ORACLE

Cgroups and Enterprise Users

Linux Plumbers 2023

Tom Hromatka, Kamalesh Babulal

tom.hromatka@oracle.com, kamalesh.babulal@oracle.com

Linux Kernel Development

Oracle Corporation

Nov 13th, 2023

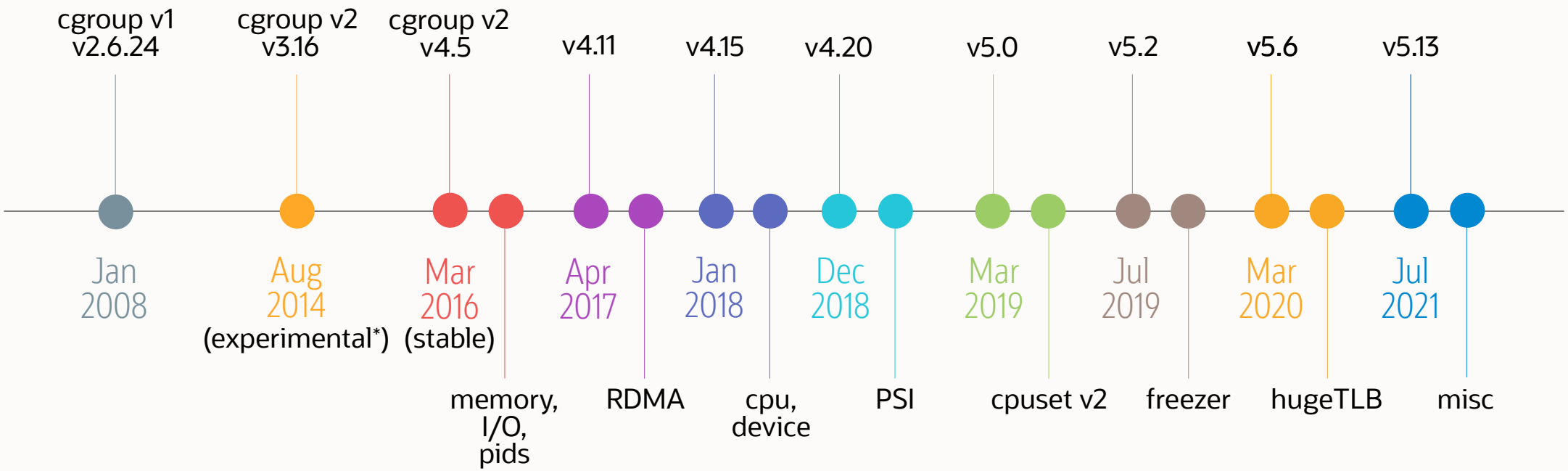
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Kernel cgroup

