软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(11):2795−2809 [doi: 10.3724/SP.J.1001.2011.03922]

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

基于时延的 Flash Crowd 控制模型*

肖 军[1,2+], 云晓春[1], 张永铮[1]

[1](中国科学院 计算技术研究所,北京 100190)

[2](中国科学院 研究生院,北京 100049)

Flash Crowd Control Model Based on Time Delay

XIAO Jun[1,2+], YUN Xiao-Chun[1], ZHANG Yong-Zheng[1]

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

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: xiaojun@software.ict.ac.cn

Xiao J, Yun XC, Zhang YZ. Flash crowd control model based on time delay. *Journal of Software*, 2011,22(11): 2795−2809. http://www.jos.org.cn/1000-9825/3922.htm

Abstract: An adaptive session-granularity admission control (SGAC) method, which combines the session and request granularities, is proposed for flash crowd control. In SGAC, the average response delay is used to measure, detect, and control the flash crowd. Once a session is allowed to access the server, it will be served until it ends. Besides preventing a server from overloading, SGAC can protect the sessions' integrity. By regulating the session served number adaptively, SGAC can improve the server's utilization. The performance of SGAC with real HTTP log are evaluated, and the result show that SGAC can effectively prevent servers from overloading, protect sessions' integrity, improve server's utilization, reduce the request arrival rate, reduce the access router's computing overhead, and protect valuable transaction sessions.

Key words: flash crowd; session-granularity; overload control; access control

摘 要: 提出了一种 session 级别的 flash crowd 控制策略 SGAC(session-granularity admission control),将 session 控制粒度和 request 控制粒度相结合,采用请求平均返回时延作为检测和控制的依据.对 session 采取一旦接受就完成的策略,在实现对服务器过载控制的同时,保护用户 session 的完整性,并能自动调节新 session 的准入速率,以提高服务器利用率.采用真实 HTTP Log 进行模拟,结果表明,SGAC 方法能够有效控制服务器过载,保护 session 的完整性,提高服务器利用率,降低接入端路由器计算开销,保护有价值的交易 session.

关键词: flash crowd;session 粒度;过载控制;准入控制

中图法分类号: TP393 **文献标识码**: A

Web 服务器作为互联网重要的组成部分,承担着信息发布和在线交易等任务,Web 服务器性能保障一直是研究的重点.Flash crowd 是指大量用户同时访问服务器,导致访问请求超过服务器处理能力,用户提交的请求无法顺利完成,服务器性能大幅度下降甚至崩溃.最近,典型的 flash crowd 事件是 2008 年奥运售票网站由于访问量

* 基金项目: 国家自然科学基金(60703021, 61070185); 国家高技术研究发展计划(863)(2007AA010501, 2009AA01Z431)
收稿时间: 2010-03-31; 修改时间: 2010-06-21; 定稿时间: 2010-07-28

过大而出现无法访问的情况和迈克尔·杰克逊的去世使得 twitter 访问量激增,导致 twitter 和 TMZ 服务器不堪重负[1].flash crowd 往往伴随着自然灾害(如地震)或突发事件(如恐怖袭击)而发生,无法准确其预测发生时间.另一方面,flash crowd 具有突发流量大的特点,发生时,访问请求量往往达到正常情况下的数十倍甚至数百倍.

Flash crowd 发生时,服务器过载,服务器缓冲区被完全消耗,大量请求数据包被丢弃,被接受的请求由于计算资源的限制,需要等待较长的时间才能分配到计算资源,因而请求返回时延较大.用户往往由于数据包丢失或者等待时间过长而放弃交易,从而造成较大损失.

接入端 flash crowd 控制对服务器透明,易于部署,因而得到广泛采用.但当前的接入端 flash crowd 控制策略具有如下不足:1) 控制粒度不合适,request 级别的控制策略忽略了保护 session 的完整性,导致大量的 session 半途而废,而 session 级别的控制策略容易导致服务器负载振荡,造成大量数据包重传;2) 只关注服务器过载控制,没有考虑到对服务器计算能力的充分利用.

鉴于现有接入端 flash crowd 控制策略的不足,本文提出了一种 session 级别的 flash crowd 控制方法(session-granularity admission control,简称 SGAC).其基本思想是:通过测量请求返回数据包平均时延和数据包流量,动态评估待保护服务器的负载情况,并据此建立一套 session 级别的准入控制机制.主要贡献在于:力图通过提出一种 session 控制粒度与 request 控制粒度相结合的 flash crowd 控制策略,控制平均请求返回时延低于初始设定值的同时,实现了对服务器过载控制和对用户 session 完整性的保护,同时实现对计算资源的充分利用.与现有基于控制论的 session 级别控制方法相比,无须建立复杂的输入输出函数,可以避免繁琐的函数系数确定过程,能够避免服务器使用率的震荡;与现有 request 级别的控制方法相比,能够保护 session 的完整性,降低部署 SGAC 的边界路由器的计算开销.采用真实访问日志进行模拟,结果表明,SGAC 方法能够在保护服务器过载的同时,使得数据包的返回时延限制的预期目标范围内,保护 session 的完整性,实现对计算资源的充分利用.SGAC 方法具有较低的计算开销,能够部署在接入端路由器或者网络安全设备(如防火墙)内部.

本文第 1 节介绍基于时延的负载控制模型.第 2 节介绍 flash crowd 的检测方法.第 3 节介绍 flash crowd 的控制方法.第 4 节进行实验分析.第 5 节比较相关工作.第 6 节对一些问题进行讨论.最后对本文进行总结.

1 基于时延的负载控制模型

1.1 时延与负载

动态 Web 服务器通常包含 3 层结构:前端 Web 服务器、中间应用层服务器以及后台数据库服务器.Web 服务器的处理能力由所有计算资源的瓶颈决定,比如,涉及大量数据库查询操作时,后台数据库往往是其瓶颈;执行加解密等大计算开销操作时,应用层服务器 CPU 往往是整体处理能力的瓶颈;而在下载大文件时,网卡往往是处理能力瓶颈.在服务器内部进行 flash crowd 控制,往往通过监测各个计算资源的使用情况,如缓冲区、网卡或 CPU 等,发现计算瓶颈,通过观察瓶颈资源,控制服务器输入,达到过载控制的目的.在瓶颈计算资源过载时,别的计算资源闲置等待,待瓶颈计算资源完成计算后才能继续工作,服务请求需要长时间排队等待.因而,数据包平均返回时延反映了整个 Web 服务器瓶颈计算资源的负载情况.如果返回时延较大,则瓶颈计算资源过载,可以判定服务器过载.

本文将整个 Web 服务器 3 层结构看成一个黑盒,将服务器抽象为如图 1 所示的一个简单结构,其中,CPU 代表了 Web 服务器的瓶颈计算资源.

