# 网络处理器的分析与研究*

谭章熹+, 林 闯, 任丰源, 周文江

(清华大学 计算机科学与技术系,北京 100084)

## Analysis and Research on Network Processor

TAN Zhang-Xi+, LIN Chuang, REN Feng-Yuan, ZHOU Wen-Jiang

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

+ Corresponding author: Phn:86-10-62782587, E-mail: xtan@csnet1.cs.tsinghua.edu.cn

http://www.cs.tsinghua.edu.cn

Received 2002-08-09; Accepted 2002-11-12

**Tan ZX, Lin C, Ren FY, Zhou WJ. Analysis and research on network processor. *Journal of Software*, 2003,14(2):253~267.**

**Abstract**: Nowadays the most salient trend with network is the increase in data rates while there is significant effort in developing new protocols and services. However, the traditional network devices which are custom logic based or pure software based, could hardly satisfy both performance and flexibility requirements. To overcome this obstacle, the parallel and programmable network processors have been involved into processing path of routers (switch). Besides network processors which are built on ASIP technology and optimized for network applications, possess the characteristic of hardware and software solution simultaneously. Network processors extend the classic store-and-forward pattern to store-process-and-forward, which makes room for complex QoS control and payload processing. In this paper, the related research work is introduced from two aspects, system and application, and the system issues and challenges of network processors are analyzed. In the end, the future evolution of network processors and associate researches are also speculated.

**Key words**: network processor; parallel processing; latency hiding; QoS control; resource scheduling

**摘 要**: 目前,网络在提高链路速率的同时出现了大量的新协议及新服务,而传统的网络设备一般采用专用硬件芯片或者基于纯粹的软件方案,很难兼顾性能与灵活性两方面的要求.为此,一种并行可编程的网络处理器被引入到路由器(交换机)的处理层面.它基于 ASIP 技术对网络程序处理进行了优化,同时还兼有硬件和软件两种方案的特点.网络处理器的出现将经典的"存储-转发"结构变为"存储-处理-转发",这为复杂的 QoS 控制和负载处理提供了可能.从网络处理器本身及其应用两个角度出发,介绍了相关的研究工作,分析了系统特点和面临的挑战,并展望其未来的发展方向.

* Supported by the National Natural Science Foundation of China under Grant Nos.90104002, 60173012 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA112080 (国家高技术研究发展计划); the National Grand Fundamental Research 973 Program of China under Grant No.G1999032707 (国家重点基础研究发展规划(973)); the Intel Internet Exchange Architecture University Program Grant No.9077 (Intel 公司 IXA 大学研究项目)

第一作者简介: 谭章熹(1980－),男,广东南雄人,硕士生,主要研究领域为计算机网络,系统性能评价.

关键词: 网络处理器;并行处理;延时隐藏;QoS 控制;资源调度
中图法分类号: TP301 文献标识码: A

随着 Internet 主干网络流量的指数性增长,主干路由器的处理线速已从 622Mbps 增加到 2.4Gbps,并且很快将达到 10Gbps[1],特别是大量的多媒体和高清晰度电视应用的出现,促使这一增长进一步加快.同时,复杂的 QoS (quality of services)控制诸如 DiffServ (differented services)[2], RED (random early detection)[3]以及新的网络协议和服务已成为研究热点.而基于 ASIC (application integrated specific circuit) 和 GPP (general purpose processor) 的传统网络处理方案已经不能同时满足处理速度和灵活性这两方面的要求.为此,基于 ASIP (application specific instruction processor) 技术的网络处理器(network processor)得到了广泛的发展.近来出现的网络处理器芯片上已经集成了多个为网络处理优化过的处理器,同时还集成了一些专用硬件处理单元,如 CRC 校验、Hash 单元等,并且在处理线速上已经达到或超过 10 Gbps.它的出现兼顾了 GPP 的灵活性和 ASIC 的执行效率,为从第 2 层到第 7 层的多种应用提供了良好的支持.

与此同时,围绕着网络处理器应用展开的相关研究也得到了飞速的发展,一些企业和学校也给予了足够的重视.比如,Intel 公司专门投资支持全球 100 所大学进行网络处理器及其相关应用的研究,并每年召开一次 Workshop 进行交流与总结.华盛顿州立大学 2002 年专门召开了以网络处理器为主题的学术会议,并在网络处理器硬件体系结构和性能评价分析等方面作了深入的探讨.综合来看,与网络处理器相关的工作主要由两方面构成,一方面是针对网络处理器硬件本身,另一方面则是面向网络处理器的应用.前者侧重于网络处理器的硬件系统设计和性能改善与提升,比如硬件体系结构的研究[4]、网络处理器的集成设计[5]等;后者则是基于网络处理器的相关应用研究,其核心问题就是要充分发挥网络处理器灵活性和高性能的特点,构建并实现一定的网络服务及应用.例如,基于网络处理器的可编程路由器研究[6]、集成 QoS 控制[1]、网络处理器的性能分析与测试[7]等.

本文针对这两方面,首先从网络处理器的自身特性入手,介绍了其硬件结构和基本技术特点.接着从应用的角度,分析了系统设计和应用所面临的主要问题及挑战.最后,本文在网络处理器现有研究成果的基础上展望了网络处理器的发展方向及相关工作.

## 1 网络处理器的硬件结构及基本处理技术

随着 DWDM[8]等光纤技术在主干网络中的广泛应用,每 12 个月光纤链路的容量就增大一倍,而按照摩尔定律,电处理器的处理速度则是每 18 个月增长一倍,因此网络上以电处理器为核心的 IP 路由器已经成为了网络的瓶颈.与此同时,大量新网络协议和网络服务的出现又进一步增加了网络处理的复杂度.文献[9]和将网络处理分为 3 个层次:底层、IP 层和应用层.底层(low or micro level),比如以太网中的 CRC 校验和表查找(table lookup)等;IP 层(IP level),如 IP 路由、DRR、NAT 等;应用层(application level),如 MD5,SSL,URL 交换等.为了弥补处理能力和链路带宽爆炸之间的差距,并兼顾处理所需要的足够高的可编程性和灵活性,网络处理器本身必须具备的要求为:(1) 拥有网络分组并行处理能力;(2) 具有高效的处理速度,能够达到分组的实时处理;(3) 拥有一定数目的网络专用协处理器;(4) 具有高度的可编程性和可扩展性;(5) 能够快速投向市场,尽量减小再开发周期.

为了迎合这些要求,网络处理器将网络处理任务划分到控制层和数据层两个层面,控制层面专门负责非实时性的管理和策略控制等,数据层面承载高速易变的数据实时处理.总体上,网络处理器主要采用以下硬件处理技术:

(1) 两种处理机制

网络处理器内部一般都是多内核(multi-core)的结构[10].这些内核一般可以分为两种:一种是具有一般运算能力和指令存储能力的处理单元 Pes(processing elements);另一种是能够完成特定处理任务的功能模块 Fus(function units),如 CRC 校验单元等.在现有的商业网络处理器中,这两种单元一般采用以下两种组织机制:

流水线:每个内核被设计成具有特定处理功能的模块,这些模块以流水线方式组织在一起完成分组的处理.这种结构的网络处理器主要有 Cisco 的 PXF,Motorola 的 C-5 DCP 等.

并行处理:每个处理单元 PE 都可以完成相似的任务,多个处理单元彼此间可并行执行.这种结构的网络处理器主要有 Intel 的 IXP1200,IBM 的 PowerNP 以及 Agere 的 PayloadPlus 等.

(2) 优化的内存管理和 DMA 单元

在一般的多处理器系统中,内存操作往往是系统开销的一大瓶颈.而一般的网络处理器设备通常要对分组进行存储和复制等处理,需要执行大量的存储器操作.网络处理器为了优化这一操作往往引入经过优化的存储器接口和一些 DMA 单元,以提高存储器的操作效率.比如,在 Intel 的 IXP1200 中,提供了 SRAM 的空闲堆栈以及相应的处理指令.

(3) 优化的运算逻辑单元 ALU

不同于一般的处理器,网络处理器提供了一些用于网络处理的专用指令[11],比如位(字)扩展、压缩指令.同时,大部分网络处理器采用了 RISC 技术,可以做到任何 ALU 运算能在一个时钟周期内完成.

