高技术通讯 2024 年 第 34 卷 第 11 期:1178-1188

doi:10.3772/j.issn.1002-0470.2024.11.005

LightShaper:一个高精度通用流量整形工具①

王浩昆② 张文力③ 张 钊 陈明宇

(中国科学院计算技术研究所 北京 100090)

摘 要 模拟流量通常是系统能力与服务质量评估和优化的常见手段,而常用工具往往存在硬件模拟成本高、不灵活、软件模拟流量特征精度差的问题。如何低成本精确再现负载流量特征是一个挑战。为此,本文提出了一种高精度、低成本的方法实现对测试负载的流量特征控制。使用独立的流量特征控制中间件 LightShaper 对流量负载进行缓冲、汇聚、同步,并使用一种基于占位包填充的软件方法,在标准网卡上实现了高精度的包间隔控制。LightShaper 能按需塑造流量负载的速率波形与包间隔分布波形,提供了流量整形、多源同步、延迟变换、流内乱序变换等不同功能,达到接近硬件的包间隔控制精度。在随机包间隔分布测试中,LightShaper 在均方误差和 KL 散度这 2 个分布拟合度指标上,相比传统基于计时器的软件方法,分别实现了 6.67 ~ 28.10 倍和 2.01 ~ 18.88 倍的提升,便于低成本灵活复现评测负载。

关键词 流量整形;速率控制;测试负载;包间隔分布

在数据中心内部,构建一个负载测试环境以精确测量服务质量指标,是一个充满挑战的任务。通常被测的高并发服务会由分布式微服务组成,这些微服务在数据中心服务器集群内运行,其中位数延迟通常在亚毫秒到微秒级别[1-4]。对这类的系统进行延迟归因等测试和分析时,需要在模拟负载时提供同级别精度的速率和包间隔分布等流量特征。以检测传输层中许多拥塞控制算法为例,拥塞算法会根据实际传输速率和数据包的包间隔判断网络的拥塞情况。如果被测负载的流量特征不准确,可能会导致拥塞控制验证的失败,导致目标网络性能下降、数据传输延迟增加以及丢包等各类问题的发生[5-6]。

目前,附带流量整形的负载发生器有硬件和软件两种不同的实现方式。硬件方式具有高精度的优势,精度级别可以达到纳秒级别[7],但相对于可自编程的软件方式,它们昂贵、闭源、专用,且灵活性差、成本较高[8]。因此目前许多软件负载发生器被广泛使用,例如 Pktgen-DPDK[9]、tRex[10]、Iperf[11]、Netperf[12]和 MCC(massive client connections)[13]等。它们均采用软件整形技术来控制发送流量的特性,使用例如漏桶、令牌桶和时间轮等不同的软件控制速率模型对发送流量进行整形。为了提升软件对发出负载模型的精度控制能力,软件负载发生器会采用多种不同的优化手段,例如,通过采用颗粒度更细的定时器控制检测速率的时间,确保整形包间隔的时间精度[14];通过将单层控速转为多层控速整形,平滑整形效果,减少负载流量突发带来的负面影响[15];通过减少整形过程对中央处理器(central processing unit,CPU)资源的需求,降低整形与应用负载处理的开销冲突风险,提升整形效果的精确度[16]。这类优化手段会导致以下 3 个问题。(1)现有的整形系统无法控制数据包向网卡交付之后发送的处理方式。网卡驱动为了减少外围设备互连快速通道(peripheral component interconnect express,PCI-E)

① 国家重点研发计划重点专项(2023YFB4503904)资助项目。
② 男,1998 年生,硕士生;研究方向:计算机网络,流量特征控制;E-mail:wanghaokun20g@ict.ac.cn。
③ 通信作者,E-mail:zhangwl@ict.ac.cn。
(收稿日期:2023-07-25)

的读写延迟对传输性能的影响[17],一般要求以批处理的方式来发送数据包。使用批处理发送数据包能提升传输性能,但同时导致数据包在发送后的包间隔不可控。(2)应用构造正确的交互内容需要通过多层次的软件栈来保证,这中间会产生大量的不可控开销[18],导致包间隔的精度无法得到保障。(3)大规模负载需要通过分布式客户端集群产生与聚合,在分布式环境中,由于多节点的流量传输和聚合,流量特征的不确定性增加,从而增加了在各个终端节点上对流量特征进行精准控制的难度。

为解决上述问题,本文将生成测试负载的内容生成部分与流量特征控制部分解耦,通过单独中间件进行流量特征控制。本文还提出了一种基于占位包填充的流量整形方法,在软件层面实现了负载流量的包间隔精准控制功能。与传统软件控速技术相比,本文主要贡献如下。

(1)提出一种将测试负载的内容生成部分与流量特征控制部分解耦的方法,构建了一个流量特征控制中间件的软件原型 LightShaper,使测试人员能够在不修改原有负载内容生成部分代码的情况下,构建具备精准流量特征控制能力的负载测试环境。

(2)设计并实现了一种采用标准网卡的基于占位包填充的软件包间隔控制方法,并通过实验验证了其可以达到亚微秒级别精度,接近硬件的包间隔控制能力。

1 背景与挑战

现有的测试负载生成方式可分为硬件方式与软件方式 2 大类。硬件方式能够提供高性能的计算处理能力和稳定的测试结果,从而提高负载生成的速度和准确性。在硬件生成负载的方法中,基于专用设备(如 IXIA[19]、Sprient[20]等公司生产的专用硬件设备)可以实现极高的发包速率和精确的速率控制。然而,专用设备具有封闭的接口和专有系统的特性,限制了其进行高度自定义开发的能力。此外,这些设备的昂贵价格远远超出了大多数负载测试的预算范围。基于开源硬件(如 NetFPGA[21])也能实现高速高精度的数据包发包,并具备一定的扩展性。然而,由于现场可编程门阵列(field programmable gate array,FPGA)本身编程较为困难,资源和功能的支持相对有限,因此主要应用于研究和教学领域,难以广泛应用于实际场景。总而言之,硬件方式开发迭代困难的特点使其难以满足测试对负载内容的需求。专有硬件和开源硬件上扩展这些内存资源的成本远高于通用硬件,这使得硬件方式也难以满足测试对并发度的要求。

