基于双散列的 DPDK 负载均衡算法设计与实现

Design and Implementation of DPDK Load Balancing Algorithm Based on Double Hash

熊义龙 曹炳尧 谢莹庆 (上海市特种光纤与光接入网重点实验室 上海大学通信与信息工程学院,上海 200444)

摘要:随着网络快速发展,业务越发复杂,RSS 和对称 RSS 分流不均衡导致高性能数据包处理框架 DPDK 性能不能充分发挥,针对这一问题提出了一种双 hash 负载均衡算法。该算法通过网卡硬件特性对称 RSS 机制和软件实现的对称 hash 算法,对数据包五元组信息进行哈希散列确定对应的逻辑核实现均衡分流。搭建了实验测试环境,和对称 RSS 算法对比,实验结果表明对称 RSS 算法分流不均衡,10 次实验结果的 CPU 使用率的均方差为 334.48。而双 hash 负载均衡算法分流均衡,各个逻辑核的 CPU 使用率均控制在 50%范围小幅波动,对于 10 次实验结果的 CPU 使用率的均方差仅为 23.92,达到了较好的均衡效果。提出的双 hash 均衡算法对于基于 DPDK 的网络处理设备研制具有一定参考价值。

关键词:DPDK;RSS;对称 RSS;负载均衡

Abstract: This paper proposes a dual-hash load balancing algorithm. The algorithm uses the symmetric RSS mechanism of the hardware characteristics of the network card and the symmetric hash algorithm implemented by software to hash the quintuple information of the data packet to determine the corresponding logical core to achieve balanced distribution. An experimental test environment is built, and compared with the symmetric RSS algorithm, the experimental results show that the symmetric RSS algorithm is not balanced, and the mean square error of the CPU usage of 10 experimental results is 334.48. In the dual-hash load balancing algorithm, the CPU usage of each logical core is controlled within 50% and fluctuates slightly. The mean square error of the CPU usage of the 10 experimental results is only 23.92, achieving a good balancing effect.

Keywords: DPDK, RSS, symmetric RSS, load balancing

随着网络技术的迅速发展,网络应用不断涌现,给人们生活带来便利的同时也导致了各区网络流量急剧提升, 数据呈现爆发式增长,各种安全问题不断涌现。为了解决安全问题, 研究学者对网络安全领域中的应用进行了大量研究[1-2],如网络安全 IDS、IPS、防火墙、数据库防护系统、信息审计系统等,这些应用都需要依赖数据包捕获技术, 因此对数据包捕获技术的研究将具有重大意义。

但随着网络速率逐渐向 10 Gbps 过渡,负责数据包捕获的处理器上的负载不断增大,丢包问题也不断涌现。因此为了提高数据包捕获性能,研究学者提出了许多流量负载均衡方法。负载均衡方法有硬件实现、软件实现两种类别。基于硬件实现的负载均衡系统虽然性能稳定, 需要相关的技术支持人员对设备进行维护保障,但是价格昂贵,依赖厂商提供技术支持。因此基于软件的负载均衡备受研究学者青睐。基于软件的负载均衡算法有轮转法、散列法、最小连接法、最快响应法等[3],这些大多是基于集群的系统架构实现的,相对来说性价比不太高。

目前多核处理器已经得到广泛应用, 多核处理器不仅具备单核处理器价格低、操作简单以及软硬件资源丰富,还具备功耗低、并发度高等优点。在高速网络环境下通过多核处理器对网络流量进行并行处理是目前主流的方向。作为高性能数据包处理框架的 DPDK,最近几年也是备受关注,对于负载均衡方面的研究,DPDK 提供了 RSS 分流机制在多核处理器上实现负载均衡,该方法是通过 DPDK 开启网卡的硬件 RSS 分流特性实现的。由于默认的 RSS 机制对同一链路上双向数据包的五元组信息进行哈希计算得到的值不同,数据包会被分配到不同的逻辑核上处理,这将导致额外的性能开销,因此提出了一种对称 RSS 算法[4],该算法能够保证同一链路上双向数据包会被分配到同一逻辑核上处理。但是对称 RSS 算法针对单一会话流将出现某个逻辑核负载过大的问题。

