基于多核优化的网络协议解析类系统应用研究

李长荣 吴 迪
(齐齐哈尔大学计算机与控制工程学院 齐齐哈尔 161006)

摘 要 针对网络流量监测系统传输数据速度过快时存在的数据丢包、传输停止、响应错误等性能问题,提出了一套针对网络流量监测系统的评价指标,其将系统的吞吐量作为核心指标,通过评价指标来对系统的优化性能进行评估;选取了网络协议解析类系统进行多核优化研究,以GTP-AS系统作为具体目标进行优化之后,根据系统的性能瓶颈提出了一套多核平台优化策略,并且通过实验证明,当核心处理器的计算核心数量增加到7个时,多核优化的网络协议解析系统的吞吐量能够达到优化之前的391.73%,有效提高了系统性能。

关键词 多核处理器,网络流量监测系统,多核优化,吞吐量,网络协议
中图法分类号 TP302 **文献标识码** A

Research on Application of Network Protocol Parsing Class System Based on Multi-core Optimization

LI Chang-rong WU Di
(School of Computer and Control Engineering, Qiqihar University, Qiqihar 161006, China)

Abstract When data transmission speed of the network traffic monitoring system is too fast, the data packet loss, transmission stops, response error will occur. To resolve the problem, this paper proposed evaluation index for the system which uses the system throughput as core index to evaluate the system optimization performance throught the index; selects the network protocol analysis system for multi-core optimization research, taking GTP-AS system as a specific objective optimization, according to the system performance bottleneck, puts forward a set of multi-core platform optimization strategy. The experiment proves when number of the core processor computing core is increased to 7, a number of kernel optimization of network protocol analysis system throughput can reach 391.73% of the optimized priorty, effectively improves the performance of the system.

Keywords Multi-core processors, Network traffic monitoring system, Multicore optimization, Throughput, Network protocol

随着我国互联网技术的飞速发展,网络带宽速率已经能够达到50G左右,这使得人们对网络流量的探索研究不断深入,期望能够合理适当地利用互联网资源。在当今学术界,对网络流量相关领域开展研究工作的人越来越多[1]。

网络带宽不断提高、新兴业务大量涌现,很多情况下由于占用了大量带宽资源,影响着其他网络用户的正常使用,如何能够利用网络资源将网络业务合理区分,保证网络能够为人们提供正常稳定的服务,正是需要通过网络流量监测技术加以解决的。同时,多核处理器平台技术也得到了全面发展和提高,为处理应用程序提供了强大的计算能力,还能够提高网络流量监测系统的性能。因此,网络流量监测技术与多核处理器平台技术的良好结合,将会拥有很高的实用价值。

本文研究的内容主要是利用多核处理器平台,使得网络流量监测软件系统的性能从各个方面得到优化,从而达到提升系统整体性能的最终目标。多核处理器具有强大的计算能力,其技术发展也在不断深入,各个相关领域都希望能够有效利用多核处理器的计算能力,来最大限度地优化软件系统的多种性能,同时节约硬件造成的成本浪费。本文将多核处理器技术与网络流量监测技术结合应用之后,提出了一套多核平台优化策略,从而提高了系统的吞吐量。

1 网络流量监测系统多核优化评价指标的构建

网络流量监测系统通过多核平台进行优化,需要综合完整的评价指标,以对系统优化效果和优化方法起指导性作用,而综合指标应该尽量将多核优化涉及到的方面全部涵盖[2,3]。

1.1 核心评价指标

吞吐量是网络流量监控软件系统最重要的核心指标,系统面临着处理网络中的海量数据信息,而系统数据处理的最高速度是最能反映性能标准的。因此,本文将网络流量监测系统吞吐量性能作为系统优化的核心评价指标。

系统吞吐量(T_r)在系统中的性能指的是单位时间以内系统能够处理的所有数据包的计数之和。很多原因都能够对系统的吞吐量性能造成一定影响,例如网络流量的特性、网络

到稿日期:2013-01-15 返修日期:2013-04-21 本文受黑龙江省自然科学基金项目(F201204)资助。
李长荣(1972—),女,硕士,副教授,主要研究方向为计算机应用,E-mail:lcrlj@126.com;吴 迪(1980—),男,硕士,讲师,主要研究方向为人工智能。

