并发线程动态性能分析

程克非[1] 汪林林[2] 张 聪[3] 张 勤[1]
(重庆大学计算机学院 重庆400044)[1] (重庆邮电学院 重庆400065)[2]
(重庆交通学院计算机系 重庆400074)[3]

摘 要 有效地分析软件应用系统的执行性能对于系统的运行稳定和性能改善有着很重要的作用,特别是一些高性能计算,如大气预测、天体运动、海量数据处理、科学数值计算等。这些应用都不可避免地涉及并发线程,在并发处理过程中,两个并发的线程(线程)之间的同步在常规的性能分析中常常被忽略,从而造成最终结果的很大偏差,同时无法分析系统的动态性能特性。本文引入一种并发线程的动态性能分析方法PERC-PAPI,以此来采集和分析具有同步并发线程特征系统的性能数据,从而获得更为精确的动态性能分析结果。

关键词 高性能计算,并发通信,PERC,PAPI,CPU硬件性能计数器

Dynamic Analysis of Performance of Simultuneous Thread

CHENG Ke-Fei[1] WANG Lin-Lin[2] ZHANG Cong[3] ZHANG Qin[1]
(College of Computer Science,Chongqing University,Chongqing 400044)[1]
(Post and Telecommunications Institute of Chongqing,chongqing 400063)[2]
(Department of Computer,Traffic Institute of Chongqing,Chongqing 400074)[3]

Abstract Effective analysis of high-end application system is very important, such as Terascale data processing, mathematics computing, …, almost all of them will employ concurrent threads to improve the computing features. Technical for analysis last days will put it aside the cost for concurrent and may will lead to large difference from standard value. This paper introduces a technology of PERC-PAPI to sample and analyzes the system performance with synchronization cost to obtain more accuracy system performance data of CPU clock cycles, PEC is proposed by sciDAC Center to analysis High-End computing system, PAPI is developed by ICL Lab of University of Tennessee to provide simple interface to access Hardware Performance Counter.

Keywords High-performance computing, Concurrent communication, PERC, PAPI, CPU hardware performance counter.

1 引言

随着软件规模和复杂性的增大和高端系统高性能计算的需要,以及为了更好地设计和分析大型软件系统,我们需要有一些有效的手段技术和工具来帮助我们对软件系统进行有效的性能分析。本文引入了一种较新的分析模型和技术来对系统的性能,特别是并发系统的性能做出比较精确的分析。一个软件系统的性能主要从它运行所占的CPU时间(或者说时钟周期)和它的I/O性能两个方面来反映,而I/O性能最终也可从CPU时间中反映,因此,对于一个系统,特别是实时系统和高性能计算系统来说,CPU时钟周期的长短和时钟数的多少就可完全反映一个特定环境和特定系统的性能,而在一个特定的计算环境中,CPU时钟周期长短已经确定,这样系统运行所占的时钟周期数就可以反映系统的性能。一种传统的性能分析方法可以采用系统仿真的形式来分析和评价系统性能,但这种方法在实现上来说,有一定的困难,同时对于一个具有并发线程(线程)的系统来说,这种仿真的结果可能很难真实反映系统的实际运行情况。在文[1]中介绍了一种行为描述调度特定实现的平均时钟周期数的技术,文[8]中描述了基于马尔科夫链模型的性能评估方法。

在很多实际的应用系统中,像通讯系统,网络程序,多媒体应用系统,消息中间件系统等都是一些并发系统的例子。对于这些例子,如果采用文[1,3,8]中所描述的技术来评估系统运行时的时钟周期数将导致分析结果的较大偏差,这是因为文[1,3,8]主要是针对单线程(进程)应用系统的分析,而忽略了可能在很大程度上影响整体性能的并发线程间的同步代价[2],在文[9]中介绍了一种利用Petri网模型的多线程系统分析方法,采用这种方法时,在数据节点增大时,整个计算复杂度将以指数级增长,很难应用于需要实时性能分析的应用中。文[2]中则采用PERC技术得到了静态的性能分析数据,但对动态性能行为分析上该文没有给出解决方法,而有时候我们根据系统实际运行时的动态时间分析系统的动态时间性能。

