ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.12, December 2007, pp.3115−3123
DOI: 10.1360/jos183115

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

基于软件的网络处理器的路由高速缓存算法研究*

刘 祯+, 刘 斌, 郑 凯

(清华大学 计算机科学与技术系,北京 100084)

Software-Based Route Cache Algorithm for Network Processors

LIU Zhen+, LIU Bin, ZHENG Kai

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62773441, Fax: +86-10-62773616, E-mail: liuzhen02@mails.tsinghua.edu.cn

Liu Z, Liu B, Zheng K. Software-Based route cache algorithm for network processors. *Journal of Software*, 2007,18(12):3115−3123. http://www.jos.org.cn/1000-9825/18/3115.htm

Abstract: Routers require fast and flexible route table lookup for incoming packets at relatively low cost. This paper describes a software-based route cache algorithm for network processors. Part of the on-chip high-speed memory space is allocated and programmed into a caching table for temporal storage of route lookup results. A suitable hash function can make a good balance between cache miss rate and update complexity, which shortens the average search time, reduces the contentions on memory bus and leaves more headroom for other network applications. Experiments with real-life packet traces show that the packet throughput of a network processor can be greatly improved with only a small number of route cache entries per processing element.

Key words: route cache; network processor; route lookup

摘 要: 路由器需要以较低的代价灵活、高速地实现路由查找这一基本功能.为网络处理器设计了一种基于软件的路由查找高速缓存算法.网络处理器片上高速存储器中的一部分空间被划分出来,由指令代码来维护一个路由查找结果缓存表.通过选择合适的哈希函数,平衡表项之间的冲突并刷新复杂度,该算法可以缩短路由查找的延迟,减少多处理单元对存储器总线的竞争,为其他网络应用提供更多的处理时间.基于真实网络流量的实验表明,即便每个处理单元中仅有少量表项,网络处理器的吞吐量仍然可以得到有效的提升.

关键词: 路由高速缓存;网络处理器;路由查找

中图法分类号: TP393 **文献标识码**: A

网络处理器(network processor,简称 NP)是一种对数据包进行灵活、高速处理的新型转发引擎,主要应用于路由器等网络设备中.网络处理器通常将多个基于精简指令集(reduced instruction set computer,简称 RISC)的处理单元(processing element,简称 PE)集成在同一块芯片上,并具有丰富的存储层次.人们往往认为网络应用中不

* Supported by the National Natural Science Foundation of China under Grant Nos.60373007, 60573121 (国家自然科学基金); the China/Ireland Science and Technology Collaboration Research Fund under Grant No.CI-2003-02 (中国-爱尔兰政府间国际科技合作项目); the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No.20040003048 (高等学校博士学科点专项科研基金); the Cultivation Fund of the Key Scientific and Technical Innovation Project under Grant No.705003 (高等学校科技创新工程重大项目培育资金项目)

Received 2006-04-21; Accepted 2006-11-03

具备足够的局部性,且数据包处理需要确定的时间性能,因此,大多数网络处理器没有 cache 结构.数据在不同存储器件中的分配和移动完全由编程人员控制.

路由器的基本功能之一是对数据包中的目的 IP 地址(destination IP address,简称 DIP)进行路由查找并根据查找结果转发数据包.有些网络处理器设置专用的查找协处理器,如 IBM PowerNP 的 Tree Search Engine[1].还有一些网络处理器侧重于保证较高的编程灵活性,缺乏对路由查找等基本操作的优化,如 Intel IXP 系列、Cisco Toaster2 等.尽管网络处理器可以依赖 TCAM 等吞吐量高且访问延迟固定的器件来完成路由查找,但这类器件价格昂贵、功耗很大,不利于网络处理器在中低速设备中的应用.因此,如何充分利用网络处理器自身的结构来提高路由查找的性能显得更为重要.

路由查找算法经过了较长时间的研究,已经比较成熟.Gupta 等人设计了一种简单、高效并且适合硬件实现的 multi-bit trie 算法.他们根据路由表中前缀长度的分布,将 trie 分为长度不等的若干级,以达到转发表所占空间和访问次数的折衷[2].例如,绝大多数 IP 地址前缀都不大于 24 位,由此可将 trie 分为长度为 24 和 8 的两级(记为 24-8 算法);此时,路由转发表占用的空间多达几十 MB,但是大多情况下只需要 1 次路由表的访问.