(4) 网络专用的协处理器(co-processors)

现在大部分网络处理器为一些特定的网络操作提供专用的硬件协处理[12],比如专用的路由表查找引擎、硬件分类引擎、缓冲和队列管理引擎.这些协处理器有些集成在网络处理器片内,有些则安排在片外.

(5) 硬件多线程技术

为了进一步提高处理器的利用率,网络处理器通常引入硬件多线程技术,并且线程间的切换开销为 0.比如 Intel IXP1200 的每一个微引擎中拥有 4 个线程,每个线程拥有独立的程序计数器 PC;IBM PowerNP 中也有类似的结构.

下面以 Intel 的 IXP1200 为例来说明上面的特点.IXP1200 属于第一代成熟的网络处理器,由 Intel 公司提出设计要求,DEC 公司完成实现.IXP1200 是 Intel IXA(Internet exchange architecture)结构的最早组成部分,它具有一般网络处理器的各种特点,并在实践中已经得到了一定的应用.IXP1200 主要是针对网络中第 2 层到第 4 层的应用,并且处理速率在 64 字节分组的情况下可达到 2.5Mpackets/s.IXP1200 由 0.28μ m 工艺制造,工作时钟频率最高 232MHz,功耗为 5W,制造时充分体现了 SOC(system on chip)的思想.其体系结构如图 1 所示.

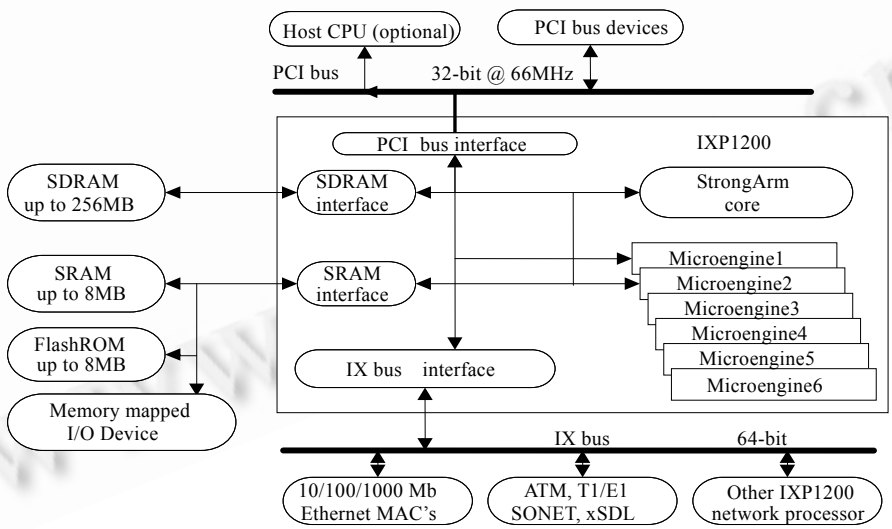



Fig.1 IXP1200 hardware architecture

图 1 IXP1200 硬件系统结构

IXP1200 主要由 6 个可编程微引擎(micro-engines)和一个工作在 200MHz 的 StrongArm 处理器组成.用峰值带宽为 6.6Gbps 的 64 位 IX Bus 将微引擎、StrongArm、存储器以及 MAC 等片外设备连接到一起.PCI 总线则用来和外部设备通信完成 4 层以上的更高层网络处理.微引擎主要用来完成数据层面分组的实时处理,每个微引擎拥有 4 个硬件线程,并且这 4 个线程共用一个寄存器堆,寄存器的分配可由软件编程控制.微引擎具有单指令 ALU 功能,此外还拥有一些用于分组处理的特殊指令,比如比特提取、设置等.每个微引擎最多可以存储

2K 的程序指令,指令的装载与微引擎的配置由 StrongArm (SA) Core 完成.此外,StrongArm 还承担其他一些慢速数据处理,它是控制层面的主要载体.为了提高 StrongArm 的性能和与微引擎的通信能力,片上 StrongArm 集成了一个 8KB 的数据缓存和一个 16KB 的指令缓存以及一个 4KB 的 Scratchpad SRAM.为了增强分组的处理能力,IXP1200 还提供了一个专用的硬件可编程 HASH 引擎以及专用 FIFO 单元,用于从 MAC 接收和发送数据.同时,IXP1200 片上还集成了 SDRAM、SRAM 控制器和 PCI 接口控制器,它最多可以同时支持 256MB 的 SDRAM 和 8MB 的 SRAM.

综上所述,IXP1200 拥有网络处理器的一般特点,从系统角度看,IXP1200 属于一个并行式的多处理器共享总线的计算机系统,多处理器(微引擎和 StrongArm)共享同一存储器,多处理器彼此间的数据交换通过共享总线完成,其他网络协处理器等资源也通过共享总线完成资源共享.

## 2 系统设计和应用所面临的问题

### 2.1 系统处理特性

网络处理器所处理的是从第 2 层到第 7 层的任务,包括从简单的 IP 路由一直到复杂的加密和压缩程序,因此它具有独特的系统处理特性.为了获得这一特性,Tilman Wolf 和 Jonathan S. Turner 在文献[7]中使用了 8 种不同层次的应用程序进行了实际的运行测试,得到了以下的系统处理复杂度的数据(见表 1),其中复杂度使用“RISC 指令/字节”来描述.

**Table 1** Complexities of different applications

**表 1** 不同应用的处理复杂度

| Application | Description | Complexity | Application | Description | Complexity |
|---|---|---|---|---|---|
| RTR | Routing table look up | 2.1 | JPEG | Image compression | 81 |
| DRR | Packet scheduling | 4.1 | CAST | Data encryption | 104 |
| FRAG | IP fragmentation | 7.7 | ZIP | Data compression | 226 |
| TCP | Traffic monitor | 10 | REED | Redundancy coding | 603 |

同时文中还宏观地总结了系统所处理任务的 3 个重要特性: (1) 短任务:大部分任务需要 1~100 个 RISC 指令/字节;(2) 大量的任务数目:一般每秒要处理大于 1 000 000 个分组;(3) 高异构的应用:任务彼此间的差异非常大,比如 IP 路由和 MPEG 编码.

这里,我们根据 Tilman Wolf 等人的数据,分析一下常用分组处理的基本特点.如果以路由查找(RTR)和数据加密(CAST)为例简单估算一下这两种任务对处理器的要求,则假设系统吞吐量为 1.2Gb/s(其余的数据见表 1),

RTR 所需要的运算量为 1.2/8*2.1=315 MIPS,

CAST 需要的运算量为 1.2/8*104=15 600 MIPS.

而现在网络处理器中单个处理元的处理能力仅为 1 500MIPS 左右,例如 IBM 的 NP2G 为 1 596MPIS、Intel 的 IXP1200 为 1 396MIPS.可见系统负载处理的开销是相当庞大的,并且处理的层次越高,开销越大.

### 2.2 系统并行性要求

从第 2.1 节中的分析可以看出,由于处理负载的开销大大超过了现有网络处理器内部单个处理元的处理能力,因此在系统的处理策略上广泛使用了并行处理机制.具体来讲,并行处理的核心就是要充分利用处理任务间的彼此独立性,将不同的任务同时交给不同的服务员处理.从网络处理器所面向的处理任务来看,不同的处理层面并行性特点也不尽相同.对于控制层面和管理层面的任务一般是实时性较差的管理及控制操作,例如路由表的维护和端口状态轮询等,对于数据层面,则是实时性较强的分组处理等操作,各自的系统并行性特点见表 2.

**Table 2** Parallel characteristic of network processor

**表 2** 网络处理器的系统并行特性

| | Performance requirement | Data parallelism | Typical application |
|---|---|---|---|
| Data plane | High | High | Packet routing |
| Control plane | High | Low | ATM VC switching |
| Management plane | Low | Low | SNMP processing |

因此,可以认为网络处理器系统的并行性要求集中体现在数据层面,为了迎合这一要求,一般采用以下 3 个

层次的并行处理技术[13]:

(1) 分组级并行(packet level parallelism,简称 PLP)

大部分网络应用都是基于分组级(packet level)的,不同的分组一般经过类似的处理,分组间一般是独立的,这样分组可以由不同的服务员并行服务.然而这种处理也存在一定的局限性,即当属于同一个流(flow)的分组经过不同服务员并行处理后必须有一个分组的重组保序过程.虽然上层的 TCP 等协议允许分组乱序,但是为了减小在终端分组重组的时间一般还要保证分组的顺序.

