计算机体系结构

Lab 1 内存缓存的设计与性能分析

Assigned Mar.23,2018 Worth XX% Cost:1 Week

http://cs.hit.edu.cn

## Summary

内存瓶颈是现代处理器设计中最重要的问题之一。正如我们在计算机性能发展的趋势中看到的那样，虽然处理器的计算速度在过去的 30 年中呈指数级增长，但由于存储器设备已经开始针对容量而不是延迟进行优化，所以主存（内存）速度相比之下仅呈线性增长。由于内存相对于处理器的速度而言是非常缓慢的，而且内存访问也非常频繁（例如每次加载指令），所以在当前情况下，一个主要的研究架构的领域就是如何改善存内存的感知延迟。

为了提高内存访问的感知延迟，我们作为架构师必须首先了解内存访问的性质。 软件对存储器的访问展现出高度的时间和空间局部性，这是架构师皆知道的经验。时间局部性意味着访问的内存位置很可能在不久的将来会被再次访问。空间局部性意味着在不久的将来可能访问刚被访问过的存储器附近的存储器位置。

在这个实验中，我们将探讨内存缓存的概念，这是一种提高内存延迟的常见且高效的机制。缓存是一个快速的，但必要的小内存，包含最近访问过的内存区域。高速缓存的访问和更新延迟比内存低得多；因此，如果 CPU 所需访问的存储器位置存在于高速缓存中，则处理器可以只花费较少量周期（与访问主存储器所花费的数百个周期相对）就能获得数据，由此大幅改进了处理器性能。我们最终通过其有效性来衡量这种高速缓存的性能，也就是它能够满足处理器请求的时间百分比，因此经常使用诸如写命中百分比和读命中百分比这种统计数据来评估高速缓存的性能。

在这个实验中，我们将使用 Pin 和 SPEC 基准测试程序来评估各种缓存组织的行为。

同之前的实验一样，这个实验需要你单独完成，你可以通过阅读附带的 Pin 指导内容来学习使用，或与同学讨论你遇到的问题。但是请注意按时提交你的成果，最终程序将会通过查重系统的检测。

## Setting Up

请从实验平台下载实验模板 CacheModel.h 和 testCache.cpp，并在 Linux 环境下打开进行编辑（在 lab0 中我们提及到，在 Windows 环境下进行编程及保存可能会改变文件信息格

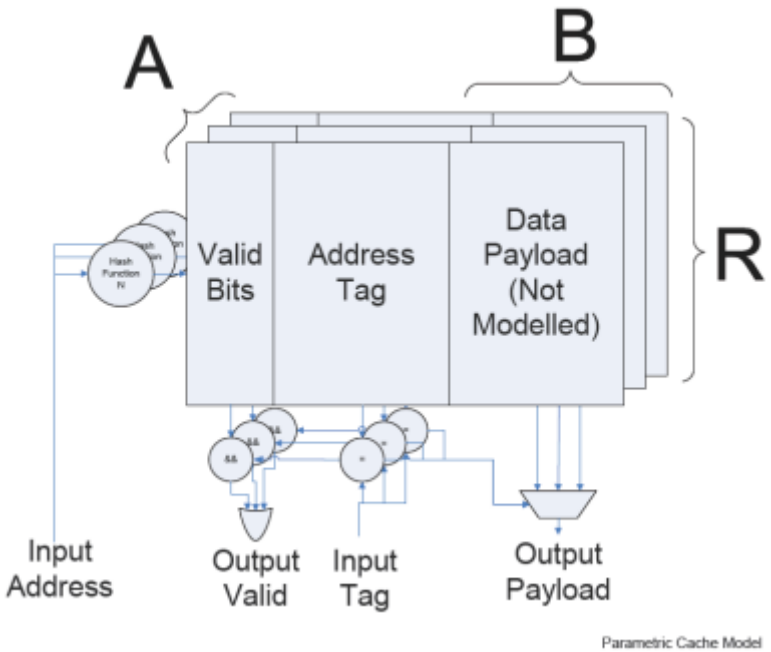
式，造成无法使用）。另外，lab1.cpp 为本次实验的支持文件，这个 Pin 工具仅供你测试使用，不需要修改。

请将本次实验所有文件放置在你的 Pin 路径下的/source/tools/ManualExamples 中，这样生成的.so 文件就在 obj-intel64 中以便使用。请不要忘记在 makefiles.rules 中添加这个工具的引用，才能正常编译。其余设置均和 lab0 保持一致。

**Lab Task**

在本次实验中，你需要完成一个高速缓存的模型。在这个缓存模型中会使用到的参数有：关联度、缓存行数以及块大小。它们决定了这个缓存应该如何构造，你所实现的缓存模型必须能根据这些参数的改变来自动调整（Tips：可以视为是 CacheModel 模型类的构造函数参数）。我们可以将一个缓存视为一个有 R 行（或 R 个集合）的数据数组，每个行包含 A 个大小为 B 的数据块和关于 B 的一些元数据（标记，有效位），这些块之间存在集合关联或组相联方式。当数据数组接收到一个地址，它输出其对应的块数据。地址可以分解为<tag> <row 或 index> <block offset>。

你将根据图设计一组参数化的 L1 数据缓存模型。其中，缓存的替换策略由你自己决定。本次实验的结果是开放式的，你可以尽可能提高缓存的最终命中率。




有了以上所有的背景，我们可以讨论位于 CacheModel.h 中的代码。所有你将要使用的缓存都将来自 CacheModel 基类，因此，请注意设计好你的这个基类。 请注意，基类为你实现了一些缓存元数据，但你必须添加一些自己的元数据，并且不要更改已经存在的变量或函数名。以下是 CacheModel 函数的描述。你需要为它们实现大胆的功能：

readReq：这个函数是从处理器到缓存的读请求。处理器将呈现一个虚拟地址。

然后，你将根据缓存类型更新适当的缓存元数据。

writeReq：这个函数是从处理器到缓存的写入请求。处理器将呈现一个虚拟地址。然后，你将根据缓存类型更新适当的缓存元数据。

dumpResults： 此函数将打印出 readReq 和 writeReq 已经跟踪的缓存统计信息。在我们提供的代码中，我们将在 Pin 运行结束时调用这个函数。这部分代码不需要你修改。

CacheModel 基类有许多构造函数参数，即前文所述的 associativityParam，logNumRowsParam 和 logBlockSizeParam。你将使用这些参数来实现你的功能，以便可以快速测试多个缓存大小和关联性。我们使用 logVariable 命名约定来表示该参数是以实际大小的 2 的对数给出的，因此如果 logBlockSizeParam 是 4，那么块大小是 2 ^ 4 = 16 个字节。请注意这一约定。

你实现的这些函数将跟踪读访问总数，写访问总数，读命中数和写命中数。本次实验将根据这些统计数据进行评估。

### Submit

请于实验系统要求的时限之前提交本次实验要求的 CacheModel.h 与 testCache.cpp，并在报告提交的位置上传你的实验报告。

### Addition

从基本配置（行数= 512，块大小= 4 字节，关联性= 1）开始，改变缓存模型的行数，块大小，关联性和容量。 你能解释一下你所观察到的所有命中和失败率的一般趋势吗（显示图表）。 三种缓存模型之间是否存在显着差异？ 为什么？

### Advice

我们的代码或基础架构中可能存在错误。如果你发现任何“有趣的”或“意外的”行为，这可能是我们提供的代码或基础架构中的问题。请立即将这些错误报告给技术援助。