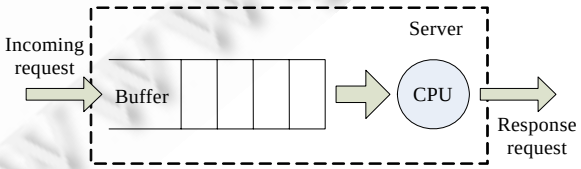


Fig.1 Simple architecture of a Web server

图 1 Web 服务器简化结构

采用模拟考察时延和服务器负载的关系.基于上述简化结构构建模拟 Web 服务器,采用 1998 年足球世界杯部分日志[2]作为访问请求,平均请求返回时延和服务器负载(CPU 使用率)的关系如图 2 所示.可见,数据包返回时延与服务器负载存在正相关关系,时延越大,CPU 的使用率越高,服务器越倾向于过载.

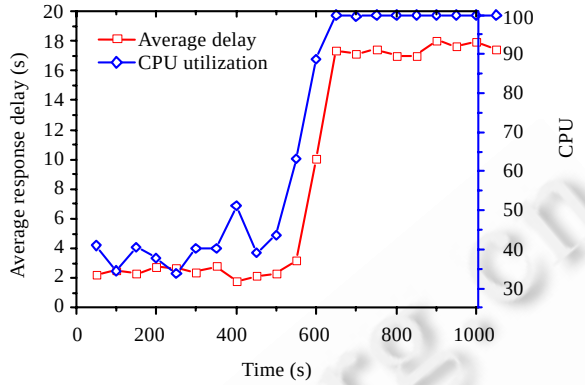


Fig.2 Average requests response delay and the load of the server

图 2 平均请求返回时延与服务器负载

1.2 平均时延

假设进入服务器的数据包和离开服务器的数据包都能通过部署 SGAC 的边界路由器.用 $p(src,dest,seq)$标识一个来自客户端的数据包,其中,src 为源地址,$dest$ 为目的地址,seq 为序列号.定义一个来自客户端的数据包 p 的返回时延 Response Delay $D(p,d)$为其通过边界路由器时间至其第一个返回数据包通过边界路由器时间,d 为时间长度.如图 3 所示,返回时延 $D(p,d)$分为 4 个部分:数据包从边界路由器到服务器的传输时间 t_0、等待处理时间 t_1、处理时间 t_2 以及第 1 个返回数据包从服务器至路由器的传输时间 t_4.

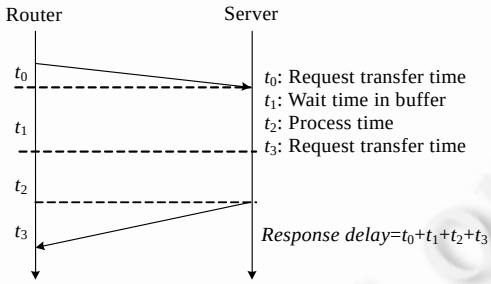


Fig.3 Response delay

图 3 返回时延

定义一段时间内所有返回时延组成的集合为 Φ,Φ的元素个数为$|\Phi|$,定义 Φ的平均返回时延为

$$AD_{\Phi}=\frac{\sum_{D\in\Phi}D(p,d)}{|\Phi|}.$$

显然,AD_{Φ}表示这个时间段内所有时延的平均值.

定义满足某一条件的所有时延组成的集合为 φ,用$|\varphi|$表示集合φ的元素个数,例如,$|\varphi|=10\%\times|\Phi|$.定义集合$\varphi$的平均时延:

$$AD_{\varphi}=\frac{\sum_{D\in\varphi}D(p,d)}{|\varphi|}.$$

1.3 基于时延的负载控制模型

本文提出的 flash crowd 控制方法 SGAC 包含 4 个模块:delay 记录模块(response delay recorder)、flash crowd 检测模块(flash crowd detector)、控制逻辑(control logic)模块以及 session 准入控制模块(session regulator),如图 4 所示.SGAC 通过对数据包时延进行统计分析,检测 flash crowd 是否发生,由 Control Logic 采取相应的控制策略,由 Session Regulator 执行.

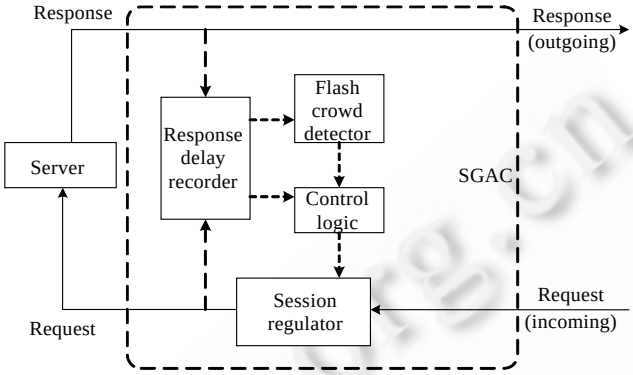


Fig.4 Architecture of SGAC

图 4 SGAC 的结构

SGAC 包括 3 个运行状态,detect,passive mitigation 和 active mitigation 状态,如图 5 所示.在 flash crowd 没有发生时,SGAC 始终运行于 detect 状态,通过数据包时延检测 flash crowd 是否发生.如果发生,则 SGAC 首先转到 passive mitigation 状态.在 passive mitigation 状态,为了避免进一步加重服务器的负担,SGAC 拒绝新的 session 连接请求进入服务器,只允许属于已被服务的 session 的请求通过,直到服务器不在繁忙,再转到 active mitigation 状态.在 active mitigation 中,根据平均时延,通过调节准入新 session 的时间间隔控制服务器过载,从而达到 session 粒度的 flash crowd 控制.

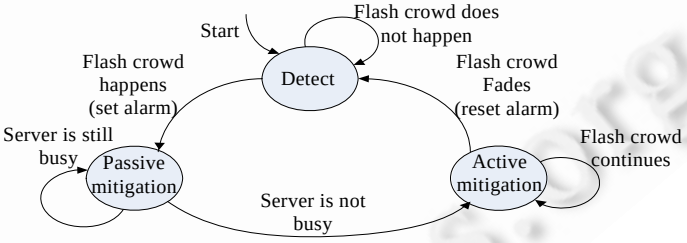


Fig.5 State diagram of SGAC

图 5 SGAC 状态图

2 Flash Crowd 检测

考察如下 flash crowd 检测参数:数据包或者 request 到达速率[3];high-bandwidth connections 的返回速率[4];90%最大时延的平均值;所有时延的平均值.

static rate limit 方法设定一个数据包或者请求通过速率阈值,如果一个时间段内到达服务器的数据包或者请求数目超过了阈值,则认为 flash crowd 发生.此方法易于实施,计算开销低.但由于不同的请求类型对服务器的开销不同,因而阈值并不能准确反映服务器的负载情况;另一方面,不同处理能力的服务器也对应于不同的阈值,因而不易精确确定合适阈值.因此,static rate limit 方法往往无法及时反映服务器的负载情况.

high-bandwidth connections 方法基于 flash crowd 发生时,流量较大的 flow 对拥塞比较敏感,返回流量会大幅度减少这一事实来检测 flash crowd 的发生.本文也考察了这种检测方法,发现其与时延检测相比具有较大的滞后性,并且不能准确判断 flash crowd 是否已经停止,导致对 flash crowd 的控制无法及时解除.