目前,大部分网络处理器都支持较大的地址空间,因此,路由转发表所占的空间对于网络处理器来说不是一个难题.然而,由于多处理单元共享存储器所导致的排队和调度等因素,网络处理器访问外部存储器件的延迟通常较大.尽管网络处理器提供了多线程等机制用来隐藏访问延迟,但网络流量中较高的突发性,往往使多线程不能很好地发挥作用[3].即便有足够多的线程投入使用,存储器件的访问延迟也会因访问请求数目的增加而进一步恶化.因此,只有减少外部存储器的访问次数,才能有效地提高网络处理器的路由查找能力.

本文针对网络处理器通常都具有低延迟的片上存储空间的结构特点,提出了一种路由查找高速缓存算法,并以 Intel IXP2800 为例,介绍了它的具体实现.

1 网络处理器的存储器层次结构

IXP2800 是 Intel 公司继 IXP1200 后的第二代网络处理器,可以实现从 OC-3 到 OC-192 的数据包处理功能[4].图 1 显示了 IXP2800 的硬件结构及存储层次的主要部分.IXP2800 集成了 16 个微引擎(microengine,简称 ME),每个微引擎支持 8 个线程,并含有一个 640 字(每字 32 位)的 Local Memory.所有的微引擎构成两个集群(cluster),通过命令总线共享片内的 Scratchpad Memory、片外的 SRAM 和 DRAM.表 1 列出了这几类存储器件的特点和典型用途,其中,cycle 以微引擎的工作频率 1.4 GHz 来衡量.

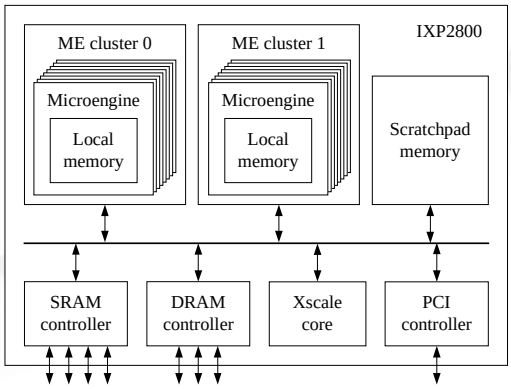


Fig.1 Block diagram of IXP2800 and its major components in memory hierarchy

图 1 IXP2800 的结构图及其存储层次的主要部分

Table 1 Characteristics of the primary memory types in IXP2800

表 1 IXP2800 主要存储器件类型的特点

Memory type	Addressable data unit	Maximum size	On chip?	Access time (cycle)		Typical use
				Read	Write	
Local memory	4B	640×4B	Yes	5	5	Caching data needed by ME
Scratchpad	4B	16KB	Yes	100	40	Processing state or shared data
SRAM	4B	64MB	No	130	53	Packet header or smaller data structure
DRAM	16B	2GB	No	295	53	Packet storage or large data structure

2 路由查找高速缓存算法

本文提出的路由查找高速缓存算法,将片上高速存储器的一部分空间划分出来,由软件来维护一个路由查找结果缓存表(route cache),同时,在片外低速存储器中用一些简单、高效但耗用空间较大的算法建立路由表.对于 IXP2800 来说,每个微引擎中的 Local Memory 就可用来建立缓存表;每个路由查找结果缓存表的表项对应一个目的 IP 地址.当一个目的 IP 地址收到后,使用该 32 位的 IP 地址进行一次哈希运算,得到一个索引号(index),用于对缓存表进行检索.检查被检索到的表项的内容,如果与目的 IP 地址相匹配,就将表项中保存的查找结果返回;否则引发一个缺失,并完整地使用路由查找算法对保存在 DRAM 中的路由表进行检索.路由查找的结果除了返回给应用程序外,还要写回路由查找结果缓存表,供后来的目的 IP 地址使用.此外,在实际使用的网络中,路由的变更是比较频繁的.路由器会接收到大量路由刷新消息(route update message),这些消息由控制平面综合分析后,把最终的修改通知路由表和路由查找结果缓存表.

2.1 表项的构成及其组织方式

IXP2800 的 Local Memory 按照字组织,它与微引擎的通信也以字为单位.为了节省访问开销,可将表项的大小限制为一个字.实际网络中路由变更频繁,表项可能会在某次路由刷新后变得无效,这就需要一个标志位来表示该表项是否有效.尽管路由查找的结果最终表现为下一跳 IP 地址,但同一个路由器所连接的不同主机不可能太多,因此可以将它们简单地映射为一个用若干位即可表示的端口号(port).表项内剩下的空间就可以保存对应的目的 IP 地址信息,称为标签(tag),用以区分经过哈希运算以后映射到同一个表项的不同目的 IP 地址.

