doi:10.11887/j.cn.201605005 http://journal.nudt.edu.cn

面向多核网络分组处理系统的线程亲和缓冲区管理机制*

杨 惠[1],陈一骄[1],李 韬[1],李世星[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:基于通用多核架构的网络分组处理系统性能受到诸如分组IO开销高、多核共享内存及进程调度竞争大、页表缓冲表项失效率高等问题的困扰。为此提出一种基于通用多核网络分组处理系统、面向高速分组转发应用的线程亲和缓冲区硬件管理机制,并在网络专用协处理引擎上实现。该机制采用无中断的线程亲和调度策略,将包含控制信息与缓冲区地址信息的描述符和分组数据按照分组处理的线程号链式地对应加载在多个地址连续的共享缓冲区中。基于通用多核和现场可编程门阵列平台进行报文转发测试,实验结果表明,采用线程亲和缓冲区管理机制能使平均报文转发处理性能提升12.4%,有效地降低IO开销和TLB表项失效率。

关键词:分组IO;线程亲和;缓冲区管理;分组转发;多核
中图分类号:TP393 **文献标志码**:A **文章编号**:1001-2486(2016)05-026-06

Thread affinity for buffer management mechanism based on multi-core network packet processing system

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

Abstract: The packet processing performance of the general multi-core architecture is plagued by many factors, including high packet IO cost, multi-core share memory and process scheduling competition, TLB entries failure rate, etc. Therefore, a TABM (thread affinity for buffer management mechanism) based on multi-core network packet processing system, which is oriented to high-speed packet forwarding application was proposed and completed on network dedicated co-processing engine. The TABM adopts thread affinity scheduling strategy with no interrupt, sends each packet data and descriptor which contains control and buffer address information to several successive shared buffers according to the corresponding thread ID, and organizes the packets and descriptors which processed in the same thread in the form of a chain. The packet forwarding performance was tested on the basis of general multi-core and field-programmable gate array platform. The experimental data show that the average packet forwarding performance is promoted by about 12.4% and the IO cost and the TLB entries failure rate are reduced by adopting the TABM.

Key words: packet IO; thread affinity; buffer management; packet forwarding; multi-core

随着新型网络业务、协议以及多核技术的发展,具有高可编程性的基于多核处理器的网络分组处理系统成为学术界和工业界研究的热点。随着通用多核处理器处理性能的不断提升,通用多核处理器与专用网络处理器芯片的差距逐步减小,具备高可编程性的通用多核处理器成为网络设备中广泛采用的数据平面处理核心器件[1-2]。基于通用多核CPU+网络专用协处理引擎(Network dedicated co-Processing Engine, NPE),能够实现其对网络处理器芯片的有效替代[3-5]。然而,在基于通用多核进行网络分组处理的过程中存在一系列不属于网络分组处理的开销,诸如中断处理、上下文切换、系统调用、数据拷贝、进程调度等,这些开销被称作分组IO开销[6-8]。将多核应用于IPv4转发,分组IO开销占整个分组处理开销的70%[1]。当前的多核处理器设计并未考虑到网络处理中分组IO的问题,在获得了高效网络处理性能的同时,也伴随着分组IO带来的长处理延迟和复杂的网卡设计问题[2]。分组处理伴随着密集的IO操作,分组处理过程中涉及的频繁的访存和页表缓冲(Translation Lookaside Buffer, TLB)查表操作等会导致严重的访存开销。

* **收稿日期**:2015-05-27
基金项目:国家自然科学基金资助项目(612024823,61301236)
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

为降低通用多核处理网络分组带来的 IO 开销,降低 TLB 表缺失率,提出一种面向高速分组转发的线程亲和缓冲区管理机制(Thread Affinity for Buffer Management mechanism, TABM)。其主要思想是系统初始化时,将预先分配的多核共享缓冲区分成大小相等、属性相同的连续的缓冲区块,每一个 CPU 线程对应处理缓冲区块中存储的分组数据;接收端采用线程亲和的无中断链式接收模式,将包含控制信息与缓冲区地址信息的描述符和分组数据,由实现 TABM 的 NPE,通过直接内存存取(Direct Memory Access, DMA)控制搬移,根据分组在多线程上处理的线程号,对应地加载在多个地址连续的共享缓冲区块中,每一个线程中处理的分组及描述符信息以链表的形式组织并地址连续地存放,无须中断多核;发送端将缓冲区地址管理的任务由多核软件管理卸载下来,各个线程通过各自的计数器到达阈值,以此来释放共享缓冲区地址空间。线程亲和的缓冲区管理方法在 NPE 上硬件实现,用于解决降低系统缓冲区的管理开销、IO 开销和线程乱序存储分组造成的 TLB 表缺失的问题,为多核实现高速分组转发提供了有效的解决方案,更好地支持了多核多线程报文处理。

