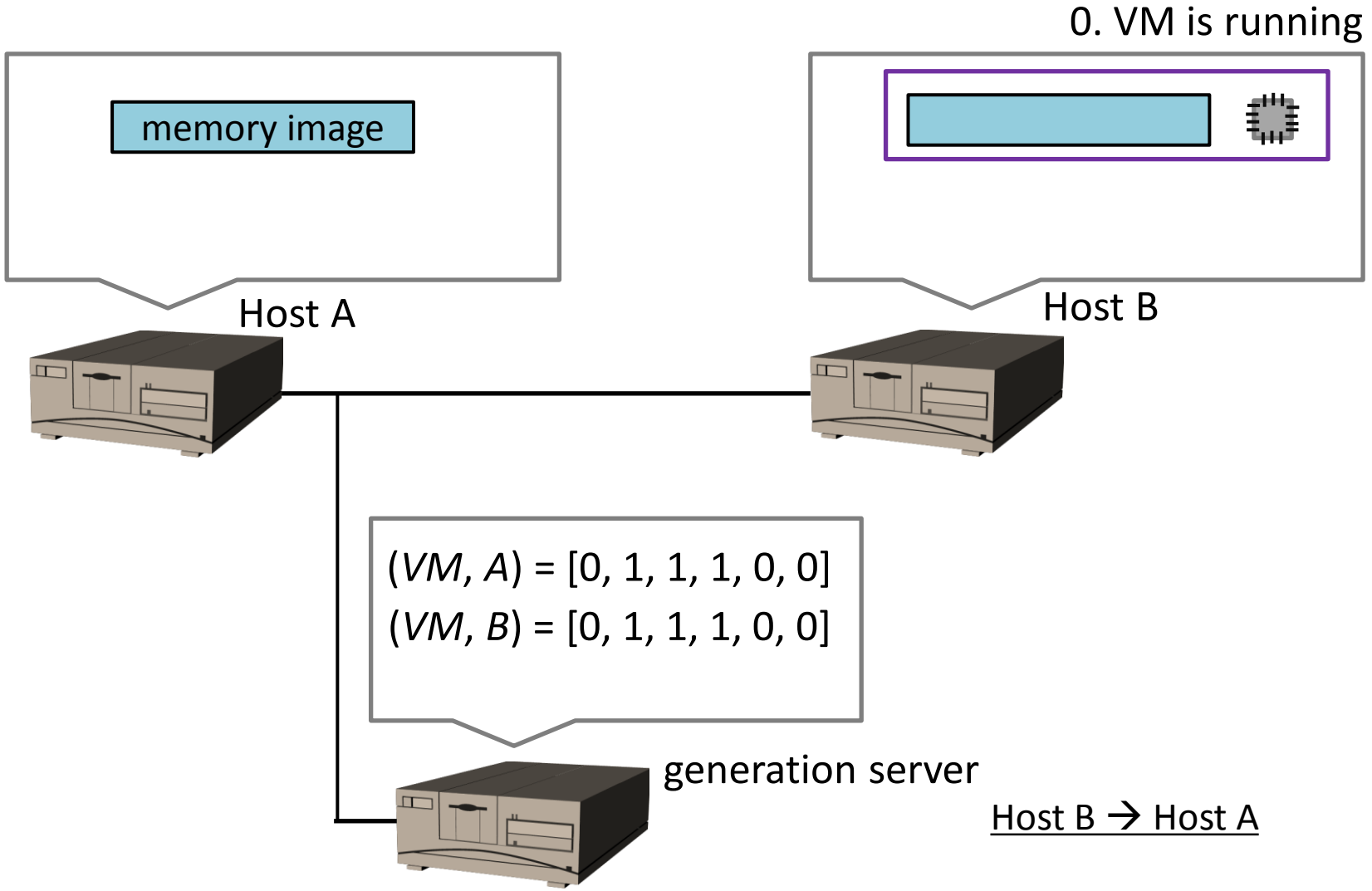
How the system works: the first migration



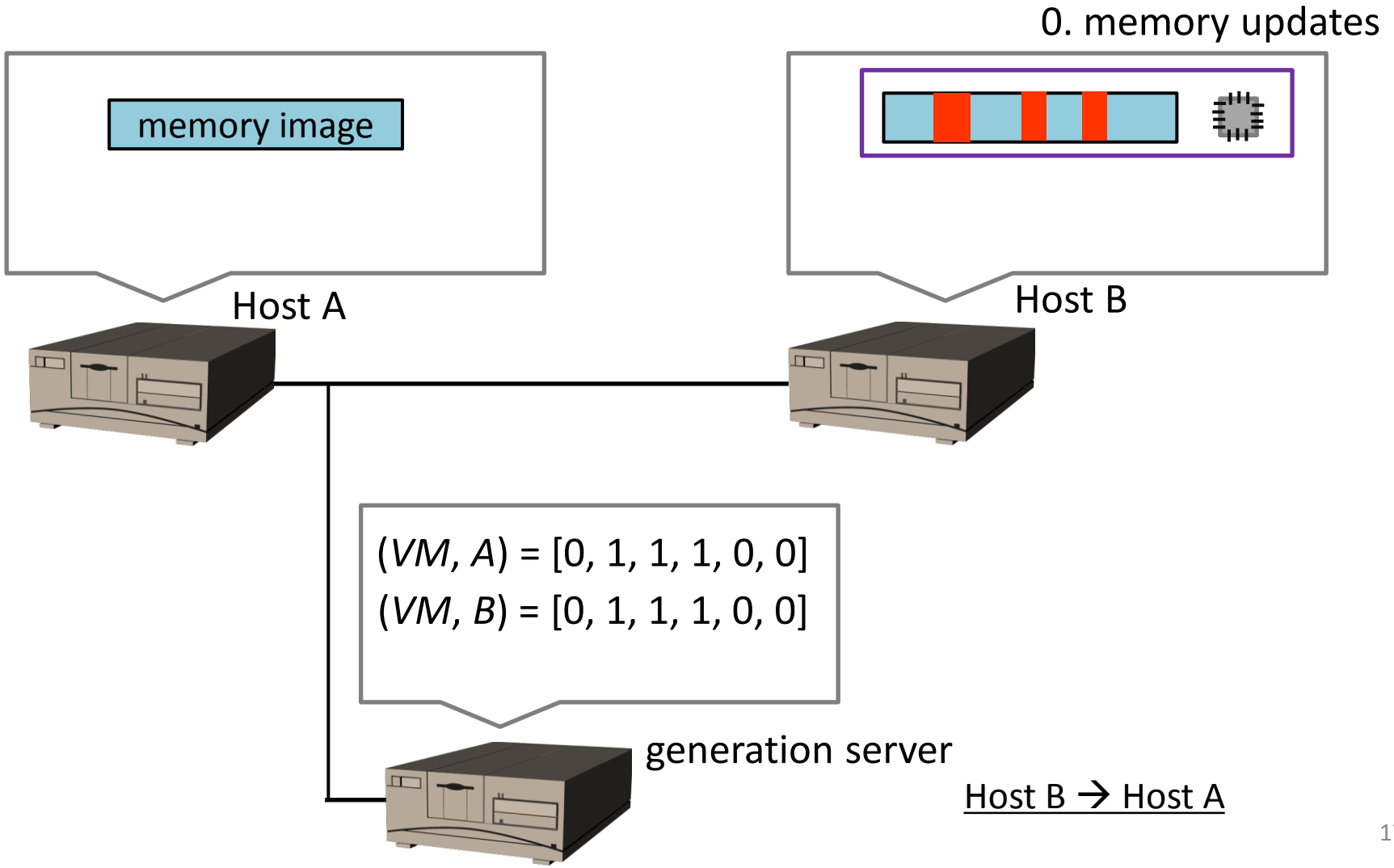
How the system works: the first migration



