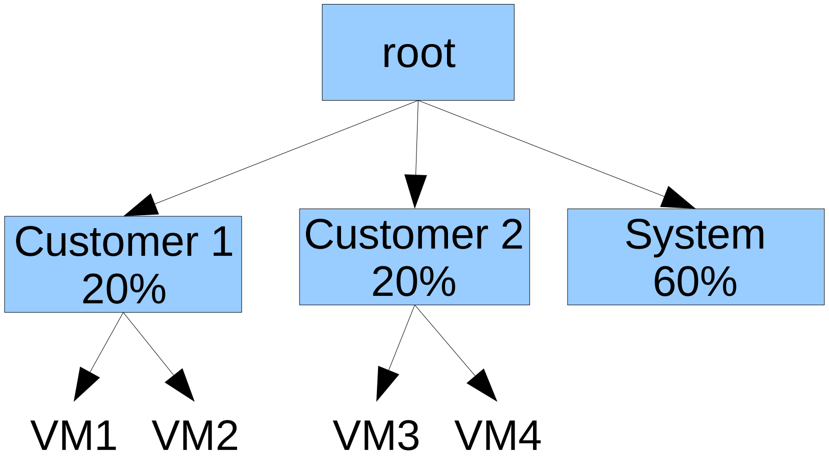
# How to design IO Controller in Linux?

