doi:10.11887/j.cn.201903007 http://journal.nudt.edu.cn

多核网络分组处理系统的数据分段卸载发送机制*

杨 惠[1],李 韬[1],吕高锋[1],全 巍[1],戴幻尧[2]
(1. 国防科技大学 计算机学院,湖南 长沙 410073;2. 中国洛阳电子装备试验中心,河南 洛阳 471003)

摘 要:为摆脱对商用网卡的依赖,降低软硬件复杂度,提出通用多核网络分组处理系统,构建面向大报文高速分组转发应用的软硬件协同数据分段卸载发送机制,并实现原型系统。该机制基于轻量级输入输出的软硬件协同多核分组处理系统,以降低大报文切分、拷贝开销以及软硬件复杂度为目的,把实现切分报文、封装报文头以及校验功能中硬件实现复杂的部分卸载到驱动中,将分段报文数据拷贝缩减为新报文头的拷贝,结合链式直接内存存取技术,为多核实现高速的大报文分组转发提供有效的解决方案。基于国产通用多核和高性能现场可编程门阵列平台进行发送性能测试。测试结果表明:采用数据分段卸载发送机制能大幅提升报文发送性能,有效解决大报文引发的多核网络分组处理性能下降的问题。

关键词:大报文;分组转发;多核;软硬件协同
中图分类号:TP393 **文献标志码**:A **文章编号**:1001-2486(2019)03-036-06

Packet segment offloading and sending mechanism based on multi-core packet processing system

YANG Hui[1], *LI Tao*[1], *LYU Gaofeng*[1], *QUAN Wei*[1], *DAI Huanyao*[2]
(1. College of Computer, National University of Defense Technology, Changsha 410073, China;
2. Luoyang Electronic Equipment Test Center of China, Luoyang 471003, China)

Abstract: The packet segment offloading and sending mechanism was proposed and the prototype system was implemented. This mechanism, based on the multi-core packet processing system with lightweight input/output, was aimed to reduce the cost of large packet segmentation, copy overhead, hardware and software complexity. It offloaded the complex parts of hardware implementation to the drive including packet segmentation, packet head encapsulation. It reduced the whole data copy cost to the copy cost of the new packets header, combined the chain direct memory access technology to provide an effective solution of packet forwarding on the multi-core system. The packet sending performance based on domestic general multi-core and field-programmable gate array platform was tested. The experimental data shows that the packet segment offloading and sending mechanism can significantly enhance the performance, and effectively solve the problem of the large packet sending on the multi-core network processing system.

Keywords: large packet; packet forwarding; multi-core; hardware-software co-design

随着互联网规模的不断扩大和新兴网络技术的应用,不断增长的网络流量对网络核心设备的处理能力提出了更高要求。多核处理器计算性能高和软件编程灵活性强的优势,使得大量的软件路由器和软件交换机在多核平台上被部署。具备高可编程性的通用多核处理器是网络设备中广泛采用的数据平面处理核心器件[1-5]。然而,传统网卡只能支持最大分段长度(Max Segment Size, MSS)大小的数据传输,当请求大量数据时,传输控制协议(Transmission Control Protocol, TCP)发送方必须将大块数据拆分成MSS大小的数据块,然后进一步封装为数据包形式,以便最终在网络中进行传输。由于多核处理器需要对每个分段进行处理,降低了其处理效率。TCP分段卸载(TCP Segment Offload, TSO)技术的提出[5],利用网卡分割大数据包,降低中央处理器(Central Processing Unit, CPU)发送数据包的负载,从而支持大报文的直接发送,报文的切分与校验等分组深度处理全部交给硬件实现。然而,基于TSO技术的网络分组深度处理需要软硬件的协同工作,在软硬件协同分组处理流程中,软硬件的通信开销过大会严重影响系统进行分组深度处理的性能。另外,报文的拆分校验等处理流程全部交由硬件实现,硬件复杂度高。

