New Cgroup Subsystem for Buffer Write IO and Network RX Control in Kernel

- DongdongChen from Tencent

- open source summit 2019

Agenda:

1、kernel cgroup for buffer write io limit

2、kernel cgroup for net_rx limit

Background:

Kubernetes manage resources on the node with the way of kernel cgroup subsystem, which is very import. The target is to guarantee pod's request resource firstly, but also the pod can borrow from other pods when there are free resources temporarily(calling it soft limit), with the mind of not influencing node's stability. Resources such as:

- **Cpu: cpu cgroup, cpuacct cgroup, cpuset cgroup**
- **Memory: memory cgroup**
- **Blkio: blkio cgroup，not working for buffer write io**
- **Network: tc or iptables, net_cls cgroup controlling net_tx，no cgroup for net_rx**
- **Disk: disk_quota**

Try to manage all resources in both hard and soft limit, we need two extra cgroup subsystems to control buffer write and net_rx in kernel.

1.1 Buffer write limit - Backgroud

Target:

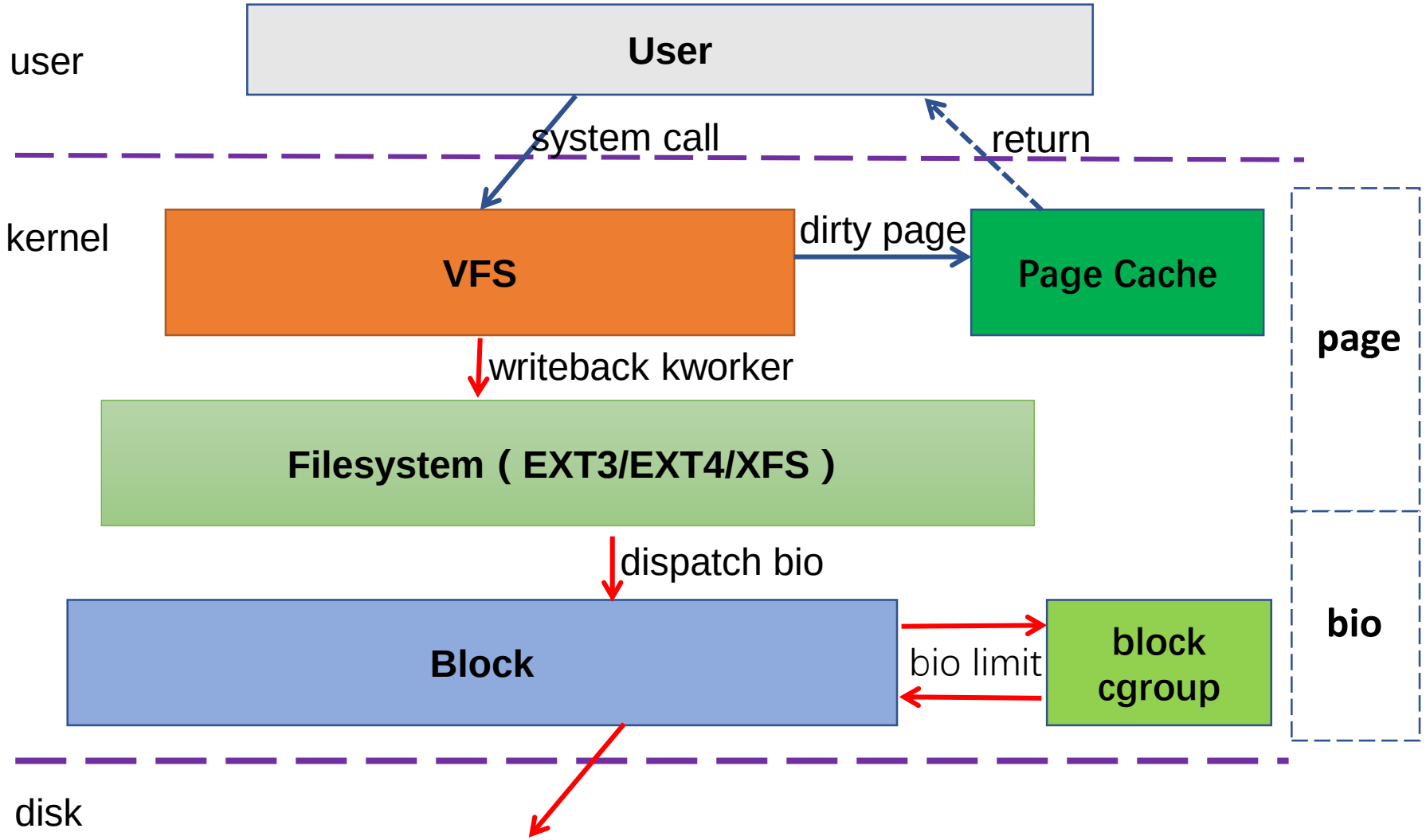
Disk io could be limited, including buffer write. Also support hard and soft limit.

Reason:

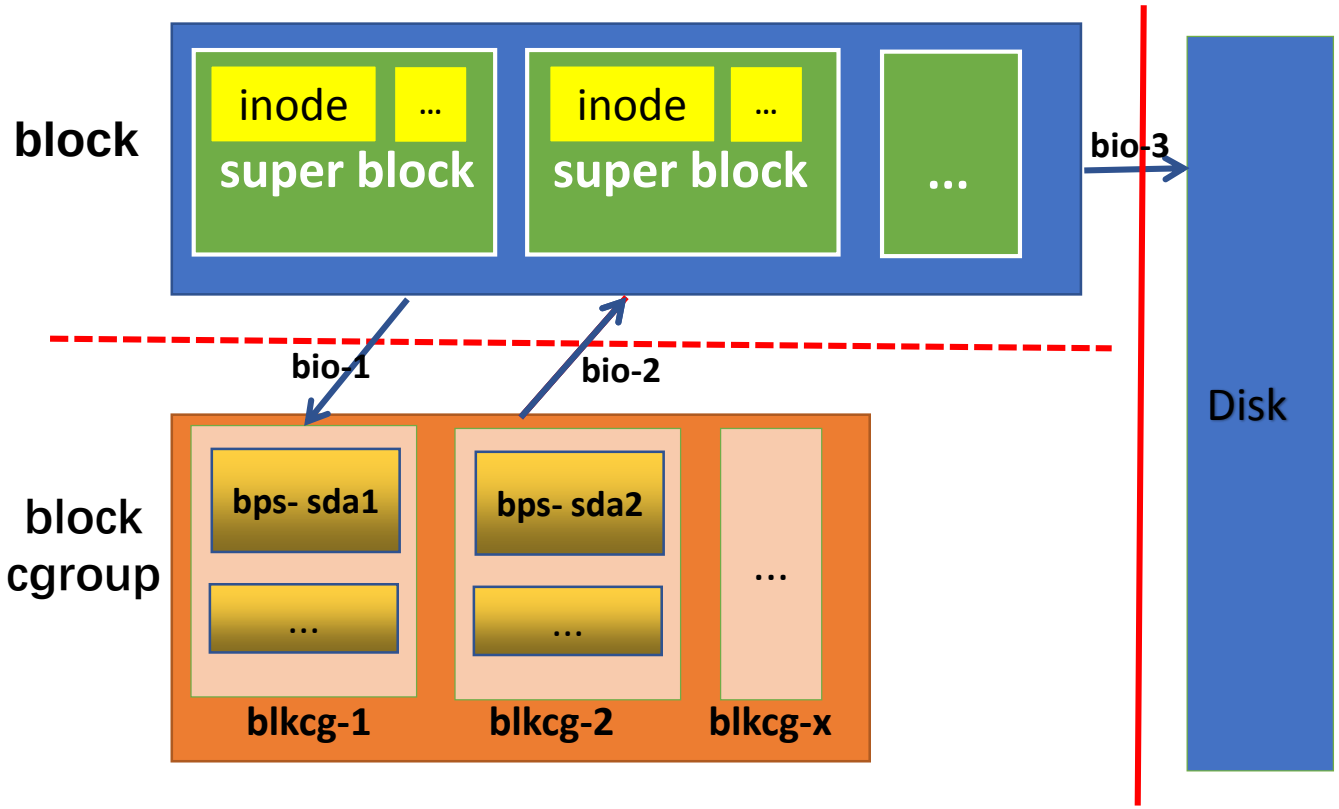
The current blkio cgroup just support direct io, but no buffer io. Because the blkio cgroup can not find the original process of the io, it is kworker in kernel that sync io to disk.

Buffer io limit has been supported by cgroup v2, while the struct has changed to unified hierarchy, and must use higher kernel version(4.5)

How Buffer write work



1.2 Assign block cgroup info to inode



inode： file related

super block： metadata

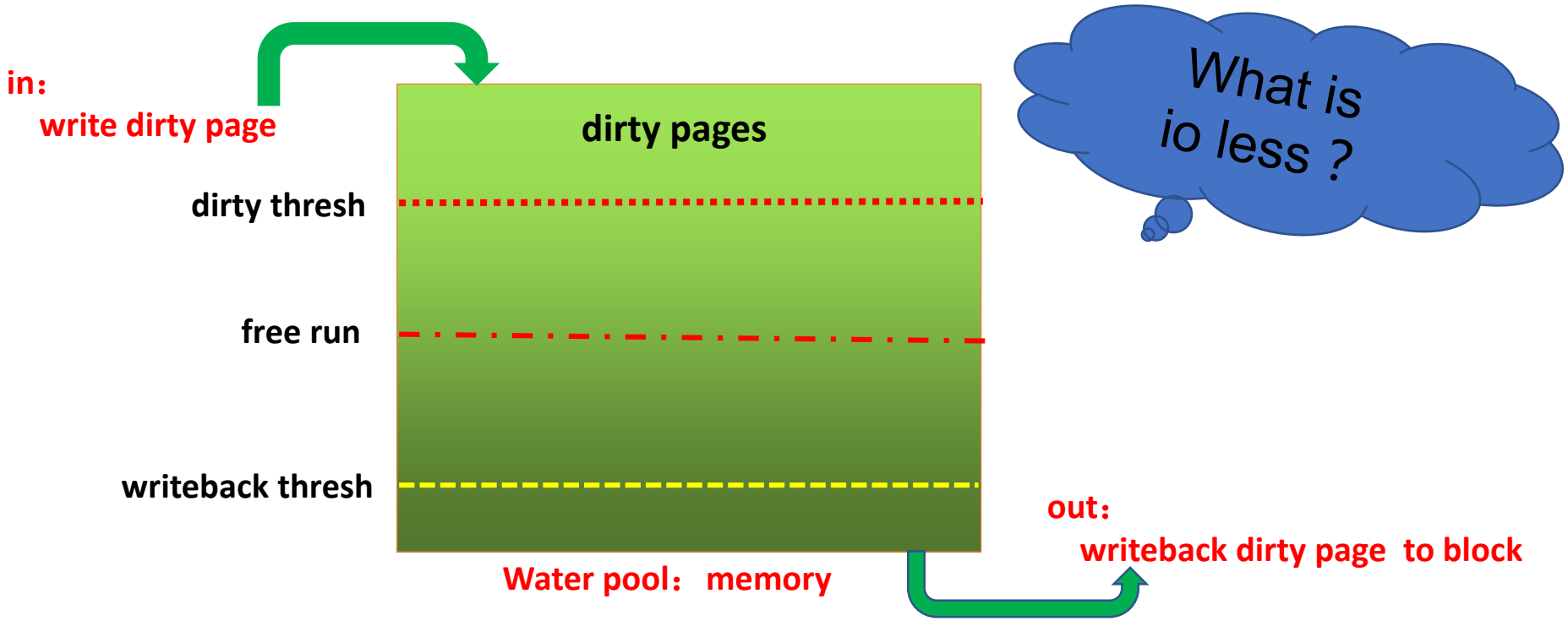
block cgroup： cgroup subsystem (/sys/fs/cgroup/blkio/)

