体系结构项目报告

1 处理器预测算法综述

1.1 分支预测器的背景

处理器分支预测算法是为了提高流水线面对条件分支指令时候的性能。早期的流水线设计策略是，如果当前指令为条件分支指令，那下一条指令会被**阻塞**在流水线上，直到分支指令进行到**执行**阶段。分支预测器的目的是，在不知道分支指令的目标时，投机地立即执行一个分支。如果预测错误，则放弃所有中间结果并重新执行正确的分支。

1.2 分支预测与分支目标预测

考虑五级流水线系统，确定一条指令是分支指令只有在两个指令周期以后，即译码阶段才能实现。只有当确定一条指令是分支指令时，才能使用分支预测器，效率很低。分支目标预测(Branch Target Prediction)可以解决这个问题。

分支目标预测以当前指令地址作为输入，通过BTB，即分支目标缓存(Branch Target Buffer)来获取跳转地址。BTB中只保存跳转指令的地址，若输入PC在BTB中能找到，则是一个跳转指令；如果找不到，则不是。

整个分支预测系统的工作流程如下：

1. 在**取址阶段**，把PC值传入BTB进行寻址，如果命中，则返回跳转地址；
2. 之后利用分支预测器进行**二次验证**，假如预测结果是跳转(taken)，则执行BTB的返回指令；如果是不跳转(not taken)，则放弃BTB的返回值，执行PC+4的指令。

1.3 分支预测算法的历史

分支预测算法的发展趋势是从简单到复杂，从**静态**到**动态**，以在预测准确率和预测代价之间做到折中和平衡(tradeoff)。

1.3.1 静态预测算法

静态预测算法的代价很小，但是较低的准确率会造成流水线的性能下降。

- **静态**的含义是，预测时不依赖以往的运行结果，每一次执行相同的指令总会返回一个相同的预测结果
- CPU在**译码阶段**进行预测，也即能确定一条指令是条件分支的时候，作出预测判断，比如：阻塞流水线，永远返回token，永远返回not taken。
- 因为CPU在**译码阶段**才会预测，因此插入一条永远会执行的分支延迟间隙(branch delay slot)

1.3.2 动态预测算法

如上所述，**动态预测**算法会参考当前的分支指令之前的**指令的执行历史**。动态预测算法有很多，以下先对动态预测算法的基础进行叙述，之后在Gem5的背景下具体描述算法流程

- 动态预测器主要有两个基本的逻辑，一个是之前描述过的、用于缓存跳转地址的BTB，另一个是**分支历史表**(Branch History Table，简称BHT)，用于保存当前指令的执行历史。
- BHT利用PC进行索引，对应的值是跳转历史。跳转历史的保存形式主要是位。比如1bit预测算法即用1表述跳转，0表述不跳转。
- 动态预测算法的发展主要有两个方向：
  1. 改进跳转历史的保存形式：比如使用2bit或更多位来保存历史
  2. 使用多级预测，即针对不同的情况选择不同的历史进行预测

2 Gem5预测算法分析与实验

2.1 2bit_local算法

2.2 tournament算法

3 预测算法实践