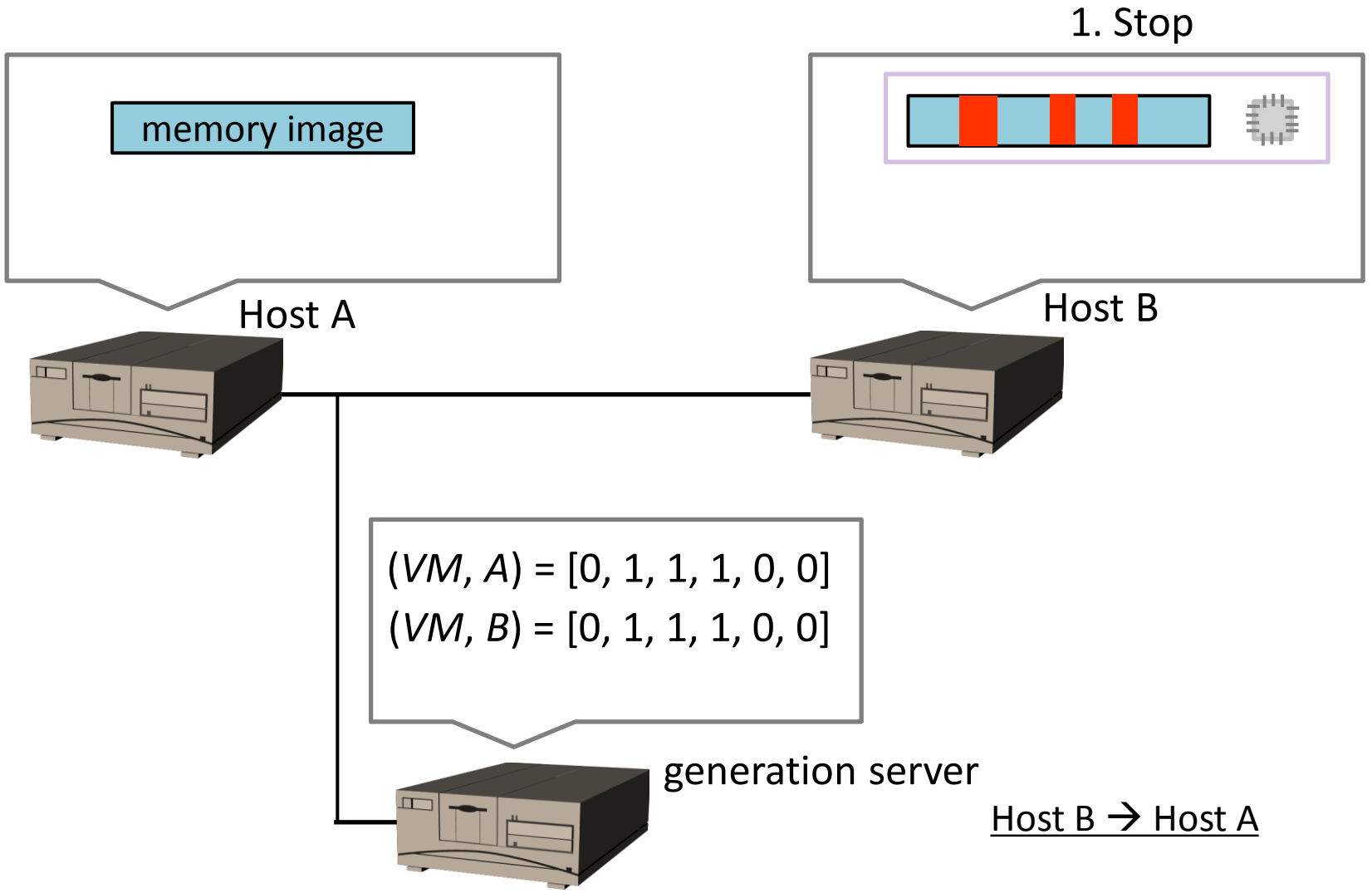
How the system works: when migrating back