为了解决对称 RSS 负载不均衡的问题,本文提出一种高效的负载均衡算法对改善数据包捕获性能具有重大研究意义。本文基于 DPDK 开发平台,在保证同一链路的双向数据包分配到同一逻辑核上进行处理的前提下, 提出了双 hash 负载均衡算法,该算法分流效果可以满足目前用户的需求。

1 相关技术

DPDK 是 Intel 公司开发的一套强大、高度优化的数据平面开发工具套件[5],它不同于 Linux 操作系统以通用设计为目的,DPDK 集中于网络应用中数据包的高性能处理。DPDK 最大的区别是应用程序运行在用户层, 使用的是自己的数据平面库发送和接收数据包,这个过程不需要 Linux 内核协议栈参与。为了使得流量被均衡的分配到各个逻辑核上, 可以对网卡接收的的数据流进行分流,DPDK 提供了 RSS 分流机制。

1.1 多队列扩展技术 RSS

RSS 是网卡的硬件特性, 需要网卡支持才能被 DPDK 开启。通过设置 rte_eth_conf 结构体中的 rss_hf 和 mq_mode 字段来开启 RSS 功能。本次实验使用的是 X710 系列网卡,该网卡支持 RSS 特性。RSS 能够利用特定的报文字段值进行哈希散列得出哈希值,根据哈希值确定数据包分配的队列,特定字段值由数据包类型确定。RSS 的工作原理如图 1 所示:

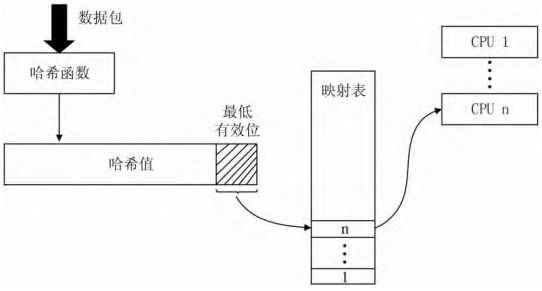

图 1 RSS 工作原理

从图 1 可以看出，当网卡接收到数据包后，首先是经过哈希函数计算哈希值，哈希函数的输入包括源 IP 地址、目的 IP 地址、源端口、目的端口、协议类型五元组信息，取哈希值的最低有效位用于索引记录队列和逻辑核关系的映射表，通过索引值找到对应的队列和逻辑核，从而实现了将数据包分配到对应的队列和逻辑核上。默认使用的哈希函数是 Toeplitz 哈希函数，函数输入包括五元组信息的 input[]数组以及随机密钥 K ，其中数组大小是 12 Byte，密钥大小为 40 Byte，经过哈希函数计算的哈希值可以通过访问数据包结构体 rte_mbuf 中的元素 hash 联合体，通过联合体的元素 rss 字段获得。

1.2 对称 RSS

由于采用 DPDK 默认的密钥值 K，会使得对于同一连接的双向数据包经过哈希散列后得到的 hash 值不一样，关系如公式(1)所示：

$$Hash(srcIP,srcPort,dstIP,dstPort,K) \neq Hash(dstIP,dstPort,srcIP,srcPort,K) \quad (1)$$

因为修改 RSS 算法比较困难，而密钥值 K 是网卡驱动的一部分，在驱动程序首次加载的时候会设置到网卡中，所以求解一个密钥值 K，使得同一通路上双向的数据包通过哈希散列得到的哈希值一样相对简单。对于一个双向的数据包，它们的五元组中协议类型位置是完全一样，所以只需要源 IP 地址、目的 IP 地址、源端口、目的端口计算得到的哈希值一样就能保证被分配到同一逻辑核。其实通过算法可以看出密钥值 K 并非每一位都参与哈希计算，对于给定的 *input*[]的一比特位，只有 32 位 K 参与异或操作，*input*[]的元素和 K 的关系如表 1 所示：

