**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： 计算机科学与技术**

**指导教师： 冯禹洪**

**报告人：黄志鹏 学号： 2017303008 班级： 计科1班**

**实验时间： 2020年 6 月 20 日至 7 月 4 日**

**实验报告提交时间： 2020年 7 月 6 日**

**教务处制**

|  |
| --- |
| **实验目的与要求：**  **目的：**   1. 观察cache对程序数据访问的加速作用； 2. 掌握编程中利用cache的基本方法；   要求：  观察代码访存模式对性能的影响、编写代码测试cache层次结构和容量等 |
| **方法、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 <stdlib.h>  #include <stdio.h>  int 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.%06ld seconds\n",time2.tv\_sec,time2.tv\_usec);    return(0);  }//main  **程序B：**  #include <sys/time.h>  #include <unistd.h>  #include <stdlib.h>  #include <stdio.h>  int 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);  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];  }  }  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.%06ld seconds\n",time2.tv\_sec,time2.tv\_usec);    return(0);  }//main  **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） | 0.0097 | 1.61 | 26.77 | 92.14 | 225.54 | 527.40 | 983.84 |   分析：由于此算法的第二个矩阵是按照列访问，没有利用Cahe的优势，所以在数据读取时耗费了大量时间。   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） | 0.0094 | 1.35 | 10.43 | 30.83 | 71.41 | 138.80 | 237.08 |   分析：此算法通过将第二个矩阵转置，将数据访问的时间减少，所以提升了程序运行效率。   1. 优化后的加速比（speedup）  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 矩阵大小 | 100 | 500 | 1000 | 1500 | 2000 | 2500 | 3000 | | 加速比 | 1.03 | 1.19 | 2.57 | 2.99 | 3.16 | 3.80 | 4.15 |   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：由于合理利用了cache，所以算法得到了优化，并且会随着矩阵的变大，而优化效率更加的明显。  图1 一般算法和优化算法耗时  图2 加速比 |
| **实验过程及内容：**  A程序:    B程序: |
| **实验结论：**  在矩阵乘法中，将第二个矩阵进行转置，这样合理地利用cache，这样极大的提高程序的运行效率。 |
| **心得体会：**  通过此次的实验，明白了在数据访问时要合理地利用cache，这样会极大的提高程序的性能，提高程序的运行效率。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：冯禹洪  2020年 月 日 |
| 备注： |