



## 基于 LoadRunner 的定制化业务背景流量生成系统

赵 瑜, 吴承荣, 严 明

(复旦大学 计算机科学技术学院, 上海 200433)

**摘 要:** 主流背景流量生成方法有导入互联网实际流量法以及利用网络测试仪器生成法等, 但是这些方法均较少关注流量的时空分布与业务流量细节分布, 无法实现流量的深度模拟。为此, 提出一个针对 Web 业务系统的基于 LoadRunner 的业务背景流量生成系统。该系统采用可扩展脚本机制、基于容器的分布式流量生成器阵列部署方式、地址修改机制、ON/OFF 模型流量时间分布控制、检查与反馈机制等方法, 实现了大规模逼真背景流量的生成, 同时, 还实现了内容异构、按需扩展、时空概率分布控制与自适应调节等特性。对以论坛网站为被测业务系统的实例进行验证, 结果表明, 实例测试生成的流量基本满足背景流量的时空特性, 并在其他特征上符合预定设置, 验证了系统设计的可行性。

**关键词:** 背景流量; 流量模型; ON/OFF 模型; 容器技术; 反馈调节

开放科学(资源服务)标志码(OSID):



**中文引用格式:** 赵瑜, 吴承荣, 严明. 基于 LoadRunner 的定制化业务背景流量生成系统[J]. 计算机工程, 2020, 46(10): 231-239, 247.

**英文引用格式:** ZHAO Yu, WU Chengrong, YAN Ming. Customized business background traffic generation system based on LoadRunner[J]. Computer Engineering, 2020, 46(10): 231-239, 247.

## Customized Business Background Traffic Generation System Based on LoadRunner

ZHAO Yu, WU Chengrong, YAN Ming

(School of Computer Science, Fudan University, Shanghai 200433, China)

**[Abstract]** The mainstream methods for background traffic generation include the introduction of actual Internet traffic and the traffic generation using network test equipment, but these methods are less concerned with the temporal and spatial distribution of traffic and the detailed distribution of business traffic, so they fail to implement in-depth simulation of traffic. To this end, this paper proposes a business background traffic generation system based on LoadRunner for Web business systems. The system adopts methods such as scalable scripting mechanism, container-based distributed traffic generator array deployment method, address modification mechanism, spatial distribution control, inspection and feedback mechanism of ON/OFF model traffic, so as to implement the generation of large-scale realistic background traffic. At the same time, the proposed system realizes content heterogeneity, on-demand expansion, temporal and spatial probability distribution control and adaptive adjustment and other features. Verification tests are carried out by taking a forum website as a to-be-tested example business system, and the results show that the traffic generated by the example tests basically meets the temporal and spatial characteristics of the background traffic, and accords with the pre-settings in other characteristics, which demonstrates the feasibility of the system design.

**[Key words]** background traffic; traffic model; ON/OFF model; container technology; feedback adjustment

**DOI:** 10.19678/j.issn.1000-3428.0055763

### 0 概述

拟态防御技术<sup>[1]</sup>等新型网络安全防御技术的实际效果评估是一个热门研究课题。目前,常用的评估方法是建立“试验场”模拟被测业务系统的日常负载,

构造可能的攻击手段,并对防御系统和业务系统的抵御攻击能力进行测试。“试验场”中网络流量的主要组成为前景流量与背景流量。在测试过程中精心构造的,与测试某项功能(如特定安全防护功能)直接相关的流量为前景流量,而与前景流量无关,且作为实

**基金项目:** 国家重点研发计划“试验场”项目(2017YFB0803203)。

**作者简介:** 赵 瑜(1997—),女,硕士研究生,主研方向为网络空间安全、网络流量仿真;吴承荣,副教授;严 明,讲师。

**收稿日期:** 2019-08-16 **修回日期:** 2019-10-18 **E-mail:** 19210240056@fudan.edu.cn

验环境存在的正常业务流量即为背景流量。对一个系统进行评估时,背景流量的作用是将其置于典型的日常运行状态,使得特定的测量数据能够真实反映其在日常情况下的表现。如拟态防御、移动目标防御等<sup>[2]</sup>新型网络安全防御理念在系统中引入了非确定性因素,防御成功与否不再是确定性事件。目标系统在典型日常运行状态下的系统性能表现与防御成功概率是反映系统防御能力的重要指标。然而,非定制化、固定内容的简单背景流量不具备逼真性,不能起到将被测系统置于日常运行典型状态的效果,无法真实评估系统防御能力。因此,背景流量的逼真度对评估结果的有效性意义重大。

背景流量生成的方法包括导入互联网实际流量法,以及使用仪器设备等测试仪器生成精确背景流量法。但是前者只能测试防火墙、路由器类二层设备,无法测试交互应用,后者自主性不足,受限于测试仪器提供的协议支持和自定义功能,通常不能支持符合特定字段的受限随机定义以及符合特定概率分布要求的定制化流量,无法提供业务操作的深度模拟。以上2种方法通常只适用于系统稳定性测试或压力测试,均不具备足够的自主性和可控性,且不能实现按需生成流量,尤其是当流量模拟的需求较复杂细致或模拟精度要求较高时,它们通常无法满足要求。理想的背景流量生成系统具备的特性有:系统应当基于少量样本规模化生成被测系统可接收的逼真背景流量,系统生成的背景流量需要符合被测系统正常运行时所遇背景流量的时空概率分布,背景流量发送系统能够对被测系统的流量接收状况进行实时监控并可自适应调节发送过程。

上述方法与现有的工具或产品均无法全面支持以上需求,因此,本文将围绕如何为被测 Web 业务系统生成复杂定制化仿真背景流量展开研究,在现有研究的基础上,提出一个针对 Web 业务系统的基于 LoadRunner(LR)的定制化业务背景流量生成系统,实现流量的实时监控以及自适应调节,完成对业务背景流量的深度模拟。

## 1 国内外研究现状

逼真的背景流量生成并非仅依靠反复发送固定的流量,或者循环批量发送实际截获的流量样本就能实现,这些无效的背景流量在交互式被测系统上将被丢弃,因此,应在识别系统流量模型的基础上,为流量模型引入随机化模拟数据,从而生成逼真流量。流量建模是对流量样本进行抽象化,参考流量识别的实验结果,构造一个以生成背景流量为基础的理论模型。从流量生成角度而言,可将背景流量模型粗略分为时间分布模型、空间分布模型和时空

