《计算机组成原理》第十次作业

网络空间安全学院 信息安全 陆皓喆 2211044

5.8

播放音频或视频文件的多媒体应用是一类被称为“流”的负载的一部分，即取回大量的数据，但是大部分数据都不会再使用。考虑一个视频流负载依次访问一个$512KiB$的工作集的情况，地址流如下：

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \ldots$$

5.8.1

题目

假设有一个$64KiB$的直接映射$cache$，$cache$块大小为32字节。那么对于上面的地址流，缺失率是多少？当$cache$容量或者工作集大小变化时，$cache$的缺失率如何随之变化？根据$3C$模型，这些缺失如何被分类？

解答

地址以字地址形式给出;每个32位块包含四个字。因此，每四次访问将是一次失败(即，失败率为$\frac{1}{4}$)。所有失误都是强制失误。缺失率对缓存大小或工作集大小不敏感。然而，它对访问模式和块大小很敏感。

5.8.2

题目

当$cache$块大小分别为16字节、64字节和128字节时，重新计算缺失率。该负载所采用的是哪种局部性？

解答

脱靶率分别是$\frac{1}{2}$、$\frac{1}{8}$和$\frac{1}{16}$。工作负载正在利用空间局部性。

5.8.3

题目

“预取”技术：当一个特殊$cache$块被访问时，利用可预测的地址模式推测地取回其他$cache$块。预取的一个例子是流缓冲区，当一个特定的$cache$块被取回时，将与其相邻的$cache$块也依次预取回一个独立的缓冲区中。如果所需的数据在预取缓冲区中，那么看成是一次命中并且将数据移入$cache$，同时预取下一个$cache$块。假设一个流缓冲区有两项，并且假设$cache$延迟满足在先前$cache$块的计算完成之前可以加载下一个$cache$块。那么对于上面的地址流，缺失率是多少？

解答

在这种情况下，不包括第一次访问时的强制错过，错过率为0:预取$buff$总是准备好下一个请求。

5.18

在本练习题中，我们将研究页表的空间/时间优化。下表是一个虚拟存储器系统的参数。

虚拟地址（位）	物理DRAM	页大小	PTE大小（字节）
43	$16GB$	$4KB$	4

5.18.1

题目

对于一个单级页表，需要多少页表项（PTE）？存放页表需要多少物理存储空间?

解答

最坏的情况是$2^{43-12}=2^{31}$个条目，需要$2^{31}\times4$字节=$2^33=8GB$。

5.18.2

题目

通过仅在物理存储器中保存活跃的PTE，多级页表可以降低消耗的物理存储空间。如果不限制段表（高一级页表）的大小，那么需要多少级的页表？如果TLB缺失，那么地址转换需要访问多少次存储器?

解答

只有两个级别，设计器可以选择每个页表段的大小。在多级模式中，读取PTE需要访问表的每个级别。

5.18.3

题目

假设将段限制为$4KiB$页大小的倍数（从而可以分页）。对于所有的页表项（包括在段表中的项），4字节大小是否足够大?

解答

是的，如果假设段表项是段页面的物理页码，并且保留一个位作为有效位，那么每个表项的有效范围为$(2^{31})\times4KiB=8TiB$，这足以覆盖机器的物理地址空间($16GiB$)。

5.18.4

题目

如果将段限制为$4KiB$页大小的倍数，那么需要多少级页表？

解答

每个页表级别包含$\frac{4KiB}{4B} = 1024$项，因此转换$log_2^{1024} = 10$位虚拟地址。使用43位虚拟地址和$4KiB$页面，我们需要装载$((43-12)/10) = 4$级转换。

5.18.5

题目

反向页表可以用来进一步优化空间和时间。存放页表需要多少PTE？假设实现一个哈希表，当TLB缺失时，在正常情况下和最差情况下的存储器访问次数分别是多少？

解答

在反向页表中，pte的数量可以减少到哈希表的大小加上冲突的代价。在这种情况下，提供$TLB\ miss$需要一个额外的引用来比较存储在哈希表中的标签。

5.23

题目

5.6节讨论了虚拟化，假设虚拟化的系统和底层硬件运行相同的ISA。然而，虚拟化的一种可能用途是对非本地的ISA进行仿真。$QEMU$就是这样一个例子，可以用来仿真多种ISA，如$MIPS$、$SPARC$以及$PowerPC$。与这种虚拟化相关的难点是什么？被模拟的系统可能比在本地ISA上运行得更快吗？