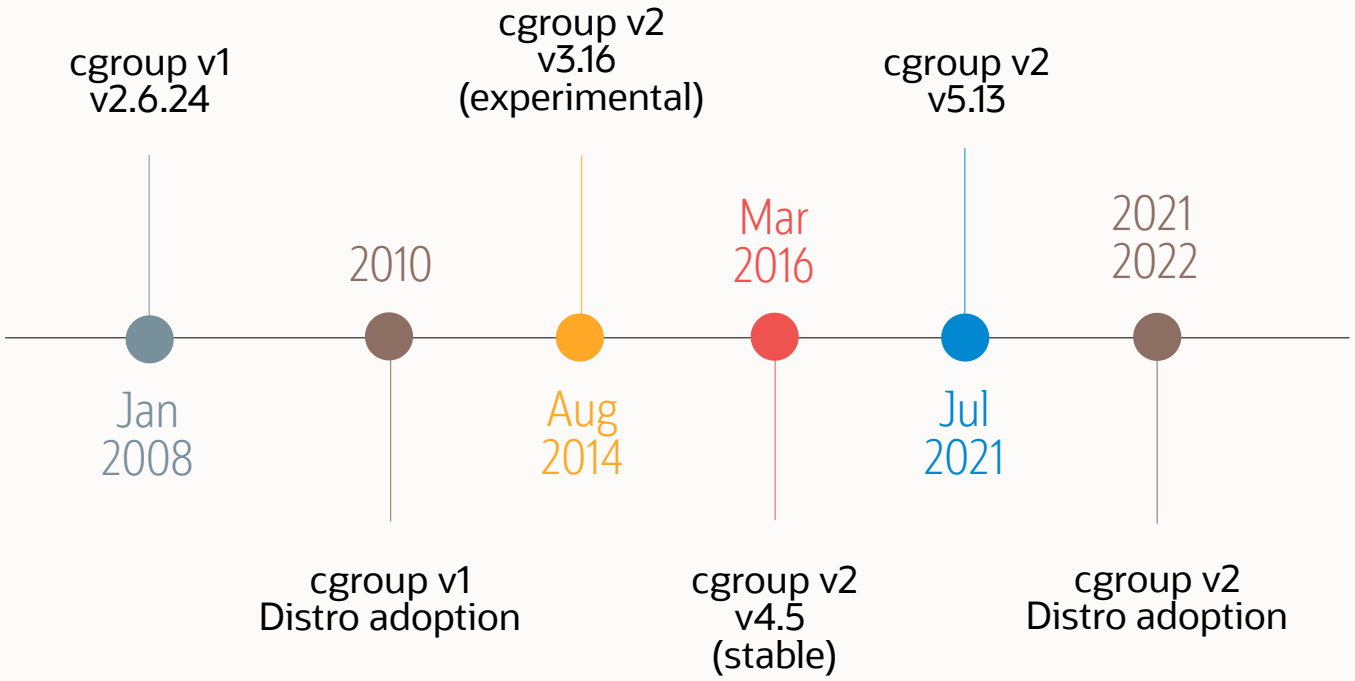
Upstream releases



*mount -t cgroup -o __DEVEL__sane_behavior cgroup <mount-point>

Enterprises cgroup

Upstream vs Distribution Adoption



Enterprise Distributions

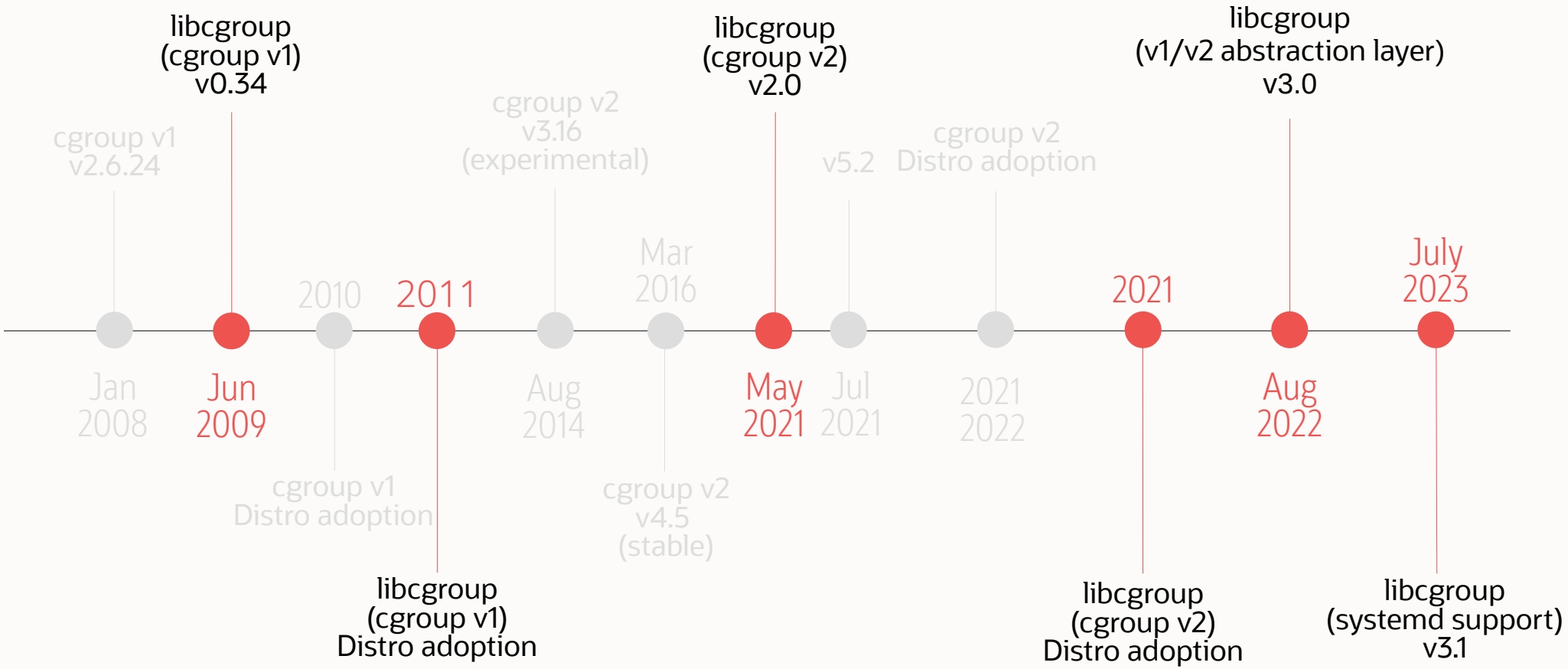
- LTS of 10 years or more