分布模型。时间分布模型是主流的网络流量模型。传统的流量模型一般是基于泊松过程,如较为经典的泊松(Poisson)模型<sup>[3]</sup>、马尔科夫(Markov)模型和回归(Regression)模型等,这些模型的共同特点为短相关性<sup>[4]</sup>。文献[5]提出了网络流量具有自相似的特性,此后,研究人员相继提出了很多基于自相似特性的流量模型。常见的自相似网络流量模型有重尾分布的 ON/OFF 模型、M/G/ $\infty$  排队模型、FBM/FGN 模型<sup>[6]</sup>与 FARIMA 模型等。空间分布模型可在链路层上考虑流量的空间分布,着眼于宏观上不同链路上的流量密度分布,如文献[7]中的重力模型是基于矩阵来体现和实现流量空间分布,空间特性还体现在网络热点的存在、集群的负载均衡策略以及业务层或用户层的数据流空间分布等方面。时空分布模型是针对流量在时间和空间2个维度上的分布,适用于更精准的模拟需求,但模型耦合度高,且实现过程较为复杂。

在以上理论研究的基础上,流量生成技术也同步推进,目前,已实现了许多开源或闭源的流量生成工具及软件。流量生成技术中最简单的分类是按照流量层次区分,主要分为数据包层次流量生成器、流层次流量生成器和应用层次流量生成器<sup>[8-9]</sup>等。现有的常见流量生成器工具<sup>[10]</sup>主要包括数据包级流量生成器 Scapy 和 Cat Karat Packet Builder,基于 libpcap 的以太网数据包生成器 Bit-Twist, Linux 内核级数据包生成器 KUTE,应用程序和数据包级网络生成器 D-ITG,还有用户友好的 GUI 和详细的数据包头定制功能的数据包级流量生成器 Ostinato,根据各种流量特性产生合成流量的 HAR-POON,以及根据真实轨迹的特征生成流量的高级流量生成器 SWING 等。虽然现已实现较多优秀的开源或商业流量生成工具,但它们的设计初衷多数是针对压力测试、极限测试等需求,日常背景流量的逼真模拟并不是其主要的追求目标。因此,它们较少关注流量的时空分布和一些涉及业务的流量特征分布,无法很好地完成流量的按需生成和对业务背景流量的深度模拟。基于此,为满足“试验场”对 Web 类型业务背景流量的仿真需求,本文设计并实现一个基于 Web 的业务背景流量生成系统,可生成定制化的深度仿真业务背景流量。

## 2 系统设计

### 2.1 总体设计思路

在“试验场”中,由于业务系统在短期内业务模式变动有限,且具有一定的可重复性和可提取性,因此可采用负载生成工具作为流量模拟系统的生成源。网络压测(或负载测试)工具在市面上种类繁多

多,著名的诸如 LoadRunner、Apache JMeter<sup>[11]</sup>等。Apache JMeter 是一个利用 Java 实现的开源压力测试工具,LoadRunner 是一款模拟实际用户操作,并对系统进行实时监测的负载测试软件。Apache JMeter 设计小巧但依赖 JDK 环境,脚本修改与扩展大多基于对功能模块的熟悉,而不依赖编程,且监测与结果分析功能没有 LoadRunner 健全,同时,软件生态系统也不如 LoadRunner 完整。LoadRunner 从版本 12.55 开始兼容 Apache JMeter 脚本,因此,本文系统的实现选择基于 LoadRunner。LoadRunner 作为一款性能测试工具,虽然为流量生成提供了数据包构造的核心功能,但单独使用时依然不能满足项目需求。首先,它不支持流量的时空分布等特征的实现。LoadRunner 对于流量在时序上的分布,仅考虑以性能为主的同步到达模式<sup>[12]</sup>,即使支持异步到达,粒度也过于粗糙且随意。同时,在压力测试过程中,用户思考时间往往被忽略,这显然不符合流量仿真模拟的需求。其次,它采用进程或线程模拟独立的虚拟用户,可能在一定程度上出现计算资源或操作系统资源受限的情形,但在真实场景中这种情况基本不会出现。最后,LoadRunner 的处理逻辑是基于重复脚本的形式,但是在实际网络中,即使是同一个操作的数据包构造也不尽相同,因此需要通过定制实现流量内容的异构化(如操作与请求的异构),同时也实现流量中数据(即内容)的异构,而不是简单的重复。

本文系统采用以下技术解决上述问题:

1) 使用以基于泊松过程的 ON/OFF 模型为主、思考时间等时间分布相关指标为辅的方式,实现流量在链路层、应用层和用户层的时间分布深度仿真。

2) 以一个单独模块实现流量中网络实体空间分布。

3) 为了更加真实地还原用户场景,流量生成采用基于容器的分布式流量发送阵列方式,这是因为选择分布式部署方式能够便捷地扩展大规模网络的流量生成,解决资源上限问题,且容器技术便于一致性部署与调控。

4) 开发可扩展脚本实现流量内容的深度异构,包括操作异构与请求异构等。可在所需粒度上对业务操作的各种变量进行精确赋值,模拟生成大规模复杂异构网络下的用户产生的业务背景流量。

5) 设计一个负责实施监控与自反馈调节的模块来自动更正参数偏移,使其具有自适应性,以保证结果的可测与可控,从而形成闭环结构。

## 2.2 本文系统整体架构

本文系统主要由9个功能模块组成,具体如图1所示。该系统基于 Web 应用场景中经典的 C/S 架构,主要在客户端之前与客户端之后扩展或插入功能模块,最后可在系统边界与服务器之间得到目标流量。

