软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(7):1841−1860 [doi: 10.13328/j.cnki.jos.004843]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

高速网络超点检测的并行数据流方法*

周爱平[1,2,3], 程 光[1,3], 郭晓军[1,3], 梁一鑫[1,3]

[1](东南大学 计算机科学与工程学院,江苏 南京 211189)

[2](泰州学院 计算机科学与技术学院,江苏 泰州 225300)

[3](计算机网络和信息集成教育部重点实验室(东南大学),江苏 南京 211189)

通讯作者: 程光, E-mail: gcheng@njnet.edu.cn

摘 要: 超点检测对于网络安全、网络管理等应用具有重要意义.由于存在着高速网络环境下海量网络流量与有限系统资源之间的矛盾,在线准确地监测网络流量是一个极大的挑战.随着多核处理器的发展,多核处理器的并行性成为算法性能提高的一种有效途径.目前,针对基于流抽样的超点检测方法存在计算负荷重、检测精度低、实时性差等问题,提出了一种并行数据流方法(parallel data streaming,简称 PDS).该方法构造并行的可逆 Sketch 数据结构,建立紧凑的节点链接度概要,在未存储节点地址信息的情况下,通过简单地计算重构超点的地址,获得了良好的效率和精度.实验结果表明:与 CSE(compact spread estimator),JM(joint data streaming and sampling method)方法相比,该方法具有较好的性能,能够满足高速网络流量监测的应用需求.

关键词: 网络测量;网络安全;超点检测;数据流;并行方法

中图法分类号: TP393

中文引用格式: 周爱平,程光,郭晓军,梁一鑫.高速网络超点检测的并行数据流方法.软件学报,2016,27(7):1841−1860. http://www.jos.org.cn/1000-9825/4843.htm

英文引用格式: Zhou AP, Cheng G, Guo XJ, Liang YX. Parallel data streaming method for detection of super points in high-speed networks. Ruan Jian Xue Bao/Journal of Software, 2016,27(7):1841−1860 (in Chinese). http://www.jos.org.cn/1000-9825/4843.htm

Parallel Data Streaming Method for Detection of Super Points in High-Speed Networks

ZHOU Ai-Ping[1,2,3], CHENG Guang[1,3], GUO Xiao-Jun[1,3], LIANG Yi-Xin[1,3]

[1](School of Computer Science and Engineering, Southeast University, Nanjing 211189, China)

[2](School of Computer Science and Technology, Taizhou University, Taizhou 225300, China)

[3](Key Laboratory of Computer Network and Information Integration, Ministry of Education (Southeast University), Nanjing 211189, China)

Abstract: Detection of super points is very important for some network applications such as network security and network management. Due to the imbalance between the massive amount of traffic data in high-speed networks and the limited system resource, it is a great challenge to accurately monitor traffic in high-speed networks online. With the development of multi-core processers, the parallelism of multi-core processers becomes an effective method of improving performance. Consider that the existing method based on flow sampling has heavy computing load, low detection accuracy and poor timeliness, this article presents a new approach, PDS (parallel data streaming).

* 基金项目: 国家高技术研究发展计划(863)(2015AA015603); 江苏省未来网络创新研究院未来网络前瞻性研究项目(BY2013095-5-03); 江苏省“六大人才高峰”高层次人才项目(2011-DZ024); 江苏省普通高校研究生科研创新计划(KYLX_0141)

Foundation item: National High Technology Research and Development Program of China (863) (2015AA015603); Jiangsu Future Networks Innovation Institute: Prospective Research Project on Future Networks (BY2013095-5-03); Six Talent Peaks of High Level Talents Project of Jiangsu Province (2011-DZ024); Research and Innovation Program for College Graduates of Jiangsu Province (KYLX_0141)

收稿时间: 2014-04-02; 修改时间: 2014-09-09; 采用时间: 2015-04-14

This method constructs parallel reversible sketch and builds a compact summary of node connection degrees. Addresses of super points are reconstructed by a simple calculation without address information of nodes. This makes PDS efficient and precise. The experimental results show that PDS is superior to the CSE (compact spread estimator) and JM (joint data streaming and sampling method) methods. Consequently, the proposed method satisfies application requirement of traffic monitoring in high-speed networks.

Key words: network measurement; network security; super point detection; data streaming; parallel method

在高速网络环境下在线准确地检测超点,对网络安全和网络管理具有重要意义.如果在测量时间区间内与源节点(目的节点)通信的不同目的节点(源节点)的数量超过预定阈值,则称源节点(目的节点)为超源(超目的).超源和超目的统称为超点,因此,超点检测是指检测在测量时间区间内与源节点(目的节点)通信的不同目的节点(源节点)的数量超过预定阈值的源(目的)节点[1].超点检测问题存在于网络安全与网络管理的许多应用中,如:端口扫描和蠕虫传播是在短时间内源节点与不同目的节点建立大量的链接引起的,而DDoS攻击是大量的节点泛洪到一个目的节点引起的.另外,在 P2P 和 CDN 中,一些服务器节点或对等节点可能会吸引大量的链接请求,而网络中其他大部分节点处于相对空闲的状态,检测这类节点有助于实现负载均衡和提高网络的整体性能[2].在高速网络环境下,超点检测的难点在于如何在有限的计算、存储资源下实现准确、实时的超点检测.

目前,高速链路上存在大量的并发活跃流,通过简单的方法维护每个流的状态是不可行的[3,4];而且,处理每个分组需要纳秒级时间.例如在 OC-768(40Gbps)链路上,设分组的平均大小为 40B,则需要 8ns 处理每个分组.在高速网络环境下,当每个报文到达时,仅仅通过 SRAM 更新每个流的状态是可行的,但 SRAM 容量小且昂贵,无法存储大量的流状态[5].然而,通过抽样和数据流方法近似测量流统计信息,在一定程度上能够缓解上述问题.抽样方法通过存储与处理少量的网络流量,极大地减少了所需的存储空间和计算资源,而基于抽样的超点检测方法获得的检测精度依赖于抽样率,未能获得高检测精度.数据流方法的主要思想是,利用 SRAM 实时地处理每个到达的报文.通过 SRAM 存储所有的信息是不切实际的,数据流方法的原理是仅存储与应用需求最相关的统计信息[2].传统的基于流抽样的超点检测方法需要维护 hash 表与多次访问内存,使其无法实时地处理到达的每个报文.另外,随着骨干链路速率的提高,基于流抽样的超点检测方法将面临更严重的实时性问题.

虽然在超点检测方面已经开展了许多研究,但实时性的问题没有得到很好的解决.近年来,随着多核处理器的发展,计算领域正在发生具有革命性影响的转变.多个厂商已经推出一批多核处理器,如:在通用多核处理器上,IBM 的 Power X Cell 8i 处理器、Sun 的 Rock 处理器、Intel 的 i7 处理器、AMD 的 Shanghai 处理器和我国的龙芯-3 处理器,它们有 4~9 个核.此外,在专用多核处理器上,Cisco 的数据包处理器、NVIDIA 的 GTX200 线程处理器、Intel 的 Larrabee 图像处理器,它们有几十至上百个核[6].在多核处理器计算平台上,并行技术是提高超点检测方法性能的一种有效途径.

因此,针对基于流抽样的超点检测方法存在计算负荷重、检测精度低、实时性差的问题,利用多点测量[7,8]的思想和数据流技术,本文提出了一种并行数据流方法.该方法构造了一个并行的可逆 Sketch 数据结构[9],该数据结构由多个二维的位数组组成.PDS 方法利用多线程并行技术实现了数据流的更新和归并过程,该过程由更新线程和归并线程构成.假设数据流被均匀分配到更新线程,当子流的每个报文到达时,更新线程更新本地的数据结构.也就是说,通过一组hash函数选择位数组的多位,将这些位置为1,测量时间周期结束后,更新线程向归并线程发送信息,然后归并线程对数据结构中对应位数组进行按位或运算,得到归并后的数据结构,在此基础上,通过概率计数方法[10]估计链接度,然后检测超点.这些 hash 函数通过下文中定理 1 构造,每个位数组共享一个hash行函数,对应一个hash列函数.虽然可逆Sketch数据结构没有存储节点源或目的信息,但能够通过简单的计算有效地重构超点的源或目的.利用真实的骨干网流量数据进行实验,理论分析和实验结果表明,并行数据流方法能够获得较好的精度和效率.

本文第 1 节介绍超点检测的相关工作和国内外研究的一些进展.第 2 节阐述并行数据流方法的主要思路.第 3 节从理论上分析并行数据流方法的存储开销、准确性及计算开销.第 4 节通过实验对并行数据流方法的性能进行分析.最后对本文进行总结及下一步研究方向的展望.

1 相关研究

由于在高速网络环境下存在着大量的网络流量数据和有限的系统资源之间的矛盾,实时、准确地测量和监控网络流量是一个极大的挑战.高速网络环境下实时、准确地检测超点,是网络测量与网络安全的难点,已经引起相关研究人员的关注.超点检测方法大致经历了“简单方法-抽样方法-数据流方法”的过程.

简单的超点检测方法利用 hash 表为每个源(目的)维护其链接的不同目的(源)[3,4],虽然该方法能够获得较高的检测精度,但却不能扩展到高速网络环境中.由于高速链路存在大量并发的活跃流,一方面,为了维护大量的流信息,需要使用大容量的存储器;另一方面,为了实时地处理到达的报文,需要使用高速存储器.虽然现有的 DRAM 容量大但其访问速度慢,且价格便宜;而 SRAM 访问速度快但其容量小,且价格相对昂贵.由于抽样与数据流方法在高速网络超点检测方面显现出优越的性能,因此已经得到广泛研究,并取得了一些研究成果.

在高速链路上,传统的基于 hash 的流抽样方法采用相同的概率对每个流进行独立抽样.节点的链接度估计通过节点的抽样链接度与抽样概率之比得到.虽然基于 hash 的流抽样方法通过存储、处理少量的网络流量减少了所需的存储空间和计算资源,然而,低抽样率导致较大的估计误差产生,从而降低了检测精度.针对基于 hash 的流抽样方法不能有效地扩展到 2.5Gbps 以上高速网络环境的问题,王洪波等人[11]提出了一种基于 Bloom Filter 流抽样的超点检测算法.该算法能够通过引入较小的空间代价而扩展到 10Gbps 高速网络环境中,然而该算法需要专用的硬件和一定量的高速存储器,实施代价过高.Venkataraman 等人[12]提出了两种基于流抽样的超点检测算法,其 one-level/two-level filtering 方法均利用传统的基于 hash 的流抽样技术来估计节点的链接度.由于受到流量突发性问题的影响,低抽样率使得算法不能获得高估计精度.另外,两种算法所需内存空间比文献[2]中第 2 种算法更高.程光等人[1]提出了一种具有自适应抽样功能的超点实时检测算法,该算法结合多种网络测量技术,表明在自适应性、资源可控性、测量精度等方面优于 Sampled 和 Bitmap[13]等算法.Kamiyama 等人[14]提出了基于流抽样的超点识别算法,通过设定测量时间区间、流数及识别概率,该算法能够在一定的内存空间内自适应地优化所有参数,同时满足处理时间的上界.Cao 等人[15]提出了基于抽样的在线超点识别算法,该算法由两级过滤、阈值 Bitmap 及误差修正这 3 部分构成.该算法首先通过两级过滤方法淘汰了大多数的小基数节点,然后利用阈值 Bitmap 方法估计候选节点的基数,最后采用误差修正方法补偿过滤与估计过程中产生的偏差.该算法的主要优势在于过滤了大部分的小基数节点,同时保留了候选节点,从而减小了所需的内存空间.

