doi:10.3772/j.issn.1002-0470.2014.09.002

高可靠处理器微体系结构设计空间的快速搜索①

尹一笑②*** 章隆兵*** 肖俊华***

(*中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)
(**中国科学院大学 北京 100049)
(***龙芯中科技术有限公司 北京 100190)

摘 要 同一个程序运行在不同微体系结构配置的处理器上时,不同处理器的软错误率通常会有非常大的差别。所以,为了设计高可靠处理器,对处理器微体系结构设计空间进行搜索,选择出高可靠的配置,是至关重要的。然而,目前的处理器软错误率预测方法忽视了执行程序的影响,导致微体系结构设计空间搜索仍然需要大量周期精确的模拟。因此,本文提出了基于同时考虑程序固有特征(与微体系结构无关的程序特征)和处理器微体系结构配置而构建的统一的软错误率预测模型的设计空间快速搜索方案。该方案对新程序或新处理器配置都不需要额外的周期精确的模拟就可以准确预测软错误率,从而极大地缩短了高可靠处理器微体系结构设计空间的搜索时间。对 SPEC CPU2000 基准测试程序的实验结果表明,与基于反应构建软错误率预测模型的方案相比较,该方案搜索高可靠处理器微体系结构设计空间所需的模拟时间减少了99.8%。

关键词 软错误, 程序固有特征, 预测模型, 设计空间搜索

0 引 言

随着芯片制造工艺的发展,芯片特征尺寸不断缩小,工作电压持续降低,尤其是片上晶体管数目指数增长,使得现代处理器越来越容易受到软错误的侵扰[1-3]。软错误是由高能粒子攻击半导体器件产生的。封装材料中的 α 粒子和大气层的中子都是常见的高能粒子。高能粒子穿过半导体器件,会直接或间接地产生电子空穴对。这些电荷可能会被晶体管的源极和漏极吸收。吸收的电荷达到一定量就会导致逻辑器件的状态发生翻转,从而将逻辑错误引入到芯片的运算过程。因为这种类型的错误不会导致器件永久性失效,所以被称为软错误。同一个程序运行在不同微体系结构配置的处理器上时,不同处理器的软错误率通常会有非常大的差别[1-4]。而且,Duan 研究发现,调节某个处理器部件的大小来降低该部件的软错误率,既可能降低也可能升高其他处理器部件的软错误率[4],而处理器软错误率是各个部件的软错误率之和。这意味着尽管处理器软错误率受微体系结构配置的影响,可是在微体系结构配置和处理器软错误率之间并不存在直观的关系。因此,为了设计高可靠处理器,对处理器微体系结构设计空间进行搜索,选择出尽可能高可靠的配置,是至关重要的[2-4]。然而,目前的处理器软错误率预测方法忽视了执行程序的影响,导致微体系结构设计空间搜索仍然需要大量周期精确的模拟。因此,本文提出了基于同时考虑程序固有特征和处理器微体系结构配置而构建的统一的软错误率预测模型的设计空间快速搜索方案。该方案对新程序或新处理器配置都不需要额外的周期精确模拟就可以准确预测软错误率,从而极大地缩短了高可靠处理器微体系结构设计空间的搜索时间。对 SPEC CPU2000 基准测试程序的实验结果表明,与基于反应构建软错误率预测模型的方案相比较,该方案搜索高可靠处理器微体系结构设计空间所需的模拟时间减少了99.8%。

① 国家自然科学基金(61221062,61100163,61133004,61232009,61222204,61221062,61303158),863 计划(2012AA010901,2012AA011002,2012AA012202,2013AA014301),国家"核高基"科技重大专项课题(2009ZX01028-002-003,2009ZX01029-001-003)和中国科学院战略性先导科技专项(XDA06010403)资助项目。

② 女,1987 年生,博士生;研究方向:计算机体系结构,处理器可靠性分析;联系人,E-mail: yinyixiao@ict.ac.cn

(收稿日期:2014-03-17)