硬件的性能以及软件系统的应用效率等等。本文主要是对网络流量监测系统的吞吐量性能进行深入研究。

1.2 辅助评价指标

辅助评价指标主要分为3类，对系统在多核优化过程中的不同方面进行分析和评价。第一类指标是关键线程时间开销和多核平均锁阻塞比，这2个指标都是用来对系统的开销分配进行评估；第二类指标是多核Cache失中开销率和多核流量分摊度，这2个指标主要用来对系统的调度方式进行评估；第三类是多核优化加速比指标，用来对系统优化效果进行评估，本文主要对第三类指标，即多核优化加速比进行介绍。

1.3 多核优化加速比的构建

由于计算资源的不断增加，使得软件系统的理想优化效果得到提升。当多核优化计算过程中有 N 个处理器核心时，软件系统的性能应该提升到单核处理器进行计算时的 N 倍。但是，需要建立一个结合了现实因素的多核优化评价指标进行实验研究，这是因为理想的优化效果在实际优化计算过程中是很难实现的，而指标的价值是需要能够得到系统进行多核优化的参考值，并且从估算性能和测量两个方面改善优化效果。

(1)系统理论加速比

多核优化加速比是在系统理论加速之上得到的，系统理论加速比是用来描述软件系统从串行转换到并行的系统优化程度，其定义式为：

$$S_{(n)}=\text{系统理论加速比}=\frac{\text{系统串行执行时间}}{\text{系统并行执行时间}}$$

上式表示的是系统从串行转换到并行的优化程度，即系统串行执行时间与系统并行执行时间的比值。

(2)多核优化加速比的计算过程

在系统进行并行优化的过程中，会引入包括原子操作、同步锁操作等不可避免的新开销，这些新的开销都会对系统理论加速比造成一定影响，而阿姆达尔法则提出的优化理想效果是几乎不存在的。由于系统并行优化引入新的开销问题，本文在阿姆达尔法则的基础之上提出了新的多核优化加速比的评价指标，并给出了其计算方法。多核优化加速比实质上是在系统理论加速比的基础上综合考虑了并行优化带来的开销后得到的系统理论加速比，多核优化加速比以 S_r 来表示。

通常情况下，系统并行优化引入的开销比例是固定不变的，本文以 f_r 表示系统并行优化带来的开销在系统并行计算时间 $T_{parallel}$ 中所占的比例。

$$T_{parallel}=(1-f)\cdot\frac{T_{total}(1)}{n} \tag{1}$$

多核优化加速比的推导过程如下：

$$T_{total}(n)=f\cdot T_{total}(1)+T_{parallel}+f_r\cdot T_{parallel} \tag{2}$$

$$T_{total}(n)=f\cdot T_{total}(1)+(1-f)\cdot\frac{T_{total}(1)}{n}+f_r\cdot (1-f)\cdot\frac{T_{total}(1)}{n} \tag{3}$$

$$\frac{T_{total}(1)}{T_{total}(n)}=\frac{n}{nf+(f_r+1)\cdot(1-f)} \tag{4}$$

由此得到：

$$S_r=\frac{1}{f+\frac{(1-f)\cdot(1+f_r)}{n}} \tag{5}$$

通过式(5)计算后可以得到多核优化加速比比值 S_r。

2 网络流量监测系统多核优化步骤

网络流量监测系统多核优化步骤逻辑如图1所示。

通过对待优化系统核心指标的全面测评，掌握系统的基本性能；对辅助评价指标进行分析之后得到系统可以优化的部分；结合网络流量的特点，以及多核优化的特性，设计一套综合多核优化策略；对完成多核优化之后的系统进行核心指标测评，掌握系统优化后的整体性能；通过辅助指标将优化后的系统在各个性能方面的提升和改进与优化前的进行对比分析；对多核优化的特点进行总结。

