

一种基于元数据的采样模拟技术优化

严 强 张为华 刘力力 臧斌宇 朱传琪

(复旦大学并行处理研究所 上海 201203)

摘 要 分析了目前主流采样模拟技术中定长样本的不足,提出了一种基于编译器元数据的采样模拟技术(BigLoopSP).首先利用编译器收集各种可能的周期行为的边界信息作为元数据.然后为了处理程序中大量存在的动态行为,基于编译器产生的元数据结合程序的动态行为进行周期行为的划分和采样点的选取.以此方案划分的变长候选样本能够在保证样本质量的前提下有效地减少所需特征样本的总数.因此比较于定长采样技术 SimPoint, BigLoopSP 在提高精确性的同时,进一步降低了模拟所需的时间(相对于 SimPoint 的平均加速比为 2.63).

关键词 编译;模拟器;元数据;采样模拟

中图法分类号 TP302 **DOI 号**: 10.3724/SP.J.1016.2008.01986

A Metadata-Driven Optimization for Sampling Simulation

YAN Qiang ZHANG Wei-Hua LIU Li-Li ZANG Bin-Yu ZHU Chuan-Qi

(Parallel Processing Institute, Fudan University, Shanghai 201203)

Abstract This paper analyzes the disadvantage of mainstream sampling simulation techniques using fixed-length samples and proposes a metadata-driven optimization for sampling simulation, BigLoopSP. In the approach, the compiler selects candidate loops and annotates the boundaries of those loops as metadata. Those metadata are used to divide the execution into varied-length candidate samples, for which each candidate sample corresponds to one iteration of the chosen loop. Since the program execution exhibits dynamic behaviors, the approach combines the knowledge from the metadata and the dynamic profiles to guide phase partition and selects simulation points for those phases. This approach effectively reduces the number of representative samples while preserving the good quality of them. So, compared with those mainstream sampling simulation techniques, such as SimPoint, our approach achieves better accuracy and reduces more simulation time (a speedup of 2.63X over SimPoint).

Keywords compiler; simulator; metadata; sampling simulation

1 引 言

模拟器作为计算机体系结构研究中的重要工具,已广泛应用于体系结构研究的方方面面.虽然模拟器具有使用灵活、成本低廉的优点,但由于通过软

件来模拟硬件行为,模拟器的执行速度与被模拟的硬件相比极其缓慢.目前最快的精确时钟模拟器的执行速度比对应硬件慢 10^5 ,如要模拟更精确的硬件行为,这种速度上的差距会上升到 10^6 甚至更高^[1].这种情况极大影响了模拟的可行性和效率.因此,如何提升模拟器的运行速度一直是体系结构领

域研究的热点之一。

作为模拟器输入的测试程序,一般运行时间较长都是由于程序中大量的循环或递归造成的。而同一循环或递归的不同迭代间一般会有比较类似的行为(周期行为)。对于这些呈现周期行为的程序段,如果能够从中找出具有代表性的样本,就可以只对这些选取的样本进行精确的时序模拟,而对样本段之间的程序段则仅使用功能模拟^[1],然后利用这些样本段的性能指标来计算推导整个程序的性能指标。由于样本段所包含的总指令数远远小于模拟完整的程序执行所需的指令数,而且在模拟完最后一个样本后就可以结束模拟过程,所以采样模拟技术能够有效地减少单次模拟实验的时间。由于仅仅对选中的样本进行时序模拟,采样模拟技术必然会引入一定的精度误差。如何在保证精度的前提下,尽可能缩短模拟时间一直是采样模拟技术所需解决的关键问题之一。

目前主流的采样模拟技术一般都采用定长样本方法^[1-3]。即把整个程序的执行过程按照固定的指令数划分成一系列等长的候选样本,随后根据收集的数据,发现不同样本的特性,从而把这些样本归类到不同的周期行为。然而,作为直接导致被模拟的程序呈现出周期性行为的循环或者递归的不同迭代的边界通常都会在程序的执行过程中发生一定的变化,即不同的循环的不同的迭代或者不同的函数的不同次序调用所包含的指令数一般都是不同的。所以使用定长样本划分样本区间可能导致样本边界与实际程序周期性行为的边界不相符,此时样本分析程序进行划分过程中,将可能导致需要划分更多的周期行为,从而需要选取更多的特征点来进行模拟。

但是,如果要使用程序中可重复结构体的边界,例如循环的单次迭代,作为划分候选样本的边界,仅仅通过分析可执行二进制文件很难准确地确定这些边界的位置和相互关系。然而编译器作为程序分析和代码生成的工具,在其分析和生成代码的过程中,可以很容易收集到各种边界信息。如果这些信息能够以元数据的形式保存下来,那么准确地识别可重复结构体的边界和相互之间的关系就会变得很容易。

本文在分析采样模拟技术中定长样本的不足之后,提出了一种以编译器元数据作为基础的新模拟器采样技术(BigLoopSP)。我们首先利用编译器收集各种可能的周期行为的边界信息作为元数据,然后基于编译器产生的元数据结合程序的动态行为进行周期行为的划分和采样点的选取。实验结果

显示,相比较于已有的定长采样技术 SimPoint^[2-3], BigLoopSP 在提高精确性的同时,进一步降低了模拟所需的时间,相对于 SimPoint 可以获得平均 2.63 的平均加速。

