一行命令，丢包率从 15％直降为

0%

有没有遇到这种情况，网络带宽使用率只有 50%,CPU
使用率也很低，但是经常会出现网络丢包的告警，伴随着响应超时和请求缓慢，应用日志中还查不到记录。

那么大概率就是受到了这个参数设置的限制， net.core.netdev_max-backlog。

查看系统当前的值

它是 Linux 内核网络子系统中一个控制网络接收队列长度的参数。新到的数据包必须先在内核的 backlog 队列里排队等候。

目前主流的发行版本这个值默认为 1000，意味着一旦瞬间有超过 1000个数据包涌来而 CPU
来不及处理，超出的部分会被内核直接丢弃，不通知任何应用程序，这也是我们为什么应用查不到记录的原因
。

显然，这个值在真实的高 PPS场景中是非常脆弱的，只要上游出现一次网络突刺就极有可能导致 backlog 队列就会瞬间溢出。

用 iperf3模拟高频 UDP请求对比下这个参数调整前后的变化

先启动一个新的窗口实时观察 backlog 丢包情况

监控丢包情况

```
#实时监控每个CPU的丢包数，每0.5秒刷新，高亮变化
$ watch -n 0.5 d 'cat /proc/net/softnet_stat | awk
"(printf "CPU%d: processed=%d dropped=%d\n", NR-1,
strtonum("Ox"$1), strtonum("Ox"$2))"
```

公众号 · 老王技术栈

□□□□□□□□ CPU □ dropped□□□ 0

□□□□□ iperf3 □□

服务端压测监听

```
#服务端，以UDP模式启动 iperf3，每秒输出一次统计
$ iperf3 s-i1
```

公众号 · 老王技术栈

□□□□□□□

客户端压测

```
#向服务端发送UDP 小包(200字节)，带宽1Gbps，持续60秒

$ iperf3 -u-c <server_lp> -b 1G-I 200-t 60
```

公众号 · 老王技术栈

□□□□□□□□ 1000 □□□□□

backlog=1000下监控输出

```
CPU0: processed=2812345 dropped=18432
CPU1: processed=2389761 dropped=15104
CPU2: processed=2678142 dropped=16896
CPU3: processed=2556712 dropped=16384
```

公众号 · 老王技术栈

□□□□□□ CPU □□□□□□□□□□□□□ 15%□

□□□□ 5000□□□□□□□

配置修改

修改值为5000后重新压测观察

```
# 临时修改(立即生效)
$ sudo sysctl -w net.core.netdev_ max_backlog=5000

永久修改(写入配置文件)
$ echo "net.core.netdev_max_backlog = 5000" | sudo
tee -a/etc/sysctl.conf
$ sudo sysctl-p
```

公众号 · 老王技术栈

□□□□□□□□□□□□□□□

backlog=5000下监控输出

```
CPU0: processed=2853742 dropped=0
CPU1: processed=2421501 dropped=0
CPU2: processed=2710184 dropped=0
CPU3: processed=2584547 dropped-0
```

