

## 33 | 关于 Linux 网络，你必须知道这些（上）

2019-02-06 倪朋飞



朗读：冯永吉

时长10:35 大小9.70M



你好，我是倪朋飞。

前几节，我们一起学习了文件系统和磁盘 I/O 的工作原理，以及相应的性能分析和优化方法。接下来，我们将进入下一个重要模块——Linux 的网络子系统。

由于网络处理的流程最复杂，跟我们前面讲到的进程调度、中断处理、内存管理以及 I/O 等都密不可分，所以，我把网络模块作为最后一个资源模块来讲解。

同 CPU、内存以及 I/O 一样，网络也是 Linux 系统最核心的功能。网络是一种把不同计算机或网络设备连接到一起的技术，它本质上是一种进程间通信方式，特别是跨系统的进程间通信，必须要通过网络才能进行。随着高并发、分布式、云计算、微服务等技术的普及，网络的性能也变得越来越重要。

那么，Linux 网络又是怎么工作的呢？又有哪些指标衡量网络的性能呢？接下来的两篇文章，我将带你一起学习 Linux 网络的工作原理和性能指标。

## 网络模型

说到网络，我想你肯定经常提起七层负载均衡、四层负载均衡，或者三层设备、二层设备等。那么，这里说的二层、三层、四层、七层又都是什么意思呢？

实际上，这些层都来自国际标准化组织制定的**开放式系统互联通信参考模型**（Open System Interconnection Reference Model），简称为 OSI 网络模型。

为了解决网络互联中异构设备的兼容性问题，并解耦复杂的网络包处理流程，OSI 模型把网络互联的框架分为应用层、表示层、会话层、传输层、网络层、数据链路层以及物理层等七层，每个层负责不同的功能。其中，

应用层，负责为应用程序提供统一的接口。

表示层，负责把数据转换成兼容接收系统的格式。

会话层，负责维护计算机之间的通信连接。

传输层，负责为数据加上传输表头，形成数据包。

网络层，负责数据的路由和转发。

数据链路层，负责 MAC 寻址、错误检测和纠错。

物理层，负责在物理网络中传输数据帧。

但是 OSI 模型还是太复杂了，也没能提供一个可实现的方法。所以，在 Linux 中，我们实际上使用的是另一个更实用的四层模型，即 TCP/IP 网络模型。

TCP/IP 模型，把网络互联的框架分为应用层、传输层、网络层、网络接口层等四层，其中，

应用层，负责向用户提供一组应用程序，比如 HTTP、FTP、DNS 等。

传输层，负责端到端的通信，比如 TCP、UDP 等。

网络层，负责网络包的封装、寻址和路由，比如 IP、ICMP 等。

网络接口层，负责网络包在物理网络中的传输，比如 MAC 寻址、错误检测以及通过网卡传输网络帧等。

为了帮你更形象理解 TCP/IP 与 OSI 模型的关系，我画了一张图，如下所示：



当然了，虽说 Linux 实际按照 TCP/IP 模型，实现了网络协议栈，但在平时的学习交流中，我们习惯上还是用 OSI 七层模型来描述。比如，说到七层和四层负载均衡，对应的分别是 OSI 模型中的应用层和传输层（而它们对应到 TCP/IP 模型中，实际上是四层和三层）。

TCP/IP 模型包括了大量的网络协议，这些协议的原理，也是我们每个人必须掌握的核心基础知识。如果你不太熟练，推荐你去学《TCP/IP 详解》的卷一和卷二，或者学习极客时间出品的[《趣谈网络协议》](#)专栏。

