第34卷 第19期 计 算 机 工 程 2008年10月
Vol.34 No.19 Computer Engineering October 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)19—0271—03 文献标识码：A 中图分类号：TP311

基于 pfmon 的性能测试与分析工具 Codemon

赵永刚[1]，付立东[2]
(1. 西安卫星测控中心喀什测控站，喀什 844000；2. 西安科技大学计算机系，西安 710054)

摘 要：通过对 Linux/IA64 平台上性能数据监测工具 pfmon 各类硬件计数器功能的分析，提出一种基于停顿时钟周期数测试存储延迟分布的监测模型，在此基础上设计和实现了程序自动测试与分析工具 Codemon。应用该工具测试二维非线性扩散方程 BNLAG2D 求解程序模型的源代码，给出程序的性能评价和优化重点。该程序优化后性能提高了 15%。

关键词：IA64 体系；硬件性能监控；程序优化

Codemon: Tool for Performance Test and Analysis Based on pfmon

ZHAO Yong-gang[1], FU Li-dong[2]
(1. Kashi Test & Control Station of Xi'an Satellite Control Center, Kashi 844000;
2. Dept. of Computer, Xi'an University of Science and Technology, Xi'an 710054)

【Abstract】 A method for using the Itanium 2 performance counters for bottleneck analysis is very complex and inconvenient. To cover this shortage, new model based on pfmon is proposed to monitor the performance issues in programs by using the "bubble" counters for stall analysis. The Codemon derived from the model is designed and implemented. In additions, the tool is used to characterize an application named "BNLAG2D", the results can tell user what about this program is and how to improve the performance. Performances achieved is 15 percent higher than the original program.

【Key words】 IA64; hardware performance monitor; program optimization

pfmon是一种为Linux/IA64 平台设计的性能监测工具，它使用IA64 体系结构提供的性能监控单元(PMU)对执行时的二进制代码进行性能相关事件的统计和采样[1]。事件统计利用Itanium处理器提供一组时钟周期统计监视器，这些监视器可用来统计消耗在不同微体系结构事件上的时钟周期数，了解程序内部的微体系结构行为。由事件统计结果可以知道哪些微体系结构事件阻碍了性能的提升，而提供的采样工具可以用来确认程序中那些比较耗时的代码。这对比较不同程序实现之间的性能差异、识别程序性能瓶颈、了解软硬件资源利用状况具有重要作用。

1 性能测试工具 pfmon

IA64 体系结构监控子系统由一组监控寄存器(PMC)控制。其中 4 个监控寄存器PMC4-PMC7 用于统计程序运行中出现的事件[2]，系统提供了 300 多个计数器事件，可以分成以下 5 类：(1)与CPU有关，如周期计数器、指令计数器；(2)与延迟有关，如停顿周期数(由指令停顿、CACH失效等引起的)；(3)与TLBS, ALAT, RSE有关；(4)与L1, L2, L3 各级cache有关；(5)与存储子系统\存储总线有关。

由于只有 4 个寄存器统计出现的 300 多个计数器事件，因此需要将pfmon程序运行几次，才能得到比较全面的性能参数。这里选SPECint库中的crafty程序作为例子，来说明如何应用pfmon对程序进行性能分析。

1.1 测试周期数、指令数及指令执行周期数

执行以下pfmon命令可以得到craft程序执行时的时钟周期数、指令数、无效指令数和停顿周期数。

```
pfmon–E cpu_cycles, ia64_inst_retired_this, nops_retired, back_end_bubble_all ./crafty <crafty.in
```

Craft程序运行时所需的输入数据从文本文件crafty.in中读入。指令执行周期数CPI = cpu_ cycles/ ia64_inst_retired_ this；有效指令数＝ia64_inst_retired_ this- nops_retired；非停顿的周期数＝cpu_cycles-back_end_ bubble_all。