1　相关研究

针对多核网络分组处理系统的分组 IO 开销大的问题,Intel 基于通用多核处理平台进行高效网络软件处理,提出了数据平面开发工具套件(Data Plane Development Kit, DPDK)[2],为高速网络设计了一套数据平面库,提供了统一的处理器软件编程模式,从而帮助应用程序有效地接收和发送数据,提高分组 IO 性能。PacketShader[1] 则采用大报文缓冲区的方式,静态地预分配两个大的缓冲区——套接字缓存(SocKet Buffer, 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 IO 开销和访存开销。Netmap[9] 通过预分配固定大小缓冲区、批处理和并行直接路径的方法,实现了内存映射,存储信息结构简单高效,能够实现报文的高速转发。直接高速缓存访问(Direct Cache Access, DCA)通过处理器硬件支持,将接收网络分组直接写入 LLC cache,减小 CPU 访问分组描述符的延时[10]。而基于多核架构的 Linux 内核数据包获取引擎[6] 接收的报文不需要通过标准协议栈处理,而是送入批处理队列进行批处理。现有研究还采用内存映射的零拷贝技术,采用将内核的内存区域直接映射到应用空间的方法,减少内存访问,这类方法需要对驱动和内核做出修改,只能解决拷贝的开销,不能解决报文缓冲区分配和释放开销。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[11] 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,但存在 TLB 页表频繁失效切换的问题,带来了很大的访存开销。

2　基于多核的线程亲和缓冲区硬件管理

2.1　多核共享缓冲区硬件管理基础机制及分组转发模型

以接收端为例,首先简要介绍在通用多核分组处理系统中报文接收的基本流程:网络端口接收到分组后,通过 DMA 将分组传送到预先分配的内核缓冲区内,软件更新描述符环、标记已经使用的缓冲区描述符,并发出中断请求、通知多核分组数据到达,操作系统处理中断并将分组数据送往协议栈。这其中存在大量的 CPU 中断处理开销软件维护描述符的系统开销以及内核缓冲区与用户数据之间的大量数据拷贝等开销。为了有效地解决缓冲区管理开销问题,本文提出的 TABM 基于已有研究——SDB 方法[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缓冲区中,也就是每个缓冲区有固定大小的单元对应存储一个报文的数据和控制信息,每个单元的报文数据对齐到 2 K 字节。为了支持多核多线程,该缓冲区为多核共享地址连续的存储区域。实现了 SDB 方法的 NPE 在初始化时,将共享缓冲区的描述符填入空闲描述符块队列中。当数据报文到达,SDB 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 PCIE 送往共享缓冲区对应的地址空间。该共享缓冲区以块为单位,由硬件进行组织、申请和释放。描述符和报文块以单向链表的形式在共享缓冲区内组织,如图 1 所示。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,无须通过中断响应,也无须多次访存缓冲区。