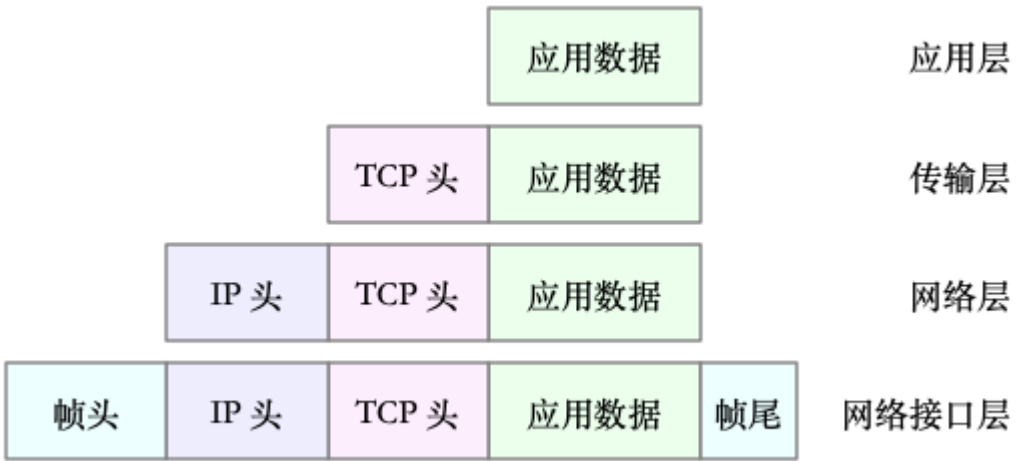
## Linux 网络栈

有了 TCP/IP 模型后，在进行网络传输时，数据包就会按照协议栈，对上一层发来的数据进行逐层处理；然后封装上该层的协议头，再发送给下一层。

当然，网络包在每一层的处理逻辑，都取决于各层采用的网络协议。比如在应用层，一个提供 REST API 的应用，可以使用 HTTP 协议，把它需要传输的 JSON 数据封装到 HTTP 协议中，然后向下传递给 TCP 层。

而封装做的事情就很简单了，只是在原来的负载前后，增加固定格式的元数据，原始的负载数据并不会被修改。

比如，以通过 TCP 协议通信的网络包为例，通过下面这张图，我们可以看到，应用程序数据在每个层的封装格式。



其中：

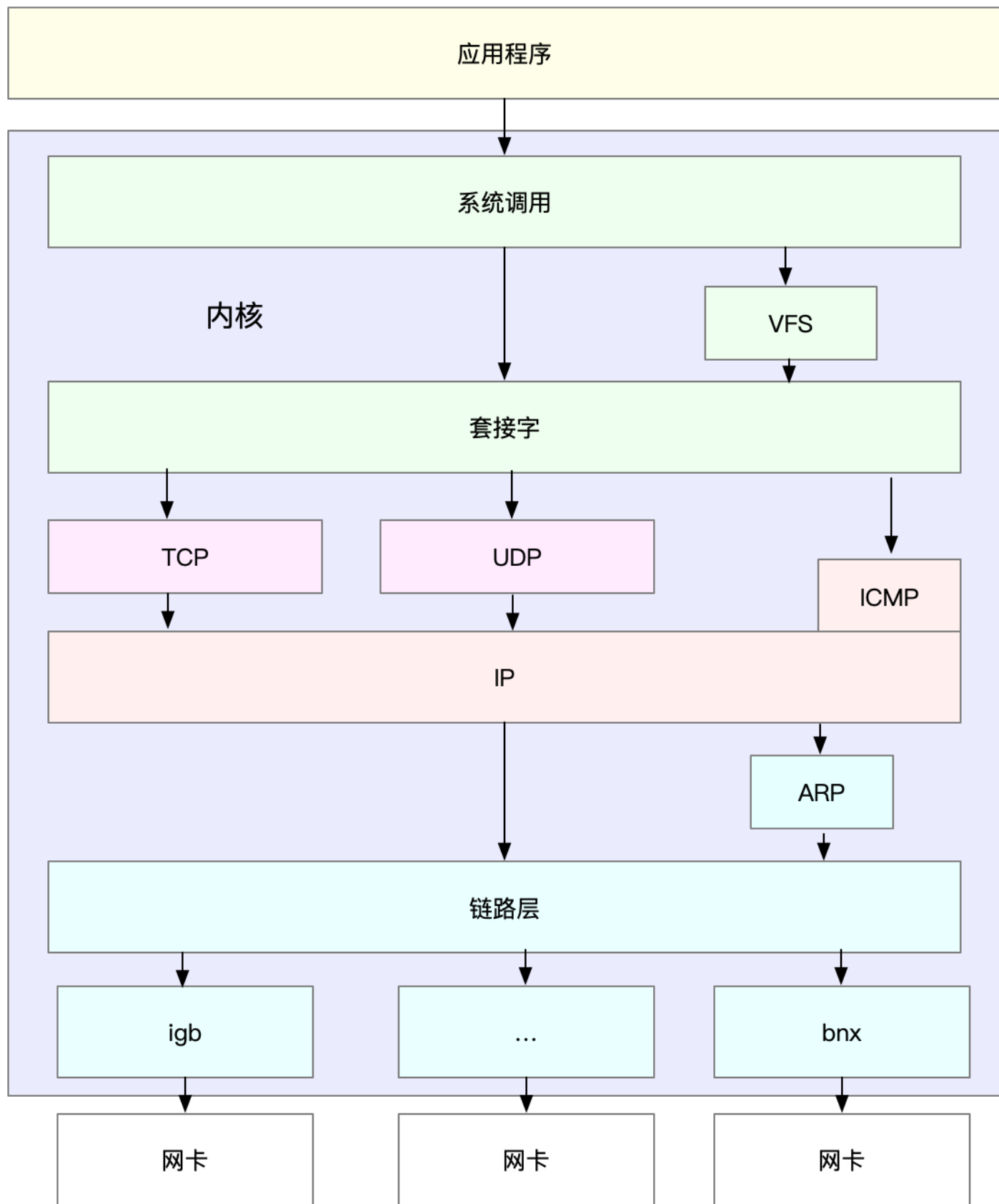
- 传输层在应用程序数据前面增加了 TCP 头；
- 网络层在 TCP 数据包前增加了 IP 头；
- 而网络接口层，又在 IP 数据包前后分别增加了帧头和帧尾。

