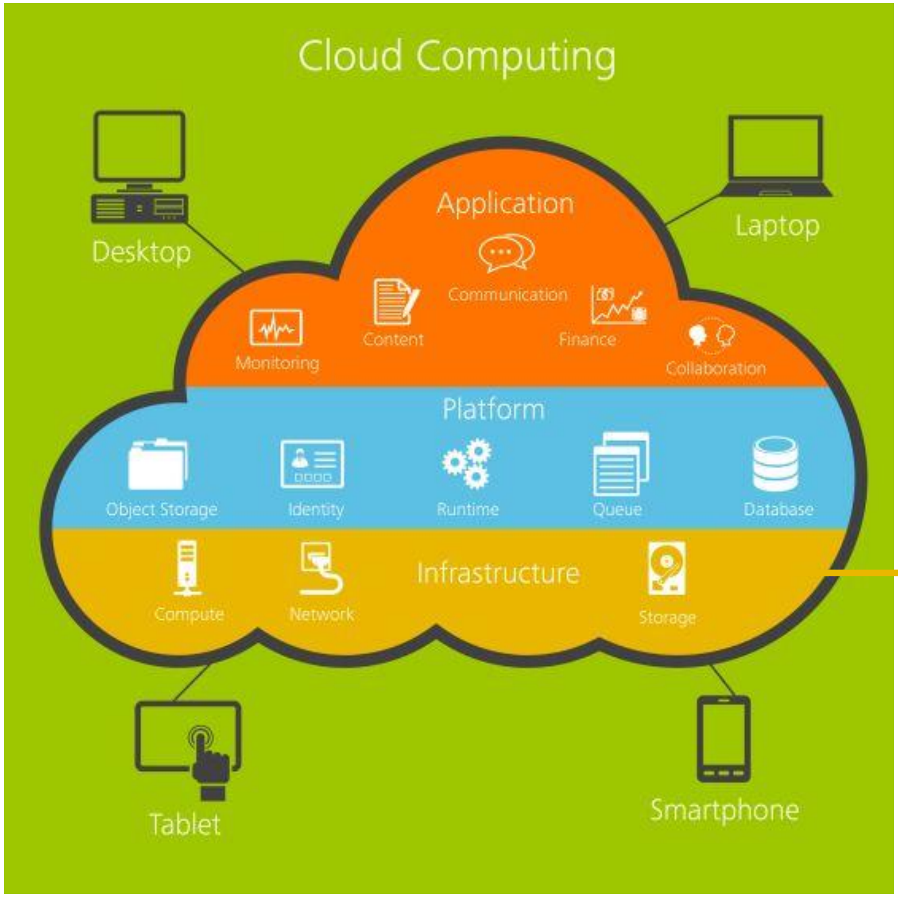
Linux cgroups

Contents

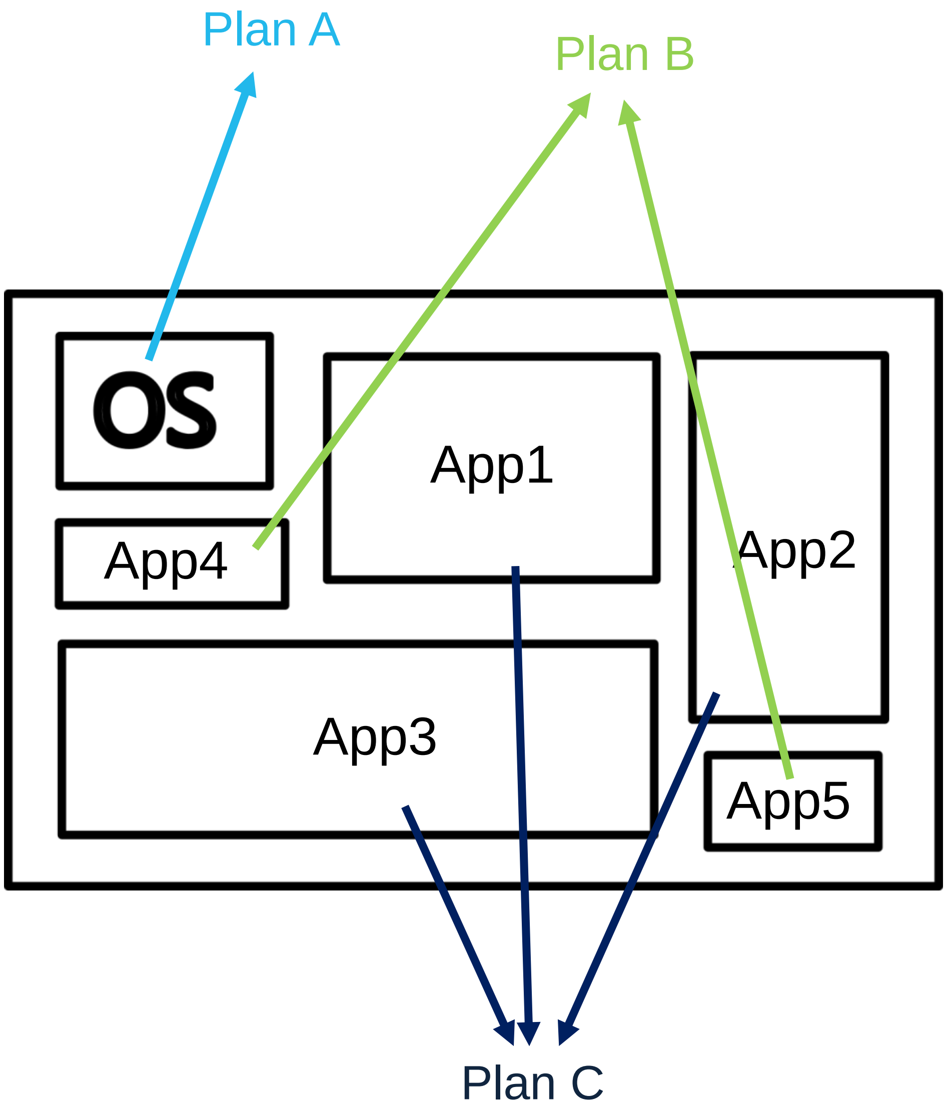
```
cgroups -$ ./tutorial

      (1) Introduction
      (2) Using cgroups

      > 1
       Introduction
            |- organization
            |- tasks and subsystems
            |- hierarchies and rules
      > _
```

