# PaSeM：并行无冲突的网络流量会话管理

张建宇[1) ] 周 渊[2)] 邹 维[1)]

[1)](北京大学计算机科学技术研究所 北京 100871)

[2)](国家计算机网络应急技术处理协调中心 北京 100029)

**摘 要** 网络会话管理是网络流量监控、状态防火墙、入侵防御、网络地址转换、负载分流等网络在线业务的关键共性技术，对于准确、快速、灵活地跟踪、分析和处置网络流量中的协议交互过程、端对端行为和通信内容起着基础性支撑作用. 近年来，随着 P2P(Peer-to-Peer)、VoIP(Voice over IP)、网络流媒体等新兴应用的快速发展，网络流量和会话呈现爆炸式增长，如何实现高效的会话管理成为人们面临的一项挑战. 文中提出了一种适用于并行执行环境的网络会话管理方案 PaSeM(Parallel Session Management)，采用基于散列表的无锁会话表设计和多种并行策略，讨论并解决了在高速网络环境下面临的各种并行冲突问题，给出了会话表查询和动态管理的高效并行算法，实现了对报文和会话的并行无冲突的高效处理. 基于 $G/G_2/n_1$ 排队模型和空竭服务多重休假 M/G+D/1 排队模型对 PaSeM 的性能进行了理论分析，对于稳态下并行处理单元(PE)数量、任务队列长度、存储开销与报文到达速率、会话到达速率之间的关系以及其它关键参数应满足的条件给出了定量计算方法. 最后，采用基于 IXP2400 网络处理器的硬件平台进行了原型开发和实验. 实验结果表明，PaSeM 对于会话管理和报文处理具有较好的并行加速效果，理论计算值与实验值能较好地吻合，报文处理的并行效率均值接近 1，当会话管理单元个数为 4 时，会话处理并行效率为 65.4%(亦即加速比为 2.62)，当会话管理单元个数为 8 时，会话处理并行效率仍然达到了 48.3%(加速比为 3.86)，能够满足当前高速网络环境流量处理的性能要求；在最大吞吐量负载下队列长度及其变化幅度都处于合理范围，会话表垃圾比率维持在较低的水平上(实验结果为小于 9%)，与已有的工作相比为优.

**关键词** 网络流量；会话管理；会话表；并行处理；并行冲突

**中图法分类号** TP301 **DOI 号**：10.3724/SP.J.1016.2010.01195

# PaSeM: Parallel and Conflict-Free Network Traffic Session Management

ZHANG Jian-Yu[1)] ZHOU Yuan[2)] ZOU Wei[1)]

[1)](*Institute of Computer Science and Technology, Peking University, Beijing* 100871)

[2)](*National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing* 100029)

**Abstract** As a mutual and essential technique in many online network operations, such as traffic monitoring, stateful firewall, intrusion prevention, network address translation (NAT), load balancing, etc., network traffic session management serves as a basic functionality to track, analyze and process protocol interactions, endpoint behaviors and communication content. In recent years, with the P2P(Peer-to-Peer), VoIP(Voice over IP), streaming media and other new applications having been springing up everywhere, network traffic and sessions have led to an explosive growth, so how to achieve efficient session management become a challenge. In this paper, an approach of parallel network session management, PaSeM(Parallel Session Management), is proposed, which introduces a lock-free session hash table structure, several parallel processing schemes, and cost-effective parallel lookup and management algorithms to achieve high speed and

收稿日期：2010-01-06；最终修改稿收到日期：2010-05-31. 本课题得到国家“八六三”高技术研究发展计划项目基金(2006AA01Z410)、国家自然科学基金(60873217，60973159)、国家发改委信息安全专项([2009]1717)、下一代互联网业务试商用及设备产业化专项(CNGI-09-01-12)、电子信息产业发展基金([2007]329)资助. **张建宇**，男，1977 年生，博士研究生，主要研究方向为网络与信息安全. E-mail：zhangjianyu@icst.pku.edu.cn. **周 渊**，男，1972 年生，博士，高级工程师，主要研究方向为信息安全. **邹 维**，男，1964 年生，博士，研究员，博士生导师，主要研究领域为网络与信息安全.

conflict-free processing of large numbers of packets and sessions. Based on the queuing model of $G/G_2/n_1$ and M/G+D/1 with exhaustive service and multiple vacations, the performance is analyzed and evaluated, and the quantitative method for the relationship between packet and session arrival rate and parallel processing unit (PE) number, the task queue length, storage overhead, condition values of other key parameters in the steady sate. Experiments show that PaSeM has high parallel efficiency, and the theoretical and experimental values agree well. The performance requirement of high-speed network can be met; the average parallel packet processing efficiency is close to 1; when the number of session management unit is 4, the parallel efficiency of session processing is 65.4% (that is, speedup 2.62); when the number of session management unit become 8, the parallel efficiency of session processing is 48.3% (that is, speedup 3.86). The queue length and its skewness are in reasonable during peak throughput. Garbage ratio of the session table is in a relatively low level (experimental result is less than 9%).

**Keywords** network traffic; session management; session table; parallel processing; parallel conflict

## 1 引 言

随着 P2P(Peer-to-Peer)、VoIP(Voice over IP)、流媒体、移动数据业务等网络应用日新月异的发展以及网络带宽的迅猛增长,网络基础设施和网络系统面临许多新的挑战.例如,网络流量监控、入侵检测和防御、状态防火墙、网络地址转换(NAT)、负载分流等网络在线业务均需要围绕网络会话实现对流量的实时、复杂的管理和控制[1-3],高速网络环境中网络会话往往达到几百万甚至上千万规模,其核心是实现高效的会话管理.

一段时间区间内在两个通信端点(endpoints)之间产生的连续报文称为一条"会话(session)".会话中的报文含有相同的关键码,如源 IP 地址、目的 IP 地址、源 TCP/UDP 端口、目的 TCP/UDP 端口和 IP 协议号.如图 1 所示,从时间轴上观察,每条会话的生命周期都包括开始、结束和老化等几个关键点[4-5].会话管理对每条会话的生命周期和状态变化进行跟踪,在会话表(session table)中记录并及时更新每条会话的信息,包括时间戳、会话状态以及各种业务模块所需的参数,并针对流经的每个报文查询会话表,找到报文所属会话记录.业务模块根据这些信息对会话和报文实施各种处理操作,实现对流量的细粒度、精确及基于状态和时间的管理控制[1-2,5].当会话结束后,则要及时删除回收会话表中过期的会话记录.

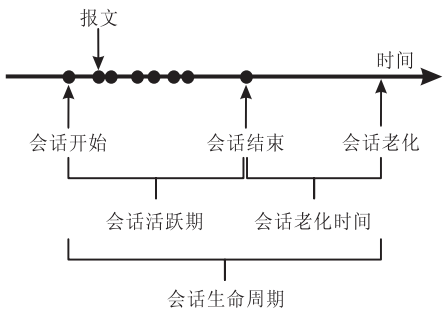



图 1 会话的生命周期

与常见的单向(unidirectional)"网络流(flow)"[4]不同,网络会话是双向(bidirectional)的,会话的第一个报文传输的方向一般定义为会话的"正向",反之称为"反向",即一条"会话"包含正向和反向两条"流".两者的应用领域和应用场景有较大的差别.前者广泛应用于网络测量领域[6-9],所涉及的测量任务往往比较简单,需要记录的流信息较少且对于少量的分类和查询错误不敏感,因而网络流管理的研究重点在于:流表结构的设计,即如何将信息压缩保存到容量小、速度快的存储器件中,以降低 I/O 访存开销、提高性能以及依托于这些流表结构的高效的查询算法;报文采样[6,10]和流采样[7,9]技术,研究如何在减轻负载的同时保证测量的精度.后者则主要应用于许多复杂的在线业务,特别是一些网络安全业务[2-3,11-14],以实现对协议交互过程、端对端行为和通信内容的准确、快速、灵活的跟踪、分析和处置.例如,状态防火墙需要通过检查 TCP 连接三次握手交互过程的完整性来抵御 SYN 洪泛攻击[1];入侵检

测系统需要通过将会话正反两方向的状态信息和检测结果进行关联分析来避免误检[3,15]；网络地址转换(NAT)在进行 IP 和 TCP/UDP 端口资源的动态分配时，也必须以会话为粒度，否则无法保持端对端通信的正确性.这些业务与上述基于流的网络测量任务不同，通常需要记录比较多的会话属性信息和状态信息，而且不允许出现会话记录和信息的缺失，因而会话表往往需要占据较大的存储空间.另一方面，这些业务对于流经的每条会话和每个报文都要进行检测和处理，不允许出现遗漏和错误[16].在这种情况下，随着会话和报文规模的增加，会话表添加、删除和查询的效率往往迅速下降，成为系统主要的性能瓶颈.因此，如何实现高效的会话管理成为一项非常具有现实意义的研究课题，其重点在于会话表结构以及会话表的动态管理机制和查询算法[5,17-18].

针对大规模网络流量会话管理问题，本文提出了一种适用于并行执行环境的网络会话管理方案 PaSeM(Parallel Session Management)，采用基于散列表的无锁会话表设计，讨论并解决了在高速网络环境下面临的各种并行冲突问题，给出了会话表查询和动态管理的高效并行算法.基于 $G/G_2/n_1$ 排队模型和空竭服务多重休假 M/G+D/1 排队模型对 PaSeM 的性能进行了理论分析，对于稳态下并行处理单元(PE)数量、任务队列长度、存储开销与报文到达速率、会话到达速率之间的关系以及其它关键参数应满足的条件给出了定量计算方法.最后，采用基于 IXP2400 网络处理器的硬件平台进行了原型开发和实验.PaSeM 在保证业务灵活性的前提下实现了对会话和报文的高速无冲突并行处理.

## 2　相关工作

对于诸如网络流计数、流报文计数[19-20]等简单的网络流量测量任务来说，精确跟踪和记录每条流的状态是没有必要的，因此人们常采用 bitmap 或 Bloom filter[21]来压缩记录流信息，并将其置于 SRAM 等高速存储器件中，从而以较小的存储空间开销和 I/O 访存开销来获得处理效率的提升.例如，Estan 等人提出了一组基于 bitmap 的流计数算法，以满足网络测量、拒绝服务攻击检测、端口扫描检测和调度等不同应用的需要[19]；Kumar 和 Xu 等人提出了带多组散列函数的 SCBF(Space-Code Bloom Filter)并采用多个具有不同采样概率的 SCBF 对报文进行计数，再基于最大似然估计和平均值估计两种方法对流大小(flow size)进行估算[20].为了实现流状态跟踪，Bonomi 等人提出了一种改进的计数型 Bloom filter，对少量流状态信息进行压缩存储和模糊查询，每个 Bloom filter 单元除了计数器外还包含一个流状态字段(一般少于 20bits)，并增加了一个 DK(Don't Know)状态作为产生碰撞时的查询结果[22].Chang 等人则通过扩展 Bloom filter 单元来缓存报文分类或路由查找的结果(如下一跳的输出设备接口位图)，同时采用两级散列的 Bloom filter 结构改善冲突，实现了报文分类和路由查找的加速[23].

上述“紧凑型”结构具有节省存储空间、减少 I/O 访存开销方面的优势，适用于一些高速链路上的简单处理任务，不过也存在非常明显的局限[22,24]：(1)能够存储的信息量太少；(2)不支持对流状态信息的精确跟踪、记录和查询；(3)难以实现记录的删除操作和超时老化机制，因此往往必须周期性地对流表进行重建；(4)涉及位级内存读写操作，如果存储器件不支持并行和位级寻址的话，则实际存储带宽开销仍然是比较大的.

