FUJITSU

shaping tomorrow with you

From mm-summit

Hiroyuki Kamezawa
Fujitsu Limited.

mm-summit

Linux Storage File and MM summit

April 18-19, 2013 - San Francisco

MM “Memory Management” summit
is co-located with Storage/File summits.
20+ engineers for mm-summit

LSF/MM summit report was done by Ric, today.
visit lwn.net for good articles.

Topics

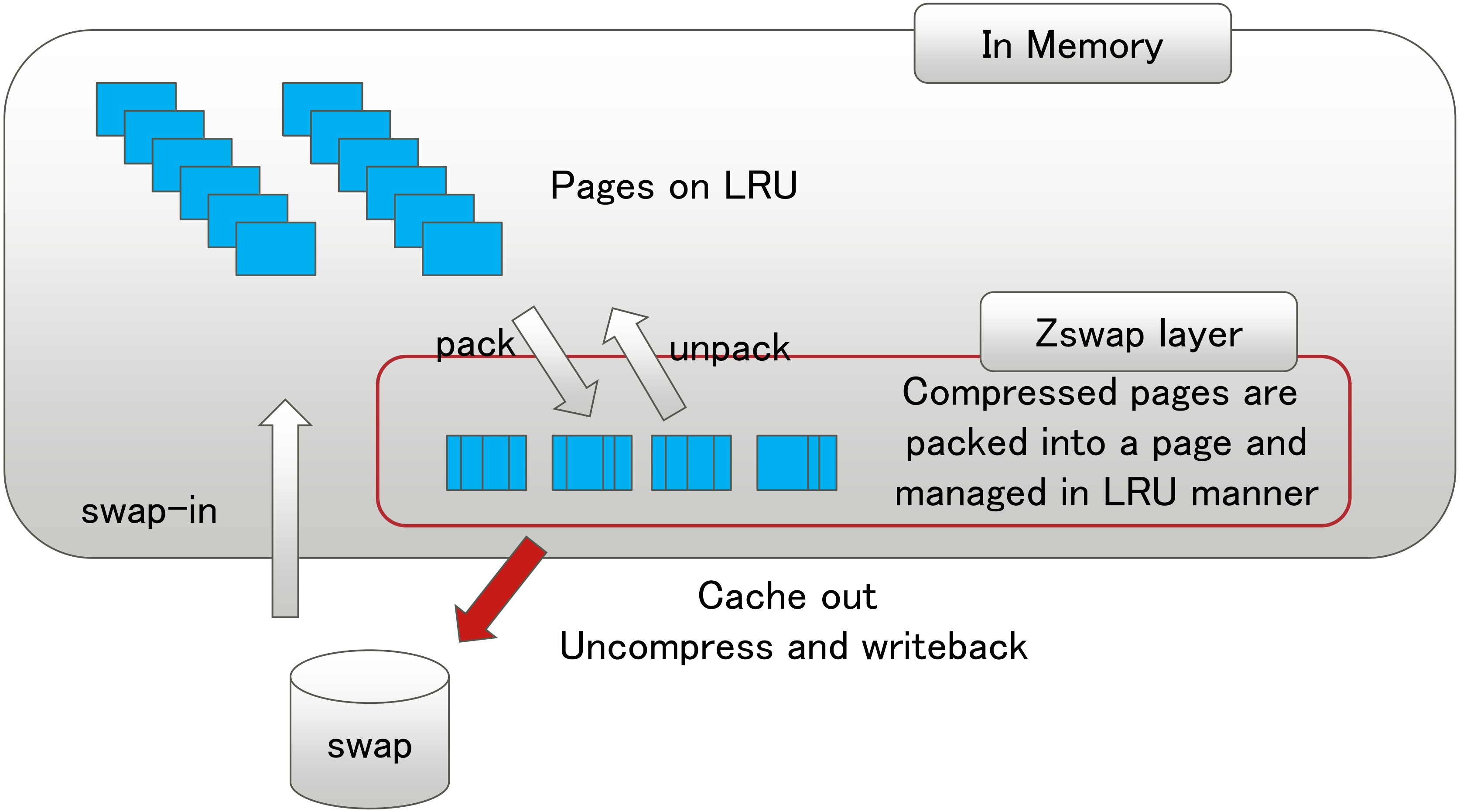
- Memory compression for swap
- volatile range of memory
- Misc. updates for better latency memory management.
- Swap improvements with SSD.
- OOM Killer redesign
- Memory cgroup and soft-limit-reclaim.
- Better page cache handling.

