基于隐马尔可夫模型的热路径预测算法研究 *

刘 魁[1], 李实英[1], 李 蕊[1,2], 李仁发[1]
(1. 湖南大学 计算机与通信学院, 长沙 410082; 2. 国防科学技术大学 计算机学院, 长沙 410073)

摘 要: 基于热路径的动态优化技术是动态二进制翻译器中提高软件运行效率的一种有效方法。如何利用基本块中已有的有限历史运行信息来识别热路径并提高它的预测命中率,同时保持计算开销没有增加是研究的重点。已有的热路径识别算法中基于模型进行预测的方法非常少,算法实现比较复杂。基于隐马尔可夫模型提出一种改进的热路径预测算法。由于状态转移序列惟一,该算法实现简单,可以提高热路径的命中率,在一定程度上改善动态二进制翻译器的性能。最后通过实验对所提出算法的有效性进行验证。

关键词: 动态二进制翻译; 动态优化; 热路径; 隐马尔可夫模型
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2010)07-2468-04
doi:10.3969/j.issn.1001-3695.2010.07.018

Algorithm for hot paths prediction using hidden Markov model

LIU Kui[1], LI Shi-ying[1], LI Rui[1,2], LI Ren-fa[1]
(1. *School of Computer & Communication, Hunan University, Changsha* 410082, *China*; 2. *School of Computer, National University of Defense Technology, Changsha* 410073, *China*)

Abstract: Method of hot paths-based dynamic optimization is effective for improving the operational efficiency of the software in dynamic binary translator. This study focused on how to identify the hot paths by using the existing limited amount of previous operational information of basic blocks, and to enhance the hit rate of the prediction, with no increase of computational cost at the same time. There had been few methods based on models among exsiting hot paths prediction algorithms, which need complicated implementation. This paper proposed an improved hot paths prediction algorithm based on hidden Markov model. Since the sequence of state transition was unique, this algorithm was easy to implement, and could improve the hit rate of hot paths as well as the performance of the dynamic binary translator. The experimental results verified the efficiency of our algorithm.

Key words: dynamic binary translation; dynamic optimization; hot path; hidden Markov model(HMM)

0 引言

动态二进制翻译技术是一种即时编译技术,它在程序的运行过程中将针对源体系结构编译生成的二进制代码(源机器码)动态翻译为可以在目的体系结构上运行的代码(目标码),此过程对用户来说是透明的。整个动态翻译过程分为两个阶段,即产生本地代码的翻译阶段和执行阶段。在代码的执行阶段,动态优化器会进行一定的优化。大多数的程序将大部分的时间花费在很小的一部分代码段上,识别并优化这一部分代码将从本质上改善软件的整体性能。

频繁执行的代码块称之为热块。代码块就是一个控制转移(如一个分支、调用或跳转指令)结束的指令序列,代码块也称为基本块[1]。当一个代码块变热时,其周围的一些代码块也将变热,由这些热块组成的执行序列称之为热路径。一个热路径就是一个指令序列。热路径是研究人员在设计 dynamo[2] 系统时提出来的概念,热路径的优化技术是目前动态二进制翻译器主要采用的技术。常见的热路径识别方法主要有分别基于基本块、边和路径的识别方法。这三种方法的预测准确度递增,但是复杂度也随之增加,尤其是基于路径的预测,随着程序的运行,负载急剧增长反而会降低软件的运行效率[3,4]。

热路径的产生一定与循环执行有关,因此预测主要针对循环进行,只有热路径优化带来的收益大于开销时才能从整体上提高系统的效率。因此,在热路径的优化过程中既要尽量提高热路径的预测准确率,同时又要控制预测过程的开销,并且优化越早启动越好。已有的热路径预测算法出于计算复杂度的考虑都没有基于模型进行预测,如 spanning tree 算法[5]、bit tracing 算法[3]、NET 算法[3]、编码算法[6],大多只是计算路径的执行次数,并取执行次数最多的作为热路径,研究的重点是如何更方便、更高效地记录热路径以及更新路径计数器的方法,尤其是当候选热路径的执行次数近似或者执行均衡时,更是很难作出合理的选择。本文基于隐马尔可夫模型(HMM),提出了改进的热路径预测算法,该算法实现简单,在预测延迟不明显增加的情况下,提高了热路径的命中率,从而减少了热路径在 cache 中的替入替出消耗,一定程度上提升了软件在动态二进制翻译器上的运行效率。

