doi:10.3772/j.issn.1002-0470.2015.01.005

基于多核平台无关属性的程序并行度分析工具①

熬 冉②*** 谭光明* 陈明宇*

(*中国科学院计算技术研究所高性能计算机研究中心 北京 100190)

(**中国科学院研究生院 北京 100049)

摘 要 为了充分利用计算机多核平台的并行能力,研究了针对特定多核平台的程序并行度分析和优化的方法,提出了一个基于平台无关属性提取的并行度分析工具 ParaInsight。用此工具可分析程序中不同区域的可并行性,同时能够根据小规模输入集下的信息预测这些属性在大规模输入集下的值。通过使用支持向量机(SVM)构建的预测模型,对大输入集下的程序属性值进行预测。选取多线程程序测试包 Parsec 中的两个程序进行了实验,实验结果表明,通过核函数选择、训练参数调优以及输入变量筛选等方法,用 ParaInsight 可以有效地预测绝大多数并行区域及其不同的属性值。

关键词 并行度分析, 动态剖析, 输入感知, SVM 预测

0 引 言

由于受到工艺与功耗的限制,计算机正朝着众核的方向发展,多核芯片已成为目前计算机的主流配置。为了充分利用多核平台的并行能力,需要对程序内不同并行区域的可并行度进行分析,以挖掘程序中的线程级并行性能。在开发多线程程序的过程中,程序员常常借助性能分析工具来帮助寻找程序中的性能瓶颈,进而优化程序。性能分析工具主要有两类:一类属结果导向型,即利用硬件提供的性能计数器统计程序在硬件上产生的行为结果来反映程序的特性,例如通过测量各级缓存缺失率了解程序局部性特征等,这种方法的问题是硬件事件无法与程序结构一一对应;另一类是面向程序自身,通过剖析的方法获取程序结构本身的信息,例如函数调用次数等信息。多线程程序的可扩展性取决于两个方面:一个是内因,即程序自身的可并行度;另一个是外因,即并行表达依赖的并行编程模型以及并行执行环境。其中内因是基础,充分了解程序自身的可并行度,对选择并行编程模型以及执行环境参数的选择都有重要的意义。本研究从内因入手,通过分析程序自身影响可扩展性的因素,提出了一种基于平台无关属性提取的线程级并行度区域分析工具,并对其进行了预测有效性验证。

1 对输入集的要求

在进行程序分析时,由于真实程序本身的执行时间比较长,程序分析方法会导致时空开销成指数增长,出于对分析效率和分析开销的考虑,通常使用适当较小的输入集来代替真实输入集。但这要求程序在两种输入集下有相似的行为特征。如果程序在两种输入集下的行为特征不相同,那么基于小规模输入集进行的分析工作就会失效,不能做出正确的优化判断。

我们用一个例子来说明这种由于输入集不同所导致的行为特征差异。针对多线程程序测试包 Par-

① 973 计划(2012CB316502,2011CB302502),863 计划(2009AA01A129)和国家自然科学基金(61272134, 31327901, 91430218, 60921002, 60925009, 61472395)资助项目。

② 女,1982 年生,博士生;研究方向:计算机系统结构;联系人,E-mail: aoran@ncic.ac.cn

(收稿日期:2014-03-24)

sec 中的 streamcluster 程序,使用性能计数器在大小两个输入集下观测其 L3 缓存缺失率以及可扩展性,测试平台为 Intel 16 核机器。如图 1(a)所示,程序在不同的输入集下呈现出不同的可扩展性,即在小输入集下可扩展性良好,而在大输入集下该程序呈现出较差的可扩展性,使用 8 核以上进行并行就不再有显著的加速比。另外,程序的局部性特征在不同的输入集下也不同。如图 1(b)所示,在小输入集下,L3 缓存缺失率在 2% 左右,显示良好的局部性特征,但是在大输入集下,L3 缓存缺失率在 13% 以上。通过这个例子可以看出,为了研究程序的可并行度,需要找出程序自身影响可并行性的因素,同时研究输入集对这些因素的影响。

