中国科学: 信息科学 2020 年 第 50 卷 第 4 期: 576–587
SCIENTIA SINICA Informationis
论文

DNS 权威服务器 FPGA 加速技术研究

李成龙*, 李韬, 韩玉浩, 冯振乾, 王宝生

国防科技大学计算机学院, 长沙 410073
* 通信作者. E-mail: lichenglong17@nudt.edu.cn

收稿日期: 2019–01–15; 修回日期: 2019–04–12; 接受日期: 2019–05–14; 网络出版日期: 2020–04–01

摘要 现有 DNS 权威服务器处理 DNS 请求及响应报文依赖软件网络协议栈, CPU 资源占用率高、开销大, 处理性能受限. 本文基于 SmartNIC 架构对 DNS 权威服务器的功能进行卸载加速, 提出并设计了高性能 DNS 权威查询响应流水线 PHDR_Pipe (perfect Hash DNS response pipeline), 基于完美哈希 (perfect Hash) 实现对区文件的预先处理, 避免哈希冲突导致的多次访存, 降低流水线最坏情况下处理延迟, 从而有效提升系统吞吐率并降低响应延迟. 基于开源的 FAST 平台的实验结果表明, 与通用的 BIND9 系统相比响应延迟降低了约 10 倍, 吞吐量接近 10 Gb 链路线速, 同时资源开销小且具有良好的可扩展性.

关键词 DNS, 权威服务器, 完美哈希, FPGA, 加速

1 引言

DNS (domain name system) [1], 即域名解析系统, 是用于完整域名和 IP 地址之间映射的一个分布式数据库系统. DNS 系统是确保现代互联网能够正常运行的重要基础, 所有运行在 IP 网络上的端点都需要依靠 DNS 系统获得不同的互联网资源. DNS 系统中通常有权威服务器 (authoritative server)、本地递归解析器 (local recursive resolver)、命名空间 (name space) 和协议 (protocol) 共 4 个组件, 它们通过相互之间的良好配合在当下的互联网中发挥着巨大作用.

从 1987 年正式提出 RFC 1034 [1] 和 RFC 1035 [1] 至今, 30 年的时间里 DNS 系统的基本结构并没有改变, 随着互联网服务的发展以及物联网的快速兴起, 传统 DNS 系统面临的压力日益增长, 尤其是对权威服务器的性能提出了更高的要求. DNS 权威服务器管理着命名空间中某个子域的域名信息, 如果性能无法满足需求, 将会增加响应延迟甚至出现宕机导致该域名失效. BIND9[1)] 系统目前广泛用于搭建 DNS 权威服务器, 但是受限于传统软件网络协议栈的固有性能瓶颈, 无法满足高性能的需求.

1) Berkeley Internet Name Domain. https://www.isc.org/downloads/bind/.

引用格式: 李成龙, 李韬, 韩玉浩, 等. DNS 权威服务器 FPGA 加速技术研究. 中国科学: 信息科学, 2020, 50: 576–587, doi: 10.1360/N112019-00009
Li C L, Li T, Han Y H, et al. Research on FPGA acceleration technology of DNS authoritative server (in Chinese). Sci Sin Inform, 2020, 50: 576–587, doi: 10.1360/N112019-00009

为了提升 DNS 系统整体性能, 研究者 [2~4] 针对本地递归解析器性能进行软件优化, 对减少客户端的响应延迟有很好的效果; 针对 BIND9 系统存在的问题, 剑桥大学的 Marinos 等 [5] 设计了面向 DNS 等服务优化的定制化软件协议栈, 但是纯软件的解决方案仍然性能受限. 也有一些研究者 [6~8] 基于现场可编程门阵列 (field programmable gate array, FPGA) 展开研究, 通过 FPGA 卸载 DNS 查询、响应操作, 但具体实现复杂, 处理流水线的性能和可扩展性仍存在问题.

值得注意的是, Microsoft 公司 [9] 在其 Azure 云平台中提出了基于 FPGA 的智能网卡 (SmartNIC) 新型架构, 用于加速云平台中服务器的性能. 该架构通过将通用网络功能和专用网络加速功能解耦, 有效降低 FPGA 实现网络功能加速的复杂度, 可以支撑 DNS 权威查询功能卸载加速的实现.

本文对使用 FPGA 加速 DNS 权威服务器进行了深入研究, 主要贡献包括:

(1) 提出基于 SmartNIC 架构实现 DNS 权威服务器卸载加速, 降低 FPGA 处理流水线实现复杂度, 减少硬件实现逻辑开销;

(2) 提出并设计了高性能 DNS 权威查询响应流水线 PHDR_Pipe, 其中响应查找过程基于完美哈希 (perfect Hash) [10] 实现, 使 DNS 权威查询最坏情况下的访存次数降低为 3 次, 从而有效提升流水线处理性能;

(3) 基于开源的可编程网络实验平台 FAST[2)] 对 PHDR_Pipe 流水线进行了原型系统设计与实现, 并与通用 DNS 权威服务器 (搭载 BIND9 系统) 进行了实验对比分析.

2 相关工作

传统的 DNS 服务器均采用软件实现, 通常客户机发出的 DNS 查询首先到达本地递归解析器, 本地递归解析器 (也称为本地递归服务器) 解析查询请求, 然后对不同 DNS 权威服务器进行递归查询, 最后将查询结果返回给客户机并按照一定的缓存策略缓存结果. DNS 权威服务器负责响应来自本地递归服务器的查询请求 (包括第 1 次请求和后续定期更新产生的请求) 和来自客户端的直接查询请求. 主流的服务器大多数使用 Linux 内核, 在处理 DNS 查询请求时需要使用网络协议栈, 会产生内核空间与用户空间之间的多次数据拷贝, 造成额外的时间开销, 成为传统软件网络协议栈的固有性能瓶颈. 高速链路下的 DNS 权威服务器需要快速处理大量的 DNS 查询请求, 这种情况下传统的网络协议栈不仅会导致响应延迟的增加, 同时还可能使得内存中的消息队列溢出造成拒绝服务.