Vivek Goyal

# Why IO Controller

**Enterprise Server**



root

Customer 1 20%

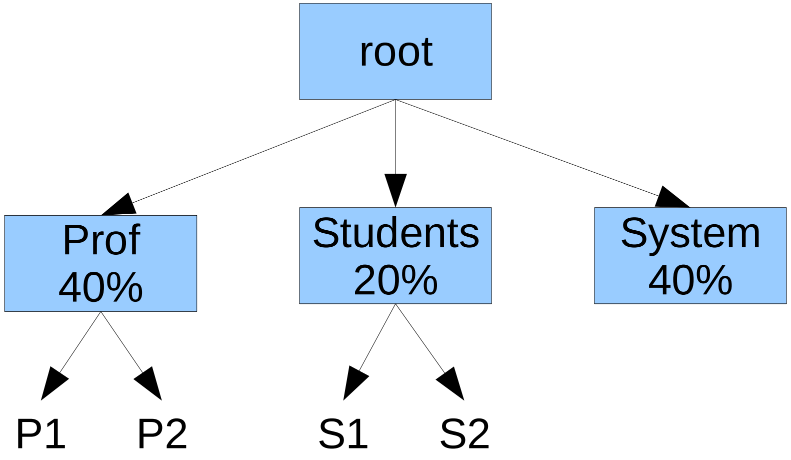
Customer 2 20%

System 60%

VM1 VM2

VM3 VM4

**University Server**



root

Prof 40%

Students 20%

System 40%

P1 P2

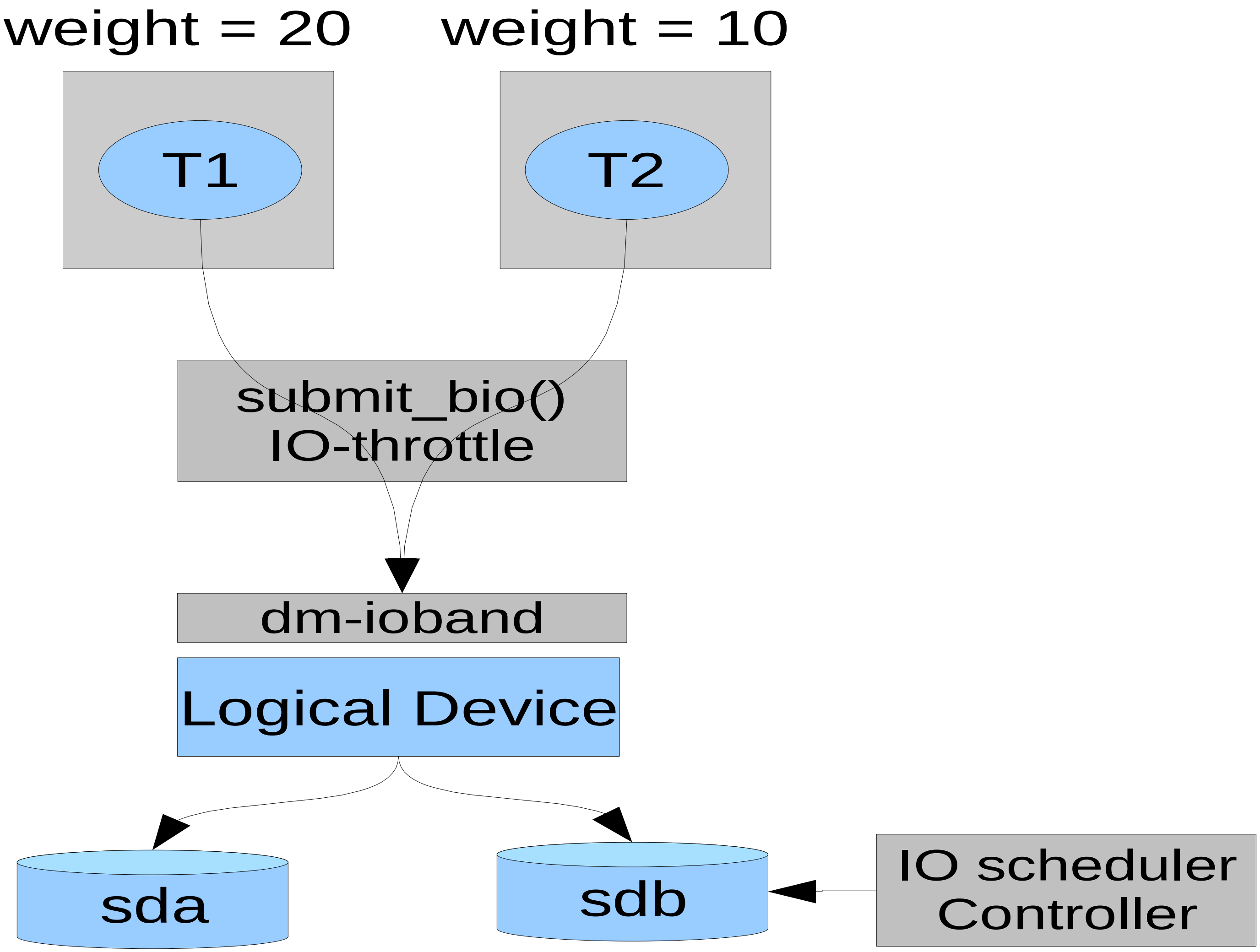
S1 S2

- More sharing needs more isolation
- Resource guarantees/Predictability
- Hierarchical group IO control

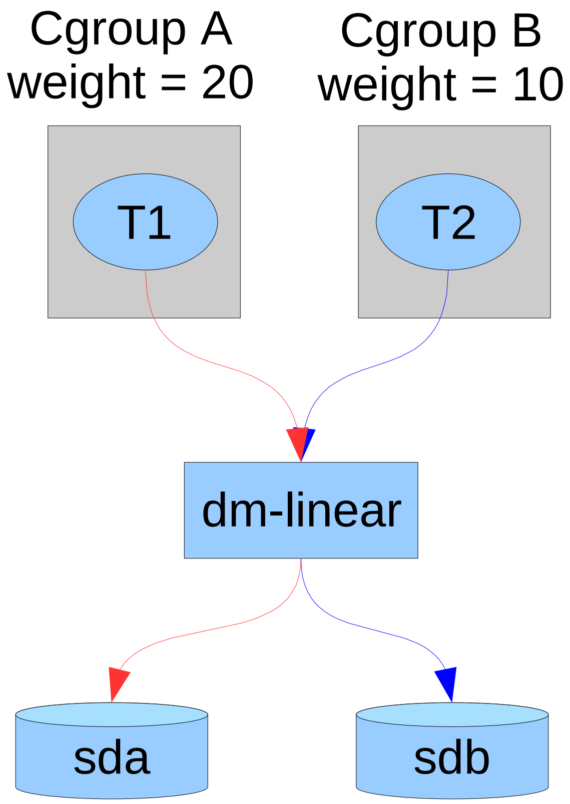
# What to Control

- Proportional Weight Controller
    - Optimal resource usage. Resource control done only if there is contention.
  - Fair share of disk time (Like CFQ)
  - Fair share in terms of number/size of IO
- Max Bandwidth/Max IOPS Control
  - 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?