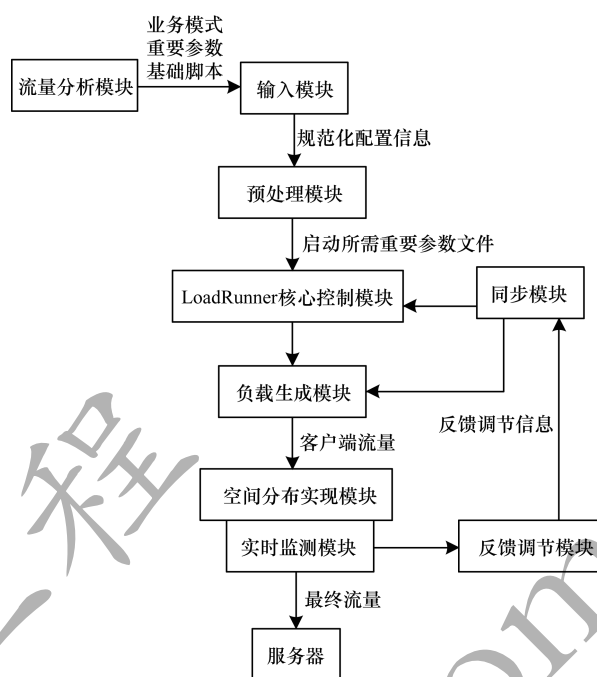


图1 本文系统整体架构

Fig.1 Overall architecture of the proposed system

本文系统整体架构中9个模块的主要功能分别为:

1) 流量分析模块。流量分析模块是系统运行的前期准备模块。针对待测业务系统进行业务操作、通信模式的识别,遵循请求-应答的基本操作对模式,整理出业务操作对双向流,选取典型的业务类型及操作并对时序组合,以此作为 LoadRunner 脚本的初期结构。

2) 输入模块。输入模块是系统一次运行的起点,也是与用户交互的窗口。输入模块向用户提供 GUI 和简单的错误反馈信息,并接收流量分析结果转化的规范化配置信息,以 YAML 配置文件作为标准输入形式。

3) 预处理模块。预处理模块接收系统首次输入的控制信息与重要参数,生成启动核心生成模块的重要参数文件和配置文件。

4) LoadRunner 核心控制模块。LoadRunner 核心控制模块与负载生成模块合作负责数据包的初步生成和发送。LoadRunner 核心控制模块主要由 LoadRunner Controller 部件扩展而来,根据配置启动 LoadRunner Controller,并同步调用负载生成器,指导负载生成器发起客户端请求并生成流量。

5) 负载生成模块。负载生成模块是实际产生流量的模块,由若干负载生成器 (Load Generator) 组成,向上承接同步模块与 LoadRunner 核心控制模块的控制命令,向下模拟真实客户端向目标服务器发起请求,并生成流量。负载生成模块应具有良好的可扩展性,其设计为分布式结构。每一个负载生成器的核心是 Load Generator 容器。

6) 空间分布实现模块。空间分布实现模块与实时监测模块均在拓扑上,处于系统的边界,且与被测服务器端邻近。空间分布实现模块实现对流量空间分布的控制,在负载生成器之后,并对数据包中代表空间分布的重要特性进行重写,如 IP 等。在空间分布实现模块之后流量将最终定型,不再有其他修改。

7) 实时监测模块。实时监测模块对负载生成器生成的、且经过空间分布实现模块转换后的流量进行实时监控。实时监测模块是系统自适应调控流量生成的依赖,且最后可根据实时监测的日志信息,得到更完整的错误提示与结果评估依据。

8) 反馈调节模块。反馈调节模块帮助系统形成一个闭环。它处理实时监测模块收集的信息,合成反馈调节指令,并通知同步模块,后者通过修改配置参数及参数文件的方式实时调整负载生成器的工作,由此保证系统流量生成的准确性。

9) 同步模块。同步模块接收反馈信息,实时修改重要参数,并利用同步模块向 LoadRunner 核心控制模块和各负载生成器发送控制信息并传输文件,协助整个系统的闭环调节机制工作。

依赖以上模块,本文系统可实现定制化流量的深度仿真功能。接下来将详细说明该系统实现的主要技术与关键方法。

## 2.3 定制化背景流量生成的关键方法

### 2.3.1 LoadRunner 核心功能的使用

LoadRunner 是原 Mercury 公司、现 HP 公司开发的一项自动负载测试工具,它能模拟实际网络用户的操作,以此产生流量。LoadRunner 主要由 Virtual User Generator (VUGen)、Controller、Load Generator、Analysis 和 Launcher 5 个部件组成。其中,VUGen 用于建立虚拟用户脚本,Controller 是实施负载测试的控制中心,Load Generator 是流量生成的真实源。LoadRunner 的工作流程<sup>[13]</sup>如图 2 所示。

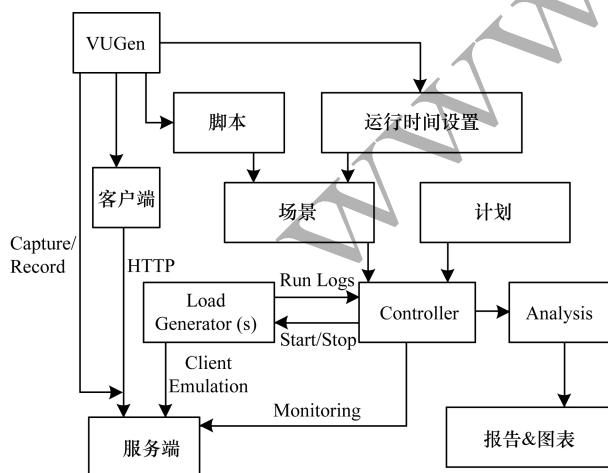


图 2 LoadRunner 工作流程  
Fig. 2 Workflow of LoadRunner

LoadRunner 具有参数化的扩展功能,但是并不提供任何关于流量分布或概率的控制。本文系统利用 LoadRunner 的核心功能,即模拟用户生成数据包功能,并着眼于扩展定制化脚本的开发与修改,在原 VUGen 脚本的基础上,扩展其他如思考时间分布化、流量异构化内容的随机生成等功能,使系统对交互的控制更佳、对测试的表现力更强,且生成的流量更细致逼真。在结合流量生成需求的场景下,再辅以其他功能模块的容器技术、分布式架构和自适应技术等,实现流量的按需发送与深度仿真。LoadRunner 支持的协议很多,鉴于本文主要针对 Web 类型流量,可选择 TCP、HTTP、HTTPS 或 WebSocket 等协议进行脚本编写。若考虑到更便捷地保留请求与附加资源的关联性,则应用层协议是更好的选择。

### 2.3.2 基于 ON/OFF 模型的流量生成

本文系统采用重尾分布的 ON/OFF 模型作为时间分布模型,负载生成模块中的各节点作为模型中的生成源。重尾分布的 ON/OFF 模型能够较好地解释流量自相似特性的形成原因<sup>[14]</sup>,因此,本文系统以其作为流量时间分布的实现参考模型。