目前关于提升 DNS 系统整体性能的研究, 主要分为从软件和硬件两种不同角度展开. 软件方面, 文献 [2] 利用域名之间关系来提高本地递归解析器的缓存命中率, 文献 [3] 采用自回归模型来提升本地递归解析器的处理速度, 文献 [4] 提出了一种本地递归解析器选择 DNS 权威服务器的策略. 上述研究均是从软件算法的角度对 DNS 系统进行性能优化, 但都会受限于传统软件网络协议栈而产生性能瓶颈. 文献 [5] 提出了一种定制化的软件网络协议栈, 结合预处理思想提升 DNS 权威服务器性能, 但其实现方式可扩展性较弱. 近年来, Intel® DPDK (data plane development kit)[3)]机制被用来提高 Linux 系统的网络数据处理能力, 基于 DPDK 机制采用旁路传统网络协议栈的思想, 虽然避免了多次 CPU 中断带来的巨大开销, 但用户需要在用户空间单独实现协议栈的功能, 带来额外的开发成本. 同时, 随着云平台中智能网卡这类新型架构的出现, 将部分网络功能卸载到可编程硬件上是更为合适的选择.

硬件方面, 由于 FPGA 具有良好可编程性和高速的特点, 因此常被用来作为软件服务功能卸载的平台, 目前也有一些基于 FPGA 对 DNS 服务进行加速的研究. 文献 [6] 基于 FPGA 利用一种简单的

2) FAST. http://www.fastswitch.org.
3) Intel DPDK. https://www.dpdk.org/.

哈希函数实现了搜索功能, 但只用于本地递归解析器. 文献 [7] 基于 FPGA 设计了一种 DNS 查询的流水线处理过程, 针对本地递归服务器的 DNS 请求响应速度进行了优化, 同时利用 MicroBlaze 微处理器完成递归查询过程; 但在对 DNS 请求进行解析时存在固定的时钟周期延迟, 也没有考虑硬件进行响应查找加速, 其设计方法无法用于请求报文间隔极短的权威 DNS 服务器; 文献 [8] 提出了一种完整的基于 FPGA 的权威 DNS 服务器的实现方法, 将区文件中的数据转化为 256-way 的 Radix 树进行 DNS 请求响应的搜索, 从而实现了 DNS 权威服务器对请求响应的加速, 但是使用 Radix 树结构进行查找会带来两个问题: 第一, 每次查找不可能单步完成, 不同长度域名会影响树结构, 也会影响搜索速度; 第二, Radix 树结构会占用过多的内存, 从而使 BRAM (block random access memory) 存储器成为了性能的主要限制.

对于硬件实现使用哈希表快速查找来说, 处理哈希冲突是重要挑战, 哈希冲突会带来不稳定的处理时间从而影响流水线处理过程. 目前也有一些关于在 FPGA 上实现哈希查找的相关研究. Istvan 等 [11] 介绍了基于 FPGA 的 MemcacheD[4)] 服务器的流水线哈希表的设计和实现. 他们设计的全流水线架构可以在 FPGA 上实现 10 Gbps 的线速性能. 但是, 其设计在遇到写后读 (read-after-write) 时存在暂停流水线的危险, 其性能高度依赖于 SET 和 GET 命令的顺序. Cuckoo 哈希 [12, 13] 是另一种常用于硬件实现哈希查找的方法, 其可以在确定时间内实现读取数据, 但是写入时可能会付出冲突的代价: 如果两个位置都被采用, 则使用贪婪算法来重新组织表格以产生高响应时间. 上述研究是从冲突解决 (collision resolution) 的思想进行开展, 即在有哈希冲突存在的情况下, 进行快速且正确的查找. 处理哈希冲突还有另一种思想是冲突避免 (collision avoidance), 即尽可能不产生哈希冲突.

Fredman 等 [10] 提出的完美哈希是采用冲突避免思想的哈希技术的典型代表, 当关键字的集合是一个不变的静态 (static) 集合时, 完美哈希可以使得最坏情况的内存访问时间为 O(1), 即所有关键字都映射到不同的位置从而完全避免了哈希冲突. 完美哈希的思想是利用两级哈希策略, 在每一级中使用全域哈希 (univeral Hash) [14] 技术. 全域哈希技术采用的思想是从一个哈希函数集合中随机选择哈希函数, 使得哈希函数与关键字集合独立, 从而确保没有一种特定的输入会产生最坏的查找情况.

基于 FPGA 实现 DNS 权威服务器要求查询过程的访存不应该成为流水线瓶颈, 最坏情况下的访存次数应尽可能少; 另一方面, 由于域名信息更新不频繁, DNS 权威服务器对区文件 (zone file) 更新及时性要求不高, 可以允许离线预先处理. 完美哈希算法可以有效满足上述特点, 提供 O(1) 的访存时间, 从而实现高效的查找性能. 此外, 因为在 DNS 查询请求中提取的关键字预先已知 (即 DNS 权威服务器的资源数据) 情况下, 完美哈希可以通过离线预处理进行更新.

3 基于完美哈希的 DNS 权威服务器 FPGA 加速框架

3.1 PHDR_Pipe 整体架构

权威服务器的主要工作流程为 (1) 接收 DNS 查询请求, 解析查询内容 (查询域名和查询类型); (2) 根据解析的查询内容, 在区文件中查找相应的资源记录; (3) 如果找到相应资源记录, 构造 DNS 查询响应数据包并发送; 否则, 发送 DNS 查询失败数据包. 从工作流程中可以看出, 解析时间、查询时间和响应封装时间是影响权威服务响应速度的 3 个重要因素.

