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

- 陈东东，腾讯

- open source summit 2019

提纲:

1、kernel cgroup支持buffer write io弹性限制

2、kernel cgroup支持网络入带宽弹性限制

背景:

Kubernetes平台的一个重要目标是资源管理，目前是通过kernel cgroup subsystem管理节点上的资源。目标是保证业务申请资源，且在宿主机有空闲资源的前提下，可以弹性借用（称之为软限制），突破其申请限制，同时确保宿主机资源稳定。目前资源有如下：

- Cpu： 通过cpu，cpuacct，cpuset cgroup管理
- Memory：通过memory cgroup管理
- Blkio：通过blkio cgroup管理，对buffer write失控
- Network：通过tc，iptables，net_cls cgroup管理网络出带宽，缺少控制网络入带宽的cgroup
- Disk: 通过disk_quota限制

为了更好地支持全维度资源管理，包括硬限制和软限制，我们在kernel中新增了两个cgroup subsystem，弹性管理buffer write和网络入带宽。

1.1 buffer write弹性限制-背景

目标：

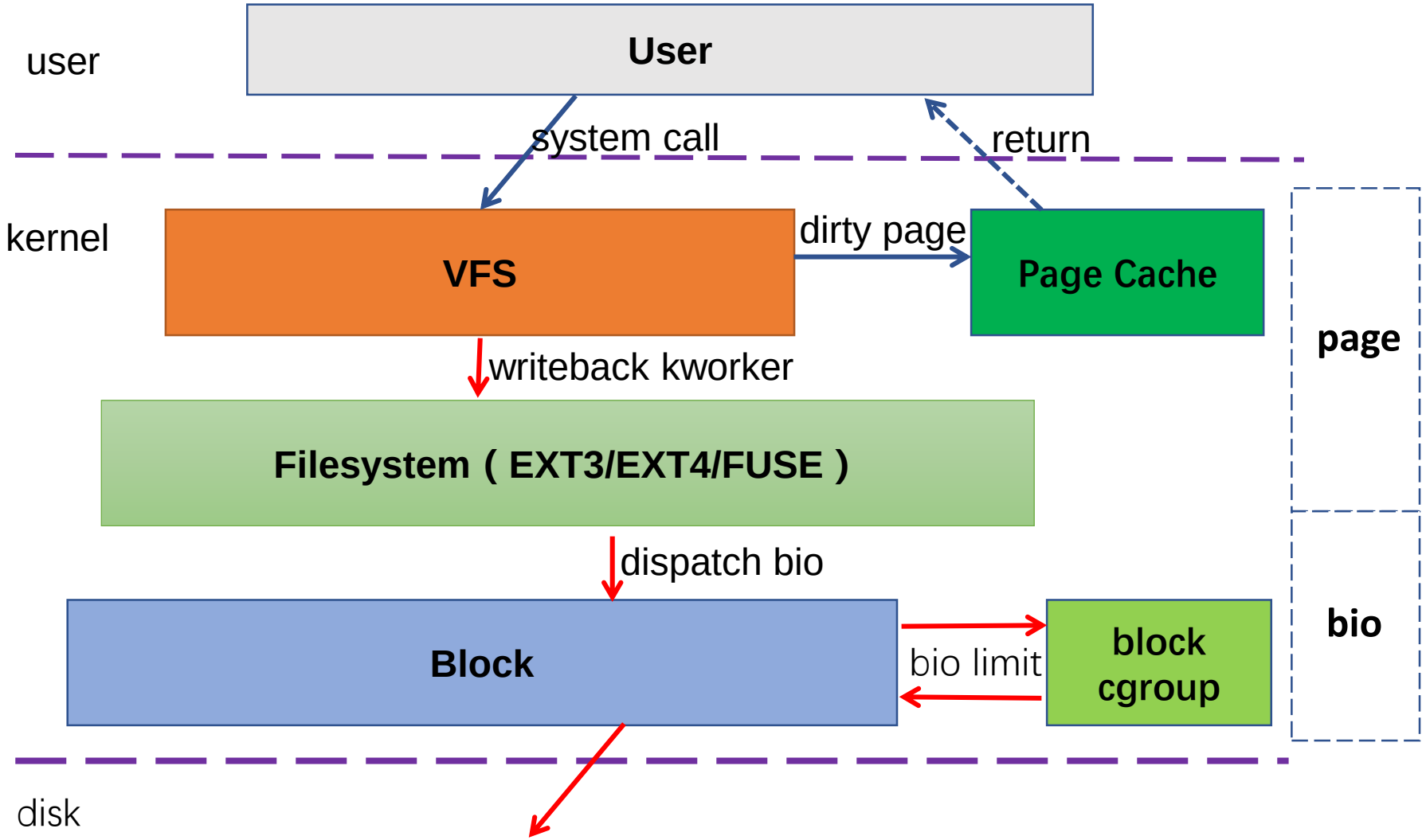
业务（进程）的磁盘IO读写可控，包括buffer write。且支持弹性借用。

现状：

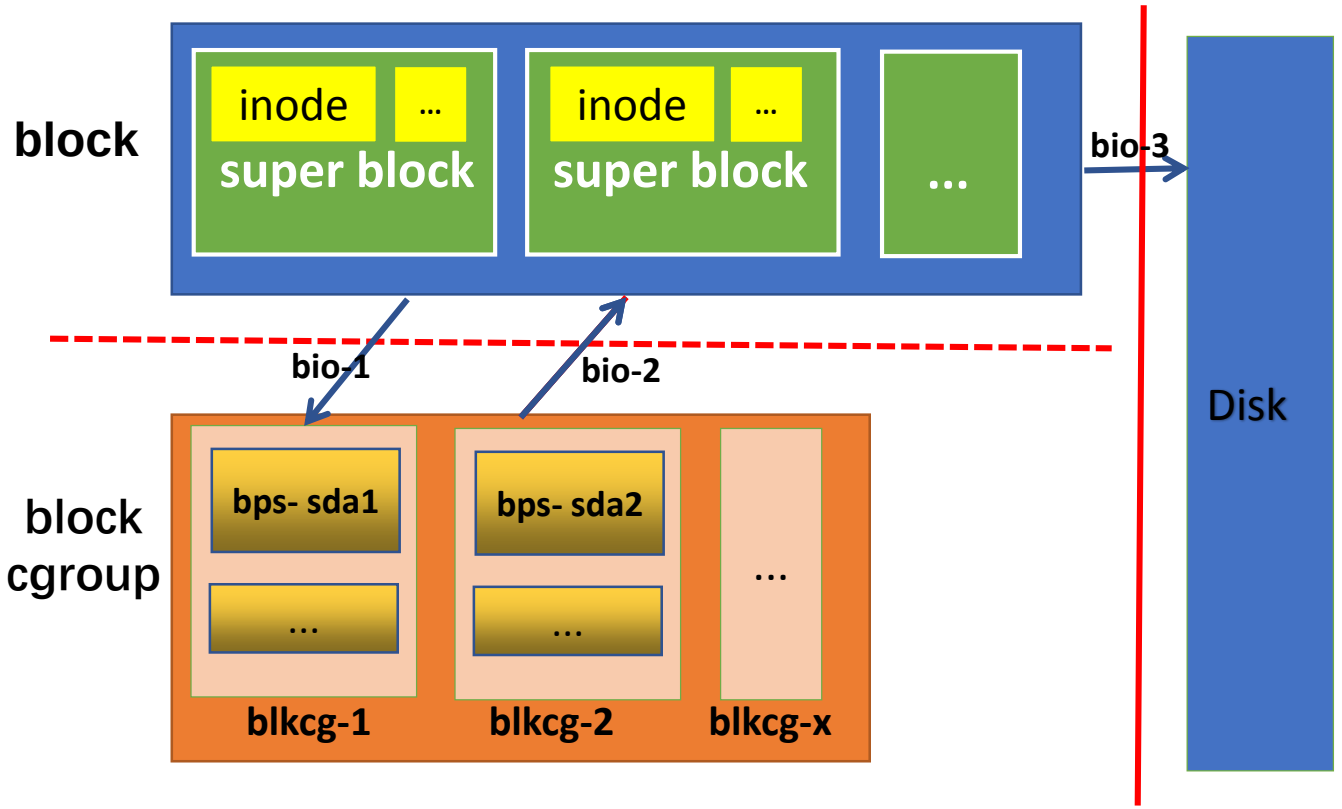
blkio cgroup目前只支持direct IO，对buffer IO失控。原因是在buffer write过程中，block io失去了原有写进程的信息，该io是由内核线程发起的。

cgroup v2可以支持buffer io限速，但cgroup的目录结构已发生变化(unified hierarchy)，也需要更高版本的内核支持（4.5）。

Buffer write流程



1.2 inode增加block cgroup属性



inode： 文件节点

super block： 元数据

block cgroup： cgroup block io模块 (/sys/fs/cgroup/blkio/)

