Yabusame: Postcopy Live Migration for Qemu/KVM

Takahiro Hirofuchi, AIST
*Isaku Yamahata, VALinux Systems Japan K.K.

Linux Plumbers Sep 9, 2011

This project is partially funded by Ministry of Economy, Trade and Industry.

Outline

- What is Postcoy Live Migration?
  - Comparison with Precopy
  - Experience from an early, ad-hoc prototype
- Postcopy for Qemu/KVM
  - Production-level design
  - Qemu/KVM upstream merge

Precopy v.s. Postcopy

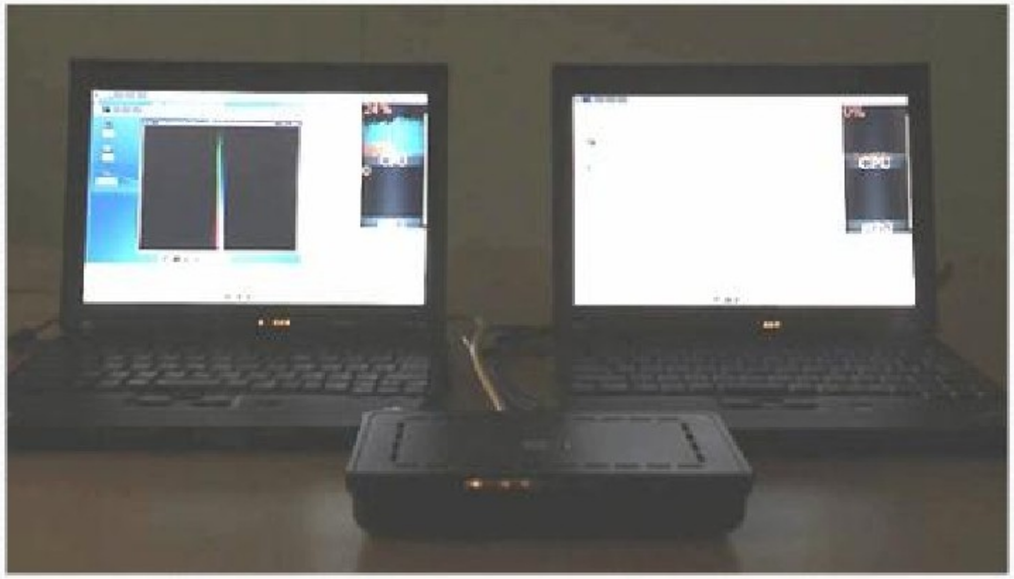
- Precopy live migration
  - Copy VM memory before switching the execution host
  - Widely used in VMMs
- Postcopy live migration
  - Copy VM memory after switching the execution host
  - Yabusame

Demo http://grivon.apgrid.org/quick-kvm-migration

A VM with 1GB RAM is live-migrated to the right PC.

Normal Live Migration

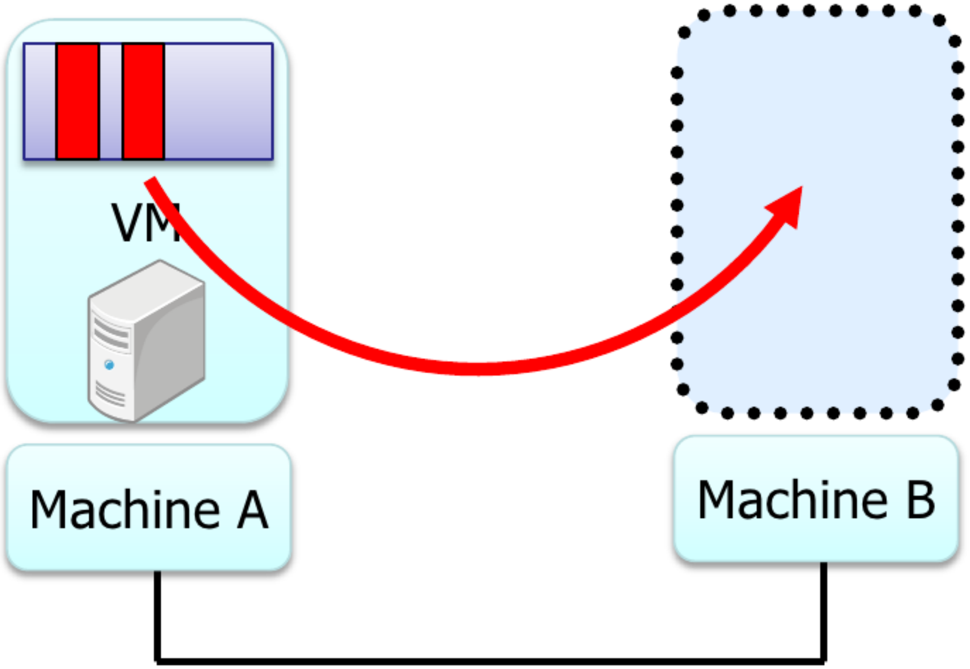
Yabusame Live Migration
(Developed by AIST)



Precopy Live Migration (1)

