Cgroup
And
Memory Resource
Controller

Japan Linux Symposium 19/Nov/2008

Kame
kamezawa.hiroyu@jp.fujitsu.com

Contents

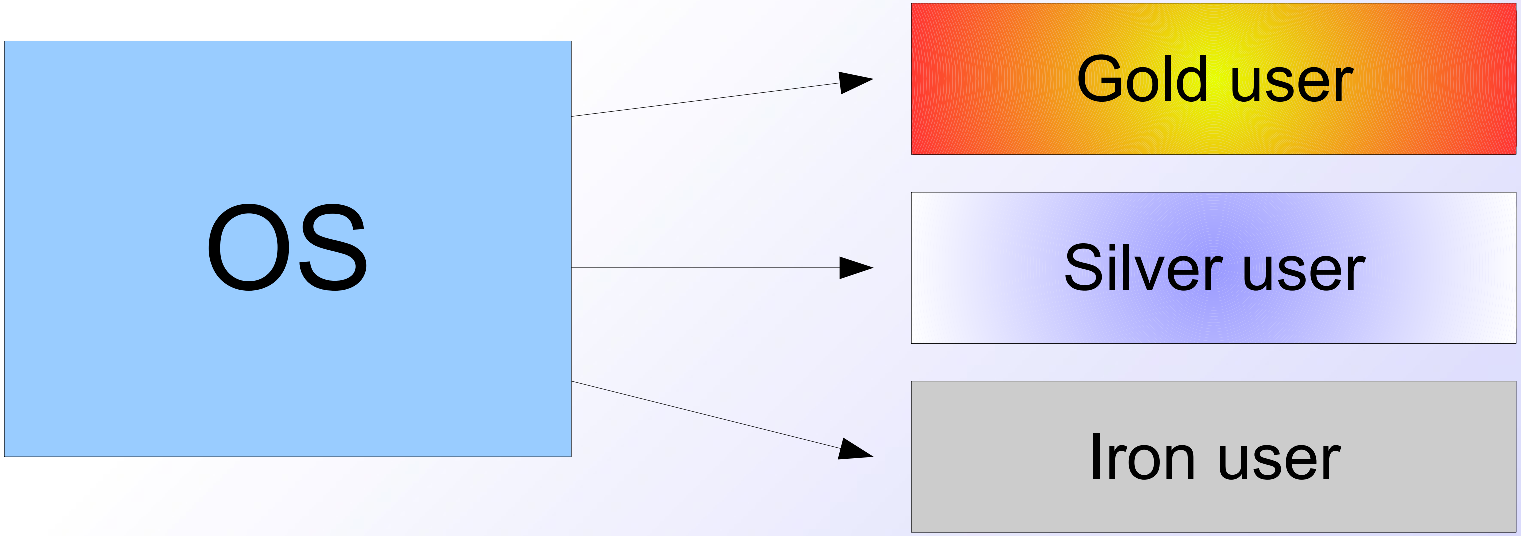
- Background
- Cgroup and subsystems
  - Subsystems Quick Tour
- Memory Resource Controller
  - Now and Future
- Demonstration

Background(1)

- In old ages, single-user system, all resource are under control of users. Resource control was simple.
- After multi-user system, Operating System(OS) controls resource instead of users and shares it in appropriate way by “Scheduling Algorithm”

Background(2)

- Scheduling algorithm works well ?
  - Depends on workload.
- In '80-90 ages, many studies for “resource control” are done. The operator can divide OS's resource into several groups.



Background(3)

- In '00 ages, interests of study are moved to security and Web.
- Cpu/Netowork getting faster and faster
- Server system is made by pc-cluster not by a big iron.
- Where is resource should be divided ?.....

But....

Background(4)

- In these days
  - ➔ Cpus are multi-core. SMP is usual machine.
  - ➔ Memory is getting cheaper and cheaper.
  - ➔ Virtual Machine is now popular system. Used in production.

