cgroup v2 migration at Google

Shakeel Butt (shakeelb@google.com)
Kamil Yurtsever (kyurtsever@google.com)

Outline

cgroups at Google: what is our history of cgroup usage and why want to migrate to cgroup v2.

Migration: how do we want to approach it?

Interesting use cases: Google use cases which rely on cgroup v1 specific properties.

What do we want to tell you?

- We don’t have specific requests for the kernel community.
- We want to share our approach to the migration and to learn what others think about it.
- We hope that some of the described use cases will be a good starting point for discussion.

cgroups at Google

Past and present

- Used since inception to ensure performance isolation between almost all workloads at Google.
- Used both by system daemons but also delegated to users.

Why cgroup v2?

- Uncertain future of cgroup v1 - ability to upstream new cgroup v1 features or bug fixes is unclear.
- New features are only being implemented in v2.

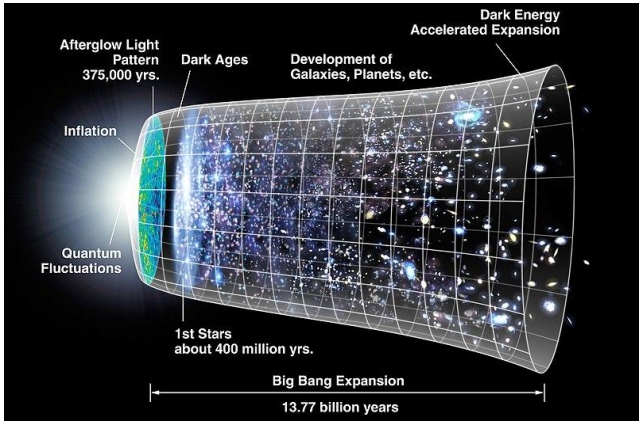
Migration

Challenges

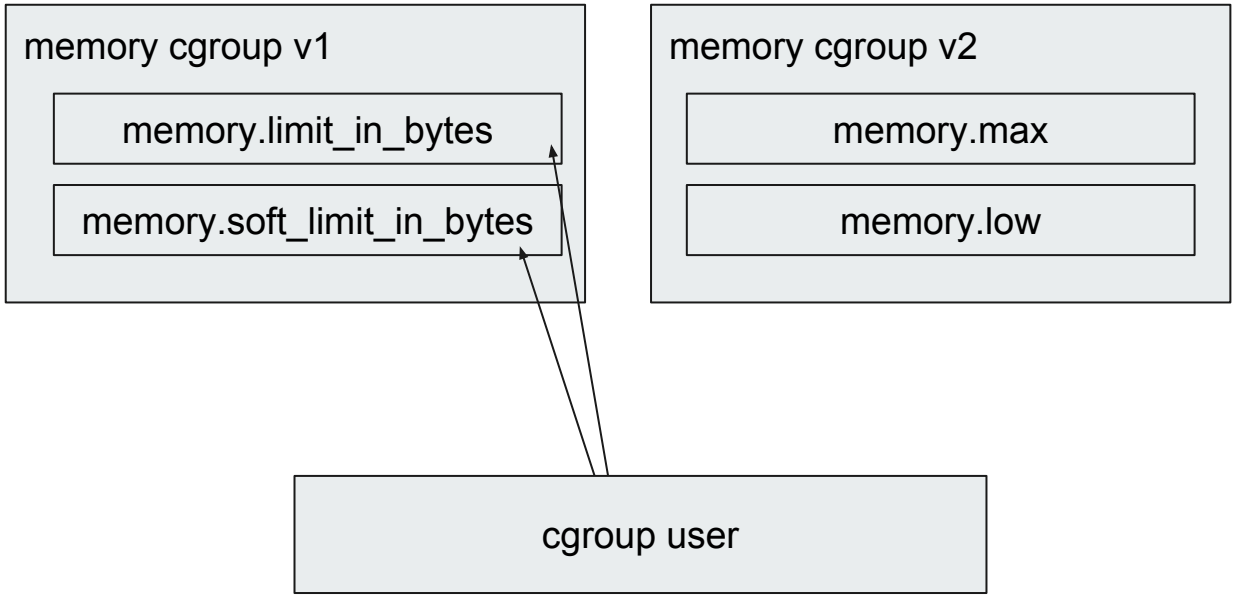
- cgroup v2 restrictions break some of our users.
- Many cgroup v1 features are missing in cgroup v2.
- No way to apply cgroup v2 restrictions incrementally.

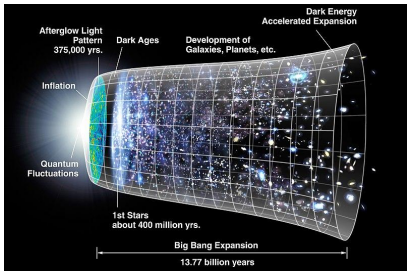
“Big bang” approach: migrating per controller

- A lot of userspace code has to be activated at the same time - “big bang” rollout instead of small steps.
- No way to detect non-conforming users beforehand.