* **收稿日期**:2018-03-28
基金项目:国家自然科学基金资助项目(61702538);国防科技大学科研计划资助项目(ZK17-03-53)
作者简介:杨惠(1987—),女,安徽萧县人,助理研究员,博士,E-mail:huihui19870124@126.com

本文通过分析基于多核的大报文发送流程中的软硬件各个部分的开销,消除和弱化大报文发送的性能瓶颈,基于软硬件协同的轻量级分组输入/输出(Input/Output,I/O)技术和支持大报文发送的传统 TSO 技术,提出了一种面向高速分组转发的数据分段卸载发送机制。

1　相关研究

TSO 技术支持 TCP 发送方 CPU 直接将大块数据(最大支持 64 KB 大小)交给网络设备处理,由网络设备进行 TCP 段的分割,将一部分 CPU 的处理工作转移到网卡,从而减少 CPU 必须处理的数据包数量,达到提高网络处理性能的目的。支持 TSO 技术的网卡,需要支持 TSO 和分散 - 聚集(Scatter-Gather, SG)技术,以及 TCP 校验和计算功能,由网卡驱动或网卡硬件完成报文分段和 TCP 校验和计算功能,因而 TSO 技术需要网络设备驱动或者网络设备提供报文分段功能,对于网络设备的要求较高,软硬复杂度高。更为通用的分段卸载(General Segment Offload, GSO)技术将大报文分段的时机推迟到将数据报文提交给网络设备驱动之前完成,并且支持 TCPv4 之外的其他协议类型,如 TCPv6、UDP 和 DCCP 等。该技术需要网络设备支持通用 GSO 和 SG 功能,性能提升效果比 TSO 技术低。

Scatter-Gather 是一种与非连续物理地址传输的块直接内存存取(Direct Memory Access, DMA)[6]方式相对应的 DMA 方式。它通过一个链表描述物理不连续的内存地址,将链表首地址送往 DMA 控制器。DMA 控制器传输完一块物理连续的数据后,不发中断,根据链表记录内容传输下一块物理不连续的数据,直到链表中所有描述符内容传输完成发起一次中断。网络设备支持该技术需要支持从多个不同区域获取报文数据并且组装在一起。

为了减少报文处理过程中软硬件交互中断的代价,Packetshader 采用大报文缓冲区的方式,静态地预分配两个大的缓冲区(skb 控制信息缓冲区和分组数据缓冲区),通过连续存储每个接收分组的 skb 控制信息和分组数据,避免缓冲区申请/释放以及描述符的转换操作,有效降低分组 I/O 开销和访存开销。[3]面向高速分组转发提出的自描述缓冲区 (Self-Described Buffer, SDB)管理机制[7-8],将描述符、skb 控制信息以及分组数据连续存储在一个缓冲区中,大大降低系统的缓冲区管理开销,实现了无中断的报文传输,但所有报文在发送给网络设备之前,都要拷贝到 SDB 管理的固定缓冲区中,拷贝代价高,成为制约大报文发送性能的瓶颈。

2　支持轻量级 I/O 的软硬件协同分组处理系统

2.1　系统描述

多核轻量级的分组 I/O 技术,是一种低开销的分组处理软硬件通信机制,通过缓冲区管理卸载技术等,实现分组的零拷贝、无中断的下发,从而降低分组在软硬件的通信开销[9-10]。支持轻量级 I/O 的软硬件协同分组处理系统[11]将包含控制信息的描述符和分组数据连续存储在多核共享的地址连续的存储缓冲区中。共享缓冲区不再采用核调度的软件管理方法,而是卸载到专用网络加速引擎上由硬件管理。在系统初始化时,将共享缓冲区的描述符填入硬件管理的空闲描述符块队列中;当数据报文到达,根据空闲描述符队列存储的地址,分配一个空闲描述符,与数据报文组装好通过外设组建高速互联(Peripheral Component Interconnect Express, PCIE)送往共享缓冲区对应的地址空间;而数据报文从共享缓冲区地址空间下发时,如图 1 所示,首先为要发送的报文获取一个共享缓冲区区域,将报文内容直接由 skb - > data 指向的线性缓冲区拷贝到共享缓冲区中,完成拷贝后,将 skb 释放,构造一个发送描述符控制块,通知专用网络加速引擎有新的报文需要发送,专用网络加速引擎根据发送描述符控制块指示的报文 DMA 地址,读取共享缓冲区中的报文内容,完成报文发送。专用网络加速引擎硬件回收对应的描述符,并重新进入空闲描述符队列,等待新的数据报文使用。该共享缓冲区以块为单位组织、申请和释