# Where to control

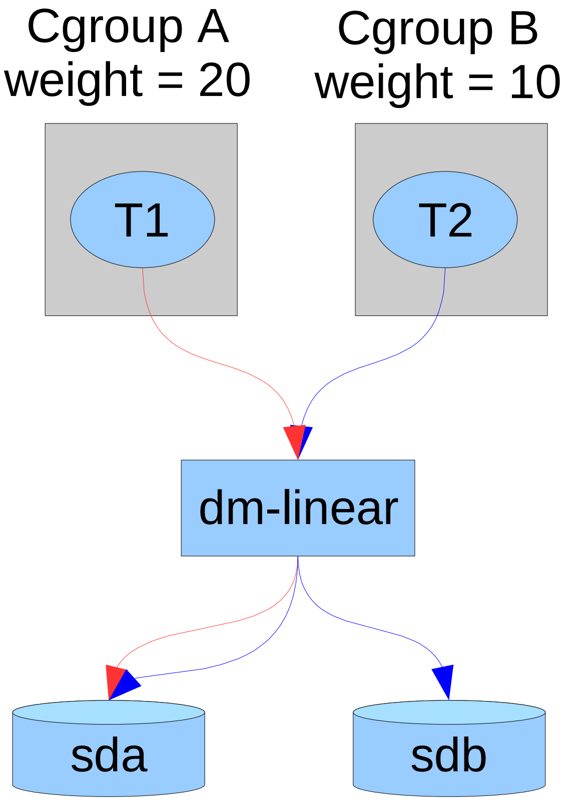


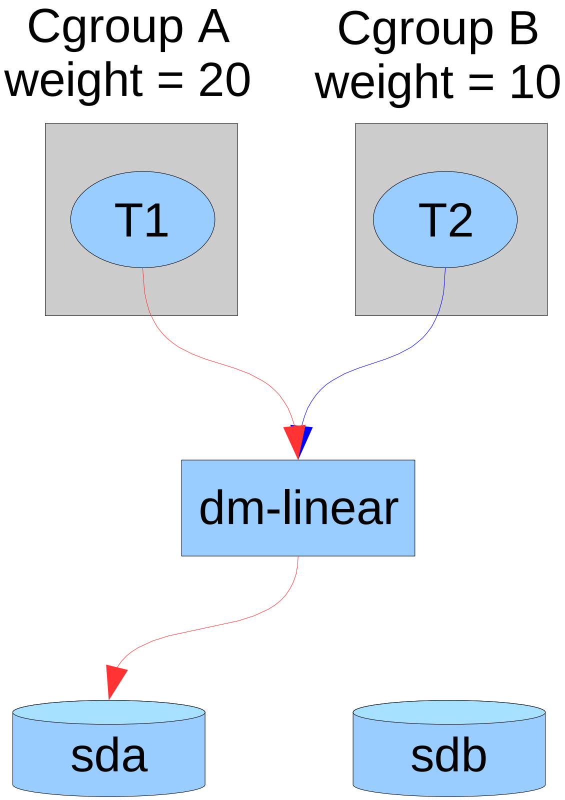
# Where to control Contd.