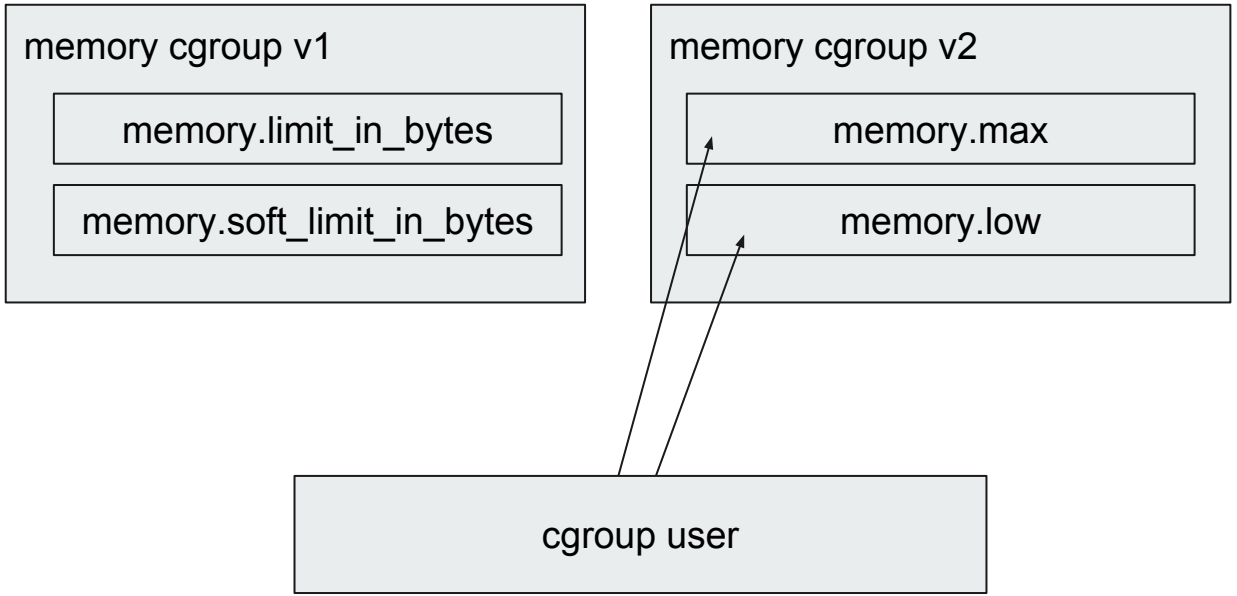


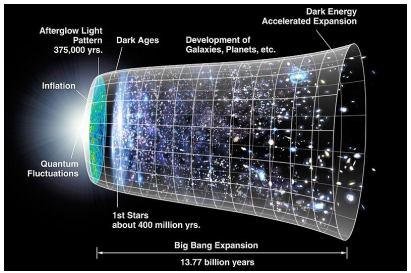
“Big bang” approach: example





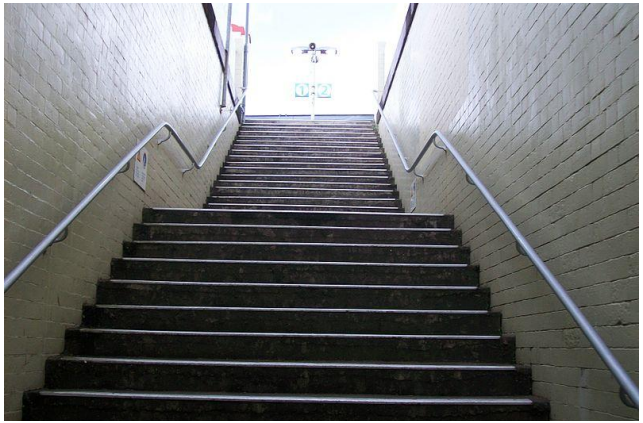
“Big bang” approach: example



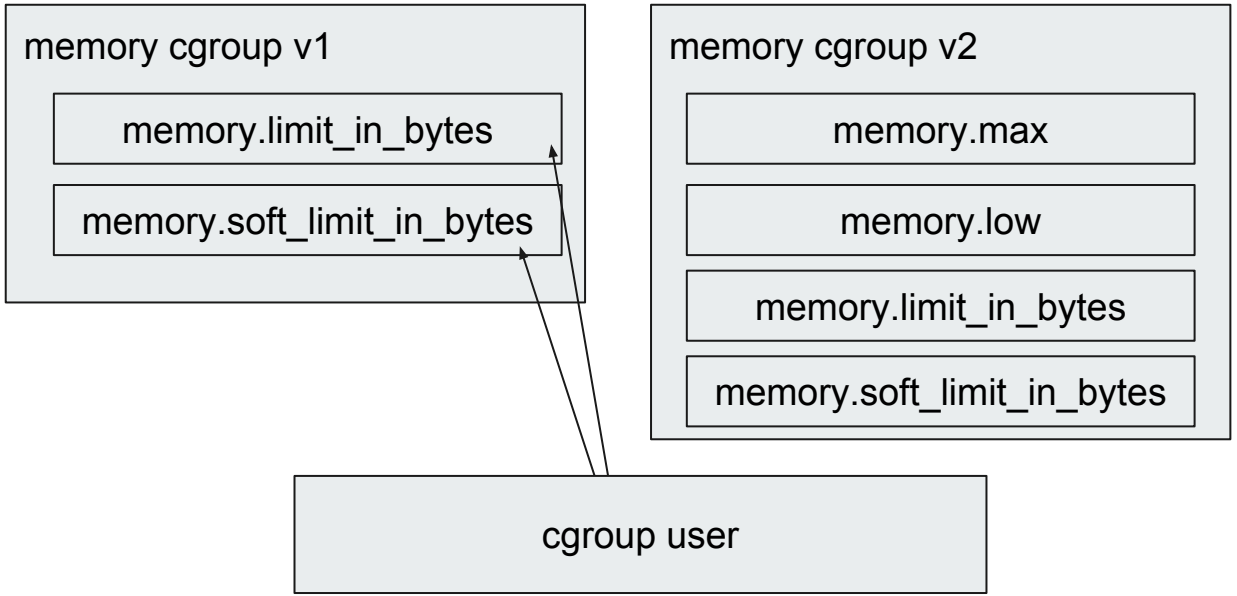


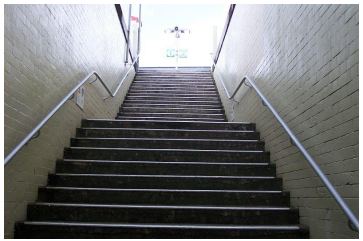
Incremental approach

- Add a way to warn on/enforce cgroup v2 constraints in cgroup v1.
- Initially v1 controller files should be exposed in cgroup v2 too.

