Reducing Memory Sharing Cost for Virtualized Infrastructures

Ryota Ozaki and Ahikiro Nakao
The University of Tokyo

2012-03-02

20m / 5m

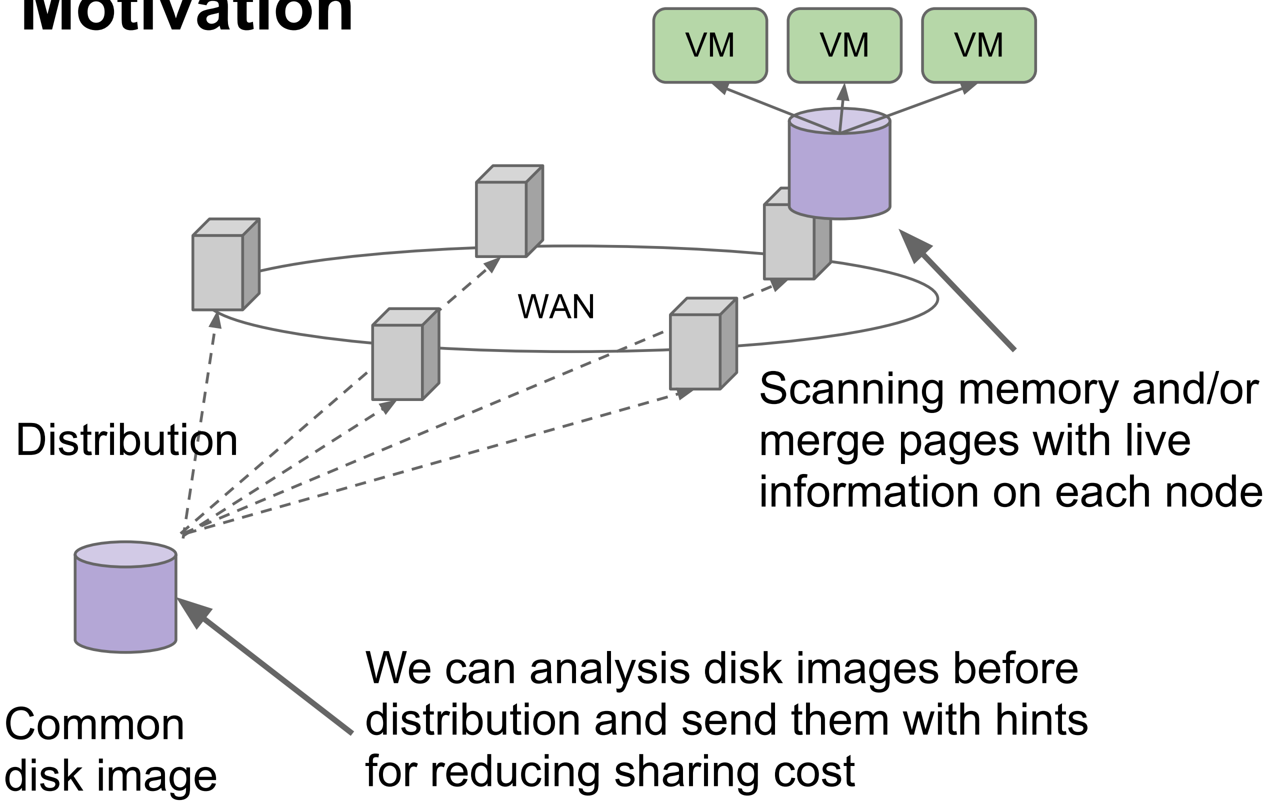
Background

- Increase the amount of memory
- Memory sharing techniques
    - A technique to share/merge identical pages
    - Increase virtual machine density
- Memory Scanning Cost
    - The cost is proportional to the amount of memory
- Virtualized infrastructures
    - Host machines cooperate over networks
    - Virtual machines that stems from a common disk image run on the host machines

Objective

- Reduce costs for memory sharing
  - The number of pages to be scanned
  - I/O tracking and page states
- Reduce total resource consumptions of an infrastructure
  - Same procedures are reproduced over machines

