软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(7):1666−1682 [doi: 10.3724/SP.J.1001.2013.04295]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 MapReduce 的大规模在线社交网络蠕虫仿真*

和 亮[1], 冯登国[1], 王 蕊[2], 苏璞睿[1], 应凌云[1]

[1](中国科学院 软件研究所,北京 100190)

[2](信息安全国家重点实验室(中国科学院 信息工程研究所),北京 100193)

通讯作者: 和亮, E-mail: windhl@yahoo.cn

摘 要: 利用云计算中的核心技术 MapReduce,提出了一种在线社交网络(online social network,简称 OSN)蠕虫的仿真方法.为了提高仿真精度,首先提出利用节点属性可调节的 OSN 有向图来描述蠕虫传播的各个过程.其次,利用运行在云环境中的多个 Map 函数和 Reduce 函数来实现对 OSN 蠕虫传播各个过程的仿真.在真实的大规模数据集上的仿真实验结果表明,提出的仿真方法不仅具有较强的可扩展性,同时也为相关领域的研究提供了一定的帮助.

关键词: 在线社交网络蠕虫;MapReduce;仿真

中图法分类号: TP309 **文献标识码**: A

中文引用格式: 和亮,冯登国,王蕊,苏璞睿,应凌云.基于 MapReduce 的大规模在线社交网络蠕虫仿真.软件学报,2013,24(7):1666−1682. http://www.jos.org.cn/1000-9825/4295.htm

英文引用格式: He L, Feng DG, Wang R, Su PR, Ying LY. MapReduce-Based large-scale online social network worm simulation. Ruan Jian Xue Bao/Journal of Software, 2013,24(7):1666−1682 (in Chinese). http://www.jos.org.cn/1000-9825/4295.htm

MapReduce-Based Large-Scale Online Social Network Worm Simulation

HE Liang[1], FENG Deng-Guo[1], WANG Rui[2], SU Pu-Rui[1], YING Ling-Yun[1]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Information Security (Institute of Information Engineering, The Chinese Academy of Sciences), Beijing 100193, China)

Corresponding author: HE Liang, E-mail: windhl@yahoo.cn

Abstract: This paper provides an approach for simulating the propagation of online social network worms, based on MapReduce, a key component of the cloud computing. In order to improve the simulation accuracy, the approach describes the phases of the worms' propagation with OSN directed graph, in which each node owns its tunable attributes. Then, the phases are simulated by different map-functions and reduce-functions, which will finally run in the cloud environment. The experimental results on the real large network datasets show that the simulating approach is scalable and helpful in the research of online social network worms.

Key words: online social network worm; MapReduce; simulation

云计算和 Web 2.0 技术的出现,使得在线社交网络(online social network,简称 OSN)逐渐成为互联网上最为流行的应用服务之一.据最新统计[1],仅 Facebook,全世界用户一个月在线时间总和已经超过 1 000 000 年.而截止到 2011 年 6 月,中国 OSN 用户数已经达到 3.8 亿,占中国人口总数的 28.6%.OSN 的大规模流行,使其成为很多不法分子的目标,将其作为恶意代码传播的主要平台[2,3].

OSN 蠕虫利用社会工程学的方法欺骗用户点击进行传播,其不依赖于特定系统的漏洞,同时还具有较高的

* 基金项目: 国家重点基础研究发展计划(973)(2012CB315804); 国家自然科学基金(61073179); 国家科技重大专项(2011ZX03002-005-2); 国家高技术研究发展计划(863)(2011AA01A203); 北京市自然科学基金(4122086)

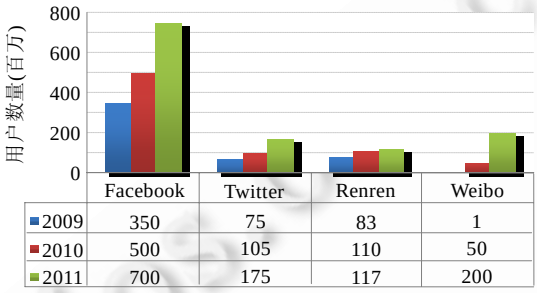
收稿时间: 2011-12-29; 定稿时间: 2012-08-14; jos 在线出版时间: 2012-12-28

CNKI 网络优先出版: 2012-12-28 13:18, http://www.cnki.net/kcms/detail/11.2560.TP.20121228.1318.001.html

© 中国科学院软件研究所 http://www.jos.org.cn

隐蔽性,是当前对 OSN 用户安全影响最大的恶意代码之一.2005 年,在 MySpace 上爆发的 Samy 蠕虫是第一个 OSN 蠕虫,在 20 小时内感染了百万用户.从 2005 年至今,每年都有新的 OSN 蠕虫爆发,例如,2008 年的 Koobface 蠕虫、2009 年的 Mikeyy 蠕虫以及 2010 年的 Clickjacking 蠕虫等.

由于 OSN 蠕虫的频繁爆发及其潜在的巨大危害(如基于浏览器的 DDoS、垃圾邮件等),安全研究人员开始针对 OSN 蠕虫传播的特性进行研究[4−7].现有研究大都将重点放在分析影响 OSN 蠕虫传播的因素方面,例如网络拓扑结构、用户点击概率以及用户行为特征等.但对这些研究进行分析可以发现,研究人员在进行实验验证时,其所采用的拓扑数据大都是人为生成的小规模数据[4−6]或者来源于小型的社交网络[7].然而,当前实际社交网络的规模已逐步扩大,如图 1 所示,国内外几个典型社交网络的用户数量均已过亿,并以较快的速度继续增长.因此,研究人员只有在接近真实的大规模环境下进行 OSN 蠕虫仿真实验,其研究结果才更加精确.而如何构建可适应大规模网络蠕虫的仿真系统,已成为国内外研究的热点和难点问题之一.



	Facebook	Twitter	Renren	Weibo
2009	350	75	83	1
2010	500	105	110	50
2011	700	175	117	200

Fig.1 Number of some top OSN sites users

图 1 几个流行 OSN 网站用户数量

目前,已有的蠕虫仿真方法主要包括数学解析模型、数据包级仿真以及网络测试床等.其中,数学解析模型作为最早出现的蠕虫仿真方法,通过使用数学上的微分方程或离散迭代式的方式来描述蠕虫的传播感染过程[8−10].这种方法可以帮助研究人员从宏观上预测蠕虫的传播过程,且不会受到规模的限制.但由于解析模型不能很好地调控每个节点的属性特征,即无法确保较高的仿真精度.数据包级仿真和网络测试床技术是目前研究网络蠕虫传播特性常用的两种方法[11−14].其中,数据包级仿真可以对单个节点的网络行为进行调控,而网络测试床技术可以同时仿真单个节点的网络行为和系统行为.由于这两种方法均需要以较大的系统开销为代价来满足对仿真精度的需求,其势必会受到仿真规模的限制.而在与 OSN 蠕虫相关的研究过程中,一方面无需对节点进行复杂的网络行为和系统行为仿真,但另一方面却需要对整个网络(节点规模在 10^7 以上)进行仿真,在这种条件下,数据包级仿真和网络测试床技术均不能很好地满足需求.

MapReduce 是云计算的核心并行计算技术,它通过将整个计算过程划分为 Map 和 Reduce 两个简单处理键值对的并行运算阶段,能够很好地处理大规模问题[15].本文针对当前蠕虫仿真研究中存在的仿真精度和仿真规模等问题,提出了一种基于 MapReduce 的大规模 OSN 蠕虫仿真方法.该方法通过分析实际 OSN 蠕虫的传播过程,将其划分为不同阶段,使用基于节点属性的 OSN 图描述方法生成 OSN 蠕虫传播的图描述;基于该描述,利用 MapReduce 技术构造相应的 Map 和 Reduce 算法分别模拟 OSN 蠕虫传播的不同阶段,并通过迭代的方式来完成对整个传播过程的仿真.该方法通过对属性信息和关键过程的描述,较好地保证了仿真精度.同时,该方法基于 MapReduce 的分布式并行计算模式,可实现大规模 OSN 蠕虫的仿真.

本文主要贡献总结如下:

(1) 提出了一种基于节点属性的 OSN 图描述方法.该方法将影响 OSN 蠕虫传播的用户状态和行为等因素抽象为属性信息,通过带有节点属性(感染状态、登陆概率以及消息响应概率等)信息的 OSN 有向图描述蠕虫传播,在描述其传播关键过程的同时描述了影响其传播的各种因素,从而提高了对 OSN 蠕虫传播的描述能力;

© 中国科学院软件研究所 http://www.jos.org.cn

(2) 提出了一种基于 MapReduce 的大规模 OSN 蠕虫仿真方法.该方法基于 OSN 蠕虫传播描述,利用 MapReduce 技术构造相应的 Map 和 Reduce 算法,通过将 OSN 蠕虫传播过程分为不同的阶段进行仿真,并通过迭代的方式仿真整个传播过程.在保证仿真精度的同时提升了大规模仿真的能力,改善了当前 OSN 蠕虫研究实验局限于小规模仿真的问题;

(3) 设计实现了原型系统并完成了相关实验.本文完成了仿真原型系统的开发,实现了 OSN 蠕虫传播的描述及仿真功能,并在真实的 Twitter 网络数据集上进行了仿真实验.实验结果表明,本文提出的方法对大规模 OSN 蠕虫具有较好的仿真能力和扩展能力.

