Linux 多核环境网卡驱动优化研究①

姚萌萌, 张 俊, 沈 亮

(无锡江南计算技术研究所, 无锡 214083)

摘 要: Linux 多核环境下网卡驱动是影响网络性能的重要因素. 网卡接收数据包首先通过硬件中断响应, 然后NAPI机制调度, 启用软件中断把数据包向网络层传递. 通过分析网卡接收数据的过程, 对网卡驱动做了优化, 做了相应实验并对结果数据进行分析, 在一定程度上提高了网络性能, 达到预期效果.

关键词: 多核环境; 网卡驱动; 中断; NAPI

Network Driver Adapter Optimization in Linux Multi-Core Environment

YAO Meng-Meng, ZHANG Jun, SHEN Liang

(Wuxi Jiangnan Inst. of Computing Techology, Wuxi 214083, China)

Abstract: In Linux multi-core environment, network driver adapter is an important factor that affects network performance. When receiving a data package, network driver adapter responses with hardware interrupt firstly. Then it schedules with NAPI mechanism and uses software interrupt to forward the data package upward to the network layer. By analyzing to know the data receiving process of network driver adapter, this paper tries to optimize the network driver adapter. Furthermore, it designs appropriate experiments, and the experiment result approves the network performance is improved to a certain extent.

Key words: multi-core environment; network driver adapter; interrupt; New API

0 引言

目前, 多核处理器在网络设备上应用的越来越广泛, Linux 作为通用的操作系统被广泛采用, Linux 自带的 SMP 技术已经可以对多核处理器有良好的支持, 系统性能也得到了很大提升. 但是, 传统的网卡驱动在多核处理器的应用环境下不能发挥多核的性能. 多核环境下, 核的数量增加并没有带来预期的网络吞吐的线性提升. 本文从如何提升多核环境下网络吞吐性能目的出发, 研究分析网卡驱动实现机制, 优化网卡驱动以充分发挥多核处理器性能, 通过实验验证数据分析, 优化后的网卡驱动使得多核处理器的网络转发性能得到了极大的提升.

1 通用网卡驱动分析

1.1 网卡驱动中的硬中断与软中断

网卡驱动向内核请求中断号, 并注册响应中断的理函数. 网卡接收数据触发硬中断, 网络数据的处理通过中断函数来实现. 在一个 cpu 或单核的环境下, cpu 不能执行其它进程, 完全属于该中断处理函数, 而且不能被抢占[1]. 内核做非可抢占设计以及等待被 cpu 服务的进程, 就会对网络性能有潜在的严重影响. 以前的网卡驱动中, 每来一帧数据触发一次硬中断, 在低负载下可以保证低延时. 但是在高流量负载网络环境中, 网络性能比较低. 现在的很多网卡驱动程序, 都是在中断期间处理多帧. 过多的触发硬中断, 会降低网络性能[2].

网络数据通过软中断处理. 软中断会轮询当前 cpu 数据结构 softnet_data 列表中有数据的设备, 然后调度响应设备驱动中注册的轮询函数, 处理数据并把数据包向上层传递. 多核的 cpu 环境, 每个核都有一个内核线程 ksoftirqd, 该线程是个后台线程, 在内核引导的时候启动. ksoftirqd 负责处理软中断, 它循环读

①收稿时间:2014-02-19;收到修改稿时间:2014-04-15

取函数 local_sofirq_pending 的返回值, 并进行相应处理. 如果本地 CPU 有一个未处理的软中断, local_sofirq_pending 返回 true. 最后, ksoftirqd 调用函数 do_softirq 处理软中断.

1.2 NAPI

本文网卡驱动使用NAPI机制. NAPI在Linux 2.5版本内核中引入, NAPI 混合了中断和轮询, 可以大幅减少 cpu 负载, 在高流量负载环境下网络性能得到了很大提升. NAPI 主要有两个优点[2]:

1) NAPI 在硬中断中处理多帧, 和以前的驱动中的处理方式一帧触发一次硬件中断相比, 减少了硬中断的次数, 减少 cpu 负载.