系统中每个正在运行的 Load Generator 容器为一个生成源,它具有 ON 态和 OFF 态 2 种状态,ON 态以恒定速率生成流量,而 OFF 态保持静默。设定该源的状态为一个两态时间随机序列  $\{S(t), t \geq 0\}$ <sup>[15]</sup>,取值为 0 时处于 OFF 态,取值为 1 时处于 ON 态,且单位时间内发送 1 个数据包。若此链路分布有若干个独立的源,则  $t$  时刻链路上流量大小为  $\sum_{n=1}^{\infty} S(t)$  个数据包。若每个源独立服从某一固定分布,则若干个源叠加产生的整体流量的时间特性取决于单个源的具体分布。在 LAN 中,对于每个源,当 ON 态和 OFF 态持续时间的方差趋于无限时,则具有 Noah 效应,而链路上的若干源叠加可使流量具有长相关性。此时若文件大小符合重尾分布<sup>[16]</sup>,则在链路层可体现出流量的自相似性。

若一个随机变量  $X$  的分布函数满足式(1),则其满足重尾分布。

$$\lim_{x \rightarrow \infty} e^{\lambda x} \Pr[X > x] = \infty, \lambda > 0 \quad (1)$$

研究表明,重尾分布的 ON/OFF 模型符合电信系统与网络流量的自相似性。

本文系统采用最常见、最简单的重尾分布,即 Pareto 分布,它的概率密度函数<sup>[17]</sup>和分布函数分别为:

$$f(x) = \begin{cases} \frac{\alpha}{\beta} \left(\frac{\beta}{x}\right)^{\alpha+1}, & x > \beta, \alpha > 0 \\ 0, & x \leq \beta \end{cases} \quad (2)$$

$$F(x) = \Pr[X \leq x] = 1 - \left(\frac{\beta}{x}\right)^{\alpha}, \alpha > 0 \quad (3)$$

其中, $\alpha$  为形状参数, $\beta$  为最小截止参数。

在流量生成测试过程中,在链路上若以恒定速率传输文件,则理论上一个文件的传输时间可以趋于无穷。但源具有最小单位的持续时间,这可能是一个数据包的发送时间,且在概率上较小的持续时间是占多数的,因此,重尾分布使得 ON 态的持续时间以短时为主,但从统计角度而言,长的持续时间起着决定性作用。实际上,文件的传输时间总是有限的,不可能出现理论上 ON 态无限持续时间的情形,因此系统采用截尾重尾分布(Truncated Heavy-Tailed Distribution),即当持续时间超过某一上限时,截取其值为上限值或重新取值即可。本文系统利用预处理模块接收模型参数,生成具有重尾分布的 ON/OFF 态时间序列的参数文件,且文件指明了每个 ON 态以及 OFF 态的持续时间。在负载生成器启动阶段,将该参数文件同步给各负载生成器,负载生成器各自根据参数文件,通过执行脚本时检查当前时间与 ON 态预定时间段是否重合,以此判断是否应当发送下一个数据包。在理论上,控制粒度应为发送一个数据包的时间,但是实际上对脚本层次进行控制时,处理的最细粒度为一次请求-响应的过程,因此该方法的实现过程对精度的控制有限。

在模型实现前需计算有关参数,实现 ON/OFF 模型的一个重点是 ON/OFF 态时间周期的确定<sup>[18]</sup>。对于服从 Pareto 分布的情形,生成符合分布的周期随机数需要确定  $\alpha$  和  $\beta$  的值,且对于 ON 态和 OFF 态需各自确定其  $\beta$  的值。 $\alpha$  为重尾特性参数, $\beta_{\text{ON}}$  和  $\beta_{\text{OFF}}$  分别为 ON 态和 OFF 态可取的最小值。在实际过程中, $\alpha$  值通常取  $\alpha_{\text{ON}}$  和  $\alpha_{\text{OFF}}$  之间的值,  $1 < \alpha_{\text{ON}}$ ,  $\alpha_{\text{OFF}} < 2$ 。 $\beta_{\text{ON}}$  取决于数据生成源发送一个数据包的最短时间,而  $\beta_{\text{OFF}}$  则由各源的负载比例,即由流量与带宽之比确定。依据文献[18]中的假设与推算,可以得到  $\beta_{\text{OFF}}$  的计算方法为:

$$\beta_{\text{OFF}} = \beta_{\text{ON}} \times \frac{T_{\text{OFF}}}{T_{\text{ON}}} \times \frac{1 - m^{T_{\text{ON}}}}{1 - m^{T_{\text{OFF}}}} \times \left( \frac{1}{L} - 1 \right) \quad (4)$$

其中,  $T_{\text{ON}} = (\alpha_{\text{ON}} - 1) / \alpha_{\text{ON}}$ ,  $T_{\text{OFF}} = (\alpha_{\text{OFF}} - 1) / \alpha_{\text{OFF}}$ 。

在实现过程中选择逆变换法生成 ON/OFF 态持续时间序列。根据已知的 Pareto 分布函数  $F(x)$ , 得到其逆函数  $F^{-1}(x)$ , 再假设随机变量  $U$  符合  $(0, 1)$  上的均匀分布, 令随机变量  $X = F^{-1}(U)$ , 则根据  $U$  的随机变量值得到  $X$  的取值序列, 符合分布  $F(x)$ 。由此, 可利用 Numpy 生成大量符合 Pareto 分布的 ON/OFF 态持续时间序列, 将其保存在中间参数文件 on\_off\_XX.dat 中, 并同步给各负载生成器, 即可实现数据生成源的自行控制。

### 2.3.3 网络实体空间分布的模拟

本文系统利用预处理模块和空间分布实现模块实现网络实体空间分布的模拟。流量空间分布主要从以下 2 个方面实现:

1) 一方面是基于链路层的客户端 IP 分布。在模拟网络流量时, IP 并不能设置为测试时的少量真实 IP, 而是要进行虚拟化。虚拟 IP 不应是随机或均匀分布, 而应依据样本流量中的 IP 频率分布进行放大模拟, 以贴近实际场景。空间分布实现模块完成了数据包 IP 等字段的修改, 本质上充当个性化网桥的功能。其实, LoadRunner 本身也提供 IP 欺骗功能, 但它需 DHCP 服务器分配真实局域网地址, 操作繁琐, 申请地址有限, 于性能有损耗, 且无法生成外部地址, 只适用于简单的 IP 欺骗场景。本文系统抛弃这种方式, 改用自定义模块实现, 且实现思路为: 在网关处拦截所有数据包, 而对业务相关数据包, 根据客户端 IP 和端口号进行 IP 替换, 再重放修改后的数据包。该操作需修改各负载生成机器的默认路由, 使其指向网关机器, 并需在网关机器上获得管理员权限。当数据包到来时, 提取 IP 和端口信息, 并在内存中的转换表中(类似 NAT)查找对应条目, 找到则替换 IP, 修改 TTL、checksum 等字段, 否则添加一条转换记录。同时, 结合转换表规模大小与数据吞吐量, 设置每条转换记录的有效时间, 过期则删除该记录。

2) 另一方面是用户级别的空间分布, 由预处理模块负责, 主要基于用户活跃度不同, 如不同客户端与服务器端的交互频繁程度以及传输数据大小等。预处理模块基于样本体现在上传量或 IP 等方面的用户活跃度分布, 度量并拟合模型及参数。在流量生成参数配置阶段, 依照用户活跃度进行资源配置, 表现为虚拟用户数量分布、数据配置差异等方面, 从而使负载生成器生成流量时, 完成用户级别空间分布的刻画。

### 2.3.4 网络流量中异构化内容的随机生成

利用负载生成工具实现按需生成流量是亟待解决的问题之一, 即如何向大量重复操作和请求添加实际流量特征生成异构化内容。异构化内容的随机生成在系统中主要由流量分析模块、输入模块、预处理模块与 LoadRunner 核心控制模块协作完成, 且主要由以下 3 个方面实现:

1) 利用与用户相关的特异性特征来实现, 如用户思考时间加权参数, 它表征一个用户完成不同业务操作的稳定平均思考时间与人群总平均思考时间的比值, 用户思考时间加权参数在测试中影响不同用户的思考时间计算生成。再如用户操作偏好, 它表现为用户对不同业务的操作频率差异等。将思考时间设计为用户各异的变量值, 以此仿真实际用户操作。思考时间分布可来自 2 个辅助参数的共同作用, 一个是平均操作思考时间  $T$ , 另一个是用户思考时间加权参数  $W$ 。对于每一种典型操作都有平均操作总思考时间  $T_{[\text{action}]}$ , 由于一个操作内部通常有多

个思考时间,因此定义操作内部第  $i$  个思考时间占比为  $R_{[action],i}$ 。各操作的平均时间依赖对样本中实际操作事件测量的参数进行统计和平均来获得。用户思考时间加权参数  $W (W > 0)$  表征一个用户思考时间长短在人群中的水平,假设此值在一定时间内是稳定的,  $W = 1$  表示用户思考时间与人群中平均值相当,  $W > 1$  表示思考时间大于平均值。在测试时,用户  $i$  的 action 操作的第  $j$  个思考时间的取值为  $W_i \times T_{[action]} \times R_{[action],j}$ 。

2) 利用可从用户中独立出来的普遍特征,如利用典型业务操作请求序列、业务人均思考时间、各操作占比、网站不同入口或板块分流比例、客户端不同浏览器标志等来实现。实现时,如论坛入口分布是对论坛各入口板块流量不均衡分布情况进行统计分析后,模拟其分布情况向各入口发送流量。浏览器标志随机化是在 HTTP 协议请求头中控制 User-Agent 字段,以此给负载测试脚本增加更多随机性等。同时,对一些主要操作为浏览或内容发布的被测网站,还可设计诸如按热度/最近/随机浏览、按热度/最近/随机回复等人性化操作。针对不同的 Web 协议,HTTP 与 WebSocket 都可根据协议的具体特点尽量多地扩展异构。另外,由于系统在同一台物理机上模拟多个虚拟用户,因此为了避免用户间的干扰而导致流量失真,处理 cache 时需要更加谨慎。

3) 利用内容的异构,特别是对于一些用户上传内容的网站,流量携带数据的形式和占比都较为重要。在测试时,显然不能简单地以随机数字或随机文本填充正常场景中具有实际内容的字段。针对该类网站,可通过爬虫爬取目标或相似网站的内容直接作为模拟流量的数据来源,或对该类网站的内容分布进一步分析与学习,采取其他方式生成内容数据。以论坛为例,需要填充的字段主要为发帖内容和回复内容两类。其中,回复内容应与被回复帖子有一定的相关性,发帖内容也需要符合该论坛的主题等特点,可对回帖内容进行相关化处理,即针对被回复帖子,截取部分作为回复内容。发帖内容来源可以是利用爬虫爬取正常活跃用户的发帖内容。

### 2.3.5 基于容器的分布式流量发送阵列

实现大规模异构网络的流量模拟,应当具备可扩展的流量生成系统。因此,本文系统的负载生成模块选择基于容器的分布式发送阵列来实现。系统运用容器技术是将一个负载生成器(Load Generator)打包成镜像,部署时直接拉取即可以容器形式(如 docker)<sup>[19]</sup> 运行。

系统的网络拓扑如图 3 所示。局域网内主要为控制器端、分布式负载生成器和一个作为网桥存在的转换器,在最右端有与转换器相连的服务器端。当然,服务器端也可以是一个集群。

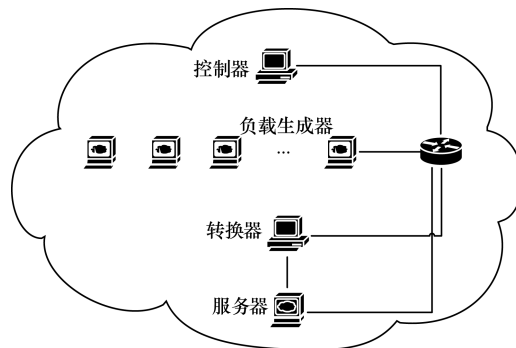


图 3 系统架构拓扑

Fig. 3 System architecture topology

分布式负载生成器集群架设在局域网内的多台物理机上,所有负载生成器在逻辑上是相同且平等的。分布式除了带来更高的性价比之外,也使系统产生了一些单个处理器不会产生的问题,如同步和一致性处理问题。根据 LoadRunner 设置,参数文件在 Controller 启动之初同步到各负载生成器,也就是说每个负载生成器容器拥有一个参数文件副本。而用户群在分布式客户端上的分布是根据用户活跃度权重随机安排的,由于虚拟用户扮演的真实用户的唯一性,用户参数文件的多个副本不能支持同时处理同一条记录。当用户数量有限以及用户选择冲突可能性较大时,需要给整个系统维护一个登陆用户列表,以保证互斥的用户选择。