blkcg-x： block cgroup name

bps-sdax： limitation for sdax in one block cgroup

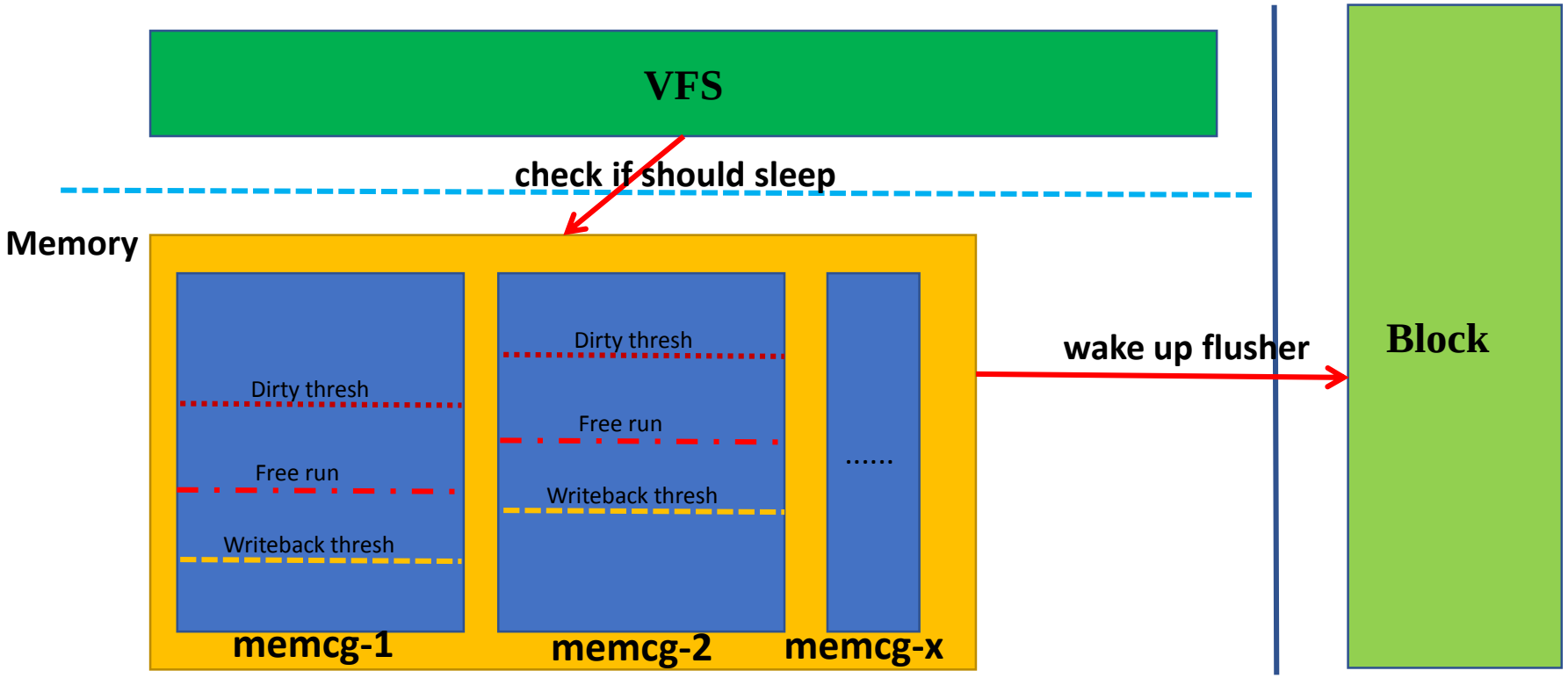
- **add block cgroup pointer in inode struct**
- **assign block cgroup info to inode when writing dirty pages**

1.3 Apply io less mechanism to memory cgroup



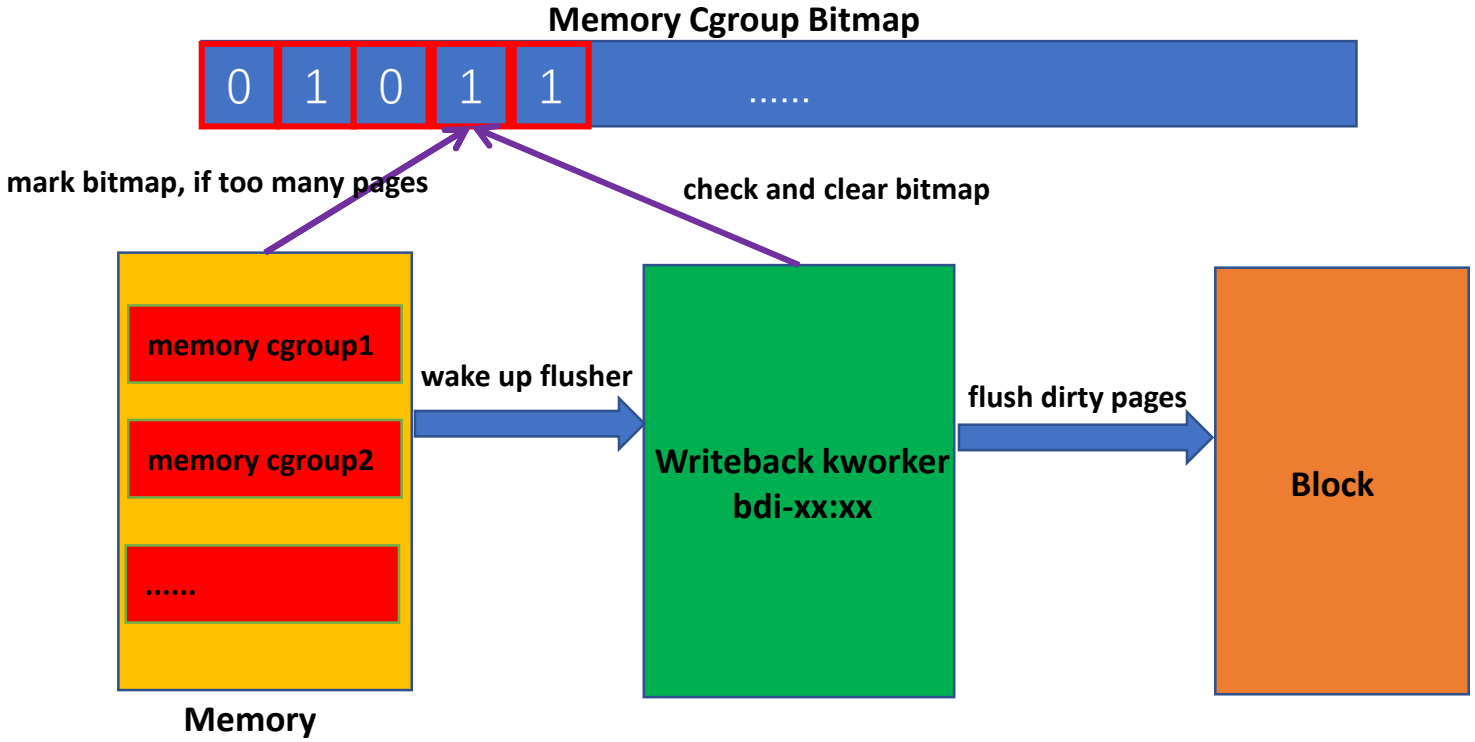
- **dirty pages**： pages with dirty and writeback flag. Pages that has not been synced to disk is marked dirty, if the page is syncing to disk, then marked writeback.
- **writeback thresh**： /proc/sys/vm/dirty_writeback_ratio(_bytes)
- **dirty thresh**： /proc/sys/vm/dirty_ratio(_bytes)
- **free run**: (writeback thresh + dirty thresh) / 2