本文第 1 节介绍相关工作.第 2 节在给出 OSN 图定义的基础上,介绍蠕虫传播过程的描述方法.第 3 节详细叙述基于 MapReduce 的仿真方法.第 4 节介绍仿真系统整体设计,并给出相应的实验结果和分析.最后在第 5 节总结全文,并提出未来的工作方向.

1 相关工作介绍

当前,在与OSN蠕虫传播相关的研究工作中,研究人员大都将重点放在分析影响OSN蠕虫传播的各种因素上,并取得了较大的突破.其中,Faghani 等人首次研究了网络拓扑结构、初始感染数量、用户点击概率等因素对OSN蠕虫传播的影响,并手工构建了规模为 10^4 的满足 Small World 拓扑特性[16]的网络,并以此为基础,分别对两种不同的 OSN 蠕虫进行仿真实验[4].罗卫敏等人在 Faghani 工作的基础上将用户的主观因素,例如社会工程学使用程度、用户安全意识程度以及访问偏向度等,作为 OSN 蠕虫传播的影响因素,他们根据 BRITE[17]生成规模为 10^5 的满足幂率特性的 OSN,并以此为基础进行 OSN 蠕虫仿真实验[5].孙鑫等人利用博弈理论对 OSN 蠕虫感染行为进行建模,该方法主要克服了无法表示用户点击概率的随机性问题[6].最近,Yan 等人不仅考虑了网络拓扑结构、用户点击概率等因素,同时将用户的网络行为作为蠕虫传播的影响因素之一[7].Yan 等人虽然使用了真实的 OSN 网络数据,但仅包含 10^4 个节点,其规模仍然与如今流行的 OSN 存在较大的差距.

通过以上叙述可以看出,在已有关于 OSN 蠕虫的研究中,所用到的网络规模都局限在小规模环境下.但目前大部分OSN网络的实际用户数量已经过亿且仍在持续增长(如图1所示),而只有在接近真实的大规模网络环境下才能全面而准确地分析研究影响 OSN 蠕虫传播的因素.

蠕虫仿真工作主要分为 3 类:① 基于数学解析模型,主要包括连续数学模型和离散数学模型;② 基于数据包级仿真,主要包括局部仿真和全局仿真;③ 基于网络测试床技术.其中,在基于数学解析模型的相关工作中,Staniford 和 Zou 利用微分方程的形式分别给出了简单传染病模型(simple epidemic model,简称 SEM 模型)[8]和双因素模型(two-factor model,简称 TFM 模型)[9];Chen 等人利用离散时间模型和确定性近似的方法描述蠕虫传播,并给出了基于离散迭代方程的解析式活跃蠕虫传播模型(analytical active worm propagation,简称 AAWP 模型)[10].虽然利用数学解析模型仿真技术可以不受规模的限制,但在研究 OSN 蠕虫过程中需要考虑单个节点处理过程(例如用户点击概率以及用户行为等)的影响,此方法无法满足该需求.在数据包级仿真工作中,研究人员大多依赖于某一个特定的网络仿真工具,其中,Liljenstam 在 SSFNet 网络仿真工具的基础上提出了局部仿真的蠕虫传播模型[11],王跃武等人在 NS2 网络仿真器的基础上提出了选择抽象技术[12],Riley 等人在 GTNets 网络仿真工具的基础上实现了全局模拟的蠕虫仿真系统[13].在网络测试床相关的工作中,Li 等人在 DETER 测试床[18]的基础上提出了基于企业环境的蠕虫传播仿真系统[14].基于数据包级仿真和测试床技术虽然可以仿真单个节点的处理过程,但由于消耗系统资源过大,通常数据包级仿真只可以处理中等规模的网络(例如校园级网络),而测试床技术则适合处理小型的局域网(例如企业级网络).因此,我们需要提出一种新的仿真方法,使其能够在满足必要的仿真精度的条件下同时具备仿真大规模 OSN 蠕虫传播的能力.

MapReduce 是云计算的核心并行计算技术,它将海量数据处理任务划分为 Map 阶段和 Reduce 阶段[15].其中,Map 阶段主要是将原始的海量数据划分为不同的数据子集,并根据用户自定义的 Map 方法来对每一个数据子集进行映射处理;而 Reduce 阶段主要是将 Map 方法处理后的结果根据用户自定义的 Reduce 方法来进行归约处理.相对于传统的并行计算技术,MapReduce 一方面隐藏了冗繁的底层分布式计算实现的细节,另一方面给

© 中国科学院软件研究所 http://www.jos.org.cn

用户提供了简单、易用的功能接口以及与性能相关的调优参数.因此,MapReduce 成为目前进行大规模并行处理的主流方法之一.本文正是在研究 OSN 蠕虫传播特点的基础上,利用 MapReduce 技术进行大规模的仿真处理.

2 OSN 蠕虫传播过程

本文首先分析 OSN 蠕虫的传播特点,给出其与传统蠕虫的不同之处;然后,通过分析两类典型 OSN 蠕虫的传播过程,归纳出其共同的传播步骤;最后,提出并利用 OSN 图来描述蠕虫传播的各个阶段的方法.

2.1 OSN蠕虫传播特点

首先,OSN 蠕虫的传播受到 OSN 网络拓扑的影响.由于 OSN 蠕虫的传播在本质上是依赖于人际关系所构成的网络,因此,属于传统蠕虫分类中的拓扑相关蠕虫.这类蠕虫能否快速传播主要依赖于网络的拓扑结构[9,12].实际的拓扑数据表明(见第 4.2 节中的实验数据),OSN 网络具有一般社交网络(例如,E-mail,IM 等所构成的网络)的特点,即拥有较小的特征路径长度、较大的聚合系数以及节点度数满足幂率分布等.这些特点都为 OSN 蠕虫的快速传播提供了有利条件.

此外,与传统蠕虫相比,另一个显著的特点是,OSN 蠕虫的传播主要受到用户的行为影响.用户的登陆频率以及对消息的响应概率都是影响 OSN 蠕虫传播的重要因素.其中,用户登陆频率反映了某一个 OSN 的流行程度,而只有在流行的 OSN 中,例如 MySpace,Facebook 以及 Twitter 等,蠕虫才会大范围迅速传播.用户对消息的响应概率则直接决定了蠕虫能否感染.例如,对于一个安全意识较高的用户来说,他可以轻易识别出普通消息和恶意消息,这类用户的消息响应概率要小于普通用户,进而导致 OSN 蠕虫很难对这类用户进行感染.本文利用仿真实验,验证了这些用户行为因素在蠕虫传播过程中所起到的关键性作用.

除了以上两个突出的特点外,OSN 蠕虫在传播过程中还具有如下特点:第一,OSN 蠕虫可以通过用户浏览网页传播,不依赖于客户端特定的操作系统及软件漏洞,这就使得 OSN 蠕虫具有更多的潜在攻击目标;第二,由于 OSN 蠕虫传播过程所产生的流量大都是在客户端和服务端之间,并且服务端通常都可以承载较大的负荷,因此,OSN 蠕虫的大规模传播不会轻易造成网络的拥塞,使其具备了很强的隐蔽性,进而造成传统安全检测和防御手段的失效;最后,虽然 OSN 蠕虫传播速度快、隐蔽性强,但由于其主要的传播过程是在浏览器内部,需要利用脚本完成,不会影响到系统本身(本地硬盘存储和内存等),因此防御的难度要小于传统蠕虫.例如,将浏览器设置禁用脚本即可阻止大部分 OSN 蠕虫的传播.

2.2 典型OSN蠕虫传播过程

根据感染策略的不同,本文将 OSN 蠕虫分为两类:类 Samy 蠕虫和类 Koobface 蠕虫,如图 2 所示.

- 类 Samy 蠕虫在传播过程中,首先找到服务端可以利用的基于 Web 应用程序的 XSS 漏洞[19],将利用 XSS 漏洞执行传播功能的恶意脚本嵌入到自身 HTML 页面中;然后,利用服务器端的系统消息推送机制,将页面更新消息自动推送给其所有好友;最后,当好友浏览该页面时会自动被感染,从而造成蠕虫的传播.虽然 Samy 蠕虫本身没有利用消息推送机制,但如今大多数的类 Samy 蠕虫均使用这种方式来加快其传播过程;
- 类 Koobface 蠕虫在传播过程中,首先将恶意软件部署到某个已被控制的服务器中,并通过某种方式感染初始账户;然后,将包含恶意 URL 的消息发送给其所有好友;最后,当用户点击该 URL 时,会被要求下载和安装经过伪装的恶意软件,当用户安装该软件后,就会自动将同样的恶意消息发送给其所有 OSN 好友,从而造成蠕虫恶意消息的传播.