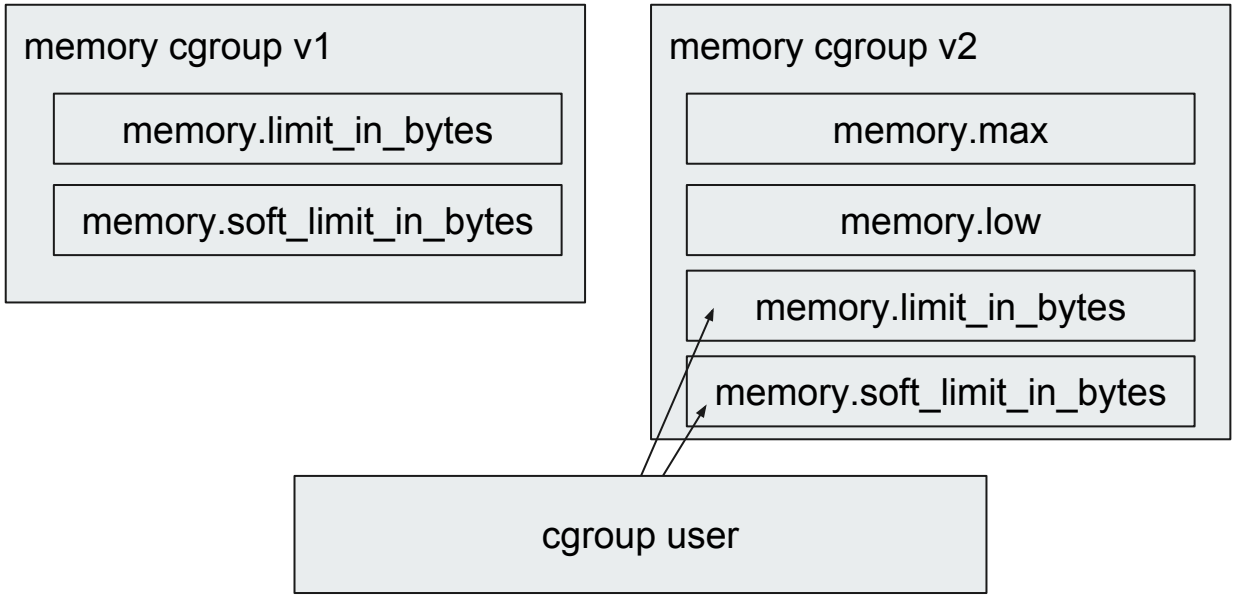


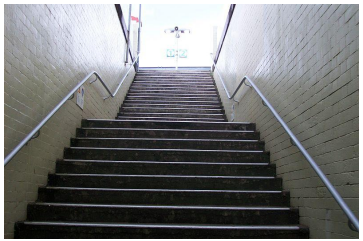
Incremental approach: controller files



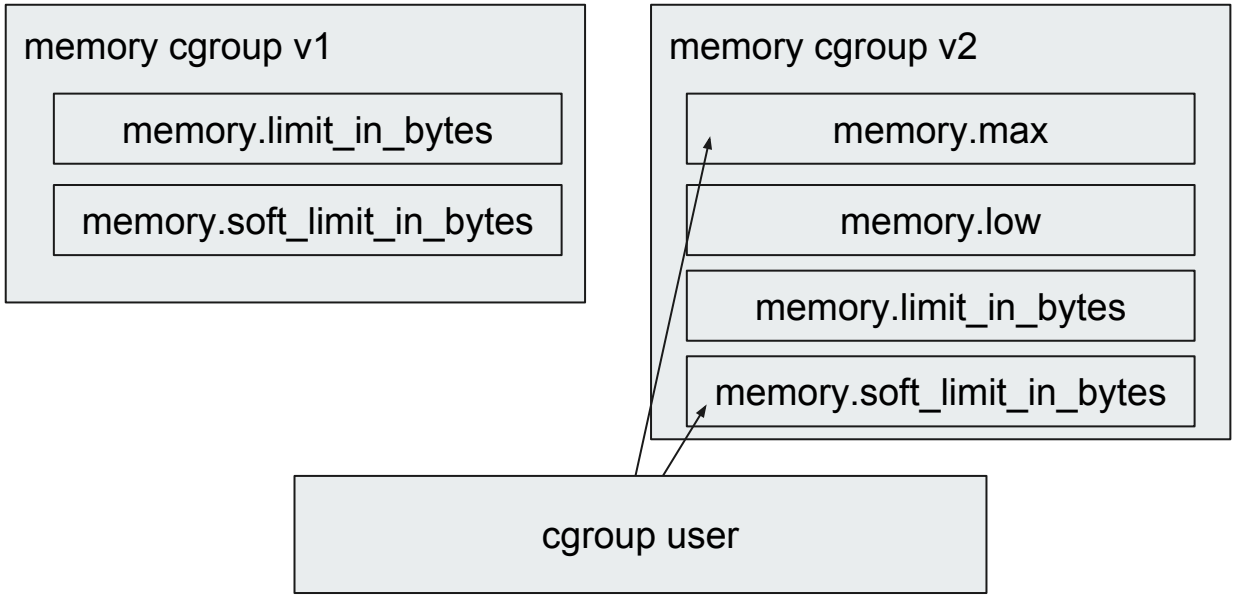


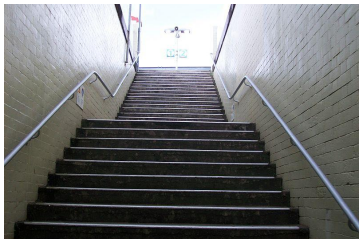
Incremental approach: controller files



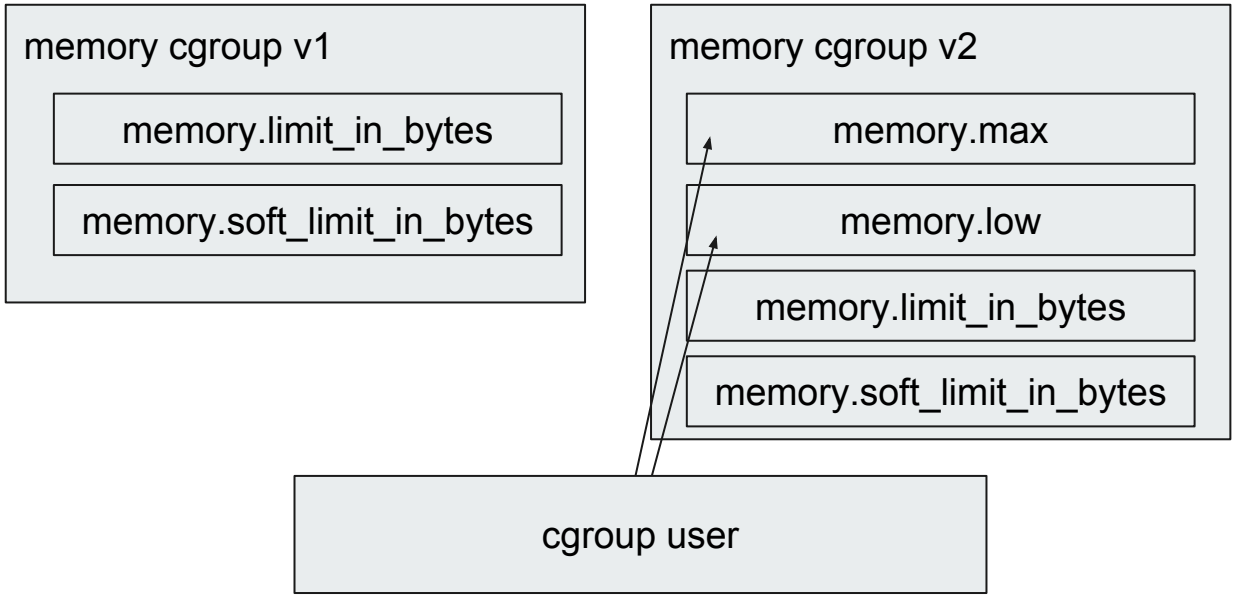


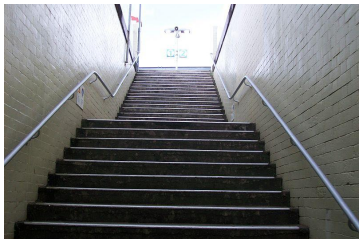
Incremental approach: controller files



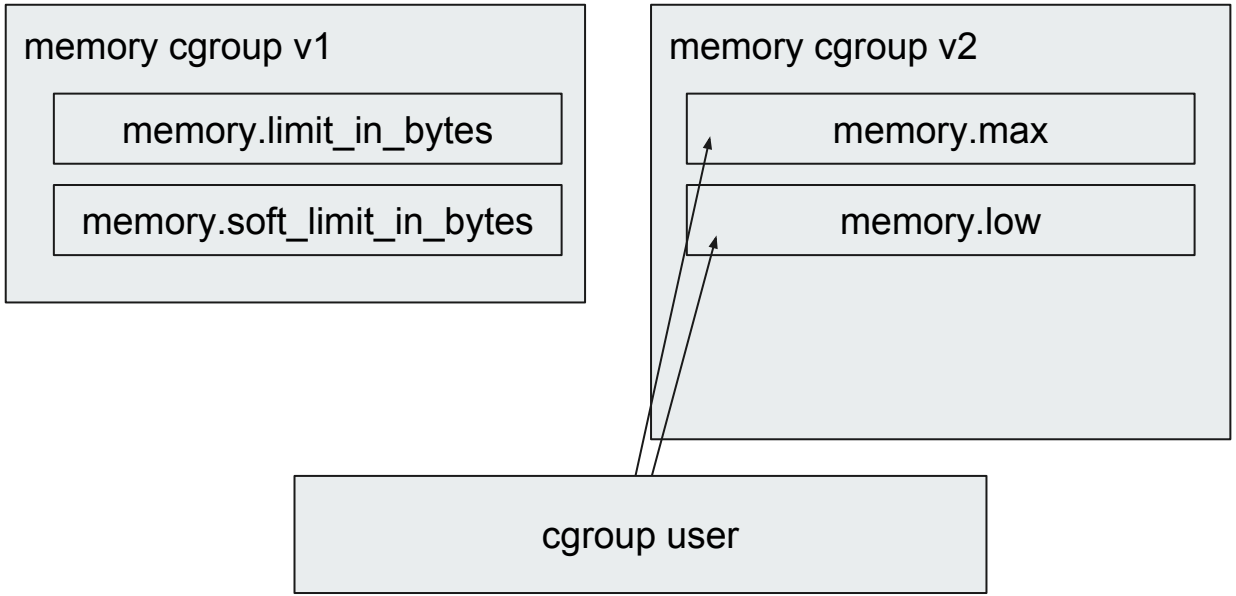


