FUJITSU

shaping tomorrow with you

Memory Cgroup
what's going on

2012.06.08

KAMEZAWA Hiroyuki
kamezawa.hiroyu@jp.fujitsu.com

Agenda

- History
- Features of memcg in linux-3.5-rc1.
- Performance brief.
- News and TODOs.

Birth of Memory cgroup

FUJITSU

- 2008.02.07 v2.6.25-rc1.

```
commit 8cdea7c05454260c0d4d83503949c358eb131d17
Author: Balbir Singh <balbir@linux.vnet.ibm.com>
Date:   Thu Feb 7 00:13:50 2008 -0800

    Memory controller: cgroups setup

    Setup the memory cgroup and add basic hooks and controls to integrate
    and work with the cgroup.

    Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
    Cc: Pavel Emelianov <xemul@openvz.org>
    Cc: Paul Menage <menage@google.com>
    Cc: Peter Zijlstra <a.p.zijlstra@chello.nl>
    Cc: "Eric W. Biederman" <ebiederm@xmission.com>
    Cc: Nick Piggin <nickpiggin@yahoo.com.au>
    Cc: Kirill Korotaev <dev@sw.ru>
    Cc: Herbert Poetzl <herbert@13thfloor.at>
    Cc: David Rientjes <rientjes@google.com>
    Cc: Vaidyanathan Srinivasan <svaidy@linux.vnet.ibm.com>
    Signed-off-by: Andrew Morton <akpm@linux-foundation.org>
    Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>
```

Way to RHEL6(2.6.32).

FUJITSU

- Some basic features are added.
  - Per-zone LRU, OOM handling, Swap, hierarchy support, softlimit etc...
- Major committers
  - Balbir Singh
  - Daisuke Nishimura
  - Hugh Dickins
  - KOSAKI Motohiro
  - KAMEZAWA Hiroyuki

Performance fixes(...2.6.36)

FUJITSU

- In this era, many many atomic ops were removed.
- Some new features.
  - Moving charge at task moving.
  - Threshold notifier (by embedded guy.)
- Major commiters
  - Daisuke Nishimura
  - Kirill A. Shutemov
  - KAMEZAWA Hiroyuki

New commiters (..v3.0)

FUJITSU

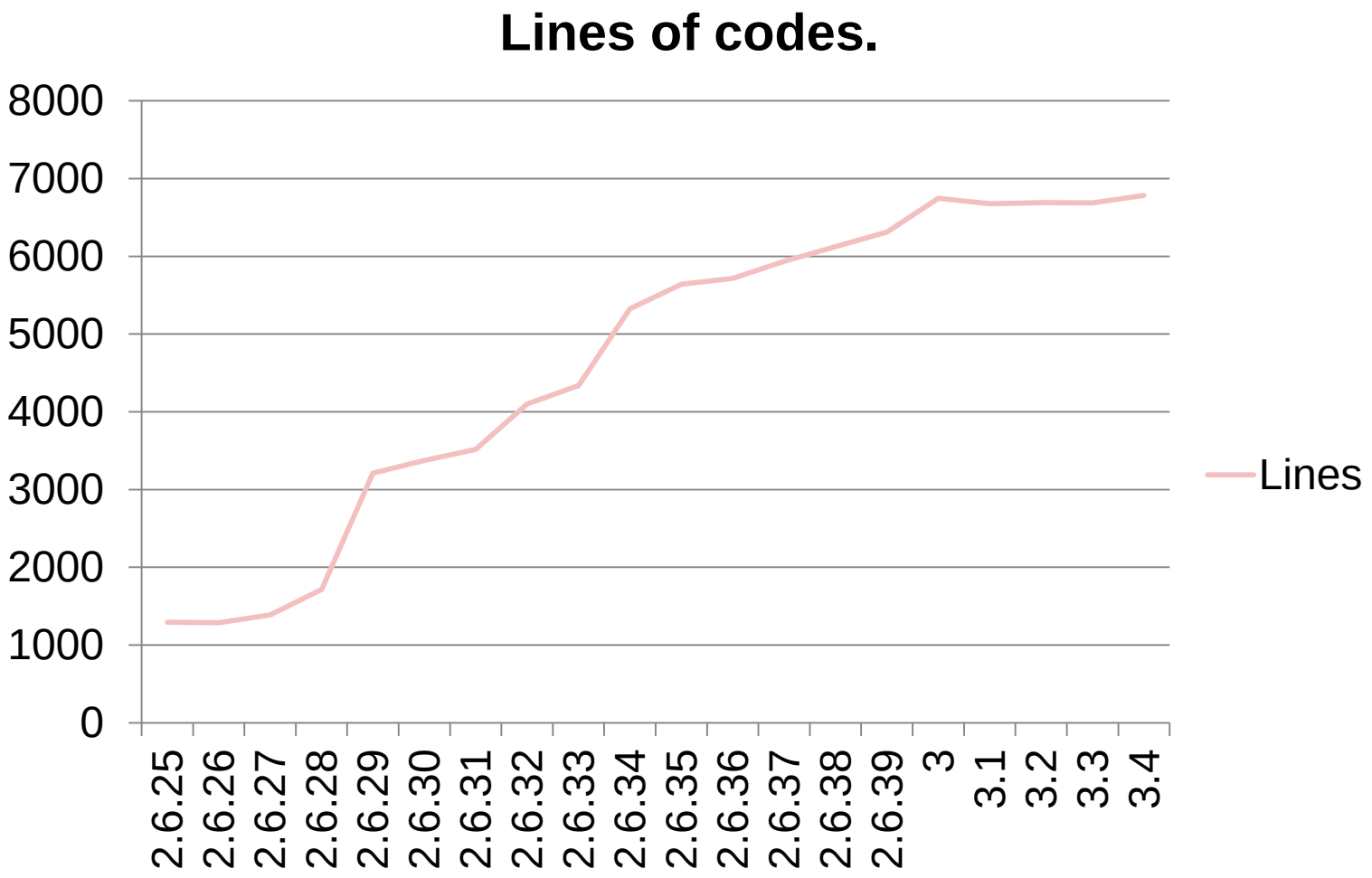
- Biggest changes in this era was new skilled committers
  - Johannes Weiner
  - Michal Hocko
  - Ying Han (+ Google team)
- Many refactoring and bug-fixes
  - Johannes and Michal are MAINTAINER now.
- More statistics

Renewal implementation (..v3.5-rc)

FUJITSU

- Hard works by Johannes and Hugh.
  - Reduced 60% of memory overhead.
- New accoutings
  - Tcp memcontrol
  - Huge Page limiting
- Recent Commiters
    - Johannes Weiner, Hugh Dickins, Glauber Costa, Michal Hocko, Konstantin Khlebnikov, Kirill A. Shutemov, KAMEZAWA Hiroyuki

Lines of codes.



Memory cgroup files (RHEL6)

```
[root@rx100-1 kamezawa]# ls /cgroup/memory/A/
cgroup.procs                    memory.move_charge_at_immigrate
memory.failcnt                  memory.soft_limit_in_bytes
memory.force_empty              memory.stat
memory.limit_in_bytes           memory.swappiness
memory.max_usage_in_bytes       memory.usage_in_bytes
memory.memsw.failcnt            memory.use_hierarchy
memory.memsw.limit_in_bytes     notify_on_release
memory.memsw.max_usage_in_bytes tasks
memory.memsw.usage_in_bytes
[root@rx100-1 kamezawa]#
```

Memory cgroup files (3.5-rc1)

```
[root@rx100-1 kamezawa]# ls /cgroup/memory/TestCgroup/
cgroup.clone_children                memory.memsw.max_usage_in_bytes
cgroup.event_control                 memory.memsw.usage_in_bytes
cgroup.procs                         memory.move_charge_at_immigrate
memory.failcnt                       memory.numa_stat
memory.force_empty                   memory.oom_control
memory.kmem.tcp.failcnt              memory.soft_limit_in_bytes
memory.kmem.tcp.limit_in_bytes       memory.stat
memory.kmem.tcp.max_usage_in_bytes   memory.swappiness
memory.kmem.tcp.usage_in_bytes       memory.usage_in_bytes
memory.limit_in_bytes                memory.use_hierarchy
memory.max_usage_in_bytes            notify_on_release
memory.memsw.failcnt                 tasks
memory.memsw.limit_in_bytes
```

memcg git tree

- Memory cgroup development tree maintained by Michal Hokko
  - git://github.com/mstsxfx/memcg-devel.git