Memory Compression/background

- Motivation
  - swap-I/O is very slow and harms system performance.
  - Today, memory compression is much faster and low overhead than I/O.
- Methods
  - There are 3 functions.
    - zram - already merged. /dev/zram can be used for swap device.
    - zswap - not merged. compressed-cache layer between swap and memory.
    - zcache – not merged. compressed-cache layer between blkdev and memory.
- Discussion
  - zswap and zcache are similar functions but doesn't sharing codes much.
    - As memory allocator, zswap uses zsmalloc(). Zcache uses zbud().
  - Which should be merged ?
    - Finally, zswap was agreed to be merged.
      - Zcache is more complex than zswap. Zswap should go 1st.
    - Then, what is zswap ?

zswap

- A compressed-cache layer between memory and swap.



You can find articles in IBM’s developerworks or lwn.net.

volatile range of memory

- Motivation
  - When applications uses memory, free memory will decrease.
  - Even if there are unused memory, it seems “used” from the kernel.
  - Hinting from user should be helpful.
- Method
  - New syscall “vrange” as volatile-range.
- Use Case
  - Userland caching, allow eviction by the kernel
  - Discard reloadable information(hidden rendered image of jpeg)
  - Better malloc() by lazy freeing of heap (glibc malloc 20x times faster in a trial)
- Discussion
  - Implementation: vrage() avoids using usual vm range management mechanism and doesn’t take mmap_sem() for better performance.
  - Sematics : FILE/ANON, Shared/Private, alloc after syscall..
  - Current LRU routine can’t handle “ready-to-discard” pages, well.

vrange()

Proposed Interface

- fvrange(fd, start_addr, length,mode,flags,&purged)

Mostly 2 functionality

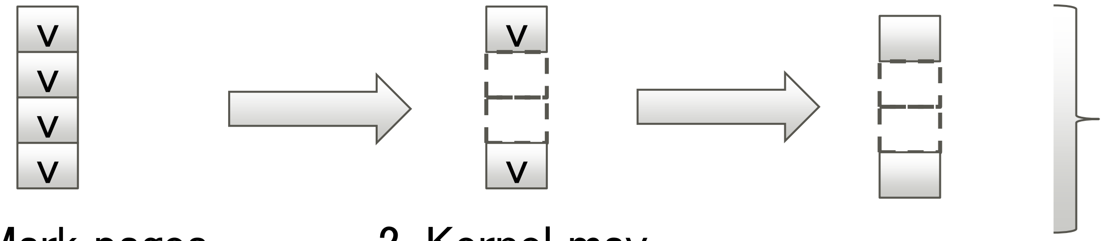
1. Mark specified range as “volatile”

   - Kernel can purge “volatile” memory if necessary

2. Mark specified range as “non-volatile”

   - Kernel returns info if pages were purged while it’s marked as volatile.

The kernel will send SIGBUS if the user touches “purged” page.