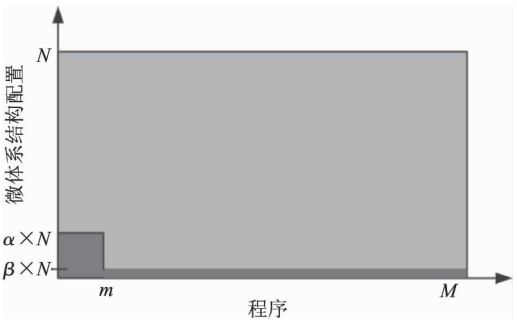
1 相关研究

近年来,很多研究工作关注如何估算处理器软错误率。体系结构级正确性执行(architecturally correct execution,ACE)分析方法[5]使用周期精确的模拟器估算体系结构易损性因子(architectural vulnerability factor,AVF)。处理器部件的 AVF 指的是在处理器部件中出现的一个故障(fault)会导致程序结果出错(error)的概率,是用来计算软错误率的重要因子。然而,使用周期精确的模拟器估算软错误率,所需的模拟时间非常长。通常微体系结构设计空间大小至少为百万量级,如果遍历整个微体系结构设计空间,则时间开销是惊人的。因此,通常只选择很少量的处理器配置进行模拟。于是,Sridharan 提出分别估算程序易损性因子(program vulnerability factor,PVF)和硬件易损性因子(hardware vulnerability factor,HVF)来加快处理器部件 AVF 的估算[6]。因为 ACE 分析方法中估算 cache AVF 的算法复杂且模拟时间长,所以 Suh 提出结合剖析的采样(profile-hybrid sampling,PHYS)来降低 cache 的 AVF 估算时间[7]。可是,由于微体系结构设计空间非常大,使用这些方法做设计空间搜索所需的时间开销仍然是不可接受的。

为了避免耗时的周期精确的模拟,一些研究工作使用机器学习算法构建 AVF 预测模型,从而加快高可靠处理器微体系结构设计空间搜索[2-4]。但是,它们需要为每一个程序构建预测模型。实际上,为了构建一个预测模型,需要做大量的周期精确的模拟才能获取到合适数量的训练样本。总的来说,目前软错误率预测方法都忽视了执行程序的影响。由于通用处理器需要适用于各种各样的应用程序[8,9](比如游戏程序、数据解压缩程序等),如果继续采用传统的软错误率预测方法,则高可靠处理器微体系结构设计空间搜索所需的模拟时间仍然是非常长的。

近年来,一些研究工作[10-13]通过构建预测模型,对处理器性能和功耗做微体系结构设计空间搜索。比如,Chen[10]和 Guo[11]使用半监督学习算法构建性能和功耗预测模型,然而他们的方法仍然需要为每一个程序构建预测模型。为了减少新程序的模拟时间,Khan[12]和 Dubach[13]提出基于反应构建预测模型,从而对新程序不需要再构建预测模型,只需要周期精确的模拟新程序在少量处理器配置上的运行,获取相应的性能或功耗结果(称为反应或签名)做为预测模型的输入,就可以获取预测的性能或功耗。可是,他们都没有显式地将程序特征做为先验知识,并且仍然需要做耗时的周期精确模拟。随着应用程序的持续增多和多样化,这些传统的设计空间搜索方法所需的模拟时间开销也会显著增加。

在本文中,我们提出同时考虑程序固有特征和处理器微体系结构配置,构建统一的软错误率预测模型。因此,对新程序或新处理器配置,我们的方案都不需要额外的周期精确的模拟,就可以准确预测软错误率,从而极大地缩短了高可靠处理器微体系结构设计空间搜索的时间。对 M 个程序和 N 个处理器配置构成的设计空间(M 和 N 是正整数),与基于反应构建软错误率预测模型的方案相比较,我们的方案的模拟时间减少了,如式 $(M-m)\times\beta\times N$ 所示,其中 $0 < m << M, 0 < \beta < 1$,如图 1 所示。因此,随着应用程序数量的持续增多,即 $m << M$,我们的方案将极大地减少设计空间搜索的模拟时间。