Features of memory cgroup

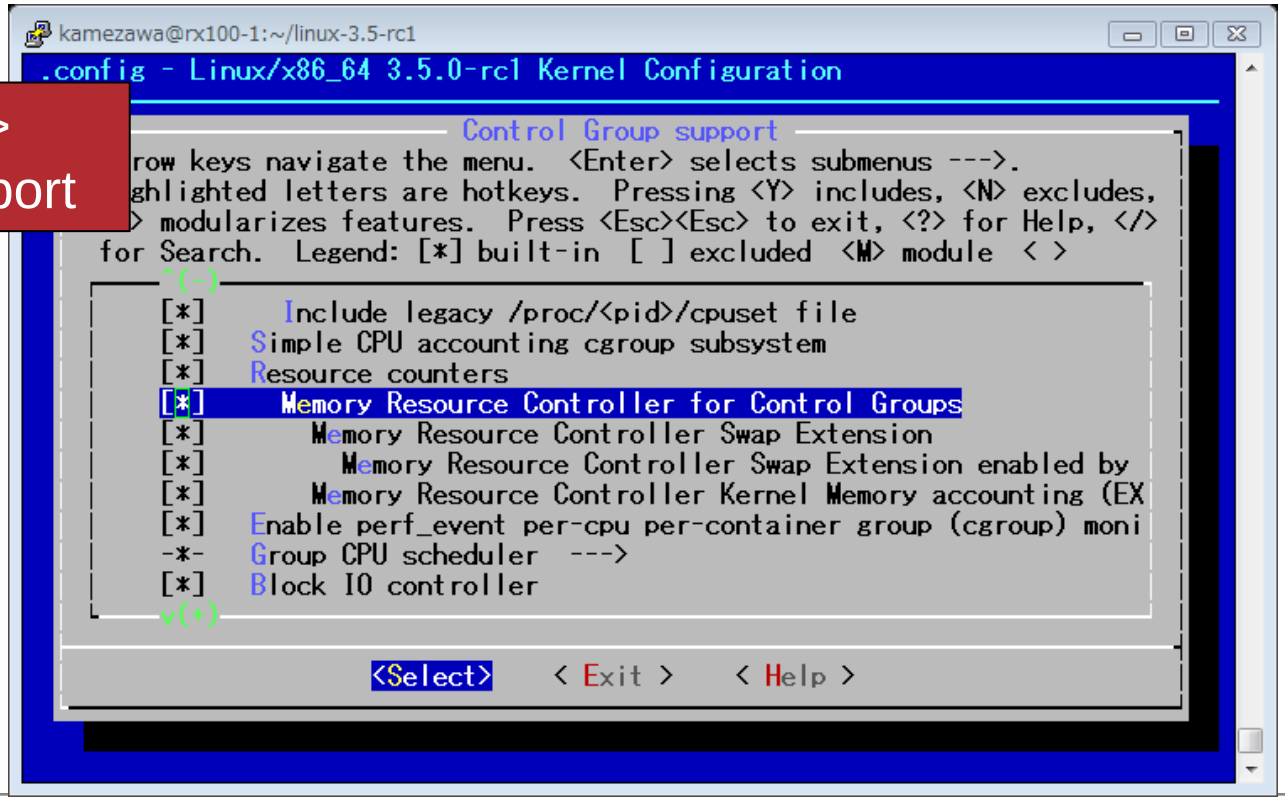
FUJITSU

- Agenda
  - Requirements/Basics
  - Per-memcg memory/swap limiting
  - Hierarchical or non-Hierarchical(Flat)
  - LRU implementation
  - Threshold/OOM notiffier
  - Tcp memcontrol

Requirements

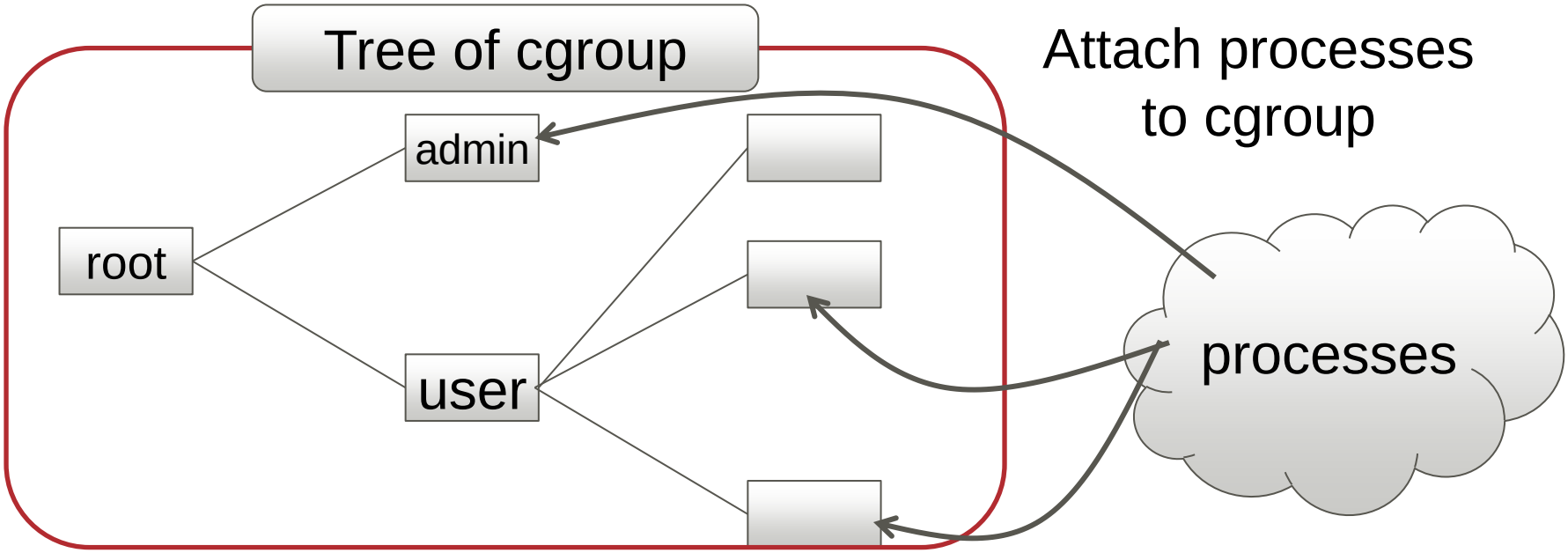
- Works on all system CONFIG_MMU=y
- Uses 16bytes/PAGE_SIZE(4096bytes) on 64bit system to record information

General setup=>
Control Group support



cgroup

- Allows users to make a group of process via cgroup file system interface
- Users can control characteristics of the group by read/write cgroup file system.



Example: task attach

```
[root@rx100-1 kamezawa]# mkdir /cgroup/memory/TestCgroup
[root@rx100-1 kamezawa]# ./very-long-task.sh &
[1] 22862
[root@rx100-1 kamezawa]# echo 22862 > /cgroup/memory/TestCgroup/tasks
[root@rx100-1 kamezawa]# cat /proc/22862/cgroup
8:blkio:/
7:net_cls:/
6:freezer:/
5:devices:/
4:memory:/TestCgroup
3:cpuacct:/
2:cpu:/
1:cpuset:/
[root@rx100-1 kamezawa]# ps -p 22862 -o pid,comm,cgroup
  PID COMMAND         CGROUP
22862 very-long-task. blkio:/;net_cls:/;freezer:/;devices:/;memory:/TestCgroup;cpuacct:/;cpu:/;cpuset:/
[root@rx100-1 kamezawa]# kill %1
[root@rx100-1 kamezawa]# rmdir /cgroup/memory/TestCgroup
[1]+  Terminated              ./very-long-task.sh
[root@rx100-1 kamezawa]#
```

Check libcgroup
For friendly UI

Memory/swap limiting.