由于标志位和端口号已经占用了部分比特,并非所有的目的 IP 地址字段都可以保存在表项内,因此在对目的 IP 地址进行哈希运算的过程中,需要有一定数量的比特保持不变,并以索引的形式反映出来.这也意味着路由查找结果缓存表至少要维持一定数量的表项,才能保证目的 IP 地址信息不丢失.例如,端口号用 7 位来表示,则该网络处理器可以支持 128 个不同的下一跳 IP 地址.此时加上标志位,表项内共有 8 位被占用,则该路由查找结果缓存表至少需要维持 2^8=256 个组;并且相应的哈希算法也需要从目的 IP 地址中选取 8 位,作为索引号的一部分.标志位、标签、端口号这 3 部分在表项中的相对位置可以自由地设定.例如在图 2 中,每个表项的低 7 位被用作端口号,最高位用作标志位,中间的 24 位用作标签.

2.2 哈希函数的选择

哈希函数的选择是路由查找高速缓存算法中比较关键的问题.它一方面决定了目的 IP 地址在缓存表中的分布是否均匀,即目的 IP 地址映射到缓存表时冲突的概率是否足够小,从而降低路由表访问发生的次数;另一方面,由于每个数据包都要经过这样的运算,使得它位于数据包处理的关键路径上,直接决定了返回路由查找结果所需的时间.尽管 IXP2800 提供了一个哈希部件,但它的运算过于复杂,不适合用在操作数仅有 32 位且需要较低延迟的路由查找结果缓存算法中.因此,这里的哈希运算只能由基本指令来完成.

为了使目的 IP 地址在路由查找结果缓存表中均匀分布,对于在哈希运算中需要保持不变的目的 IP 地址字段,应该选择那些被置为“1”和“0”的概率相近的字段.文献表明,目的 IP 地址的高位部分(即网络地址部分)比低位部分(即主机地址部分)更为稳定,即 DIP[15..0]比 DIP[31..16]更富于变化;并且由于目前绝大多数地址前缀都为 24 位,但并非每个这样的子网都能保证有足够多的主机,DIP[15..8]的分布比 DIP[7..0]更为均匀[5].从这些分析来看,DIP[15..8]附近的字段是最适合于用来保留的.此外,对于表项数目足够多的缓存表,其索引号中并非直

接目的IP地址中提取的部分,则可以由整个目的IP地址或者其中的一部分经过诸如异或操作等运算得到.异或操作除较为简单外,还可以在4种不同的操作数组合中,产生2个“1”和2个“0”,即产生了一个均匀的分布.这是其他操作,如与、或等所不能达到的.哈希运算完全由软件来实现,可以根据实际的流量进行调整,从而使缓存表的命中率达到最大.

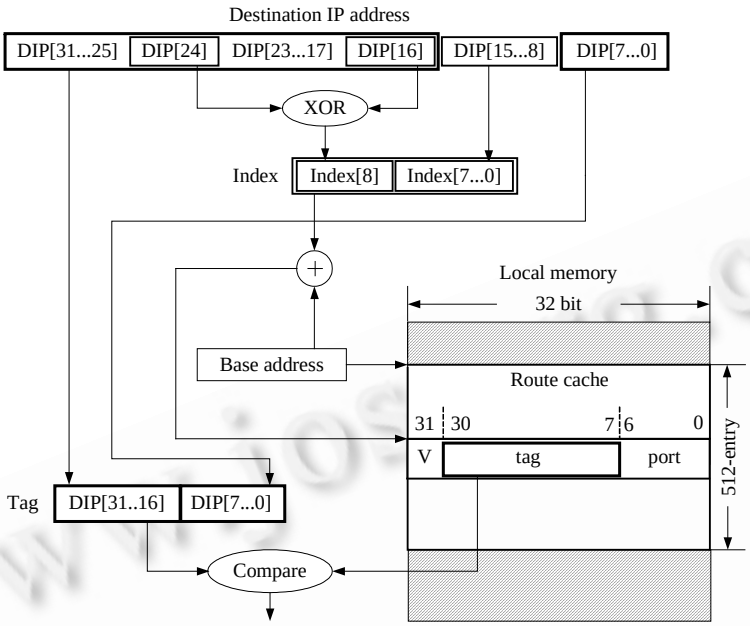


Fig.2 A possible implementation of route cache entry and Hash function

图2 一种可能的路由缓存表表项构成以及哈希函数的实现

图2除了显示一种路由查找结果缓存表表项构成方法以外,还提供了一种与该构成方法相应的可能的哈希函数计算方法.其索引号的最高位(即Index[8])由DIP[24]和DIP[16]经过异或产生,低位部分由DIP[15...8]直接提取得到.整个哈希函数可以表示为

$$Index[8]=DIP[24]\ XOR\ DIP[16],$$

$$Index[7..0]=DIP[15..8].$$

相应地,除DIP[15...8]以外,目的IP地址中的其他部分,即DIP[31..16]和DIP[7..0]被连接起来作为标签.