虽然两类蠕虫在感染策略上有所差别,但通过分析,上述两类 OSN 蠕虫的传播过程是相似的,可以将其归纳为如下 3 个步骤:

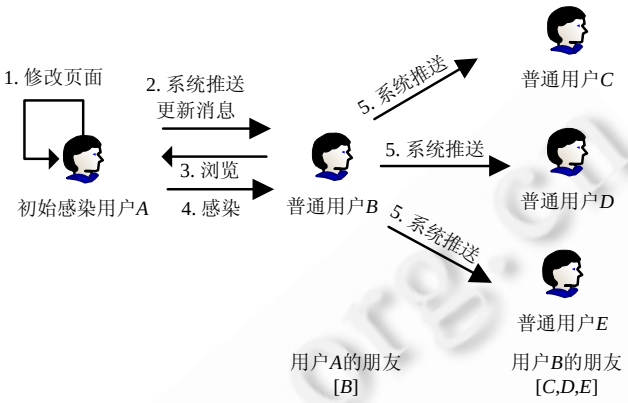
步骤 1. 感染初始节点,类 Samy 蠕虫在页面中内嵌恶意脚本,类 Koobface 蠕虫拥有某一账号即可;

步骤 2. 利用某种机制(系统推送、主动发送)将包含恶意内容的消息发送给感染节点的所有好友;

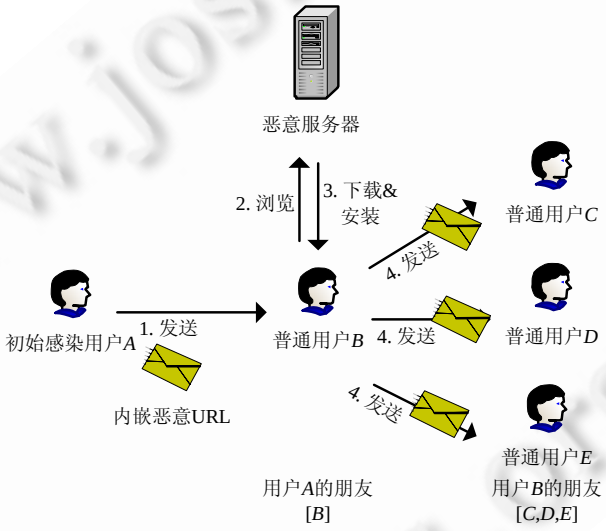
© 中国科学院软件研究所 http://www.jos.org.cn

步骤 3. 好友以某一概率响应该消息(浏览页面或者下载、安装恶意软件),即被感染.

分析以上 3 个步骤可以发现:其本质与描述传统蠕虫传播的 SEM 模型有一定的相似之处;不同之处在于:在 SEM 模型中,当已感染节点探测到含有漏洞的易感染节点后,易感染节点一定会被感染;但在 OSN 蠕虫传播中,收到恶意消息的易感染节点却不一定被感染,其是否被感染主要取决于用户行为相关的因素,例如用户登陆概率和对消息的响应概率等.



(a) 类 Samy 蠕虫传播



(b) 类 Koobface 蠕虫传播

Fig.2 Propagation of OSN worms

图 2 OSN 蠕虫传播过程

2.3 OSN蠕虫传播过程的图描述

为了能够具体地描述 OSN 蠕虫的传播,我们首先分析了 OSN 网络中用户关系的有向性问题;然后,在有向图的基础上给出了基于节点属性的 OSN 图描述方法;最后,利用 OSN 图描述了蠕虫传播的各个过程.

2.3.1 用户关系的有向性问题

在以 Facebook 为代表的 OSN 网络中,用户之间的关系是双向的[20],即如果 A 是 B 的“好友”,则 A 和 B 可以互相接收到对方的更新消息.因此,对于这类 OSN 网络,可以使用无向图 $G=(V,E)$来表示,其中,V 表示用户节点集合$\{v_i\}$,E 表示用户之间的好友关系集合$\{e_n|e_n=(v_i,v_j),v_i\neq v_j\}$.

© 中国科学院软件研究所 http://www.jos.org.cn

而在以 Twitter 为代表的 OSN 网络中,用户之间的关系是单向的[20],即如果 A 是 B 的“粉丝(fans)”,则只有 A 可以接到 B 的更新消息.因此,对于这类 OSN 网络,可以使用有向图 $G=(V,E)$来表示,其中,V 表示用户节点集合 $\{v_i\}$,而 E 表示用户节点之间的关系集合$\{e_n|e_n=\langle v_i,v_j\rangle,v_i\neq v_j\}$,并且规定 v_j 是 v_i 的“粉丝”.

本文用有向图 G 来研究 OSN 网络,对于双向关系的 OSN 网络,我们通过将无向边 $e_n=(v_i,v_j)$表示为两条有向边 $e_s=\langle v_i,v_j\rangle$和 $e_t=\langle v_j,v_i\rangle$的方式即可处理.

2.3.2 基于节点属性的 OSN 图

我们首先扩展了有向图 G 的定义,将 OSN 网络表示为

$$G_{\mathrm{OSN}}=(V,E,A),$$

其中,V 表示用户节点集合$\{v_i\}$,E 表示用户之间的关系集合$\{e_n|e_n=\langle v_i,v_j\rangle,v_i\neq v_j\}$,而 A 则表示用户的属性信息.通过分析 OSN 蠕虫传播的实际过程,本文的用户属性包括如下 4 种:

(1) 标识 ID.用来表示用户在 OSN 网络中的唯一标识,根据实际情况,这里我们将 ID 的取值范围设为正整数集,且满足$\forall i\neq j$,均有 $v_i.id\neq v_j.id$,其中,$i=1,2,\ldots,|V|$,$j=1,2,\ldots,|V|$;

(2) 登陆概率 P_t.用来表示用户在时刻 t 登陆的概率.由于在 OSN 蠕虫传播过程中,用户登陆是否频繁是影响其传播速度的关键因素之一,因此这里我们将登陆概率作为用户的属性之一,并用 $v_i.p_t$ 表示节点 v_i 的登陆概率;

(3) 消息响应概率 P_r.用来表示用户在接收到消息(包括正常消息和恶意消息)时对其进行响应的概率,也即用户登陆时 OSN 蠕虫传播过程中的感染概率,这里用 $v_i.p_r$ 表示节点 v_i 的响应概率;

(4) 感染状态 S.用来表示用户在 OSN 蠕虫传播过程中的感染状态.在传统的 SEM 模型中,用户状态分为“正常”和“已感染”,而在 OSN 蠕虫传播过程中,还存在用户登陆之前已收到恶意消息的情况(保存在服务端),因此,这里我们增加“预感染”状态.为了表述方便,我们用 $v_i.s=0$ 表示“正常”,$v_i.s=1$ 表示“预感染”,$v_i.s=2$ 表示“已感染”.

接着,我们给出了在 G_{OSN} 上定义的几类节点集合:

定义 1. 节点 v_i 的后继集合(successor set),记为 $SS(v_i)$:令 $SS(v_i)$表示所有与节点 v_i 存在有向边$\langle v_i,v_j\rangle\in E$ 的节点 v_j 集合,即 $SS(v_i)=\{v_j|\langle v_i,v_j\rangle\in E,v_j\in V\}$,称 $SS(v_i)$为节点 v_i 的后继集合.

定义 2. 节点 v_i 的前驱集合(precursor set),记为 $PS(v_i)$:令 $PS(v_i)$表示所有与节点 v_i 存在有向边$\langle v_j,v_i\rangle\in E$ 的节点 v_j 集合,即 $PS(v_i)=\{v_j|\langle v_j,v_i\rangle\in E,v_j\in V\}$,称 $PS(v_i)$为节点 v_i 的前驱集合.

定义 3. 叶子节点集合(leaf set),记为 LS:令 LS 表示所有没有后继节点的节点集合,即 $LS=\{v_i|SS(v_i)=\varnothing,v_i\in V\}$,称 LS 为叶子节点集合,称 v_i 为叶子节点.

定义 4. 初始感染集合(initial infection set),记为 IIS:若存在映射 f,$f(V)=IIS_f$,将 IIS_f 中所有 v_i 的感染状态置 $v_i.s=2$,我们称 f 为初始化策略,并将 IIS_f 称为在策略 f 下的初始感染集合,即$\forall v_i\in IIS_f$,$v_i.s=2$.特别地,如果存在映射 g 使得$|IIS_g|>0$,则称 g 为有意义的初始化策略.

定义 5. 感染集合(infection set),记为 IS:令 IS 表示某一时刻 t 集合 V 中所有已感染节点的集合,即 $IS=\{v_i|v_i.s=2,v_i\in V\}$,称 IS 为 G_{OSN} 的感染集合.

定义 6. 预感染集合(pre-infection set),记为 PIS:令 PIS 表示由感染集合 IS 中节点的后继节点组成的可能被感染的节点集合,并将其中节点的状态置为 1,即 $PIS=\{v_i|v_i.s=1,v_i\in V\}$,称 PIS 为预感染集合.并用 $PIS\text{-}C_i$ 表示对于节点 v_i,其前驱节点中被感染节点的个数,即集合$\{v_j|v_j\in PS(v_i),v_j\in IS\}$中 v_j 的个数.