2.2　线程亲和缓冲区管理机制描述

基于 SDB 管理机制的分组转发模型能支持多核多线程的调度,然而不同线程处理的数据报文将乱序排列在内存共享缓冲区中。如图 1 所

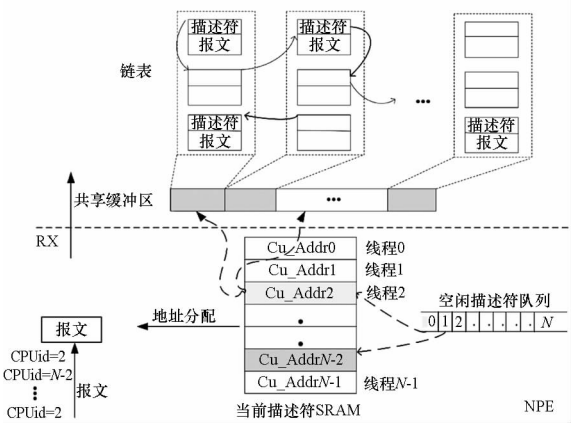

图 1　无中断缓冲区硬件管理机制示意图

Fig. 1　Hardware buffer management mechanism without interruption

示,当前描述符静态随机存取存储器(Static Random Access Memory, SRAM)中,与每个 CPU 线程号一一对应地存储着当前即将接收报文的空闲描述符,而当前描述符 SRAM 中对应的报文空闲描述符被分配给报文后,从空闲描述符队列中产生下一个当前描述符,存入 SRAM 对应的线程号中。例如,首先将要送往线程 2 号处理的报文,从网络端口经由 NPE 分配描述符后,送往多核共享缓冲区;接着,送往线程号为 $N-2$ 等的报文陆续到达,由于空闲描述符队列依次给各个线程提供当前空闲描述符,下一个线程号同样为 2 的报文对应的缓冲区地址就不能与上一个线程号为 2 的报文连续,即同一线程上处理的报文被乱序地加载到共享缓冲区内存储,这将导致多核处理报文时 TLB 查表的高失效率,致使 TLB 表项反复冲刷,严重影响系统性能。

如果将送往相同内核线程进行处理的报文对应的顺序存放到连续的共享缓冲区空间,将缓冲区管理进行线程亲和,会大大降低 TLB 查表的失效率。于是,将共享缓冲区按线程数进行划分,以 8 线程为例,如图 2 所示,多核共享缓冲区按地址顺序划分成 8 块。实现 TABM 的 NPE 进行共享缓冲区描述符初始化、维护与分配、释放与回收的机制如下:

1)共享缓冲区描述符在 NPE 中的初始化:系统首先在内存空间中初始化连续的共享缓冲区块,再将要初始化的缓冲区描述符块通过写寄存器的方式通知 NPE,于是缓冲区描述符按线程填充到对应的空闲描述符队列中;

2)共享缓冲区描述符的维护与分配:从线程对应的空闲描述符队列中取出当前块基地址,并顺序以固定长度 2 K 为偏移量,形成当前描述符存放入 SRAM 中,当报文到达,将对应线程的报文数据与对应线程的当前描述符,通过 PCIE 发送到共享缓冲区中;

3)共享缓冲区释放和描述符回收:按块释放缓冲区,每个缓冲区块均包含 M 个 2 K 大小的缓冲区,那么当该块对应的计数器为 M 时,表明该缓冲区块全部释放,该块对应的描述符可以回收到对应线程的空闲描述符队列中。