在高速网络环境下,超点检测的数据流方法能够实时地测量和监控节点链接度,从而有效地检测超点[16].研究表明:抽样方法和数据流方法适用于捕获信息谱中不同和互补的区域,抽样方法和数据流方法的结合能够恢复完整的信息.在此基础上,Zhao 等人[2]提出了两种基于抽样与数据流的超点检测方法:一种超点检测算法是在基于 hash 的流抽样算法的基础上提出的,通过数据流模块进一步过滤抽样流量,允许更高的抽样率,获得了比基于 hash 的流抽样算法更高的检测精度;另一种超点检测算法结合了数据流在高效存储、估计源(目的)的出度(入度)能力和抽样在产生候选源(目的)的能力,虽然该算法更加复杂,但却获得了比前一种算法更高的检测精度.后者的不足在于存储空间的利用率低,由于大多数节点具有小链接度,少数节点具有大链接度,从而位数组的大部分列被分配给了小链接度的节点,其包含大多数 0.为了提高空间利用率,Yoon 等人[5]利用一组 hash 函数从共享的位数组中随机地选择位,为每个节点建立虚拟 Bitmap,通过每个节点对应的虚拟 Bitmap 估计节点链接度,有效地利用了存储空间.为了检测超点,Li 等人[17]结合 CSE[5]与流抽样,提出了具有更高内存利用率的扫描检测方法,该方法减少了 3~20 倍的内存开销.然而,即使其内存开销减少 20 倍,也不足以检测较低速率的攻击,该方法只能解决扫描检测问题而不适用于一般的超点检测问题.另外,该方法还存在一些未解决的问题:位数组中每一位被所有的节点共享,使得节点的虚拟 Bitmap 可能没有被此节点的链接更新而是被其他节点更新,这样给该节点的虚拟 Bitmap 引入了噪声,从而影响了节点链接度估计的精度.由于 hash 冲突,虚拟 Bitmap 中不同位的数量发生变化,也会影响节点链接度估计的精度.节点链接度估计的精度分析基于每个链接独立、均匀地 hash 到共享位数组这样的假设,然而,一个节点的链接并没有独立 hash 到共享位数组.针对上述问题,Wang 等人[18]提出了两种基于数据流的超点检测方法:一种基于虚拟链接度概要的超点检测方法利用 hash 表存储节点的地址;然而,另一种基于可逆虚拟链接度概要的超点检测方法以牺牲较小的估计精度为代价识别超点,且不需要额外的内

存空间存储节点的地址.而且,上述两种方法与均匀流抽样技术相结合能够进一步减小内存复杂度.Shi 等人[19]提出了基于抽样与数据流技术的在线架构,该算法提高了检测精度,同时减少了内存使用和 CUP 处理时间.

随着骨干链路的速率提高以及海量并行数据流的存在,为了满足高速网络环境下在线准确地检测超点的应用需求,并行技术必然成为提高超点检测方法性能的一种有效途径.目前,并行技术在超点检测方面的研究较少,而其在数据挖掘方面已经开展了一些研究工作.例如,Shin 等人[20]的研究表明,现有方法存在测量时间周期短的局限性,从而提出了基于 GPU 的超点检测方法,该方法在 1GB 内存空间内获得多达 160Gbps 的吞吐量.Das 等人[21,22]提出了基于协作的加锁方法,通过一个并行流概要数据结构并行处理频繁项挖掘问题,然而,该方法的不足在于缺乏可扩展性.Cafaro 等人[23,24]提出了频繁项挖掘的两种并行算法,其不足在于不能够提供连续的全局频繁项.Zhang 等人[25,26]研究了在多核处理器平台上的带权值数据流的并行频繁项挖掘问题,提出了精度合成法,其基本思想是在多个线程中运行同一种算法的多个拷贝.该方法既降低了汇聚开销又保证了实时性,并已将其应用于分布式大流监控中.

2 并行数据流方法

本节将描述超点检测的并行数据流方法(parallel data streaming,简称 PDS).假设数据流 $S=p_1,\ldots,p_t,\ldots,p_n$ 被均匀分为 R 个子流 $S_1,S_2,\ldots,S_R$,子流 $S_l(1\leqslant l\leqslant R)$为由二元组$(s_{lt},d_{lt})$构成的时间序列.这样,数据流 S 就是子流 S_1,$S_2,\ldots,S_R$ 依时间戳 t 重新排列后的时间序列.并行数据流方法中,每个子流被分配到一个更新线程,每个更新线程维护一个相同的可逆 Sketch 数据结构,执行相同的操作,当测量时间周期结束后,更新线程将流量概要信息发送到归并线程,然后进行链接度估计和超点检测.

2.1 方法描述

PDS 方法的流程如图 1 所示.首先,初始化数据结构,对于到达的报文流,更新数据结构;测量时间结束后,对数据结构进行归并;然后,通过概率计数方法估计链接度;再设置阈值,确定超列;通过简单的逆计算重构节点,估计节点的链接度,将节点的链接度与阈值进行比较,从而检测超点.PDS 方法由更新模块、归并模块、估计模块、检测模块组成.

- 更新模块包含一个并行的可逆 Sketch 数据结构,该数据结构由一组二维位数组构成,对于到达的每个报文,取出报文的源-目的对的值,并计算其 hash 值,根据 hash 值确定数据结构中相应的位置,然后更新相应的位置对应的值;
- 归并模块对更新的数据结构中对应二维位数组进行按位或运算,得到归并的数据结构;
- 估计模块通过概率计数方法估计归并的数据结构中每一列位向量对应的节点链接度;
- 检测模块设置超点检测的阈值和确定超列,然后对数据结构中任意两个二维位数组的超列进行组合,利用定理 1 进行逆计算构造节点的 IP 地址,估计节点的链接度:如果节点链接度大于阈值,则认为该节点是超点.最后输出超点.重复上述步骤,直到处理完所有的超列组合.其中,阈值依据节点链接度估计值确定,超列依据节点链接度的估计值和阈值确定.

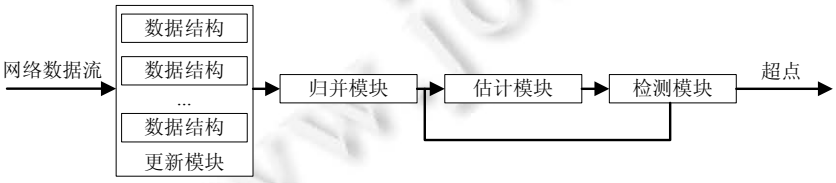


Fig.1 Flowchart of PDS

图 1 PDS 方法的流程图

下面给出了 PDS 方法的相关定义、数据结构的构造、数据结构的更新和归并过程、链接度估计方法及超

点检测方法.

2.2 相关定义

定义 1(网络数据流(network data stream)). 一个时间区间内顺序到达的报文序列,称为网络数据流,记为 $S=p_1,\ldots,p_t,\ldots,p_n$.其中,$p_t=(s_t,d_t)$表示第 t 个报文信息,s_t,d_t 分别表示该报文的源和目的.一个源由报文头的一个或多个源字段构成,如:源 IP、源端口、源 IP 和源端口的二元组;类似地,一个目的由一个或多个目的字段构成.

定义 2(超源(super source)). 节点的出度大于预定阈值的所有源构成的集合,称为超源,记为

$$SS(S,D)=\{s|OD(s)\geqslant \phi_{OD}F_{OD},s\in S\},$$

其中,$\phi_{OD}(0<\phi_{OD}<1)$是一个阈值;$OD(s)$是源 s 的出度,表示在测量时间区间内源链接的不同目的数;F_{OD} 是所有源的出度总和.

定义 3(超目的(super destination)). 节点的入度大于预定阈值的所有目的构成的集合,称为超目的,记为

$$SD(S,D)=\{d|ID(d)\geqslant \phi_{ID}F_{ID},d\in D\},$$

其中,$\phi_{ID}(0<\phi_{ID}<1)$是一个阈值;$ID(d)$是目的 d 的入度,表示在测量时间区间内目的链接的不同源数;F_{ID} 是所有目的的入度总和.

定义 4(超列(super column)). 对于任一列向量 $A_i[\cdot][j]$,通过 hash 函数映射到该列的总链接度为

$$D(A_i[\cdot][j])=-m\ln\frac{U_{A_i[\cdot][j]}}{m}.$$

当且仅当 $D(A_i[\cdot][j])\geqslant \phi F$ 时,称为超列.其中,$U_{A_i[\cdot][j]}$ 表示位向量 $A_i[\cdot][j]$中 0 的位数;F 是总链接度,通过 Bitmap 算法估计[11].

定义 5(链接度(connection degree)). 在一定的测量时间区间内,不同的源-目的对的总数,称为链接度,记为 n.链接度的估计表示为

$$\hat{n}=-m\ln\frac{U_n}{m},$$

其中,U_n 表示位向量中 0 的位数,m 表示位向量的大小.

定理 1[27]. 令 $n=n_1n_2\ldots n_H$,其中,$n_1,n_2,\ldots,n_H$ 是两两互质的正整数,则对任意整数 $c_1,c_2,\ldots,c_H$,联立方程组

$$x\equiv c_i(\mathrm{mod}\ n_i),i=1,2,\ldots,H$$

对模 n 有唯一解:

$$x\equiv\sum_{i=1}^{H}c_i(m_i(m_i^{-1}\bmod n_i))\ (\mathrm{mod}\,n),$$

其中,$m_i=n/n_i$, $m_i^{-1}\bmod n_i$ 表示 m_i 对模 n_i 的乘法逆元,且 $m_im_i^{-1}\equiv 1\ (\mathrm{mod}\,n_i)$.

2.3 数据结构

可逆 Sketch 数据结构表示为

$$A^l=(A_1^l,A_2^l,\ldots,A_H^l)(1\leqslant l\leqslant R),$$

其中, $A_i^l\ (1\leqslant i\leqslant H)$是一个二维位数组 $A_i^l[j][k]\ (0\leqslant j\leqslant m-1,0\leqslant k\leqslant n_i-1)$;$R$ 表示可逆 Sketch 的数量;H 表示二维位数 $A_i^l\ (1\leqslant i\leqslant H)$的个数, $A_i^l\ (1\leqslant i\leqslant H)$对应于一个列 hash 函数 $h_i(1\leqslant i\leqslant H)$,共享一个行 hash 函数 f.行 hash 函数 f、列 hash 函数 h_i 分别表示为

$$f:\{0,1,\ldots,M-1\}\to\{0,1,\ldots,m-1\},$$
$$h_i:\{0,1,\ldots,N-1\}\to\{0,1,\ldots,n_i-1\},$$

其中,M 表示所有源、目的的二元组数,N 表示源数,m 表示二维位数组 $A_i^l\ (1\leqslant i\leqslant H)$的行数,$n_i$ 表示第 i 个二维位数组 $A_i^l\ (1\leqslant i\leqslant H)$的列数.参数的具体说明见表 1.