针对以上 3 个因素, 本文基于 SmartNIC 架构设计了 DNS 权威查询响应流水线 PHDR_Pipe (perfect Hash DNS response pipeline), 实现对查询请求的快速解析和响应数据的封装; 并将完美哈希

4) Memcached. http://memcached.org.

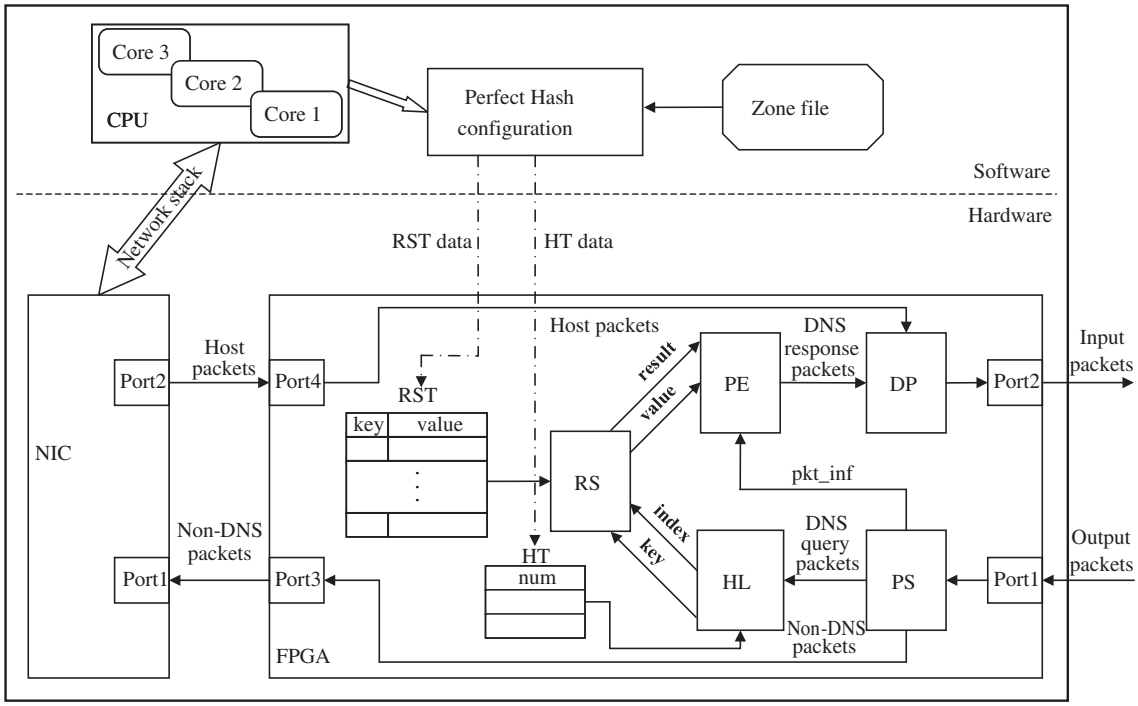

图 1 PHDR_Pipe 整体架构

Figure 1 PHDR_Pipe architecture

查找卸载到硬件上用于加速 DNS 权威查询响应的查找; 由于权威服务器的响应数据是根据区文件生成, 所以预先生成响应数据报文模板, 进一步增加响应速度.

Microsoft 公司提出的 SmartNIC 新型架构已经用于加速 Azure 云平台中主机的性能, 利用 FPGA 的可编程性将一部分软件功能卸载到硬件上, 实现功能加速并且可以节省 CPU 资源. SmartNIC 架构有着明显的优势: FPGA 不需要卸载网卡所有功能, 只需要专注于所加速的服务, 简化工作量并且缩短开发周期. 基于以上, 本文基于 SmartNIC 架构设计了流水线处理框架, 整体架构如图 1 所示, 框架整体分为软件和硬件两大部分. 软件部分主要是完美哈希配置模块和 CPU 处理通过传统网络协议栈送达的非 DNS 的其他报文. 硬件部分主要是 FPGA 和网卡两部分, FPGA 负责处理 DNS 权威查询请求报文, 网卡负责处理非 DNS 的其他报文.

完美哈希配置模块主要用于处理区文件中的资源记录, 提取每一条资源记录的关键字集合, 根据关键字集合生成完美哈希函数及相关数据结构, 同时生成每个关键字对应的响应数据报文模板. 非 DNS 的其他流量直接从 FPGA 送往网卡, 通过传统的软件网络协议栈送至 CPU 进行处理. 云平台主机的其他应用不会被专用加速的 FPGA 所影响, 同时 FPGA 节约出的资源可以用来支持并行加速.

FPGA 部分主要由解析 (parser, PS)、哈希查找 (HASH lookup, HL)、响应存储 (response store, RS)、封装 (packet encapsulate, PE) 和调度 (dispatcher, DP) 这 5 个模块组成. 解析模块对输入数据进行判断, 如果是非 DNS 的其他报文则直接送往与网卡相连的端口; 如果是 DNS 查询请求则对报文进行解析, 提取传输信息 pkt_inf (包括源 IP、源端口号和 DNS TransactionID 等) 送至封装模块. 哈希查找模块首先根据查询报文提取用于查找的 key, 然后使用软件部分配置下来的哈希查找数据结构 (Hash table, HT) 计算出查询内容在响应存储表 (response store table, RST) 中对应位置的索引 index. 响应存储模块根据索引 index 取出对应的 key 和 value, 将取出的 key 与由查询内容生成的 key 进行

表 1 区文件常见记录类型

Table 1 The common record types of zone file

Record type	Meaning
SOA	Authoritative information
NS	Domain name server
MX	Mail server
A	IPv4 address
AAAA	IPv6 address
CNAME	Alias

比较, 如果一致则发送 value, 如果不一致则将 result 设置为未找到. 封装模块接受传输信息 pkt_inf 以及查询结果 result 和 value, result 表明是否找到应答, 如果找到应答则将 value 封装为响应数据报文, 如果未找到应答则构造查询失败数据报文. 调度模块同时接收 DNS 响应数据和由主机发出的数据, 根据调度策略将数据从 FPGA 的端口 2 送出.