1 总体思路

本文的总体思想是以程序的基本块为单位,将程序的流程

收稿日期: 2009-12-21; **修回日期**: 2010-02-18 **基金项目**: 国家自然科学基金资助项目(60673061);中国博士后科学基金资助项目(20080441285)

作者简介:刘魁(1984-),男,湖南岳阳人,硕士研究生,主要研究方向为可信软件(liukui359@163.com);李实英(1970-),女,江西九江人,副教授,博士,主要研究方向为计算机视觉、计算机图形学、图像处理、模式识别;李蕊(1980-),男,湖南湘乡人,讲师,博士,主要研究方向为可信计算;李仁发(1957-),男,湖南宜章人,教授,博士,主要研究方向为计算机系统结构、计算机应用技术、信息与通信系统.

构造成一个满足隐马尔可夫模型(HMM)[7]的有向图,并以图中自上至下的每层为一个状态,每层中的一个代码块可视为一个观察值,热路径即为模型的一个观察序列,程序的流转变为状态的转移,同时,任何时刻每个状态的转移只与前一个状态相关,而与时间以及其他状态无关。很显然,此时基于块的热路径预测过程满足马尔可夫性,是一个马尔可夫过程,即马尔可夫链;而且一个状态中有多个观察值,通过扩展后(详见 1.2 节),一个观察序列不能直接确定状态的转移序列,因此该过程又是一个隐马尔可夫过程。依据 HMM 的估算方法,一旦循环程序段的入口代码块的计数器 counter 达到阈值 trigger 时,就启动基于该模型的热路径预测,并选取候选热路径 candiHp 中估算概率最大的路径作为热路径。总体思路如图 1 所示。

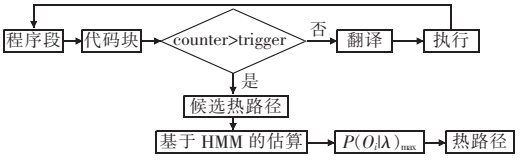

图 1　算法总体思路

1.1　隐马可夫模型介绍

如果一个过程的“将来”仅依赖于“现在”而不依赖“过去”,则此过程具有马尔可夫性,或称此过程为马尔可夫过程。设 S 是一个由有限个状态组成的集合 $S=\{1,2,3,\cdots,n-1,n\}$,随机序列 X 在 t 和 s 时刻所在的状态分别为 q_t、q_s($q_t,q_s\in S$)。若有 $P(q_t=j|q_{t-1}=i,q_{t-2}=k,\cdots)=P(q_t=j|q_{t-1}=i)$,或者 $P(q_t=j|q_{t-1}=i)=P(q_s=j|q_{s-1}=i)$,则随机序列 X 构成一个一阶马尔可夫链。令 $a_{ij}=P(q_t=j|q_{t-1}=i)(1\leqslant i,j\leqslant n)$,则对于所有的 i, j 有下面的关系成立:$\sum_{j=1}^{n}a_{ij}=1(a_{ij}\geqslant 0)$。

一阶马尔可夫模型可以描述为一个二元组 $\lambda(S,\boldsymbol{A})$。$S$ 是状态的集合,而 $\boldsymbol{A}$ 是所有状态转移概率 a_{ij} 组成的一个 n 行 n 列的矩阵,其中每一个元素 a_{ij} 表示从状态 i 转移到 j 的概率,可表示为 $\boldsymbol{A}=[a_{ij}]$,$a_{ij}=P(q_{t+1}=j|q_t=i)(1\leqslant i,j\leqslant n)$。一阶马尔可夫模型的状态与观察序列一一对应,因此根据观察序列能直接推断出状态转移序列。隐马尔可夫模型是对马尔可夫模型的一种扩展,观察序列不能确定状态的转移。隐马尔可夫模型 λ 可以表示为一个五元组 $\lambda=(S,V,A,B,\pi)$。S 和 $\boldsymbol{A}$ 与一阶马尔可夫模型一样分别表示一组状态的集合和状态转移矩阵;V 是一组输出符号组成的集合,$V=\{v_1,v_2,v_3,\cdots,v_m\}$;$B$ 是输出符号的概率分布,$B=\{b_j(v_k)\}$,其中 $b_j(v_k)$ 表示在状态 j 时输出符号 v_k 的概率,$b_j(k)=P(v_k|j)(1\leqslant k\leqslant m,1\leqslant j\leqslant n)$;$\pi$ 是初始状态概率分布,$\pi=\{\pi_i\}$, $\pi_i=P(q_1=i)(1\leqslant i\leqslant n)$,表示初始时刻选择某个状态 i 时的概率。

