|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程编号 1502760001-01  题目类型 实验5 | |  |  |  | | --- | --- | --- | | **得分** | **教师签名** | **批改日期** | |  | 冯禹洪 |  | |

**深 圳 大 学 实 验 报 告**

**课程名称： 计算机系统(2)**

**实验项目名称： Cache实验**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 冯禹洪**

**报告人： 叶茂林 学号： 2021155015 班级： 腾班**

**实验时间： 2023年6月6日至6月19日**

**实验报告提交时间： 2023年6月19日**

**教务处制**

|  |
| --- |
| **一、实验目的：**   1. 加强对Cache工作原理的理解； 2. 体验程序中访存模式变化是如何影响cache效率进而影响程序性能的过程； 3. 学习在X86真实机器上通过调整程序访存模式来探测多级cache结构以及TLB的大小。 |
| **二、实验环境**  X86真实机器。 |
| **三、实验内容和步骤**  **1、分析Cache访存模式对系统性能的影响**   * 1. 给出一个矩阵乘法的普通代码A，设法优化该代码，从而提高性能。   2. 改变矩阵大小，记录相关数据，并分析原因。   **2、编写代码来测量x86机器上（非虚拟机）的Cache 层次结构和容量**   1. 设计一个方案，用于测量x86机器上的Cache层次结构，并设计出相应的代码； 2. 运行你的代码获得相应的测试数据； 3. 根据测试数据来详细分析你所用的x86机器有**几级Cache**，**各自容量**是多大？ 4. 根据测试数据来详细分析**L1 Cache行**有多少？   **3、尝试测量你的x86机器TLB有多大？（选作）** |
| **四、实验结果及分析**  **1、分析Cache访存模式对系统性能的影响**  如图1所示，分析未进行优化的原始代码可知，在进行矩阵乘法的时候，对矩阵b进行元素访问的时候是按照列取元素相乘，而Cache访存模式是取出局部顺序存取数据备用，所以会比较慢。    图1  所以可以修改一下代码，如图2所示，将b矩阵存储改为转置矩阵存储，这样在对矩阵b进行元素访问的时候就可以按照行取元素相乘，从而利用cache顺序存取的特点。    图2  将优化后的代码文件另存为optimization.c文件保存好。下面开始测试。  首先运行未优化的矩阵乘法代码，测试结果如图3所示。    图3  然后运行优化过的矩阵乘法代码，测试结果如图4所示。    图4  具体数据如表1所示，其中加速比=优化前系统耗时/优化后系统耗时，加速比越高，表明优化效果越明显。  表1普通矩阵乘法与及优化后矩阵乘法之间的性能对比    对比图如图5所示，可见随着数据规模的增大，矩阵乘法优化的效果越明显。  图5  加速比的变化情况如图6所示，可知随着矩阵的增大，优化的效果越明显。  图6  **分析原因：**  在计算矩阵乘法时，按行读取元素比按列读取元素更快的原因与CPU缓存有关。  当我们按行读取矩阵时，由于相邻的元素在内存中是连续存放的，因此这些元素都能被缓存在同一个缓存行中。因此，当我们读取第一个元素时，它会将其余元素所在缓存行一起加载到CPU缓存中，这样一来，当我们接下来需要读取这些元素时，它们已经位于缓存中了，不需要再次从内存中读取，因此可以大大提高运算速度。  相比之下，按列读取矩阵时，由于相邻的元素并不在同一个缓存行中，因此每次读取元素时都需要从内存中读取，这会导致频繁的访存操作，从而降低运算速度。  因此，按行读取矩阵能够更充分利用CPU的缓存机制，提高矩阵乘法的计算效率。  **2、测量分析出Cache 的层次结构、容量以及L1 Cache行有多少？**   1. **实验原理**   现代CPU通常拥有三级缓存，分别是L1缓存、L2缓存和L3缓存。L1缓存是CPU中距离处理器最近的一个缓存，由于其接近CPU，因此访问速度非常快。相比之下，L2和L3缓存则较远离CPU，因此访问速度会慢一些。   1. **测量方案及代码**    1. **测量cache的层次结构和容量**   我们可以通过测量随机访问不同大小的内存空间的时间来测量cache的层次结构和容量，具体来说，我们先准备好一系列大小的内存空间，如图7所示。    图7  然后，进行内存随机访问一千万次，记录访问的时间，如图8所示。    图8   * 1. **测量L1Cache行大小**   在L1的高速缓存中以不同的步长顺序访问内存，记录访问的时间，具体来说，先设定好高速缓存的大小和不同的步长，如图9所示。    图9  然后以不同的步长顺序访问L1的高速缓存空间，记录访问的时间，如图10所示。    图10   1. **测试结果**    1. Cache层级结构和容量测试结果如图11所示。     图11   * 1. ② Cacheline缓存行大小测试结果如图12所示。     图12   1. **分析过程；**   随机访存的耗时结果如图13所示，从中可以看出L1缓存的大小大概在512KB左右，L2缓存的大小大概在4MB到8MB之间，L3缓存的大小大概在16MB到18MB左右。    图18  不同步长的访问耗时结果如图14所示，当步长在cache line之外时访问耗时增加，由图可知，L1的缓存行大小大概在64B到96B之间。    图14   1. **验证实验结果**   如图15所示，查看任务管理器的CPU参数可知，L1缓存的大小为512KB，L2缓存大小为4.0MB，L3缓存的大小为16.0MB，这与我们测量的基本相符。    图15  如图16所示，使用CPU-Z软件查看我们的电脑CPU缓存可知，L1缓存行的大小为64B，这也与我们测量出的结果基本相符。    图16 |
| **五、实验结论与心得体会**  本次实验的目的是加强对Cache工作原理的理解，体验程序中访存模式变化是如何影响cache效率进而影响程序性能的过程，以及学习在X86真实机器上通过调整程序访存模式来探测多级cache结构。  在本次实验中，我们首先对矩阵乘法的代码进行了优化，我们发现，随着矩阵规模的增大，按列读取矩阵时，由于相邻的元素并不在同一个缓存行中，所以每次读取元素都需要从内存中读取，导致频繁的访存操作，从而降低了计算速度。因此，我们可以改成按行读取元素，当我们按行读取矩阵时，由于相邻的元素在内存中是连续存放的，因此这些元素都能被缓存在同一个缓存行中。因此，当我们读取第一个元素时，它会将其余元素所在缓存行一起加载到CPU缓存中，这样一来，当我们接下来需要读取这些元素时，它们已经位于缓存中了，不需要再次从内存中读取，因此可以大大提高运算速度。  接着，我们编写了测量Cache层次结构和容量的代码，并运行该代码以获取相应的测试数据。我们根据测试数据分析所使用的X86机器有几级Cache，各自容量是多少。通过分析和验证后，我们可以确定L1 Cache的容量为512KB，L2 Cache的容量为4MB，L3 Cache的容量为16MB。最后，我们确认了L1 Cache的行大小为64B。  通过本次实验，我们深入了解了Cache的工作原理及运作方式，并探究了程序中访存模式变化对Cache效率和性能的影响。此外，我们还学会了在X86真实机器上通过编写程序来访存探测多级cache结构以及缓存行的大小，这对于进一步深入理解计算机体系结构以及代码优化具有重要意义。 |

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2023年 月 日 |
| 备注： |