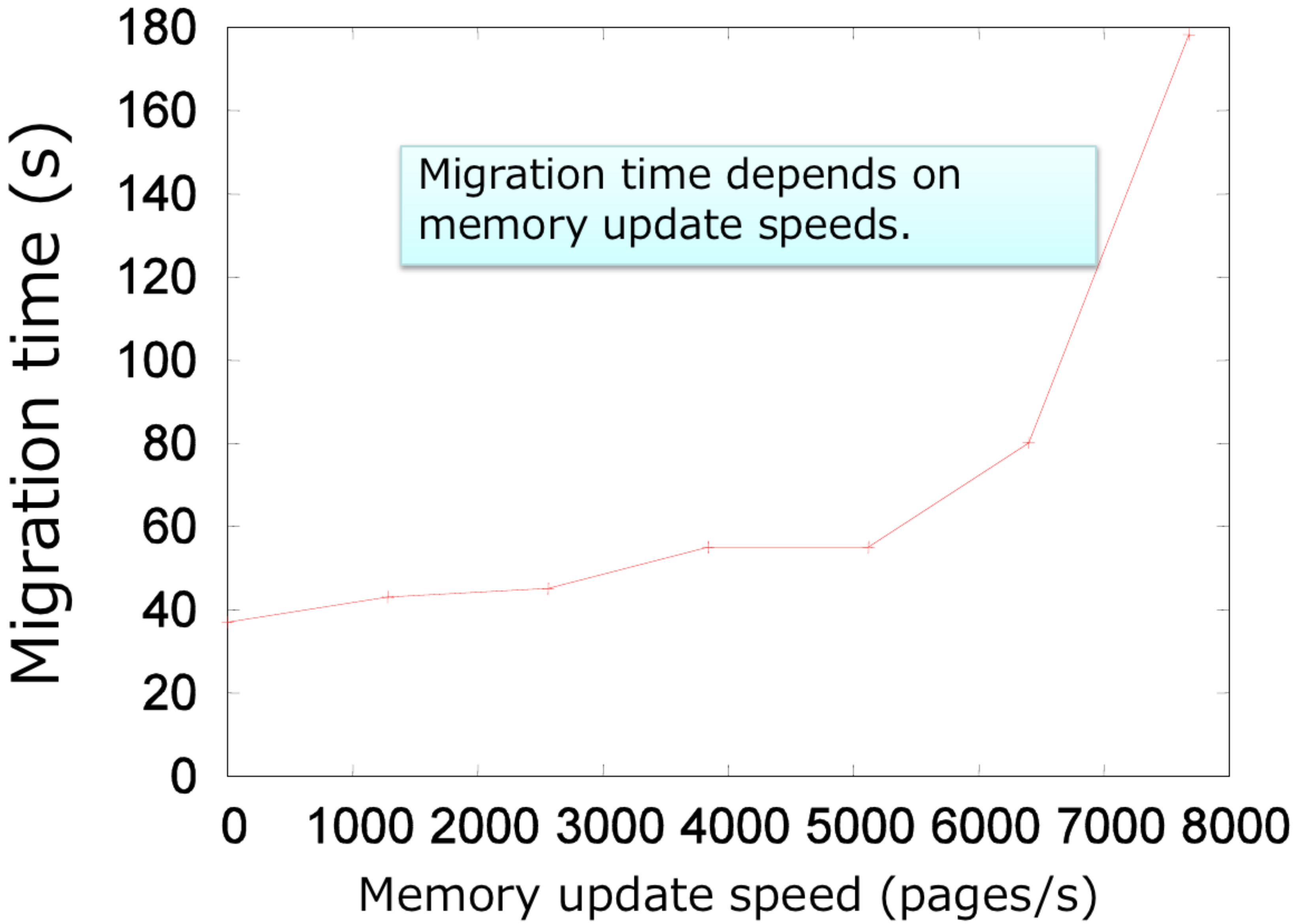
Copy VM memory before relocation

1. Copy all memory pages to destination
2. Copy memory pages updated during the previous copy again
3. Repeat the 2nd step until the rest of memory pages are enough small
4. Stop VM
5. Copy CPU registers, device states, and the rest of memory pages.
6. Resume VM at destination



Precopy Live Migration (2)

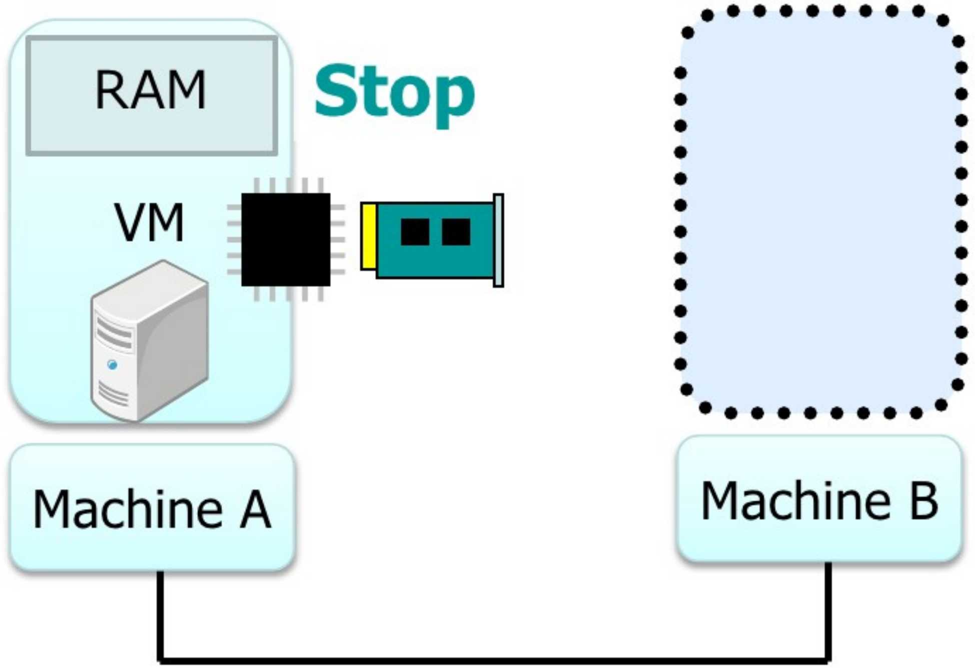
Copy VM memory before relocation



Migration time depends on memory update speeds.

Postcopy Live Migration (1)

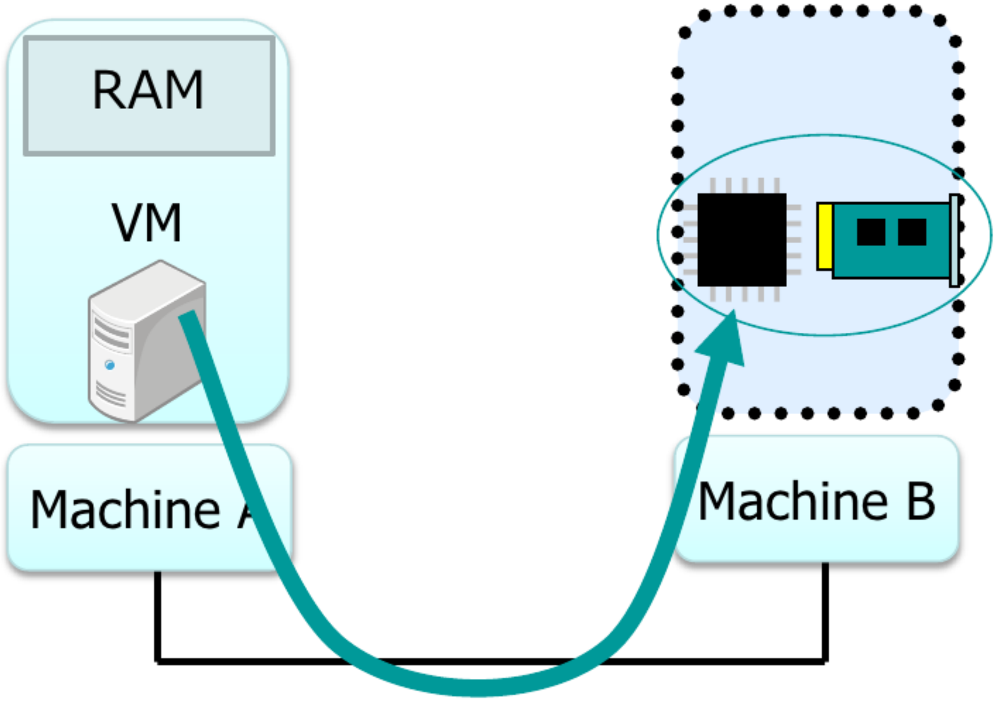
Copy VM memory after relocation



1. Stop VM
2. Copy CPU and device states to destination
3. Resume VM at destination
4. Copy memory pages

Postcopy Live Migration (2)