因此，从实用性和灵活性角度考虑[25]，散列表仍然是最常见的流表结构[5-6,9,12-14].NetFlow[6]是 Cisco 路由器中广泛采用的流缓存与测量技术，其底层流表结构和流表管理的技术细节尚未公开，有文献宣称是基于散列表的实现[9,22]，在高速网络环境中通常需要采用报文采样[6,10]或流采样[7,9]技术来减轻负载.Xu 等人详细研究了高速 QoS 路由器中的流表设计，针对分布式线卡和集中式转发引擎两种路由器体系结构，分别提出了硬件和软件两套流表设计方案[5]，其中硬件流表采用多路组关联缓存，通过动态组关联机制显著降低了缓存溢出比率，并采用流水线设计以实现 100+Gbps 的吞吐量；软件流表则采用散列表形式，基于随机模型对过期会话记录(垃圾)回收机制的性能和存储开销进行了分析，但是对于流表本身的查询性能反而没有做深入研究，而且所采用的实验数据集中网络流规模只有 6.5 万条左右，与当前实际网络环境的状况存在比较大的差异.

流量监控、状态防火墙、入侵防御、网络地址转换(NAT)、负载分流等等复杂业务的网络会话表也通常采用散列表的结构，在高速网络环境中往往存在比较严重的性能问题.开源项目 Netfilter[12]基于散列表结构实现了一个比较典型的面向网络在线业

务的会话管理机制 Conntrack，具有良好的业务可扩展性，但是由于存在如下一些问题而导致其性能表现相当不尽如人意：(1) 采用表级全局锁进行会话表的读写互斥，这种大粒度的锁机制造成了严重的性能瓶颈；(2) 会话表的查询(快通道任务)与会话记录的添加、删除(慢通道任务)串行执行，导致大量报文的执行路径过长，不仅使得系统吞吐量受到很大影响，而且会带来处理时延的抖动问题；(3) 没有考虑和利用软硬件平台的 SMP(Symmetrical Multiprocessing)、硬件多线程、多核、内核多线程以及多通道内存等并行机制来改善性能. 表 1 给出了采用双核 Intel Xeon 1.86GHz 处理器的 Linux 系统在启用 Netfilter Conntrack 前后桥转发的性能变化情况，可以看出，启用 Netfilter Conntrack 后系统支持的最大并发会话速率下降了 59.78%，吞吐量下降了 63.78%，而 CPU 负载则增加了一个数量级.

**表 1 Netfilter Conntrack 的性能(桥转发模式，报文大小 1024 字节)**

| | 最大并发会话速率/(条·$s^{-1}$) | 桥转发吞吐量(/Kpps)/(/Mbps) | CPU 负载(并发会话速率 8000 条/s)/% |
|---|---|---|---|
| 启用 Netfilter Conntrack 前 | 20466 | 28.2/235.8 | 6.2 |
| 启用 Netfilter Conntrack 后 | 8231 | 10.2/85.4 | 62.1 |

为了提高散列表的查询命中率、减少查询开销，文献[18]采用计数型 Bloom filter 对散列表进行改进，其基本思想是：为每个散列桶维护一个计数器(可保存在片上多通道 SRAM 等高速存储器件中)；添加记录时，将关键码输入到 $k$ 个散列函数中进行计算得到散列索引，然后修改散列索引对应的计数器并将记录插入到对应的散列桶或者计数最小的散列桶中；查询时先判断散列表中是否已存在对应的记录(即报文关键码对应的 $k$ 个散列桶的计数器均不为零)后，再在散列桶中执行顺序匹配，找到对应的记录. 这种方法虽然减少了散列表查询失败次数，但是以牺牲增量更新的效率为代价，导致每次添加和删除操作都需要对 $k$ 个散列桶中的所有记录进行扫描处理. 文献[26]则针对快速报文分类转发业务，提出在查询散列表之前先进行端口比较和端口匹配：通过端口比较，将生命周期较短的流识别出来，直接进行过滤匹配而无需保存记录，以节省存储空间；通过端口匹配，对散列表中是否存在对应的记录进行初步判断，以减少查询失败次数.

为了更好地利用存储空间并避免查表时间开销的抖动，文献[22]采用 d-left 散列对散列表进行负载均衡，即将散列表切分成 $d$ 个相同大小的子表，散列桶大小固定，采用一个小的 TCAM(Ternary Content Addressable Memory)来处理溢出，每个子表对应独立的散列函数，添加记录时从各子表对应的散列桶中选择负载最小的那个散列桶. 这种方法虽然比计数型 Bloom filter 占用更少的存储空间，但是在进行查询操作时需要同时对 $d$ 个子表进行查询.

为了降低散列表的碰撞率、减少散列函数的计算开销，Jenkins Jr 对散列算法进行了研究，提出了一种针对 IP 地址和 TCP/UDP 端口的散列算法(jhash)[27]并已被 Linux 内核所采纳. Xu 等人则研究了适合于动态组关联硬件流表的散列算法，对位提取(bit extraction)、线性散列函数、CRC、Rabin 散列函数和 $H_3$ 等散列算法进行了比较分析，指出 $H_3$ 算法适于硬件实现且具有良好的性能[5].

综上所述，高速网络环境中网络业务的会话管理机制面临着业务弹性与处理性能两个方面的挑战. 实际的应用系统通常为了满足业务的复杂性和灵活性要求而忽视了性能；已有的研究工作则一般着眼于性能而舍弃了业务的弹性，而且往往过于强调局部操作的性能(如查询命中率). 因此，我们将研究的焦点集中于在多核/多线程处理器、软件多进程/多线程等并行执行环境中如何通过并行机制优化会话管理机制性能，并同时保持良好的业务弹性，从已有的文献来看，这方面的研究还开展得比较少.

## 3 会话表设计

### 3.1 会话表结构

如图 2 所示，PaSeM 采用与 Netfilter[12] 相似的、基于散列表的会话表结构，散列表长度为 $L$，散列桶为单向链表结构，采用拉链法解决散列碰撞. 为了实现访问并行，散列表和散列桶均不设访问互斥锁.

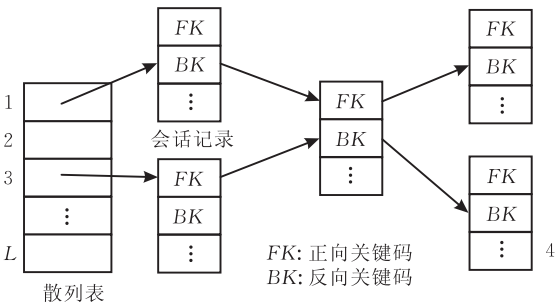



图 2 会话表结构

每条会话记录同时存在于两个散列桶之中，这主要是考虑到实际应用[12,28]中网络地址转换、四层交换、负载分流等业务涉及对报头 IP 地址、TCP/UDP 端口等会话关键码字段的修改，另一方面，会话关键码中也可能包含有诸如输入设备接口[12]等与流量相关的信息，导致会话正向关键码和反向关键码分别映射到不同的 ID(即散列索引)上.

### 3.2　回收缓冲区

会话记录采用动态存储分配，并设置一个会话记录回收缓冲区，如图 3 所示，以先进先出队列(称为 to-be-free list)的方式进行组织，并设置全局锁实现对缓冲区的读/写互斥. 缓冲区中最多包含 $S_f$ 条回收的会话记录，当一条会话记录从会话表中删除回收时，暂时先不释放其存储空间，而是放置到缓存区的队尾，直至该条记录到达缓存区队首并且队列长度超过了 $S_f$ 时，才将其从队列中摘除并释放它的存储空间. $S_f$ 可以根据当前会话到达速率、会话表垃圾回收速率(稳态下，与会话到达速率一致)以及报文的处理时延进行动态调整.

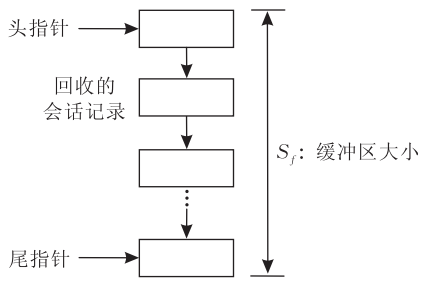



图 3　会话记录回收缓冲区

### 3.3　会话记录和状态机

如表 2 所示，会话记录通常包括如下一些信息：(1)正向关键码(*FK*)和正向链表指针(*Next*)，其中关键码字段存储从报头中提取的字段信息或其散列值，链表指针指向散列桶中的下一条会话记录；(2)反向关键码(*BK*)和反向链表指针(*reNext*)；(3)报文链表头尾指针(*Pktlist*)，用于在创建会话记录的过程中缓存到达的报文；(4)状态信息，包括会话记录建立时间(*Starttime*)、最近报文到达时间戳(*Timestamp*)、会话状态(*Sessionstate*)、老化时间间隔(*Agetime*)、会话记录状态(*Recordstate*)等；(5)业务参数(*Actionpara*)和统计信息(*Statinfo*)；(6)互斥写锁(*Wlock*)，用于对会话记录字段和报文链表的互斥写. 其中，*Sessionstate* 字段保存了会话本身的状态信息，通常与会话的协议类型相关；*Agetime* 字段则与会话的协议类型和 *Sessionstate* 字段密切相关，一般需要针对不同协议类型以及会话的不同状态设置不同的老化时间间隔，以反映特定应用的特点和需求；*Recordstate* 字段则保存了会话记录本身的状态，各状态的定义如表 2 所示.

**表 2　会话记录**

| 字段 | 说明 |
|---|---|
| *flNext* | 回收缓冲区中下一条会话记录指针 |
| *FK* | 会话正向关键码 |
| *Next* | 正向链表中下一条会话记录指针 |
| *BK* | 会话反向关键码 |
| *reNext* | 反向链表中下一条会话记录指针 |
| *Pktlist* | 报文链表头尾指针 |
| *Starttime* | 会话记录建立时间 |
| *Timestamp* | 最近报文到达时间戳 |
| *Agetime* | 老化时间间隔 |
| *Sessionstate* | 会话状态 |
| *Recordstate* | 会话记录状态 |
| *Actionpara* | 业务参数，如下一条路由、访问控制规则 ID 等 |
| *Statinfo* | 统计信息，如报文技术、平均报文长度等 |
| *Wlock* | *Pktlist*、*Sessionstate*、*Recordstate*、*Actionpara*、*Statinfo* 字段的互斥写锁 |

**定义 1.**　“FREE”状态. 会话记录在空闲缓冲区中，未被分配使用.

**定义 2.**　“CREATING”状态. 会话记录正处于创建过程中，还不能正常使用.

**定义 3.**　“WORKING”状态. 会话记录已创建完毕，可正常使用.

**定义 4.**　“HALF-DELETED”状态. 会话记录已经从正向或反向链表中删除，正等待从另一条链表中删除后回收. 此状态用于会话记录回收过程中正反向两个链表操作的同步.

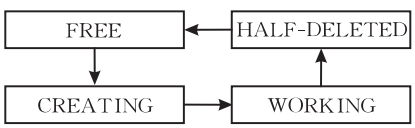



图 4　会话记录的状态变迁

## 4　并行无冲突的会话管理

### 4.1　基本思想

PaSeM 针对多核/多线程处理器、软件多进程/多线程等并行环境，通过并行机制来实现高效的会话管理和高速网络流量的处理，主要思想包括：(1)通过快慢通道任务分离实现“任务并行”. 即将针对报文的快通道任务和针对会话的慢通道任务分别交给不同的独立运行模块执行，前者主要包括会话表的查询和会话记录字段信息的更新，后者则主

要涉及会话记录的添加和删除回收；(2)采用多个处理单元(PE)负责快通道任务的高速并行处理，实现“报文并行”. PE 遵循先来先服务(FCFS)的原则处理报文；(3)采用多个 PE 负责慢通道任务的高速并行处理，将会话表切分成不同的子表，交由不同的 PE 来负责管理，实现“会话并行”；(4)将会话表的不同子表分别放置于多通道存储器件的不同存储单元中，通过“I/O 并行”来提高访存效率；(5)不同模块以及不同 PE 之间采用异步通信机制，通过队列(ring)传递消息或者报文，进一步提高操作的并行度.