上文已验证了请求平均返回时延和服务器负载的对应关系,本文采用返回时延作为 flash crowd 检测和控制的指标.通常,如果用户等待请求返回时间过长,则用户往往会放弃交易.根据用户对返回时延的容忍程度,设置一个阈值 *DELAY_LIMIT* 来检测 flash crowd 和控制 flash crowd.不同的业务访问对应不同的返回时延容忍度,如浏览页面,容忍度为数秒;如果是视频请求,则时延应更小.本文以页面访问请求进行性能分析,在下文模拟实验中将 *DELAY_LIMIT* 设为 3s.本文分别考察了 10%最小时延$\varphi_{10\%\min}$的平均值、90%最小时延$\varphi_{90\%\min}$的平均值、10%最大时延$\varphi_{10\%\max}$的平均值、90%最大时延$\varphi_{90\%\max}$的平均值以及所有时延Φ的平均值对检测滞后的影响,见表 1.采用 10%最大时延平均值在 flash crowd 发生前产生了误报;而采用 10%最小时延平均值和 90%最小时延平均值作为检测参数,分别滞后 26s 和 11s;以 90%的最大时延的平均值做参数,能最早检测到 flash crowd 发生,滞后了 5s;以所有时延的平均值做参数,滞后 8s 检测到 flash crowd 发生,high-bandwidth connections 则滞后 29s.

Table 1 Detection parameters and detection delay

表 1 检测参数和检测滞后时间

Average 10% minimum response delay	*Detection latency*=26s
Average 90% minimum response delay	*Detection latency*=11s
Average 10% maximum response delay	*Detection latency*=−257s
Average 90% maximum response delay	*Detection latency*=5s
Average response delay	*Detection latency*=8s
Average 10% fast connection response rate (bytes)	*Detection latency*=29s

虽然 90%最大时延能够更前地检测到 flash crowd 发生,但是由于其计算复杂性,本文采取了所有时延平均值作为检测参数.这两种方法的计算复杂性会在后面详细分析.

为了及时检测 flash crowd,本文采用了滑动窗口(sliding-time window,简称 TSW)[5]计算 TSW 时间段的平均时延.在检测 flash crowd 的同时,采取滑动窗口统计进入服务器的请求数和新 session 数.令 R 为一个时间段(detect 阶段为 TSW、passive mitigation 或 active mitigation 阶段为采用的固定时隙)内平均每秒数据包数,S 为上述一个时间段内平均每秒新 session 数目.如果一个 TSW 时间段内的平均时延大于 *DELAY_LIMIT*,则认为发生了 flash crowd,此时,统计出的平均数据包数和平均新 session 数目为 R_f 和 S_f.如下所示,如果条件(1)成立,则认为 flash crowd 发生;如果条件(2)~条件(4)同时满足,则认为 flash crowd 停止,其中,条件(3)、条件(4)是为了减少误报.

(1) $AD_\Phi > DELAY\_LIMIT$;

(2) $AD_\Phi \leqslant DELAY\_LIMIT$;

(3) $R<\alpha\times R_f$;

(4) $S<\alpha\times S_f$.

其中,$0<\alpha<1$.

3 Flash Crowd 控制策略

3.1 Passive mitigation

在 passive mitigation 阶段,本文采取了如下控制策略:

(1) 由于在检测到 flash crowd 发生后服务器已经过载,为避免服务器进一步过载,不允许新的 session 进入服务器,而让服务器处理已建立连接的 session;

(2) 为了避免已建立连接的 session 导致服务器过载,对每秒进入服务器的请求数进行限制,阈值为$\alpha\times R_f$;

(3) 如果在 1s 内到达 SGAC 的数据包数小于$\alpha\times R_f$,并且平均时延小于 *DELAY_LIMIT*,则 passive mitigation 结束,进入 active mitigation 阶段.

Passive mitigation 算法如图 6 所示.

```
        The pseudocode for passive mitigation
reqLimit=α×Rf
WHILE receive a request req
IF time slot over THEN
IF avgDelay≤DELAY_LIMIT AND reqCount<Rf THEN
        go to active mitigation state
   ELSE start new time slot
   END
    END
IF req is response packet THEN
     Allow req to pass
ELSE
  IF reqCount>reqLimit THEN
drop req
  ELSE IF from a new session s THEN
refuse s
          ELSE
            Allow req to pass;
         END
  END
END
END
```

Fig.6 Control algorithm of passive mitigation

图 6 Passive mitigation 控制算法

3.2 Active mitigation

在 active mitigation 状态中,通过调节新准入 session 之间的时间间隔(interval)来达到控制的目的,对属于已被服务的 session 的请求,直接允许其通过.

Active mitigation 共有 4 个状态:wait,under control,fine tune 和 out of control,状态转换如图 7 所示.每一个固定时间单元(time unit)结束后,计算此时间单元内的平均时延和平均数据包速率,依据结果转入对应的状态.详细状态转换过程描述如下:

(1) Wait 状态持续一个时间单元,并且设置此时间单元内允许进入的新 session 数目为 S_f.在时隙结束计算此时间单元内的平均时延.如果时延小于或等于 *DELAY_LIMIT*,则进入 under control 状态,否则进入 out of control 状态.

(2) 如果从 wait 转入 under control,则降低 interval,转入 fine tune 状态;如果从 out of control 状态转入 under control 状态,则保持 interval 不变;如果前一个状态为 under control,并且时间单元内平均时延小于 *DELAY_LIMIT*,平均请求数小于 $\alpha\times R_f(0<\alpha<1)$,则转入 fine tune 状态;如果前一个状态为 under control,并且时间单元内平均时延小于 *DELAY_LIMIT*,平均请求数大于 $\alpha\times R_f(0<\alpha<1)$,则保持 interval 大小不变,仍然处于 under control 状态;如果时间单元内平均时延大于 *DELAY_LIMIT*,则转入 out of control 状态.

(3) fine tune 的目的是为了充分地利用服务器的处理能力.在 fine tune 中,如果在一个时间单元内平均时延小于 *DELAY_LIMIT*,则继续降低 interval,令 R_f 为此时间单元内通过的平均数据包数,继续处于 fine tune 状态,直到平均时延大于 *DELAY_LIMIT*,转入 out of control 状态.

(4) 如果从 fine tune 转入 out of control,则采取回滚的策略,即采用 fine tune 中最后一个时间单元的 interval 值;否则增加 interval 值,直到平均时延小于 *DELAY_LIMIT*,令 R_f 为此时间单元内通过的平均数据包数,转入 under control 状态.