- T1 and T2 are seemingly contending for logical disk but no contention at physical level
- No contention at physical level

# Where to control Contd.



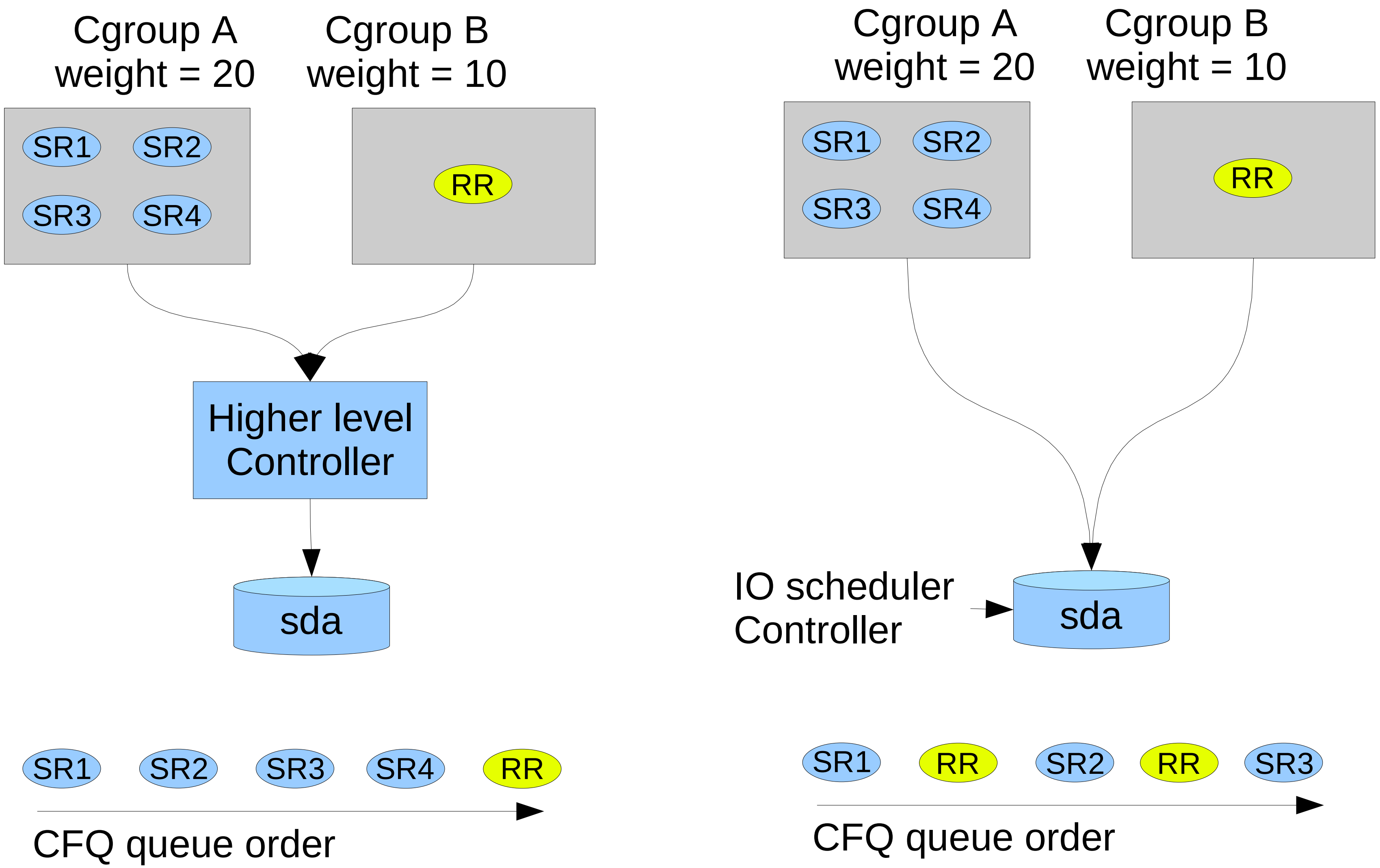


- T1 and T2 are sharing the disk sda
- One could have got faster response for T1 by throttling T2 at higher layer

# Where to Control Contd.

- Lower level control
  - Good for overall throughput
  - But application might not always see fair share at logical device level
- Higher level control
  - Good for fairness numbers at logical devices
  - Not an optimized scheme for throughput

# Latency issues with Second level Controller



# Challenges with Second level controller

- Number or size of IO not best for seeky media
- How to get timing information up there?
- Can't use token bucket kind of model and allow IO from multiple group at same time.
    - Latency issues
    - Pre-emptions across group; Poor isolation
    - No idling at higher layer means no fairness for readers; Idling means reduced throughput.
    - Possibly, Increased number of seeks due to throttling; Reduced throughput