### 2.3.6 系统的检查与反馈机制

本文的定制化业务背景流量生成系统设计包括检查与反馈机制,负责处理运行时数据,具有记录服务器反馈、实时流量监控以及自适应调节等功能。检查与反馈机制主要依赖实时监测模块、反馈调节模块和同步模块共同协作。检查系统的主要职能是监测和结果评估,监测的结果一部分作为中间生成信息,交由反馈系统处理,一部分作为结果评估原始数据,以日志形式记录下来,供测试完成后查看分析。检查系统的实现主要考虑以下 2 个方面:

1) 检查系统作为生成流量最后的出口总监控,可以利用类似 Wireshark、tcpdump 等包捕获工具,或其自身编程完成截获。最基本的代码结构也应该至少实现 2 个进程/线程,一个专注于截获,另一个处理数据。此处监测的原始数据为单个数据包,需整理提取会话流信息,这可能会产生一定的时延,若作为反馈调节信号,则需考虑过大的时延带来的影响。

2) 在 LoadRunner 核心控制模块中,编写脚本时,对关键请求/事务/操作的 response 等信息进行检查,主要分为 2 种方式,第 1 种方式可利用 LoadRunner 内置的检查函数(如 Web\_reg\_find() 函数)对报文进行包含性检查,并返回布尔值,可提示返回信息错误或其他错误情况。第 2 种方式则是对第 1 种方式的补



充,对于其他一些内置检查函数不能覆盖的特征检查,需要自身实现检查逻辑,如关键操作的超时检查、思考时间的偏差计算等。检查系统在此处的实现是从流量生成的源头进行检查,更适合一些基于逻辑结构完整性的检查,如事务、操作的检查等。

反馈系统向流量生成系统提供自适应调控功能,提高了其准确性、稳定性和表现力。反馈系统也基于以下方面实现:

前 2 个方面同检查系统类似,在得到总出口的截获结果后,对于反馈系统应当敏感的指标,在对数据进行处理后,将反馈指令传递给同步模块,对负载生成集群实现实时调控。在 LoadRunner 核心控制模块中,自检函数提供对下一步起决定作用的检查信息,然后依照确定流程分支执行不同代码路径,如检查关键操作是否成功,若没有成功是否重做,或检查关键资源加载是否完成,若失败如何处理等。

前文提到,反馈系统从检查系统处获取数据到最后反馈指令得到正确执行,可能带来附加的时延。反馈系统在处理数据时应尽可能简洁,且模块或设备之间的通信也应尽量迅速。首先,认真筛选重要参数指标,筛除一些反馈无意义或无法进行实时监控的指标(如 ON/OFF 时间分布模型的实现等),通常可选择思考时间粗略分布、用户活跃度分布和内容分布等作为反馈关注指标。反馈系统网络通信部分的实现可基于 Socket 或 RPC 或其他方式。Socket 的主要优点是开发简单,不同平台、不同语言甚至不同库对于 Socket 的支持都十分完备。但若是系统复杂且规模较大,则需要更高级别的安全和结构分类功能,特别是分布式系统需要更高效的通信,则 RPC 是一个更好的选择,流行的 RPC 框架有:由 Java 实现阿里巴巴开发的流行 RPC 框架 Dubbo,对多种语言支持更好的由 Google 开发的 RPC 框架 gRPC<sup>[20]</sup> 等。

### 3 系统实现与验证

#### 3.1 开发、部署及运行环境

基于上文中设计的定制化业务背景流量生成系统,本文实现了一个以论坛网站为被测业务系统的具体实例。系统至少由局域网内的 4 台机器实现:1)安装 LoadRunner 12.60 版本的完整套件和输入模块、预处理模块等;2)为承载空间分布实现模块和实时监测模块等的网关物理机;3)为装有多台虚拟机的预置负载生成器,并装有相应版本的 Docker;4)搭建有目标业务系统,即论坛网站,用来充当服务端。论坛网站利用 XAMPP 3.2.2 集成 Apache、MySQL 和 FileZilla,更便捷地建立了站点后端相关配置软件,并采用基于 PHP 的 Discuz! 开源代码实现论坛前端与主要逻辑部分,论坛网站的主要网络协议为 HTTP。开发和运行环境为 Windows Server 2016 和

Ubuntu 16.04 LTS 等。开发语言主要使用 ANSI C 和 Python 3.5.0。开发过程中使用的工具主要包括 LoadRunner 各部件、Docker、Wireshark 与 tcpdump 等,使用的库主要包括 Python 前端库 PyQt5,数据计算与分析库 Numpy、Scipy,相关网络编程的 Scapy、Winsock 和 Linux 下核心函数等。

#### 3.2 样本业务分析与特征参数计算

系统使用 Wireshark 截获样本流量,获得样本流量后,选择与业务相关的数据字段,采用统计或机器学习等方式进行通信模式的识别,并提取具有代表性的业务模版。业务模版后续将逻辑化为 LoadRunner 脚本,并控制流量的生成。在分析流量样本时,首先对样本进行过滤和分离。过滤是除去原始数据中的部分噪音,过滤的依据通常是服务器 IP 及端口。然后需要对流量进行分离,分离的标准是一次用户登录的完整事件,这也是基本分析单位,包括用户的该次登录以及登录活跃期间的操作。最后,可能以登出操作、cookie 过期或覆盖登陆结尾。流量分离后,得到一次用户登录的完整事件的流量集合。

当对一次登陆进行业务识别时,在一般场景下,使用聚类可分类各流量隐藏的典型业务操作。但对于该论坛网站和该样本来说,业务种类易于区分且种类简单,基于文本匹配和统计即可完成。该论坛的业务操作可基本分为 6 种,即登陆(log in)、登出(log out)、浏览(scan)、发帖(post)、回帖(reply)和删除(delete)。

由此则可进一步对样本流量的数据包按照操作进行划分,得出各操作的占比结果如表 1 所示。从样本流量中提取高频时序操作序列,则可得到业务系统的业务模版,高频时序操作序列如表 2 所示。