Enterprise Customers

- Applications runs on multiple distros/release
- Long term Application support
- Need to support cgroup v1/v2

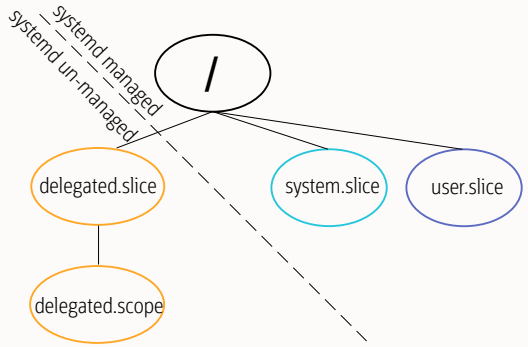
Libcgroup Releases

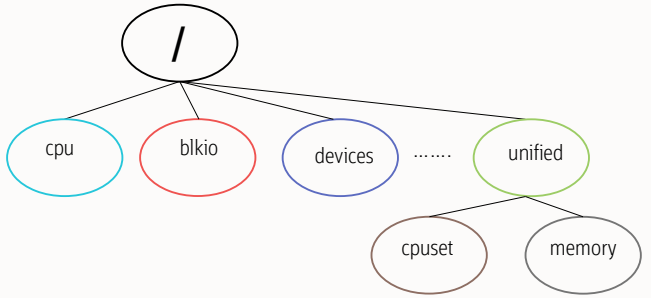
Upstream vs Distribution adoption vs libcgroup release