完美哈希配置模块决定了 DNS 权威查询处理的正确性, 硬件哈希查找过程决定了整个架构处理性能的瓶颈, 因此本文在 3.2 和 3.3 小节中对这两个部分进行重点讲解.

3.2 完美哈希预处理与配置

区文件中存储了权威服务器能够给出响应的所有查询信息, 这些信息以资源记录 (resource record, RR) 的形式存放. RR 数据通常由 5 部分组成: 名称 (name)、TTL (time to live, 即过期时间)、记录种类 (class)、记录类型 (type) 和记录数据组成. 记录种类通常为 IN, 即指互联网地址, 但是也存在其他种类 (例如 CHAOS, 指 Chaosnet 网络). 表 1 列举了常见的 6 种记录类型.

从 RR 数据的组成可以看出, 名称、记录种类和记录类型可以唯一确定一条 RR 数据, 所以将这 3 个信息用于组成完美哈希的关键字 key. DNS 权威查询请求的报文中包含 Queries 字段, 其内容包括名称、记录种类和记录类型, 可以直接提取作为关键字 key.

本文使用 gperf[5)] 生成完美哈希函数, 对给定的关键字 key 集合, gperf 会以 C 或 C ++ 代码的形式生成哈希函数和哈希表. 生成的哈希函数是完美的, 即对提取出的关键字 key 集合不会产生哈希冲突. 根据输入的关键字集合, gperf 会生成两种关键的数据结构: 固定长度的 HT (Hash table) 数组作为第一级哈希表和变长的 RST (response store table) 数组作为第二级哈希表; 同时生成根据第一级哈希表确定第二级哈希表中对应位置索引的哈希函数. HT 数组的每项是一个整数, RST 数组的每项是自定义数据结构 (包括关键字 key 和值 value, value 的内容可以自定义). 查找过程如图 2 所示, 对于待查找的 keyIn, 首先根据第一级哈希表和哈希函数得出的位置索引, 然后可以直接取出 RST 对应位置的关键字 key 和值 value, 将取出的关键字 key 和待查找的 keyIn 进行比较就可以确定 keyIn 是否是关键字. 对于任何待查找的关键字都只需要一次, 这种特性使得硬件流水线进行哈希查找具有稳定的性能.

完美哈希配置模块调用 gperf 工具, 使用从区文件中提取出的 RR 数据关键字集合生成对应的数据结构 HT、RST 和哈希函数. HT 的表项数设置为 256 对应整个 ASCII 码表; RST 的表项数和 RR 数据的数量有关, 通常是 RR 数据数量的 3 倍左右. 将 RST 中的 value 设置为其关键字 key 对应的查询响应数据报文模板. 对于同一域名内的 RR 数据, 哈希函数通常是固定的. 通过内核驱动模块将

5) GNU gperf. https://www.gnu.org/software/gperf/manual/gperf.html.

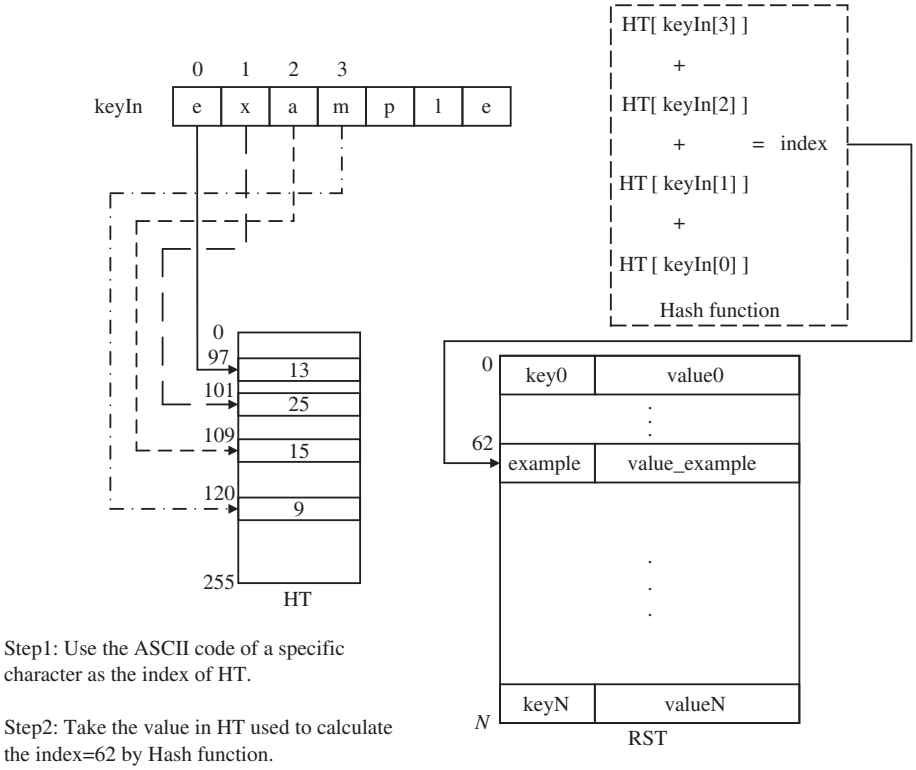

图 2 完美哈希函数查找示例

Figure 2 An example of perfect Hash function lookup

HT、RST 和哈希函数配置到 FPGA 上指定位置的寄存器中, 进行硬件哈希查找时可以直接访问这些数据, 从而快速准确找到查询响应结果.