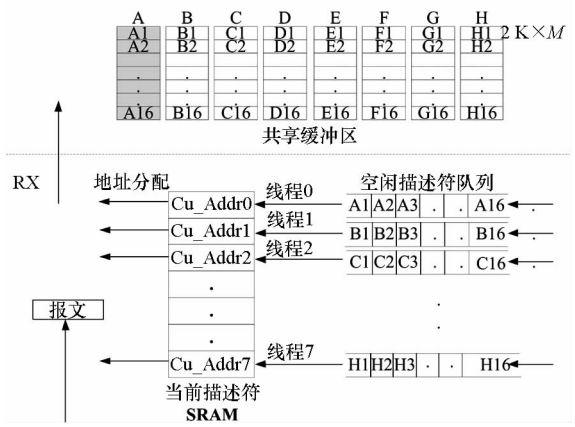

图 2　线程亲和缓冲区管理机制示意图

Fig. 2　Thread affinity for buffer management mechanism

2.3　线程亲和缓冲区管理机制硬件实现

如图 3 所示,线程亲和缓冲区硬件管理机制结构实现主要包括缓冲区地址分配模块、缓冲区描述符回收模块。

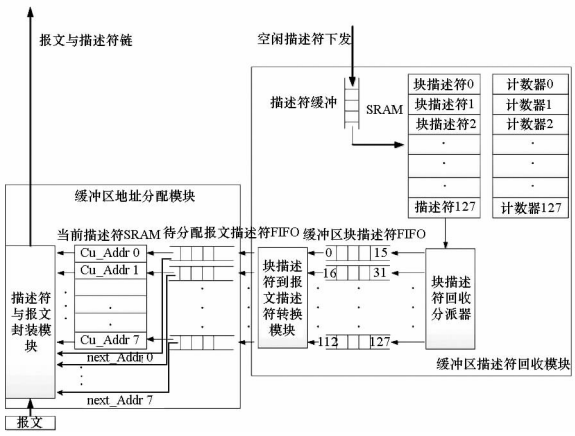

图 3　线程亲和缓冲区管理机制硬件结构图

Fig. 3　TABM architecture

缓冲区地址分配模块:接收报文,并根据该报文对应的 CPU 线程号,从当前描述符 SRAM 中获取当前报文存放到共享缓冲区内的描述符信息 Cu_Addr$i$,同时从各自待分配报文描述符先入先出队列(First Input First Output, FIFO)中获取下一报文存放到共享缓冲区内的描述符信息 next_Addr$i$ ($i=0,1\cdots,7$),同报文一起构造报文链表,并通过 PCIE 发送到多核的共享缓冲区内存储。

缓冲区描述符回收模块:主要负责初始化和回收共享缓冲区描述符队列,并为报文链形成可用的描述符。多核下发需要释放的共享缓冲区对应的描述符信息到下发描述符缓冲。根据描述符信息,将该描述符对应的块地址存放入缓冲区块描述符 SRAM 中,每个块地址都对应一个释放计数器。仍旧以每个缓冲区块包含 M 个 2 K 大小的缓冲区为例,当释放计数器计数到 M 时,释放这一个块地址,并将对应的块描述符写入缓冲区块描述符 FIFO 中。由于每一个 CPU 线程对应一个缓冲区块描述符 FIFO,因而图 2 中的 A1 至 A16 缓冲区块描述符经由块描述符回收分派器对应加载到缓冲区块描述符 FIFO 0 中,H1 至 H16 缓冲区块描述符对应加载到缓冲区块描述符 FIFO 7 中。

块描述符到报文描述符转换模块:实现将 8 个块的地址信息分配到对应的 8 个待分配报文描述符 FIFO 中。一个缓冲区块基地址包含 M 个待分配报文描述符,每形成一个待分配报文描述符,计数器加 1,地址偏移 2 K。当计数器等于 M 时,则这个缓冲区块的地址分配完了。

2.4　基于 TABM 的报文处理流程

为实现无中断地接收由网络接口传送来的数据报文,接收端接收的报文将被存储到共享报文缓冲区中。接收报文需要给报文分配存放在内存的地址,那么首先需要对报文存放的内存地址进行初始化,这部分初始化工作由软硬件配合完成。在描述符初始化流程中,系统首先将要初始化的共享报文缓冲区描述符经过图 3 中所示的下发描述符缓冲,再下发到块描述符 SRAM 中,同时将对应的缓冲区块计数器进行累加,当计数器数值到达块大小时,缓冲区块地址经由块描述符回收分派器按照线程对应地加载到缓冲区块描述符 FIFO 中,同时将块描述符 SRAM 中的计数器清零。初始化流程的描述在表 1 中给出。

表 1　硬件初始化伪码描述

Tab. 1　Hardware initialization

initial_Addr
/* 描述符初始化流程 */
1. **begin**
2. 系统驱动将缓冲区块地址描述符信息写到对应 8 个块描述符队列表中去;
3. 块地址号写入缓冲区块描述符管理 SRAM 中;
4. 对应的释放计数器置 0;
5. **end**

当缓冲区块地址在缓冲区块描述符 FIFO 中准备好以后,数据报文便可以从网络接口接收进来。根据接收进来的数据报文的线程号 CPUid,从线程对应的当前描述符 SRAM 中获取描述符信息,将描述符和数据报文组成报文链表,构造成 PCIE 的写请求 TLP 报文,送往共享报文缓冲区内与线程对应的存放空间中,与此同时将该线程的下一描述符信息从空闲描述符队列 FIFO 中填充到当前描述符 SRAM 中已被分配出去的位置。只要共享报文缓冲区中存在空闲的地址,分组处理系统就可以持续地接收网络接口接收上来的报文。分组处理系统之上运行的应用程序通过轮询的方式检测到新数据的到达,然后对数据进行处理,无须中断端系统进行数据接收。RX 接收流程的硬件描述在表 2 中给出。