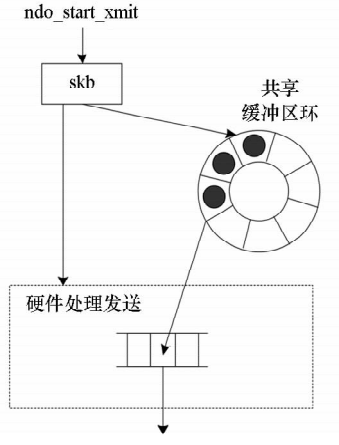

图 1　支持轻量级 I/O 的报文发送机制

Fig. 1　Packet sending mechanism with lightweight I/O

放,解放了核对共享缓冲区地址的管理。描述符和报文块以单向链表的形式,在共享缓冲区内组织。因此,系统在处理分组时,只要空闲描述符队列具有空闲描述符,数据报文即可上传至共享缓冲区,等待 CPU 核处理,无须中断响应,也无须多次访存缓冲区。

2.2 瓶颈分析

为对报文发送开销进行分析,做出如表 1 所示的假设。

表 1 发送分段开销假设

Tab. 1 Assumption of the packet sending overhead

开销分类	名称	开销代价
系统调用处理开销	N_a	每 1.5 KB 大小报文处理开销设为 N_a
协议栈处理开销	N_S	每 1.5 KB 大小报文处理开销设为 N_S;协议栈处理 1.5 KB 报文和 64 KB 报文,其处理开销相等,均为 N_S
网卡驱动处理开销	N_D	每 1.5 KB 大小报文处理开销设为 N_D;不拷贝的情况下,网卡驱动处理 1.5 KB 报文和 64 KB 报文,其处理开销相等,均为 N_D
网卡报文拷贝开销	N_c	每 1.5 KB 大小报文处理开销设为 N_c
DMA 报文开销	$N_{DMA\text{-}R}$	每发送一个报文需 DMA 报文处理开销设为 $N_{DMA\text{-}R}$
协议栈分段开销	N_{SF}	每段 1.5 KB 报文处理开销设为 N_{SF}
标准网卡硬件分段处理开销	N_{HF}	每段 1.5 KB 报文处理开销设为 N_{HF}
驱动分段报文处理开销	N_{DF}	每段 1.5 KB 报文处理开销设为 N_{DF}

标准网卡不支持大报文发送流程如图 2 所示,skb 指向应用需要传输的数据及其报文头内容,使用线性缓冲区存放报文内容,构造报文时,skb 指向的报文长度最大仅为 1514 B。网卡硬件根据描述符指向的地址获取报文内容并将报文发送。标准网卡普通发送 n 个 1.5 KB 报文的报文处理开销可表示为:$n \cdot N_a + n \cdot N_S + n \cdot N_D + n \cdot N_{DMA\text{-}R}$。

图 3 为标准网卡支持大报文发送 TSO 机制流程图,大报文被发送到网卡硬件才进行报文的分段。发送 $n \cdot 1.5$ KB 报文处理开销可表示为:$n \cdot N_a + N_S + N_D + n_2 \cdot N_{DMA\text{-}R} + n \cdot N_{HF}$。