这样可以定义其他 3 个参数，CPUI 表示每个有效指令的 CPU 周期数，UCPUI 表示每个有效指令的非停顿 CPU 周期数，UCPI 表示每个指令的非停顿 CPU 周期数。从测试结果虽然计算出 UCPUI 值是 0.36，说明每个无停顿周期中将近包括 3 个有用的指令，但停顿周期数占总周期却超过 50%，因此，接下来对停顿周期数进行进一步的测试。

1.2 停顿周期计数器测试

Itanium 2 处理器中有 5 个统计各种停顿的计数器，停顿可以按以下公式细分：

```
Back_end_bubble_all=Be_flush_bubble_all + Be_L1d_fpu_
bubble_all + Be_exe_bubble_all+ Be_RSE_bubble_all +Back_end_
bubble_fe
```

停顿计数器可以分解是因为每个子停顿计数器都与流水线中特定的阶段有关系，所以只要有停顿插入流水线的特定阶段就可以增加相应的计数器。再次运行 pfmon 工具，测试出 4 个子停顿计数器的值，由于停顿周期的总数已知，因此可以知道这 5 个子计数器的值。

Be_exe_bubble_all 计数器的值在其中占比重较大，说明在流水线执行阶段的停顿占有优势。将这些子计数器进一步分解，理解引起停顿的更深层次的微体系结构事件。

作者简介：赵永刚(1976－)，男，工程师、硕士，主研方向：高性能计算；付立东，讲师、硕士

收稿日期：2007-11-15 **E-mail：**ygzhao@nudt.edu.cn

Be_flush_bubble_all= Be_flush_bubble_bru+ Be_flush_bubble_xpn，其中 Be_flush_bubble_bru 计数器值反映了分支预测失败引起的停顿周期数，Be_flush_bubble_xpn 计数器的值反映出由异常或中断事件引起的停顿周期数。Be_l1d_fpu_bubble_all ＝ Be_l1d_fpu_bubble_l1d ＋ Be_l1d_ fpu_bubble_fpu，一般来说，在程序中，Be_l1d_fpu_bubble_ l1d 计数器值占有较大的比重。

1.3 全局停顿分类

通过对上面测试的各种停顿计数器进行重新分类组合，可以对全局停顿有总体的了解。从表 1 可以看出，虽然停顿计数器多而复杂，还是可以把它分成 6 个有意义的类别。从性能的观点来看，大多数程序是和craft相似的，一般来说，停顿周期通常是由数据存储访问引起的，分支失败预测的延迟数一般占第 2 位。

表 1 全局停顿分类

停顿分类名	计算公式	计数值($\times 10^9$)	比例/(%)
数据访问停顿	Be_exe_bubble_grall-Be_exe_bubble_ grgr+Be_l1d_fpu_bubble_l1d	42.60	50.0
分支预测失败	Be_flush_bubble_bru+Fe_bubble_ bubble +Fe_bubble_branch	16.80	19.7
指令失效停顿	Fe_bubble_imiss	9.80	11.5
RSE 停顿	Be_rse_bubble_all	7.50	8.8
FLP 单元	Be_exe_bubble_frall+Be_ ld_fpu_bubble_fpu	7.00	8.2
GR 记分牌停顿	Be_exe_bubble_grgr	0.95	1.1
合计		84.565	99.3

1.4 存储计数器测试

性能监控子系统可以统计每级 Cache 中出现的事件。为了描述这些事件与停顿周期数的关系，现在主要测试与数据访问和失效有关的事件。在 2 级 Cache 中需要测试的计数器有 L2_data_references_L2_all, L2_misses。在 3 级 Cache 中需要测试的计数器有 L3_references, L3_misses, L2dtlb_misses。

1.5 联系存储计数器和停顿计数器的公式

给定如下公式，它可以将存储计数器和停顿计数器联系起来。这样做的目的是找出存储层次系统中产生大量停顿周期数的部分。

Bubbles =(L2_data_references_L2_all – L2_misses_adjusted)×N1+
(L2_misses_adjusted – L3_missed_adjusted)×N2+
L3_misses_adjusted×N3+ L2dtlb_misses×N4