HMM 的状态是不确定或不可见的,只有通过观察序列的随机过程才能表现出来。观察到的观察序列与状态不是一一对应的,而是通过一组概率分布相联系。HMM 是一个双重随机过程(图 2),有两个组成部分:马尔可夫链和一般随机过程。前者描述状态的转移,用转移概率描述;后者描述状态与观察序列间的关系,用观察值概率描述。

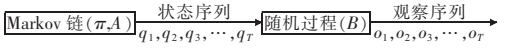

图 2　HMM 双重随机过程

基于模型 λ,若给定状态转移序列 $q=\{q_1,q_2,q_3,\cdots,q_T\}$,则产生观察序列 $O=\{o_1,o_2,o_3,\cdots,o_T\}$ 的概率 $P(O|\lambda)$ 为[7]

$$P(O|\lambda)=\sum_{q}P(O,q|\lambda)=\sum_{q_1,q_2,\cdots,q_T}\pi_{q_1}b_{q_1}(o_1)a_{q_1q_2}b_{q_2}(o_2),\cdots,a_{q_{T-1}q_T}b_{q_T}(o_T)$$

1.2　基于程序的图解及扩展

动态二进制翻译器的热路径优化针对的是频繁执行的代码块序列,主要是程序中循环执行的代码段。图 3 是一种典型的循环代码段,其中包括结构化程序中常见的顺序、选择和循环三种结构。图 4 为该程序段基于基本块的图解。由图 4 可以发现,程序中任何时刻基本块的转移仅与前一基本块有关,而与其他基本块以及时间无关。图中基本块旁边的数字表示在启动热路径预测时代码块的执行次数,由程序运行时的 profile 获取。

```
do
{switch(A)
  {cae 1:B;
       if(E)
       {I;continue;}
       else
       {J;break;}
case 2:while(C);
       {F;}break;
case 3:if(D)
       {G;}
       else
       {H;}
       J;break;
  }
}while(K)
```

图 3　循环程序段

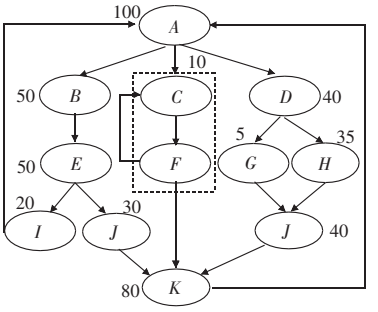

图 4　基于块的程序段图解

通过对图 4 的扩展,可使其满足隐马尔可夫性。首先将循环中的内循环合并成一个块,本文以代码块较少、循环次数较少的常见内循环为例进行介绍,当然,当内循环较复杂时可多次调用本算法。将图中的每个块视为 HMM 中一个输出或者是观察值,也即模型 λ 中集合 V 的元素,$V=\{A,B,C,D,E,F,G,H,I,J,K\}$。并使得自顶向下处于同一层次的代码块属于一个状态,如图 5 所示,也即模型 λ 中的状态集合 S,其中 $S=\{q_1,q_2,q_3,q_4,q_5\}$。通过对图 5 的再次扩展,可使其转换成以代码块为单位的图 6,也就是使得程序段的每条执行路径在每个状态 $q_i(1\leqslant i\leqslant 5)$ 都有输出。