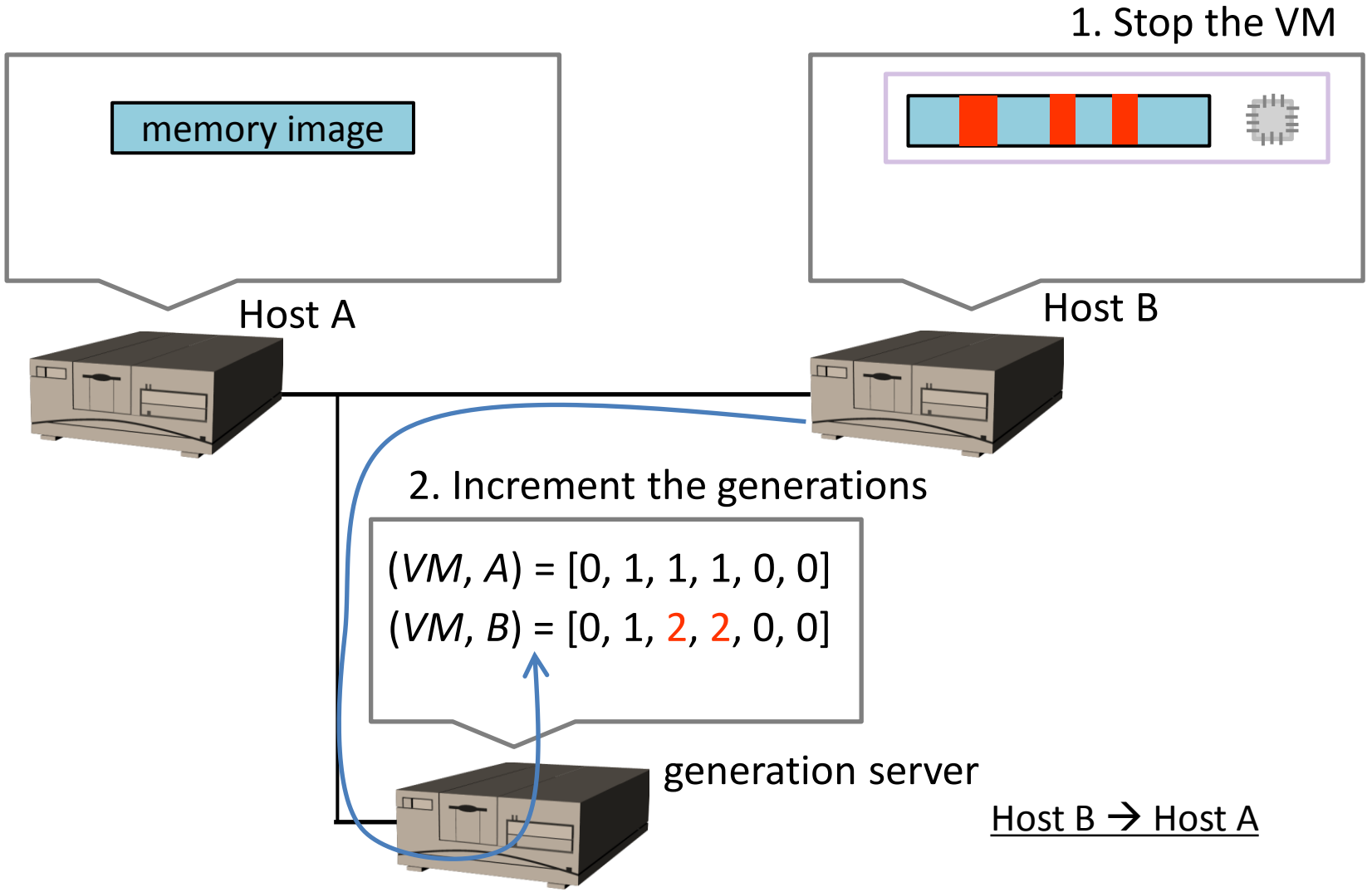
How the system works: when migrating back



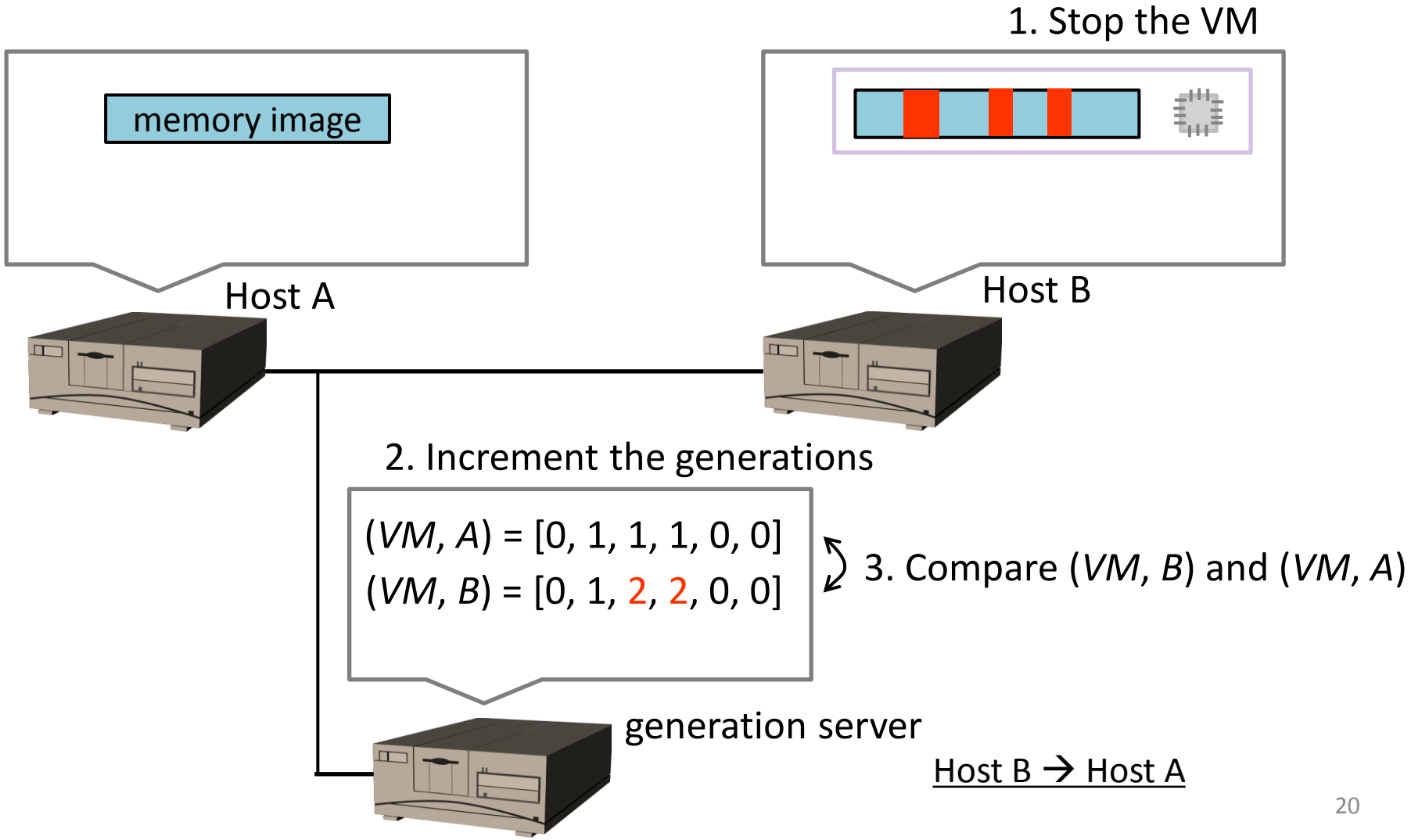
How the system works: when migrating back