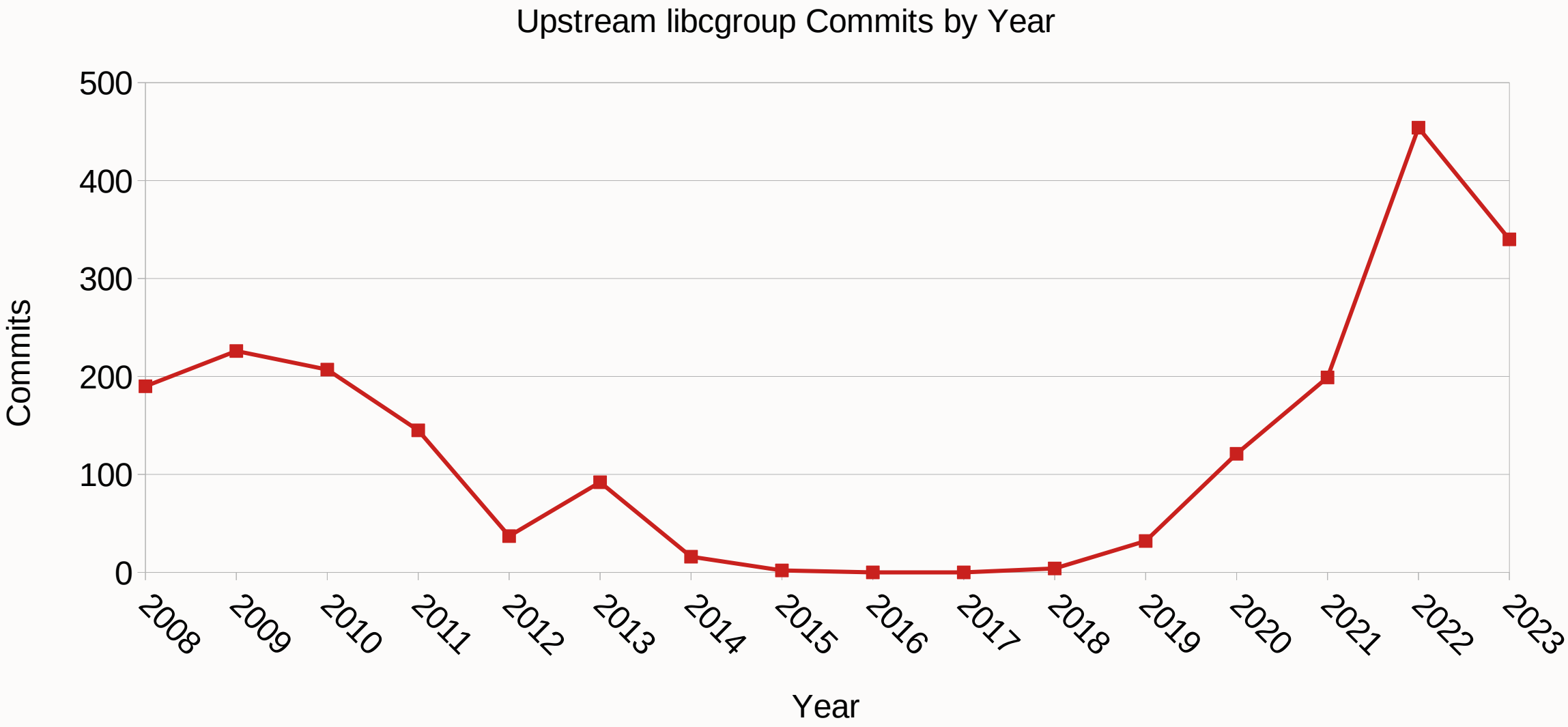
Why/When to Use libcgroup

- Programmatic management of delegated scopes
- CLI management of delegated scopes
- Complex cgroup hierarchies
- Different hierarchies across v1 controllers
- Different hierarchies across v1 and unified controllers
- Threaded cgroups
- Transient cgroups
- Cgroup settings not managed by systemd, e.g. cpusets
- Support for Python bindings