1. Mark pages
As volatile

2. Kernel may
purge pages
If there is memory pressure

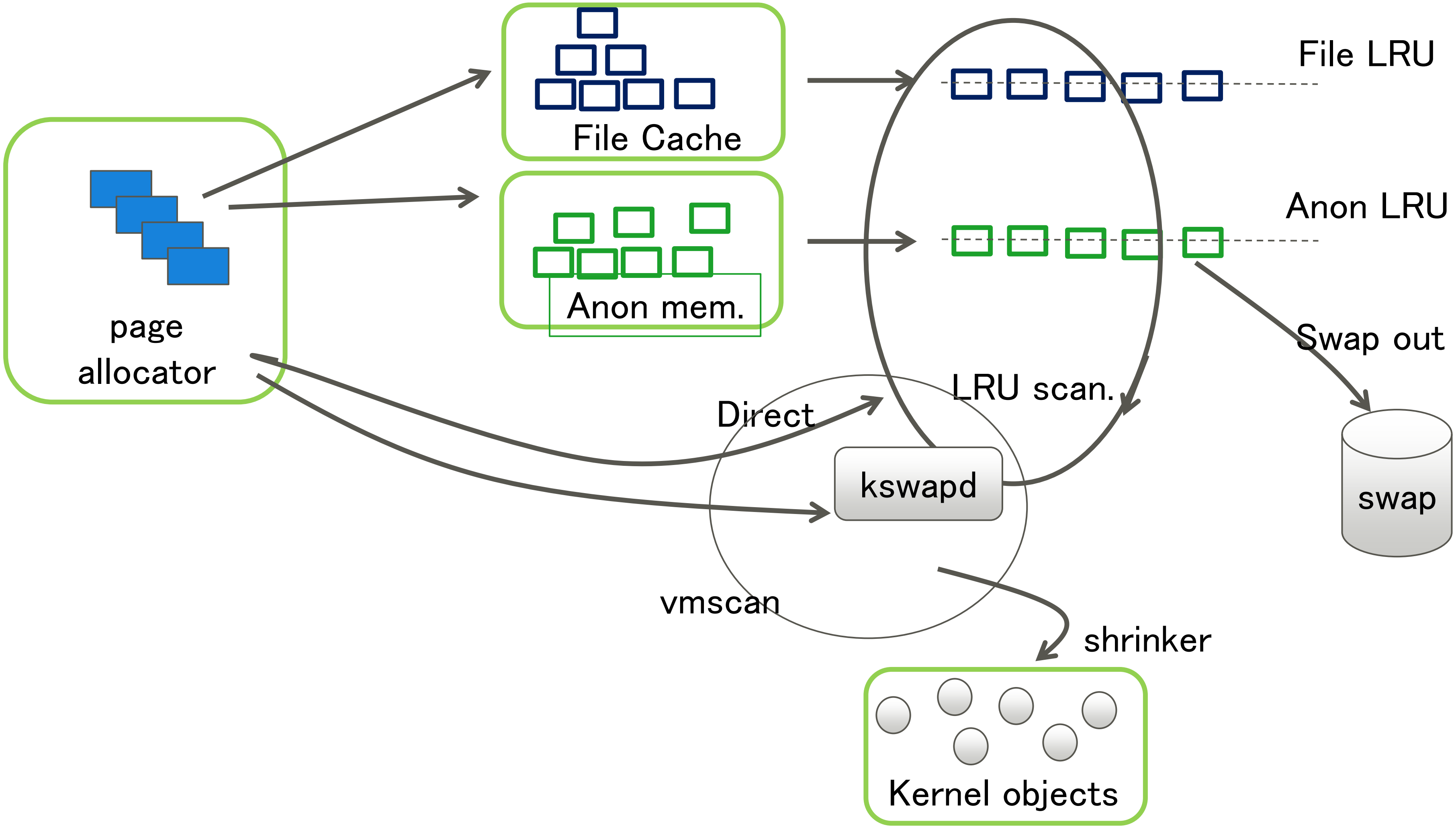
3. unmark

Part of pages
are purged.
Some are not.

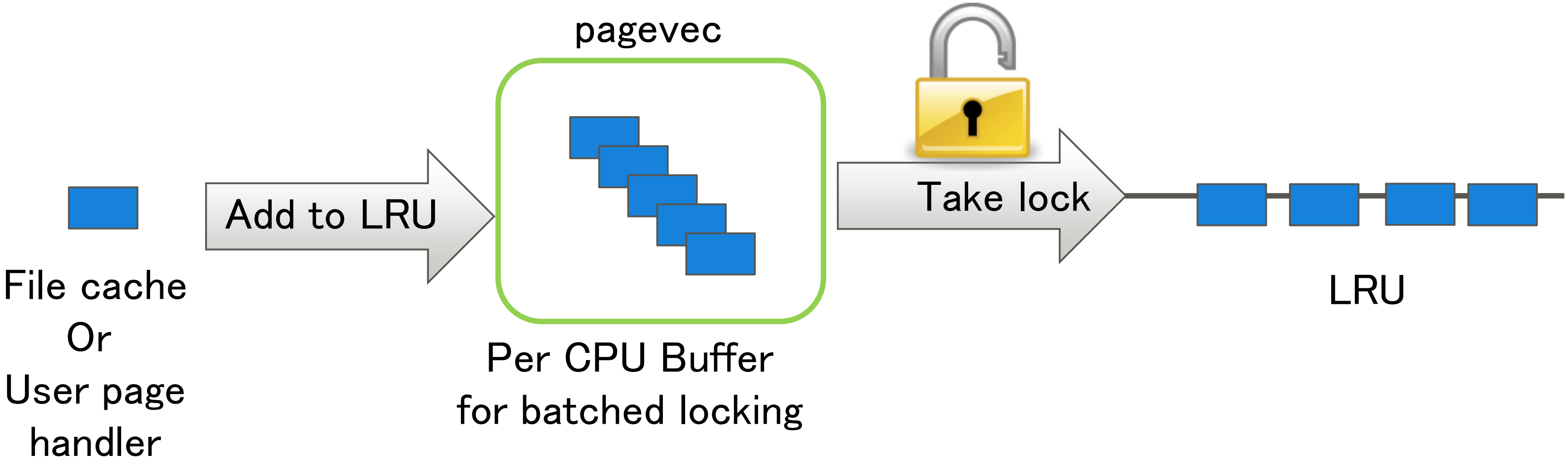
Towards better mm.