# Challenges with second level controller

  - Can't exploit group locality feature; Interleaved IO across groups;

- Allowing IO from single group only reduces parallelism at higher level devices
  - Reduced throughput
- No per process queues at higher layer. How to maintain ioprio model.
- sync/async IO ratio with-in group
  - IO scheduler property

# Timed group fairness

- Not suitable for higher level logical devices
  - Introduces more serialization.
- Two ways to implement
  - Keep group and queues together
    - Current IO scheduler based controller implementation
  - Keep groups independent of queues

# Issues with separate group and queues

- Group scheduler will hold bios and release in FIFO manner.
  - Back to issue of ioprio with-in group
  - Issue of Reader/Writer ratio
- How to sync between Group slices and queue slices
- How to sync with AS read/write timed batches
  - Save state per group otherwise we will see sekwed read/write ratios with-in group

# Issues with separate group and queues

- What's the advantage of queuing at two levels?
  - Group level queue and IO scheduler level queuing
  - Group scheduler most likely will be queuing bio and can't take advantage of merging feature.

# Buffered Writes

- pdflush/flusher threads evens out the writeback flow
- Need per memory cgroup dirty ratio to differentiate in page cache share
- Also possibly need facility to writeback pages from a particular cgroup

# Summary Of Test Results

| TEST CASE | IOC | IOBAND | IOT |
|---|---|---|---|
| Mult Sequential Reader Vs Random Reader | 🟩 | 🟥 | 🟥 |
| Mult Random Writer Vs Random Reader | 🟩 | 🟥 | 🟩 |
| Mult Sequential Reader Vs Sequential Reader | 🟩 | 🟥 | 🟥 |
| Mult Buffered Writer Vs Buffered Writer | 🟥 | 🟨 | 🟩 |
| Multiple Random Reader Vs Sequential Reader | 🟩 | 🟥 | 🟨 |

| TEST CASE | IOC | IOBAND | IOT |
|---|---|---|---|
| Mult Sequential Reader Vs Mult Random Reader | 🟩 | Throughput? | No results |
| Mult Sequential Reader Vs Mult Sequential Reader | 🟩 | Latency? | No results |

# Advantages of dm-ioband

- IO control can be enabled both at lower level devices as well as higher level devices
- Provides multiple control policies
  - Number of IO
  - Size of IO
  - Max BW

# Issues with dm-ioband

- Fairness in terms of number of IO/size of IO does not do very well on seeky media
- Weak isolation between groups
- Poor latencies
- No fairness for low volume IO group
- Changes the properties of underlying IO scheduler
  - Reader Vs Writer ratio
- IO priority with-in group is not maintained
  - Some tasks in the group can starve

# IO Throttling

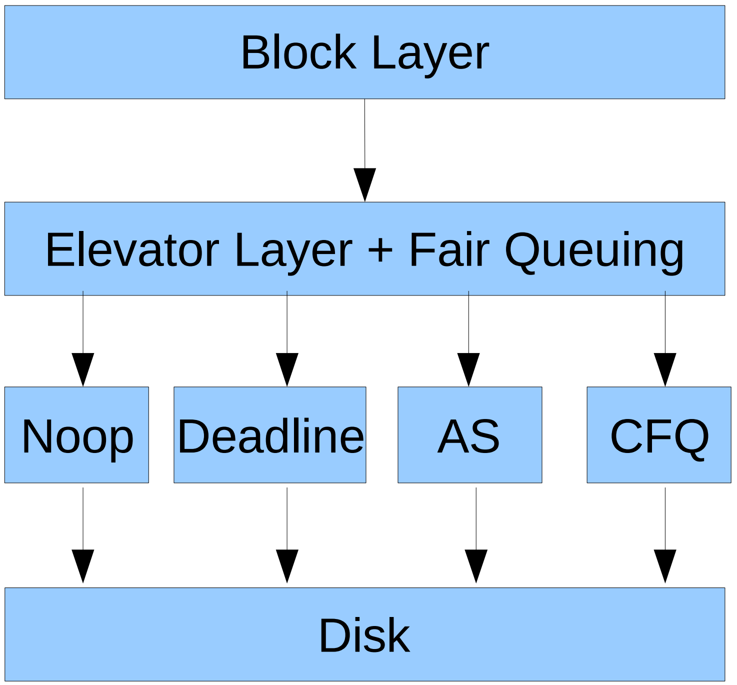
- Higher level controller, can be used for both physical and logical devices
- Provides max BW policy at higher layer