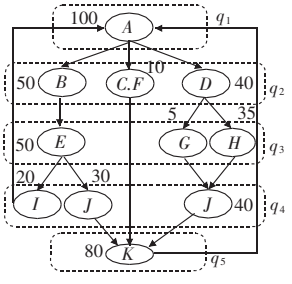

图 5　基于 HMM 的程序段图

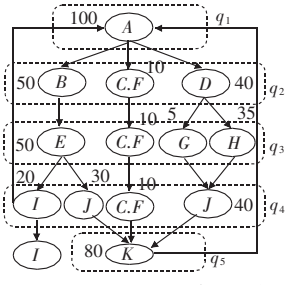

图 6　基于 HMM 的程序段图扩展

基于模型 λ 的概率估算只在循环入口代码块 A 的执行次数达到优化触发器的值 trigger 时才启动预测,本例为了方便计算设 trigger = 100。候选热块 candiBB 是基本块的计数器 count 大于阈值 baseCount 的块,一般 baseCount = trigger × 40%[8],可表示为 candiBB = $\{x|x\in \text{BBset}\cdot\ \text{count}(x)>\text{baseCount}\}$。BBset 指循环内的所有基本块,此处即为集合 V。由候选热块组成的执行路径称为候选热路径, candiHp = $\{A,x,y|x,y\in\text{candiBB}\cdot A\rightarrow x\wedge x\rightarrow y\wedge y\rightarrow A\}$。

至此,热路径的预测即可通过模型 λ 进行,图 6 中的路径 *ADHJK* 表示 HMM 的一个输出,也即观察序列 $O_i(1\leqslant i\leqslant M)$,$M$ 为循环内路径总数,并且是一条候选热路径。热路径的预测就是基于模型 λ 预测其中输出序列 O_i 出现概率 $P(O_i|\lambda)$ 最大的序列过程。

2　基于 HMM 的建模

经过分析,此时热路径的预测变成如下问题的求解过程。

已知:a)模型 $\lambda=(S,V,A,B,\pi)$;b)候选热路径为 $O_i=(o_1,o_2,\cdots,o_i,\cdots,o_5)$。其中 $o_i\in V$。

求解:候选热路径中 $P(O_i|\lambda)$ 值最大的路径,也即求解 O_i 中出现概率最大的观察序列。

其中:状态集合 $S=\{1,2,3,4,5\}$,$q_i(1\leqslant i\leqslant 5)$ 表示某一个状态;观察值集合为 $V=\{A,B,C,D,E,F,G,H,I,J,K\}$。状态转移矩阵 A 如下所示,经过扩展后,程序的执行可视为自上至下的执行过程,因此只存在 (q_1,q_2,q_3,q_4,q_5) 的状态转移序列,其他出现的概率均为 0。

$$A=\begin{Bmatrix}0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\\1&0&0&0&0\end{Bmatrix}$$

B 表示各代码块的执行概率分布,$B=\{b_j(v_k)\}$。其中:$b_j(v_k)=\text{count/trigger}$,如 $b_{q_2}(B)=40/100$,即为 q_2 状态下出现 B 的概率;π 是初始状态概率分布,$\pi=\{\pi_i\}$ 且 $\pi_i=P(q_1=i)$,本处很显然 $\pi_1=1,\pi_1=0(2\leqslant i\leqslant 5)$。

3　算法及复杂度分析

1)给定 λ 以及状态转移序列 $q=(q_1,q_2,q_3,\cdots,q_T)$,则观察序列 $O_i=(o_1,o_2,o_3,\cdots,o_T)$ 的出现概率为 $P(O_i|q,q,\lambda)=b_{q_1}(o_1)b_{q_2}(o_2)b_{q_3}(o_3),\cdots,b_{q_T}(o_T)$;给定 λ,则状态转移序列 $q=(q_1,q_2,q_3,\cdots,q_T)$ 的出现概率为 $P(q|\lambda)=\pi_{q_1}a_{q_1q_2}a_{q_2q_3},\cdots,a_{q_{T-1}q_T}$。因此 O_i 和 q 的联合概率 $P(O_i,q|\lambda)=P(O_i|q,\lambda)$[7]。

2)给定 λ,若考虑所有的状态转移序列,则 $P(O_i|\lambda)=\sum_q P(O,q|\lambda)=\sum_q P(O|q,\lambda)P(q|\lambda)$。