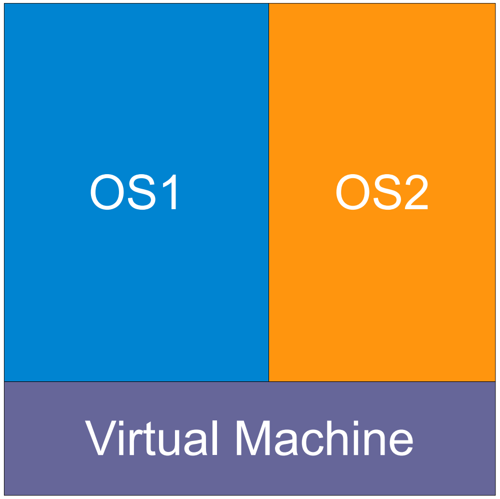
How about OS level control ?

Background(5)

- Proprietary Operating Systems (UNIX) provides “resource management system”
- Popular design is 3-level.
  - Virtualization by Virtual Machine
  - Divide system into independent blocks. (container, jail)
  - Precise and Flexible control per group of processes.

3Levels of resource control

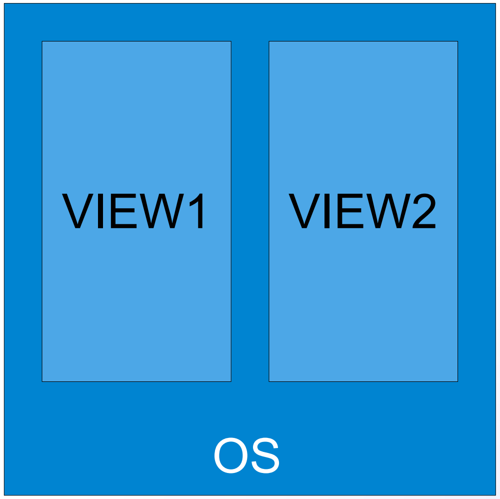
Isolation by
Virtual Machine


OS1

OS2

Virtual Machine

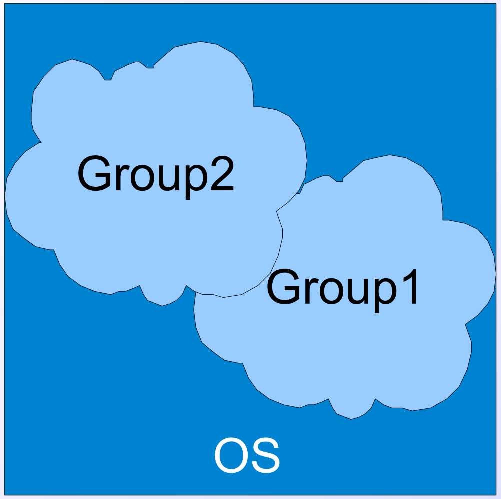
Isolation by OS(Virtual OS)
(Container/Jail)


VIEW1

VIEW2

OS

Flexible Resource Control


Group2

Group1

OS

	Virtual Machine	Container	RC
Performance	Not good	Very good	Good
Isolation/Security	Very good	Good	Not good
Runtime Flexibility	Not good	Good	Very good
Maintenance	Not good	Good	Good

About Linux ?

- Out-of-tree controls
  - Virtuozzo/OpenVZ
  - Linux Vserver

  need out-of-tree kernel patches.
- Several proposals are done and

  Paul Menage(google) finally implemented "cgroup" as base technology for control.

Contents

- Background
- Cgroup and subsystems
  - Subsystems Quick Tour
- Memory Resource Controller
  - Now and Future
- Demonstration

Cgroup

- Cgroup is a method to put processes into groups.
- It was “container group” but is “control group”
- Has following characteristics
  - Implemented as pseudo filesystem.
  - Grouping can be done by a unit of thread.
  - Many functions are implemented as “subsystem”
  - A child process is automatically put into a group under which its parent is.

Cgroup interface

1.mount

```
# mount -t cgroup none /cgroup -o subsystem
```

2.mkdir

```
# mkdir /cgroup/group01
```

3.attach

```
#echo <PID> > /cgroup/group01/tasks
```

After Work.