公众号 · 老王技术栈

所有 CPU 的 dropped均为 0，丢包率 0%，网卡 dropped计数器也停止增长。

丢包率从 15％下降到了 0%。

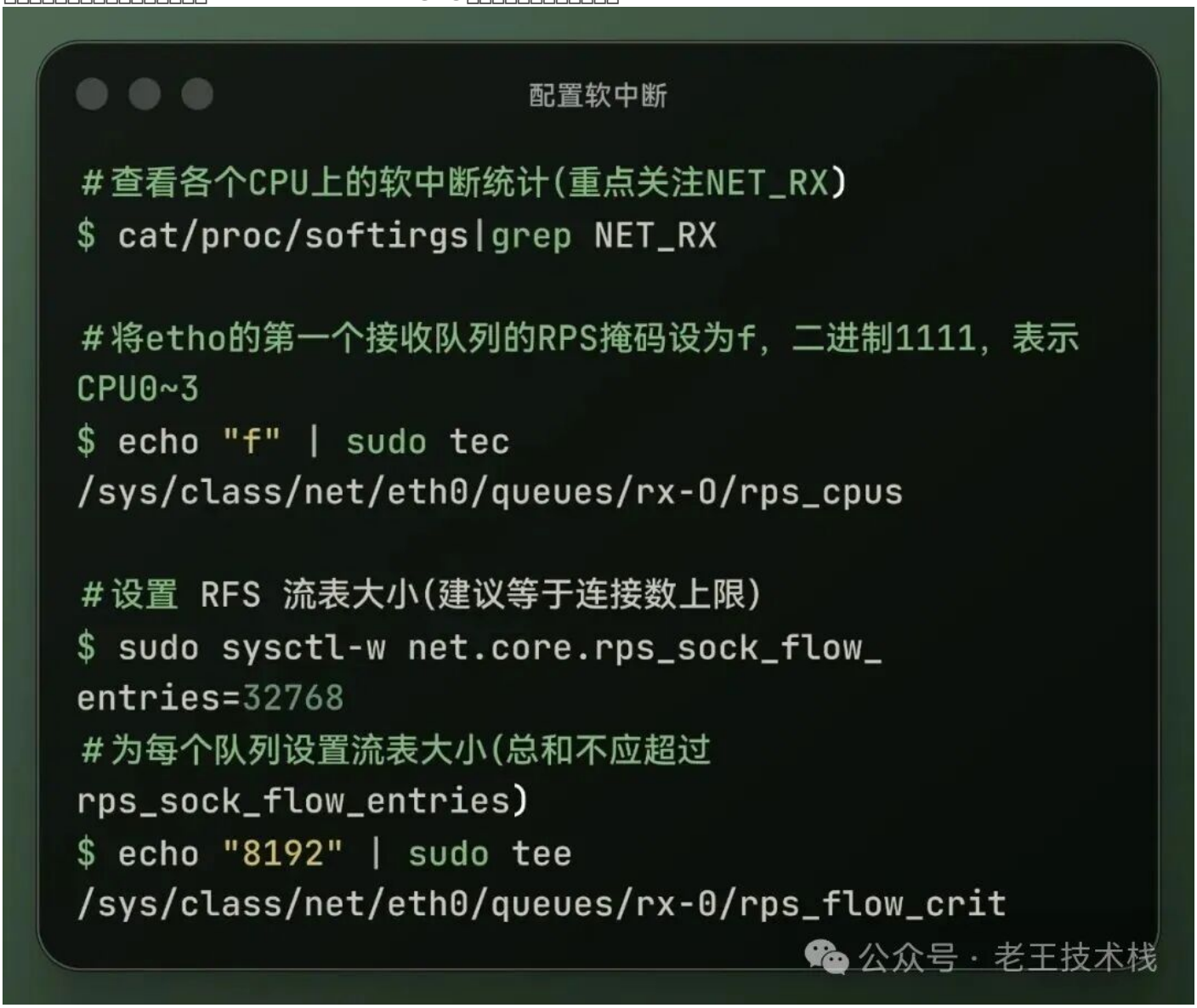
那真实环境需要调整多大合适呢？

常规场景调整到 5000~10000;
高 PPS场景调整到 30000~50000（需压测验证）。

需要注意的是，如果还想进一步调优就需要配合 RPS提升软中断能力。

调大 netdev_max_backlog 只是解决了队列容量问题，但如果没有足够的软中断处理能力，
队列依然会持续积压。

提高软中断处理能力的就是是让多个 CPU核共同分担网络收包处理。



开启 RPS(Receive PacketSteering)LRPS将收到的数据包哈希分发到多个 CPU 的 backlog 队列上，让软中断并行处理。

开启 RFS(Receive Flow Steering): RFS 是 RPS 的升级版，能将同一条流（例如同一个 TCP连接) 数据包送到同一个 CPU

net.core.netdev_max backlog是 Linux 网络接收路径上容易被忽视的参数，目前的默认值很有可能成为高
PPS场景下的网络瓶颈。

Revision #1
Created 2026-06-13 07:21:37 UTC by Admin
Updated 2026-06-13 07:24:24 UTC by Admin