本文引入了PERC-PAPI技术来采集和分析多线程并发系统的动态时间性能。PERC是sciDAC中心针对高端计算而提出的性能分析模型,PAPI(Performance Application Interface)是美国田纳西州立大学ICL实验室开发的基于Perfctr的CHPC(CPU Hardware Performance Counter)应用接口,

程克非 博士生,讲师,主要研究方向为高性能计算应用系统的性能分析研究。汪林林 教授,硕士生导师,主要研究方向为数据库理论,数据挖掘,GIS与空间数据库等。张 勤 博士生导师,主要研究方向为故障诊断,人工智能等。

CHPC是现代CPU中提供的保存CPU工作状态的一组特殊的寄存器，包括Cache命中率，分支预测准确度，指令周期数等。对这些寄存器的读写都是和正常的指令流并行，基本上不影响指令流本身的执行特性。但对于普通应用程序来说，读写这些寄存器在操作系统中是不允许的，要想存取CHPC的话，需要很多底层特权指令操作。PAPI则通过插入系统核心工作，提供一个简单的用户接口访问这些寄存器。在本文中，就采用PAPI技术读取这些寄存器值(主要是时钟周期数)，从而最大限度地减少因为监控运行带来的性能特征影响，这样得到的性能数据将更为准确，最终的分析结果也就更为精确有效。同时因为系统中采用PAPI来得到系统的底层性能数据，这样在得到时间性能的同时，也可以很容易得到应用系统的其他性能特征曲线(如MFlops)，这对要求除时间性能外的系统行为分析将有一定的帮助。

在第2节中，我们将详细讨论线程间通信时延对多线程并发系统整体性能的重要影响，并给出本文所要用到的分析实例及其线程间的同步时序图；第3和第4节讨论通过系统的同步图识别通信层；第5节将简单介绍PAPI如何应用于系统中采集性能数据；第6节根据采集到的数据分析系统性能，并作出比较；最后为结论。

2 通信时延对并发系统性能的影响

在叙述线程间通信时延之前，我们首先引入本文后续内容所要用到的一个实例，如图1所示，这是作者参与设计和开发的一个GPS车辆管理项目中用到的一个消息中继服务器(或者说这是一个定制的消息中间件)的发送部分的同步时序图(STG)。如果某个消息帧发送失败，将尝试重发预定的次数，如果重发次数为0，那么发送失败，放弃该帧消息的发送，进入下一帧消息发送。线程1控制发送过程，并调度重发管理，线程2负责消息帧发送处理，线程3调度帧间发送，在每一帧之间插入等待，保证消息发送有足够时间完成，同时负责启动线程1。这三个线程间必然存在同步通信问题。

在图中的三个线程间的通信可以采用很多种方式实现，如共享内存区(含寄存器交换)，管道，TCP/UDP传输等各种协议，在这个实现中，采用了较为简单有效的共享内存区的方式来实现。相互关联着的并发实体线程之间必须按照一定的信号量规则才可向前推进。这种信号量在整个系统中是属于全局性的，保存在共享内存区中。这样线程又可以从信号量的角度划分为生产者和消费者两类，一个线程在某一个时刻可能是生产者，在下一个时刻可能是消费者，但在同一个时刻，只能是其中之一。同时信号量又根据其作用的不同，分为同步信号量和数据信号量两类。同步信号量用于不同线程间的同步，数据信号量则用来在线程间传递消息。在图1中的Send_Ready，Frame_Wait，Send_Start，Send_End属于同步信号量，而Send_Fault则属于数据信号量。

现在我们来看在这种具有多线程并发控制的情况下的系统性能分析。首先，根据系统实现的源代码，可以计算出一个期望的理论性能值，见表1。第二步，我们采用对单个线程分别分析的技术，在不考虑同步通信代价情况下(也就是假设当前的发送数据流是连续的，可以通过简单的写发送缓冲区来模拟)，通过PERC[10]静态分析，分别得到三个并发线程所用的时钟周期数分别为6500，2800，4100，见表1，系统的整体性能为6500时钟数。它和期望的比率为−11.43%，这个值很显然有较大的误差。

