《计算机组成原理》第十次作业

信息安全 胡博浩 2212998

5.8

1)

地址以字地址形式给出，每个32位数据块包含四个字。因此，每四次访问就会有一次未命中，即缺失率为1/4。所有未命中都是强制未命中，未命中率对cache容量或工作集大小并不敏感。但是，它对访问模式和块大小很敏感。

2)

cache块大小为16、64、128字节时，缺失率分别为 1/2、1/8 和 1/16。该负载利用的是空间局部性原理。

3)

在这种情况下，如果不包括首次访问时的强制未命中，由于预取缓冲区总是为下一个请求做好准备，所以缺失率为0。

5.18

1)

考虑最坏的情况，需要 $2^{43-12} = 2^{31}$ 页表项，

物理存储空间需要 $2^{31} \times 4$字节 $= 2^{33}$字节 $= 8GB$ 。

2)

如果可以选择每个页表段的大小，则只需要两级页表。在多级页表中，读取PTE需要访问表的每一级。

3)

是的，如果将段表项假定为段页的物理页码，并保留一位作为有效位，那么每个段表项的有效范围为 2^{31} * 4KB = 8TB，足以覆盖机器的物理地址空间（16GB）。

4)

每个页表级包含4KB/4B = 1024个页表项，可转换为log2(1024) = 10 位虚拟地址。因此使用了43位虚拟地址和4KB页，则需要(43 - 12)/10) = 4 级页表。

5)

在反向页表中，PTE的数量可以减少为哈希表的大小加上碰撞的代价的总和。在这种情况下，处理TLB缺失需要额外的引用来比较哈希表中存储的一个或多个标记位。

5.23

模拟不同的ISA需要对ISA的API进行特定处理。每个ISA在执行指令、中断、捕获到内核模式等过程中都有特定的行为，因此必须进行仿真。这可能需要执行比目标ISA最初所需更多的指令来模拟每条指令。这会导致性能大幅下降，并且难以与外部设备正常通信。

如果能对仿真代码进行动态检查和优化，仿真系统的运行速度可能会比在本地ISA上更快。例如，如果底层机器的ISA有一条指令可以处理执行仿真系统的几条指令，那么就有可能减少执行指令的数量。这与最近英特尔处理器的微操作融合类似，可以用较少的指令处理多个指令。

5.27

1)

srcIP和 refTime字段。平均每项有两次cache缺失。

2)

将srcIP和refTime字段分组到一个单独的数组中(即创建两个并行数组，一个包含 srcIP 和 refTime，另一个包含其余字段）。

3)

peak_hour (int status); // 给定状态的峰值时间

将srcIP、refTime和状态合并在一起。

6.4

1)

如下图所示，循环的每次迭代需要16个周期，循环运行了999次。因此，运行的周期总数为16 × 999 + 3 = 15984。

```
1         li   $s0, 8000
2         add  $s1, $a0, $s0
```

```
3          addi $s2, $a0,  16
4  loop:   l.d  $f0, -16($s2)
5          l.d  $f2, -8($s2)
6          stall
7          stall
8          stall
9          stall
10         stall
11         stall
12         add.d $f4, $f0, $f2
13         stall
14         stall
15         stall
16         stall
17         s.d  $f4, 0($s2)
18         addi $s2, $s2, 8
19         bne  $s2, $s1, loop
```

2)

下面的代码删除了每次迭代中的一次阻塞

```
1          li    $s0, 8000
2          add   $s1,  $a0, $s0
3          addi  $s2,  $a0, 16
4  loop:   l.d   $f0, -16($s2)
5          l.d   $f2, -8($s2)
6          stall
7          stall
8          stall
9          stall
10         stall
11         stall
12         add.d $f4, $f0, $f2
13         addi  $s2, $s2,   8
14         stall
15         stall
16         stall
17         s.d   $f4, 0($s2)
18         bne   $s2, $s1, loop
```

因此，新的代码需要15×999=14958个周期

3)

数组元素D[j]和D[j-1]存在循环相关性，因为在第i次迭代中载入D0的值是在第i-1次迭代中产生的。

4)

```
        li   $s0, 8000
        add  $s1, $a0, $s0
        addi $s2, $a0,  16
        l.d  $f0, 0($s2)
        l.d  $f2, 8($s2)
        addi $s2, $s2, 8
        stall
        stall
        stall
        stall
        stall
loop:   add.d $f4, $f0, $f2
        addi  $s2, $s2,  8
        mov.d $f0, $f2
        mov.d $f2, $f4
        stall
        s.d   $f2, 0($s2)
        ble   $s2, $s1,loop
```

每次循环需要7个周期，运行999次。因此，总的周期数为$7 \times 999 + 10 = 7003$。

5)

```
        l.d   $f0, 0($s2)
        l.d   $f2, 8($s2)
        li    $s0, 8000
        add   $s1, $a0, $s0
        addi  $s2, $s2,16
        stall
        stall
        stall
loop:   add.d $f4, $f0, $f2
        stall
        stall
        stall
        stall
        add.d $f0, $f4, $f2
        s.d   $f2, 0($s2)
        stall
```

```
        stall
        stall
        add.d $f2, $f4, $f0
        s.d   $f0, 0($s2)
        addi $s2, $s2,24
        stall
        stall
        s.d  $f2,-8($s2)
        bne  $s2,$s1,loop
```

循环展开后每次循环需要17个周期，但只需要循环333次。因此，总的周期数为17 × 333 + 10 = 5671。

6)

会导致包括两个循环的情况： 未展开的循环和原始循环。假设你将循环展开U次，运行未展开的循环，直到剩下的迭代次数小于U，即for (i = 0; i + U < MAX; i+= U)。此时，切换到未展开的循环，即for (; i < MAX; i++) 。

7)

即使消息传递系统没有延迟，也不可能利用消息传递来提高性能。并行处理的工作不足以由于使用多个CPU而提高性能。所有可以并行处理的工作都可以安排在从属的浮点指令之间进行。

6.6

1)

涉及的计算有 (m × p × n) 乘法和 (m × p × (n- 1)) 加法。与C中单个元素相关的乘法和加法是相互依赖的，并且在有两个乘积之前，我们不能开始对一个元素的乘法结果求和。因此，在本题中，加速比应该非常接近4。

2)

由于四个核都在处理映射到同一cache行的不同矩阵元素，因此会造成cache缺失，从而影响速度。每次更新都会产生缓存缺失，因此加速比会降低，降低幅度是缓存缺失代价的 3 倍。

3)

解决伪共享问题的最简单方法是通过跨列而不是跨行遍历矩阵（即使用索引j而不是索引i）来计算C中的元素。这些元素将被映射到不同的cache行中。然后，我们只需确保在同一个核上处理计算出的矩阵索引（i, j）和（i + 1, j）。因此消除了伪共享问题。