(2) 分组内并行(intra packet parallelism,简称 IPP)

在对分组内部处理的过程中,有些任务也是彼此独立的.这种思路类似于微处理器设计中的线程级并行(thread level parallelism).例如,一个二层交换程序,源 MAC 地址的处理和目的 MAC 地址的查找是完全独立的,可以并行处理.同时还可引入一些专用的硬件协处理器,以增加 IPP 的并行效果.另外,增加一些专用的处理指令也是提高 IPP 效果的有效手段[13].

(3) 指令级并行(instruction level parallelism,简称 ILP)

在网络处理器内部通过特殊的硬件结构完成处理器指令的并行执行,比如 VLIW 和超标量技术.此外,有些处理器还使用了线程级并行(thread level parallelism)技术[14,15].然而,经模拟研究表明,在一般非特定的整数科学计算应用中,ILP 是非常有限的,因此现在大部分网络处理器出于成本的考虑并没有大量使用 ILP 技术.

以上 3 种技术同时出现在网络处理器硬件结构和应用设计中.现在的商用网络处理器大都采用 PLP 并行技术,即将多个简单的 RISC 处理引擎(PE)集成到一个芯片上,并通过集成软件开发环境完成对多个处理引擎的编程[16].为了提高集成度,RISC 内核被设计得尽量简单,并且一般都不具备浮点运算指令及超标量等 ILP 技术.

在以上的 3 种并行技术中,硬件实现 PLP 是最简单的,但是对于相同的芯片面积存在着 PE 数目和 PE 复杂度的一个设计均衡(tradeoff)问题.如果单个 PE 很简单,则可以在相同的面积上集成多个 PE,但是单个 PE 的处理能力会比较差;如果减少 PE 的集成数目以提高单个 PE 的处理能力(包括引入 IPP 和 ILP 机制),将减小分组在单个 PE 中的处理时间,不过,系统的 PLP 并行度会受到一定的损失.一些模拟的结果表明[13],通过增加 PE 数目的做法虽然可以增加 PLP 效果,但是随着 PE 的增加其利用率将下降,同时增大分组重组缓冲将有助于提高 PE 的利用率,不过随着缓冲区增大,PE 利用率很快将达到饱和.因此对于网络处理器本身,在设计时将尽量减小 PE 的数目,同时要求 IPP 与 ILP 两种策略的合理应用.

对于应用设计,一般是依靠网络处理器提供的硬件并行特点,在分组处理的各个阶段适当地引入并行策略以提高分组处理效率.由于现在的商用网络处理器都提供 PLP 硬件支持,因此现有的大部分应用设计主要体现的是 PLP 这种技术.

### 2.3 建立Gigabit链路系统的挑战

使用网络处理器构建 Gigabit 链路系统(如可编程 QoS 路由器等)时,在系统的结构设计、策略使用、应用开发等各个层次存在着如下的问题.

- 并行策略所引入的问题

如第 2.2 节所述,网络处理器中大量采用并行处理技术,它虽然提高了分组处理的效率,但是仍带来了以下 3 点问题:

(1) 同步问题

在 PLP 策略中,分组交由不同的 PE 处理.但由于分组的特点,不同 PE 所处理的分组并不总是相互独立的,有时分组间也存在着一定的制约关系.这种制约关系主要体现在服务时序和资源操作两个方面,对于服务时序引起的同步问题可通过维护分组处理状态来解决,对于资源操作冲突则需要通过加锁机制来解决.同样的同步问题在 IPP 和 ILP 策略中也存在.

(2) 处理器资源调度问题

简单的讲,就是当一个待处理的分组到达时将由哪一个 PE 来服务以及如何来服务的问题.在调度时需要考虑包括系统负载和流量特性在内的多种信息,并要保证效率和公平性.目前此种资源调度算法一般分为静态和

动态两类.文献[6,17]中采用了静态算法,即 PE(或硬件线程)静态地对应不同的端口,不同的 PE(或硬件线程)只负责特定端口的收发操作.文献[18]中采用了基于 EHDA(enhanced hash-based distribution algorithm)的动态分配算法,算法考虑了 TCP 与 UDP 两种流的特性以及 PE 的负载情况.

实际上,网络处理器内的其他共享资源(如网络协处理器等)也存在着同样的问题.

(3) 模块间通信问题

为了解决同步以及系统控制等问题,网络处理器内部 PE 间的通信以及与外部设备间的通信越来越多,甚至成为系统性能的瓶颈.为了提高系统集成度,大部分网络处理器内部采用共享总线的通信方式,内部没有其他专用的数据通路.但是这种结构却严重地影响了系统的同步及通信策略.不过这一问题可以通过增加专用的数据总线得到改善,然而它将增加系统设计的开销.

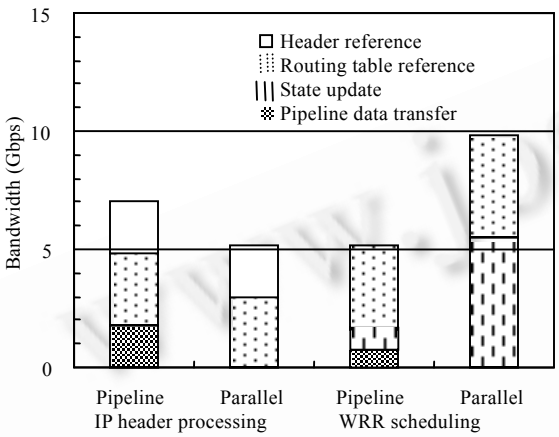



Fig.2 Parallel and pipeline processing cost

图 2 并行处理与流水线处理数据量

(4) 并行处理与流水线处理的关系

并行处理虽然可以提高分组处理的效率,但是它将引入大量的分组状态维护工作,如果分组状态存取过于频繁反而会降低分组处理的效率.例如,Diffserv 边界路由器中引入了基于类的调度(class based scheduling)和基于流的调度(per-flow scheduling).对于基于流的调度使用 WRR 算法[19],文献[1]给出了 2.4Gbps Diffserv 边界路由器进行普通报头处理和 WRR 调度时,并行处理与流水线处理的数据量,如图 2 所示.对于 WRR 调度,由于使用并行处理要频繁地对分组状态进行更新反而增大了运算量.因此,何时采用并行处理也是十分讲究的.

- 系统复杂度和 SOC(system on chip)优化问题

为了适应新一代 Gigabit 网络的灵活性和高性能,通常要求网络处理器系统采用 SOC 的设计方式.SOC 的基本目的就是为了降低系统的成本和提高整体性能.同时为了满足复杂的实时性处理要求,系统还需要在网络处理器内部储存一定数目的指令代码,如 Intel 的 IXP1200 为每个微引擎提供了 2KB 的指令空间以及一些用于高速运算和分组状态存储的存储区域[15].然而目前芯片设计技术还存在着诸多困难[20],在一块硅片上集成特性不同的异种功能单元电路增加了 SOC 的困难,同时,不同单元所占芯片面积也存在着彼此的均衡问题,其中最突出的就是 cache 大小和 IO 通道数目的设计.文献[5]对网络处理器的 SOC 优化得出了一些结论,并通过模型的方法分析了多处理器进行 SOC 时的性能,并认为对于特定面积的网络处理器其 cache 的大小和 IO 通道数目随着片上处理器的增加而减少,并存在一个最优配置.

- 延时隐藏问题(latency hiding)

对于 Gigabit 的链路来说,分组到达的间隔为几百纳秒,通常的 SRAM 访问时间在 10ns 左右,而处理器在访问存储器时处于空闲状态,因此存储器访问延时将带来巨大的处理器周期浪费.同样,这种延时出现在对协处理器以及其他一些外部资源的访问过程中,例如 Intel 的 IXP1200 对于 SRAM 的访问速度为 30 个微引擎周期,对片外 FIFO 的访问速度在 40 个微引擎周期左右.

通过调整并行执行的时序可以有效地将这些延时隐藏掉,其基本思路如图 3 所示.图中实线代表处理器实际运行的时间,虚线代表访问延时等待时间.通过调整处理时序,合理的利用访问等待时处理器的空闲时间,在宏观上可以达到隐藏访问延时的效果.