另一方面，如果我们在做全局系统性能评估时，同时考虑线程间的通信代价，结果应该就能比较精确地反映系统实际运行特性。在表1中第三行给出了采用PERC-PAPI技术得到的系统稳定运行后的某时刻的性能数据，现在三个并发的线程的时钟周期数分别为7665，3853，5142，系统整体性能则为7665，和期望值的误差比率为4.79%，明显小于刚才的误差，实际上，因为这个时间参数也包含了系统运行时动态特性，因此比理论值更能反应系统动态表现。

从上面的分析比较中，可以看出在多线程并发系统中通信时延代价对整个系统性能分析的重要性，是不能随意忽略的。在第3、4节中，我们将介绍如何利用同步通信层来快速准确地决策数据采集点。

3 系统同步通信层定义

在图1中，我们用SC:(消费者)或者SP:(生产者)开头表示一个信号量为同步信号量，否则为数据信号量。一个循环l_i如果含有同步信号量且该信号量不属于该循环的嵌套子循环，则我们称l_i为同步循环，而一个线程本身可以多次调用，所以一个线程本身我们也认为其为一个循环。在图1中的并发管理循环因为含有SC:$frame\_waiting$，因此它就是一个同步循环。

接下来我们将引入系统同步图来抽象化表示系统的线程间的同步关系。一个同步图是由带权重的边集E和顶点集V构成的有向图。其中顶点集V含循环的起点和终点，同步事件SC:和SP:；对于边集E，有两种类型，一类为在图1中的同一线程内相关顶点间的自然顺序连接关系，我们称其为线程边，边的权重则决定于两个相连顶点间执行所需的时钟周期数，另一类为线程间的同步连接，我们称其为同步边。如果存在顶点v_1与v_2分别属于两个不同的线程，其中v_1为某个信号量的生产者SP，v_2为该信号量的消费者，或者反之，则v_1与v_2存在一条同步边，方向为v_1到v_2(v_1为SC)或者v_2到v_1(v_2为SC)，其权重为CPU读取寄存器值所需的周期数，但在实际运行环境的实验结果中，该值要大于这个数量(见第6节实验结果分析)。另外因为数据信号量对线程间同步没有影响，因此在同步图中忽略。按照上面的定义和第5节实验所得到的性能数据，我们得到分析实例的系统线程同步图，见图2。

图中顶点以$V_{x,y}$标记，x表示属于那个线程，y为线程内节点编号。现在我们可以得到系统的所有同步循环集合L，同时每一个同步环都和这样一个四维组$\{N(l_i), P(l_i), C(l_i), T(l_i)\}$相关联。$N(l_i)$表示嵌套在$l_i$中的所有子循环的集合，$P(l_i)$表示所有拥有消费由$l_i$中生产信号量的同步循环集合，$C(l_i)$表示所有生产$l_i$中消费信号量的同步循环集合，而$T(l_i)$则表示包含$l_i$循环的所有线程集合。

为了能够将线程同步代价在系统性能中表现出来，在文[2]中引入了通讯层的概念，这里采用用类似的方法用于同步结构分析，以便于快速准确决策数据采集点。

循环l_i和l_j存在生产消费关系，如果$l_i \in (P(l_j) \cup C(l_j))$。如果在一个给定的时刻(CPU运行周期)内，$l_i$处于激活状态，则称$l_i$激活；如果存在两个循环$l_i$，$l_j$在这个时刻内都处于激活状态，则称$l_i$，$l_j$并发执行。

同步通讯层CL是这样一些同步循环的最大集合，这些循环都相互关联，并且满足以下两个条件：

1. 对于所有$\{l_i, l_j | l_i$与l_j存在直接的信号量生产消费关系，$i \in [1,n], j \in [1,n], i \neq j, n$为系统中的循环总数$\}$；

2. 对于 CL 中的循环 $l_i \in L, l_j \in L, l_i, l_j$ 并发执行。

在图1中，考察 $CL_1 = \{l_2, l_3, l_4\}$，每一个循环都和其他循环存在生产消费关系满足条件1，同时所有循环都并发执行，满足条件2，因此 CL_1 是通讯层。