3.3 硬件哈希查找与更新

硬件哈希查找过程由哈希查找模块和响应存储模块共同完成. 为了支持硬件上的在线更新配置, 我们将 HT 表拆分成 MHT (master Hash table) 表和 SHT (salve Hash table) 表, 同时引入中间表 MMT (master match table) 表和 SMT (salve match table) 表. 如图 3 所示, 查找过程如下:

(a) 关键字提取子模块首先根据 DNS 协议格式取出 Queries 字段内容作为 keyIn, 判断 keyIn 的长度是否在关键字长度范围内 (由区文件内容可知);

(b) 若在范围内, 则将控制信号 len 置为有效, 并将 keyIn 送至哈希函数子模块; 若不在, 则将控制信号 len 置为无效, 并将 keyIn 置为全 0;

(c) HT 判断子模块根据 upd 信号来判断当前有效的 MHT 表还是 SHT 表, 然后将有效数据作为 HT 表送至哈希函数子模块;

(d) 当 len 有效时, 哈希函数子模块对取出 keyIn 的不同位置, 并行地对 HT 表进行查找, 然后将得到的 idx1, keyIn 送至响应存储模块;

(e) 当 len 无效时, 将 index1 置为全 1 (响应表中最后一项存储为查询失败数据报文模板);

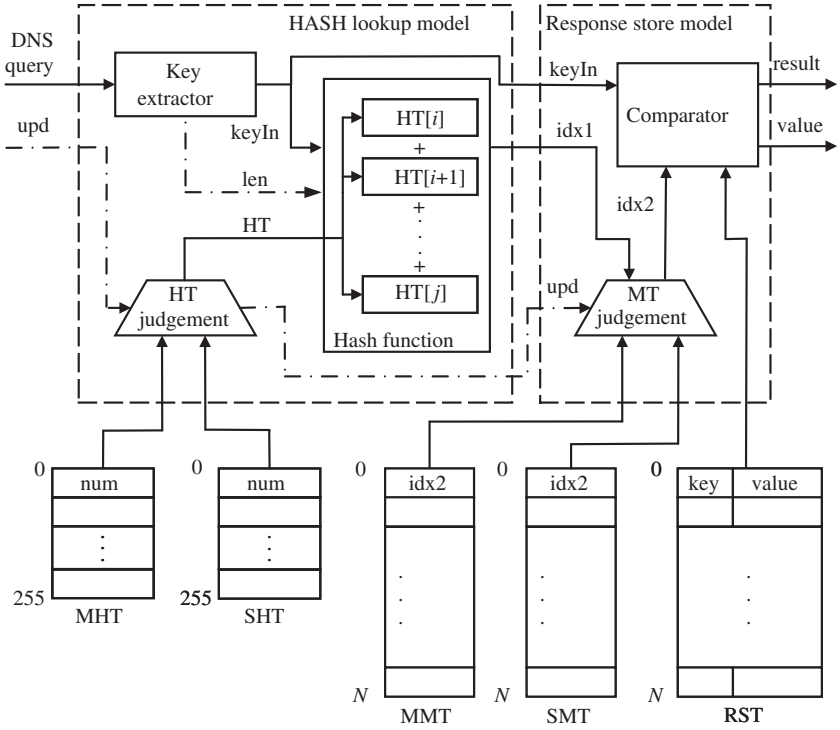

图 3　HL 模块架构

Figure 3　The architecture of HL module

(f) MT 判断子模块根据 upd 信号来判断当前有效的 MMT 表还是 SMT 表, 然后根据 idx1 取出当前有效表中相应位置的 idx2;

(g) 比较子模块根据 idx2 在 RST 表中取出 key, 再将收到的 keyIn 与取出的 key 逐比特进行对比, 若相同则将 value 内容输出, 并将 result 设置为查询成功; 若不同则输出查询失败数据报文模板, 并将 result 设置为查询失败.

整个查找过程的时间开销等于各个子模块的时间开销总和. 哈希查找模块中, 更新判断子模块只需要一个时钟周期就可以完成处理, 计算公式子模块由于对不同位置并行处理所以也只需要一个时钟周期就可以完成处理. 关键字提取子模块的处理时间与 DNS 查询请求中的域名有关, 如果请求域名的长度不在区文件的关键字长度范围内则在一个时钟周期就可以给出结果, 否则处理时间周期与区文件的关键字长度范围成正比. 但是对于给定的区文件来说, 关键字提取子模块的处理时间开销是固定的. 响应存储模块至多需要进行一次比较, 其时间开销情况与关键字提取子模块类似, 也是固定的. 所以整个查找过程的时间开销只与区文件关键字集合有关.