2.3 路由查找结果缓存表的刷新

路由刷新消息通常以前缀的形式到达路由器,而路由查找结果缓存表的表项记录的是目的IP地址,因此,一个前缀可能会映射到多条表项.理想的缓存表刷新算法是逐个检查前缀所映射到的所有表项,并将匹配的表项作废.这种做法占用大量的存储器带宽,且耗时较长,只能用于路由不经常更新且链路速率较低的情形.一种与该做法相反的算法是在路由更新发生时作废所有的表项,这样虽然软件开销不大,但如果路由更新频繁并且表项较多,则缓存表大多数时间都处于未被写满的状态,降低了算法的有效性.因此,可以采用一种折衷的办法,即将所有映射到的表项不加检查地直接作废.

哈希函数决定了每个前缀映射的表项个数,从而影响到更新复杂度.从减少受路由表刷新影响的表项数目的角度来说,选择目的IP地址的高位部分用来进行哈希运算是比较有利的.图3和图4显示了从Route Views项目得到的两个路由刷新消息数据集的前缀长度分布[6].

这两个数据集分别记录了2003年11月1日和2004年3月1日,路由器route-views2.oregon-ix.net所收到的15分钟内的BGP路由刷新通告消息(BGP update advertisement).从中可以看到,超过50%的更新地址前缀要长于24位.因此,选择不同于DIP[7...0]的字段都可以大幅度地缩减可能受到路由刷新影响的缓存表表项数目.

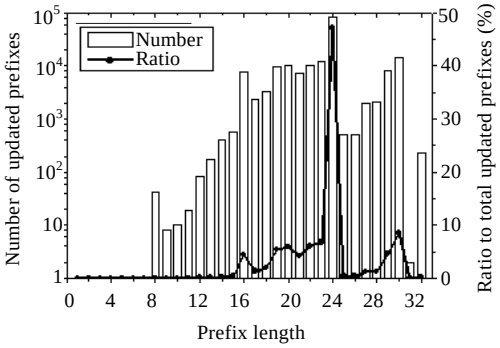


Fig.3 Prefix length distribution for dataset 1

图 3 数据集 1 的前缀长度分布

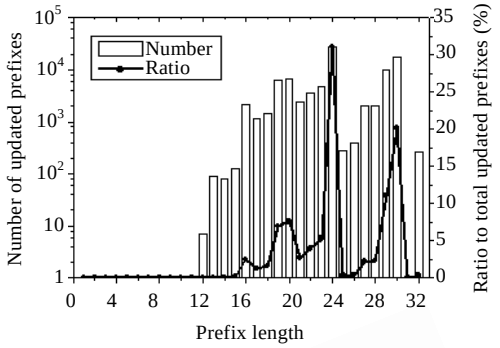


Fig.4 Prefix length distribution for dataset 2

图 4 数据集 2 的前缀长度分布

2.4 路由查找高速缓存算法在IXP2800中的具体实现

针对网络应用的特点对指令集进行优化,是网络处理器中经常采用的技术.网络处理中频繁出现的操作被提取出来,由专用的硬件或者指令来完成,可以缩短程序的长度,节省处理的时间.经常在网络处理器中出现的特殊指令,如位操作等,就能极大地简化路由查找高速缓存算法中的哈希运算以及标签生成.

图 5 显示了使用 IXP2800 中的算术-移位指令(alu_shf)和字节加载指令(ld_field/ld_field_w_clr)[7]来实现的图 2 中的哈希函数以及标签构造算法.相对于未经优化的指令集(如 MIPS 等),支持位操作的指令集只用少数几条指令就可以轻松地完成诸如字段的提取以及移位等操作.在程序开始执行前,寄存器 a_0 中保存的是目的 IP 地址,寄存器 a_1 中保存操作数 0x00000100.该程序段的详细解释如下:

a) 指令 A.0 将目的 IP 地址右移 16 位后与操作数 0x00000100 进行与操作,将 DIP[24]提取出来,保存在寄存器 b_0 中;指令 A.1 则将 DIP[16]提取出来保存在寄存器 b_1 中同样的位置.指令 A.2 将 DIP[24]和 DIP[16]进行异或操作,形成索引号的最高位 Index[8],并保存在寄存器 b_0 中.指令 A.3 将目的 IP 地址右移 8 位,即将 DIP[15...8]移动到最低一个字节,然后位掩码 0001 掩码(每个位掩码对应一个字节,位掩码为 1 表示需要提取该字节)将这个字节提取出来,写入寄存器 b_0 的相应位置,同时保持 b_0 其他部分不变,形成最终的索引号.