最后,在研究过程中,本文同样使用了离散时间模型[10],即在任意时刻 t,如果感染用户将恶意消息传播到其好友用户,则在同一时刻,该用户好友以 P_t 的概率登陆并以 P_r 的概率响应恶意消息,即被感染.

2.3.3 基于 OSN 图的传播过程

根据第 2.2 节的归纳,我们将 OSN 蠕虫的传播过程分为 3 个阶段,分别称为初始化阶段、传播阶段和感染阶段.而整个 OSN 蠕虫的传播过程可以看做是由一次初始化阶段和多次传播、感染阶段迭代完成的.根据离散时间模型,我们假设初始化阶段在 t_0 时刻完成,而在 t_m 时刻完成第 m 次传播和感染阶段的迭代,其中,$m=1,2,\ldots,N$.

© 中国科学院软件研究所 http://www.jos.org.cn

在初始化阶段,OSN 蠕虫根据有意义的策略 g 获取初始感染集合 IIS_g.这里,从蠕虫首次感染 OSN 网络开始描述其传播过程,因此,在初始化之前,集合 IS 和集合 PIS 均为空集,即 $IS=PIS=\varnothing$.而根据策略 g,获取集合 IIS_g 后,对$\forall v_i\in IIS_g$,将 $v_i.s$ 置 2,即将其添加至集合 IS.这里,有意义的策略 g 可以是不同的规则,例如"选择度数最高"或者"随机选取"等,不同的策略会导致 OSN 蠕虫传播过程的不同(见第 4.2 节的实验结果).

在传播阶段,OSN 蠕虫从已感染节点传播到其相邻节点.在时刻 t_m,m=1,2,…,N,已感染的节点均包含在集合 IS 中,则对$\forall v_i\in IS$,获取 v_i 的后继集合 $SS(v_i)$,将满足 $v_j\in SS(v_i)$且 $v_j.s$=0 的节点 v_j 的状态置 1,即添加到集合 PIS 中.如果根据集合 IS 中的节点 v_i 获取不到任何满足 $v_j.s$=0 条件的后继节点,则表明节点 v_i 已经感染了其所有的后继节点.此外,由于多个已感染节点有可能将恶意消息传播到同一个节点 v_j,则我们用集合 PIS 定义中的 $PIS\text{-}C_j$ 来记录节点 v_j 在时刻 t_m 接收恶意消息的次数.

在感染阶段,如果用户登陆并响应了蠕虫恶意消息,则被 OSN 蠕虫感染.为了描述用户登陆和响应消息行为对蠕虫传播的影响,这里,我们对集合 PIS 中的每一个节点 v_i 按照以下公式(1)计算其感染概率 p_i.

$$p_i = v_i.p_t \times (1-(1-v_i.p_r)^{PIS\text{-}C_i}) \tag{1}$$

在公式(1)中,计算了用户以概率 $v_i.p_t$ 登陆的情况下收到 $PIS\text{-}C_i$ 条恶意消息时被感染的概率.此时,如果节点 v_i 以概率 p_i 被感染,则将 v_i 从集合 PIS 中删除,并置 $v_i.s$=2,即将其添加至集合 IS 中.如果未被感染,则说明有两种情况发生:第 1 种情况是用户以$(1-v_i.p_t)$的概率未登陆,根据真实的场景分析,用户在下一次迭代过程中仍有机会被感染,我们只需将节点 v_i 的状态置 0,并从 PIS 删除即可;第 2 种情况是用户以 $v_i.p_t$ 的概率登陆后,识别恶意消息并将其删除,这里假设用户识别恶意消息后永远不会再被感染,那么我们不仅需要将节点 v_i 从集合 PIS 删除,同时还要将其从集合 V 中删除.严格来说,用户有可能在下次登陆时再次接收到其他感染用户的恶意消息并对其进行响应,为了简化描述过程,本文暂不考虑这种情况.

当感染阶段完成后,会有新的节点被添加至 IS 中,进而开始下一次传播、感染的迭代过程.如果在任意一次迭代过程中,当传播阶段完成后,集合 PIS 为空,则表明 OSN 蠕虫已经感染了所有可能被感染的节点,此时,迭代过程结束,即整个 OSN 蠕虫传播过程结束.这里,我们给出了基于有向图的 OSN 蠕虫传播算法.

算法 1. 基于有向图的 OSN 蠕虫传播算法.

Input:$G_{OSN}=(V,E,A)$,初始化策略 g;

Output:蠕虫传播过程.

```
1.  IS←∅, PIS←∅
2.  for each v_i∈V
3.      为每一个节点的属性赋初始值
4.  IIS←selectNodesByG(V,g)                ▷初始感染集合
5.  for each v_i∈IIS
6.      v_i.s←2
7.      IS.add(v_i)
8.  loop←0
9.  while TRUE                             ▷开始迭代
10.     loop++
11.     for each v_i∈IS                    ▷传播阶段
12.         for each v_j∈SS(v_i)
13.             if v_j.s=0
14.                 v_j.s←1
15.                 PIS.add(v_j)
16.                 PIS-C_j++
17.     if PIS=∅
```

© 中国科学院软件研究所 http://www.jos.org.cn

```
18.     return
19.   infectNum←0
20.   for each v_j∈PIS                                  ▷感染阶段
21.     p_infect ← v_j.p_t × (1 − (1 − v_j.p_r)^(PIS-C_j))
22.     p_random←getRandomProbability()
23.     if p_random<p_infect
24.       v_j.s←2
25.       IS.add(v_j)
26.       infectNum++
27.     else
28.       p_random←getRandomProbability()
29.       if p_random<v_j.p_t
30.         V.delete(v_j)
31.       else
32.         v_j.s←0
33.     PIS.delete(v_j)
34.     report(loop,infectNum)
```

在该算法中,主要关注影响蠕虫传播过程的几个因素:首先,在根据策略 g 选取初始感染节点集合时,不同的策略 g 可能会形成不同的传播过程;其次,在给节点属性赋初始值时,不同的登陆概率和消息响应概率同样会形成不同的传播过程.我们在第 4 节仿真实验结果的基础上,再详细分析这些因素对 OSN 蠕虫传播的影响.

3 基于 MapReduce 的 OSN 蠕虫仿真方法

本节根据 MapReduce 技术,介绍如何利用这种技术进行高效的大规模 OSN 蠕虫仿真,包括如何利用键值对来表示 OSN 图以及如何利用 Map 和 Reduce 方法来仿真蠕虫的传播和感染过程,并对仿真迭代过程的结束条件加以讨论.

3.1 OSN图的*k-v*形式

在通过 MapReduce 处理大规模问题的过程中,主要是利用 Map 和 Reduce 方法将原始的键值对映射为新的键值对,该过程可以简单地描述为如下形式:

$$\langle okey,ovalue\rangle \xrightarrow{map} \langle mkey,mvalue\rangle \xrightarrow{reduce} \langle nkey,nvalue\rangle \tag{2}$$

利用 MapReduce 进行计算的前提就是要将待处理的数据转化为$\langle key,value\rangle$形式(以下简称 *k-v* 形式).由于本文要处理的是 OSN 图 $G_{OSN}=(V,E,A)$,下面,我们讨论如何将 G_{OSN} 转化为 *k-v* 形式:

首先,对于图来说,有两种标准的表示方法,即邻接表和邻接矩阵[21].但在真实情况下,由于 OSN 图普遍呈现出明显的稀疏性,如果使用邻接矩阵,则会导致空间上的浪费,因此这里使用邻接表来表示 OSN 图;其次,与普通图不同,在 OSN 图中还要包括每个节点的属性相关的信息,故在邻接表中,不仅要包含每个节点所有的相邻节点,同时还要包括该节点的属性内容.根据以上分析,最终,我们将 OSN 图表述为如下 *k-v* 形式:

$$\langle v_i.id,edges\text{-}v_i.p_t\text{-}v_i.p_r\text{-}v_i.s\rangle \tag{3}$$

这里,将节点的 ID 作为 *key*,而将节点相邻的节点集合、登陆概率、响应概率和感染状态作为对应的 *value*.我们将公式(3)称为 OSN 图的 *k-v* 形式.

3.2 基于MapReduce的仿真方法

本文将完成形如公式(2)的过程称为一次 MR-Job.根据 MapReduce 的设计原理,为了完成包含有迭代过程的图算法,需要多趟地外部链式调用 MR-Job[22].这里,链式调用的实现方式是将前一次 MR-Job 的输出作为下一

© 中国科学院软件研究所 http://www.jos.org.cn

次 MR-Job 的输入.通过第 2.3.3 节的描述,OSN 蠕虫的传播过程可以由 1 次初始化阶段和多次传播、感染阶段迭代完成,而根据对每个阶段具体任务的分析,其中初始化阶段可以由 1 次完整的 MR-Job 来完成,而传播、感染阶段则可以由 1 次 MR-Job 的 Map 和 Reduce 方法分别完成.如图 3 所示,我们给出了利用 MapReduce 技术分阶段迭代完成 OSN 蠕虫传播的基本流程.