4.rmdir

```
# rmdir /cgroup/group01
```

Cgroup Subsystems(1)

- Can be specified as mount option of cgroupfs.

  ex) #mount -t cgroup none /cgroup -o cpu
- 2 types of subsystem in general

  A) Isolation and special controls

  cpuset, namespace, freezer, device, checkpoint/restart

  B) Resource control

  cpu(scheduler), memory, disk i/o
- Each subsystem can be mounted independently. => next

Cgroup subsystems(2)

- Ex) mount each subsystem independently

```
# mount -t cgroup none /cpu -o cpu
# mount -t cgroup none /memory -o memory
# mount -t cgroup none /devices -o device
```

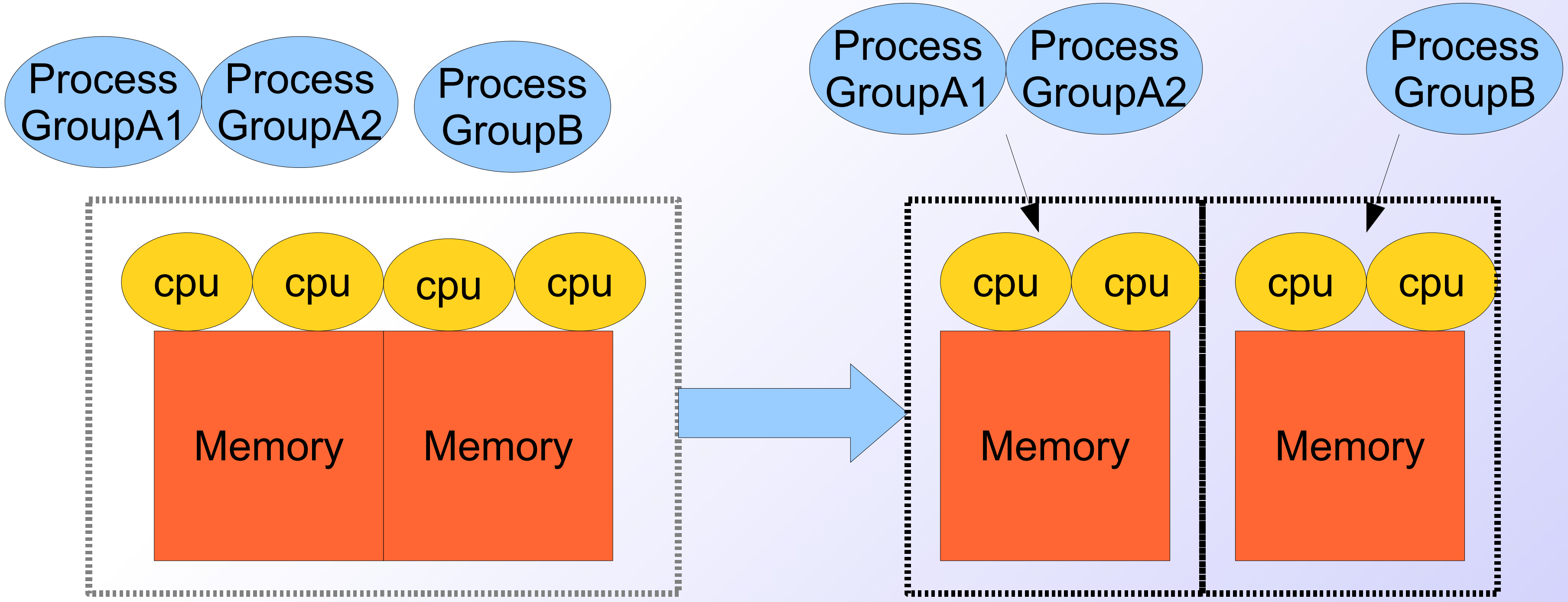
- Ex) mount at once

```
# mount -t cgroup none /xxx -o cpu,memory
```

- /proc/cgroups
- /proc/<PID>/cgroups

Cpuset (feature for isolation)