公式简化了每级 Cache/TLB 产生的停顿数的计算方法，停顿数等于各级存储器的访问数乘以对应的平均访问时钟周期数。其中，前 2 项对应 cache 延迟，第 3 项对应内存延迟，最后一项对应 TLB 延迟。调整 L2 失效数和 L3 失效数的原因是失效数也应该包括由读或预取指令引起的失效。因此，计算 L2 访问失效数采用了值 2/3 来调整原来的值，见表 2。

表 2 访问周期数估计

	L2	L3	内存	TLB
访问时间	5＋	12＋	110－190	32
推荐因数	$N1 = 2$	$N2 = 10$	$N3 = 150$	$N4 = 30$

将前面测试的与 Cache 有关的计数器的值代入公式，数据访问耗费的周期数如下：

Bubbles=(L2_data_references_L2_all–L2_misses)×2+(L2_misses–
L3_missed)×10 + L3_misses×150 + L2dtlb_misses×
30 =40.4

计算结果 40.4 与由停顿计数器测出的数据访问停顿周期数 42.6 值基本相当。计算 cache 延迟、内存延迟、TLB 延迟所占的百分比，cache 延迟比率接近 100%，可以知道 crafty 所需的全部数据都在 cache 中。

2 pfmon 测试方法小结

上面综合介绍了 3 类性能计数器，提出一种理解应用问题性能特征的方法[3]，该方法测试的内容包括：

(1)CPU 周期和指令计数器。它可以计算出 CPI(每条指令占有的 CPU 周期数)，这样可以得到程序运行好坏的总体情况。

(2)延迟计数器。通过它可以更好地理解应用程序中延迟的特征，延迟可以分为以下几种：D-CACH(数据CACH)延迟，分支预测失败延迟，I-CACH(指令CACH)延迟，RSE(寄存器堆栈引擎)延迟，FLP延迟，GR记分牌延迟，Front-end刷新延迟。其中最主要的是前 2 种延迟。

(3)存储计数器。通过它可更详细地了解程序中 D-CACH 延迟产生的具体情况。

上面给出了程序 crafty 性能测试与分析的过程，这是在实践中总结的应用 pfmon 测试程序的一种方法，即基于停顿周期数的测试分析方法。

3 pfmon 工具的不足

从上节对测试工具pfmon使用方法的介绍看，它提供了丰富的性能监控硬件。就事件统计这一方面，提供了300多个计数器用于描述可能出现的微体系结构事件，事件集丰富，几乎对程序运行时的状态都进行了详细的记录。虽然工具功能强大，可采样和统计的事件全面，但系统庞大，自动化程度不够，提供的是对处理器内部微体系结构事件的基本统计数据，没有对数据分析，使用起来不方便，结果不直观。

另一方面，由于 pfmon 是基于硬件寄存器测试方式的软件，而且只有 4 个寄存器(PMC4~PMC7)统计出现的事件，为了得到全部相关方面的性能，需要将性能分析程序运行几次，而且由于内在的硬件冲突，一些计数器在同一次运行中不能同时使用，因此要求用户理解各事件之间的关系或应用 pfmon 命令检测，以确保一次运行中不存在相冲突的事件，并且 pfmon 提供的参数选项繁多，这都增加了用户测试操作的难度。在实际的测试中，由于硬件寄存器在代码运行期间对处理器内部操作进行现场监控，因此本来就需要长时间计算的数值程序运行周期更长。

4 codemon 工具的设计与实现

本文提出一种基于 pfmon 的性能测试和分析模型，一方面希望涉及的参数尽量少，以减少测试次数节省时间，另一方面也希望它能较全面地反映程序某一方面的基本特征。第 2 节介绍了 pfmon 的一种使用方法，该方法分 3 个方面进行测试，较系统地得到了程序运行时特征。它是一种对科学计算程序可行的性能测试方法，测试中共用到 48 个计数器，最少需要测试 12 次，也相当耗时。