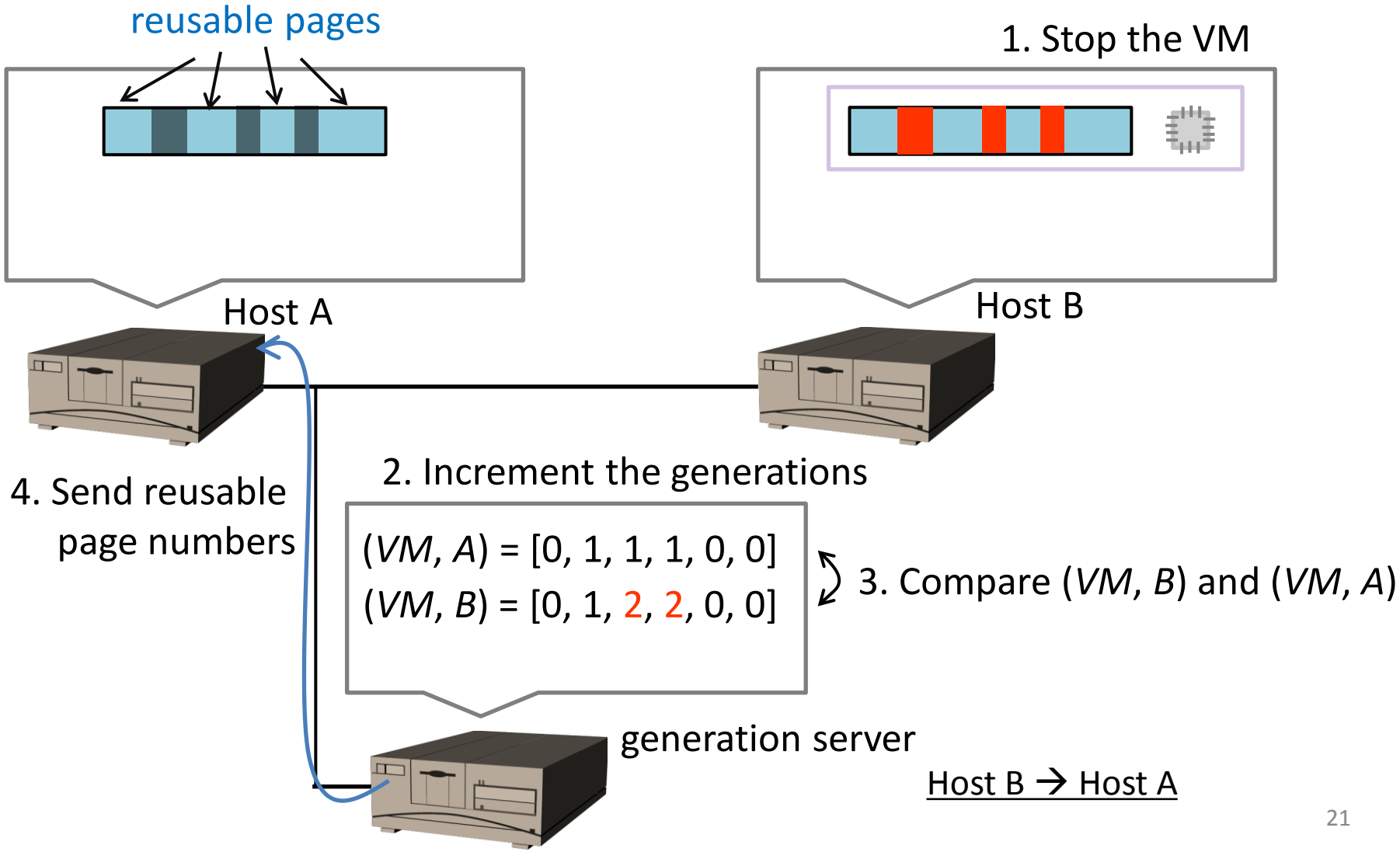
How the system works: when migrating back



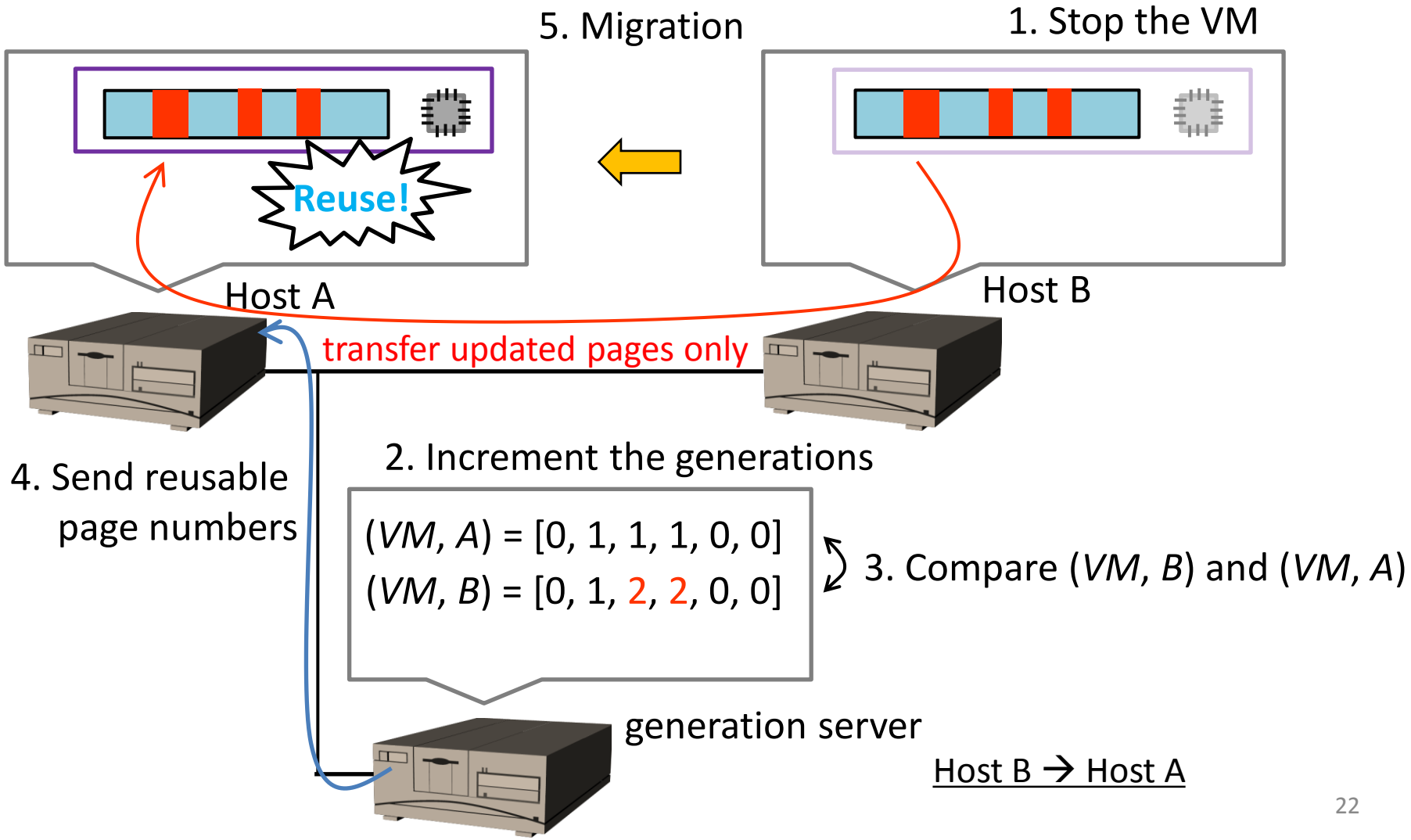
How the system works: when migrating back



