第33卷 第18期 计 算 机 工 程 2007年9月
Vol.33 No.18 Computer Engineering September 2007
·网络与通信· 文章编号：1000—3428(2007)18—0116—03 文献标识码：A 中图分类号：TP393

网络处理器并行度研究

秦思林，张盛兵，周昔平
(西北工业大学计算机学院，西安 710071)

摘 要：介绍了网络处理器基本配置方式：并行配置和串行配置。在并行配置的基础上，分析了 IPv4 对网络处理器报文处理的要求，提取了不同网络处理器报文处理的共同特征。讨论了在网络处理器饱和工作条件下内部线程个数和引擎个数之间的关系以及网络处理器的效率问题，为在相同的资源条件下，如何提高其性能作出了前期研究。

关键词： 网络处理器；IPv4；微引擎；线程；协处理器

Research of Parallel Degree in Network Processor

QIN Si-lin, ZHANG Sheng-bing, ZHOU Xi-ping
(Institute of Computer, Northwestern Polytechnical University, Xi'an 710071)

【Abstract】 This paper introduces the basic configuration mode of network processor, including parallel configuration and serial configuration. Based on parallel configuration, the action of router in IPv4 is analyzed. After abstracting the common characteristic of diverse NP of processing packet, the relationship between the number of thread and number of micro-engine is analyzed. The efficiency of the NP is discussed with focus on how to improve the efficiency under limited resources.

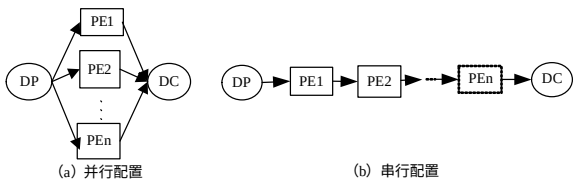
【Key words】 network processor; IPv4; micro-engine; thread; co-processor

为了使网络处理器能够达到更高的报文处理能力，大多数网络处理器都具有多个微引擎。微引擎越多，每个微引擎的线程就越多，网络处理器能够同时处理的报文数目也就越多，网络处理器的报文处理能力也就增强了。但根据Amdahl定律[1]：仅仅对系统的一部分进行改进对整个系统性能的提高是有限的，即在PE的个数达到某个数目以后，再增加PE对性能没有明显的改善；另外，由于受集成电路制造工艺的影响，网络处理器的微引擎个数和每个微引擎线程的个数并不是能够无限制地增加的，况且还会带来线程切换开销增加。因此，有必要研究网络处理器性能和PE数目以及线程数目的关系，以确定在不同的情况下如何配置PE的数目和线程的数目，使系统的性能达到最优。

1 网络处理器报文处理时间的分析

1.1 网络处理器的2种配置

网络处理器的结构比较复杂，目前已经面世的网络处理器在结构上也存在较大的差异。由于本文讨论的是网络处理器并行报文处理的一些共同特征，因此有必要对网络处理器报文处理结构进行抽象，提取出网络处理器报文处理的一些共同特征。忽略那些对本研究影响不大的因素，既不失去其一般意义，又使研究具有针对性。网络处理器报文处理模块结构一般分为2种常用的配置：并行配置和串行配置[1](图1)。



(a) 并行配置 (b) 串行配置

DP: 包分派部件; PE1,PE2,…,PEn; 包处理单元; PC：包收集单元

图1 网络处理器报文处理配置

由于并行配置处理单元运用比较广泛，本文只针对并行配置结构的报文处理时间进行研究。

1.2 IPv4对网络处理器报文处理的要求

RFC1812 定义了 IPv4 对路由器报文处理的要求。主要内容如下:(1)对入栈报文进行 IP 头的校验和，如果错，则丢弃；(2)验证并修改报文头的 TTL(time to live)时间，如果小于 0 则丢弃该报文。防止该报文在网络中无限传播；(3)利用相关协议对该报文进行分组；(4)将修改好的报文头和源报文组装，并将其送到对应的组；(5)查找转发表，找出该报文在网络的下一跳 IP 地址，以及使用哪个输出端口发送该报文；(6)分段处理。如果报文长度大于该网络处理器(MUT)，则需要进行分段处理。在组装时候要有一定的机制保证其发送顺序；(7)对于特殊报文，应给予特殊处理方法，如 IP 广播报文等。