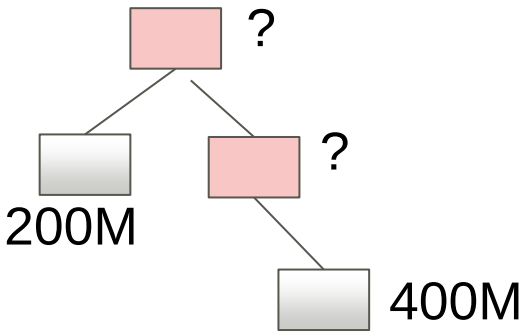
- limit usage of Anon, File Cache
- limit of memory+swap usage.

```
[root@rx100-1 kamezawa]# mkdir /cgroup/memory/TestCgroup
[root@rx100-1 kamezawa]# echo 300M > /cgroup/memory/TestCgroup/me
mory.limit_in_bytes
[root@rx100-1 kamezawa]# echo 300M > /cgroup/memory/TestCgroup/me
mory.memsw.limit_in_bytes
[root@rx100-1 kamezawa]# ls -l rhel-server-6.2-x86_64-dvd.iso
-rw-r--r-- 1 kamezawa kamezawa 3589636096 Jun  4 14:53 rhel-serve
r-6.2-x86_64-dvd.iso
[root@rx100-1 kamezawa]# echo $$ > /cgroup/memory/TestCgroup/task
s
[root@rx100-1 kamezawa]# grep Cache /proc/meminfo
Cached:            38904 kB
SwapCached:            0 kB
[root@rx100-1 kamezawa]# cat rhel-server-6.2-x86_64-dvd.iso > /dev/
null
[root@rx100-1 kamezawa]# grep Cache /proc/meminfo
Cached:           350340 kB
SwapCached:            0 kB
```

3.6G file

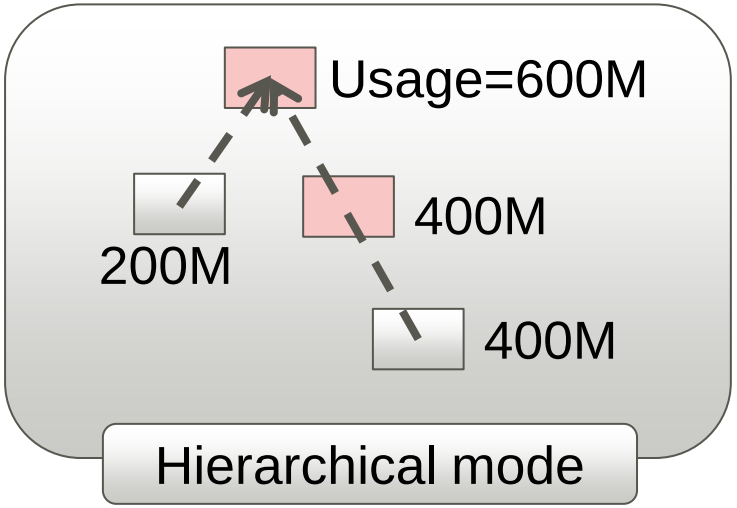
Only 300M usage of cache

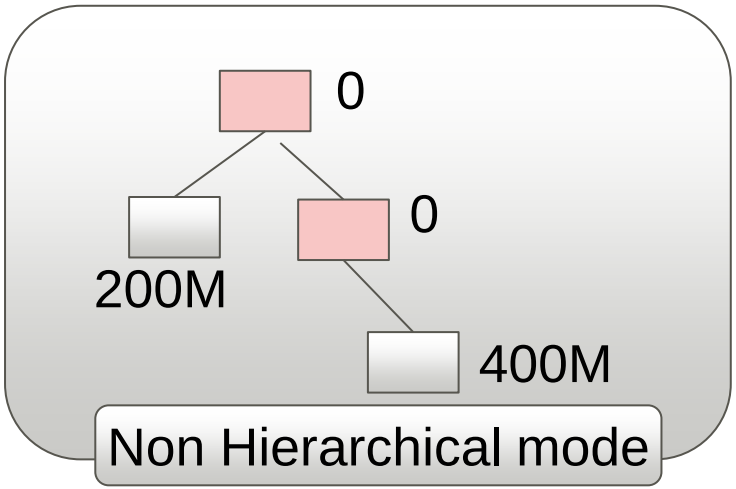
Hierarchical/non-Hierarchical



Assume memory usage in leaf cgroups are 200M and 400M.
What the parent's should be ?

choice






```
#cd /cgroup/memory/TestCgroup/
#echo 1 > memory.use_hierarchy
```

- User can choice accounting policy of tree

Example: hierarchical mode.

```
#pwd
/cgroup/memory/TestCgroup
#echo 1 > memory.use_hierarchy
#echo 300M > memory.limit_in_bytes
#cat ./memory.usage_in_bytes
1093632
#mkdir Child
#cat ./Child/memory.usage_in_bytes
0
#echo $$ > ./Child/tasks
#cat /home/kamezawa/rhel-server-6.2-x86_64-dvd.iso > /dev/null
#cat ./Child/memory.usage_in_bytes
314458112
#cat ./memory.usage_in_bytes
314494976
#grep Cache /proc/meminfo
Cached:          350508 kB
```

Set hierarchical mode.

Memory usage is propagated.

Example: non-hierarhical mode

```
#pwd
/cgroup/memory/TestCgroup
#echo 0 > memory.use_hierarchy
#mkdir Child
#echo 300M > memory.limit_in_bytes
#echo 200M > Child/memory.limit_in_bytes
#echo $$ > ./Child/tasks
#cat /home/kamezawa/rhel-server-6.2-x86_64-dvd.iso > /dev/null
#cat ./Child/memory.usage_in_bytes
209580032
#cat ./memory.usage_in_bytes
716800
#grep Cache /proc/meminfo
Cached:          251404 kB
```

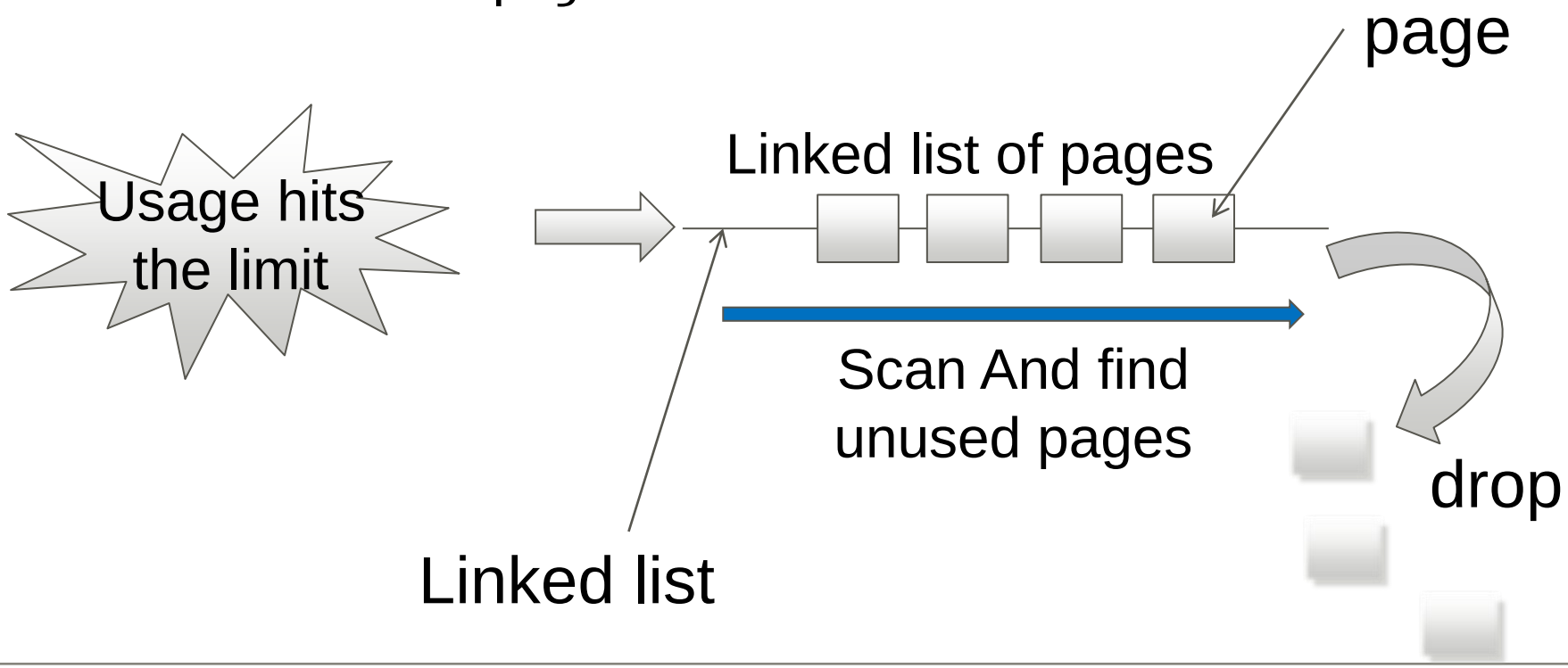
Unset hierarchical mode

No propagation to the parent

Under non-hierarchila mode, parent/child cgroup are independent from each other.