How the system works: when migrating back



How the system works: when migrating back



Evaluation: Setting

Machine Environment

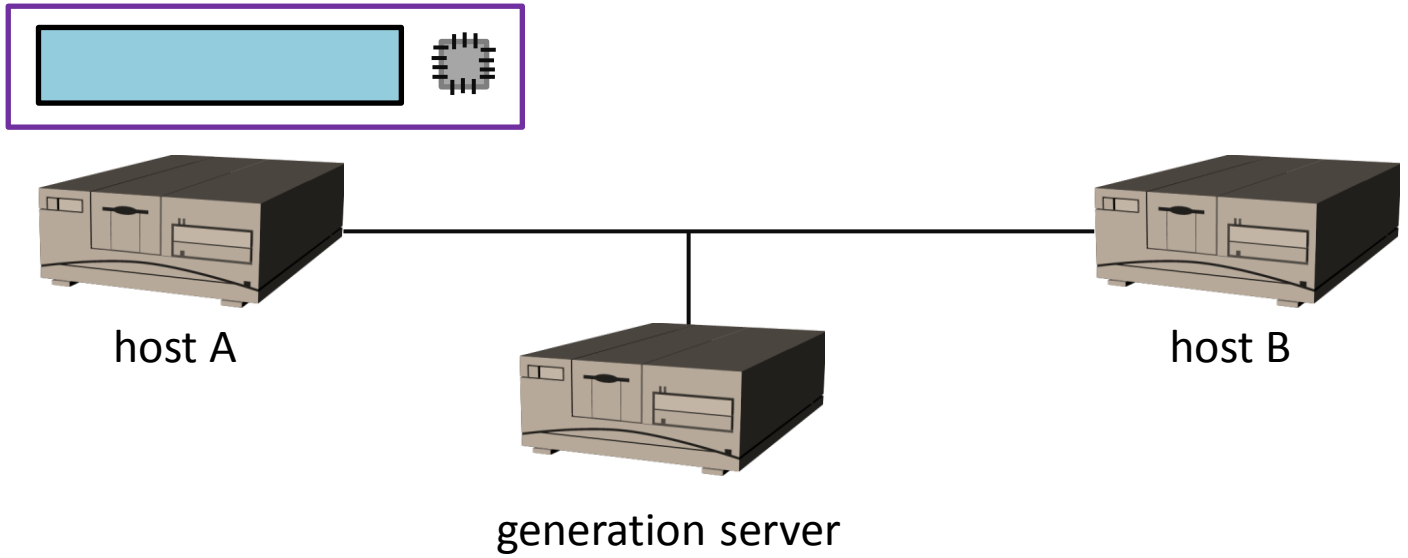
- 2 hosts, 1 generation server
  - Intel Xeon X5460 (4 cores), 8GB RAM
  - Debian squeeze (64 bit version)
  - QEMU 0.13.0 with our modification, KVM 2.6.38
  - Gigabit Ethernet
- 1 VM
  - 1 vCPU, 1GB RAM
  - Ubuntu 10.10 server (64 bit version)

Benchmarks

1. Busy Loop CPU-intensive (100%)
2. Apache Static web service, Network-intensive (100Mbps)
3. Video Video transcoding, IO-intensive (read, write)
4. TPC-C DB simulation of a net shop (CPU, memory, IO)

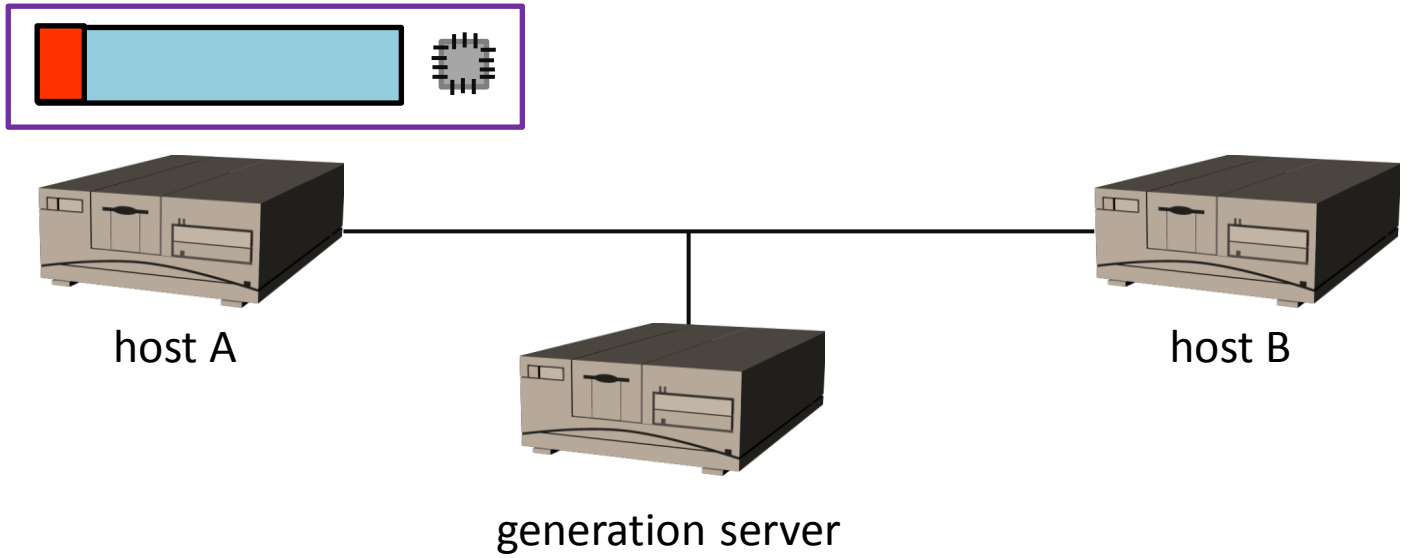
Evaluation: Procedure

1. Exec a benchmark on host A for *N* mins
   5, 10, 15
2. Migration from host A to host B
3. Exec the bench on host B for another *N* mins
4. Migration back from host B to host A