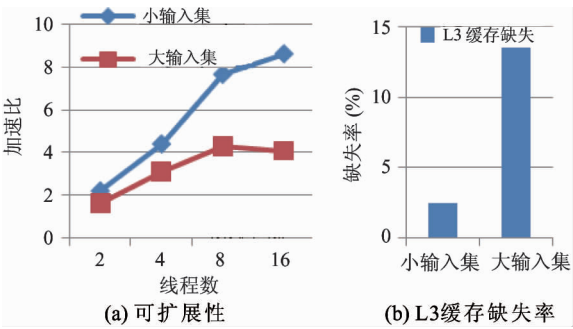


图 1　Streamcluster 在不同输入集下的行为特征

2　影响并行度分析的属性

程序的不同并行区域通常存在不同的最优并行度,这就需要通过对每个并行区域单独进行分析确定该区域自身的可并行度,以此决定物理并行资源的具体分配。线程并行的对象可以是基本块、函数、循环体,我们的工具可以针对不同粒度进行分析。这里以下面的循环并行化为例,说明在分析该区域的并行度时,会对并行效果产生影响的属性。这些属性包括工作量百分比、并行上限、并行收益等。

2.1　工作量百分比

传统程序分析的一个重要功能就是找出程序热点,通常都是统计各个函数执行所占的时间百分比。根据阿姆达定律,程序可并行比例 p 越大,潜在并行收益越大。为了提高分析效率,我们的工具也会给出程序的热点区域信息,统计各个循环所占的百分比。本文作者开发的分析工具能直接记录各循环执行时间,但是这样的结果是依赖具体执行平台的。为了反映程序自身的工作量特征,本文进行了改进,以指令条数代表的工作量代替执行时间:每个循环的工作总量 Work(Loop) = Work (i) × n × N, 其中 N 是指该循环作为一个并行区域被执行的次数, n 是该并行区域一次执行时包含的迭代次数。由这个式子可知,判断循环(Loop)是否适合并行时,不仅要看该循环总的工作时间,还要看这个循环被调用一次的平均执行时间,这会影响到在具体的并行编程模型和执行平台上是否能够将该循环并行执行。

2.2　并行上限

当要对一个循环进行并行时,还需要看这个循环的迭代次数上限,这决定了该循环的并行上限,当迭代次数上限 n 小于处理器核数时,无法充分利用多核的并行性。

2.3　并行收益

当迭代次数上限 n 大于等于核数时,可以使每个核都分到任务,为了便于分析其它影响并行收益的因素,这里假设循环迭代上限等于核数。由于迭代间可能存在数据相关,并行收益会受具体的相关性影响。根据迭代间的读写关系可以将相关性分为两类:数据依赖和数据共享。数据依赖的分析见文献[1]。本文侧重数据共享的分析。由于多核平台共享缓存(cache)的影响,根据迭代之间数据访问的不同特性,并行收益会受到不同的影响。

2.3.1　串行执行

当循环 L 的迭代串行执行时,该循环的执行时间 Ts = (∑(Inst(j) × latency)) × n, 一般的指令延迟(latency)与具体的体系结构有关[2]。这里需要关注的是访存指令,由于 cache 的影响,不是每条访存指令都会访问内存,这里为了说明迭代并行后由于多核间共享数据对 cache 访问的影响,将每次迭代作为一个整体,做两条假设:

(1) 不考虑冷失效(cold miss),只考虑容量失效(capacity miss),即当迭代的工作集小于等于最后一级缓存(Last Level Cache, LLC)容量时,假设预取和空间局部性使得数据全部能在 LLC 中命中;

(2) 数据重用距离等于 LLC 容量,即当迭代的

工作集大于 LLC 容量时,发生缓存缺失(cache miss)。

根据迭代间数据共享度及关系的不同,迭代并行后对访存产生的影响也不同。

2.3.2　并行执行时迭代间数据共享关系

迭代间数据共享关系主要有三种,即无共享数据、共享读及共享读写(真依赖)。并行后的执行时间 $Tp = \max(\sum \mathrm{Inst}(j) \times \mathrm{latency} + \mathrm{Overhead})$。其中 Overhead 是开销,由三部分组成:一是并行开销(+),二是依赖导致的通讯开销(+),三是共享收益(+/-)。其中前两项都会导致时间增加,而第三项由于共享带来的影响关系就比较复杂,可能有正有负,与并行前迭代工作集大小、迭代间共享程度、并行执行的核的个数都有关系。

这里进行简单的上下界分析。

(1) 迭代间无数据共享

当迭代间无共享数据时,迭代并行后会争用最后一级缓存(LLC),当并行执行的核的个数大于(LLC 容量/单个迭代工作集大小)时,并行会导致最后一级缓存失效(LLCmiss),产生访存代价(penalty);如果串行执行时迭代本身的工作集就超出了 LLC 容量,那么这样的并行更会加剧 LLC 冲突,增加内存带宽压力。

