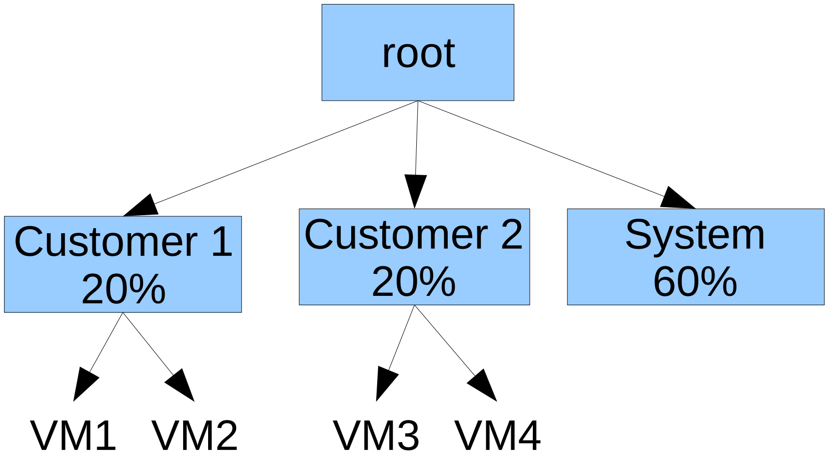
IO Controller
&
BIO Tracking

Fernando Luis Vázquez Cao
Nauman Rafique
Vivek Goyal

Why IO Controller

Enterprise Server



root

Customer 1 20%

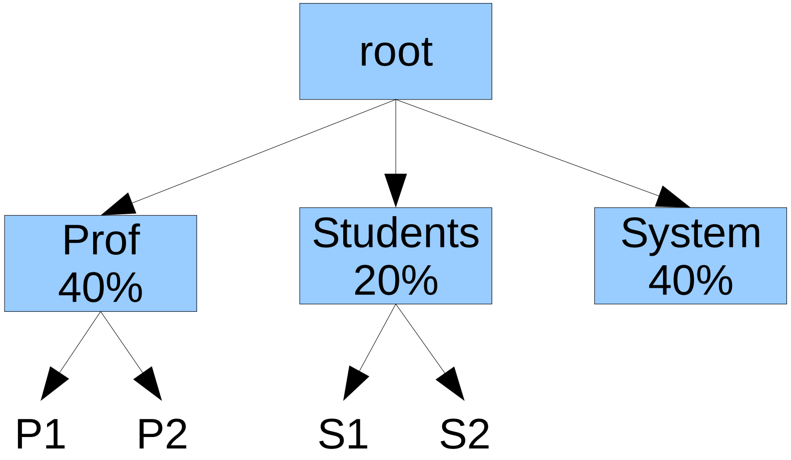
Customer 2 20%

System 60%

VM1 VM2

VM3 VM4

University Server



root

Prof 40%

Students 20%

System 40%

P1 P2

S1 S2

More sharing needs more isolation

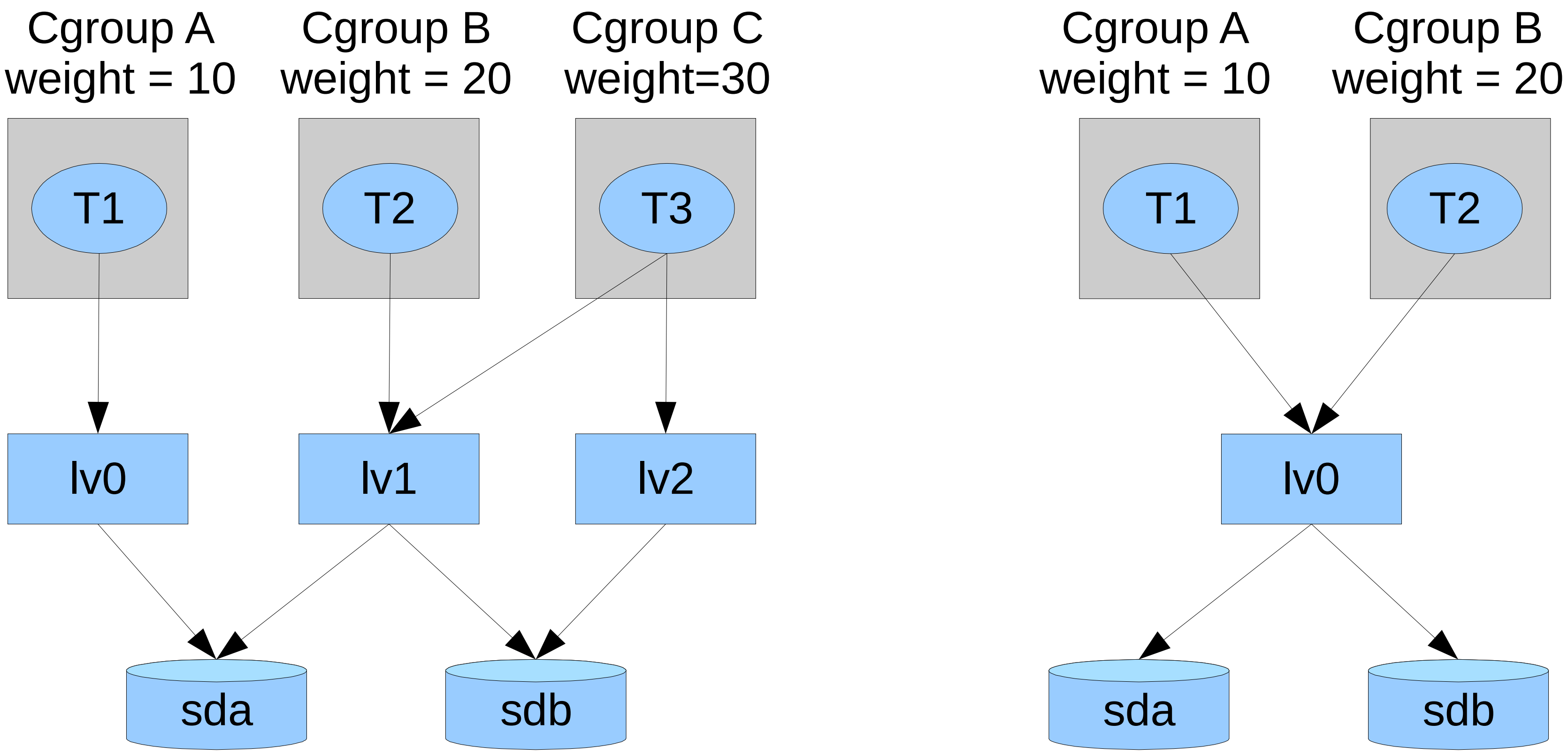
Resource guarantees/Predictability

Hierarchical group IO control

What to Control

- Proportional Weight/Prio Controller (CFQ)
  - Fair share of disk time (As CFQ does)
  - Fair share in terms of number of sectors transferred
  - Good throughput. Resource control done only if there is contention.
- Max Bandwidth Control (In terms of IO rate)
  - Don't allow usage of more resource if customer has paid for lower level of service.
  - How would one know the BW of a device to divide that in absolute numbers
- Both Proportional and Max Bandwidth Rate?