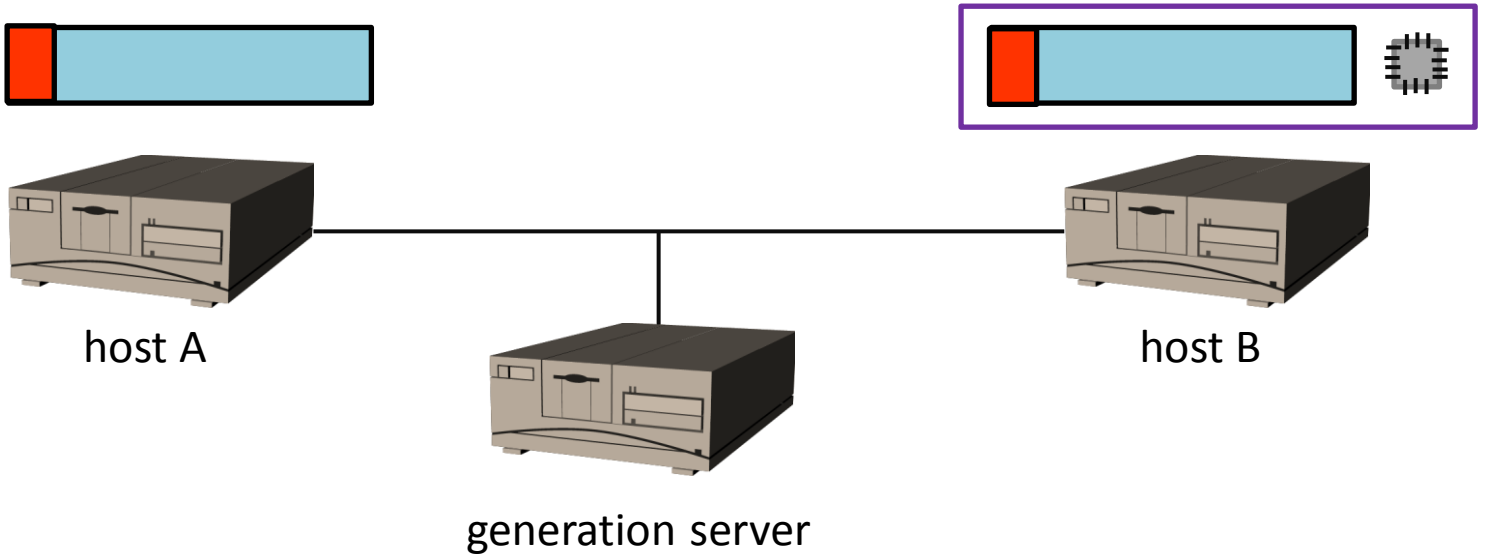
Evaluation: Procedure

1. Exec a benchmark on host A for *N* mins
   5, 10, 15
2. Migration from host A to host B
3. Exec the bench on host B for another *N* mins
4. Migration back from host B to host A



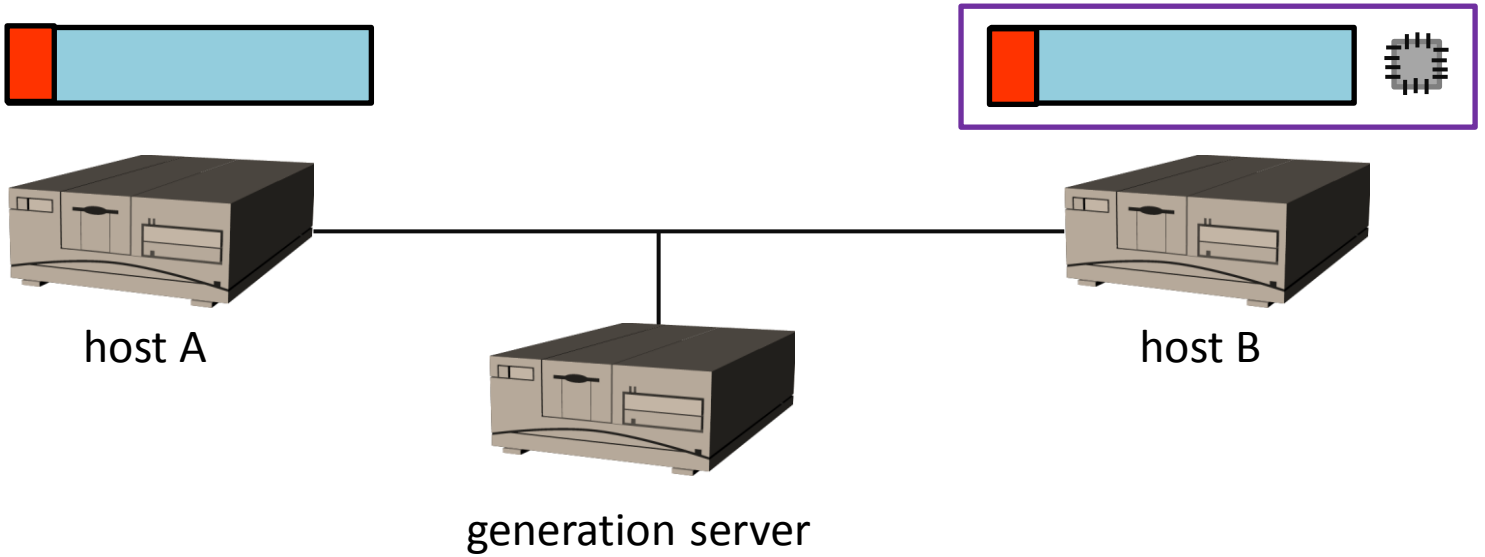
Evaluation: Procedure

1. Exec a benchmark on host A for *N* mins
   5, 10, 15
2. Migration from host A to host B
3. Exec the bench on host B for another *N* mins
4. Migration back from host B to host A