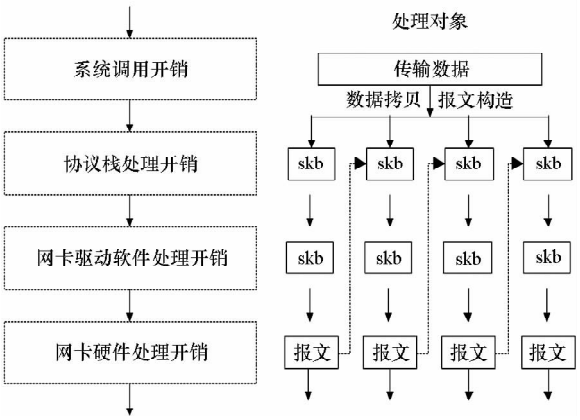

图 2 标准网卡不支持大报文发送流程图

Fig. 2 Sending flow on standard network card (large packet sending is not supported)

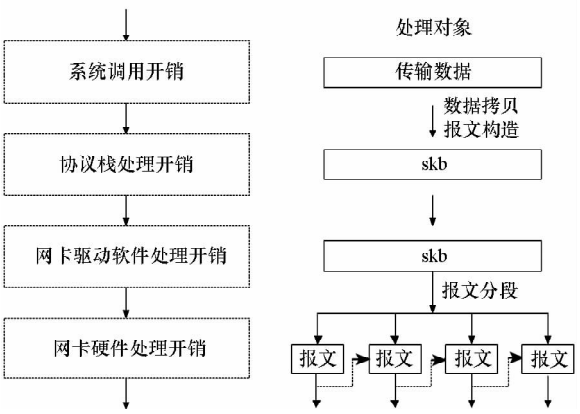

图 3 标准网卡支持大报文发送 TSO 机制流程图

Fig. 3 Sending flow on standard network card (TSO)

图 4 为支持轻量级分组 I/O 的网卡实现大报文发送流程图,大报文被发送到网卡驱动软件进行处理后,完成报文的分段,为每个分段添加报文头部、重新计算每个分段的校验和等功能,并拷贝到指定的多核共享缓冲区中,也就是 skb 指向报文的内容需分段拷贝至共享缓冲区后才能发送出去,无须硬件支持报文拆分组装功能,减小了大报文在协议栈处理的开销,但驱动层存在报文拷贝

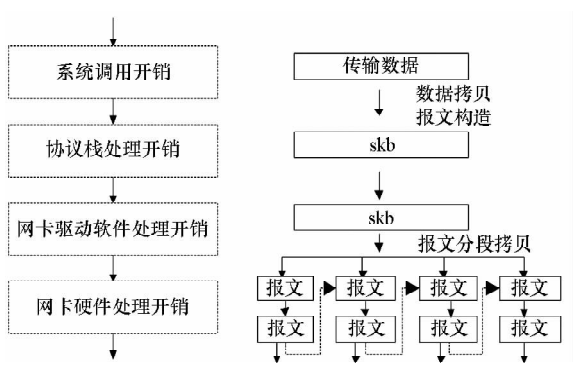

图 4 支持轻量级分组 I/O 的网卡大报文发送流程图

Fig. 4 Diagram of sending flow for based on the multi-core packet processing system with lightweight I/O

开销,且报文分段效率相较于硬件分段较低。发送 $n \cdot 1.5$ KB 路径报文处理开销约表示为:$n \cdot N_a + N_S + n \cdot N_D + n \cdot N_c + n \cdot N_{DF} + n \cdot N_{DMA\text{-}R}$。

图 5 为提出的数据分段卸载开销示意图,大报文被发送到网卡驱动软件并被处理后,完成报文的分段,省去了报文拷贝到指定软件缓冲区中的开销。发送 $n \cdot 1.5$ KB 报文发送路径报文处理开销约表示为:$n \cdot N_a + N_S + N_D + n \cdot N_{DF} + n \cdot N_{DMA\text{-}R}$。