软件方式具有灵活与低成本的特点,相比硬件实现方式,软件方式在可移植性、功能可扩展性、开发成本与易用性等方面具有优势,如 Iperf、Netperf 等软件被广泛使用。软件方式通过内核提供的网络接口或用户态驱动、用户态协议栈等内核旁路方式提供的网络接口来收发数据包,软件方式生成的负载支持协议交互和事务交互,能满足对负载内容的需要。但由于构造复杂负载内容的过程引入了事务处理与协议栈调用等开销[22],不能以确定的时间生成数据包,就无法用确定的包间隔或速率发送,难以满足负载测试对流量特征进行精准控制的需要。

最后在面向大规模用户的场景下,服务端承受的并发度峰值可达到千万乃至上亿级[23],而单机平台受到资源瓶颈的限制[24],难以生成满足并发度的测试负载,这要求流量负载生成系统具备多机扩展的能力,通过分布式集群产生大规模测试负载,这进一步对产生精确的负载流量造成了困难。

解决这些问题主要面临 3 个挑战。

(1) 测试负载方案中负载内容生成部分和流量特征控制部分的高度耦合,不利于进行流量特征控制。正如前文所述,复杂负载的生成时间具有不确定性,而流量特征控制需要将负载以预设的时间间隔发送,存在对确定的负载生成时间的需求,这个固有矛盾使得对复杂负载进行流量特征控制存在困难。

(2) 负载测试环境中分布式集群的流量传输与汇聚会进一步增加流量特征的不确定性,这使得即便在分布式节点上进行了流量特征控制,汇聚负载的流量特征也往往不能恰如预期。

(3) 基于软件的流量特征控制方案精度不足。软件方案虽然在开发灵活性和泛用性上有优势,但

由于无法控制硬件底层的开销和排队时延，导致难以进行高精度的包间隔控制。在数据包实际发送过程中，CPU 将数据包交付给网卡与网卡发送数据包是异步过程，即使调用最接近底层的 I/O 接口实现，即便 CPU 对数据包交付给网卡的时间间隔进行了尽可能精准的控制，但由于标准网卡队列按批对数据包进行发送，并不对包间隔作保证，因此无法精准控制包间隔。其过程如图 1 所示。网卡的批处理发包在高速网络下是无法避免的，这是由于在高速网络中，操作系统与网卡之间的 PCI-E 通信开销是不可忽视的。为了实现 128 B 大小数据包的 10 Gbps 的传输速率，需要每 140 ns 左右的时间发送一个数据包，但一次单独的 PCI-e 写入操作可能需要高达 900 ns 左右的开销[17]。为了提高 PCI-E 通信的效率，现代网卡及其驱动程序只能按照突发的方式进行批量发送处理。若需对数据包的间隔进行精确控制，改为对每个数据包实施独立处理，则将导致可实现的带宽显著降低。

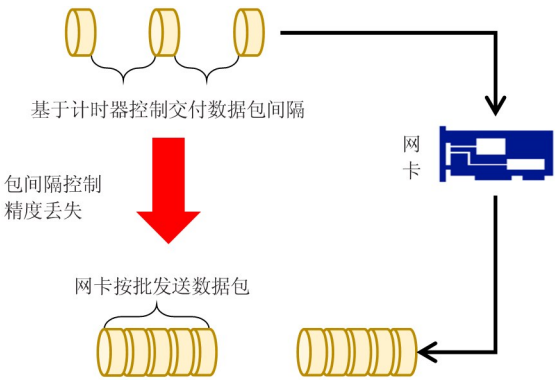


图 1　基于计时器的软件包间隔控制方法

2　设　计

为解决上一节提到的挑战，本文基于 2 个核心思想进行设计。首先，解耦负载内容生成与流量特征控制，通过独立的转发中间件来管理负载的流量特征。这种解耦方法有助于消除负载生成时间的不确定性，满足对确定负载生成时间的需求（挑战 1），同时也能对分布式集群的汇聚流量进行统一的速率与包间隔控制，避免分布式节点流量汇聚造成的不确定性（挑战 2）。此外，采用基于占位包填充的软件包间隔控制方法，通过标准网卡来实现亚微秒精度的包间隔控制（挑战 3）。使用这 2 种方法不仅降低了负载流量的测试成本，使研发人员更专注于事务交互部分的代码，还解决了基于软件的流量特征控制方案精度不足的问题。

综上，通过解耦与占位包填充的方法，系统能够更好地控制流量特征，同时降低负载生成器和测试成本并提高精度。基于以上思想，本文设计并实现了流量特征控制中间件 LightShaper。LightShaper 通过流量缓冲和包调度实现流量特征控制，并且通过占位包填充方法在标准网卡上实现高精度的包间隔控制。

2.1　整体架构

LightShaper 整体架构如图 2 所示，由数据包收包处理模块、流量变换处理以及数据包发送处理模块组成，同时需要外部能有占位包丢弃的能力进行支撑。整体系统实现了将复杂的混合流量按需精确整形为特定波形的能力。

