可扩展路由器 FIB 表分解存储模型

陈文龙[1),2)] 徐明伟[3)] 杨 扬[2)] 韩 冬[4)]

[1)](首都师范大学信息工程学院 北京 100048)

[2)](北京科技大学计算机与通信工程学院 北京 100083)

[3)](清华大学计算机科学与技术系 北京 100084)

[4)](北京大学软件与微电子学院 北京 100080)

摘 要 FIB 表急剧增长是互联网高速发展面临的重要问题之一,FIB 表分解存储能有效解决该问题. 现有的 SPAL 技术将 FIB 表较均匀地分解存储在不同线卡,但仍然存在较多的表项冗余存储现象,并且实现复杂. 对此设计了一种新型的转发表分解存储模型(Decomposed Storage of FIB,DSF),它依据 IP 前缀的前若干 bit 位实现线卡对转发表的分解存储,并只带来极少的冗余存储. DSF 的改进方案——EDSF,更可使各线卡非常均衡地完成分解存储. 提出的分解存储模型缓解了 FIB 表项急剧增长问题的解决压力,同时大大节省了硬件资源. 对于线卡数量更多的可扩展路由器尤为适合. 通过对当前运营的路由表的分解存储实验研究及与其它方案的比较,验证了文中模型良好的存储性能.

关键词 路由器;路由;转发表;分解存储;IP 前缀

中图法分类号 TP393 **DOI 号**: 10.3724/SP.J.1016.2011.01611

Decomposed Storage Model of FIB for Cluster Router

CHEN Wen-Long[1),2)] XU Ming-Wei[3)] YANG Yang[2)] HAN Dong[4)]

[1)](*Information Engineering College, Capital Normal University, Beijing* 100048)

[2)](*School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing* 100083)

[3)](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

[4)](*School of Software and Microelectronics, Peking University, Beijing* 100080)

Abstract The fast increasing of FIB has put unprecedented pressure to the Internet routers. The decomposed storage of FIB has theretofore been widely suggested, which explores the balance of storage overhead over routers' line cards. SPAL can achieve the decomposed storage of FIB, but its implement is very complex, and there exists many redundancy storage of FIB. In this paper, we propose a novel decomposed storage model of FIB, DSF(Decomposed Storage of FIB), which can reduce the unnecessary redundancy of FIB storage across line cards. Moreover, we discuss an enhanced mechanism: EDSF. The real-FIB experiments show that EDSF can further minimize the diversity of line cards' storage. The design and the implementation of our proto type is also validated in detail showing that the proposed architecture is compatible with the existing commercial core-routers.

Keywords router; route; FIB; decomposed storage; IP prefix

收稿日期:2010-07-15;最终修改稿收到日期:2011-08-06. 本课题得到国家"九七三"重点基础研究发展规划项目基金(2009CB320502)、国家"八六三"高技术研究发展计划项目基金(2009AA01Z251)、国家科技支撑计划项目基金(2008BAH37B03)和国家自然科学基金(60873192)资助.**陈文龙**,男,1976 年生,博士,讲师,主要研究方向为路由器体系结构、网络协议. E-mail: chenwenlong2008@gmail.com; wenlongchen@sina.com. **徐明伟**,男,1971 年生,博士,教授,主要研究领域为网络体系结构、高速路由器体系结构和协议测试. **杨 扬**,男,1955 年生,博士,教授,主要研究领域为网络通信、图像处理. **韩 冬**,男,1989 年生,硕士研究生,主要研究方向为计算机网络.

1 引 言

互联网快速发展带来的一个重大问题就是核心路由器 FIB 表的急剧增长[1]. 我们需要在路由器中部署更大容量的路由查找及存储芯片. 而且,商用路由器普遍采用全冗余备份方法存储 FIB 表项,该方法使得每块线卡(Line Card,LC)都要大量扩充硬件资源. 随之带来的芯片供电和散热处理还会消耗更多的电力资源. 随着路由器体系向可扩展结构的发展[2],该问题显得更为突出. FIB 表分解存储是解决上述问题的重要手段. 现有 FIB 表研究[3-4]主要集中于通过并行处理提高查询速度,只有文献[5]提出的 SPAL 技术是一个完备的 FIB 表分解存储方案,但它仍存在大量转发项在多块线卡的冗余存储.

本文设计了转发表分解存储方法(Decomposed Storage of FIB,DSF). 它依据 IP 前缀的前若干 bit 位进行转发表分解存储,实现线卡对 FIB 表的部分存储. 每线卡只需存储系统转发表的一个子表,且每线卡平均存储转发项数量随着线卡数增加而减少. 模型只需为每块线卡增加存储极少虚拟路由,与真实转发项共存于一个转发表中. 线卡转发引擎通过最长前缀匹配(Longest Prefix Matching,LPM),既可实现转发下一跳查询,又可实现报文到转发子表的对应. 针对真实网络中路由前缀分布不均衡特征,DSF 模型的改进方案 EDSF,能使各线卡存储的转发项数非常均衡. DSF/EDSF 模型可实施于所有进行最长前缀匹配的分布式转发体系. 而且,针对该模型设计的线卡功能结构与现有商业路由器线卡设计兼容,易实施. 理论分析及对实际运营路由器 FIB 表的分解结果都表明它能极大地减少整个系统及每线卡存储的 FIB 表项.

本文提出的分解存储模型较之 SPAL 技术,能实现更为平均的分解存储以及更少的冗余存储. 其主要特点包括:(1)各线卡分解存储非常均衡,使得线卡所需的最大存储容量变小;(2)只需增加存储极少的虚拟路由即可实现;(3)方案适用范围广:所有遵循 LPM 的分布式路由转发体系;(4)与现有商用路由器兼容,我们只需对现有商用路由器线卡功能做极小的改动,便可完成 DSF 模型的实施,易推动其产业化进程.

本文第 2 节介绍相关研究工作;第 3 节介绍 DSF 模型;第 4 节介绍 DSF 模型实施算法及线卡功能结构;第 5 节介绍优化的 DSF 模型——EDSF;第 6 节为实验及结果分析;第 7 节总结全文.

2 相关研究

文中"核心路由表"是指系统对各路由协议所学路由计算后的最佳路由,"转发表(FIB)"是指存储在线卡上指导报文转发的信息. 传统模型中,每块线卡的转发表都是核心路由表的映像. 然而,该描述在本文提出的新型模型中并不成立. 另外,"真实路由"或"路由"也表示核心路由,它对应 DSF 模型中的"虚拟路由"描述.