LRU

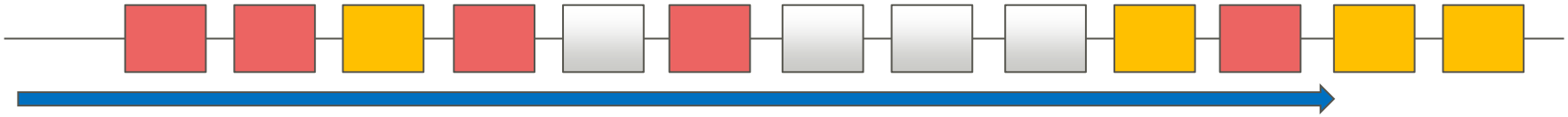
- Memory cgroup reclaims memory when the usage hit limits.
- All pages are tracked by linked list, LRU.
- At reclaiming memory, memory cgroup scans s its own LRU list and select victim pages.



LRU implementation(1)

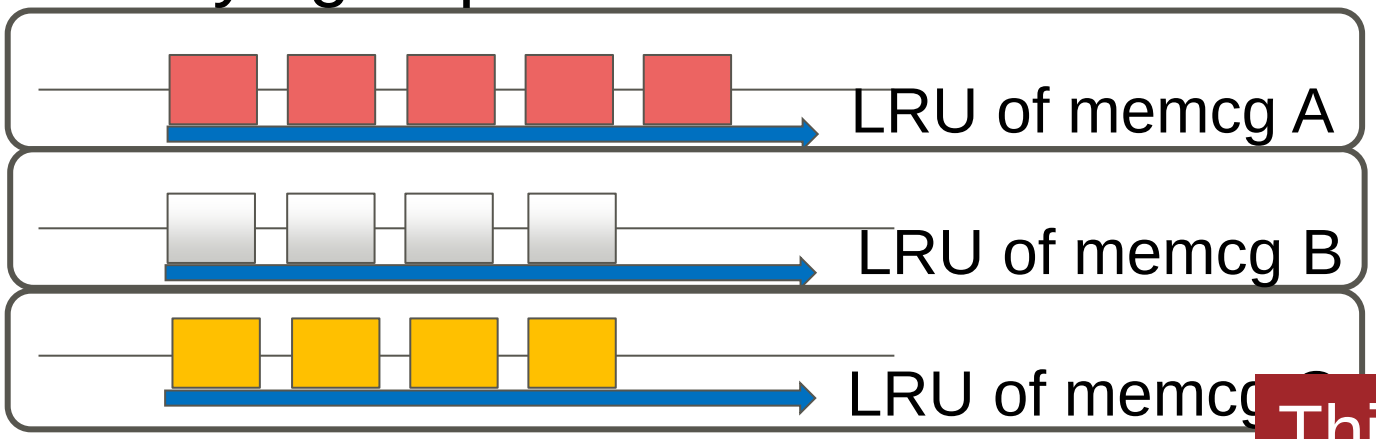
FUJITSU

- Before 3.3.....LRU was Duplicated.



Global per-zone-page-LRU-list scans all pages.

Memory cgroup scans its own LRU.

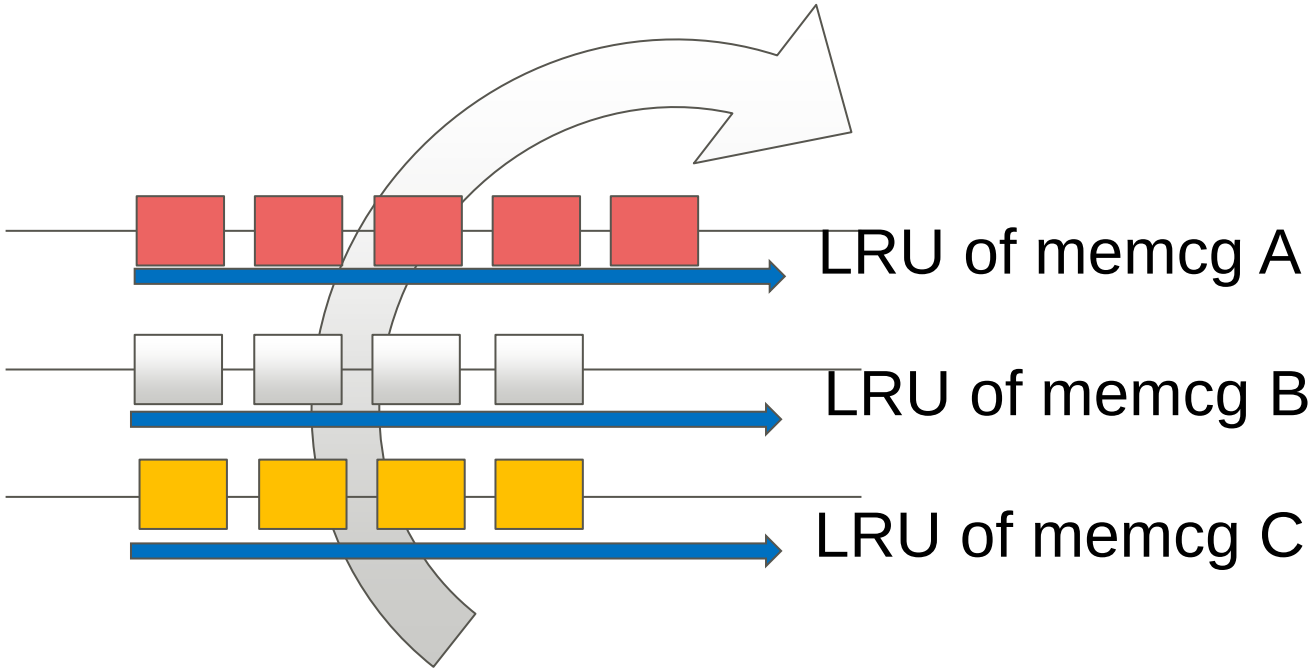


This consumes 16bytes/page

LRU implemenations(2)

