Toward A Unified Block IO Controller

Shaohua Li <shli@fb.com>

Software Engineer, Facebook

Agenda

- Overview
- Unified IO Controller and Challenges
- The Solution
- Benchmark Data
- TODO

Overview

IO Controller

- Share IO resources between tasks
- Maintain fairness with specific policy
- 2 policies
  - CFQ
  - Block-throttle

CFQ IO Controller

- Based on CFQ ioscheduler
- Proportion based
- Time slice/IOPS accounting
- Fair*
- Performance issues
  - Not scale
  - Idle disk for fairness

Block-throttle

- Throttle cgroup to upper limit
- Bandwidth/IOPS based
- No proportional scheduling
  - User sets upper limit

Blk-mq Challenges

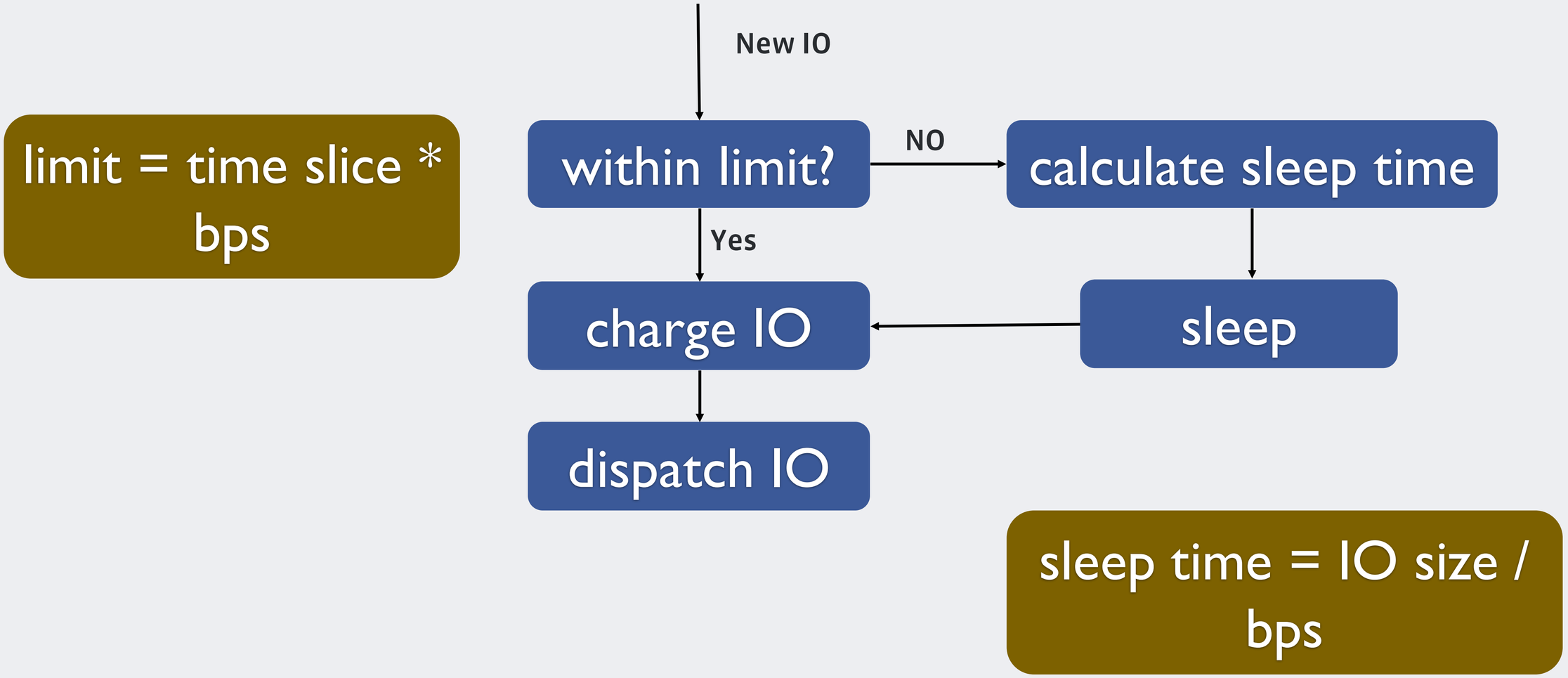
- Multiple queues
- Target devices have high queue depth
- Scalable design
- No IO scheduler