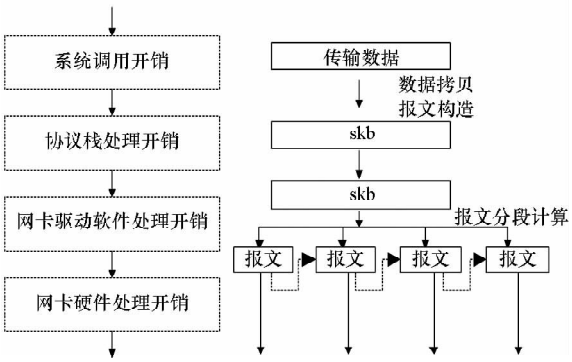

图 5　数据分段卸载开销

Fig. 5　Cost for packet segment offloading

3　数据分段卸载发送机制

3.1　驱动分段及分段报文存储机制

为了降低硬件分段的处理和缓冲区报文拷贝开销,将大报文在驱动层完成切分,只拷贝报文头到共享缓冲区,拷贝的同时驱动完成报文头内容的更新,报文体切分后依然存放在原内存空间。相较于网卡支持的 TSO 机制,硬件切分报文头,更新报文头内容的工作卸载到驱动实现,同时降低了发送拷贝代价。于是驱动拆分报文后,分段报文的存储方式为,分段报文头存储在共享缓冲区中,分段报文体依然存放在线性缓冲区和页缓冲区中。

图 6 为数据分段卸载方法中驱动分段及分段报文存储原理图。解析报文头,对于 TCP 报文,确认报文分段数目以及是否需要进行报文分片。对于需要分片的报文,根据需要分片的数目,将一个大报文的报文头拷贝多份至共享缓冲区中,并同时更新报文头,形成切分后的报文头 1、2、3、4。分段后的小报文,第一段的报文头存储在共享缓冲区,第一段的报文体依旧存储在线性缓冲区,第二段的报文头存储在共享缓冲区,第二段的报文体一部分存储在线性缓冲区,其余部分存储在页缓冲区中。为所有的分段报文的头和体构造一个发送描述符链,硬件根据发送描述符链获取每个报文分段或所有分片的每个报文分段所在位置,并获取每个报文分段或所有分片的每个分段报文内容。将多个报文分段拼装成一个报文或多个分片报文,完成 TCP 校验和计算以及循环冗余校验(Cyclic Redundancy Check,CRC)和计算后,完成报文发送。

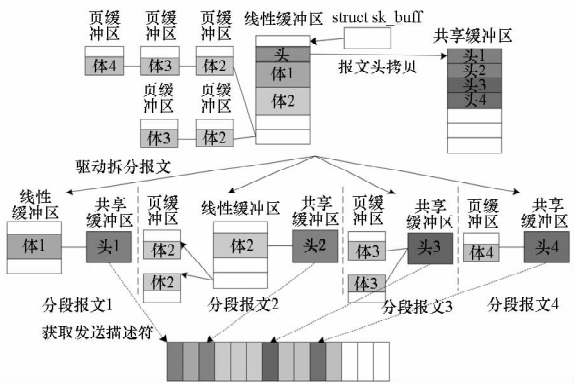

图 6　驱动分段及分段报文存储原理图

Fig. 6　Drive segmentation and storage principle diagram

3.2　数据分段卸载发送方法

数据分段卸载发送方法流程如图 7 所示。

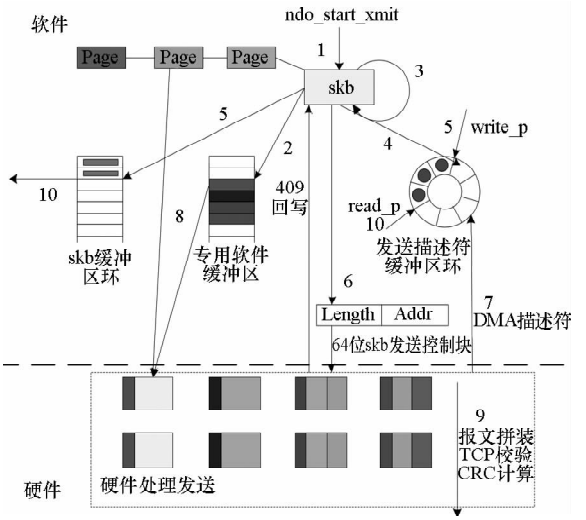

图 7　数据分段卸载发送方法流程图

Fig. 7　Sending flow for data segment offloading and sending mechanism

步骤 1:获取 skb 线性缓冲区和页缓冲区数目,解析报文头,对于 TCP 报文,确认报文分段数目以及是否需要进行报文分片(skb - >len >1514 B);

