天津大学

计算机系统课程实验

选择题目：第五章实验

学生姓名 王雨朦

学生学号 2016229082

学院名称 国际工程师学院

专业 计算机

时间 2017/10/5

目录

一、 实验条件

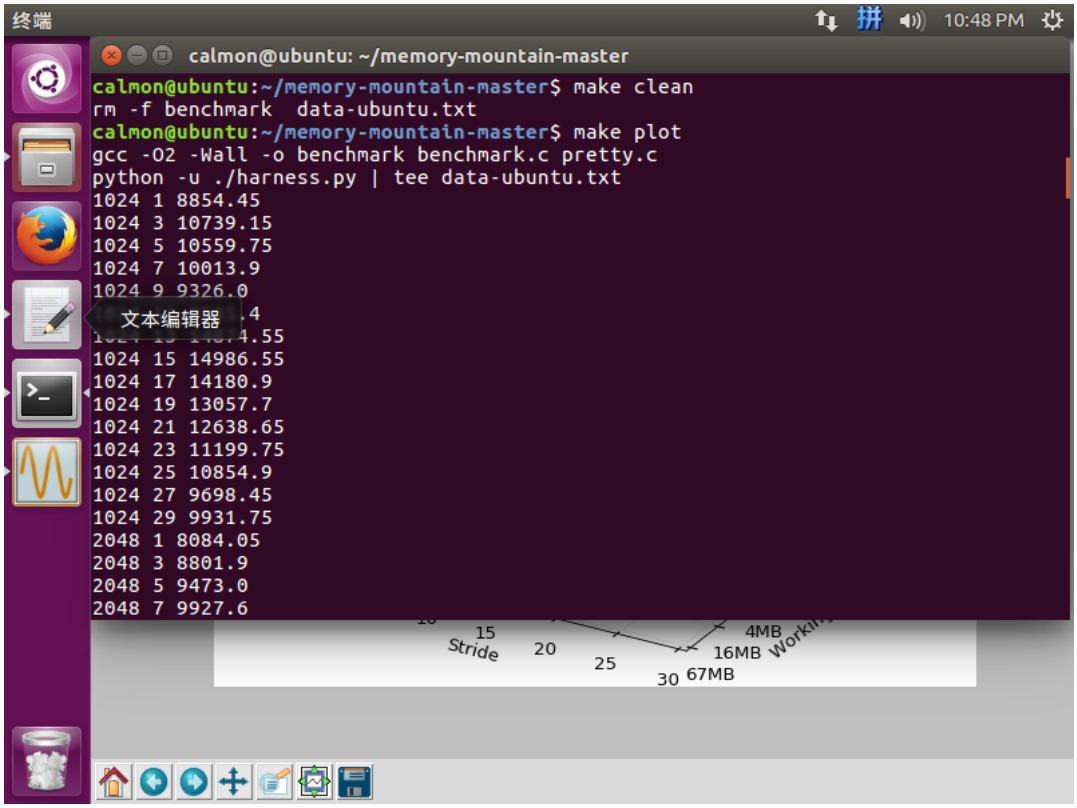
机器型号	Acer Aspire E1-471G	
内存大小	10GB	
系统类型	64 位操作系统	
SSD	Sumsung 256GB	
CPU	Intel® Core™ i5-3210M CPU @2.50GHz 2.50GHz	
高速缓存	L1	2*32KB，8 路
	L2	2*256KB，8 路
	L3	3.0MB，12 路
操作系统	Windows 7 旗舰版 64 位	