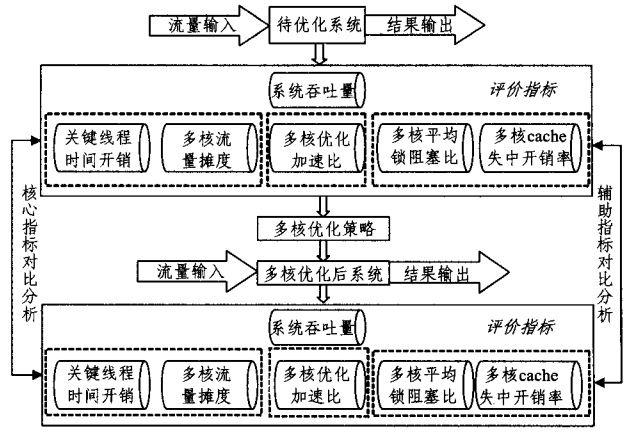

图1 网络流量监测系统多核优化步骤逻辑图

3 网络协议解析类系统的多核性能优化研究

3.1 研究目标

网络流量监测系统中代表性较强的就是网络协议解析类系统，本文选取网络协议解析类系统作为研究对象，并通过多核优化评价指标对网络协议解析类系统的多核平台性能优化进行研究。

本文选取自主研发的典型网络协议解析系统，即GTP-AS作为优化实例，GTP-AS通过对互联网中的GTP(GPRS隧道协议)流量进行分析，得到用户对互联网访问时的上线和下线信息，从而为在互联网中监测其他业务流量提供准确的用户信息。在多核平台上对GTP-AS进行系统性能优化，能够探索网络协议解析系统在多核优化过程中的性能特点。

3.2 优化过程

(1)通过对GTP-AS进行测评性能分析，得出GTP-AS经过单核处理器平台上的信息吞吐量性能，分析出系统存在的瓶颈问题。

(2)将GTP-AS置于多核处理器平台中进行性能测评分析，分析GTP-AS在经过多个处理器同时计算后，性能是否得到提高。

(3)根据GTP-AS在多核处理器平台的瓶颈问题，与多核优化理论相结合，提出新的多核平台优化策略，GTP-AS经过优化之后记作MGTP-AS。

(4)对优化后得到的MGTP-AS进行性能测评分析，并计算MGTP-AS与GTP-AS相比性能提升的比例。

4 GTP-AS 系统性能评测实验分析

4.1 实验说明

本文在单核处理器平台中的实验研究是通过利用多核处理器平台中的一个核心处理器来完成的；实验平台共有 8 个核心处理器(核心处理器 0-核心处理器 7)，始终以核心处理器 0 作为网络流量的输入，剩下的 7 个核心处理器进行多核性能优化测试。实验使用的网络流量是在中国移动市级网络链路中获取的 GTP 业务流量，如表 1 所列。

表 1 网络流量说明

	数据包数量	数据总长度(bytes)	报文长度(bytes)
Trace	7884399	1151923568	146.10

4.2 GTP-AS 在单核处理器平台上的性能分析

GTP-AS 在单核处理器平台上的吞吐量性能分析如表 2 所列。

表 2 GTP-AS 单核处理器平台吞吐量

处理器类型	吞吐量(mbps)	报文速率(pps)
单核处理器	102.77	92281

4.3 GTP-AS 在多核处理器平台上的性能分析

GTP-AS 在多核处理器平台上进行性能测试，随着处理器核数的增长得到的吞吐量性能如表 3 所列。

表 3 GTP-AS 多核处理器平台吞吐量

处理器核数	吞吐量(mpbs)	报文速率(pps)	与单核处理器比较
2	129.18	115895	125.70%
3	119.88	107546	116.62%
4	118.66	106450	115.46%
5	115.05	103213	111.95%
6	111.32	99869	108.32%
7	111.32	101484	110.27%

通过对表 3 进行观察能够发现，当 GTP-AS 在 2 核处理器运行时，能够得到最好的性能提高，与单核处理器相比性能提高了 25%，但是随着处理器核数不断增加，吞吐量性能提高逐渐趋于缓慢，由此能够说明当网络协议解析系统运行在逐渐增加核数的多核处理器平台上时，并不能有效提高网络协议解析系统的吞吐量性能。