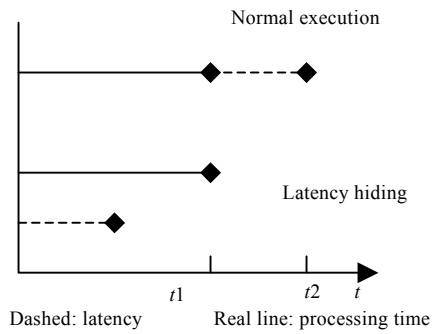



Fig.3 Principle of latency hiding

图 3 延时隐藏的基本原理

对于一般的计算机系统解决内存访问延时问题,可通过对内存地址预取来实现.但这种方法对于网络处理器中具有大量流操作的数据层面处理是没有什么效果的,而对其控

制层面的诸如路由表更新等操作却是非常有效的.此外,这种方法不需要额外的寄存器或线程调度等额外开销.在数据层面上,大部分商用网络处理器使用硬件多线程技术来解决内存访问延时的问题.在运行时,这些线程一部分处于活动状态,另一部分处于等待状态,当一条活动线程遇到访问延时情况就立即交出执行权,由另一条等待线程获得处理器资源.如 Intel 的 IXP1200,每个微引擎拥有 4 个线程,其中 1 条可执行;IBM 的 PowerNP,每个 PE 拥有 4 条线程,其中两条可执行.此外,网络处理器一般还引入零开销的硬件线程切换.

虽然多线程技术得到了广泛的应用,但是线程间的调度和使用策略仍对系统公平性和利用率起着重要作用.例如,在文献[6]的设计中线程被静态地指派,当一条线程触发内存访问等操作时,由系统硬件仲裁器决定下一个可以激活的线程.然而,模型分析显示这种简单的被动线程调度方式的处理器利用率受外部负载的重大影响,当分组长度增加时,处理器利用率将下降,其原因就是由于随着分组长度的增加,内存访问量增加,这种被动的线程分配方式往往导致处理器内部所有线程都处于访问延时等待状态.

## 3 网络处理器的应用研究

### 3.1 基于网络处理器的现有研究工作

网络处理器被认为是推动下一代网络向灵活性和高性能发展的核心技术,围绕其本身以及相关应用展开了大量的研究,并且已成为热门话题[21,22].OPENARCH 2002 专门为网络处理器开设了专栏进行讨论,INFOCOM 等重要国际会议也在可编程路由器等领域的研究中加大了网络处理器的比重.综合起来,现有的关于网络处理器研究成果主要有以下几方面.

- 基于网络处理器的路由器体系结构研究

随着网络处理器的出现,将其用于可编程 QoS 路由器的研究[14,15]得到了飞速的发展.其中主要是关于体系结构和相关算法的研究,并且在体系结构的研究上有了完整的结论和部分实现.其中比较成功的有,Princeton 大学的可扩充路由器 VERA[6]以及 Columbia 大学的 Genesis[17].以 Princeton 的 VERA 为例,它主要是基于 Diffserv 的体系结构,并兼顾可扩展性、兼容性、效率性三方面特点,不过总的看来,VERA 更侧重服务的灵活性和可扩充性.其分组处理逻辑流程如图 4 所示.图中 C 代表一个层次化的分类器(classifier),F 代表传送处理单元(forward),它负责完成分组的数据处理,S 为一个输出调度器(scheduler).在具体实现上,该项目采用一种基于 Intel IXP1200 和 PC 的层次化体系结构,如图 5 所示.Intel IXP1200 中的微引擎主要负责数据层面的实时数据处理,StrongArm 和 PC 负责更高层次的非实时数据处理.此外,该项目组还在体系结构的性能评价方面做了一定的工作,文献[23]中通过实际测量的手段考察了 VERA 各个层面的数据吞吐量、存储器访问频率、代码长度,并分别使用了 3 种不同的队列模型(单队列、多私有队列、多共用队列).在性能测试的同时,他们还对 3 种不同队列模型所引入的 QoS 调度机制和同步控制策略进行了一定的研究.在可扩展应用方面,VERA 给出了一些在 IXP1200 上实现的典型应用扩展所带来的开销(见表 2),进一步证明了其扩展的可行性.

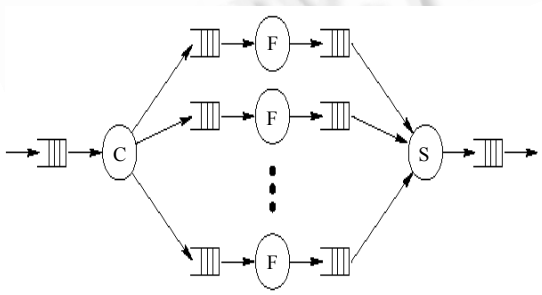



Fig.4 Packet processing flowchart of VERA

图 4 VERA 的分组处理流程图

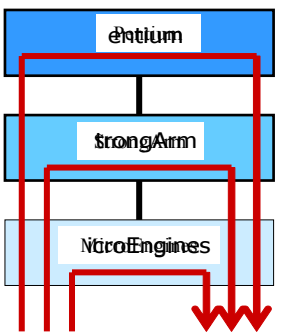


Fig.5 Hierarchy of VERA

图 5 VERA 层次化结构

**Table 3** Time and space expenses of typical extensions

**表 3** 一些典型扩展的时间及空间开销

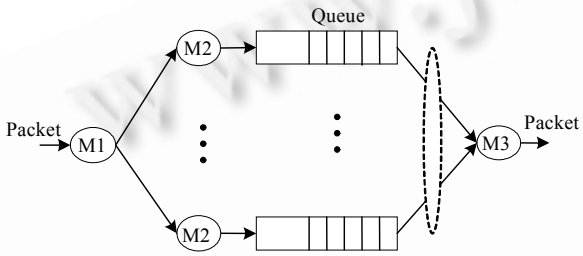
| | SRAM R/W (byte) | Register operations | Registers needed |
|---|---|---|---|
| TCP slicing | 24 | 45 | 7 |
| ACK monitor | 12 | 15 | 4 |
| SYN monitor | 4 | 5 | 0 |
| Smart dropper | 8 | 28 | 4 |

不过,VERA 在系统资源和处理器调度策略上只采用了简单的静态分配,一些代码的运行加载和卸载等动态特性与 Columbia 的 Genesis 中采用的 NetBind[24]相比还有不足.同时,VERA 虽然在体系结构上有了完整的声明,但是在综合 QoS 控制策略以及系统的理论建模等工作方面还不太完善.

- 综合 QoS 控制策略的研究

简单来讲,QoS 控制是指网络能够提供有保证的、可预测的数据传输服务,满足不同用户的应用需求.最近围绕着分组处理 QoS 控制已经展开了很多的研究,比如任务模型[25]、任务调度[26]、操作系统相关问题[27]、分组处理器(packet processor)体系结构[28].

然而这些研究大部分是面向网络处理器分组处理中的某一阶段.方案对一个阶段是最优未必对全局就是最优.比如从系统的角度考虑,我们要提高系统的利用率使之获得最大的吞吐量,同时还要确保不同用户间的服务公平性;从用户的角度,对一些分组延时抖动及丢失率则要给予充分的重视.




M1: Classification, M2: Buffer management, M3: Packet

Fig.6 Multiply queue system

图 6 多队列系统

清华大学参加的 Intel IXA 大学项目就是以 QoS 控制中最关键的缓冲管理及队列调度为对象进行综合优化方案的研究的.该项目以 Intel 的 IXP 系列网络处理器为实现平台,在研究综合方案的同时对其在网络处理器上的优化也做了一定工作.该项目采用如图 6 所示的基于分类(class based)的多队列系统,在系统的前端,M1 是一个多维分类器(classifier),它是整个系统的重要基础,并且要能够满足多维性(报文分类的字段)和实时性.根据充分利用分类规则、索引排序、增加预处理时间以及网络处理器的硬件特性,该项目已经构建了 5 维的实时分类算法.M2 是缓冲管理模块,它主要负责缓冲区的分配,并根据一定的策略对分组进行丢弃,这主要影响分组丢失率和公平性.目前,该项目已经在 Intel 的 IXP1200 上实现了 Tail Drop[29]和 PBS(partial buffer sharing)[30]等经典算法,同时还在 PBS 的基础之上提出了动态阈值的 DPBS(dynamic partial buffer sharing)算法[31].M3 是分组的输出调度器,它主要实现对链路带宽管理,即按照一定的规则来决定从队列中选择分组进行发送,它影响的主要性能参数包括带宽分配、时延和时延抖动等. 同样,项目在实现 WRR(weighted round robin)[19],DRR(deficit round robin)[32]等经典调度算法的同时还对这些算法与缓冲管理及分类的综合组合方案进行了优化研究. 在整体方案上还提出了基于 Diffserv 的拥有不同延时丢弃优先级的集成队列调度和缓冲管理方案[33],并用 QLT(queue length threshold)[34]调度算法和 PBS 及 RED 缓冲管理算法结合作为分析实例.