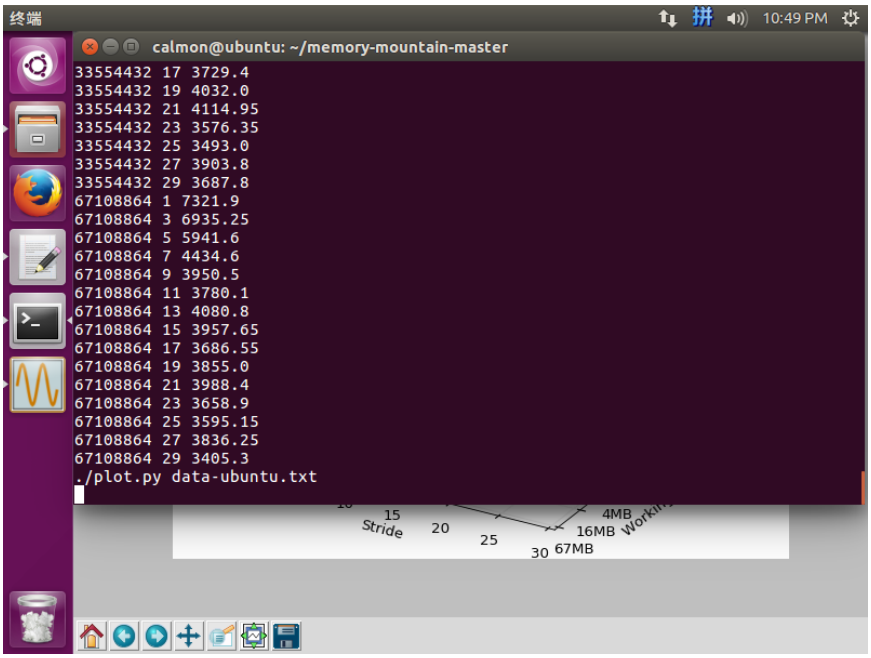
二、 大实验

实验一：存储器山

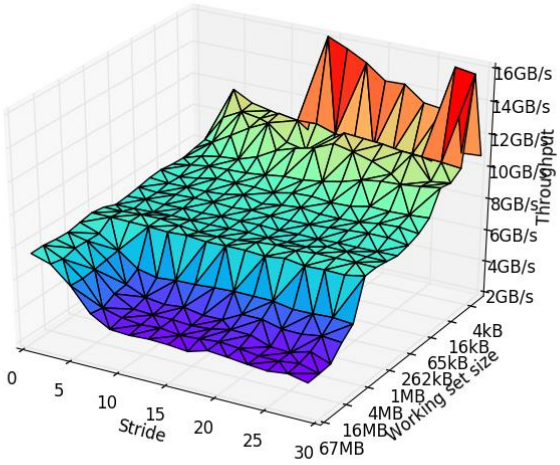
A. 实验描述

实验运行在 linux 系统下，python 实现高速缓存性能测试和存储器山的画图。





B. 实验结果



C. 实验分析

存储器山的三维如上图所示，能看到如图所示三条山脊，由山坡的由高到低分别对应于 L1 高速缓存、L2 高速缓存和 L3 高速缓存。

坐标轴中 working set size 指工作集的大小，stride 是访问数据的步长，stride 为 1，就是按顺序访问数据，为 2 就是间隔一个访问，这和数据的局部性有关。Throughput，即吞吐量是指存储器的访问速度，这个值越高越好。

通过看 working set size，可以发现，橙色和黄绿色的边界大概是 32K，黄绿

色和和蓝色的边界大概是 256K，蓝色的边界大概是 3M，这正好是三级 cache 的大小。

从 stride 来看，步长越长，速度越慢。大步长的访问方法会破坏时间局部性，因此会造成访问速度下降。

存储器山很好的体现了时间局部性和空间局部性。

固定步长不变，可以明显看到，当工作集尺寸不大于 32K 时，读速率最高。当工作集尺寸介于 32K 与 256K 之间时，读速率明显降低。当工作集尺寸大于 256K 低于 3M 时，读速率再次降低。这是由于工作集的尺寸决定了由哪一层存储器来提供待读取的数据。