本文第 2 节分析了定长样本采样技术的不足;第 3 节具体阐述了 BigLoopSP 样本选取算法;第 4 节给出了对该算法评估的过程和结果;第 5 节列举了模拟器加速和编译器元数据领域的相关研究工作;最后在第 6 节中给出结论。

2 定长样本采样技术的不足

定长样本采样技术是目前应用最为广泛的模拟器采样技术之一。虽然该方法在保证精确性的同时可以极大地减少模拟所需的时间,但这种方法也存在一些不足。最主要的不足是定长样本的边界可能与实际造成程序周期行为(如循环或递归的不同迭代)的边界不匹配。由此导致对实际的周期行为刻画不够精确,从而造成采样点选取的冗余。我们将使用如下例子来说明该不足。

图 1 和图 2 中分别描述了 173.applu 中使用 100M 定长指令数划分和以循环迭代为边界划分候选样本之后,样本行为的特征伴随时间的变化。在这里使用了基本块向量 BBV 作为衡量程序运行过程中行为变化的基本指标。基本块向量 BBV (Basic Block Vector)^[2]是学术界常用的刻画程序行为的指标。它是一段程序执行的指令流中带权重的基本块执行频率的向量,向量中的每个值是对应的基本块在该程序段中被执行的次数乘上该基本块内指令总数。由于 BBV 是多维向量,图 1 和图 2 中的曲线是使用主分量分析 PCA (Primary Component Analysis)^[4]降维之后得到的特征曲线,其中横坐标是样本编号,纵坐标是降维之后得到的特征值。

通过比较这两组变化曲线,可以发现以循环迭代为边界划分候选样本之后,BBV 的特征值的变化很规则。图 2 中曲线基本成一直线,而在图 1 中,曲线的变化则呈现出复杂的模式。造成这一现象最主要的原因是 100M 指令数定长样本的边界与实际的行为周期的边界不匹配,在 173.applu 中实际的行为周期长度为 1540M 条指令。因此,对于这样的情况使用定长样本造成的后果是程序被划分为更多的周期行为,从而需选取更多的采样点。

从上面的示例中可以看到,这种边界不匹配可能对收集的样本行为的特征数据产生严重的影响,

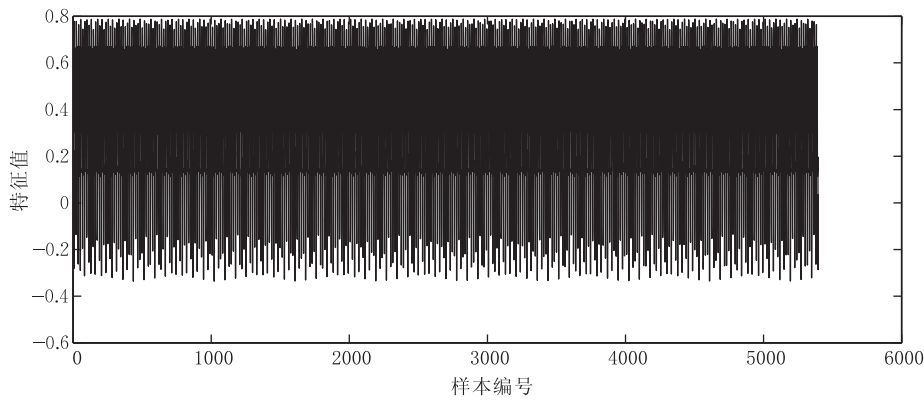


图 1 173.aplu 的 100M 定长样本的 BBV 随程序执行过程的变化

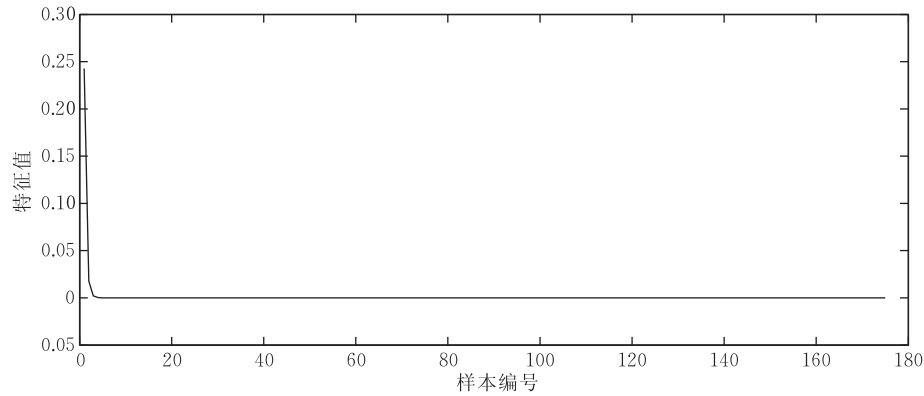


图 2 173.aplu 以循环迭代为边界划分样本的 BBV 随程序执行过程的变化

使得原本简单的行为模式呈现出一种复杂的外部表现,由此增大了样本分析程序从中提取出行为特征的难度.对于这种边界不匹配造成的不良影响,可以从以下两方面来进一步分析:

(1) 当定长样本包含了多个实际的行为周期时(图 3 中的情况 A),定长样本不能准确地找出单个周期.这种情况会导致冗余的周期被选取,尤其当定长样本的大小不是实际周期的整数倍时会导致定长样本内部包含不完整的周期,从而增加样本特征数据的复杂度,使得更多冗余的行为周期被选为模拟样本,进一步增加了模拟时间.

