分布式系统作业

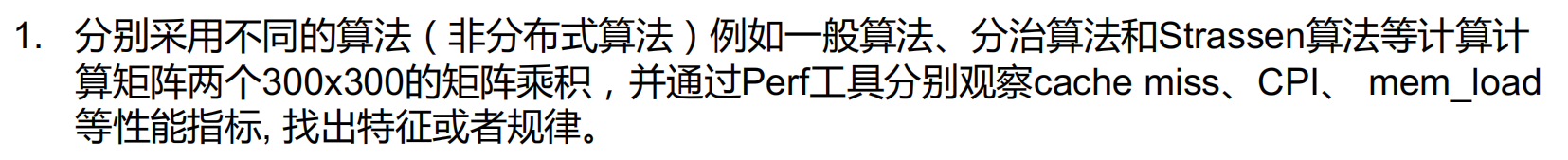
第2次作业

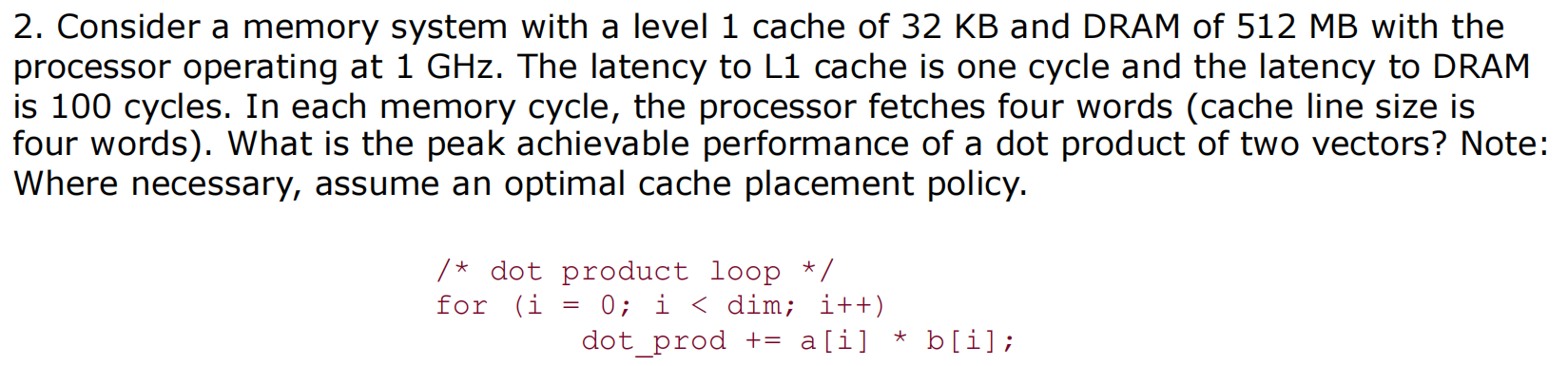
姓名：唐晨轩

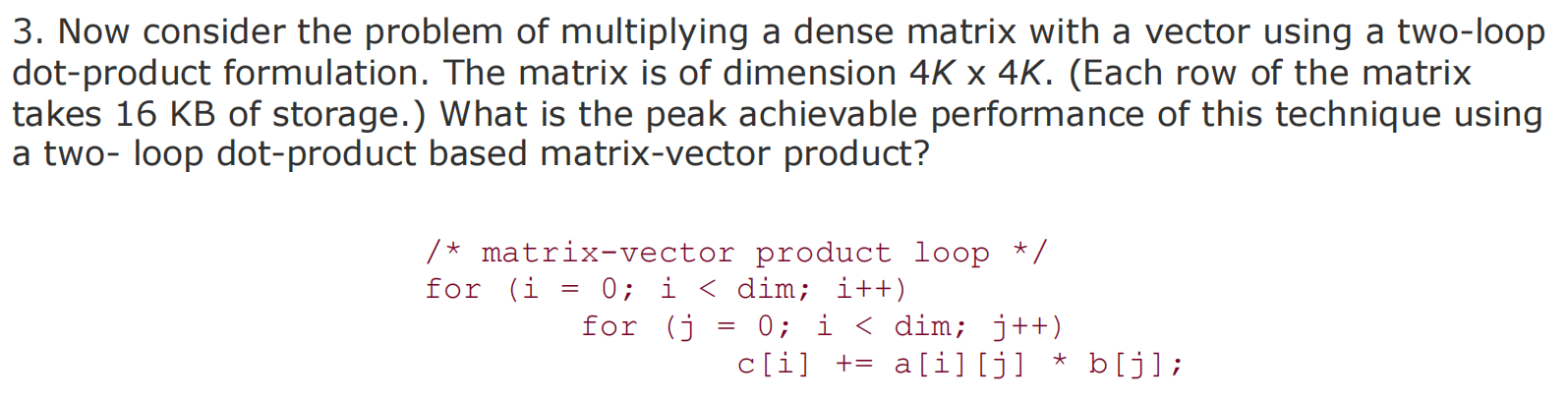
班级：人工智能与大数据

学号：19335182

1. 问题描述



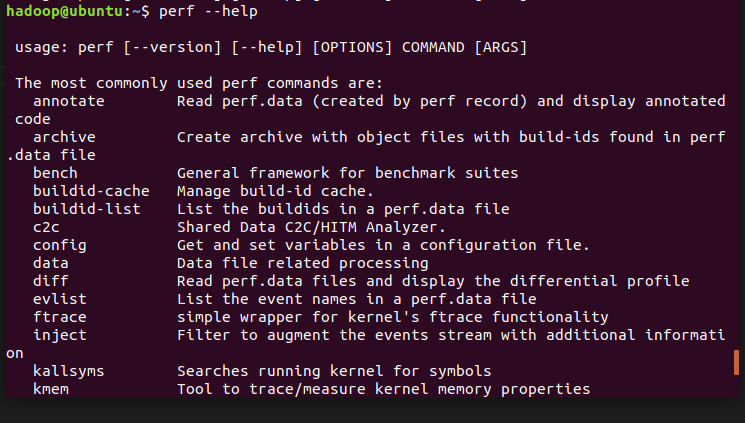




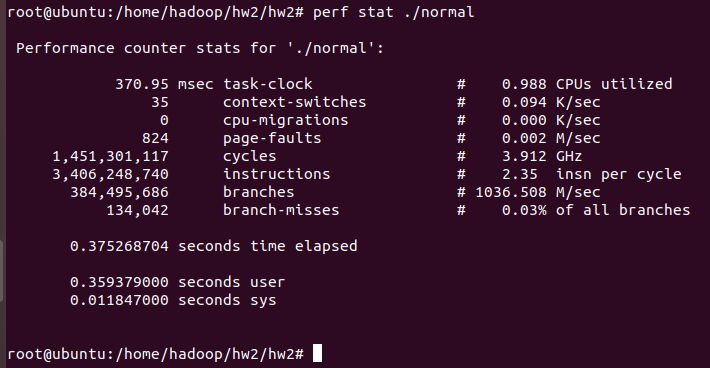
1. 解决方案
2. 根据传统算法，两个n阶矩阵相乘，对于n2个元素，每个元素想要被计算出来，至少要进行n次乘法和n-1次加法，算法复杂度达到O(n^3)。
3. 这里的分治算法是利用矩阵分块将矩阵分为4个子矩阵，这样就可以对两个二乘二的矩阵进行乘法，最后再合并起来得到结果。简单的分治算法的时间复杂度仍然是O(n^3)的。  