Copy VM memory after relocation



1. Stop VM
2. Copy CPU and device states to destination
3. Resume VM at destination
4. Copy memory pages

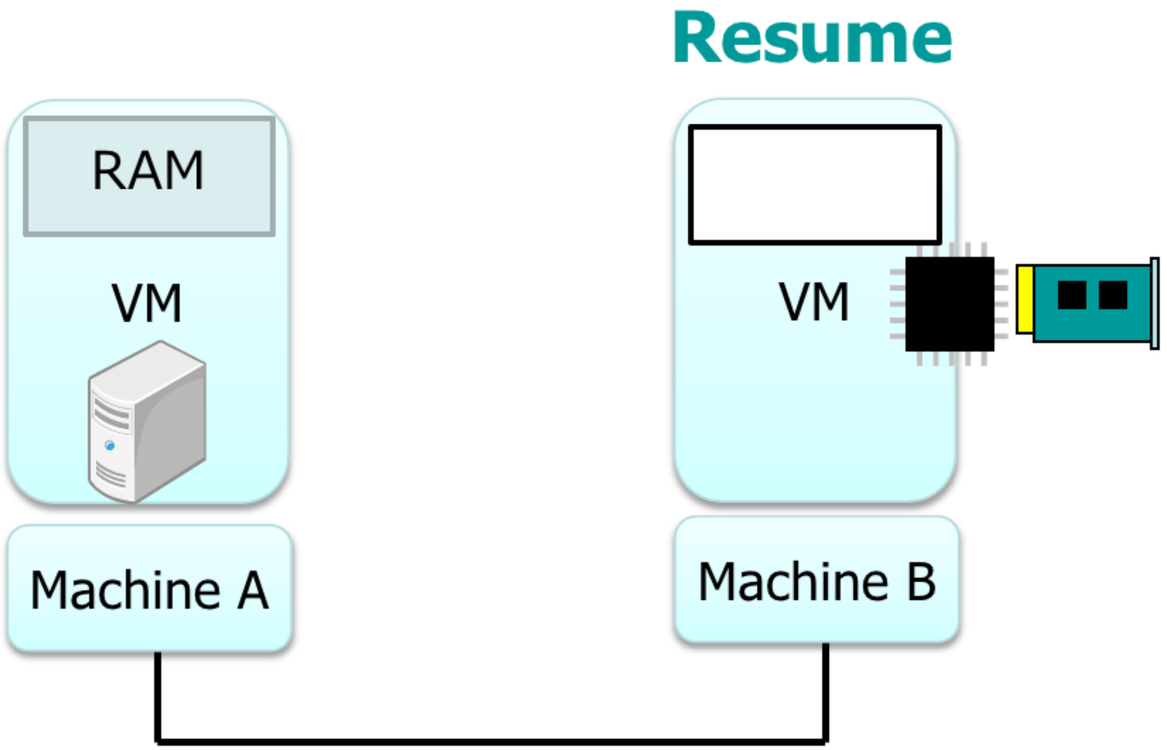
Copy CPU and device states

512KB or so (w/o VGA)

=> Less than 1 sec for relocation

Postcopy Live Migration (3)

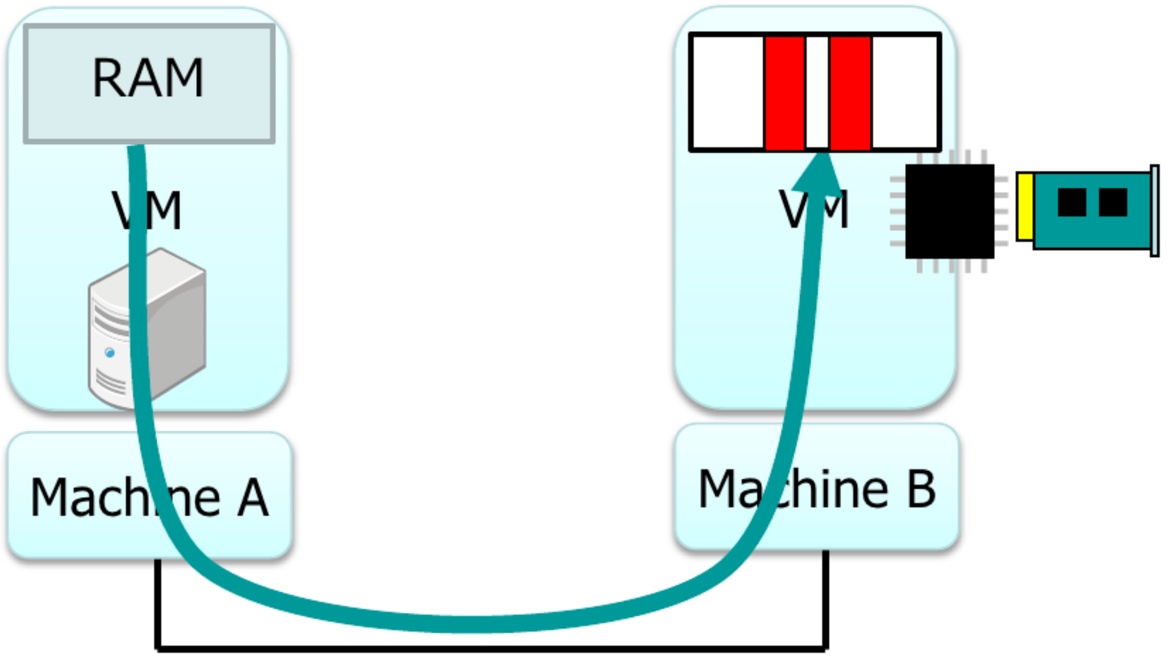
Copy VM memory after relocation



1. Stop VM
2. Copy CPU and device states to destination
3. Resume VM at destination
4. Copy memory pages

Postcopy Live Migration (4)

Copy VM memory after relocation



1. Stop VM
2. Copy CPU and device states to destination
3. Resume VM at destination
4. Copy memory pages

Copy memory pages
- **On-demand**
- **Background (Precache)**

Precopy v.s. Postcopy

	Precopy	Postcopy
	Before the execution host is switched, memory pages are transferred to the destination.	After the execution host is switched, memory pages are transferred to the destination.
The time until the execution host is switched	$\frac{\text{RAM size}}{\text{Network speed}} + \text{alpha}$	**200-300ms**
The time until all states are removed	$\frac{\text{RAM size}}{\text{Network speed}} + \text{alpha}$	$\frac{\text{RAM size}}{\text{Network speed}}$