列 hash 函数 h_i 的构造如下:

$$\begin{cases} h_i(x) = g(x) \bmod n_i, 1 \leqslant i \leqslant H \\ g(x) = (ax + b) \bmod p \end{cases} \tag{1}$$

其中,$n_1,n_2,\ldots,n_H(\leqslant p)$是两两互质的整数;$p$ 是大于 N 的质数;a,b 是整数,且 $1\leqslant a\leqslant p,0\leqslant b\leqslant p$.如果 $n_i(1\leqslant i\leqslant H)$是一个质数,则有:

$$\Pr_{1\leqslant i\leqslant H}\{h_i(x)=h_i(y)\}\leqslant\frac{1}{n},$$

其中,x 与 $y(0\leqslant x,y\leqslant n-1)$是不相同的数,$h_i$ 是任意一个满足方程组(1)的列 hash 函数.

Table 1 Interpretation of symbols for data structure

表 1 数据结构的符号说明

符号	符号说明
R	数据结构的数量
H	位数组 A_i^l 的数量
H^*	位数组 B_i^l 的数量
A^l	第 l 个数据结构的可逆 Sketch
A_i^l	位数组 A_i^l 的第 l 个位向量
B_i^l	位数组 B_i^l 的第 l 个位向量
$A_i^l[j][k]$	位数组 A_i^l 中位于(j,k)的元素
$B_i^l[j][k]$	位数组 B_i^l 中位于(j,k)的元素
n_i	位数组 A_i^l 的列数
n_i^*	位数组 B_i^l 的列数
m	位数组 A_i^l 或 B_i^l 的行数
f	行 hash 函数
h_i	位数组 A_i^l 的第 i 个列 hash 函数
h_i^*	位数组 B_i^l 的第 i 个列 hash 函数

2.4 更新和归并过程

PDS 方法的更新和归并架构如图 2 所示,由 R 个更新线程和一个归并线程组成.更新线程处理子流与更新本地数据结构,测量时间周期结束后,更新线程与归并线程通信,归并线程对流量概要信息进行归并.

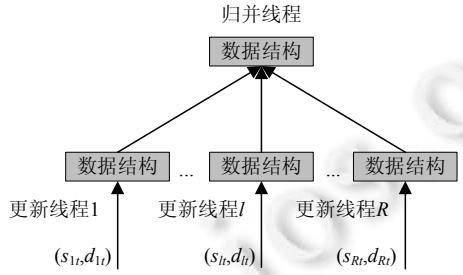


Fig.2 Updating and merging architecture of PDS

图 2 PDS 方法的更新和归并架构

更新与归并线程的具体实现、线程的调度策略、多线程优化机制及任务分配与调度说明如下.

(1) 更新线程 $l(1\leqslant l\leqslant R)$为子流 S_l 维护一个可逆 Sketch 数据结构.

首先,将 A_i^l $(1\leqslant i\leqslant H,1\leqslant l\leqslant R)$中的每一位初始化为 0,当报文 $p_t=(s_t,d_t)$到达时, A_i^l $(1\leqslant i\leqslant H,1\leqslant l\leqslant R)$中第 $f(s_t\|d_t)$行和第 $h_i(s_t)$列的位置设置为 1,即:

$$A_i^l[f(s_t\,\|\,d_t)][h_i(s_t)]=1,1\leqslant i\leqslant H,1\leqslant l\leqslant R.$$

测量时间周期结束后,更新线程 l 向归并线程发送流量概要信息,再将 A_i^l $(1\leqslant i\leqslant H,1\leqslant l\leqslant R)$中的每一位重置为 0.

(2) 归并线程接收更新线程发送的信息,对流量概要信息进行归并,维护一个相同的数据结构.

对 R 个可逆 Sketch 数据结构相应位数组中的位向量进行按位或运算,得到位向量 $A_i[\cdot][j]$,即:

$$A_i[\cdot][j]=A_i^1[\cdot][j]|A_i^2[\cdot][j]|...|A_i^R[\cdot][j],1\leqslant i\leqslant H,1\leqslant j\leqslant n_i,$$

其中, $A_i^l[\cdot][j]$ 表示第 l 个可逆 Sketch 的第 i 个二维位数组的第 j 列,|表示按位或运算符.

(3) 线程流水线是提高多核处理器性能的关键因素,将线程流水线引入线程调度策略.

线程调度策略是指将更新线程划分为不同的线程流水线,调度到不同的核内,通过核内的缓存共享数据,使得核内多个更新线程能够执行线程流水线,实现更新线程并行执行,达到提高超点检测方法性能的目的.为了实现线程流水线:首先,定义测度 P 与 F,P 表示更新线程划分为某个线程流水线的概率,F 表示线程流水线的当前特征指标(更新线程相关性、更新线程同步等);然后,通过比较更新线程的测度 P 值与每个线程流水线的测度 F 值,找到一个最适合的线程流水线,将该更新线程插入到线程流水线所在核的更新线程队列中,完成该更新线程的调度,并重新计算该线程流水线的测度 F 值,再进行下一更新线程的调度.在所有更新线程执行完成后,调度归并线程.

(4) 主要从线程数量、CPU 亲和力和 Cache 优化这 3 个方面阐述多线程优化机制.

在大多数情况下,线程数量比硬件线程数量要多.但是线程过多可能会严重地降低程序的性能,主要体现在两个方面:首先,将给定的工作量划分给过多的线程会造成每个线程的工作量过少,可能导致线程启动、终止以及线程上下文切换比程序实际执行的时间还要长;其次,太多并发线程的存在将导致共享有限硬件资源的开销增大.因此,根据实际的硬件资源和应用需求,通过不断地调优来确定最优的更新线程数量,使得更新线程数量与硬件线程数量之间保持合理的平衡,将有利于提高超点检测方法的性能.

CPU 亲和力分为软亲和力和硬亲和力两类:软亲和力是指内核中的线程调度器会尝试着将某个线程绑定到特定的 CPU 核上,如果这个 CPU 核一直处于忙碌状态,那么调度器会将此线程调度到其他处于空闲状态的 CPU 核上,这意味着线程通常不会在 CPU 核之间频繁迁移;硬亲和力是指线程固定在某个 CPU 核上运行而不是在不同 CPU 核之间进行频繁的迁移,这样不仅改善了程序的性能,还提高了程序的可读性.从以上分析不难看出:在某种程度上,硬亲和力比软亲和力具有一定的优势.因此,本文方法利用硬亲和力将更新线程和归并线程绑定到不同的核上.

为了节省带宽,将数据压缩得更加紧凑,或减少其在 CPU 核之间的移动,这种方法对串行程序是有益的,而对多核多线程程序是性能降低的.对于多核多线程程序,数据不仅在 CPU 核和存储器之间传输,还会在 CPU 核之间传输.根据数据相关性,写后读与写后写模式会涉及到数据的移动.这两种读写模式引起数据的竞争,从而影响了超点检测的性能.因此,在超点检测的并行数据流方法设计过程中,需要综合考虑多个更新线程的访存带宽和竞争问题,将更新线程使用的数据放在不同的 Cache 行中,将只读数据和可写数据分离开.

(5) 任务分配是多核时代提出的新概念,任务分配过程就是将线程分配给处理器核的过程.

为了在多核处理器并行系统中有效地进行任务分配,通过建立任务分配模型和利用启发式算法实现线程到处理器核的分配.不失一般性,设多核处理器并行系统包含 N_{node} 个计算节点 $D_1,D_2,...,D_{N_{node}}$,每个处理器包含 N_{core} 个核 $C_1,C_2,...,C_{N_{core}}$;待分配的并行程序有 N_{proc} 个进程 $P_1,P_2,...,P_{N_{proc}}$,进程 P_i 包含 M_i 个线程 $T_1,T_2,...,T_{M_i}$.因此,线程的总数为 $N_{thread}=\sum_{k=0}^{N_{proc}}M_k$.待分配的并行程序可以表示为一个无向图 $G=(V,E)$,其中,

- $V=\{V_i\}$,节点 V_i 表示一个二元组$\langle\{T_i\},\{P_i\}\rangle$,其中,$T_i$ 表示节点对应的线程号,P_i 表示线程所属的进程号;
- $E=\{E_{ij}\}$,无向边 E_{ij} 表示线程 T_i 与 T_j 之间通信或共享数据的关系,无向边 E_{ij} 的权值 W_{ij} 表示线程 T_i 与 T_j 之间通信或共享数据的频繁程度.

任务分配可以通过函数 f 将线程映射到处理器核.任务分配算法的目标是将无向图 G 划分为 $N_{node}\times N_{core}$ 个不相交的子图 G_{ij}.子图 G_{ij} 中的节点对应了分配给计算节点 D_i 中处理器核 C_j 的线程.子图 G_{ij} 划分满足以下条

件:同一进程的线程被划分给同一计算节点;各子图包含的节点数量基本相等;分配给不同计算节点的线程之间的权值之和在所有的子图划分方案中最小;尽量不将多个进程分配给同一处理器核.由于子图 G_{ij} 划分问题的最优解是 NP 难的,通过启发式算法获得子图 G_{ij} 划分的近优解,从而完成了线程到处理器核的分配.由于不同处理器的内核之间的通信开销远大于同一处理器的内核之间的通信开销,通过任务复制将相互关联的子图 G_{ij} 变成相互独立的子图,消除任务之间的通信开销,使得处理器相互独立地执行任务.对每个独立子图分别使用遗传算法,将线程分配给处理器中的某个内核,并对其进行合理的调度.

假设存在两个超点,$A_i(1\leqslant i\leqslant H)$中可能有两个超列,通过基于可逆 Sketch 的检测方法至少获得 2^H 个超点.为了降低基于可逆 Sketch 的检测方法产生的误报,建立另一个 Sketch 数据结构 $B^l=(B_1^l,B_2^l,...,B_{H^*}^l)(1\leqslant l\leqslant R)$ 用于验证通过基于可逆 Sketch 检测方法获得的超点是否映射到 B_i 中的超列,其中, $B_i^l(1\leqslant i\leqslant H^*)$ 是一个二维位数组 $B_i^l[j][k](0\leqslant j\leqslant m-1,0\leqslant k\leqslant n_i^*-1)$. 更新线程 $l(1\leqslant l\leqslant R)$和归并线程需要进行类似的更新和归并,如下所示:

$$B_i^l[f(s_t\,\|\,d_t)][h_i^*(s_t)]=1,1\leqslant i\leqslant H^*,1\leqslant l\leqslant R,$$

$$B_i[\cdot][j]=B_i^1[\cdot][j]\,|\,B_i^2[\cdot][j]\,|\,...\,|\,B_i^R[\cdot][j],1\leqslant i\leqslant H^*,1\leqslant j\leqslant n_i^*,$$

其中,H^*表示二维位数 B_i^l 的个数, B_i^l 对应一个列 hash 函数 h_i^*, 共享一个行 hash 函数 f;列 hash 函数 h_i^* 与 h_i 是相互独立的,列 hash 函数 h_i^* 的构造方法类似于 h_i.