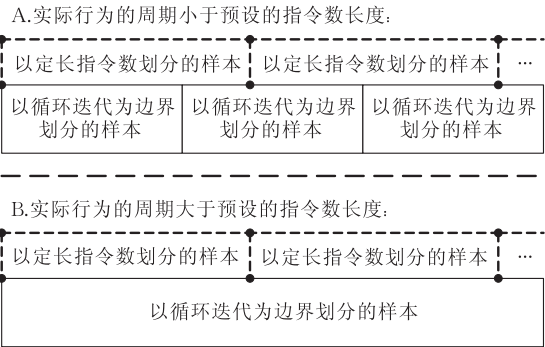


图 3 定长样本的边界与实际的行为周期的边界不匹配的两种情况

(2) 当定长样本不能够容纳单个实际的行为周期(图 3 中的情况 B),此时单个定长样本就不能表征整个周期行为的特征,那么就需要使用多个样本来代表这一周期.

而且无论是出现哪种不匹配的情况,由此造成的样本数目增多除了会增加时序模拟的时间之外,也有可能大幅增加那些样本之间的功能模拟所需的时间.图 4 中给出了由于一个样本分散引入额外模拟时间的示例.图中假定循环中每次迭代的行为都是一致的,如果使用以循环迭代为边界划分的样本,可能只需要选取第一个迭代所在的样本作为特征样本即可.但是如果使用以定长指令数划分的样本,由于划分之后的样本边界与实际循环迭代的边界不匹配,从而导致需要选两个样本才能表征这个循环,那么对这两个样本之间的过渡指令段进行模拟就会引入额外的功能模拟时间.由此可见,选取合适的非定

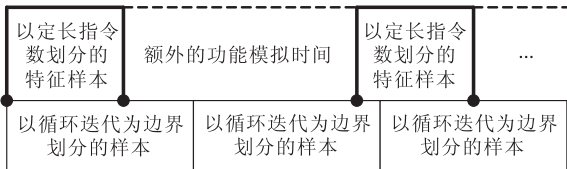


图 4 由于定长的特征样本的数目增多引入的额外的功能模拟时间

长边界划分候选样本有利于收集到更为精确的样本信息,从而提高模拟速度。

3 BigLoopSP 样本选取算法

基于上一节的分析,本节中给出一种基于编译器元数据的优化模拟样本选取算法 BigLoopSP. 由于程序中的循环和递归行为是导致程序周期行为最常见的可重复结构,所以 BigLoopSP 算法中使用循环迭代的边界作为划分候选样本的依据. 对于递归的处理将是我们未来工作的一部分。

- 该算法的主要步骤如下。
- (1)生成元数据. 编译器分析源程序,在生成二进制可执行文件的同时生成记录候选循环边界的元数据,如果遇到嵌套循环还要筛选出位于最合适层次上的循环。
 - (2)选取主循环. 找出一组循环使其覆盖率足以表述整个程序运行中的行为特征。
 - (3)捕获动态行为. 通过 Profiling 对上一步选出的循环收集以各自循环迭代为边界的候选样本的动态行为。

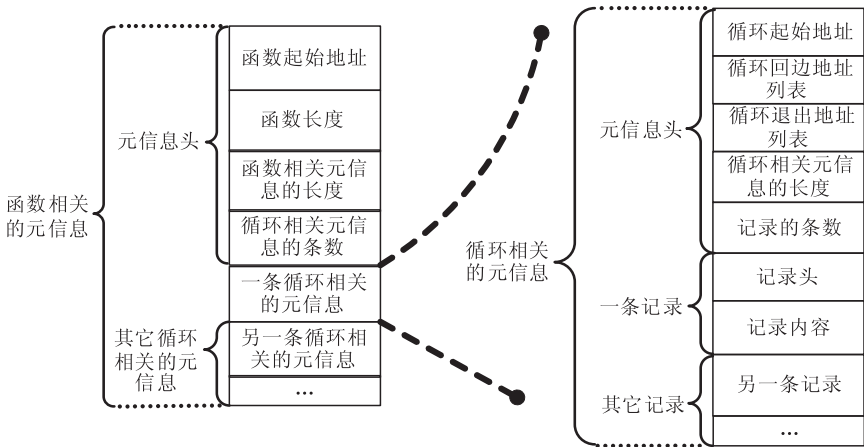


图 5 编译器元数据的格式

在生成元数据时如果遇到嵌套循环,还需要对其进行筛选,选出位于最适合层次上的循环,没被选中的循环将不会出现在最终生成的元数据中. 最适合层次的具体定义是从外向里最低层次的完美循环嵌套(perfect loop nests)所在的层次. 下面的代码给出了一个在循环嵌套中选择最适合层次上的循环的示例。

```
for (i=0; i<100; i++){//LOOP0
    for (j=0; j<50; j++){//LOOP1
        statementA;
    ...
    }
```