1.4 Apply io less mechanism to memory cgroup



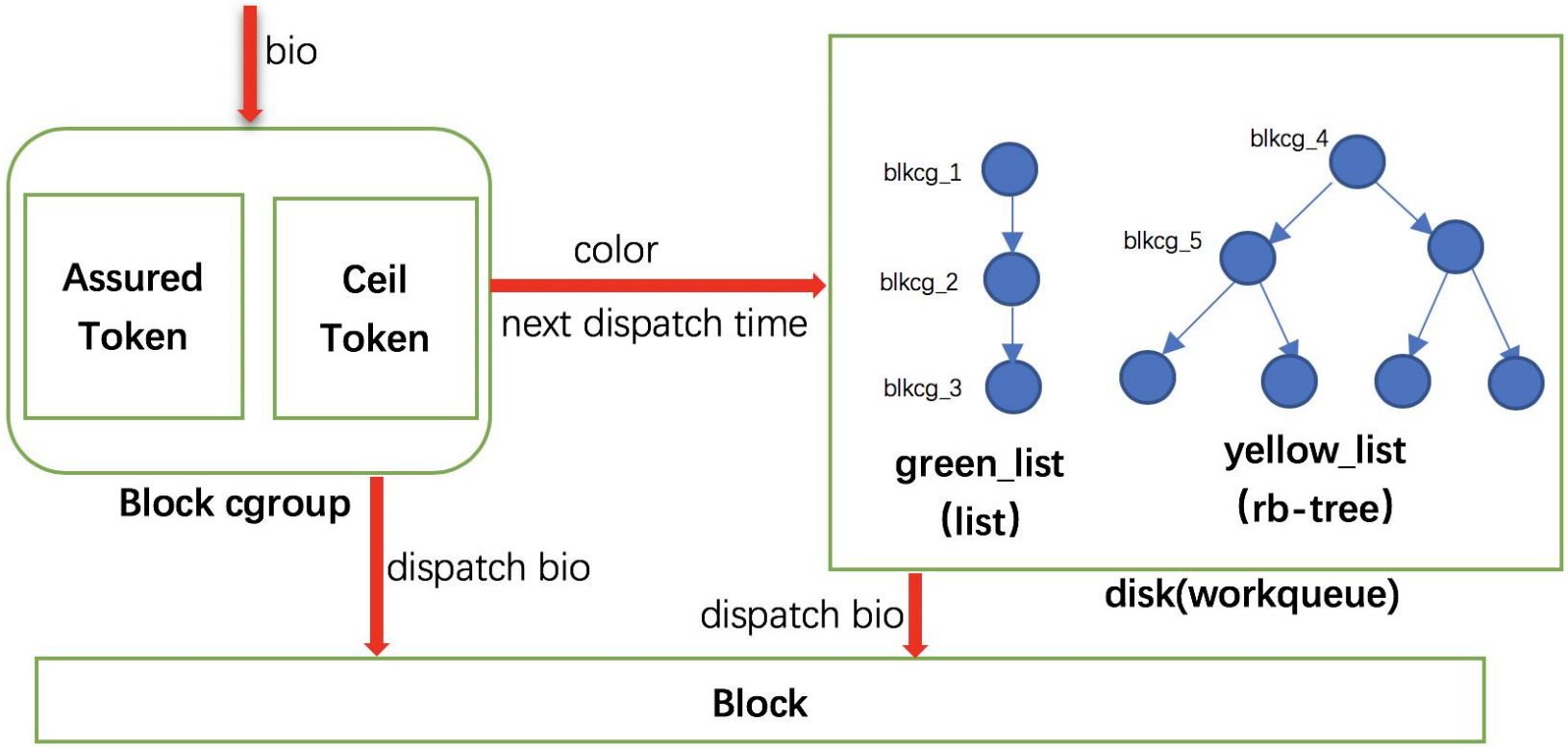
- memory cgroup should know how many pages with dirty and writeback flag in charge
- memory cgroup has its own dirty and writeback thresh level
- apply io less mechanism memory cgroup

1.5 Writeback kworker should know memory cgroup



- use global bitmap to mark if the memory cgroup should sync dirty pages
- mark bitmap to 1 when memory cgroup has too many dirty pages
- writeback kworker check bitmap and clear bitmap to 0 when syncing io

1.6 Buffer write soft limit



(1) every block cgroup has two token buckets: assured and ceil

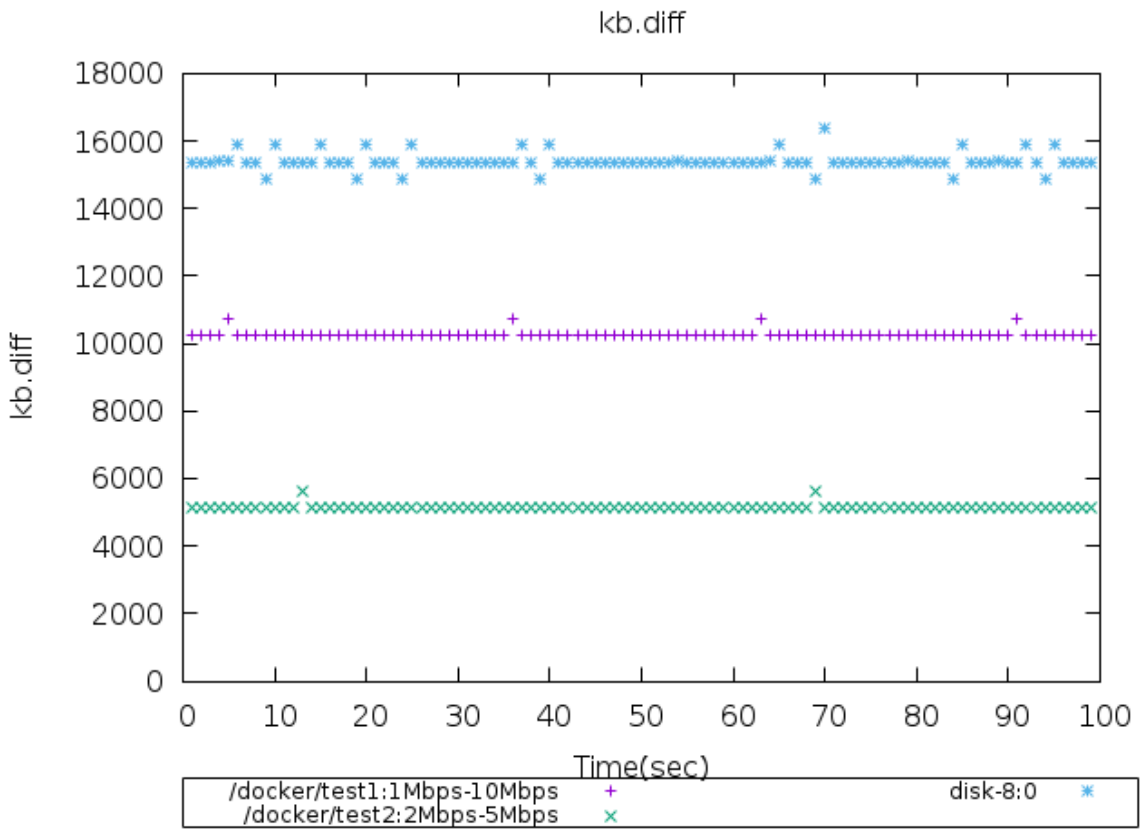
(2) every block cgroup has the color attribute, which Green indicates both the two buckets have token, and Yellow indicates the assured bucket has no token, while the ceil bucket still has token, the Red indicates none of the two buckets have token