列 hash 函数 h_i^* 表示为

$$h_i^*:\{0,1,...,N-1\}\to\{0,1,...,n_i^*-1\},$$

其中, n_i^* 表示第 i 个二维位数组 B_i^l 的列数,参数的具体说明见表 1.

由于更新线程和归并线程均维护了一个可逆 Sketch 数据结构,从而极大地降低了更新线程和归并线程之间的通信开销.

2.5 链接度估计

对于任意源 s,令映射到 $A_i(s)=A_i[\cdot][h_i(s)](1\leqslant i\leqslant H)$的所有流为 $T_i(s)$.在测量时间区间内,如果没有其他的源通过列 hash 函数 $h_i(1\leqslant i\leqslant H)$映射到 $A_1(s),A_2(s),\ldots,A_H(s)$,那么 s 的出度 $OD(s)=|T_1\cap T_2\cap\ldots\cap T_H|$;否则,$|T_1\cap T_2\cap\ldots\cap T_H|$是通过列 hash 函数 $h_i(1\leqslant i\leqslant H)$映射到 $A_1(s),A_2(s),\ldots,A_H(s)$的所有源的出度总和.当 $n=n_1n_2\ldots n_H\geqslant p$ 时,不存在源 $s'\in S$,使得 s'被列 hash 函数 $h_i(1\leqslant i\leqslant H)$映射到 $A_1(s),A_2(s),\ldots,A_H(s)$.因此,$s$ 的出度 $OD(s)=|T_1\cap T_2\cap\ldots\cap T_H|$.

$OD(s)$的估计为

$$\hat{OD}(s)=-m\ln\frac{U_{A(s)}}{m},$$

其中,$A(s)=A_1(s)\ \&\ A_2(s)\ \&\ \ldots\ \&\ A_H(s)$,&表示按位与运算符;$U_{A(s)}$表示 $A(s)$中 0 的位数;m 表示 $A(s)$的位数.

2.6 超点检测

为了检测超点,首先根据定义 4 计算 $A_i(1\leqslant i\leqslant H)$的超列.由于超列可能包含超点,然后利用超列的任意组合,通过可逆 Sketch 重构节点.为了解决可逆 Sketch 的逆问题,考虑两种情况,具体的过程如下所示.

(1) 简单的情形

假设可逆 Sketch 数据结构中每个位数组 $A_i(1\leqslant i\leqslant H)$只有 1 个超列,分别用 $c_1,c_2,\ldots,c_H$ 表示超列.检测的超点就是通过 hash 函数映射到 $A_i(1\leqslant i\leqslant H)$的超列对应的源地址,问题转化为联立方程组的求解,联立方程组为

$$\begin{cases}h(x)\equiv c_i \bmod (n_i),1\leqslant i\leqslant H\\ g(x)=(ax+b)\bmod p\\ 0\leqslant x<n\end{cases}\tag{2}$$

令 $y=h(x)$,则有:

$$y\equiv c_i \bmod (n_i),1\leqslant i\leqslant H.$$

利用定理 1,联立方程组(2)的解为

$$y \equiv \sum_{i=1}^{H} c_i (m_i (m_i^{-1} \bmod n_i)) (\bmod n) \tag{3}$$

其中,$m_i=n/n_i$, $m_i^{-1} \bmod n_i$ 表示 m_i 对模 n_i 的乘法逆元,且 $m_i m_i^{-1} \equiv 1 (\bmod\ n_i)$. 因为 $y \in \{0,1,\ldots,p-1\}$,联立方程组的解表示为

$$X = \{x \mid x = a^{p-2} \cdot (k \cdot M + \hat{y} - b) \bmod p, 0 \leqslant k \leqslant \lfloor (p-\hat{y})/M \rfloor, 0 \leqslant x < n\} \tag{4}$$

其中, $\hat{y}$ 是联立方程组(2)中小于 M 的唯一解.如果 $M>p$,有至多一个解;否则,有至多 $\lfloor (p-\hat{y})/M \rfloor + 1$ 个解.

(2) 一般的情形

假设可逆 Sketch 数据结构中每个位数组 $A_i (1 \leqslant i \leqslant H)$有 k_i 个超列,令超列的所有组合的集合表示为 $\{D_1, D_2,\ldots,D_X\}$,其中, $X = \sum_{1 \leqslant i < j \leqslant H} \binom{H}{2} k_i k_j$. 检测的超点就是通过 hash 映射到 $A_i (1 \leqslant i \leqslant H)$的超列对应的源地址.对于超列的任意一个可能组合 D_i,利用联立方程组(1)的求解方法,得到一个超点子集 $U_i (1 \leqslant i \leqslant X)$,整个超点集合就是 $U_i (1 \leqslant i \leqslant X)$的并集.

由于超列的错误组合使得基于可逆 Sketch 的超点检测产生误报,为了减少误报,如果节点的链接度小于 ϕF,则丢弃通过可逆 Sketch 重构产生的节点;否则,认为通过可逆 Sketch 重构产生的节点是超点.这样,便提高了超点检测精度.

3 性能分析

本节从存储开销、准确性及计算开销的角度来分析 PDS 方法的性能.PDS 方法的目标是在有限的存储空间内实现实时准确的超点检测,因此,方法的存储开销是首先需要考虑的影响因素.在一定的存储空间下,尽量地提高超点检测的准确性,满足高速网络的特定应用需求.同时,计算开销也是影响算法性能的重要因素.存储开销是指存储链接状态所占用的内存空间.准确性是指超点的链接度估计尽可能地接近其链接度的真实值.计算开销是指处理每个报文所需的 hash 运算和访问内存的次数.

3.1 存储开销

可逆 Sketch 数据结构由 (A^l, B^l)构成,其中,A^l 是由 H 个二维位数组 $A_i^l[j][k]\ (0 \leqslant j \leqslant m-1, 0 \leqslant k \leqslant n_i-1)$组成,$B^l$ 是由 H^* 个二维位数组 $B_i^l[j][k] (0 \leqslant j \leqslant m-1, 0 \leqslant k \leqslant n_i^*-1)$ 组成.因此,PDS 方法所占用的内存空间为

$$(\sum_{i=1}^{H} n_i + \sum_{i=1}^{H^*} n_i^*) m \tag{5}$$

式(5)表明,存储开销与 PDS 方法中的参数有关.在 PDS 方法的设计过程中,需要权衡存储空间和检测精度.

3.2 准确性

检测精度与存储空间是密切相关的,从而需要在检测精度与存储空间之间进行折中.链接度估计是 PDS 方法准确性的重要影响因素,因此,链接度估计精度的大小直接对 PDS 方法的准确性产生不同程度的影响.根据定义 5,节点 s 的链接度估计表示为

$$\hat{n}_s = -m \ln \frac{U_{A(s)}}{m},$$

其中,$A(s)$表示以 s 为源形成的位向量,$U_{A(s)}$表示位向量 $A(s)$中 0 的位数,m 表示位向量 $A(s)$的大小.

从 $\hat{n}_s / n_s$ 的数学期望和标准差分析链接度估计的精度.

定理 2. 如果 $t=n_s/m$ 是一个常数,那么估计量 $\hat{n}_s / n_s$ 是一个渐近无偏估计量,即, $E[\hat{n}_s / n_s] \approx 1$.

证明:令 A_j 表示位向量 $A(s)$中第 j 个位为 0 的事件, 1_{A_j} 表示相应的指示器随机变量,即:如果 A_j 发生,则 1_{A_j} 的取值为 1;否则, 1_{A_j} 的取值为 0.因为不同的源-目的对的 hash 取值是相互独立的,所以,

$$P(A_j) = \left(1 - \frac{1}{m}\right)^n, P(A_j \cap A_k) = \left(1 - \frac{2}{m}\right)^n, j \neq k.$$

因为 $U_{A(s)}$是位向量 $A(s)$中 0 的位数,所以,

$$U_{A(s)}=\sum_{j=1}^{m}1_{A_j}.$$

从而,

$$E[U_{A(s)}]=\sum_{j=1}^{m}P(A_j)=m\left(1-\frac{1}{m}\right)^{n_s}\approx me^{-n_s/m}=me^{-t},n_s,m\to\infty,$$

$$E[U_{A(s)}^2]=E\left[\left(\sum_{j=1}^{m}1_{A_j}\right)^2\right]=m\left(1-\frac{1}{m}\right)^{n_s}+m(m-1)\left(1-\frac{2}{m}\right)^{n_s}.$$

所以,

$$Var[U_{A(s)}]=E[U_{A(s)}^2]-(E[U_{A(s)}])^2\approx m(e^{-t}-e^{-2t}-te^{-2t}),m,n_s\to\infty=me^{-t}(1-(1+t)e^{-t}).$$

令 $V_{A(s)}=U_{A(s)}/m$,则有:

$$E[V_{A(s)}]=e^{-t},Var[V_{A(s)}]=\frac{1}{m}e^{-t}(1-(1+t)e^{-t}).$$

令 $f(V_{A(s)})=-\ln(V_{A(s)})$,则有:

$$\hat{n}_s=m\times f(V_{A(s)}).$$

因为 $f(V_{A(s)})$在 $x_0=e^{-t}$ 处的泰勒级数为

$$f(V_{A(s)})=\sum_{n_s=0}^{\infty}\frac{f^{(n_s)}(x_0)}{n_s!}(V_{A(s)}-x_0)^{n_s},$$

所以,

$$\hat{n}_s=m\left(\sum_{n_s=0}^{\infty}\frac{f^{(n_s)}(x_0)}{n_s!}(V_{A(s)}-x_0)^{n_s}\right).$$

将 $f(V_{A(s)})$的泰勒级数的常数项、一阶导数项、二阶导数项之和作为 n_s 的估计 $\hat{n}_s$,并计算其数学期望,则有:

$$E[\hat{n}_s]=mt+\frac{m}{2x_0^2}E[V_{A(s)}-x_0]^2=n_s+\frac{e^t-t-1}{2}.$$

因此,

$$E[\hat{n}_s/n_s]=1+\frac{e^t-t-1}{2n_s}.$$

当 t 为常数,$n_s\to\infty$时:

$$E[\hat{n}_s/n_s]\approx 1.$$

从而定理 2 得证. □

将 $f(V_{A(s)})$的泰勒级数的常数项和一阶导数项之和作为 n_s 的估计 $\hat{n}_s$,即:

$$\hat{n}_s=m\left(t-\frac{V_{A(s)}-x_0}{x_0}\right).$$

从而,估计量 $\hat{n}_s/n_s$ 的方差为

$$Var[\hat{n}_s/n_s]=\frac{1}{n_s^2}\left(\frac{m^2}{x_0^2}Var[V_{A(s)}-x_0]\right)=\frac{m(e^t-t-1)}{n_s^2}.$$

因此,估计量 $\hat{n}_s/n_s$ 的标准差为

$$Std[\hat{n}_s/n_s]=\frac{\sqrt{m}(e^t-t-1)^{1/2}}{n_s}.$$

3.3 计算开销

除了存储开销和检测准确性外,计算开销也是影响 PDS 方法性能的重要因素.为了简化问题,将处理每个报文所需的 hash 运算和访问内存次数作为计算开销的衡量标准.在 PDS 方法的更新过程中,对到达的每个报文,

需要进行(H+H^*)次行 hash 运算、(H+H^*)次列 hash 运算及(H+H^*)次内存访问(写操作).因此,PDS 方法需要消耗较少的计算资源,有助于其部署在入侵检测系统中进行实时的流量监控.