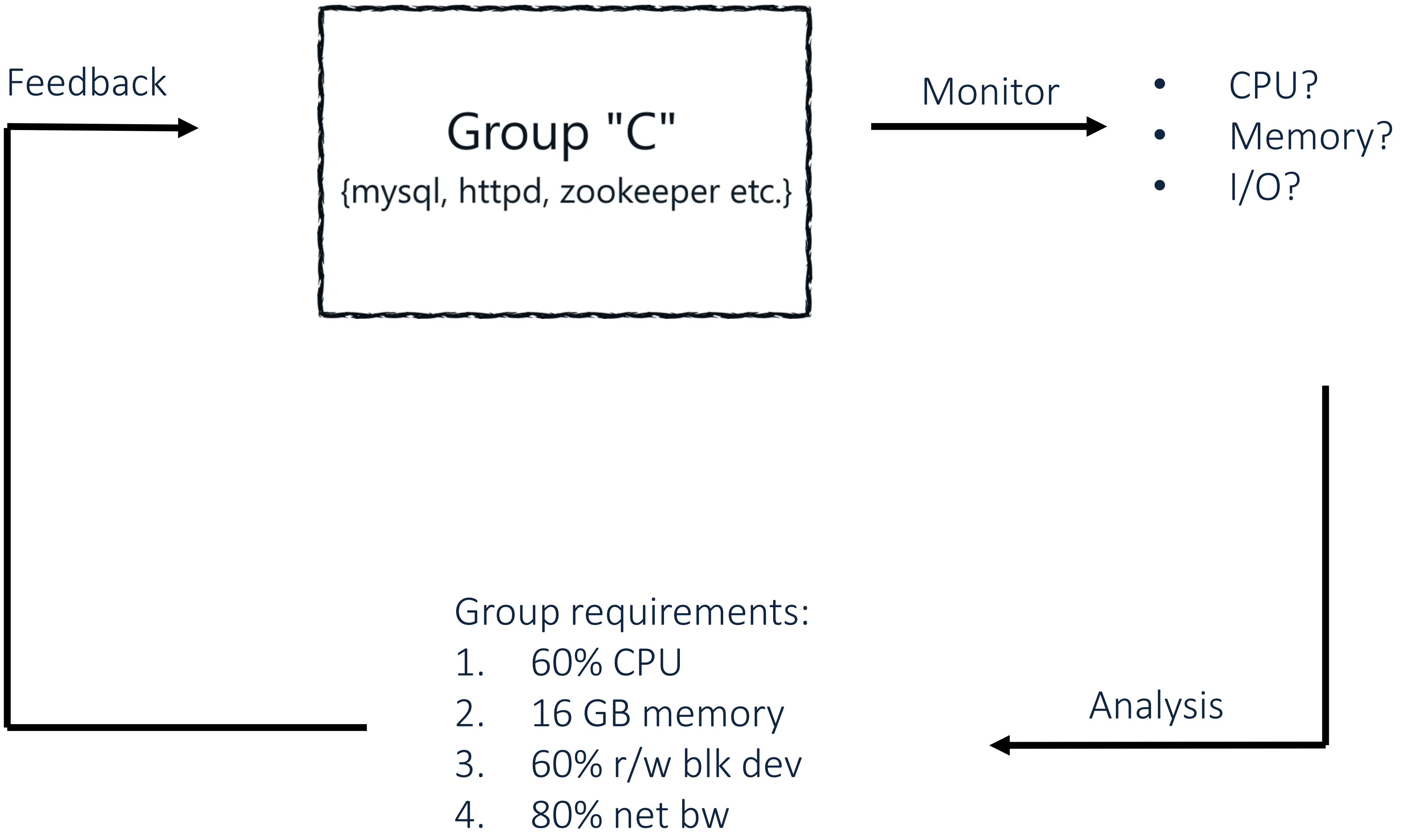
Whetting your appetite



App specs:

- CPU time
- System memory
- Disk bandwidth
- Network bandwidth
- Monitoring





Group "C"

{mysql, httpd, zookeeper etc.}

Monitor

- CPU?
- Memory?
- I/O?

Analysis

Group requirements:

1. 60% CPU
2. 16 GB memory
3. 60% r/w blk dev
4. 80% net bw

Feedback

What are cgroups?

“Control Groups is a mechanism for aggregating/partitioning sets of **tasks,** and **all their future children**, into hierarchical groups described by specialized behavior. “

- lxr

What are control groups about?