Motivation


VM
VM
VM
WAN
Distribution
Scanning memory and/or merge pages with live information on each node
Common disk image
We can analysis disk images before distribution and send them with hints for reducing sharing cost

Related Work

- Content-based sharing
  - Scanning & merging
  - ex. VMware ESX, Difference Engine, KSM, Satori
- CoW disk
  - Uses a knowledge that data from a shared read-only CoW disk will be identical if not modified
  - Track guest's I/O requests and updates of memory pages
  - ex. Disco, Satori
- Our proposal supports the latter
  - While the former can be used with ours

Design Decision

- Static analysis of disk images in advance
  - The results can be applied to many virtualization nodes
- Diversity of applications
  - Their memory may not have much chances to be merged
- Shared CoW disks and page caches are still important candidates to be merged
  - Many researches agreed
- Merge pages in a backend block driver
  - It can merge pages without data copy and interfering guests

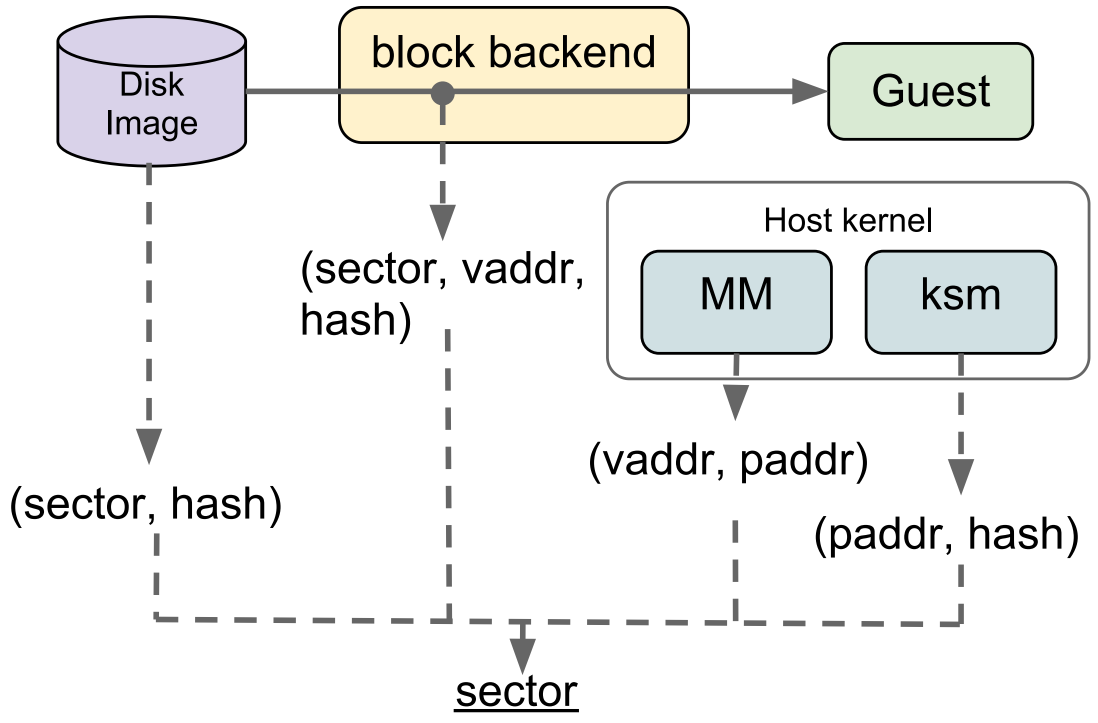
Static Analysis

- With running a pair of virtual machines, run KSM for a while, and collect several data and statistics
  - # of merged pages
    - Memory size
  - Merged pages characteristics
    - Zero-filled pages
      - To merge them is not good idea
    - Which pages are read from a disk image and stay as they are (*)
- Sector number list
  - Pages of (*) have a corresponding blocks on a disk image

How to Use a Sector Number List?