(深灰色区域表示基于反应构建软错误率预测模型方案的模拟时间,浅灰色区域表示可预测的设计空间)

图 1 对 M 个程序和 N 个处理器微体系结构配置构成的设计空间进行搜索所需的周期精确的模拟时间

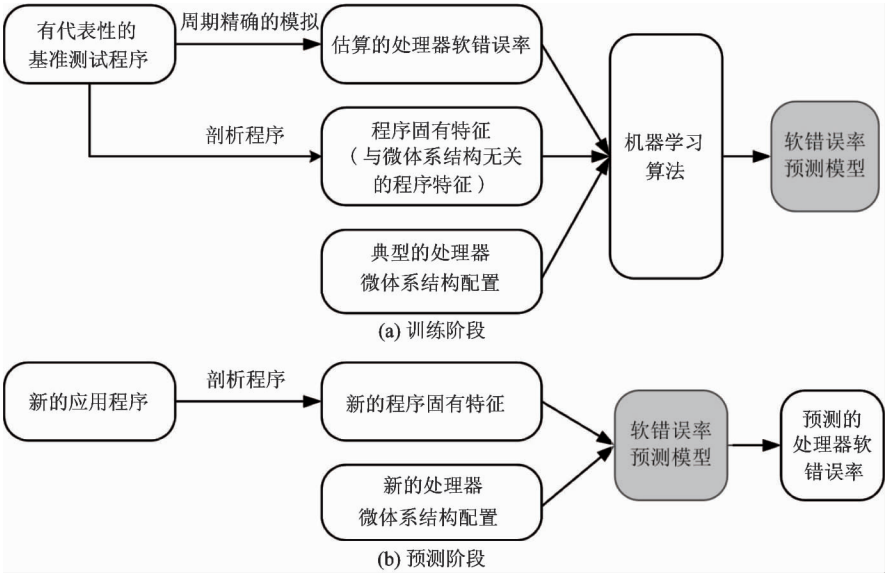
2 设计空间搜索系统设计

这一节介绍方案的总体框架。本方案分成两个阶段:训练阶段和预测阶段,如图 2 所示。

在训练阶段,首先选择出有代表性的基准测试程序和典型的处理器微体系结构配置。然后利用周期精确的模拟器获取每个基准测试程序在典型的微体系结构配置上运行时的处理器软错误率。接着,使用探测工具剖析基准测试程序,获取它们的固有特征。因为这些程序特征是与微体系结构无关的,所以一个程序只需要剖析一次。最后,将基准测试程序的固有特征、处理器微体系结构配置和周期精

确的模拟得到的处理器软错误率输入到机器学习算法,构建软错误率预测模型。

在预测阶段,将新的应用程序的固有特征和处理器微体系结构设计空间中的典型配置做为预测模型的输入,不需要进行周期精确的模拟,就可以快速预测该程序在相应处理器上运行时的软错误率,极大的缩短了高可靠处理器微体系结构设计空间搜索的时间。



(a) 描述的是构建软错误率预测模型;(b) 描述的是使用软错误率预测模型

图 2　方案的总体框架

3　机器学习算法

在获取了有代表性的程序固有特征和典型的处理器微体系结构配置以及相应的处理器软错误率之后,我们使用这些训练数据来构建机器学习预测模型。本文选用了增强学习树模型。它将多个精度一般的树模型组合成精度较强的学习模型,从而更好地完成学习任务。由于该方法可以很好地控制每颗树的复杂度,因此,它的泛化效果比单棵树更好。而且,在训练阶段,该方法采用对样本和特征双重采样的策略,极大地提升了模型的训练效率。增强学习树模型在机器学习领域得到了广泛的研究,是目前应用性最广的模型之一。

增强学习树模型是一种自适应的集成回归树模型。该模型的结构示例如图 3 所示。图中箭头的走向表示预测阶段一个待预测数据所经过的路径。图中每棵树的分裂节点是在模型学习过程中,通过计算训练样本中每一维特征与样本标记结果的信息增益来确定的。预测阶段,待预测数据分别经过每棵树的分裂节点路径到达叶节点,从而获取每棵树对该待预测数据的预测值,然后将所有树的预测值加权累加得到该待预测数据的最终预测值。本文中,待预测数据是由程序固有特征和处理器微体系结构配置组成的,而预测值是相应处理器的软错误率。