(4)确定模拟样本. 分析收集到的动态信息以此确定模拟样本的位置和权重。

3.1 生成元数据

为了在选取样本时准确获取各个周期行为样本的边界信息,首先需要使用编译器产生标记循环相关信息的元数据. 这类元数据有助于在指令流中准确地识别出循环的边界以及了解循环之间的关系。

这里对我们先前工作^[5]中的元数据的基本框架进行扩展,来保存所需的循环边界的元数据. 扩展后的元数据格式如图 5 所示. 目前该结构记录的元数据为候选循环的边界信息. 元数据在组织上呈级联结构,一共可以分成以下三层. 顶层的元数据是函数相关的信息,第二层是的循环相关的信息,第三层是描述循环内部细节的记录信息. 这些循环相关的信息是本文为了划分候选样本而新加的信息,其中包括了循环的起始指令的地址,回边指令和出边指令的地址列表. 第三层的记录信息是为了保持当前元数据格式的扩展性,用于描述循环内部更细节的特征. 元数据的生成可以通过编译器中的控制流分析^[6]获得,生成之后的元数据将以额外的元数据段的形式附加在原始的二进制可执行文件中。

```
for (k=0; k<25; k++){//LOOP2
    ...
}
...
statementB;
}
```

在上例中,LOOP0 和 LOOP1 就属于一组完美循环嵌套. 由于 LOOP1 和 LOOP2 之间存在一些循环控制语句之外的其它语句,所以它们就不属于一组完美循环嵌套. 根据前面的规则选择 LOOP1 作

为代表这一循环嵌套的最适合层次上的循环. 在这里之所以不选 LOOP2 所在的层次作为最适合层次上的循环, 是因为这样做会导致 statementA 和 statementB 等语句不会被任何候选循环的范围所包含, 由此导致的循环覆盖面不足可能会对最终结果的准确性造成严重的影响. 这种倾向于选择靠外层循环的策略, 可能会选到包含指令数比较多的循环(大循环), 但是大循环往往更具有代表性且受冷启动误差(Cold Start Bias)的影响也比较小. 这也是我们把优化之后的算法称之为 BigLoopSP 的原因.

3.2 选取主循环

主循环是程序中最终参与候选样本选取的循环. 为了选出主循环, Profiler 会收集上一步中作为元数据标记的那些循环在程序运行过程中实际所占指令数的比例. 然后筛选出所占指令数比例大于 0.5% 的循环, 并记录下它们在整个程序运行过程中所占的具体权重. 对于那些所占指令数比例不大于 0.5% 的循环(小循环), 将不会在其中选取模拟样本. 一方面是因为忽略它们不会对整体的各方面性能指标造成很大的影响, 另一方面是因为如果该循环位于程序执行的末端的话, 可能会引入一段很长的功能模拟作为过渡, 由此会大幅增加模拟所需要的总时间. 图 6 的示例说明了这种情况, 表征指令数比例大于 0.5% 的循环的最后一个样本可能与指令数比例不大于 0.5% 的循环的最后一个样本之间相隔很长的一段距离, 由此会额外引入很长的一段功能模拟的时间, 而且即便包含这一小循环, 也不会对整体结果的精确性有明显提升. 所以在选取主循环的过程中, 算法将舍弃这类小循环.

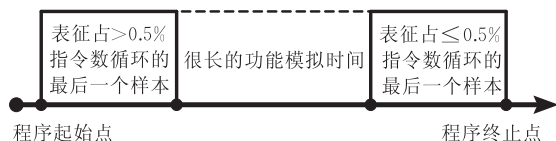


图 6 引入小循环可能造成样本间过渡的功能模拟时间大幅增加

3.3 捕获动态行为

由于一个循环中通常会包含一些由于分支引起的行为变化, 所以不能简单地认为一个循环中所有的迭代的行为都是一成不变的. 所以, 算法需要通过 Profiling 捕获这些动态行为. BigLoopSP 算法也使用 BBV^[2] 来衡量循环中迭代之间行为的变化. Profiling 工具分析获得主循环各个迭代的 BBV 数据后, 就可以进入下一步确定模拟样本.

3.4 确定模拟样本

SimPoint 中提供的 k -mean 聚类分析工具^[3]是学术界主流的样本聚类分析工具. BigLoopSP 算法中使用它对主循环的 BBV 数据进行分析, 以此确定最终的模拟样本. 使用这一工具最关键的步骤就是确定 k -mean 算法的关键参数 K_{\max} . 在 100M SimPoint 中^[7], K_{\max} 的推荐值是 10. 但是这个值是针对定长样本设置的, 而 BigLoopSP 算法中的单个样本与定长样本相比会更具有代表性, 为了避免选择冗余的模拟样本, 需要适当减小 K_{\max} 的值.

因此, 我们定义了以下公式来计算 K_{\max} 的值.

$$K_{\max} = \min\{10, \max\{1, \alpha \cdot \sqrt{\text{COV}_{\text{instructionCount}}}\}\}$$

在这里, $\text{COV}_{\text{instructionCount}}$ 指代主循环中每次迭代指令数的变异系数 COV (Coefficient Of Variation), α 是放大因子, 当前值设为 10. 这里我们使用 100M SimPoint 中 K_{\max} 的推荐值 10 作为 K_{\max} 的极大值.