实验二：图像增强中用均方误差的计算来测试空间局部性

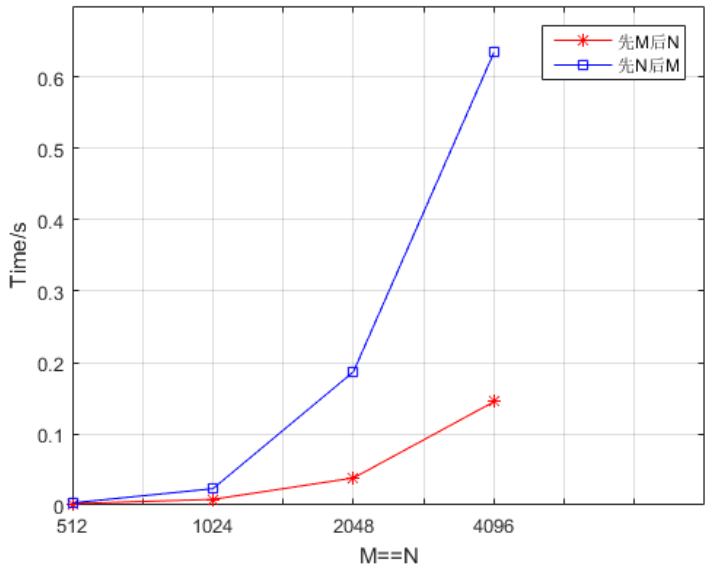
A. 实验描述

此实验中以指针二维数组模拟图像，进行图像增强中均方误差的计算，改变两重循环中的变量顺序，来测试空间局部性的差异。

B. 实验数据

M==N	Time（ij）	Time(ji)
512	0.002	0.003
1024	0.008	0.023
2048	0.038	0.187
4096	0.145	0.635

C. 实验结果



D. 实验分析

如图中，矩阵的存储是先行后列，所以先按照行顺序访问数组的循环相对于先按照列访问矩阵的要快很多。

实验三：分块矩阵相乘

A. 实验描述

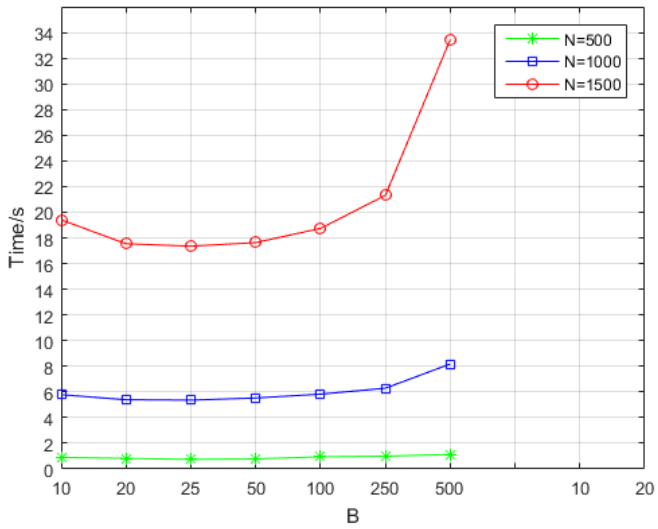
实验中的变量有 N 和 B 两个，随着 N 的增大，即矩阵的大小增大，矩阵相乘运行的时间一定是增加了；而当 N 一定时，随着分块 B 的增大，矩阵相乘的运行时间先是逐渐减少，随后又逐渐增加。因为分块越接近缓存大小，性能越好，所以曲线中有个相对的最低点。