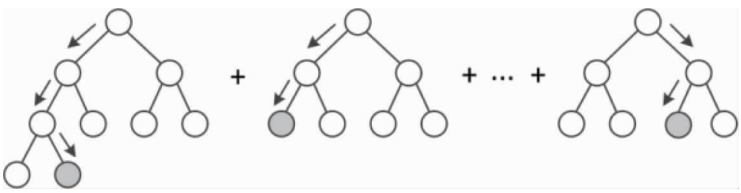


图 3　增强学习树模型的结构示例

定义 X 为程序固有特征和处理器微体系结构配置组成的向量,$f(X)$ 表示增强学习模型预测的处理器软错误率。预测阶段基于 $f(X)=\sum_{j=1}^{J}\beta_j T(X;\theta_j)$,其中 θ_j 包含了第 j 棵树的结构信息,比如树节点所使用的特征和特征分裂值;β_j 为第 j 棵树的权重,J 是树的总棵数。通过最小化损失函数 L 来确定每棵树的 θ_j 和 β_j:$\min\sum_{i=1}^{E}L[y_i,\sum_{j=1}^{J}\beta_j T(X_i,\theta_j)]$,其中 E 为训练样本数,y_i 为第 i

个训练样本中的周期精确的模拟得到的处理器软错误率。

θ_j 和 β_j 的计算过程如下：

(1) 初始化 $f_0(X_0) = 0$；

(2) For $j = 1, \cdots, J$ 计算

$$(\beta_j, \theta_j) = \arg\min \sum_{i=1}^{E} L[y_i, f_{j-1}(X_i) + \sum_{j}^{J} \beta_j T(X_i; \theta_j)]$$

增强学习树模型中每棵树的节点数表示每棵树的复杂度，可以根据具体应用情况设定。在本文中，设定每棵树的节点数为 4 ~ 8，树的深度为 3。此外，我们使用 Friedman 提出的树权重计算方法[14]。

4 实验设置

在这一节中，首先介绍影响处理器软错误率的程序固有特征，它们是与微体系结构无关的，然后介绍本文的处理器微体系结构设计空间，接着介绍本文所使用的周期精确模拟器和基准测试程序，最后介绍软错误率预测模型的构建过程。

4.1 程序固有特征

近年来，一些研究工作将实时采样的微体系结构统计特征输入到机器学习预测模型，预测在程序运行过程中处理器部件的体系结构易损性因子(AVF)[1]。尽管这些方法表明微体系结构统计特征与处理器软错误率存在因果关系，但是它们不适用于早期的高可靠处理器设计阶段。受这些研究的启发，我们利用机器学习预测模型，模拟体系结构无关的程序特征和处理器微体系结构配置与处理器软错误率之间的因果关系。一些研究工作发现与体系结构无关的程序特征能够准确反映程序的固有特性[15]。为了避免周期精确的模拟，我们基于 simplescalar 的功能模拟，增加了剖析程序的探测功能。

程序的可并行化程度。程序本身的可并行化程度决定了处理器能够达到的最佳并行化程度。数据冲突、结构冲突和控制冲突是阻碍理想流水线的主要因素。因此，我们从数据相关性、各种指令所占比例和程序的分支行为这三个方面来评估程序的可并行化程度。

程序的局部性：Cache 缺失和旁路转换缓冲(translation lookaside buffer，TLB)缺失，也是导致流水线阻塞的重要原因。它们都依赖于程序的局部性。程序的局部性包括时间局部性和空间局部性。它不仅与程序固有特征有关，也与处理器微体系结构配置有关。许多工作研究如何精确量化程序的局部性行为。Zhong 等发现，数据再次被使用的距离这一特征可以很好地反映程序固有的局部性[16]。数据再次被使用的距离指的是连续两次使用同一数据之间的指令数。根据定义中数据的指代不同，我们统计的数据再次被使用的距离可以分为数据时间局部性和空间局部性，以及指令时间局部性和空间局部性。