步骤 2:对于需要分片的报文,根据需要分片的数目,将报文头拷贝至共享缓冲区的多个分区中,完成每个分片报文头内容的更新,无须分片则跳过此步骤;

步骤 3:根据报文分段,为每个分段完成 DMA 映射;

步骤 4:为每个分片的每个报文分段,包括专用缓冲区、线性缓冲区、页缓冲区分别获取一个发送描述符,填充相关字段,构造描述符链表。在获

取描述符时需获得锁,确保不会有多个进程获得同一个描述符;

步骤5:将 skb 缓存到 skb 缓冲区环,并更新发送描述符缓冲区环的写 write_p 指针;

步骤6:构造一个发送描述符控制块,通知硬件有新的报文需要发送;

步骤7:硬件根据发送描述符控制块的内容 DMA 读发送描述符链,获取所有分片中每个报文分段所在的位置;

步骤8:根据描述符中指示报文分段地址,DMA 读取所有分片的每个分段报文内容。将属于同一个分片的多个报文分段拼装成一个分片报文;

步骤9:网卡硬件计算 TCP 校验以及 CRC,完成报文发送后,向发送描述符进行回写,通知软件报文发送完成;

步骤10:驱动处理中断或软中断,检查发送描述符的回写状态,若发送完成则将 skb 从 skb 缓冲区环出队,完成 skb 的释放,并更新发送描述符缓冲区环的 read_p 指针。

上述步骤4中,为每个分片的每个报文分段,包括专用缓冲区、线性缓冲区、页缓冲区分别获取的发送描述符所构造的描述符链表,以支持链式 DMA,允许碎片化的存储中的数据一次 DMA 完成,描述符链表中的每个发送描述符,都包含了64位存储地址信息、长度信息等。

上述步骤6中,构造的描述符控制块描述的信息为描述符链表存储的地址及长度信息,描述符控制块由驱动构造好之后,以写寄存器的方式通知网卡硬件,以实现将整个描述符链表读取到硬件的功能。

4 性能评估

为有效验证大报文数据分段卸载发送功能和性能,设计并实现了原型系统。原型系统基于国防科技大学计算机学院自主研发的高性能通用64位 CPU FT-1500A[12]与自主研制的网络专用协加速引擎网络处理引擎(Network Processing Engine,NPE)[7]构建。其中网络专用协加速引擎在现场可编程门阵列(Field-Programmable Gate Array,FPGA)上实现,FPGA 器件采用 Stratix V 5SGXMA3K3F40C2X。

本实验的软件测试环境包括:ubuntu 14.04 操作系统、NPDK 2.0 版本软件开发环境提供的 NPE 网口驱动[13-14]、用户配置程序。

NPDK 环境将所有软硬件的初始化函数封装在环境库内,调用初始化函数 create_net_device (dma_cnt,disp_mode),dma_cnt 表示硬件启动多少个 DMA 通道,与软件处理线程数对应,每个 DMA 通道对应一个处理线程,绑定在一个指定的 CPU 核上运行;disp_mode 指定报文分派模式,循环分派或端口绑定分派。在不超过硬件最大支持 DMA 通道数情况下,通过重复调用 pthread_create (&p_t,&attr,start_npe_thread,tp)函数,选择自由创建多个处理线程,创建线程数与 dma_cnt 数相同,start_npe_thread 即为业务处理线程。线程创建后,显示指定线程绑定到哪个 CPU 核上运行,线程的创建与 CPU 亲和设置都使用标准的 libpthread 库函数。线程轮询到报文后,立即调用报文发送函数进行发送,完成一个报文的转发操作。发送函数 send_pkt(*pkt, outport, pkt_len),pkt 表示报文指针,outport 表示输出端口号,pkt_len 表示发送长度。