DNS 权威服务器的区文件通常不会频繁更新, 因此当区文件被更新后, 采用离线更新的方式重新生成相应数据结构. 通过设置主、从寄存器的方式支持 FPGA 的在线更新. 在 FPGA 上指定主寄存器存储 MHT 表, 指定从寄存器存储 SHT 表. 更新信号 upd 为 1 代表 MHT 表有效, 为 0 代表 SHT 表有效. 更新当前无效的寄存器, 更新完成后修改更新信号 upd. 同一域名的区文件在更新时, 通常不会大规模改变资源数据, 即仅涉及较少数据的增加、删除和更新, 所以 RST 表的大部分内容是可以复用的. 为了避免不必要的频繁更新, 节省内存资源, 对于 RST 表不采用主、从寄存器方式, 而是引入 MMT 表和 SMT 表 (其原理同上) 来存储更新后的 RST 表与原 RST 表之间相同响应数据的映射关

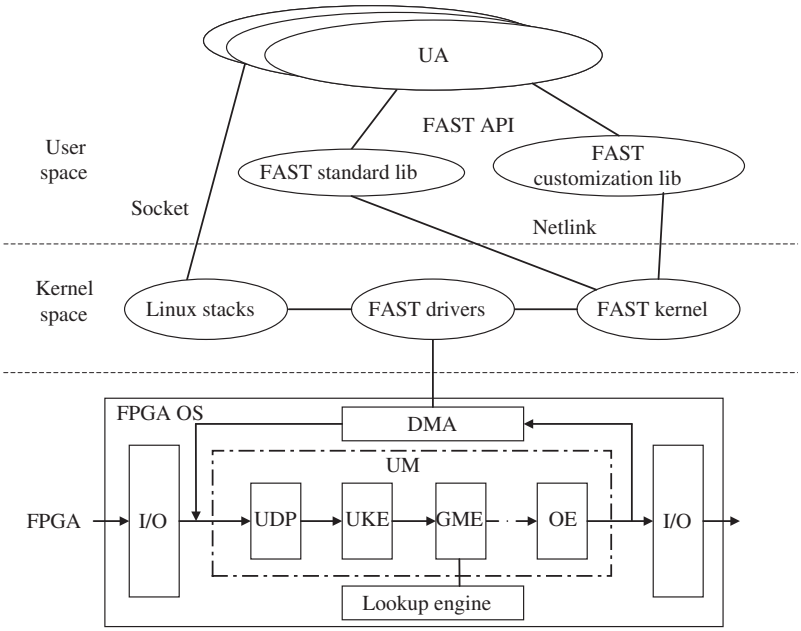

图 4 FAST1.0 架构

Figure 4 The architecture of FAST1.0

系. RST 表的表项数通常是 RR 数据的 3 倍, 所以我们将增加和修改的响应数据插入到原 RST 表的空闲位置中. 由于删除的数据不会出现在中间表中, 所以不需要对原 RST 表进行处理. 当原 RST 表的空闲位置小于阈值时, 可以认为区文件已经有了很大的变化, 此时应重新生成所有数据结构.

MHT 表和 SHT 表的大小均为 256×4 B $= 1$ KB, 占用内存资源少; MMT 表和 SMT 表每个表项为 4 bits, 以 100 bits DNS 响应报文为例, 可以节约 92% 的内存占用即可支持在线更新.

4 实现与评估

4.1 基于 FAST 平台的实现

FAST 是基于多核 CPU 和 FPGA 的开源平台, 采用可扩展的软硬件紧耦合分组处理模型, 支持用户在平台已有的基础上, 通过对 FPGA 硬件流水线的扩充, 或基于 FAST 库编写用户空间程序实现对网络设备数据平面的扩充, 从而实现对各种新型的交换技术进行实验研究.

FAST 平台的 1.0 版本架构如图 4 所示, FPGA OS 为用户屏蔽了不同 FPGA 平台的异构性, 使得 FPGA 中的硬件流水线的实现 (即用户模块, UM) 与具体的平台无关. UA 为用户应用, 即用户基于 FAST API 编写的用户空间程序. 本文基于 FAST 平台, 将完美哈希配置模块作为 UA 进行实现, 将 FPGA 上设计的流水线作为 UM 模块实现.

4.2 测试与分析

实验环境主要包括一台测试仪、一台高性能 DNS 权威服务器和通用 DNS 权威服务器, 实验设备相关信息见表 2. 将 PHDR_Pipe 架构部署到搭载有万兆网卡和 Arria® 10 智能板卡 (如图 5) 的服务器作为高性能 DNS 服务器, 万兆网卡和 Arria® 10 智能板卡都通过 PCIe 与 CPU 通信. 测试仪通过

表 2　实验设备信息

Table 2　Experimental equipment information

Equipment name	Version / Model
BIND9	1:9.10.3.dfsg.P4-8ubuntu1.10
Operating system	Ubuntu 16.04 LTS
Kernel	Linux 4.4.0-31-generic
CPU	Intel Xeon E5-3650
NIC	BROADCOM 5719
FPGA	Arria® 10 10AS027H3F34E2SG
Network tester	IXIA XM2-P0633116

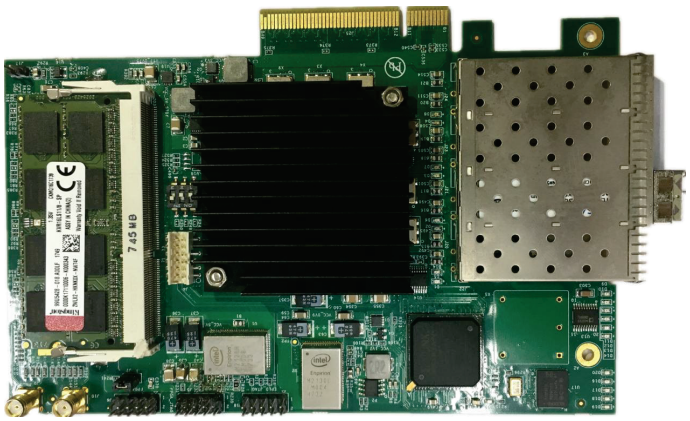

图 5　(网络版彩图) Arria® 10 FPGA 板卡实物图

Figure 5　(Color online) Arria® 10 FPGA board

光纤分别与 PHDR_Pipe 权威服务器和通用权威服务器连接. 使用 BIND9 系统搭建通用 DNS 权威服务器, 并同样安装万兆网卡.