4.2 处理器微体系结构设计空间

为了构建高可靠超标量乱序执行处理器的微体系结构设计空间，选取了 12 个微体系结构参数，每一个参数有一系列可以选择的数值，如表 1 所示。

表 1 处理器微体系结构设计空间

微体系结构参数	取值范围和步长	数值个数
机器宽度	2,4,8	3
取指队列大小	4,8,16	3
ROB 大小	32 ~ 160:32 +	5
发射队列大小	8 ~ 72:16 +	5
访存队列大小	8 ~ 72:16 +	5
Gshare 大小	1k ~ 32k:2 ×	6
BTB 大小	1k,2k,4k	3
L1 Icache 大小	8k ~ 128k:2 ×	5
L1 Dcache 大小	8k ~ 128k:2 ×	5
L2 Ucache 项数	64k ~ 1M:2 ×	5
L2 Ucache 延迟	4 ~ 20cycle:4 +	5
L2 Ucache 关联度	4,8	2

本文的设计空间大小为 25.3125 兆，其中 L2 Ucache 的延迟是随着它的项数相应变化的。我们选取的这些参数对处理器软错误率有较大的影响，并且和其他研究处理器微体系结构设计空间搜索的工作所考虑的参数类似[2-4]。尽管 cache 可以使用纠错码(error correcting code，ECC)进行保护，但是它的大小会影响其他处理器部件的软错误率，所以将 cache 和其他处理器部件综合起来考虑，从而选择出高可靠处理器微体系结构配置。如果选择出的高可靠处理器的 cache 软错误率已经比较低时，则选择低开销的保护措施(如奇偶校验)就可以满足绝大多数制造商的可靠性设计要求。为了减小由于其他微体系结构的设置不同对处理器软错误率的影响，有些微体系结构参数是固定设置的，如表 2 所示。

表 2　处理器固定设置的微体系结构

微体系结构参数	固定设置
流水线深度	5
整型加法器/乘法器	4/2
浮点加法器/乘法器	2/1
BTB 关联度	4
L1 cache 关联度	4 路
L1 cache 块大小	32B
L1 cache 延迟	1 cycle
L2 cache 块大小	64B
ITLB	256k
DTLB	512k

4.3　模拟器和基准测试程序

为了获取程序在不同微体系结构配置的处理器上运行时的软错误率,将 AVF 计算方法[5]添加到周期精确的模拟器 Wattch。我们的模拟器可以估算处理器绝大多数的微体系结构部件的软错误率:发射队列、功能部件、重排列缓冲队列(reorder buffer, ROB)、一级数据 cache、一级指令 cache、load/store 队列和二级 cache 等。

通过使用 SPEC CPU2000 基准测试程序集来评估我们的方案。采用 reference 输入集,并选择最高级编译选项-O3 来编译 SPEC CPU2000。实际上,每个程序都可以由一些程序片段来表示,并且每个程序片段的特性通常都有很大的差异[17]。因此,为了获取更多典型的程序特征,对每个基准测试程序,我们使用 Simpoint[17]选择出最有代表性的 10 个程序片段。每个程序片段的大小为 1000 万条指令。

4.4　构建机器学习预测模型

为了构建机器学习预测模型,我们首先需要获取训练数据。训练数据的内容由三部分组成:程序固有特征,处理器微体系结构配置和相应的处理器软错误率。其中的处理器软错误率是使用周期精确的模拟器模拟程序在不同微体系结构配置的处理器上运行得到的。在理想情况下,训练数据应该尽可能有代表性并且越多越好。然而由于周期精确的模拟需要很长的时间,因此,为了权衡模拟时间开销和模型准确性,对每个程序片段,从处理器微体系结构设计空间中随机均匀采样 1000 个配置。这是我们通过简单实验得出的尽可能少的采样数目。在未来的工作中,我们也可以使用一些先进的采样方法,比如动态学习方法,来提高采样点的多样性。