### 4.2 PaSeM 并行处理框架

基于上述基本思想，提出 PaSeM 的处理框架，如图 5 所示. 分类查询模块(CM)中包含 $n_1$ 个 PE，这些 PE 从输入队列(RR)中取走报文，以报头中包含的关键码信息代入散列函数，根据计算得到的散列索引找到会话表中对应的散列桶(会话记录链表)，然后将报文的关键码依次与会话记录链表中各会话记录的关键码进行比较，如果关键码相等且会话记录状态 *Recordstate* 为“WORKING”或“CREATING”，则表明找到了报文对应的会话记录，将报文交给后续的业务模块进行处理即可；否则，CM PE 通知会话表管理模块(MM)创建并添加新的会话记录. MM 中同样包含 $n_2$ 个 PE，每个 PE 都拥有自己独立的输入队列(记为 $SR_1, SR_2, \cdots, SR_{n_2}$)，以避免与其他 MM PE 的访问竞争. 此外，每个 MM PE 还各自拥有一个与其他 MM PE 通信用的队列(CR)，记为 $CR_1, CR_2, \cdots, CR_{n_2}$.

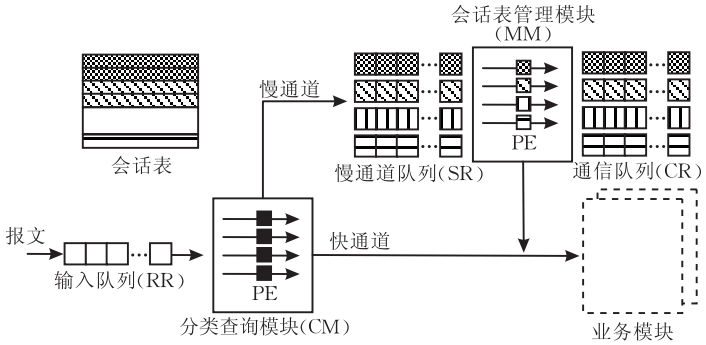



图 5 PaSeM 并行处理框架

在会话表中添加一个会话记录涉及“正向”和“反向”两条会话记录链表的插入操作. CM PE 先通过散列索引确定正向会话记录链表的写者是哪个 MM PE，然后通过该 MM PE 的 SR 队列向其发送一个至少包含报文指针、散列索引信息的消息(记为 $MSG_{new}$). MM PE 收到这个 $MSG_{new}$ 消息后，创建一条会话记录并将其插入正向会话记录链表头，然后再通过 CR 向负责反向会话记录链表插入操作的 MM PE 发送消息(记为 $MSG_{ins}$ 消息)，通知其将会话记录插入反向会话记录链表头. 同样地，从会话表中删除一条会话记录也需要分成两步完成，两个 MM PE 各自通过检查会话记录状态将会话记录从链表中删除，不需要通过消息通信进行异步协同.

会话记录的老化回收也由 MM PE 负责. MM PE 定期或不定期对会话表中的会话记录进行扫描检查，将已经超时老化的会话记录删除回收.

### 4.3 并行冲突

为了保证并行逻辑的正确性并取得较好的并行效率，必须解决高速网络环境下的各种并行冲突问题，包括互斥、同步和死锁.

#### 4.3.1 MM PE 间写互斥

首先，MM PE 之间存在对散列桶的写互斥(即对链表的插入和删除操作互斥)，设置会话表的全局锁或散列桶级锁的做法都会造成很大的性能损失. 为此，PaSeM 将会话表切分成若干子表，如图 5 所示，每个 MM PE 各负责其中一个子表的管理，包括会话记录的插入和删除，以确保每个散列桶都只有固定且唯一的写者.

#### 4.3.2 MM PE 间通信死锁

其次，MM PE 之间如果不是通过 CR 队列而是通过 SR 队列进行通信，则可能造成死锁. 设想如下情况：有两个 MM PE，$PE_i$ 和 $PE_j$，它们各自的 $SR_i$ 和 $SR_j$ 队列都已满，如果 $PE_i$ 要向 $PE_j$ 发送 $MSG_{ins}$ 消息，必须先等待 $PE_j$ 从 $SR_j$ 队列中取走 $MSG_{new}$ 或 $MSG_{ins}$ 消息，同样 $PE_j$ 向 $PE_i$ 发送 $MSG_{ins}$ 消息也必须先等待 $PE_i$ 从 $SR_i$ 队列中取走 $MSG_{new}$ 或 $MSG_{ins}$ 消息，导致这两个 MM PE 进入相互等待的死锁状态.

为此，如前所述，为每个 MM PE 增设一个独立的通信队列 CR，并规定：(a) CR 的长度大于 $n_2\gamma(\gamma$

为大于等于 1 的常数因子)；(b) CR 队列的优先级高于 SR 队列，即 MM PE 必须优先处理 CR 队列中的 $MSG_{ins}$ 消息；(c) MM PE 每次只能从 SR 队列中取走并处理一个 $MSG_{new}$ 消息. 下面给出无死锁证明.

证明.　采用反证法证明.

如果发生了死锁，则存在一个由 $m$ 个($2\leq m\leq n_2$) MM PE 组成的封闭环路，它们的 CR 队列都已满，且每个 MM PE 都因为试图向环路中它的前驱 MM PE 的 CR 队列发送 $MSG_{ins}$ 消息而进入等待状态. 注意到 $MSG_{ins}$ 消息是在处理 $MSG_{new}$ 消息的过程中产生的，因此只有在此过程中 MM PE 才可能进入死锁状态. 设 $PE_m$ 为在时刻 $t_m$ 最后一个进入等待状态的 MM PE，它进入等待状态前最后一次从 $SR_m$ 队列取走 $MSG_{new}$ 消息的时刻为 $t'_m$. $PE_k$ 为封闭环路中 $PE_m$ 的后继，它进入等待状态的时刻为 $t_k$，$t_k\leq t_m$.

由限定(b)和(c)可知，只有在 $t'_m$ 时刻 $CR_m$ 为空时，$PE_m$ 才能从 $SR_m$ 队列中取走一个 $MSG_{new}$ 消息进行处理. 由此可知 $t_k$ 必定落在$[t'_m, t_m]$区间内，且 $CR_m$ 必须在$[t'_m, t_k]$区间内被填满，否则 $PE_k$ 就不能在 $t_k$ 时刻进入等待状态. 设 MM PE 从 SR 队列中取走一个 $MSG_{new}$ 消息进行处理到向 CR 队列发送 $MSG_{ins}$ 消息所费时间的下限和上限分别为 $T_a$ 和 $T_b$ ($T_a\leq T_b$，且$\lceil T_b/T_a\rceil=\gamma$)，即一个 MM PE 往 CR 队列发送 $MSG_{ins}$ 消息的时间间隔至少为 $T_a$. 从而有

$$CR_m\text{的长度}\leq n_2\lceil(t_k-t'_m)/T_a\rceil\leq n_2\gamma$$

(这里 $t_k-t'_m\leq t_m-t'_m\leq T_b$).

与限定条件(a)矛盾. 因此不会产生死锁.　　证毕.

#### 4.3.3　CM PE 与 MM PE 间读写互斥

CM PE 与 MM PE 之间同样存在对会话记录链表的读写互斥，为此在执行链表插入和删除操作过程中通过“写”原子指令和严格规定的指令次序确保对于 CM PE(读者)来说任一时刻链表结构的完整性.

如果 MM PE(写者)要在会话记录 A 和 C 之间插入一个新的会话记录 B，则应先读取 A 的 *Next/reNext* 字段(即 C 的地址)，然后用“写”原子指令将读取的值填入 B 的 *Next/reNext* 字段，再用“写”原子指令将 B 的地址填入 A 的 *Next/reNext* 字段. 同理，当要删除会话记录 A 和 C 之间的会话记录 B 时，首先应读取 B 的 *Next/reNext* 字段(即 C 的地址)，然后用“写”原子指令将读取的值写入 A 的 *Next/reNext* 字段.

此外，在从链表中删除会话记录 B 的过程中，可能有一些 CM PE 正在读取 B 的信息(如查询链表时正好“经过”B)，而将 B 删除回收后仍然可能存在一些正“使用”B 的报文还没有处理完. 为了减少不必要的互斥开销并防止产生脏数据，B 删除后将先暂时存入会话记录回收缓冲区中，并且不释放内存空间，也不清空其中的数据(包括 *Next/reNext* 字段)，而是等到 B 被移出会话记录回收缓冲区时再执行清空和释放内存空间的操作. 为了给上述对 B 的访问留出足够多的时间，设置会话记录回收缓冲区大小的阈值 $S_f$，如图 3 所示，使得会话记录回收缓冲区中一直保留有 $S_f$ 个回收的会话记录，从而保证 B 被回收后不会即刻被重新分配出去. 稳态下，$S_f$ 的取值与会话到达速率以及 CM PE 处理报文的服务时间相关.

#### 4.3.4　重复的 $MSG_{new}$ 消息

快慢通道任务并行的策略以及 CM PE 与 MM PE 之间的异步消息通信机制还可能导致产生重复的 $MSG_{new}$ 消息. UDP 或 ICMP 等非连接通信、TCP 报文重传或者“洪泛”攻击等都有可能使得会话首报文与同方向后续报文之间到达间隔过小，MM PE 来不及创建会话记录并将其插入正向会话记录链表，从而导致 CM PE 连续发送相同的 $MSG_{new}$ 消息给 MM PE. 由于散列桶的写者固定且唯一，因此重复的 $MSG_{new}$ 消息依然会按照先后次序交由同一个 MM PE 处理. 为此，MM PE 需要在处理每个 $MSG_{new}$ 消息前确认一下链表中是否已存在对应的会话记录，如果已经存在，则丢弃重复的 $MSG_{new}$ 消息. 由于新会话记录总是插入会话记录链表的头部，因此重复消息的查询开销基本可以忽略不计.

另一方面，反向报文与会话首报文之间存在协议交互时延，足够让系统完成添加会话记录的工作，因此会话的反向流量通常不会触发重复的 $MSG_{new}$ 消息.

#### 4.3.5　包序控制

最后，并行处理还会带来报文的包序控制问题. 为了保证输入和输出的报文次序一致，在创建和添加会话记录时 PaSeM 将已经到达的报文按照先后次序缓存到 *Pktlist* 链表中，等待会话记录添加完成后再按照次序将报文交给后继的业务模块处理，从而保证在这个阶段中会话的报文次序. 在快通道任务执行阶段，由于 CM 对输入的报文采取 FCFS 的策略，且快通道任务相对比较简单且执行分支少，各执行分支的时间开销粒度通常比较均匀，属于同一

条流的报文的处理时间不会有太大波动，因此 CM 基本能够保持同一条会话同方向的报文次序.

### 4.4 并行查询算法

会话表的并行查询算法如下.