- Several topics…
  - Pagevec is harmful.
    - Pagevec is a per-cpu caching for batched LRU ops.
    - Page cannot be marked as “Active”, if it’s in pagevec.
    - Remove or reduce it.
  - kswapd needs to wake up earlier.
    - Then, reduce direct reclaim.
    - Reduce calling shrinker ….call shrinker only by kswapd.
  - Transparent hugepage + compaction may delay memory allocation.
  - Guarding ext4 block bitmap not to be purged.
  - Tools for measurement performance?
    - Ftrace
  - Batching TLB flush.

Memory reclaim.

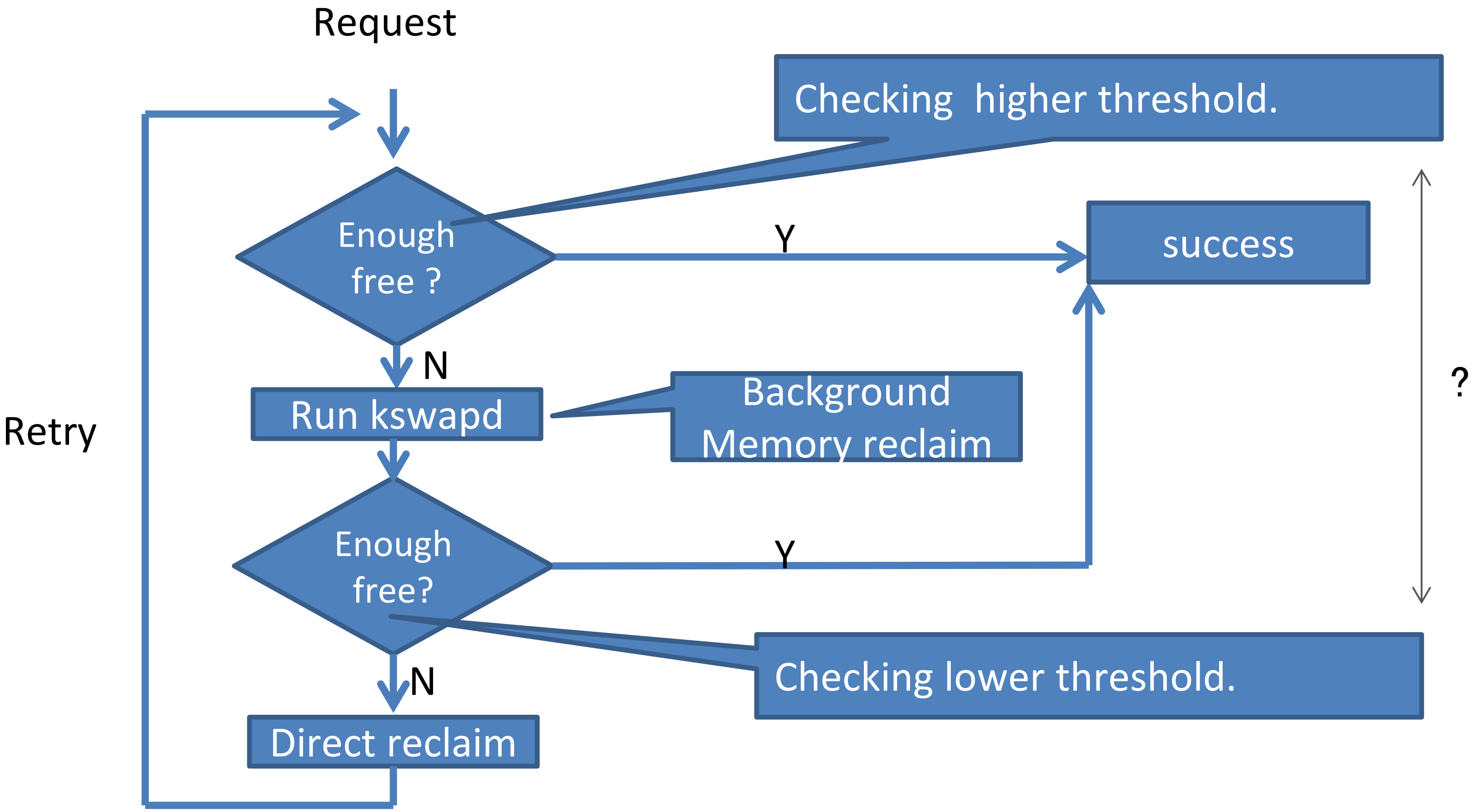


Problem of Pagevec.



While pages are in pagevec, it’s not recognized as they are on LRU.
Because of this, some page aging information cannot be recorded
to the page correctly.