由图 2 可以看出，随着计算核数的不断增加，GTP-AS 的吞吐量性能提升比例逐渐下降。

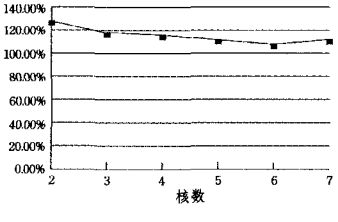

图 2 GTP-AS 核数增加的吞吐量提升比例变化

5 GTP-AS 的多核平台优化策略

对上述实验结果进行分析后得知，将 GTP-AS 从单核处理器平台转到多核处理器平台进行性能测试之后，仅仅增加计算机核心的数量，其性能并没有得到理想效果的提升，反而出现了性能下降的现象。本小节根据实验结果分析出的 GTP-AS 性能瓶颈，提出了一套新的多核优化策略，最后完成了实验验证。

5.1 多路并行的执行方式

GTP-AS 是由于线程 F1 负载过重而出现瓶颈问题的，但是已经无法重新对 F1 线程进行拆分以执行并行，所以，对 GTP-AS 系统整体线程进行多路并行执行是能够减轻系统瓶颈问题的最好方法，图 3 为多路并行执行方式。

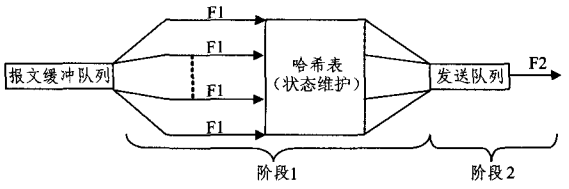

图 3 多路并行执行方式

图 3 中，不能将 GTP-AS 系统的 F1 线程进行进一步拆分，只能通过多路并行执行的方式分摊系统瓶颈问题。当有网络报文数据输入的时候，多个 F1 线程能够在阶段 1 中对其进行并行处理，计算得出 GTP-AS 系统中最耗时的功能模块，从而在增加系统的吞吐量的同时，提高了系统数据处理的能力。

5.2 协议消息类型的报文分发

当使用多路并行执行模式进行处理之后，系统前端的单个接收队列出现了性能瓶颈。多路并行的处理线程 F1 从队列中获取消息进行处理时，必然需要实现同步。同步带来的竞争会严重削减并行处理性能增益[6,7]。

当系统采用多路线程并行执行方式之后，系统的瓶颈所在转变成为前段单队列报文接收。F1 线程需要同步地从单队列中获取报文进行处理，而同步造成的竞争等待情况会降低本来并行处理能够实现的性能提高。由此，本文的报文分发方式选择的是基于协议的消息类型，从而有效减少同步竞争带来的性能下降，当对单一报文接收对列进行优化时需要考虑到 GTP 协议的特征，本文按照消息类型将输入的 GTP 报文分别发送到 6 个缓冲队列中，将原有的单一报文接收队列优化改造为基于消息类型的 6 个队列，由此能够大幅度减少多路 F1 现场同时从队列中获取报文产生的竞争等待时间。

5.3 报文消息处理的优化策略

在协议消息的基础上进行报文分发之后，生成的新问题是 GTP 报文发生顺序混乱的处理，而这是协议解析类系统完成并行优化之后的共性问题。

由此，本文在消息处理方式上采取了组间并行、组内优先的原则，当报文分配到 6 个缓冲队列之后，将 6 个队列划分成为 3 个小组，分别是 Create、Update 和 Delete。F1 能够对每组之间进行并行处理，对于每个小组中的 2 个队列(Request，Response)进行优先级设置，即 Request 的优先级要在 Response 的优先级之上。在接到报文消息之后，根据类型将其分配到各自的队列，如果 F1 可以在 3 组之间获得报文消息且完成并行处理，则需要根据预设的优先级在每个小组内处理报文消息，高优先级队列中如果存在没有处理结束的报文，必须对其进行处理。组间并行、组内优先的消息处理方法实质上是将并行处理与串行处理结合折中所得，虽然不能够达到完全并行处理的速度，但是能够有效避免报文数据发生乱序