- We can preload a set of pages that correspond to sector numbers
- By doing so, we can merge a page being read from a guest to a preloaded one
  - directly with low cost
    - just check if the number exists
  - without tracking a page lifecycle
    - modified or not

How to Get a Sector Number List



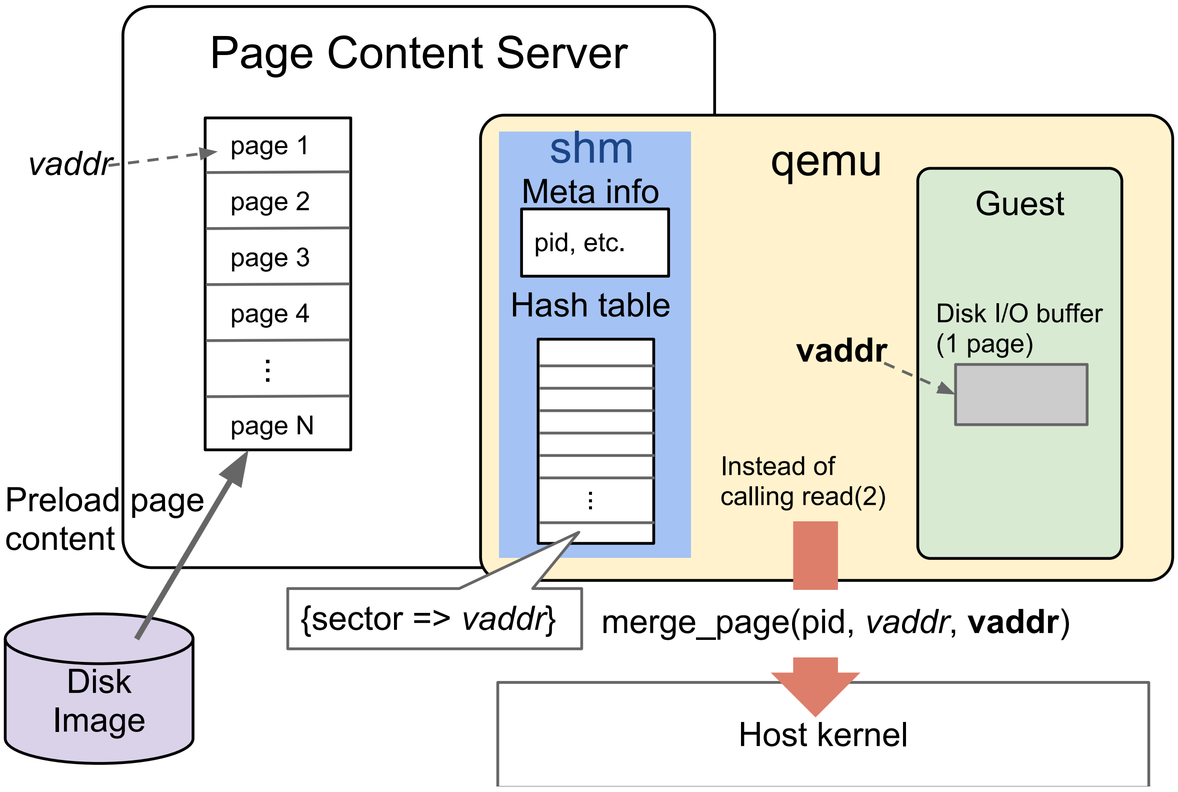
Implementation

- qemu-kvm-0.14.0
- linux-2.6.35.13 (support only EPT)
- crash-5.1
- analysis scripts
  - bash/ruby/python and crash extension
- Modifications
  - qemu-kvm: 1,500 LOC
  - linux: 2,000 LOC
- analysis scripts
  - bash/ruby/python: 1,000 LOC
  - crash extension: 500 LOC

Implementation

- Page content server
  - Preloads pages in accordance with a sector number list
  - Exports a hash table to quickly convert a sector number to a virtual address of a page
    - shm
- merge_page(s) syscall
  - called by qemu
  - Takes three arguments: pid, *vaddr*, **vaddr**
  - pid: of a page content server
  - *vaddr*: specifies a page in a page content server
  - **vaddr**: specifies a target page to be merged with a page of *vaddr*