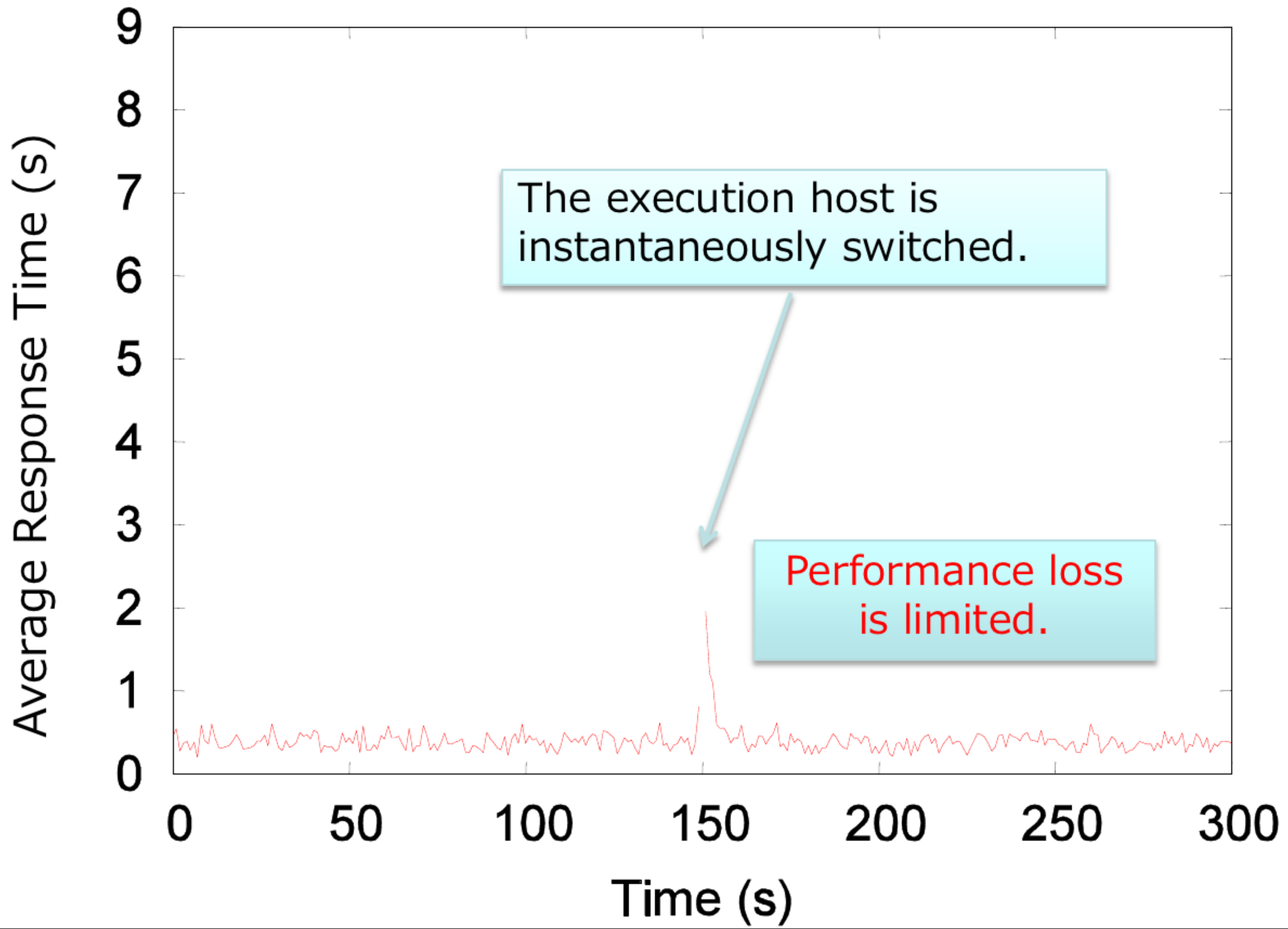
- alpha: depends on memory update speed (non deterministic!)
- Note the above values are the worst case.
  - Qemu skips zero-filled page.

Question (1)

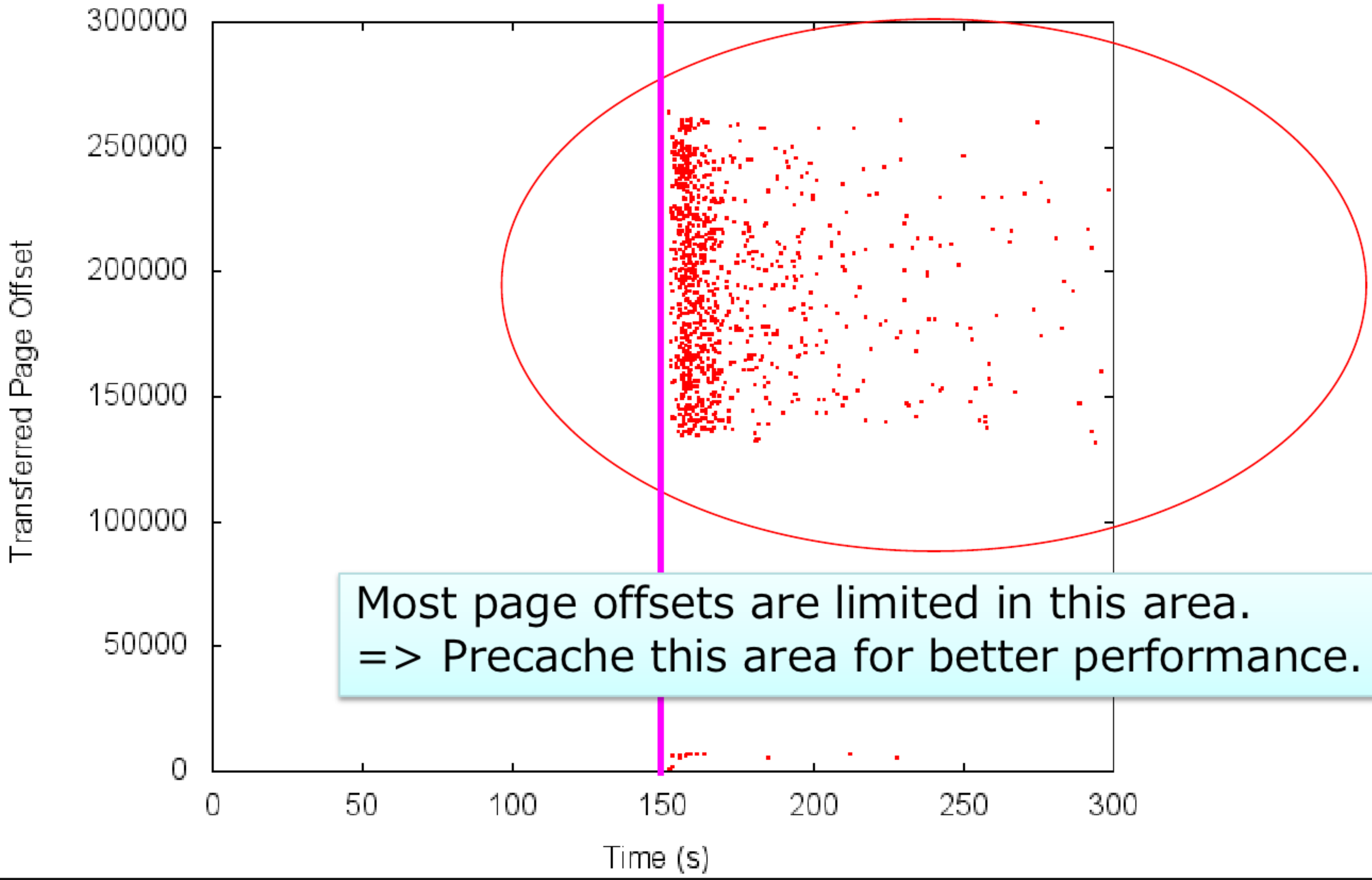
- Is there performance loss after relocation?
  - Yes, (hopefully?) slightly.
  - The working set of memory pages is limited.
  - These pages are precached as soon as possible.

Migrate an heavily-loaded Web Server VM

Using the SPECweb 2005 Banking benchmark.



