**深 圳 大 学 实 验 报 告**

**课 程 名 称： 计算机系统(2)**

**实验项目名称： Cache实验**

**学 院： 计算机与软件学院**

**专 业： 计算机与软件学院所有专业**

**指 导 教 师： 贾森**

**报告人： 洪继耀 学号： 2014150120**

**班级： 02**

**实 验 时 间： 2016年6月13日**

**实验报告提交时间： 2016年6月20日**

**教务处制**

**一、实验目标：**

了解Cache对系统性能的影响

**二、实验环境：**

1、个人电脑（Intel CPU）

2、Fedora 13 Linux 操作系统

**三、实验内容与步骤**

1. 编译并运行程序A，记录相关数据。
2. 不改变矩阵大小时，编译并运行程序B，记录相关数据。
3. 改变矩阵大小，重复1和2两步。
4. 通过以上的实验现象，分析出现这种现象的原因。

**程序A：**

#include <sys/time.h>

#include <unistd.h>

#include <stdio.h>

main(int argc,char \*argv[])

{

float \*a,\*b,\*c, temp;

long int i, j, k, size, m;

struct timeval time1,time2;

if(argc<2) {

printf("\n\tUsage:%s <Row of square matrix>\n",argv[0]);

exit(-1);

} //if

size = atoi(argv[1]);

m = size\*size;

a = (float\*)malloc(sizeof(float)\*m);

b = (float\*)malloc(sizeof(float)\*m);

c = (float\*)malloc(sizeof(float)\*m);

for(i=0;i<size;i++)

for(j=0;j<size;j++) {

a[i\*size+j] = (float)(rand()%1000/100.0);

b[i\*size+j] = (float)(rand()%1000/100.0);

}

gettimeofday(&time1,NULL);

for(i=0;i<size;i++)

for(j=0;j<size;j++)

{

c[i\*size+j] = 0;

for (k=0;k<size;k++)

c[i\*size+j] += a[i\*size+k]\*b[k\*size+j];

}

gettimeofday(&time2,NULL);

time2.tv\_sec-=time1.tv\_sec;

time2.tv\_usec-=time1.tv\_usec;

if (time2.tv\_usec<0L) {

time2.tv\_usec+=1000000L;

time2.tv\_sec-=1;

}

printf("Executiontime=%ld.%6ld seconds\n",time2.tv\_sec,time2.tv\_usec);

} //for

return(0);

}//main

**程序B：**

#include <sys/time.h>

#include <unistd.h>

#include <stdio.h>

main (int argc,char \*argv[])

{

float \*a,\*b,\*c,temp;   
long int i,j,k,size,m;   
struct timeval time1,time2;   
if (argc<2)   
{

printf("\n\tUsage:%s <Row of square matrix>\n",argv[0]);   
exit(-1);

}

size=atoi(argv[1]);

m=size\*size;

a=(float\*)malloc(sizeof(float)\*m);

b=(float\*)malloc(sizeof(float)\*m);

c=(float\*)malloc(sizeof(float)\*m);

for(i=0;i<size;i++)   
 for(j=0;j<size;j++)   
 {

a[i\*size+j]=(float)(rand()%1000/100.0);

c[i\*size+j]=(float)(rand()%1000/100.0);

}

gettimeofday(&time1,NULL);

for (i=0;i<size;i++)

for (j=0;j<size;j++)

{

b[i\*size+j] = c[j\*size+i];

for (i=0;i<size;i++)

for(j=0;j<size;j++)

{

c[i\*size+j] = 0;

for (k=0;k<size;k++)

c[i\*size+j] += a[i\*size+k] \* b[j\*size+k];

} //for

gettimeofday(&time2,NULL);

time2.tv\_sec-=time1.tv\_sec;

time2.tv\_usec-=time1.tv\_usec;

if(time2.tv\_usec<0L)

{

time2.tv\_usec+=1000000L;

time2.tv\_sec-=1;

}

printf("Executiontime=%ld.%6ld seconds\n",time2.tv\_sec,time2.tv\_usec);

}//for

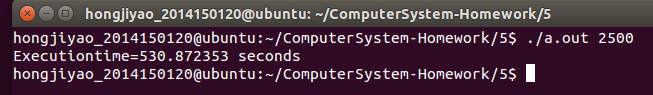
return(0);

}

**四、实验结果及分析**

1. 用C语言实现矩阵（方阵）乘积一般算法（程序A），填写下表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 矩阵大小 | 100 | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
| 一般算法执行时间(s) | 0.5293 | 0.846603 | 9.208 | 58.68610 | 157.725569 | 530.872353 | 715.772059 |

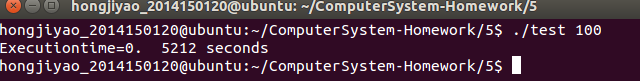


分析：

矩阵越大 运行结果越长 而且越大 运行时间增加幅度越大

1. 程序B是基于Cache的矩阵（方阵）乘积优化算法，填写下表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 矩阵大小 | 100 | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
| 优化算法执行时间(s) | 0.5212 | 0.834076 | 6.768957 | 23.462820 | 56.745567 | 105.607632 | 165.579817 |



分析：

优化后速度变得很快

1. 优化后的加速比（speedup）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 矩阵大小 | 100 | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
| 加速比 | 1.015541059 | 1.015019015 | 1.360327743 | 2.501238129 | 2.779522302 | 5.026837009 | 4.322821899 |

加速比定义：加速比=优化前系统耗时/优化后系统耗时；

所谓加速比，就是优化前的耗时与优化后耗时的比值。加速比越高，表明优化效果越明显。

分析：

在低阶数的时候加速不明显

随着阶数增加，优化效果变得明显

这是因为cache是连续读写的时候占优势

而矩阵就是连续读写的典型例子

**五、实验总结与体会**

Cache真好玩

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字： 刘刚    2016年 3月25日 |
| 备注： |