利用 Itanium2 的 PMU 部件开发程序性能分析工具

张宇峰
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:Itanium2 处理器以寄存器组的形式提供的性能监视单元实现了在程序运行过程中捕捉微结构事件的功能。文中介绍了以 Linux 为 Itanium2 的性能监视单元提供的接口 perfmon 为基础的开发相对高端的性能分析工具的方法,以实现对这些由性能监视硬件提供的数据进行综合处理利用。

关键词:程序性能分析;安腾2;性能监视单元
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2006)08-0069-03

Developing Performance Analysis Tool Using Itanium2 PMU

ZHANG Yu-feng
(Computer Institute, National University of Defence Technology, Changsha 410073, China)

Abstract: Itanium2 processors are providing hardware support in the form of performance registers that help to capture microarchitectural events that occur during the running of a program. In this paper, we introduce the method to develop the application performance analysis tool based on perfmon under Linux that can assemble various types of complex performance related data available from the performance registers and provide a high level summary of the data.

Key words: application performance analysis; Itanium2; performance monitoring unit

0 引 言

近年来,微处理器速度的提高极大地推动了程序性能的提高。然而,计算机的其余部件的发展不仅跟不上微处理器速度提高的脚步,甚至成为了妨碍程序获得峰值性能的瓶颈所在。比如,在层次存储器体系结构中,一个产生了较长延迟的 Cache Miss 也许就会对处理器性能产生非常大的影响;在流水线中,对分支情况的错误预测也有可能使处理器陷于停顿状态。因此,全面分析程序的动态行为,指出并修正程序的瓶颈所在(如 Cache Miss、分支的错误预测),将是程序优化以使程序达到最高性能的一个重要方面。然而,随着现代处理器引入的指令级并行、处理器流水线、层次存储结构等技术,性能分析工作变得越来越复杂并难以开展了。为了简化性能分析工作,现代处理器大多以性能寄存器组的方式提供了捕捉那些与能够影响程序性能的事件相关的数据的硬件机制。其中,IA-64 架构的处理器都提供了一组被称为性能监视单元[1](Performance Monitoring Unit,PMU)的部件,它能够提供底层的 Profiling 信息,并捕捉那些在程序运行期间发生的微结构[1](microarchitecture)事件的信息。PMU 可由程序设定在什么时候捕捉什么事件,尽管这些由硬件寄存器提供的信息都是非常有价值的,但还需要对这些底层信息和数据进行大量的后续处理工作才能确定程序性能瓶颈所在并分析出产生瓶颈的原因[2]。从这个方面来看,开发有效的性能分析工具,对这些底层的数据进行分析处理并生成用户可接受并理解的更具指导性的信息,也将是非常有现实意义的。

收稿日期:2006-01-20
作者简介:张宇峰(1978-),男,湖南长沙人,硕士研究生,助理研究员,研究方向为并行与分布处理。

1 Itanium2 的硬件性能监视部件 PMU

IA-64 架构的处理器通过 PMU 机制来收集程序性能信息。PMU 由一组专用寄存器构成,可对诸如 L1 Cache Miss 等微结构事件进行计数。PMU 主要包括收集被监视事件数据的性能监视数据寄存器(Performance Monitor Data Register,PMD)和控制被监视事件的性能监视配置寄存器(Performance Monitor Configuration Register,PMC)。

Itanium2 的 PMU 提供了 4 个 48 位的性能计数器并可同时对 4 个微结构事件进行监视,这 4 个计数器使用了 4 对标注为 PMC4-PMC7 和 PMD4-PMD7 的寄存器。PMC4-PMC7 用于指定对应的 PMD4-PMD7 中进行计数的事件。当 PMCn 指定的事件发生时,PMDn 中的计数器就加一,当计数器溢出并归零时,溢出状态寄存器 PMC0-PMC3 就会检测到并保存这次溢出情况。PMU 可由程序设定来产生中断而不管 PMD 寄存器是否真正溢出。这一中断机制可被操作系统用来对每个硬件计数器维护一个 64 位的软件计数器。