Where to control

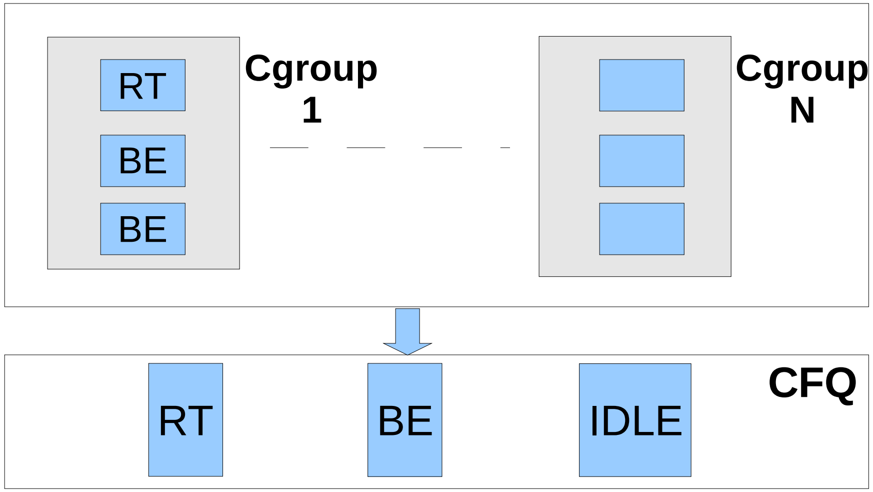


Control resources where real contention is?

Higher level control can be bad for throughput

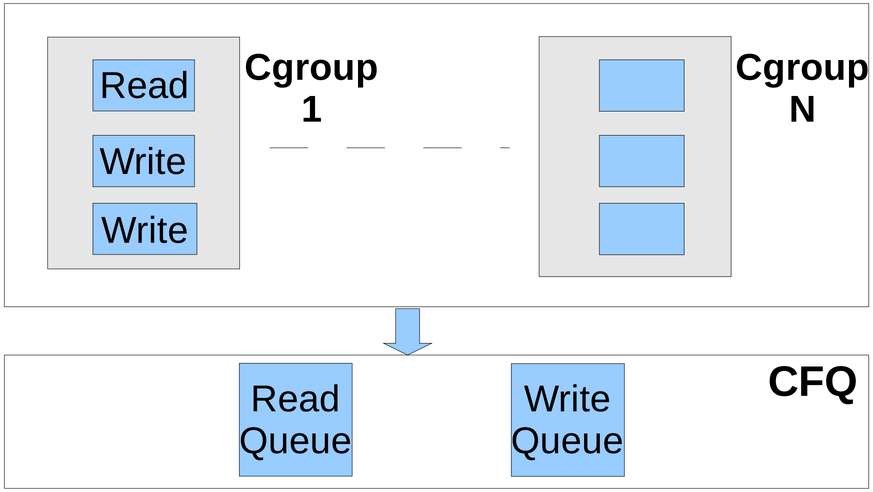
Two Level vs One Level Control

Priority Inversion



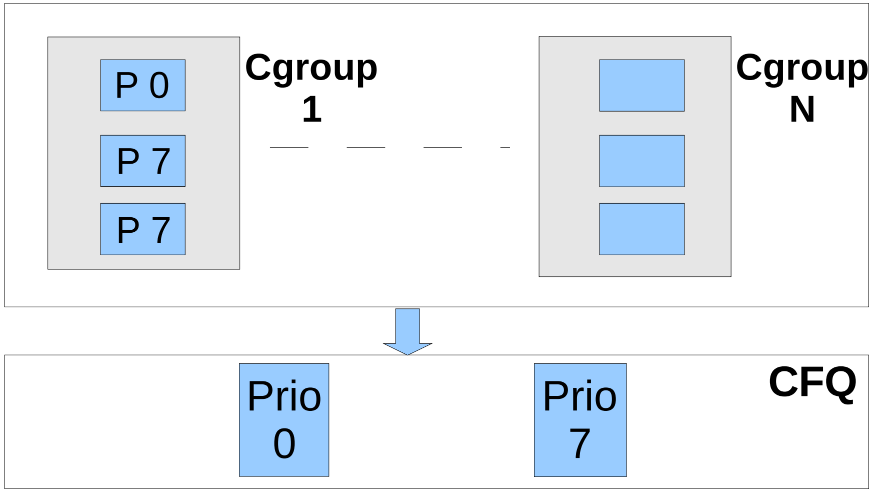
RT
BE
BE
Cgroup 1
Cgroup N
CFQ
RT
BE
IDLE

