

10 | 案例篇：系统的软中断CPU使用率升高，我该怎么办？

2018-12-12 倪朋飞



朗读：冯永吉

时长12:47 大小11.72M



你好，我是倪朋飞。

上一期我给你讲了软中断的基本原理，我们先来简单复习下。

中断是一种异步的事件处理机制，用来提高系统的并发处理能力。中断事件发生，会触发执行中断处理程序，而中断处理程序被分为上半部和下半部这两个部分。

上半部对应硬中断，用来快速处理中断；

下半部对应软中断，用来异步处理上半部未完成的工作。

Linux 中的软中断包括网络收发、定时、调度、RCU 锁等各种类型，我们可以查看 `proc` 文件系统中的 `/proc/softirqs`，观察软中断的运行情况。

在 Linux 中，每个 CPU 都对应一个软中断内核线程，名字是 ksoftirqd/CPU 编号。当软中断事件的频率过高时，内核线程也会因为 CPU 使用率过高而导致软中断处理不及时，进而引发网络收发延迟、调度缓慢等性能问题。

软中断 CPU 使用率过高也是一种最常见的性能问题。今天，我就用最常见的反向代理服务器 Nginx 的案例，教你学会分析这种情况。

案例

你的准备

接下来的案例基于 Ubuntu 18.04，也同样适用于其他的 Linux 系统。我使用的案例环境是这样的：

机器配置：2 CPU、8 GB 内存。

预先安装 docker、sysstat、sar、hping3、tcpdump 等工具，比如 `apt-get install docker.io sysstat hping3 tcpdump`。

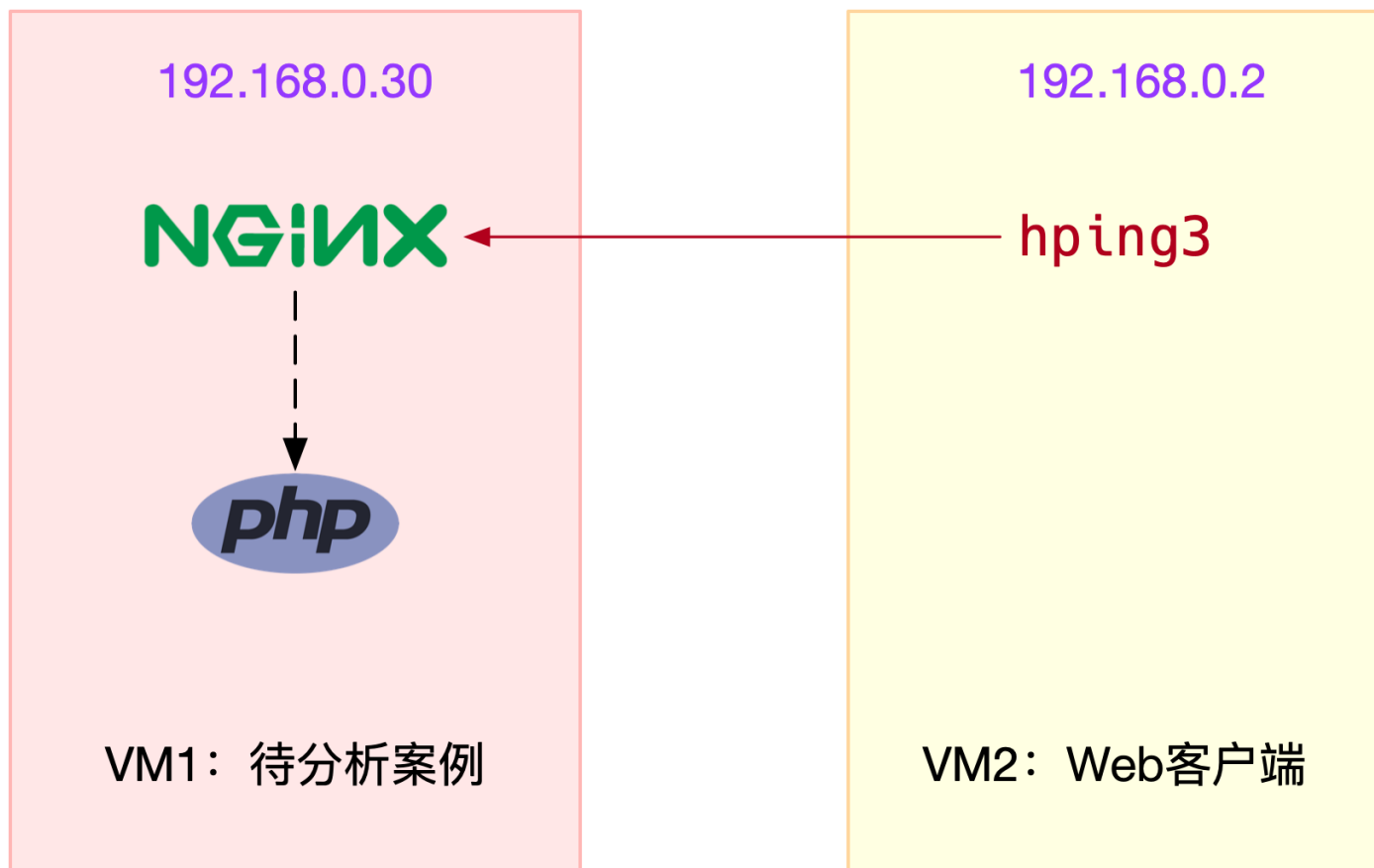
这里我又用到了三个新工具，sar、hping3 和 tcpdump，先简单介绍一下：