(3) every block cgroup has an attribute of timestamp to dispatch bio for next time according bps limitation

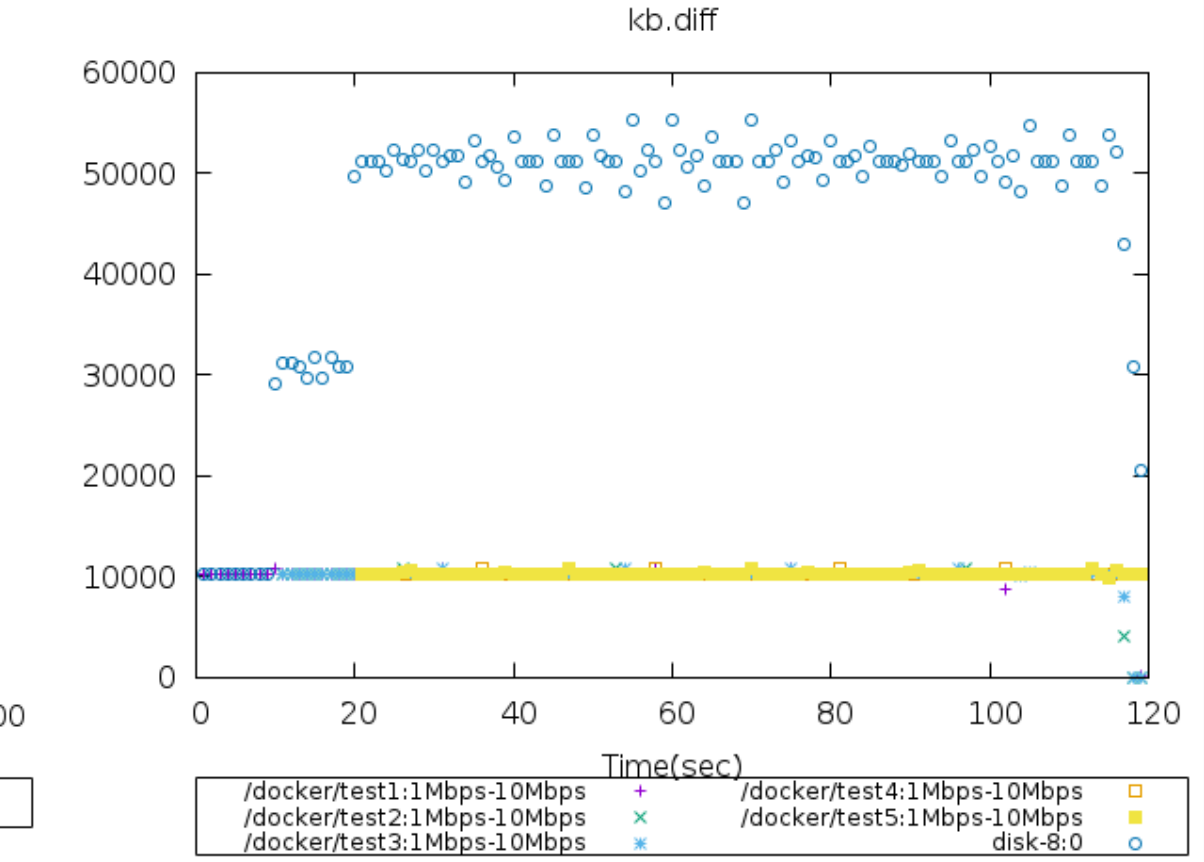
(4) all block cgroup with green color related to the same disk are organized by list, and all block cgroup with yellow color related to the same disk are organized by red-black tree

(5) when dispatching bio, all block cgroup with green color should be checked, while for yellow list, block cgroup can borrow more resources with higher priority by the way of ‘Deficit Round Robin’