b) 指令 B.0 通过使用字节掩码 1100,将目的 IP 地址的高 16 位,即 DIP[31...16]提取出来,写入寄存器 b_2,同时将寄存器 b_2 的其他部分置为 0;指令 B.1 将目的 IP 地址左移 8 位,并通过位掩码 0010 将目的 IP 地址的低 8 位,即 DIP[7...0]提取出来,连接在寄存器 b_2 中 DIP[31...16]的后面,形成标签.

```
A.0: alu_shf [b0,a1,AND,a0,>>16]        B.0: ld_field_w_clr [b2,1100,a0,>>0]
A.1: alu_shf [b1,a1,AND,a0,>>8]         B.1: ld_field [b2,0010,a0,<<8]
A.2: alu [b0,b0,XOR,b1]
A.3: ld_field [b0,0001,a0,>>8]
```

(a) Hash function (b) Tag construction

(a) 哈希函数 (b) 标签构造

Fig.5 Example implementation of part of route cache algorithm using optimized instructions of IXP2800

图 5 使用 IXP2800 的优化指令集来部分地实现路由高速缓存算法的实例

3 性能评价

3.1 对网络处理器吞吐量的影响

为了对路由查找高速缓存算法的有效性进行评价,本文使用了 4 个不同的数据包 trace 来对 IXP2800 路由查找的过程进行模拟.这些数据包 trace 由 NLANR 提供,截取自真实的网络流量,涵盖了从边缘、接入到骨干网的各种链路速率[8].表 2 列出了这些数据包 trace 的主要特点.由于这里考察的是路由查找高速缓存算法的处理

能力,我们将链路中的时间戳修改为最大可能实现的数据包到达速率.注意到,现实中的网络往往遵循“过度提供(over provision)”的原则,大部分链路的负荷率都较低,并且边缘链路的负荷高于核心链路.因此,修改后的链路负荷远高于实际情况,造成数据包间局部性的提高,会对实验结果有一定的影响.

Table 2 Major characteristics of packet traces

表 2 数据包 trace 的主要特点

Name	Site description	Link	Bandwidth	Date
BWY	Columbia university (BroadWaY)	OC-3	155 Mbps	September 15th, 2004
UFL	University of florida at gainesville	OC-12	622 Mbps	December 5th, 2004
Abilene-I	Links at IPLS router node	OC-48	2.5 Gbps	March 8th, 2006
Abilene-III	Link between Indianapolis router node and Kansas city	OC-192	10 Gbps	June 1st, 2004

我们使用 NePSim 作为基础的模拟器,并使用微指令编写了相应的代码[9].为了提高路由查找结果缓存表的命中率,我们将具有相同目的 IP 地址的数据包分配给同一个线程.每个微引擎中的 Local Memory 仅有 640 个字,因此,我们仅对缓存表中有 64,128,256 和 512 个表项,以及没有使用路由查找高速缓存算法的配置进行了模拟.路由查找算法采用的是 16-8-8 的 multi-bit trie 方法.

图 6 是不同数目的微引擎以及路由查找结果缓存表表项下网络处理器的吞吐量.表 3 和图 7 分别显示了单微引擎配置下缓存表的缺失率和微引擎的利用率;表 4 和图 8 分别显示了四微引擎配置下缓存表的平均缺失率和微引擎的平均利用率.

Table 3 Route cache miss rate for single ME

表 3 单微引擎配置下的路由缓存缺失率

Trace	Route cache entries			
	64 (%)	128 (%)	256 (%)	512 (%)
BWY	30.97	22.71	15.31	9.42
UFL	55.95	46.46	37.21	27.02
Abilene-I	43.70	35.40	26.42	19.70
Abilene-III	55.38	45.76	37.67	28.30

Table 4 Average route cache miss rate for four-ME

表 4 四微引擎配置下的平均路由缓存缺失率

Trace	Route cache entries per ME			
	64 (%)	128 (%)	256 (%)	512 (%)
BWY	22.12	16.57	11.86	7.98
UFL	36.67	29.39	23.31	17.67
Abilene-I	23.93	19.61	15.41	12.05
Abilene-III	36.06	29.32	23.63	18.02

从图 6 中可以看出,当不使用路由查找结果缓存表时,网络处理器的吞吐量与数据包 trace 基本无关.而路由查找结果缓存表的效果,则依赖于数据包 trace 中目的 IP 地址之间的局部性.一般情况下,数据包 trace 中携带的目的 IP 地址数目越多,缓存表的命中率越低.无论链路的速率如何,即便是使用了一个很小的缓存表,相对于没有使用的情况,网络处理器的吞吐量都会有很大的提高.对于增长幅度最小的 UFL,当缓存表中有 64 个表项,并且仅有一个微引擎时,吞吐量仍然由 17.70 Mpps 增加到 22.60 Mpps,提高了 27.71%.