What

Resource allocation management:

- ➢ CPU time
- ➢ System memory
- ➢ IOPS
- ➢ Network bandwidth

demanded by a group of tasks (processes) → **cgroup**

How

Operations – functionality:

- ✓ Monitoring
- ✓ Resource access denial
- ✓ Cgroup reconfiguration on-the-fly

Why

Fine grained control over:

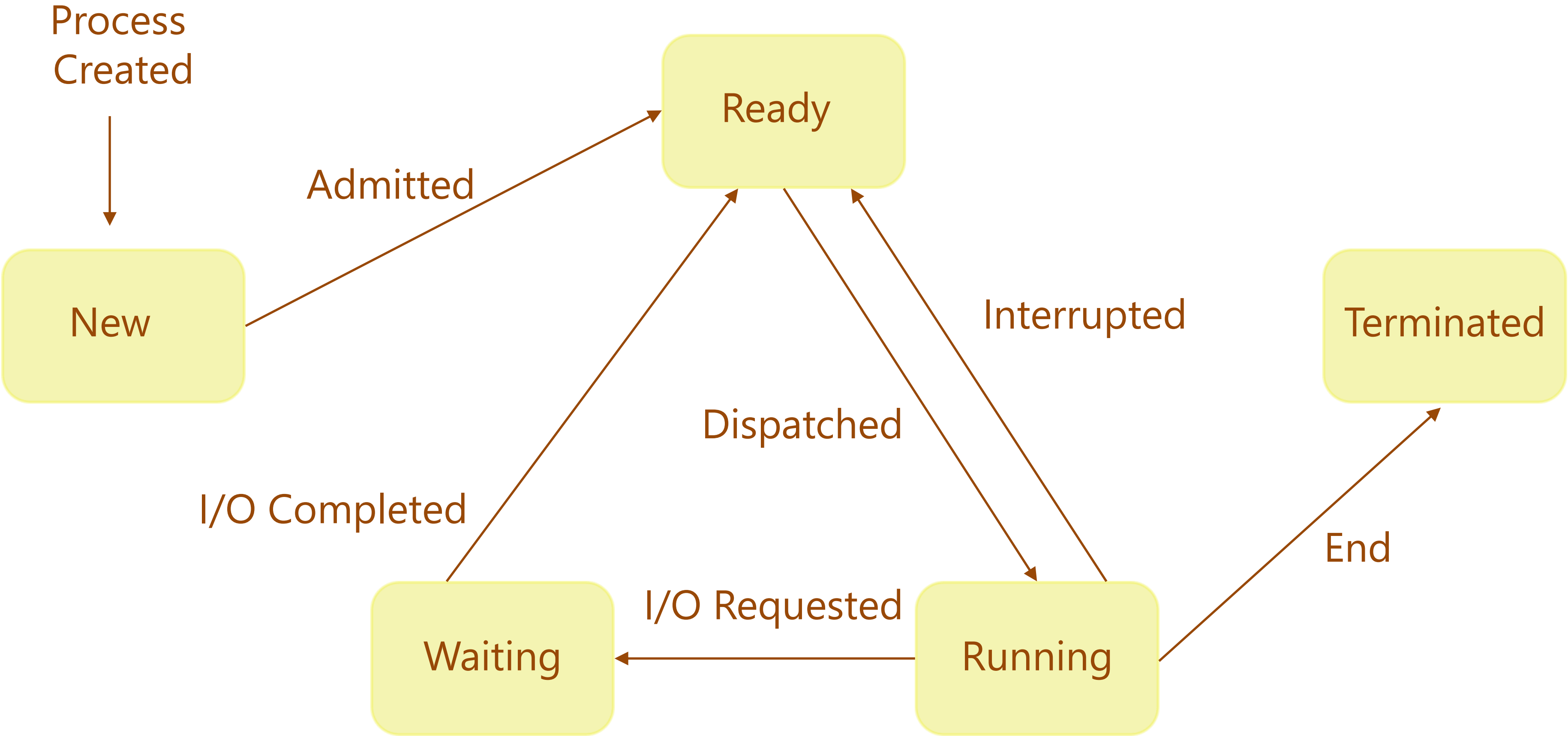
allocating, prioritizing, denying, managing and monitoring system resources to provide