这些新增的头部和尾部，都按照特定的协议格式填充，想了解具体格式，你可以查看协议的文档。比如，你可以查看[这里](#)，了解 TCP 头的格式。

这些新增的头部和尾部，增加了网络包的大小，但我们都知道，物理链路中并不能传输任意大小的数据包。网络接口配置的最大传输单元（MTU），就规定了最大的 IP 包大小。在我们最常用的以太网中，MTU 默认值是 1500（这也是 Linux 的默认值）。

一旦网络包超过 MTU 的大小，就会在网络层分片，以保证分片后的 IP 包不大于 MTU 值。显然，MTU 越大，需要的分包也就越少，自然，网络吞吐能力就越好。

理解了 TCP/IP 网络模型和网络包的封装原理后，你很容易能想到，Linux 内核中的网络栈，其实也类似于 TCP/IP 的四层结构。如下图所示，就是 Linux 通用 IP 网络栈的示意图：



(图片参考《性能之巅》图 10.7 通用 IP 网络栈绘制)

我们从上到下来看这个网络栈，你可以发现，

最上层的应用程序，需要通过系统调用，来跟套接字接口进行交互；

套接字的下面，就是我们前面提到的传输层、网络层和网络接口层；

最底层，则是网卡驱动程序以及物理网卡设备。

这里我简单说一下网卡。网卡是发送和接收网络包的基本设备。在系统启动过程中，网卡通过内核中的网卡驱动程序注册到系统中。而在网络收发过程中，内核通过中断跟网卡进行交互。

再结合前面提到的 Linux 网络栈，可以看出，网络包的处理非常复杂。所以，网卡硬中断只处理最核心的网卡数据读取或发送，而协议栈中的大部分逻辑，都会放到软中断中处理。