(2) 迭代间有数据共享

当迭代间有共享数据时,一种情况是迭代本身工作集小于 LLC 容量,共享的数据比例越大,就能同时允许更多的核并行执行而不会给内存带宽造成更大的压力;另一种理想的情况是,虽然每个迭代的工作集大于 LLC 容量,但是迭代间共享度很大,这样当并行时,一个核放入 LLC 的数据可以被其它的核利用,从而减少了原本迭代在串行执行时的本应产生的最后一级缓存失效。上述共享数据在不同工作集大小时的影响见表 1。

表 1　数据共享的影响

迭代串行执行	迭代并行执行			
	数据共享关系	对 LLC 影响	对带宽影响	收益
迭代工作集小于 LLC 容量	无共享	加剧竞争	加剧竞争	~
	有共享	较上一项缓和	较上一项缓和	~
迭代工作集大于 LLC 容量	无共享	恶化	恶化	~
	有共享	有改善空间	有改善空间	局部性改善收益较大

('~'代表收益不确定,受其它并行因素影响)

3　并行区域分析工具

3.1　ParaInsight

为了研究程序自身的可并行性,需要抽取程序中平台无关的特性。本文基于动态 profiling 的并行区域分析设计并实现了ParaInsight框架。其核心是基于动态实例关系的多粒度对象调用图,以基本块为基本单位,由其组成的函数和循环体的每一个实例都作为图中的节点。通过底层虚拟机(low level virtual machine,LLVM)中间代码插桩得到程序访存序列,由于 LLVM 的中间代码有一套类似汇编的指令集,这样既保留了程序的结构信息,又含有底层指令信息,因此很容易将访存信息与程序结构对应起来。

3.2　热点区域描述改进

文献[1]详细描述了 ParaInsight 的框架以及程序特性的分析。其中热点区域是通过记录程序执行时间计算的,不算是程序的平台无关特性。因此,本文在改进的 ParaInsight 中,对热点区域的衡量用工作量代替,因为每个基本块的指令条数是固定的,一旦一个基本块执行,其中的每条指令都会执行。在程序静态分析阶段,可以将每一个基本块的指令分布作为基本块的属性值记录在文件中。在实际执行中每个区域的工作量受调用次数影响,在动态实例调用图上,通过自下而上遍历,很容易算出每个区域的工作量。这个数值是程序本身固有的属性,而每个区域具体的执行时间则与底层的体系结构以及区域之间的数据相关性有关。

3.3　区域间共享数据分析

从表 1 中可以看出,迭代并行后共享数据对局部性的影响与迭代本身工作集大小以及平台属性有关系,尤其是出现最后一行情形时,在程序优化过程中考虑这个因素对程序的性能提升收益非常大,而这一点要结合具体的平台参数进行计算。通过分析能够给出的程序本身的信息主要有循环工作集大

小、迭代的工作集大小和数据共享度。数据共享度可以用各迭代工作集之和与循环工作集的比值来衡量。

4 程序属性值预测

4.1 输入集对程序行为的影响

通过引言中的例子我们可以看到,程序在不同的输入集下会表现出完全不同的行为特征,因此在一个输入下得到的某些结论并不适用于另一个输入。文献[3]对函数在不同输入集下所占时间百分比的研究也同样印证了这一点。在本研究开发的并行分析工具中,时空开销最大的部分是对访存踪迹以及对象实例调用的记录。通过前面的介绍可以得知,本研究开发的工具的目的是提取程序平台无关的属性值,而引言例子中程序的扩展性、缓存失效率等是程序在不同外因下的表现。程序本身的平台无关的属性值与输入之间存在数值对应关系,通过找到这样输入敏感的属性,我们可以预测这些属性在大输入集下的数值,而不必通过实际运行得到这些数值,进而使用这些数值对程序行为进行分析。

在并行区域工作量占比的分析中,实例调用图中随着输入改变的参数只有循环的迭代次数 n,而每个基本块内的指令条数是固定的,因此,知道了大输入集下的循环迭代次数 n,就可以结合预存的基本块信息计算出各个区域的工作量占比。通过预测得到各循环在大输入集下的迭代次数 n 之后,对工作量的计算就不需要遍历全部实例调用图,只需要用循环一次迭代的工作量与预测的 n 值相乘即可。这里假设循环迭代间的行为具有相似性。除此之外,这个数值也代表了这个区域的并行度上限。同样在共享数据的分析中,可以预测大输入集下的工作集大小。