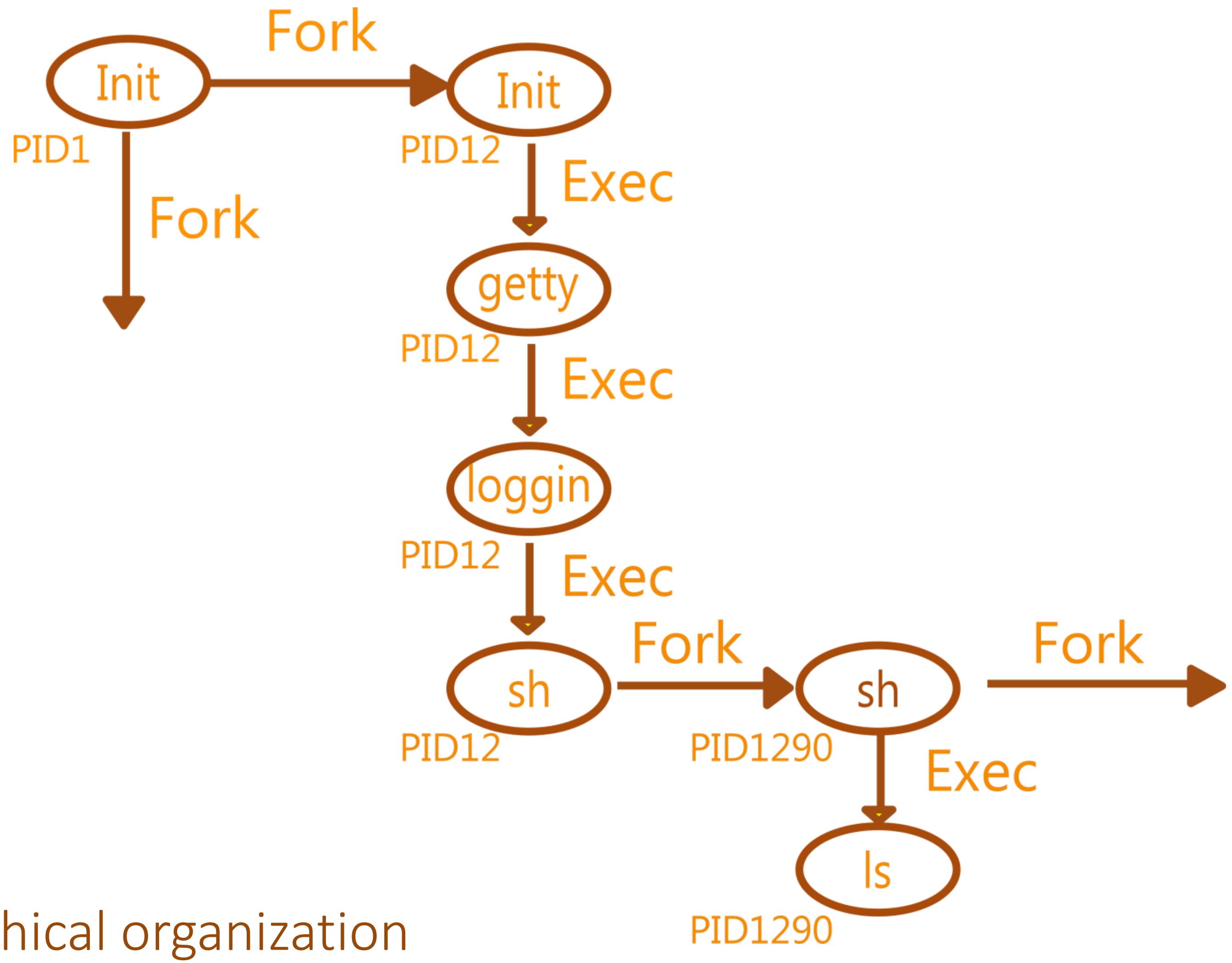
overall system efficiency

National Technical University of Athens

CSLab

Let’s recall the Linux Process Model!





> Hierarchical organization

> Attribute inheritance

Control group organization

"cgroup ::= user defined group of processes attached
to user defined resource management plans/policies"

- Hierarchical organization
  - Consider linux processes organization → tree
  - Cgroups organization → forest (with trees sharing leaves!)

- Attribute inheritance

+ many different **hierarchies** of cgroups can exist simultaneously

+ each hierarchy is **attached** to one or more **subsystems**

Subsystem ?

“ A *subsystem* is a module that makes use of the task grouping facilities provided by cgroups to treat groups of tasks in particular ways.

A subsystem is typically a "**resource controller**" that **schedules** a resource or applies per-cgroup **limits**, but it may be **anything that wants to act on a group of processes**, e.g. a virtualization subsystem. ”