Unified IO Controller and Challenges

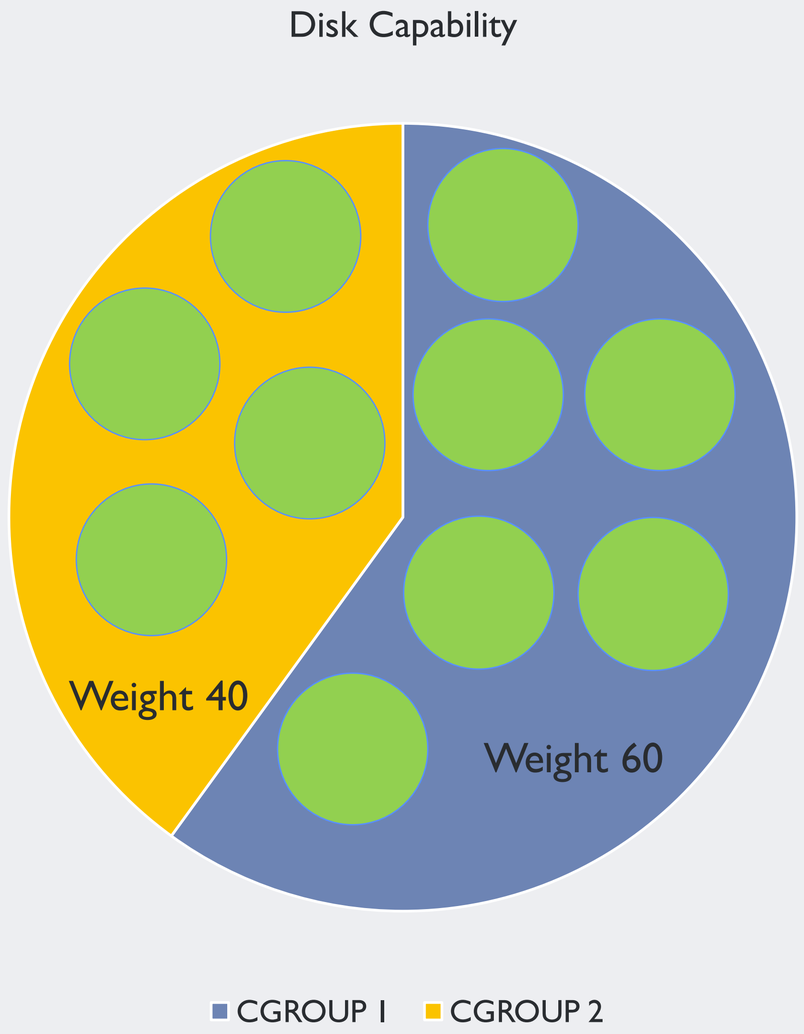
Unified IO Controller

- Has both proportion and upper limit policy
- Work for blk-mq
- Scalable
- Block-throttle is a good candidate
  - Work for blk-mq
  - Has global lock but not too bad
  - Potentially per-cpu cache for better scalability
  - Must add proportional policy