4 实验评价

本节通过在真实网络流量数据集上进行实验(实验中将网络流量均匀分配到各个线程),对 PDS 的性能进行评估.针对不同的应用,例如端口扫描检测、DDoS 攻击检测、热点检测,深入分析 PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP),PDS(DPort||DIP,SIP),同时与 CSE[5]、文献[2]中第 1 种算法(用 JM 表示)、SVCDS[18]进行比较.

4.1 实验数据及评价标准

实验的硬件平台配置:一个英特尔至强 4 核处理器(E5-2403,1.80GHz)、8GB 内存及 Fedora r14 操作系统,内核版本是 2.6.35.6-45.fc14.i686、共享 10MB L3 缓存,每个内核支持一个硬件线程,因此,整个服务器可提供 4 个硬件线程资源.将上述任务分配与调度策略和该多核处理器相结合,制定相应的任务分配与调度方案.设置 4 个线程,包括 3 个更新线程和 1 个归并线程.将 4 个线程分别绑定到不同的核上,为每个线程建立本地数据结构,这样使得线程之间不需要因共享存储而产生竞争问题.采用数据分解(数据及并行)方式使得每个线程对不同的数据块执行相同的操作.为了使实验数据具有代表性,本节利用 CERNET 的流量数据[28]和 CAIDA 数据集[29],用 Trace1 和 Trace2 分别表示 CERNET 数据集、CAIDA 数据集.实验中,将一个子流分配到相同的线程,对超点检测方法的性能进行评估.CAIDA 数据集采集于 OC48 骨干链路的流量,只包含报文头部信息.CERNET 数据集采集于中国教育和科研计算网的边界路由器,仅包含匿名的报文头部信息.表 2 显示了网络流量的部分统计信息.

Table 2 Statistics of data sets

表 2 数据集的统计信息

	Trace1 (450s)	Trace2 (15s)
报文数	7 485 823	7 282 241
源 IP 数	68 998	438 360
目的 IP 数	72 620	408 365
(SIP,DPort||DIP)的数量	357 836	188 292
(Sport||SIP,DIP)的数量	335 219	175 128

在本文的实验中,利用平均相对误差评价节点链接度,平均相对误差(average relative error)用于衡量节点链接度的测量值与实际值之间的差异程度.令节点链接度实际值为 F,测量值为 $\hat{F}$,则平均相对误差表示为

$$average\ relative\ error = E\left[\frac{|\hat{F}-F|}{F}\right] \tag{6}$$

平均相对误差越小,表明节点链接度越精确.

利用误报率和漏报率评价超点检测精度:误报率(false positive rate)是指被错误识别的超点数与被识别的超点总数的比率;漏报率(false negative rate)是指未被识别的超点数与实际的超点数的比率.令实际的超点集为 A,被识别的超点集为 B.

- 误报率表示为

$$false\ positive\ rate = \frac{|B-A|}{|B|} \tag{7}$$

- 漏报率表示为

$$false\ negative\ rate = \frac{|A-B|}{|A|} \tag{8}$$

误报率、漏报率越小,表明检测精度越高.

4.2 链接度评价

链接度是准确检测超点的前提.利用平均相对误差评价节点链接度.图 3 显示了节点的出度分布情况,横轴

表示节点的出度,纵轴表示节点数,纵坐标取对数.从图 3 可知,节点的出度具有重尾特性,也就是说,出度小的节点占据了绝大部分,出度大的节点占据了极少部分.

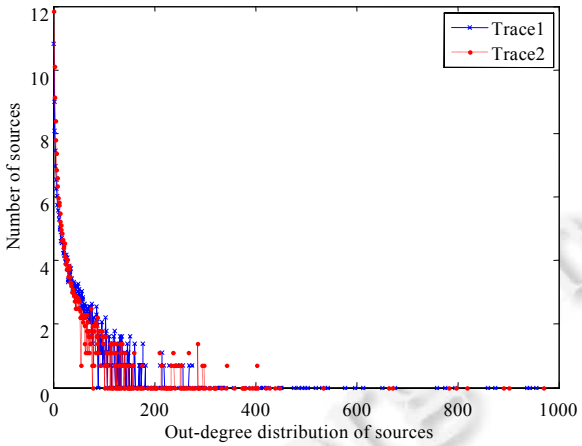
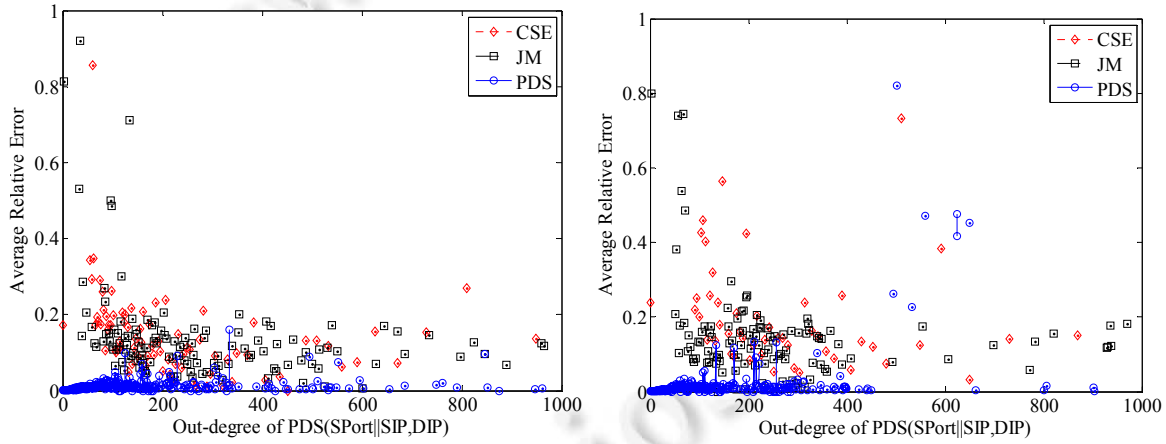


Fig.3 Out-Degree distribution of sources

图 3 源的出度分布

图 4 显示了在参数 H^*,m,n 一定的情况下,CSE,JM,PDS(SPort||SIP,DIP)的出度的平均相对误差分布,参数 H^*,m,n 设置为 H^*=2,n 为接近 32K 的质数,m=4K.从图 4 可知:PDS(SPort||SIP,DIP)的出度估计主要集中在图 4 所示的左下角,其也有极少部分出度估计的平均相对误差比较大,表明总体上 PDS(SPort||SIP,DIP)出度的平均相对误差小于 CSE,JM 方法.因此,说明 PDS 方法适用于教育网、ISP 的节点链接度准确估计.



(a) Trace1 (b) Trace2

Fig.4 Distribution of average relative error of out-degree of PDS(SPort||SIP,DIP)

图 4 PDS(SPort||SIP,DIP)出度的平均相对误差分布

4.3 参数对检测精度的影响

下面通过实验分析并行数据流方法中,参数 H^*,m,n 的不同取值对检测精度的影响.利用误报率和漏报率评价检测精度.利用每流方法得到实际的超点,作为超点检测精度的评价标准,见表 3.

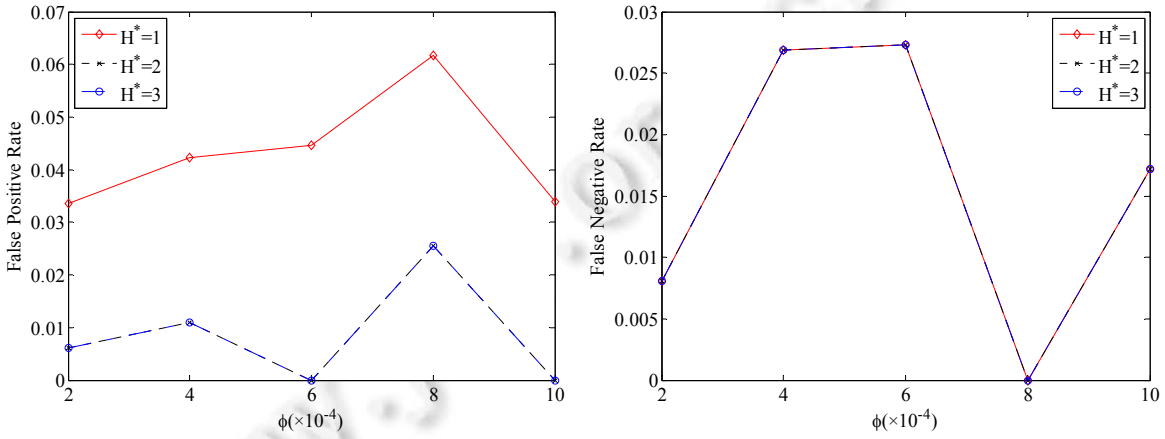
Table 3 Actual numbers of the super points

表 3 实际的超点数

超点	阈值 $\phi(\times 10^{-4})$				
	2	4	6	8	10
SS(SPort||SIP,DIP)的数量	495	186	110	76	58
SS(SIP, DPort||DIP)的数量	594	316	209	142	103
SD(SIP, DPort||DIP)的数量	500	184	115	79	62

4.3.1 参数 H^*

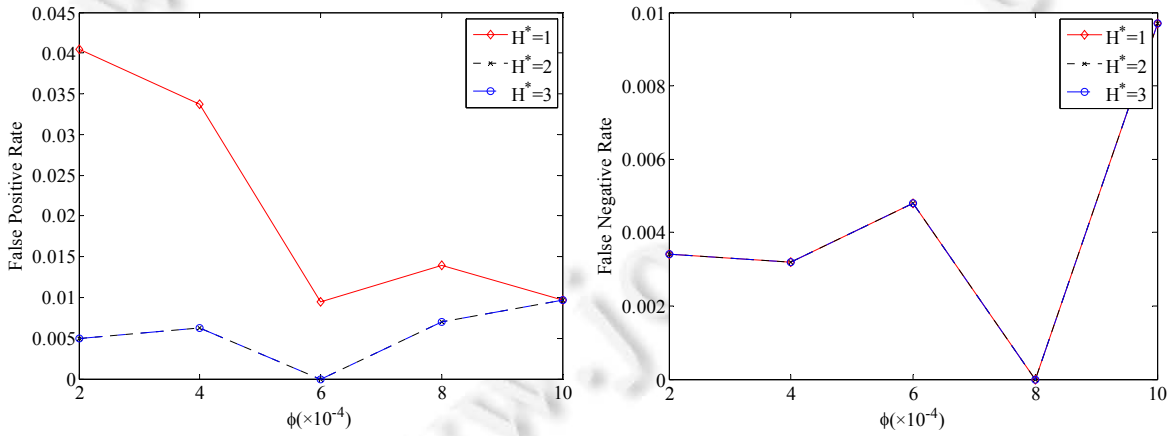
在本文所提出的超点检测方法中,H^*是指可逆 Sketch 数据结构中位数组的数量.下面通过实验研究在其他参数一定的情况下,参数 H^*对超点检测精度的影响,如图 5、图 6 所示.