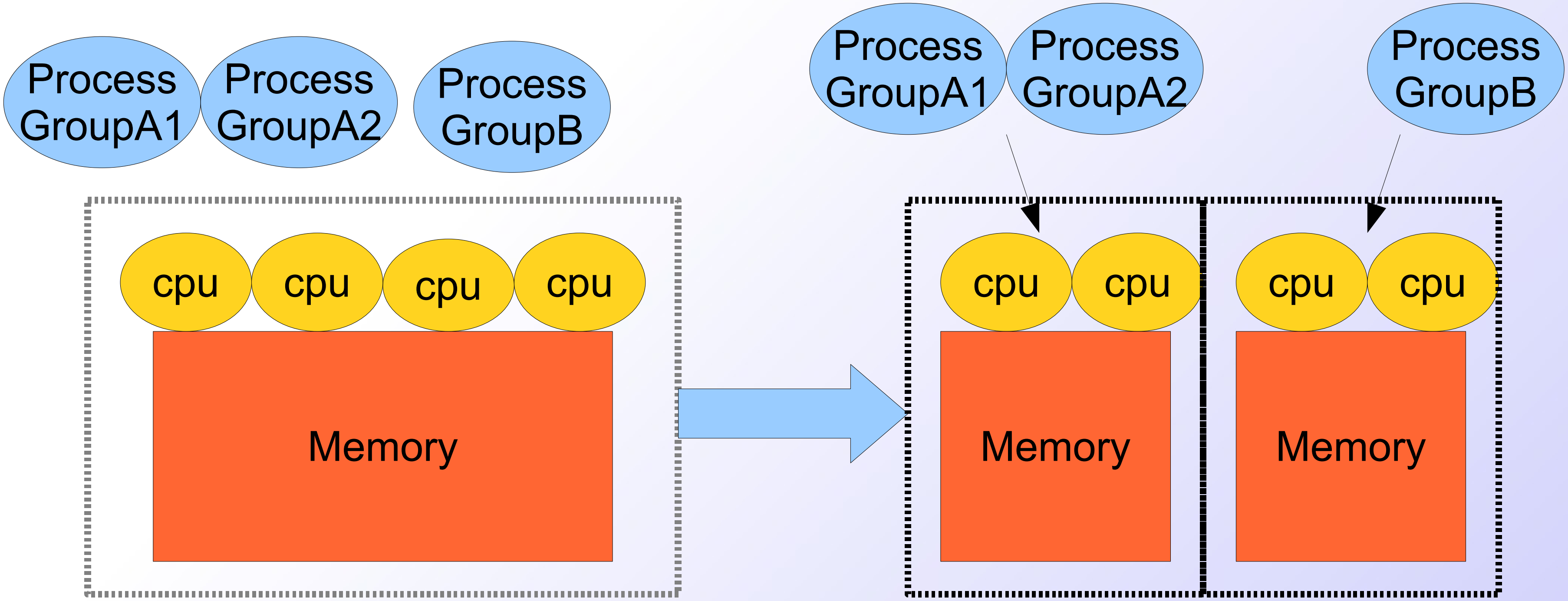
- Cpuset if for tying processes with cpu and (NUMA) memory.
- Used in production



Cpuset + Fake NUMA

- For SMP, Fake-NUMA is available(x86-64)