Page allocation , kswapd, direct reclaim.


Request
Checking higher threshold.
Enough free ?
Y
success
N
Run kswapd
Background Memory reclaim
Retry
?
Enough free?
Y
Checking lower threshold.
N
Direct reclaim

Distance between 2 thresholds are important.

Swap improvements with SSD.

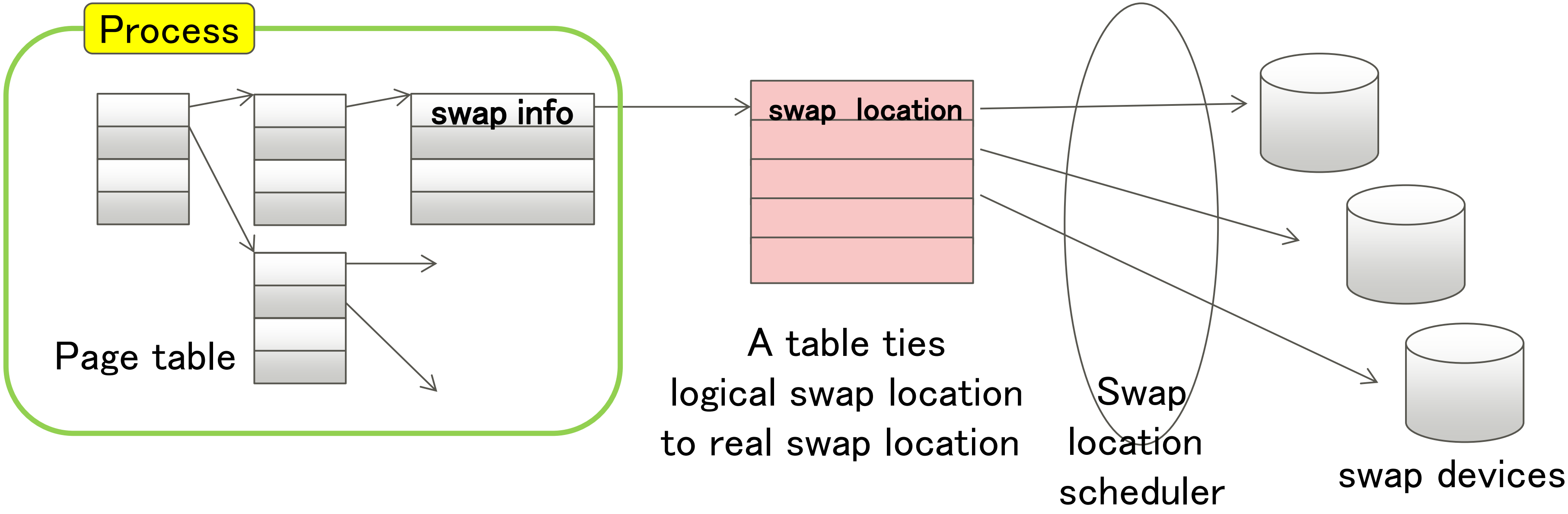
- Motivation
  - At using SSD for swap, performance is not good rather than expected.
- Method
  - Modify LRU to be SSD aware.
  - update swap subsystem SSD aware.
- Discussion

  Hi-jacked by more generic swap discussion…

  - If swap is enough quick and smart for random-access, anon memory can be swapped-out easily.
    - Current LRU thinks “page cache” is easier to discard.
    - If file systems are on rotated device….

