第 36 卷 第 7 期 计 算 机 工 程 2010 年 4 月
Vol.36 No.7 Computer Engineering April 2010
·开发研究与设计技术· 文章编号：1000—3428(2010)07—0249—03 文献标识码：A 中图分类号：TP391

基于超块的激进执行模型可预测性分析

赵灿明[1]，安 虹[1,2]，任永青[1]，从 明[1]
(1. 中国科学与技术大学计算机科学与技术系，合肥 230027；2. 中国科学院计算机体系结构重点实验室，北京 100080)

摘 要：分析基于超块的激进执行模型中超块级预测可行性，给出满足超块级预测的预测器设计方案。对不同应用深度预测可行性高低、期望预测深度及其影响因素等进行论证。实验结果表明，大部分应用具有较高的期望预测深度，适合激进执行，但不同的应用期望深度相差较大。

关键词：激进执行模型；超块级预测；期望预测深度

Predictability Analysis of Aggressive Execution Model Based on Hyperblock

ZHAO Can-ming[1], AN Hong[1,2], REN Yong-qing[1], CONG Ming[1]
(1. Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027;
2. Key Laboratory of Computer System and Architecture, Chinese Academy of Sciences, Beijing 100080)

【Abstract】This paper analyses the feasibility of the hyperblock-level prediction in hyperblock-based aggressive execution model, puts forward a multi-exit predictor which adapts to hyperblock-based execution model. It demonstrates the feasibility and prediction-depth of hyperblock-level prediction based on this model. Experimental results show that a lot of applications have a higher expected prediction-depth, suitable for aggressive execution, but they have great difference in different applications.

【Key words】aggressive execution model; hyperblock-level prediction; expected prediction-depth

1 概述

随着大规模集成电路技术的发展，可用的片上资源越来越多。如何把这些晶体管资源有效利用起来，进一步挖掘应用本身的并行性及加速程序执行成为体系结构研究亟需解决的问题[1]。

基于超块的执行模型，可以很好地将硬件资源转化为计算性能。通过编译技术，把原来程序的基本指令块通过一定方式有机组合在一起，成为固定或可变大小的指令块，叫做超块。在近年出现的许多体系结构中，如 TRIPS 和 Multiscalar 等，超块被作为程序执行过程中取指和提交的基本单位，这种执行模型就叫做基于超块的执行模型[2]。

超块作为取指和提交的基本单位，能获得比传统超标量处理器更大的取指和发射窗口，加速了程序的执行。但它并不能完全解决资源的利用问题：一方面更多的可用资源要求更大超块的出现；另一方面受应用本身以及编译技术的制约，难以构造出更大的有效超块。在这种情况下，基于超块的激进执行模型就成为一个较好的选择。

基于超块的激进执行模型是指利用超块级预测技术推测执行程序执行路径上的后续块。如果对后续块的预测准确率较高，执行路径上很多的后续块将被提前取指、发射和执行，应用并行性得以提高。可见，激进的超块执行模型是否可行，首先取决于能否进行有效的超块级预测，因此，对超块级预测技术的研究十分必要。

本文对基于超块的激进执行模型中超块级预测可行性进行分析，给出满足超块级预测的预测器设计方案。

2 超块级预测可行性分析

分支预测的本质都是基于历史的信息预测后续信息，其可行性取决于分支执行历史再现的可能性。如果程序中存在大量的循环，一定的分支指令的跳转行为和目标地址将在不久的将来重复再现，这样，当执行到某条分支指令时，记录该分支的跳转行为和分支目标后，下一次循环到此时利用分支指令地址和记录信息就能正确获取后续指令地址而无须等待分支指令结果计算出来，这就是分支预测的原理。但是，像 if-else 或迭代次数很少的 for(while)循环，分支行为的再现性很低，很难通过分支历史预测分支行为及转移的目标地址，这样的程序比较难以预测。

超块是在传统编译器产生的控制流图的基础上产生的。为了产生超块，编译器一方面利用贪心策略把一些基本块结合在一起，另一方面利用较少次数的循环展开或者断言策略，把基本块间的控制依赖关系转换成数据依赖关系，最后结合成超块。这样，大的循环迭代在新的控制流图中得以保留，而 if-else 或者迭代次数较少的循环不再存在，减少了难以预测的分支的存在[3-4]。