sar 是一个系统活动报告工具，既可以实时查看系统的当前活动，又可以配置保存和报告历史统计数据。

hping3 是一个可以构造 TCP/IP 协议数据包的工具，可以对系统进行安全审计、防火墙测试等。

tcpdump 是一个常用的网络抓包工具，常用来分析各种网络问题。

本次案例用到两台虚拟机，我画了一张图来表示它们的关系。



你可以看到，其中一台虚拟机运行 Nginx ，用来模拟待分析的 Web 服务器；而另一台当作 Web 服务器的客户端，用来给 Nginx 增加压力请求。使用两台虚拟机的目的，是为了相互隔离，避免“交叉感染”。

接下来，我们打开两个终端，分别 SSH 登录到两台机器上，并安装上面提到的这些工具。

同以前的案例一样，下面的所有命令都默认以 root 用户运行，如果你是用普通用户身份登陆系统，请运行 `sudo su root` 命令切换到 root 用户。

如果安装过程中有什么问题，同样鼓励你先自己搜索解决，解决不了的，可以在留言区向我提问。如果你以前已经安装过了，就可以忽略这一点了。


操作和分析

安装完成后，我们先在第一个终端，执行下面的命令运行案例，也就是一个最基本的 Nginx 应用：

 复制代码


```
1 # 运行 Nginx 服务并对外开放 80 端口
2 $ docker run -itd --name=nginx -p 80:80 nginx
```

然后，在第二个终端，使用 curl 访问 Nginx 监听的端口，确认 Nginx 正常启动。假设 192.168.0.30 是 Nginx 所在虚拟机的 IP 地址，运行 curl 命令后你应该会看到下面这个输出界面：

 复制代码

```
1 $ curl http://192.168.0.30/
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <title>Welcome to nginx!</title>
6 ...
```

接着，还是在第二个终端，我们运行 hping3 命令，来模拟 Nginx 的客户端请求：

 复制代码


```
1 # -S 参数表示设置 TCP 协议的 SYN（同步序列号），-p 表示目的端口为 80
2 # -i u100 表示每隔 100 微秒发送一个网络帧
3 # 注：如果你在实践中现象不明显，可以尝试把 100 调小，比如调成 10 甚至 1
4 $ hping3 -S -p 80 -i u100 192.168.0.30
```

现在我们再回到第一个终端，你应该发现了异常。是不是感觉系统响应明显变慢了，即便只是在终端中敲几个回车，都得很久才能得到响应？这个时候应该怎么办呢？

虽然在运行 hping3 命令时，我就已经告诉你，这是一个 SYN FLOOD 攻击，你肯定也会想到从网络方面入手，来分析这个问题。不过，在实际的生产环境中，没人直接告诉你原因。