PMU 还提供操作码匹配器和地址范围匹配器以对要被监视的事件实现更好的控制，通过这些分类控制器，PMU 可将停顿节拍分类从而定位程序中存在的问题。比如如果有大量的停顿节拍是由指令的存取访问导致的，那么增加对指令的预取就有可能会改进程序的性能。

对处理器的节拍的计数可以有助于对程序的全局行为的了解，但并不能了解程序中哪些地方是与影响程序性能的事件相对应的。为了解决这个问题，Itanium2 还提供了一组事件地址寄存器（Event Address Registers，EARs），EARs 可以通过与事件相关的指令指针（Instruction Pointer，IP）来捕捉事件信息。在没有 EARs 时，当某事件发生时，IP 地址中保存的也许是多条导致该事件发生的指令中的一条，遇到这种情况时，程序员或者性能分析工具就必须进行进一步的分析以“猜测”哪条指令才是真正导致该事件发生的指令。而现在，EARs 就可以直接确定引发事件的指令地址了。

除了 EARs，PMU 还提供了一个容量为 8 的分支路径缓冲（Branch Trace Buffer，BTB）以记录分支指令的执行路径。这样就可以通过设置 BTB 以专门捕捉某一类的分支（如 taken branches，not taken branches，correctly predicted，incorrectly predicted branches）。BTB 可以记录 4 组分支事件的源地址和目标地址。

2 Linux 对 PMU 的支持

Linux 的 kernel 中包含了一个名为 perfmon[3,4] 的子系统以实现对 IA－64 的 PMU 的访问。perfmon 接口包含了一个简单的系统调用 perfmonctl()，由该系统调用来完成对 PMD 和 PMC 寄存器的读写访问的设置、评测和信息收集工作。为使 PMU 完成性能监视工作，主要需要设定 PMC 的事件信息、程序权限级别、计数器溢出是否产生中断等信息。对于基于事件的信息采样过程，还至少需要设置一个能够在采样周期结束时产生中断的计数器（采样周期是指两个样本之间的事件数目）。

在采样周期结束时，perfmon 溢出中断处理器将指定的 PMD 寄存器的信息送入采样缓冲，当采样缓冲填满后，perfmon 溢出中断处理器给被监视的任务发送一个信号，这样被监视的任务就可以通过信号处理器来处理采样缓冲中的信息。

为了简化对 PMU 的设置操作，可以使用一个叫 pfmon 的用户层的工具。pfmon 工作流程如图 1 所示。

为了理解 pfmon 在一次监视过程中收集的数据，同时也为了更好地理解被监视程序的行为过程，因此有必要对 pfmon 收集的数据进行更进一步的处理和分析，以使用户能够从更高的层次来理解和观察这些数据中蕴含的内容。