4.2 程序属性值预测

当两组数据间具有统计意义上的相关性时,常用回归分析的方法得到它们之间的数值关系。本文以预测循环迭代次数和工作集大小这两个属性值为例介绍预测的方法并检验这种预测的有效性。这里的自变量就是程序的输入,因变量是循环迭代上限 n 以及循环的工作集 w。每一个区域的 n 和 w 都需要有与输入的映射关系。

$$\left.\begin{aligned} N_i &= f(X, B_i) \\ W_i &= f(X, B_i) \end{aligned}\right\} \quad (i \in \text{并行区域个数})$$

上式中的 X 是程序的输入参数向量,对于每一个区域来说,不是所有的输入参数都会影响到该区域。因此,为了提高模型预测的准确率,要先进行参数筛选,只保留相关系数最高的参数。回归分析主要分为线性回归和非线性回归两大类,在传统的线性回归中主要采用最小二乘法建立线性模型,对于非线性模型,主要方法有回归树和模型树,譬如文献[4]就采用线性回归和回归树的方式对每个基本块执行次数进行了预测。

4.3 基于 SVM 的回归预测

除了上述传统的回归分析方法外,随着机器学习理论和方法的发展,一些分类技术如后向传播神经网络(back propagation neural network, BPNN)、k 最近邻(k-nearest neighbor, KNN)、支持向量机(support vector machine, SVM)也可以用于回归预测。本文采用 SVM 来进行预测。SVM 的原理是寻找最大边缘超平面,当在 N 维空间中只有最大边缘超曲面时,SVM 通过向更高维度进行映射的方式,寻找在更高维度对应的最大边缘超平面。SVM 的核心理论是在高维空间进行的点积运算可以用核函数在原始输入空间上的计算代替,因此 SVM 的核心是对核函数的选择。

5 预测有效性验证

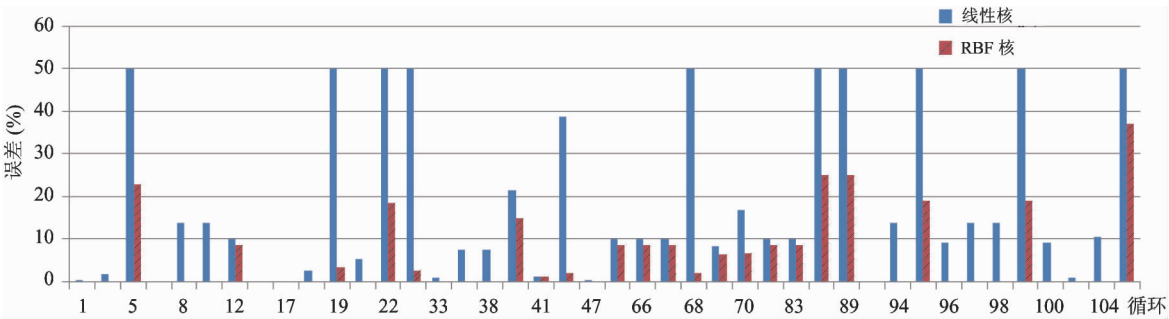
5.1 实验设置

ParaInsight 执行环境配置见文献[1],本文的预测部分使用 Matlab2010b,安装 SVM 工具箱 LIBSVM[5]。目标程序为 Parsec 中的两个程序 Splash2x. fmm 和 streamcluster,其中 fmm 受单一输入参数影响,streamcluster 受多个输入参数影响。对预测方法的验证主要是预测循环迭代次数 n,最后给出一个例子证明对其它属性如工作集大小的预测同样有效。

5.2 核函数对预测结果的影响

输入对于不同的循环会有不同的影响,因此需

要针对每个循环建立预测模型。输入与循环迭代次数间最简单的映射关系是线性相关,但是由于循环深度和输入参数间耦合的影响,这种相关性有时是非线性的。图 2 中的灰色柱状代表使用简单的线性相关对 fmm 程序中各循环建立预测模型。从图中可以看出,使用这种预测模型,某些循环预测误差高于 50%。对于这些循环,线性相关的预测模型不能正确反映输入与循环迭代次数 n 之间的关系。除了线性相关,LIBSVM还提供了建立非线性映射关系的三种核函数,即多项式核、高斯径向基函数(RBF)核和 S 形核。图 2 中的黑色代表使用常用的 RBF 核函数重新建立预测模型。可以看出,对不适用线性相关的循环,使用非线性模型可以大幅降低预测误差,使具有预测意义。