处理的现象。图 4 为报文消息处理优化策略示意图。

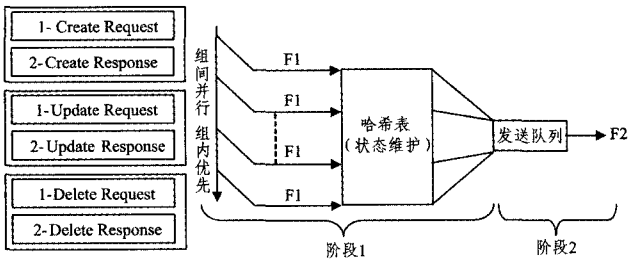

图 4　报文消息处理优化策略示意图

5.4　线程固定调度策略

在应用层空间，通过核心处理器的亲和力对软件线程和硬件核心进行调配，是最为关键和有效的方法。核心处理器的亲和性指的是线程在核心处理器上能够尽可能地长时间运行，且不会被其他核心处理器迁移。

本文采用固定调度策略对 GTP-AS 系统进行优化，实质上是通过核心处理器的亲和性将系统线程与核心绑定，从而根据系统的特征来完成线程的调配，如图 5 所示。例如：假设阶段 1 的 F1 线程共有 N_{F1} 个，阶段 2 的 F2 线程共有 N_{F2} 个，利用核心处理器的亲和性，将 $N_{F1}+N_{F2}$ 个线程绑定在多核处理器的核心上，使得每个线程都能实现并行处理执行。

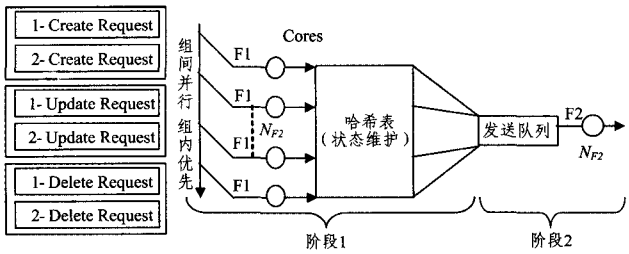

图 5　固定调度策略的多路并行执行方式

6　MGTP-AS 在多核平台上的性能实验分析

6.1　吞吐量性能分析

将 GTP-AS 在多核平台进行优化处理之后得到的系统为 MGTP-AS，下面是对 MGTP-AS 系统吞吐量的性能测试结果。

表 4　MGTP-AS 系统多核处理器平台吞吐量

处理器核数	N_{F1} : N_{F2}	吞吐量 (mbps)	报文速率 (pps)	单核吞吐量	与单核处理器比较
2	1 : 1	137.88	123700	102.77	134.16%
3	2 : 1	209.32	187782		203.68%
4	3 : 1	248.89	223282		242.18%
5	4 : 1	304.43	273111		296.22%
6	5 : 1	347.89	312100		338.51%
7	6 : 1	402.58	361164		391.73%

随着核心处理器计算核心数量的不断增加、N_{F1} : N_{F2} 的比例不断升高，主体线程的计算资源正在以较快的速度增加，因此，MGTP-AS 系统的性能也得到进一步提升，当核心数量上升为 7 个的时候，MGTP-AS 系统的吞吐量是优化之前的 391.73%。

由图 6 可以知道，MGTP-AS 系统经过优化后，随着核心数量的增加，系统吞吐量呈现出逐渐上升的趋势，与图 2 对比后能够得知，本文提出的多核平台优化策略在提升系统性能方面达到了较好的效果。

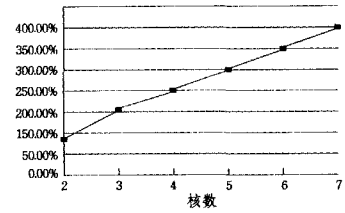

图 6　MGTP-AS 核数增加的吞吐量提升比例变化

6.2　多核优化加速比分析

本文介绍的多核优化加速比是系统的辅助指标之一，并给出了多核优化加速比的定义和计算公式，本小节通过多核优化加速比指标来对网络协议解析系统的优化效果进行评估。

表 5 给出了串行比例 f 的计算结果。

表 5　不同核数时串行比例 f 的计算

核数	串行函数	Gtp System Init / Gtp System Clean	系统发生事件总数	f
2		17,101,765,427	167,664,366,932	0.102
3		17,835,099,258	156,448,239,105	0.114
4		21,617,125,262	196,519,320,570	0.110
5		23,328,812,105	220,083,133,066	0.106
6		23,774,098,109	224,28,3944,422	0.106
7		21,321,205,373	226,821,333,752	0.094