numa=fake=2

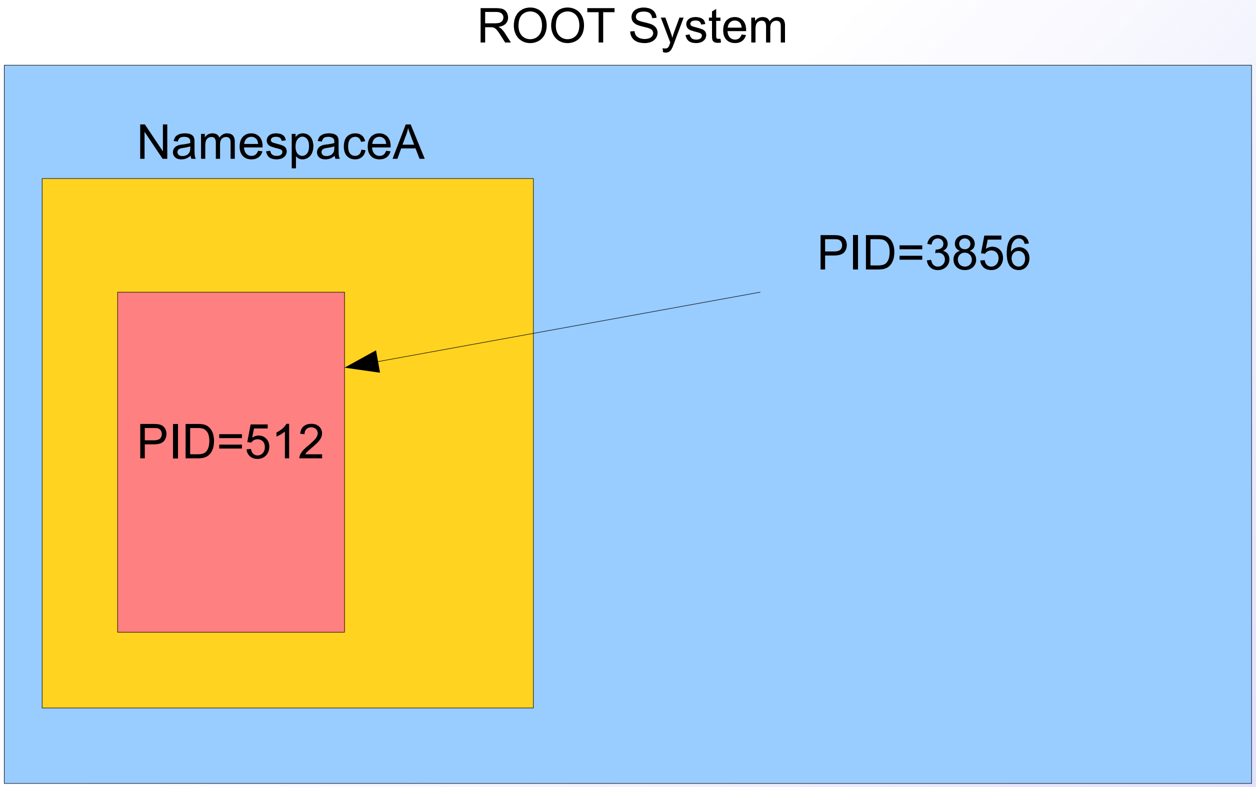


Namespace(feature for isolation)

- Namespace is for showing private view of system to processes in cgroup. Mainly used for OS-level virtualization. This subsystem itself has no special functions and just tracks changes in namespace via clone()/unshare().
  - UTS namespace (for uname())
  - IPC namespace (for SYSV ipc)
  - USER namespace (for UID/GID)
  - PID namespace (for PID)

/cgroups/(...)/node_<pid>/node_<pid>/....

Namespace(cont.)



Freezer(feature for control)

- Freezer cgroup is for freezing(stopping) all tasks in a group.

```
#mount -t cgroup  none /freezer -o freezer
....put task into /freezer/tasks...
#echo FROZEN > /freezer/freezer.state
#echo RUNNING > /freezer/freezer.state
```

Device(feature for isolation)

- Device cgroup as device-white-list.
- A system administrator can provide a list of device can be accessed by processes under group.
- Allow/Deny Rule.
- Allow/Deny : READ/WRITE/MKNOD

Device (Cont.)

Limits access to device (file system on device) of tasks in specified cgroup.

```
#echo [b|c] MAJOR MINOR r/w/m > devices.allow
```

```
# cat devices.list to see list
```

checkpoint/restart(feature for control)

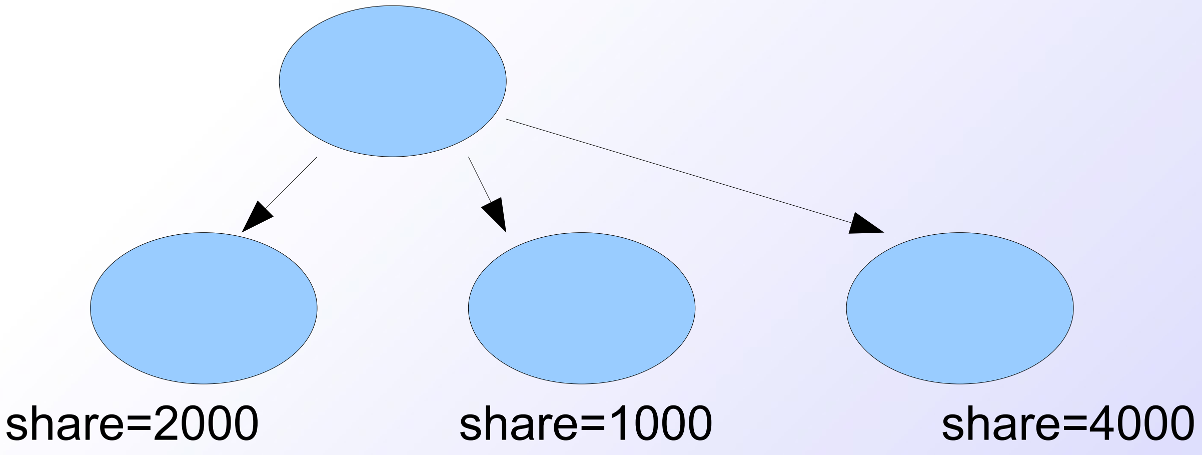
- Save all process's status in a cgroup to a dump file, restart it later. (or just save and continue.)
- For allowing “saved container” moved between physical machines.(as VM can do.)
- Dump all process's image to a file.