表 2　RX 端接收报文伪码描述

Tab. 2　Packets receiving of RX terminal equipment

RX
1. **begin**
2. **if**(存在可接收报文 && 待分配报文描述符 FIFO 不为空)**begin**
3. 根据接收报文中的 CPUid 从 Cu_Addr 八个队列中选择对应 metadata 字段;
4. next_Addr 将从描述符 FIFO 中获得的新 metadata 信息更新至 Cu_Addr;
5. 将报文与描述符组合好发送至主机;
6. **end　else return**;
7. **end**

当共享缓冲区中的报文被处理完之后,需要将数据通过发送端下发出去,TX 发送分组到端系统及描述符的回收如表 3 所示。TABM 为了降低描述符回收代价,将描述符回收下放到硬件实现,因此发送端实现了数据报文的发送以及描述符的回收两种功能。首先系统将需要下发的数据的描述符信息,包括线程号、下发端口信息、数据存放地址、长度信息等寄存器写的方式,存入下发描述符缓冲中。一方面,描述符缓冲中提取的描述符信息写入块描述符 SRAM 中,将对应的缓冲区块计数器进行累加,当计数器数值到达块大小时,缓冲区块地址经由块描述符回收分派器按照线程对应地加载到缓冲区块描述符 FIFO 中,并将块描述符 SRAM 中的计数器清零。另一方面,根据描述符下发缓冲中提取的描述符信息,包括需回收报文缓冲区地址和长度信息,构造 PCIE 的内存读请求 TLP

报文。发生端接收到多核分组处理系统返回的报文缓冲区内需下发数据报文的 Completion 报文后解析并进行报文重定序和拼接,然后将组装完成的需下发的数据报文下发至网络接口。

表 3 TX 主机下发分组到端及描述符回收

Tab. 3 Packets sending from TX CPU to terminal and descriptor recycle

	TX
1.	**begin**
2.	**if**(报文下发描述符缓冲不为空)**begin**
3.	将要发送报文 metadata 信息,根据线程号以寄存器方式写入 TX 发送引擎中,具体指写入缓冲区描述符回收模块;
4.	**if**(释放计数器 > 1023)**begin**
5.	释放计数器置 0;
6.	缓冲区描述符 SRAM 对应地址清除;
7.	根据线程对应的缓冲区块号,将释放的可用缓冲区地址写入对应线程号的缓冲区块描述符 FIFO 中;**end**
8.	**else begin**
9.	释放计数器累加; **end**
10.	根据报文地址和长度构造 DMA 读请求;
11.	**if**(Completion 报文返回)**begin**
12.	转发该报文;
13.	**end else return**;
	end

3 性能评估

为有效验证 TABM 技术的功能和性能,本文设计并实现了 TABM 原型系统。原型系统基于国防科学技术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 NPE[11] 构建。其中 NPE 的核心部件在现场可编程门阵列(Field-Programmable Gate Array, FPGA)上实现,FPGA 器件采用 Stratix IV EP4SGX230KF40C2。

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] 及用户配置程序。

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),其中 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。发送函数为 send_pkt(* pkt, outport, pkt_len),其中 pkt 表示报文指针,outport 表示输出端口号,pkt_len 表示发送长度。

在该原型系统的软硬件基础上,通过 Ixia 网络测试仪连续发送大小为 64 B 至 1500 B 大小不等的报文。由一个万兆端口接收的报文,采用公平轮询的方式分配到 1,2,4,8 个 CPU 线程处理,每个 CPU 线程被设置访问对应的共享缓冲区块;通过软件设置 DMA 通道数,每个线程对应一个 DMA 通道,创建不同的软件线程。测试结果如图 4 所示,可以看出采用 TABM 的原型系统,当发包速率为 10 Gbps、转发分组大小为 1500 B 时,速率最高达 7.93 Gbps。当分组较小时,支持线程数越大,转发性能越好。

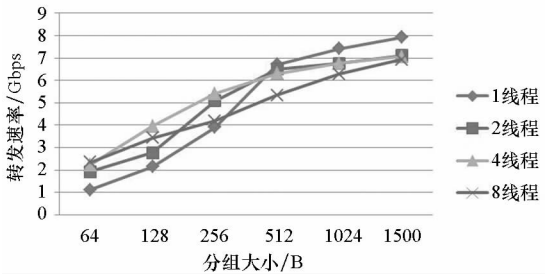

图 4 TABM 分组转发性能

Fig. 4 Packet forwarding performance of TABM

另外,在该原型系统的基础上,通过 Ixia 网络测试仪连续发送大小为 64 B 的报文,分别在 FT - 1500A加 NPE 万兆加速引擎子卡的测试平台上对 SDB 机制和 TABM 进行了裸转发性能测试和比对。测试结果如图 5 所示,可以看出,TABM