**算法 1.** 会话表并行查询算法.

$h$：散列函数，$L$：散列表长度

$R.key$：会话记录中的 $FK$ 或 $BK$ 字段

1. forall CM PE $P_i$，$i=1,2,\cdots,n_1$
2. 从队列 RR 中取出一个报文 $PK$
3. $j \leftarrow h(PK.key) \% L$
   //计算散列桶编号，$PK.key$ 为报文的关键码
4. for 散列桶 bucket[$j$]中的每个会话记录 $R$
5. if ($R.key=PK.key$) then
6. 获得 $R.Wlock$ 锁，$rs \leftarrow R.Recordstate$
7. if ($rs$=WORKING) then
8. 更新 $R.Sessionstate$，$R.Timestamp$，$R.Agetime$
9. else if ($rs$=CREATING) then
10. 将报文 $PK$ 插入 $R.Pktlist$ 链表尾
11. endif
12. 释放 $R.Wlock$ 锁
13. if ($rs \neq$ CREATING) then 将报文交给后继的业务模块处理
14. exit
15. endif
16. endfor
17. if (在 bucket[$j$]中找不到对应的会话记录) then //missing
18. $k \leftarrow j \% n_2$//计算该散列桶对应的 MM PE 编号
19. 向队列 $SR_k$ 发送一个包含 $j$ 和 $PK$ 指针的 $MSG_{new}$
20. endif
21. Endforall

各个 CM PE 从队列 RR 中取出报文，根据报文的关键码查询会话表. 如果查询命中(hitting)，则将报文交给后继业务模块，根据会话记录的 $Recordstate$ 字段执行不同的处理. 当 $Recordstate$ 为“HALF-DELETED”或“FREE”时，表明有 MM PE 正在执行或已经完成该会话记录的删除回收操作，不过会话记录的数据还暂时不会被清空，因此并不会影响到 CM PE 的执行. 如果查询失败(missing)，则 CM PE 向对应的 MM PE 发送 $MSG_{new}$ 消息，创建和添加新的会话记录. 各个 CM PE 之间的查询操作可以完全并行，CM PE 与 MM PE 之间也没有散列桶的互斥读写问题，两类 PE 可以各自并行无冲突地工作. 整个过程中只有在修改会话记录的状态和统计信息以及报文链表 $Pktlist$ 时，才需要加上写锁.

### 4.5 会话表的动态管理

MM PE 按照优先处理 CR 队列(每次处理完队列中的所有 $MSG_{ins}$ 消息)、其次处理 SR 队列(每次处理 1 个 $MSG_{new}$ 消息)、最后执行垃圾回收(扫描会话表、删除回收过期会话记录)操作的次序循环工作，算法如下.

**算法 2.** 会话表动态管理算法.

$R$：$MSG_{ins}$ 消息所指向的会话记录

$j$：$MSG_{ins}$ 消息所指向的反向会话记录链表(散列桶)的编号

1. forall MM PE $Q_i$，$i=1,2,\cdots,n_2$
2. while (队列 $CR_i$ 非空)
3. 从队列 $CR_i$ 中取走一个 $MSG_{ins}$ 消息，
4. 将会话记录 $R$ 插入散列桶 bucket[$j$]的链表头部
5. endwhile
6. if (队列 $SR_i$ 非空) then
7. 从队列 $SR_i$ 中取走一个 $MSG_{new}$ 消息
8. 执行会话记录的创建和添加操作
9. else
10. 执行扫描会话表、删除回收过期会话记录的操作
11. endif
12. endforall

MM PE 利用“空闲”时间扫描它负责的会话子表，将超过老化期限(当前时刻$-Timestamp > Agetime$)或者 $Recordstate$ 为“HALF-DELETED”的会话记录从会话表中删除，并放置到回收缓冲区. 为了控制一批次垃圾回收操作开销的粒度，避免 CR 和 SR 队列发生溢出，可指定一次扫描的散列桶个数($S_r$). 此外，为了控制老化扫描的频率、避免耗费过多的 I/O 访存开销，可采取一定的保护措施，如限定 MM PE 扫描一遍会话子表的最小时间间隔.

#### 4.5.1 创建添加会话记录

在会话表中添加一个会话记录涉及到正向和反向两条会话记录链表的插入操作，需要分别由对应的两个 MM PE(记为 $PE_i$、$PE_j$)来完成. $PE_i$ 先创建新的会话记录并将其插入到正向会话记录链表中，然后通过 $CR_j$ 向 $PE_j$ 发送 $MSG_{ins}$ 消息，由 $PE_j$ 将会话记录插入反向会话记录链表. 完整的算法如下.

**算法 3.** 创建添加会话记录.

$PK$：$MSG_{new}$ 消息所指向的报文

$k$：$MSG_{new}$ 消息所指向的正向会话记录链表(散列桶)

的编号

$l$：$MSG_{ins}$消息所指向的反向会话记录链表(散列桶)的编号

$h$：散列函数，$L$：散列表长度

1. $PE_i$从队列$SR_i$中取走一个$MSG_{new}$消息
2. if (查询 bucket[$k$]找到对应的会话记录) then
3. 忽略重复的$MSG_{new}$消息，根据会话记录对$PK$进行处理，exit
4. endif
5. 创建一个新的会话记录$R$
6. $R.FK \leftarrow PK.key$，$R.Recordstate \leftarrow$CREATING
7. $R.Starttime$，$R.Timestamp \leftarrow$now
8. 将$PK$插入$R.Pktlist$链表尾部
9. 将会话记录$R$插入散列桶 bucket[$k$]的链表头部
10. 填写会话记录$R$的其它字段
11. $l \leftarrow h(R.BK)\%L$，$j \leftarrow l\%n_2$
    //计算反向的散列桶及其对应的 MM PE 编号
12. 向$CR_j$队列发送一个包含$l$和$R$的指针的$MSG_{ins}$消息
13. 获得$R.Wlock$锁
14. 从头至尾依次释放并处理$R.Pktlist$链表中的报文，$R.Pktlist \leftarrow$NULL
15. $R.Recordstate \leftarrow$WORKING
16. 释放$R.Wlock$锁

为了检查重复的$MSG_{new}$消息，MM PE 需要先查询一次散列桶(见算法第 2 步)，如果散列桶已经存在对应的会话记录，则可知当前的$MSG_{new}$消息为重复发送的消息.由于新建的会话记录总是插入到散列桶会话记录链表的头部，因此重复的$MSG_{new}$消息的处理开销很小，基本可以忽略不计.而对于非重复的$MSG_{new}$消息，考虑到会话首报文的数量通常在总流量中只占较小的比例，因此多余一轮会话记录链表的扫描查询开销还是可以承受的.

在创建一个新的会话记录时，如果存储空间不足，可以采取丢弃报文的“抑制”策略.从安全性的角度来考虑，对于过载的流量通常都采取丢弃的做法.如果是正常的突发业务流量导致的网络会话激增，则 TCP 协议的重传机制可保证峰值过后业务会话仍然能够正常建立；如果是由于拒绝服务攻击所导致的，则这种策略在一定程度上能够起到类似于“首报文丢弃”的防御效果.

#### 4.5.2　删除回收会话记录

同理，在会话表中删除回收一个会话记录也需要对正向和反向两条会话记录链表进行操作.$PE_i$负责将会话记录从一条记录链表中删除，$PE_j$负责将会话记录从另一条会话记录链表中删除并回收到会话记录回收缓存区中.为了避免死锁，$PE_i$执行删除操作后不会向$PE_j$发送同步消息，即$PE_i$、$PE_j$各自通过扫描所负责的会话子表完成删除和回收过程.这样做虽然会增加过期会话记录删除回收的平均时延，但是并不会增加最坏情况下的时延.具体算法如下.

**算法 4.**　删除回收会话记录.

$i$：$MSG_{new}$消息所指向的正向会话记录链表(散列桶)的编号

$j$：$MSG_{ins}$消息所指向的反向会话记录链表(散列桶)的编号

$h$：散列函数，$L$：散列表长度

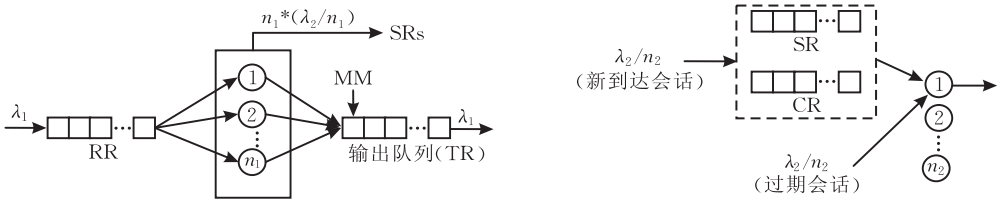
1. $PE_i$将会话记录$R$从它负责的会话记录链表中删除
2. 获得$R.Wlock$锁
3. if ($R.Recordstate$ =HALF-DELETED) then
4. $R.Recordstate \leftarrow$FREE
5. elseif ($R.Recordstate \neq$FREE)
6. $R.Recordstate \leftarrow$HALF-DELETED
7. endif
8. 释放$R.Wlock$锁
9. if ($R.Recordstate$=FREE) then
10. 将$R$回收到缓冲区中，并保留会话记录的数据
11. if (会话记录回收缓存区已满) then
12. 将缓冲区链表头的会话记录摘除，并释放其存储空间
13. endif
14. endif

## 5　性能分析评价

### 5.1　性能评价模型

报文到达具有自相似性，假定报文到达为时间间隔服从均值为$ab/(a-1)$、方差为$ab^2/[(a-1)^2(a-2)]$(其中$a>2$)的 Pareto 分布，报文到达平均速率$\lambda_1=(a-1)/(ab)$.对于会话的首报文和后续报文，CM PE 在查询会话表时将产生失败(missing)和命中(hitting)两种不同的情况，其服务时间也将遵循两种不同的分布，因而 CM 构成了$G/G_2/n_1$型排队系统，如图 6(a)所示，其中$G_2$表示 CM PE 对于 RR 队列的服务时间独立且服从两类一般概率分布，$n_1$为 CM PE 的数目.

文献[5]和文献[29]中指出，在骨干网层次观察，网络会话到达和活跃会话数量并没有呈现自相似特性，会话的到达仍然可以视为平均速率为$\lambda_2$的泊松过程，到达时间间隔服从参数为$1/\lambda_2$的负指数




(a) CM性能评价: $G/G_2/n_1$排队模型

(b) MM性能评价: 空竭服务、多重休假的M/G+D/1型排队模型

图 6 性能评价模型

分布. 假定各 SR 和 CR 队列输入独立同分布,均值为 $\lambda_2/n_2$(忽略产生重复 $MSG_{new}$ 消息的情况),其中 $n_2$ 为 MM PE 的数目,亦即 $MSG_{new}$ 和 $MSG_{ins}$ 消息的处理负载将平均分担到各个 MM PE 上. 由图 6(a)可知,SR 队列的输入为 $G/G_2/n_1$ 系统的输出,视其为泊松过程,即 $MSG_{new}$ 消息的到达时间间隔服从参数为 $n_2/\lambda_2$ 的负指数分布. 而 CR 队列的输入取决于 SR 队列的输出,由于 CR 有最高的处理优先级,因此 CR 中 $MSG_{ins}$ 消息的等待时间可以忽略不计;又,为方便起见,假定创建添加一条会话记录的工作全部由同一个 MM PE 完成,而不是分由两个 MM PE 负责,由于各 MM PE 之间无差异且 CR 队列的服务时间 $T_{CR}$ 为定长分布,因此这个假定不会对性能评价造成影响. 综上所述,每个 MM PE 都构成空竭服务(exhaustive service)、多重休假(multiple vacations)的 M/G+D/1 型排队系统,如图 6(b)所示,其中 M 表示会话到达过程为泊松分布,G 表示 MM PE 对于 SR 队列的服务时间独立且服从一般概率分布,D 表示 MM PE 对于 CR 队列的服务时间服从定长分布.