B. 实验数据

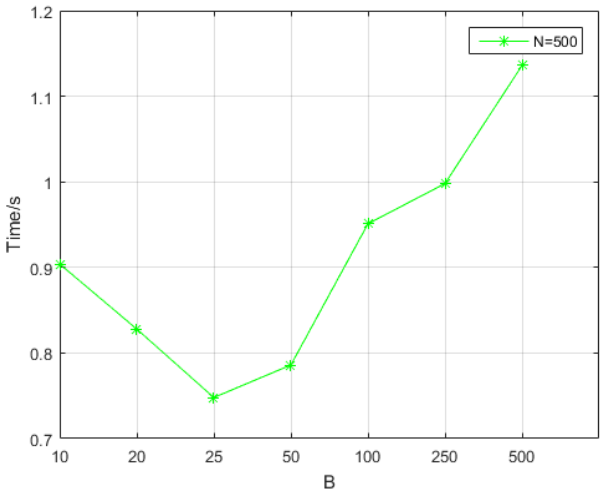
N*N B	10	20	25	50	100	250	500
500*500	0.904	0.828	0.748	0.786	0.951	0.998	1.136
1000*1000	5.787	5.395	5.362	5.532	5.83	6.289	8.179
1500*1500	19.41	17.544	17.354	17.634	18.741	21.313	33.445

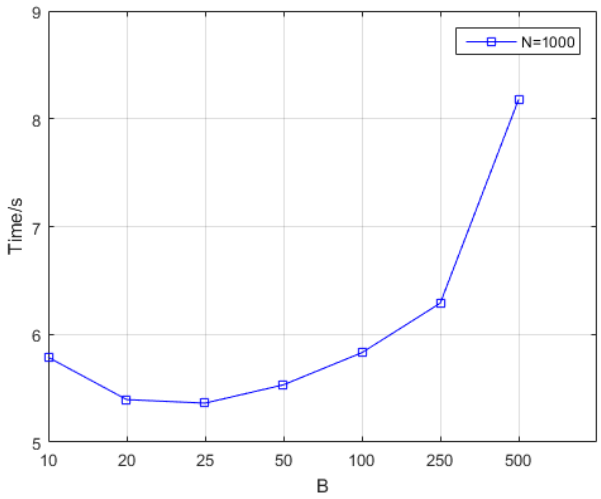
C. 实验结果

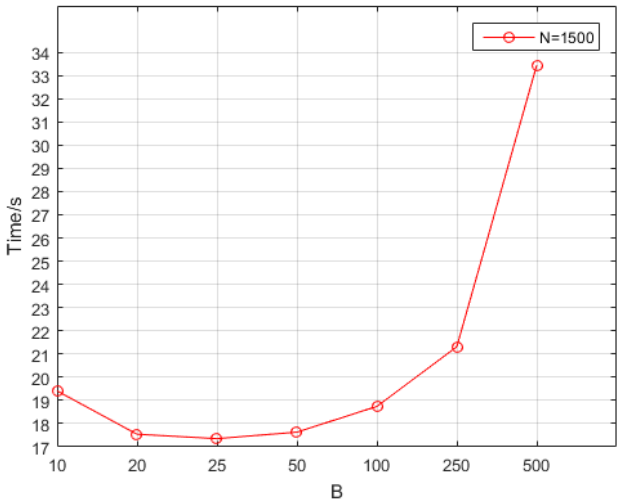
如下图，为不同大小的矩阵，在分块不同的情况下的运行时间的曲线图：



由上图，三种 N 不同的情况绘制在一个大图里看起来较模糊，因为时间差很大，这样蓝色和绿色曲线的变化就不能够明显的展示出来。故而又将每个 N 一定，B 变化的情况下，分开画图，如下：







D. 实验分析

此题考察的时间局部性，由于对矩阵进行了分块，那么读的时候，装入的是一整个块，而不是一个矩阵，因而命中率会提高。而分块大小和缓存大小相对合适时，执行时间最小，由此题的实验中，上图所得，最合适的分块大小应为 B=25 附近，因为三个大小的矩阵都显示得到当 B 为 25 时，执行时间最小，程序性能最好。