对于预测误差大于 50% 的数值进行截取,红色非线性模型仅保留对线性模型有改进的值

图 2　核函数选择对预测效果的影响

5.3　训练过程中参数的选择对预测结果的影响

对于图 2 给出的预测结果,无论是选择线性模型还是非线性模型,在建立预测模型的训练过程中,都没有进行参数寻优,而是随机取值。在 LIBSVM 的模型训练函数中,有多个可以调节的参数,会对预测模型的准确性产生影响。本文不对具体的参数寻优过程进行探讨,仅通过对其中两个参数的优化展示参数寻优对最后预测结果的影响。在 LIBSVM 的训练函数 svmtrain 的参数中,对模型精度影响较大的参数是其中的'-c'、'-g'参数,使用文献[6]提供的扩展 LIBSVM 工具箱,对这两个参数进行优化。通过参数优化后,在图 3 中可以看出,对于大多数循环,预测误差都可以得到降低。因此为了更好地对目标结果进行预测,在建立预测模型的过程中,参数优化是一项非常有必要进行的工作。

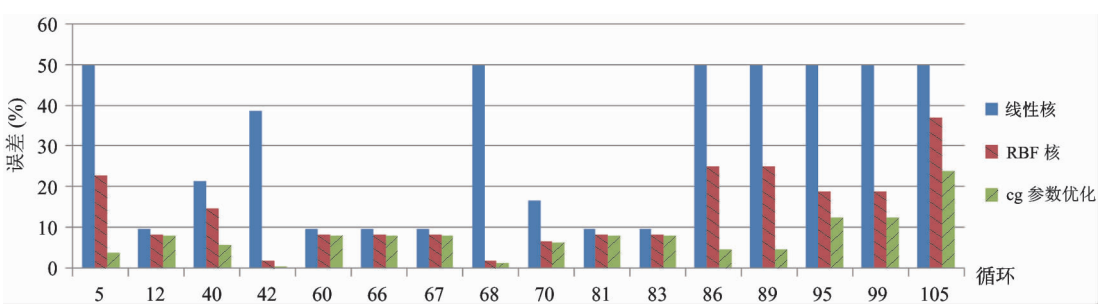


图 3　训练过程中参数选择对预测模型准确率的影响

5.4　输入参数筛选

程序的输入通常提供多个参数,这些参数对于程序整体来说都会产生影响,但是具体到不同的循环以及不同的行为特征,并不是每个参数都有相关性。使用 SVM 建立预测模型的过程中,过多的不相关自变量或相关性较低的自变量一方面会影响模型的复杂度,更重要的是会影响模型的预测能力。为了建立更合理的预测模型,需要对自变量进行降维处理,即筛选出对因变量影响最大的自变量。这里采用平均影响值(mean impact value,MIV)来评估变

量相关性。文献[6]提供的扩展 LIBSVM 工具箱对 MIV 的计算方法是:使用原始训练样本建立初始预测模型,然后对于每一个自变量,将训练样本中的数据分别增减 10% 形成两个新的样本,使用初始预测模型对这两个新的样本进行预测得到两组预测结果,这两组预测结果的差值反映了变动该自变量对预测结果产生的影响。每一个自变量都可以得到一个 MIV,根据 MIV 的大小即可选择相关性较大的那些自变量。在我们的实验中仅筛选掉 MIV 为零的自变量,即对循环迭代次数影响完全不相关的参数,需要注意的是,不同的循环筛选掉的参数是不一样的。在图 4 中可以看出,对于循环 31、32 通过筛选参数后可以得到更加准确的预测模型,预测误差大幅下降。

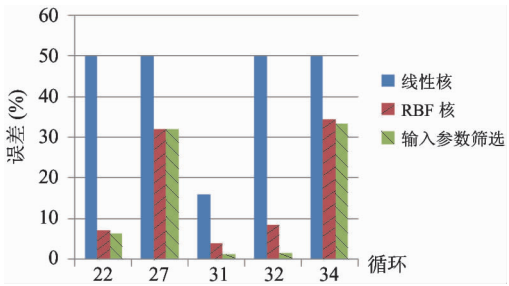


图 4　输入参数筛选(仅选择有变化的循环)