所以，我希望你把 hping3 模拟 SYN FLOOD 这个操作暂时忘掉，然后重新从观察到的问题开始，分析系统的资源使用情况，逐步找出问题的根源。

那么，该从什么地方入手呢？刚才我们发现，简单的 SHELL 命令都明显变慢了，先看看系统的整体资源使用情况应该是个不错的注意，比如执行下 top 看看是不是出现了 CPU 的瓶颈。我们在第一个终端运行 top 命令，看一下系统整体的资源使用情况。

 复制代码

```
1 # top 运行后按数字 1 切换到显示所有 CPU
2 $ top
3 top - 10:50:58 up 1 days, 22:10,  1 user,  load average: 0.00, 0.00, 0.00

4 Tasks: 122 total,   1 running,  71 sleeping,   0 stopped,   0 zombie
5 %Cpu0  :  0.0 us,   0.0 sy,   0.0 ni, 96.7 id,   0.0 wa,   0.0 hi,  3.3 si,   0.0 st
```

```

6 %Cpu1 :  0.0 us,  0.0 sy,  0.0 ni, 95.6 id,  0.0 wa,  0.0 hi,  4.4 si,  0.0 st
7 ...
8
9  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
10    7 root        20   0       0       0       0 S   0.3   0.0    0:01.64 ksoftirqd/0
11   16 root        20   0       0       0       0 S   0.3   0.0    0:01.97 ksoftirqd/1
12  2663 root        20   0  923480  28292  13996 S   0.3   0.3    4:58.66 docker-containe
13  3699 root        20   0       0       0       0 I   0.3   0.0    0:00.13 kworker/u4:0
14  3708 root        20   0   44572   4176   3512 R   0.3   0.1    0:00.07 top
15    1 root        20   0  225384   9136   6724 S   0.0   0.1    0:23.25 systemd
16    2 root        20   0       0       0       0 S   0.0   0.0    0:00.03 kthreadd
17 ...

```

这里你有没有发现异常的现象？我们从第一行开始，逐个看一下：

平均负载全是 0，就绪队列里面只有一个进程（1 running）。

每个 CPU 的使用率都挺低，最高的 CPU1 的使用率也只有 4.4%，并不算高。

再看进程列表，CPU 使用率最高的进程也只有 0.3%，还是不高呀。

那为什么系统的响应变慢了呢？既然每个指标的数值都不大，那我们就再来看看，这些指标对应的更具体的含义。毕竟，哪怕是同一个指标，用在系统的不同部位和场景上，都有可能对应着不同的性能问题。


仔细看 top 的输出，两个 CPU 的使用率虽然分别只有 3.3% 和 4.4%，但都用在了软中断上；而从进程列表上也可以看到，CPU 使用率最高的也是软中断进程 ksoftirqd。看起来，软中断有点可疑了。

根据上一期的内容，既然软中断可能有问题，那你先要知道，究竟是哪类软中断的问题。停下来想想，上一节我们用了什么方法，来判断软中断类型呢？没错，还是 proc 文件系统。观察 /proc/softirqs 文件的内容，你就能知道各种软中断类型的次数。

不过，这里的各类软中断次数，又是什么时间段里的次数呢？它是系统运行以来的**累积中断次数**。所以我们直接查看文件内容，得到的只是累积中断次数，对这里的问题并没有直接参考意义。因为，这些**中断次数的变化速率**才是我们需要关注的。

那什么工具可以观察命令输出的变化情况呢？我想你应该想起来了，在前面案例中用过的 watch 命令，就可以定期运行一个命令来查看输出；如果再加上 -d 参数，还可以高亮出变化的部分，从高亮部分我们就可以直观看出，哪些内容变化得更快。

比如，还是在第一个终端，我们运行下面的命令：

 复制代码

```
1 $ watch -d cat /proc/softirqs
2
3           CPU0      CPU1
4   HI:           0        0
5   TIMER:    1083906    2368646
6   NET_TX:       53         9
7   NET_RX:   1550643   1916776
8   BLOCK:        0         0
9   IRQ_POLL:     0         0
10  TASKLET:    333637    3930
11  SCHED:     963675    2293171
12  HRTIMER:     0         0
13  RCU:      1542111    1590625
```


通过 `/proc/softirqs` 文件内容的变化情况，你可以发现，`TIMER`（定时中断）、`NET_RX`（网络接收）、`SCHED`（内核调度）、`RCU`（RCU 锁）等这几个软中断都在不停变化。