    => LRU should be aware of disk speed of backing store.
  - Swapping out huge-page as it is ?
  - Swap hierarchy including zswap
  - Swap indirect table and assign better swap location ? Defrag ?.

Swap indirect table



This allows

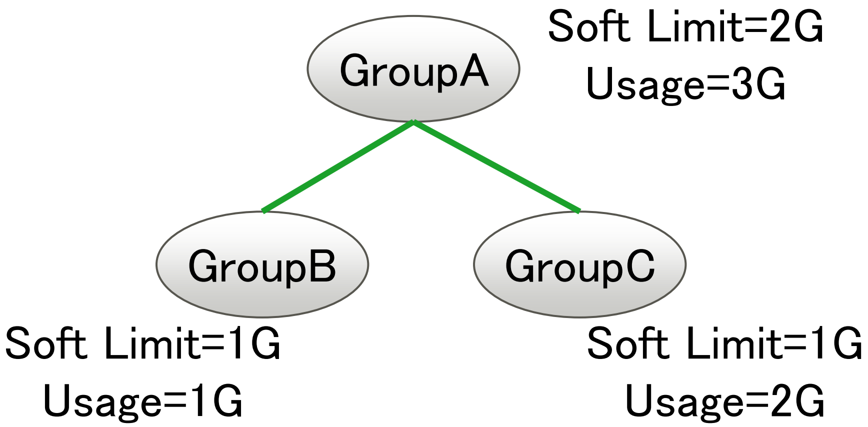
- Lazy allocation of real swap entry. i.e. choose the best device with regard to page’s activity.
- migrating swap locations.

OOM Killer

- Motivation
  - OOM-Killer kills processes automatically at out-of-memory.
  - Everyone dislikes OOM-Killer.
- Discussion
  - Policy module for OOM-Killer
  - Notification from the kernel to user-land
    - User-land OOM Killer …… won’t work.
  - Notification to some kernel subsystem
    - Trigger Kernel module.
    - Trigger In-Kernel script.
  - No conclusion.

Memory cgroup and soft-limit-reclaim

- Motivation
  - Re-implmenting memory cgroup’s softlimit reclaim.
- Method
  - Replace almost all existing codes with simpler ones.
  - More integration with vmscan codes.
- Discussion
  - Semantics : how to handle hierarchy.



GroupA

Soft Limit=2G
Usage=3G

GroupB

Soft Limit=1G
Usage=1G

GroupC

Soft Limit=1G
Usage=2G

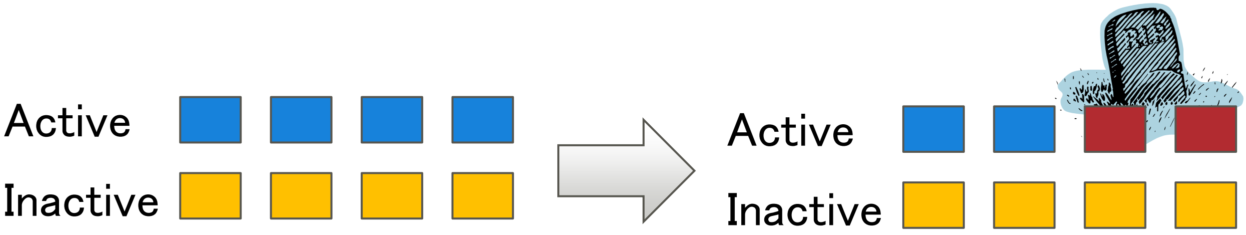
In hierarchy, GroupA’s usage includes propagated usage of children, B and C.

Because GroupA is over softlimit

- We should cull memory from both of children
- We should cull memory from GroupC 1st.

Better reclaim control

- Motivation
  - 2 Level LRU(Active/Inactive List) cannot work well in some case.
- Method
  - Recording the distance between page-in/out and make use of the info.
- Discussion
  - Recording time info in the radix-tree of page cache.



as an application runs,
Frequently accessed
Data will go to active lit

If the application ends, some
data in Active list will be dead.

Even if dataset size is
larger than inactive list,
unused cache in active list
cannot be purged easily.

Changing inactive/active ratio dynamically･･･

FUJITSU

shaping tomorrow with you