另外,在综合方案的性能评价方面,该项目也取得了一定的成果.文献[31]指出,综合 QoS 方案是一个多目标问题,每个目标不可能同时满足,文中还给出了综合吞吐量、队列延时、分组丢失率以及公平性在内的综合评价方案.其中比较重要的就是对 Power 公式[35]的改进.原始的 Power 公式是基于单队列系统的,增大 Power 值被认为是计算机网络的一个主要目标:

$$Power = \frac{Throughput^{a}}{Delay}, 0 < a < 1.$$

改进后的 Power 公式是基于多服务多类的,并在原有基础上综合考虑延时公平参数(delay fairness parameters,简称 DFPs)和丢失率公平参数(loss rate fairness parameters,简称 LFPs):

$$Power=\sum Power_i=\sum\frac{T_i^{\alpha}}{w_1\cdot\bar{d}_i+w_2\cdot\bar{l}_i},0<\alpha<1,$$

式中 $T_i,\bar{d}_i,\bar{l}_i$ 分别为分类 $i$ 的吞吐量、平均队列延时、平均丢失率;$w_1,w_2$ 为权重,它根据不同的环境和应用而变化.

- 局部资源调度算法的研究

现在大部分商用网络处理器系统是一个典型的多 RISC 内核的并行处理系统,同时存在着大量共享资源,如共享存储器、共享内存和协处理器等.依据系统特性对处理器、存储器等共享资源的有效调度已成为当前的热门课题.其中围绕着资源调度所展开的研究主要表现在两方面:一是围绕处理器所展开的负载平衡和处理器调度算法的研究;另一方面是根据系统特性(外部特性和内部特性)所展开的分组调度算法的研究.

关于负载平衡方面的算法分类一般遵循 Casavant 和 Kuhl 在文献[36]中所采用的基于策略的分类规则,另外,Lazowska 和 Zahorjan[37]在研究了特定的适应性(adaptive)负载平衡策略后发现,简单的基于阈值的负载平衡策略可以在本质上改善系统的性能.在网络领域,随着 Web Server,Web Caching 和分布式数字图书馆的出现[38,39],一些基于 HRW(highest random weight)的负载平衡算法也被应用到网络处理器中.不过这些基于 HRW 的方法大多从用户请求空间角度入手,系统适应性较差.但也有些算法从服务空间出发,同时结合 TCP/IP 分组在路由器中并行处理算法[40],并加上了一些动态执行预测的技术,很好地解决了适应性的问题.综合来看这类算法具有以下几个特性[41]:

(1) 基于一些随机映射(random mapping)算法,如 Hash-Based 和 HRW 等.同时要保证一定的容错性(faults tolerance).

(2) 定义具有动态特性的系统利用率函数,即将系统利用率定义为时间的函数.

(3) 定义触发阈值,只有当系统利用率达到阈值时才触发负载平衡策略.这样是为了避免平衡策略触发得过于频繁而降低系统的性能.

但这类算法也有它的局限性,它通常是基于流映射的(flow mapping)[42],虽然它不需要对流状态进行保留,可是它假定流与流之间是同构的,对流处理预测时采用相同的算法.这对于采用 Diffserv 结构的路由器等基于多分类的应用是不适宜的.

对于分组调度算法,简单来讲,就是当一个分组到来时将进行怎样的处理,哪一个分组先处理,哪一个分组后处理.通常,基于网络处理器的分组处理系统逻辑结构如图 7 所示[43].

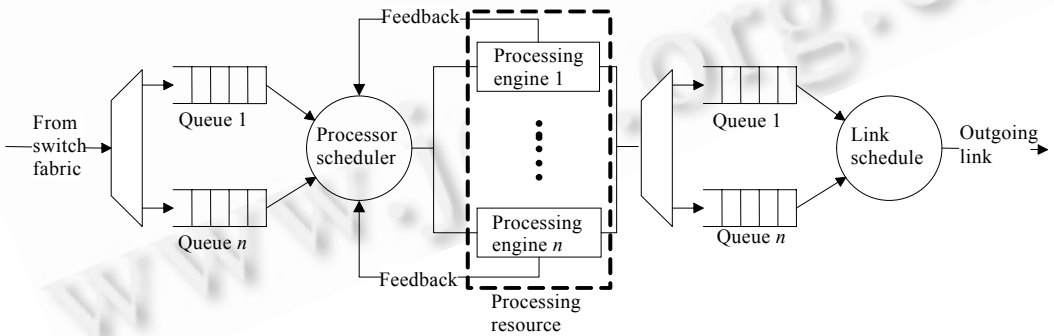



Fig.7 Packet processing system outline

图7 分组处理系统的结构图

系统采用一种类似输入输出队列的结构(CBIO),输入输出队列间是一组并行且同构的服务员,在这组服务员前设有一个处理器调度器(processor scheduler)负责分组的处理调度,在输出端队列后设有一链路调度器(link scheduler)负责分组的发送调度.分组调度及处理算法的核心就是处理器调度器和链路调度器的算法设计.而传统的分组处理系统只是简单的存储-转发过程,仅涉及到链路调度器.因此,对于网络处理器系统则要两方面综合考虑.

无论是哪种调度器,其核心问题就是对分组处理时间进行预测[44],对于具有多个通用处理器(general

purpose processor)的网络处理器系统尤为明显.在链路调度器方面,分组调度策略(基于带宽调度)已经有了广泛的研究[44],并且一些性能评价参数如端到端延时、公平性等也和 GPS(general processor sharing)策略[45]进行了相应的比较.但是一些基于 PFQ(packet fair queue)的分组调度策略则不能被用来做处理器的调度,因为诸如 WFQ,$WF^2Q$[46]等使用了虚拟时间的概念,这些虚拟时间要在调度器中被精确地更新,也就是说,调度器需要精确地知道分组处理所耗费的时间,而对于处理器是十分困难的.虽然一些算法,诸如 SFQ(start-time fair queuing)[47]不需要知道分组处理的时间,但是 SFQ 在最坏情况下,系统延时将随着流数目的增加而增加,并在一些流相关负载(correlated cross-traffic)下变得更糟[48].

因此一些研究在避免模拟 GPS 策略复杂性的同时尽力提高算法理论延时和公平性等系统性能参数[49].为此,这些算法使用预测的分组服务时间作为调度依据,文献[43]中给出了一种常用的预测分组处理开销的算法:

$$c=\alpha_a+\beta_a\cdot l_i,$$

$$\beta_a=\frac{\sum_n c_i\cdot l_i-\sum_n c_i\cdot\sum_n l_i/n}{\sum_n l_i^2-\sum_n l_i\cdot\sum_n l_i/n},$$

$$\alpha_a=\sum_n c_i-\beta_a\cdot\sum_n l_i,$$

式中 $c$ 是长度为 $l$ 的分组进行 $a$ 处理的开销,$\alpha_a$ 为平均每个分组进行 $a$ 处理的开销,$\beta_a$ 为平均每字节进行 $a$ 处理的开销;调度器运行时为每一种处理 $a$ 维护如下的统计信息:$\sum c_i,\sum l_i,\sum c_i^2,\sum l_i^2,\sum c_i\cdot l_i$.

在分组调度策略的选择上,链路调度器大部分使用现有的 PFQ 策略,而处理器调度大部分采用 FCFS(first come first serve),T-Opt(throughput-optimal),LA(locality-aware),LAP(locality-aware predictive)[50].

- 网络处理的性能评价

网络处理器性能评价研究的主要目的是为了在众多系统方案中选择一个最适合需要的方案,同时对现有系统方案性能的缺陷和瓶颈进行改进以及提高,再对未来设计的系统进行性能预测.具体的性能评价工作包括两方面:一方面是对系统应用方案进行性能评价;另一方面是对网络处理器本身进行性能评价.前者的目的是在现有系统上评价应用方案的性能,为应用方案的设计与改进提供依据.后者则是从网络处理器本身出发,对其体系结构作性能分析,为网络处理器的设计与体系结构的改进提供必要的支持.