MM PE 只有在 SR 和 CR 队列为空时才能进入"休假"期(即遵循"空竭服务"规则),在"休假"期间 MM PE 执行扫描回收过期会话记录的工作,每次"休假"时间(扫描 $S_r$ 个散列桶)为独立于到达和服务过程的独立同分布. 当一次"休假"结束后,若 SR 和 CR 队列仍然为空,就开始另一次"休假"(即遵循"多重休假"规则),对扫描会话子表的频率不做限制;在休假结束时,若 SR 或 CR 队列不为空,就进入新的"忙期".

性能评价模型的主要参数见表 3.

**表 3 性能评价模型的参数**

| 参数 | 说明 | 实验值 |
|---|---|---|
| $\lambda_1$ | 报文到达平均速率 | |
| $\lambda_2$ | 会话到达平均速率, $\lambda_2 \le \lambda_1$ | |
| $L$ | 散列表长度 | |
| $d$ | 散列桶深度 | |
| $T_d$ | 会话活跃时间 | |
| $T_f^x$ | $x=m$,匹配一条会话记录的时间开销 | (0,0; 1,0; 18,187)* |
| | $c$,创建一条会话记录的时间开销 | (10,3; 10,3; 356,2920) |
| | $i$,在散列桶中插入一条会话记录的时间开销 | (0,1; 0,1; 2,114) |
| | $u$,更新一条会话记录状态信息的时间开销 | (0,2; 0,1; 13,158) |
| | $s$,在扫描回收过期会话记录时,检查一条会话记录的时间开销 | (0,0; 1,0; 21,170) |
| | $d$,从散列桶中删除一条会话记录的时间开销 | (0,0/1; 0,1/0; 1,70/44)# |
| | $r$,将一条会话记录回收到缓存区中的时间开销 | (1,3; 0,1; 10,293) |
| $T_r^x$ | $x=r$,队列 RR、SR 或 CR 的读时间开销 | (1,0; 0,0; 1,109) |
| | $w$,队列 RR、SR 或 CR 的写时间开销 | (0,1; 0,0; 1,91) |
| $T_p$ | 读取报文数据的时间开销 | (0,0; 1,0; 1,187) |
| $T_h$ | 计算散列值并读取散列表项的时间开销 | (1,0; 0,0; 1,91) |
| $T_b$ | 在散列桶中查找一条会话记录的时间开销 | |
| $T_a$ | 会话老化时间间隔 | |
| $T_v$ | 执行一批次扫描和回收过期会话记录的时间开销(即一次"假期"的时长) | |
| $T_w$ | MM PE"忙期"的时长 | |
| $T_{mv}$ | 两次"忙期"之间多重"假期"的时长 | |
| $T_X$ | 队列 $X$($X$=RR, SR, CR)的服务时长 | |
| $S_r$ | 一次"假期"中扫描的散列桶个数, $1 \le S_r \le \lceil L/n_2 \rceil$ | |

注: * (读 SRAM 次数,写 SRAM 次数; 读 SDRAM 次数,写 SDRAM 次数; 执行指令数,指令的总时钟周期开销),下同.

# $T_f^d = 44 \times \min(L, 2N)/(2N) + 70 \times \max(2N-L, 0)/(2N)$. 通常 $N > L$,因此 $T_f^d \approx 70$(时钟周期).

### 5.2 性能指标

主要性能指标如表 4 所示. 除了排队模型关注的 PE 数目、队列长度、等待时间(waiting time)、逗留时间(sojourn time,即等待时间+服务时间)等指标外,还包括会话表中会话记录总数和活跃会话记录的数量、垃圾比率、会话记录回收缓冲区大小等指标.

**表 4 性能指标**

| 指标 | 说明 |
|---|---|
| $n_1$ | CM PE 数量 |
| $n_2$ | MM PE 数量 |
| $L_X^q$ | 队列 $X(X=\mathrm{RR},\mathrm{SR})$长度,即在队列中等待的报文/消息数 |
| $W_X^q$ | 队列 $X(X=\mathrm{RR},\mathrm{SR})$的等待时间 |
| $W_X$ | 队列 $X(X=\mathrm{RR},\mathrm{SR})$的逗留时间(等待时间+服务时间) |
| $N$ | 会话表中会话记录总数均值 |
| $N_a$ | 会话表中活跃会话记录数量均值 |
| $GR$ | 会话表的垃圾比率,即$(N-N_a)/N_a$ |
| $S_f$ | 会话记录回收缓冲区大小 |

### 5.3 报文处理性能分析

下面对稳态(steady state)下队列 RR 的长度 $L_{\mathrm{RR}}^q$、等待时间 $W_{\mathrm{RR}}^q$ 和逗留时间 $W_{\mathrm{RR}}$ 以及 CM PE 数量 $n_1$ 的下限进行定量分析.

给定会话表中的平均会话总数 $N$,假设散列桶深 $d$ 服从二项分布,则会话首报文查询时间的概率分布为

$$P(T_b=xT_f^m\mid\mathrm{missing})=P(d=x)=\binom{2N}{x}\left(\frac{1}{L}\right)^x\left(1-\frac{1}{L}\right)^{2N-x},\quad x=0,1,2,\cdots,2N \tag{1}$$

会话后续报文查询时间的概率分布为

$$P(T_b=xT_f^m\mid\mathrm{hitting})=\frac{L}{2N}\sum_{y=x}^{2N}P(d=y,y>0),\quad x=1,2,\cdots,2N \tag{2}$$

采用递推法,可得会话表散列桶深 $d$ 的期望:

$$E(d)=\sum_{x=1}^{2N}xP(d=x)=\frac{2N}{L},$$

以及

$$\sum_{x=1}^{2N}x^2P(d=x)=\frac{2N}{L}\left(1+\frac{2N-1}{L}\right),$$

$$\sum_{x=1}^{2N}x^3P(d=x)=\frac{2N}{L}\left(1+\frac{3(2N-1)}{L}+\frac{2(N-1)(2N-1)}{L^2}\right) \tag{3}$$

从而得到会话表散列桶查询时间的概率分布为

$$E(T_b\mid\mathrm{missing})=\sum_{x=0}^{2N}xT_f^mP(T_b=xT_f^m\mid\mathrm{missing})=T_f^mE(d)=\frac{2N}{L}T_f^m \tag{4}$$

$$E(T_b\mid\mathrm{hitting})=\sum_{x=1}^{2N}xT_f^mP(T_b=xT_f^m\mid\mathrm{hitting})=\frac{LT_f^m}{2N}\sum_{x=1}^{2N}\sum_{y=x}^{2N}(xP(d=y,y>0))=T_f^m\left(1+\frac{2N-1}{2L}\right) \tag{5}$$

以及

$$E(T_b^2\mid\mathrm{missing})=\sum_{x=0}^{2N}(xT_f^m)^2P(T_b^2=(xT_f^m)^2\mid\mathrm{missing})=(T_f^m)^2\sum_{x=0}^{2N}x^2P(T_b=xT_f^m\mid\mathrm{missing})=\frac{2N(T_f^m)^2}{L}\left(1+\frac{2N-1}{L}\right) \tag{6}$$

$$E(T_b^2\mid\mathrm{hitting})=\sum_{x=1}^{2N}(xT_f^m)^2P(T_b^2=(xT_f^m)^2\mid\mathrm{hitting})=\frac{L(T_f^m)^2}{2N}\sum_{x=1}^{2N}\sum_{y=x}^{2N}(x^2P(d=y,y>0))=(T_f^m)^2\left(1+\frac{3(2N-1)}{2L}+\frac{2(N-1)(2N-1)}{3L^2}\right) \tag{7}$$

$\mathrm{G/G_2}/n_1$ 排队系统的服务时间:

$$T_{\mathrm{RR}}=\begin{cases}T_r^r+T_p+T_h+T_b\mid\mathrm{hitting}+T_f^u+T_r^w, & \text{会话的首报文}\\ T_r^r+T_p+T_h+T_b\mid\mathrm{missing}+T_r^w, & \text{会话的后续报文}\end{cases} \tag{8}$$

由式(4)～(7),可计算出队列 RR 的服务时间的期望 $E(T_{\mathrm{RR}})$以及方差 $E(T_{\mathrm{RR}}^2)$:

$$\begin{aligned}E(T_{\mathrm{RR}})&=P(\mathrm{hitting})E(T_{\mathrm{RR}}\mid\mathrm{hitting})+P(\mathrm{missing})E(T_{\mathrm{RR}}\mid\mathrm{missing})\\&=\frac{\lambda_1-\lambda_2}{\lambda_1}(T_r^r+T_p+T_h+E(T_b\mid\mathrm{hitting})+T_f^u+T_r^w)+\frac{\lambda_2}{\lambda_1}(T_r^r+T_p+T_h+E(T_b\mid\mathrm{missing})+T_r^w)\\&=T_r^r+T_p+T_h+T_r^w+T_f^u\frac{\lambda_1-\lambda_2}{\lambda_1}+\frac{T_f^m}{2L\lambda_1}((2N+2L-1)\lambda_1+(2N-2L+1)\lambda_2)\end{aligned} \tag{9}$$

$$\begin{aligned}E(T_{\mathrm{RR}}^2)&=P(\mathrm{hitting})E(T_{\mathrm{RR}}^2\mid\mathrm{hitting})+P(\mathrm{missing})E(T_{\mathrm{RR}}^2\mid\mathrm{missing})\\&=\frac{\lambda_1-\lambda_2}{\lambda_1}((T_r^r+T_p+T_h+T_f^u+T_r^w)^2+2(T_r^r+T_p+T_h+T_f^u+T_r^w)E(T_b\mid\mathrm{hitting})+E(T_b^2\mid\mathrm{hitting}))+\frac{\lambda_2}{\lambda_1}((T_r^r+T_p+T_h+T_r^w)^2+2(T_r^r+T_p+T_h+T_r^w)E(T_b\mid\mathrm{missing})+E(T_b^2\mid\mathrm{missing}))\end{aligned} \tag{10}$$

(1) 当 CM PE 的数量 $n_1=1$ 时，CM 为 G/G/1 排队系统，当 $\rho_1=\lambda_1 E(T_{RR})<1$ 时，有[30]

$$\frac{\lambda_1 E(T_{RR}^2)-E(T_{RR})(2-\lambda_1 E(T_{RR}))}{2(1-\lambda_1 E(T_{RR}))}\leq E(W_{RR}^q)\leq\frac{\lambda_1(E(T_{RR}^2)+ab^2/((a-1)^2(a-2)))}{2(1-\lambda_1 E(T_{RR}))} \quad (11)$$

当 CM PE 的负载 $\rho_1=\lambda_1 E(T_{RR})/n_1<1$ 且充分接近 1(即处于 heavy traffic 状态)时，在稳态下 $W_{RR}^q$ 近似服从负指数分布，从而有[30-31]

$$E(W_{RR}^q)=\frac{\lambda_1(E(T_{RR}^2)+ab^2/((a-1)^2(a-2)))}{2(1-\lambda_1 E(T_{RR}))} \quad (12)$$

由 Little 公式可算出此时队列 RR 长度的期望为

$$E(L_{RR}^q)=\lambda_1 E(W_{RR}^q)=\frac{\lambda_1^2(E(T_{RR}^2)+ab^2/((a-1)^2(a-2)))}{2(1-\lambda_1 E(T_{RR}))} \quad (13)$$

(2) 当 CM PE 的数量 $n_1>1$ 时，由文献[30,32]可知：

$$\frac{\lambda_1 E(T_{RR}^2)-E(T_{RR})(2-\lambda_1 E(T_{RR}))}{2(1-\lambda_1 E(T_{RR}))}-\frac{(n_1-1)E(T_{RR}^2)}{2n_1 E(T_{RR})}\leq E(W_{RR}^q)\leq\frac{\lambda_1(n_1^2(ab^2/((a-1)^2(a-2)))+n_1 E(T_{RR}^2)-(n_1-1)E^2(T_{RR})}{2(n_1^2-n_1\lambda_1 E(T_{RR}))} \quad (14)$$

当 CM PE 的负载

$$\rho_1=\lambda_1 E(T_{RR})/n_1<1 \quad (15)$$

且充分接近 1(即处于 heavy traffic 状态)时，在稳态下 $W_{RR}^q$ 近似服从负指数分布，从而有[30-31]

$$E(W_{RR}^q)\approx\frac{\lambda_1(n_1(ab^2/((a-1)^2(a-2)))+E(T_{RR}^2))}{2(n_1-\lambda_1 E(T_{RR}))} \quad (16)$$

又，实际情况下采用随机性足够好的散列函数且散列表足够大时，会话记录在会话表中趋向于均匀分布，使得 $E(T_{RR}^2)$ 趋近于 $E^2(T_{RR})$，从而有

$$E(W_{RR}^q)\approx\frac{\lambda_1(n_1(ab^2/((a-1)^2(a-2)))+E^2(T_{RR}))}{2(n_1-\lambda_1 E(T_{RR}))} \quad (17)$$

同式(13)，根据 $E(W_{RR}^q)$ 由 Little 公式可求得 $E(L_{RR}^q)$.