近年来学者们研究了如何通过并行路由查询,提高转发引擎查询速度. 文献[3]针对 Trie 树路由存储提出了一种基于 SRAM 的多管道并行 IP 查询体系,通过缓存常用路由来均衡各管道流量. 文献[4]设计了一种基于分块路由并行查询的报文转发体系:IFPLUT. 它根据转发出接口将转发表分成若干子部分. 下一跳查询时,所有子部分并行进行路由查询,所有结果中 IP 前缀长度最大的路由即为报文转发的依据. 文献[3-4]的主要贡献是提高了 FIB 表查询效率,虽没有实现 FIB 表分解存储,但确为分解存储提供了有价值的参考.

文献[5]为解决 FIB 表扩张问题设计的 SPAL 技术是真正的 FIB 表分解存储方案. SPAL 根据目的 IP 前缀中任意 n 位的值将转发表分解成若干子集,分别存储在不同线卡. 线卡分析报文目的地址该 n 位值,确定其转发项所在线卡,并向该线卡进行转发查询获得结果. 虽然 SPAL 缓存查询结果,但新到流量转发项查询时涉及两次线卡间消息传递,增加了转发时延. 而且,缓存维护(新项插入、删除等维护)行为带来很大开销. 最重要的,该方法基于 SRAM 实现且硬件逻辑实现复杂,与现有商业核心路由器实现不兼容. 另外,该方案中仍会出现较多的转发项冗余存储.

路由器已向可扩展体系结构发展[2]. 可扩展路由器是由若干个子路由器级连而成的一个统一的路由系统. 它在功能、性能、接口规模等方面,具有极大的可扩展性. 其主要优点有:保护前期运营投资,提高系统性能,增强可靠性,简化网络拓扑等. 主要的路由器厂商已开发出相应产品[6-7]. 可扩展路由器的一个重要特征是线卡数量急剧增多,所以 FIB 表容量增大导致的消耗更多硬件资源及电能的问题在可扩展路由器中显得更为突出.

3 DSF 模型

商用核心路由器通常为分布式体系结构,普遍

采用的转发表存储方法是每块线卡存储所有核心路由表项[6-8]，称为完全备份存储模型(Full Backup Storage，FBS). 假设可扩展路由器系统的核心路由表项数量为 m，线卡总数为 n，满足 $n>1$(本文不考虑系统仅有一块线卡的情况). 传统 FBS 模型中，系统存储的转发项总数为 $m\times n$. 随着路由器路由数量逐渐增大，设备存储代价及查找开销都难以承受. 而且，因为 IPv6 路由占用更多存储空间，IPv6 网络的部署将加剧这一矛盾. 本文的 DSF 模型正是为解决这一矛盾而设计的.

3.1　分解存储

定义 1.　对于两个给定 IP 前缀 PFX_1：$Address_1/Masklen_1$；PFX_2：$Address_2/Masklen_2$，若满足 $Masklen_2\geq Masklen_1$，且前缀地址 $Address_1$ 和 $Address_2$ 的二进制前 $Masklen_1$ 位完全相同，则称作 PFX_2 归属 PFX_1，或 PFX_1 包含 PFX_2，记作：$PFX_2\prec PFX_1$. 如有 192.0.0.0/2≺128.0.0.0/1、10.0.0.0/8≺10.0.0.0/8.

定义 2(单元前缀 PFX_{unit}).　单元前缀是分解存储中依据某种策略指定的一些 IP 前缀. 单元前缀具有下述特性：目的 IP 前缀归属同一单元前缀的若干路由项，存储在同一线卡，一块线卡可拥有多个单元前缀.

定义 3(聚集前缀 PFX_{aggr}).　包含多个不同单元前缀的 IP 前缀被称作聚集前缀.

令所有可能的单元前缀集合为 S_{unit}，所有可能的聚集前缀集合为 S_{aggr}. DSF 模型中，单元前缀和聚集前缀的指定方法分别通过规则 1 和规则 2 完成.

规则 1.　对于给定整数 $k(0<k<32)$，所有掩码长度等于 k 的 IP 前缀被指定为单元前缀. k 被称作分解位，有 $|S_{\text{unit}}|=2^k$，并令 $PFX_{\text{unit}}(i)$ 为前 k 位二进制数等于 i 的单元前缀.

规则 2.　对于给定整数 $k(0<k<32)$，掩码长度小于 k 的 IP 前缀被指定为聚集前缀.

以 $k=2$ 为例进一步说明. 此时单元前缀个数为 4，包括 $PFX_{\text{unit}}(0)$=“0.0.0.0/2”，$PFX_{\text{unit}}(1)$=“64.0.0.0/2”，$PFX_{\text{unit}}(2)$=“128.0.0.0/2”，$PFX_{\text{unit}}(3)$=“192.0.0.0/2”；而“128.0.0.0/1”就是一个聚集前缀. 所有前缀长度大于等于 2 的 IP 前缀，总归属于 4 个单元前缀中的某一个. 我们将任一 IP 地址视为掩码长度为 32 的 IP 前缀，则任一 IP 地址总是属于且只属于某一单元前缀.

DSF 模型根据线卡数 n 来确定分解位 k 的取值，满足

$$k=\lceil\log_2 n\rceil \tag{1}$$

DSF 模型的单元前缀和聚集前缀都是根据满足式(1)的 k 值指定而得. 以 4 块线卡为例，n 等于 4 则 k 等于 2，即根据 IP 前缀的前两位进行路由表的分解存储. 那么，共有 4 个单元前缀，每块线卡分得一个单元前缀. IP 前缀归属“0.0.0.0/2”的路由项存储在 LC_0 中，IP 前缀归属“64.0.0.0/2”的路由项存储在 LC_1 中，以此类推. 对于目的前缀为聚集前缀(掩码长度小于 2)的路由项，会在所有线卡存储. 当然，路由系统中的线卡数可以为任意值，系统总是以满足式(1)的 k，进行单元前缀的指定. 所以，单元前缀的个数为