Page Content Server



Merging Pages

- We implement page merging facility based on KSM
  - It pus merged pages in KSM a RB tree
  - We can run KSM as usual
- We also need to modify mm
  - To prevent EPT violation (host page fault for guest) from breaking merged pages

Security Concern

- merge_page(s) has a potential to steel arbitrary pages
- madvise(MADV_ALLOWMERGED)
  - specifies a memory region where can be merged by merge_page(s)
- A page content server uses this to expose page contents to others
  - Contents of a share CoW disk image are not usually hidden

Evaluations

1. The time of a single syscall execution
   - read vs. merge_page
2. Overhead of our method
   - Our method checks if merge-able or not for every I/O requests
3. CPU and memory usage
   - Comparison with KSM
   - Our method also runs KSM but it scans only non-zero pages

Execution time

- Compare our syscall with read syscall
  - merge 1 page vs. read 4 Kbytes
  - the average of one thousand repetition

read	1.60 μs
merge_page	0.17 μs

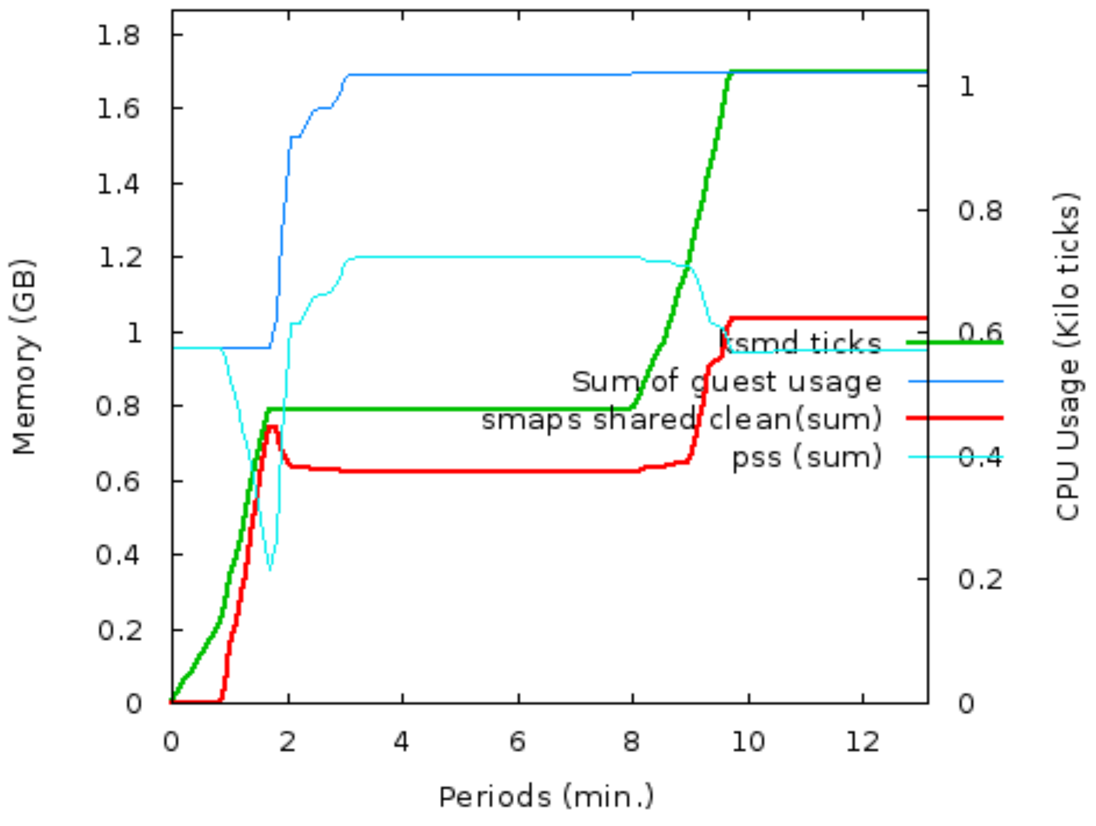
- Faster than read syscall because no data copy is necessary