Transferred Page Offsets (Precache Disabled)



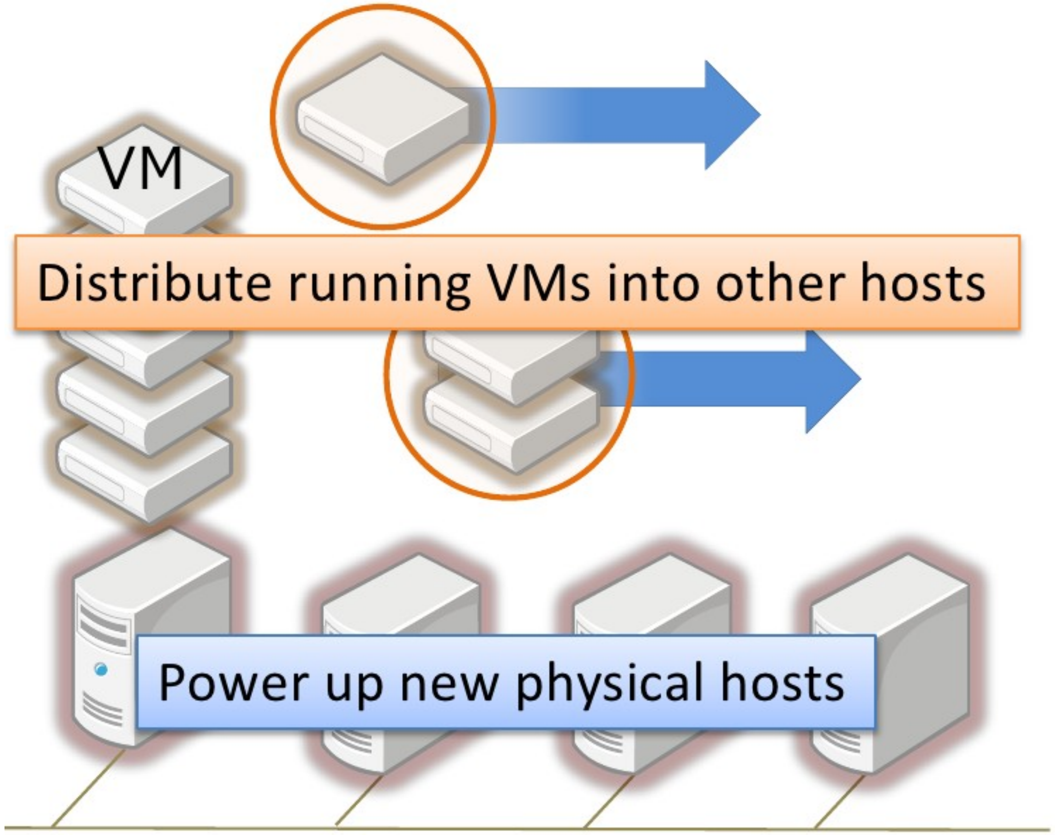
Most page offsets are limited in this area.
=> Precache this area for better performance.

Question (2)

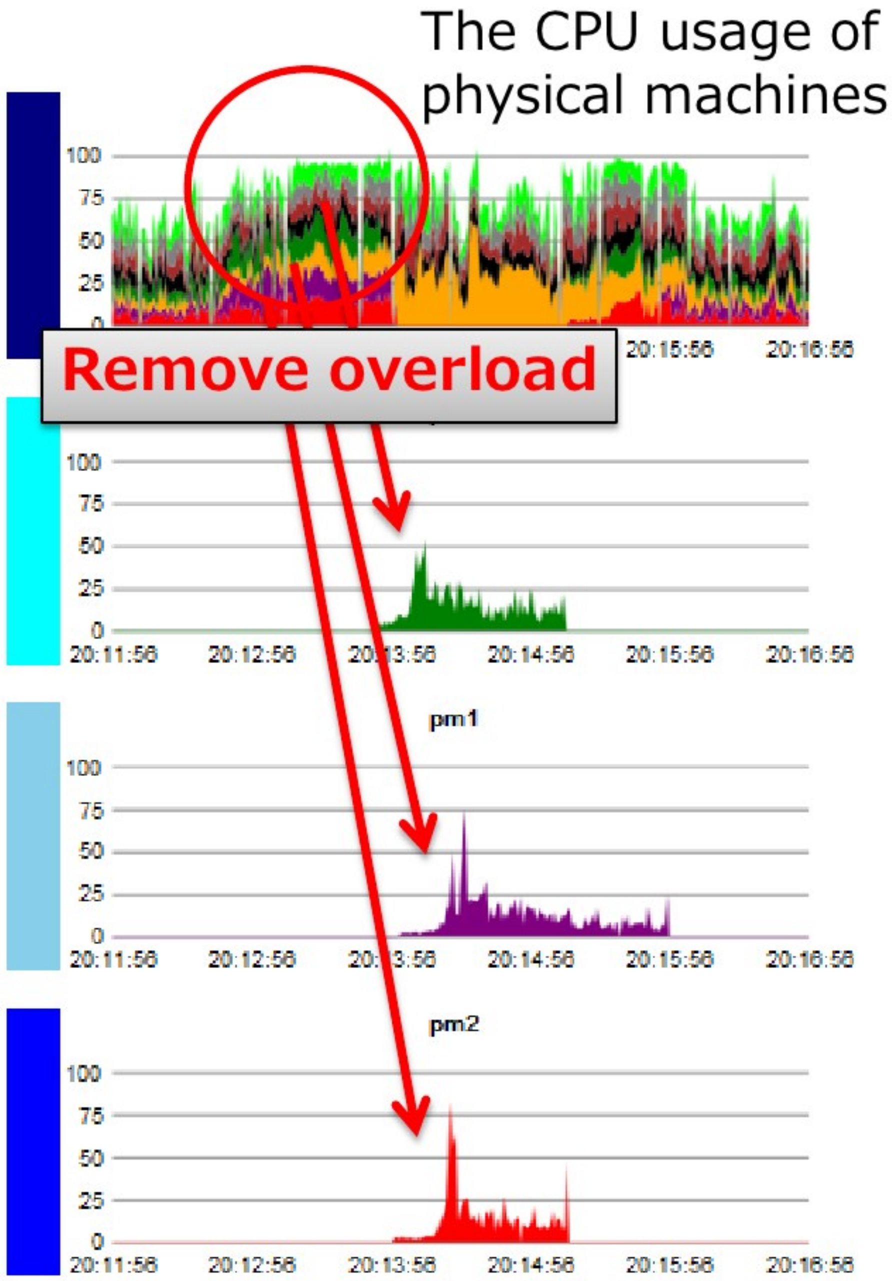
- Useful for dynamic server consolidation?
  - Hopefully, yes.
  - Enables quick, deterministic load balancing.