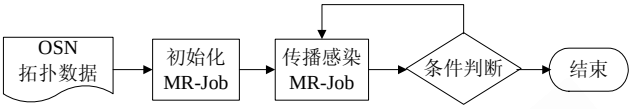


Fig.3 Propagation with MR-Job

图 3 利用 MR-Job 的传播过程

3.2.1 初始化 MR-Job

假设最原始收集到的 OSN 图由有序二元组$\langle v_i.id,v_j.id\rangle$表示,那么在初始化 MR-Job 中,就需要将原始的二元组转化为 OSN 图的 *k-v* 形式.算法 2 给出了转化的具体过程.

算法 2. 初始化仿真算法.

Input:原始二元组$(v_i.id,v_j.id)$表示的 OSN 图;

Output:OSN 图的 *k-v* 形式.

Map 阶段:

▷这里,*key* 为二元组的编号,*value* 为$\langle v_i.id,v_j.id\rangle$

1. ***map***(*key*,*value*)
2. $v_i.id,v_j.id\leftarrow$*parse*(*value*)
3. *collect*($v_i.id,v_j.id$)

Reduce 阶段:

▷这里,*key* 为 $v_i.id$,*value* 为邻接节点集合

4. ***reduce***(*key*,*value*)
5. *edges*←∅
6. **for** each *id* **in** *value* ▷*value* 为邻接点集合
7. *edges*←*edges*+*id*
8. $p_t,p_r,s\leftarrow$*assign*(*key*)
9. *collect*(*key*,*edges*-p_t-p_r-*s*)

这里,利用 Map 方法来收集原始 OSN 图中所有二元组的信息,并将 $v_i.id$ 和 $v_j.id$ 作为 Map 收集的 *key* 和 *value*(即公式(2)中的 *mkey* 和 *mvalue*).在 Reduce 方法中,系统会提供由 Map 方法收集的每一个 *key* 及其对应的所有 *value*,这里也即每一个节点对应的邻接节点.而在此基础上,再为每一个 *key* 进行属性赋值,即得到了 OSN 图的 *k-v* 形式.此外,根据算法 2,这里只为非叶子节点进行初始化赋值.而对于叶子节点,则在下一节中讨论其处理过程.

3.2.2 传播和感染 MR-Job

我们用一次 MR-Job 的 Map 方法来仿真传播阶段,用 Reduce 方法来仿真感染阶段.在传播阶段,要将集合 *IS* 中所有节点的后继集合中的正常节点放入 *PIS* 集合中,并将节点状态置 1.观察公式(3),对于任意节点 v_i,在实际计算过程中根据 $v_i.s$ 来判断,如果 $v_i.s$=2,则表明该节点被感染,那么就需要将 v_i 对应的 *edges* 中的所有节点的状态置 1,并放入集合 *PIS* 中.而为了“放入集合 *PIS*”,我们需要根据每一个状态被置 1 的节点构造新的$\langle$*key*,*value*$\rangle$,其形式如下:

$$\langle v_j.id,null\text{-}null\text{-}null\text{-}1\rangle \tag{4}$$

© 中国科学院软件研究所 http://www.jos.org.cn

在感染阶段,要判断集合 *PIS* 中所有的节点是否被感染,也就是判断所有状态为 1 的节点是否可能被感染.但在实际的 Reduce 方法中,根据 Map 方法收集的所有结果,对于任意一个节点 v_i,其对应的 *value* 集合中可能存在的值包括:

① (*edges*-v_i.p_t-v_i.p_r-0),即原始的节点值;

② (*null*-*null*-*null*-1),被放入 *PIS* 集合中的节点值;

③ (*edges*-v_i.p_t-v_i.p_r-2),被感染的节点值.

下面分别讨论可能出现的情况:

1. 存在类型③的情况.说明该节点已被感染,则在本阶段无需处理,但要保留到下次传播阶段继续处理;
2. 存在类型①和类型②的情况.说明该节点在传播阶段接收到恶意消息,那么此时就需要判断其是否被感染;
3. 只存在类型①的情况.说明该节点尚未接收到恶意消息,需要保留到下次迭代过程处理;
4. 只存在类型②的情况.此时,这类节点为叶子节点,需要为叶子节点属性赋值,并判断其是否被感染.

算法 3. 传播和感染仿真算法.

Input:*k*-*v* 形式的 OSN 图;

Output:传播感染过程.

Map 阶段:

▷这里,*key* 为 v_i.*id*,*value* 为(*edges*-v_i.p_t-v_i.p_r-*S*)

```
1.  map(key,value)
2.    s,edges←parse(value)
3.    if s=2                                  ▷传播开始
4.      newValue←null-null-null-1
5.      for each v_j.id in edges
6.        collect(v_j.id,newValue)
7.    collect(key,value)                      ▷收集所有记录
```

Reduce 阶段:

▷这里,*key* 为 v_i.*id*,*value* 为{(*edges*-v_i.p_t-v_i.p_r-v_i.*s*)}集合

```
1.  reduce(key,value)
2.    newEdges←∅;newP_t,newP_r,PIS-C_i←0
3.    for each v in value
4.      edges,p_t,p_r,s←parse(v)
5.      if s=2                                ▷情况 1
6.        collect(key,v)
7.        return
8.      if s=1
9.        PIS-C_i++                           ▷记录收到恶意消息的次数
10.     newEdges←newEdges+edges
11.     newP_t←max(newP_t,p_t)
12.     newP_r←max(newP_r,p_r)
13.   if PIS-C_i==0                           ▷情况 3
14.     collect(v_i.id,newEdges-newP_t-newP_r-0)
15.     return
16.   if newP_t=0||newP_r=0                   ▷情况 4
```

© 中国科学院软件研究所 http://www.jos.org.cn

17. $newP_t,newP_r \leftarrow assign(key)$
18. $p_{infect} \leftarrow v_i.p_t \times (1-(1-v_i.p_r)^{PS\text{-}C_i})$ ▷情况 2
19. $p_{random} \leftarrow getRandomProbablity()$
20. **if** $p_{random}<p_{infect}$ ▷被感染
21. $reportInfect(v_i.id)$
22. $collect(v_i.id,newEdges\text{-}newP_t\text{-}newP_r\text{-}2)$
23. $p_{random} \leftarrow getRandomProbablity()$
24. **if** $p_{random}<newP_t$ ▷删除登陆但未响应的节点
25. $collect(v_i.id,newEdges\text{-}newP_t\text{-}newP_r\text{-}2)$
26. **else** ▷未登陆
27. $reportPreInfect(v_i.id)$
28. $collect(v_i.id,newEdges\text{-}newP_t\text{-}newP_r\text{-}0)$

这里对算法 3 进行如下说明:首先,在 Reduce 方法的第 16 行,如果判断某节点为叶子节点,则需要为其属性赋值,因为在初始化算法 2 中并没有为叶子节点属性赋值(见第 3.2.1 节),之后,与其他“预感染”节点一样,判断其是否被感染;其次,在 Reduce 方法的第 24 行,如果判断该节点登陆但不响应恶意消息,那么我们就认为该节点永远不会被感染.为了避免下次重复计算,在实际计算过程中,我们将其状态置为 2,但不对其进行感染统计.

3.2.3 迭代结束条件

由于本文使用迭代的方式进行 OSN 蠕虫传播过程的仿真,因此,这里需要讨论迭代过程的结束条件,包括具体的判断条件和实现方式.这里,我们将“不会出现的新感染节点”和“感染了大部分节点”作为两个判断条件,满足其中任何一条即可终止迭代.

首先,讨论“不会出现的新感染节点”情况.

在算法 3 中,利用 *reportInfect* 方法和 *reportPreInfect* 方法(传播和感染 MR-Job 中 Reduce 阶段的第 21 行、第 27 行)可以获取一次迭代过程中新感染节点数量 n 和接收恶意消息但未登陆的节点数量 m.那么,根据 n 和 m 的可能取值,这里作如下讨论:

① $n\neq0$ 且 $m\neq0$,说明在本次迭代过程中出现了新感染节点和尚未登陆的节点,那么很明显需要继续迭代;

② $n=0$ 且 $m\neq0$,说明本次迭代没有新感染的节点,但是因为存在尚未登陆的节点,那么仍要在下一次迭代时来判断是否有新节点被感染;

③ $n\neq0$ 且 $m=0$,说明有新感染的节点,但没有未登陆的节点,而只要有新感染的节点,就需要继续迭代;

④ $n=0$ 且 $m=0$,说明本次迭代过程中既没有新感染节点,也没有尚未登陆的节点,那么在下次节点过程中就不可能再出现新的感染节点.

因此我们将情况④,即 $n=0$ 且 $m=0$ 作为迭代结束的判断依据.

其次,分析“感染了大部分节点”的情况.

为了研究蠕虫的传播感染过程,通常只需要感染率达到设定的阈值即可[4−14].这里,我们设感染率阈值为 T,则累计每次迭代过程中通过 *reportInfect* 方法获取到新感染的节点数量为 $n_i,i=1,2,\ldots,N$,当满足 $\left(\sum n_i/|V|\right)>T$ 时,即可判断迭代过程结束.