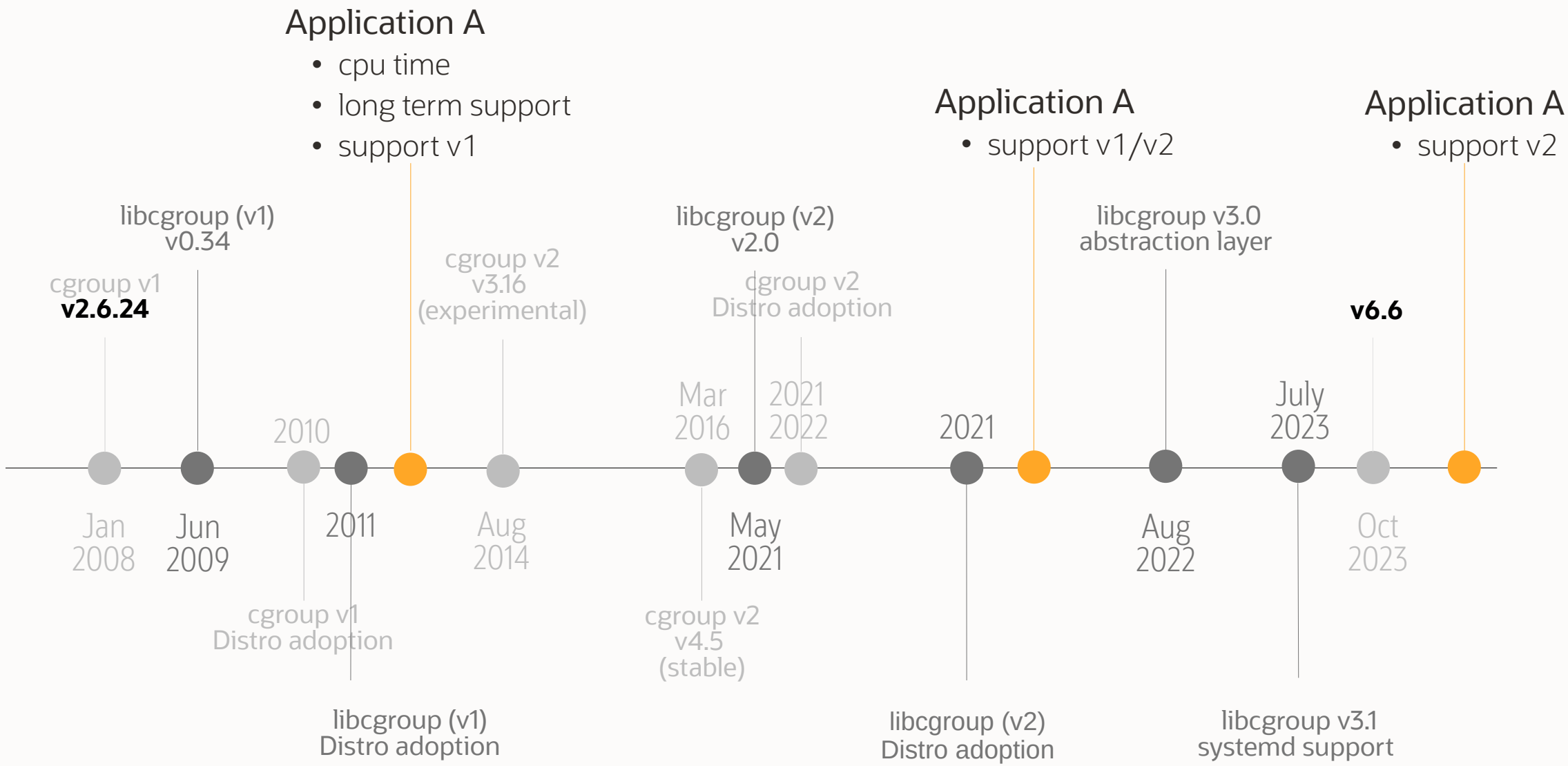




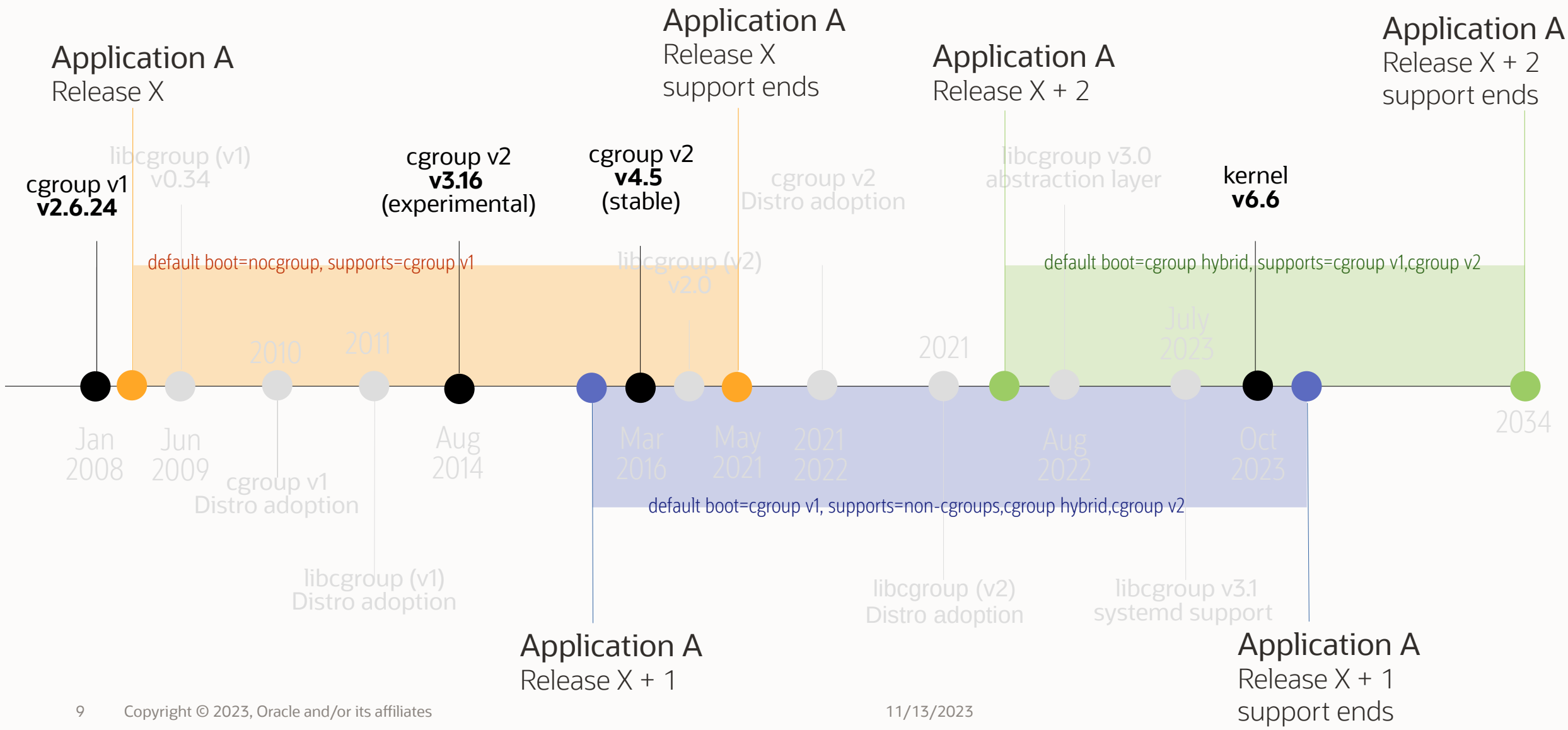
Upstream libcgroup Activity



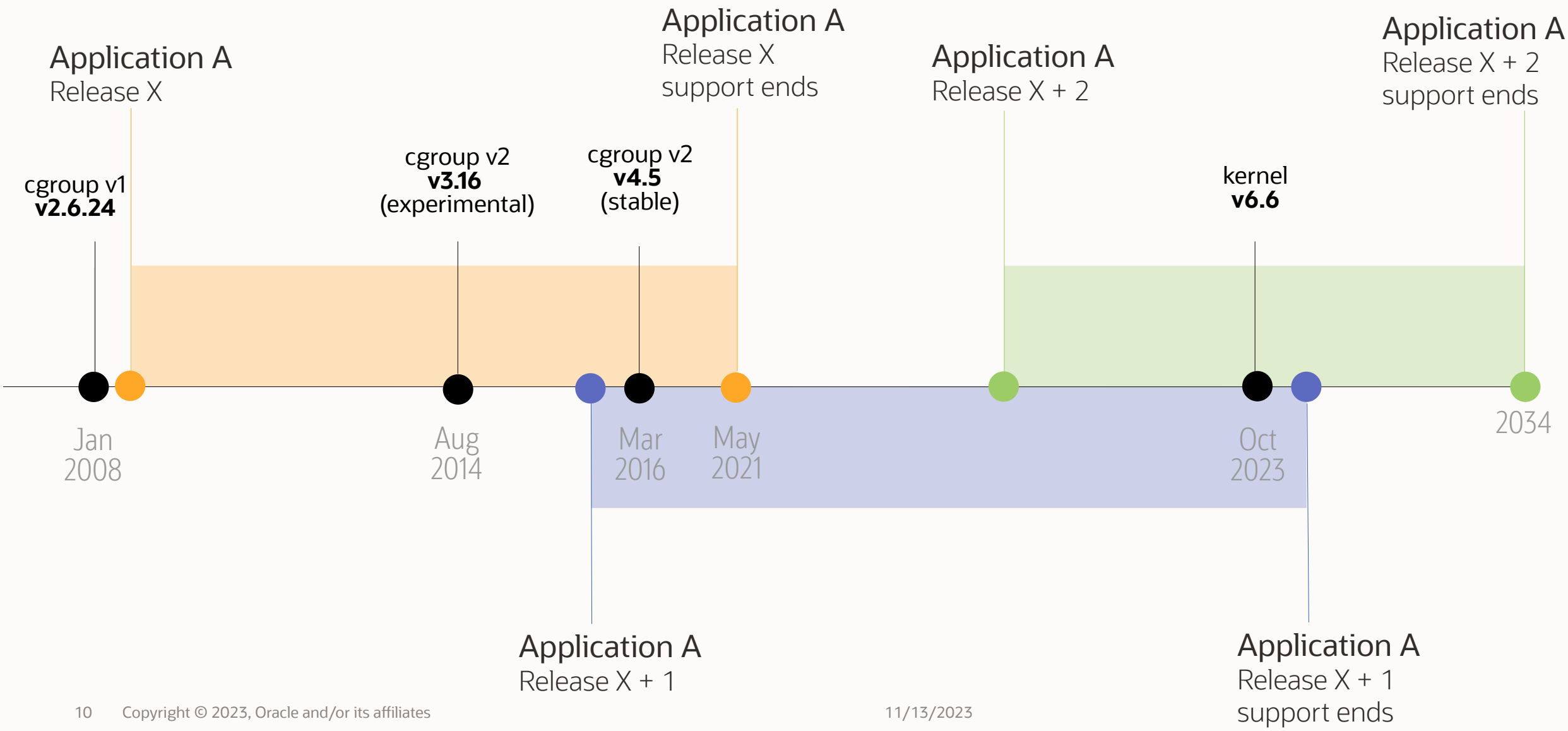
Enterprises Application Example



Enterprise Application Life Cycle



Enterprise Application Life Cycle



Challenges for Enterprise Users

Exclusive cpusets and root cgroup

- v2 - a process in the root cgroup cannot access a child cgroup's exclusive cpuset cpus
- v1 - root cgroup entirely ignores child cgroup's exclusive cpusets

Exclusive cpusets and child cgroups

- v6.6 - parent cgroups set as root can exclusively allocate all cpus to their children
- v5.15 - parent cgroups must keep at least 1 cpu available to themselves

cpu controller and root cgroup

- Given the default cpu.shares/cpu.weight, one process in the root cgroup has equal weight to a child cgroup
- Inconsistent spawning of processes in the root cgroup vs a child cgroup like system.slice

Single writer rule

- systemd owns the root cgroup and its children in the root cgroup
- A delegated cgroup (company.slice/app.scope) is the only approved method for a user-managed cgroup
- Generally more strictly enforced on cgroup v2 and newer versions of systemd
- This rule didn’t exist during early cgroup adoption