FUJITSU

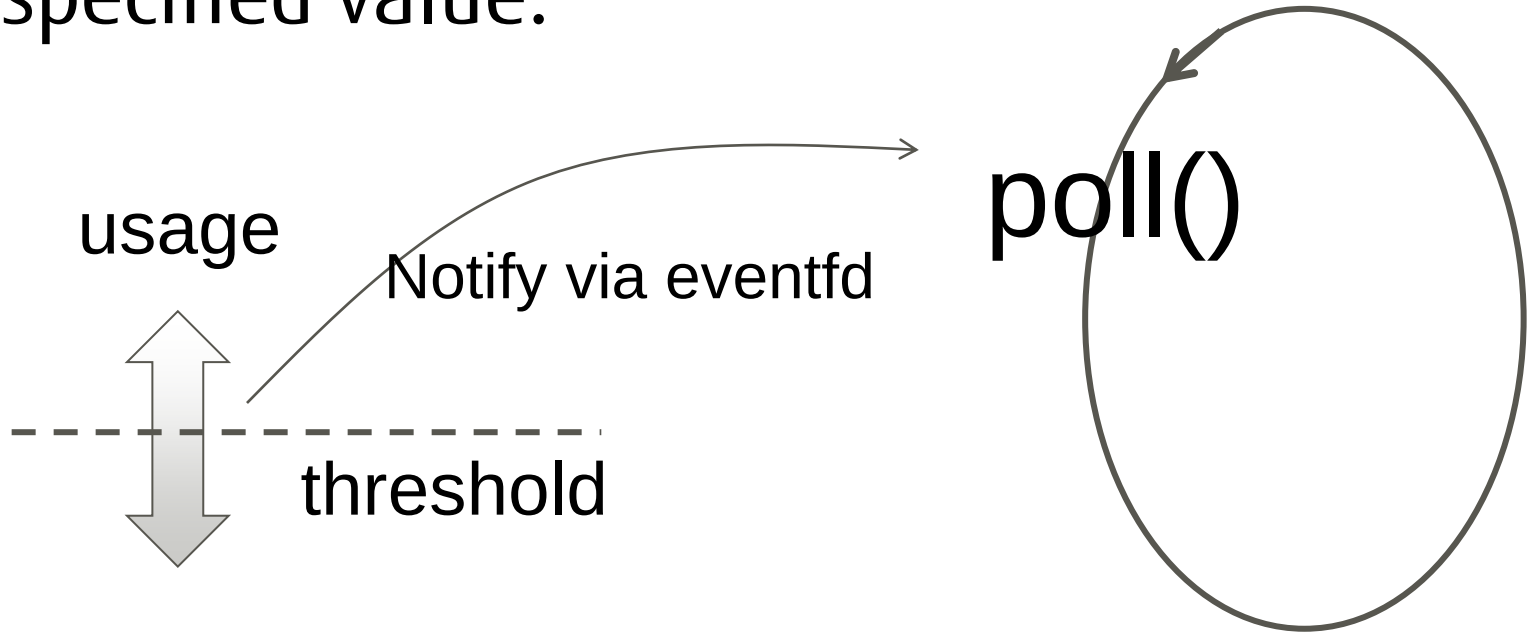
- Since 3.3, LRUs are unified.



Global LRU is re-implemented as a group of all per-memcg-per-zone-LRU linked list.
This saves 16bytes/page.

Threshold notification

- Notify via eventfd when the usage crosses the specified value.



Check: Documentation/cgroup/cgroup_event_listener.c

Example: threshold

```
#gcc -o cgroup_event_listener cgroup_event_listener.c
#mkdir /cgroup/memory/TestCgroup
#./cgroup_event_listener /cgroup/memory/TestCgroup/memory.usage_in_bytes 300M
/cgroup/memory/TestCgroup 300M: crossed
```

Wait for usage reaching 300M bytes on TestCgroup

OOM block/notifier

FUJITSU

- Memory cgroup can
  - Block OOM-Kill under a memcg
  - Notification of OOM via eventfd
- If OOM-Killer is blocked
  - All tasks under the cgroup will stop.
  - Tasks run again if
    - Some resources are freed.
    - Get signal
    - Limit is raised.
    - A task is moved to other cgroup

Example: OOM

FUJITSU

Set limt and wait for OOM

```
#/sbin/swapoff -a
#cat /cgroup/memory/TestCgroup/memory.oom_control
oom_kill_disable 0
under_oom 0
#echo 1 > /cgroup/memory/TestCgroup/memory.oom_control
#echo 300M > /cgroup/memory/TestCgroup/memory.limit_in_bytes
#./cgroup_event_listener /cgroup/memory/TestCgroup/memory.oom_control dummy
/cgroup/memory/TestCgroup dummy: crossed
```

Check status and kill by hand

```
#cat /cgroup/memory/TestCgroup/tasks
3919
4472
4473
#ps --pid 3919 4472 4473
  PID TTY      STAT   TIME COMMAND
 3919 pts/1    S      0:00 bash
 4472 pts/1    S+     0:00 ./stress --vm 1 --vm-bytes 400M
 4473 pts/1    D+     0:00 ./stress --vm 1 --vm-bytes 400M
#kill 4473
#cat /cgroup/memory/TestCgroup/tasks
3919
#ps --pid 3919
  PID TTY          TIME CMD
 3919 pts/1    00:00:00 bash
```

All tasks are stopped

kill

Run again

tcp memcontrol.

- Controls memory usage for TCP (3.3)
  - If memory usage hits limit....
    - INPUT: packets will be dropped.
    - OUTPUT: wait for available memory.

```
#ls memory.kmem.tcp.*
memory.kmem.tcp.failcnt
memory.kmem.tcp.limit_in_bytes
memory.kmem.tcp.max_usage_in_bytes
memory.kmem.tcp.usage_in_bytes
```

For now, this works independent from other memory controls for anon,file,swap

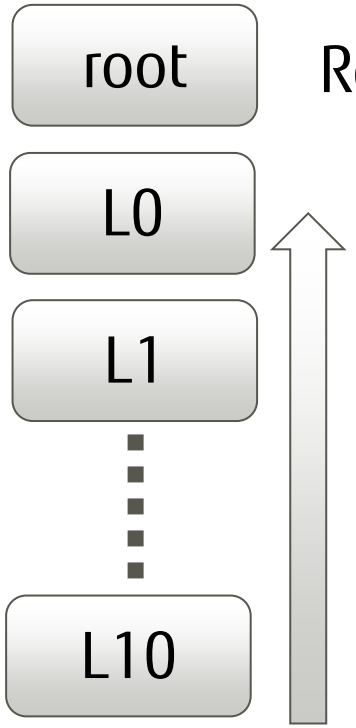
Performance

- Comparison between 2.6.32 and 3.4
- Transparent Hugepage is disabled.
- Check overheads of memory cgroup.
  - create a tree /cgroup/memory/L0/L1/L2/L3/L4/L5/L6/L7/L8/L9/L10 with use_hierarchy=1
  - Run mini-benchmark on root,L1,L2,L10.

Overhead of hierarchy

FUJITSU

- If use_hierarchy=1, need to update several counters at once.