现在商业网络处理器正在以惊人的速度增长,已经有 30 个厂商超过 500 个设计[51],这些网络处理器的体系结构、内存及系统接口都不尽相同,这给性能评价和方案间的相互比较带来了巨大的困难[10].因此在具体的性能评价工作上一般采用实际测量(benchmark)和模型两种评价方式.由于网络处理器的高异构性,现有的大部分性能评价工作都是基于实际测量的.Nemirovsky 在文献[52]中讨论了实际测量的要求以及所面临的挑战, 同时为网络处理器的测量方法定义了一组评价指标和测量软件的设计原则. 除此之外,Mel Tsai[53]等人还给出了用于网络处理器的通用测量软件的两项基本原则:测试标准的通用性;测试例程符合真实网络应用.

在具体测量软件方面,目前一些流行的测量软件如 SPEC[54],Dhrystone[55],MediaBench[56]是面向同构系统的,对于网络处理器系统并不太适合,因此出现了一些专门评价网络处理器的软件[7,9,57,58],这些软件按照不同的分类提供了一系列的测试程序,并且使用它们已经得到了一些关于网络处理器的系统负载[7]、嵌入式集成[5]等重要数据.其中比较著名的测试工具有 CommBench[7],NetBench[9],EEMBC[57]及 NPF-BWG[58],它们的技术特性见表 4.

**Table 4** Characteristics of typical benchmark suits

**表 4** 典型测试软件技术特性

| | CommBench | NetBench | EEMBC | NPF-BWG |
|---|---|---|---|---|
| Number of benchmarks | 8 | 9 | 34 | |
| Benchmark classification basis | Workload | Processing levels | Application oriented | Processing levels |
| Executable description | Yes | Yes | No | Unknown |
| Method of benchmark choice | Yes | Yes | Yes | Yes |
| Environment and system benchmark | No | No | No | Yes |

它们基本上都是面向网络处理器应用的,除 NPF-BWG 外都没有从系统的角度对整个系统进行评价.比如,在 CommBench 提供的 8 组测试程序中,有 4 个被用作报头处理测试,另外 4 个被用来作负载测试,它们很好地反映了第 2.1 节中提到的网络处理器的系统特性.但是 CommBench 是基于单服务员且没有明确地说明如何将

其应用于一些多内核的网络处理器,如 Intel 的 IXP1200.类似地,NetBench 也定义了 9 组测试程序,并将这 9 组程序分为底层、IP 层、应用层 3 个不同处理层次,经过这种分类,NetBench 的测试在底层结构(micro-architecture)反映了网络处理器的异构特性(heterogeneous),并且已经在 SimpleScalar[59]模拟器和 Intel IXP1200 系统上得到了一定的结果,但是对于其他层次,这种异构特性并不十分明显.EEMBC 则从网络应用对象出发,给出了面向工业、普通消费者等终端对象的不同测试程序,其中包括一些 OSPF 等协议测试.

在理论建模方面,Patrick Crowley[60]等人给出了一种基于可编程网络接口的网络处理器的模型,同时将模型划分为系统模型、应用模型、流量负载模型 3 个子模型,并对模型的负载特性作了详细的评价分析.但是他们的模型还不十分完善,只涉及到真实网络处理器系统的一小部分,并且没有将系统功能模块与环境模块分开考虑.总的来看,目前围绕着网络处理器所展开的性能评价工作大部分是基于实际测量的,模型化的方法还不十分完善.

- 网络处理器的应用

网络处理器的应用研究得到了广泛的开展,与此同时,涌现出了一些成功的应用范例.这些应用集中地体现在以下几个方面:

(1) 基于网络处理器的智能路由交换设备.包括路由器体系结构的研究和功能模块的构建.比如 Alcatel and Asante Technologies 使用 IBM 的 PowerNP 构建其核心路由及交换设备[61],Broadband Access Systems,Mayan Systems,NorthChurch[62]及 Cloudshield Technologies[63]使用 Intel 的 IXP1200 构建起网络路由设备.

(2) 智能安全设备的应用:防火墙、VPN 及入侵检测设备等.

(3) 无线网络的应用:Ad Hoc 无线网络中的路由设备,3G 协议栈的实现等.

(4) 系统的监测控制或模拟设备:一些研究机构将网络处理器作为其系统测试和实验设备,比如 Empirix 使用 Motorola 的 C-5 作为其网络模拟器用于在线速下获得网络延时、抖动、丢失率等参数[64].

- 网络处理器其他相关研究

(1) 主动网络和可编程网络的研究

这些研究包括执行代码的动态移植和发布、代码的安全执行、资源共享调度等.其中一些研究成果也被应用到网络处理器中,比如 Columbia 大学的 Genesis 将可编程虚拟网络中的概念移植到网络处理器系统中,在 Intel IXP1200 的平台上构建了可在运行时装卸代码的动态数据层面[17].

(2) 处理器系统体系结构的研究

这部分研究与网络处理器相关的主要是对处理器内部组件的研究(如 RISC 内核、协处理器等)以及内存接口、高速数据总线等接口的研究.这些研究主要是着手在硬件体系结构上解决网络处理器的性能瓶颈.

(3) 第二代商用网络处理器的开发与研制

大部分厂商都展开了对下一代商用网络处理器的研究,并将其产品定位到核心网络设备.其中以 Intel IXP 2K 系列为代表的第二代网络处理器并没有增加过多的并行处理内核,而是大幅度提高了处理器的执行效率,同时加强了多处理器间的通信能力,并增设一些硬件指令执行单元,如乘法硬件指令.不过,这些研究与改进主要还是面向提高报头处理效率,至于加密等复杂的数据计算处理暂时还没有专门涉及.

### 3.2 网络处理器的发展方向和相关工作

随着网络应用的数据码率呈指数性增长,网络处理器的研究与发展越来越受到网络中边缘和独立节点的影响,今天的核心设备支持 2.5Gbps 链路,到了下一代就是 10Gbps 甚至是 40Gbps,而边缘设备虽然在链路带宽方面落后于核心设备一代,但是它将承载更为复杂的处理任务.围绕着网络处理器所开展的相关工作也将迎合网络中边缘设备的发展需求.下面将从网络应用、网络处理器结构和相关研究 3 个方面展望网络处理器的发展.

(1) 网络应用的发展

如今的网络应用除了链路速度不断提高之外,传统 OSI 模型也已被打破,越来越多的高层信息被用来决定低层次操作.比如,Web 交换或者 URL 负载平衡使用 TCP 端口和 HTTP 协议信息来决定相应的路由操作.但由于高层协议和应用一般遵循端到端(peer to peer)的原则,它们的变化非常频繁而且很难预测.因此,对网络处理

器灵活性和性能的要求也越来越强.

另外,IPv6 的出现给现有基于 IP 地址的查找运算带来了巨大变化,其中多维查找算法(多个字段)将越来越普遍,平均每关键字查找数据量将大于 300bits,而现有的协处理器最多只能处理每关键字 288bits 的数据量[10].为此,人们对 IPv6 的高速路由查找和分类算法方面的研究正在积极地展开.此外,IPv6 需要实现 IPSec 协议,因此在通常的报头处理之余,网络处理器还需进行大量的加密解密运算.所以,IPv6 的出现会使下一代网络处理器向能胜任大计算量任务的方向发展.

最后,随着 MPLS 的加速应用,MPLS 将作为 Ethernet,SONET,ATM 等多种主流协议的协同工作标准,作为标签交换路由器(label switch router)将同时支持多种协议,而网络处理器的灵活高效性必将在标签路由系统中得到广泛应用,但是网络处理器这种通用处理计算结构是否能很好地适应其特殊的交换式结构还有待研究,相关的算法和体系结构研究也必将得到重视.

(2) 网络处理器自身体系结构的进化

随着网络应用的发展和摩尔定律的限制,网络处理器自身体系结构也在不断地进化,主要表现在并行结构、协处理器、通信体系结构这 3 个方面.

♦ 并行结构.数据的增长将会超过摩尔定律的增长,第 2.2 节已分析过并行结构将是弥补这一缺陷的有效手段.随着半导体工艺的改进,新一代网络处理器的运行时钟已经超过了第一代网络处理器 1 倍以上,其中 Broadcom 的 Mercurian 运行速度达到了 1GHz[65].而新一代网络处理器中 PE 的数目并不会有过多的增长,而是向着提高 PE 速度方向发展.