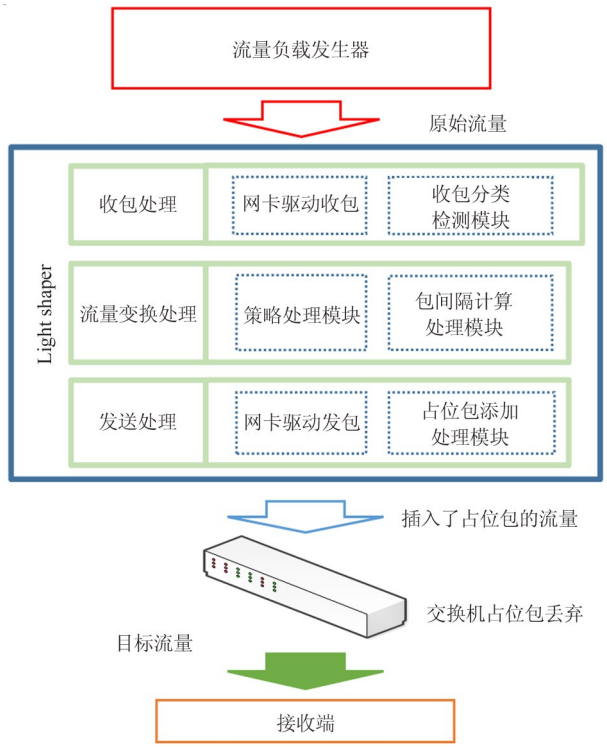


图 2　系统整体架构图

收包处理模块，主要负责数据包的接收以及数据包检测与分类功能，负责从数据包中提取并检测特征信息，并将这些信息与分类规则进行匹配，根据不同的处理策略，将数据包交付到后续相应的流量

变换处理部分。这些用于分类的数据包特定信息包括五元组、特定报头域内容、应用层负载的内容与标志位等,也可以由用户自行定义。为最小化数据包传递的开销,分类部分与流量变换部分间的数据包传递通过无锁指针队列进行。

流量变换处理部分包括速率变换、包间隔分布变换、延迟变换、流内乱序变换和多源同步变换等不同的数据包处理策略。其中延迟变换模拟策略的主要目的是模拟广域网高延迟环境,数据包添加时间戳并缓存至最小堆,延迟一段时间,等待数据包发送时间到期后数据包才放入待发送缓冲区进行发送。流内乱序变换策略的主要功能是模拟广域网传输路径不同导致的流内乱序,每条流的数据包根据哈希算法映射到对应存储结构,按预设策略调整顺序放入待发送缓冲区。最后多源同步变换策略解决了分布式集群多源客户端数据流量汇聚顺序问题,数据包通过多队列缓冲与调整,按预期顺序放入待发送缓冲区。

发送处理部分包括网卡驱动发送以及占位包填充 2 个功能。其中占位包的填充需要根据流量变换处理中包间隔分布的计算结果进行。

2.2 流量整形

LightShaper 的主要功能是流量整形,即对到达被测系统流量负载的速率与包间隔分布进行塑造,这分别对应速率变换策略与包间隔分布 2 种变换策略。在实现流量整形的过程中,需要将输入流量的不确定到达时间转化为以预期时间序列精确发送的输出流量。LightShaper 通过控制缓冲计时与包间隔时间来实现这一目标。

缓冲计时为 LightShaper 接收到待整形流量与开始发送之间的等待时间。LightShaper 在缓冲计时期间接收并缓冲获取足够的数据包,缓冲计时结束时,缓冲的数据包以预期波形发送。缓冲计时是测试的关键参数,当缓冲计时过长时一方面会加大通信的延迟,进而扰乱客户端与服务端间的正常协议交互;另一方面缓冲计时过短时则无法获取足够的数据包以构造理想的预期波形。为确定最佳缓冲计时长度,待整形流量波宽可以通过 LightShaper 实时输出观察,确认实际需要的缓冲大小。

包间隔时间定义为发送时间序列上 2 个相邻数据包之间的间隔时间。为了计算包间隔时间,目前采用了 2 种变化策略。第 1 种策略是包间隔分布变换,其中 LightShaper 使用逆变换法从预期分布生成随机数表。在计算包间隔时间时,它从随机数表中均匀地随机取值,并通过平移和放缩来获取指定区间内符合预期分布的包间隔时间。第 2 种策略是速率变换,其中 LightShaper 将预期速率与网卡线速的比例映射为数据包发送时间与包间隔时间的比例。通过这种映射,可以计算出预期速率下的包间隔时间。这 2 种策略都旨在确保数据包的发送时间序列具有所需的分布和速率特征。

2.3 包间隔控制

在计算出包间隔时间之后,如何将预期包间隔时间精确地转化为网卡最终发送数据包时的包间隔是目前主要的挑战。第 1 节讨论了传统基于计时器的软件方法通过控制调用发包接口间的时间间隔来控制包间隔不够精准的原因,标准网卡需要按批对数据包进行发送,并不对包间隔作保证,因此不能精准控制包间隔。

LightShaper 使用了一种称为“占位包填充”的方法来控制包间隔。这种方法利用了标准网卡的 2 个基本机制。首先,标准网卡队列不会改变相邻数据包的发送顺序。其次,网卡在全速传输时,数据包间的间隔总是以以太网帧的间隙长度进行发送。因此,占位包填充的基本原理是在正常数据包之间插入对应包间隔时间的不定长占位包,使得到达目标节点或交换机的数据包成为正常数据包与占位包的混合数据包。通过在接收端进行配置,占位包可以被直接丢弃或由交换机转发至非目标端口,从而确保只有正常数据包到达接收端,实现精准控制包间隔的效果。此时,包间隔时间的大小等于网卡线速发送时当前长度的占位包的发送时间与固有帧间隙时间之和。由于这种方法借助了网卡底层机制在硬件层进行包间隔控制,因此能够准确地保证包间隔时间的准确性。整个包间隔的控制处理流程如图 3 所示。在 LightShaper 中,构造占位包的方法是赋予占位包一个特定的目的媒体存取控制(media access control,MAC)地址。将该 MAC 地址配置为交换机