# Issues with IO throttling

- Max BW policies have got limited usage and are not very suitable for dynamic workload environment.
- Inherits all the issues of higher level controller mentioned in previous slides
  - Weak isolation
  - No strong control on latencies
  - Preemptions across groups
  - Looses notion of ioprio and class with-in group

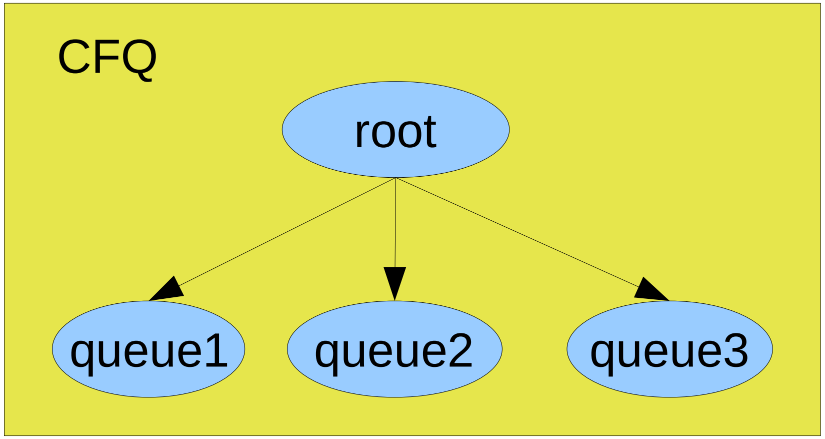
# IO Scheduler based Control

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

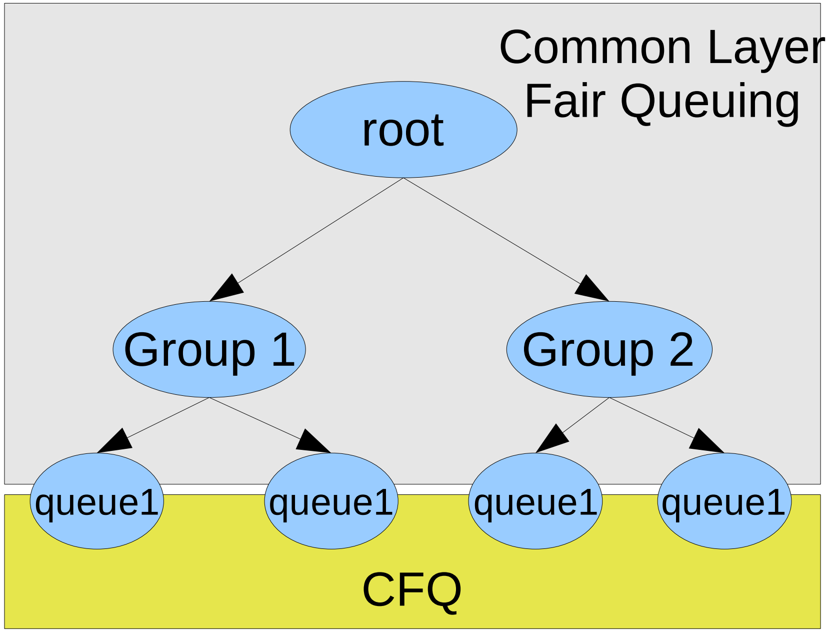


# IO Scheduler based Control Contd..

**CFQ**
**Weighted Round Robin**



**Hierarchical CFQ**



# One possible way to move forward...

- Implement more than one controller in kernel
  - One CFQ level for more efficient and optimal control
    - Implement time based fairness policy
  - One higher level for
    - Control on logical devices
    - size/number of IO policies
    - Max BW policies
- Let user choose based on the need.

That's it.