- lxr

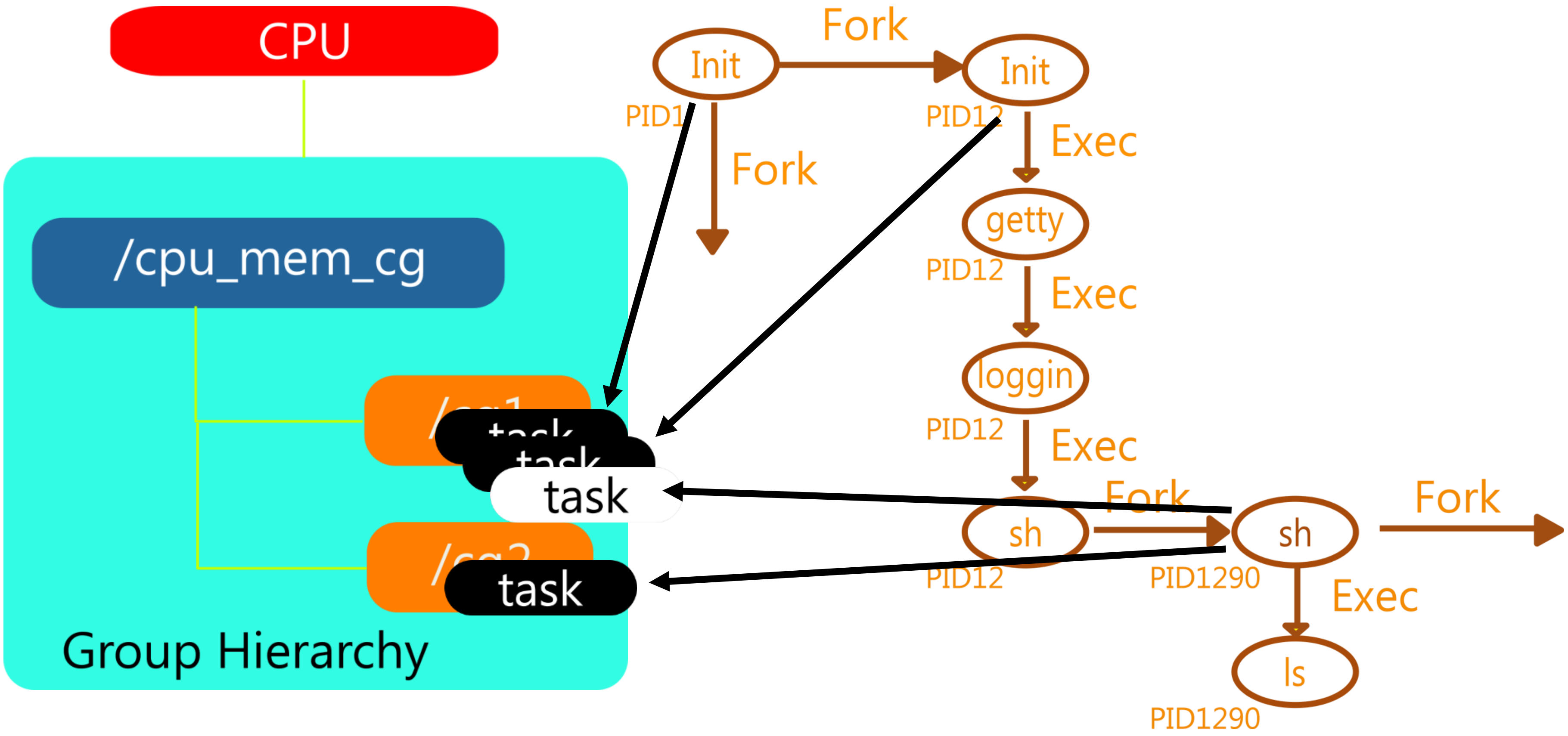
Available Subsystems

- blkio: limits per cgroup block I/O (disk, solid state, USB, etc.)
- cpu: enables setting of scheduling preferences on per-cgroup basis
- cpuacct: generates reports on CPU resources used by tasks in a cgroup
- cpuset: facilitate assigning a set of CPUS and memory nodes to cgroups. Tasks in a cpuset cgroup may only be scheduled on CPUS assigned to that cpuset
- devices: controls the ability of tasks to create or use devices nodes using either a blacklist or whitelist
- freezer: provides a way to 'freeze' and 'thaw' whole cgroups. Tasks in the cgroup will not be scheduled while they are frozen
- memory: allows memory, kernel memory, and swap usage to be tracked and limited
- net_cls: provides an interface for tagging packets based on the sender cgroup. These tags can then be used by traffic controller to assign priorities
- net_prio: allows setting network traffic priority on a per-cgroup basis
- perf_event: enables per-cpu mode to monitor only threads in certain cgroups

Hierarchy?

A *hierarchy* is a set of cgroups arranged in a tree, such that every task in the system is in exactly one of the cgroups in the hierarchy, and a set of subsystems; each subsystem has system-specific state attached to each cgroup in the hierarchy. Each hierarchy has an instance of the cgroup virtual filesystem associated with it.