展开后得到

$$P(O_i|\lambda)=\sum_{q_1,q_2,\cdots,q_T}\pi_{q_1}b_{q_1}(o_1)a_{q_1q_2}b_{q_2}(o_2)a_{q_2q_3},a_{q_{T-1}q_T}b_{q_T}(o_T)$$

由于只存在 $q=(q_1,q_2,q_3,\cdots,q_T)$ 转移序列,其他情况的状态转移计算结果为 0,无须纳入计算,即 $\pi_1=1,\pi_i=0(2\leqslant i\leqslant 5)$。此时计算变成:

$$P(O_i|\lambda)=b_{q_1}(o_1)a_{q_1q_2}b_{q_2}(o_2)a_{q_2q_3}\cdots a_{q_{T-1}q_T}b_{q_T}(o_T)$$

整理后即为

$$P(O_i|\lambda)=b_{q_1}(o_1)b_{q_2}(o_2)\cdots b_{q_T}(o_T)a_{q_1q_2}a_{q_2q_3}\cdots a_{q_{T-1}q_T}$$

则计算 $P(O_i|\lambda)$ 共需要进行 $2(T-1)$ 次乘法,整个算法需要计算 $N\times 2(T-1)$ 次乘法(N 为候选热路径数)。

本例对于每条候选热路径只需计算 8 次乘法运算,候选热路径数 3 条,则总的计算次数为 24 次。通过计算获得候选热路径的最大可能执行概率,选取该候选热路径作为热路径。

在本例中

$$P(O_i|\lambda)=b_{q_1}(o_1)a_{q_1q_2}b_{q_2}(o_2)a_{q_2q_3}b_{q_3}(o_3)a_{q_3q_4}b_{q_4}(o_4)a_{q_4q_5}b_{q_5}(o_5)$$

候选热路径有 $\{ABEII,ABEJK,ADHJK\}$。

通过计算可以得到 $P(ABEII|\lambda)=100/100\times 50/100\times 50/100\times 20/100\times 20/100=0.01$;同理可计算得 $P(ABEJK|\lambda)=0.06$;$P(ADHJK|\lambda)=0.0448$(由于 $b_j(v_k)$ 的基数都是 trigger,计算时可直接使用基本块的执行次数 count 进行计算,本处为了更直观,仍使用比值计算)。

$P(ABEJK|\lambda)>P(ADHJK|\lambda)>P(ABEII|\lambda)$,那么最热的热路径为 $ABEJK$。

若采用传统的热路径执行次数判断法,则 $C\{ABEII\}=20$;$C\{ABEJK\}=30$;$C\{ADHJK\}=35$。

$C\{ADHJK\}>C\{ABEJK\}>C\{ABEII\}$,因为路径 $ADHJK$ 的执行次数最多,所以最热的路径为 $ADHJK$。

由以上复杂度分析可知,尽管思路完全不同,基于隐马尔可夫模型的计算,其计算过程并不复杂,而且实现非常简单。

4　实验结果与分析

为了评价热路径预测算法的有效性,下面分别给出了预测命中率、噪声比、预测延迟的定义。假设启动预测后,候选热路径 p_i 的执行次数为 $f(p_i)(1\leqslant i\leqslant N)$,则所有候选热路径的执行次数约为 $\sum_{i=1}^{N}f(p_i)+N\times\text{trigger}$,热路径的预测命中率为

$$\text{hitRate}(p)=f(p)/(\sum_{i=1}^{N}f(p_i)+N\times\text{trigger})$$

其中:p 表示预测出的热路径。

噪声比是指将非热路径作为热路径的概率,可表示为

$$\text{noise}=(\sum_{i=1}^{N}f(p_i)-f(p))/(\sum_{i=1}^{N}f(p_i)+N\times\text{trigger})$$

其中:N 为候选热路径数。

预测延迟 preDelay 是指启动热路径预测前所有路径执行的次数与程序段总的执行次数的比值,可表示为

$$\text{perDelay}=\sum_{i=1}^{M}f(p_i(t))/T$$

其中:$f(p_i(t))$ 表示 t 时刻启动预测前路径 i 的执行次数;M 表示路径总数;T 是程序段总的执行次数。预测延迟越大,则热路径的丢失机会成本[3]越高,同时运行时的负载也将越高。