其中，`NET_RX`，也就是网络数据包接收软中断的变化速率最快。而其他几种类型的软中断，是保证 Linux 调度、时钟和临界区保护这些正常工作所必需的，所以它们有一定的变化倒是正常的。

那么接下来，我们就从网络接收的软中断着手，继续分析。既然是网络接收的软中断，第一步应该就是要观察系统的网络接收情况。这里你可能想起了很多网络工具，不过，我推荐今天的主人公工具 `sar`。

`sar` 可以用来查看系统的网络收发情况，还有一个好处是，不仅可以观察网络收发的吞吐量（BPS，每秒收发的字节数），还可以观察网络收发的 PPS，即每秒收发的网络帧数。

我们在第一个终端中运行 `sar` 命令，并添加 `-n DEV` 参数显示网络收发的报告：

 复制代码

```
1 # -n DEV 表示显示网络收发的报告，间隔 1 秒输出一组数据
2 $ sar -n DEV 1
3 15:03:46      IFACE  rxpck/s   txpck/s   rxkB/s   txkB/s   rxcmp/s   txcmp/s  rxmcsst/
4 15:03:47      eth0  12607.00   6304.00   664.86   358.11     0.00     0.00     0.0
5 15:03:47    docker0   6302.00  12604.00   270.79   664.66     0.00     0.00     0.0
6 15:03:47        lo     0.00     0.00     0.00     0.00     0.00     0.00     0.0
7 15:03:47  veth9f6bbcd   6302.00  12604.00   356.95   664.66     0.00     0.00     0
```

对于 sar 的输出界面，我先来简单介绍一下，从左往右依次是：

第一列：表示报告的时间。

第二列：IFACE 表示网卡。

第三、四列：rxpck/s 和 txpck/s 分别表示每秒接收、发送的网络帧数，也就是 PPS。

第五、六列：rxkB/s 和 txkB/s 分别表示每秒接收、发送的千字节数，也就是 BPS。

后面的其他参数基本接近 0，显然跟今天的问题没有直接关系，你可以先忽略掉。

我们具体来看输出的内容，你可以发现：

对网卡 eth0 来说，每秒接收的网络帧数比较大，达到了 12607，而发送的网络帧数则比较小，只有 6304；每秒接收的千字节数只有 664 KB，而发送的千字节数更小，只有 358 KB。

docker0 和 veth9f6bbcd 的数据跟 eth0 基本一致，只是发送和接收相反，发送的数据较大而接收的数据较小。这是 Linux 内部网桥转发导致的，你暂且不用深究，只要知道这是系统把 eth0 收到的包转发给 Nginx 服务即可。具体工作原理，我会在后面的网络部分详细介绍。

从这些数据，你有没有发现什么异常的地方？

既然怀疑是网络接收中断的问题，我们还是重点来看 eth0：接收的 PPS 比较大，达到 12607，而接收的 BPS 却很小，只有 664 KB。直观来看网络帧应该都是比较小的，我们稍微计算一下， $664 \times 1024 / 12607 = 54$ 字节，说明平均每个网络帧只有 54 字节，这显然是很小的网络帧，也就是我们通常所说的小包问题。

那么，有没有办法知道这是一个什么样的网络帧，以及从哪里发过来的呢？

使用 tcpdump 抓取 eth0 上的包就可以了。我们事先已经知道，Nginx 监听在 80 端口，它所提供的 HTTP 服务是基于 TCP 协议的，所以我们可以指定 TCP 协议和 80 端口精确抓包。

接下来，我们在第一个终端中运行 tcpdump 命令，通过 -i eth0 选项指定网卡 eth0，并通过 tcp port 80 选项指定 TCP 协议的 80 端口：


```
1 # -i eth0 只抓取 eth0 网卡, -n 不解析协议名和主机名
2 # tcp port 80 表示只抓取 tcp 协议并且端口号为 80 的网络帧
3 $ tcpdump -i eth0 -n tcp port 80
4 15:11:32.678966 IP 192.168.0.2.18238 > 192.168.0.30.80: Flags [S], seq 458303614, win 512,
5 ...
```