$$|S_{\text{unit}}|=2^k=2^{\lceil\log_2 n\rceil} \tag{2}$$

进一步可得

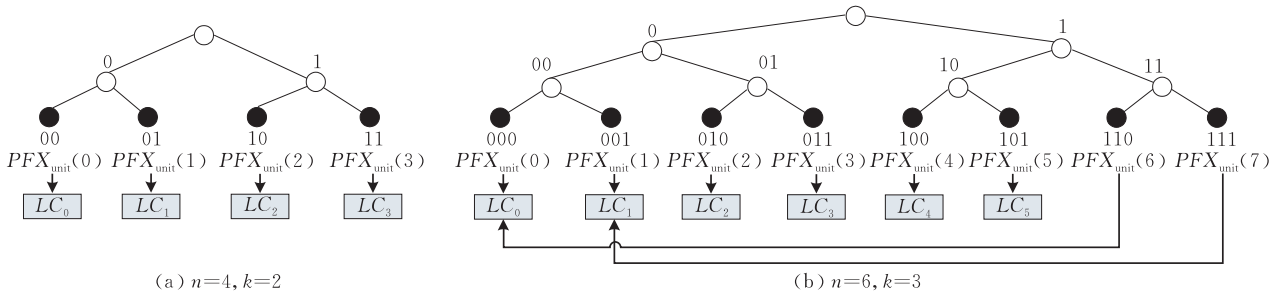
$$\begin{cases}|S_{\text{unit}}|=n, & \text{当 } n \text{ 等于 2 的指数}\\ n<|S_{\text{unit}}|<2n, & \text{当 } n \text{ 不等于 2 的指数}\end{cases} \tag{3}$$

根据均分原则分配单元前缀. 由式(3)可得每块线卡拥有 PFX_{unit} 数为 1 或 2，线卡会存储 IP 前缀归属其任一单元前缀的路由项.

定义 4.　路由系统中，若第 i 个单元前缀 $PFX_{\text{unit}}(i)$ 被分配给第 j 块线卡 LC_j，记作 $PFX_{\text{unit}}(i)\sqsubseteq LC_j$.

为了描述方便，假设线卡标号为 0、1、2 依次递增，则 DSF 方案中线卡与单元前缀的分配关系为式(4). 图 1 所示分解树以线卡数等于 4 和 6 为例说明了单元前缀的分配方法.

$$\begin{cases}PFX_{\text{unit}}(i)\sqsubseteq LC_i, & i<n\\ PFX_{\text{unit}}(i)\sqsubseteq LC_{(i-n)}, & i\geq n\end{cases} \tag{4}$$



(a) $n=4, k=2$　　(b) $n=6, k=3$

图 1　$n=4$ 和 $n=6$ 时单元前缀分配示例

单元前缀分配后，就可实施对 FIB 表的分解存储. 一条路由的 IP 前缀，要么归属某个单元前缀，要么本身就是一个聚集前缀. 对于系统任一条核心路由项，其存储方法遵循规则 3.

规则 3. 令核心路由项的目的 IP 前缀为 PFX_{dst}，若 PFX_{dst} 为聚集前缀，则通告该路由项给所有线卡存储；否则，该 IP 前缀必归属某个单元前缀，则只通告给某块线卡 LC_j 存储，满足 $PFX_{dst} \prec PFX_{unit}(i)$ 和 $PFX_{unit}(i) \sqsubseteq LC_j$.

分解存储过程总结如下. 首先，根据系统线卡数 n 及式(1)计算得分解位 k. 接着，根据 k 值及规则 1 得到所有单元前缀. 最后，依据规则 3 对核心路由表进行分解存储.

3.2 虚拟路由及数据转发

FIB 表在各线卡分解存储后，DSF 模型实施的另一关键则是数据层基于此存储规则的报文转发机制. DSF 模型对处理报文的线卡进行角色定义. 针对系统转发的某个 IP 报文，目的地址归属的单元前缀所在线卡，被称作该报文的宿主线卡，记作 LC_{attach}. 以 4 块线卡为例，单元前缀“64.0.0.0/2”属于 LC_1，则 LC_1 是目的地址为 64.0.0.1、65.10.0.1、100.0.0.1 等报文的宿主线卡. 另外，定义 LC_{in} 为接收该报文的线卡，LC_{out} 为发送该报文的线卡.

DSF 模型中报文转发思想是：对于待转发 IP 报文，总是在其宿主线卡进行 LPM 查找时才能获得最终转发信息，包括查找失败也在宿主线卡中决策. 当报文的接收线卡不是其宿主线卡时，通过增加虚拟路由并按 LPM 查询机制将报文发向宿主线卡. 也就是说，线卡接收报文后只需进行 LPM 处理，若接收线卡就是报文的宿主线卡则可直接获得最终转发信息进行转发处理，否则发送报文到它的宿主线卡再次进行 LPM 查找. DSF 方案用简化、统一的硬件 LPM 查询，集成了报文处理的两个重要功能：获取最终转发信息及获取宿主线卡信息.

一条转发项的描述包括以下主要字段：〈目的网段，出线卡号，出接口号，下一跳地址〉. 对任一单元前缀 $PFX_{unit}(i)$，若其分配给 LC_j，系统会构造一条目的网段为该单元前缀的虚拟路由：〈$PFX_{unit}(i)$，j，Inv，Inv〉(Inv 表示一个无效值)，并通告该路由给除 LC_j 外的所有线卡存储. 这样做的目的是，当除 LC_j 外的其它线卡收到目的地址属于 $PFX_{unit}(i)$ 的报文，LPM 查找后必定会匹配该条虚拟路由，从而转发报文到 LC_j. 仍以 4 块线卡为例，单元前缀“64.0.0.0/2”属于 LC_1，则归属“64.0.0.0/2”的路由全部存储在 LC_1 中. 并且，系统会在 0、2、3 号线卡增加一条虚拟路由：〈“64.0.0.0/2”，1，Inv，Inv〉. 若 LC_0 或 LC_2 或 LC_3 收到目的地址为 64.0.0.1 的报文，LPM 查找必匹配该虚拟路由，从而转发报文到 LC_1. 每个单元前缀对应着一条虚拟路由，而每条虚拟路由会存储到($n-1$)块线卡上，由式(2)可得路由器系统总共增加存储的虚拟路由数为

$$((n-1)\times 2^{\lceil \log_2 n \rceil}) \tag{5}$$

总结每块线卡存储的 FIB 表包括：(1) 所有聚集前缀路由项；(2) IP 前缀归属本线卡所拥有单元前缀的路由项；(3) 其它线卡的单元前缀对应的虚拟路由. 其中，前 2 类路由都是系统真实存在的核心路由项.