在下一节我们将简单介绍如何从系统同步图中识别通讯层。

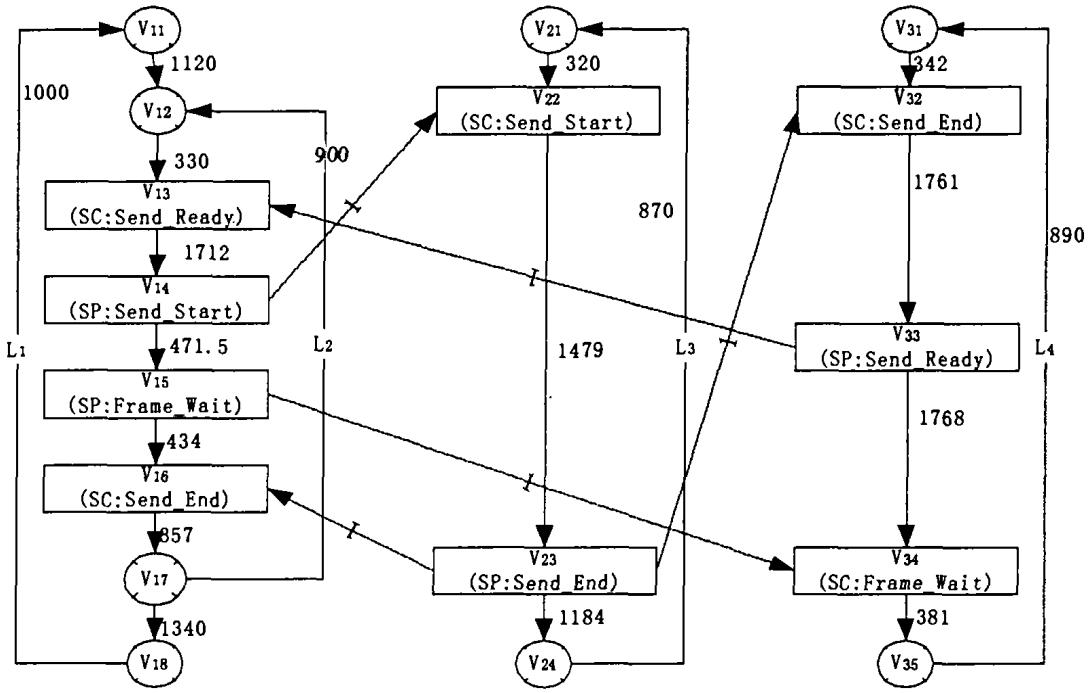

图1 消息发送同步图

4 通信层的划分和识别

从上面的分析我们可以知道要识别一个同步通信层 CL_i，首先要给出并发运行的同步循环，这个可以从系统的设计和系统的特征得到。在下面的论述中，我们都假定各循环间的同步信息已经存在，并且系统中的循环集合表示为 L。结合上一节通信层的定义，就得到下面识别系统中同步通讯层 CL 的伪代码算法。

算法1 IdentifyCL

1. 令 L 为系统中一个满足各循环相互关联的最大集合；

2. 在 L 中搜索并找出所有满足条件1的循环的最大集合 M(可直接从系统设计和实现中得到)；

3. 对于每一个 M，如果其中任意两个循环 l_i, l_j 是并发执行的，则该集合就是一个通讯层 CL_i，算法结束。否则继续执行第3步；

4. 从 M 中去除 l_i, l_j，同时令 $M_1 = M \cup \{l_i\}$，$M_2 = M \cup \{l_j\}$，然后对 M_1 和 M_2，分别返回第2，3，4步递归执行，直到所有递归返回。

为了便于分析和识别处于系统不同位置的通信层 CL_i，我们还要对每一个 CL_i 进行位置编号，其算法如下。

算法2 LocateCL

1. 令 $level = 1$，L 为系统中所有同步循环集合，$Layer[level] = \{\}$；

2. 如果 L 为空，则结束；否则，对于每一个已得到的通信层 $CL_i = IdentifyCL$ 中不存在循环 l_i，使得 l_i 中嵌套有 L 中的任一个循环 $l_{j,(j \neq i)}$，那么 $Layer[level] = Layer[level] \cup \{CL_i\}$；