State: RFC. (not in -mm)

CPU(for resource control)

- Share cpu bandwidth between groups by group scheduling function of CFS(a new scheduler)
- Mechanically complicated
- Latency problem still ? (default=n, now)

(bandwidth is well controlled.but..)



Memory(for resource control)

- For limiting memory usage of processes.
- Just limit LRU pages (anonymous and file cache)
- No limits for further kernel memory
  - maybe in another subsystem if needed
- Details in later.

Disk I/O(for resource control)

- 6~7 proposals have been done by many players.
- Recently, it seems they will be able to make a consensus.
- In recent discussion,
  - Developing 2-level scheduler will break something.
  - Developing per-io-scheduler cgroup callback.
  - Supporting both of “weight/share” and “limit”

Contents

- Background
- Cgroup and subsystems
  - Subsystems Quick Tour
- Memory Resource Controller
  - Now and Future
- Demonstration

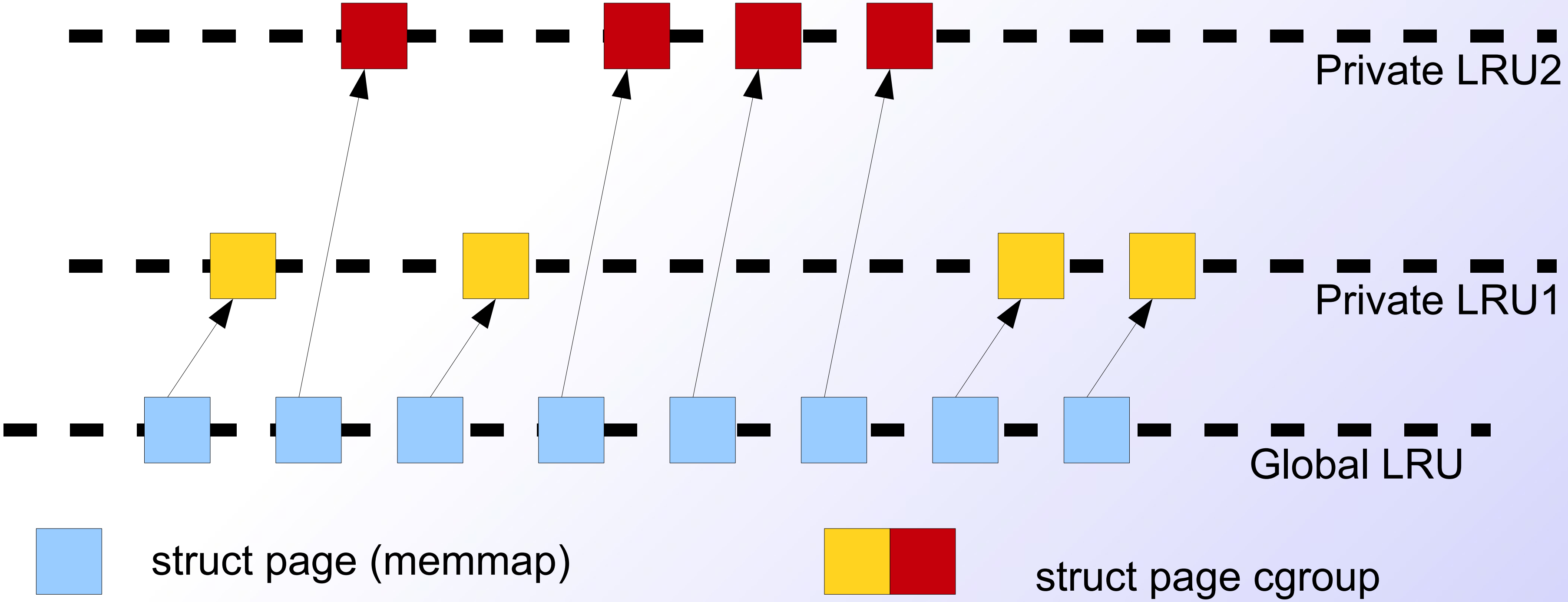
Features of memory resource controller

- Limiting usage of anon and file-caches.
- Optionally limiting usage of memory+swap.