## Linux 网络收发流程

了解了 Linux 网络栈后，我们再来看看，Linux 到底是怎么收发网络包的。

注意，以下内容都以物理网卡为例。事实上，Linux 还支持众多的虚拟网络设备，而它们的网络收发流程会有一些差别。

### 网络包的接收流程

我们先来看网络包的接收流程。

当一个网络帧到达网卡后，网卡会通过 DMA 方式，把这个网络包放到收包队列中；然后通过硬中断，告诉中断处理程序已经收到了网络包。

接着，网卡中断处理程序会为网络帧分配内核数据结构（sk\_buff），并将其拷贝到 sk\_buff 缓冲区中；然后再通过软中断，通知内核收到了新的网络帧。

接下来，内核协议栈从缓冲区中取出网络帧，并通过网络协议栈，从下到上逐层处理这个网络帧。比如，

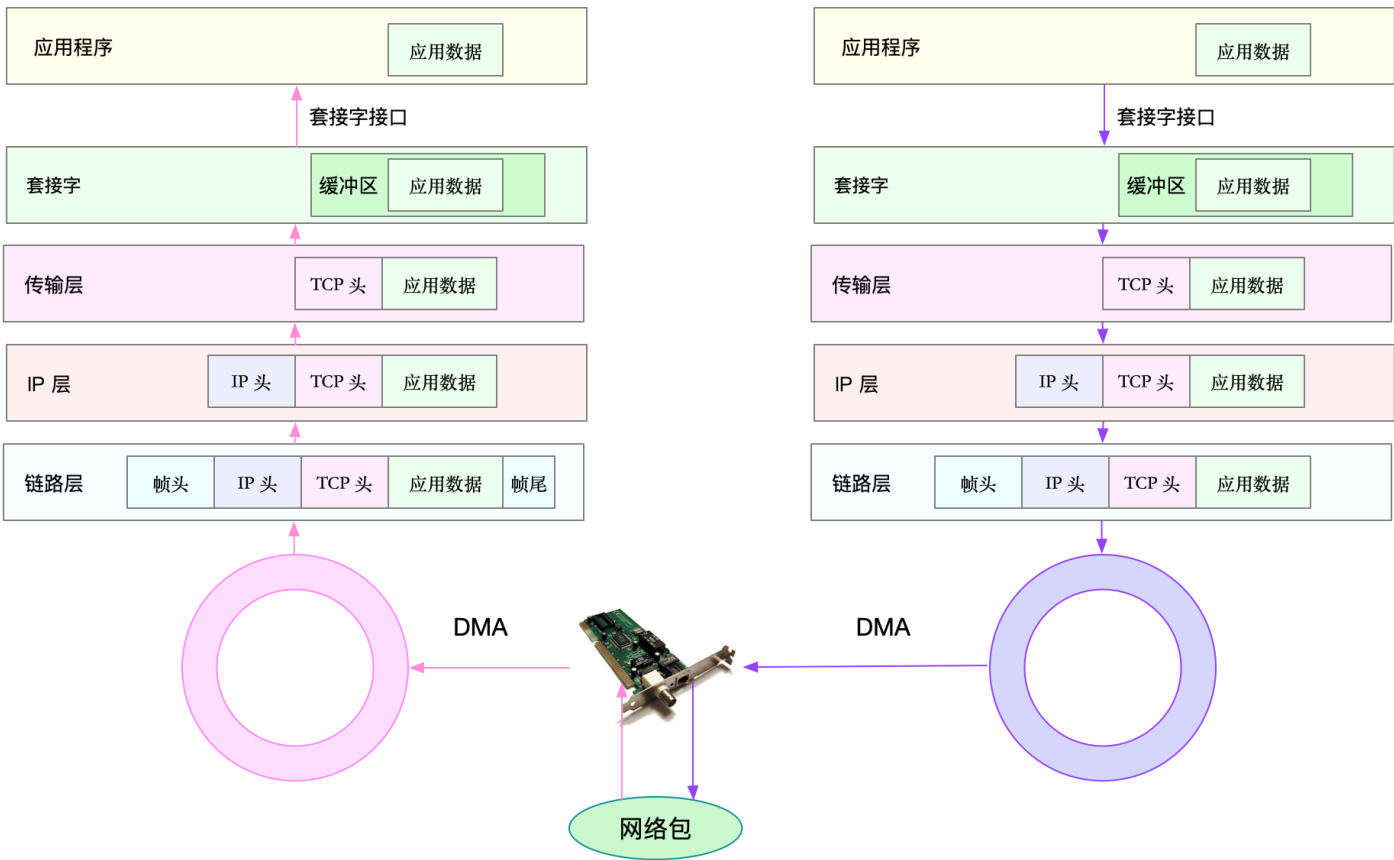
在链路层检查报文的合法性，找出上层协议的类型（比如 IPv4 还是 IPv6），再去掉帧头、帧尾，然后交给网络层。

网络层取出 IP 头，判断网络包下一步的走向，比如是交给上层处理还是转发。当网络层确认这个包是要发送到本机后，就会取出上层协议的类型（比如 TCP 还是 UDP），去掉 IP 头，再交给传输层处理。

传输层取出 TCP 头或者 UDP 头后，根据 < 源 IP、源端口、目的 IP、目的端口 > 四元组作为标识，找出对应的 Socket，并把数据拷贝到 Socket 的接收缓存中。

最后，应用程序就可以使用 Socket 接口，读取到新接收到的数据了。

为了更清晰表示这个流程，我画了一张图，这张图的左半部分表示接收流程，而图中的粉色箭头则表示网络包的处理路径。



## 网络包的发送流程

了解网络包的接收流程后，就很容易理解网络包的发送流程。网络包的发送流程就是上图的右半部分，很容易发现，网络包的发送方向，正好跟接收方向相反。

首先，应用程序调用 Socket API（比如 `sendmsg`）发送网络包。

由于这是一个系统调用，所以会陷入到内核态的套接字层中。套接字层会把数据包放到 Socket 发送缓冲区中。

接下来，网络协议栈从 Socket 发送缓冲区中，取出数据包；再按照 TCP/IP 栈，从上到下逐层处理。比如，传输层和网络层，分别为其增加 TCP 头和 IP 头，执行路由查找确认下一跳的 IP，并按照 MTU 大小进行分片。