2) 如果设备的入口队列中有数据, NAPI 会以相当公平的循环方式予以访问. 确保其它设备负载很高时, 低流量设备的延时在接受的范围内.

NAPI 机制调度的时候会调用函数 __raise_softirq_irqoff(NET_RX_SOFTIRQ), 然后调用软中断注册的函数 net_rx_action, net_rx_action 函数调用网卡注册的 poll 函数, 过程如图 1:

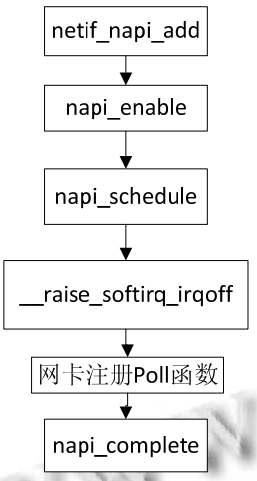

图 1　NAPI 驱动注册流程

NAPI 机制网卡接受数据过程如图 2 所示.

网卡注册的 poll 函数是由软中断处理流程调用, 主要功能就是从网卡接收队列中读取数据, 初始化 skb_buff,调用 netif_receive_skb 函数向上层传递数据包[3].

poll 函数处理多帧数据, 处理的帧数由 weight 决定, 本文使用默认值 32. 从上图可以看出, 硬件中断函数调用后, 会把该中断屏蔽, 然后 NAPI 调度, 触发软中断, 当处理完 weight 个帧时, 使能硬中断. 在整个处理的过程中, 尽管网卡接收队列中有数据包, 但是不会去处理. 这个过程是非抢占的.

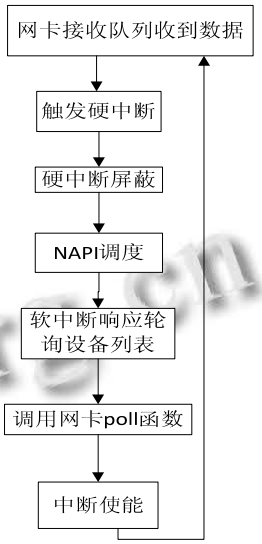

图 2　NAPI 收包流程图

2　网卡驱动优化设计

2.1 通用处理模式分析

Linux 开启 SMP 功能运行在多核处理上, 网卡收包时, 通常采用单硬中断配合单软中断的方式进行[4]. 该非抢占模式下网卡处理数据流是以单进程方式进行, 难以发挥多核的并行性能.

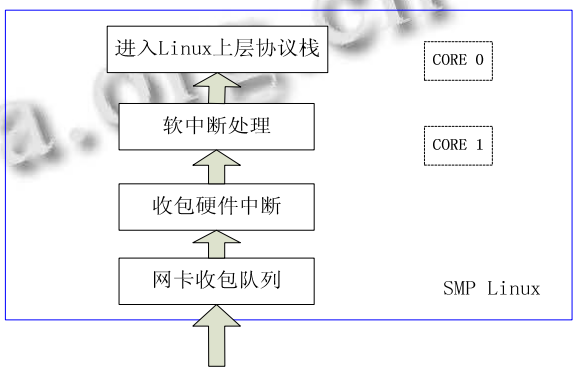

图 3　多核环境下传统网卡驱动处理模式

采用 Cavium 的 CN5020 的评估板进行实验验证, 该处理器主频为 400Mhz, 含有 2 个 CPU 核心, 启动 Linux 系统时可以通过参数设置选择启动的 CPU 核心数量, 该平台具有三个千兆以太网口(eth0、eth1、eth2), 实验以 eth0 和 eth2 为测试网口进行. 实验的拓扑如图 4 所示, 采用两台测试微机分别连接测试平台的两个

网口进行转发的性能测试.