Reader Starvation



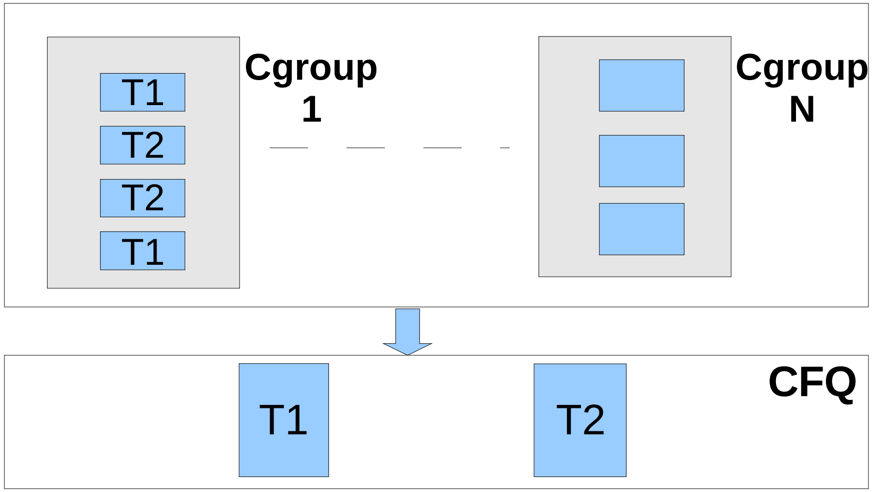
Read
Write
Write
Cgroup 1
Cgroup N
CFQ
Read Queue
Write Queue