表 1 *input*[]对应的密钥值 K 的范围

输入	比特位数/位	输入位范围	K 的位范围
srcIP	32	1~32	1~63
dstIP	32	33~64	33~95
scrPort	16	65~80	65~111
dstPort	16	81~96	81~127

假设 $K[1:N]$表示为密钥值 K 的第 1 位到第 N 位的范围，要使公式(2)成立：

$$Hash(srcIP,srcPort,dstIP,dstPort,K) = Hash(dstIP,dstPort,srcIP,srcPort,K) \quad (2)$$

需要使 srcIP 和 dstIP，srcPort 和 dstPort 对应的 K 的范围分别相等，即满足公式(3)、公式(4)的关系，这样对于哈希函数的输入参数 1 和参数 3，参数 2 和参数 4 只要值相同即使位置互换，它们的哈希值也将相同。

$$K[1:63]=K[33:95] \quad (3)$$

$$K[1:63]=K[33:95] \quad (4)$$

从公式(3)、公式(4)可以看出对于有重叠部分，将其关系拆成三个不重叠的部分得到公式(5)~(7)。

$$K[1:15]=K[17:31]=K[33:47]=K[49:63]=K[65:79]=K[81:95]=K[97:111]=K[113:127] \quad (5)$$

$$K[16]=K[48]=K[80]=K[96]=K[112] \quad (6)$$

$$K[32]=K[64] \quad (7)$$

满足公式(5)~(7)的一个密钥值 K 如图 2 所示：

$$\begin{bmatrix} 0x6d5a & 0x6d5a & 0x6d5a & 0x6d5a \\ 0x6d5a & 0x6d5a & 0x6d5a & 0x6d5a \\ 0x6d5a & 0x6d5a & 0x6d5a & 0x6d5a \\ 0x6d5a & 0x6d5a & 0x6d5a & 0x6d5a \\ 0x6d5a & 0x6d5a & 0x6d5a & 0x6d5a \end{bmatrix}$$

图 2 对称 RSS 密钥值 K

对称 RSS 相比 RSS 解决了同一链路上双向的数据流分配到不同逻辑核上的问题，但是对于单一会话流还是会出现某一逻辑核上负载过重的情况，就需要提出新的负载均衡方法。

2 双 hash 负载均衡算法

2.1 系统框架设计

因为对称 RSS 存在负载分配不均衡问题，为了保证同一链路上的流量尽量分配到同一逻辑核上处理，且流量能平均分配到各个逻辑核上处理，提出了双 hash 负载均衡算法充分利用多核处理器的性能，系统的架构图如图 3 所示：

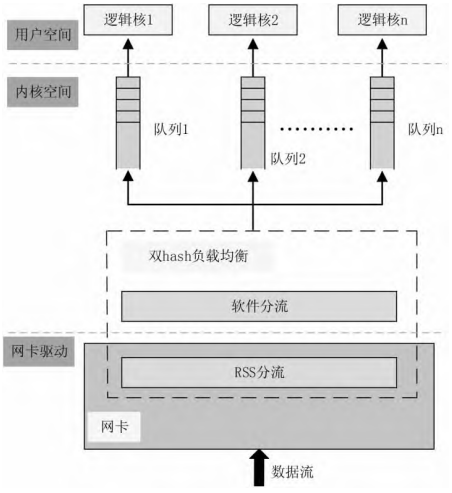

图 3 双 hash 负载均衡算法系统架构图

从图 3 可以看出，当网卡接收到数据包，将通过 RSS 分流机制和软件分流两个子模块计算对应的五元组信息的哈希值，其中 RSS 分流机制和软件分流组成了系统的双 hash 负载均衡模块。RSS 分流机制是网卡的硬件特性，为了保证同一链路的流量分配到同一逻辑核，这里采用的是对称 RSS 算法，软件分流采用的是通过软件实现的对称哈希算法，软件分流的哈希计算是在主核上完成的，所以不影响负责数据包处理的逻辑核的性能。经过双 hash 算法得到数据包五元组信息的哈希值，将根据哈希值和队列的映射关系将数据包分配到对应的接收队列，进入对应的逻辑核上处理。经过双 hash 散列之后数据包能够进一步解决单 hash 出现的 hash 碰撞问题，从而将数据包更加均衡的分配到各个逻辑核上。