表 6 给出了并行开销占并行计算时间 f_r 的计算结果。

表 6　不同核数时并行开销比例 f_r 的计算

核数	并行开销函数	Put Recv Queue / Get Recv Queue / Hash Item Lock / Hash Item Un Lock	并行计算部分事件总数	f_r
2		30,564,208,106	150,562,601,505	0.203
3		28,970,146,228	138,613,139,847	0.209
4		34,805,536,864	174,902,195,298	0.199
5		40,334,635,797	196,754,320,961	0.205
6		40,703,498,802	200,509,846,313	0.203
7		40,072,525,034	205,500,128,379	0.195

根据多核优化加速比的计算公式，结合表 7 得出了 MGTP-AS 系统的多核优化加速比。

表 7　MGTP-AS 系统多核优化加速比

	f	f_r	S_r	实际值
2	10.20%	20.30%	155.73%	134.16%
3	11.40%	20.90%	212.29%	203.68%
4	11.00%	19.90%	265.41%	242.18%
5	10.60%	20.50%	311.09%	296.22%
6	10.60%	20.30%	350.57%	338.51%
7	9.40%	19.50%	402.14%	391.73%

图 7 给出了多核优化加速比与吞吐量实际优化相比的提升比例。

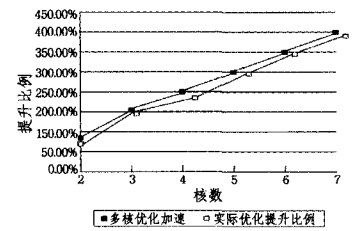

图 7　多核优化加速比与吞吐量实际优化相比的提升比例

(下转第 121 页)

方式的时间性能低于邻接链表方式。

(3)不同算法的时间性能

如图 8 所示，AC_SC 算法的时间性能最优，匹配所需时间大约是 AC-bitmap 算法的 62%，CA-AC 算法的 74%。

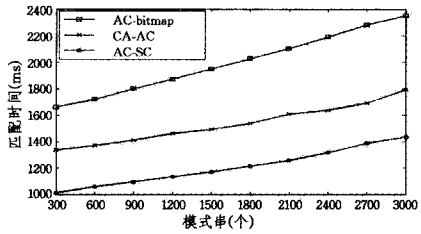

图 8　不同算法的时间性能

结束语　本文提出了一种适合中文的多模式匹配算法。该算法采用邻接链表方式存储有限状态自动机，减少了空间开销，同时将扇出系数较大的状态“0”的单链表转化为散列链表，以便快速查找下一状态，提高了算法的时间性能。最后对算法空间和时间性能进行了测试，测试结果表明，本文提出的邻接链表方式，其时间性能高于状态矩阵存储方式，略低于完全 Hash 表存储方式，而其空间性能远优于完全 Hash 表、状态矩阵和位图方式。AC_SC 算法的时间性能优于 AC-bitmap和 CA-AC 算法，适合大规模中文模式匹配。

参 考 文 献

[1] 杜大军，费敏锐，宋扬，等. 网络控制系统的简要回顾及展望[J]. 仪器仪表学报，2011，3(32)：713-720

[2] Wheeler D L，Barrett T，Benson D A，et al. Database resources of the National Center for Biotechnology Information [J]. Nucleic Acids Research，Database issue，2007，35：5-12

[3] Knuth D E，Morris J H，Pratt V R. Fast Patter Matching in Strings[J]. SIAM Journal of Computer，1977，6(2)：323-350

[4] Boyer R S，Moore J S. A Fast String Searching Algorithm[J]. Communications of the ACM，1977，20(10)：762-772

[5] Horspool R N. Practical fast searching in strings [J]. Software-Practice & Experience，1980，10(6)：501-506