最后,给出具体的实现方式.

根据 MapReduce 分布式设计原理,一次 MR-Job 中的每一个 Reduce 方法(Map 方法)是无交互同时运行的,因此不存在“共享变量”的机制,即无法直接获取所有 Reduce 方法的结果总和.为了解决“共享变量”的问题,这里我们考虑使用 MapReduce 赖以运行的底层分布式文件系统来解决.即,将每一个 Reduce 方法的结果保存到新生成的分布式文件中,当 MR-Job 结束后,通过统计所有新生成文件中的内容,从而可以判断是否满足结束条件.

© 中国科学院软件研究所 http://www.jos.org.cn

4 系统设计及实验评估

4.1 系统设计

本文在 MapReduce 的开源实现——Apache Hadoop(以下简称 Hadoop)[23]的基础上设计并实现了原型系统.图 4 给出了原型系统的整体架构,从下到上依次为硬件层、Hadoop 层和仿真层.

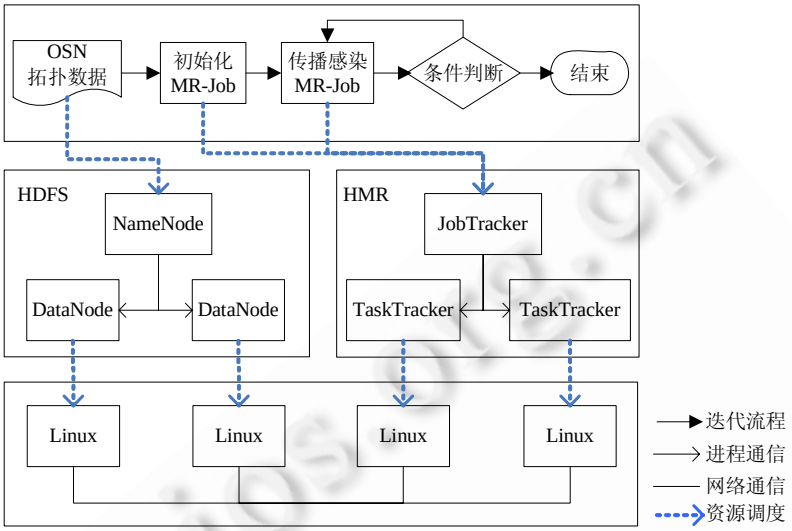


Fig.4 Structure of prototype system

图 4 原型系统结构

如图 4 所示,在硬件层主要包含运行 Linux 操作系统的主机节点.为了降低硬件部署的难度,在实验过程中,我们使用了虚拟化技术,用虚拟主机来代替传统的物理主机.在 Hadoop 层,我们使用 Hadoop Distributed File System(简称 HDFS)[23]作为分布式存储环境,利用 Hadoop MapReduce(简称 HMR)[23]来部署分布式计算环境.在仿真层,我们将待输入的 OSN 拓扑数据提交到 HDFS 存储环境中,并将与 OSN 蠕虫传播仿真相关的 MR-Job 提交到 HMR 所提供的分布式计算环境中,以实现仿真过程.通过以上 3 层,共同构成了我们的仿真系统原型.

4.2 实验评估

为了验证仿真系统的有效性,本文在 Twitter 数据集上进行了 OSN 蠕虫传播仿真实验.首先介绍仿真实验的环境以及相关的实验数据集;然后给出了 OSN 蠕虫传播仿真实验的结果并分析各种因素对传播过程的影响;最后,通过增加实验节点从而减少实验时间的方式,验证了仿真系统的可扩展性.

4.2.1 实验环境及数据集

实验主要利用了虚拟化技术,即通过开源的虚拟化软件 Oracle Virtualbox 将高性能 Dell R910 服务器虚拟出多个主机节点,并以此作为底层的分布式硬件环境.表 1 给出了实验过程中所用到的各种软、硬件参数.根据硬件配置参数的设定,我们可以在一台服务器中虚拟 15 台~20 台处理节点.

Table 1 Software and Hardware

表 1 实验所用软件和硬件

Software			Hardware			
	Type	Version		CPU	Memory	Disk
Host OS	Windows	Server 2008 R2	Host computer	E7520×48	120G	1T SAS
Virtual OS	Ubuntu	10 LTS (64bit)	Virtual computer	E7520×2	4G	40G SAS

在实验过程中,我们用到了两个在不同时期收集的 Twitter 数据集:A-DS 和 S-DS,其中,A-DS 是由 KAIST

© 中国科学院软件研究所 http://www.jos.org.cn

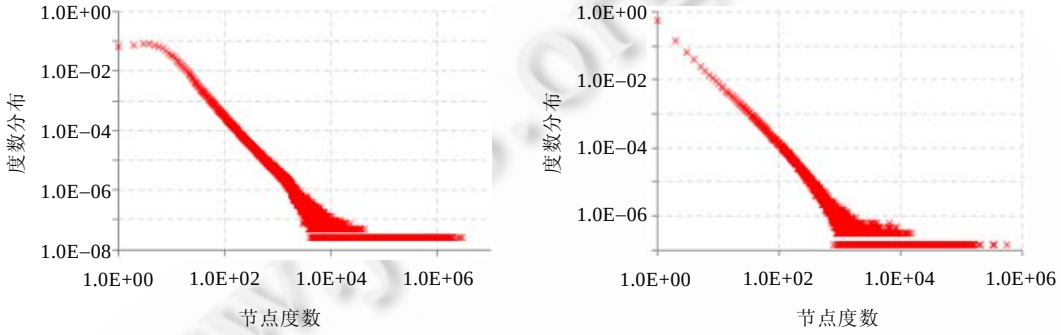
的 Advanced Networking LAB 收集的[24],S-DS 是由 ASU 的 Social Computing Data Repository 提供的[25].原始的 A-DS 和 S-DS 的数据格式分别为(userID,friendID)和(friendID,userID),为了适应本文的实验场景,我们将数据格式统一为(userID,friendID)格式.表 2 给出了这两个数据集的一些特征参数,其中,A-DS 的特征路径长度和聚合系数可从文献[20,26]中直接获取,S-DS 中这两个特征参数则根据文献[20,27]所提供的算法计算得出.从两个数据集参数可以看出,Twitter 网络符合 Small World 特性,即拥有较小的特征路径和较大的聚合系数.

Table 2 DataSet characteristic parameter

表 2 数据集特征参数

	Size (GB)	Users (M)	Relations (M)	Average degree	Average path length	Clustering coefficient
A-DS	24.3	47.1	1472.4	36.7	4.1	0.182
S-DS	1.1	11.3	85.3	12.9	5.3	0.238

此外,Scale Free 特性是社交网络的另一个特点,即节点度数满足幂率分布.而为了验证本文所使用的 Twitter 网络具备该特性,我们利用双对数图(log-log plot)给出了节点的度数分布情况,如图 5 所示.从图 5 所给出的统计结果可以看出,A-DS 和 S-DS 两个数据集的度数也均满足幂率特性.



(a) A-DS (b) S-DS

Fig.5 Degree distribution of the Twitter datasets

图 5 Twitter 数据集的度数分布

综上所述,根据表 2 和图 5 给出的特征路径长度、聚合系数以及度数分布可以看出,实验所使用的数据符合一般社交网络的特性,即 Small World 特性和 Scale Free 特性,进而说明本文通过仿真实验所得出的结论同样适用于其他满足社交网络特性的数据集.

4.2.2 实验结果及分析

实验首先在 A-DS 和 S-DS 数据集和 15 个虚拟主机节点上运行 OSN 蠕虫仿真实验,以验证系统的有效性.同时,根据实验结果分析了影响蠕虫传播的相关因素;然后,通过扩展主机节点的方式减少实验所需时,以说明仿真系统具有较好的可扩展性.

在进行 OSN 蠕虫传播和感染仿真实验之前,我们需要通过初始化 MR-Job 将原始的二元组格式转化为第 3.1 节公式(3)所给出的 *k*-*v* 格式.表 3 给出了转化前后数据集的相关信息,这里,我们将转化后的数据集称为 *A-DS*(*kv*)和 *S-DS*(*kv*).从表中可以看出,通过本文所提出的转化方式不仅减小了待处理数据本身的体积,同时也很大程度地减少了待处理的记录数——转化后 *A-DS*(*kv*)的记录数不到原始个数的 3%,由此可以看出,本文所给出的转化方式在一定程度上提高了仿真实验的运行效率.