Dynamic Consolidation

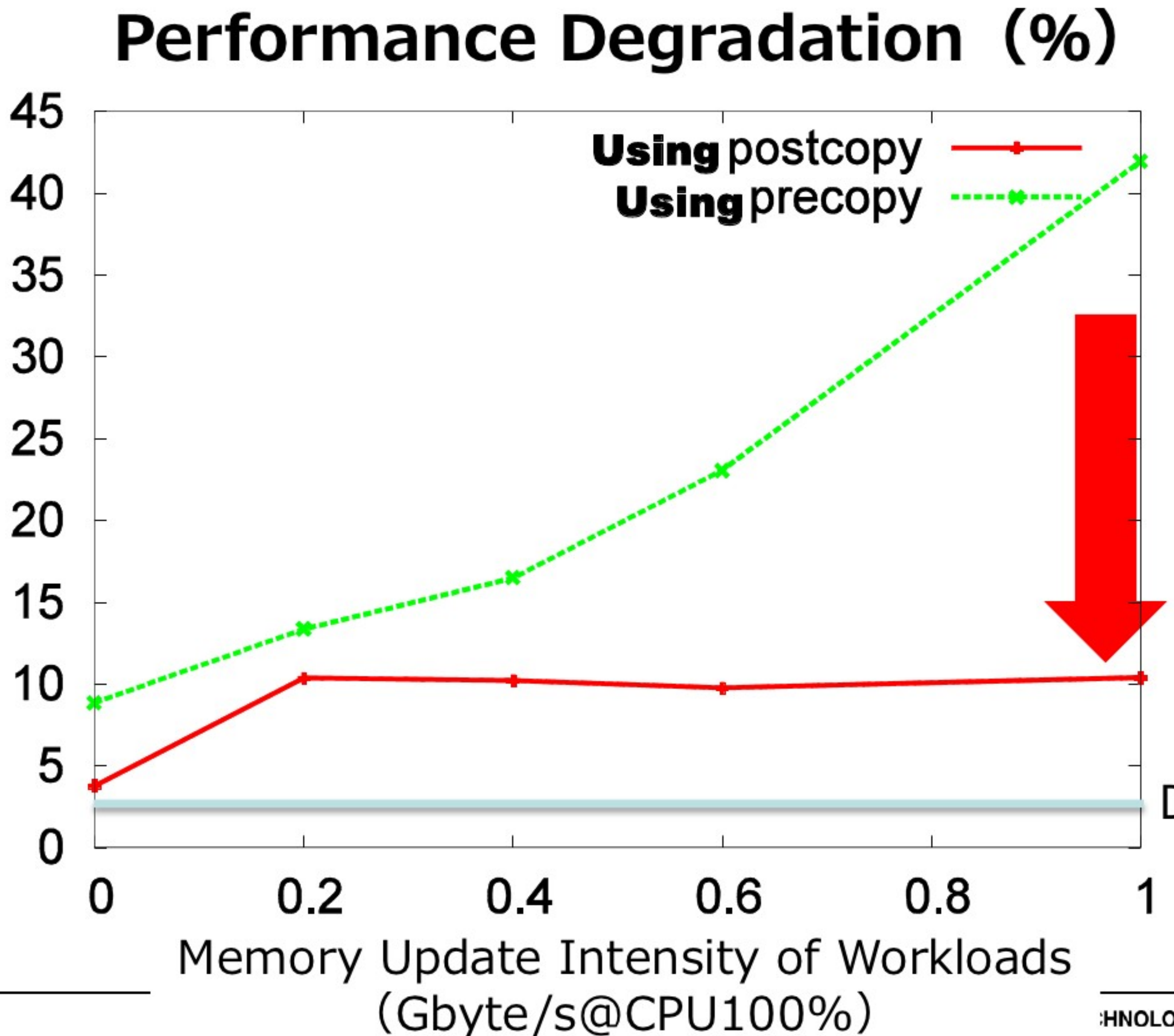
When datacenter load becomes high,


VM
Distribute running VMs into other hosts
Power up new physical hosts

Assure VM Performance



Postcopy v.s. Precopy for dynamic consolidation



The consolidation system using precopy cannot quickly migrate VMs, resulting serious performance loss on sudden load changes.

Postcopy contributes alleviating performance loss.

Detection overhead

Summary of the first half

- Yabusame
  - Postcopy Live Migration for Qemu/KVM
  - The execution host is switched in 200-300ms.
  - The total migration time is shorter than precopy. It is deterministic!

  - An early-stage, ad-hoc, ad-hoc, concept-proof prototype is here.
    - http://grivon.apgrid.org/quick-kvm-migration
    - Do not use it!