从 tcpdump 的输出中, 你可以发现

192.168.0.2.18238 > 192.168.0.30.80 , 表示网络帧从 192.168.0.2 的 18238 端口发送到 192.168.0.30 的 80 端口, 也就是从运行 hping3 机器的 18238 端口发送网络帧, 目的为 Nginx 所在机器的 80 端口。

Flags [S] 则表示这是一个 SYN 包。

再加上前面用 sar 发现的, PPS 超过 12000 的现象, 现在我们可以确认, 这就是从 192.168.0.2 这个地址发送过来的 SYN FLOOD 攻击。

到这里, 我们已经做了全套的性能诊断和分析。从系统的软中断使用率高这个现象出发, 通过观察 /proc/softirqs 文件的变化情况, 判断出软中断类型是网络接收中断; 再通过 sar 和 tcpdump , 确认这是一个 SYN FLOOD 问题。

SYN FLOOD 问题最简单的解决方法, 就是从交换机或者硬件防火墙中封掉来源 IP, 这样 SYN FLOOD 网络帧就不会发送到服务器中。

至于 SYN FLOOD 的原理和更多解决思路, 你暂时不需要过多关注, 后面的网络章节里我们都会学到。

案例结束后, 也不要忘了收尾, 记得停止最开始启动的 Nginx 服务以及 hping3 命令。

在第一个终端中, 运行下面的命令就可以停止 Nginx 了:

```
1 # 停止 Nginx 服务
2 $ docker rm -f nginx
```

然后到第二个终端中按下 Ctrl+C 就可以停止 hping3。

小结

软中断 CPU 使用率（softirq）升高是一种很常见的性能问题。虽然软中断的类型很多，但实际生产中，我们遇到的性能瓶颈大多是网络收发类型的软中断，特别是网络接收的软中断。

在碰到这类问题时，你可以借用 sar、tcpdump 等工具，做进一步分析。不要害怕网络性能，后面我会教你更多的分析方法。

思考

最后，我想请你一起来聊聊，你所碰到的软中断问题。你所碰到的软中问题是哪种类型，是不是这个案例中的小包问题？你又是怎么分析它们的来源并解决的呢？可以结合今天的案例，总结你自己的思路和感受。如果遇到过其他问题，也可以留言给我一起解决。

欢迎在留言区和我讨论，也欢迎你把这篇文章分享给你的同事、朋友。我们一起在实战中演练，在交流中进步。



Linux 性能优化实战

10 分钟帮你找到系统瓶颈

倪朋飞

微软资深工程师
Kubernetes 项目维护者



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得转载

上一篇 09 | 基础篇：怎么理解Linux软中断？

下一篇 11 | 套路篇：如何迅速分析出系统CPU的瓶颈在哪里？

精选留言 (57)

写留言



倪朋飞 置顶

2018-12-12

18

统一回复一下终端卡顿的问题，这个是由于网络延迟增大（甚至是丢包）导致的。比如你可以再拿另外一台机器（也就是第三台）在 hping3 运行的前后 ping 一下案例机器，ping -c3 <ip>

hping3 运行前，你可能看到最长的也不超过 1 ms: ...

展开 ▼



2xshu

2018-12-12

9

老师，网络软中断明明只占了百分之四左右。为什么终端会感觉那么卡呢？不是很理解这点呢

作者回复: 参考置顶回复



赵强强

2018-12-12

7

倪老师，案例中硬中断CPU占用率为啥是0呢，硬中断和软中断次数不是基本一致的吗？



Days

2018-12-13

4

软终端不高导致系统卡顿，我的理解是这样的，其实不是系统卡顿，而是由于老师用的ssh远程登录，在这期间hping3大量发包，导致其他网络连接延迟，ssh通过网络连接，使ssh客户端感觉卡顿现象。

作者回复: 正解👍



卿卿子衿

2018-12-12

4

有同学说在查看软中断数据时会显示128个核的数据，我的也是，虽然只有一个核，但是会显

示128个核的信息，用下面的命令可以提取有数据的核，我的1核，所以这个命令只能显示1核，多核需要做下修改

```
watch -d "/bin/cat /proc/softirqs | /usr/bin/awk 'NR == 1{printf \"%13s %s\\n\\\", \"...
```

展开 ∨

作者回复: 谢谢分享



我来也

2018-12-12

👍 3

[D10打卡]

"hping3 -S -p 80 -i u100 192.168.0.30" 这里的u100改为了1 也没觉得终端卡,top的软中断%si倒是从4%上升了不少,吃满了一个cpu.