Table 3 Result of the initialization MR-Job

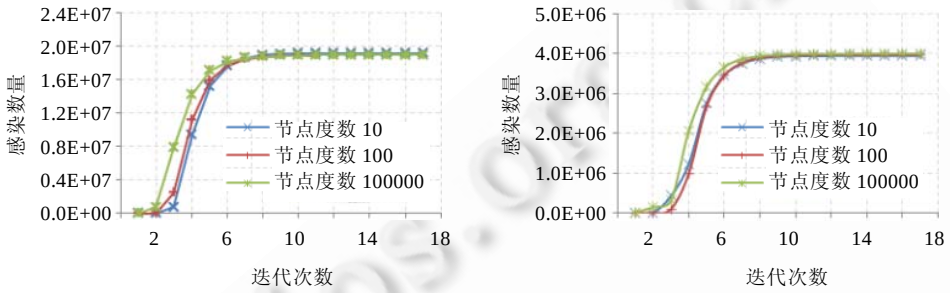
表 3 初始化 MR-Job 运行结果

	Size (Byte)	Records		Size (Byte)	Records
A-DS	26 172 280 241	1 468 365 182	S-DS	1 233 949 980	85 331 845
A-DS(*kv*)	13 956 781 561	40 113 281	*S-DS*(*kv*)	740 304 329	6 626 985

© 中国科学院软件研究所 http://www.jos.org.cn

为了验证仿真系统的有效性,同时也为了分析影响蠕虫传播的相关因素,本文进行了如下 3 类仿真实验:第 1 类实验是利用不同的策略 g,分析在固定登陆概率 P_t 和固定消息响应概率 P_r 下的传播情况,称为 g 实验;第 2 类实验是在选定 g 和 P_r 的情况下,分析 P_t 对蠕虫传播的影响,称为 P_t 实验;第 3 类是 P_r 实验,即通过选定 g 和 P_t,研究 P_r 所造成的影响.以下分别给出这 3 类实验的结果及其分析,其中,每类实验的结果是 10 次实验的平均值.

在 g 实验过程中,我们选定 P_t 和 P_r 均为固定值 0.5,进而分析在不同策略下的 OSN 蠕虫传播情况.根据对传统蠕虫仿真实验[8,9]的分析,选择感染能力不同的初始节点很大程度上会影响蠕虫传播的速度和感染数量.但从图 6 所示的实验结果可以看出,虽然我们选择了拥有不同好友数量(即初始感染能力)的用户作为初始化感染节点,但是无论从感染速度还是最终的感染数量上来看,效果几乎相同.分析造成这种情况发生的原因,是由于 OSN 蠕虫在传播过程中主要依赖于自身的网络拓扑结构,即如果节点 A 被感染,则其只会传播给周围的好友,而不能像传统蠕虫那样具有随机扫描并感染网络中任意节点的能力.

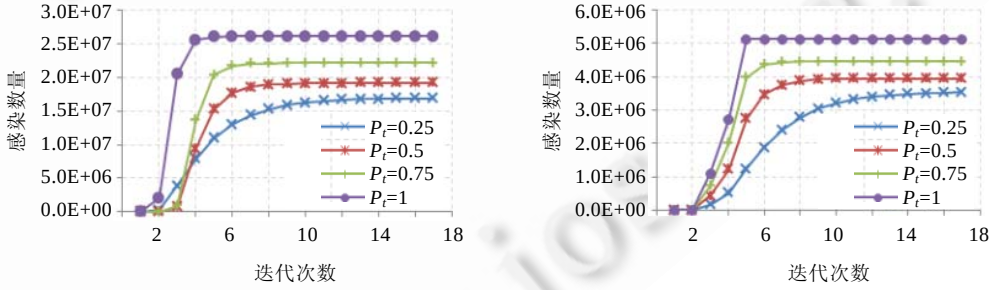


(a) *A-DS*(*kv*) [P_t=0.5,P_r=0.5] (b) *S-DS*(*kv*) [P_t=0.5,P_r=0.5]

Fig.6 Simulation results of g experiment

图 6 g 实验仿真结果

图 7 给出了在两个数据集上的 P_t 实验结果.可以看出:当 P_t 为最小值 0.25 时,OSN 蠕虫需要 10 次~11 次迭代才能达到感染峰值;而当 P_t 为最大值 1 时,OSN 蠕虫只需 4 次~5 次迭代即可到达感染峰值.此外,观察实验结果还可以看出,随着 P_t 的增加,感染峰值均有大幅度的提高.由于用户登陆的概率一定程度上反映了某个 OSN 网络的流行程度,因此通过 P_t 实验我们可以得出,越是流行的 OSN 网络,当蠕虫爆发时,其受危害程度越大.



(a) *A-DS*(*kv*) [g='random',P_r=0.5] (b) *S-DS*(*kv*) [g='random',P_r=0.5]

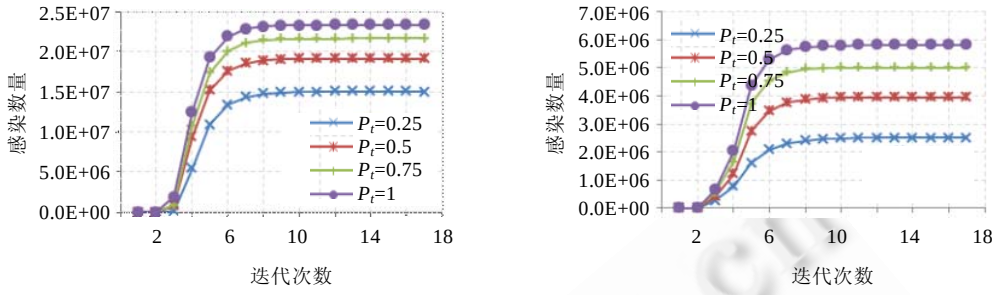
Fig.7 Simulation results of P_t experiment

图 7 P_t 实验仿真结果

图 8 给出了在两个数据集上的 P_r 实验结果.与 P_t 实验类似,随着用户对消息响应概率的增加,OSN 蠕虫传播速度加快的同时感染峰值得到了提高.与 P_t 实验不同,P_r 对传播速度的影响并不如 P_t 的影响明显.在 P_t 实验中,当 P_t=1 时,通过 4 次~5 次迭代即可达到感染峰值;而在 P_r 实验中,即使 P_r=1,仍然与其他取值情况一样,需要通过 8 次~10 次迭代才能达到感染峰值.此外,通过综合 P_r 实验和 P_t 实验可以得出,在 OSN 蠕虫的传播过程中,

© 中国科学院软件研究所 http://www.jos.org.cn

用户登陆的概率 P_t 和对消息响应的概率 P_r 直接影响了 OSN 蠕虫感染的规模和速度.而在现实中,P_t 和 P_r 分别反映了 OSN 网站的流行程度以及用户的安全意识程度[5].因此,对于目前流行的 OSN 网站来说,提高用户的安全意识程度,降低用户响应恶意消息的概率,是有效抑制 OSN 蠕虫快速传播的前提.



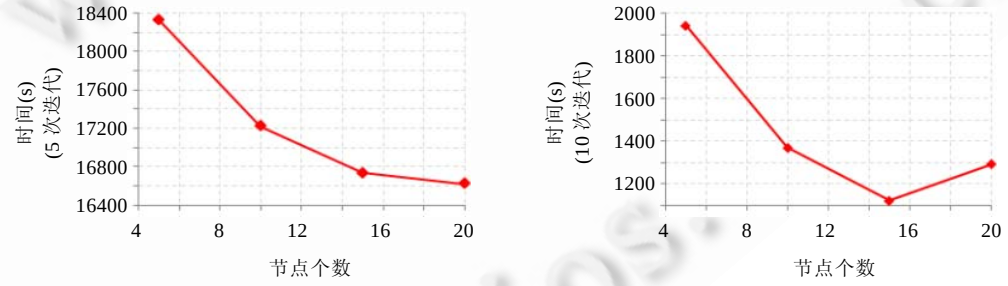
(a) *A-DS*(*kv*) [*g*='random',P_t=0.5] (b) *S-DS*(*kv*) [*g*='random',P_t=0.5]

Fig.8 Simulation results of P_r experiment

图 8 P_r 实验仿真结果

我们通过以上 3 个仿真实验,较好地验证了本文所提出的 OSN 蠕虫仿真系统的有效性.同时我们还可以看出,该系统很好地克服了传统仿真方法的不足,即:一方面,通过为每个节点引入属性特征,从而解决了基于数学模型的仿真方法无法针对单个节点进行仿真的问题.例如,在本文进行的实验过程中,正是通过调节单个节点的 P_t 和 P_r 等不同属性值,进而观察 OSN 蠕虫在不同条件下的传播情况;另一方面,该系统利用 MapReduce 技术很好地克服了基于数据包级和基于测试床的仿真方法很难对大规模网络蠕虫传播进行仿真的问题.正如表 2 所示,本文仅利用 15 个计算节点即完成了节点规模在 10^7 以及关系规模在 10^9 的大规模 OSN 蠕虫仿真.

最后,由于本文所提出的仿真系统利用了"线性可扩展"的 MapReduce 技术,因此理论上通过简单增加处理节点数量即可满足可扩展性.为了验证该结论,本文还给出了在不同节点数量上运行同一实验所需时间的结果,如图 9 所示.其中,图 9(a)是在 *A-DS*(*kv*)数据上运行 5 次迭代的时间,图 9(b)是在 *S-DS*(*kv*)数据上运行 10 次迭代的结果.