测试过程中, 使用测试仪模拟 10 Gb 链路线速发送 DNS 查询报文, 通过设置不同长度的查询域名和不同资源记录数量的区文件, 分别对两种 DNS 权威服务器进行测试分析.

4.2.1　性能测试

测试主要从吞吐量和响应延迟两方面对 PHDR_Pipe 权威服务器和通用权威服务器进行对比. 图 6 展示了两种服务器的 DNS 查询吞吐量测试结果, 测试中使用 4 种查询关键字长度和类型不同的数据集, 分别是: 20 bits 域内域名 (20_rs)、20 bits 域外域名 (20_rf)、60 bits 域内域名 (60_rs) 和 60 bits 域外域名 (60_rf). 域内域名表示查询域名在区文件中有相应资源记录, 域外域名表示查询域名不属于当前区文件. 目前最常见的三级域名长度都在 20 bits 左右, 而 60 bits (左右) 域名一般是少见的超长域名.

从图 6 中可以看出, 通用权威服务器最大吞吐量仅为 6 Gbps, 而 PHDR_Pipe 权威服务器接近 10 Gb 链路线速, 这体现了硬件流水线对吞吐量的有效提升. 实验结果还表明域外域名会极大地影响通用权威服务器的吞吐量, 这种情况可能与 BIND9 系统的查询查找算法有关. 如果攻击者构造大量域外域名查询对通用权威服务器发起 DNS 查询请求, 就可能会造成系统性能骤降或拒绝服务. 而

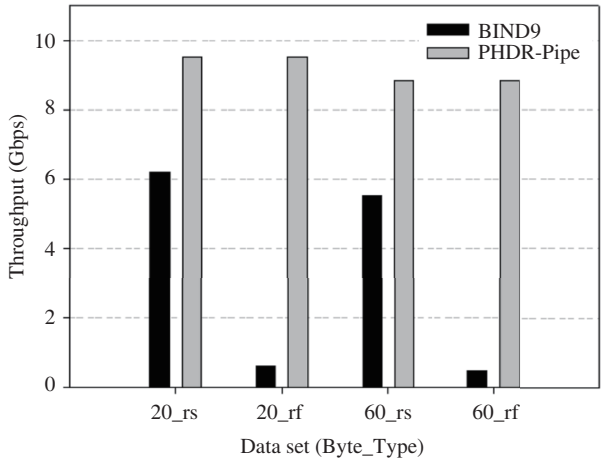

图 6 DNS 查询吞吐量

Figure 6 The throughput of DNS query

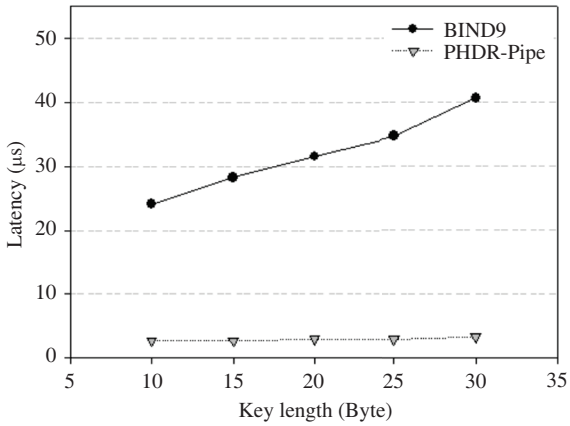

图 7 DNS 查询延迟

Figure 7 The letency of DNS query

表 3 资源占用信息

Table 3 Resource occupancy information

Resource type	Used	Total	Usage ratio (%)
Adaptive logic modules (ALMs)	1431	101620	1.41
Block memory (bits)	366592	15360000	2.39
Registers	2298	–	–

PHDR_Pipe 架构使用的完美哈希保证了最坏情况下查询性能的稳定性, 因此域外域名对 PHDR_Pipe 权威服务器并没有影响. 在处理超长域名时, PHDR_Pipe 架构中硬件流水线的处理周期会稍有增加, 因此吞吐量略有所下降.

考虑到通用权威服务器处理域外域名的性能会极大下降, 因此本文使用 3 种查询关键字长度不同 (20, 40 和 60 bits) 的域内域名数据集进行 DNS 查询响应延迟测试, 结果如图 7 所示. 从实验结果中可以看出, 即使在最好情况下, PHDR_Pipe 服务器的响应延迟也比通用服务器降低了约 90%, 达到了 2.676 μs. 随着查询关键字的长度增加, PHDR_Pipe 架构中硬件流水线对关键字提取和封装需要更多的时间周期, 但查找过程的时间周期固定, 因此 DNS 查询响应延迟成线性增长并远小于通用权威服务器.

4.2.2 资源开销分析

PHDR_Pipe 架构的资源开销分别来自软件和硬件两部分, 硬件部分中 FPGA 的总体资源少, 是制约整体性能的关键因素, 因此本文重点分析硬件部分的资源开销.

FPGA 流水线的资源开销由逻辑单元、模块间 FIFO 和存储数据表 (MHT, SHT, MT 和 RST) 的寄存器组成, 部分关键资源占用统计见表 3. 从统计结果可以看出, 与资源记录数量无关的逻辑部分占用资源很少. 在本文实现 PHDR_Pipe 架构的 Arria® 10 上, 逻辑资源开销仅占全部逻辑资源的 1.41%, MHT 表和 SHT 表的大小是固定的, 即 $256 \times 4 \times 8 \times 2 = 16384$ bits, 以上部分仅占用该 Arria® 10 片上内存的约 2.5%. 对于最常见的 20 bits 左右的三级域名, DNS 查询响应数据报文通常为 100 bits 左

右, MMT 表和 SMT 和 RST 的表项数为资源记录数量的 3 倍, PHDR_Pipe 架构最多可支持近 2 万条资源记录; 若按照 5 千条资源记录存储, 则 PHDR_Pipe 架构可以扩展支持 40 Gb 链路线速.