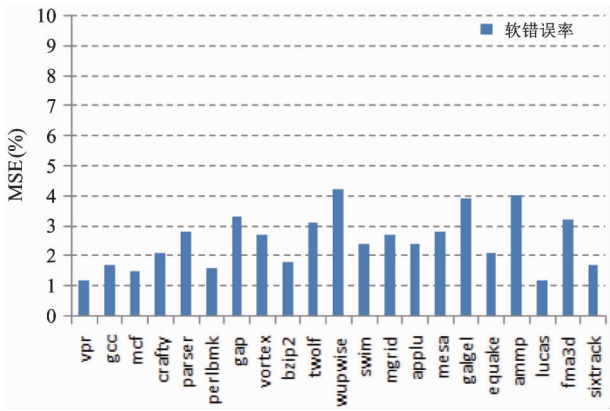
在获取了训练数据之后,为了降低过拟合的风险和充分利用所有的训练数据,使用交叉验证机制来构建软错误率预测模型。我们采用最常用的一种交叉验证机制:10 折交叉验证。首先将所有的训练数据分成 10 等份,然后使用第 1 – 9 份做为预测模型的训练数据,第 10 份做为预测模型的测试数据。接着,将第一轮中的训练数据依次轮转做为测试数据。比如,在第二轮中,使用第 2 – 10 份做为预测模型的训练数据,第 1 份做为预测模型的测试数据。为了尽可能最小化所有测试数据的误差,重复以上过程。

5　实验结果

在这一节中,首先评估软错误率预测模型的准确性,然后实验表明,对新的应用程序,本文方案极大地减少了高可靠处理器微体系结构设计空间搜索所需的模拟时间。

5.1　预测模型的准确性

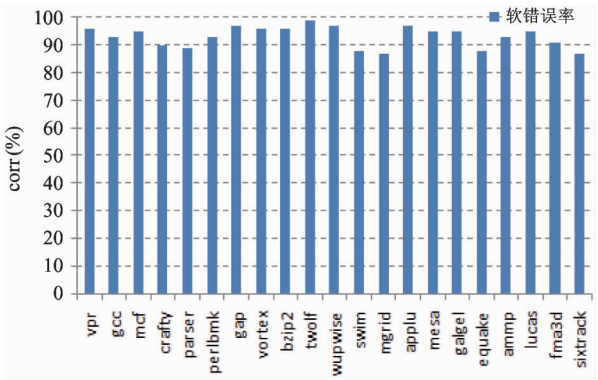
为了评估软错误率预测模型的准确性,使用均方差(mean square error, MSE)这一评估指标。MSE 的计算公式为 $MSE = \frac{1}{K}\sum_{i=1}^{K}(P_i - M_i)^2$,其中 P_i 表示预测模型预测的软错误率,M_i 表示周期精确的模拟器估算的软错误率,K 表示总的采样数。如果 MSE 的数值越小,则表示构建的软错误率预测模型越准确。对 SPEC CPU2000 基准测试程序,我们构建的软错误率预测模型的 MSE 如图 4 所示。对整个 SPEC CPU2000 基准测试程序,预测模型的平均 MSE 为2.5%。



(对 SPEC CPU2000 基准测试程序)

图 4　本文构建的软错误率预测模型的均方差

为了进一步评估软错误率预测模型的准确性，我们使用相关系数来评估预测值和估算值之间的线性关系的强度和方向。相关系数的计算公式为 corr $=\frac{\mathrm{cov}(Y,Y')}{\sigma Y \cdot \sigma Y'}$，其中 σY 和 $\sigma Y'$ 分别表示变量 Y 和变量 Y' 的标准差，cov(Y, Y')表示变量 Y 和变量 Y' 的协方差。Y' 表示预测模型预测的软错误率，Y 表示周期精确模拟器估算的软错误率。相关系数的取值范围为[-1,1]。当相关系数为 1 时，表明预测值的线形和估算值的线形完全相同。当相关系数为 0 时，表明预测值和估算值完全没有关系。对 SPEC CPU2000 基准测试程序，本文构建的软错误率预测模型的相关系数如图 5 所示。对整个 SPEC CPU2000基准测试程序，预测模型的平均相关系数为 93%。综上所述，我们构建的预测模型可以准确预测新的应用程序在新微体系结构配置的处理器上运行时的软错误率。