   3.strassen乘法：通过数学构造，将8个不同的小方阵转化为7个不同的小方阵，从而减少相乘的次数。时间复杂度大概为O(n^2.8)

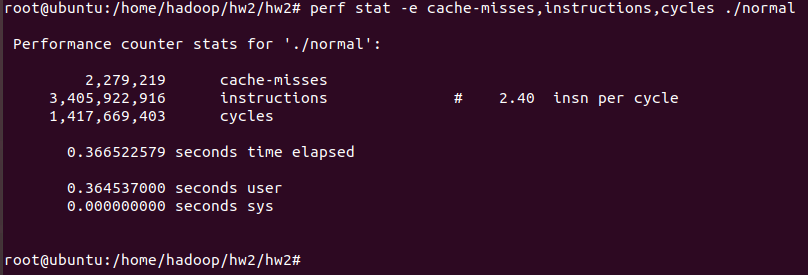
使用perf观察cache-misses,cpi,mem-loads 等性能指标:

先安装好perf：



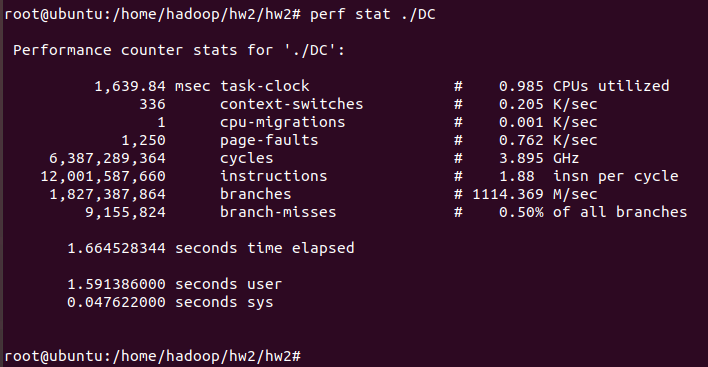
一般算法：

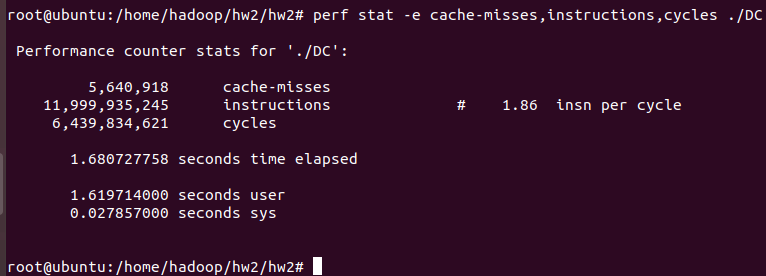




可以看到cache-misses为2279219次，指令数量为3485922916条，平均cpu cycles只有1417669483条，计算得到cpi = 1417669483/3485922916=0.4067，mem\_loads我查看不了，一查看就报错所以这里没有贴出。