的黑洞 MAC,使得占位包在到达交换机时就能被交换机直接丢弃。这样便实现了精准控制包间隔的效果。

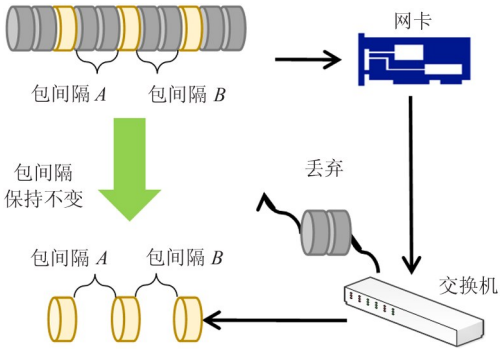


图 3　占位包填充方法示意图

3　实　现

本文利用数据平面开发套件(Intel Data Plane Development Kit,DPDK)实现了流量特征控制中间件 LightShaper 的软件原型[25],本节对 LightShaper 的架构实现(3.1节)与功能实现(3.2节)进行阐述。目前,LightShaper 相关代码已开源发布在 https://gitee.com/acs-dcn/lightshaper。这一创新方法具有广泛的潜力,能够满足不同场景的需求,为实现高效的网络负载测试提供了新的途径。

3.1　架构实现

LightShaper 采用了多线程流水线架构,每个线程都独占一个 CPU 核。线程间的数据包传递通过无锁队列进行,避免了争用导致的性能开销。其逻辑实现过程如图 4 所示。在分类模块中,LightShaper 由接收线程和分类线程组成。借助于 DPDK 的高效数据包 I/O 性能,该部分实现了 10 Gbps 线速的数据包接收与分类。在流量变换部分,LightShaper 包括延迟变换、流内乱序变换、多源同步变换等处理线程。最后在发送部分,LightShaper 由一个决策线程和一个发送线程组成。决策线程持续轮询待发送队列中的数据包数量,决定即时发送速率。发送线程根据即时发送速率或包间隔分布持续获取下一次发送数据包的时间间隔,将其映射为占位包的长度,并在正常数据包之间填充占位包并发送。

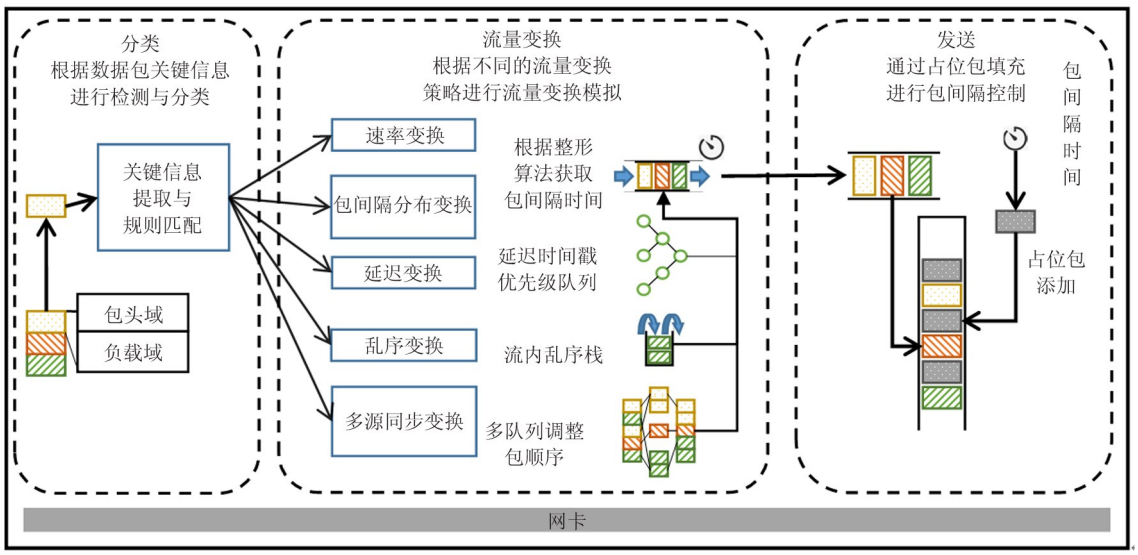


图 4　LightShaper 逻辑架构结构图

3.2　功能实现

LightShaper 具备流量整形功能,能够对速率和包间隔分布进行调整。在速率控制方面,支持设定线速万分之一精度的发送速率,并支持常速率波形、多突发波形与渐变速率波形 3 种速率整形模式。在包间隔控制方面,支持设定亚微秒级别精度的包间隔,并支持固定间隔与随机分布间隔 2 种包间隔分布整形模式。此外,LightShaper 还支持多源同步变换,在分布式集群环境下进行流量汇聚与整形时,通过多缓冲区管理和多源同步功能,可以将来自不同客户端的波峰顺序按用户设定进行调整,产生可控的聚合负载。同时,LightShaper 也提供了延迟变换

与流内乱序变换2种流量变换功能来模拟网络抖动状况。在流内乱序变换上,实现了对指定流进行特定比例的乱序处理。在延迟变换上,实现了固定延迟与随机延迟。在随机延迟中,提供了附带相关系数的随机波动延迟与符合统计分布模型的分布延迟2种随机方式。

为了使流量整形中的包间隔分布与网络抖动模拟中的延迟分布更好地贴合实际负载,LightShaper提供了正态分布、卡方分布与泊松分布3种随机分布模型用于包间隔分布与延迟分布,并对用户通过类似NistNet[26]的方式自行构造分布提供了支持。通过读取用户提供的实验数据文件或分布模型文件,理论上LightShaper可以接入任意分布。

4 实验评估

本节旨在评估LightShaper的功能,并针对以下问题展开讨论。

(1)LightShaper使用的占位包填充方法与传统的软件负载发生器的控速功能相比,其精度如何?