实验四：矩阵转置

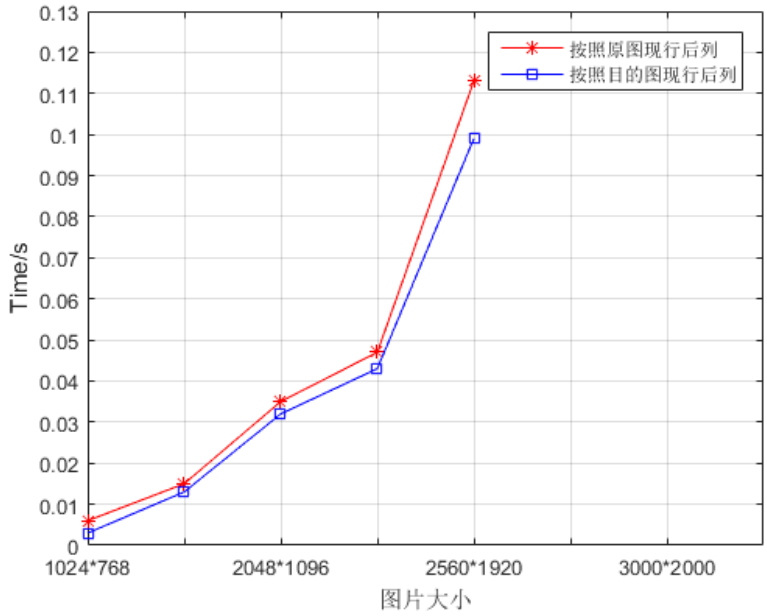
A. 实验描述

此次实验针对灰度图进行转置，当进行转置算法时，有按照原始图先行后列和按照结果图先行后列两种算法。本实验针对两种方法进行实验。

B. 实验数据

图片大小	Time(按原始图)/s	Time(按结果图)/s
1024*768	0.006	0.003
2048*1096	0.015	0.013
2560*1920	0.065	0.032
3000*2000	0.047	0.043
4000*3000	0.123	0.099

C. 实验结果



D. 实验分析

实验结果都是后一种比较快一些，即按照结果图先行后列访问比较快。

原因猜想是因为按照结果图先行后列，即按照原始图先列后行进行访问，而这些图片都是原始图的列比行少，所以不命中的情况少。于是改变了一下实验数据中的行和列，让行数少于列数，但是发现依然是后一种情况所需时间短。这说明猜想错误。