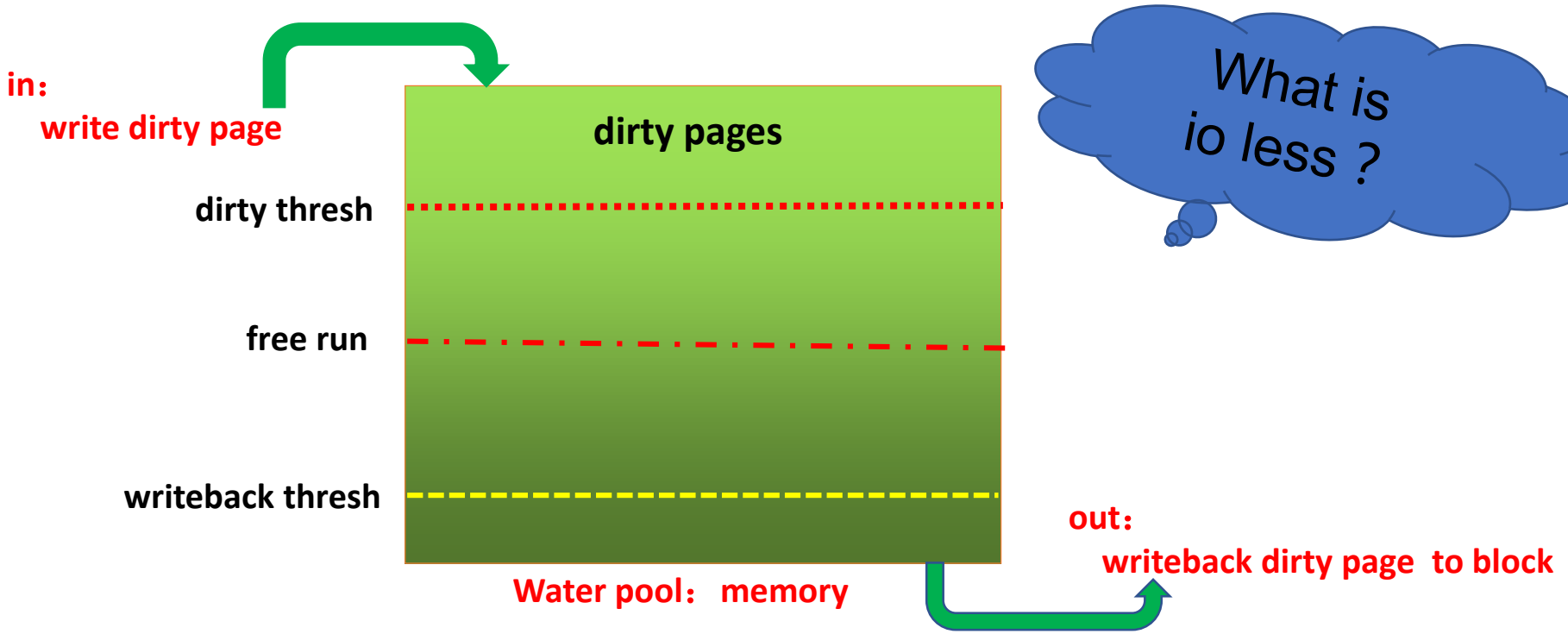
blkcg-x： 一个block cgroup名称

bps-sdax： 磁盘sdax在该cgroup设置的速率

1、在inode结构体中记录进程所属block cgroup

2、进程写脏页阶段赋值进程所属的block cgroup到inode结构体

1.3 io less下沉到memory cgroup



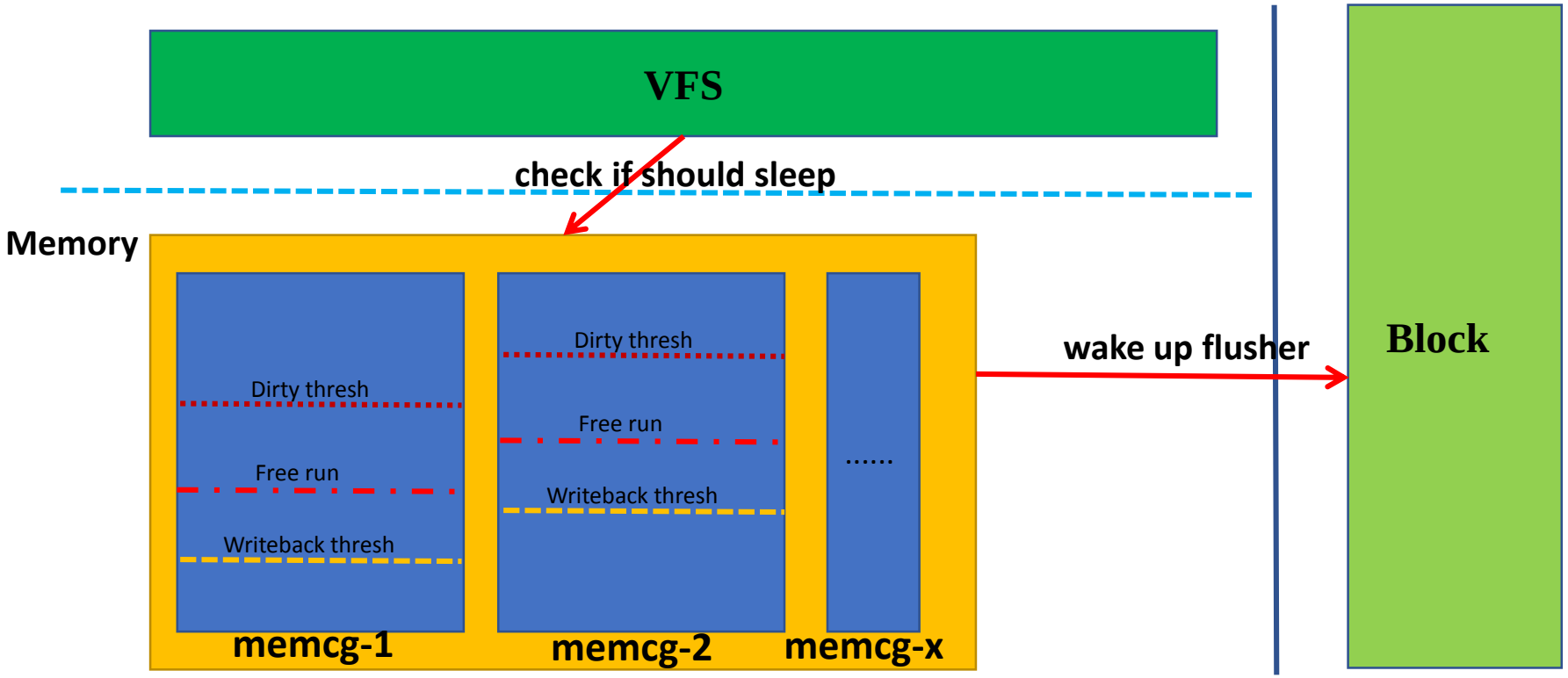
1、dirty pages：内存中处于dirty及writeback状态的页数目。dirty表示page上的数据还未写到磁盘。writeback表示page上的数据正在写入磁盘

2、writeback thresh：/proc/sys/vm/dirty_writeback_ratio(_bytes)

3、dirty thresh：/proc/sys/vm/dirty_ratio(_bytes)