真实路由及虚拟路由全部存储进 FIB 表后，各线卡就可通过对报文进行 LPM 处理实现快速转发. 根据接收报文的角色不同，可以分为两种情况：

(1) 非宿主线卡接收. LC_{in} 经过 LPM 查找后必定匹配某条虚拟路由，查找结果中出线卡就是该报文的 LC_{attach}，且得到的出接口号和下一跳都是无效值. 报文通过内部交换网络到达 LC_{attach} 后，LC_{attach} 发现边带信息中出接口和下一跳是无效值，于是再次进行 LPM 查找，若查询失败则丢弃报文. 否则，必匹配某条真实路由，并得到全部转发信息. 若最终出线卡就是 LC_{attach}，即 LC_{attach} 与 LC_{out} 角色重合，则直接发送；否则，通过交换网络发往最终的 LC_{out}，LC_{out} 发现边带信息已包含所有有效转发信息，则发送报文.

(2) 宿主线卡接收. 此时报文的 LC_{in} 与 LC_{attach} 角色重合，报文所需的转发项只可能存储在接收线卡. LC_{in} 进行 LPM 查找，若查询失败则丢弃报文；否则，必匹配某条真实路由，并得到全部有效转发信息. 若最终出线卡就是 LC_{in}，即 LC_{in}、LC_{attach} 与 LC_{out} 三种线卡角色重合，则直接发送；否则，通过交换网络发往最终的 LC_{out}，LC_{out} 发现边带信息已包含所有有效转发信息，发送报文.

DSF 模型的报文转发过程中，板间转发次数由两个条件决定，条件 1：$LC_{in} = LC_{attach}$；条件 2：$LC_{attach} = LC_{out}$，而 LC_{in} 与 LC_{out} 的关系并不影响板间转发次数. 条件 1 表示线卡从外部接收的报文的目的地址所需路由在入线卡；条件 2 表示转发时报文出线卡恰好存储了报文转发所需真实路由. DSF 模型中，上述两个条件都成立则需 0 次板间转发，任一条件成立则需 1 次板间转发，都不成立则需 2 次板间转发.

3.3 无效地址考虑

互联网单播路由中，有些 IP 前缀是不可能出现

的，如私有地址和组播地址. DSF 模型中，如果一个单元前缀覆盖范围完全是这些地址，就无需分配. 例如针对组播地址，它是二进制 1110(十进制 14)开始的地址范围. 当单元前缀划分位数为 4 时，组播地址恰好归属于单元前缀：$PFX_{\text{unit}}(14)$. 当单元前缀的划分位数为 5 时，组播地址恰好归属 2 个单元前缀：$PFX_{\text{unit}}(28)$和 $PFX_{\text{unit}}(29)$. 以此类推，当单元前缀的划分位数 k 大于等于 4 时，有 $2^{(k-4)}$ 个单元前缀属于组播地址空间，分别是 $PFX_{\text{unit}}(14\times 2^{(k-4)})$，$PFX_{\text{unit}}(14\times 2^{(k-4)}+1)$，…，$PFX_{\text{unit}}(14\times 2^{(k-4)}+(2^{(k-4)}-1))$. 考虑无效地址的分解存储算法实现时，在本文算法基础上略作修改即可.

4　设计及算法

4.1　线卡设计

图 2 是基于普遍施用的 TCAM 结构，设计的 DSF 模型线卡硬件功能结构. 关键模块包括：Module_1，外部接口报文收发模块；Module_2，FIB 表查询及转发处理模块；Module_3，板间报文分析处理模块. 其中，Module_2 是整个转发引擎的核心，通过 TCAM 芯片存储转发项的 IP 前缀并实现 LPM 处理. 静态随机存储器(SRAM)在路由器查找系统中配合 TCAM 查找，用于存储匹配表项信息，即 LPM 查找结果：出线卡、出接口、下一跳等信息. 其中，Module_1 和 Module_2 是当前商用高端路由器普遍具备的功能[9]，目前的线卡设计中没有标识①所指流程，收到交换网络发来的报文就直接走标识②所指流程，利用边带信息发出. Module_3 是专门为 DSF 模型设计的. 报文从其它线卡经交换网络发到本线卡，由板间报文分析处理模块对报文的边带信息(报文内容以外的相关处理信息，如入接口、出接口、下一跳地址等)进行分析处理，有两种可能：

(1) 报文已获得出接口及下一跳信息则直接发送(即图 2 中标识②)，对于该报文来说本线卡的角色是 LC_{out}，该类报文在本线卡会直接通过外部接口发送.

(2) 报文未获得有效的出接口及下一跳信息(即图 2 中标识①)，到本线卡进行最终 LPM 查询及转发处理. 对于该报文来说本线卡的角色是 LC_{attach}，该类报文会在本线卡进行最终 LPM 查询及转发处理.

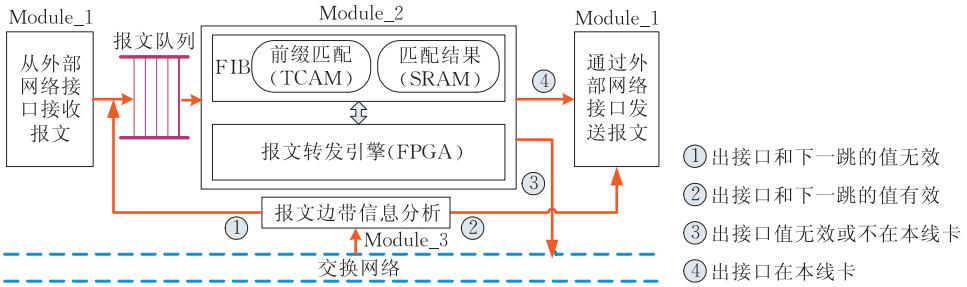


图 2　支持 DSF 模型的线卡功能结构

无论 DSF 模型还是下文介绍的 EDSF 模型，都有一部分报文需要多一次 LPM 查找或多一次板间转发，这就带来两个问题：报文转发时延增加、消耗更多的内部交换带宽. 不过，由于核心路由器普遍采用 TCAM 芯片完成 LPM 查找，TCAM 查找速度可达到纳秒级，但这对端到端至少毫秒级的传输时延来说影响很小，不影响设备提供的网络服务质量. 而对于交换带宽，由于目前的分步式核心路由器处理速度瓶颈在线卡本身而不在交换网络上，我们只需通过设置提高交换结构的加速比就可解决内部交换带宽增加的问题. 另外，我们还可在线卡增加存储常用路由(Popular Routes)，使得两次 LPM 查找或两次板间转发的情况发生概率大大减小. 限于篇幅，本文不对该问题深入探讨.