基金项目：国家自然科学基金资助项目(60633040, 60736012)；国家“973”计划基金资助项目(2005CB321601)；国家“863”计划基金资助项目(2006AA01A102, 2009AA01Z106)；教育部-英特尔信息技术专项科研基金资助项目
作者简介：赵灿明(1983－)，男，硕士研究生，主研方向：微处理器体系结构；安 虹，副教授、博士；任永青、从 明，博士研究生
收稿日期：2009-12-12 **E-mail：**zhaocm0806@gmail.com

由以上的分析可知，在基于超块的执行模型中，分支预测所依赖的历史再现性没有被改变，甚至在某些编译策略中一定程度上得到提高，因此，基于超块的执行模型是完全可以进行块级预测的。

3 超块级预测器设计

3.1 超块级预测器与传统预测器的区别

超块级预测与传统的分支预测存在很大的区别，主要表现在以下 3 个方面：

(1)传统的分支预测器中，每一个分支只有一个出口，用一位来表示跳转或者不跳转；但在超块执行模型下，每一个块存在多个出口。因此，仅采用一位表示分支是否跳转不再可行，而应该对出口号进行预测。

(2)在超快执行模型下，不同的出口号对应不同的目标地址，而且由于块的大小不定，因此需要预测每一个出口，包括顺序出口的目标地址。

(3)在超块执行模型中，分支出口的类型需要到块执行结束时才能获得，因此，如果要使用 RAS，则必须对出口类型进行预测，而这又势必降低整体预测准确率。

3.2 超块级预测器的设计

本文的预测器设计方案包括出口预测和块目标地址预测 2 个部分，出口预测器利用块地址来预测块出口号，类似 Alpha21264 上的分支预测器设计,采用一种可配置的预测器；根据需要，可被配置成全局预测器、局部预测器或基于全局和局部历史的锦标赛预测器等[5]。利用块地址和预测出的分支出口号，可以在分支目标预测器中索引出块转移地址。预测器的基本构造如图 1 所示。

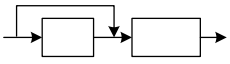


图 1 预测器基本构造示意图

3.2.1 出口预测

全局预测器：这种预测器利用全局分支历史和分支地址相结合作为模式表的索引。

局部预测器：与全局预测器不同，这种预测器的历史寄存器对应每一个分支地址，即局部历史。

锦标赛预测器：锦标赛预测器利用一个计数器表记录全局和局部预测的命中情况，对任一分支地址都可以索引出一个计数值，根据值的大小选择全局或局部预测器的结果。

3.2.2 块目标地址预测

对目标地址的预测，传统分支预测器采用分支目标缓存和返回地址栈相结合的方式。使用 RAS 必需提前知道分支出口类型，在超块执行模型中，很难获得出口类型，而对出口类型进行预测势必降低分支预测的准确性。因此，对于分支目标的预测，只采用 BTB。如图 2 所示，分支目标缓冲的每一项包含几个出口目标地址和预防抖动位，利用块地址和预测出的出口号对其进行索引，即可获得相应块的目标地址。

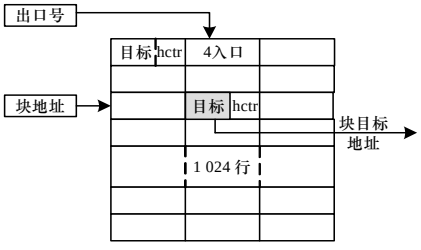


图 2 分支目标缓存结构

4 实验平台及结果分析

4.1 实验平台及相关工具

本文的研究工作以 TRIPS 仿真平台为基础，利用其产生各种应用执行过程的 PC 序列。采用 TRIPS 作为研究工作的平台基于以下考虑：首先 TRIPS 具有超块执行模型的一般特征，具有通用性；TRIPS 公布了其相关的仿真环境和实验工具，便于在其上面进行研究工作。

