cgroup writeback support

getting lost between memcg and blkcg

What's writeback?

The regulator

- triggered by memory pressure (dirty_background_ratio, dirty_ratio)
- propagates IO backpressure to dirtiers
- writeback is regulated by memory and IO at each end

implementation

- IO device is represented by backing_dev_info
- each bdi hosts one write worker
- bdi is also the focal junction for congestion state
- each bdi has one bdi_writeback embedded in it

memcg and blkcg

- memcg defines overall memory pressure for the cgroup
- blkcg regulates IO for the cgroup and thus should be able to generate backpressure which is propagated to the dirtiers

nothing worked

- no way to make controllers work together
- memcg tracks each page's ownership but couldn't communicate with blkcg
- blkcg couldn't control writeback IOs at all
- memcg had to implement custom ad-hoc writeback regulation logic
- people ended up doing weird stuff

let’s do it right

- controllers can refer to each other on unified hier
- shouldn’t be different from global case except for being split according to cgroups
- IO pressure from IO device - cgroup combination is propagated all the way up.
- !cgroup is a degenerate case of cgroup

cgroup writeback

plumbing

- backing_dev_info->bdi_writeback
- convert wb path to deal with bdi_writeback
- bdi-wide ops span across all existing wb's
- balance_dirty_pages() updated to deal with nested split mem domains and parameters

complications

- huge amount of plumbing through the layers splitting everything
- device-wide ops need to coordinate multiple writebacks and guarantee forward progress
- granularity mismatch between memcg and blkcg (page vs. inode, foreign inode detection using a variation of Boyer-Moore majority vote alg)

current status

- it works - global and memcg memory pressure and IO backpressure from blkcg shape page dirtying in the same way as in the global case
- scheduled for the coming merge window
- currently supported only on ext2 w/ blk-throttle