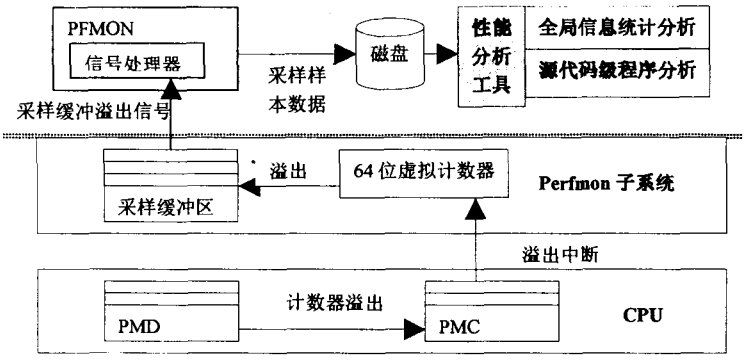

图 1 Linux 对 Itanium2 的 PMU 的支持

3 基于 pfmon 的性能分析工具的开发

pfmon 测试的数据多数是对原始性能事件计数，包括 CPU 周期数、净指令数、无效指令数、停顿周期数、各级 cache 的失效数、到达各级 cache 的访问次数、延迟周期数、机器的主频和各级 cache 大小等，而且由于 Itanium2 硬件的限制，pfmon 关于程序的一次监视过程只能监视其 4 个事件，为了全面而系统地了解程序的运行行为，就必须将程序运行多次，同时由于一般用户更关心源程序的浮点性能、cache 失效率、利用率、指令并行性等一些可以宏观反映程序性能的指标，这就需要对得到的计数器原始性能数据分析、整理和加工，以便呈现给用户直观地反映程序运行特征的数据，因此有必要开发基于 pfmon 的性能分析工具。从图 1 中可以看到，基于 pfmon 的性能分析工具的开发有以下两个主要开发方向：

1）站在全局的角度，对数据进行分类统计分析。

这种开发方向的思想是建立在程序运行多次且将被监视事件分类并自动统计，然后给出用户关心的指标信息的基础上的。开发流程如图 2 所示。可以将 pfmon 支持的事件和计数器分为以下 3 类：CPU 周期和指令计数器类、延迟计数器类、存储器计数器类。在将事件分类的基础上，再定义用户需要的性能指标，如程序浮点性能、利用率、各级 cache 失效率、延迟比、指令并行度（由参数 CPUI，CPI，UCPUI，UCPI 来反映）、程序运行时间等。然后建立事件计数器与性能指标之间的映射关系，如定义访问二级 Cache 的缺失率为 L2_MISS_RATION = L2_MISSES/L2_REFERENCE，其中 L2_MISS 和 L2_REFERENCE 为在程序运行过程中发生的二级 Cache 的不命中事件次数和对二级 Cache 的访问事件次数。最后则可编写程序对 pfmon 进行自动调用并完成事件分配设置等

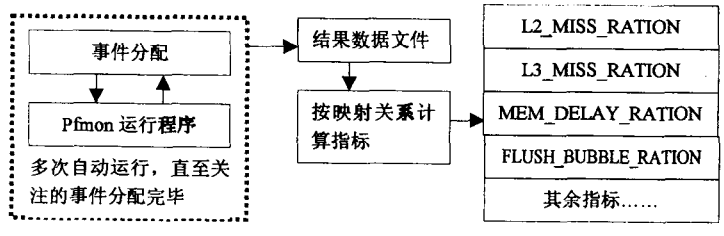

图 2 对数据进行分类统计分析的工具的组织结构

工作,待程序运行完毕后则可利用 pfmon 生成的数据,按照事件计数器与性能指标的映射关系进行计算从而得到对于程序的全局性信息。

这种方法的好处在于:一方面可使部分分析过程自动实现,减轻用户手工分析的负担;另一方面,将软件开发人员在机器模型、性能模型、程序性能优化等方面的专业知识融入到数据分析中,能够得到一些更具指导意义的性能指标,对程序性能优化更具指导意义。

2)从局部出发,将数据进行逆向映射,对应到源代码级别。

基于这种思想开发的工具是针对体现程序性能中的某一种事件进行分析的,并建立程序源代码语句与被监视的事件的关联,有以下几个方面可以利用来实现从事件到源代码的精确定位。

首先,可以按照事件地址寄存器 EARs 中收集的 IP 地址对采样样本进行排序和分类以找出出现频率最高的 IP 地址[5],在大多数情况下,这些地址就代表着反映性能瓶颈的程序中的"热区(hot spots)"所在。比如,如果监视的是指令 Cache Miss 事件,那么 EARs 的 IP 地址中出现的最多的就应该是 Cache 中缺失最频繁的指令。按照程序局部性原理,80% 的程序执行时间是集中在 20% 的代码段上,因此,确定程序的"热区"对于改进程序性能是非常重要的一个环节。