经过查资料，和同学交流。觉得应该是读的不命中和写不命中的时间不同，对于第二种情况，是读数据的时候不命中，而读数据不命中的时间明显短。第一种写不命中耗时长。

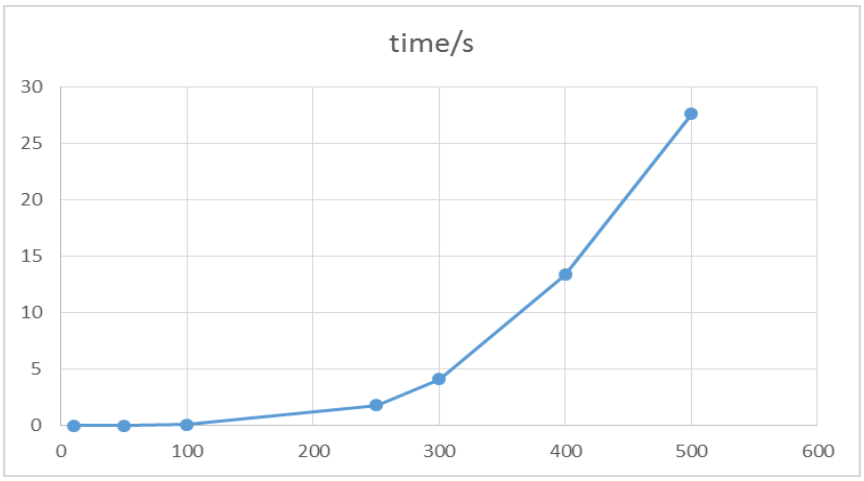
三、 小实验

6.7 题

A. 数据

N	time/s
10	0
50	0.009
100	0.084
250	1.782
300	4.11
400	13.371
500	27.608

B. 画图



C. 分析：

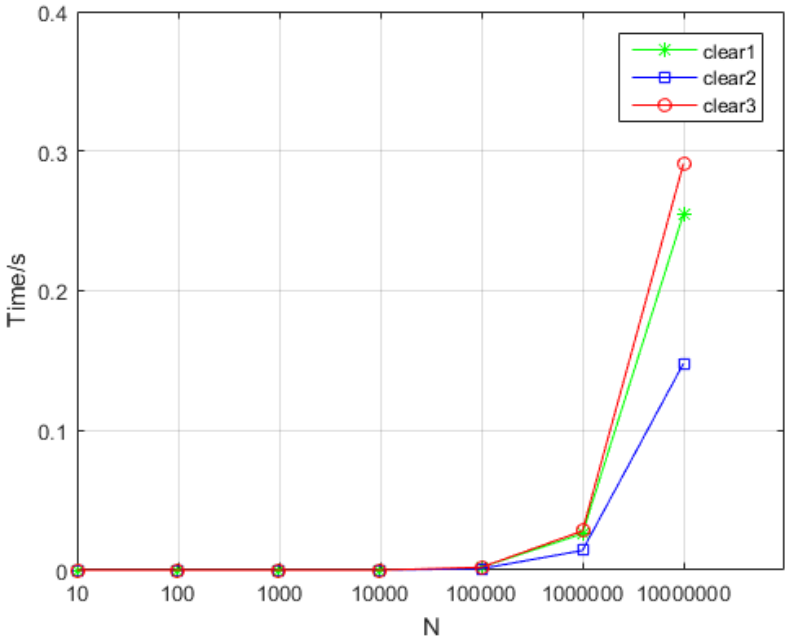
题中考察的是空间局部性，即应该按照矩阵的行列顺序依次访问的时候，命中率最高。而此题中将 N 依次增大，看到随着 N 增大，程序访问时间是呈指数增长的。

6.8 题

A. 数据

N	time/s		
	clear1	clear2	clear3
10	0	0	0
100	0	0	0
1000	0	0	0
10000	0	0	0
100000	0.002	0.001	0.002
1000000	0.026	0.014	0.028
10000000	0.255	0.148	0.291

B. 画图



C. 分析

Clear2 函数是按照先行后列访问，因此时间最短，由图中蓝色曲线可以看到；Clear3 函数是按照先列后行访问，因此时间最久，由绿色曲线所示；Clear1 函数虽然按照先行后列，但是列在每行的情况下访问了两次，因而时间比单纯的先行后列要慢一些，不过还是比先列后行快一些。

四、 作业

6.37 题

函数	N=64	N=60
SumA	0.25	0.25
SumB	1	0.25
sumC	0.5	0.25

由于整型 int 占 4 个字节，块大小为 16 字节，一个块可以放四个整型。高速缓存有 4KB，4K/16=256，高速缓存有 256 块可以直接映射。

N=64 时，sumA 为行优先访问，每块中第一个元素不命中，因此不命中率为 0.25；SumB 列优先访问时，每行 64 个数据，需要 64/4=16 个块来进行保存，所以高速缓存的 256 块可以存 256/16=16 行，前 16 行按照列访问，每次都不命中，到第 17 次以后，每次会产生覆盖，因此不命中率为 1；sumC 按照先列后行和先行后列相结合进行访问，每次访问的第一个元素都不命中，第二个元素都会命中，所以不命中率是 0.5。

N=60 时，sumA 情况和以上相同；sumB 时，每行 60 个数据，需要 60/4=15 个块来进行保存，比之前少一个块，这样会导致错位，所以不会产生覆盖，只有第一个不会命中，后面三个命中，不命中率 0.25；sumC 情况相同，不命中率 0.25。

6.41 题

每行 4 个字节，而 char 为一个字节，循环时，第一个不命中，后三个命中，所以不命中率为 0.25。