(2)LightShaper进行包间隔分布整形的实际效果如何?

(3)LightShaper解耦后,对现有负载应用的流量整形能力如何?

为此本实验搭建了一个"客户端-转发端(LightShaper)-服务端"架构的测试平台。流量负载从客户端发出后经交换机到达转发端,转发端流量特征控制之后的流量负载经交换机发送到服务端,通过专用硬件设备IXIA网络测试仪在转发端出口处通过镜像端口的方式捕获流量负载,并对其流量特征数据进行分析。在具体硬件配置上,客户端与服务端配备了2个Intel Xeon Gold 5118 2.30 GHz 12核CPU、128 GB RAM和10 Gbps标准网卡的服务器,转发端搭载LightShaper,配备Intel Xeon Gold 6130 CPU 2.10 GHz 16核CPU 128 GB RAM和10 Gbps标准网卡的服务器。服务器的操作系统均为CentOS 7,内核版本3.10.0-957.el7。

4.1 流量整形精确度测试

在流量整形精度验证测试中,本文采用了LightShaper、Pktgen-DPDK与tRex这3种方案进行测试。为了确保公平性,本文让它们以相同的预设速率或者包间隔向IXIA网络测试仪发送数据包。其中,Pktgen-DPDK和tRex作为对照组,用于与LightShaper进行整形精度的对比。根据Pktgen-DPDK和tRex源码可以得知,其中Pktgen-DPDK采用了令牌桶模型进行速率控制,而tRex则采用了时间轮的模型进行速率控制。

在流量吞吐精度测试中,本文使用LightShaper、Pktgen-DPDK以及tRex这3种不同的方式向IXIA网络测试仪发送5 Gbps固定速率的数据包,并对测试仪端抓取的流量记录进行了后续分析,其结果如图5所示。

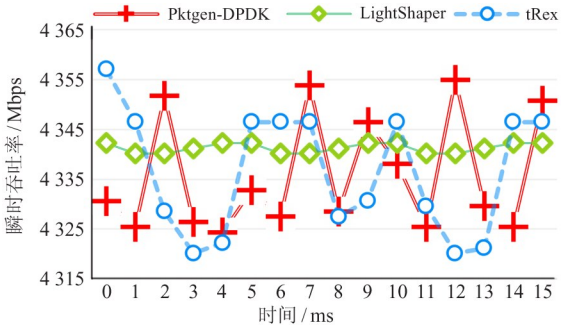


图5 速率整形波动性对比

实验结果显示LightShaper发出的测试负载比特率的最大值与最小值的极差值为2 112 kB·s^{-1},而Pktgen-DPDK发出的测试负载比特率极差值为30 624 kB·s^{-1},是LightShaper的14.5倍。tRex发出的测试负载比特率极差值为55 300 kB·s^{-1},是LightShaper的26.1倍。此外,LightShaper的包间隔标准差为243 ns,相比Pktgen-DPDK的包间隔标准差676 ns降低了64%,相比tRex的包间隔标准差921 ns降低了73%。从测试结果来看,LightShaper在速率精确度方面表现出最佳性能,tRex表现最差。

在进行固定包间隔控制精度测试中,IXIA网络测试仪接收来自LightShaper与tRex的发送流量,测试10万个长度在64~1 500 Byte之间随机大小的数据包的包间隔精度。LightShaper与tRex以预期10.0 μs的包间隔大小的流量送入IXIA网络测试仪

中进行检测。表 1 呈现了使用 IXIA 网络测试仪捕获的 Trace 的测试结果。通过采用占位包方法获得的包间隔的实际均值为 10 000.4 ns,与预期包间隔 10 μs 之间的差异仅为 0.4 ns。相比之下,使用 tRex 进行测试的结果为 9 996.0 ns,与预期包间隔差异为 4.0 ns,这一误差值是使用占位包方法所产生误差的 10 倍。此外,占位包方法的测试结果表现出更小的波动性。具体来说,占位包的最大最小包间隔极差值为10.0 ns,这一数值远低于 tRex 测试结果的 310.9 μs。另外,占位包包间隔的实际标准差为 3.4 ns,只有 tRex 测试结果的 1/1 500。这些数据进一步证实了使用占位包方法在网络性能测试中的优越性和准确性。

表 1　固定包间隔测试数据对比

包间隔统计量	包间隔控制方法	统计值/ns
最大最小极差值	LightShaper	10.00
	tRex 整形	310 953.01
标准差	LightShaper	3.40
	tRex 整形	5 192.22
均值	LightShaper	10 000.40
	tRex 整形	9 996.69

图 6 提供了更为直观的比较,展示了 tRex 和 lightShaper 在包间隔控制方面的差异。观察使用时间轮的 tRex,可以发现大多数包的包间隔极低。为了维持平均包间隔在 10 μs,tRex 会在发送固定数量的数据包后产生一个较大的包间隔。相比之下,LightShaper 能够自始至终保持所有包的包间隔稳定在 10 μs 级别,不会有巨大的起伏变化。LightShaper