由于 IXP2800 工作在比较高的频率,并且线程数目较多且大,访问片外 DRAM 时会引发较长的延迟.由图 7 可知,当没有使用路由查找结果缓存表时,很多情况下,整个微引擎内的所有线程都会处于等待 DRAM 返回结果的挂起状态(suspend),即微引擎处于空闲(idle)状态,从而浪费了微引擎的处理能力.在图 8 中,当使用的微引擎数目增多时,则出现由于访存请求队列已满而导致的迟延(stall)状态.即便在缺失率比较高的情况下(比如 Abilene-III 的缺失率达到 55.38%),访问低延迟的 Local Memory 也会显著地减少 DRAM 访问的次数,从而缩短微引擎中空闲和迟延状态,提高处理资源的使用效率.

另一方面,随着微引擎数目的增加,路由查找高速缓存算法的效果更加明显.随着访存请求的增加,存储器总线的竞争加剧,导致访存延迟急剧增加.从图 7 和图 8 的对比中可以看出,当微引擎的数目从 1 增加到 4 时,每个微引擎的有效工作时间却缩短到一半左右.在图 6 中,从微引擎的数目为 8 开始,微引擎数目的增加已经不能抵偿每个微引擎工作效率的减少,网络处理器的吞吐量开始下降.而对于路由查找结果缓存表来说,多个微引擎的分流,使得进入每个缓存表的不同目的 IP 地址的数目有所下降,缓存表的命中率也相应地得到提高.尽管每个微引擎之间负担的目的 IP 地址数目并不均衡,但 DRAM 访问次数的进一步减少以及单个微引擎工作效率的提高,使得网络处理器的吞吐量得到更加有效的提升.从图 6 中可以看到,使用缓存表后,网络处理器的吞吐量随着微引擎数目的增加,呈现近乎线性的增长.

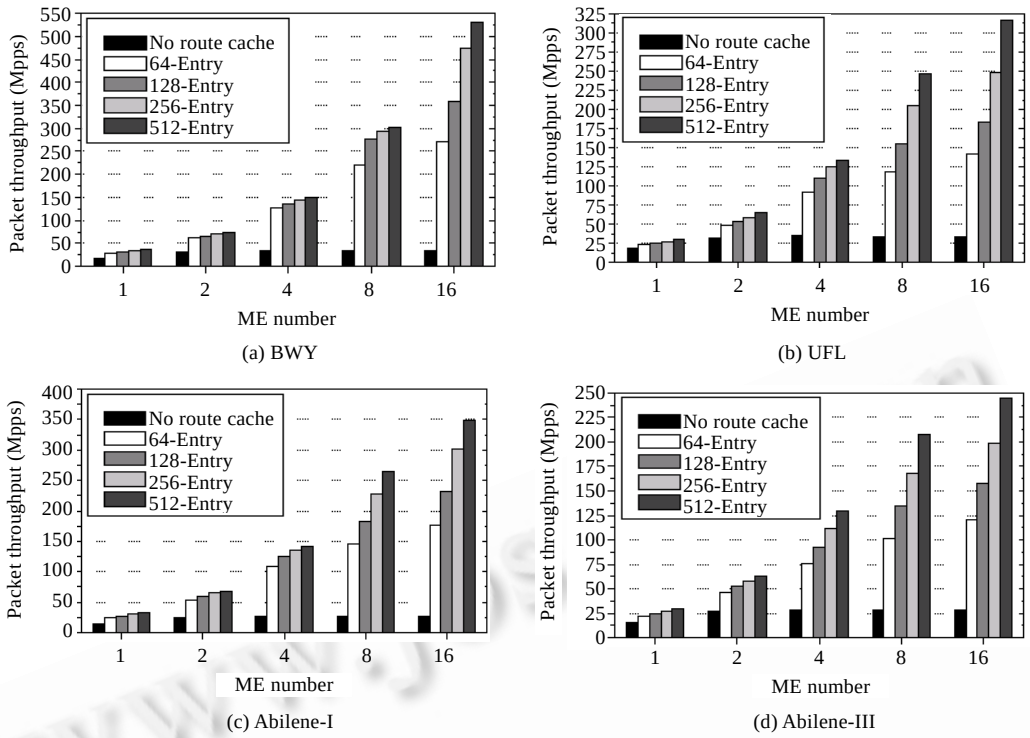


Fig.6 Packet throughput of NP under different number of MEs and route cache entries