为支持提出的数据分段卸载发送机制,对软件做出的修改有:增加驱动对大报文的各个分段进行 DMA 映射功能;增加对 TSO 功能的支持(netdev->features 属性修改);增加驱动报文分段功能;驱动下发描述符由单个变为多个,修改 skb 软中断,读取发送状态寄存器,修改 skb 回收机制。对硬件做出的修改有:增加 skb 发送状态计数器寄存器;支持一次处理多个描述符,同一报文内容可能需要从不同的区域获取,将这几个区域的报文内容获取后拼装,完成 TCP 校验、计算和更新。

数据发送性能测试场景如图8所示。将由国产 CPU、高性能 FPGA 和存储阵列构建的支持数据分段卸载发送机制的原型系统,通过光纤和千兆网线连接服务器的万兆网卡和千兆网卡接口。服务器万兆网卡接口和原型系统配置成同一网段,由串口登录到存储板 CPU。

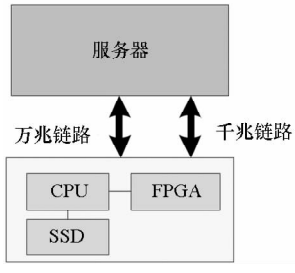

图8 数据发送性能测试场景

Fig. 8 Test scenario for data transmission performance

表2为支持数据分段卸载发送机制前后性能对比。由表2实验数据可以看出,当配置性能测试参数为单线程时,不管是 iscsi 的读性能还是 iperf 的发送性能,在系统支持大报文的数据分段卸载发送机制后,降低了协议栈分段、驱动拷贝等

开销,性能有较大的提升。

表 2　支持数据分段卸载发送机制前后性能对比

Tab. 2　Performance comparison before and after supporting data segmentation offloading and sending mechanism

	iscsi 读性能/(MB/s)	iperf 发送性能/(Gbit/s)
普通发送	192.8	1.94
数据分段卸载发送	272.64	6.11

受限于国产高性能 CPU 面向通用计算,并没有针对网络处理应用的性能瓶颈,单线程单队列单分区的 iscsi 读性能并没有达到线速。因此,建立 iscsi 多个链接,测试在 CPU 不构成性能瓶颈的前提下,发送性能是否达到线速,如图 9 所示。

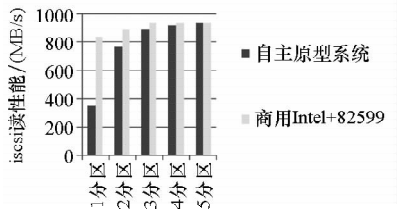

图 9　自主原型系统与商用网卡系统 iscsi 读性能对比

Fig. 9　The iscsi read performance comparison of domestic prototype system and business-used network

使用 fdisk /dev/dfa 命令,将 SSD 固态盘依次分为 1 ~5 个区进行测试,在/etc/iet/ietd. conf 文件中为每个分区增加一个 iscsi 节点设置。每次分区完之后使用/etc/init. d/iscsi-target start 命令,启动 iscsi-target 服务,建立基于多个分区的多个 iscsi 链接。在服务器上与存储板建立 1 个或多个 iscsi 连接后,依次测试 1 ~5 个分区时 iscsi 读性能,即数据发送性能,使用 fio 测试工具统计测试 iscsi 性能。

对比打开 TSO 功能的 82599 商用网卡[13]加 Intel 处理器构建的系统[15],当 iscsi 建立链接为 5 个时,支持数据分段卸载发送机制的自主原型系统与商用网卡构建系统均达到了线速水平。

5　结论

实验结果显示,支持数据分段卸载发送机制的自主原型系统,iscsi 读性能和 iperf 发送性能均有较大提升,相较于商用网卡构建系统,均达到了线速水平。数据分段卸载发送机制支持轻量级分组 I/O,软硬件复杂度较低,能够更好地支撑基于通用多核多线程的高速分组转发。

参考文献(References)

[1]　Gandhi R, Liu H H, Hu Y C, et al. Duet: cloud scale load balancing with hardware and software [J]. ACM SIGCOMM Computer Communication Review, 2015, 44(4): 27 – 38.