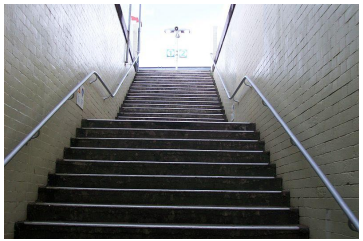
Incremental approach: controller files





Incremental approach: controller files





Detecting non-conforming users

- Requires kernel support.
- Ultimately no way to tell that the newly activated code will work correctly if we can’t verify it on cgroup v1.

Enforcing cgroup v2 constraints

- Warning and enforcement mode separately for non uniform hierarchy, thread splitting and processes in non-leaf cgroups.
- Expose cgroup.subtree_control and cgroup.type on cgroup v1.

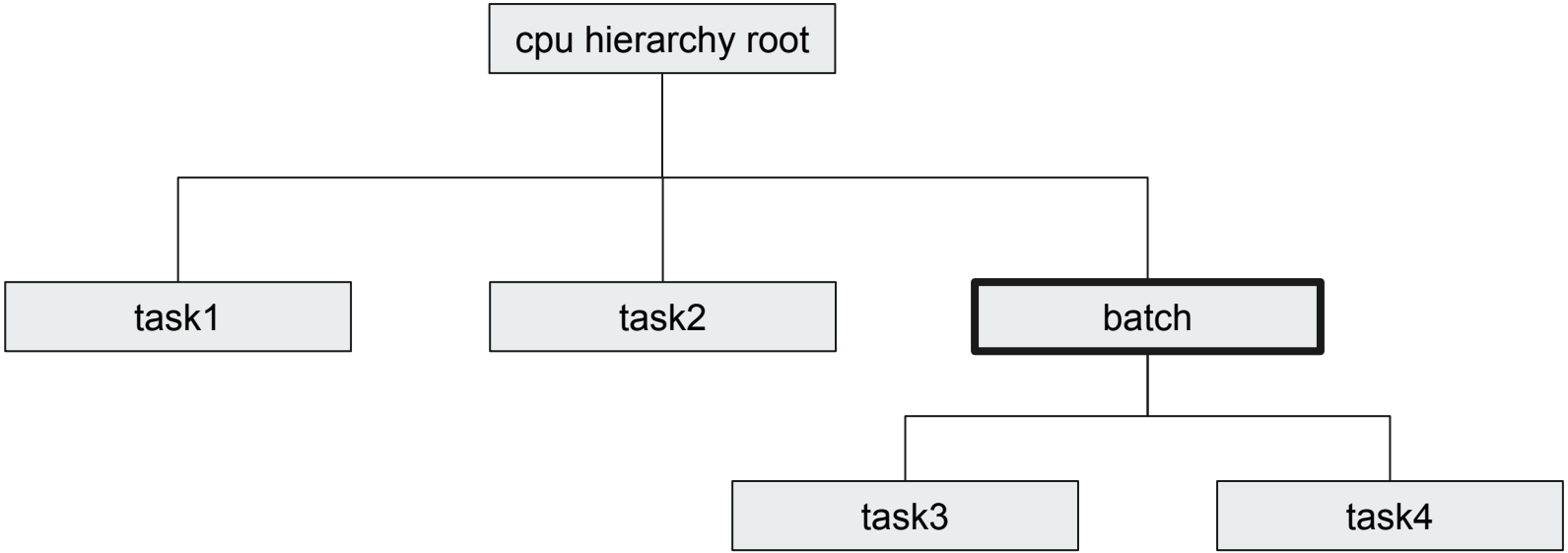
Interesting use cases

Isolating latency tolerant

- Some workloads are very latency sensitive (e.g. serving a user facing query).
- Other workloads care only about overall throughput (batch processing, long running computations).
- Latency is harder to guarantee than throughput.