- Next, discuss a upstream merge-able design…

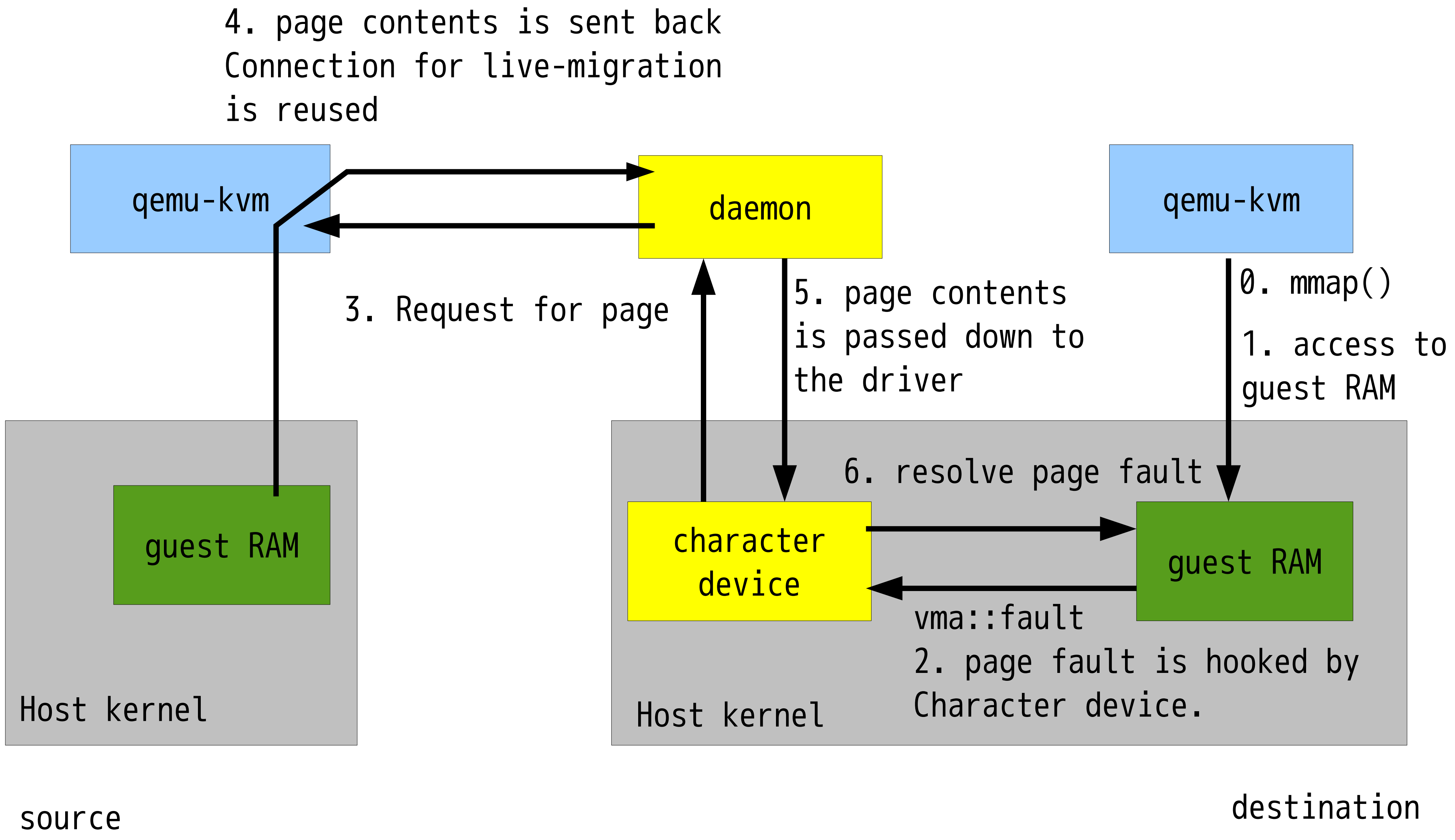
Why re-design/implementation

- Next step of YABUSAME project
  - We'd like to merge post copy livemigartion into the upstream
- The existing patch was implemented for
  - academic research
  - proof-of-concept
- So, its design/implementation is
  - Ad-hoc, quick-hack
  - Not suitable for the upstream merge
- => re-design/implement it for the upstream merge

Prerequisite

- New implementation should satisfy
- Allow qemu/kvm features
  - target/host agnostic
  - accelerator(tcg, kvm)
  - Devices
    - emulated devices, virtio, vhost
    - assigned-device won't be supported
- Allow kvm host features
  - Swap, KSM, THP/hugetlb, async page fault, coredump...
- Zero overhead after migration completes
  - CPU, memory
- Minimal administrative operation
  - Don't require special preparation on migration source

Implementation proposal



Design points

- Who on the destination handles page requests
  - An independent daemon or a thread in qemu-kvm
- connection between source/destination
  - Re-use the connection of live-migration or
  - Create new connection
- Page transfer protocol. Based on
  - Qemu live migration or
  - other protocol: nbd, iSCSI, AOE
- How to hook guest RAM access

Hooking guest RAM access

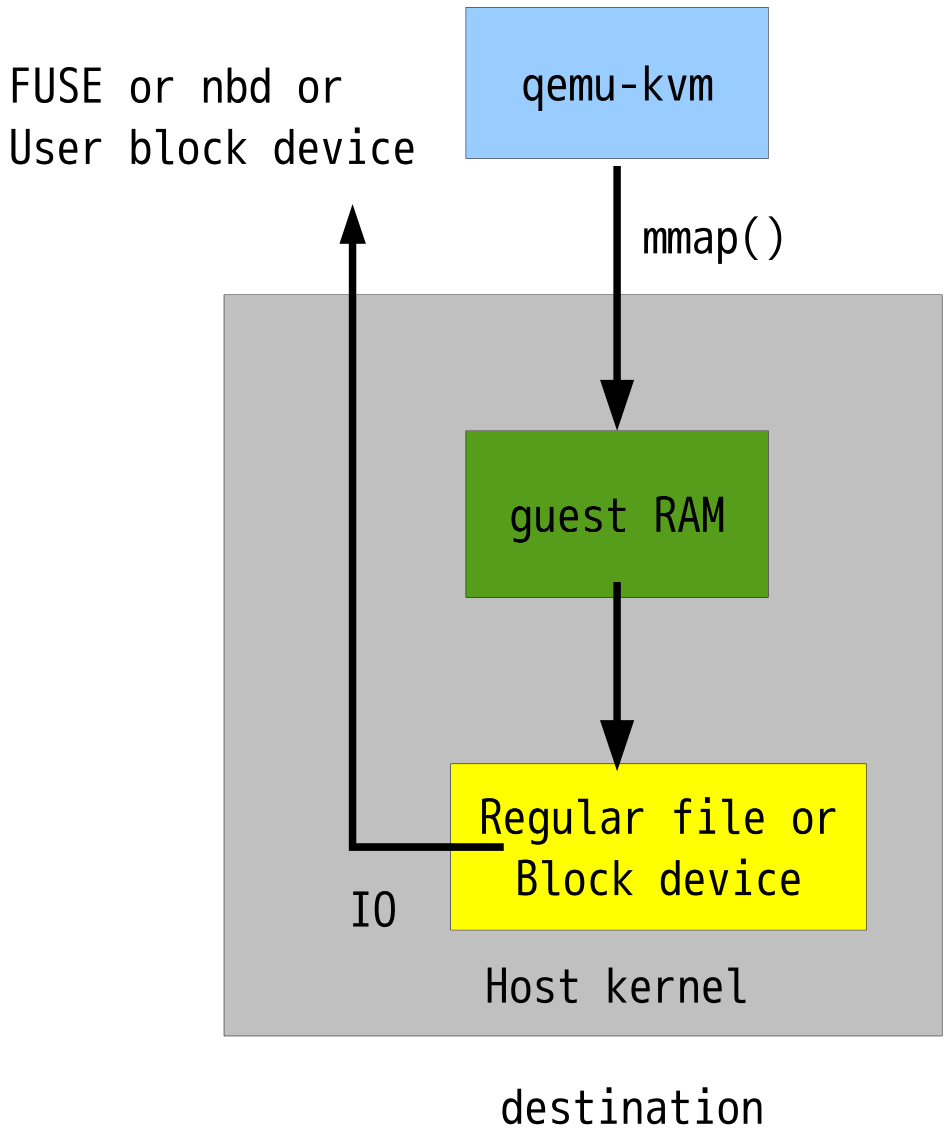
- Insert hooks all accesses to guest RAM
- Character device driver
- Backing store(block device or file)
- Swap device

Insert hooks

- Insert hooks all accesses to guest RAM
- Carefull code inspection is required
- Pros
  - Portable. May work with qemu tcg without any kernel drivers
- Cons
  - Impractical

