计算机系统结构实验

实验（四）Cache性能分析

姓　　名	熊恪峥
学　　号	22920202204622
日　　期	2023年5月6日
学　　院	信息学院
课程名称	计算机系统结构

实验（四）Cache性能分析

目录

1 补充实验

1.1 实验目的

简单设计实验，理解Cache预取

1.2 实验结果

Cache预取的思路是发生未命中时将本块和下一块都调入Cache。为了进行实验，首先需要对Cache块大小进行配置。首先，进行如下配置：

1. Cache大小：4KB
2. 相联方式：直接相联
3. 块大小：16B

确定Cache参数，主要是为了确定块大小。确定了块大小，就可以构造访问序列。例如如图 1a序列，它的访问方式如图 1b可以看到，访问序列中，每次访问都是下一块的第一个字节。因此，如果没有预取，Cache不会命中。通过预取，可以将下一块也调入Cache，这样就可以减少未命中次数。

(a) 访问方式

```
0       10
0       20
0       30
0       40
0       50
0       60
0       1000
0       1010
0       1020
0       1030
0       1040
0       1050
0       1060
```

(b) 访问位置

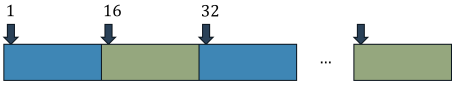

图 1: 访问地址序列

实验结果如图 2，可见未命中率确实减少了。

(a) 不预取

(b) 预取

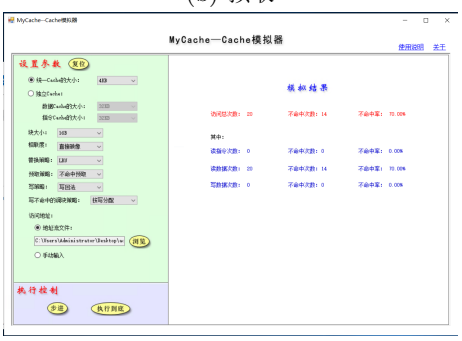

图 2: 实验结果

1.3 分析

虽然在本例中确实降低了未命中率，但是这种预取方式并不总是能提高性能。例如如果下次访问不落在下一块内，这样的地址序列会导致预取过程读入无用的数据，反而增加了额外的开销。因此，预取对性能是否有提升需要根据实际情况进行分析。

2 探究性实验

2.1 实验目的

从Cache的角度，分析以下代码的效率，以及改进方法：

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    for(k=0;k<n;k++)
      C[i][j]+=A[i][k]*B[k][j];
```

2.2 实验结果

可以通过重排循环的方式，使得Cache的命中率提高。如图 **??**。

(a) `ijk`

```
#include <stdio.h>

#define n 1024
int A[1024][1024],
    B[1024][1024],
    C[1024][1024];

int init(){
  for(int i=0;i<1024;i++)
    for(int j=0;j<1024;j++)
      A[i][j]=B[j][i]=i+j;
}

int main(){
  init();
  for(int i=0;i<n;i++)
     for(int j=0;j<n;j++)
       for(int k=0;k<n;k++)
         C[i][j]+=A[i][k]*B[k][j];
  return 0;
}
```

(b) `ikj`

```
#include <stdio.h>

#define n 1024
int A[1024][1024],
    B[1024][1024],
    C[1024][1024];

int init(){
  for(int i=0;i<1024;i++)
    for(int j=0;j<1024;j++)
      A[i][j]=B[j][i]=i+j;
}

int main(){
  init();
  for(int i=0;i<n;i++)
     for(int k=0;k<n;k++)
       for(int j=0;j<n;j++)
         C[i][j]+=A[i][k]*B[k][j];
  return 0;
}
```

图 3: 不同的循环次序

其中图 4a是题目中所给的`ijk`次序，而图 4b是重排过后效果较好的 `ikj`次序。实验结果如图 4，可见重排后的效果确实更好。

(a) `ijk`

```
(base) bear@DESKTOP-E68U1HV [15:43:11] [/mnt/d/Proj
-> % time ./ijk
./ijk  8.45s user 0.02s system 99% cpu 8.474 total
```

(b) `ikj`

```
(base) bear@DESKTOP-E68U1HV [15:43:29] [/mnt/d/Proj
-> % time ./ikj
./ikj  3.71s user 0.01s system 99% cpu 3.725 total
```

(c) 对比

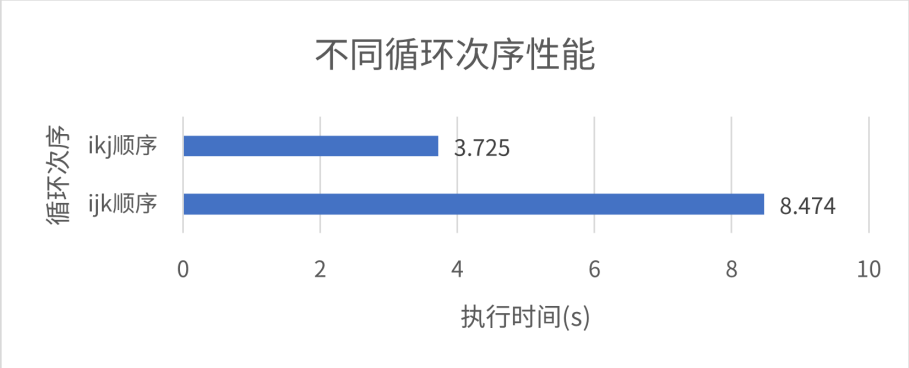

图 4: 结果

2.3 分析

矩阵在内存中常常以行优先的方式存储。因此，在`ijk`顺序中，对B的访问是按列的，会造成缓存不命中，性能降低。而在`ikj`顺序中则调整成了按行访问，可以提高缓存命中率。如图 5。

(a) `ijk` (b) `ikj`

图 5: 访问方式

然而，对C的访问虽然具有空间局部性，但缺乏时间局部性，因此如果缓存不够大或者相联程度比较低，这种方法可能不能提高性能。

3 实验总结

本次实验我探究了Cache的命中率与预取的关系，然后通过调整循环次序探究了局部性对Cache命中率的影响，使得矩阵乘法得到了性能提升。通过本次实验，加深了我对Cache的理解。