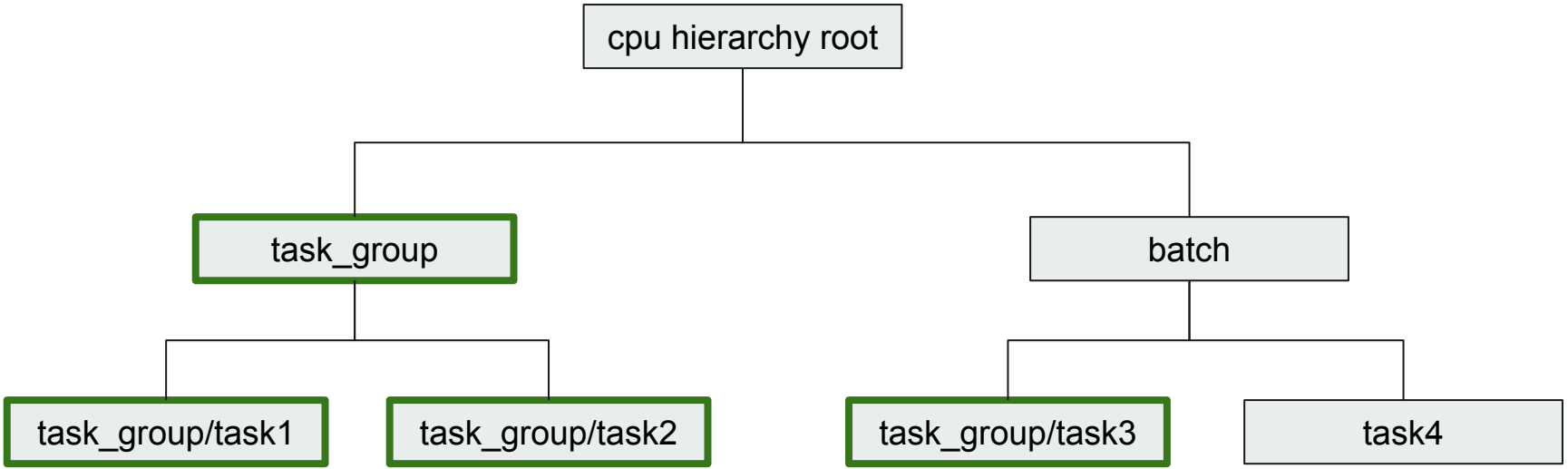
Isolating latency tolerant: CPU priority bands

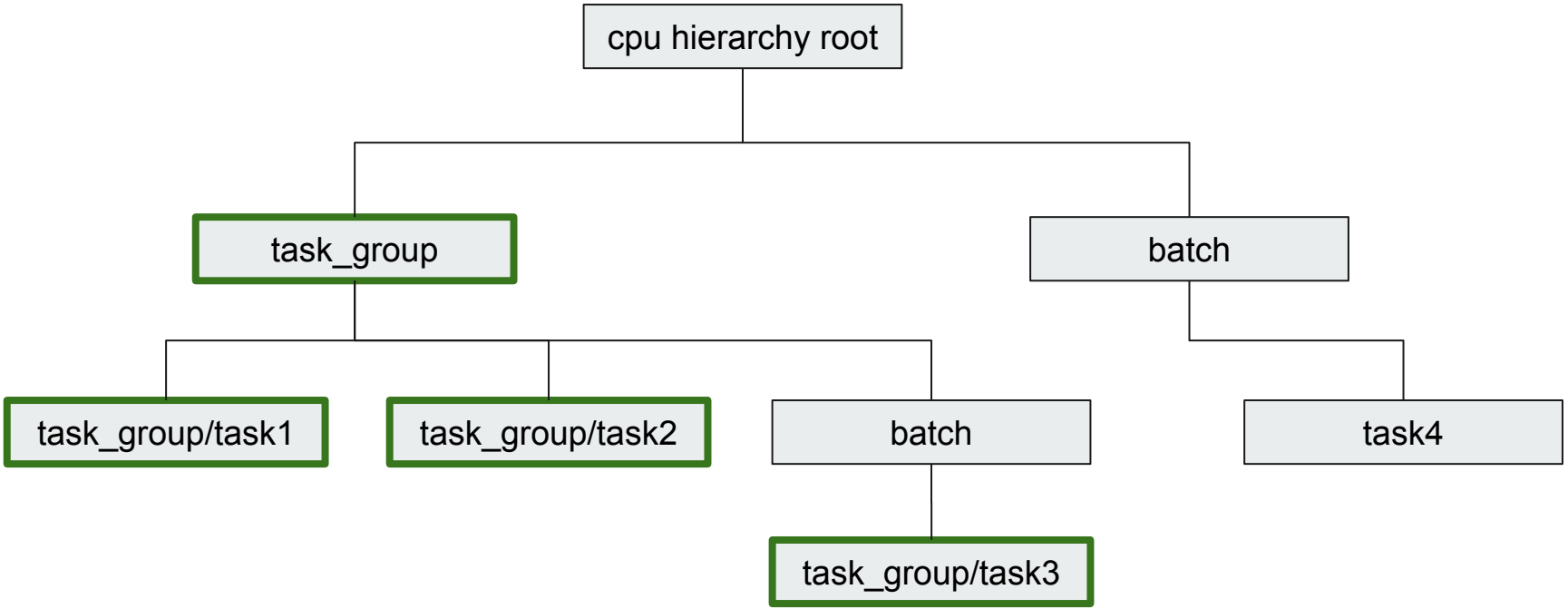
- Top level cgroup “batch” with minimal amount of cpu.shares.
- Ensures that latency tolerant jobs get CPU only if latency sensitive jobs have already been served.


cpu hierarchy root
task1
task2
batch
task3
task4

Isolating latency tolerant: nested

- Situation gets more complicated once we allow nesting.
- The “batch” property can be treated as hierarchical or not.