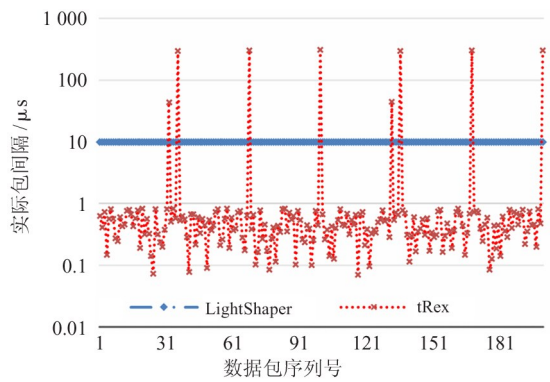


图 6　包间隔变化图

的占位包技术使其能够实现更为精确的包间隔控制能力。

4.2　包分布整形功能精度测试

本实验主要验证通过 LightShaper 能否生成符合特定统计分布的包间隔分布波形。通过 IXIA 网络测试仪向 LightShaper 节点发出 10 万个大小为 64 Byte 的数据包,LightShaper 按照占位包与 DPDK 高精度定时器 2 种方法,分别构建均值为 100 μs、分布范围在 80 ~ 120 s 的卡方分布与泊松分布的包间隔进行数据包转发(为凸显 3 种分布形态上的差别,本实验选用 λ 值为 2 的泊松分布、自由度为 6 的卡方分布进行测试)。

本实验通过双采样 KS(Kolmogorov-Smirnov)校验法[27]对实际测试出的包间隔分布数据与理论分布进行校验。KS 校验法可以量化样本的经验分布函数与参考分布的累积分布函数之间的距离,或者 2 个样本的经验分布函数之间的距离。KS 校验中,当 2 个样本累积分布函数差值的上确界 D 小于界限值 $D_{n,\alpha}$时,KS 校验认为两样本服从同一分布,其中 n 为样本量,α 为显著性水平。在卡方分布中,D 值为 $0.004\,092 < D_{100\,000,0.05} = 0.006\,073$。在泊松分布中,$D$ 值为 $0.004\,461 < D_{100\,000,0.05} = 0.006\,073$。二者均可认为在置信度为 95% 的情况下,获得的实际包间隔实际分布与理论分布服从同一分布。

为了进一步证实使用占位包技术相对于常规计时器方案在精确度上的优势,本文采用了均方误差(mean square error,MSE)来衡量实际分布与理论分布之间的距离,以及 KL 散度(Kullback-Leibler divergence,KL)来衡量实际分布与理论分布之间的相似度。测试对比的结果表明,使用占位包方法生成的包间隔分布与理论分布的拟合度指标相比计时器方法有显著的提升。如表2所示,使用占位包方法

表 2　分布拟合度指标对比

分布模型	控制包间隔方法	*MSE*	KL 散度
卡方分布	占位包	1.65×10^{-6}	2.98×10^{-3}
	计时器	1.10×10^{-5}	6.01×10^{-3}
泊松分布	占位包	5.69×10^{-6}	5.59×10^{-4}
	计时器	1.60×10^{-4}	1.06×10^{-2}

在 MSE 上的提升分别达到了 6.6 ~ 28.1 倍,在 KL 散度上的提升达到了 2.01 ~ 18.88 倍。通过使用占位包技术,可以获得相比常规计时器方案更高的精确度提升。

4.3 实际应用接入测试

本实验使用实际负载进行实验,验证了用户态大规模并发连接(massive concurrent connection,MCC)负载发生器[13]产生的多源同步负载以及验证内核态 Netperf 客户端负载流量变换测试。

MCC 是一个物联网场景下的软件负载发生器,支持开环、半开环、闭环多种工作模式,在本实验中使用开环工作模式。服务端软件为 iotepserver[28],iotepserver 是一个物联网服务器测试基准,它服务于高并发的突发工作负载。在本实验中,客户端使用 3 台机器组成分布式集群向服务端发出负载,LightShaper 通过多源同步变换,将原本来自不同源客户端的流量顺序按预期进行调整,使每个客户端的数据包在汇聚波峰中以单独的波峰形式存在,多源同步变换的效果如图 7 所示。

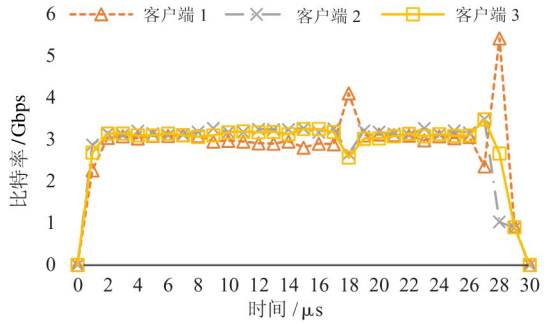
图 7(a)为分布式环境下,通过 LightShaper 进行多个波峰汇聚与整形后的速率图像,图 7(b)为在汇聚与整形的基础上进一步进行波峰错峰同步的速率图像。该实验证明在分布式集群环境下,LightShaper 的多源同步变换能有效地将不同客户端的波峰顺序调整,有效地实现了预期的流量变化。

Netperf 是惠普公司开源的一种常见内核态网络性能的测量工具,主要针对基于传输控制协议(transmission control protocol,TCP)或用户数据报协议(user datagram protocol,UDP)的传输。Netperf 可以进行批量数据传输(bulk data transfer)和请求/应答(request/reponse)2 种不同的测试模式。测试 LightShaper 的延迟变换与流内乱序变换功能对实际负载造成的影响,本实验选用了 Netperf 提供的 3 种具有代表性的 TCP 流量负载。