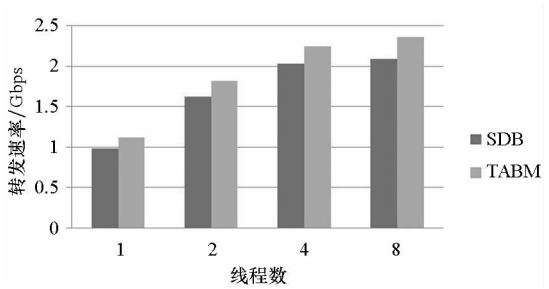

图 5 TABM 与 SDB 机制的性能比对

Fig. 5 Performance comparison of TABM and SDB mechanism

相较于 SDB 机制,分组转发性能有明显的提升,在内核没有支持更大 TLB 页表的保守情况下,平均性能提升 12.4%。TABM 更加适应于通用多核多线程的报文处理平台。

4　结论

为了解决通用多核分组处理的分组 IO 开销问题,提升转发性能,基于通用多核加网络专用协处理引擎的平台,提出了一种基于多核网络分组处理系统的线程亲和缓冲区硬件管理机制 TABM,并实现了支持 TABM 的原型系统。实验结果显示,当发包速率为 10 Gbps 时,相较于无中断 SDB 管理机制,TABM 能使系统整体转发速率提高约 12.4%。TABM 能够有效降低系统缓冲区的管理开销、IO 开销和线程乱序存储分组造成的 TLB 表缺失,更好地支持基于通用多核多线程的高速分组转发。

参考文献(References)

[1] 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.

[2] Wind River Systems. High-performance multi-core networking software design options[R]. White Paper Intel, 2011.

[3] Huggahalli R, Iyer R, Tetrick S. Direct cache access for high bandwidth network I/O [J]. ACM SIGARCH Computer Architecture News, 2005, 33(2): 50 – 59.

[4] Dashtbozorgi M, Abdollahiazgomi M. A high-performance and scalable multi-core aware software solution for network monitoring [J]. Journal of Supercomputing, 2012, 59(2): 720 – 743.

[5] Kekely L, Pus V, Benacek P. 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 (FPL), Munich, 2014: 1 – 4.

[6] Bonelli N, Pietro A D, 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, Berlin/Heidelberg: Springer, 2012: 64 – 73.

[7] Biersack E, Callegari C, Matijasevic M. Data traffic monitoring and analysis—from measurement, classification, and anomaly detection to quality of experience [J]. Lecture Notes in Computer Science, 2013, 7745: 45 – 56.

[8] García-Dorado J L, Mata F, Ramos J, et al. High-performance network traffic processing systems using commodity hardware [J]. Lecture Notes in Computer Science, 2013, 7754: 3 – 27.

[9] Rizzo L. Netmap: a novel framework for fast packet I/O[C]// Proceedings of the USENIX Annual Technical Conference, 2012:1 – 12.

[10] Rizzo L, Deri L, Cardigliano A. 10 Gbit/t line rate packet processing using commodity hardware: survey and new proposals[EB/OL]. [2015 – 05 – 08]. http://luca. ntop. org/log. pdf.

[11] 唐路. 通用多核网络处理器高速报文 I/O 技术研究[D]. 长沙:国防科学技术大学, 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)

[12] 中国电子报. 中国电子重量级芯片“飞腾”亮相[EB/OL]. (2015 – 04 – 13)[2016 – 03 – 07]. www. news. ifeng. com/a/20150413/43539854_0. shtml.
Journal of China Electronic. The heavyweight chip of Chinese electronic FT appear [EB/OL]. (2015 – 04 – 13)[2016 – 03 – 07]. www. news. ifeng. com/a/20150413/43539854_0. shtml. (in Chinese)

[13] 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:国防科学技术大学, 2015.
XU Donglai. Research and realization of high-performance I/O acceleration on general multi-core CPU[D]. Changsha: National University of Defense Technology, 2015. (in Chinese)