1.7 Buffer write limit - data

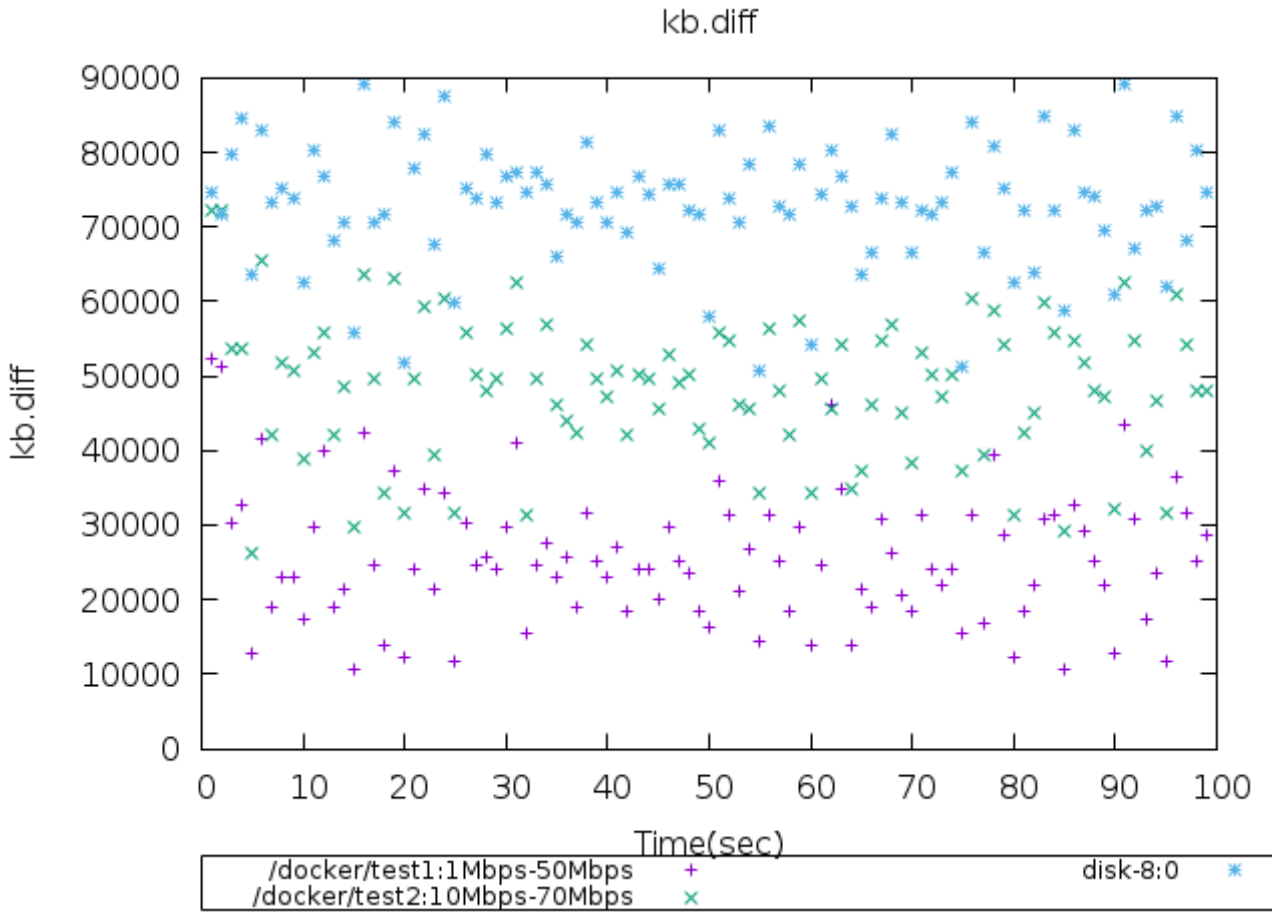


5Mbps vs10Mbps



concurrency with different start time

1.8 Buffer write limit - data



assured=1Mbps, ceil=50Mbps vs
assured=10Mbps, ceil=70Mbps

Block cgroup with higher assured bps limitation(higher priority) could borrow more resources

2.1 Net_rx limit - background

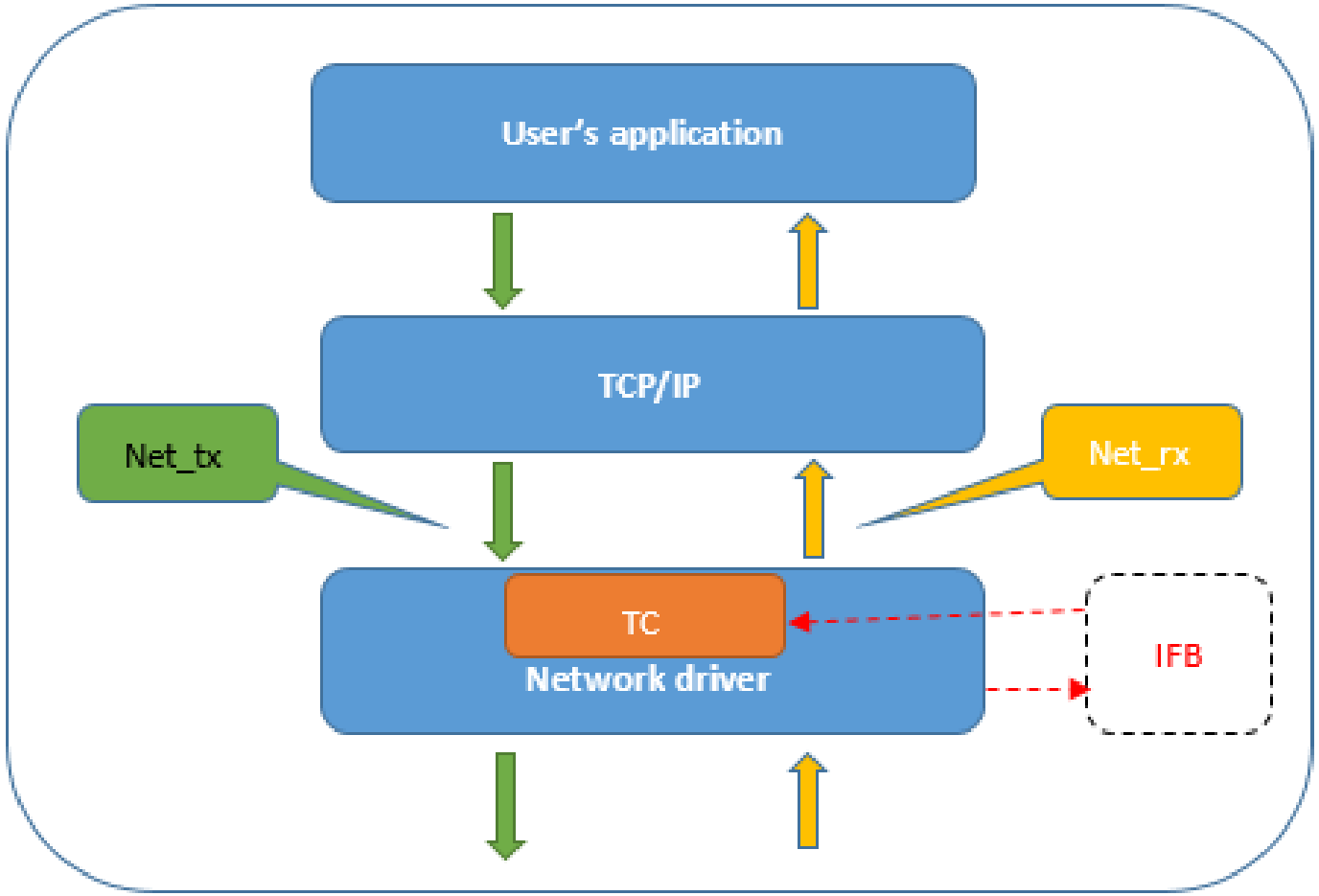
Target:

Net_rx could be limited by cgroup with less package dropped, also should support hard and soft limit

Current net_rx control:

- tc egress matching dst ip
- tc policing(ingress)
- IFB(Intermediate Functional Block device), mirroring net_rx to net_tx