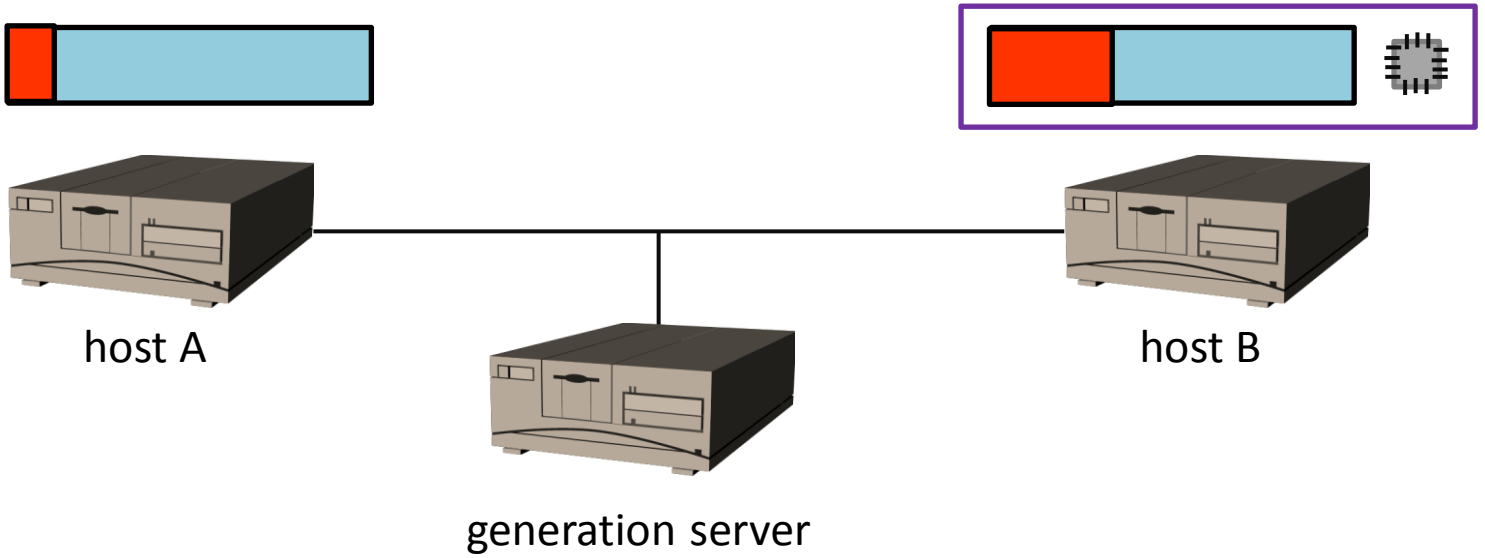
Evaluation: Procedure

1. Exec a benchmark on host A for *N* mins (5, 10, 15)
2. Migration from host A to host B
3. Exec the bench on host B for another *N* mins
4. Migration back from host B to host A



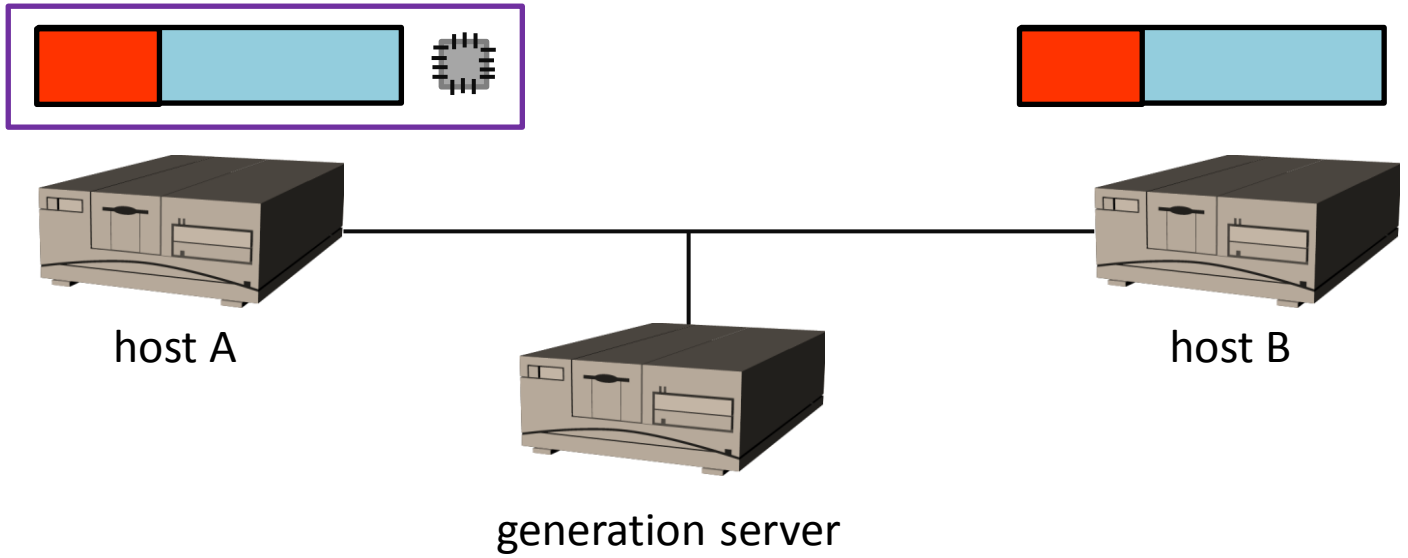
Evaluation: Procedure

1. Exec a benchmark on host A for *N* mins
   5, 10, 15
2. Migration from host A to host B
3. Exec the bench on host B for another *N* mins
4. Migration back from host B to host A