4.2　实施步骤及算法

DSF 模型实施时包括以下步骤(它们全在主控板中完成，各线卡只需对主控发来的转发项进行存储即可)：

1. 参数初始化. 主控在向线卡分发转发项之前设置参数. 设置系统核心路由数为 m；设置线卡数为 n；根据式(1)计算单元前缀分解位 k，即单元前缀掩码长度；计算单元前缀数量 $s=2^k$，也等于虚拟路由数.

2. 真实路由分解存储. 路由引擎对真实路由 IP 前缀分析，若为聚集前缀，对应路由项分发给所有线卡存储；否则，只通告所属单元前缀所在的线卡存储. 见算法 1.

3. 虚拟路由存储. 路由引擎为每个单元前缀构造一条虚拟路由，通告除该单元前缀所属线卡之外的所有线卡. 见算法 2.

算法 1. Distribute-Actual-RT.

```
1. for (i=1; i<=m; i++)
2.     get masklen of PFX_dst of RT_i;
3.     if (masklen<k)
4.         send RT_i to all LC;
5.     else
6.         get PFX_dst of RT_i;
7.         for (j=1; j<=|PFX_unit|; j++)
8.             if PFX_dst ≺ PFX_unit[j]
9.                 send RT_i to LC PFX_unit[j] belongs to;
10.                break;
11. return;
```

算法 2. Distribute-Virtual-RT.

```
1. for (i=0; i<s; i++)
2.     if (i<n)
3.         construct virtual_rt:
               ⟨PFX_unit(i), LC_i, Inv, Inv⟩;
4.         send virtual_rt to all LC except LC_i;
5.     else
6.         construct virtual_rt:
               ⟨PFX_unit(i), LC_(i-n+1), Inv, Inv⟩;
7.         send virtual_rt to all LC except LC_(i-n+1);
8. return;
```

算法假设系统中线卡编号从 0 开始依次增加，实际环境中并不一定如此. 此时，算法略作改进，只需给每块线卡额外分配一个依次加的序号，并记载序号与线卡号的对应关系，算法主体思想不变.

4.3 存储分析

通过式(5)，我们可以分析虚拟路由总数、每线卡平均虚拟路由数随线卡数的变化关系，见图 3. 图中虚拟路由总数随线卡数增多有一定的增加，但数量较小，16 线卡时也不到 250 条虚拟路由. 相对核心路由器多达几十万的路由数来说，影响不大. 另一方面，每线卡平均存储的虚拟路由数量也随线卡数增加而略有增加，但数值一直较少，16 线卡时每线卡增加存储 15 条虚拟路由. 所以，DSF 方案在真实 FIB 表分解存储过程中所增加的虚拟路由数极少，额外存储开销很小.

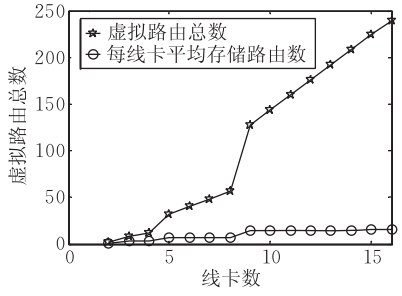


图 3 DSF 模型虚拟路由数量分析

接着，我们分析路由器总共存储的转发项数及每线卡平均存储的转发项数. FBS 方案系统存储的转发项总量为 $m\times n$. DSF 模型中，路由存储分为三块：聚集前缀真实路由、非聚集前缀真实路由和虚拟路由. 令聚集前缀路由数为 m'，它们会在每块线卡存储，共为 $n\times m'$. 非聚集前缀路由只会在某一块线卡存储，共存储 $m-m'$. 而对于虚拟路由数量，依据式(5)计算可得. 所以，DSF 方案中系统的转发项总量为 $(m-m')+n\times m'+(2^{\lceil\log_2 n\rceil}\times(n-1))$，即 $m+(m'+2^{\lceil\log_2 n\rceil})\times(n-1)$. 由于聚集前缀路由数 m' 总是较小，所以相对 FBS 方案中 $m\times n$ 的转发项数量，DSF 方案在系统转发项总数和每线卡平均转发项数有很大的减少，大大节省了硬件存储资源.

5 优化模型

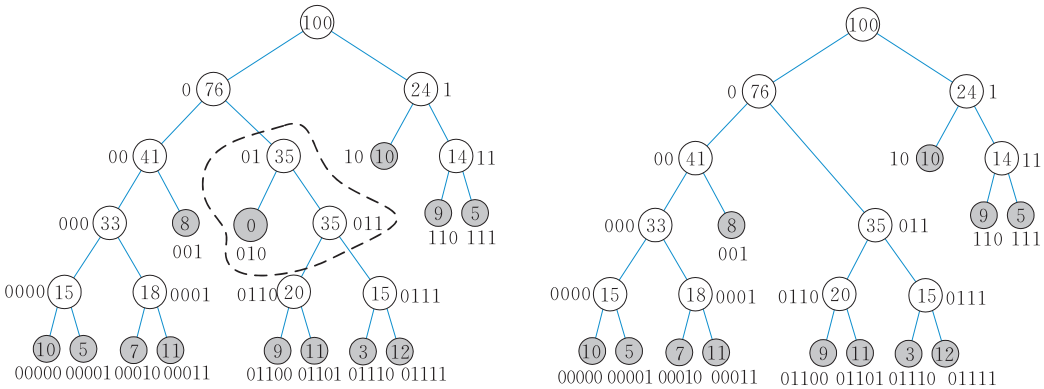
互联网中，地址分配零乱且不延续，导致路由表 IP 前缀分布极不均衡. DSF 模型虽然能将系统路由分解到各线卡，但各线卡存储转发项数量差别极大. DSF 模型对两个真实路由表的实施结果都说明了该问题的存在，见表 2 和表 3. 所以，我们设计了优化模型：EDSF(Enhanced DSF)，希望能将 FIB 表项尽量均匀地分解到各线卡. 优化方案的转发引擎，仍然通过虚拟路由完成报文到宿主线卡 LC_{attach} 的传送，线卡功能结构与图 2 一致.