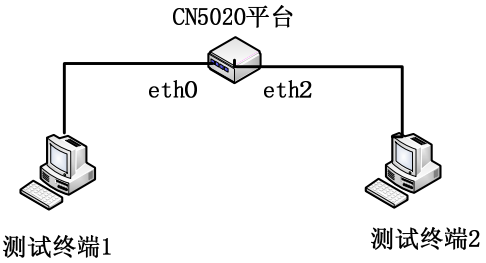

图 4 测试验证拓扑

本文测试软件使用 BWTest. 在一台测试终断微机上启动 BW 服务端, 另一台测试终端微机上启动 BW 客服端, 进行 TCP 的数据收发, 测试网络的性能.

经过性能测试, 发现 CN5020 平台 Linux 系统运行在单核和双核两种情况下, 网络性能相差不大, 吞吐量为 233MKb 左右. 本文测试过程中 netfilter 开启.

当 Linux 运行在多核上进行性能测试时, 通过查看设备状态后发现, 3 个网口共用同一个硬中断和软中断, 软中断在 Core1 上响应. ksoftirqd/0 进程 cpu 使用率为 0, 而 ksoftirqd/1 进程 cpu 占用率 70%左右. 可以看出, 内核只在 Core1 上处理软中断, Core0 上几乎不做任何处理.

性能测试如下图:

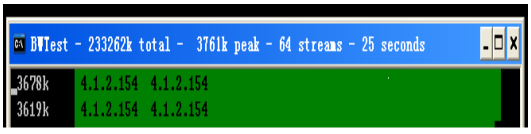

图 5 通用网卡驱动性能测试图

中断数量如下表:

表 1 通用网卡驱动中断统计表

网口	Core0 硬中断次数	Core 1 硬中断次数	BW 运行时间
eth0/eth2	14230	0	25S

2.2 网卡驱动的优化设计方式一

Linux 开启 SMP 功能运行在多核处理上, 网卡收包时, 采用单硬件中断配合多软件中断的方式进行, 软件中断的线程数量根据 CPU 核的数量进行分配.

如图 6 所示, 通过检测 CPU 的核心数量并行启动多个软中断进行同时处理, 例如 CN5020 平台具有两个 CPU 核心, 同时注册两个软中断服务程序, 两个核并行执行软中断. 这个过程在网卡注册的 poll 函数中. 其过程如下: 接收一个帧时, 读取相关寄存器获取网卡队列中的帧数. 如果网卡队列中帧数积压过多, 并且当前还有其它核可以用, 则调用函数 smp_call_function_single, 通过 IPI(处理器之间中断)启用另外一个核进行 NAPI 调度, 两个核同时处理网卡队列上数据. 其详细过程如图 7 所示. 这样将数据包的处理流程从单进程优化成多进程并行处理.