Block-throttle Workflow

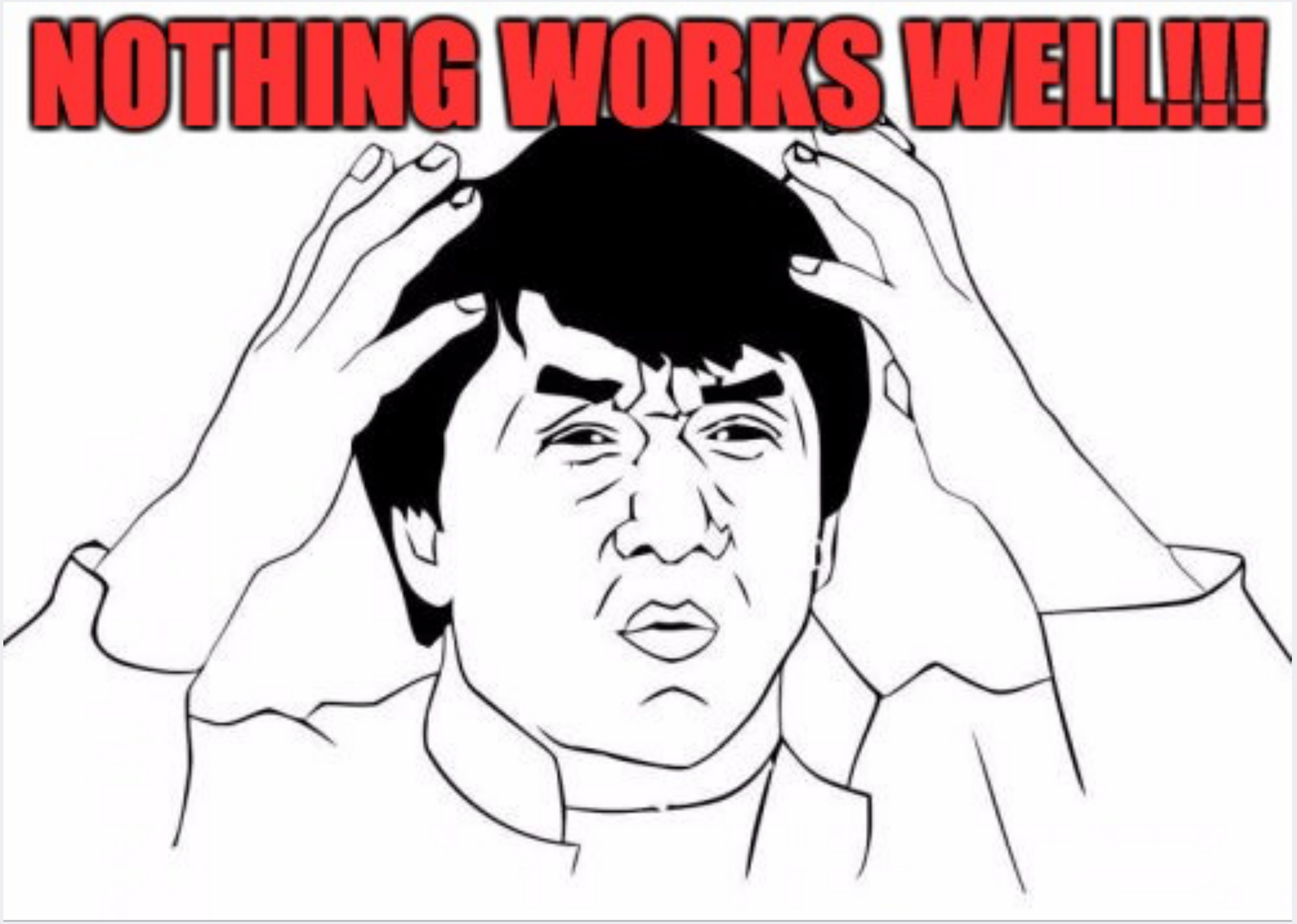


A Magic Disk



The Cruel World

- Disk capability isn't fixed
  - IO size
  - read/write ratio
  - IO depth
  - Queue depth
  - Sequential/random
- Measure IO cost
  - IO size
  - IOPS
  - IO time
  - Combine them?


NOTHING WORKS WELL!!!

The Solution

Suboptimal Solution

- Use bandwidth or IOPS
  - Capability is total bandwidth or IOPS
  - IO cost is IO size or 1 per request
  - User choose
- Adaptive
  - No fixed total bandwidth or IOPS
  - Feedback system, try and push to steady state

Disk Bandwidth

- Estimate bandwidth
  - Bandwidth = current bandwidth / disk utilization
- Disk could be underutilized even with 100% utilization
  - Throttled tasks can’t dispatch more IO
- Always slightly over estimate
  - Over estimate bandwidth
  - Workload gets bigger limit
  - Workload sends more IO
  - Estimate higher bandwidth
  - Reach steady state in max bandwidth

Cgroup Throttle

- cgroup share = weight / total weight
- cgroup bandwidth limit = cgroup share * estimated disk bandwidth
- Using existing block-throttle mechanism to throttle

Inactive Cgroup - Example

- Disk bandwidth is 100M/s
- 2 cgroups with 50% share, each gets 50M/s
- Cgroup1 is idle, cgroup2 is 50M/s
- Estimated bandwidth is 50M/s
- Each gets 25M/s
- cgroup1 is idle, cgroup2 is 25M/s
- Estimated bandwidth is 25M/s
- ...

Inactive Cgroup - Solution

- Dynamically adjust cgroup share
- Feedback system
  - Gradually decrease share if underutilized
  - Recovery to original share if acting limit hits
  - Defer share decrease if recovery happens recently
  - Eventually cgroup acting limit = bandwidth

Benchmark Data

Benchmark

- NVMe disk
- Fio job with 1 iodepth, 4k IO and 8 threads
  - Randread ~330M/s
  - Randwrite ~1.4GB/s
- Emulate inactive cgroup with fio ‘–rate=2M/s’

Benchmark Result

- Cgroup1 weight 200, cgorup2 weight 800; randomwrite
  - Cgroup1: 322042KB/s, cgroup2: 1020.4MB/s
- Cgroup1 weight 200, cgroup2 weight 800 with rate limit 2M/s; randomwrite
  - Cgroup1: 1367.3MB/s, cgroup2: 2047KB/s

Benchmark Result

- Cgroup1 weight 200, cgroup2 weight 800; randomread
  - Cgroup1: 296690KB/s, cgroup2: 308275KB/s
- Cgroup1 weight 200, cgorup2 weight 800 with rate limit 2M/s; randomread
  - Cgroup1: 338551KB/s, cgroup2: 2040KB/s

Benchmark Result

- Cgroup1 weight 200 randomwrite, cgorup2 weight 800 randomread
  - Cgroup1: 168443KB/s, cgroup2: 287954KB/s
- Cgroup1 weight 200 randomread, cgorup2 weight 800 randomwrite
  - Cgroup1: 79038KB/s, cgroup2: 1127.8MB/s

TODO

- More tuning
- Separate weight for read and write
- Throttle write

Thank You!

facebook