2.2 算法设计

双 hash 负载均衡算法的思想是采用对称 RSS 和对称哈希算法实现流量均衡分配，对称 RSS 第 2 小节已经介绍过，对称哈希算法采用的是源 IP 地址、目的 IP 地址、源端口和目的端口进行异或运算。由于异或操作不关乎两个操作数的顺序，这样对于双向的数据流只要满足公式(2)就能被分配到同一逻辑核上进行处理。哈希算法的具体实现步骤如下：

1)将源 IP 地址 S_IP、目的 IP 地址 D_IP、源端口 S_Port 和目的端口 D_Port 以 8 比特位进行拆分得到多个字节段。

$S\_IP1=S\_IP[1:8], S\_IP2=S\_IP[9:16]$

$S\_IP3=S\_IP[17:24], S\_IP4=S\_IP[25:32]$

$D\_IP1=D\_IP[1:8], D\_IP2=D\_IP[9:16]$

$D\_IP3=D\_IP[17:24], D\_IP4=D\_IP[25:32]$

$S\_Port1=S\_Port[1:8], S\_Port2=S\_Port[9:16]$

$D\_Port1=D\_Port[1:8], D\_Port2=D\_Port[9:16]$

S_IP[1:8]表示源 IP 地址 S_IP 的第 1 到第 8 位，其他依次类推。

2)对步骤 1)得到的字节段进行平方运算，为了保证各字段平方的结果位数一致，对平方得到的结果只取低 8 位。例如对于 S_IP1 字节段平方截断之后表示为 $S\_IP_8^2$。

3)对步骤 2)的结果进行异或运算，如公式(8)所示：

$$HASH(S\_IP,D\_IP,S\_Port,D\_Port)=S\_IP1_8^2\otimes S\_IP2_8^2\otimes S\_IP3_8^2\otimes S\_IP4_8^2\otimes D\_IP1_8^2\otimes D\_IP2_8^2\otimes D\_IP3_8^2\otimes D\_IP4_8^2\otimes S\_Port1_8^2\otimes S\_Port2_8^2\otimes S\_Port3_8^2\otimes S\_Port4_8^2\otimes D\_Port1_8^2\otimes D\_Port2_8^2\otimes D\_Port3_8^2\otimes D\_Port4_8^2 \quad (8)$$

该哈希算法将数据包的四元组信息的每一比特位都参与了运算,且异或操作的操作数取的是平方值,能够充分避免哈希冲突的产生。该哈希算法只涉及简单的位运算和平方运算,时间复杂度为 $O(1)$。双哈希负载均衡算法的程序设计流程图如图 4 所示:

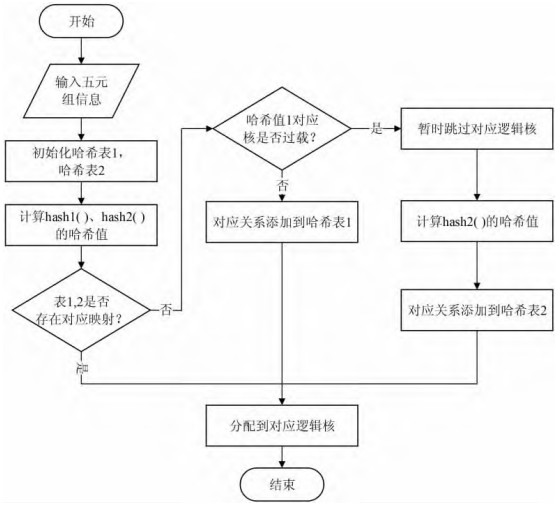

图 4 双 hash 负载均衡算法程序设计流程

从图 4 可以看出当接收到数据包,先是获取数据包的五元组信息,建立两个哈希映射表记录链路的哈希值和接收队列的关系。对于一个数据报文,通过对称 RSS 和对称 hash 函数对五元组信息进行哈希计算,通过遍历两个哈希表判断当前的哈希值是否存在,存在说明是已记录的映射关系,直接分配到对应的队列。不存在说明是新的链路数据流,如果对称 RSS 哈希计算得到的哈希值对应的逻辑核的 CPU 负载小于或等于分配的逻辑核的平均负载,则将哈希值和队列的映射关系添加到哈希映射表 1 中,该数据流被分配到哈希值对应的逻辑核上。如果当前逻辑核负载大于当前的平均 CPU 负载,暂时跳过该逻辑核,将该数据流分配到对称哈希散列值对应的逻辑核上,同时将该哈希值和队列的映射关系更新到哈希表 2 中。

获取各个逻辑核的平均 CPU 负载情况是整个双 hash 负载均衡算法的关键,它的准确率直接影响了分流的均衡效果。为了尽量减小 CPU 负载的误差,该系统通过前后两个时间点的各个 CPU 使用率的差值来衡量平均 CPU 负载情况。通过访问文件系统/proc 下的 stat 文件来获取各个逻辑核的用户时间 user time、系统时间 system time、等待时间 waiting time、空闲时间 idel time、系统优化调整时间 nice time、硬件中断时间 hard irq time、软件中断时间 soft irq time 和强制等待时间 steal time。这样就可以获得 CPU 的总的时间 total,其关系如公式(9),这样通过前后两个时间点 t_1、t_2 之间的 CPU 负载情况可以确定当前的 CPU 平均负载情况 cpu_usage ,如公式(10)所示:

$$total=user+system+waitiong+idle+nice+hard\ irq+soft\ irq+steal \quad (9)$$

$$cpu\_usage=((user2+system2+nice2)-(user1+system1+nice1))/(total2-total1) \quad (10)$$

3 实验与分析

3.1 实验平台

搭建的数据包捕获系统主要由一台服务器、一台 Lenovo 的 PC 机、一台 TestCenterC1 思博伦测试仪和一台显示器组成,整个系统的网络拓扑结构如图 5 所示:

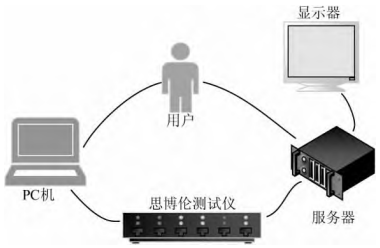

图 5 系统网络拓扑图

本次实验 DPDK 用的版本是 19.11,服务器的配置参数如表 2 所示。

表 2 服务器配置信息

配置信息	配置值
CPU 型号	Intel Xeon CPU E5-2640 V4
CPU 频率	2.4GHz
物理核/逻辑核	2/20
内存	64G
网卡	Intel X710 系列
操作系统	Centos7.0
内核版本	Linux 3.10.0-1160.25.1.el7.x86_64

3.2 实验结果

按照上述的系统网络结构拓扑图搭建硬件测试环境,首先测试了在单核单队列实现数据包捕获的场景下,流量速率分别在 1 Gbps、2 Gbps、5 Gbps、8 Gbps 和 10 Gbps 情况下 CPU 的负载情况,如图 6 所示:

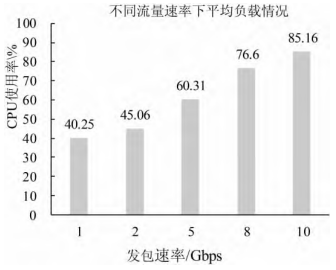

图 6 不同速率下单核的负载情况

从图 6 可以看出,使用 DPDK 进行数据包捕获,随着发包速率增大,逻辑核的负载情况不断加重。对于 1 Gbps、2 Gbps 的流量速率,系统的 CPU 使用情况低于 50%,系统的性能相对稳定。但对于 8 Gbps、10 Gbps 的流量速率,系统内核处于过载状态,一般需要避免这种 CPU 负载长期高于 60%的情况。该实验结果也证实对于高速网络环境引入负载均衡技术改善系统的性能的必要性。

目前基于 DPDK 的负载均衡技术有 RSS、对称 RSS 等。目前性能较好的对称 RSS 虽然解决了同一链路上的双向数据流分配到同一逻辑核上处理的问题,但是负载均衡效果不太理想。实验在使用 4 个逻辑核用于数据包捕获的场景下,配置了对称RSS,通过测试仪发送 10 Gbps 的 128B 的数据包,通过比较多核处理器各个逻辑核的使用率来对结果展开衡定与测量,得到的实验结果如图 7 所示:

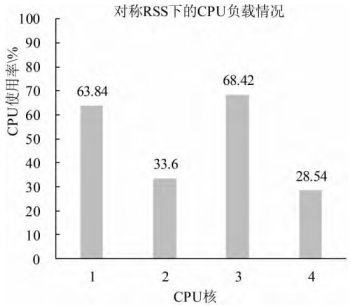

图 7 对称 RSS 下的 CPU 使用情况

从图 7 可以看出在 10 Gbps 速率下,使用对称 RSS 算法

(下转第 70 页)

示,局部位置与跟随曲线如图 7 所示。从反馈数据可以看出从站电机能够跟上给定位置,符合预期目标。

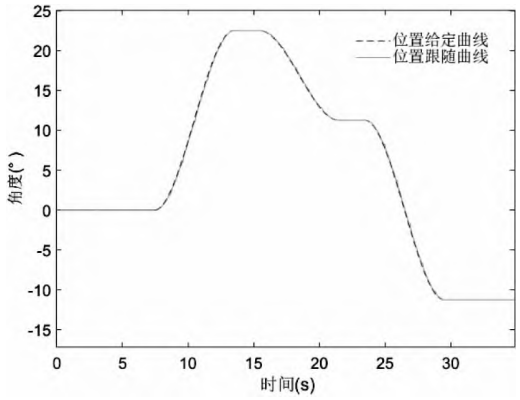

图 6 第五轴整体位置给定与反馈曲线

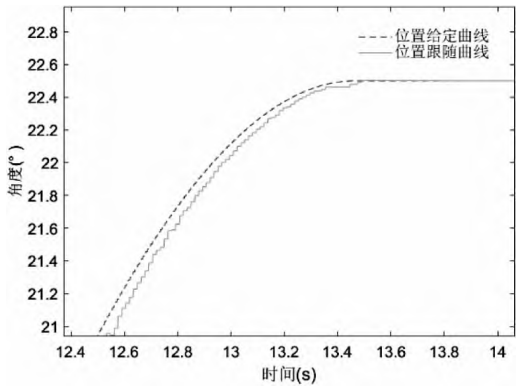

图 7 第五轴局部位置给定与反馈曲线

3 结束语

针对机器人控制系统底层协议不开放、机器人控制算法不开源等问题,设计了一种基于 EtherCAT 和 Matlab 的协作机器人控制系统。采用 Matlab 下 Simulink 设计机器人控制算法,并通过 EtherCAT 主站将运动数据发送给各关节,同时关节数据也能实时采集显示、实现曲线绘制、离线分析等功能。最终在机器人开放式结构中验证了系统的可行性和稳定性,同时实验证明运动效果良好,达到了预期的设计要求。