(a) 误报率 (b) 漏报率

Fig.5 Accuracy of PDS(SPort||SIP,DIP) for different H^*

图 5 参数 H^*对 PDS(SPort||SIP,DIP)的精度的影响



(a) 误报率 (b) 漏报率

Fig.6 Accuracy of PDS(SIP,DPort||DIP) for different H^*

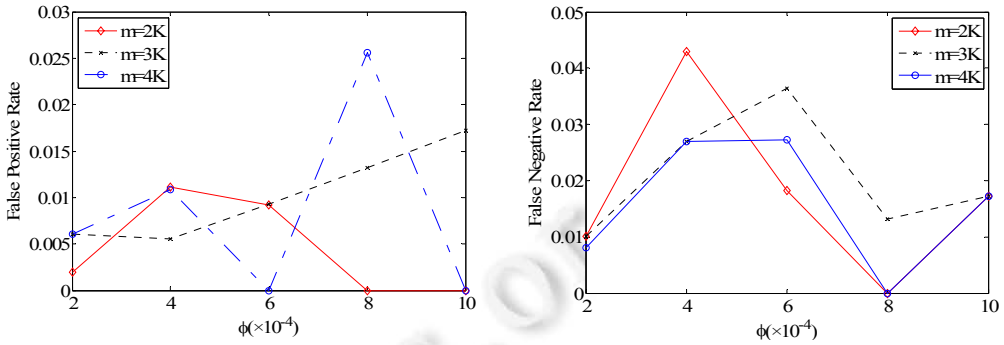
图 6 参数 H^*对 PDS(SIP,DPort||DIP)的精度的影响

图 5、图 6 显示了已知参数 m,n,参数 H^*的不同取值对超点检测精度的影响.参数 m,n 设置为 m=4K,n 为接

近 32K 的质数.从图 5、图 6 可知:随着 H^*的增加,PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP)超点检测的误报率有所下降.当 H^*=2,3 时,PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP)超点检测的误报率接近,说明当 H^*增加到一定值之后,H^*对 PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP)超点检测的误报率影响很小;随着 H^*的增加,PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP)超点检测的漏报率非常接近,说明 H^*的变化对超点检测的漏报率影响较小.

4.3.2 参数 m

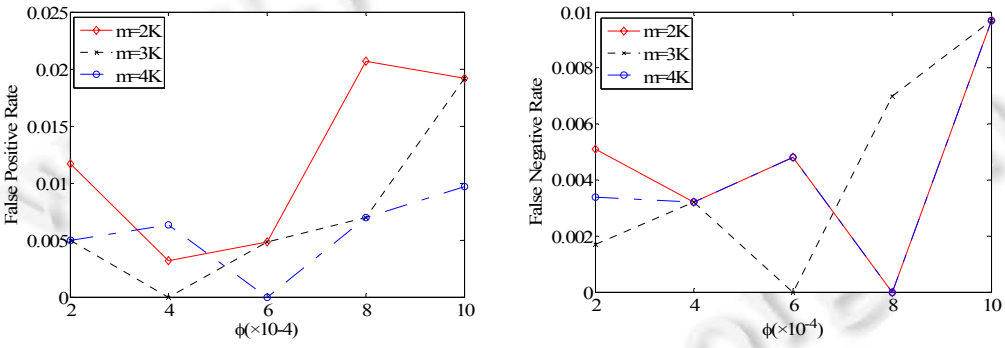
在并行的数据流方法中,m 是指可逆 Sketch 数据结构中位数组的行数.下面通过实验研究在其他参数一定的情况下,参数 m 对超点检测精度的影响,如图 7、图 8 所示.



(a) 误报率 (b) 漏报率

Fig.7 Accuracy of PDS(SPort||SIP,DIP) for different m

图 7 参数 m 对 PDS(SPort||SIP,DIP)的精度的影响



(a) 误报率 (b) 漏报率

Fig.8 Accuracy of PDS(SIP,DPort||DIP) for different m

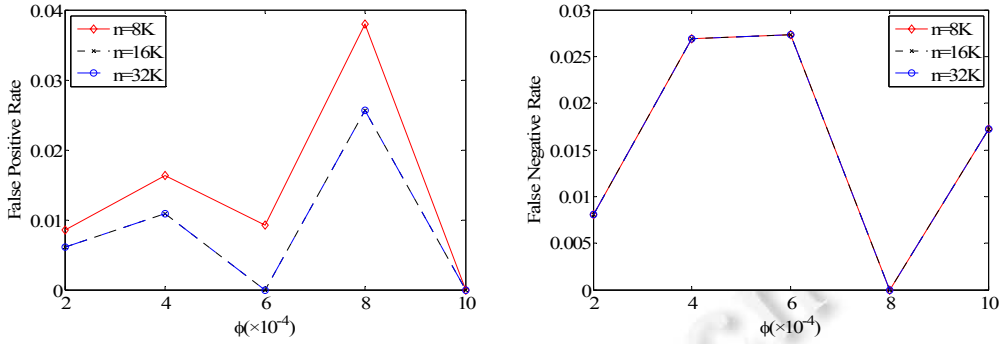
图 8 参数 m 对 PDS(SIP,DPort||DIP)的精度的影响

图 7、图 8 显示了已知参数 H^*,n,参数 m 的不同取值对超点检测精度的影响.参数 H^*,n 设置为 H^*=2,n 为接近 32K 的质数.从图 7、图 8 可知:随着 m 的增加,PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP)的误报率、漏报率发生波动.同时,ϕ 的变化对 PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP)的误报率、漏报率影响较大.

4.3.3 参数 n

在 PDS 方法中,n 是指可逆 Sketch 数据结构中位数组的列数.下面通过实验研究在其他参数一定的情况下,参数 n 对超点检测精度的影响,如图 9、图 10 所示.图 9、图 10 显示了在参数 H^*,m 一定的情况下,参数 n 的不同取值对超点检测精度的影响.参数 H^*,m 设置为 H^*=2,m=4K.从图 9、图 10 可知:随着 n 的增加,PDS(SPort||SIP,DIP)超点检测的误报率有所下降.当 n=2,3 时,超点检测的误报率接近,说明当 n 增加到一定值之后,n 对超点检测的误报率影响很小,而 PDS(SPort||SIP,DIP)超点检测的漏报率几乎没有变化;而随着 n 的增加,PDS(SIP,DPort||

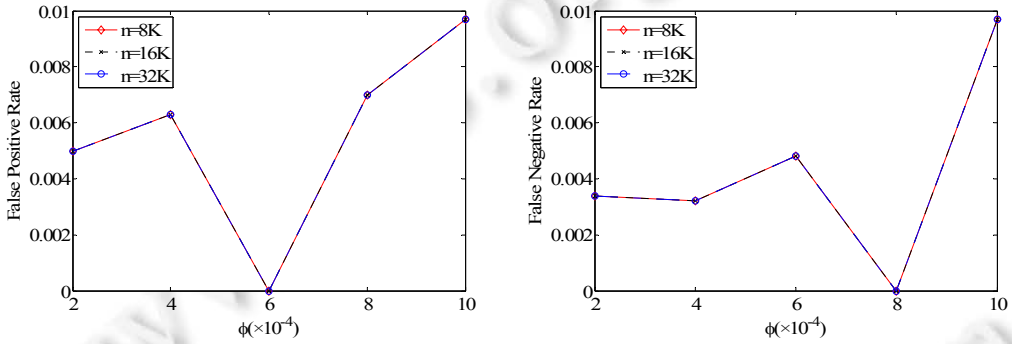
DIP)超点检测的误报率、漏报率几乎没有变化,说明 n 的变化对 PDS(SIP,DPort||DIP)超点检测的精度影响极小.



(a) 误报率 (b) 漏报率

Fig.9 Accuracy of PDS(SPort||SIP,DIP) for different n

图 9 参数 n 对 PDS(SPort||SIP,DIP)的精度的影响



(a) 误报率 (b) 漏报率

Fig.10 Accuracy of PDS(SIP,DPort||DIP) for different n

图 10 参数 n 对 PDS(SIP,DPort||DIP)的精度的影响

4.4 与相关方法的比较

实验中,我们选择 CSE 方法[5]、JM 方法[2]、SVCDS 方法与 PDS 方法进行比较.CSE 方法由存储模块和估计模块构成,存储模块负责将链接存储到虚拟向量中,估计模块负责估计节点的链接度.该方法不仅具有较高的估计精度,而且能够实施有效的在线操作.JM 方法是结合抽样与数据流技术而提出来的,在流抽样之后,对抽样流进行过滤,从而允许在最坏情况下,抽样率非常接近于 hash 表速率与链路速率之比.与基于 hash 的流抽样算法相比,可获得更高的检测精度.SVCDS 方法是结合数据流技术与流抽样而提出来的,提高了检测精度,同时降低了内存复杂度.在一定的存储空间条件下,我们对 4 种方法在检测精度和效率方面进行比较.

4.4.1 检测精度

下面对 3 种方法的检测精度进行比较分析.JM 方法抽样率取 0.5,PDS 的参数配置为 H^*=2,n 为接近 32K 的质数, m=4K.图 11~图 13 显示了 CSE,JM,SVCDS,PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP),PDS(DPort||DIP,SIP)的超点检测精度比较情况.从图 11~图 13 可知:PDS(SPort||SIP,DIP),PDS(SIP,DPort||DIP),PDS(DPort||DIP,SIP)的误报率、漏报率最低;SVCDS 次之;JM 方法较高;CSE 方法最高.

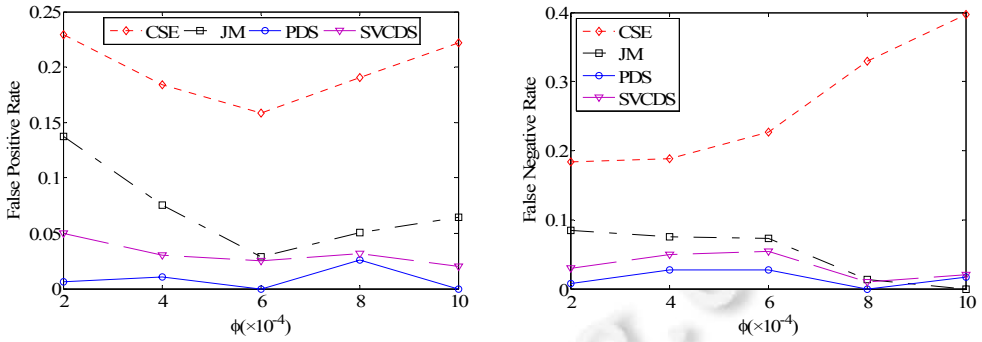
本文方法中,

- 一方面通过并行的可逆 Sketch 数据结构建立紧凑的节点链接度概要,更加准确地估计出节点链接度,

从而可以降低漏报;

- 另一方面,通过另一个 Sketch 数据结构验证因超列的错误组合产生的误报,从而能够有效地降低误报.