libcgroup features to assist moving from cgroup v1 to v2

Abstraction layer support

- Maps v1 controller settings to v2 controller settings and vice versa

```
cgxset -1 -r cpu.shares=512 <cgroup>

cgroup_convert_cgroup(out_cgroup, out_version,
                in_cgroup, in_version);
```

- Not all mappings are available, especially multiline controller settings.

Systemd delegation support

- Allow creation of systemd slice and scope (transient) programatically and tools to support them

```
cgcreate -S -c
-gcpu,memory:libcgroup.slice/database.scope

cgroup_create_scope2(cgroup, ignore_ownership,
                  cgroup_systemd_scope_opts);
```

Abstraction layer support cgconfig.conf

- /etc/cgconfig.conf will automap v1 settings to v2 settings

```
group foo {
     cpu {
          cpu.shares=512;
     }
}
```

Systemd delegation support cgconfig.conf

- /etc/cgconfig.conf will automap v1 settings to v2 settings

```
systemd {
     slice = database.slice;
     scope = db.scope;
     setdefault = yes;
}
```

Discussion Topics

- Re-adoption of libcgroup into distros
- Are there upcoming features (cgroups, systemd, etc.) that could break assumptions made by these legacy applications?
- Are there upcoming features that may be of interest to these legacy applications?
