September 12, 2022

systemd cgroup delegation and control processes

Michal Koutný

What is delegation?

- single writer design rule

`Delegate=yes`

Units where this is enabled may create and manage their own private subhierarchy of control groups below the control group of the unit itself.
...concept of ownership is established the control group tree above the unit's control group (i.e. towards the root control group) is owned and managed by the service manager of the host, while the control group tree below the unit's control group is owned and managed by the unit itself.

Service with a control command

```
[Service]
#Delegate=no
ExecStart=/usr/bin/server
ExecReload=/usr/bin/reload
```

Service without delegation and control command

```
/system.slice/plain-control.service
├─20001 /usr/bin/server
└─40001 /usr/bin/reload
```

What control commands are there

- `ExecStartPre=`, `ExecCondition=` run before payload
- `ExecStartPost=`
- `ExecReload=`
- `ExecStopPre=`, `ExecStop=`, `ExecStopPost=`

Service with delegation

```
/system.slice/delegate-control.service
├─cgroup.subtree_control  (rw)
├─custom-main
│ └─20001 /usr/bin/server
└─custom-side
  └─30001 /usr/bin/helper
```

Internal node constraint

```
/system.slice/delegate-control.service
├─custom-main
│ └─20001 /usr/bin/server
├─custom-side
│ └─30001 /usr/bin/helper
└─40001 /usr/bin/reload   (!!!)
```

Internal node constraint provision

```
/system.slice/delegate-control.service
├─custom-main
│ └─20001 /usr/bin/server
├─custom-side
│ └─30001 /usr/bin/helper
└─.control                  B-)
  └─40001 /usr/bin/reload
```

But also “unconsumed” delegation

```
/system.slice/delegate-control.service
├─cgroup.subtree_control  (rw)
├─20001 /usr/bin/server
└─.control
  └─40001 /usr/bin/reload
```

- constraint applies only with controllers

Motivational example

```
/-.slice/rt.service       cpuset.cpus=1-3
├─sensitive               cpuset.cpus=1,2
│ ├─20001 thread-1
│ └─20002 thread-2
└─auxiliary               cpuset.cpus=3
  └─30001 helper-thread
```

to create own hierarchy

```
[Service]
Delegate=yes
```

Motivational example – threaded subtree

- background: threaded and non-threaded controllers

```
/-.slice/rt.service       cgroup.type=domain threaded
├─sensitive               cgroup.type=threaded
│ ├─20001 thread-1
│ └─20002 thread-2
├─auxiliary               cgroup.type=threaded
│ └─30001 helper-thread
```

Motivational example – threaded subtree

- background: threaded and non-threaded controllers

```
/-.slice/rt.service       cgroup.type=domain threaded
├─sensitive               cgroup.type=threaded
│ ├─20001 thread-1
│ └─20002 thread-2
├─auxiliary               cgroup.type=threaded
│ └─30001 helper-thread

└─.control                cgroup.type=domain invalid
  └─xxxxx /usr/bin/reload
```

Motivational example – resource allocation

```
/-.slice/rt.service       cpuset.cpus=1-3
├─sensitive               cpuset.cpus=1,2 .partition=root
│ ├─20001 thread-1
│ └─20002 thread-2
├─auxiliary               cpuset.cpus=3
│ └─30001 helper-thread
└─.control                cpuset.cpus=???
  └─40001 /usr/bin/reload
```

My proposal

```
[Service]
Delegate=yes
DelegateControlControlGroup=my-control:my-payload
ExecStart=/usr/bin/server
ExecReload=/usr/bin/reload

/system.slice/delegate-control.service
├─my-payload
│ └─...
└─my-control
  └─40001 /usr/bin/reload
```

- defaults to `.control:.` (backwards compatible)
- payload wrapper is optional, control is `threaded` when needed
- PR#22937

Direct usage

```
# user@.service
[Service]
...
Delegate=pids memory cpu
ExecStart=/usr/lib/systemd/systemd --user
ExecReload=systemctl --user daemon-reload
DelegateControlControlGroup=init.scope
...
```

- allows reload by admin
- utilizes `init.scope` in the delegated subtree

Other approaches

- hardcoded partitions

```
/system.slice/delegate-control.service
├─fixed-payload
│ └─...
└─fixed-control
  └─40001 /usr/bin/reload
```

- global

```
├─/init.scope
│ ├─1 systemd
│ └─???
│   └─40001 /usr/bin/reload
└─/system.slice/delegate-control.service
  └─...
```

- status-quo + special flag for threaded delegation

Discussions

- mandatory payload wrapper
  - where are limits configured
  - how are allocations passed down
  - what with hybrid setups
  - purpose in `.scope` units
- weights of payload vs control
- control under payload (instead of opposite)
- payload's resource reflection
- controller implementation details
  - depth for cpu, partitioning for memory
- (restart) instance wrap cgroups

Summary

- problem
  - control commands and payload with delegation side by side
  - resource allocation, threaded mode
- proposed solution
  - merge?
  - what to change