通过对上面提到的测试方法中停顿周期数的分析可以发现：Be_l1d_fpu_bubble_all 的值与 Be_l1d_fpu_bubble_l1d 的值相等；Be_exe_bubble_all 的值和(Be_exe_bubble_ grall-Be_exe_bubble_grgr)值相差不大；因为 D_CACH 延迟= Be_exe_bubble_grall-Be_exe_bubble_grgr+Be_L1d_fpu_bubble_L1d ，所以将求数据 CACH 延迟的公式简化成以下：D_CACH 延迟=Be_exe_bubble_all+Be_l1d_fpu_bubble_all。

这样在延迟数的测试中，只需 2 个计数器的值，不用再测试下一级子停顿计数器的值，大大减少所测试的计数器的数量，但对数据 CACH 延迟周期数的影响不大，不会影响到测试结果。D_CACH 延迟周期计算表达式可以简化的原因是

由科学计算程序内在的特性决定的。

对于大多数程序来说，用户希望了解的运行时特征基本是相同的。先定义需要的性能指标，与测试平台有关的：CPU 主频，主存容量，各级 Cache 容量，各级 Cache 块大小，操作系统和测试工具的版本等；与测试程序有关的：程序浮点性能，利用率，各级 Cache 失效率，延迟比，DCache 延迟比，各级 Cache 延迟比，L2TLB 延迟比，内存延迟比，指令并行度(由参数 CPUI, CPI, UCPUI, UCPI 来反映)，程序运行时间。

由前面的介绍可知，通过 pfmon 获得的性能参数主要是程序执行期间完成的各种操作的计数，希望测试软件能够在这些原始数据的基础上，进行自动的分析与计算，向用户提供更高层次的数据，比如上面提到性能指标。这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，见表 3。

表 3　Codemon 用到的 16 个计数器及其各自的含义

计数器名称	含义	计数器名称	含义
cpu_cycles	时钟周期数	Ia64_inst_retired_this	指令数
Nops_retired	无效指令数	Be_rse_bubble_all	寄存器堆栈引擎停顿周期数
L3_references	3 级 cache 访问数	Be_l1d_fpu_bubble_all;	l1d 和 fpu 微流水线停顿周期数
L3_misses	3 级 cache 失效数	Be_exe_bubble_all	流水线执行单元停顿周期数
L2dtlb_misses	TLB 失效数	Be_flush_bubble_all	分支预测失败引起的停顿周期数
L2_references	2 级 cache 访问数	L2_data_references_L2_all	2 级 cache 的数据访问次数
L2_misses	2 级 cache 失效数	FP_OPS_RETIRED	浮点操作计数器
Back_end_bubble_all	停顿周期数	Back_end_bubble_fe	后端执行停顿周期数

Codemon 定义的各指标名称、含义和计算公式如下：

(1)反映程序总体性能

Prog_cpui=Cpu_cycles/(Ia64_inst_retired_this-Nops_retired);

Prog_cpi=Cpu_cycles/Ia64_inst_retired_this;

Prog_ucpui=(Cpu_cycles-Back_end_bubble_all)/(Ia64_inst_retired_this-Nops_retired);

Prog_ucpi=(Cpu_cycles-Back_end_bubble_all)/Ia64_inst_retired_this;

Prog_deley_ration=Back_end_bubble_all/Cpu_cycles;

其中，Prog_deley_ration 反映停顿数占整个时钟周期数的比率，由这 5 个参数可以分析出程序整体性能的好坏。

(2)停顿分布情况

Be_flush_bubble_all_ration=Be_flush_bubble_all/Back_end_bubble_all;

Be_rse_bubble_all_ration=Be_rse_bubble_all/Back_end_bubble_all;

Back_end_bubble_fe_ration=Back_end_bubble_fe/Back_end_bubble_all;