根据以上要求，将报文处理分为以下 3 个阶段(图 2)：

(1)拆分和校验阶段($\Delta t1$)。PE 被分派 1 个报文以后，分离出 IP 报文头信息(例如 TTL 域、头校验和、IP 选项域以及源和目的 IP 地址等)。检查各个字段是否满足标准的 IP 报文格式，主要检查是否为 IP 报文，报文头格式是否正确，是否需要分段，是否是组播报文，TTL 域是否为 0 等。若不满足，则作特殊标记，将其作为特殊报文交给路由器操作系统去处理。这一阶段在 PE 上完成。

(2)查表阶段($\Delta t2$)。主要是完成查找转发表，路由查找等工作。这一阶段主要在协处理器上完成。

基金项目：国家自然科学基金资助项目(60570101)
作者简介：秦思林(1980－)，男，硕士研究生，主研方向：专用集成电路设计；张盛兵，教授；周昔平，博士研究生
收稿日期：2006-09-20 **E-mail：**qinsilin@mail.nwpu.edu.cn

(3)转发阶段($\Delta t3$)。当处理器完成了报文的路由查找工作，将获得的下一跳的端口和目的地址加入到报文头中，同时修改相应的域，将修改好的报文头和有效负载组装，然后按一定的规则转发出去。这一阶段也是在 PE 上完成的。

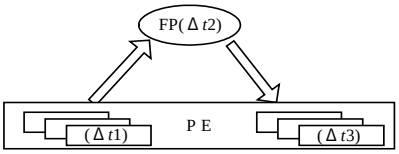


图 2　路由器报文处理时间模型

在该模型中，FP，PE 均代表集合，并分别表示为协处理器集合和微引擎集合，同时每个微引擎有多个线程。

2　基于该模型的数学分析

2.1　分析参数

在分析参数之前，先做如下假设：

(1)报文缓冲区总是有报文分派的，这个假设是可以实现的。网络处理器成为网络传输的瓶颈，而线路速度大于网络处理器处理速度；

(2)忽略报文从报文缓冲区向 PE 单元的传输速度。这个可以通过在网络处理器内部设置一定的缓冲区来实现；

(3)忽略 PE 单元内部 cache 指令失效引起的额外开销。当指令 cache 失效时，用线程切换来隐藏其带来的额外开销。另外，由于 PE 线程程序具有很强的局部性，cache 的失效率是很低的，因此几乎可以忽略不计；

(4)忽略对同一报文流的定序问题。在一般情况下，只要增加适当硬件，定序问题对于上述报文处理没有影响。

分析参数确定如下：

(1)Tp：PE 的处理时间。$Tp = Tp1 + Tp2 + ... + Tps + 1$。

(2)Tf：协处理器处理时间。$\dot{Tf} = Tf1 + Tf2 + ... + Tfs$。

(3)Tw:PE 在等待协处理器的排队时间。$Tw=Tw1+Tw2+\ldots+Tws$。

(4)n：PE 的数目。

(5)m：PE 内线程的数目。

(6)L：协处理器内请求队列的长度。

要想对实际的网络处理器做精确的数学分析相当困难，本文仅对下述理想情况作分析：$Tp = Tp1 + Tp2$，并且 $Tp2=0$。即 PE 接收 1 个报文请求后，对该报文进行 $TP1$ 的处理，然后向协处理器发出请求，同时进行线程切换。当协处理器对该报文进行 Tf 的处理，将处理结果返回给 PE，此时，认为该报文处理已经结束，同时释放该线程。

基于对这个过程的分析，可得出以下结论：

(1)协处理器是网络处理器的关键资源，所有的报文处理必须经过协处理器。因此，网络处理器的吞吐率最大为 $1/Tf$

(2)当 $Tf > Tp$，系统的瓶颈在 PF 阶段，没有必要设置多个 PE，一个 PE 就能使协处理器工作在饱和状态。

(3)当 $Tf < Tp$，系统的瓶颈在 PE 阶段，可以设置多个 PE 来隐藏报文处理的实际延迟，使协处理器达到饱和状态。使系统的吞吐率达到最大为 $1/Tf$。

当协处理器的吞吐率达到 $1/Tf$ 时，协处理器工作在饱和工作状态；当 PE 的吞吐率达到 $1/Tp$ 时，PE 工作在饱和工作状态。但仅当两者同时工作在饱和工作状态时称系统工作在饱和工作状态，否则称其在未饱和工作状态。以下分析都基于 $Tf<Tp$ 的情况下。

2.2　关于 m，n，L 的分析

现假设系统有 n 个 PE，每个 PE 有 m 个线程，且假设系统工作在饱和状态(这种假设是有道理的，因为，本文分析的目的就是为了让系统工作在饱和状态下)。从单个 PE 来看：一个报文进入 PE 后经过 Tp 的处理，将该报文送往 FP 队列，并向 FP 发出请求，同时切换线程，经过排队等待和协处理器的处理后($Tf + Tw$)返回处理结果，同时释放该线程。其工作状况如图 3 所示。

