

（深圳）

实验报告

开课学期： 2021春季

课程名称： 计算机体系结构(实验)

实验名称： 分支预测器设计

实验性质： 综合设计型

实验时间： 6 地点： T2210

学生班级： 2018级计算机5班

学生学号： 180110505

学生姓名： 胡聪

评阅教师：

报告成绩：

实验与创新实践教育中心印制

2021年4月

# 1. 实验内容

本实验将基于Pin插桩分析的工作机理，使用C/C++实现分支预测器的软件模型，从而在进一步熟悉插桩工具使用的同时，加深对分支预测原理的理解。

设计锦标赛分支预测器；

至少使用2种动态分支预测方法；

只需预测分支跳转方向，不需预测分支目标地址。

# 2. 设计与实现

## 2.1 题目分析

BHT——Branch History Table，用于记录分支历史信息的表格，用于判断一条分支指令是否token，记录跳转信息。简单的BHT只需要用1bit位进行记录，但是这样记录还不够准确，所以一般使用2bit位进行记录，预测时，用分支指令的地址查BHT，获得相应的饱和计数器值。若饱和计数器的最高位为1，预测分支跳转，否则预测分支不跳转。当分支指令的实际跳转方向被确定时，不管预测是否正确，都对BHT中的饱和计数器进行更新，从而达到动态调整分支预测结果的目的。

基于BHT的分支预测方法忽视了分支指令之间的关联性。为此，基于全局历史的分支预测方法在BHT的基础上增加了GHR（Global History Register，全局历史寄存器）来将所有分支指令关联起来。

基于全局历史的分支预测方法使用一个k比特的GHR来记录所有最近k条分支指令的历史跳转方向，并使用PHT（Pattern History Table，模式历史表）来记录各分支指令的分支历史。其中，PHT的结构类似于BHT。

预测时，首先将分支指令的地址和GHR进行异或，再用异或操作的结果来查PHT，然后根据PHT当前行的分支历史和分支目标地址，对该分支指令的分支跳转方向和分支目标地址进行预测。

基于局部选择历史的选择方法使用LSHT（Local Selection History Table，局部选择历史表）来记录子预测器预测结果的历史选择情况。

LSHT一般具有4096条记录，每条记录均包含Tag和局部选择历史2个字段。其中，Tag字段是分支指令地址的一部分，局部选择历史字段则是2bit的饱和技术器，其作用等同于GSHR。

预测时，先取分支指令的地址查LSHT，得到相应的选择历史LSHT[i]。若LSHT[i]的最高位为0，则输出子预测器1的预测结果；否则输出子预测器2的预测结果。当分支指令的实际跳转行为被确定时，需要同时对子预测器和LSHT[i]进行更新。对于子预测器，根据分支指令的实际跳转行为和锦标赛预测结果等信息，使用子预测器自身的更新策略进行更新。

## 2.2 实验过程

// Pin calls this function every time a new instruction is encountered

void Instruction(INS ins, void \*v)

{

    if (INS\_IsControlFlow(ins) && INS\_HasFallThrough(ins))

    {

        INS\_InsertCall(ins, IPOINT\_TAKEN\_BRANCH, (AFUNPTR)predictBranch,

                       IARG\_INST\_PTR, IARG\_BOOL, TRUE, IARG\_END);

        INS\_InsertCall(ins, IPOINT\_AFTER, (AFUNPTR)predictBranch,

                       IARG\_INST\_PTR, IARG\_BOOL, FALSE, IARG\_END);

    }

}

在已经给定的实验框架中，搭建好了Pin插桩的指令代码框架，这部分代码从目标程序中选择控制流指令，然后在分支成功处和失败处插入分析代码。分析代码获取分支指令地址，调用分支预测器模型进行分支预测，并且记录模型的预测结果数据。

BHT：预测时，使用分支指令地址查询BHT，获得相应的饱和计数器值。最高位为1则分支跳转，否则不跳转。当跳转方向确定，无论预测是否正确，均对饱和计数器进行更新，实现动态调整分支预测结果。

GHR：预测时，首先将分支指令的地址和GHR进行异或，再用异或操作的结果来查PHT，然后根据PHT当前行的分支历史和分支目标地址，对该分支指令的分支跳转方向和分支目标地址进行预测。指令跳转时：

    BOOL predict(ADDRINT addr)

    {

        UINT64 tag = cut(addr ^ GHR.getVal(), L);

        return bhist[tag].isTaken();

    }

锦标赛预测分支，基于局部选择历史的选择方法：

BOOL predict(ADDRINT addr)

    {

        if (!GSHR.isTaken())

        {

            return BPs[0]->predict(addr);