Be_l1d_fpu_bubble_all_ration=Be_l1d_fpu_bubble_all/Back_end_bubble_all;

Be_exe_bubble_all_ration=Be_exe_bubble_all/Back_end_bubble_all;

上面 5 个指标是各个停顿计数器的值占总停顿数的比率，反映停顿的分布情况。

Prog_access_delay_ration=(Be_l1d_fpu_bubble_all+Be_exe_bubble_all)/Back_end_bubble_all

该参数反映存储延迟占总停顿数的比率，可以得出数据局部性的好坏。

(3)各级存储器访问延迟分布情况

Prog_l2_miss_ration=L2_misses/L2_reference;

Prog_l3_miss_ration=L3_misses/L3_reference;

Prog_Cache_delay_ration=((L2_data_references_L2_all-L2_misses×adjusted)×access_time.N1+(L2_misses×adjusted-L3_missed×adjusted)×access_time.N2)/Bubbles;

Prog_mem_delay_ration=L3_misses×adjusted×access_time.N3/Bubbles;

Prog_tlb_delay_ration=L2dtlb_misses×access_time.N4/Bubbles

前 2 个参数描述 L2,L3 cache 的命中率，后 3 个参数描述 cache 延迟、内存延迟和 TLB 延迟占总存储延迟的百分比，Bubbles 是 3 项延迟之和。

(4)系统参数

Sys_info：描述 CPU 的主频、型号、主存容量、各级 cache 容量及行大小

Exec_time：给出程序执行时间

FP_speed=FP_ops_retired/Exec_time：浮点性能，由浮点操作数除以程序运行时间得到

SYS_utilize=FP_speed/PEAK; 利用率，其中，PEAK 是机器峰值性能

通过上面定义的 18 个指标参数，可以比较全面地反映程序性能特征，能指导应用程序面向体系结构特点进行性能优化。测试中共用到 16 个计数器，通过对计数器内在硬件冲突测试，可以将它们分为 4 组。

codemon 的组成主要有 3 个部分：(1)监控执行模块。负责通过系统调用启动 pfmon 命令行性能监控环境，按照给定参数对硬件寄存器进行控制与访问，获得程序执行期间的 16 个原始性能参数和程序执行时间。(2)数据分析模块。在原始性能参数、机器参数等数据的基础上进行分析计算，得到 18 个综合性的性能数据。(3)显示模块。将综合性的性能数据显示到用户终端或输出到格式化的报告文件中。

5　Codemon 应用实例

BNLAG2D程序用于求解二维非线性扩散问题[4]，应用codemon进行测试后结果见表 4。

表 4　程序的性能指标

指标名称	值	指标名称	值
Prog_cpui	1.22	Prog_l2_miss_ration;	0.04
Prog_cpi	0.89	Prog_l3_miss_ration;	0.77
Prog_ucpui	0.30	Prog_Cache_delay_ration;	0.31
Prog_ucpi	0.22	Prog_tlb_delay_ration;	0.00
Prog_deley_ration	0.75	Prog_mem_delay_ration;	0.69
Prog_access_delay_ration	1.00	Be_flush_bubble_all_ration	0.00
Be_l1d_fpu_bubble_all_ration	0.80	Be_rse_bubble_all_ration	0.00
Be_exe_bubble_all_ration	0.18	Back_end_bubble_fe_ration	0.00
Exec_time/s	772	FP_speed	5.1E+8
SYS_utilize	0.09	Sys_info	

由表 4 可以看出：(1)在总的延迟中存储访问延迟大约占 75%；(2)cache 访问延迟约占存储访问延迟 31%，内存延迟占 69%；(3)存储延迟主要是由 3 级 cache 失效引起的。

6　结束语

针对 pfmon 测试软件的不足及实际程序性能分析与优化的需要，在分析 pfmon 工具的基础上开发了一个基于 Linux/IA64 的用 C 语言编写的程序性能测试软件 Codemon。该软件有下列特点：

（下转第 282 页）