如果主循环迭代之间的变异系数值很小, K_{\max} 就会取到 1. 在变异系数上加上了平方根之后进一步限制了 K_{\max} 随着变异系数的增长, 但是这并不影响使用优化算法获得精确的模拟结果. 在后面的实验中可以看到, 对于总体结果, 即便当 $K_{\max} = 1$ 时, 与最精确的 100M SimPoint ($K_{\max} = 10$) 相比, 优化算法的模拟结果在精确性上也会有所提高, 而且模拟所需要的总时间也会进一步地减少.

4 实验结果

由于测试时间的限制, 我们选用了 SPECfp2000 中的 9 个基准程序 (wupwize, swim, mgrid, ap-
plu, mesa, facerec, lucas, sixtrack 和 aspi) 来评估 BigLoopSP 的效果. 这些测试程序都使用 ref 输入集. 附带元数据的 Alpha 可执行文件是通过修改过的 GCC 4.0 编译产生. 模拟器方面选用的是 SimpleScalar Tool Set 3.0^[8]. 模拟器具体的参数配置请见表 1.

由于 100M SimPoint^① 是 SimPoint 精确性最高的配置(如无特殊说明, K_{\max} 取为文献[7]中的推荐值 10), 所以我们使用它作为 BigLoopSP 的比较对象, 并使用 CPI 作为比较结果精确性的衡量标

① SimPoint 最新的工作 Software Phase Marker^[9] 是跟我们工作最类似的工作, 但是由于其仅仅通过分析二进制文件获取周期行为的边界信息, 造成其准确性不如使用定长样本的 SimPoint 准确, 同时由于它也没实现到 SimPoint 的最新版中, 因此, 我们无法比较直接与它的结果进行比较.

准. 精确性比较的基准数据通过运行 SimpleScalar Tool Set 中原始版本的 sim-outorder 获得的. 此外为了对 K_{\max} 取值的有效性进行评估, 我们也尝试对 100M SimPoint 选用较小的 $K_{\max}(1, 2, 3, 4, 5)$, 以此来与 BigLoopSP 的结果进行比较.

表 1 模拟器配置	
参数	值
乱序执行	8 路解码, 发射, 提交宽度
ROB/LSQ 大小	128/64
寄存器	32 个整形寄存器, 32 个浮点寄存器
功能单元	6 个 integer ALU, 2 个 load/store unit, 6 个 FP adder, 4 个 integer MULT/DIV, 4 个 FP MULT/DIV
L1 指令缓存	32KB, 直接映射, 32 字节块大小, 1 个时钟周期延时
L1 数据缓存	128KB, 2 路组相联, 32 字节块大小, 1 个时钟周期延时
混合 L2 缓存	1MB, 4 路组相联, 64 字节块大小, 23 个时钟周期延时
分支预测器	Combined, 16KB 个 BHT 表项
内存访问延时	330, 20 个时钟周期延时(first, following)

4.1 性能评估

图 7 和图 8 中分别给出了 BigLoopSP 和 100M SimPoint 之间相对加速比和 CPI 误差的比较. 这里的平均值 AVG 使用的是几何平均. 从总体结果上看, 即便与 SimPoint 精确性最高的配置 100M SimPoint 相比, BigLoopSP 在快 2.63 倍的前提下, 取得了更高的 CPI 精确度. BigLoopSP 的平均 CPI 误差是 0.27% (最大 CPI 误差 1.08%), 而 100M SimPoint 的平均 CPI 误差是 0.56% (最大 CPI 误差 2.94%). 在 9 个测试程序中, 有 4 个测试程序的 BigLoopSP 的 CPI 误差略大于 100M SimPoint, 但无论从更精确结果的个数, 误差的相对偏移值还是平均结果, 都说明 BigLoopSP 可以获得更精确的结果.

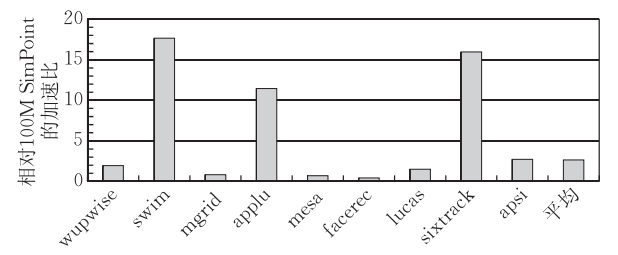


图 7 BigLoopSP 相对于 100M SimPoint ($K_{\max}=10$) 的相对加速比

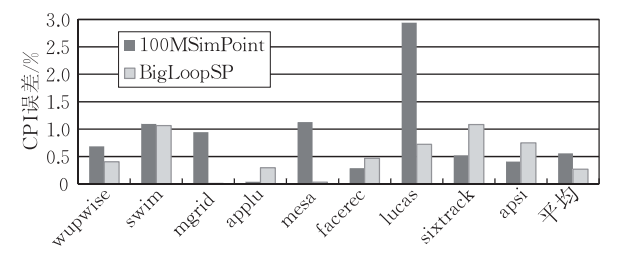


图 8 BigLoopSP 与 100M SimPoint 在 CPI 误差方面的比较

表 2 中的数据解释了 BigLoopSP 能够相对于 100M SimPoint 取得 2.63 倍平均加速的原因. 其中给出了 BigLoopSP 与 100M SimPoint 中时序模拟和功能模拟的指令数所占完整模拟整个程序执行所需指令数的比例, 其中的 K_{\max} 值* 是 BigLoopSP 为基准程序中占指令数比例最大的主循环选取的 K_{\max} 值. 从中可以看到: (1) 由于 BigLoopSP 使用了合适的变长样本, 大幅减少了模拟样本的数量; (2) 尽管采用倾向于选择靠外层循环的策略可能会增加一定的时序模拟的时间, 但是由此带来的样本总数减少直接导致了样本之间功能模拟时间的大幅减少, 进而提高了模拟加速比.