- lxr



CPU

/cpu_mem_cg

/cg1

task

task

task

/cg2

task

Group Hierarchy

Init

PID1

Fork

Init

PID12

Fork

Exec

getty

PID12

Exec

loggin

PID12

Exec

sh

PID12

Fork

sh

PID1290

Fork

Exec

ls

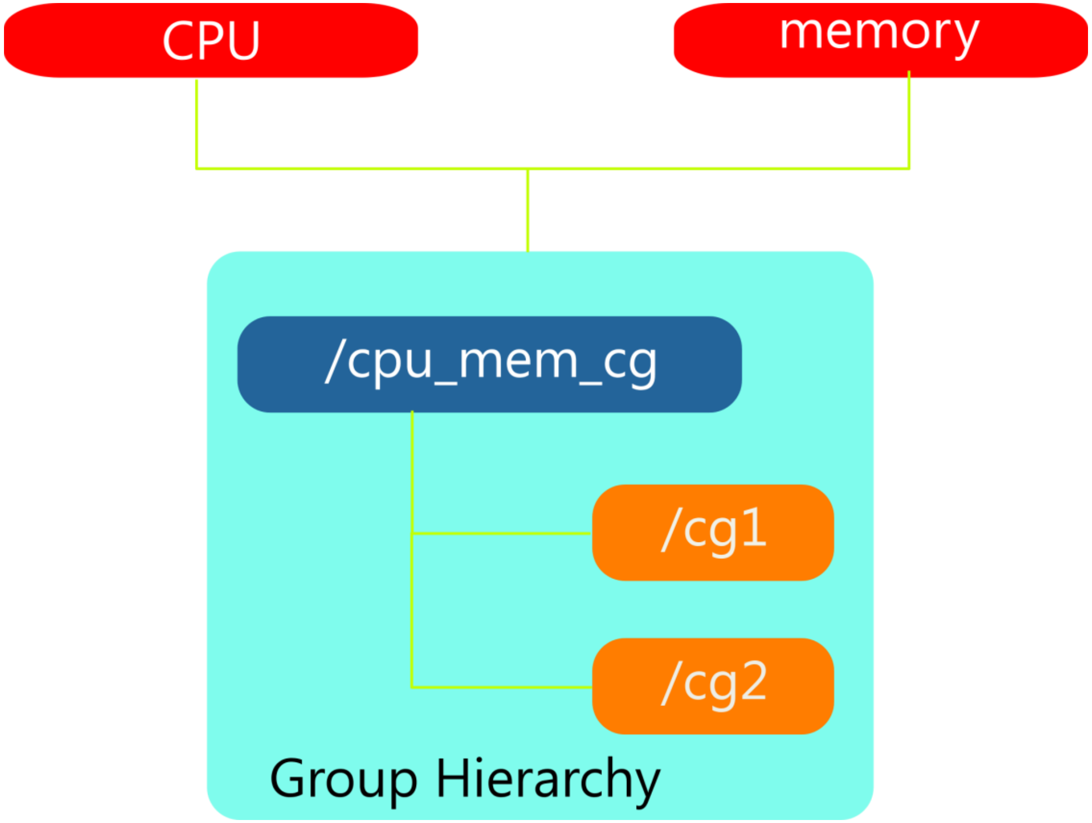
PID1290

Attaching tasks to subsystems via cgroups

- rules and principles

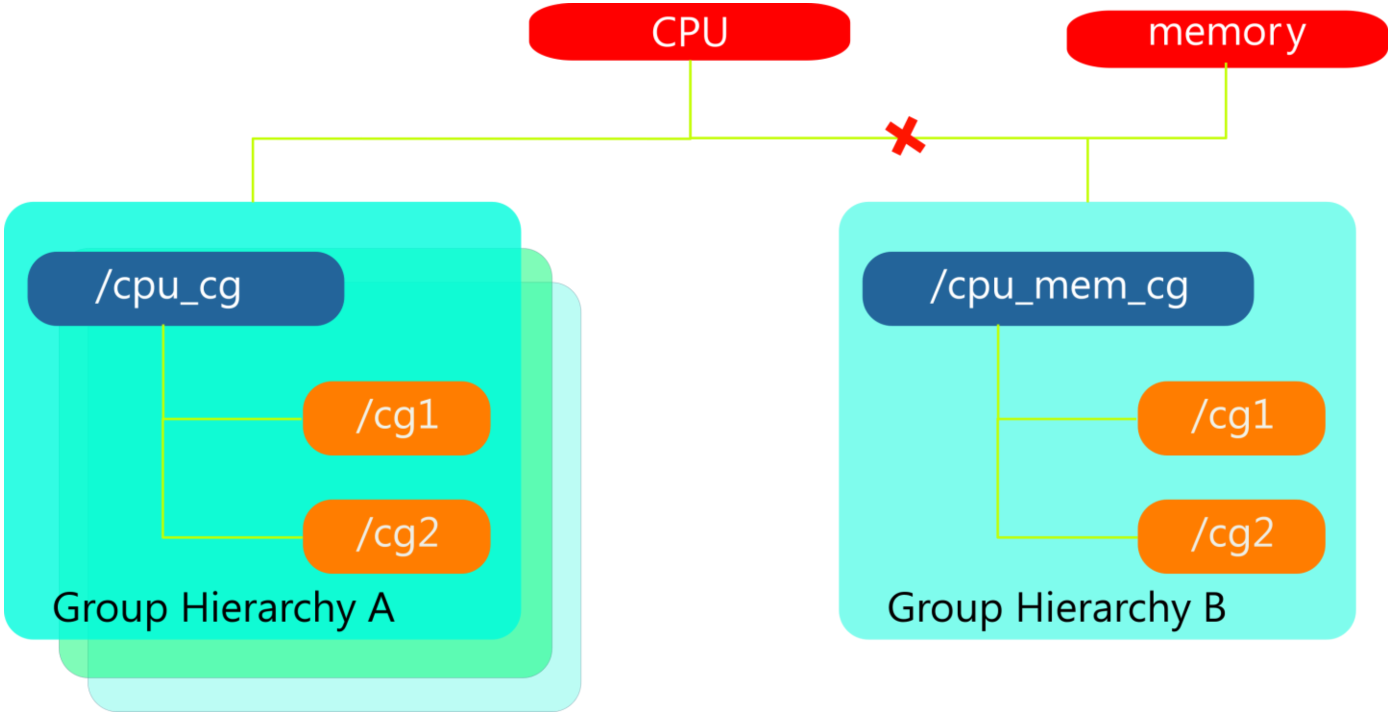
Rule I

“A single hierarchy can have one or more subsystems attached to it.”



Rule II

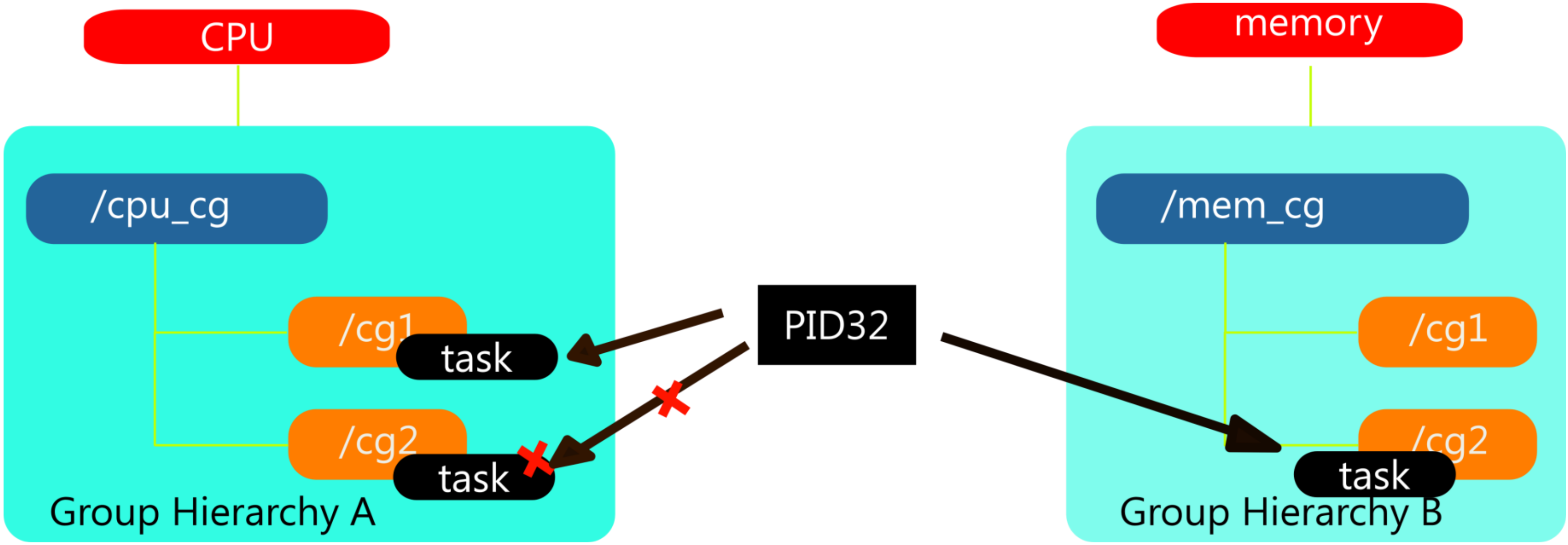
Any single subsystem (such as cpu) cannot be attached to more than one hierarchy if one of those hierarchies has a different subsystem attached to it already.