4、free run: (writeback thresh + dirty thresh) / 2

1.4 io less下沉到memory cgroup

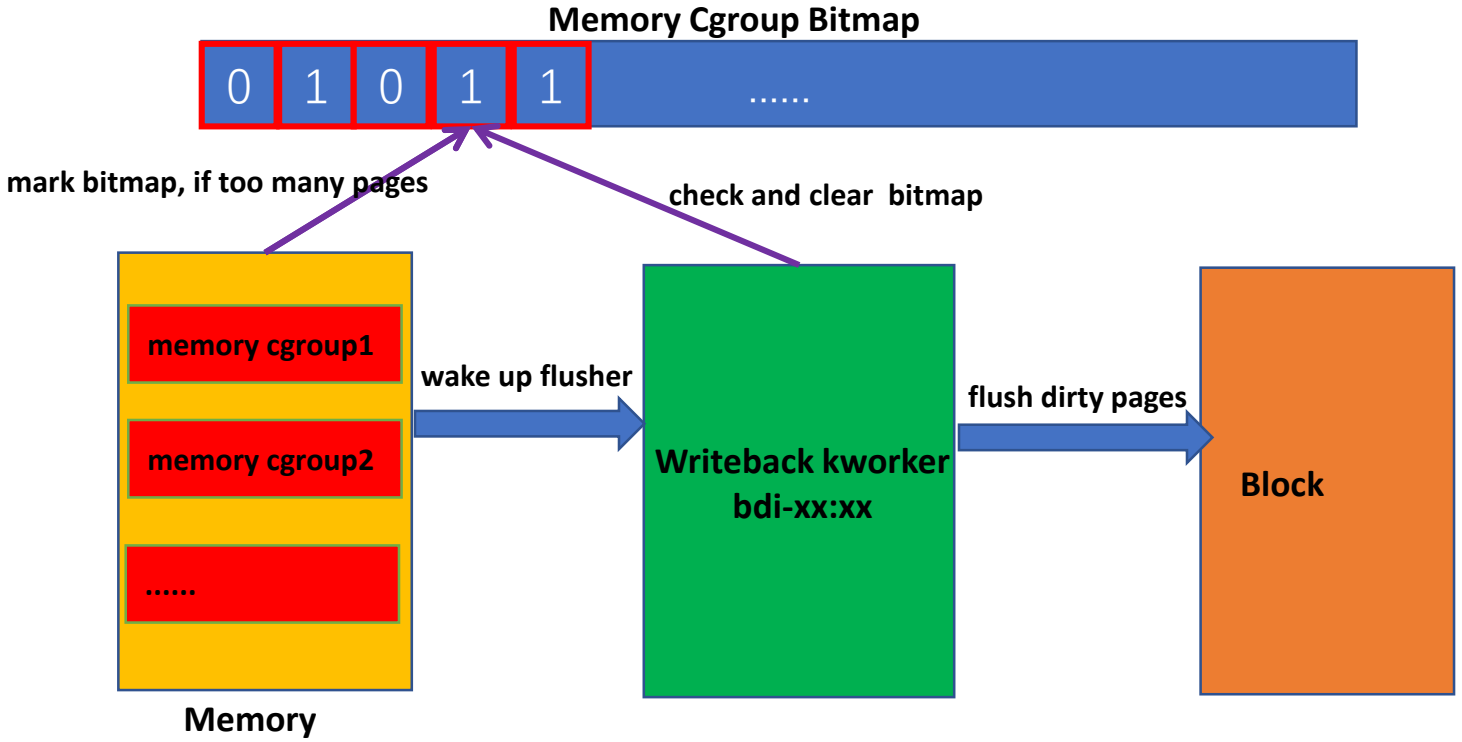


1、memory cgroup(/sys/fs/cgroup/memory)增加统计处于dirty和writeback状态的page

2、memory cgroup有自己的writeback和dirty thresh（/proc/sys/vm/）

3、应用io less机制到memory cgroup

1.5 writeback过程感知memory cgroup