解答

模拟不同的ISA需要对该ISA的API进行特定的处理。每个ISA都有特定的行为，这些行为将在指令执行、中断、捕获内核模式等情况下发生，因此必须对其进行模拟。这可能需要为了模拟每条指令而执行的指令比最初在目标ISA中所需要的要多。这可能会导致性能大幅下降，并使其难以与外部设备进行正确通信。如果可以动态检查和优化仿真代码，则仿真系统可能比在其本机ISA上运行得更快。例如，如果底层机器的ISA有一条指令可以处理多个仿真系统指令的执行，那么执行的指令数量可能会减少。这类似于最近的英特尔处理器做微操作融合，允许几个指令被更少的指令处理。

5.27

本练习题给出了网络服务器日志的定义，并且考察了代码优化以改进日志处理速度。日志的数据结构定义如下：

```
struct entry{
    int srcIP;
    char URL[128];
    long long refTime;
    int status;
    char brower[64];
}log[NUM_ENTRIES];
```

假定日志的处理函数如下：

```
topK_sourceIP (int hour);
```

该函数决定了给定时间内最频繁观察到的源IP。

5.27.1

题目

对于给定的日志处理函数，一个日志项中的哪些字段将被访问？假设$cache$块为64字节，没有预取，那么给定的函数平均每项会引发多少次$cache$缺失？

解答

`srcIP`和`refTime`字段将会被访问。每次缺失两次。

5.27.2

题目

为了改善$cache$的效用和访问局部性，如何重新组织数据结构？

解答

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。一个带有`srcIP`和`refTime`，另一个带有剩余的字段。)

5.27.3

题目

请举例说明另一种不同数据结构的日志处理函数。如果两个函数都很重要，为了改进整体性能，将如何重写程序？用代码片段和数据补充讨论。

解答

```
peak_hour (int status); // peak hours of a given status
```

将`srcIP`，`refTime`和`status`绑定在一起。

6.4

考虑下面的C代码片段：

```
for(j=2;j<1000;j++){
    D[j]=D[j-1]+D[j-2];
}
```

与之对应的$MIPS$代码如下所示：

```
     li $s0,8000
     add $s0,$a0,$s0
     addi $s2,$a0,16
loop:l.d $f0,-16($s2)
     l.d $f2,-8($s2)
     add.d $f4,$f0,$f2
     s.d $f4,0($s2)
     addi $s2,$2,8
     bne $s2,$s1,loop
```

每种指令的延迟如下（以周期为单位）：

add.d	l.d	s.d	addiu
4	6	1	2

6.4.1

题目

执行该代码需要多少周期?

解答

这个问题说明，如果重组串行代码，一些计算可以并行完成。但是，更重要的是，我们可能希望在ISA中提供$SIMD$操作，并在对多个数据项执行相同操作时允许数据级并行性。

如下所示，循环的每次迭代需要16个循环。E循环运行999次。因此，循环总数为$16\times 999+3=15984$。

```
li x5, 8000
add x12, x10, x5
addi x11, x10, 16
LOOP: fld f0, -16(x11)
fld f1, -1(x11)
stall
stall
stall
stall
stall
stall
```

```
fadd . d f2, f0, f1
stall
stall
stall
stall
fsd f2, 0(x11)
addi x11, x11, 8
ble x11, x12, LOOP
```

6.4.2

题目

对该代码重排序以减少阻塞。重排序之后执行该代码需要多少周期？（提示：可以通过改变fsd指令的偏移量来减少额外的阻塞。）

解答

下面的代码每次迭代会移除一个失速:

```
li x5, 8000
add x12, x10, x5
addi x11, x10, 16
LOOP: fld f0, -16(x11)
fld f1, -1(x11)
stall
stall
stall
stall
stall
stall
fadd . d f2, f0, f1
addi x11, x11, 8
stall
stall
stall
fsd f2, -8(x11)
ble x11, x12, LOOP
```

而新的循环需要$15\times 999 = 14958$个周期。

6.4.3

题目

在循环中，如果后面迭代中的指令会依赖于前面迭代指令（同一循环中）产生的结果，我们说循环的迭代存在循环相关性（$loop-carried\ dependence$）。请分析上面代码中的循环相关性，识别其中相关的程序变量和汇编级寄存器。可忽略循环变量j。

解答

数组元素$D[j]$和$D[j-1]$将具有循环携带依赖关系。在迭代i期间加载到$D0$的e值是在迭代$i-1$期间产生的

6.4.4

题目

重写代码，使用寄存器保存迭代之间的数据（与从内存中存储和重载数据相反）。指出代码在哪里阻塞，并计算执行所需的周期数。注意，你会用到汇编伪指令“`mov.drd，rs`”，表示将浮点寄存器rs中的数值写入浮点寄存器rd中。假设`mov，d`的执行需要一个周期。

解答

```
li x5, 8000
add x12, x10, x5
fld f0, 0(x11)
fld f1, 8(x11)
addi x11, x11, 8
stall
stall
stall
stall
stall
LOOP: fadd . d f2, f0, f1
addi x11, x11, 8
fmv . d f0, f1
fmv . d f1, f2
stall
fsd f2, 0(x11)
ble x11, x12, LOOP
```