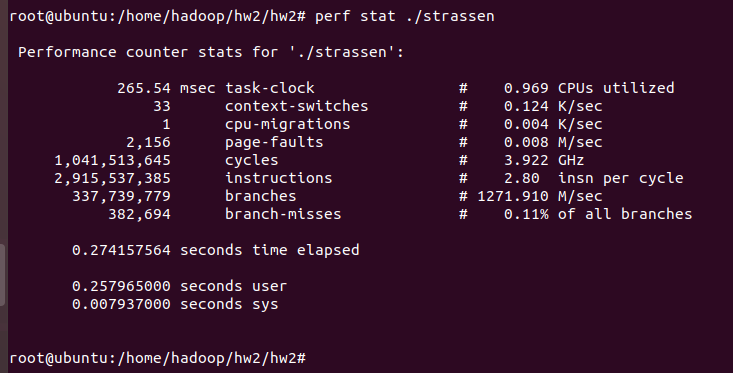
普通分治：

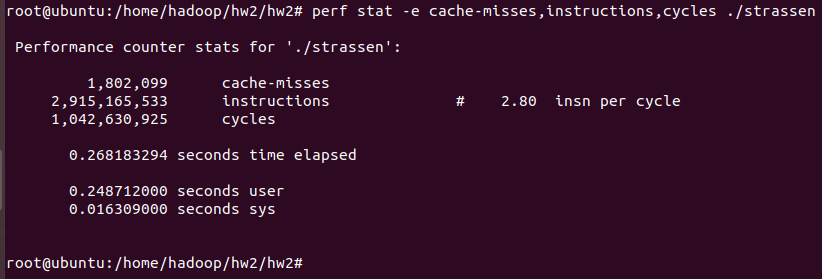




cache-misses为5640918次，与一般算法在一个数量级，指令数量为11999935245条，平均cpu cycles有6439834621条，计算得到cpi = 6439834621/11999935245=0.5367，CPI有所提升，但是可以看到平均时间消耗从0.36秒上升到1.68秒，说明程序在递归调用时消耗大量时间。

Strassen算法：





cache-misses为1802899次，与上面是一个数量级，指令数量为2915165533条，平均cpu cycles有1042630925条，计算得到cpi = 1042630925/2915165533=0.3577.

1. 在这里我假设数组的元素都是4字节的float类型，一个word是4个字节，cache分配时不会把上一次刚分配的块覆盖掉，i,dot\_prod两个变量放在寄存器里。则对每一次循环：  
     ①如果i%4==0，则读取a[i]时需要从内存中取出a[i],a[i+1],a[i+2],a[i+3]放在cache里，需要100ns，读取b[i]时需要从内存中取出b[i],b[i+1],b[i+2],b[i+3]放在cache里，需要100ns。则这次循环需要 100ns+100ns+4ns(比较i与dim要1ns,i++要1ns,a[i]b[i]要1ns，将结果加到dot\_prod要1ns)，共204ns。  
     ②如果i%4！=0，则a[i],b[i]可以从cache里取出，需要2ns。则这次循环需要 2ns+2ns+4ns，共8ns。

所以，每次循环平均需要 (8\*3+204\*1)/4=57ns故每秒钟能进行17.5M次循环。考虑到每次循环是进行了两次浮点数运算，所以能达到的最高性能是35MFLOPS

3.我们仍然假设cache不会把刚刚分配的块覆盖掉，在取a[i][j]时只需访问一次内存cache，i和j两个变量放在寄存器里，且第一重循环中i<dim，i++这两个操作可以忽略不计。值得注意的是cache的大小为32k，而b数组大小为16k，矩阵a的一行也是16k，如果cache分配策略是最优的，则b数组会在i=0时全部装进cache里，只有a数组需要重复取出，所以在内存里取出b数组的时间也可以忽略。  
  则对每一次循环：  
  ①如果j%4==0，则读取a[i][j]时需要从内存中取出a[i][j],a[i][j+1],a[i][j+2],a[i][j+3]放在cache里，需要100ns。c[i]只有一开始会从内存中取到cache里(这一次的用时也忽略不计)，之后一直在寄存器里放着。  
则这次循环需要 100ns+6ns(假设比较j与dim要1ns,j++要1ns,a[i][j]b[j]要1ns，取出c[i]要2ns，将结果加到c[i]要1ns)，共106ns。  
  ②如果i%4！=0，则a[i][j],b[j]可以从cache里取出，需要2ns。则这次循环需要 2ns+2ns+6ns，共10ns。  
  所以，每次循环平均需要 （3\*10+106\*1）/4=34ns,每秒钟能执行29.4M次循环，能达到的最高性能是59.8MFLOPS。

1. 实验结果

四、遇到的问题及解决方法