Linux系统针对网卡中断的优化处理

中断：

当网卡接收到数据包后，会触发硬中断，通知CPU来收包。硬中断是一个CPU和网卡交互的过程。这其实会消耗CPU资源。特别是在使用速度极快的万兆网卡之后，大量的网络交互使得CPU很大一部分资源消耗在网卡中断处理上。此时，瓶颈并不在网卡，而是在CPU上。因此，现在的网卡都采用多队列的技术，用于充分利用多核心CPU。

中断的详细解释：《Linux的中断和异常扫盲笔记》

SMP IRQ affinity

为了防止多个设置发送相同的中断, Linux设计了一套中断请求系统, 使得计算机系统中的每个设备被分配了各自的中断号, 以确保它的中断请求的唯一性. 从2.4 内核开始, Linux改进了分配特定中断到指定的处理器(或处理器组)的功能. 这被称为SMP IRQ affinity, 它可以控制系统如何响应各种硬件事件. 允许你限制或者重新分配服务器的工作负载, 从而让服务器更有效的工作..。

以网卡中断为例，在没有设置SMP IRQ affinity时， 所有网卡中断都关联到CPU0, 这导致了CPU0负载过高，而无法有效快速的处理网络数据包，导致了瓶颈。 通过SMP IRQ affinity， 把网卡多个中断分配到多个CPU上，可以分散CPU压力，提高数据处理速度。

RPS和RFS

RPS（Receive Packet Steering）

RFS主要是把软中断的负载均衡到各个cpu，简单来说，是网卡驱动对每个流生成一个hash标识，这个HASH值得计算可以通过四元组来计算（SIP，SPORT，DIP，DPORT），然后由中断处理的地方根据这个hash标识分配到相应的CPU上去，这样就可以比较充分的发挥多核的能力了。通俗点来说就是**在软件层面模拟实现硬件的多队列网卡功能**，如果网卡本

身支持多队列功能的话RPS就不会有任何的作用。该功能主要针对单队列网卡多CPU环境，如网卡支持多队列则可使用SMP irq affinity直接绑定硬中断。

RFS(Receive Flow Steering)

由于RPS只是单纯的把同一流的数据包分发给同一个CPU核来处理了，但是有可能出现这样的情况，即给该数据流分发的CPU核和执行处理该数据流的应用程序的CPU核不是同一个：数据包均衡到不同的cpu，这个时候如果应用程序所在的cpu和软中断处理的cpu不是同一个，此时对于cpu cache的影响会很大。

这时候就需要RFS来配合使用了，这也是Tom提交的内核补丁，它是用来配合RPS补丁使用的，是RPS补丁的扩展补丁，**它把接收的数据包送达应用所在的CPU上，提高cache的命中率**。这两个补丁往往都是一起设置，来达到最好的优化效果, 主要是**针对单队列网卡多CPU环境**。