3. 从 l 中去掉所有在 $Layer[level]$ 中的循环，$level = level + 1$，然后递归执行第2步，直到所有递归返回。

通过调用 LocateCL 算法，就可以得到最终性能数据采集和分析所需要的系统特征点。在图2中，有同步循环集 $L = \{l_1, l_2, l_3, l_4\}$，通信层 $CL_1 = \{l_2, l_3, l_4\}$，$CL_2 = \{l_1\}$，则可以得到 $Layer[1] = \{CL_1\}$，$Layer[2] = \{CL_2\}$。在分析的时候，首先分析第一层，然后再向外扩展，直到最外层为止，系统的整体性能由最外层的性能给出。同时因为在分析 $Layer[i], i > 1$ 的时候，所有 $Layer[j]$，$j <= i$ 的层就作为一个整体计算在 $Layer[i]$ 中。这样就自然地将并发线程间的同步代价计算在内，同时也就逐层简化，最后得出结果。在实际的分析过程中，根据 LocateCL 和 IdentifyCL 得到同步通讯层，然后利用下一节中讲到的 PAPI 性能数据采集技术，得到分析数据，然后逐层向外分析计算，当到达最外层时就得到了系统当前运行态的动态时间性能。

5 性能数据的采集

在上一节的讨论中，我们得到了系统中的关键性能采集位置，同步通信层。这一节中，将讨论如何利用 PAPI 在这些位置可靠地采集性能数据。在文[2]中讨论了 PERC 用于静态系统性能分析，因此其数据是通过一个专用的调度系统得到的，而对于动态运行时的性能分析，文[2]中的技术就很难实现。本文中将动态收集系统性能数据，文[4]中介绍了在并行系统中性能分析数据的采集如果以传统方式进行的话，可能改变并行系统的计算特征，因为传统技术的仿真或者监控可能会改变并行进程间的时间同步特征，对于并发的线程也一样。因此在这里我们采用了另外一种技术，PAPI 采集性能数据。这种技术是通过采集现代 CPU 中提供的硬件性能计数器(CHPC)的数据得到应用系统的性能数据，而不再监控系统本身的运行，从而达到最小限度影响应用系统的运行特性的目的[7]。这种可靠的并行读取是因为 CHPC 寄存器的值由 CPU 中的硬件实现并行采集[11]。

上一节中讨论得出的同步通信层 CL 实际上就是系统的并发通信所在，这样我们在每一个通讯层的入口点和出口点

(下转第221页)

porttask analyses. In. Proc. of INTERACT'90, 1990. 259～264
8 Johnson H P. Task knowledge structures: psychological basis andintegration into system design. Acta Psychologica, 1991, 78: 3～26
9 Lim K Y, Long J B. Instantiation of Task Analysis in a StructuredMethod for User Interface Design. In: Proc. of Task Analysis in H-C-I, 11th Interdisciplinary Workshop on "Informatics and Psychology", Schaerding, Austria, June, 1992
10 Dumas J S, Redish J C. A practical guide to usability testing. Ablex Publishing Corporation, Norwood, 1994

(上接第208页)

分析当前的性能数据，可以得到系统运行到该节点的CPU执行的总周期数，出口与入口两者的差就是该层运行所消耗的CPU时钟周期数，也就是该层的时间性能。考察图2，为了得到线程1各个关键边的执行代价，我们在顶点 V_{13}, V_{14}, V_{15}, V_{16}, V_{17} 处分别插入PAPI CPU时钟周期数 *PAPI_TOT_CYC* 采集函数 *PAPI_read* (*EventSet*, *values*)，其中EventSet为当前所监控的事件集{*PAPI_TOT_CYC*}，values返回监控的事件值。之后，运行系统，就可以得到所有监控点的 *PAPI_TOT_CYC* 值，然后通过计算分析，得到表1第3行的各个线程的性能数据，关于PAPI的详细描述可参阅文[11]。

6 实验结果与分析

在图1中标明了通过实验得到的运行时各个线程的动态时钟周期数，然后按照第5节给出的方法，在第2步的时候我们得到图2(a)的结果，就是在简化第一层后的简化的同步图。然后进一步简化得到图2(b)，这样就得到最后的系统性能数据表1。