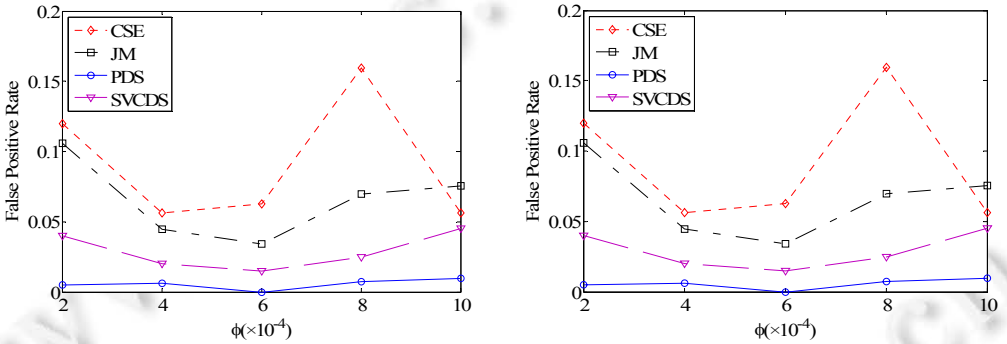
因此,实验结果表明,本文方法具有较高的精度、较低的漏报率和误报率.



(a) 误报率 (b) 漏报率

Fig.11 Comparison of accuracy of CSE, JM, SVCDS, PDS(SPort||SIP,DIP)

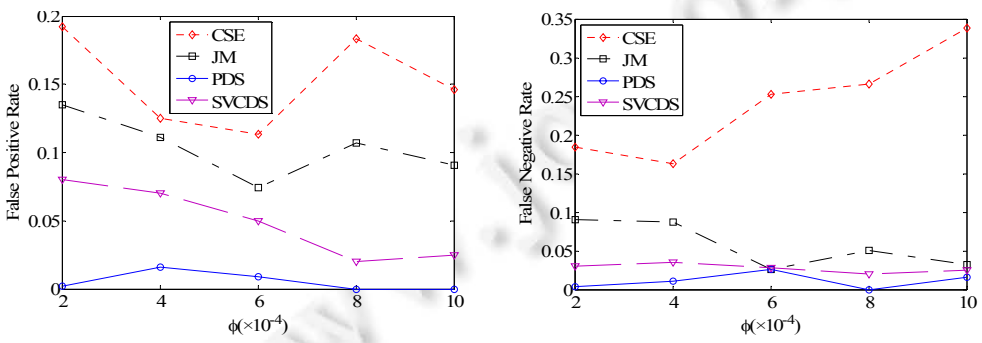
图 11 CSE,JM,SVCDS,PDS(SPort||SIP,DIP)的精度比较



(a) 误报率 (b) 漏报率

Fig.12 Comparison of accuracy of CSE, JM, SVCDS, PDS(SIP,DPort||DIP)

图 12 CSE,JM,SVCDS,PDS(SIP,DPort||DIP)的精度比较



(a) 误报率 (b) 漏报率

Fig.13 Comparison of accuracy of CSE, JM, SVCDS, PDS(DPort||DIP,SIP)

图 13 CSE,JM,SVCDS,PDS(DPort||DIP,SIP)的精度比较

4.4.2 存储开销

为了减少内存开销,CSE,JM,SVCDS 方法均使用位向量标识每个节点的链接信息.因此,节点链接度的准确估计依赖于位向量的大小.虽然 CSE 方法为每个节点建立了一个虚拟位向量,但其占用的内存空间主要由位向量的大小决定.JM 方法利用 hash 表存储每个节点的链接度信息,其占用的内存空间主要由 hash 表的大小决定.SVCDS 方法为每个节点建立了多个虚拟位向量,但其占用的内存空间主要由位向量的大小决定.假设测量的时间区间内总链接度为 N,对于每个不同的链接,PDS 方法更新(H+H^*)个位,则其所需的内存空间为 $N\times(H+H^*)$;SVCDS 方法更新 H 个位,则其所需的内存空间为 $N\times H$;JM 方法查询位向量与更新 hash 表,则其所需的内存空间为 $N+64\times N$;CSE 方法更新 1 位,则其所需的内存空间为 N.假设总链接度为 1M,分析在不同的阈值下,4 种方法的内存开销.为 PDS 方法分配 256KB 内存空间,每个位数组的大小为 128KB(32bits×32K),从而使得该方法能够在不同的阈值下进行有效的检测.已知位向量的大小为 m,仅能准确估计链接度小于 $m\ln m$ 的节点.其原因在于:如果节点链接度的真实值大于 $m\ln m$,位向量中的所有 1 以高概率出现,从而获得的唯一信息就是节点链接度不小于 $m\ln m$.只要阈值小于 $m\ln m$,就认为该节点为超点.当 m=32 时,$m\ln m$=111>70($\phi=2\times10^{-4}$).因此,对于阈值为 70($\phi=2\times10^{-4}$),140($\phi=4\times10^{-4}$),210($\phi=6\times10^{-4}$),280($\phi=8\times10^{-4}$),350($\phi=10\times10^{-4}$),m 为 32,64,64,128,128,CSE 方法中位向量的大小为 128KB,256KB,256KB,512KB,512KB.假设 JM 方法的抽样率为 1,为其分配 64bits×32K 内存空间,m=64,$m\ln m$=266>210.因此,对于阈值为 70($\phi=2\times10^{-4}$),140($\phi=4\times10^{-4}$),210($\phi=6\times10^{-4}$),280($\phi=8\times10^{-4}$),350($\phi=10\times10^{-4}$),m 为 64,64,64,128,128,JM 方法中位向量的大小为 256KB,256KB,256KB,512KB,512KB.JM 方法的抽样率越小,其所需的内存空间也较小.假设 SVCDS 方法的抽样率为 1,对于阈值为 70($\phi=2\times10^{-4}$),140($\phi=4\times10^{-4}$),210($\phi=6\times10^{-4}$),280($\phi=8\times10^{-4}$),350($\phi=10\times10^{-4}$),m 为 32,64,64,128,128,SVCDS 方法中位向量的大小为 128KB,256KB,256KB,512KB,512KB.

4 种方法所需的内存开销见表 4.

Table 4 Comparison of memory overhead of CSE, JM, SVCDS, PDS based super point detection (KB)

表 4 CSE,JM,SVCDS,PDS 的超点检测的内存开销比较 (KB)

超点检测方法	阈值 ϕ($\times10^{-4}$)				
	2	4	6	8	10
CSE	128	256	256	512	512
JM	256	256	256	512	512
SVCDS	128	256	256	512	512
PDS	256	256	256	256	256

由表 4 可知:随着阈值的增加,4 种方法所需的内存空间均呈上升趋势.

4.4.3 时间开销

PDS 方法涉及 hash 运算、内存访问等,这些操作均需消耗一定的时间.为了能够直观地呈现 PDS 方法的时间效率优势.

- 一方面,将 4 种方法的执行时间进行比较.图 14 显示了 CSE,JM,SVCDS,PDS(SIP,DPort||DIP)的超点检测的处理时间.从图 14 可知:随着 ϕ 的增加,PDS(SIP,DPort||DIP)方法的超点检测的执行时间逐渐缩短,而 CSE,JM,SVCDS 方法的超点检测的执行时间只发生微小的变化,当 $\phi=8\times10^{-4}$ 时,PDS(SIP,DPort||DIP)方法的超点检测的执行时间开始低于 CSE,JM,SVCDS 方法,表明 ϕ 的变化影响了超点检测的时间效率;
- 另一方面,通过实验将 PDS 方法与利用单线程实现的超点检测方法进行比较,用 DS 表示单线程实现的超点检测方法.两种方法的时间开销如图 15 所示.由图 15 可知:随着 ϕ 的增加,PDS 方法、DS 方法的时间开销先是显著下降然后再缓慢下降,而 PDS 方法的时间开销始终小于 DS 方法的时间开销,从而表明了 PDS 方法的实时性优势.

本文只给出了 Trace1 的超点检测的性能分析,Trace2 具有相似的结论.

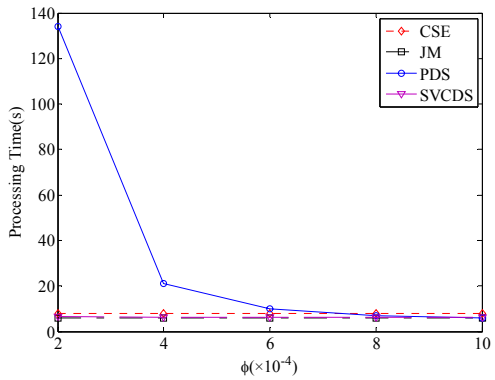


Fig.14 Comparison of processing time of CSE, JM, SVCDS, PDS

图 14 CSE,JM,SVCDS,PDS 的执行时间比较

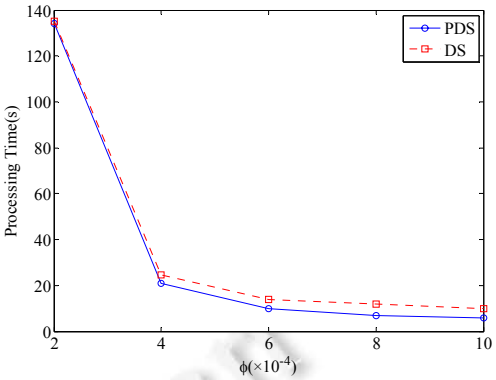


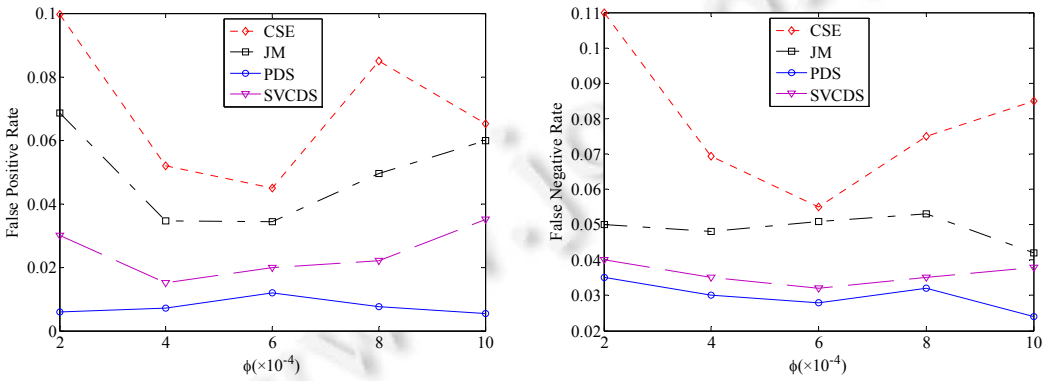
Fig.15 Comparison of processing time of DS, PDS

图 15 DS,PDS 的执行时间比较

4.5 PDS在分布式超点监控中的应用

目前,大部分的超点监控任务是在单个监测点上实施的.实际上,许多网络安全事件(如 DDoS 攻击、蠕虫传播、端口扫描)需要在多个监测点上协作完成检测任务.例如,要在多个网络接入点检测 DDoS 攻击,根据归并后的链接度大小判断遭受攻击的牺牲者.当 DDoS 攻击发生时,由于大量的攻击者可能分布于互联网的不同位置,攻击者所产生的流量通过不同的路径到达目标服务器,此时,依据单个监测点上检测目标服务器的链接度大小可能不足以识别超点,只有归并多个监测点上的链接度才能发现牺牲者.因此,分布式超点监控有助于检测网络安全事件.