- Strategies for applications that must support cgroup v1/v2, new and old systemd versions, and a wide range of kernel versions
  - See inconsistencies slide from earlier
- Ways to increase “proper” usage of cgroups – single writer rule, no-inner-node processes, etc.
- Ways to discourage improper usage of cgroups

Thank You


ORACLE

Backup Slides

cgroup v2

Controllers	Kernel Version		libcgroup Release
memory, I/O, pids controllers	- v4.5	(Mar 2016)	
rdma controller	- v4.11	(Apr 2017)	
cpu controller	- v4.15	(Jan 2018)	
block I/O latency controller	- v4.19	(Oct 2018)	
Pressure Stall Information	- v4.20	(Dec 2018)	
cpuset v2	- v5.0	(Mar 2019)	
freezer	- v5.2	(Jul 2019)	
Kernel Release	- v5.5	(Jan 2020)	libcgroup v0.42
hugetlb	- v5.6	(Mar 2020)	
Kernel Release	- v5.12	(Apr 2020)	libcgroup v2.0
misc	- v5.13	(Jun 2021)	
Kernel Release	- v5.17	(Mar 2022)	libcgroup v2.0.1
Kernel Release	- v5.18	(May 2022)	libcgroup v2.0.2
Kernel Release	- v5.19	(Jul 2022)	libcgroup v3.0
Kernel Release	-v6.2	(Feb 2023)	libcgroup v2.0.3

cgroup adoption

Distribution	Release Version	Release Date	libcgroup version
Oracle Linux	9	Jun 2022	3.0
RedHat Enterprise Linux	9	May 2022	N/A
SuSE Linux Enterprise Server			
Ubuntu Server	21.10	Oct 2021	2.0.2.2