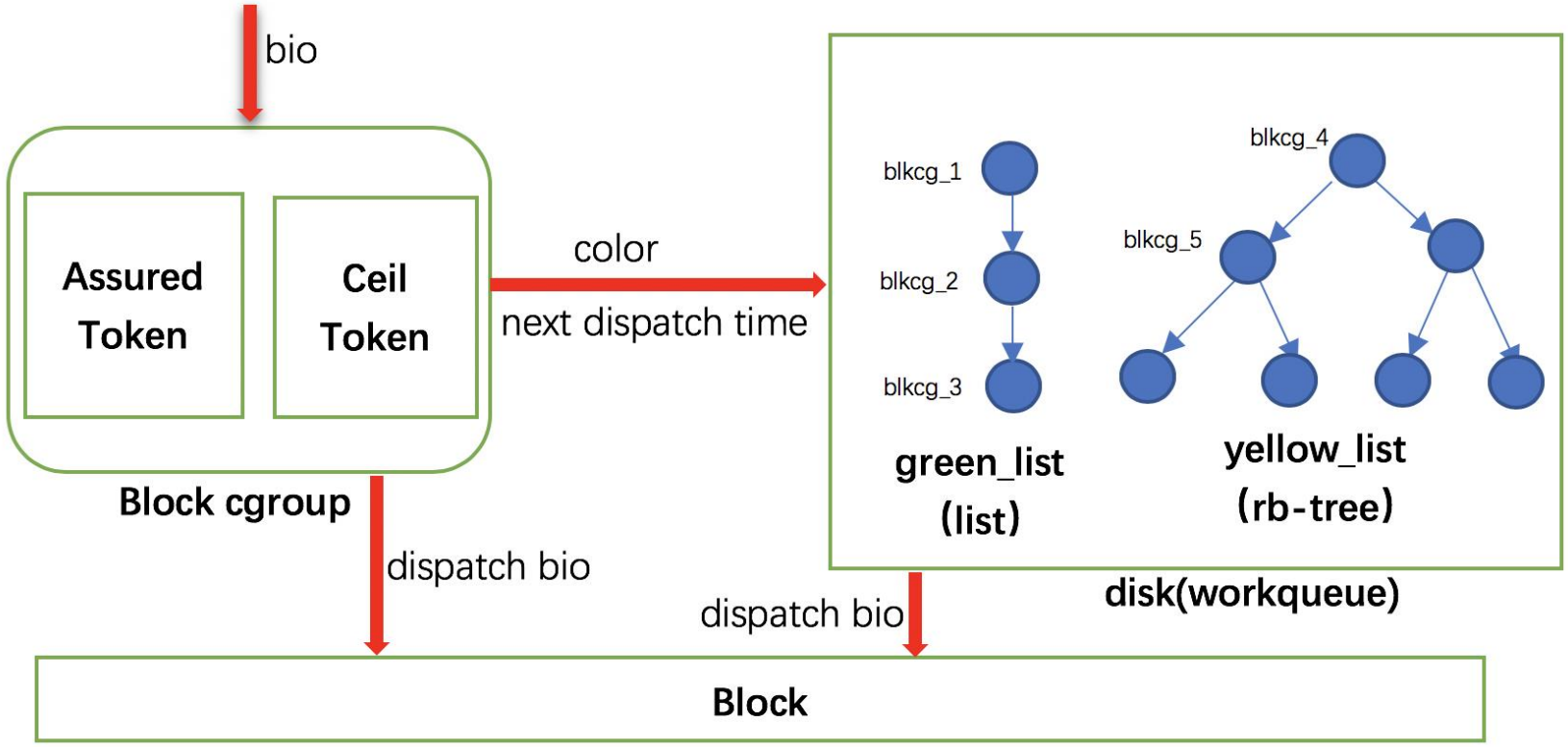


1、增加全局bitmap，记录memory cgroup是否脏页过多

2、memory cgroup脏页过多时，标记对应的bitmap为1

3、writeback过程中检测bitmap是否有被标识为1，在下发bio时，清空bitmap为0

1.6 buffer write弹性限制



（1）每个block cgroup维护两个令牌桶：保证速率和最大速率

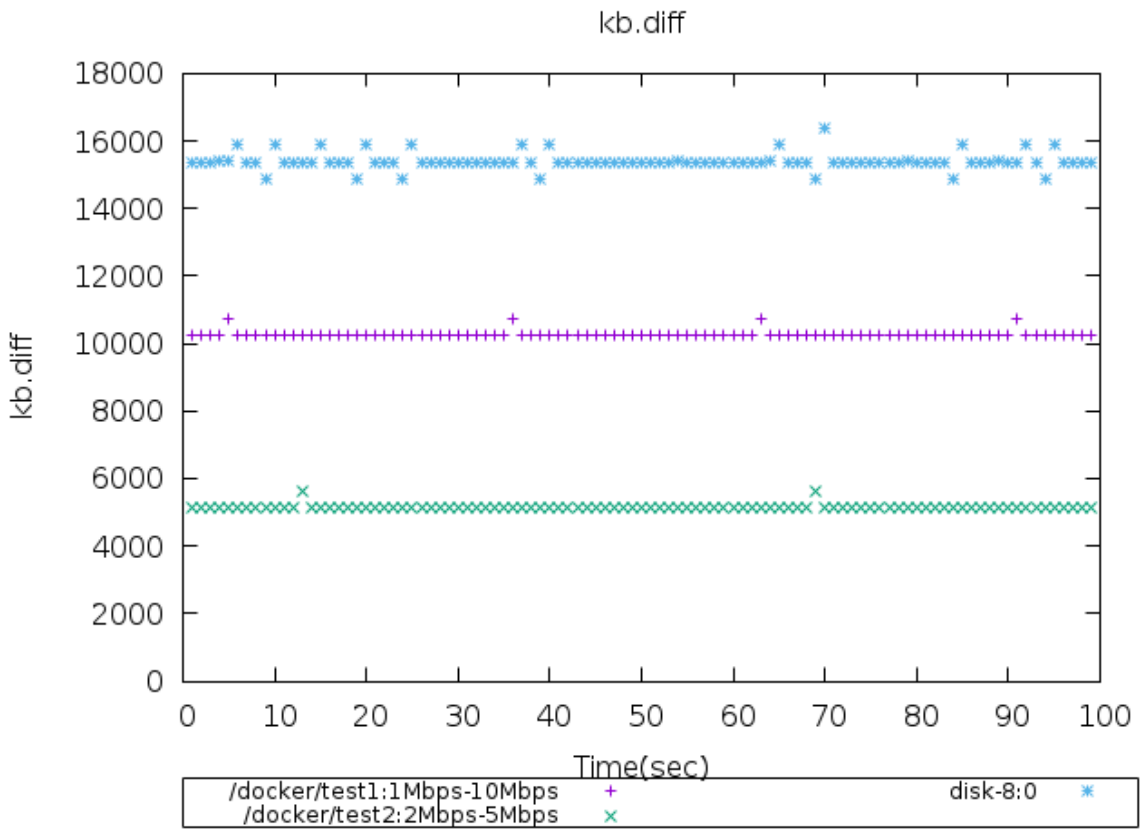
（3）block cgroup根据令牌数目维持一种颜色属性：绿色表示两个令牌桶都有令牌；若assured无令牌，ceil有令牌，颜色为黄色；若都没有令牌，颜色为红色