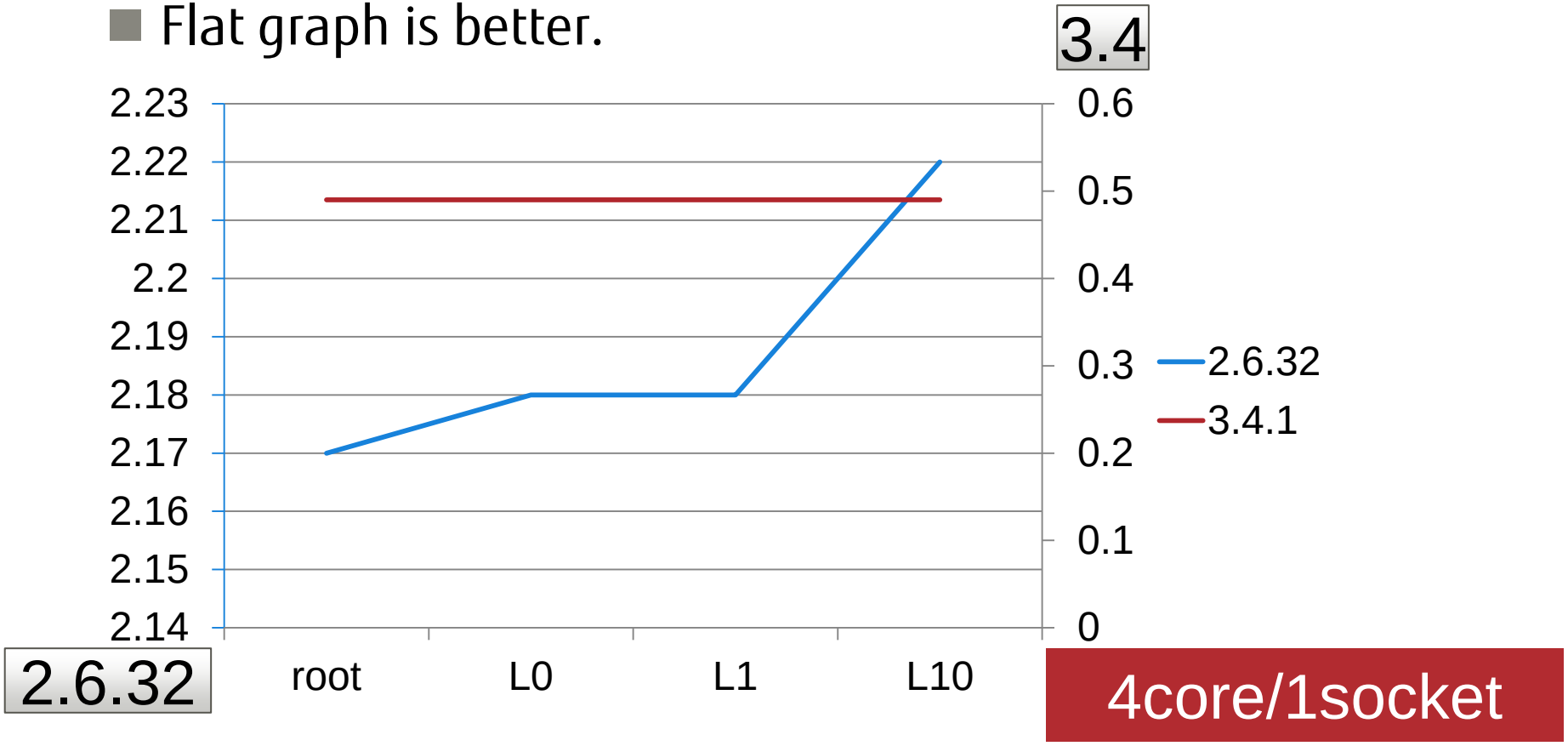


Root cgroup is out of control. No counters

In this test, All usage in L1...L10 are propagated up to L0. Then, the number of counters and Overheads will be big in deep groups.

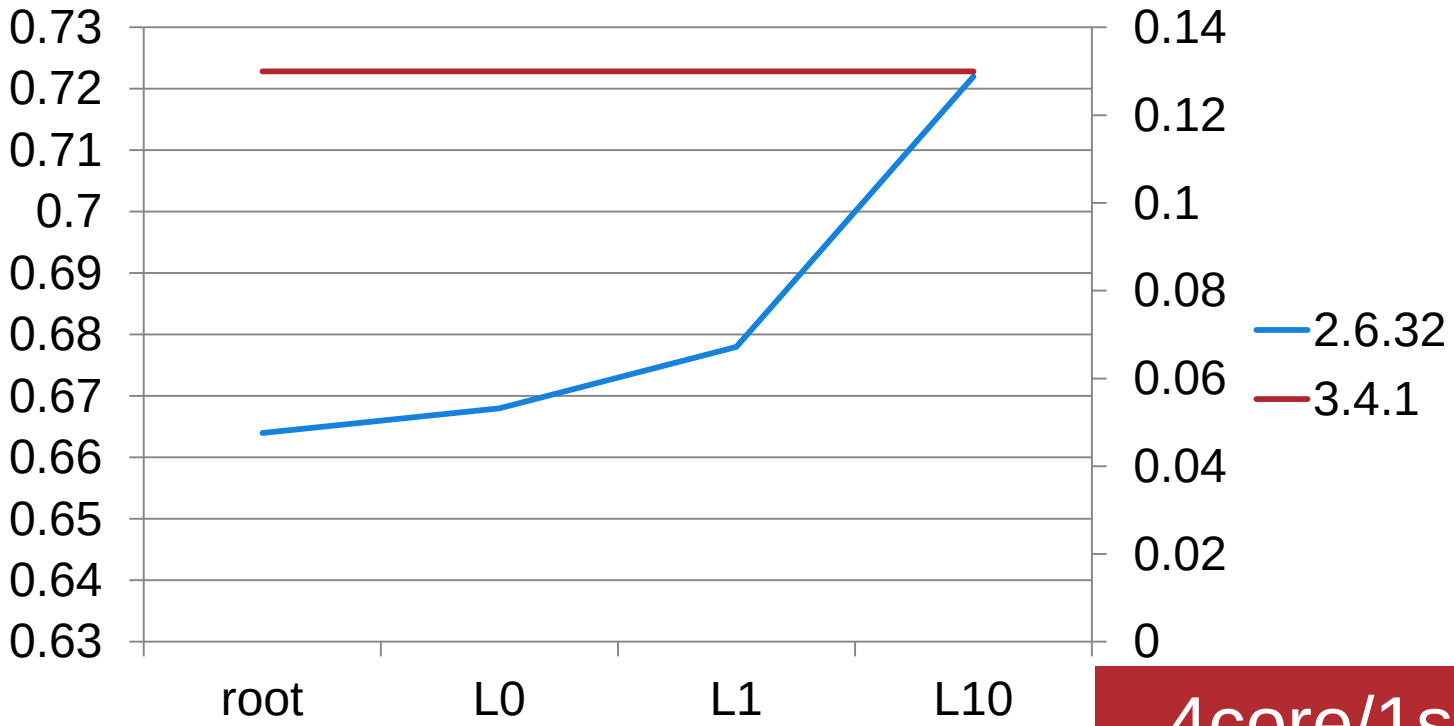
tar -xpf

- tar –xpf linux-3.5.tar onto tmpfs.
  - Checking file cache creation overheads in 'sys' time
  - Flat graph is better.



rm -rf

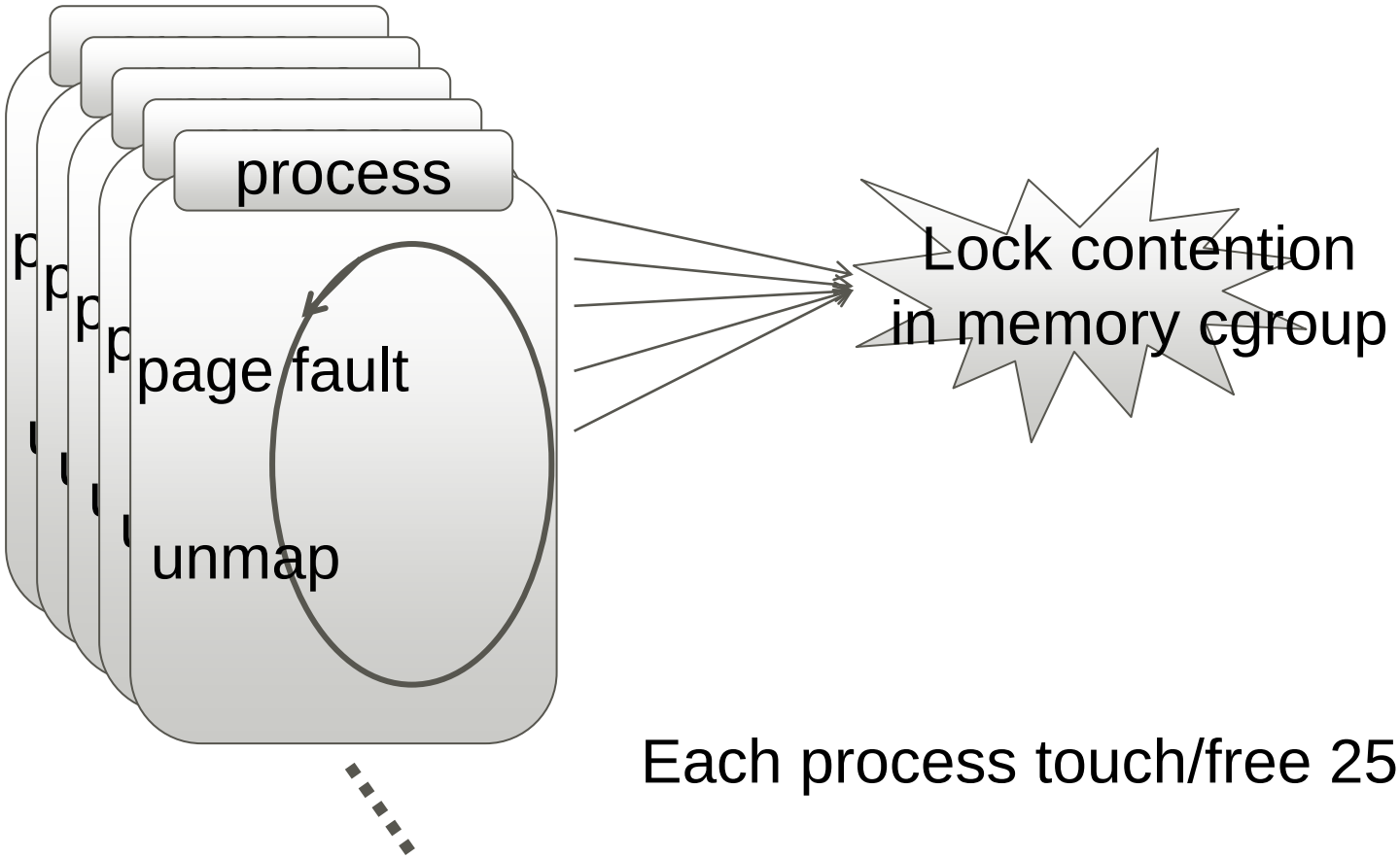
- # rm –rf linux-3.5 on tmpfs
- Checking file cache deletion overheads in 'sys'.
  - Flat graph is better.