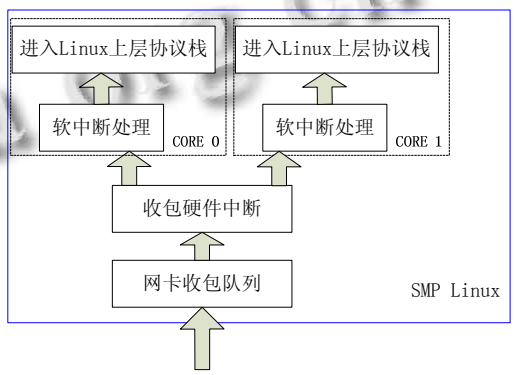

图 6 优化设计方式一

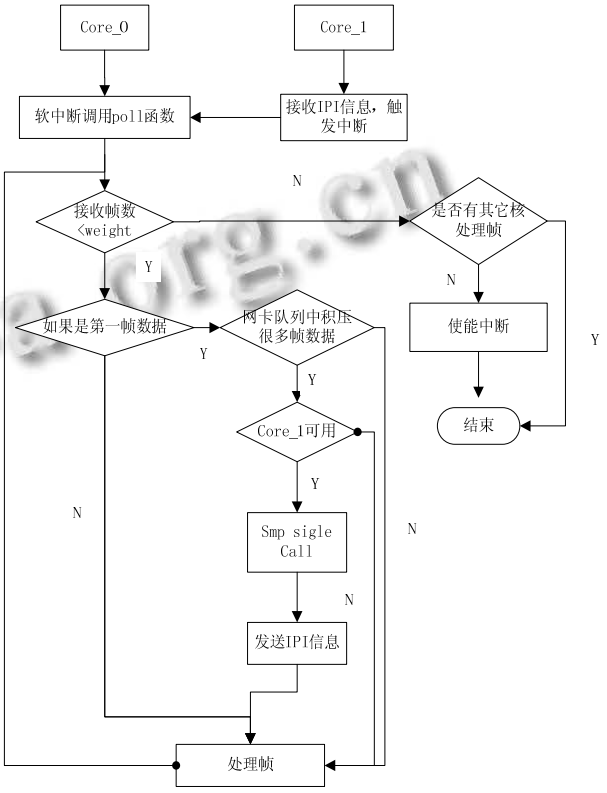

图 7 优化设计一详细过程

通过将驱动进行优化后, 按照优化前同样的实验

拓扑环境进行测试, 测试结果如下:

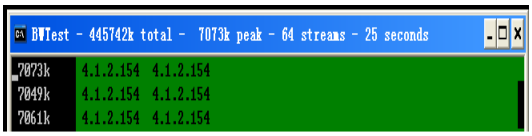

图 8　优化设计一的性能测试结果

表 2　优化设计一得中断统计表

网口	Core 0 硬中断次数	Core 1 硬中断次数	BW 运行时间
eth0/eth2	14770	0	25S

通过测试数据可以看出双核环境下网络性能得到了大幅度提升, 由于吞吐的增加, 硬件中断次数也有所增加.

2.3 网卡驱动的优化设计方式二

Linux 开启 SMP 功能运行在多核处理器上, 网卡收包时, 采用多硬件中断配合多软件中断的方式进行, 硬件中断的数量根据不同的网口绑定到不同的核上的方式进行分配[5], 软件中断的线程根据硬件中断的分配方式进行同步多核分配.

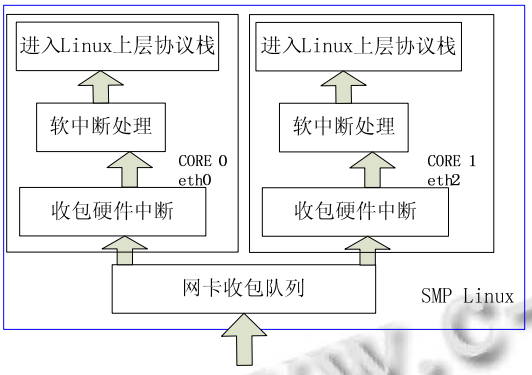

图 9　优化设计方式二

如图所示, 将两个网口的硬中断绑定到两个核上分别进行响应[6]. 两个核同时处理中断函数.

按照此前同样的实验拓扑环境进行测试, 测试结果如下:

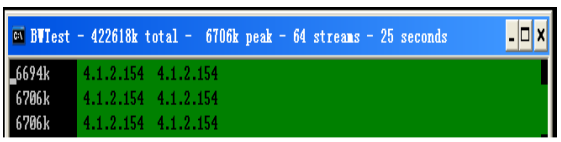

图 10　优化设计二的性能测试结果

表 3　优化设计二得中断统计表

网口	Core 0 硬中断次数	Core 1 硬中断次数	BW 运行时间
eth0	10891	0	25S
eth2	0	10684	25S

通过测试数据可以看出优化方式二网络性能比通用模式下也得到了大幅度提升, 由于硬件中断的并行设计, 硬件中断次数大幅度增加. 由于中断次数增加比较多, 处理中断的 CPU 资源消耗增加, 在该平台上的性能的提升相比优化方式一略微有所下降.