（4）block cgroup维持一个下次下发bio的时刻，即按照速率限制，何时可以下发bio

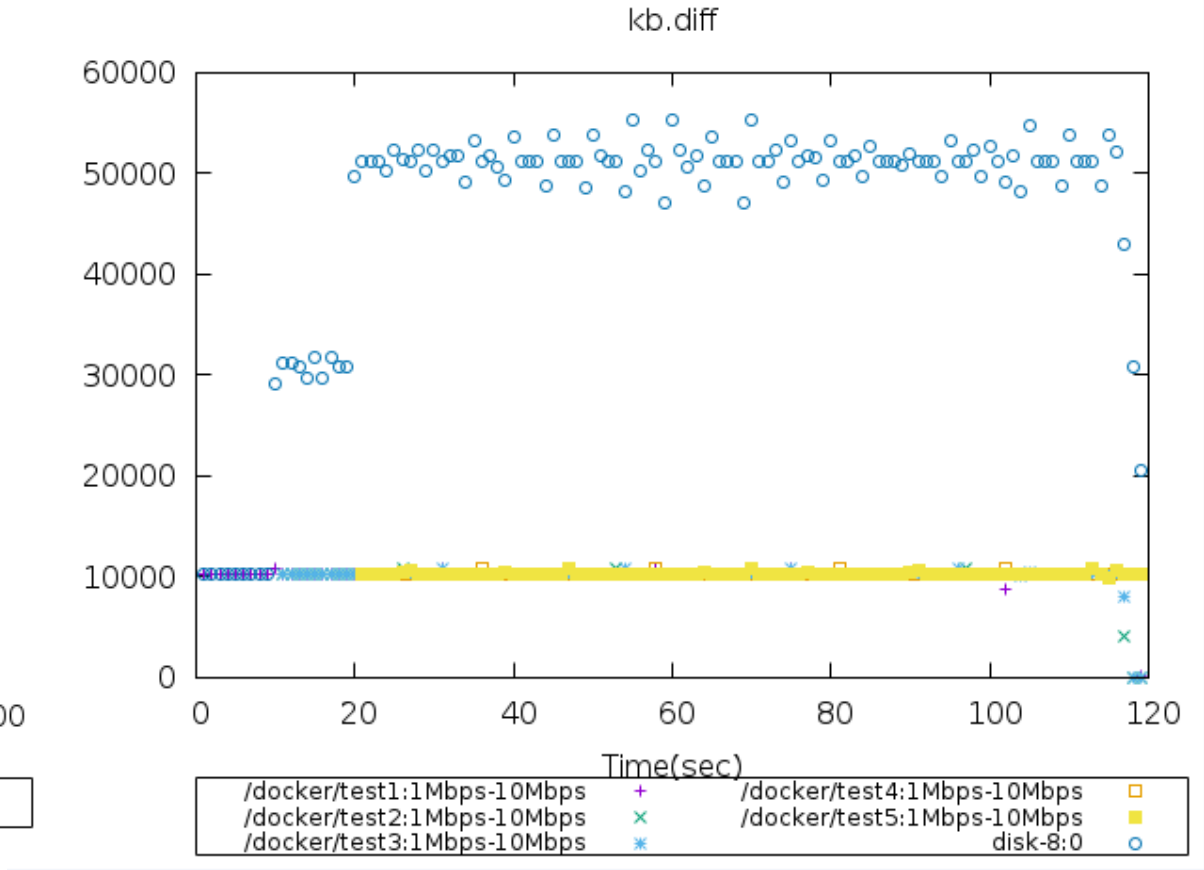
（5）每个磁盘维护其关联的block cgroup，其中绿色的cgroup采用链表维护，而黄色的采用红黑树维护