5.5　对不同程序特征值的预测

上述实验都是以预测循环迭代次数为例介绍预测方法,除了循环迭代次数之外,还可以针对其它的程序属性建立预测模型。图 5 给出的是对程序 fmm 不同循环的工作集大小进行预测的结果。

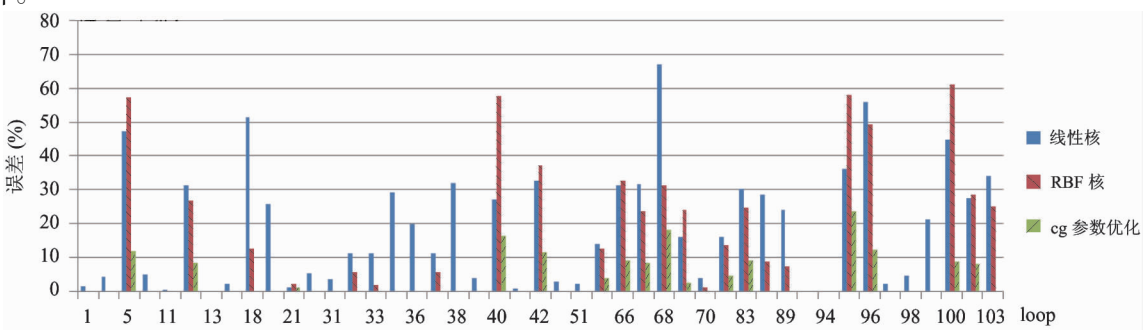


图 5　对不同程序特征值进行预测的误差率

6　结　论

为了充分利用多核的并行性和提升并行性能,需要对程序内不同并行区域的可并行度进行分析。本研究对前期开发的并行度分析工具 ParaInsight 一方面进行了改进,对于热点判断不使用执行时间,消除平台相关因素,改进的方式由固定的基本块信息与变化的迭代次数 n 联合计算确定,同时在数据相关对并行性影响的分析中,除了之前的数据依赖,又添加了数据共享的影响;另一方面,考虑分析工具的开销,为了满足在大输入集下的分析,采用 SVM 构建预测模型,对大输入集下的程序属性值进行预测。实验结果表明,通过核函数选择、训练参数调优以及输入变量筛选等方法,可以有效地预测绝大多数并行区域以及其不同的属性值。

参考文献

[1] Ao R, Tan G M, Chen M Y. ParaInsight: An assistant for quantitatively analyzing multi-granularity parallel region. In: Proceedings of the 15th IEEE International Conference on High Performance Computing and Communications, Zhangjiajie, China, 2013. 698-707

[2] Granlund T. Instruction latencies and throughput for AMD and Intel x86 processors. 2012-02-13

[3] Hsu W C, Chen H, Yew P C. On the predictability of program behavior using differentInput data sets. In: Proceedings of the Sixth Workshop on Interaction between Compilers and Computer Architectures, Washington DC, USA, 2002. 45-53

[4] Tian K, Jiang Y L, Zhang E Z, et al. An input-centric paradigm for program dynamic optimizations. In: Proceedings of the 25th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, Reno/Tahoe, USA, 2010. 125-139

[5] Chang C C, Lin C J. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2011. http://www.csie.ntu.edu.tw/~cjlin/libsvm/: IEEE, 2011

[6] Li Y(Faruto). LIBSVM-FarutoUltimate: a toolbox with implements for support vector machines based on libsvm. 2011. http://www.matlabsky.com

A tool for program parallelism analysis based on the program properties independent of multi-core platforms

Ao Ran[*,**], Tan Guangming[*], Chen Mingyu[*]

([*]High Performance Computer Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]University of Chinese Academy of Sciences, Beijing 100049)

Abstract

To make full use of a multi-core computing platform's paralleling ability, the study on analysis and optimization of a specific multi-core program was conducted, and a tool for program parallelism analysis based on the program properties independent of multi-core platforms, called the ParaInsight, was proposed. The tool can be used to analyze the parallelism of different areas in a program, and to predict the values of the program properties independent of a multi-core platform under a large input set according to the information under a small input set. The values of program properties under a large input set were predicted by using a prediction model constructed by using a support vector machine. Two programs of Splash2x.fmm and streamcluster of the Parsec were used to perform the experiment on the predicting effectiveness, and the results show that the ParaInsignt can effectively predict most parallel areas and their different property values through core function selection, optimization of training parameters and screening input variables.

Key words: parallelism analysis, dynamic profiling, input-aware, SVM-prediction