参考文献

[1]潘炼东.开放式机器人控制器及相关技术研究[D].武汉:华中科技大学,2007
[2]包光旋,黄家才,李耀,等.基于视觉的并联机器人智能分拣系统设计与实现[J].南京工程学院学报(自然科学版),2021,19(1):7-11
[3]袁俊杰,刘海涛,潘国庆,等.基于 EtherCAT 和 TwinCat3 的协作机器人控制系统设计[J].科学技术与工程,2021,21(8):3159-3168
[4]刘海涛.六自由度协作机器人控制系统设计研究[D].北京:北方工业大学,2021
[5]王惠娇.嵌入式平台 EtherCAT 主站的实现[J].计算机应用,2018,38(S1):165-169
[6]谢锴.基于 EtherCAT 总线的多轴控制系统设计[D].杭州:浙江大学,2017
[7]冯文.基于 EtherCAT 通信的多轴运动实时控制系统研究与设计[D].镇江:江苏科技大学,2021
[8]郑林,何岭松,陈驰.基于 EtherCAT 的机器人教学实验平台设计[J].实验室研究与探索,2021,40(5):9-13,30
[9]段超,黄家才,包光旋,等.基于 Matlab 的开放式六自由度机器人控制系统研究[J].南京工程学院学报(自然科学版),2021,19(2):25-29

[收稿日期:2022-07-11]

(上接第 67 页)

各个逻辑核的 CPU 使用情况相对于单核场景下 CPU 使用率 85.16%都有所降低,但是出现了某些逻辑核负载过重、流量分配不均衡的问题,图 7 中逻辑核 1 和逻辑核 2 的 CPU 使用率相差的近 40%,10 次测试结果的 CPU 使用率的均方差为 334.48。在相同的测试环境下,通过双 hash 负载均衡算法得到的 CPU 使用情况如图 8 所示:

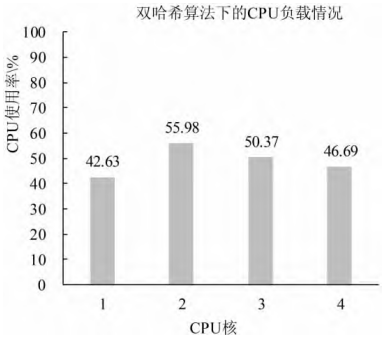

图 8 双 hash 负载均衡算法下 CPU 使用情况

从图 8 可以看出使用双哈希算法,各个逻辑核的 CPU 使用情况相对于单核场景下有了大幅降低,且相比于对称 RSS 分流机制,流量分配更均衡,基本在 50%左右小幅波动,10 次的测试结果的 CPU 使用率的均方差仅为 23.92,所以表明该双 hash 负载均衡算法对于当前高速网络环境负载均衡效果理想。

4 结束语

为了解决 DPDK 数据处理使用 RSS 和对称 RSS 出现负载不均衡问题,本文提出了一种双 hash 负载均衡算法,改善了 RSS 和对称 RSS 的负载不均衡。和对称 RSS 性能相比,双 hash 负载均衡算法在保证同一链路上的双向数据流分配给同一逻辑核的前提下,分流更均衡,10 次实验结果的 CPU 使用率的均方差仅为 23.92,而对称 RSS 的 CPU 使用率的均方差为 334.48。该算法对当前的高速网络环境下解决负载均衡问题具有一定应用价值。

参考文献

[1]LIN H, YAN Z, CHEN Y, et al. A survey on network security-related data collection technologies[J]. IEEE Access, 2018, 6: 18345-18365
[2]FERDIANA R. A systematic literature review of intrusion detection system for network security: research trends, datasets and methods [C]//2020 4th International Conference on Informatics and Computational Sciences (ICICoS). IEEE, 2020:1-6
[3]黄冰.基于 DPDK 的高性能负载均衡系统设计与实现[D].西安:西安电子科技大学,2015
[4]WOO S, PARK K S. Scalable TCP session monitoring with symmetric receive-side scaling [R]. Daejeon, Korea: KAIST, 2012: 163-169
[5]唐宏,柴卓原,任平,等.DPDK 应用基础[M].电信科学,2016,32(8):170

[收稿日期:2022-05-09]