将式(9)代入式(15)，展开后可知若 CM 存在稳态则 CM PE 的数量 $n_1$ 应满足

$$n_1>\frac{2N(\lambda_1+\lambda_2)T_f^m+(\lambda_1-\lambda_2)((2L-1)T_f^m+2LT_f^u)+2L\lambda_1(T_r^r+T_p+T_h+T_r^w)}{2L} \quad (18)$$

### 5.4 会话处理性能分析

接下来对稳态下队列 SR 的长度 $L_{SR}^q$、等待时间 $W_{SR}^q$、逗留时间 $W_{SR}$ 以及 MM PE 数量 $n_2$ 进行定量分析.

MM PE 创建添加一条会话记录所花费的总时间(即 M/G+D/1 排队系统的服务时间)为 $T_{SR}+T_{CR}$，其中

$$T_{SR}=T_r^r+T_h+T_b|\text{missing}+T_f^c+T_f^i+T_f^u+T_r^w \quad (19)$$

$$E(T_{SR})=T_r^r+T_h+T_f^c+T_f^i+T_f^u+T_r^w+E(T_b|\text{missing})=T_r^r+T_h+T_f^c+T_f^i+T_f^u+T_r^w+\frac{2N}{L}T_f^m \quad (20)$$

$$T_{CR}=E(T_{CR})=T_r^r+T_h+T_f^i+T_r^w \quad (21)$$

将系统中的有效会话记录(active flow records)看成 M/G/∞型排队系统中正在接受服务的“顾客”，服务时间即为会话的生命周期(session lifespan)，则由文献[33]可知，系统中有效会话记录的数目服从泊松分布. 假定会话的活跃期(session duration) $T_d$ 服从均值为 $e^{a+\delta^2/2}$(其中 $\sigma>0$)的对数正态分布[34]，则会话的生命期均值为 $e^{a+\delta^2/2}+T_a$，由 Little 公式可得会话表中有效会话记录数量的均值为

$$N_a=\lambda_2(T_d+T_a)=\lambda_2(e^{a+\delta^2/2}+T_a) \quad (22)$$

稳态下系统内部将维持一种动态的平衡关系，即单位时间内平均新产生的过期会话记录数(亦即需要回收的会话记录数)与平均新到达的会话数量相同，均为 $\lambda_2$. 由 MM PE 扫描散列桶回收过期会话记录的批处理策略(粒度为 $S_r$)可知，稳态下 MM PE 在一次“休假”期间扫描会话记录的次数服从参数为 $S_r/L$ 的二项分布，均值为 $2NS_r/L$. 同理，稳态下 MM PE 在一次“休假”期间删除过期会话记录次数和回收会话记录次数也服从参数为 $S_r/L$ 的二项分布，均值分别约为 $2(N-N_a)S_r/L$ 和 $(N-N_a)S_r/L$. 因此，稳态下执行一批次扫描和回收过期会

话记录(即一次“假期”)的时间开销 $T_v$ 的期望和方差分别为

$$E(T_v)=\frac{S_r}{L}(2NT_f^s+2(N-N_a)T_f^d+(N-N_a)T_f^r) \tag{23}$$

$$D(T_v)=E(T_v)\left(1-\frac{S_r}{L}\right) \tag{24}$$

对于空竭服务、多重休假的 M/G+D/1 排队系统，当

$$\rho_2=\lambda_2(E(T_{SR})+E(T_{CR}))/n_2<1 \tag{25}$$

时，SR 队列中消息数量的期望为[30,35]

$$E(L_{SR}^q)=\frac{\lambda_2^2E((T_{SR}+T_{CR})^2)}{2n_2^2(1-\rho_2)}+\frac{\lambda_2E(T_v^2)}{2n_2E(T_v)} \tag{26}$$

则队列 SR 中消息等待时间、逗留时间(等待时间+服务时间)的期望分别为

$$E(W_{SR}^q)=\frac{E(L_{SR}^q)}{\lambda_2/n_2}=\frac{\lambda_2E((T_{SR}+T_{CR})^2)}{2n_2(1-\rho_2)}+\frac{E(T_v^2)}{2E(T_v)},$$

$$E(W_{SR})=E(T_{SR}+T_{CR})+E(W_{SR}^q) \tag{27}$$

其中 $E(T_v^2)=D(T_v)+E^2(T_v)$，实际情况下会话记录在会话表中趋向于均匀分布时，$D(T_v)$ 趋近于 0；由式(19)和式(21)可知：

$$\begin{aligned}&E((T_{SR}+T_{CR})^2)\\&=(2(T_r^r+T_h+T_f^i+T_r^w)+T_f^c+T_f^u)^2+\\&\quad 2(2(T_r^r+T_h+T_f^i+T_r^w)+T_f^c+T_f^u)\cdot\\&\quad E(T_b|\text{missing})+E(T_b^2|\text{missing})\end{aligned} \tag{28}$$

又，将式(20)、式(21)代入式(25)，展开后，再由 $N\geq N_a$ 可得稳态下 MM PE 的数量 $n_2$ 应满足

$$\begin{aligned}n_2&>\lambda_2(2T_r^r+2T_h+2T_f^i+2T_r^w+T_f^c+T_f^u)+\frac{2\lambda_2T_f^mN}{L}\\&\geq\lambda_2(2T_r^r+2T_h+2T_f^i+2T_r^w+T_f^c+T_f^u)+\frac{2\lambda_2T_f^mN_a}{L}\end{aligned} \tag{29}$$

由式(29)可知，在不限制队列 SR 长度的情况下，调整 MM PE 每批次扫描回收过期会话记录的粒度 $S_r$ 不会对 MM PE 的数量 $n_2$ 造成影响.

### 5.5　存储开销分析

下面对稳态下会话表大小、垃圾比率以及回收缓冲区大小做定量分析.

由文献[35]可知，MM PE“忙期”(添加会话记录的时间)的平均长度为

$$E(T_w)=\frac{\rho_2E(T_v)}{(1-\rho_2)(1-LS(T_v))} \tag{30}$$

其中 $LS(T_v)$ 为 $T_v$ 的 Laplace-Stieltjes 变换；MM PE“全假期”(前后两次“忙期”之间的若干次连续“假期”之和，扫描回收操会话记录的时间)的平均长度为

$$E(T_{mv})=\frac{E(T_v)}{1-LS(T_v)} \tag{31}$$

从而稳态下对于 MM PE 的一个“忙循环”(即一次“全假期”加上其后的忙期)存在平衡方程：

$$(E(T_{mv})+E(T_w))\frac{\lambda_2}{n_2}=(N-N_a)\frac{E(T_{mv})S_r}{E(T_v)L} \tag{32}$$

即在“忙循环”期间新产生的过期会话记录数应等于回收的会话记录数. 将式(20)、(21)、(23)、(25)和式(30)、(31)代入式(32)后，展开计算得到

$$N=\frac{B-\sqrt{B^2-4AC}}{2A} \tag{33}$$

其中，$A=2\lambda_2T_f^m$，$B=n_2L-2\lambda_2(LT_f^s-N_aT_f^m)-\lambda_2L(\theta_1+\theta_2)$，$C=(n_2L-\lambda_2L(\theta_1+\theta_2))N_a$，$\theta_1=2T_f^d+T_f^r$，$\theta_2=2(T_r^r+T_h+T_f^i+T_r^w)+T_f^c+T_f^u$.

再由 $N$、$N_a$ 可计算出会话表的垃圾比率(garbage ratio)[5]：$GR=(N-N_a)/N_a$.

考虑一个报文从进入系统到离开的时间间隔内回收的过期会话记录数目(稳态下亦即新到达会话数目)，则回收记录缓冲区大小 $S_f$ 应满足

$$S_f\geq\left(E(W_{RR})+\frac{\lambda_2}{\lambda_1}E(W_{SR})\right)\lambda_2 \tag{34}$$

## 6　实　验

以 Intel IXP2400 网络处理器、8MB SRAM、1GB SDRAM 和 4 个 GE 多模光网口组成的高速网络处理板卡为实验平台. IXP2400 的工作主频为 600MHz，内部包含 8 个微引擎(ME)和 1 个 XScale 处理器. 微引擎支持 32 位 RISC 指令集，每个微引擎又包含 8 个有独立寄存器组的硬件线程，线程轮流获得执行权限且线程之间切换为零开销. 每个硬件线程可以看做一个 PE.

IXP2400 片外存储单元主要包括：(1) 工作频率最高为 200MHz 的双通道 QDR SRAM，每个通道的峰值带宽为 1.6GB/s、最大容量 64MB，支持不包括原子减操作在内的各种原子操作；(2) 工作频率最高为 150MHz 的 DDR SDRAM(4 个 Bank)，峰值带宽为 2.4GB/s，最大容量 2GB.

在原型系统中，散列表、会话记录写锁组成的位串(bitmap)、回收记录缓冲区头尾指针等存储在 SRAM 中，会话记录本身则根据所属子表分别存储

于 SDRAM 的不同 Bank 中. 采用源 IP 地址、源 TCP/DUP 端口、目的 IP 地址、目的 TCP/DUP 端口和协议号作为会话关键码，散列表的散列算法采用 jhash 算法[27].

对原型系统中各项操作的开销(读 SRAM 次数，写 SRAM 次数；读 SDRAM 次数，写 SDRAM 次数；操作所含指令数，操作的时钟周期开销)进行了统计，见表 3，其中创建会话记录操作的平均开销达到了 2920 个时钟周期，要远大于其它操作的时间开销.

### 6.1 实验设置

通过若干台流量发生器向实验平台发送 64 字节大小的 UDP 报文，总发包速率为 $\lambda_1$. 其中，以均匀的时间间隔发送带新的目的 IP 地址和 UDP 端口的报文，发送速率为 $\lambda_2$，以这样的报文作为会话首报文，触发原型系统执行创建添加会话记录的操作；其余的报文采用轮询方式分别通过不同的 UDP 会话发送出去，每条会话的生命周期持续 $T_d$ 秒.

通过调整流量发生器的参数 $\lambda_1$、$\lambda_2$、$T_d$ 以及原型系统的参数 $L$、$T_a$、$S_r$，观察在不丢包条件下系统处于稳态时的 $n_1$、$n_2$、$N$ 和 $GR$，并周期性地记录 $L^q_{RR}$、$L^q_{SR}$ 和 $S_f$ 的变化，从而对报文吞吐并行加速性能、会话吞吐并行加速性能以及存储开销(队列长度、回收缓冲区大小)进行评估.