2.1 Net_rx limit - background



(1) use iptables or net_cls to tag network package

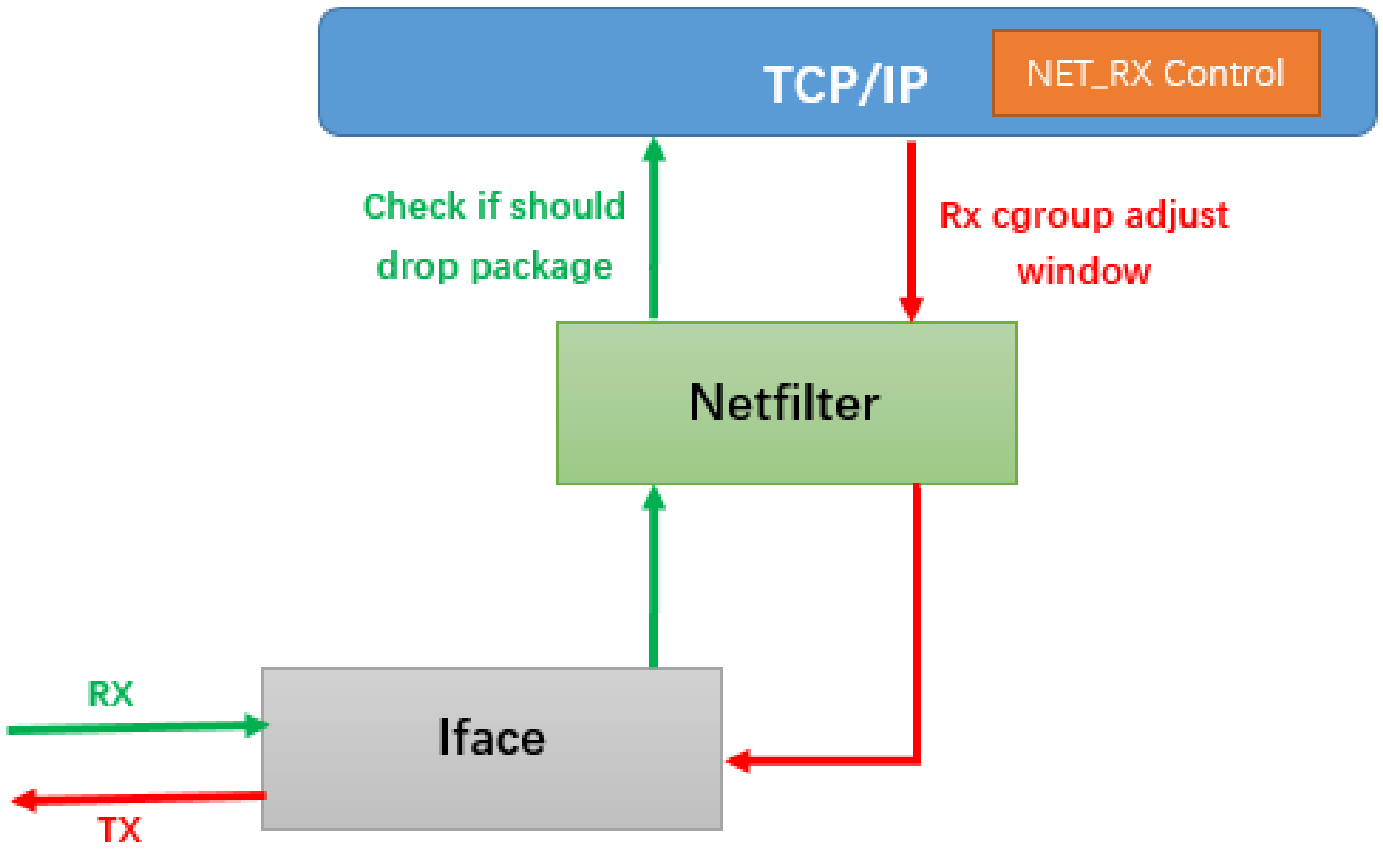
(2) use tc command to limit network bandwidth

(3) for net_rx, there is no specific process info when passing to TC

(4) IFB, mirroring net_rx to net_tx, may consume more cpu

(5) drop the package when net_rx has been limited, increasing network congestion

2.2 Net_rx limit



Net_rx control:

(1) add net_rx control during receiving socket buffer, by the way of token bucket

(2) check if should drop network package according token number

(3) adjust window size before transmitting socket buffer, notify the other end to send slowly

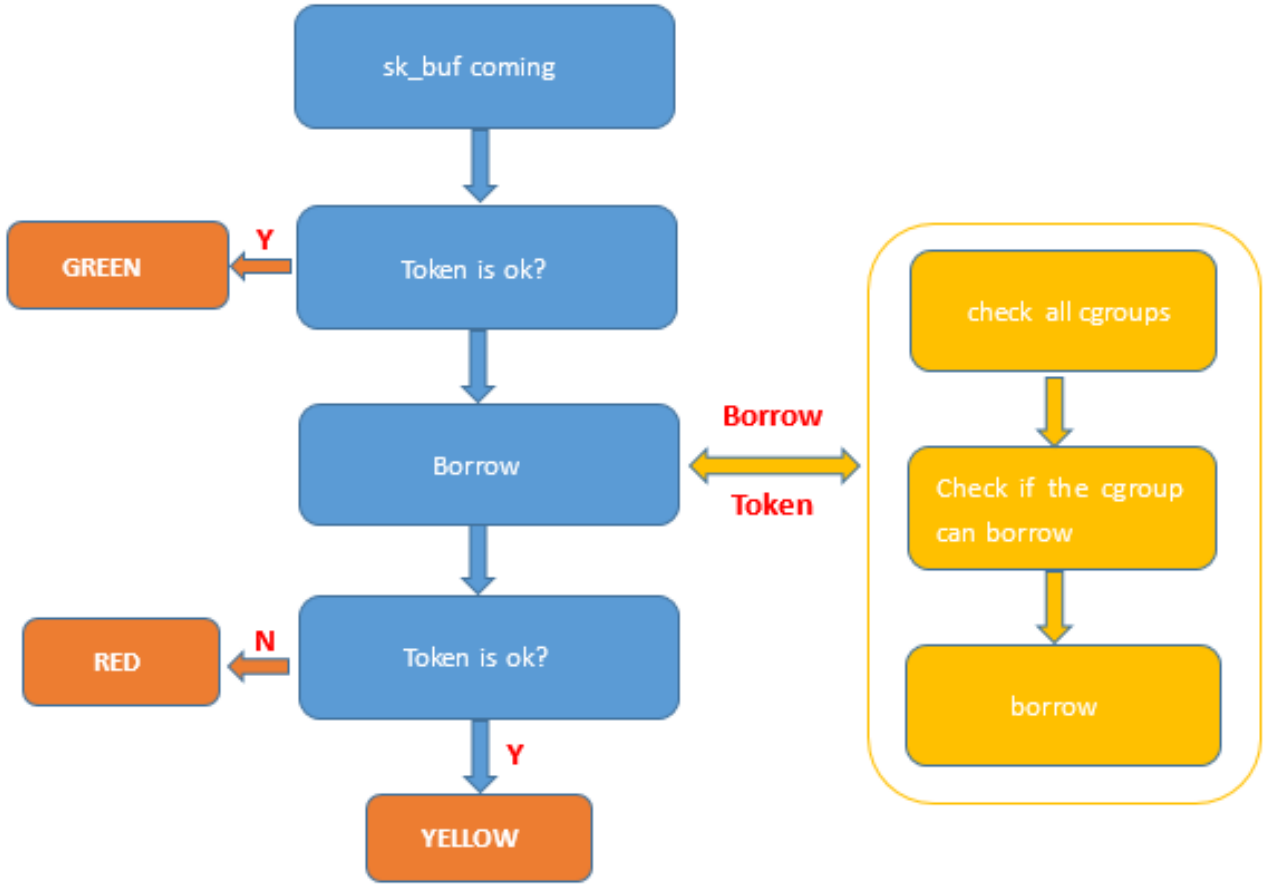
(3) you can assign processes into different cgroup to control net_rx, not ip:port