Priority Inversion



P 0
P 7
P 7
Cgroup 1
Cgroup N
CFQ
Prio 0
Prio 7

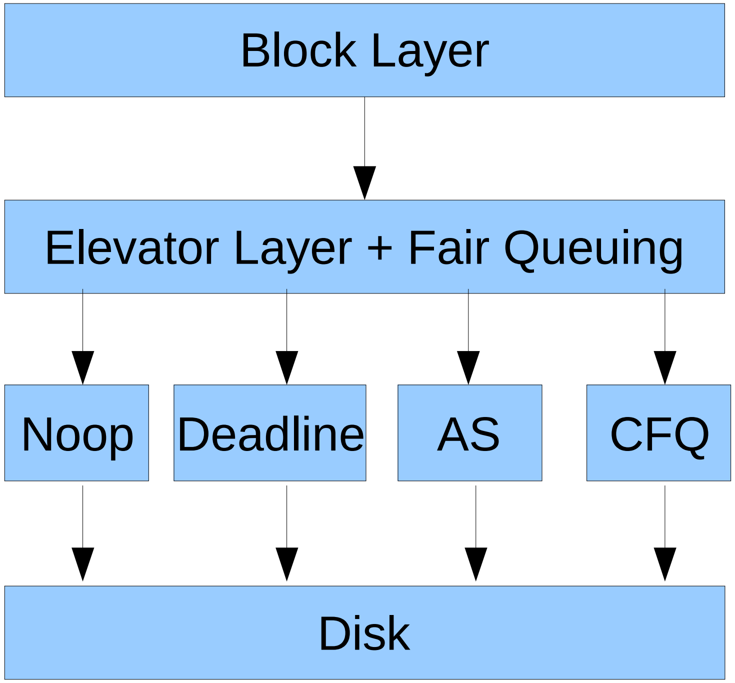
Idling Issues



T1
T2
T2
T1
Cgroup 1
Cgroup N
CFQ
T1
T2

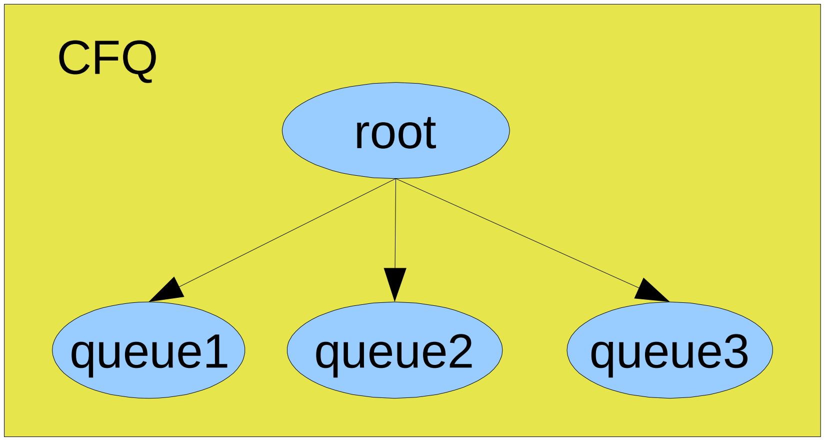
IO Scheduler based Control

- Proportional Weight Controller
- One Level Control at leaf nodes
- Common fair queuing elevator layer
- Extend to implement upper limit control later

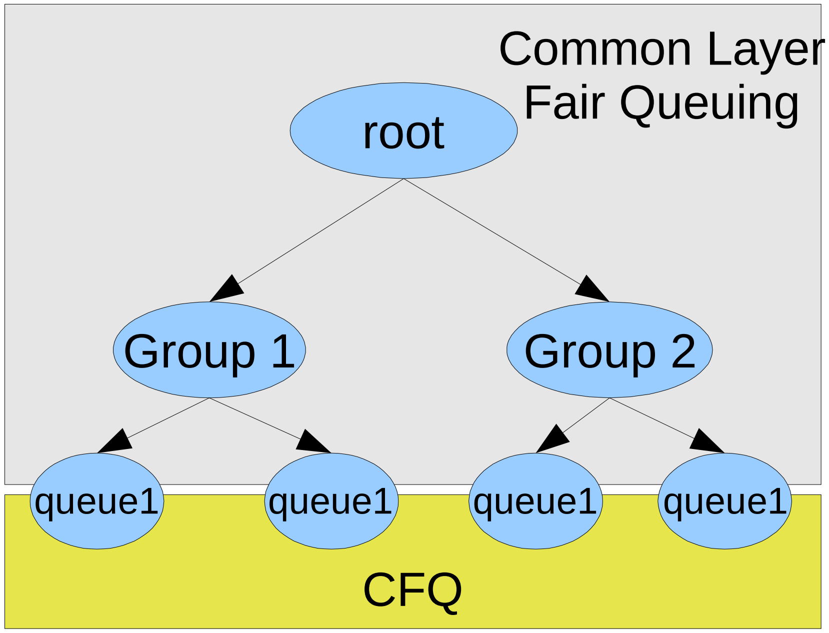


IO Scheduler based Control Contd..