图 6 不同数目的微引擎以及路由查找结果缓存表表项下网络处理器的数据包的吞吐量

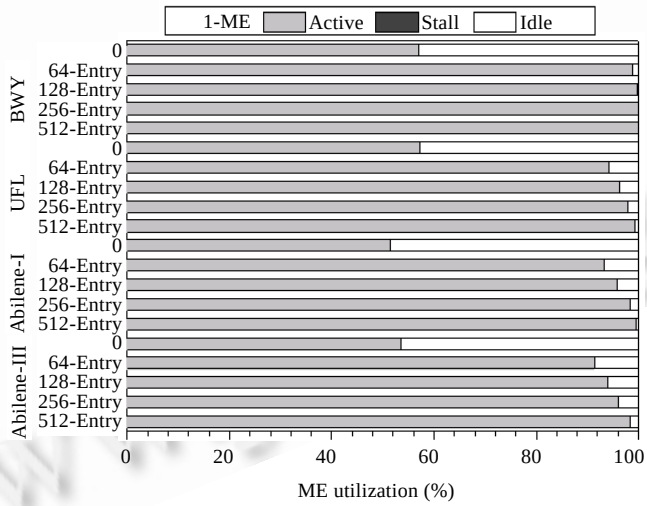


Fig.7 Utilization rate for single ME

图 7 单微引擎配置下的 ME 使用率

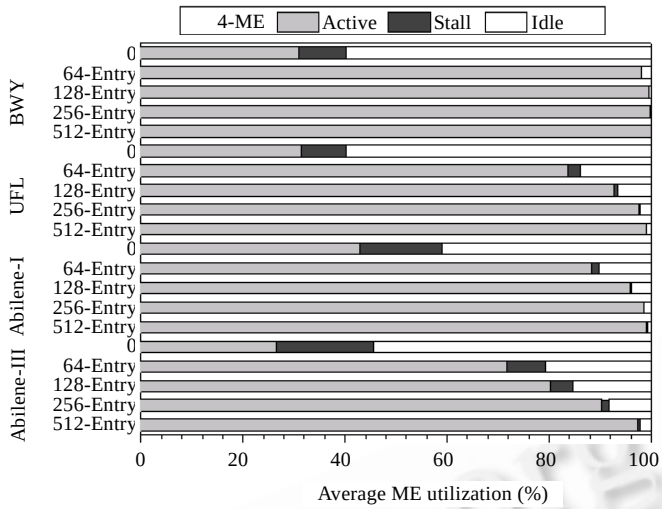


Fig.8 Average utilization rate for four-ME

图 8 四微引擎配置下的平均 ME 使用率

3.2 刷新的影响

表 5 列出了图 3 和图 4 中的两个数据集作用在路由查找结果缓存表时,不同哈希函数下需要作废的平均表项数目.这些哈希函数均为直接从目的 IP 地址中提取相应数目的连续比特,而表 5 的第 1 列给出了这些比特的最低位.从中可以看到,当最低位为 DIP[10]时,每次刷新仅有 5 个左右的表项需要作废;尽管当最低位为 DIP[7]的时候有超过 10 个表项需要作废,但是表项数目越多,相对而言受影响表项的比例越低.

Table 5 Average invalidated route cache entries for each update

表 5 每次刷新时需要作废的平均路由缓存表表项数

Index ending bit	Dataset 1				Dataset 2			
	64-Entry	128-Entry	256-Entry	512-Entry	64-Entry	128-Entry	256-Entry	512-Entry
DIP[7]	13.23	18.60	26.89	40.06	12.42	16.05	21.24	28.34
DIP[8]	9.381	13.53	20.11	22.00	8.209	10.80	14.36	15.23
DIP[9]	7.081	10.37	11.32	12.40	5.741	7.517	7.952	8.466
DIP[10]	5.537	6.010	6.551	7.039	4.125	4.342	4.599	4.879
DIP[11]	3.386	3.657	3.901	4.139	2.557	2.685	2.825	2.845

4 相关工作

在早期的路由器设计中,路由查找操作的设计多集中在如何利用传统的 cache 结构[10].Chiueh 等人提出了一种应用于通用处理器的路由查找结果缓存算法,其 cache 机制结构复杂、容量较大,具有多种配置和映射方式,在网络处理器中通常是不具备的[11].