(4) the requested net_rx resource will be assured

(5) the net_rx cgroup can borrow when other cgroups have free resources

(6) when more cgroups share free resources, the borrowed resource will be divided according priority

2.2 Net_rx limit - soft

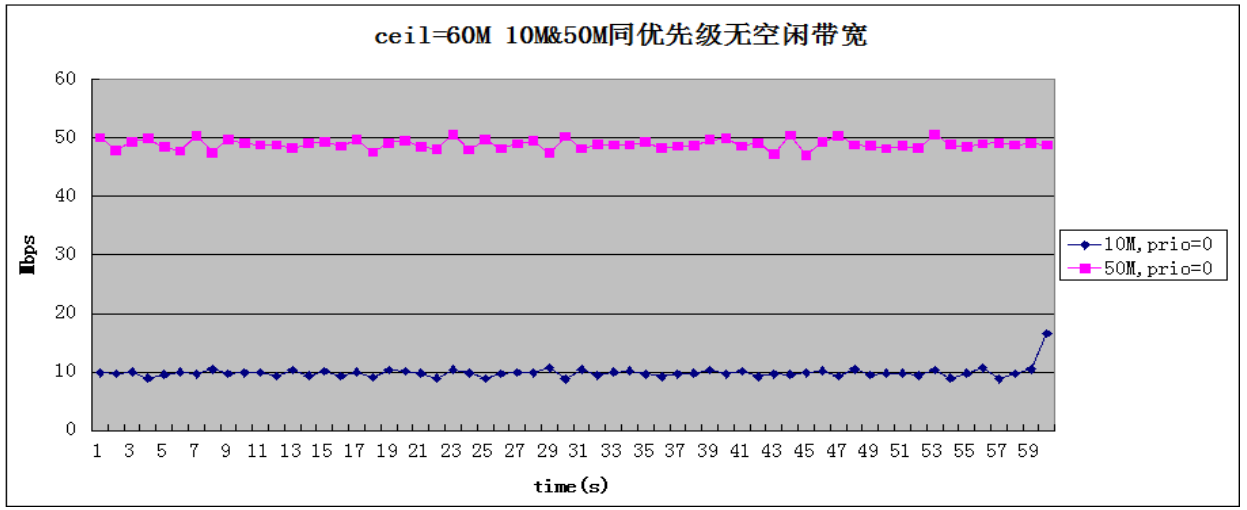


(1) there is a root cgroup assigning the total limitation value, the summary of other sub cgroups should not exceed the value

(2) all cgroup has its own token bucket. The root cgroup's token is the remaining resources without all sub cgroups

(3) When borrowing, the cgroup with higher priority could share more resources

2.3 Net_rx limit - data

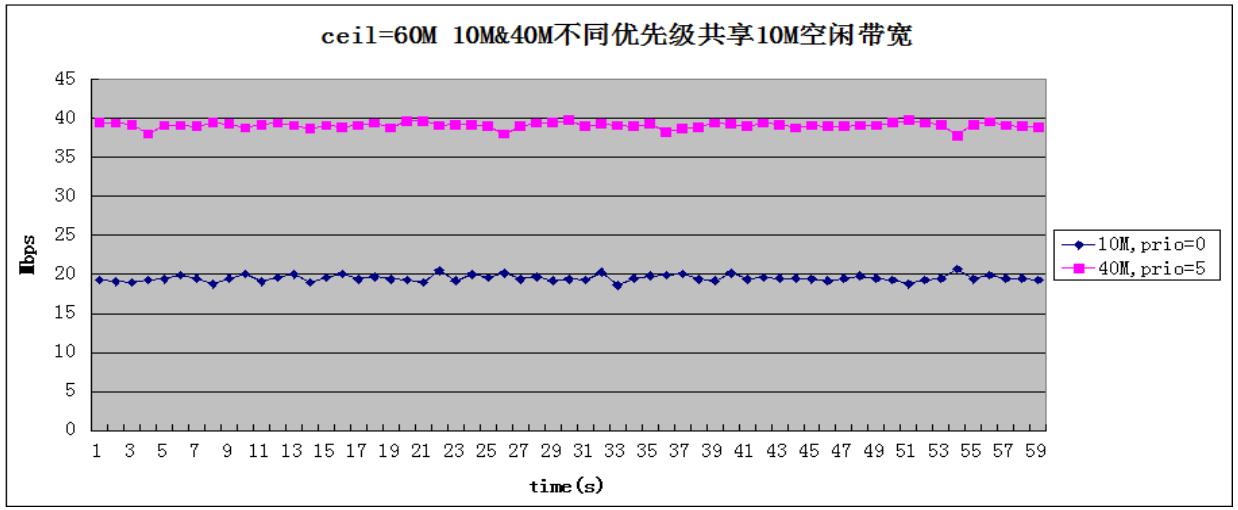


ceil=60M

bps=10M, priority=0

bps=50M, priority=0

No free resources



ceil=60M

bps=10M, priority=0

bps=40M, priority=5

Free resources of 10M

Thank you