Note: A single subsystem can be attached to more than one hierarchies as long as they do not have any other subsystem already attached

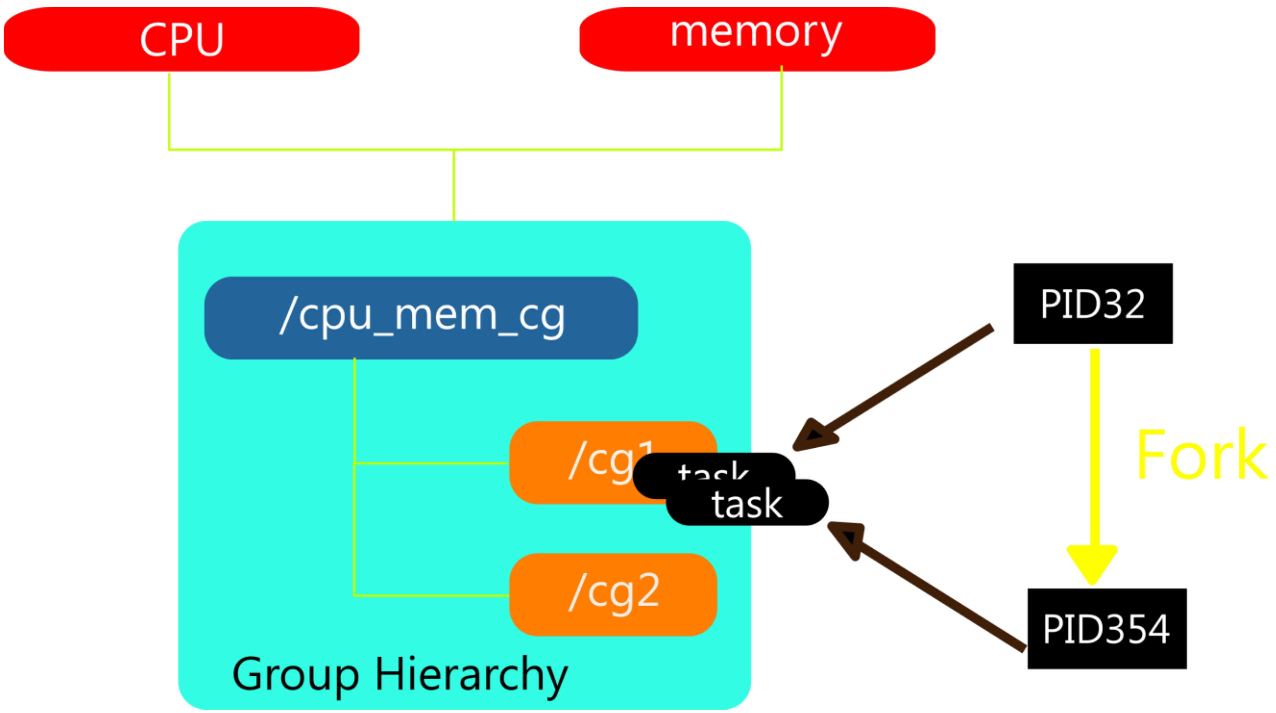
Rule III

- ❑ New hierarchy → all tasks belongs to root cgroup (recall process organization)
- ❑ A single task can belong to distinct cgroups of distinct hierarchies
  - ❑ Recall "sharing leaves"
- ❑ A single task can not belong in distinct cgroups of the same hierarchy



Rule IV

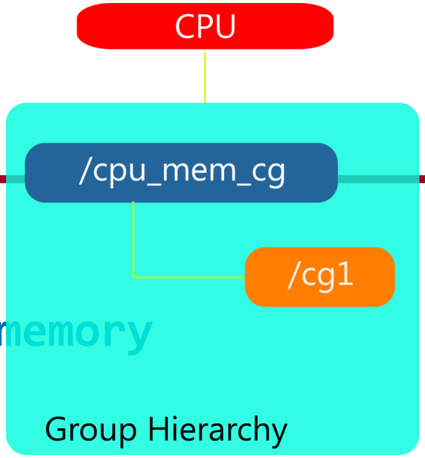
- ❑ Process fork → child task
- ❑ Parent membership inheritance (recall process fork inheritance)



Getting our hands dirty!

```
cgroups -$ ./tutorial

    (1) Introduction

    (2) Using cgroups

    > 2
     Using cgroups
          |- Files and cgroup VFS
          |- Hands on
          |- The cpu subsystem


    > _
```

Files and cgroup VFS



```
cgroups -$ ls /sys/fs/cgroup
blkio cpu cpuacct cpuset devices freezer hugetlb memory
perf_event  system

cgroups -$ ls /sys/fs/cgroup/cpu_mem_cg
cgroup.clone_children cgroup.procs cpu.cfs_period_us cpu.shares
notify_on_release tasks cgroup.event_control
cgroup.sane_behavior  cpu.cfs_quota_us cpu.stat release_agent
cg1
```

Hint:

- tasks → list of attached tasks by pid
- cgroup.procs → list of thread group IDs in the cgroup
- notify_on_release → flag, “run the release agent on exit?”
- release_agent → path to use for release notifications
- **Other files** → depends on the policy expression model

Files and cgroup VFS

```
cgroups -$ ls /proc
      … cgroups …              → available cgroup hierarchies

cgroups -$ ls /proc/764
      … cgroup …               → what this task is attached to?
                               path relative to the cgroup file system
```

Hands on

Temporary fs

Device type

```
1. mount -t tmpfs cgroup_root /sys/fs/cgroup
2. mkdir /sys/fs/cgroup/cpuset
3. mount -t cgroup –o cpuset cpuset /sys/fs/cgroup/cpuset
```