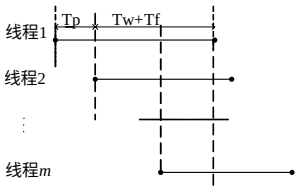


图 3　单个 PE 的处理时间

在理想情况下，协处理器处理时间和等待时间完全被隐藏。则 PE 每经过 Tp 则送出 1 个结果，即单个 PE 的吞吐率为 $1/Tp$，因此，系统总的吞吐率为

$$Throughtout=n/Tp \tag{1}$$

又因为协处理器也工作在饱和状态，故有

$$Throughtout=n/Tp=1/T \Rightarrow n=\lceil Tp/Tf \rceil+1 \tag{2}$$

从图 3 可以看出，线程个数 m 与协处理器处理时间 Tf、PE 的处理时间 Tp、PE 在等待协处理器的排队时间 Tw 之间的关系

$$m=\left\lceil \frac{Tw+Tf}{Tp} \right\rceil+2 \tag{3}$$

另外，可以很明显地看出 $Tw=(L-1)*Tf$，将其代入式(4)

$$\Rightarrow m=\left\lceil \frac{L*Tf}{Tp} \right\rceil+2 \tag{4}$$

假设在某一时刻，所有的 PE 向协处理器发出 n 个请求，协处理器处理完这些请求需要 $n*Tf$ 的时间，在这段时间内，则需要由 PE 再次发出请求，否则协处理器就会出现空闲的状态。而 1 个 PE 发出最短请求的时间为 Tp。故有

$$Tp<n*Tf \Rightarrow n>Tp/Tf \tag{5}$$

以下分 2 种情况讨论：

(1)当 m=1 时，即如果 PE 只有 1 个线程，并且报文调度是基于一种公平策略，那么第 1 个发出协处理器请求的 PE 必须在$(n-1)*Tf$ 时间内再一次发出请求，才能保证协处理器不空闲。即有

$$Tp<(n-1)*Tf \Rightarrow n>Tp/Tf+1 \tag{6}$$

(2)当 m>2 时，即当 PE 内的线程数目多于 1 个时，当 1 个 PE 接收到 1 个返回结果后释放一个线程资源，接着又启动 1 个线程处理报文。而结果的平均返回时间 $L*Tf$，如果 PE 内部的线程调度也是公平的，那么应该有 m－1 个线程优于这个进程发出请求，协处理器在这段时间内处理了 m*n 请求，故有

$$Tp+L*Tf=m*n*Tf \Rightarrow L=m*n-Tp/Tf \tag{7}$$

$$\Rightarrow m=\frac{\left(L+\left\lceil \frac{Tp}{Tf} \right\rceil+1\right)}{n} \tag{8}$$

如果考虑理想情况下 Tp/Tf 与 n 近似，则式(8)与式(4)基本相同。

综上讨论，系统的工作饱和点、n、队列长度 L 和 Tp/Tf 的关系如表 1 所示。

表 1　工作饱和点和 n、L 的关系

Tp/Tf	2	4	6	8
n (m=1)	4	6	8	10
n (m=2)	3	5	7	9
L (m=1)	2	2	2	2
L (m=2)	4	6	8	10

2.3 关于 PE、FP 利用率的分析

(1)当协处理器工作在饱和状态时，吞吐率达到最大 $1/Tf$ 。当协处理器利用率为 1，即 Epf =1，经过时间 T，共输出 T/Tf 个报文，而每个 PE 完成 $T/(Tf*n)$ 个报文，因为每个 PE 处理 1 个报文需要 Tp 时间，故有 $(n*Tf)/Tp$

$$Epe = Tp/(Tf*n) \tag{9}$$

可知，增加 PE 的个数会导致其利用率的下降。

(2)当协处理器处于未饱和工作状态时，PE 的多线程机制将使其工作在饱和状态，此时，Epe =1，单个 PE 的吞吐率为 $1/Tp$，在时间 T 内，吞吐量为 T/Tp，协处理器的吞吐量为 $(T*n)/Tp$，其消耗的协处理器时间为 $(T*n*Tf)/Tp$，故其利用率为