♦ 协处理器的广泛应用.网络处理器本身是基于 ASIP 技术的,为了提高性能网络处理器中部分常用单元将以 ASIC 协处理器的形式出现,同时原先一些片外协处理器也将逐步集成到网络处理器芯片内部.另外,协处理器所处理的任务也将越来越复杂,Vissers 等人[66,67]已经展示了一种具有多媒体处理协处理器的设计.

♦ 通信体系结构:第一代网络处理器的通信体系结构一般采用简单的共享总线、共享内存的方式,新一代的网络处理器不但增强了处理器内部 PE 间的通信能力(使用专用数据通路和寄存器),而且在处理器外部接口通信能力的提高方面也作了很多的工作,比如 Intel 的 IXP2400 增加了交换光纤通信接口.同时为了解决多 PE 竞争内存的瓶颈,一些设计还采用了增加内存操作通道的方法进行改进.

表 5 以 Intel 的 IXP2400 与 IXP1200 的比较为例,说明了以上 3 个方面的改进.

**Table 5** Comparison between IXP1200 and IXP2400 network processor

**表 5** IXP1200 与 IXP2400 网络处理器的比较

| | | IXP2400 | IXP1200 |
|---|---|---|---|
| Parallel structure | Core frequency | 600Mhz | 200Mhz |
| | Number of micro engines | 8 | 6 |
| Co-Processor | Hardware multiplication | Yes | No |
| | ATM segmentation and reassembly | Yes | No |
| Inter-Communication | Neighbor registers | Yes | No |
| | Local memory | 640Byte | No |

(3) 相关研究工作的开展

第 3.1 节总结了一些现有的工作,虽然有些已经取得了一定成果,但是仍有很多问题有待解决,集中地体现在以下几点:

♦ 综合资源调度方案的研究.目前大部分资源调度都是面向某一局部的[43,50],从系统角度看,资源调度方案以及相应的综合 QoS 控制策略将是未来一段时间内研究的热点.

♦ PE 内硬件线程的调度以及延时隐藏技术的研究.第 2.3 节中提到了延时隐藏技术,同时 PE 内多硬件线程的结构正是为了达到延时隐藏这一要求.但现在大多数设计[6,17]只是静态地使用 PE 内的线程,仅依靠线程硬件执行特点而没有使用任何主动线程调度算法,这使得延时隐藏的效果受到了一定影响.今后将会有包括 PE 线程调度在内的更多的延时隐藏技术受到重视,相应的理论模型以及分析工作也将得到广泛开展.

♦ 网络处理器性能评价问题.正如第 3.1 节中所提到的,目前网络处理器的性能评价主要依靠实际测量的

方法,今后其性能评价研究将向模型、模拟、测量三者结合的方向发展.完整的系统模型和相应的模拟器将是性能评价的首选.同时,测量软件也将逐步完善,测量评价的标准也将得到统一.

♦ 网络处理器软件开发工具和操作系统.随着网络处理的发展,一些相关的开发工具也得到了飞速发展,大部分网络处理器已经可以使用 C 语言进行开发,同时也出现了一些支持网络处理器运行的操作系统.在开发工具方面,将向着集成化开发环境(IDE)方向发展,如 Intel 的 WorkBench.并且在开发套件里提供一些专门用于 IP 协议的 API 和子程序.另外,Network Processing Forum 也正在为网络处理器开发通用的软件接口.在操作系统方面,将从现有的嵌入式操作系统(如 Linux,VxWorks 等)向专用的网络处理器操作系统过渡,比如 Princeton 的 Vera 所使用的 Scout OS[27].

## 4 总 结

本文立足于网络处理器自身及其应用研究两个方面,对网络处理器作了系统的概述.首先介绍了网络处理器的硬件处理特性和系统结构特点.接着,从应用的角度出发,指出了应用网络处理器构建 Gigabit 链路系统所面临的问题,分析了一些解决问题的基本方法.同时,本文重点总结了围绕网络处理器所开展的应用研究工作,并涉及到当前研究与应用中的一些不足.最后,本文结合现有的研究成果,从网络处理器自身和应用两个角度展望了其研究发展的方向.

**References**:


[1] Shimonishi H, Murase T. A network processor architecture for flexible QoS control in very high-speed line interfaces. In: Proceedings of the 2001 IEEE Workshop on High Performance Switching and Routing (HPSR 2001). Dallas: IEEE Computer Society Press, 2001.402~406.

[2] Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. An architecture for differentiated services. IETF RFC 2475, 1998.

[3] Floyd S, Jacobson V. Random early detection gateways for congestion control. IEEE/ACM Transactions on Networking, 1993, 1(4):397~413.

[4] Thiele L, Chakraborty S, Gries M, Künzli S. Design space exploration of network processor architectures. In: Patrick C, *et al*., eds. Proceedings of the 8th International Symposium on High Performance Computer Architecture. Cambridge, MA: Morgan Kaufmann Publishers, 2002.

[5] Wolf T, Franklin MA, Spitznagel E. Design tradeoffs for embedded network processors. Technical Report, WUCS-00-24, St. Louis: Department of Computer Science, Washington University, 2000.

[6] Karlin S, Peterson L. VERA: an extensible router architecture. Computer Networks, 2002,38(3):277~293.

[7] Wolf T, Franklin MA. CommBench—a telecommunica-tions benchmark for network processors. In: Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software. Austin, TX, 2000. 154~162.

[8] Hecht J. Wavelength division multiplexing.1999. http://www.technologyreview.com/articles/hecht0399.asp.

[9] Memik G, Mangione-Smith B, Hu W. NetBench: a benchmarking suite for network processors. In: Proceedings of the International Conference on Computer-Aided Design (ICCAD). San Jose: IEEE Computer Society Press, 2001. 39~43.

[10] Shah N. Understanding network processors [MS. Thesis]. Berkeley: Department of Electrical Engineering and Computer Sciences, University of California, 2001.

[11] Nie XN, Gazsi L, Engel F, Fettweis G. A new network processor architecture for high-speed communications. In: Proceedings of the IEEE workshop on signal processing systems. IEEE Computer Society Press, 1999. 548~557.

[12] McAuley A, Francis P. Fast routing table lookup using CAMs. In: Proceedings of the Infocom'93, Vol 3. IEEE Computer Society Press, 1993. 1382~1391.

[13] Liu H. A trace driven study of packet level parallelism. In: Proceedings of the International Conference on Communications (ICC). New York, NY: IEEE Computer Society Press, 2002. 2191~2195.

[14] IBM Corp. Rainier network processor. 2000. http://www.ibm.com/.

[15] Intel Corp. Intel IXP1200 Network Processor. 2002. http://developer.intel.com/design/network/products/npfamily/ixp1200.htm.

[16] Wolf T, Turner JS. Design Issues for High-performance active routers. IEEE Journal on Selected Areas in Communications, 2001, 19:404~409.

[17] Kounavis ME, Campbell AT, Chou S, Modoux F, Vicente J, Zhuang H. The genesis kernel: a programming system for spawning network architectures. IEEE Journal on Selected Areas in Communications, 2001,19(3):511~526.

[18] Wang J, Nahrstedt K. Parallel IP packet forwarding for tomorrow's IP routers. In: Proceedings of the 2001 IEEE Workshop on High Performance Switching and Routing (HPSR 2001). Dallas: IEEE Computer Society Press. 2001. 353~357.

[19] Katavenis M, Sidiropoulos S, Courcoubetis C. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. IEEE Journal on Selected Areas in Communication, 1991,9(8):1265~1279.

[20] Keutzer K. Enabling Fully Programmable Embedded System Solutions. Presentation. Gigascale Silicon Research Center Annual Review. 1999.

[21] Network Processor Forum. http://www.npforum.org.

[22] Network Processor Conference. http://www.networkprocessors.com.

[23] Spalink T, Karlin S, Peterson L, Gottlieb Y. Building a robust software-based router using network processors. In: Proceedings of the 18th SOSP. ACM Press, 2001. 216~229.

[24] Campbell AT, Chou ST, Kounavis ME, Stachtos VD. NetBind: a binding tool for constructing data paths in network processor-based routers. In: Proceedings of the 5th IEEE Conference on Open Architectures and Network Programming (OPENARCH). IEEE Computer Society Press. 2002. 91~103.