3　实验数据对比分析

本文提供的两种优化方式核心思路都是通过优化 Linux 内核网络收包处理流程进一步提升在多核环境下的网络处理性能. 分别对优化前、优化方式一以及优化方式二进行数据对比分析.

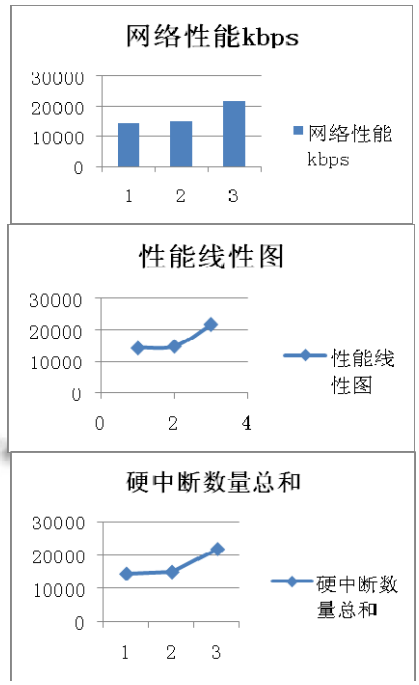

图 11　优化设计后的性能和优化前数据对比

通过图表可以看出网络驱动进行优化方式一采用共享硬中断和并行软中断的设计后, 改进后性能和 CPU 核心的数量接近线性增长. 在 CN5020 的平台下, 性能为原网络驱动的 1.91 倍.

网络驱动进行优化方式二采用多核多硬中断配合多软中断的设计后, 改进后的性能为原驱动的 1.81 倍.

理论上采用方式二进行优化设计的多核并行度更加高, 但实际验证测试中性能反而比方式一有所下降. 分析原因, 由于内外网在两个核上各占一个中断号, 反而产生了大量的硬中断和软中断, 硬中断数量总和是方式一的 1.41 倍, 中断的切换占用了大量的 CPU 资源, 反而降低了 CPU 的网络处理性能. 方式一在通过多路软件中断提升 CPU 核心并行处理的同时共享了硬中断资源, 硬中断的数量总和是优化前的 1.03 倍. 因而在性能提升中, 表现最佳.

4 结语

经过实验验证, 通过本文提供的两种网卡驱动方式可以极大地提升在 Linux 多核环境下网络转发处理性能. 随着多核处理器的广泛流行, 通过网络驱动的优化设计可以真正发挥多核处理器的网络性能. 在优化的过程中发现, 一方面良好的并行度可以很好的提升性能, 另一方面网卡在处理数据帧的时候中断响应次数也同时是影响网络性能的一个重要因素. 更好的并行度和更少的中断资源消耗存在一个平衡点, 在多次实验中得到比较好的方案. 在多核多网口的设备中, 除了分析硬中断、软中断的执行过程外, 也需要多次实验已得到最佳的解决方案, 使能网络性能在多核环境下发挥最佳性能.

参考文献

1 Daniel P, Marco Cesati. 深入理解 linux 内核(第 3 版).北京: 中国电力出版社,2007.

2 Benvenuti C.深入理解 linux 网络技术内幕.北京:中国电力出版社,2007.

3 周敬琼,周凤星.基于 ARM 的 Linux 网络设备驱动程序开发. 计算机工程与设计,2009,30(22):5124–5127.

4 毕学尧,刘宝旭,许榕生.Linux/SMP 体系网络处理性能研究.计算机工程,2003,22(29):126–127.

5 Cavium Networks. CN50XX-HM-v1.0P. [2009-07-07]

6 彭海运,李亚.基于 Linux 的多网卡负载均衡技术.实验室研究与探索,2012,31(9):77–78.