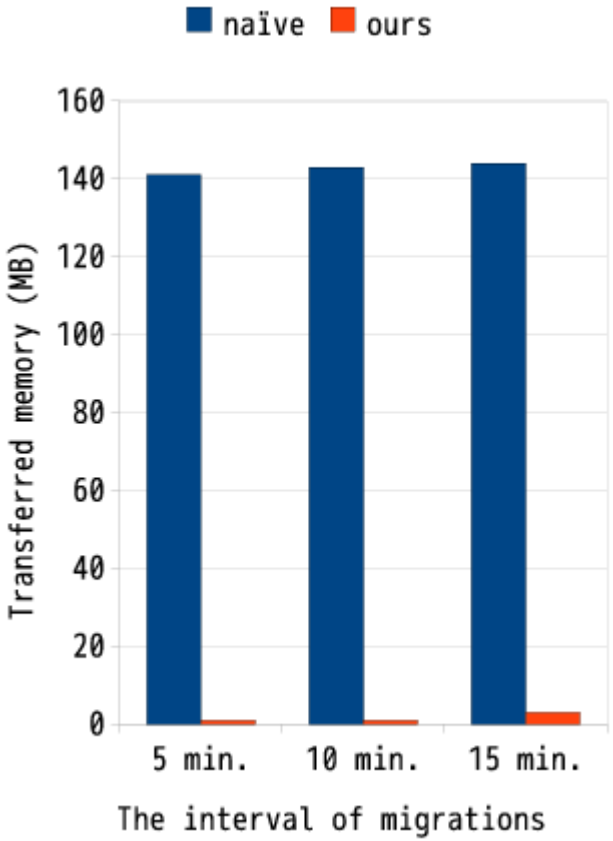


Evaluation: Procedure

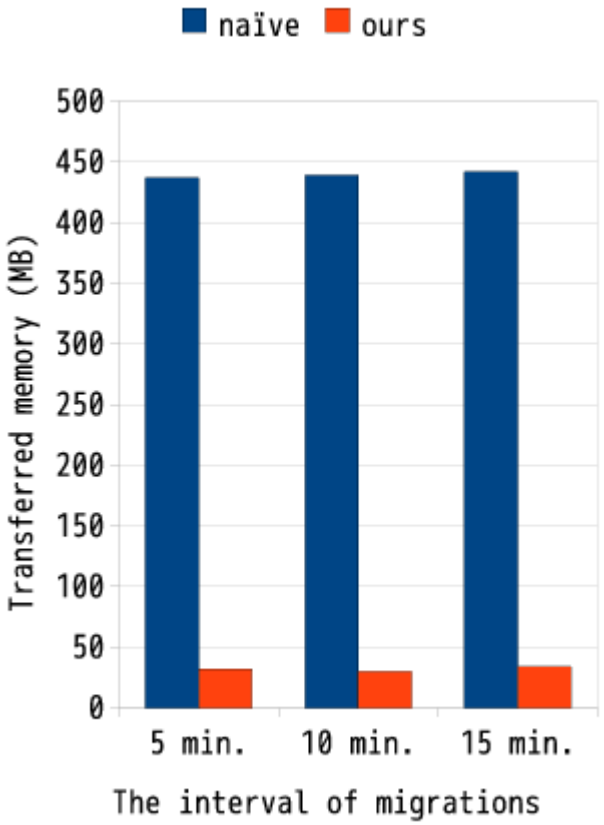
1. Exec a benchmark on host A for *N* mins
   5, 10, 15
2. Migration from host A to host B
3. Exec the bench on host B for another *N* mins
4. Migration back from host B to host A



Evaluation: Results



Busy Loop

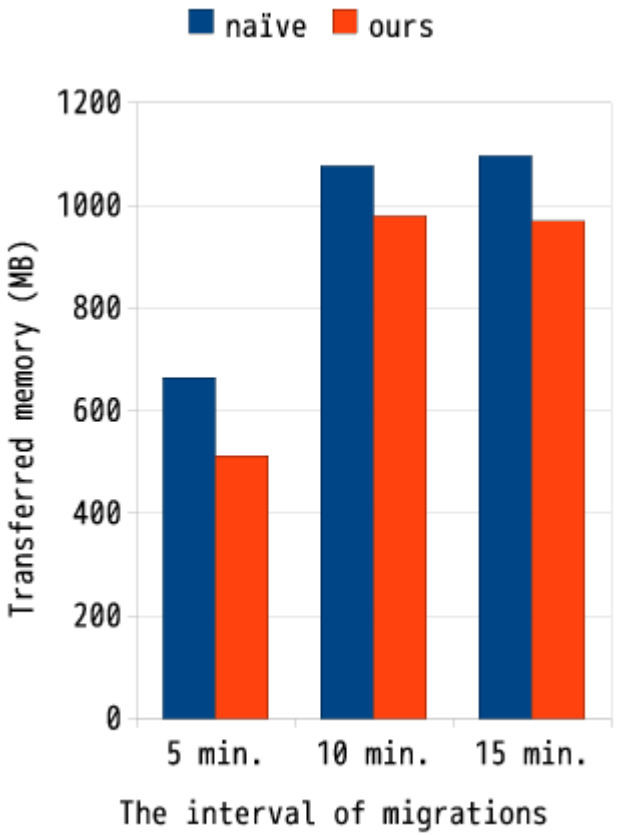


Apache

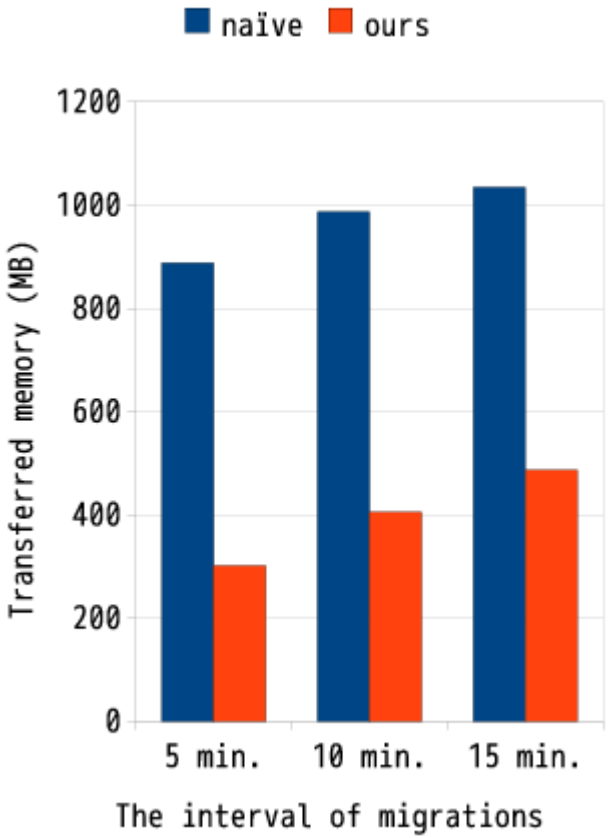
Busy Loop Ideal case, reused more than 90%

Apache Uses much memory for file cache, but I/O with static data does not update the memory

Evaluation: Results



Video



TPC-C

Video Big converted video dirties much memory

TPC-C Memory access pattern is complex, but memory reusing can be utilized

Out proposal is beneficial for practical transaction systems

Related work

- Researches on improved live migration
  - compress diff between older version of the page in iterative copying [1]
  - Transfer execution logs, instead of the memory [2]
- Uniqueness of our proposal
  - we focus of the initial copy of the memory
  - our work and related ones can be utilized together

[1] Evaluation of Delta Compression Techniques for Efficient Live Migration of Large Virtual Machines, *VEE2011*, Svärd *et al*.
[2] Live Migration of Virtual Machine Based on Full System Trace and Replay, *HPDC2009,* Liu *et al*.

Future work

- Memory cache management
  - currently all the memory pages must be cached
  - better cache management is needed
- Granularity of reusing
  - Reuse ratio in TPC-C:

    50% (page-wise) → more than 80% (byte-wise)