$$Epf = (n*Tf)/Tp \tag{10}$$

可知，此时增加 PE 的个数，可以导致协处理器的利用率的提高。

综上讨论，当取 Tp/Tf=4，m=2 时，系统吞吐量、协处理器利用率、PE 利用率与 n 关系如图 4～图 6。

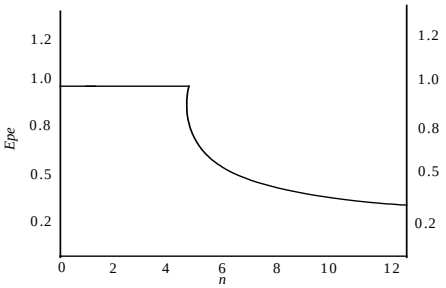


图 4　*Epe* 随 *n* 的变化关系

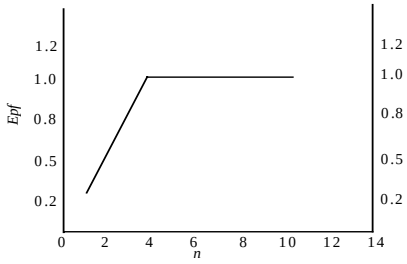


图 5　*Epf* 随 *n* 的变化的关系

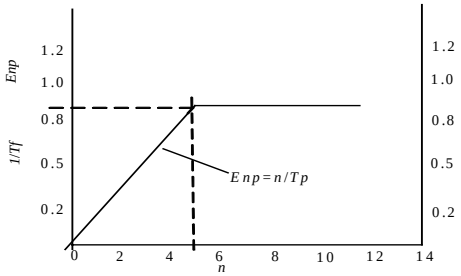


图 6　系统吞吐率随 *n* 的变化关系

由图 4～图 6 可知，当作上述配置时，系统吞吐率和协处理器利用率随 n 增加到一定程度后(n=4 或者 n=5)均不再变化。PE 的利用率当 n 达到 5 时开始降低。因此，在上述配置情况下，当 n=5 时，系统工作在饱和状态。

3　结束语

本文提取了不同网络处理器报文处理的共同特征，抽象了网络处理器报文处理结构，在此基础上，分析了网络处理器性能和 PE 数目以及线程数目的关系。对在相同的资源条件下，提高网络处理器的性能有一定的指导意义。

参考文献

1 张宏科, 张思东, 刘文红. 路由器原理与技术[M]. 北京: 国防工业出版社, 2005.

2 白建军. Internet 路由结构分析[M]. 北京: 人民邮电出版社, 2002.

3 郑维民, 汤志忠. 计算机系统结构[M]. 2 版. 北京: 清华大学出版社, 1998.

4 刘云勇. 基于 IPv6 的下一代互联网[M]. 北京: 电子工业出版社, 2004.

5 石晶林, 程　胜, 孙江明. 网络处理器原来、设计与应用[M]. 北京:清华大学出版社, 2003.

6 Douglas E. Comer 网络处理器与网络系统设计[M]. 北京: 机械工业出版社, 2004.

7 李海龙. 网络处理器分组转换引擎PTE的研究与设计[D]. 西安:西北工业大学, 2003.

(上接第 96 页)

(2)最小合并文档数(min merge document number)。该参数同样对索引时间有重要影响，即当内存中的文档数达到某个值时，Lucene 才进行写磁盘的操作，当内存空间允许时，该参数值越大，建立索引所需要的时间也就越少。

为验证这两个参数在索引建立中时所起的作用，本文进行了如下试验：

对 10 000 个文件建立索引，文件总大小为 1GB，通过调整这两个参数来观察索引建立所用的时间。由表 2 可知，在机器内存允许的情况下，这两个参数的值越大，索引建立所需时间就越少。

表 2　两个参数对索引时间的影响

文档数	文档总大小/GB	合并因子	最小合并文档数	所需时间/s
10 000	1	10	10	1 560
10 000	1	100	10	1 296
10 000	1	100	100	1 228
10 000	1	1 000	1 000	1 196

7　总结

作为一个开源的全文信息检索包，Lucene 正在为越来越多的应用程序提供搜索功能。Lucene 也是目前常用的基于 Java 的全文信息检索包，随着搜索技术正在成为计算机行业研究的热点领域，Lucene 将得到更加广泛的应用。

参考文献

1 Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval[M]. 北京: 机械工业出版社, 2004.

2 Hatcher E, Gospodnetic O. Lucene in Action[M]. Greenwich: Manning Press, 2004.

3 Singhal A. Modern Information Retrieval: A Brief Overview[EB/OL]. (2001-11). http://www.cs.uiuc.edu/class/fao5/cs511/Spring05/lectures/ieee-2001, pdf.

4 Apache Software Foundation. Lucene Query Syntax[EB/OL]. (2005-06). http://lucene.apache.org/java/docs/.

5 张　岭. 智能信息检索中的 Web 挖掘研究[D]. 上海: 上海交通大学, 2002.