这个循环需要7个周期，运行999次。因此，循环总数为$7\times 999+10=7003$。

6.4.5

题目

第4章中描述了循环展开。对上述循环进行展开并优化从而使每个展开的循环处理3个之前循环的迭代过程。指出代码在哪里阻塞，并计算执行所需的周期数。

解答

```
fld f0, 0(x11)
fld f1, 8(x11)
li x5, 8000
add x12, x10, x5
addi x11, x11, 16
stall
```

```
stall
stall
LOOP: fadd . d f2, f0, f1
stall
stall
stall
stall
fadd . d f0, f2, f1
fsd f2, 0(x11)
stall
stall
stall
fadd . d f1, f2, f0
fsd f0, 8(x11)
addi x11, x11, 24
stall
stall
fsd f1, -8(x11)
bne x11, x12, LOOP
```

展开的循环需要17个循环，但只运行了333次。因此，循环的总次数为$17\times 333+10=5671$。

6.4.6

题目

因为循环迭代的次数恰好为3的倍数，所以练习题6.4.5中的循环展开运行效率高。如果编译时无法知道要迭代的次数，那么会发生什么？如果总的迭代次数不是循环展开的迭代次数的整数倍，那么该如何有效地处理这些迭代呢？

解答

包括循环的两个副本:展开的循环和原始循环。

假设展开循环U次。运行未展开循环，直到迭代次数小于u(在某种意义上，未展开循环将这样做:`for(i=0;i+U<MAX;i+=U)`。此时，切换到展开的循环。(在某种意义上，你的原始循环将这样做:`for( ;i<MAX;i++)`

6.4.7

题目

考虑将此代码运行在一个具有2个节点的分布式存储器消息传递系统中。假定我们采用6.7节描述的消息传递机制，操作$send(x，y)$可向节点x发送值y，操作$receive()$等待接收传递给它的消息。假定$send$操作的发射需要1个周期（也就是说，同一节点的后续指令可在下个周期执行），而接收节点需要多个周期来接收。$receive$指令会阻塞接收节点上后续指令的执行，直到接收到消息。你能使用这样的系统来加速本例中的代码吗？如果可以，那么可以接收到消息的可容忍最大延迟是多少？如果不行，为什么？

解答

使用消息传递来提高性能是不可能的——即使消息传递系统没有延迟。这里没有足够的工作可以并行完成，从而从使用多个cpu中获益。所有可以并行完成的工作都可以在相关的操作点指令之间进行调度。

6.6

矩阵乘在大量应用中都扮演重要角色。两个矩阵可以相乘的条件是第一个矩阵的列数和第二个矩阵的行数相同。
假设有一个$m \times n$的矩阵A,还有一个$n \times p$的矩阵B与之相乘。乘法结果为一个$m \times p$的矩阵AB(或$A \cdot B$)。如果令$C = AB, c_i, j$代表在矩阵C中(i, j)位置处的值，则$1 \leqslant i \leqslant m$且$1 \leqslant j \leqslant p$,
$c_{i,j} = \sum_{i=1}^{n} a_{i,j} = b_{j,k}$。现在将考虑是否可以将$C$的计算并行化。假设矩阵在存储器中的存放顺序为:
$a_{1,1}, a_{2,1}, a_{3,1}, a_{4,1}, \cdots$。

6.6.1

题目

假设我们分别在单核/四核共享内存的系统计算 C,请问四核相对于单核的预期加速比是多少？可忽略存储器相关的问题。

解答

这个问题提出了一个“令人尴尬的并行”计算，并要求学生和在四核系统上获得的加速。E计算包括:
$(m \times p \times n)$乘法和$(m \times p \times (n-1))$加法。$C$中与单个元素相关的乘法和加法是相关的(在得到两个乘积之前，我们不能开始对一个元素的乘法结果求和)。所以在这个问题中，加速应该非常接近4。

6.6.2

题目

如果对C的更新会导致$cache$缺失(例如更新一行中连续的元素时可能引起伪共享),重新计算练习题$6.6.1$中的问题。

解答

他的问题是关于如何加速是一个由于四个核心都工作在事件矩阵元素映射到相同的缓存线造成的缓存丢失。每次更新都会产生缓存缺失的成本，因此会将获得的加速降低为缓存缺失服务成本的3倍。

6.6.3

题目

有什么办法消除可能出现的伪共享问题？

解答

在这个问题中，我们被问到如何x这个问题。解决虚假共享问题的最简单方法是通过跨列而不是行遍历矩阵来计算C中的元素(即使用$index - j$而不是$index - i$)。Ese元素将被映射到不同的缓存线。我们只需要确保我们处理的矩阵索引是在同一个核上计算的(i, j)和$(i + 1, j)$这将消除虚假分享。