另外, 从中也能看到倾向于选择靠外层循环的策略带来的一个副作用: 当功能模拟时间的减少不能抵消时序模拟时间增加时, 模拟所需的总时间

表 2 BigLoopSP 与 100M SimPoint 的模拟行为统计					
基准程序	K_{\max} 值*	时序模拟的指令数/%		功能模拟的指令数/%	
		BigLoopSP	100M SimPoint	BigLoopSP	100M SimPoint
wupwise	1	1.31	0.16	6.37	72.98
swim	1	0.12	0.29	1.00	73.38
mgrid	1	4.00	0.10	12.00	77.43
applu	1	0.29	0.17	1.57	97.28
mesa	2	0.20	0.28	80.02	99.29
facerec	1	1.46	0.21	87.56	90.22
lucas	1	0.81	0.63	21.03	92.21
sixtrack	1	0.04	0.08	1.78	78.56
aspi	1	1.42	0.17	5.97	96.81
平均值	1.1	0.52	0.20	8.30	85.85

就会变长. 这就解释了 mgrid 和 facerec 在使用 BigLoopSP 之后所获得模拟加速比会比使用 100M SimPoint 所获得的模拟加速比低的原因.

4.2 K_{\max} 值的有效性分析

为了评估 K_{\max} 值的有效性, 我们尝试减小了 100M SimPoint 中的 K_{\max} 值. 图 9 和图 10 中给出了 100M SimPoint 平均相对加速比(相对于 $K_{\max}=10$ 的 100M SimPoint)和平均 CPI 误差伴随 K_{\max} 的变化趋势. 从中可以看到, 在 K_{\max} 大于 2 之后, 100M SimPoint 的模拟所需的平均时间已经与 $K_{\max}=10$ 的 100M SimPoint 所需的平均时间基本相近, 这是因为此时样本之间的功能模拟时间已经成为决定模拟所需的总时间的主要因素. 而且对于 K_{\max} 小于 5, 100M SimPoint 所取得 CPI 误差明显大于 $K_{\max}=10$, 最大误差都超过了 5%, $K_{\max}=1$ 时最大误差甚至达到了 18.54%. 由此可见使用以循环迭代为边界划分候选样本的确有利于在保证样本质量的前提下减少样本总数.

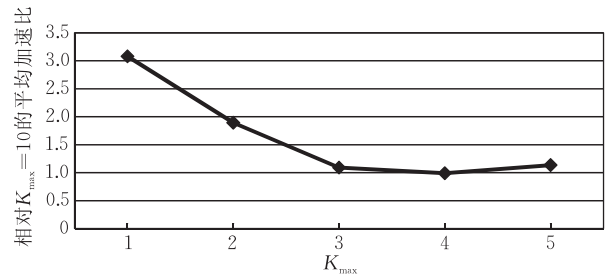


图 9 100M SimPoint 的相对加速比伴随 K_{\max} 的变化趋势

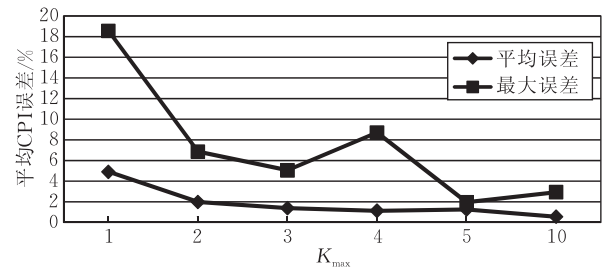


图 10 100M SimPoint 的 CPI 误差伴随 K_{\max} 的变化趋势

4.3 元数据的开销

图 11 中给出了可执行文件在添加元数据之后,

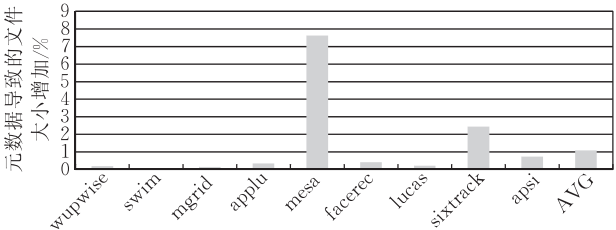


图 11 BigLoopSP 中使用元信息导致可执行文件大小增加量占原始文件大小的比例

文件大小相对于原始文件大小的增加量的比例. 其中除了 mesa 和 sixtrack 之外, 其他基准程序可执行文件大小的增量都在 1% 以内. mesa 和 sixtrack 的文件大小的增量超过 2% 的原因是这些程序内部含有大量循环结构.

5 相关工作

5.1 模拟器加速相关

模拟器加速领域的研究工作主要可以分成以下几类: (1) 精简输入集; (2) 截断执行; (3) 采样模拟; (4) 并行模拟.