### 6.2 报文并行处理性能

**实验 1.** 报文到达速率 $\lambda_1$ 处于高位时的报文并行加速性能.

将 $\lambda_2$、$L$、$E(T_d)$、$T_a$、$S_r$ 分别取值为 1 万条/s、524288、300s、60s、1 个，则稳态下会话表中的活跃会话记录数 $N_a$ 应为 360 万条. 实际测得会话表中的会话记录总数 $N$ 约为 362.5 万条，垃圾比率 $GR$ 为 0.69%，散列桶深均值 $E(d)$ 为 14；所需 MM PE(硬件线程)数 $n_2$ 为 1 个.

定义报文处理的并行效率(parallel efficiency)=

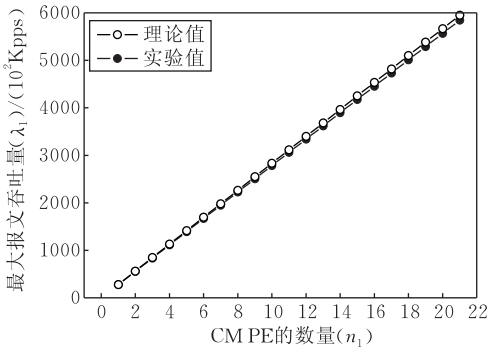
稳态下 $n_1$ 个 CM PE 的最大报文吞吐量/

($n_1$ × 稳态下单个 CM PE 的最大报文吞吐量) (35)

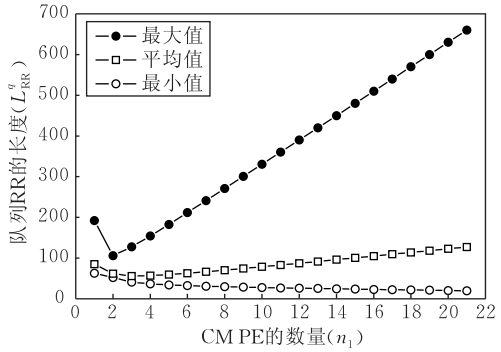
测得此时单个 CM PE(硬件线程)的最大报文吞吐量约为 278.3Kpps. 逐步增加 CM PE(硬件线程)的数量 $n_1$，测试对应的最大报文吞吐量以及报文输入队列 RR 长度变化情况. 受限于实验平台硬件本身的吞吐能力(峰值流量为 4Gbps)，CM PE(硬件线程)总数只能增加到 21 个，约占 IXP2400 硬件线程总数的三分之一.

图 7(a)给出了在不丢包情况下最大报文吞吐量随 CM PE(硬件线程)数目 $n_1$ 的变化情况. 由图中可知，最大报文吞吐量与 $n_1$ 呈线性增长关系，且与理论值能较好地吻合，报文处理的并行效率均值接近 1. 当 $n_1=1$ 时，最大报文吞吐量达到 278.3Kpps，对比表 1 中 Netfilter Conntrack 的实验结果可知，在硬件平台性能较逊的情况下，PaSeM 的该项性能指标仍然超过 Netfilter Conntrack 一个数量级. 当 $n_1=21$ 时，最大报文吞吐量达到了 5.844Mpps(约 3.93Gbps).

图 7(b)给出了报文输入队列 RR 的长度 $L^q_{RR}$ 的变化情况. 随着 $\lambda_1$ 和 $n_1$ 的增加，$L^q_{RR}$ 的均值呈缓慢上升，$L^q_{RR}$ 的最小值基本上保持不变且略有下降，而 $L^q_{RR}$ 的最大值呈明显上升趋势，反映出在所有 CM PE 接近满负载的状况下等待被各个 CM PE 处理的报文在队列 RR 中产生“累积”，使得 $L^q_{RR}$ 的变化幅度变大. 当报文吞吐量达到 5.844Mpps，接近系统流量负载峰值时，$L^q_{RR}$ 的最大值达到 660($n_1=21$)，仍然在比较合理的水平上.



(a) 最大报文吞吐量



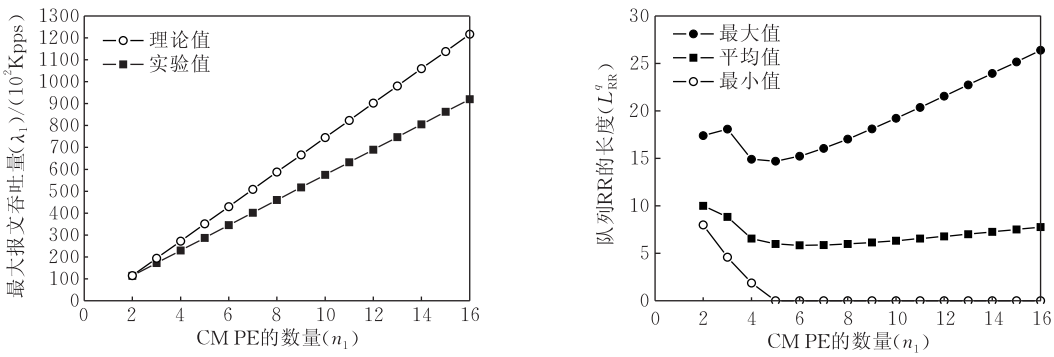
(b) 队列RR长度(CM PE满负载情况下)

图 7 报文吞吐并行加速性能和排队队列长度变化情况(会话速率 10000 条/s)

**实验 2.**　会话到达速率 $\lambda_2$ 处于高位时的报文并行加速性能.

将 $\lambda_2$ 调整为 5 万条/s，其它条件与实验 1 相同，则稳态下会话表中的活跃会话记录数 $N_a$ 应为 1800 万条. 实际测得会话表中的会话记录总数 $N$ 约为 1907 万条，垃圾比率 $GR$ 约为 5.9%，散列桶深均值 $E(d)$ 为 73；所需 MM PE(硬件线程)数 $n_2$ 为 2 个.

测得此时 2 个 CM PE(硬件线程)的报文吞吐量约为 114Kpps(单个硬件线程的吞吐量已经小于 $\lambda_2$). 由于此时会话表的负载因子(load factor) $N/L$ 较高，导致查询会话表的 I/O 访存开销增加，从而使得最大报文吞吐量明显降低. 将 CM PE(硬件线程)的数量 $n_1$ 从 2 逐步增加到 16，测得最大报文吞吐量的变化情况如图 8(a)所示，报文输入队列 RR 的长度 $L^q_{RR}$ 的变化情况则如图 8(b)所示. 由图中可知，最大报文吞吐量与 $n_1$ 依然呈现线性增长关系，当 $n_1=16$ 时，最大报文吞吐量达到了 920Kpps；报文处理的并行效率均值仍接近 1. 当 $n_1$ 为 2 时，尽管此时会话首报文数量 $\lambda_2$ 在总流量中占据了相当高的比例，达到 43.85%，最大报文吞吐量的实验值与理论值仍然能够较好地吻合；另一方面，由于总流量 $\lambda_1$ 相对于 $n_1$ 来说并不高，$L^q_{RR}$ 的变化幅度也相应地比较小，且依然呈现随着 $n_1$ 逐步递增的趋势——这两点都表明 CM PE 对报文的处理并没有受到 MM PE 的干扰，体现出 PaSeM 无冲突并行加速的效果是比较明显的. 当 $n_1$ 从 2 增加到 16 时，会话首报文数量 $\lambda_2$ 在总流量中的比例逐步下降到了 5.43%，但是最大报文吞吐量的实验值与理论值之间的差距反而逐步扩大到 24%. 这主要是由于理论分析模型并没有将 I/O 访存带宽的限制考虑进来，因此当 I/O 访存开销存在瓶颈时理论值产生了一定的偏差.




(a) 最大报文吞吐量　　(b) 队列RR长度(CM PE满负载情况下)

图 8　报文吞吐并行加速性能和排队队列长度变化情况(会话速率 50000 条/s)

### 6.3　会话并行处理性能

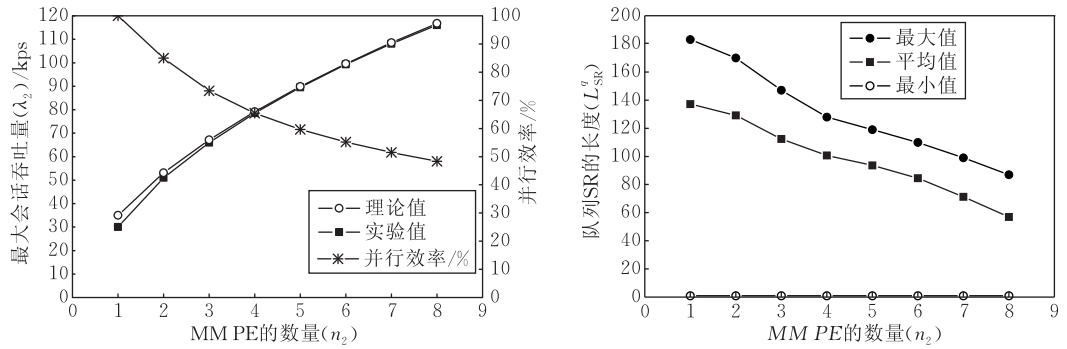
**实验 3.**　将 $L$、$E(T_d)$、$T_a$、$S_r$ 分别取值为 524288、300s、60s、1 个，报文发送速率 $\lambda_1$ 固定为 900Kpps. 逐步增加 MM PE(硬件线程)的数量 $n_2$，测试对应的最大会话吞吐量、并行效率以及队列 SR 长度变化情况.

定义会话处理的并行效率(parallel efficiency)=

稳态下 $n_2$ 个 MM PE 的最大会话吞吐量/

($n_2$ ×稳态下单个 MM PE 的最大会话吞吐量)　(36)

测得此时单个 MM PE(硬件线程)的最大会话吞吐量 $\lambda_2$ 约为 3 万条/s. 将 MM PE(硬件线程)的数量 $n_2$ 从 1 逐步增加至 8，则对应的最大会话吞吐量、并行效率的变化情况如图 9(a)所示. 由图中可以看出，最大会话吞吐量与 $n_2$ 呈递增关系，且实验值与理论值能较好地吻合. 当 $n_2=1$ 时，最大会话吞吐量达到 3 万条/s，对比表 1 中 Netfilter Conntrack 的实验结果可知，在硬件平台性能较逊的情况下，PaSeM 的该项性能指标仍然达到了 Netfilter Conntrack 的 3.64 倍. 当 $n_2=8$ 时，最大会话吞吐量达到了 11.6 万条/s，会话表中的会话记录总数达到 4470 多万条，其中活跃会话约为 4176 万条，已经达到相当于 100+Gbps 网络流量的会话到达速率和并发会话规模[5]. 当 $n_2=2,3,4,5,6,7,8$ 时，并行效率分别为 85%、73.3%、65.4%、59.7%、55.2%、51.5%和 48.3%，取得了较好的会话并行加速效果. 随着 $n_2$ 的增加，MM PE 的并行效率呈递减趋势，这主要是由于会话表负载因子 $N/L$ 的增长导致查询会话表的 I/O 访存开销增加，反过来限制了最大会话吞吐量 $\lambda_2$ 的进一步增长. 图 9(b)给出了队列 SR 的长度 $L^q_{SR}$ 的变化情况. 由图中可知，当 $n_2=1$ 时 $L^q_{SR}$ 最大值达到 183，之后随着 $n_2$ 的增加，$L^q_{SR}$ 及其变化幅度均呈下降趋势.




(a) 最大会话吞吐量与并行效率 (b) 队列SR长度(MM PE满负载情况下)

图 9 会话吞吐并行加速性能和排队队列长度变化情况(报文速率 900Kpps)

### 6.4 存储开销

对实验 3 中稳态下的会话表规模 $N$ 进行记录,并根据式(22)可算出同时活跃的会话数 $N_a$,由此可得会话表中过期会话记录以及垃圾比率 GR 的变化情况,如图 10 所示.由图中可知,在系统处于最大会话吞吐量的情况下,GR 仍然处于 9%以下,低于文献[5]的指标(28%).