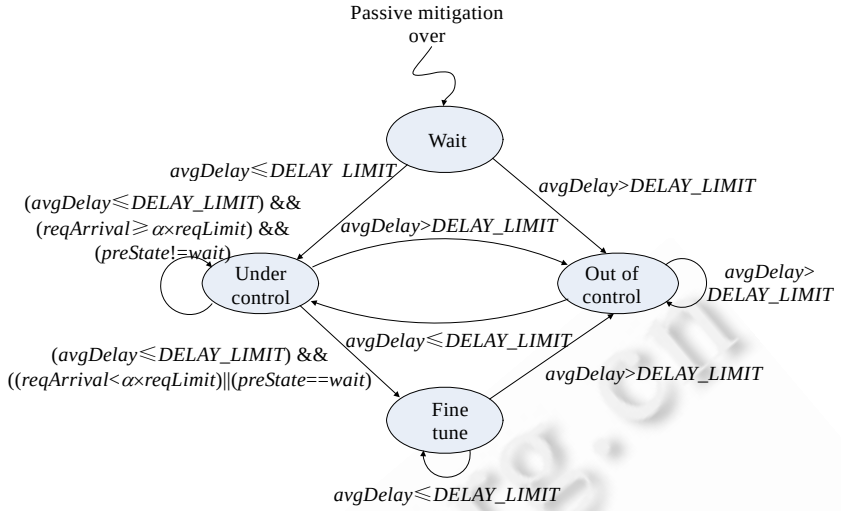


Fig.7 State diagram of active mitigation

图 7 Active mitigation 状态转换图

3.3 计算复杂性分析

本节分析 SGAC 在 flash crowd 检测和控制两个方面的计算复杂性.分析比较以 90%最大时延$\varphi_{90\%max}$的平均值以及所有时延Φ的平均值作为 flash crowd 检测和调整的参数,各自在 detect,passive mitigation 和 active mitigation 这 3 个阶段不同的内存开销和计算时间开销,见表 2.

Table 2 Computational complexity of two parameters in 3 phases

表 2 两种参数在 3 个阶段的计算复杂性

	Average delay of Φ	Average delay of $\varphi_{90\%max}$
Detect	Space complexity: $O(n)$ Time complexity: $O(1)$	Space complexity: $O(n)$ Time complexity: $O(n)$
Passive mitigation	Space complexity: $O(n)$ Time complexity: $O(1)$	Space complexity: $O(n)$ Time complexity: $O(n)$
Active mitigation	Space complexity: $O(n)$ Time complexity: $O(1)$	Space complexity: $O(n)$ Time complexity: $O(n)$

在 detect 阶段,如果采用$\varphi_{90\%max}$的平均值作为检测和调整的参数,假设每秒进入服务器的数据包数为 n,记录所有请求的到达时间和其对应返回包的到达时间的时间复杂性为 $O(1)$.记录滑动窗口大小内的时延,无论全部记录或者采用抽样方式记录[6],空间复杂性均为 $O(n)$.计算$\varphi_{90\%max}$的平均值,n 个数据包对应的计算时间复杂性为 $O(n^2)$.那么,对每个数据包而言,其计算时间复杂性为 $O(n)$.另一方面,如果采用所有时延Φ的平均值作为参数,记录每个数据包的到达时间、返回时间和时延之和的时间复杂性为 $O(1)$,空间复杂性为 $O(n)$.在一个时间单元结束后,计算平均计算时间复杂性为 $O(1)$,所以计算复杂性为 $O(1)$.

在 passive mitigation 阶段,与 detect 状态相比只是采用固定时隙替代滑动窗口,操作过程相同,因此,以$\varphi_{90\%max}$的平均值或Φ的平均值为参数,空间复杂性都是 $O(n)$.区别在于,以$\varphi_{90\%max}$的平均值为参数,时间复杂性为 $O(n)$;而以Φ的平均值为参数,计算复杂性为 $O(1)$.

active mitigation 状态的分析与 passive mitigation 相似,区别在于时间单元长度变大,两者分析类似,以$\varphi_{90\%max}$的平均值或Φ的平均值为参数,空间复杂性都是 $O(n)$;以$\varphi_{90\%max}$的平均值为参数,时间复杂性为 $O(n)$;而以Φ的平均值为参数,计算复杂性为 $O(1)$.

所以,以所有时延Φ的平均值为参数,空间复杂性为 $O(n)$,而计算复杂性为 $O(1)$;以 90%最大时延$\varphi_{90\%max}$的平均值为参数,空间复杂性为 $O(n)$,而计算复杂性为 $O(n)$.由于以 90%最大时延$\varphi_{90\%max}$的平均值为参数的计算复杂

性过高,本文选用所有时延Φ的平均值为参数.虽然检测时间比以 90%最大时延$\varphi_{90\%max}$ 的平均值为参数稍稍滞后,但是实验证明仍然能够在 passive mitigation 状态中迅速而有效地控制 flash crowd.

4 实 验

4.1 实验设置

本文采用 1998 年足球世界杯,巴西与荷兰半决赛之前 5 000s 的真实 HTTP log[2]进行模拟.共有 30 个服务器提供服务,0~1 000s 采用位于 Santa Clara 的 4 个服务器的 log 来模拟,1 000s 后,开始逐渐增加别的服务器的 log 来模拟请求,直至全部 30 个服务器.

为了更接近真实情况,依据地址编号和 request 间隔时间将 log 分成 session 形式,并将 session 进一步划分为多个页面.对一个页面而言,第 1 个 request 为主页面请求(main page request),后面的都是嵌入请求(embedded request).如果请求页面无法满足,则后面的嵌入请求就不会提交.只有在前一个页面的所有请求都满足后,才会进行下一个页面的请求.如果一个 session 的请求都被满足,则此 session 完成.所有 session 开始时间分布如图 8 所示.

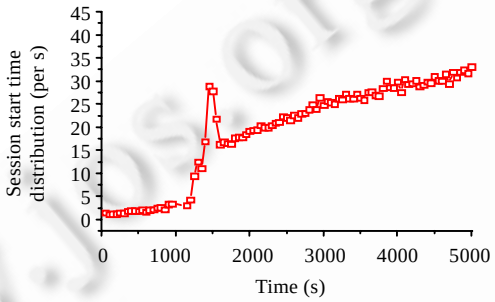


Fig.8 Session start time distribution

图 8 Session 开始时间分布

处理用户请求的服务器结构如图 1 所示,处理能力为 5MB/s,缓冲区最多可容纳 2 000 个请求,如果缓冲区已满,则后面的请求会被丢弃.控制策略部署在与服务器相连的边界路由器上,通过边界路由器计算请求的返回时延.如果一个请求被控制策略许可进入服务器,则边界路由器马上进行转发,否则丢弃.

4.2 性能及比较

本文考察 SGAC 在如下 5 方面的性能:1) session 完成数;2) 完成 session 的平均数据包重传次数;3) 到达部署 SGAC 的路由器的数据包速率;4) session 分布情况;5) 服务器利用率.

本文采用 NEWS 方法[4]与 SGAC 在前 4 个性能指标方面做比较,采用工程中最常用的 PID control 方法[7]与 SGAC 在上述 5 个方面进行比较.令 *DELAY_LIMIT*=3s.

所有时延平均值变化如图 9 所示.在第 1 次平均时延超过 *DELAY_LIMIT* 后进入 passive mitigation 状态,进入 active mitigation 状态后,每个时间单元允许进入服务器的新 session 数目为 S_f,导致服务器过载,平均时延超过 *DELAY_LIMIT*.第 3 次时延超过 *DELAY_LIMIT* 是由于为了提高服务器的利用率,进行 fine tune 调整.