CFQ
Weighted Round Robin



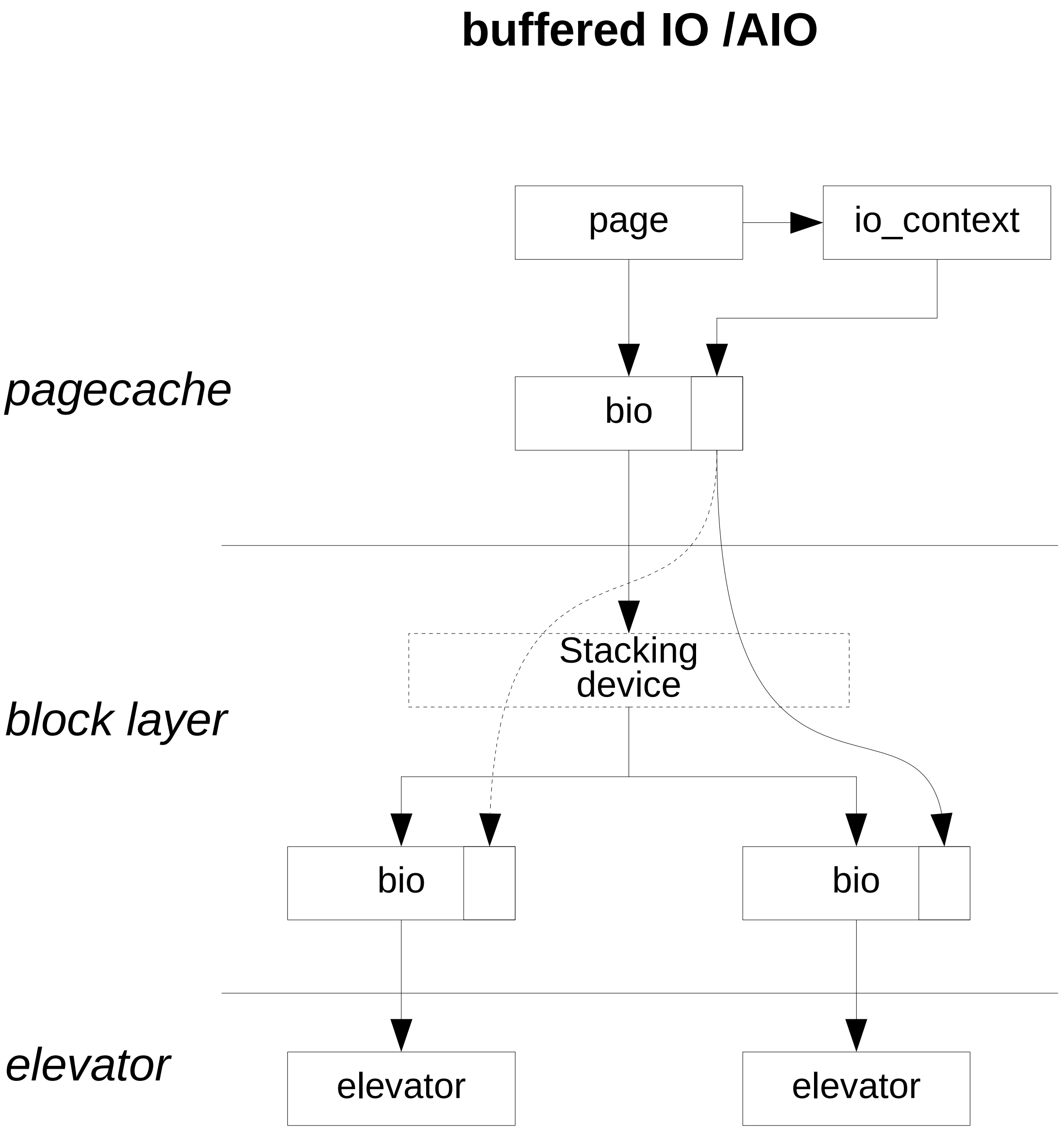
Hierarchical CFQ
B-WF2Q+

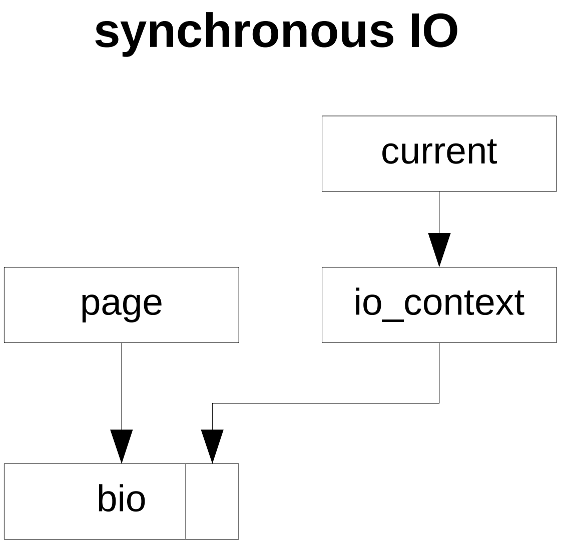


Other Proposals

- Elevator/IO Scheduler based Controllers
  - CFQ IO controller (Satoshi Uchida, NEC)
  - Another CFQ based IO Controller (Vasily, OpenVZ)
  - AS IO scheduler based control (Naveen Gupta, Google)
- dm-ioband (valinux)
  - Proportional weight controller
  - Two level IO scheduling
  - Device mapper based driver
  - Additional grouping mechanism other than cgroup