（6）下发bio时，绿色链表全部下发，黄色链表采用Deficit Round Robin方式轮训，实现高优先级抢占更多的资源

1.7 buffer write弹性限制-实验

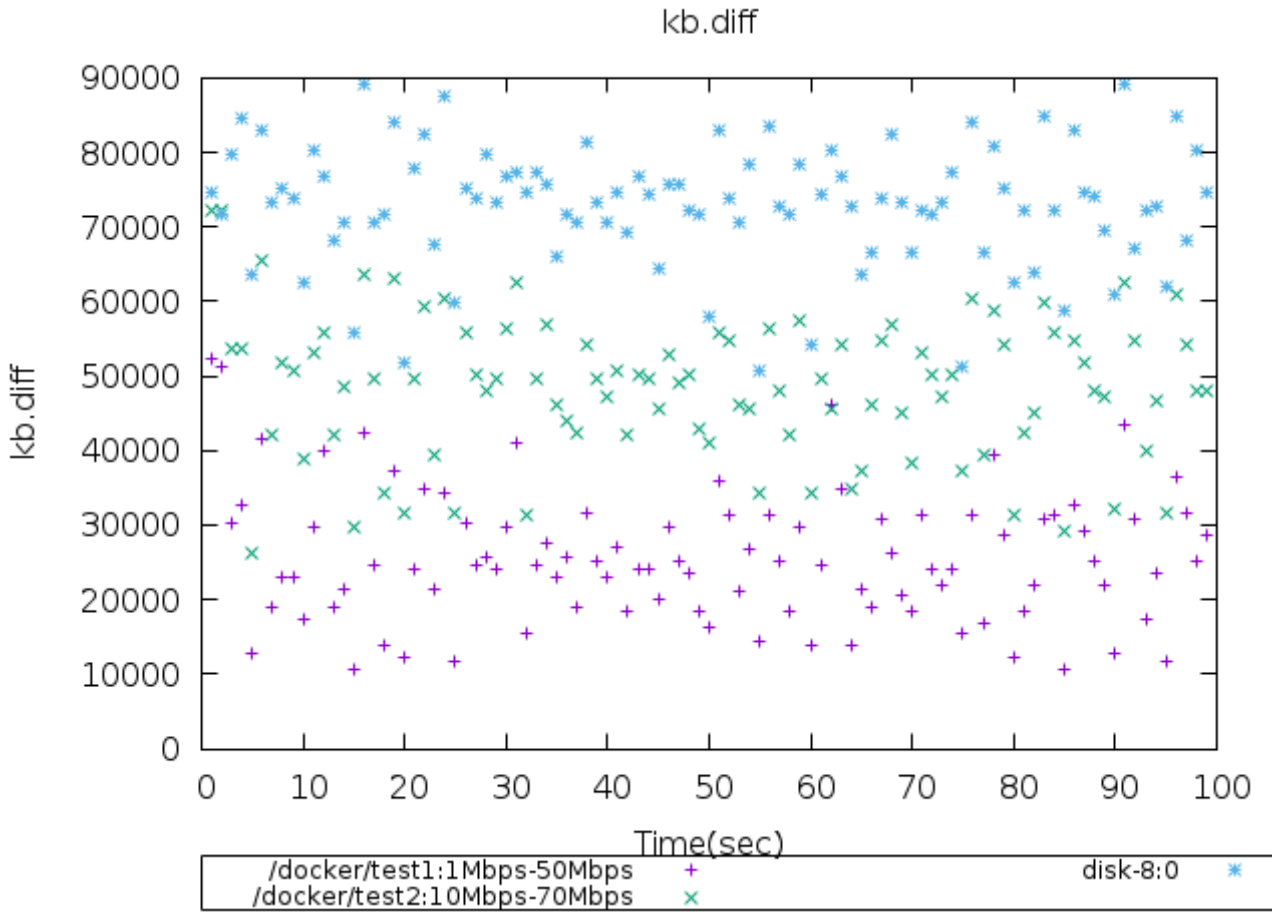


5Mbps和10Mbps并行



5个10Mbps延时并发

1.8 buffer write弹性限制-实验



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

注：若空闲带宽不能满足并行IO的ceil值总和，则会优先分配给assured值比较大（更高优先级）的IO。

2.1 网络入带宽限制-背景

目标：

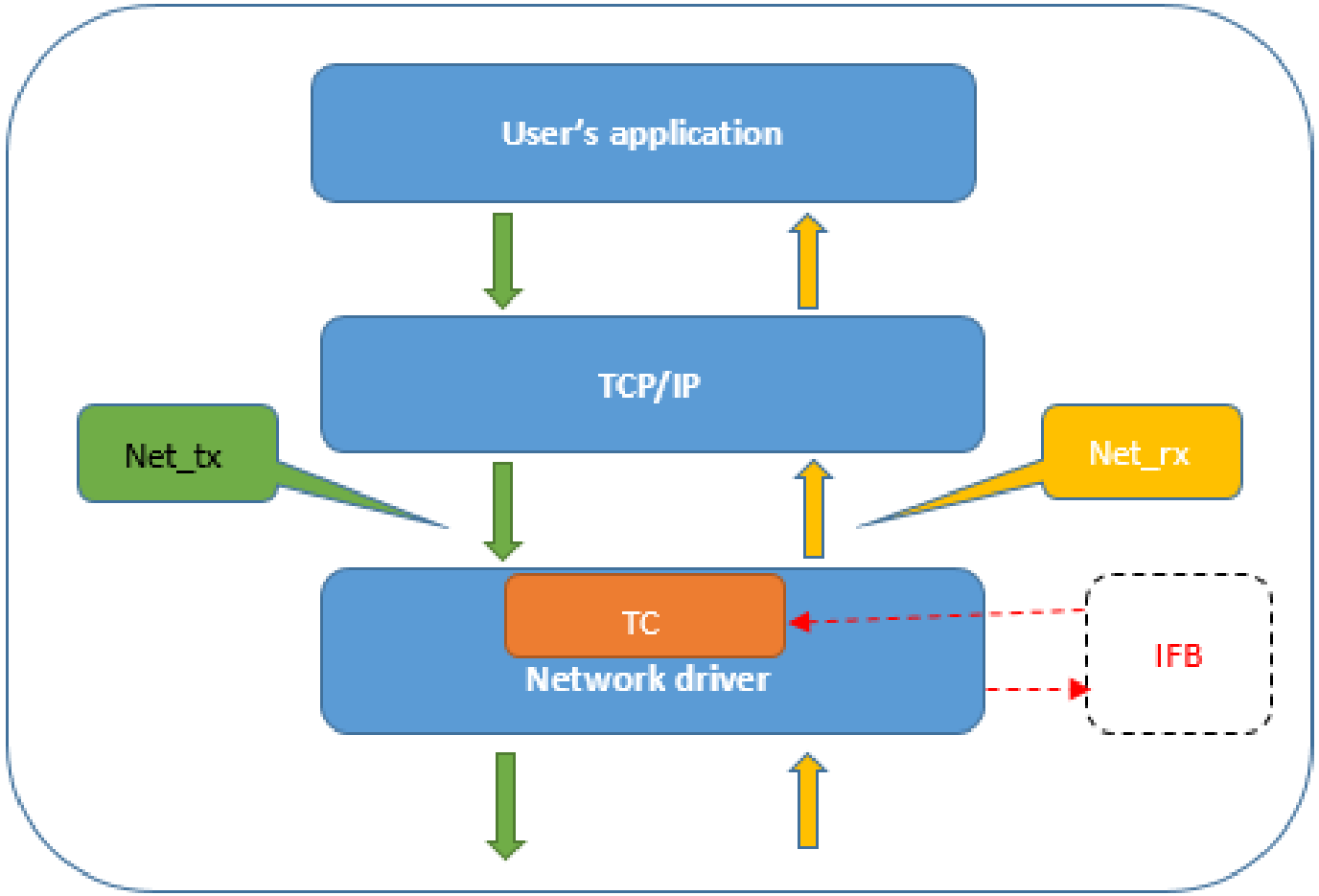
网络入带宽可通过cgroup控制，尽量减小丢包，支持弹性借用。

现状：

现有网络入带宽控制方式：

- tc egress，需配置dst ip
- tc policing(ingress)
- IFB(Intermediate Functional Block device)，把入带宽转换为出带宽，再通过tc进行控制