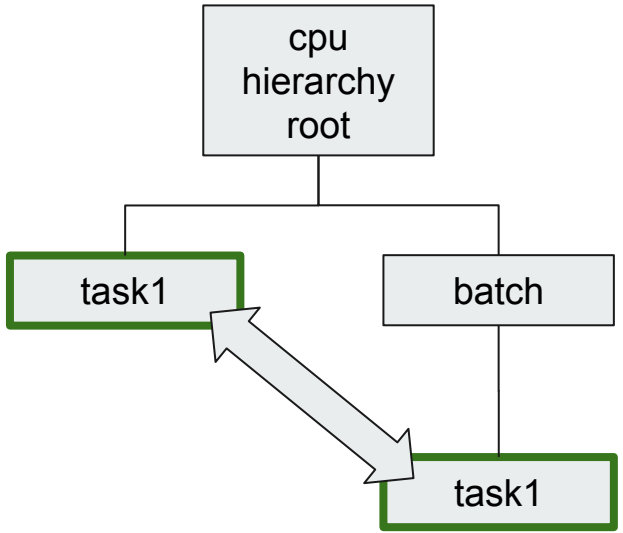

cpu hierarchy root
task_group
batch
task_group/task1
task_group/task2
task_group/task3
task4


cpu hierarchy root
task_group
batch
task_group/task1
task_group/task2
batch
task4
task_group/task3

High/low priority thread splitting

- Create high priority and low priority versions of user cgroup.
- Allow user to move between them.

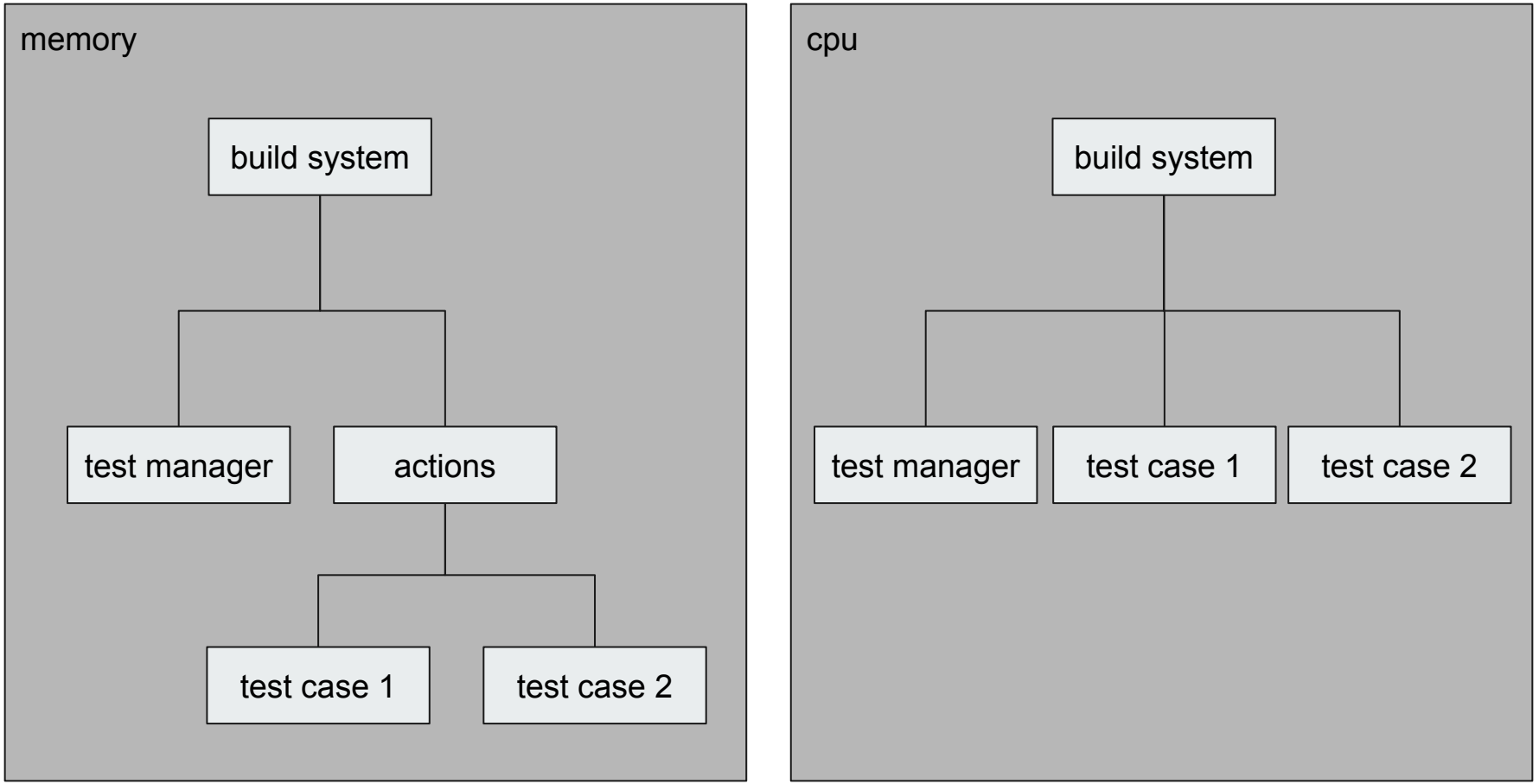
Example use case: server which handles high and low priority queries.