4. Create the new cgroup by doing mkdir's and write's (or echo's) the /sys/fs/cgroup/cpuset virtual file system.
5. Start a task that will be the "founding father" of the new job.
6. Attach that task to the new cgroup by writing its PID to the /sys/fs/cgroup/cpuset tasks file for that cgroup.
7. fork, exec or clone the job tasks from this founding father task.

cpu subsystem

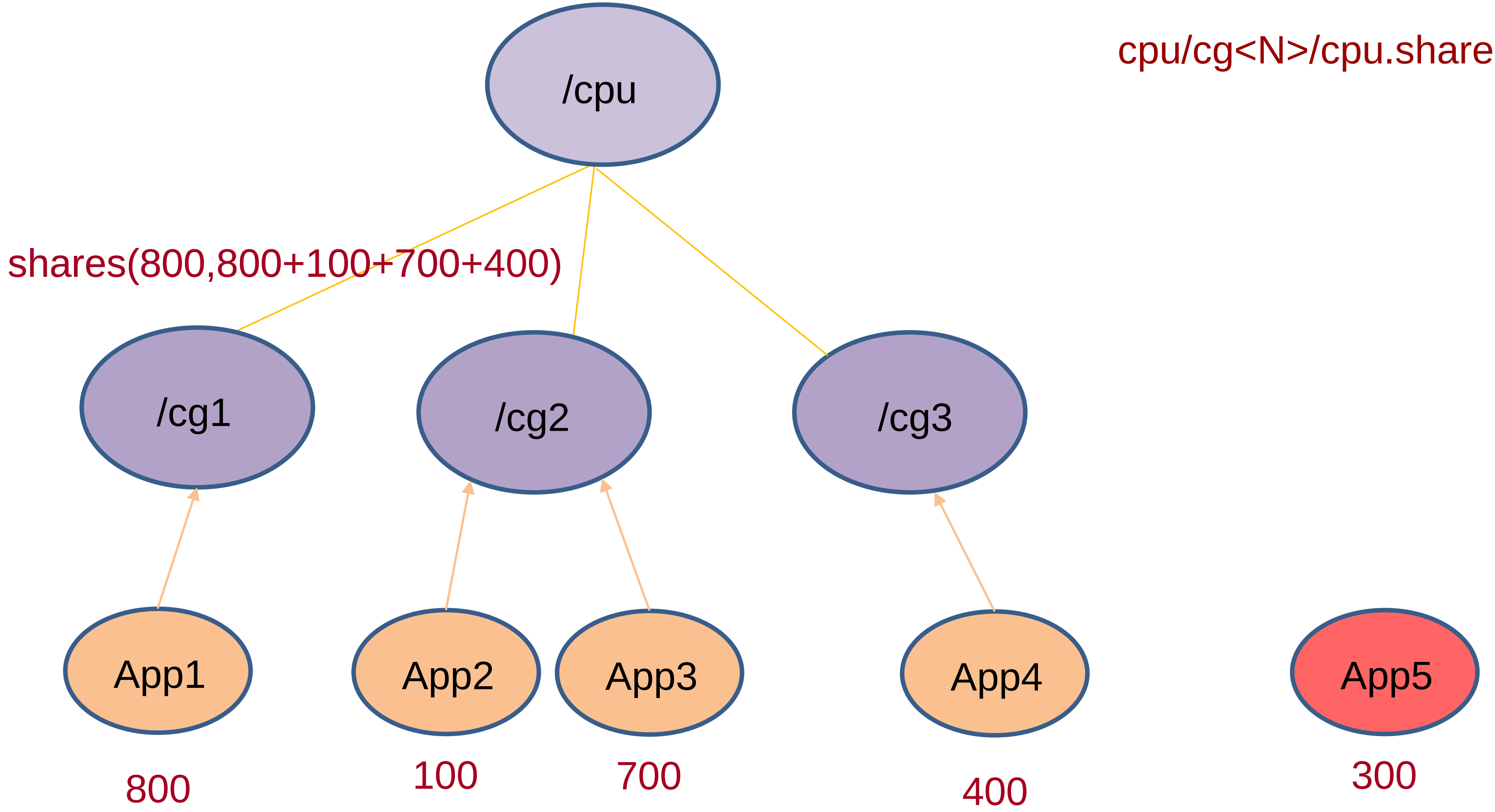
1. **Completely Fair Scheduler (CFS)** — a proportional share scheduler which divides the CPU time (CPU bandwidth) proportionately between groups of tasks (cgroups) depending on the priority/weight of the task or shares assigned to cgroups.

2. **Real-Time scheduler (RT**) — a task scheduler that provides a way to specify the amount of CPU time that real-time tasks can use.

   e.g.: All tasks in a cgroup are allowed to run 0.1 seconds in every 1 second

cpu subsystem, CFS

cpu/cg<N>/cpu.shares



Άσκηση

Εκτελέσιμα

1. **Δαίμων cgmond**
2. **cgmon-policy**
   - **Input:** policy:<application name>:cpu:<value>
   - **Output:** score:<float>
     set_limit:<application name>:cpu.shares:<value>
3. **cgmon-limit**
   1. Δημιουργία cgroup για μια νέα εφαρμογή:
      - create:<monitor>:cpu:<application name>
   2. Κατάργηση του cgroup μιας εφαρμογής που έχει τερματίσει:
      - remove:<monitor>:cpu:<application name>
   3. Εγγραφή μιας διεργασίας στο cgroup μιας εφαρμογής:
      - add:<monitor>:cpu:<application name>:<process id>
   4. Ρύθμιση της τιμής cpu.shares για το cgroup μιας εφαρμογής:
      set_limit:<monitor>:cpu:<applicationname>:cpu.shares:<value>

For extra info you may refer to …

1. http://lxr.free-electrons.com/source/kernel/cgroup.c
2. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch01.html
3. http://stackoverflow.com/