Session 完成情况如图 10 所示.与 NEWS 方法相比,SGAC 能够在 flash crowd 发生时,完成更多的 session,并且完成能力不随 flash crowd 规模的增加而变化;而 NEWS 在 flash crowd 增加到一定规模后,完成 session 的能力会逐渐下降.SGAC 与 PID control 方法在完成 session 的数量方面并无较大差异,区别在于 SGAC 完成 session 的速率相对稳定,而 PID control 完成 session 速率振荡较为明显.

平均重传数据包数如图 11 所示.SGAC 除了在 passive mitigation 状态和在 active mitigation 的 fine tune 状态以及 out of control 状态中数据包平均重传次数比 NEWS 高以外,其他时候均比 NEWS 要低,且平均重传次数比 NEWS 要低.而 PID control 方法导致平均重传数据包震荡.

SGAC 数据包到达速率如图 12 所示.与 NEWS 相比,SGAC 能够有效降低请求到达速率,有助于降低边界路由器计算负担.与 SGAC 方法相比,PID control 方法的数据包速率振荡状态,对边界路由器负担也更重.

服务器利用率如图 13 所示,SGAC 在 fine tune 调整后,在保持平均时延低于 *DELAY_LIMIT* 的同时,服务器的利用率约为 70%;在检测出 flash crowd 发生后,SGAC 下的服务器利用率振荡次数低于 PID control 控制策略下的服务器振荡次数,而 PID control 方法使得服务器的利用率在过载和负载不足之间振荡.SGAC 下服务器的利用率也出现了振荡,原因在于,SGAC 通过对请求数和 session 数进行限制,实现对负载的控制.在批准一个 session 进入服务器时,此 session 的资源消耗无法预测,因此系统负载与请求数以及 session 数之间无法得到精确的对应关系.如 1 000 个读静态页面请求和 1 000 个读数据库请求对系统的资源消耗差别巨大.因此,采用 SGAC 后,由于无法实现对资源消耗的精确控制,服务器仍然会存在负载振荡.在请求或者 session(资源消耗)分布均匀时,SGAC 振荡次数会较少.PID control 方法导致的频繁振荡是由于其方法本身造成,不可避免.

完成 session 包含的请求数分布见表 3.与 NEWS 相比,SGAC 完成包含较多请求的 session 比率较高.通常,交易 session 比浏览 session 包含较多的数据包,因此,SGAC 能够更好地保护更有价值的交易 session.PID control 与 SGAC 均为 session 级别的 admission control 方法,两者完成的 session 分布情况大致相同.

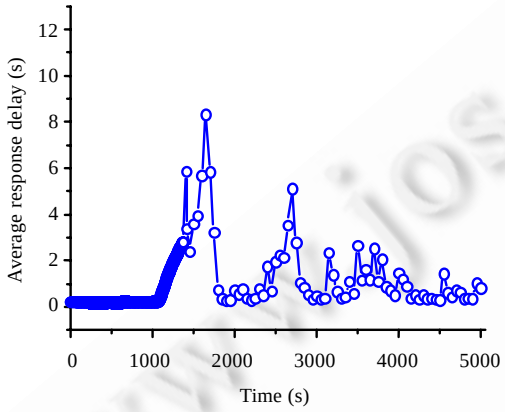


Fig.9 Average delay

图 9 平均时延

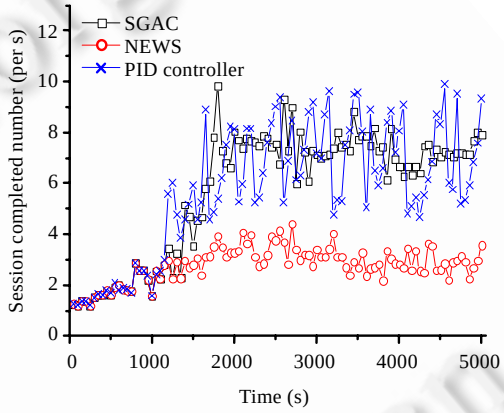


Fig.10 Session number finished under SGAC and NEWES

图 10 SGAC 和 NEWS 的 session 完成情况

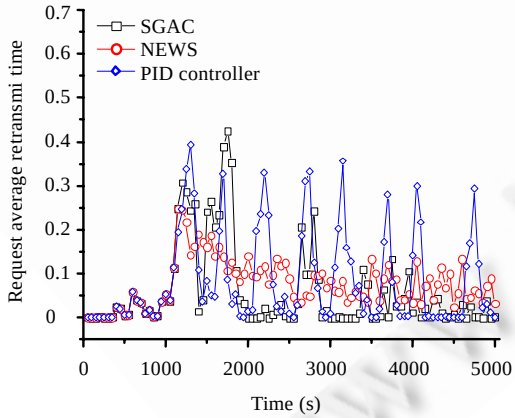


Fig.11 Average packet number retransmitted

图 11 平均重传数据包

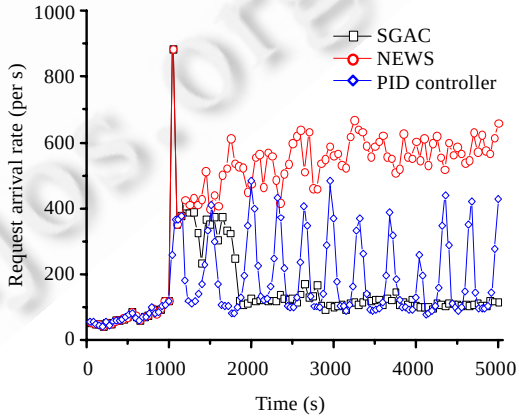


Fig.12 Packet arrival rate under SGAC

图 12 SGAC 数据包到达速率

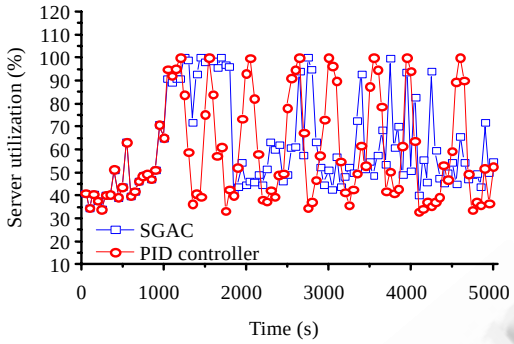


Fig.13 Server utilization

图 13 服务器利用率

Table 3 Session distribution with different packets number

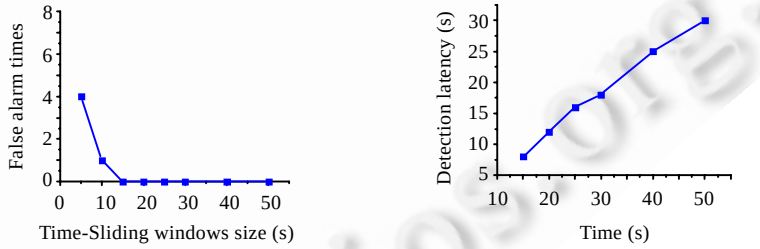
表 3 包含不同数据包数的 session 分布

	SGAC (%)	NEWS (%)	PID control (%)
1≤*Packet number*<5	50.31	72.54	54.68
5<*Packet number*≤10	3.61	5.41	4.15
10<*Packet number*≤20	33.38	8.30	27.64
20<*Packet number*≤50	4.73	4.88	4.58
50<*Packet number*≤100	7.19	7.35	7.60
Packet number>100	0.78	1.52	1.35

4.3 滑动窗口大小对和检查精度和检测滞后的影响

滑动窗口大小的选择应兼顾到检测的准确性和及时性.较小的滑动窗口可以较早检测到攻击,但是容易受到流量波动的影响,突发流量往往导致较小的滑动窗口机制产生误报;较大的滑动窗口可以避免流量波动产生的误报,但检测滞后较大.