精简输入集. 指的是通过调整程序输入集, 使得完整模拟所需要指令数变少, 从而达到减少模拟时间的效果. KleinOsowski 等人为 SPEC CPU 2000 创建了 MinneSPEC^[10]. Alameldeen 等人对 TPC-C 提供了精简输入集^[11]. 虽然使用该技术不需要对模拟器本身进行修改, 但是在精简输入之后, 为了保持程序在接受原始输入集时的行为特征, 往往需要花费很大的代价来对程序本身进行分析. 然而对于那些大型程序, 要比较准确地提取程序特征通常是很困难的.

截断执行指的是当程序模拟完一段预先设定的长度的程序段之后就停止模拟. 其最常见的形式可以分成 3 个阶段: 首先是使用功能模拟跳过被认为是初始化过程的 X 条指令, 然后进行 Y 条指令的预热(warmup), 最后在完成随后的 Z 条指令进行时序模拟之后就停止模拟. 这里的 X, Y, Z 都是预先设定好常量, 并且只对最后 Z 条指令收集所关注的特征数据. 这个技术可以应用在各种需要模拟的场景里, 但由于不同程序在不同输入时所需特征段的长度和位置存在着很大的差异, 所以该技术很难精确地刻画程序的行为特征.

采样模拟可以进一步细分成基于统计理论的采样模拟和基于特征样本的采样模拟. 此外一些预热技术诸如^[12-13]也常常使用在采样模拟中, 以此来减少冷启动所带来的非样本误差(code-start bias). 基于特征样本的采样模拟是指通过分析程序执行时的特征来选取能够最好地表现这些特征的样本. SimPoint^[2-3] 就是其典型的代表. 基于统计理论的采样模拟则不需要对被模拟程序的特征进行分析, 而仅仅是通过统计理论确定当前样本的有效性. 这类技术的代表是 SMARTS^[1]. SMARTS 的样本点均匀分布在整个模拟过程中, 如果当前模拟完成之后结

果的精度达不到指定的置信度,可以根据当前结果中的反馈数据为下一次模拟做参数调整(诸如提高采样频率等)。在速度方面,由于 SMARTS 选取的样本点均匀分布在整个程序段中,即必定有些样本段的位置靠近整个程序模拟的结束点,所以其模拟时间更多地受限于完整模拟整个程序执行所需的时间,因此 SMARTS 模拟所需的时间通常都比 SimPoint 所需的时间长。在结果的精确性方面, SimPoint 由于只使用基于程序指令流 BBV 作为选取定长指令数的样本段的依据,所以其内存方面的结果比较差;而 SMARTS 不存在根据程序特征选点的问题,所以其各项结果总体上都要比 SimPoint 好。Software Phase Marker^[9]对 SimPoint 进行了扩展,使之能够选取变长样本。它通过分析二进制文件获取周期行为的边界信息,但是由于获得边界信息不精确导致其准确性不如使用定长样本的 SimPoint 准确。

采样模拟是与本文工作最为相近的一类模拟加速技术,但这些主流的采样模拟技术都是以定长指令数划分候选样本,而且都没有借助编译器来优化采样模拟的过程。而本文的贡献就在于提出了借助编译器提供的元数据对采样模拟技术提供更好的变长候选样本划分方案,以此达到优化采样模拟的目的。

并行模拟指的是通过把模拟过程分配到多个处理器上来提高模拟速度。Girbal 等人^[14]提出了一种分布式框架 DiST。它可以通过观察模拟重叠程序段的 CPI 误差来动态地调整每个处理器上预热过程的长度,以此在保证减小预热代价的同时最大可能地减少冷启动所带来的误差。Penry 等人^[15]基于 Liberty Simulation Environment (LSE)^[16]提出了一种自动化的模拟器并行技术。其他与并行模拟相关的工作还包括文献^[17-18]所述的工作。

5.2 元数据相关

编译器元数据主要应用于动态优化领域,编译器元数据可以与收集到的动态程序行为相结合提高程序的运行效率。JAVA Anotation^[20-21]是其典型的应用,其中编译器生成的元数据记录了与动态优化的相关信息,以此减少动态翻译或优化 JAVA 字节的代价。在文献^[5]中,元数据被用来放宽 IA-32 EL^[22]动态优化时对寄存器分配的限制,由此提高翻译代码的运行效率。而本文所提及的利用编译器元数据与程序动态行为结合对采样模拟的过程进行优化的方法则为本文首创。

6 结 论

本文分析了采样模拟技术中定长样本的不足,提出了一种基于编译器元数据的采样模拟技术优化 BigLoopSP。该优化技术借助编译器提供的元数据准确地标识出程序中循环的边界,并以循环的单次迭代作为划分候选样本的边界,而不是采用主流采样模拟技术中依照定长指令数来划分样本边界的方案。以此划分变长候选样本有利于收集到更为精确的行为特征信息。在与 100M SimPoint 的比较实验中,以循环的单次迭代划分的候选样本在保证样本质量的前提下有效地减少了所需特征样本的总数。因此相比较于 100M SimPoint, BigLoopSP 在提高精确性的同时,也进一步降低了模拟所需的时间。

