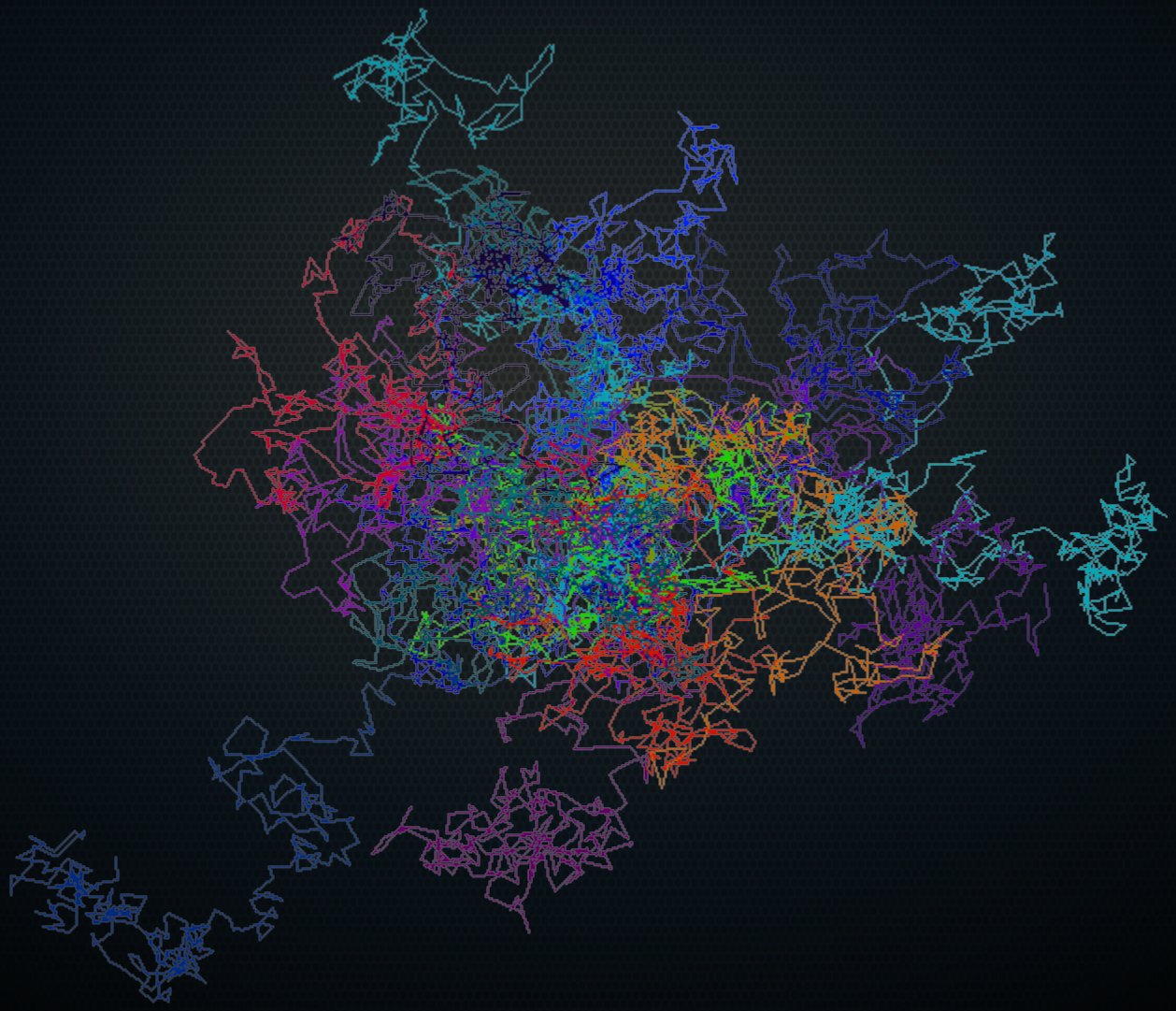
cgroup v2

Issues with the v1 interface



Goals of the v2 interface

Goals of the v2 interface

- Consistency.
- One way to achieve one goal. This ain’t perl.
- Usability not implementability.
- Flexibility with purpose.

A set of consistent interfaces and behaviors to provide **a** hierarchical grouping of processes and overlay various resource controls over **the** hierarchy.

The unified hierarchy

- The one hierarchical grouping of processes.
- A process’s cgroup membership can be specified with a single path.
- Different controllers have a common ground to talk to each other.
- per-process, not per-thread.
- Separation between organization and control.

cgroup.subtree_control

- Enable / disable controllers on the **children** cgroups of a given cgroup.
- Top-down.
- No tasks in internal nodes.

```
A(b,m) - B(b,m) - C (b)
          \     - D (b) - E
```

A is the root and has all controllers enabled.

1. echo +blkio > A/cgroup.subtree_control
   - blkio enabled on B
2. echo +blkio > A/B/cgroup.subtree_control
   - blkio enabled on C and D
3. echo +memory > A/cgroup.subtree_control
   - memory enabled on B

```
A(b,m) - B(b,m) - C (b)
          \     - D (b) - E
```

	A	B	C	D	E
blkio	A	B	C	D	D
memory	A	B	B	B	B

```
A(b,m) - B(b,m) - C (b)
          \     - D (b) - E
```

- Any controller which has any controller enabled in its cgroup.subtree_control can’t have processes in them.
- Root is the only exception.
- A, C, D and E can have processes in them but B can’t.

Groups and processes never compete directly. Groups compete against groups. Processes compete against processes within its group.

```
A(b,m) - B(b,m) - C (b)
          \     - D (b) - E
```

	A	*B*	C	D	E
blkio	A	*B*	C	D	D
memory	A	*B*	B	B	B

The unified hierarchy

- Clear separation between cgroup core’s role (process organization) and cgroup controllers’ role (resource control).
- Structural constraints which rule out ambiguous situations.
- Presents interface with consistent behavior to userland. Enforces controllers to conform to common conventions.
- Flexible enough to fulfill the core functionalities but rigid enough to encourage consistency.

Per-controller changes

Given the brownian motion each controller did, their behaviors need to be updated to conform to the new standard.

- Fully hierarchical.
- Organization and configuration should be orthogonal.
- Restructuring of messed-up interfaces and functionalities.
- General cleanups.

cpuset

- Explicit distinction between configured and effective configurations.
- A new child always has the same effective configuration as its parent.
- Organization is now mostly orthogonal to configuration and its enforcement.

memory

- The current interface is schizophrenic, especially selectable hierarchical behavior.
- Mostly useless softlimit. Heavy dependence on hardlimit ends up shifting dynamic control features to OOM killer.
- On-going cleanup of interface and implementation.
- Clearly defined min, high, max limits to make softlimit actually useful.

freezer

- The whole thing is braindamaged. A process state which should never be visible to userland is being exposed.
- And then working around it by bypassing OOM killing and whatnot.
- The wait state should be merged with jobctl stop.

blkio

- Interface being simplified without losing functionality.
- Resource control didn’t work on writeback IO traffic which is the majority of the write IOs on most configurations. Being worked on.

Timeline

- Experimental implementation already working in the upstream kernel.
- I suck at predicting timelines but it really isn’t too far out.
- Once writeback IO control is done. The v2 interface will be made officially available.
- Controllers will be gradually enabled on the v2 interface.

Questions?