分片后的网络包，再送到网络接口层，进行物理地址寻址，以找到下一跳的 MAC 地址。然后添加帧头和帧尾，放到发包队列中。这一切完成后，会有软中断通知驱动程序：发包队列中

有新的网络帧需要发送。

最后，驱动程序通过 DMA，从发包队列中读出网络帧，并通过物理网卡把它发送出去。

## 小结

在今天的文章中，我带你一起梳理了 Linux 网络的工作原理。

多台服务器通过网卡、交换机、路由器等网络设备连接到一起，构成了相互连接的网络。由于网络设备的异构性和网络协议的复杂性，国际标准化组织定义了一个七层的 OSI 网络模型，但是这个模型过于复杂，实际工作中的事实标准，是更为实用的 TCP/IP 模型。

TCP/IP 模型，把网络互联的框架，分为应用层、传输层、网络层、网络接口层等四层，这也是 Linux 网络栈最核心的构成部分。

应用程序通过套接字接口发送数据包，先要在网络协议栈中从上到下进行逐层处理，最终再送到网卡发送出去。

而接收时，同样先经过网络栈从下到上的逐层处理，最终才会送到应用程序。

了解了 Linux 网络的基本原理和收发流程后，你肯定迫不及待想知道，如何去观察网络的性能情况。那么，具体来说，哪些指标可以衡量 Linux 的网络性能呢？别急，我将在下一节中为你详细讲解。

## 思考

最后，我想请你来聊聊你所理解的 Linux 网络。你碰到过哪些网络相关的性能瓶颈？你又是怎么样来分析它们的呢？你可以结合今天学到的网络知识，提出自己的观点。

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 Linux 性能优化专栏加餐（一）

下一篇 34 | 关于 Linux 网络，你必须知道这些（下）

## 精选留言 (10)

💬 写留言



空空

2019-02-08

👍 1

老师过年好！

曾经在Linux3.10测试netlink收发包效率，发现一个问题，正常情况下每收一个包大概需要10us，但是每隔8秒会出现一次收包时间30-50ms，就是因为固定间隔8秒会出现一次收包时间过长，导致收包效率降低。请教一下老师每隔8秒系统会做什么？或者是因为什么系统配置？希望老师解答一下疑惑，谢谢！



陈云卿

2019-02-06

👍 1

系统出口带宽被打满，导致大量请求超时

作者回复: DDoS 😊



Penn

2019-02-06

1

中断不均，连接跟踪打满

作者回复: 嗯 这是最常见的两个问题



vvccoe

2019-02-12



老师，新年快乐。

之前忘了在哪里读的，关于分片的说法，和你在文章中的说法，有一点差异，如果是TCP连接，在三次握手中的前两次会带一个MSS值，MSS值来表示接收单包最大量，MSS值根据MTU计算出来。以避免在IP层分片，IP层分片的缺点在于如果丢失一个包，会导致所有分片包重传。...

展开 ▼



Anker

2019-02-12



网络报文传需要在用户态和内核态来回切换，导致性能下降。业界使用零拷贝或intel的dpdk来提高性能。



Orcsir

2019-02-11



Flag

2019/02/11



我来也

2019-02-08



[D33打卡]

可能是公司业务规模不大，峰值带宽未超过100m。以前遇到的网络性能问题不太多，现在都是云服务器了，理论上出口带宽可以动态调整，遇到的问题就更少了。

但自从阿里云切到腾讯云后，还真遇到了一个对我来说无解的网络问题。

云服务器控制台经常提示带宽已达峰值，但上控制台观察，只能查到最小粒度10s的流量图...

展开 ▼



J

2019-02-07



先拜年，再问问题:)。好多时候性能问题是由于网络造成的，后续应该有些相关案例分析吧。从centos7.4升级到7.5之后，网卡busy-poll: off 被设置成off了。如果用tuned-profile network-latency, 就会有很大的性能下降。必须把里面的net.core.busy\_read net.core.busy\_poll 设置为0关闭。不太明白这个内核变化的原因以及性能下降的原因。

---



道无涯

2019-02-06



本机有两张网卡，如果路由配成第二张网卡也走第一张网卡，会不会导致第二张网络收不到数据？

---



威

2019-02-06



老师您好，请问最后一张图下方的两个大圈圈代表的是什么意思，是代表loop吗

作者回复: ring buffer