[6] Sunday D M. A very fast substring searching algorithm [J]. Communications of the ACM，1990，33(8)：132-142

[7] Aho A V，Margaret J. Corasick Efficient String Matching[J]. An Aid to Biographic Search Communications of the ACM，1975，18(6)：333-340

[8] Commentz-Walter B. A String Matching Algorithm Fast on the Average [J]. Proceedings of 6th ICALP，1979，71：118-132

[9] Wu Sun，Manber U. Fast algorithm for multi-pattern searching [R]. Tucson：Department of computer science university of Arizona，1994

[10] 王永成，沈州，许一霞. 改进的多模式匹配算法[J]. 计算机研究与发展，2002，39(1)：55-60

[11] Norton M. Optimizing pattern matching for intrusion detection [EB/OL]. http://does. idsreseareh. Org/OptimizingPatterm-Matching-ForIDS. pdf，2006-05-11

[12] Tuck N，Sherwood T，Calder B，et al. Deterministic Memory Efficient String Matching Algorithms for Intrusion Detection[C]// IEEE Iafocom. 2004：333-340

[13] 巫喜红，曾锋. AC 多模式匹配算法研究[J]. 计算机工程，2012，38(6)：279-281

[14] 张元竞，张伟哲. 一种基于位图的多模式匹配算法[J]. 哈尔滨工业大学学报，2010，42(2)：277-280

[15] 王培凤，李莉. 一种改进的多模式匹配算法在 Snort 中的应用[J]. 计算机科学，2012，39(2)：72-74

(上接第 88 页)

本文对 MGTP-AS 系统的串行代码比例 f 和并行处理引入的开销比例 f_r 进行测量之后得到了多核优化加速比 S_r。图 7 说明了在系统进行多核优化时，如果能够得到准确的 f 和 f_r，通过计算得到的 S_r 就可以对实际优化效果和理想效果之间的差距进行评估，从而给出系统优化后性能上升的趋势。

结束语　随着网络新型业务的不断出现、原有业务的更新升级，都要求网络流量监测系统软件能具有良好的性能，同时，多核处理器平台为上层应用软件提供了强大计算能力的支撑，本文将网络流量监测与多核处理器结合之后，对网络流量监测系统在多核处理器平台上的性能优化进行了深入研究。总之，多核处理器平台强大的计算能力不仅可以支持网络流量监控软件，还可以使优化后的系统性能得到提高，有效处理网络中的大量数据处理，在保证网络服务质量、发掘网络资源价值、分析网络关键数据、保障网络稳定运行等方面具有良好的应用前景。

参 考 文 献

[1] 刘热. OpenMP 多核技术研究及其在遗传算法中的应用[J]. 沈阳大学学报，2010(05)：21-35

[2] 吴俊杰，潘晓辉，杨学军. 面向非一致 Cache 的智能多跳提升技术[J]. 计算机学报，2009(10)：31-42

[3] 肖俊华，冯子军，章隆兵. 片上多处理器中延迟和容量权衡的 cache 结构[J]. 计算机研究与发展，2009(01)：19-21

[4] 黄国睿，张平，魏广博. 多核处理器的关键技术及其发展趋势[J]. 计算机工程与设计，2009(10)：43-49

[5] 黄国睿，张平，魏广博. 多核处理器的关键技术及其发展趋势[J]. 计算机工程与设计，2009(10)：67-72

[6] Gummaraju J，Rosenblum M. Stream Programming on General-Purpose Processors[M]. 2005：113-127

[7] McCool M D. Scalable Programming Models for Massively Multicore Processors[M]. 2008：89-101

[8] Bell S，Edwards B，Amann J，et al. Tile64processor：A64-core soc with mesh interconnect[C]// Proc of Int Solid-State Conference. 2008：145-152

[9] Ros A，Acacio M E，Garcia J M. Scalable directory organization for tiled cmp architectures[C]// Proc of Int Conf on Computer Design. 2008：94-99

[10] Guz Z，Keidar I，Kolodny A，et al. Nahalal：Cache organization for chip multiprocessors[C]// IEEE Computer Architecture Letters. 2007：54-68