(a) *A-DS*(*kv*) (b) *S-DS*(*kv*)

Fig.9 Scalability experiment

图 9 可扩展性实验

从图 9 所给出的实验结果可以看出,随着处理节点个数的增加,实验所需的时间基本呈现出线性下降的趋势,进而说明了仿真系统具备了较好的可扩展性.但在图 9(b)中,随着处理节点规模增加到一定程度(20 个节点),执行时间有所回升.这主要是因为在计算过程中,并发执行的任务个数主要由原始的输入数据的大小所决定,而当处理节点个数满足最大并发任务数时,即达到了最优化结果,如果继续增加处理节点,反而会造成额外的系统开销(例如数据传输、任务协调等).因此在进行实际的仿真实验时,我们需要根据待处理数据的大小来分配合理的节点个数,以达到最优化处理,同时也避免了对硬件资源的浪费.

© 中国科学院软件研究所 http://www.jos.org.cn

5 总 结

为了更好地进行 OSN 蠕虫的研究,本文利用云计算的核心并行计算技术——MapReduce,提出了一种针对大规模 OSN 蠕虫传播的仿真方法.该方法首先构建基于节点属性的 OSN 图,并利用其描述了 OSN 蠕虫传播过程的不同阶段;然后,通过不同的 Map 方法和 Reduce 方法来实现整个蠕虫的仿真过程.为了验证该方法的有效性和可扩展性,本文在实现仿真系统原型的基础上,在两个大规模数据集上(节点数量均在千万级以上)进行了仿真实验.实验结果显示,本文所提出的仿真方法可以在相关研究的过程中起到基础性支持作用.后续工作将分为两个方面:一是将关于 OSN 蠕虫的最新研究应用到该仿真系统中,例如,考虑用户行为特征[28]对 OSN 蠕虫传播的影响等;另一个是考虑如何利用仿真系统对检测和防御 OSN 蠕虫的方法进行研究和实际评估.

References:

[1] 20 social media statistics. 2011. http://www.hongkongwebdesign.com.hk/blog/2011/09/13/20-social-media-statistics-infographic

[2] Patsakis C, Asthenidis A, Chatzidimitriou A. Social networks as an attack platform: Facebook case study. In: Proc. of the 8th Int'l Conf. on Networking. Gosier: IEEE Computer Society, 2009. 245−247. [doi: 10.1109/ICN.2009.77]

[3] Xu W, Zhang F, Zhu S. Toward worm detection in online social networks. In: Proc. of the 26th Annual Computer Security Applications Conf. New York: ACM Press, 2010. 11−20. [doi: 10.1145/1920261.1920264]

[4] Faghani MR, Saidi H. Malware propagation in online social networks. In: Proc. of the 4th Int'l Conf. on Malicious and Unwanted Software (MALWARE 2009). Montreal: IEEE Computer Society, 2009. 8−14. [doi: 10.1.1109/MALWARE.2009]

[5] Luo WM, Liu JB, Liu J, Chen XF. Propagation analysis of XSS worms in social network. Computer Engineering, 2011,37(10): 128−130 (in Chinese with English abstract).

[6] Sun X, Liu YH, Zhu JQ, Li FP. Research on simulation and modeling of social network worm propagation. Chinese Journal of Computers, 2011,34(7):1252−1260 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01252]

[7] Yan GH, Chen GL, Eidenbenz S, Li N. Malware propagation in online social networks: Nature, dynamics, and defense implications. In: Proc. of the 6th ACM Symp. on Information, Computer and Communications Security (ASIACCS 2011). New York: ACM Press, 2011. 196−206. [doi: 10.1145/1966913.1966939]

[8] Staniford S, Paxson V, Weaver N. How to own the Internet in your spare time. In: Proc. of the 11th USENIX Security Symp. Berkeley: USENIX Association, 2002. http://www.icir.org/vern/papers/cdc-usenix-sec02/

[9] Zou CC, Gong W, Towsley D. Code red worm propagation modeling and analysis. In: Proc. of the 9th ACM Symp. on Computer and Communication Security. New York: ACM Press, 2002. 138−147. [doi: 10.1145/586110.586130]

[10] Chen ZS, Gao LX, Kwiat K. Modeling the spread of active worms. In: Proc. of the IEEE INFOCOM 2003. Piscataway: IEEE Press, 2003. 1890−1990. [doi: 10.1109/INFCOM.2003.1209211]

[11] Liljenstam M, Yuan Y, Premore BJ, Nicol DM. A mixed abstraction level simulation model of large-scale Internet worm infestations. In: Proc. of the 10th IEEE/ACM Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2002). Washington: IEEE Computer Society, 2002. 109−116. [doi: 10.1109/MASCOT.2002.1167067]

[12] Wang YW, Jing JW, Xiang J, Liu Q. Topology aware worm simulation and analysis. Ruan Jian Xue Bao/Journal of Software, 2008,19(6):1508−1518 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1508.htm [doi: 10.3724/SP.J.1001.2008.01508]

[13] Riley GF, Sharif MI, Lee W. Simulating Internet worms. In: Proc. of the 12th IEEE/ACM Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2004). Washington: IEEE Computer Society, 2004. 268−274. [doi: 10.1109/MASCOT.2004.1348281]

[14] Li LQ, Jiwasurat S, Liu P, Kesidis G. Emulation of "single-packet" UDP scanning worms in large enterprise. In: Proc. of the 19th Int'l Teletraffic Congress (ITC-19). Beijing, 2005. http://130.203.133.150/showciting;jsessionid=D7A14EA7A2F2EE5E0BD411C1136AFF33?cid=935424

[15] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008,51(1):107−113. [doi: 10.1145/1327452.1327492]

© 中国科学院软件研究所 http://www.jos.org.cn

[16] Watts D, Strogatz S. Collective dynamic of small world networks. Nature, 1998,393(6684):440−442. [doi: 10.1038/30918]

[17] BRITE. http://www.cs.bu.edu/brite/

[18] DETERlab testbed. http://www.isi.edu/deter

[19] Grossman J. Cross-Site scripting worms & viruses: The impending thread & the best defense. 2006. http://www.whitehatsec.com/downloads/WHXSSThreads.pdf

[20] Kwak H, Lee C, Park H, Moon S. What is Twitter, a social network or a news media? In: Proc. of the 19th Int'l Conf. on World Wide Web (WWW 2010). New York: ACM Press, 2010. 591−600. [doi: 10.1145/1772690.1772751]

[21] Cormen TH, Leiserson CE, Rivest RL. Introduction to Algorithms. Cambridge: MIT Press, 1990.

[22] Thusoo A, Sarma S, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R. Hive: A warehousing solution over a map-reduce framework. In: Proc. of the Conf. on Very Large Databases (VLDB 2009). Lyon, 2009. 137−150. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.151.2637

[23] Apache Hadoop. http://hadoop.apache.org

[24] Twitter social graph. http://an.kaist.ac.kr/~haewoon/release/twitter_social_graph

[25] Twitter data set. http://socialcomputing.asu.edu/datasets/Twitter

[26] Suri S, Vassilvitskii S. Couting triangles and the curse of the last reducer. In: Proc. of the 20th Int'l Conf. on World Wide Web (WWW 2011). New York: ACM Press, 2011. 607−614. [doi: 10.1145/1963405.1963491]

[27] Schank T, Wagner D. Approximating clustering coefficient and transitivity. Journal of Graph Algorithms and Applications, 2005, 9(2):265−275. [doi: 10.7155/jgaa.00108]

[28] Benevenuto F, Rodrigues T, Cha M, Almeida V. Characterizing user behavior in online social networks. In: Proc. of the 9th ACM SIGCOMM Conf. on Int'l Measurement Conf. (IMC 2009). New York: ACM Press, 2009. 49−62. [doi: 10.1145/1644893.1644900]

附中文参考文献:

[5] 罗卫敏,刘井波,刘静,陈晓峰.XSS 蠕虫在社交网络中的传播分析.计算机工程,2011,37(10):128−130.

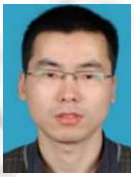
[6] 孙鑫,刘衍珩,朱建启,李飞鹏.社交网络蠕虫仿真建模研究.计算机学报,2011,34(7):1252−1260. [doi: 10.3724/SP.J.1016.2011.01252]

[12] 王跃武,荆继武,向继,刘琦.拓扑相关蠕虫仿真分析.软件学报,2008,19(6):1508−1518. http://www.jos.org.cn/1000-9825/19/1508.htm [doi: 10.3724/SP.J.1001.2008.01508]

和亮(1985－),男,河南焦作人,博士生,主要研究领域为网络与系统安全,在线社交网络恶意代码研究.
E-mail: windhl@yahoo.cn

苏璞睿(1976－),男,博士,副研究员,主要研究领域为网络与系统安全.
E-mail: supurui@is.iscas.ac.cn

冯登国(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为密码学,信息安全.
E-mail: feng@is.iscas.ac.cn



应凌云(1982－),男,博士,助理研究员,主要研究领域为僵尸网络分析.
E-mail: yly@is.iscas.ac.cn

王蕊(1981－),女,博士,助理研究员,CCF 会员,主要研究领域为网络与系统安全.
E-mail: wangrui@iie.ac.cn

© 中国科学院软件研究所 http://www.jos.org.cn