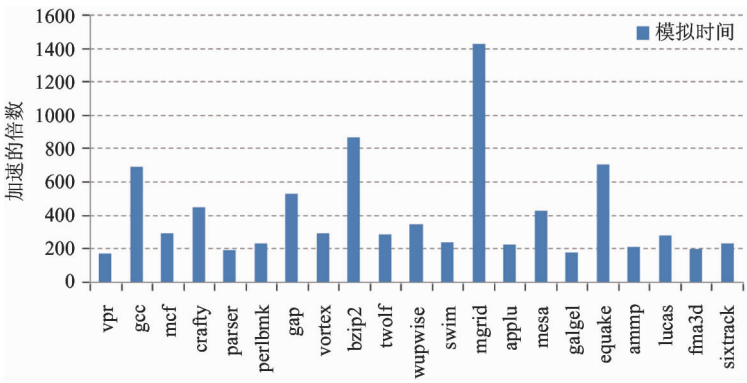
（对 SPEC CPU2000 基准测试程序）

图 5　本文构建的软错误率预测模型的相关系数

5.2　加速设计空间搜索

为了便于描述，本文是基于特征（characteristic-based，CB）的方案，简写为 CB，基于反应（reaction-based，RB）构建软错误率预测模型的方案简写为 RB。为了评估本文的方案效果，对 SPEC CPU2000 基准测试程序，我们分别使用 CB 和 RB 进行高可靠处理器微体系结构设计空间搜索，再比较它们各自所需的模拟时间。RB 采用与 CB 相同的机器学习算法。并且 RB 使用在 8 个处理器配置上运行所得到的反应来表现新程序的特征。

对新的应用程序，RB 需要通过周期精确模拟获取反应数据，再将反应数据做为预测模型的输入，从而预测软错误率。然而，我们的方案 CB 只需要使用探测工具剖析新程序，快速获取程序固有特征做为预测模型的输入，就可以预测软错误率。与 RB 相比较，对每一个新程序，我们的方案 CB 减少模拟时间176.0 ~ 1427.1倍，如图 6 所示。对整个 SPEC CPU2000 基准测试程序，我们的方案减少了99.8% 的模拟时间。随着程序的增多，我们的方案减少的模拟时间将更多，从而极大地加速了高可靠处理器微体系结构设计空间搜索。



（对每一个新程序，CB 减少模拟时间 176.0 ~ 1427.1 倍。对整个 SPEC CPU2000 基准测试程序，CB 减少了99.8% 的模拟时间，极大地加速了高可靠处理器微体系结构设计空间搜索）

图 6　本文方案（CB）与基于反应构建软错误率预测模型的方案（RB）的比较

6　结　论

目前的软错误率预测方法都忽视了执行程序的影响。如果继续采用传统的软错误率预测方法，则高可靠处理器微体系结构设计空间搜索所需的模拟时间非常长。本文提出同时考虑程序固有特征和处理器微体系结构配置，构建统一的软错误率预测模型，从而极大的缩短了高可靠处理器微体系结构设计空间搜索的时间。与基于反应构建软错误率预测

模型的方案相比较,对 SPEC CPU2000 基准测试程序,使用本文方案,高可靠处理器微体系结构设计空间搜索所需的模拟时间减少了99.8%。在将来的工作中,我们可以基于增强决策树机器学习算法,量化分析影响软错误率的微体系结构参数的优先级和权重,从而指导体系结构架构师更好地设计高可靠处理器。

参考文献

[1] Duan L, Li B, Peng L. Versatile prediction and fast estimation of architectural vulnerability factor from processor performance metrics. In: Proceedings of the 15th International Symposium on High Performance Computer Architecture, Raleigh, USA, 2009. 129-140

[2] Nair A A, Eyerman S, Eeckhout L, et al. A first-order mechanistic model for architectural vulnerability factor. In: Proceedings of the International Symposium on Computer Architecture, Portland, USA, 2012. 273-284