4core/1socket

Parallel page faults

- Causing page fault in parallel.

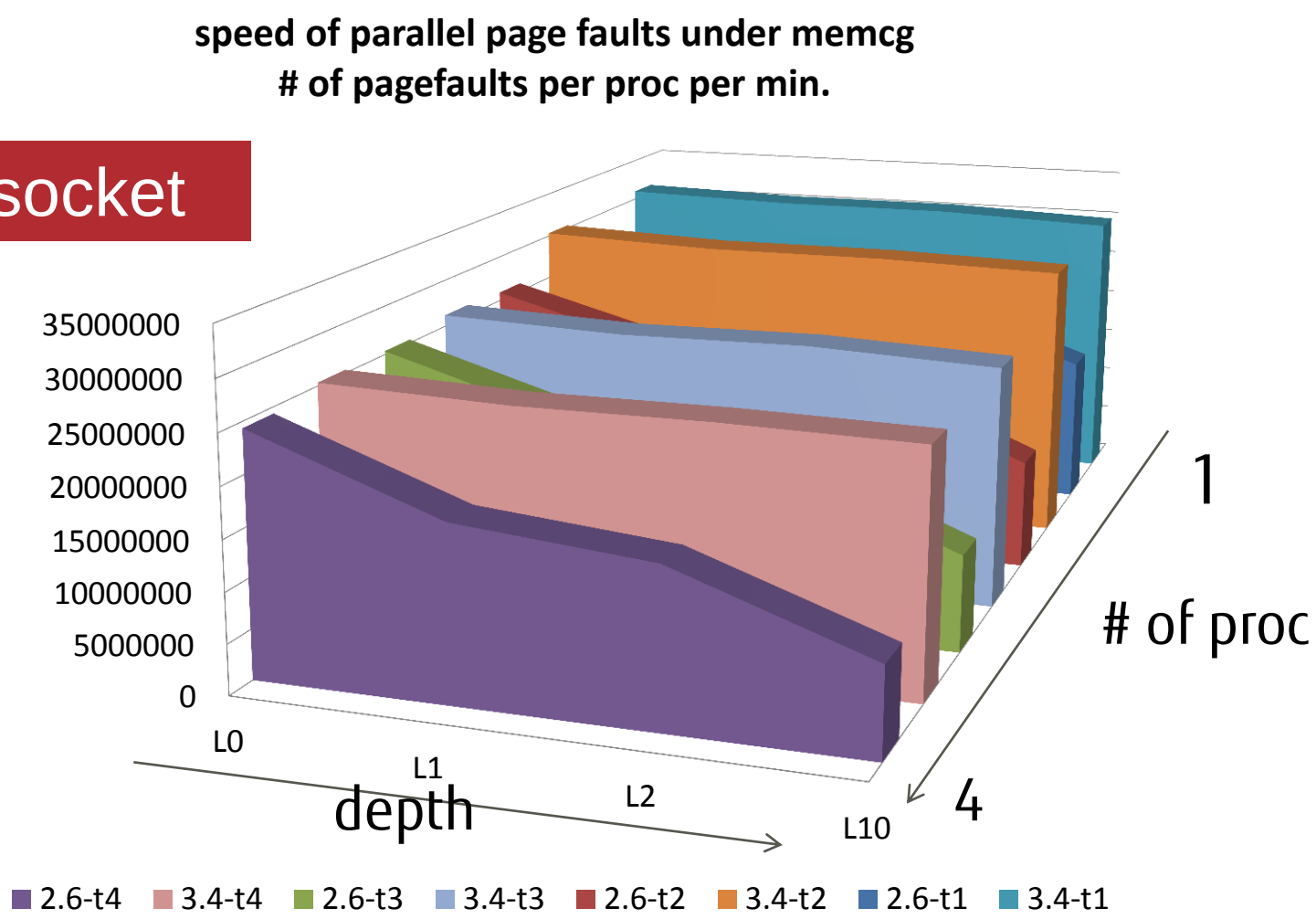


Each process touch/free 256k mem.

Parallel page faults(1)

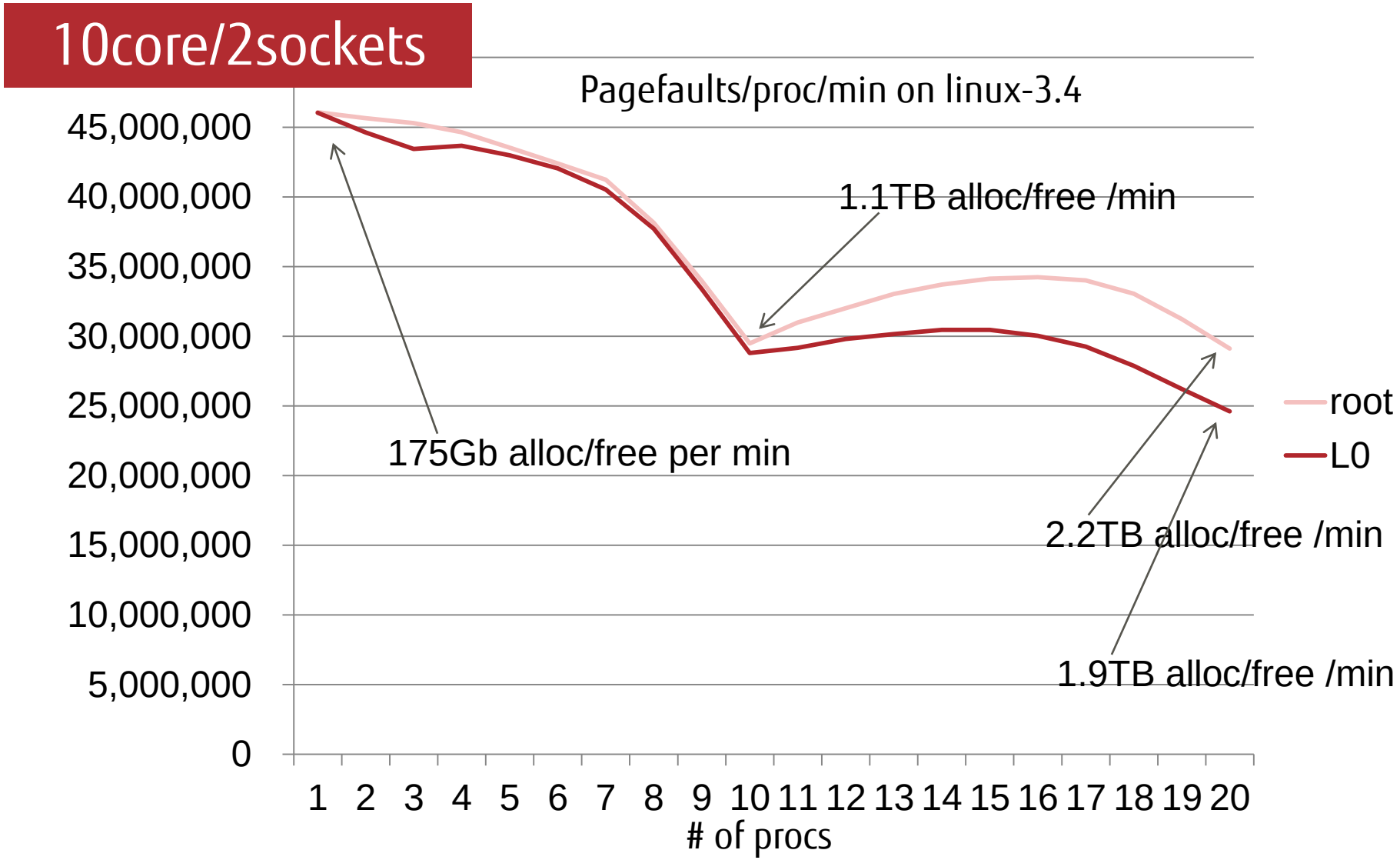
FUJITSU

4core/1socket



Parallel page faults (2)