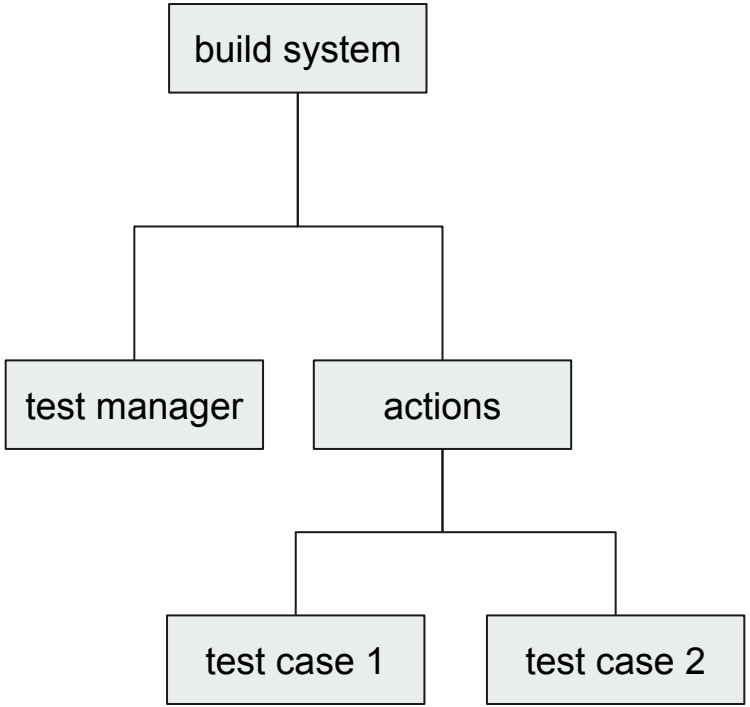
Build system

- Receives test cases to run from users.
- Shared actions memory cgroup which allows test cases to be safely overcommitted on memory.
- CPU competition on an equal level.

cgroup v1

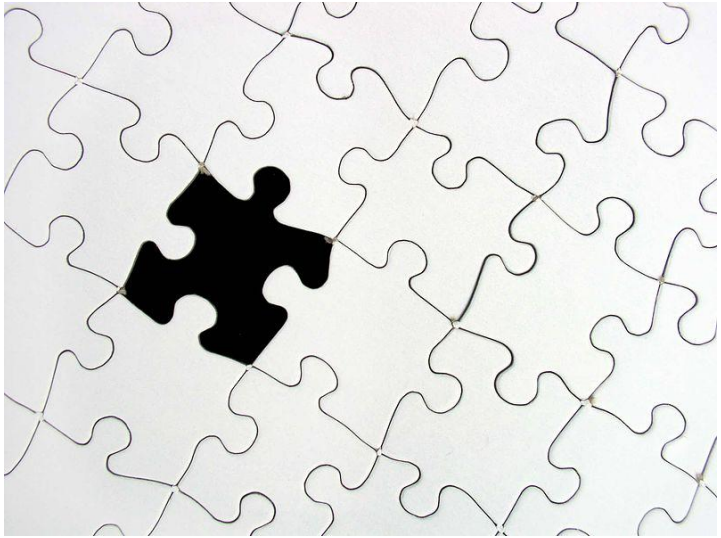

memory
build system
test manager
actions
test case 1
test case 2
cpu
build system
test manager
test case 1
test case 2

cgroup v2?



Missing v1 features: memswap

- We use swap as a tool for overcommitment.
- User limits should not be affected by how much we manage to swap out.



Thanks!

Discussion topics:

- If you are a cgroup v1 user too - how do you want to approach migration?
- Are the split hierarchy use cases something that should be addressed in cgroup v2?
- Are there v1 features that explicitly should not be migrated to cgroup v2?

Image sources

Sources for images in this presentation:

https://commons.wikimedia.org/wiki/File:Needle_and_red_thread.jpg
https://commons.wikimedia.org/wiki/File:Puzzle_black-white_missing.jpg
https://www.publicdomainpictures.net/en/view-image.php?image=80472&picture=rubber-bands-b-w
https://commons.wikimedia.org/wiki/File:Russian-Matroshka_no_bg.jpg
https://commons.wikimedia.org/wiki/File:Question_Mark.svg
https://commons.wikimedia.org/wiki/File:Climbing_through_the_Yellow_Band,_Mt._Everest,_-May_2007_a.jpg
https://commons.wikimedia.org/wiki/File:Grus_grus_-migrating_north-6a.jpg
https://commons.wikimedia.org/wiki/File:CMB_Timeline300_no_WMAP.jpg
https://commons.wikimedia.org/wiki/File:Macdonaldtown_Railway_Station_stairs_to_platform.jpg
https://commons.wikimedia.org/wiki/File:Shipping_containers_at_Clyde.jpg
https://commons.wikimedia.org/wiki/File:Metal_detector_on_the_beach.jpg
https://commons.wikimedia.org/wiki/File:Police_in_riot_gear_at_Ferguson_protests.jpg