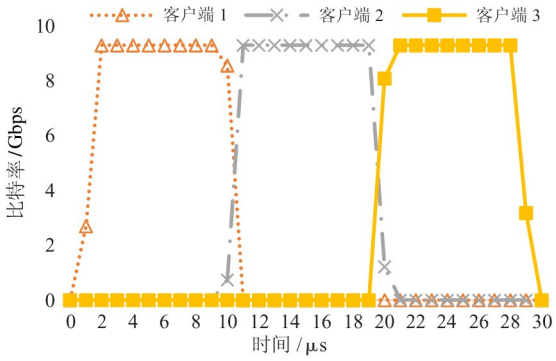
(1) 基础 TCP 流测试(basic TCP stream test,TCP _ STREAM)。Netperf 通过建立多条 TCP 连接与服务器通信。在每条连接中,数据传输是单向且持续的批量传输,这代表了文件传输场景下的典型流量负载,例如互联网用户访问网盘等互联网存储工具的情况。

(2) TCP 请求/响应测试(TCP request/response test,TCP _ RR)。Netperf 通过建立一个单一的 TCP 连接与服务器通信。在单个连接内,测试以闭环模式持续发起请求和响应,这代表了分布式节点间通信场景下的典型流量负载,例如服务器集群中业务服务器访问 Redis 服务器的场景。

(3) TCP 连接重置请求/响应测试(TCP connection reset request/response test,TCP _ CRR)。该模式下 Netperf 与服务端建立批量 TCP 连接。在每个连接中,进行一次请求和响应后即关闭连接。这代表了 Web 访问场景下的典型流量负载,例如互联网用户访问门户网站的场景。



(a) 波峰同步前



(b) 波峰同步后

图 7 多源同步测试

本实验测试了单纯转发、添加单向固定 50 ms 延迟与添加 25% 流内乱序 3 种不同流量变换策略下的性能指标。其中,TCP _ RR 负载与 TCP _ CRR

负载的工作模式决定了每条流在同一时刻只有单个数据包在发送通路上,这种模式并不适用于添加乱序的测试,因此本文只对其测试单纯转发与添加单向固定延迟的流量变换策略。

本实验参数配置为:客户端与服务端的 TCP 收发缓冲区大小均为 32 MB,TCP _ RR 与 TCP _ CRR 的请求响应消息大小为 128 Byte,每次测试时长为 5 s,在 TCP _ STREAM 的流量负载下,吞吐量的单位为 Gbps,在 TCP _ RR 与 TCP _ CRR 的流量负载下吞吐量的单位为 TPS(transaction per second)。

延迟与流内乱序对于吞吐量的影响如表 3 所示。在单向固定 50 ms 延迟模拟中,TCP _ STREAM 负载的吞吐量下降至未加入延迟时的 10.4%;在 25% 流内乱序模拟中,客户端与服务端间的重传过程使 TCP _ STREAM 的吞吐量下降至未加入乱序时的 5.7%。

表 3 延迟与流内乱序对于吞吐量的影响

负载类型	流量变换策略	吞吐量/Gbps
TCP _ STREAM	无	9.16
	延迟	0.96
	乱序	0.53

延迟对于吞吐量的影响如表 4 所示。单向增加 50 ms延迟增加了通信的往返时延(round-trip time,RTT),在实验数据中对该效果体现最明显的是 TCP _ RR 的测试结果,TCP _ RR 负载的测试中,客户端需要获得上一次请求消息的响应后才能发出下一消息,RTT 从 0.05 ms 上升到 50.29 ms 使得该时延升高了 1 005 倍。

表 4 延迟对吞吐量的影响

负载类型	流量变换策略	平均时延/ms
TCP _ RR	无	0.05
	延迟	50.29
TCP _ CRR	无	0.14
	延迟	100.57

实验结果表明,LightShaper 能在不修改原始应用程序的前提下,有效实现预期的流量延迟与乱序变换功能。

5 结 论

本文针对面向大规模用户的高并发互联网服务难以构建流量特征满足测试要求的负载测试系统的问题,提出了一种高精准、低成本进行测试负载流量特征控制的方法,即通过流量特征控制中间件,对不确定的流量负载进行缓冲、汇聚与同步,再生成符合预期特征的精准负载流量。本文还提出了一种高精度、低开销的基于占位包填充的软件包间隔控制方法,可实现亚微秒级别的包间隔控制。

实验评估表明,本文基于 DPDK 实现的流量特征控制中间件软件原型 LightShaper,能按需塑造精准的速率图像与包间隔分布图像,能在分布式环境下对来自多个客户端节点的流量负载进行同步与波峰调整,能通过延迟变换与流内乱序变换对实际流量负载造成预期的影响。该工具有助于测试人员快速构建具备精准流量特征控制能力的负载测试环境。

参考文献

[1] NAMIOT D, SNEPS-SNEPPE M. On micro-services architecture[J]. International Journal of Open Information Technologies, 2014,2(9):24-27.

[2] KANEV S, HAZELWOOD K, WEI G Y, et al. Tradeoffs between power management and tail latency in warehouse-scale applications[C]//Proceedings of the IEEE International Symposium on Workload Characterization. Raleigh, USA:IEEE, 2014:31-40.

[3] MICHIGAN U O, SRIRAMAN A, WENISCH T F. μTune: auto-tuned threading for OLDI microservices[C]//Proceedings of the Operating Systems Design and Implementation. Berkeley, USA:USENIX Association,2018:177-194.

[4] SRIRAMAN A, WENISCH T F. μSuite: a benchmark suite for microservices[C]//Proceedings of the IEEE International Symposium on Workload Characterization. Raleigh, USA: IEEE,2018:1-12.

[5] ALIZADEH M, KABBANI A, EDSALL T, et al. Less is

more: trading a little bandwidth for ultra-low latency in the data center[C]// Usenix Conference on Networked Systems Design and Implementation. San Jose, USA: USENIX Association, 2012:19.

[6] SHAN D, REN F. Improving ECN marking scheme with micro-burst traffic in data center networks[C]// IEEE INFOCOM 2017 IEEE Conference on Computer Communications. Atlanta, USA: IEEE, 2017:1-9.