如图 14 所示,5s 和 10s 的滑动窗口在 flash crowd 发生之前产生误报,窗口越小,产生的误报次数越多.本文将滑动窗口设置为 15s,因其在不产生误报的同时具有最小的检测滞后.



(a) False alarm times under different TSW (b) Detection latency under different TSW

(a) 不同大小滑动窗口的错报次数 (b) 不同大小滑动窗口的检测滞后时间

Fig.14 Impact of TSW size to the detection accuracy and latency

图 14 滑动窗口大小对检测精度和检测滞后时间的影响

4.4 Time unit大小对session完成情况影响

下面分析 active mitigation 阶段中,时隙(time unit)大小对 session 完成数的影响.Time unit 值过小,能够较快地进行 interval 调节,但对流量波动比较敏感,容易导致不必要的波动;如果 time unit 过大,则对流量波动有较好的忍耐能力,但是调节灵敏度较低.如图 15 所示,在 time unit=50s 时所完成的 session 最多,因而本实验中令 time

unit 为 50s.

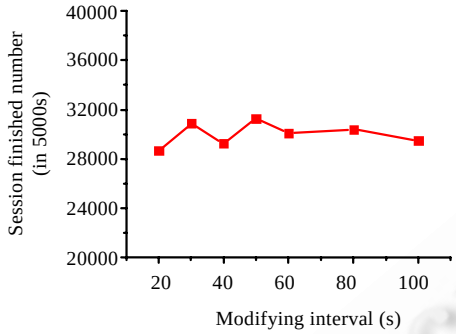
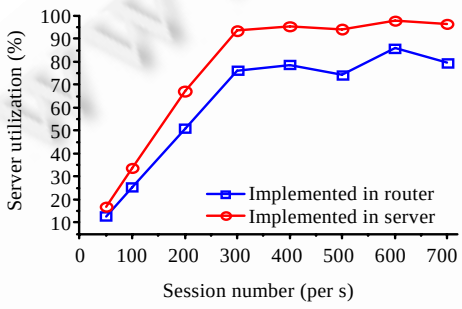


Fig.15 Impact of time unit size to the session number finished

图 15 Time unit 大小对 session 完成情况的影响

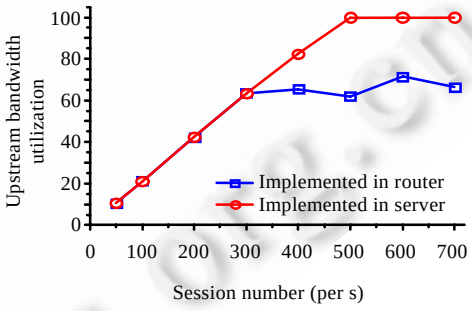
4.5 与部署在服务器上的控制策略比较

采用模拟方法,分析 SGAC 与部署在 Web 服务器上的 PI[8]控制策略,比较两者所保护 Web 服务器的负载和服务器上行带宽的利用率.为便于比较,采用的服务请求和 session 均相同.不同的 session 请求速率下,服务器平均利用率和上行带宽平均利用率如图 16 所示.为了更好地考察部署在 Web 服务器上的控制策略消耗的计算资源,赋予控制操作比请求处理操作更好的优先级,控制操作能够抢占 CPU.与部署在服务器上相比,控制策略部署在边界路由器能够保证服务器的利用率处于合理的范围,同时能够避免上行带宽饱和.部署在服务器上的控制策略无法为上行带宽提供有效保护,同时增加了 CPU 的计算开销,并且所抢占的计算资源的动态变化,使得控制策略往往无法准确调整请求准入速率.在访问请求增加到一定数目后,服务器过载(利用率接近 100%).可见,与部署在服务器上的控制策略相比,控制策略部署在接入端,能够有效节省服务器本身的计算资源.



(a) Server utilization with different implementation position

(a) 部署位置对服务器利用率的影响



(b) Upstream bandwidth utilization with different implementation position

(b) 部署带宽对服务器上行带宽利用率的影响

Fig.16 Impact of implementation position to the server and upstream bandwidth utilization

图 16 不同部署位置对服务器和上行带宽利用率的影响

采用模拟方法,比较 SGAC 与部署在 Web 服务器上的 PI 控制策略的 session 完成情况.采用 1998 年足球世界杯日志作为访问请求,请求 session 数如图 8 所示.比较前 2 500s 内两者完成的 session 数,如图 17 所示.控制策略部署在服务上完成的 session 较少,原因在于控制策略消耗了相当部分的计算能力.且随着访问量的增加,控制策略消耗的服务器的计算能力也相应增加,session 完成速率下降.

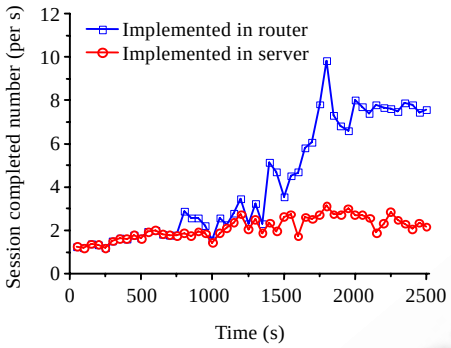


Fig.17 Session finished number under two different implementation positions

图 17 两种部署位置对应 session 的完成情况

5 相关工作

对 flash crowd 检测的研究主要集中于区分 flash crowd 和 DDoS 攻击.Flash crowd 与 DDoS 有两个明显的区别[9].Flash crowd 访问源地址分布相对固定,DDoS 源地址分布比较平均.每个网站的源地址分布均具有各自的特征,DDoS 源地址不易伪装成特定网站的源地址分布形式.另外,虽然两者均表现为流量的大幅度增加,但两者流量变化有较大差异.Flash crowd 的流量呈现逐渐增加趋势,而 DDoS 则在较短时间内(数秒或数十秒)迅速增加到最大值,并保持相对稳定.Flash crowd 发生时,请求返回速率变化较大.正常情况下,请求速率和返回速率间符合幂律分布.根据请求速率预测返回速率,如果实际返回速率与预测值偏离较大,则认为 flash crowd 发生[10].