5 结论

本文提出了一种基于 FPGA 的 SmartNIC 架构下高性能 DNS 权威响应流水线 PHDR_Pipe. PHDR_Pipe 实现了对 DNS 权威查询请求进行并行解析和快速响应, 利用完美哈希技术实现了对查询内容的快速查找. SmartNIC 架构让我们在开发 FPGA 时不需要卸载网卡所有功能, 只需要专注于所加速的服务, 简化工作量并且缩短开发周期. 完美哈希技术避免了哈希冲突, 使得在区文件确定的情况下, 面对各种查询情况具有稳定的性能. 最后, 我们基于 FAST 开源平台实现了 PHDR_Pipe 架构, 实验结果表明, 相比通用的 DNS 权威服务器, 其吞吐量接近 10 Gb 链路线速, 响应延迟大大降低; 同时 FPGA 的资源开销较少, 具有良好的可扩展性.

下一步我们将研究 SmartNIC 架构下传统软件服务卸载加速的通用框架. 同时, 在卸载出现问题时, 对 SmartNIC 架构进行优化.

参考文献

1 Mockapetris P. Domain Names, Concept and Facilities, Implementation and Specification. RFC 1034, RFC 1035, 1987. https://tools.ietf.org/html/rfc1034

2 Shang H, Wills C E. Piggybacking related domain names to improve DNS performance. Comput Netw, 2006, 50: 1733–1748

3 Deb S, Srinivasan A, Pavan S K. An improved DNS server selection algorithm for faster lookups. In: Proceedings of International Conference on Communication Systems Software and MIDDLEWARE and Workshops, Comsware, 2008. 288–295

4 Yu Y, Wessels D, Larson M, et al. Authority server selection in DNS caching resolvers. SIGCOMM Comput Commun Rev, 2012, 42: 80

5 Marinos I, Watson R N M, Handley M. Network stack specialization for performance. In: Proceedings of the 12th ACM Workshop on Hot Topics in Networks. New York: ACM, 2014. 1–7

6 Saito T, Kimura S, Ebihara Y, et al. An FPGA implementation of DNS Servers Using a Simple Hash Function. IEICE Tech Report, 2005, 105: 43–47

7 Li P J, Zhang X M, Shen J L. Design of local DNS server based on FPGA. Appl Res Comput, 2014, 31: 1102–1104

8 Sadoun R, Belouchrani A, Bourennane E B, et al. An FPGA based soft multiprocessor for DNS/DNSSEC authoritative server. Microprocessors Microsyst, 2011, 35: 473–483

9 Firestone D, Putnam A, Mundkur S, et al. Azure accelerated networking: SmartNICs in the public cloud. In: Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation, 2018. 51–66

10 Fredman M L, Komlós J, Szemerédi E. Storing a sparse table with O(1) worst case access time. J ACM, 1984, 31: 538–544

11 Istvan Z, Alonso G, Blott M, et al. A flexible Hash table design for 10GBPS key-value stores on FPGAS. In: Proceedings of International Conference on Field Programmable Logic and Applications. New York: IEEE, 2013. 1–8

12 Pagh R, Rodler F F. Cuckoo Hashing. J Algorithms, 2004, 51: 122–144

13 Kirsch A, Mitzenmacher M, Wieder U. More robust Hashing: cuckoo Hashing with a stash. In: Proceedings of European Symposium on Algorithms. Berlin: Springer, 2008. 611–622

14 Carter J L, Wegman M N. Universal classes of Hash functions. J Comput Syst Sci, 1979, 18: 143–154

Research on FPGA acceleration technology of DNS authoritative server

Chenglong LI*, Tao LI, Yuhao HAN, Zhenqian FENG & Baosheng WANG

College of Computer, National University of Defense Technology, Changsha 410073*, China*
* Corresponding author. E-mail: lichenglong17@nudt.edu.cn

Abstract The existing, authoritative DNS servers process DNS requests and response packets depending on the software network protocol stack, which use DNS resources and have high overhead and limited processing performance. Based on the SmartNIC architecture, this paper quickens the unloading of DNS authority server and proposes and designs a high-performance DNS authority query response pipeline PHDR_Pipe (Perfect Hash DNS Response Pipeline), which realizes the preprocessing of region files based on perfect Hash. To avoid multiple memory access, caused by Hash collision, and reduce the processing delay in the worst case of a pipeline, the system throughput is effectively improved and reduces the response delay. The experimental results based on the open-source FAST platform show that the response latency is reduced by approximately 10 times compared with the general BIND9 system, and the throughput is near the 10 Gb link-line speed. Moreover, the resource overhead is small, and the system is scalable.

Keywords DNS, authoritative server, perfect Hash, FPGA, accelerate

Chenglong LI is now a graduate student at the National University of Defense Technology, China. His main research interests include packet processing on FPGA and computer networks.

Tao LI is currently an associate professor of computer science at the National University of Defense Technology, China. He received a Ph.D. degree in computer science from the National University of Defense Technology. His main research interests include network architecture, network processors, and routers.

Baosheng WANG is a professor and Ph.D. supervisor at the College of Computer at National University of Defense Technology in China. He received his B.S., M.S., and Ph.D. degrees in computer science from the National University of Defense Technology in 1992, 1995, and 2005, respectively. His research interests include router architecture, routing protocol, and cybersecurity.