表 1 各操作的占比结果

Table 1 Percentage result of each operation %

操作	比例
scan	55.2
post	12.1
reply	26.4
delete	6.3

表 2 高频时序操作序列

Table 2 High-frequency sequential operation sequence

业务操作	操作时序	比例/%	业务操作	操作时序	比例/%
1	scan, post, reply, delete	5.2	5	post, scan, reply	5.7
2	scan, reply, post, delete	3.1	6	scan, reply	7.8
3	scan, delete, reply, post	2.2	7	scan, reply	21.4
4	scan, reply, delete	8.2	8	scan	26.0

分析样本流量的空间分布特性,空间分布特性主要基于 IP 随机化和用户活跃度分布。后者分析时主要针对不同 IP(表现为不同用户)对网站流量的贡献分布进行研究。由于原样本容量过小,因此通过爬虫爬取相似论坛发帖和回复情况,对其中的用户活跃度进行调查,取用户发布内容的数量作为流量贡献的重要依据,并以文本字数作为统计对象,最终得到用户对网站的贡献分布,如图 4 所示。用户活跃度近似服从偏态分布,考虑到样本量的局限性,最终修正为均值为 1.0、方差为 12.5 的正态分布。

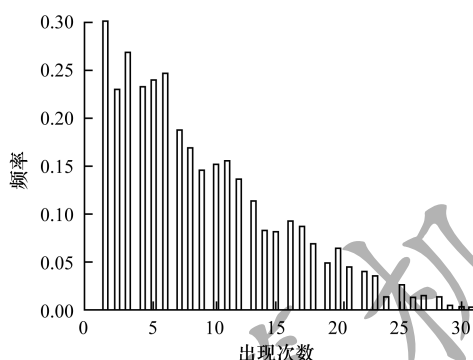


图 4 用户活跃度

Fig. 4 User activity

为实现流量异构化内容的随机生成,流量仿真还增加了以下维度的模拟。首先,思考时间的分布,也是基于用户角度的时间分布。通过统计得到各操作平均思考时间及内部思考时间分布,具体如表 3 所示。根据目标论坛各入口热度统计得到不同操作的论坛入口分布,具体如表 4 所示。另外爬取相似论坛发帖内容作为测试时发帖数据。

表 3 各操作平均思考时间及内部分配比值

Table 3 Average thinking time and internal allocation ratio of each operation

操作	平均思考时间/s	时间分配比值
scan	42	4:4:2
post	79	2:8
reply	24	6:2:2
delete	32	7:3

表 4 论坛入口分布

Table 4 Forum entrance distribution

板块号	比例	%
2	49.0	
36	17.8	
37	14.8	
38	18.4	

### 3.3 主要功能模块的开发实现

流量生成系统功能代码在运行与部署安排上,主要分为 3 个部分进行开发。一是与 LoadRunner Controller 协作的部分,包括前端输入、预处理功能代码、同步功能代码、可扩展脚本代码与为脚本开发的自定义库等。二是跟随 Load Generator 打包为镜像的同步代码和反馈调节代码,后者主要根据反馈指令修改重要参数和参数文件。三是空间分布实现模块的截获、修改与转发代码,以及对数据包的实时监测统计代码。

预处理阶段系统接受用户输入,并对参数和参数文件进行预处理,主要包括参数检查、参数生成、重要配置文件生成及重要数据生成。初始化控制通路的连接发送重要指令和参数文件,直至所有设备准备完全,再开启监测进程,最后触发 LoadRunner Controller 准备发送流量。

可扩展脚本的开发为脚本和操作的分布控制、流量内容的异构等功能提供支持。可扩展脚本的实现方式主要有以下 3 种:与预处理中指定分布参数生成相结合的参数化机制;编写并载入自定义 C 库,实现更多自定义功能函数;其他异构代码,包括不同操作逻辑的重复类型与重复次数的参数化设计,思考时间的参数化实现等。

检查与反馈机制主要依赖 2 个线程。一个是空间分布实现模块的截获、修改与转发线程,它会截获数据包并定量转发给另一个分析线程,触发实时分析。分析线程对数据包集合中的感兴趣指标进行统计分析后,会将所有计算结果作为日志保存在文件中。若数据达到一定量且偏移达到一定阈值,则发送反馈指令给同步模块上监听的线程,等待其对生成参数和参数文件进行调整。

### 3.4 运行测试与结果评估

系统部署完成之后,测试时先启动各机器,在负载生成器上运行各容器,接着启动用户输入端。输入 YAML 配置文件形式的用户需求和相关参数,配置文件说明用户思考时间加权参数服从均值为 1、方差为 0.5 的截断正态分布,空间分布用户活跃度指标符合均值为 1、方差为 12.5 的截断正态分布,并输入各操作的平均思考时间、操作内部思考时间分布、脚本和操作比例分布、入口分布以及其他参数文件路径等。场景文件 Scenario. lrs 规划了系统虚拟用户的分配与负载生成器分配等内容。测试时,选择 20 个虚拟用户并按照脚本比例进行分配,计划每 15 s 初始化一个新的虚拟用户,当全部虚拟用户初始化完成,则持续执行脚本 20 min,最后每隔 10 s 终止一个虚拟用户。提交输入后系统进入流量生成状



态,自动生成流量并实时自行调整,直至运行规定时间截止。最后,流量生成的结果以实时图表的形式呈现,还有分布式日志可供查询,测试结果如图5、图6所示。其中,流量时序图表示单位时间转发数据包个数,由此可得测试结果符合网络流量自相似性。空间分布图符合正态分布,根据思考时间运行日志得出其按既定参数执行,脚本中操作重复次数为213、50、109与20,符合样本提取的比例。从论坛网站流量/发帖量/回复量来看,结果符合Url入口分布规律,其他性能指标也基本与预期相符。