- IO Throttling (Andrea Righi)
  - Max bandwidth controller

1. I/O tracking: io context inheritance

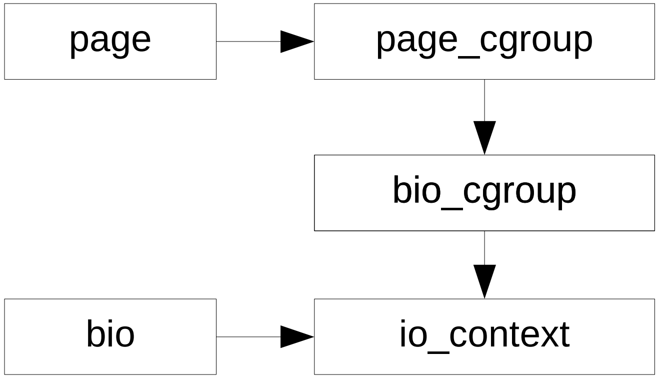




Combining page tracking and storing io context information in struct bio it is possible to track all bio-based I/O

2. I/O tracking: page tracking

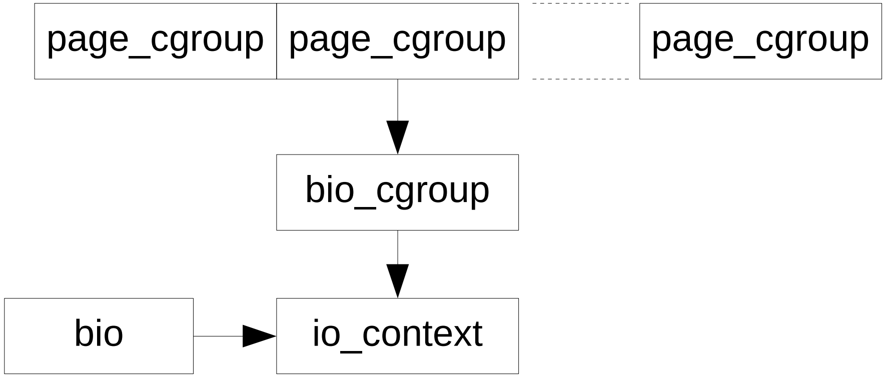
struct page approach (cgroups)



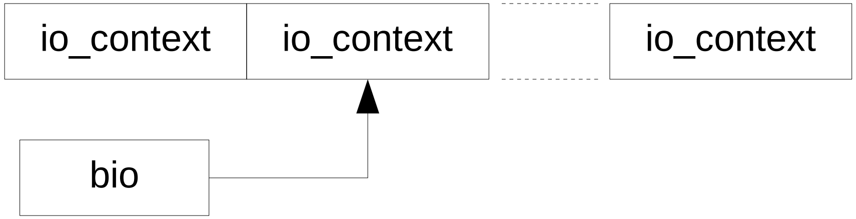
- To track the io context of io pages struct page could be extended, but using a an array of io_contexts à la mem_map