Overhead (worst case)

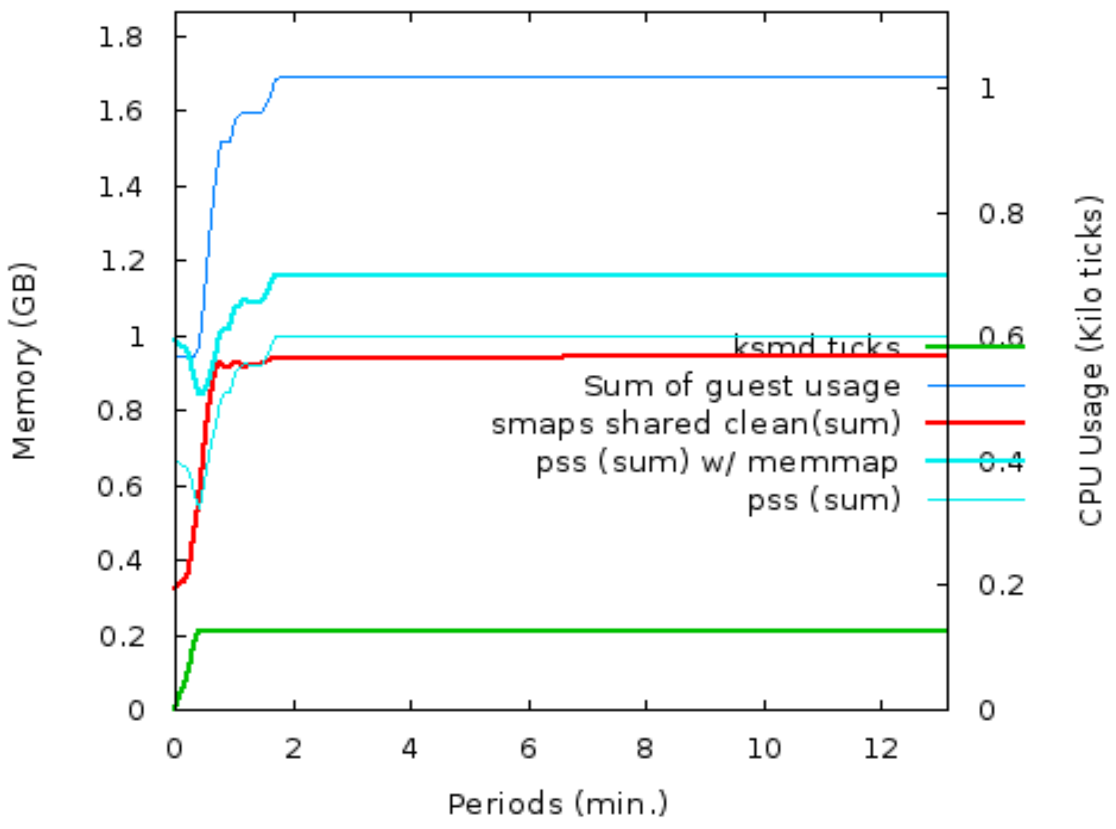
- Issues I/O requests which cannot be merge by our method at all
  - Overhead here is a cost to check if pages in question are merge-able

	Single request from a block driver of qemu	Elapsed time of `grep` in a guest
w/o our method	14.1 µs	6.57 sec.
w/ our method	14.5 µs	6.69 sec.
Overhead	2.8 %	1.8 %

CPU and Memory Usage



CPU and Memory Usage



Conclusions

- Our contributions
  - We proposed that static analysis of disk images is useful to support efficient memory merging facilities
  - We designed and implemented prototype to demonstrate our proposal
  - Experimental results have shown that using a sector number list and a page content server can reduce CPU consumptions to merge pages
- Future work
  - We will explore another usages of static analysis of disk images
    - Dynamic analysis and feedback loop