  (now under test)
- Remaining page caches in obsolete cgroup can be dropped.

Account logic(1)

- page_cgroup, new struct per page, is used for tracking pages.
- Memory resouce controller has its own LRU.



Account logic(2)

- A page is accounted when
  - Anonymous page is allocated (page fault)
  - File cache is added. (add to page cache)
- When account_swap=enabled
  - Swap entry is also accounted.
  - Swapped-in page goes back under its original allocator.

Limiting memory

- Account logic works even if cgoup is not mounted. (To disable, pass boot option.)
- When memory usage reaches limit, the kernel try to reduce memory usage as global LRU does by using private LRU.

Limiting memory (cont.)

```
#mount -t cgroup none /memory -o memory

#mkdir /memory/group01

#echo 128M > (...)/memory.limit_in_bytes

#echo $$ > (...)/tasks
#cp veryverybigfile  tmpfile
```

(memory usage doesn't exceeds 128M)

```
#echo $$ > /memoy/tasks (moves back to..)
#rmdir group01
```

Out-Of-Memory(OOM)

- At OOM, a process in the cgroup will be killed by oom-killer.
- Special OOM handler development is in plan.
- If global LRU hits OOM, usual OOM killer is invoked.

Limiting Mem+Swap

- Now, tested under -mm kernel.
- Limiting usage of Memory+Swap.

```
# echo 512M > memory.limit_in_bytes.
# echo 1G > memory.memsw.limit_in_bytes.
```

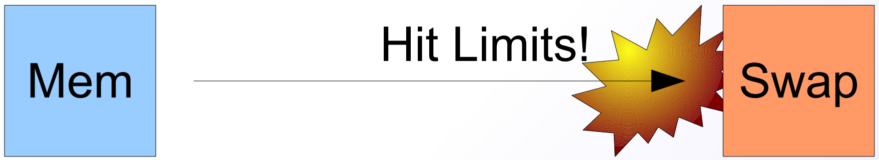
In above case, memory usage will be limited to 300M when swap usage is 700M.

- Can be disabled by boot option.

Why Mem+Swap ?

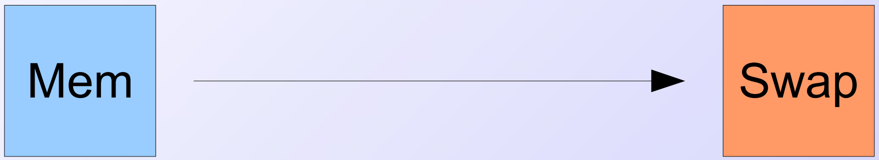
- “swap” controller can be worked as a kind of mlock(). This is bad.
- In Mem+Swap controller, global LRU will not be affected by Mem+Swap controller.

Have to charge against swap



Swap controller

No changes in count



Mem+Swap

Overhead

- Implicitly accounted(means overhead) even when not mounted. (can be disabled by boot option)
- My personal goal is 3~5%.(My boss's request is 3% ;)
- Unixbench on x86-64/8cpu/2.6.28-rc4mm, bigger is better.

test	disabled	enabled
Execl	1778	1731
shell(8)	2262	2207
Arithmetic	1558482	1557442
File Read/Write	773977 / 109065	751117 / 109092
C compier	1193	1165

TODO

- Hierarchy support
- User Land Tools!
- Stabilization/optimization/clean up
- Support for vm parameters,
  - dirty_ratio , swapiness, etc....
- Fix LRU algorithm to be the same as global's
- Documentation
- And Disk I/O controller will be necessary.....

Memory Resource Controller: this year

- Almost one year of development.

2.6.25

2.6.28-rc4mm

Special thanks to
Balbir Singh(IBM) and Daisuke Nishimura(NEC), Hugh Dickins(Veritas)
and all folks

Will be bigger ....should be careful about maintenance

(from http://sozai-free.com/)

Questions?

2.6.X?