预测的重点就是如何提高命中率 hitRate(p),并保持低的预测延迟和噪声比 noise。本算法的实现在 Skyeye[9] 上进行。Skyeye 是国内一款开源的虚拟机,目标是模拟常见的嵌入式计算机系统,可在 SkyEye 上运行 Linux、μCLinux 等多种嵌入式操作系统和各种系统软件。Skyeye 的动态二进制翻译模块(DBCT)对运行在其上的操作系统进行加速,该模块的实现基于 QEMU。

下面是在 Skyeye 的 Linux 系统上运行系统性能测试工具 nbench benchmark suite 的结果,并针对上面介绍的三个指标进行分析。表 1 给出了 benchmark 程序中的绝对热路径数(A-Hotpaths)与相对热路径数(R-Hotpaths)占路径总数比例的情况。其中绝对热路径是指在循环内执行次数远远多于其他分支的路径;相对热路径是指循环内平均执行的路径。大多数情况下循环内绝对热路径要比相对热路径多,可以发现其中 assignment 和 huffman 程序中相对热路径的数目较多。同时,从表 1 可以了解到 huffman 程序绝对路径与相对路径数相似,assignment 程序的相对热路径比绝对热路径稍多,idea 程序两者相差较大。这三个程序具有一定的代表性,因此下面就采用这三个程序对本文所提出的算法性能进行分析。

表 1　路径情况

benchmark	paths	A-Hotpaths/%	R-Hotpaths/%
numeric sort	700	2.429	1.571
string sort	750	2.800	1.600
bitfield	713	2.104	1.543
fp emulation	827	1.693	0.967
fourier	697	2.296	1.578
assignment	762	2.887	3.281
idea	763	2.752	1.180
huffman	739	2.030	1.894

图 7 给出了所提出算法下 huffman、idea 和 assignment 程序的预测延迟与命中率的关系图。由图可以看出,由于 huffman 的路径数较少,在盲测时的命中率稍高;同时由于 idea 的绝对热路径与相对热路径占用比例相差较大,idea 的命中率总体呈上升趋势。

图 8 给出了上述三个程序的噪声比与预测延迟的关系图。由图可以看出,与图 7 相对应,idea 的噪声比明显要小;同时由于 assignment 的相对路径较多,整体噪声比较大。结合图 7 和 8 可以发现,若在预测延迟的 20% 处启动预测,将在hitRate与noise 之间达到一个较好的平衡。

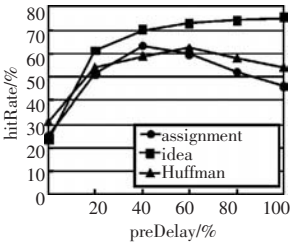

图 7　命中率与预测延迟的关系图

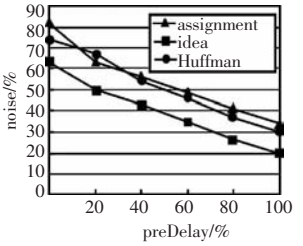

图 8　噪声比与预测延迟关系图

预测命中率的计算相对比较复杂,并且预测的目的是通过提高命中率来减少热路径重复翻译产生的开销,从而提高程序的运行效率,减少程序的运行时间,因此本文直接针对程序的运行时间进行分析,运行时间越少自然命中率越高,算法更优越。图 9 给出了所提出的算法与基于路径数预测方法以及本地的运行时间比较。从图中可以发现,利用基于 HMM 模型的热路径算法程序运行时间有不同程度的减少,其中相对热路径较多的 assignment 等程序减少明显。由于显示的需要,本文以代码块较少的简单程序作为实例进行分析。如果程序更加复杂,如大型系统程序,该算法的效果将更加显著。当然,如果直接在本地运行,速度差别仍然较大。以上实验结果和分析表明,本算法在保持预测延迟不明显增加的情况下可以提高热路径的命中率,从而提升软件的总体运行效率。