在分布式超点监控应用中,每个监测点执行超点检测算法产生计算和存储开销.同时,由于监测点分布在互联网的不同位置,监测点之间需要通信协作来完成检测任务,此过程产生一定的通信开销.为了减少通信开销,只有在测量时间周期结束后才归并各个监控点的链接度信息,将链接度超过总链接度的一定百分比的节点检测为超点. DDoS Attack 2007 数据集[30]是 CAIDA 在 2007 年 8 月 4 日 DDoS 攻击发生过程中采集的流量,流量 trace 的持续时间为 1 小时,大小为 5.3GB(压缩后),只包含向牺牲者的攻击流量和向攻击者的响应流量.利用上述部分 DDoS 攻击流量来测试超点检测方法的性能.通过误报率和漏报率评价超点检测的精度,利用执行时间来评价超点检测的时间效率.如图 16、图 17 所示:对多种超点检测方法进行对比,本文方法在检测精度和时间方面具有较好的性能优势,满足分布式超点监控的应用需求.



(a) 误报率 (b) 漏报率

Fig.16 Comparison of accuracy of super point detection

图 16 超点检测的精度比较

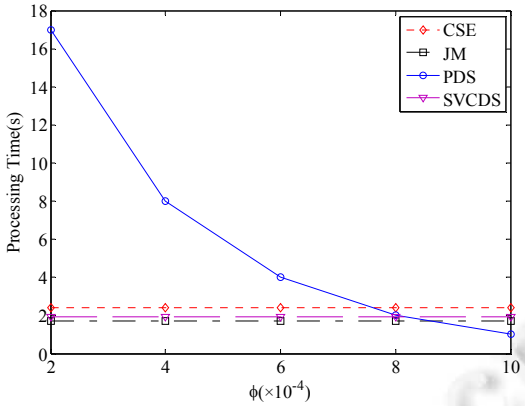


Fig.17 Comparison of processing time of super point detection

图 17 超点检测的执行时间比较

5 结束语

由于基于流抽样的超点检测方法存在计算负荷重、检测精度低、实时性差等问题,在多核处理器计算平台上,本文提出了一种并行数据流方法.本文从理论上分析了 PDS 方法的存储开销、准确性及计算开销,讨论了可逆 Sketch 数据结构中参数对检测精度的影响.通过实验验证了 PDS 方法的有效性,并和相关方法进行了比较.实验结果表明:PDS 方法的链接度估计精度、超点检测精度及处理时间均优于 CSE,JM 方法.同时,PDS 方法占用的存储空间也较小.因此,PDS 方法能够满足高速网络流量监测的应用需求.将 PDS 方法部署到实际网络中的多个测量点,实施在线的分布式的流量监控,以进一步验证 PDS 方法的有效性,这将是下一步的研究方向.

References:

[1] Cheng G, Gong J, Ding W, Wu H, Qiang SQ. Adaptive sampling algorithm for detection of superpoints. Science in China Series (E: Information Sciences), 2008,38(10):1679−1696 (in Chinese with English abstract).

[2] Zhao Q, Xu J, Kumar A. Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation. IEEE Journal on Selected Areas in Communications, 2006,24(10):1840−1852. [doi: 10.1109/JSAC.2006.877139]

[3] Roesch M. Snort: Lightweight intrusion detection for networks. In: Proc. of the LISA. Washington: USENIX Association, 1999. 229−238. http://www.usenix.org

[4] Plonka D. FlowScan: A network traffic flow reporting and visualization tool. In: Proc. of the LISA. USENIX Association, 2000. 305−317. http://www.usenix.org

[5] Yoon MK, Li T, Chen S, Peir JK. Fit a compact spread estimator in small high-speed memory. IEEE/ACM Trans. on Networking, 2011,19(5):1253−1264. [doi: 10.1109/TNET.2010.2080285]

[6] Wu KD. Microprocessor: Multicore has become main stream. 2009 (in Chinese). http://tech.sina.com.cn/h/2009-04-09/1337298708 2.shtml

[7] Sekar V, Reiter MK, Willinger W, Zhang H, Kompella RR, Andersen DG. cSamp: A system for network-wide flow monitoring. In: Proc. of the NSDI. San Francisco: USENIX Association, 2008. 233−246. https://www.usenix.org

[8] Henke C, Schmoll C, Zseby T. Empirical evaluation of hash functions for multipoint measurements. ACM SIGCOMM Computer Communication Review, 2008,38(3):39−50. [doi: 10.1145/1384609.1384614]

[9] Wang P, Guan X, Qin T, Huang Q. A data streaming method for monitoring host connection degrees of high-speed links. IEEE Trans. on Information Forensics and Security, 2011,6(3):1086−1098. [doi: 10.1109/TIFS.2011.2123094]

[10] Whang KY, Vander-Zanden BT, Taylor HM. A linear-time probabilistic counting algorithm for database applications. ACM Trans. on Database Systems, 1990,15(2):208−229. [doi: 10.1145/78922.78925]

[11] Wang HB, Cheng SD, Lin Y. On flow sampling for identifying super-connection hosts in high speed networks. Acta Electronica Sinica, 2008,36(4):809−818 (in Chinese with English abstract).

[12] Venkataraman S, Song D, Gibbons PB, Blum A. New streaming algorithms for fast detection of superspreaders. In: Proc. of the NDSS. San Diego: ISOC, 2005. 1−18. http://repository.cmu.edu/

[13] Estan C, Varghese G, Fisk M. Bitmap algorithms for counting active flows on high speed links. In: Proc. of the 3rd ACM SIGCOMM Conf. on Internet Measurement. New York: ACM Press, 2003. 153−166. [doi: 10.1145/948205.948225]

[14] Kamiyama N, Mori T, Kawahara R. Simple and adaptive identification of superspreaders by flow sampling. In: Proc. of the INFOCOM. Anchorage: IEEE, 2007. 2481−2485. [doi: 10.1109/INFCOM.2007.305]

[15] Cao J, Jin Y, Chen A, Bu T, Zhang Z. Identifying high cardinality internet hosts. In: Proc. of the INFOCOM. Rio de Janeiro: IEEE, 2009. 810−818. [doi: 10.1109/INFCOM.2009.5061990]

[16] Guan X, Wang P, Qin T. A new data streaming method for locating hosts with large connection degree. In: Proc. of the GLOBECOM. Honolulu: IEEE, 2009. 1−6. [doi: 10.1109/GLOCOM.2009.5426280]

[17] Li T, Chen S, Luo W, Zhang M. Scan detection in high-speed networks based on optimal dynamic bit sharing. In: Proc. of the INFOCOM. Shanghai: IEEE, 2011. 3200−3208. [doi: 10.1109/INFCOM.2011.5935169]

[18] Wang P, Guan X, Towsley D, Tao J. Virtual indexing based methods for estimating node connection degrees. Computer Networks, 2012,56(12):2773−2787. [doi: 10.1016/j.comnet.2012.03.025]

[19] Shi X, Chiu DM, Lui J. An online framework for catching top spreaders and scanners. Computer Networks, 2010,54(9):1375−1388. [doi: 10.1016/j.comnet.2009.12.003]

[20] Shin S, Im E, Yoon M. A grand spread estimator using a graphics processing unit. Journal of Parallel and Distributed Computing, 2014,74(2):2039−2047. [doi: 10.1016/j.jpdc.2013.10.007]

[21] Das S, Antony S, Agrawal D, Abbadi AE. CoTS: A scalable framework for parallelizing frequency counting over data streams. In: Proc. of the ICDE. Shanghai: IEEE, 2009. 1323−1326. [doi: 10.1109/ICDE.2009.231]

[22] Das S, Antony S, Agrawal D, Abbadi AE. Thread cooperation in multicore architectures for frequency counting over multiple data streams. In: Proc. of the VLDB. Lyon: ACM Press, 2009. 217−228. [doi: 10.14778/1687627.1687653]

[23] Cafaro M, Tempesta P, Pulimeno M. A paralleld space saving algorithm for frequent items and the Riemann-Hurwitz zeta distribution. Technical Report, CRM-3322, 2012. http://www.crm.umontreal.ca/pub/Rapports/3300-3399/3322.pdf

[24] Cafaro M, Tempesta P. Finding frequent items in parallel. Concurrency and Computation: Practice and Experience, 2011,23(15): 1774−1788. [doi: 10.1002/cpe.1761]

[25] Zhang Y, Fang B, Zhang Y. Parallelizing weighted frequency counting in high-speed network monitoring. Computer Communications, 2011,34(4):536−547. [doi: 10.1016/j.comcom.2010.04.026]

[26] Zhang Y, Sun Y, Zhang J, Xu J, Wu Y. An efficient framework for parallel and continuous frequency item monitoring. Concurrency and Computation: Practice and Experience, 2014,26(18):2856−2879. [doi: 10.1002/cpe.3182]

[27] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 3rd ed., Cambridge: The MIT Press, 2009. 926−954.

[28] Zheng X. IP Trace distribution system. 2011. http://iptas.edu.cn/src/system.php

[29] Claffy KC. The CAIDA UCSD anonymized Internet traces 2012. 2012. http://www.caida.org/passive passive_2012_dataset.xml

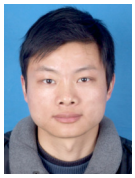
[30] Claffy KC. The CAIDA DDoS attack 2007 dataset. 2007. http://www.caida.org/data/passive/ddos-20070804_dataset.xml

附中文参考文献:

[1] 程光,龚俭,丁伟,吴桦,强士卿.基于自适应抽样的超点检测算法.中国科学(E 辑:信息科学),2008,38(10):1679−1696.

[6] 吴康迪.微处理器:多核已成为主流.2009. http://tech.sina.com.cn/h/2009-04-09/13372987082.shtml

[11] 王洪波,程时端,林宇.高速网络超连接主机检测中的流抽样算法研究.电子学报,2008,36(4):809−818.



周爱平(1982－),男,江苏泰州人,博士生,CCF 学生会员,主要研究领域为网络测量,网络安全.

郭晓军(1983－),男,博士生,讲师,主要研究领域为网络测量,网络安全.

程光(1973－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络测量,网络安全,网络管理.

梁一鑫(1980－2016),男,博士生,讲师,主要研究领域为流量管理,传输控制.