2.1 网络入带宽限制-背景



（1）通过iptables或net_cls对数据包打标签

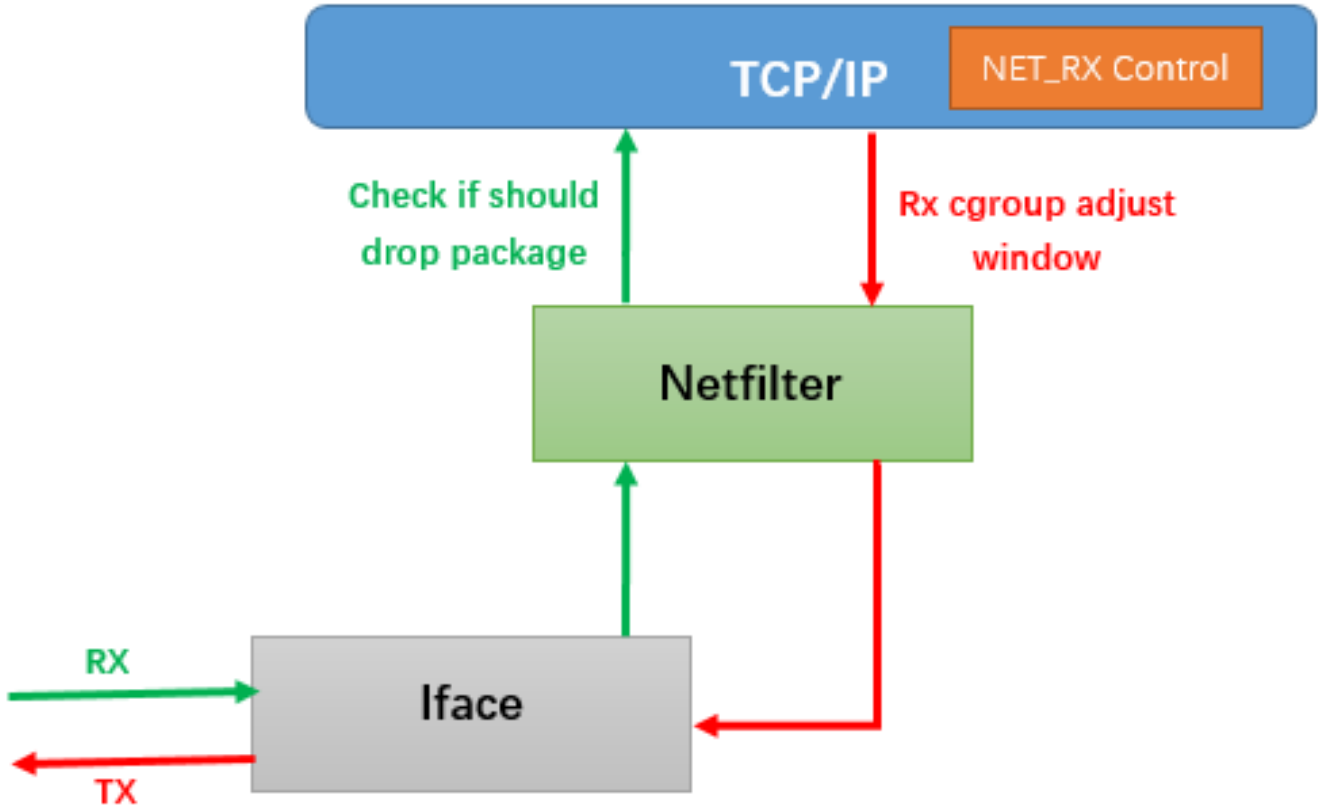
（2）通过tc限制网络带宽

（3）对于入流量，先经过TC模块，此时是没有进程信息的，无法匹配到具体哪个业务

（4）增加ifb模块，把入流量转化为虚拟网卡的出流量，再利用TC进行限流，开销相对比较大

（5）入流量受限后丢包重传，增加网络传输量

2.2 网络入带宽弹性限制-实现



入流量:

（1）在sockt接收到包时增加网络入带宽控制，令牌桶方式精确控制速率 。

（2）收包过程根据令牌数目判断是否丢弃。

（3）回包过程配置滑动窗口大小，通知对端慢点发送，减少丢包。

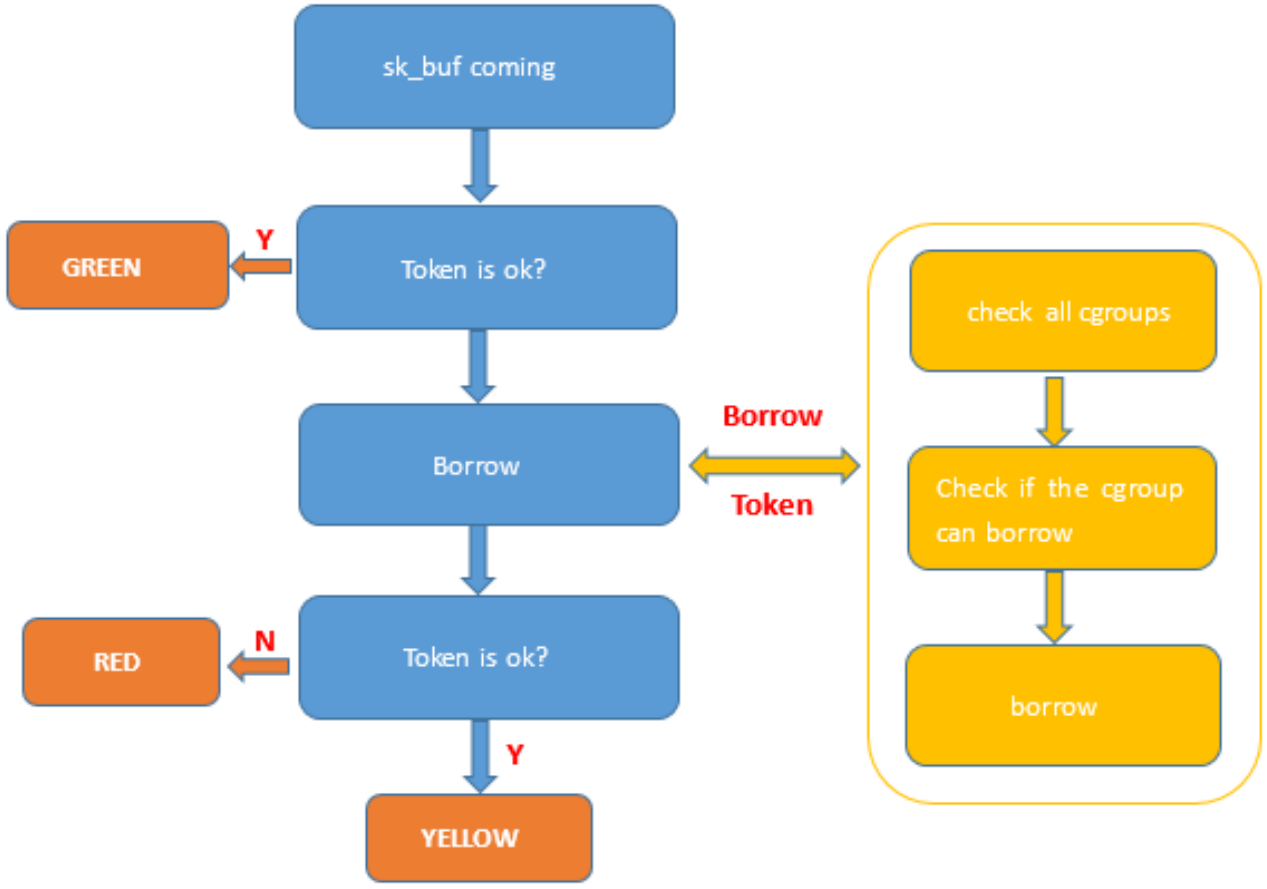
（3）把需要限制的进程放入cgroup中，摈弃ip+端口。

（4）cgoup繁忙时保证带宽资源不会被挤占，即入带宽速率受保证。

（5）cgroup不繁忙时，空闲带宽资源可弹性借出。

（6）多个cgroup共享带宽时，按照优先级分配空闲资源。

2.2 网络入带宽弹性限制-实现

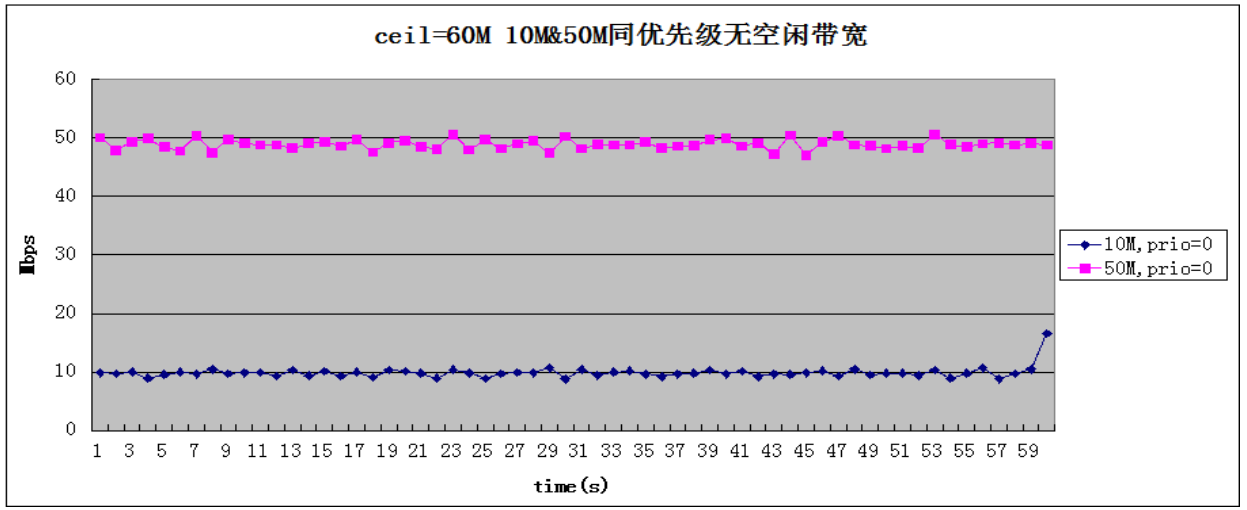


（1）cgroup的根目录配置一个总带宽值，各个子cgroup目录之和不能超过这个值

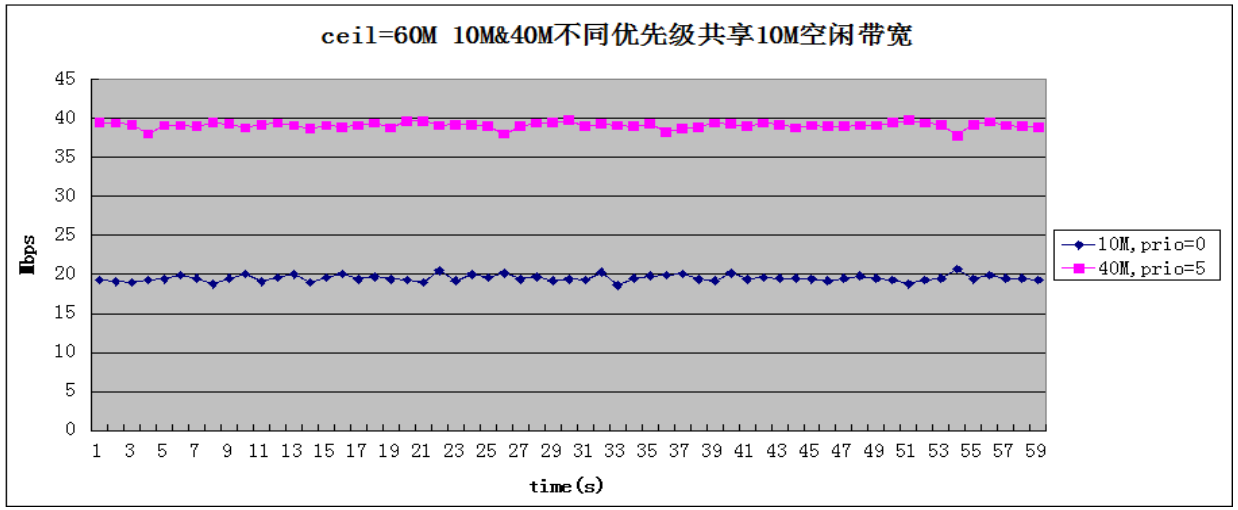
（2）每个cgoup维护一个令牌桶，其中根目录令牌桶的令牌为所有子cgroup申请之后剩余的资源

(3) 借用过程中，遵循优先级原则，高优先级可借用更多的令牌

2.3 网络入带宽弹性限制-实验



总带宽为60M,
10M优先级为0，
50M优先级为0，
无共享带宽



总带宽为60M
10M优先级为0，
40M优先级为5，
有10M共享带宽

Thank you