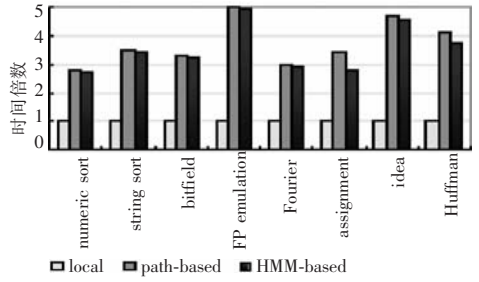

图 9　运行时间比较

5　结束语

热路径是程序在某一个时间频繁执行的代码块的集合,识别并优化热路径能显著改善程序在动态二进制翻译器上的运行性能。热路径预测的目标是如何利用有限的已有运行信息预测出未来将要频繁执行的路径,预测算法要尽量简单,容易实现,并且只有预测带来的效益大于开销时,预测才有意义。本文运用隐马尔可夫模型提出了改进的基于块的热路径预测算法,该算法的优越性主要体现在热路径的预测基于预测模型,尤其当候选热路径执行次数近似时,热路径的选择更具有效性。该算法实现简单,提高了热路径的命中率,降低了热路径在 cache 中替入替出的消耗,从而能够提高软件在虚拟机中的运行速率。下一步工作将针对热路径的阶段变化情况进行研究。

参考文献:

[1] SCOTT K, KUMAR N. Overhead reduction techniques for software dynamic translation [C]//Proc of the 18th International Parallel and Distributed Processing Symposium. 2004.

[2] EBCIOGLU K. Dynamic binary translation and optimization [J]. IEEE Trans on Computers, 2001, 50(6): 529-548.

[3] DUESTERWALD E, BALA V. Software profiling for hot path prediction: less is more [C]//Proc of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press, 2000: 202-211.

[4] BALL T, MATAGA P, SAGIY M. Edge profiling versus path profiling: the showdown [C]//Proc of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York: ACM Press, 1998: 134-148.

[5] BALL T, LARUS J R. Efficient path profiling [C]//Proc of the 29th Annual ACM/IEEE International Symposium on Microarchitecture. Washington DC: IEEE Computer Society, 1996: 46-57.

[6] 史辉辉,管海兵,梁阿磊. 动态二进制翻译中热路径优化的软件实现[J]. 计算机工程, 2007, 33(23): 78-83.

[7] RABINER L R. A tutorial on hidden Markov models and selected applications in speech recognition [J]. Proceedings of IEEE, 1989, 77(2): 257-286.

[8] UNG D, CIFUENTES C. Optimizing hot paths in a dynamic binary tanslalor [C]//Proc of ACM SIGARCH Computer Architecture News. New York: ACM Press, 2001: 55-65.

[9] CHEN Yu, REN Jie, ZHU Hui, *et al*. Dynamic binary translation and optimization in a whole-system emulator-skyEye [C]//Proc of International Conference on Parallel Processing. Washington DC: IEEE Computer Society, 2006: 327-336.

(上接第 2464 页)

[4] CHEN Sheng, HONG Xia, LUK B L, *et al*. Orthogonal-least-squares regression: a unified approach for data modeling [J]. Neurocomputing, 2009, 72(10-12): 2670-2681.

[5] CHEN S, COWAN C F N, GRANT P M. Orthogonal least squares learning algorithm for radial basis function networks [J]. IEEE Trans on Neural Networks, 1991, 2(2): 302-309.

[6] CHO K B, WANG B H. Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction [J]. Fuzzy Sets and Systems, 1996, 83(3): 325-339.

[7] RIVALS I, PERSONNAZ L. A recursive algorithm based on the extended Kalman filter for the training of feedforward neural models [J]. Neurocomputing, 1998, 20(1): 279-294.

[8] SIMON D. Training radial basis neural networks with the extended Kalman filter [J]. Neurocomputing, 2002, 48(1): 455-475.

[9] WU Shi-qian, ER M J, GAO Yang. A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks [J]. IEEE Trans on Fuzzy Systems, 2001, 9(4): 578-594.

[10] WU Shi-qian, ER M J. Dynamic fuzzy neural networks: a novel approach to function approximation [J]. IEEE Trans on Systems, Man and Part B-Cybernetics, 2000, 30(2): 358-364.

[11] RIGATOS G, ZHANG Q. Fuzzy model validation using the local statistical approach [J]. Fuzzy Sets and Systems, 2009, 160(7): 882-904.