FUJITSU



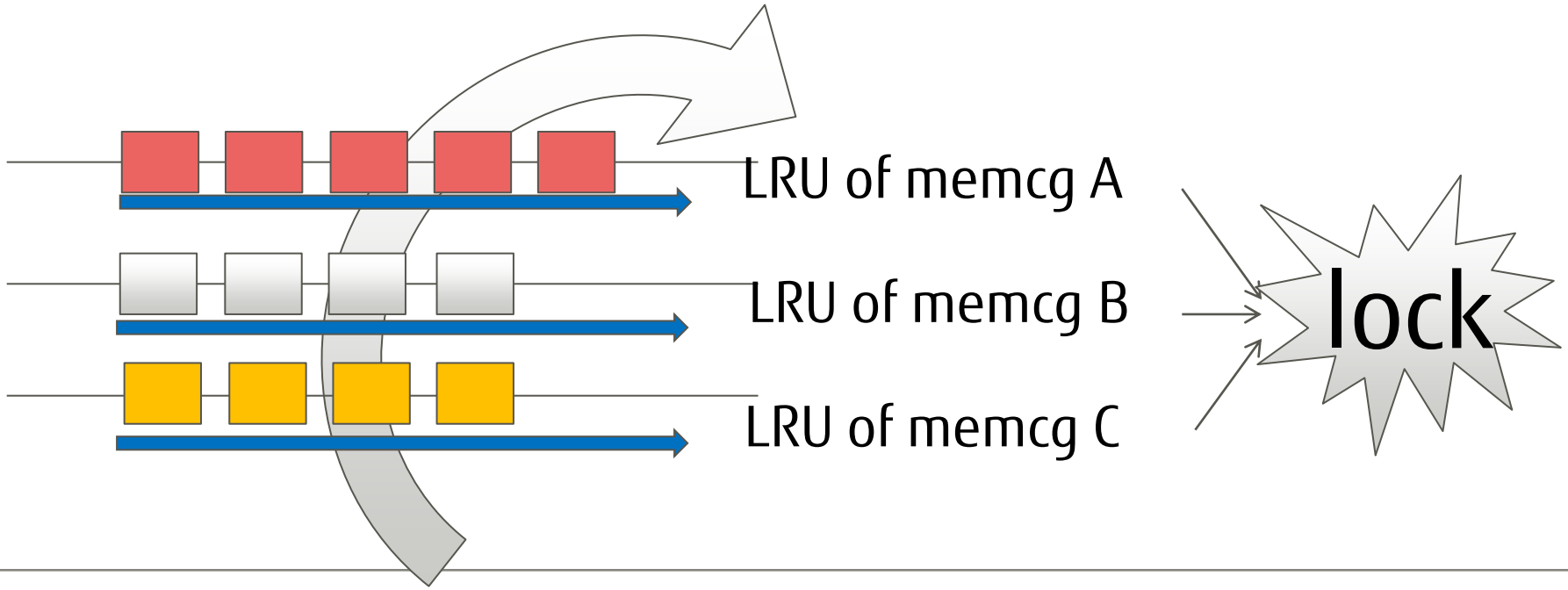
News & TODO

FUJITSU

- Many things are still in TODO list...
  - Split locks.
  - Hugetlb controls
  - Kmem(slab) controls
  - Softlimit renewal
    - Idle memcg.
  - Dirty Throttling
  - Reduce memory overhead(16bytes)
  - Per-memcg kswapd.

Split locks

- Now, per-zone-memcg-lru list is maintained.
- But, lock is now shared among memcgs.
  - Implementation is very complicated.
  - Hugh and Konstantin working on this.



Hugetlb Controls

FUJITSU

- Controls for hugetlbfs
  - Implements per cgroup hugetlbfs quota.
  - The development was almost finished but the author started re-designing because of rjections by other commiters.
  - Maybe hugetlbfs cgroup will be added rather than enhancing memcg.
  - Aneesh is working on this.

Kmem(slab) controls

FUJITSU

- Function to account/limit kernel memory
  - Under development for a half year...
  - Supporting slab/slub
  - Discuss: per-page accounting v.s. per-objs
  - The biggest concern is performance.
    - Isolation & Performance.....challenge.
  - Costa & Suleiman works on this.
    - Seems co-operative with Christoph Lameter

Soft limit renewal

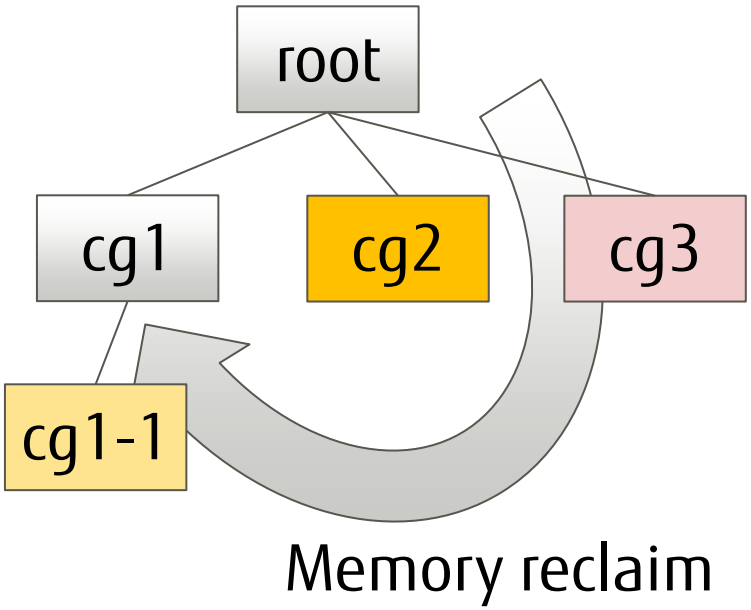
FUJITSU

- Now, memory cgroup has softlimt
  - for hinting the system memory reclaim priority.
- Some people complaints
  - isolation using softlimit doesn't work well.
  - it doesn't work as expected..
- Total re-implementation is planned.
  - Ying Han, Johannes, Michal working on this.

Idle memcg

FUJITSU

- In softlimit discussion..



- If soft limit is set, kswapd will choose victim cgroup by it.
- What happens when a cgroup has been idle for a long time but it doesn't hit softlimit ?

Dirty Throttoling

FUJITSU

- In todo list since 3 years ago...
- Without this :
  - background write-out doesn't start enough quick.
  - Memory recalim will see too much dirty pages.
- Patches were posted several times.
- New implementation of I/O less dirty throttoling
  - Need updates.

Reduce memory overhead.

FUJITSU

- Now, using 16bytes/page (onx86-64)
  - 4Mbytes/ 1Gbytes.

```
 */
struct page_cgroup {
        unsigned long flags;
        struct mem_cgroup *mem_cgroup;
};
```

- Seems it's possible to make this 8bytes/page.
  - merge this into 'unused' 8bytes in 'struct page'...

Per-memcg kswapd

FUJITSU

- Now, memory cgroup has no kswapd.
- Run kthread for reduce memory usage when the usage is near to the limit.
  - Move cpu usage for memory reclaiming from applications to kthread.
  - By this, memory will be reclaimed in 'idle' time and applications will get better latency.
    - Should be able to control cpu prio of kthread for kswapd?
- Old patch shown good result.
  - Waiting for lock-splitting patches for avoid contention.

FUJITSU

shaping tomorrow with you

Parallel Page fault /touch 2M+THP(3)

FUJITSU

- Increase bufsize as 2M and enable THP.