[3] Cho C, Zhang W, Li T. Informed microarchitecture design space exploration using workload dynamics. In: Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture, Washington, D. C., USA, 2007. 274-285

[4] Duan L, Zhang Y, Li B, et al. Universal rules guided design parameter selection for soft error resilient processors. In: Proceedings of the International Symposium on Performance Analysis of Systems and Software, Austin, USA, 2011. 247-256

[5] Mukherjee S S, Weaver C, Emer J, et al. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In: Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, Washington, D. C., USA, 2003. 29

[6] Sridharan V, Kaeli D R. Using hardware vulnerability factors to enhance AVF analysis. In: Proceedings of the International Symposium on Computer Architecture, New York, USA, 2010. 461-472

[7] Suh J, Annavaram M, Dubois M. PHYS: profiled-hybrid sampling for soft error reliability benchmarking. In: Proceedings of the International Conference on Dependable Systems and Networks, Budapest, Hungary, 2013. 1-12

[8] Hu W W, Wang J, Gao X, et al. Godson-3: a scalable multicore RISC processor with x86 emulation. *IEEE Micro*, 2009, 29: 17-29

[9] Hu W W, Wang R, Chen Y J, et al. Godson-3B: a 1GHz 40W 8-Core 128GFlops processor in 65nm CMOS. In: Proceedings of the 58th IEEE International Solid-State Circuits Conference, San Francisco, USA, 2011. 76-78

[10] Chen T S, Chen Y J, Guo Q, et al. Effective and efficient microprocessor design space exploration using unlabeled design configurations. *ACM Transactions on Intelligent Systems and Technology*, 2013, 5:. 17-27

[11] Guo Q, Chen T S, Chen Y J, et al. Effective and efficient microprocessor design space exploration using unlabeled design configurations. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Spain, 2011. 1671-1677

[12] Khan S, Xekalakis P, Cavazos J, et al. Using predictive modeling for cross-program design space exploration in multicore systems. In: Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques, Brasov, Romania, 2007. 327-338

[13] Dubach C, Jones T, Boyle M. Microarchitectural design space exploration using an architecture-centric approach. In: Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture, Chicago, USA, 2007. 262-271

[14] Friedman J, Hastie T, Tibshirani R, et al. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 1998, 28: 337-655

[15] Chen J, John L K, Kaseridis D. Modeling program resource demand using inherent program characteristics. *SIGMETRICS Performance Evaluation Review*, 2011, 39: 1-12

[16] Gu X, Christopher I, Bai T, et al. A component model of spatial locality. In: Proceedings of the International Symposium on Memory Management, New York, USA, 2009. 99-108

[17] Sherwood T, Perelman E, Hamerly G. Automatically characterizing large scale program behavior. In: Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, Vancouver, Canada, 2000. 83-94

Fast exploration of the microarchitectural design space of highly reliable processors

Yin Yixiao[*,**], Zhang Longbing[*,***], Xiao Junhua[*,***]

([*] Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

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

([***] Loongson Technology Corporation Limited, Beijing 100190)

Abstract

Since the soft error rate of the same program varies widely across the different microarchitecture configurations, it is crucial to do the microarchitecture design space exploration to design highly reliable processors. However, the current methods of predicting soft error rate do not consider program characteristics, so they need excessive time-consuming simulations for microarchitecture program pairs. In this paper, the authors propose a novel approach which utilizes inherent program characteristics coupled with microarchitectural configurations to construct a universal predictive model of soft error rate so as to accelerate the microarchitectural design space exploration. For a new program or a new microarchitectural configuration, the proposed approach could predict the soft error rate of processors accurately, without additional cycle-accurate simulations. Therefore, it significantly reduces the overall simulation time for design space exploration of the highly reliable processors. The results of the experiment on SPEC CPU2000 benchmarks show that the proposed approach can reduce 99.8% of the simulation time compared with the previous reaction-based approaches.

Key words: soft error, inherent program characteristics, predictive model, design space exploration