[7] Keysight Technologies. Network-test-hardware[EB/OL]. (2022-08-12)[2023-09-25]. https://www.keysight.com/us/en/products/network-test/network-test-hardware.html.

[8] EMMERICH P, SEBASTIAN G, ANTICHI G, et al. Mind the gap a comparison of software packet generators[C]//2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems. Beijing, China: IEEE, 2017:191-203.

[9] Intel. Pktgen-DPDK[EB/OL]. (2014-12-23)[2022-02-15]. https://github.com/pktgen/Pktgen-DPDK.

[10] Cisco. tRex[EB/OL]. (2017-02-21)[2023-05-28]. https://trex-tgn.cisco.com/trex/doc/trex_manual.html.

[11] iPerf. iperf3[EB/OL]. (2003-03-13)[2016-06-08]. https://iperf.fr/.

[12] Github. Netperf[EB/OL]. (2005-10-18)[2021-07-22]. https://github.com/HewlettPackard/netperf.

[13] WU W Q, FENG X, ZHANG W L, et al. MCC: a predictable and scalable massive client load generator[C]// Proceedings of the International Symposium on Benchmarking, Measuring and Optimizing. Denver, USA: Springer, 2019:319-331.

[14] SAEED A, DUKKIPATI N, VALANCIUS V, et al. Carousel: scalable traffic shaping at end hosts[C]// Proceedings of the Conference of the ACM Special Interest Group on Data Communication. Los Angeles, USA: ACM, 2017:404-417.

[15] 华为. 流量整形与流量控速介绍说明[EB/OL]. [2023-10-15]. https://support.huawei.com/enterprise/zh/doc/EDOC1100096333/d7505117.

[16] KUMAR P, DUKKIPATI N, LEWIS N, et al. PicNIC: predictable virtualized NIC[C]// Proceedings of the ACM Special Interest Group on Data Communication. Beijing, China: ACM, 2019:351-366.

[17] NEUGEBAUER R, ANTICHI G, ZAZO J F, et al. Understanding PCIe performance for end host networking[C]// Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. Budapest, Hungary: ACM, 2018:327-341.

[18] ADELEKE O A, BASTIN N, GURKAN D. Network traffic generation: a survey and methodology[J]. ACM Computing Surveys (CSUR), 2022, 55(2):1-23.

[19] Keysight Technologies. IXIA[EB/OL]. (2022-08-12)[2023-09-25]. https://support.ixiacom.com.

[20] Sprient. TestCenter hardware[EB/OL]. (2017-07-17)[2023-09-25]. https://www.spirent.com/products/testcenter-hardware.

[21] ZILBERMAN N, AUDZEVICH Y, KALOGERIDOU G, et al. NetFPGA: rapid prototyping of networking devices in open source[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4):363-364.

[22] JEONG E, WOO S, JAMSHED M A, et al. mTCP: a highly scalable user-level TCP stack for multicore systems[C]// Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation. Berkeley, USA: USENIX Association, 2014:489-502.

[23] ZUJIE R, JIAN W, WEISONG S, et al. Workload analysis, implications, and optimization on a production hadoop cluster: a case study on taobao[J]. IEEE Transactions on Services Computing, 2013, 7(2):307-321.

[24] CHANGCHANG W, AGARWAL S, CURLESS B, et al. Multicore bundle adjustment[C]// Proceedings of the Computer Vision and Pattern Recognition. Piscataway, USA: IEEE, 2011:3057-3064.

[25] HAOKUN W. Lightshaper[EB/OL]. (2022-10-01)[2022-12-15]. https://gitee.com/acs-dcn/lightshaper.

[26] Mahousu. NistNet[EB/OL]. (2004-09-15)[2016-01-10]. https://sourceforge.net/projects/nistnet.

[27] SMIRNOV N V. Approximate laws of distribution of random variables from empirical data[J]. Russian Mathematical Survey, 1944, 10:179-206.

[28] WENLI Z, KE L, YIFAN S, et al. Labeled network stack: a high-concurrency and low-tail latency cloud server framework for massive IoT devices[J]. Journal of Computer Science and Technology, 2020, 35(1):179-193.

LightShaper：a high-precision and generic traffic shaping tool

WANG Haokun, ZHANG Wenli, ZHANG Zhao, CHEN Mingyu
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

Abstract

Simulated traffic is frequently adopted as a standard methodology for evaluating and enhancing the performance and quality of services within systems. However, commonly used tools are often characterized by high costs associated with hardware emulation, a lack of adaptability, and suboptimal precision in replicating traffic characteristics through software simulation. The challenge lies in the accurate replication of load traffic features at an affordable cost. Therefore, this paper introduces a novel approach that combines high precision with economic efficiency for the control of traffic features in test loads. Leveraging the independent traffic feature control middleware, LightShaper, to buffer, aggregate, and synchronize traffic loads, the method also incorporates a software technique based on placeholder packet filling, enabling the precise control of packet intervals on standard network cards. LightShaper is capable of dynamically shaping the network traffic waveform and packet interval distribution waveform of traffic loads, offering a suite of functionalities including traffic shaping, synchronization across multiple sources, delay transformation, and intra-flow disorder transformation, thereby attaining a level of packet interval control accuracy comparable to that of hardware. In the context of random packet interval distribution testing, LightShaper has demonstrated significant improvements over traditional timer-based software methods, with enhancements ranging from 6.67 to 28.10 times in mean square error and from 2.01 to 18.88 times in KL divergence, thus enabling the cost-effective and agile reproduction of evaluation loads.

Key words: traffic shaping, rate control, test load, inter-departure distribution