- The io tracking mechanism should not be cgroup-specific

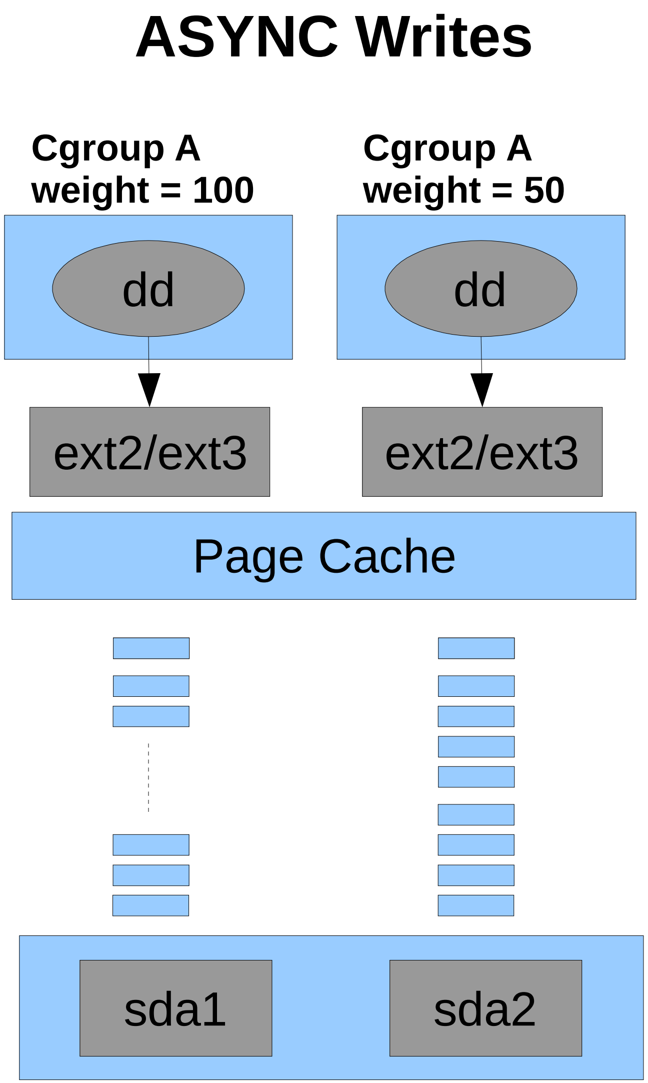
page_cgroup array (cgroups)

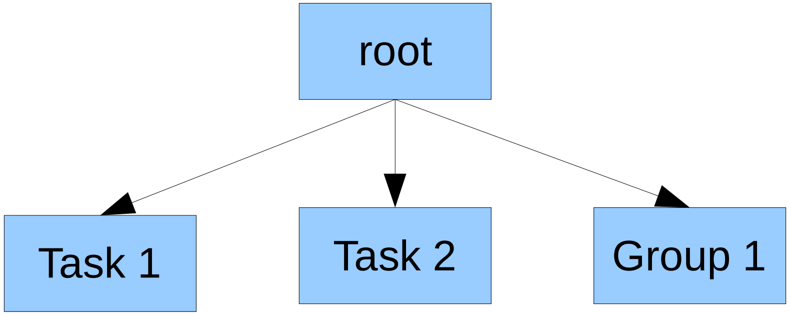


io_context array (no cgroups)



Miscellaneous





Treat task and group at same level (33% each)

50% BW to group 1 and 50% BW is shared by T1 and T2

That's it.

Backup Slides

Disadvantages of dm-ioband

- Two level control
  - Lots of duplication of code from cfq
  - FIFO release of bio from second level buffering
  - Tasks and group can't be treated at same level
- One ioband device for every block device
- Configuration complexity
- Additional Grouping logic
- No hierarchical support yet
- No concept of multiple classes

Disadvantages of IO-throttling

Max Bandwidth Controller only

Two level control. Will suffer from same issues as mentioned dm-ioband

No hierarchical support yet

Can't treat task and groups at same level

3. I/O throttling

Need to make sure that I/O accounting and control is performed in the context of the task that generated or will generate I/O (buffered I/O) (io-throttle's approach)

Trivial for synchronous reads

The controller kicks in each time a page is newly dirtied

Direct I/O controlled at the submit_page_section level