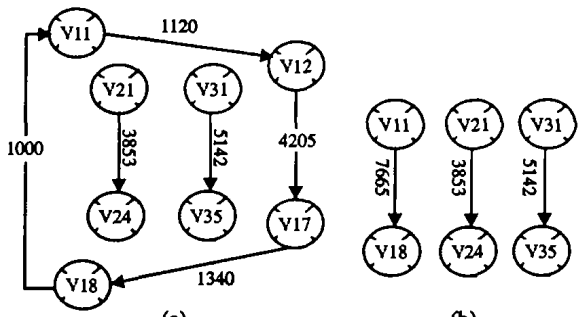

图2 同步简化图

从图1的边的权重值可以看到，所有的信号量消费者的位置，其时钟周期数在1700左右，而在生产者的位置，为400左右，这个差值很明显就是两个线程间的同步代价，因为对于信号量的消费者来说，它需要等待信号量生产者发出信号，而对于生产者来说，直接赋值即可产生信号量(此处忽略同步临界区进入时间，因为无论对于生产者还是消费者，都有这样一个消耗)，可以看出对于一个并发多线程的系统，同步通信代价是不可随意忽略的，而采用PERC-PAPI技术得到的性能数据和分析结果有比较高的精确度。

表1 在不同分析技术下的各个进程的性能数据比对

类型	P1	P2	P3	系统	误差比
期望值(E)	7300	3600	4800	7300	0%
静态分析数据(S) 忽略同步代价	6500	2800	4100	6500	−11.43% ((S-E)/E)
采用PAPI动 态采集(P)	7665	3853	5142	7665	+4.79% ((P-E/E)

结论 本文引入了一种动态性能分析技术，并将其应用到了并发线程通信系统的性能评估中。通过本文的论述，我们可以看到在实际应用中并发线程间的同步通信代价对系统的整体性能影响是不可随意忽略的，通过引入PERC-PAPI技术，使得对于这一类系统的性能分析和评估的精确度得到较大提高。在今后的研究中，将考虑应用这种技术到集群并行处理应用中，同时因为现在的分析是完全依赖于源代码级的"探针"插入方式得到，使其应用有一定的局限性，因此也要考虑如何完成不需要源代码而直接插入和监控目标代码进行分析。通过PAPI监控技术实际可以得到更多的除了时间性能之外的程序行为的底层表现，如Cache(高速缓存)和TLB(Tranlate Lookaside Buffer)的使用情况，分支预测情况等，从而可以提供给系统分析和结构设计师更多和更小粒度的性能数据，进一步优化和完善应用系统，特别是并行系统和多线程并发系统，也可用于这类系统的运行状态监控中。

参考文献

1 Bhattacharya S, Dey S, Brglez F. Performance Analysis and Optimization of Schedules for Conditional and Loop-Intensive Specifications. In: Proc. Design Automation Conf. June 1994. 491～496
2 Dey S, Bommu S. Performance analysis of a system of communicating system. C&C Research Laboratgories, IEEE, 1997
3 Zhang Tianhao, Cao Feng, Dewey A M, et al. Performance Analysis of Microelectrofluidic Systems Using Hierarchical Modeling and Simulation. IEEE, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, 2001, 48(5)
4 Wilkinson B, Allen M. 陆鑫达译. 并行程序设计. 机械工业出版社, 2002. 50～62
5 Kuehlmann A, Bergamaschi R A. Timing Analysis in High-Level Synthesis. In: Proc. of the IEEE Intl. Conf. on computer-Aided Design, Aug. 1992
6 Narayan S, Gajski D D. System clock Estimation based on Clock Slack Minimization. In: Proc. of the European Design Automation Conf. 1992
7 London K, Dongarra J, Moore S, Mucci P, Seymour K, Spencer T. End-user Tools for Application Performance Analysis, Using Hardware Counters. In: Intl. Conf. on Parallel and Distributed Computing Systems, Dallas, TX, Aug. 2001
8 Brewer F, Gajski D. Chippe: A System for Constraint Driven Behavioral Synthesis. IEEE Transactions on Computer Aided Design, 1990, 9: 681～695
9 Miller J, Kramer H. Analysis of multi-process specifications with a petri-net model. In: Proc. EURODAC'93, European Design Automation conference with EURO-VHDL'93, 1993. 474～479
10 Bailey D H, Supinski B D, et al. Performance Technologies for Peta-Scale Systems: A White Paper Prepared by the Performance Evaluation Research Center and Collaborators. Response to the Invitation to Submit White Papers on High End Computing, May 2003
11 ICL of University of Tennessee. PAPI Programmer's Reference. http://icl.cs.utk.edu/