随着网络设备的设计向 ASIC(application specific integrated circuit)发展,一些称为 route cache 专用的硬件模块开始出现.Tzi-cker Chiueh 等人为网络处理器设计了一种称为 Intelligent Host Address Route Cache 的结构,可将多个不相邻的 IP 地址前缀均匀地映射到同一个表项,从而提高命中率[12].Liu 提出将 TCAM 嵌入到网络处理器中,从而可以直接使用 IP 地址前缀进行查找[13].Rajan 等人为 Level Compressed trie 设计了一种分段式的 cache,不同的段用来缓存不同类型的节点,提高 cache 的使用效率[14].但是,这些硬件无法在已经制成产品的网络处理器中使用.而现有网络处理器的开发主要集中于网络安全、QoS 保证等复杂应用,缺乏对路由查找等基本操作的讨论[15,16].

5 结 论

本文描述了一种灵活并且性能较高的基于网络处理器的路由查找高速缓存算法.该算法在网络处理器的片上高速存储器中划分出一部分空间,由软件来维护一个路由查找结果缓存表.算法的关键在于哈希函数的设计,以期达到提高缓存表命中率与降低刷新复杂度的平衡.缓存算法的具体实现,可以利用网络处理器提供的针对网络应用进行过优化的指令来减少代码的长度,缩短程序执行的时间.采用从真实网络上截取下来的数据包 trace 作为输入,对路由查找算法进行模拟得到的实验结果表明,即便每个处理单元中仅有少量表项,网络处理器的吞吐量也可以得到大幅度的提升.

References:

[1] Peyravian M, Davis G, Calvignac J. Search engine implications for network processor efficiency. IEEE Network, 2003,17(4): 12−20.

[2] Gupta P, Lin S, McKeown N. Routing lookups in hardware at memory access speeds. In: Charny A, ed. Proc. of the 17th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'98). San Francisco: IEEE Communications Society, 1998. 1240−1247.

[3] Liu H. A trace driven study of packet level parallelism. In: Proc. of the IEEE Int'l Conf. on Communications (ICC 2002). New York: IEEE Computer Society Press, 2002. 2191−2195.

[4] Intel IXP2800 network processor hardware reference manual. 2003. http://www.intel.org

[5] Talbot B, Sherwood T, Lin B. IP caching for terabit speed routers. In: Proc. of the IEEE Global Telecommunications Conf. (Globecom'99). Rio de Janeiro: IEEE Computer Society Press, 1999. 1565−1570.

[6] Route views project, University of Oregon. 2004. http://www.routeviews.org

[7] Intel IXP2400 and IXP2800 network processor programmer's reference manual. 2005. http://www.intel.org

[8] National Laboratory for Applied Network Research. Passive measurement and analysis. 2006. http://pma.nlanr.net/

[9] Luo Y, Yang J, Bhuyan LN, Zhao L. NePSim: A network processor simulator with a power evaluation framework. IEEE Micro, 2004,24(5):34−44.

[10] Partridge C, Carvey PP, Burgess E, Castineyra I, Clarke T, Graham L, Hathaway M, Herman P, King A, Kohalmi S, Ma T, Mcallen J, Mendez T, Milliken WC, Pettyjohn R, Rokosz J, Seeger J, Sollins M, Storch S, Tober B, Troxel GD, Waitzman D, Winterble S. A 50-Gb/s IP router. IEEE/ACM Trans. on Networking, 1998,6(3):237−248.

[11] Chiueh T, Pradhan P. High-Performance IP routing table lookup using CPU caching. In: Proc. of the 18th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'99). New York: IEEE Communications Society, 1999. 1421−1428.

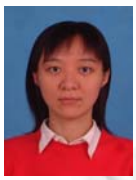
[12] Chiueh T, Pradhan P. Cache memory design for Internet processors. IEEE Micro, 2000,20(1):28−33.

[13] Liu H. Routing prefix caching in network processor design. In: Proc. of the 10th Int'l Conf. on Computer Communications and Networks (ICCCN 2001). Scottsdale: IEEE Press, 2001. 18−23.

[14] Rajan K, Govindarajan R. A heterogeneously segmented cache architecture for a packet forwarding engine. In: Proc. of the Int'l Conf. on Supercomputing (ICS 2005). Boston, New York: ACM Press, 2005. 71−80.

[15] Tan Z, Lin C, Yin H, Li B. Optimization and benchmark of cryptographic algorithms on network processors. IEEE Micro, 2004, 24(5):55−69.

[16] Lin YD, Lin YN, Yang SC, Lin YS. DiffServ edge routers over network processors: Implementation and evaluation. IEEE Network, 2003,17(4):28−34.



刘祯(1980－),女,山东兖州人,博士生,主要研究领域为路由查找与流分类算法,网络处理器系统结构.

郑凯(1978－),男,博士生,主要研究领域为路由查找与流分类算法,网络安全.

刘斌(1964－),男,博士,教授,博士生导师,主要研究领域为交换与调度理论,核心路由器设计,网络处理器设计,网络安全.