[25] Thiele L, Chakraborty S, Gries M, Maxiaguine A, Greutert J. Embedded software in network processors models and algorithms. In: Proceedings of the 1st Workshop on Embedded Software. Lecture Notes in Computer Science 2211, Lake Tahoe, CA: Springer-Verlag, 2001. 416~434.

[26] Qie X, Bavier A, Peterson L, Karlin S. Scheduling computations on a software-based router. In: Proceedings of the SIGMETRICS. IEEE Computer Society Press, 2001. 13~24.

[27] Peterson L, Karlin S, Li K. OS support for general purpose routers. In: Proceedings of the 7th Workshop on Hot Topics in Operating Systems. IEEE Computer Society Press, 1999. 38~43.

[28] Lahiri K, Raghunathan A, Dey S. System level performance analysis for designing on-chip communication architectures. IEEE Transactions on Computer Aided-Design of Integrated Circuits and Systems, 2001,20(6):768~783.

[29] Peterson LL, Davie BS. Computer Networks: a System Approach. 2nd ed., Morgan Kaufmann Publisher, Inc., 2000.

[30] Causey JW, Kim HS. Comparison of buffer allocation schemes in ATM switched: complete sharing, partial sharing and dedicated allocation. In: Proceedings of the International Conference on Communications. IEEE Computer Society Press, 1994. 1164~1168.

[31] Jiang Y, Lin C, Wu J, Sun X. Integrated performance evaluation criteria for network traffic control. In: Proceedings of the IEEE Symposium on Computers and Communications. IEEE Computer Society Press, 2001. 438~443.

[32] Shreedhar M, Varghese G. Efficient fair queueing using deficit round-robin. IEEE Transactions on Networking, 1996,4(3):375~385.

[33] Lin C, Sheng L, Wu J, Xu MW. An integrative scheme of differentiated services. In: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2000). IEEE Computer Society, 2000. 441~448.

[34] Yong J, Kim YH. Performance analysis of a hybrid priority control scheme for input and output queueing ATM switches. In: Proceedings of the IEEE INFOCOM'98, Vol 3. IEEE Computer Society Press, 1998. 1470~1477.

[35] Giessler A, Haanle J, Konig A, Pade E. Free buffer allocation-An investigation by simulation. Computer Networks, 1978,2: 191~204.

[36] Casavant TL, Kuhl JG. A taxonomy of scheduling in general purpose distributed computing systems. IEEE Transactions on Software Engineering, 1988,14(2):141~154.

[37] Eager DL, Lazowska ED, Zahorjan J. Adaptive load sharing in homogenous distributed systems. IEEE Transactions on Software Engineering, 1986,SE-12(5):662~675.

[38] Barish G, Obraczka K. World wide web caching: trends and techniques. IEEE Communications Magazine, 2000,38(5):178~184.

[39] Goldszmidt G, Hunt G. Scaling Internet services by dynamic allocation of connections. In: Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management. IEEE Computer Society Press, 1999. 24~28, 171~184.

[40] Koufopavlou OG, Tantawy AN, Zitterbart M. Analysis of TCP/IP for high performance parallel implementations. In: Proceedings of the 17th IEEE Conference on Local Computer Networks. Minneapolis: IEEE Computer Society Press, 1992. 576~585.

[41] Kencl L, Le Boudec J-Y. Adaptive load sharing for network processors. In: Proceedings of the IEEE INFOCOM 2002. IEEE Computer Society Press, 2002. 545~554.

[42] Thaler DG, Ravishankar CV. Using name-based mappings to increase hit rates. IEEE/ACM Transactions on Networking, 1998,6(1): 1~14.

[43] Pappu P, Wolf T. Scheduling processing resources in programmable routers. In: Proceedings of the IEEE INFOCOM 2002. IEEE Computer Society Press, 2002. 104~112.

[44] Zhang H. Service disciplines for guaranteed performance service in packet switching networks. Proceedings of the IEEE, 1995, 83(10):1374~1396.

[45] Parekh AK, Gallager RG. A generalized processor sharing approach to flow control: the single node case. In: Proceedings of IEEE INFOCOM'92. IEEE Computer Society Press, 1992. 915~924.

[46] Bennett J, Zhang H. Worst case fair weighted fair queuing. In: Proceedings of the IEEE INFOCOM'95. IEEE Computer Society Press, 1995. 120~128.

[47] Goyal P, Vin HM, Cheng HC. Start-Time fair queuing: a scheduling algorithm for integrated services packet switching networks. In: Proceedings of the ACM SIGCOMM. ACM Press, 1996. 157~168.

[48] Bennett JCR, Zhang H. Hierarchial packet fair queuing algorithms. In: Proceedings of the ACM SIGCOMM. ACM Press, 1996. 43~56.

[49] Golestani SJ. A self clocked fair queuing scheme for broadband applications. In: Proceedings of the IEEE INFOCOM'94. IEEE Computer Society Press, 1994. 636~646.

[50] Wolf T, Franklin M. Locality-Aware predictive scheduling of network processors. In: Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE Computer Society Press, 2001. 152~159.

[51] Matsumoto C. Technical trial-by-fire awaits NPUs. EE Times. http://www.eetimes.com/story/OEG20011221S0027.

[52] Nemirovsky A. Towards characterizing network processors: needs and challenges. XStream Logic, Whitepaper, 2000.

[53] Tsai M, Kulkarni C, Sauer C, Shah N, Keutzer K. A benchmarking methodology for network processors. In: Patrick C, *et al*., eds. Workshop on Network Processors. Cambridge, MA: Morgan Kaufmann Publishers, 2002.

[54] Standard Performance Evaluation Corporation (SPEC), http://www.spec.org/.

[55] Weicker R. Dhrystone benchmark: rationale for version 2 and measurement rules. SIGPLAN Notices, 1988,23(8):1~2.

[56] Lee C, Potkonjak M, Mangione-Smith W. MediaBench: a tool for evaluating and synthesizing multimedia and communications systems. In: Proceedings of the International Symposium on Microarchitecture. IEEE Micro-30, IEEE Computer Society Press, 1997. 330~335.

[57] Embedded Microprocessor Benchmark Consortium (EEMBC). http://www.eembc.org/.

[58] Audenaert S, Chandra P. (NPF Benchmarking Working Group co-chairs), Network processors benchmark framework. NPF Benchmarking Workgroup, http://www.npforum.org/.

[59] Burger D, Austin T. The simpleScalar tool set. Version 2.0. Technical Report, CS-TR-97-1342. University of Wisconsin, 1997.

[60] Crowley P, Baer J-L. A modeling framework for network processor systems. In: Patrick C, *et al*., eds. Proceedings of the 8th International Symposium on High Performance Computer Architecture. Cambridge, MA: University of Washington, 2002.

[61] Mathew J. IBM ropes in partners for network processors. Electronic News Online. 2000. http://www.e-insite.net/electronicnews/index.asp?layout=article&articleId=CA49443.

[62] Matthew J. Network processor companies face the same tough issues. Electronic News Online. 2000. http://www.e-insite.net/electronicnews/index.asp?layout=article&articleId=CA49900.

[63] Matsumoto C. CloudShield pushes net processors to next performance level. EE Times. 2001. http://www.eetimes.com/story/OEG20010625S0088.

[64] Austin TX, Wilmington MA. Motorola's C-Port network processor chosen as key technology in the hammer packetSphere platform from empirix. 2001. http://apspg.motorola.com/press/022801/cport.html.

[65] Atondo A. Fabless semiconductor start-up SiByte announces breakthrough MIPS64 microprocessor core——delivers up to 1 GHz at less than 2.5 Watts world's most power-efficient core in its performance class. Press Release, http://www.sibyte.com /pressroom/docs/pr_20000612_sb1.html.

[66] van Eijndhoven JTJ, Sijstermans FW, Vissers KA, Pol EJD, Tromp MJA, Struik P, Bloks RHJ, van der Wolf RHJ, Pimentel AD, Vranken HPE. TriMedia CPU64 Architecture. In: Proceedings of the International Conference on Computer Design (ICCD'99). Austin: IEEE Computer Society Press, 1999. 586~592.

[67] Hekstra GJ, La Hei GD, Bingley P, Sijstermans FW. TriMedia CPU64 design space exploration. In: Proceedings of the International Conference on Computer Design (ICCD'99). Austin: IEEE Computer Society Press, 1999. 599~607.