[2]　Lei G Q, Dou Y, Wan W W, et al. CPU-GPU hybrid accelerating the Zuker algorithm for RNA secondary structure prediction applications[J]. BMC Genomics, 2012, 13: S13.

[3]　Han S J, Jang K, Park K S, et al. PacketShader: a GPU-accelerated software router[J]. ACM SIGCOMM Computer Communication, 2010, 40(4): 195 – 206.

[4]　Intel. New network processor for next-generation networks[R]. White Paper, Intel Developer UPDATE Magazine, 2001.

[5]　Wan W. Understanding TCP segmentation offload (TSO) and large receive offload (LRO) in a VM ware environment[EB/OL]. (2016 – 01 – 19)[2018 – 01 – 12]. https://kb. vmware. com/s/article/2055140.

[6]　Kavianipour H, Bohm C. High performance FPGA-based scatter/gather DMA interface for PCIe[C]//Proceedings of Nuclear Science Symposium & Medical Imaging Conference, 2012: 1517 – 1520.

[7]　唐路. 通用多核网络处理器高速报文 I/O 技术研究[D]. 长沙: 国防科技大学, 2012.
TANG Lu. Research on packet I/O technology for general-purpose multi-core network processor [D]. Changsha: National University of Defense Technology, 2012. (in Chinese)

[8]　徐东来. 面向通用多核 CPU 的高性能网络 I/O 加速研究与实现[D]. 长沙: 国防科技大学, 2015.
XU Donglai. Research and implementation of high performance network I/O acceleration for general multi-core CPU [D]. Changsha: National University of Defense Technology, 2015. (in Chinese)

[9]　Kekely L, Pus V, Benacek P, et al. Trade-offs and progressive adoption of FPGA acceleration in network traffic monitoring [C]//Proceedings of the 24th International Conference on Field Programmable Logic and Applications, 2014.

[10]　Bonelli N, Di Pietro A, Giordano S, et al. On multi-gigabit packet capturing with multi-core commodity hardware [C]// Proceedings of the 13th International Conference on Passive and Active Measurement, 2012: 64 – 73.

[11]　杨惠, 陈一骄, 李韬. 面向多核网络分组处理系统的线程亲和缓冲区管理机制[J]. 国防科技大学学报, 2016, 38(5): 26 – 31.
YANG Hui, CHEN Yijiao, LI Tao. Thread affinity for buffer management mechanism based on multi-core network packet processing system [J]. Journal of National University of Defense Technology, 2016, 38(5): 26 – 31. (in Chinese)

[12]　Liu J. The heavyweight chip of Chinese electronic FT appear [EB/OL]. (2015 – 04 – 13)[2016 – 03 – 07]. http://www. news. ifeng. com/a/20150413/43539854_0. shtml.

[13]　García-Dorado J L, Mata F, Ramos J, et al. High-performance network traffic processing systems using commodity hardware [C]//Proceedings of Lecture Notes in Computer Science, Data Traffic Monitoring and Analysis, 2013, 7754: 3 – 27.

[14]　Rizzo L. Netmap: a novel framework for fast packet I/O [C]// Proceedings of the USENIX Annual Technical Conference, 2012.

[15]　Bonelli N, Di Pietro A, Giordano S. On multi-gigabit packet capturing with multi-core commodity hardware [C]// Proceedings of International Conference on Passive and Active Network Measurement, 2012: 64 – 73.