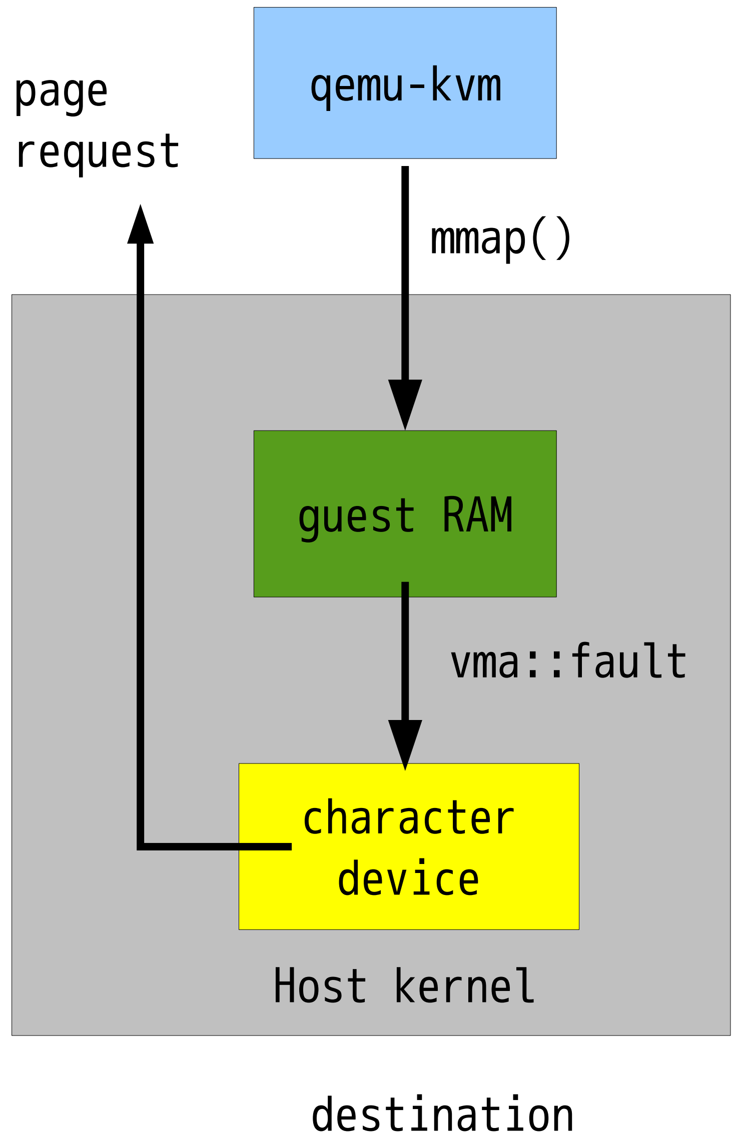
Backing device/file approach

- Use block device or file as backing store for guest RAM
- Pros
  - New device driver isn't needed
- Cons
  - Future improvement would be difficult
  - Some KVM host features wouldn't work.(KSM, THP)



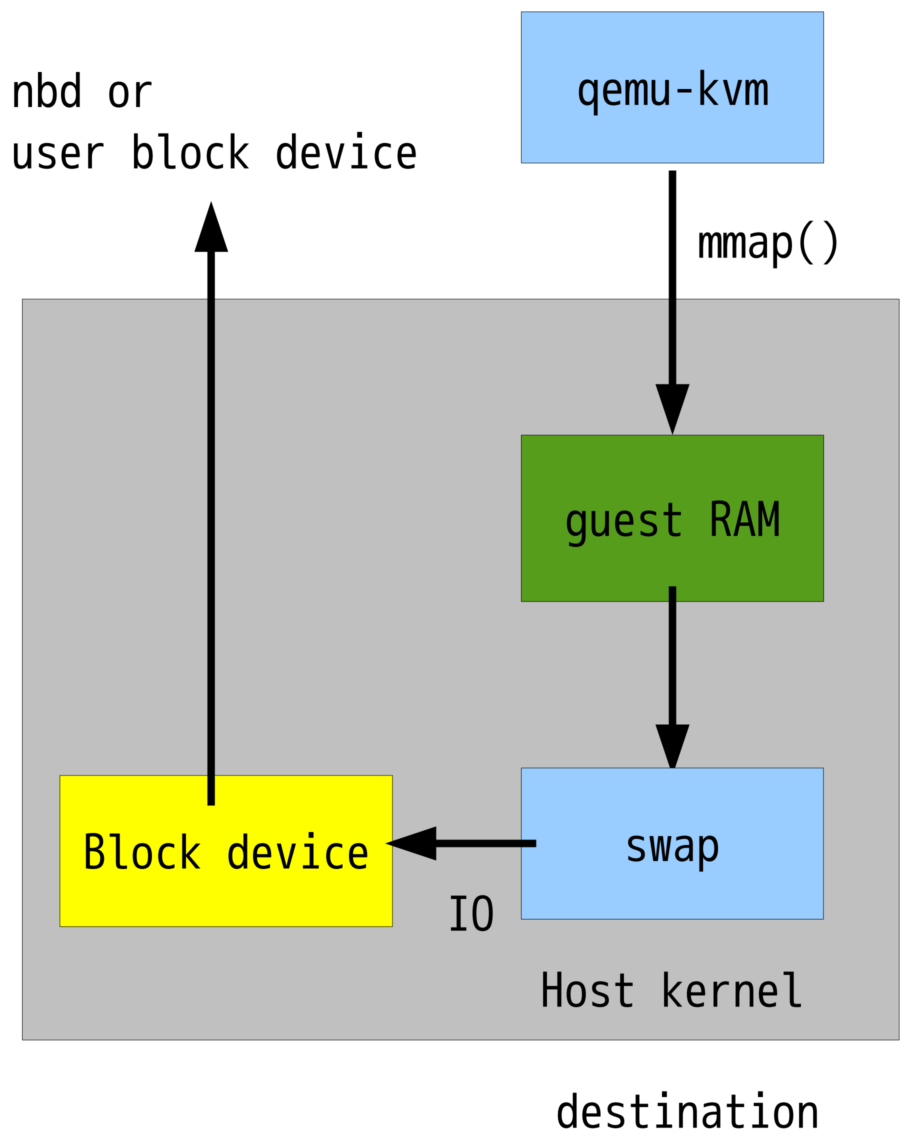
Character device approach

- Character device to handle page fault on guest RAM area
- Pros
  - Straight forward
  - Future improvement would be easy
- Cons
  - New driver is necessary
  - Some KVM host features wouldn't work.(KSM, THP)
    - VMA isn't anonymous
    - Can be fixed



Swap device approach

- On destination, set up such that all guest RAM are swapped out
- Pros
  - Every thing would be normal after migration completes
  - New device driver isn't needed
- Cons
  - Future improvement would be difficult
  - Administration
    - setting up swap device



Comparison

	Pros	Cons
Modify VMM	portability	impractical
Backing store	No new device driver	Difficult future improvement Some kvm host features wouldn't work
Character Device	Straight forward Future improvement	Need to fix kvm host features
Swap device	Everything is normal after migration	Administration Difficult future improvement

Future work after the merge

- Finish implementation
  - Investigate cuse
- Evaluation/benchmark
- Optimization
  - Another connection for background page transfer
    - Bandwidth control
  - Reduce unnecessary page fault
  - Mix precopy/postcopy
  - Avoid memory copy
  - Hint not to send page contents
  - Not to fetch pages when writing/clearing whole page
    - cleancache/frontswap might be good candidate
- Libvirt support is necessary?
- Cooperate with Kemari

Questions/discussions

- Project page
  - http://sites.google.com/site/grivonhome/quick-kvm-migration
- Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension: proof-of-concept, ad-hoc prototype. not a new design
  - http://grivon.googlecode.com/svn/pub/docs/ccgrid2010-hirofuchi-paper.pdf
  - http://grivon.googlecode.com/svn/pub/docs/ccgrid2010-hirofuchi-talk.pdf
- Reactive consolidation of virtual machines enabled by postcopy live migration: advantage for VM consolidation
  - http://portal.acm.org/citation.cfm?id=1996125
  - http://www.emn.fr/x-info/ascola/lib/exe/fetch.php?media=internet:vtdc-postcopy.pdf
- Qemu Wiki
  - http://wiki.qemu.org/Features/PostCopyLiveMigration