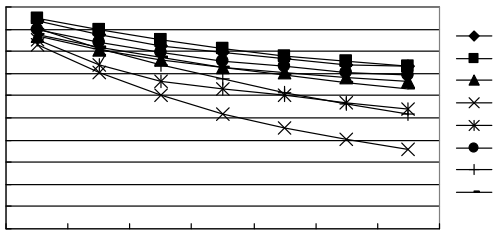
TRIPS 仿真平台包括：编译器 SCALE，利用它能够把一些基于 C 或 PASCAL 的基准测试程序编译成超块结构；TRIPS 模拟器，包括功能模拟器 tsim_arch 和时钟精确模拟器 tsim_ proc；其他一些相关剖析工具等。

实验选取了 SPEC2000 中的部分测试程序，包括整型测试程序中的 gzip, bzip, mcf, vortex 和 parser 以及浮点测试程序中的 ammp, art 和 equake 等。这些测试程序基本涵盖了 SPEC2000 中的各种应用，具有较高的代表性。

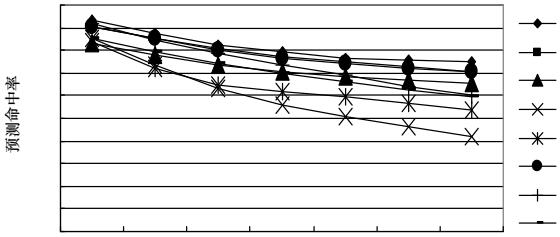
4.2 一定深度下预测性能分析

为了观察不同应用深度推测的可行性情况，将采用全局预测器和局部预测器对测试程序在深度为 1~7 的情况进行预测。

如图 3 所示，随着预测深度的增加，预测的准确率呈下降趋势，但下降幅度并不大，并随着深度增加，下降趋势越来越缓，大多数应用到了第 7 级仍能达到较高的预测准确率，说明采用深度预测及激进执行模型是可行的。



(a)全局预测器



(b)局部预测器

图 3 预测性能随深度变化

4.3 不同应用期望深度及其影响因素分析

在 4.2 节中，能够看到在不同预测器、不同深度下应用的预测准确率的变化趋势，比较直观地反映了应用本身的可预测深度情况，但还存在一定的缺陷：它不能量化地给出不同应用的可预测深度大小，对于研究激进的超块执行模型合适的推测深度选取缺乏有效的指导，另外，这种模型很难反映预测器种类、配置情况及应用本身对期望深度的影响大小。

为此本文提出了期望预测深度的概念：通过对处于不同预测深度的指令块按比例求和，就能得到相应的期望深度。

期望深度直观地反映了不同应用的可预测深度大小，并且能够保证在相应的预测深度下，推测本身能够获得很高的预测准确度，对研究激进执行模型推测深度情况有较好的指导作用。图 4 描述了不同应用的期望预测深度，从图中可以看出，大多数应用都具有 6 以上的期望预测深度，说明这些应用是适合深度推测执行的。

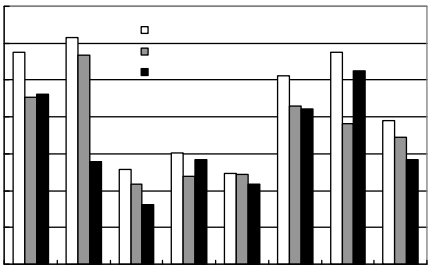


图 4　应用期望预测深度

无论对什么应用，锦标赛预测器都能获得比全局预测器和局部预测器高的期望深度，而全局和局部预测器对不同应用各有所长。对同一预测器，整体上 art 期望深度最高，其次是 vortex 和 ammp，而 bzip 和 gzip 可推测深度则最低。不同应用的期望深度差异非常明显，在锦标赛预测器下，art 的期望深度达到 12 以上，而 gzip 却不到 6，可见应用本身对期望深度的影响非常大。

当把模式表的大小扩大 16 倍时，在锦标赛预测器下，期望深度与原期望深度的比较如图 5 所示。

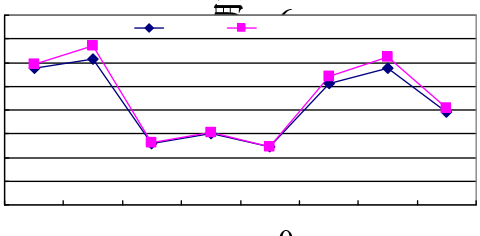


图 5　改进前后期望预测深度变化

不难发现，bzip, equake 和 gzip 几乎没有变化，而其他应用也只有很少的提高，因此，预测器配置对于期望深度的影响是较小的。这是因为对绝大部分应用，预测性能偏低并不是由于模式表的冲突造成的，而是由应用本身特性决定的。