Flash crowd 的处置方法分为增加处理设备和计算资源、客户端分担和对访问量进行限制 3 类.

Flash crowd 流量规模比平时有很大幅度的增加,如增加服务器来处理 flash crowd,需要较大投资,并且会造成在平时大量设备闲置.除了简单增加处理设备,如服务器以外,采用 Web cache 和 content delivery networks (CDN)[11]是最常见的处理方法.Web cache 和 content delivery network 能够缓存 Web 服务器的内容,在 flash crowd 发生时分担部分访问请求.但是部署 Web cache 和 content delivery network 也需要较大的投资,并且 Web cache 在 flash crowd 发生时需要对热点内容进行缓存后才能发挥作用,并且无法处理涉及数据库操作的 flash crowd 事件(如 2008 年奥运售票网站 flash crowd 事件).文献[12]提出了一种灵活的 CDN 网络结构,在 flash crowd 发生时,依据严重程度动态灵活调整网络结构.FCAN 将 Peer-to-Peer(P2P)与 Client-Server(C/S)模式相结合.检测 flash crowd 程度,在 C/S 模式无法处理时,转到 P2P 模式,cache 间组成 P2P 网络,同时采用 DNS 重定向技术分配请求负载.

客户端分担策略是在 flash crowd 发生时让客户端组成 P2P 网络[13],由已完成服务的客户端为未完成服务的客户端提供请求内容.此方法需要在浏览器中增加新的功能,涉及用户隐私,未必能够得到用户的允许.另外,客户端只能提供只读页面,防火墙也可能阻止 P2P 网络的建立.

在 flash crowd 发生后,接入端路由器通知上游路由器进行流量控制,在骨干路由器上对访问流量进行 pushback 策略[10].此方法能够有效控制服务器的访问量,避免服务器过载,但是需要路由器具有此项功能,同时也会加重骨干路由器的处理负担.并且由于 ISP 无法获益,往往不会提供此项功能.另一方面,pushback 策略不能保持 session 的完整性.

在服务器过载时,吞吐量会急剧下降,传统的性能分析模型无法对这一现象给出合适的解释.文献[14]提出了一个两层队列的服务器模型,根据请求重传、用户超时放弃、资源竞争和服务器过载处理方式,针对 FIFO 和 LIFO 两种队列,详细分析了过载行为.分析结果表明,LIFO 在大部分过载情况下能够有较好的吞吐量.较小比率的请求消耗了大量的计算资源,严重影响了哪些资源消耗教少的请求和系统的吞吐量.与单纯采用准入控制相比,检测和停止此类大资源消耗的请求,能够有效提高系统的服务质量.文献[15]通过检测请求的运行时间和资

源消耗,动态调整选择阈值,停止某些对其他请求无干扰的大资源消耗的服务(如只读请求),大幅度提高了短作业的处理数量和系统吞吐量.为了避免服务器性能在流量突发时大幅度下降,文献[16]提出了一种轻量级的flow 数目估计方法,无须维持每个 flow 的状态,可用于决定一个 flow 是否允许进入系统.通过随机选择缓冲区中的新到底数据包属于一个流的概率,估算当前活跃的 flow.此方法具有较低的计算时间开销和内存开销,借助于控制论中的反馈控制策略[17]决定是否允许新的请求进入,实现了对服务器负载控制.在服务器端内部实现过载控制,往往需要对服务器如 Apache[18]进行修改,无法实现对服务器的透明.另一方面,在服务器内部对过载进行控制,需消耗服务器的处理能力.反馈控制策略需要确定平均响应时间和请求准入速率之间的函数关系,然而对动态 Web 服务器来说,不同的服务请求类型,不同的处理能力使得确定一个长期适用的函数变得比较困难.

NEWS[4]通过对 fast-connection 的返回流量进行测量,检测 flash crowd 的发生,通过对进入服务器的请求数目进行限制,达到flash crowd控制的目的.在接入端路由器进行控制,可以充分利用接入路由器的空闲计算能力,并且实现了对服务器的透明;同时,控制策略往往也比较容易实现.但是,当前研究往往只是基于对服务器过载的控制,简单地采取请求数目限制的目的,而忽略了用户本身,导致大量的 session 都无法完成,半途而废;或处理时间过长,导致用户失去耐心,放弃交易.文献[19]根据服务器的处理能力,采取了 measurement-based 的准入控制,根据请求的执行情况,在线估计请求的资源消耗,同时采取了 shortest-job first(SJF)策略来减少平均请求响应时间,并采取 aging mechanism 来避免长作业饥饿.

6 讨 论

6.1 3种控制方法比较

NEWS 为避免服务器过载,限制每个时间单元进入服务器的请求数目,忽略了保持 session 的完整性,导致大量的 session 仅仅完成了部分 request,后续的 request 因 NEWS 对请求数目的限制而无法完成.大量未完成的request 进行重传,导致接入端路由器接收到大量的数据包.

PID control 采取让返回时延趋近预期返回时延的策略,导致了新 session 准入速率出现震荡,从而导致服务器负载、session 完成速率、数据包重传速率以及到达边界路由器的数据包速率均出现震荡.

本质上,SGAC 在 flash crowd 发生时,通过限制每个时间单元进入服务器的新 session 数,控制服务器过载.采取了一旦服务就完成的策略,使得每个进入的请求都能迅速得到满足.如果一个 session 被 SGAC 拒绝,则只会重传第 1 个 SYN 包,从而重传数据包总数较小,有效降低了路由器的处理负担.

本文通过对时延的控制,达到控制服务器过载的目的.在 passive mitigation 状态中,通过只服务已建立连接的 session 和对允许请求进入速率进行限制的策略,可以在短时间内让服务器从过载中恢复,并能完成已开始服务的session.Fine tune 状态中,通过逐渐缓慢增加每个 time unit 中的新 session 准入数,逐渐提高服务器的利用率,在避免服务器过载和保证服务质量的同时,兼顾到对服务器计算能力的充分利用.从控制粒度上看,本文所提的SGAC 将 session 控制粒度与 request 控制粒度相结合,与 request 粒度控制策略相比,能够实现 session 的完整性;与 session 粒度的控制策略相比,控制粒度更细,对服务器过载的控制更加及时,同时也实现了对服务器计算资源的充分利用.从控制方法上来看,基于返回时延进行准入控制,仍然是反馈控制的方法.但与以往研究工作的不同在于,本文避免了反馈控制方法确定函数系数的繁琐过程,同时也避免了服务器负载的震荡.本文所提的方法实现了对服务器的透明,无须修改操作系统和 Web 服务器.最后,本文所提出的方法具有较低的计算开销,便于实施部署.

