计 算 机 组 成 原 理 实 验 报 告

题目： cache 和程序访问的局部性（一）

姓　　名： 段欣然

专　　业： 计算机科学与技术

年　　级： 2020 级

学　　号： 202011081033

任课教师： 王志春

完成日期： 2021 年 5 月 20 日

人工智能学院

一、 实验要求

通过实际程序的执行结果，了解程序访问的局部性对带有 cache 的计算机系统性能的影响。

正文中文字体统一为宋体，英文及数字字体为 Times New Roman，字号小四，行间距最小值 20 磅，段落首行缩进 2 字符

二、 实验结果与分析

程序段 A

```
Assign-array-rows(  )
{
int i, j, a[M][N]
……….
For  (i=0;i<M;i++)
     For(j=0;j<N;j++)
        A[i][j]=i+j
……….
}
```

程序 B

```
Assign-array-cols(  )
{
Int i. j, a[M][N]
……….
For  (i=0;j<N;j++)
     For(i=0;i<M;i++)
         A[i][j]=i+j
……….
}
```

figure 1 **程序代码示例**

在如 figure 1 所示程序中，修改或添加必要的语句（如计时函数等），以计算和打印主体程序段的执行时间。分别以$M = 100000, N = 10$、$M = 1000, N = 1000$、$M = 10, N = 10000$，执行程序 A 和程序 B，以比较两个程序执行时间的长短。下为实验代码及运行结果示例。

```cpp
int main(){
    double time = 0;
    double counts = 0;
    LARGE_INTEGER nFreq;
    LARGE_INTEGER nBeginTime;
    LARGE_INTEGER nEndTime;
    QueryPerformanceFrequency(&nFreq);
    QueryPerformanceCounter(&nBeginTime);
    cout<<"Assign-array-rows running"<<endl;
    Assign_array_rows();
    QueryPerformanceCounter(&nEndTime);
    time = (double)(nEndTime.QuadPart - nBeginTime.QuadPart) / (double)nFreq.QuadPart;
    cout << "Assign-array-rows spends: " << time * 1000 << "ms" << endl;
    QueryPerformanceFrequency(&nFreq);
    QueryPerformanceCounter(&nBeginTime);
    cout<<"Assign-array-cols running"<<endl;
    Assign_array_cols();
    QueryPerformanceCounter(&nEndTime);
    time = (double)(nEndTime.QuadPart - nBeginTime.QuadPart) / (double)nFreq.QuadPart;
    cout << "Assign-array-cols spends: " << time * 1000 << "ms" << endl;
    return 0;
}
```

figure 2 实验主要代码示例

```
\tempCodeRunnerFile }
Assign-array-rows running
Assign-array-rows spends: 3.0641ms
Assign-array-cols running
Assign-array-cols spends: 5.106ms
PS C:\Users\24636\AppData\Local\Temp>
```

figure 3 程序运行结果示例

1. 实验结果

3 次实验结果下表所示。

实验序号	M	N	Assign-array-rows（ms）	Assign-array-cols（ms）
1	100000	10	3.0641	5.106
2	1000	1000	3.6502	4.0192
3	10	10000	1.467	0.468

table 1 实验结果

2. 结果分析

在实验 1 中，由于 Assign-array-rows 按先行后列的顺序访问数组，每次都访问

离上次访问位置最近的元素，故从缓存中调用数据次数较多，从主存中调用数组次数较少。而 Assign-array-cols 按先列后行的顺序访问数组，每次访问有固定跨度且跨度较大（100000），故基本都需要从主存中读取数据。因此实验 1 结果呈现出 Assign-array-rows 耗时少于 Assign-array-col。

在实验 2 中，数组行列大小一致，但 Assign-array-rows 仍然每次访问离上次访问位置最近的元素，故读取缓存中数据次数较多，读取主存中数据次数较少；而 Assign-array-cols 同样每次具有固定跨度地访问数组，因此 Assign-array-rows 运行时间少于 Assign-array-cols。

在实验 3 中，数组每行数据较多，当 cache 大小较小时，可能发生多次替换，命中可能性减小。所以结果呈现出 Assign-array-rows 运行时间多于 Assign-array-cols。

三、 实验小结

我们编写程序时可以充分利用局部性原理以加快运行速率，但需要注意缓存中每行的块大小不能过大，否则可能类似实验 3 的结果，先列后行反而比先行后列的顺序访问更快。