5　结束语

分支预测的本质是由历史信息预测后续执行信息，预测的可行性取决于分支的历史再现性；超块执行模型不仅没有破坏这种再现性，并且在某种意义上增强了可预测性。实验结果表明，大部分应用都具有较高的期望预测深度，但不同应用期望深度相差很大，期望深度的大小反映了应用可预测性深度的高低，更多地取决于应用本身而不是预测器的配置。

在未来的工作中，将进一步拓展研究范围，不仅对 SPEC 基准测试程序进行研究，而且还要对一些专门领域，如多媒体、科学计算等进行研究。

参考文献

[1] Smith A, Burrill J, Gibson J, et al. Compiling for EDGE Architectures[C]//Proc. of International Symposium on Code Generation and Optimization. NY, USA: [s. n.], 2006.

[2] Vijaykumar T N. Compiling for the Multiscalar Architecture[D]. Wisconsin, USA: Philosophy at the University of Wisconsin, 1998.

[3] Ranganathan N, Nagarajan R, Jimenez D, et al. Combining Hyperblocks and Exit Prediction to Increase Front-end Bandwidth and Performance[R]. Department of Computer Sciences, Texas University, Technical Report: TR-02-41, 2002.

[4] Allen J R, Kennedy K, Porterfield C, et al. Conversion of Control Dependence to Data Dependence[C]//Proc. of the 10th Annual ACM Symposium on the Principles of Programming Languages. Austin, TX, USA: [s. n.], 1983.

[5] Kessler R. The Alpha 21264 Microprocessor[J]. IEEE Micro, 1999, 19(2): 24-36.

编辑　索书志

（上接第 248 页）

各个方法在内存文件中的分配情况如表 5 所示。

表 5　方法运行空间和偏移量

相关项方法	Offset	ThisMethodAddr	MethodAdr	pre-len	aft-len	run-size
Object: <init>	16	32	91+32	1	1	41
Object: getClass	0	-	-	0	0	0
Object: hashCode	0	-	-	0	0	0
Object: equals	28	73	91+73	11	11	51
SuperClass: <init>	32	168	91+168	10	18	58
SuperClass: getInt	36	226	91+226	5	6	46
SubClass: <init>	44	332	91+328	6	13	53
SubClass: <add>	0	385	91+385	4	4	44
SubClass: <main>	0	429	91+429	28	38	78

与<clinit>方法相比，表 5 中为每个非<clinit>方法增加了 40 Bytes 的运行空间，目的是为了扩展的需要。offset 表示相对于类起始地址 ThisClassAddr 的偏移量；ThisMethodAddr 则是相对于 Cls_base 的偏移量。

5　结束语

本文提出的类预处理器，仅支持标准的 Java 类库。为了支持 RTSJ 规范，在接下来的工作中要对 API 进行改进并重新实现其类库，生成适合 Jpor-32 使用的实时类库。此外，为保证最坏情况执行时间(WCET)的可预测，需要通过预处理器提供预取指令条数等信息来指导可预测指令 cache 的设计。

参考文献

[1] Bollela G, Gosling J, Brosgol B, et al. The Real-time Specification for Java[M]. [S. l.]: Addison-Wesley, 2000.

[2] Puffitsch W, Schoeberl M. picoJava-II in an FPGA[C]//Proc. of the 5th International Workshop on Java Technologies for Real-time and Embedded Systems. Vienna, Austria: [s. n.], 2007.

[3] Schoeberl M, Pedersen R. WCET Analysis for a Java Processor[C]// Proc. of Conference on Design, Automation and Test in Europe. Paris, France: [s. n.], 2006.

[4] 柴志雷，高丽强，陈章龙，等. 一种用于实时 Java 处理器的类转换器设计及实现[J]. 小型微型计算机系统, 2006, 27(12): 2336.

[5] Venners B. 深入 Java 虚拟机[M]. 北京: 机械工业出版社, 2003.

编辑　顾逸斐