        }

        else

        {

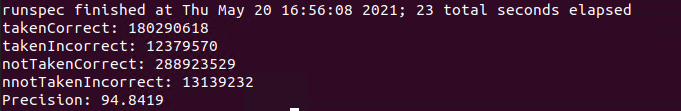
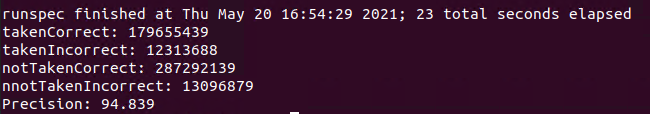
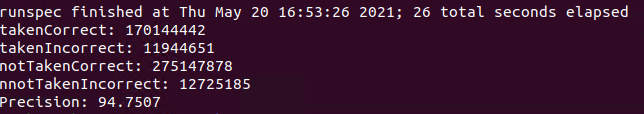
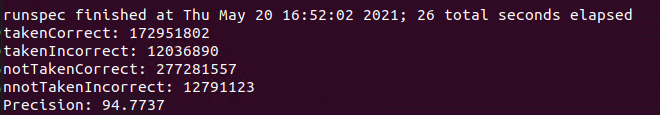
            return BPs[1]->predict(addr);

        }

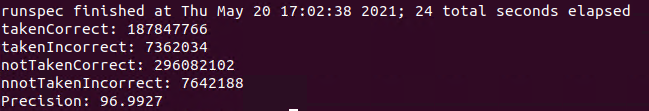
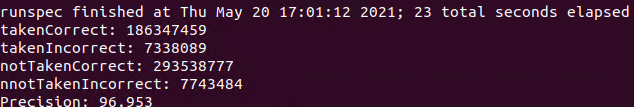
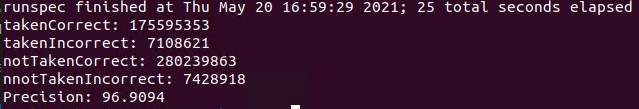
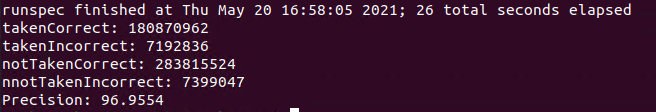
    }

## 2.3 实验结果及分析

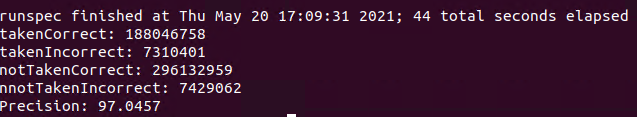
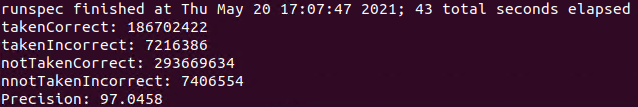
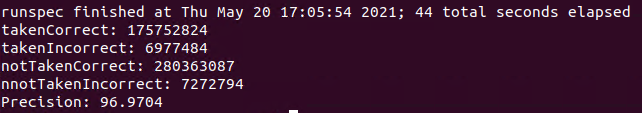
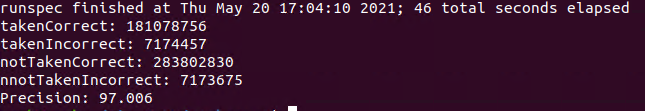
BHT



GHR



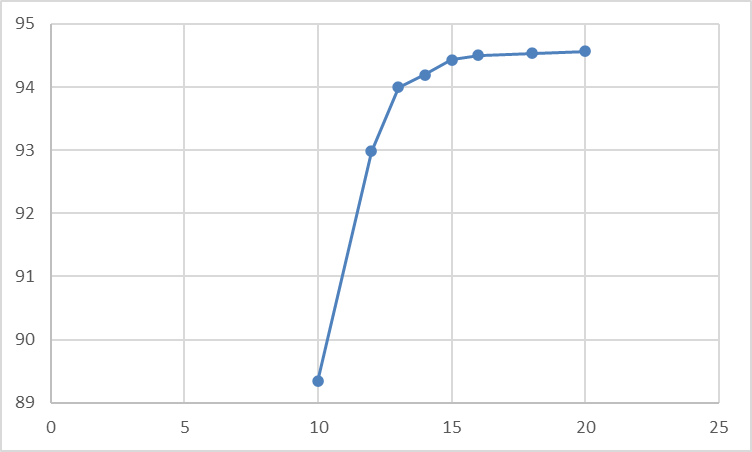
锦标赛



结果表格

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | bzip2 | sjeng | wrf | sphinx3 |
| BHT预测器 | 94.7737 | 94.7507 | 94.839 | 94.8419 |
| GHR预测器 | 96.9554 | 96.9094 | 96.953 | 96.9927 |
| 锦标赛预测器 | 97.006 | 96.9704 | 97.0458 | 97.0457 |

通过改变L的值，可以找到准确率与L值之间的关系：



通过观察最终的实验结果，对bzip2、sjeng、wrf和sphinx3算法进行测试的结果中，同一预测算法的结果大致相同，而全局分支预测器的准确率较高，锦标赛分支预测器的结果也是接近全局分支预测器的，这比较符合最初的预期。

# 3. 总结和感想

在本次实验过程中，首先学习了各个动态分支预测器的原理，以及各种寄存器的使用方式，通过在实验过程中对于参数的调节，一步一步慢慢改善预测算法的准确性，锻炼了自己对于问题的处理能力和调试能力。