其次,可以将收集的采样样本与分支路径缓冲的内容联系起来,以建立与被监视事件相符合的程序执行路径。

最后,可以利用在程序编译过程中生成的反映执行过程信息的调试符号表(debugging symbol table)来将 IP 地址映射为源代码中的相应位置,也就是指出产生被监视事件的语句的行号。可以通过 Linux 下的工具软件 GDB 将调试符号表保存为文件,然后对文件进行分析,通过从 EARs 中读取的 IP 地址与调试符号表中的内容进行对照匹配,得到产生 EARs 中 IP 地址的指令与源程序中某一行语句的对应关系,这将有助于定位那些产生瓶颈的代码和数据结构并分析产生瓶颈的原因。这种开发工具的处理流程如图 3 所示。

这两种开发方向各有利弊,第一种全局性的数据分析方法虽然只能概略性地指出程序性能瓶颈存在于哪个方面,但其实现过程简单;而第二种局部性的方法能够具体精确地指出产生性能瓶颈的语句所在,但其实现过程较为烦琐,开发周期长,因此很难评说这两种开发方向的优劣,只能说,应该在面对不同的应用环境下可以选用不同的开发方向,当然,两者的结合将是更加理想的状态,只不过其开发更为复杂。

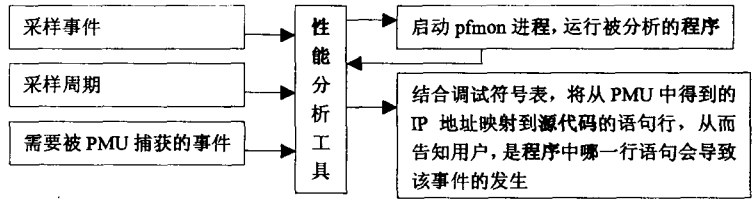

图 3　对数据进行逆向映射的工具的组织结构

4　结束语

程序性能分析对于程序优化的意义将随着计算机结构的不断更新发展而显得越来越重大。中国古语云:工欲善其事,必先利其器。因此,开发适应用户需求的性能分析工具必将极大地推进程序性能分析工作。文中介绍的基于 pfmon 的开发方法,加以变通,必将能够适应 IA-64 家族中的其余微处理器。

参考文献:

[1] Intel Corporation. Intel Itanium 2 Processor Reference Manual for Software Development and Optimization [EB/OL]. http://developer.intel.com/design/itanium2/manuals/index.htm. 2002-06.

[2] Jarp S. A Methodology for using the Itanium 2 Performance Counters for Bottleneck Analysis, Tech-Report HP Labs [EB/OL]. http://www.gelato.org/pdf/Performance-counters-final.pdf, 2002-08-27.

[3] Mosberger D, Eranian S. IA-64 linux kernel: Design and Implementation[M]. [s.l.]: Prentice Hall PTR, 2002.

[4] HP Labs. Perfmon Project web site[EB/OL]. http://www.hpl.hp.com/research/linux/perfmon/, 2005-02.

[5] Lingamneni A. PerfView - A Tool for Source-Level Performance Analysis on the Itanium Processor Family[EB/OL]. http://dynopt.dtc.umn.edu/documents/perfview-ananth-planb-report.pdf. 2004-06.

(上接第 68 页)

[12] Buyya R, Abramson D, Giddy J. Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid[A]. In Proceedings of the 4th International Conference and Exhibition on High Performance Computing in Asia-Pacific Region (HPC ASIA 2000)[C]. [s.l.]: IEEE Computer Society Press, 2000.283-289.

[13] Czajkowski K, Fitzgerald S, Foster I, et al. Grid Information Services for Distributed Resource Sharing[A]. In Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)[C]. [s.l.]: IEEE Press, 2001.181-195.

[14] Abawajy J H, Fault-Tolerant Scheduling Policy for Grid Computing Systems[A]. In Proceedings of the IEEE 18th International Parallel and Distributed processing Symposium (IPDPS'04)[C]. [s.l.]: IEEE Computer Society Press, 2004.238-244.