EDSF 模型区别于 DSF 模型的主要特征是：不同的单元前缀指定策略. DSF 模型的单元前缀掩码长度总时相等，EDSF 模型却无此约束. 优化方案主要思想是从第 1 个 bit 位开始，每次根据路由 IP 前缀一个 bit 位的值对核心路由表进行分解. 针对每次分解，令该 bit 位值为 0 的一份为 SEG_0，值为 1 的一份为 SEG_1. 只要分得部分的路由数大于基准值 $m/2n$，即平均每线卡路由数的 1/2，就根据下一 bit 位的值继续分解. 分解完成后，根据树根节点到每一叶子节点的 bit 位值，生成一个单元前缀 PFX_{unit}，单元前缀个数就是该分解树的叶子节点数. 每一 PFX_{unit} 只分配到某一线卡，一块线卡可分得 1 个或多个 PFX_{unit}. 分配依据是归属到每一线卡的多个 PFX_{unit} 所辖真实路由数尽量接近. 后续处理，如根据单元前缀生成虚拟路由，以及真实路由存储过程与标准 DSF 一致. 需要说明，上述基准值的指定与方案所需分解均衡粒度相关，基准值越小分解得越平均，但会带来更多的虚拟路由.

下面我们举例说明分解树产生过程. 假设某系

统核心路由表共有 100 条路由，4 块线卡，则基准值 $m/2n$ 为 12.5. 根据前述规则，只要分解后的节点对应的路由数大于 12.5，就根据下一 bit 位继续分解. 图 4(a)是该系统对应的分解树，其中树节点圈内数值表示归属该节点前缀的路由数，圈外数表示该节点对应前若干 bit 位的值. 如第一次分解，根据 IP 前缀第 1 bit 位分解. SEG_0 有 76 项，对应节点 0，SEG_1 有 24 项，对应节点 1. 由于节点对应路由数都大于 12.5，需要根据下一 bit 位继续分解，直至所有叶子节点对应的路由数都不大于 12.5. 图 4(a)中，虚框内节点 01 分解后，所有路由都被节点 011 继承，此次分解没有意义. 所以，可对图 4(a)中虚框内结点压缩为一个节点，压缩后的分解树如图 4(b)所示.

分解树生成后，根据路由均分原则将单元前缀归属到各线卡. 如表 1，各线卡分得路由数分别为 25，25，24，26. 表中每个叶子节点对应一个单元前缀，如节点 00000 对应 PFX_{unit}：0.0.0.0/5，节点 10 对应 PFX_{unit}：128.0.0.0/2.



(a) 未压缩的分解树　　(b) 压缩后的分解树

图 4　基于 EDSF 的分解树示例

表 1　基于 EDSF 的转发表分解存储示例

	叶节点	每节点路由数	叶节点	每节点路由数	叶节点	每节点路由数	每线卡上总路由数
LC_0	00000	10	00010	7	001	8	25
LC_1	00001	5	00011	11	01100	9	25
LC_2	10	10	01101	11	01110	3	24
LC_3	01111	12	110	9	111	5	26

所有单元前缀都是分解树的叶子节点，则任意两个单元前缀的覆盖范围交集为空. 所以对于任一路由前缀 PFX_{dst}，最多只可能归属于一个 PFX_{unit}. 由于分解树覆盖了 IP 前缀的全集，而且是从树根开始，依次根据 1 个 bit 位进行分解的. 所以，如果一个 PFX_{dst} 不归属任一 PFX_{unit}，则必有对应的非叶子节点. 以其为根的子树中，叶子节点代表的单元前缀全部归属该 PFX_{dst}. 如图 4 中，前缀 128.0.0.0/1 不归属分解树任一 PFX_{unit}，其对应非叶子节点 1. 以节点 1 为根的子树所有单元前缀：128.0.0.0/2(节点 10)，192.0.0.0/3(节点 110)和 224.0.0.0/3(节点 111)都归属 128.0.0.0/1.

EDSF 模型在指定单元前缀并分属到各线卡后，对于系统的任一核心路由项，其分解存储方法遵循规则 4.

规则 4.　令一条核心路由目的 IP 前缀为 PFX_{dst}，若 PFX_{dst} 归属某单元前缀 $PFX_{\text{unit}}(i)$，即 $PFX_{\text{dst}} \prec PFX_{\text{unit}}(i)$，则对应转发项存储在 $PFX_{\text{unit}}(i)$ 所属的线卡；若 PFX_{dst} 不归属任何 PFX_{unit}，则对应转发项存储在 PFX_{dst} 对应子树的单元前缀所属线卡.

以图 4 及表 1 进行说明. 若一条路由前缀为 33.0.0.0/8，它归属节点 001 对应的单元前缀 32.0.0.0/3，该单元前缀分配给了 LC_0，所以将该路由存储在 LC_0. 若一条路由前缀为 128.0.0.0/1，不归属任一单元前缀，而对应非叶子节点 1. 以节点 1 为根的子树有 3 个叶子结点，其单元前缀分别属于 LC_2 和 LC_3. 所以，前缀为 128.0.0.0/1 的路由项存储在 LC_2 和 LC_3. 现实互联网中，没有前缀长度小于 8 的路由. 所以，只有当分解树的深度达到 9 以上时，真实路由的 IP 前缀才可能出现在分解树的非叶子节点上，此时出现真实路由的冗余存储. 而一般情况，EDSF 分解树的深度很难达到 9 以上. 所以，EDSF 模型出现真实路由在多块线卡存储的概率很小.

真实路由分解存储后，如 3.2 节描述思想，系统根据单元前缀产生虚拟路由并存储. 对任一单元前缀 $PFX_{\text{unit}}(i)$，若其被分配给第 j 块线卡，系统增加一条 IP 前缀为单元前缀的虚拟路由：$\langle PFX_{\text{unit}}(i), j, Inv, Inv\rangle$($Inv$ 表示一个无效值)并通告该路由给

除 LC_j 外的所有线卡存储.

下面给出 EDSF 模型相关算法.

算法 3 用于 EDSF 模型分解树生成,是一个递归算法. 参数 SET_{rt} 表示本次待分解的路由集合,参数 i 表示依据第 i 个 bit 位进行分解. 首次调用中,参数 SET_{rt} 是所有路由项的集合,i 等于 1. 只要分得的 SEG_0 容量大于 $m/2n$,就对 SEG_0 进行分解,之后再回朔分解 SEG_1,分解过程是深度优先进行. 另外,可以对单脉相传的节点进行压缩. 即一次分解后所有的路由项都由某一子节点继承,则该次分解无需进行,并根据下一 bit 位进行分解. 算法 4 描述了系统真实路由项的存储方法. 若目的 IP 前缀 PFX_{dst} 归属某单元前缀 $PFX_{unit}(i)$,则该路由项只存储在一块线卡:$PFX_{unit}(i)$ 所属的线卡,算法会跳出内层循环. 否则,该路由存储在 PFX_{dst} 对应子树所有叶子节点的单元前缀所属线卡. 算法会遍历所有单元前缀,只要满足 $PFX_{unit} < PFX_{dst}$,就把路由存储到该单元前缀所属线卡. 算法 5 实现虚拟路由存储. 依次对每一 PFX_{unit},以其为目的 IP 前缀生成虚拟路由,虚拟路由的出线卡是该单元前缀所属线卡,出接口和下一跳为无效值. 该虚路由存储在 PFX_{unit} 所属线卡之外的其它线卡.