目前这种技术还不能优化由于递归引起的周期性行为,递归结构相对于循环结构有着更为复杂的行为模式,对于这些模式的分析和处理将是我们将来的工作重点。

参 考 文 献

- [1] Wunderlich R E, Wenisch T F, Falsafi B, Hoe J C. Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling//Proceedings of the 30th Annual International Symposium on Computer Architecture. New York, USA, 2003: 84-97
- [2] Sherwood T, Perelman E, Calder B. Basic block distribution analysis to find periodic behavior and simulation points in applications//Proceedings of the International Conference on Parallel Architecture and Compilation Techniques. Washington, DC, USA, 2001: 3-14
- [3] Sherwood T, Perelman E, Hamerly G, Calder B. Automatically characterizing large scale program behavior//Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems. New York, USA, 2002: 45-57
- [4] Johnson R A, Wichern D A. Applied Multivariate Statistical Analysis. 5th Ed. Upper Saddle River, NJ, USA: Prentice Hall, 2002
- [5] Xu C, Li J, Bao T, Wang Y, Huang B. Metadata driven memory optimizations in dynamic binary translator//Proceedings of the 3rd international conference on Virtual execution environments. New York, USA, 2007: 148-157
- [6] Muchnick S S. Advanced Compiler Design and Implementation. San Francisco, CA, USA: Morgan Kaufmann, 1997
- [7] Perelman E, Hamerly G, Calder B. Picking statistically valid and early simulation points//Proceedings of the International

- Conference on Parallel Architectures and Compilation Techniques. Washington, DC, USA, 2003; 244-255
- [8] Burger D, Austin T M. The simplescalar tool set, version 2.0. Madison, Wisconsin, USA: Computer Sciences Department, University of Wisconsin - Madison, Technical Report 1342, 1997
- [9] Lau J, Perelman E, Calder B. Selecting software phase markers with code structure analysis//Proceedings of the International Symposium on Code Generation and Optimization. Washington, DC, USA, 2006; 135-146
- [10] Klein Osowski A J, Lilja D J. Minnespec: A new spec benchmark workload for simulation-based computer architecture research. IEEE Computer Architecture Letters, 2002, 1(1): 7-11
- [11] Alameldeen A R, Martin M M K, Mauer C J, Moore K E, Xu M, Hill M D, Wood D A, Sorin D J. Simulating a \$2m commercial server on a \$2k pc. Computer, 2003, 36(2): 50-57
- [12] Haskins J W, Skadron K. Minimal subset evaluation: Rapid warm-up for simulated hardware state //Proceedings of the International Conference on Computer Design: VLSI in Computers & Processors. Washington, DC, USA, 2001; 32-39
- [13] Haskins J W, Skadron K. Memory reference reuse latency: Accelerated warmup for sampled microarchitecture simulation//Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software. Washington, DC, USA, 2003; 195-203
- [14] Girbal S, Mouchard G, Cohen A, Temam O. Dist: A simple, reliable and scalable method to significantly reduce processor architecture simulation time//Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. New York, USA, 2003; 1-12
- [15] Penry D, Fay D, Hodgdon D, Wells R, Schelle G, August D, Connors D. Exploiting parallelism and structure to accelerate the simulation of chip multi-processors//Proceedings of the 12th International Symposium on High-Performance Computer Architecture. Austin, Texas, USA, 2006; 29-40
- [16] Vachharajani M, Vachharajani N, Penry D A, Blome J A, August D I. Microarchitectural exploration with liberty//Proceedings of the 35th Annual ACM/IEEE international Symposium on Microarchitecture. Los Alamitos, CA, USA, 2002; 271-282
- [17] Chidester M, George A. Parallel simulation of chip-multiprocessor architectures. ACM Transactions on Modeling and Computer Simulation, 2002, 12(3): 176-200
- [18] Eeckhout L, Bosschere K D. Efficient simulation of trace samples on parallel machines. Parallel Computing, 2004, 30(3): 317-335
- [19] Poulsen D K, Yew P-C. Execution-driven tools for parallel simulation of parallel architectures and applications//Proceedings of the 1993 ACM/IEEE conference on Supercomputing. New York, USA, 1993; 860-869
- [20] Azevedo A, Nicolau A, Hummel J. An annotation-aware Java virtual machine implementation. Concurrency: Practice and Experience, 2000, 12(6): 423-444
- [21] Krintz C, Calder B. Using annotation to reduce dynamic optimization time//Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation. New York, USA, 2001; 156-167
- [22] Baraz L, Devor T, Etzion O, Goldenberg S, Skaletsky A, Wang Y, Zemach Y. Ia-32 execution layer: A two-phase dynamic translator designed to support ia-32 applications on itanium-based systems//Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture. Washington, DC, USA, 2003; IEEE Computer Society, 2003; 191-201



YAN Qiang, born in 1984, master candidate. His main research interests are computer architecture and compiling optimization.

ZHANG Wei-Hua, born in 1974, Ph. D., assistant professor. His main research interests are computer architecture and compiling optimization.

LIU Li-Li, born in 1985, master candidate. His main research interests are computer architecture and compiling optimization.

ZANG Bin-Yu, born in 1965, Ph. D., professor, Ph. D. supervisor. His current research interests include parallel compiler and computer architecture.

ZHU Chuan-Qi, born in 1943, professor, Ph. D. supervisor. His main research interests include parallel processing and compiling.