可能是我直接在宿主机上开终端的原因,本身两个虚拟机都在这个宿主机上,都是走的本地网络.本地网卡可能还处理的过来....

展开 ∨

作者回复: 👍

最后一个问题其实前面已经看到PPS了



向阳胜

2018-12-27

👍 2

搞运维好些年了。一些底层性能的东西，感觉自己始终是一知半解，通过这个专栏了解的更深入了，确实学到了很多。而且老师也一直在积极回复同学们的问题，相比某些专栏的老师发出来就不管的状态好太多。给老师点赞。

作者回复: 也很高兴看到大家有所收获😊



xfan

2018-12-12

👍 2

ssh的tty其实也是通过网络传输的，既然是经过网卡，当然会卡，这就是攻击所带来的结果

作者回复: 对的



Vicky🍊🔒...

2018-12-12

👍 2

1. 网络收发软中断过多导致命令行比较卡，是因为键盘敲击命令行属于硬中断，内核也需要去处理的原因吗？
2. 观察/proc/softirqs，发现变化的值是TIMER、NET_RX、BLOCK、RCU，奇怪的是SCHED一直为0，求老师解答

作者回复: 我们是SSH登陆的机器，还是走网络而不是键盘中断😓



黄海峰

2018-12-12

👍 2

这真是非常干货和务实的一个专栏，这么便宜，太值了。。。

作者回复: 😊



几叶星辰

2019-01-24

👍 1

怎么让网卡中断平衡呢，可以请教下linux 2.6.40。中断平衡问题吗，以及内核版本更高的版本？

作者回复: 配置 smp_affinity 或者开启 irqbalance 服务



Vicky🍊🔒...

2018-12-12

👍 1

执行了一下hping3，机器直接卡死了，登录不上去了，哈哈

作者回复: 可能太猛了，调整下参数再试试



chenjt

2018-12-12

👍 1

同问，这种情况下cpu使用率这么低，为什么会感到卡顿呢

作者回复: 参考置顶回复



bluefantasy...

2018-12-12



1

老师，既然软中断并没有占用太多cpu资源，为啥会影响其他任务的性能？

作者回复：参考置顶回复



zqing

2018-12-12



1

同问:老师，网络软中断明明只占了百分之四左右。为什么终端会感觉那么卡呢？不是很理解这点呢

作者回复：参考置顶回复



Maxwell

2019-02-12



我用的是vmware虚拟机，网络连接是NAT,CPU是 I7 8750U 4C8T,运行案例场景时，并没有任何卡顿？

还有，我这边使用 sar命令查看的结果和你的差别很大是什么原因呢？

18时23分07秒 ens33 4462.00 2237.00 261.45 127.87 0.00 0.00 0.00 31.89



bruceding

2019-02-12



找网络相关的错误，可以有几种方式。

1. 找系统类的错误， `dmesg | tail`
2. 直接的网络错误 `sar -n ETCP 1` 或者 `sar -n EDEV 1`
3. 查看网络状态， `netstat -s` 或者 `watch -d netstat -s`
4. 网络状态的统计 `ss -ant | awk '{++s[$1]} END {for(k in s) print k,s[k]}'`



keyboard_ch...

2019-02-08



这个案例刚好是其他进程cpu利用率低，假如其他进程利用率比较高时，是不是比较难定位是软中断的问题了？



maoxiajun

2019-01-24



lesson 10打卡



李实健

2019-01-08



老师，我的电脑一点都不卡，core i5 5 代 cpu，MacBook Pro 2014 13 寸，环境 vmware fusion，两台 ubuntu 1804，iTerm ssh 到 a ubuntu 开启 docker 80 端口，iTerm ssh 到 b ubuntu 运行 hping3，a ubuntu 中断 30% 左右，所有数据跟文章里的一样，但终端就是不卡，一点也不卡...

作者回复: 看来是压力还不够大 试试ping的延迟是不是没多大变化? 也可以再增大压力试试