算法 3. Distribute-Tree(SET_{rt}, i).

1. SEG_0 is subset of SET_{rt}, satisfied with "value of the ith bit of PFX_{dst} is equal 0";
2. SEG_1 is subset of SET_{rt}, satisfied with "value of the ith bit of PFX_{dst} is equal 1";
3. if ($|SEG_0| > m/2n$) Distribute-Tree(SEG_0, $i+1$);
4. if ($|SEG_1| > m/2n$) Distribute-Tree(SEG_1, $i+1$);
5. return;

算法 4. Hand-Actual-RT.

1. for ($i=1$; $i<=m$; $i++$)
2. get PFX_{dst} of RT_i;
3. for ($j=1$; $j<=|PFX_{unit}|$; $i++$)
4. if $PFX_{dst} < PFX_{unit}[j]$
5. get the LC $PFX_{unit}[j]$ belongs to;
6. RT_i is stored in this LC;
7. break;
8. if $PFX_{unit}[j] < PFX_{dst}$
9. get the LC $PFX_{unit}[j]$ belongs to;
10. RT_i is stored in this LC;
11. return;

算法 5. Hand-Virtual-RT.

1. for ($i=1$; $i<=|PFX_{unit}|$; $i++$)
2. LC_j is the LC $PFX_{unit}[i]$ belongs to;
3. construct $virtual\_rt$:〈"$PFX_{unit}(i)$, LC_j, Inv, Inv"〉;
4. send $virtual\_rt$ to all LC except LC_j;
5. return;

算法实施在主控板完成,各线卡只需对主控引擎发来的转发项进行存储即可. EDSF 模型完成转发项分解存储后,报文转发过程与 DSF 模型相同. 所以,EDSF 模型的线卡功能设计与 DSF 模型完全一样,图 2 所示.

EDSF 模型中,每一单元前缀生成一条虚拟路由,而每条虚拟存储在($n-1$)块线卡. 所以,系统总共存储虚拟路由数为($n-1$)×$|S_{unit}|$. 虚拟路由数与线卡数量或单元前缀数成正比关系. 而对于单元前缀个数,在线卡数一定的情况下,与真实路由数量及路由 IP 前缀分布是否均匀相关. 路由 IP 前缀分布得越均匀,需要的单元前缀越少;而真实路由越多,则需要的单元前缀越多. EDSF 模型虚拟路由数比 DSF 模型略有增加,但基本维持一个数量级,参见表 2、表 3 实际路由表分析.

6 实验分析

实验系统利用 5 台安装 Linux 操作系统的 PC 机完成. 交换机模拟路由器内部交换网络,$PC_{0\text{-}3}$ 模拟 4 块线卡,PC_4 模拟主控. 主控主要负责管理路由表及向线卡分发路由. 这样,1 块主控及 4 块线卡构成一台可扩展分布式路由系统. $PC_{0\text{-}3}$ 都安装两块以太网卡,Eth0 用于联接内部网络,Eth1 用于路由器系统的外部通信. 各线卡利用 CLICK[10] 模块仿真路由器转发引擎. CLICK 现有功能可支持传统 FBS 模型,而按图 2 功能整改又可支持 DSF 模型和 EDSF 模型. 线卡可支持几种模型的功能切换.

我们获取了真实路由表对各种存储模型进行分析. Data1① 和 Data2② 分别是 CERNET 和 AS65000 真实运营中的路由表. 在主控上分别注入 Data1 和 Data2 路由表,再查看各线卡转发表存储情况. 实验分别基于 4 种转发存储模型进行,实验结果如图 5、图 6 所示. 其中,SPAL 模型面向 Data1 最优选择第 10 及第 13 位进行分解,面向 Data2 最优选择第 14 及第 15 位进行分解.

我们对数据的进一步整理分析为表 2 和表 3. 可以看出,相对传统的 FBS 模型,无论线卡数量多少或者不同的路由表,DSF 模型和 EDSF 模型都能

① Routing Table Data, http://www.cernet.edu.cn/, Dec. 2009

② BGP Routing Table Data, http://bgp.potaroo.net, Dec. 2009

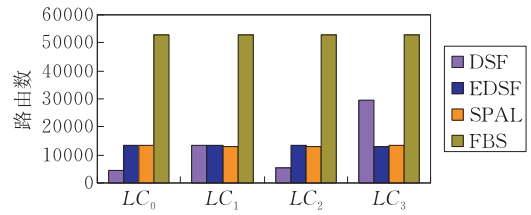


图 5　不同模型下面向 Data1 的分解存储

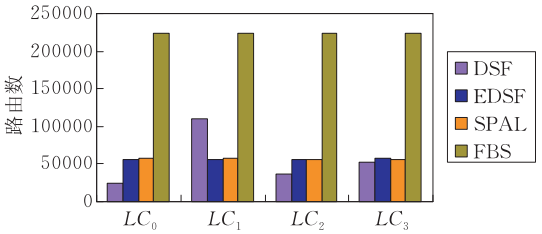


图 6　不同模型下面向 Data2 的分解存储

大量减少系统转发项数量. DSF 模型中，虽然实现转发项分解存储，但各线卡存储的转发项数量差异非常大. 同时，EDSF 模型却能实现将路由非常平均地分解存储在各块线卡. DSF 模型和 EDSF 模型增加的虚拟路由数量都很小，EDSF 较之 DSF，虚拟路由增加略多. 相对 SPAL，EDSF 模型分解更为平均，带来的额外存储也更少，即最后存储的路由总数更少.