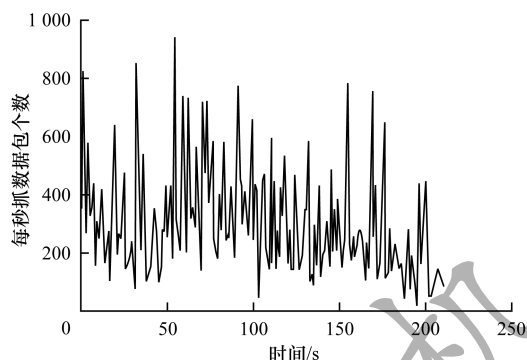


图5 流量时序图

Fig.5 Flow timing chart

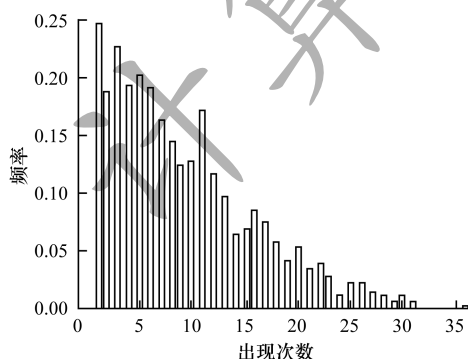


图6 空间分布图

Fig.6 Spatial distribution chart

#### 4 结束语

本文提出一个基于Web的定制化业务背景流量生成系统,并实现一个以论坛网站为被测系统的系统实例。该系统的设计基于LoadRunner核心功能的使用,利用基于容器的分布式流量发送阵列以及流量内容的异构等技术与方法实现了流量的深度仿真。实验结果表明,系统实例具有较好的模拟效果,能正确还原实际网络流量的重要特性。但是在测试过程中也存在不足,如未添加图片参数化支持,检查反馈机制对于加密协议中的内容不能完全截获分析等。下一步将围绕多样化场景及性能提升进行研究。

#### 参考文献

- [1] WU Jiangxing. Research on cyber mimic defense [J]. Journal of Cyber Security, 2016, 1(4): 1-10. (in Chinese) 郭江兴. 网络空间拟态防御研究[J]. 信息安全学报, 2016, 1(4): 1-10.
- [2] LEI Cheng, MA Duohe, ZHANG Hongqi, et al. Network moving target defense technique based on optimal forwarding path migration [J]. Journal on Communications, 2017, 38(3): 133-143. (in Chinese) 雷程, 马多贺, 张红旗, 等. 基于最优路径跳变的网络移动目标防御技术[J]. 通信学报, 2017, 38(3): 133-143.
- [3] BONALD T. The erlang model with non-poisson call arrivals [J]. ACM SIGMETRICS Performance Evaluation Review, 2006, 34(1): 276-286.
- [4] ZHANG Bin, YANG Jiahai, WU Jianping. Survey and analysis on the Internet traffic model [J]. Journal of Software, 2011, 22(1): 115-131. (in Chinese) 张宾, 杨家海, 吴建平. Internet 流量模型分析与评述[J]. 软件学报, 2011, 22(1): 115-131.
- [5] LELAND W E, TAQQU M S, WILLINGER W, et al. On the self-similar nature of Ethernet traffic [J]. ACM SIGCOMM Computer Communication Review, 1993, 23(4): 183-193.
- [6] LEDESMA S, LIU D R. Synthesis of fractional Gaussian noise using linear approximation for generating self-similar network traffic [J]. ACM SIGCOMM Computer Communication Review, 2000, 30(2): 4-17.
- [7] ZHANG Y, ROUGHAN M, DUFFIELD N, et al. Fast accurate computation of large-scale IP traffic matrices from link loads [J]. ACM SIGMETRICS Performance Evaluation Review, 2003, 31(1): 206-217.
- [8] BOTTA A, DAINOTTI A, PESCAPÉ A. A tool for the generation of realistic network workload for emerging networking scenarios [J]. Computer Networks, 2012, 56(15): 3531-3547.
- [9] WALDMANN S, MILLER K, WOLISZ A. Traffic model for HTTP-based adaptive streaming [C]//Proceedings of 2017 IEEE Conference on Computer Communications Workshops. Washington D. C., USA: IEEE Press, 2017: 1-6.
- [10] KUANG X H, LI J, XU F. Network traffic generator based on distributed agent for large-scale network emulation environment [C]//Proceedings of International Conference on Intelligent Science and Big Data Engineering. Berlin, Germany: Springer, 2018: 68-79.
- [11] HALILI E H. Apache JMeter: a practical beginner's guide to automated testing and performance measurement for your websites [EB/OL]. [2019-07-12]. <http://www.openisbn.org/download/1847192955.pdf>.
- [12] BRADY J, GUNTHER N J. How to emulate Web traffic using standard load testing tools [EB/OL]. [2019-07-12]. <https://arxiv.org/abs/1607.05356>.
- [13] WLILSON M. Load runner architecture [EB/OL]. [2019-07-12]. <http://www.wilsonmar.com/1loadrun.htm>.

(下转第247页)

(上接第 239 页)

- [14] LIU Q Y, ZHAO X H, WILLINGER W, et al. Self-similarity in social network dynamics [J]. ACM Transactions on Modeling and Performance Evaluation of Computing Systems, 2016, 2(1): 1-26.
- [15] NA Zhenyu, LIU Yi, CUI Yang, et al. Research on aggregation and propagation of self-similar traffic in satellite network [J]. International Journal of Hybrid Information Technology, 2015, 8(1): 325-338.
- [16] CROVELLA M E, BESTAVROS A. Self-similarity in world wide Web traffic: evidence and possible causes [J]. ACM Transactions on Networking, 1997, 5(6): 835-846.
- [17] SHI Jiangtao, WANG Yonggang, DAI Xuelong, et al. On study and implementation of self-similar network traffic [J]. Journal on Communications, 2005, 26(6): 112-117. (in Chinese)  
石江涛, 王永纲, 戴雪龙, 等. 自相似网络业务流量的研究与实现 [J]. 通信学报, 2005, 26(6): 112-117.
- [18] HAO Zhenhua, JIAO Wencheng, HAO Dawei, et al. Design and implementation of network traffic generator based on ON/OFF model [J]. Science Technology and Engineering, 2008, 8(12): 3219-3223. (in Chinese)  
郝震华, 矫文成, 郝大为, 等. 基于 ON/OFF 模型的网络流量产生器的设计与实现 [J]. 科学技术与工程, 2008, 8(12): 3219-3223.
- [19] KURT L. Understanding docker [EB/OL]. [2019-07-12]. [https://www.usenix.org/system/files/login/articles/login\\_winter17\\_11\\_lidl.pdf](https://www.usenix.org/system/files/login/articles/login_winter17_11_lidl.pdf).
- [20] WANG Xingwei, ZHAO Hong, ZHU Jiakeng. GRPC: a communication cooperation mechanism in distributed systems [J]. ACM SIGOPS Operating Systems Review, 1993, 27(3): 75-86.

编辑 刘继娟