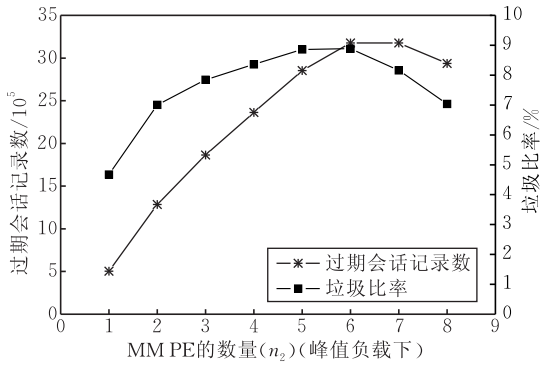



图 10 过期会话记录数与垃圾比率

对于实验 3 中队列 RR 和 SR 的长度 $L^q_{RR}$ 和 $L^q_{SR}$ 进行记录并求均值,再通过 Little 公式和式(34)可计算出回收缓冲区大小 $S_f$ 的条件值,如图 11 所示.采用此条件值为 $S_f$ 的实验值,在实验过程中没有发生会话丢失的现象,表明计算出的条件值是合理的.

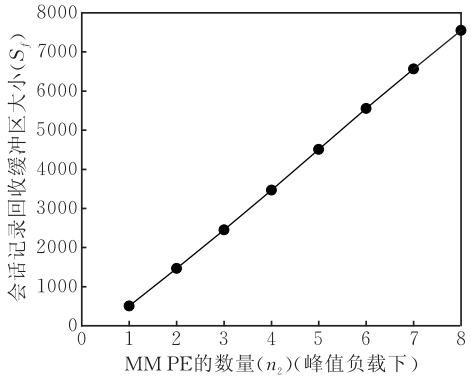



图 11 回收缓存区大小($S_f$)的条件值

随着会话吞吐量 $\lambda_2$ 的增加,$S_f$ 也需要相应地增加,当 $\lambda_2$ 达到 11.6 万条/s 时,$n_2$ 和 $N$ 分别为 8 和 4470 万条,此时 $S_f$ 只需设置为 7548 即可满足要求,仅为 $N$ 的万分之 1.69.

## 7 结论与展望

本文的工作主要体现在如下几个方面:(1)提出了一种基于并行的网络流量会话管理方案 PaSeM,基于散列表的无锁会话表设计,采用了任务并行、报文并行、会话并行、I/O 并行和异步通信等多种并行策略;(2)讨论并解决了在高速网络环境下面临的各种并行冲突问题,给出了高效的会话管理和查询的并行算法;(3)基于排队理论建立了相应的性能评价模型并进行了理论分析,对于稳态下的性能指标以及关键参数应满足的条件给出了定量计算方法,与这部分类似的理论建模和分析工作还只见于与文献[5].该模型的理论计算值与实验值有较高的吻合度,对于实际应用以及今后的研究具有指导意义.(4)实现了一个 PaSeM 的原型并进行了实验,实验结果表明 PaSeM 对于会话管理和报文处理具有较好的并行加速效果,报文处理的并行效率均值接近 1,当会话管理单元个数为 4 时,会话处理并行效率为 65.4%(亦即加速比为 2.62),当会话管理单元个数为 8 时,会话处理并行效率仍然达到了 48.3%(加速比为 3.86),能够满足当前高速网络环境流量处理的性能要求;在最大吞吐量负载下排队队列长度及其变化幅度都处于合理范围,会话表垃圾比率维持在较低的水平上(实验结果为小于 9%),与已有的工作相比为优.

与已有的研究工作相比,PaSeM 在保证业务灵活性的前提下实现了对会话和报文的高速无冲突并行处理,具备很好的实用性.针对并行执行环境中大

规模网络会话管理问题，根据已有文献来看，这方面的研究工作还开展得比较少.

进一步的方法改进和研究工作包括：(1)考虑基于松耦合分布式并行处理系统(如电信级防火墙或者运营商级流量监控系统)以及异构并行系统的应用场景，将 PE 之间通信信道的不可靠性因素考虑进来，对会话记录状态机和并行算法进行扩展，以满足 10+Gbps～100+Gbps 高速带宽下的性能要求；(2)考虑在专用硬件支持下(如 TCAM、Bloom filter、Hash 硬件)的方法改进，以获得局部性能的改善，如提高散列表的查询命中率、减少查询失败开销.

## 参 考 文 献


[1] Guo F L, Chiueh T. Traffic analysis: From stateful firewall to network intrusion detection system. Stony Brook University, Technical Report TR-164, 2004

[2] Cranor C, Johnson T et al. Gigascope: A stream database for network applications//Proceedings of the ACM SIGMOD. San Diego, CA, 2003: 647-651

[3] Dreger H, Feldmann A et al. Dynamic application-layer protocol analysis for network intrusion detection//Proceedings of the 15th USENIX Security Symposium. Vancouver, B. C., Canada, 2006: 257-272

[4] Claffy K, Braun H, Polyzos G. A parameterizable methodology for Internet traffic flow profiling. IEEE Journal on Selected Areas in Communications, 1995, 13(8): 1481-1494

[5] Xu J, Singhal M. Cost-effective flow table designs for high-speed routers: Architecture and performance evaluation. IEEE Transactions on Computers, 2002, 51(9): 1089-1099

[6] NetFlow Services Solutions Guide. 2005, from: http://www. cisco. com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite. htm

[7] Estan C, Varghese G. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. ACM Transactions on Computer Systems, 2003, 21(3): 270-313

[8] Brownlee N, Mills C, Ruth G. Traffic Flow Measurement: Architecture. RFC 2722, 1999

[9] Sekar V, Reiter M K et al. CSAMP: A system for network-wide flow monitoring//Proceedings of the NDSI. San Francisco, CA, 2008: 233-246

[10] Estan C Keys K et al. Building a better NetFlow//Proceedings of the ACM SIGCOMM. Portland, OR, 2004: 416-427

[11] Handelman S, Stibler S et al. RTFM: New attributes for traffic Flow measurement. RFC 2724. 1999

[12] Netfilter, from: http://www. netfilter. org/

[13] Paxson V. Bro: A system for detecting network intruders in real-time. Computer Networks, 1999, 31(23-24): 2435-2463

[14] Roesch M. Snort — Lightweight intrusion detection for networks//Proceedings of the 13th USENIX Conference on Systems Administration. Seattle, WA, 1999: 229-238

[15] Ptacek T H, Newsham T N. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical Report, Secure Networks, Inc., 1998

[16] Mai J, Chuah C N. Is sampled data sufficient for anomaly detection?//Proceedings of the IMC. Rio de Janeriro, Brazil, 2006: 165-176

[17] Che H, Li S Q. MPOA flow classification design and analysis//Proceedings of the INFOCOM. New York, NY, 1999: 1497-1504

[18] Song H Y, Dharmapurikar S et al. Fast Hash table lookup using extended bloom filter: An aid to network processing//Proceedings of the SIGCOMM. Philadelphia, PA, 2005: 181-192

[19] Estan C, Varghese G, Fisk M. Bitmap algorithms for counting active flows on high speed links//Proceedings of the IMC. Miami Beach, FL, 2003: 153-166

[20] Kumar A, Xu J et al. Space-code bloom filter for efficient per-flow traffic measurement//Proceedings of the INFOCOM. New York, 2004: 1762-1773

[21] Broder A, Mitzenmacher M. Network applications of bloom filters: A survey. Internet Mathematics, 2003, 1(4): 485-509

[22] Bonomi F, Mitzenmacher M et al. Beyond bloom filters: From approximate membership checks to approximate state machines//Proceedings of the SIGCOMM. Pisa, Italy, 2006: 315-326

[23] Chang F, Feng W C, Li K. Approximate caches for packet classification//Proceedings of the IEEE INFOCOM. Hong Kong, 2004: 2196-2207

[24] Chang F, Feng W C et al. Efficient packet classification with digest caches//Proceedings of the HPCA NP3 Workshop. Madrid, Spain, 2004: 13-24

[25] Lakshman T V, Stidialis D. High-speed policy-based packet forwarding using efficient multi-dimensional range matching//Proceedings of the ACM SIGCOMM. Vancouver, Canada, 1998: 203-214

[26] Tung Y, Che H. A flow caching mechanism for fast packet forwarding. Computer Communications, 2002, 25(14): 1257-1262

[27] Hash Functions for Hash Table Lookup. 1997, from: http://www. burtleburtle. net/bob/hash/evahash. html

[28] Dharmapurikar S, Paxson V. Robust TCP stream reassembly in the presence of adversaries//Proceedings of the USENIX Security Symposium. Berkeley, CA, 2005

[29] Barakat C, Thiran P et al. Modeling Internet backbone traffic at the flow level. IEEE Transactions on Signal Processing, 2003, 51(8): 2111-2124

[30] Bose S K. An Introduction to Queueing Systems. Kluwer Academic/Plenum Publishers, 2001

[31] Köllerström J. Heavy traffic theory for queues with several servers: I. Journal of Applied Probability, 1974, 11(3): 544-552

[32] Brumelle S L. Some inequalities for parallel-server queues. Operations Research, 1971, 19(2): 402-413

[33] Sheng You-Zhao. Queueing Theory and Its Application in Computer Communications. Beijing: Beijing University of Posts and Telecommunications Press, 1998: 134(in Chinese)
(盛友招. 排队论及其在计算机通信中的应用. 北京: 北京邮电大学出版社, 1998: 134)

[34] Mori T, Uchida M, Goto S. Flow analysis of Internet traffic: World wide web versus peer-to-peer. Systems and Computers in Japan, 2005, 36(11): 70-81

[35] Tian Nai-Shuo. Vacation Random Service System. Beijing: Peking University Press, 2001(in Chinese)
(田乃硕. 休假随机服务系统. 北京: 北京大学出版社, 2001)

**ZHANG Jian-Yu**, born in 1977, Ph. D. candidate. His research interests include network and information security.

**ZHOU Yuan**, born in 1972, Ph. D., senior engineer. His research interests include information security and IPv6.

**ZOU Wei**, born in 1964, professor, Ph. D. supervisor. His research interests include network and information security.

**Background**

One of the most important challenges faced by the session-based network traffic processing systems, such as traffic monitoring, stateful firewall, intrusion prevention, network address translation(NAT), load balancing, etc., is how to achieve real-time, efficient session management while network traffic and sessions have led to an explosive growth in high-speed network environments. Practical systems are usually to meet service flexibility at the expense of performance, while previous studies have generally focused on performance, likely to give up the processing flexibility, and often too much emphasis on the performance of local operations (such as search hit rate). Therefore, this paper focuses on how to both optimize the performance of session management via parallel mechanisms in parallel execution environments (such as multi-core /multi-threaded processors, software multi-process /multi-threaded) and mantain good service flexibility. Few studies in existing literature focus on this issue. This paper proposes an approach of parallel conflict-free network session management, and analyzes its performance based on the queuing model of $G/G_2/n_1$ and M/G+D/1 with exhaustive service and multiple vacations. The authors also implement a prototype and make some experiments. The results show that the approach has high parallel efficiency for packet and session processing. This work was supported by the National High Technology Research and Development Program (863 Program) of China under grant No. 2006AA01Z410, the National Science Foundation of China under grant Nos. 60873217, 60973159, National Development and Reform Commission Project of China under grant Nos. [2009]1717, CNGI-09-01-12 and the Development Fund for Electronic and Information Industry of China under grant No. [2007]329.