表 2　基于 Data1(CERNET 路由表)的不同模型存储分析

	每线卡最多路由数	每线卡最少路由数	冗余路由数	虚拟路由数	线卡路由数最大差值	路由总数
FBS	53018	53018	159054	0	0	212072
DSF	29738	4485	0	12	25253	53030
EDSF	13347	13093	0	45	254	53063
SPAL	13654	12933	66	0	721	53084

表 3　基于 Data2(AS65000 路由表)的不同模型存储分析

	每线卡最多路由数	每线卡最少路由数	冗余路由数	虚拟路由数	线卡路由数最大差值	路由总数
FBS	224435	224435	673305	0	0	897740
DSF	110370	24178	0	12	86192	224447
EDSF	57105	55448	0	39	1657	224474
SPAL	57699	55709	2651	0	1990	227086

上述 DSF/EDSF 分解过程中，没有出现某条路由目的前缀是聚集前缀的情况，所以不存在真实路由的冗余存储. 整个分解存储过程中，DSF/EDSF 模型只增加了少量的虚拟路由，并且不存在一条真实路由的冗余存储，具有很好的存储性能. 从线卡路由数最大差值和路由存储总数两方面分析，显然 EDSF 模型具有最好的存储性能. 最后，我们将仿真路由器的外部接口与普通 PC 相连，验证 DSF/EDSF 模型能根据灌入路由进行正常报文转发.

7　总　结

本文设计的转发表分解存储模型以 IP 前缀的前若干 bit 位进行分解存储，实现线卡对 FIB 表的分解存储. 该模型在各线卡增加少量虚拟路由与真实路由项共存于一个转发表. 线卡转发引擎的 LPM 查询集成了对虚拟路由和真实路由项的查找，既可实现转发下一跳查询，又可实现报文到转发子表的对应. DSF 模型大大降低了系统总共存储转发条目及各线卡存储转发条目，节省了硬件资源及电能开销. 而且，本文还针对真实网络中路由 IP 前缀分布不均衡特征，设计了改进方案 EDSF，它使各线卡存储的转发项数非常均衡. 与现有 SPAL 技术比较，在路由存储的均衡度及冗余率方面都有了进一步的改进及提高.

根据对当前正在运营的两个路由器路由表数据进行分析，证明了本文分解存储方法的优越性. 而且，无论 DSF 模型或 EDSF 模型，都有较强兼容性，只需在现有商业路由器上做极小的改动即可实现. 所以，该方法可快速进入产业化进程，具有很好的现实意义. 对于线卡数量更多的可扩展路由器，本文的分解存储方法尤为适合.

致　谢　本论文工作在清华大学完成，感谢清华大学网络研究所的支持！

参　考　文　献

[1] Meyer D, Zhang L, Fall K. Report from the IAB workshop on routing and addressing. RFC 4984, 2007

[2] Xu Ke, Wu Jian-Ping, Xu Ming-Wei. Advanced Computer Networks: Architecture, Protocol Mechanism, Algorithm Design and Router Technology. Beijing: Mechanism Industry Press, 2009(in Chinese)
(徐恪，吴建平，徐明伟. 高等计算机网络：体系结构、协议机制、算法设计与路由器技术. 北京：机械工业出版社，2009)

[3] Jiang W, Wang Q, Prasanna V K. Beyond TCAMs: An SRAM-based multi-pipeline architecture for terabit IP lookup// Proceedings of the INFOCOM 2008. Phoenix, USA, 2008: 1786-1794

[4] Akhbarizadeh Mohammad J, Nourani Mehrdad. An IP packet forwarding technique based on partitioned lookup table// Proceedings of the IEEE International Conference on Communications(ICC2002). New York, NY, 2002

[5] Tzeng Nian-Feng. Routing table partitioning for speedy packet lookups in scalable routers. IEEE Transactions on Parallel and Distributed Systems, 2006, 17(5): 481-494

[6] Cisco Systems, Cisco 12016 Gigabit Switch Router, Data Sheet. http://www.cisco.com, 2001

[7] Juniper Networks, Inc. T-Series Routing Platforms: System and Packet Forwarding Architecture. white paper, http://www.juniper.net, Apr. 2002

[8] Hitachi, Ltd. The Hitachi GR2000 Gigabit Router Series. http://www.internetworking.hitachi.com, 2002

[9] Xu Ming-Wei, Xu Ke. Design and implementation of high performance security router BW7000. Engineering Sciences, 2002, 4(3): 54-62(in Chinese)
(徐明伟, 徐恪. 高性能安全路由器 BW7000 的设计与实现. 中国工程科学, 2002, 4(3): 54-62)

[10] Morris R, Kohler E, Jannotti J, Kaashoek M F. The click modular router//Proceedings of the 17th Symposium on Operating Systems Principles (SOSP'99). Kiawah Island, SC, USA, 1999: 217-231

CHEN Wen-Long, born in 1976, Ph. D., lecturer. His research interests include network protocol and network architecture.

XU Ming-Wei, born in 1971, Ph. D., professor. His research interests include network architecture, high-performance router architecture and protocol test.

YANG Yang, born in 1955, Ph. D., professor. His research interests include network protocol and network architecture.

HAN Dong, born in 1989, M. S. candidate. His research interests focus on computer network.

Background

This research is supported by the National Basic Research Program (973 Program) of China under Grant No. 2009CB320502; the National High Technology Research and Development Program (863 Program) of China under Grant No. 2009AA01Z251; the National Key Technology R&D Program under Grant No. 2008BAH37B03; the National Natural Science Foundation of China under Grant No. 60873192.

Along with rapid expanding of Internet and implementation of multi-homing, capacity of core routing table of Internet has been increasing radically for past years, and size of some routing tables are more than 300,000. In current distributed routers, every line cards must store all FIB of system. Therefore, increasing of FIB will lead to great storage consume, and maybe influence the speed of route lookup. In addition, the architecture of core routers has gone through three phases: single CPU and centralized bus structure, multi-CPU and distributed bus structure, multi-CPU and distributed switching structure. Cluster router is a trend of the next generation router development. Its main advantages include saving prophase investment, improving system performance, enhancing the reliability of the functional module, predigesting the network topology and so on. Current main router vendors have already supported this kind of routers such as CRS-1 of Cisco, Routing Matrix of Juniper, and TSR of Avici.

In recent years, lots of research focus on improvement of storage and lookup of FIB. However, research on decomposed storage of FIB aiming at cluster router, like SPAL, is very little. SPAL implements decomposed storage of FIB according to the prefix matching algorithm. Nevertheless, its implement is very complex, and there exists many redundancy storage of FIB. This paper propose a novel approach called DSF to decompose FIB, which can reduce the unnecessary redundancy of FIB storage across multiple line cards. Moreover, an enhanced mechanism, EDSF, is proposed to further minimize the diversity of the storage of line-cards. Storage decomposition model can alleviate the pressure from the fast increasing size of FIB can save lots of hardware resources.