6.2 时延作为检测指标的准确性分析

依据服务器的使用率来判断是否发生了 flash crowd 是最为准确、可靠的方法,在服务器上实现的 flash crowd 控制方法大都采用服务器利用率作为 flash crowd 检测和控制的参数.本文是在服务器外实现对 flash crowd 的检测和控制,由于对服务器透明的要求,无法从服务器获得其利用率信息,只能通过时延来推测服务器的使用率.如本文第 1.1 节所示,模拟实验结果表明,时延与服务器利用率存在正相关关系.此外,我们在过载控制

系统项目开发时,在真实网络环境中也验证了时延与服务器利用率的正相关关系.另外,后续的模拟实验结果也表明,当平均返回时延较大时,服务器的利用率相应较大.因此,时延可作为 flash crowd 检测的参数.如上所述,*DELAY_LIMIT* 依据用户对时延的容忍度而定.在平均时延达到 *DELAY_LIMIT* 时,服务器的利用率已较高(如第 1.1 节和后续的模拟实验所示),需对服务器进行过载控制.由于时延体现了服务器利用率,而在平均返回时延达到或超过 *DELAY_ LIMIT* 时,服务器的利用率已较高,因此,时延与 *DELAY_LIMIT* 结合,可用于判断 flash crowd.

7 结 论

当前,request 粒度的 flash crowd 控制策略关注对服务器的拥塞控制,而忽略了保护用户 session 的完整性,基于反馈控制方法 session 粒度的控制策略则会导致服务器负载频繁震荡.本文提出一种 session 级别与 request 级别相结合的 flash crowd 控制策略 SGAC,以平均返回时延为检测和调整的参数,在 detect 状态中实现了对 flash crowd 的检测;在 passive mitigation 状态实现了服务器过载的迅速处置;在 active mitigation 状态通过控制新 session 的准入速率,实现了对服务器过载的控制和用户 session 完整性的保护,同时兼顾了计算能力的充分利用.采用真实 HTTP log 进行模拟,结果表明,本文所提出的 SGAC 方法能够及时检测 flash crowd 的发生,控制请求数据包的平均返回时延低于设定值,使得用户对服务具有较好的满意度,在对服务器过载控制的同时,有效保护 session 的完整性和保护有价值的交易 session,提高服务器的利用率,有效降低数据包到达边界路由器的速率,有效降低边界路由器的计算开销.与部署在服务器上的控制策略相比,部署在接入端路由器的控制策略避免了对服务器计算能力的大量消耗.SGAC 方法的时间复杂性为 $O(1)$,空间复杂性为 $O(n)$,可部署在接入端路由器或防火墙等网络安全设备中.

References:

[1] http://www.matchtt.com/dp-bbsthread-369328.html

[2] http://ita.ee.lbl.gov/html/contrib/WorldCup.html

[3] Barford P, Plonka D. Characteristics of network traffic flow anomalies. In: Proc. of the 1st ACM SIGCOMM Workshop on Internet Measurement. San Francisco, 2001. 69−73. [doi: 10.1145/505202.505211]

[4] Chen X, Heidemann J. Flash crowd mitigation via adaptive admission control based on application-level observations. ACM Trans. on Internet Technology, 2005,5(3):532−569. [doi: 10.1145/1084772.1084776]

[5] Clark DD, Fang WJ. Explicit allocation of best-effort packet delivery service. ACM/IEEE Trans. on Networking, 1998,6(4): 362−373. [doi: 10.1109/90.720870]

[6] Chen F, Lambert D, Pinheiro JC. Incremental quantile estimation for massive tracking. In: Proc. of the 6th ACM KDD Int'l Conf. in Knowledge Discovery and Data Mining. Boston, 2000. 516−522. [doi: 10.1145/347090.347195]

[7] Franklin GF, Powell JD, Emami-Naeini A. Feedback Control of Dynamic Systems. 6th ed., Prentice Hall, 2010. 179−200.

[8] Kamra A, Misra V, Nahum EM. Yaksha: A self-tuning controller for managing the performance of 3-tiered Web sites. In: Proc. of the Internet Workshop of Quality of Service. Montreal, 2004. 47−56. [doi: 10.1109/IWQOS.2004.1309356]

[9] Le Q, Zhanikeev M, Tanaka Y. Methods of distinguishing flash crowds from spoofed dos attacks. In: Proc. of the 3th EURO-NGI Conf. on Next Generation Internet Design and Engineering. Trondheim, 2007. 167−173. [doi: 10.1109/NGI.2007.371212]

[10] Xie LL, Smith P, Hutchison D, Banfield M, Leopold H, Jabbar A, Sterbenz JPG. From detection to remediation: A self-organized system for addressing flash crowd problems. In: Proc. of the IEEE Int'l Conf. on Communications. Beijing, 2008. 5809−5814.

[11] http://www.akamai.cn

[12] Pan CY, Atajanov M, Hossain MB, Shimokavwa T, Yoshida N. FCAN: Flash crowds alleviation network. In: Proc. of the 21st Annual ACM Symp. of Applied Computing. Dijon, 2006. 759−765. [doi: 10.1145/1141277.1141452]

[13] Deshpande M, Amit A, Chang M, Vankatasubramanian N, Mehrotra S. Flashback: A peer-to-peer Web server for flash crowds. In: Proc. of the 27th Internet Conf. on Distributed Computing Systems. Toronto, 2007. [doi: 10.1109/ICDCS.2007.112]

[14] Mathur V, Apte V. An overhead and resource contention aware analytical model for overloaded Web servers. In: Proc. of the 6th Int'l Workshop on Software and Performance. Buenes Aires, 2007. 39−55. [doi: 10.1145/1216993.1216999]

[15] Zhou JY, Yang T. Selective early request termination for busy Internet services. In: Proc. of the 15th Int'l World Wide Web. Edinburgh, 2006. 605−614. [doi: 10.1145/1135777.1135866]

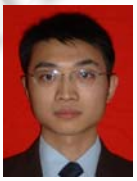
[16] Prasad RS, Thottan M, Lakshman TV. A stateless and light-weight bandwidth management mechanism for elastic traffic. In: Proc. of the 27th IEEE Int'l Conf. on Computer and Communications (Infocom). Phoenix, 2008. 2153−2161. [doi: 10.1109/INFOCOM.2008.207]

[17] Lu C, Lu Y, Abdelzaher TF, Stankovic JA, Son SH. Feedback control architecture and design methodology for service delay guarantees in Web servers. IEEE Trans. on Parallel and Distributed Systems, 2006,17(9):1014−1027. [doi: 10.1109/TPDS.2006.123]

[18] Wei JB, Zhou XB, Xu CZ. Robust processing rate allocation for proportional slowdown differentiation on Internet servers. IEEE Trans. on Computer, 2005,54(8):964−977. [doi: 10.1109/TC.2005.135]

[19] Elnikety S, Nahum E, Tracey J, Zwaenepoel W. A method for transparent admission control and request scheduling in E-commerce Web sites. In: Proc. of the 13th Int'l World Wide Web. New York, 2004. 276−286. [doi: 10.1145/988672.988710]

肖军(1979－),男,江苏大丰人,博士,助理研究员,CCF 会员,主要研究领域为 DDoS 攻击检测和过滤,拥塞控制.



张永铮(1978－),男,博士,副教授,CCF 会员,主要研究领域为网络安全.

云晓春(1971－),男,博士,教授,博士生导师,主要研究领域为网络安全,互联网建模.