## Atitit.性能提升 提高Cache命中率

## 提高Cache命中率

注意：这里讲的提高Cache命中率方法与前面提高共享Cache命中率方法不完全一样）：

·数组合并

利用块长，改善空间局部性。比如：

    /\* Before: 2 sequential arrays \*/

int val[SIZE]; int key[SIZE];

/\* After: 1 array of stuctures \*/

struct merge {

        int val;

        int key;

};

struct merge merged\_array[SIZE];  
    通过这种方式的修改，意识减少了val与key之间的Cache冲突，二是改善了Cache空间局限性。

·循环交换

改变嵌套循环中访问内存的次序。

/\* Before \*/

for (k = 0; k < 100; k = k+1)

{

    for (j = 0; j < 100; j = j+1)

    {

         for (i = 0; i < 5000; i = i+1)

{

             x[i][j] = 2\_ \* x[i][j];

        }

    }

}

/\* After \*/

for (k = 0; k < 100; k = k+1)

{

    for (i = 0; i < 5000; i = i+1)

    {

         for (j = 0; j < 100; j = j+1)

        {

             x[i][j] = 2\_ \* x[i][j];

    }

    }

}

通过上面的调整，将步长为100字的跳跃式访问变为顺序访问，改善了Cache的空间局部性，提高了Cache命中率。

·循环合并

增强数据的可重用性（时间局部性）。

/\* Before \*/

for (i = 0; i < N; i = i+1)

{

    for (j = 0; j < N; j = j+1)

    {

         a[i][j] = 1/b[i][j]\_ \* c[i][j];

    }

}

for (i = 0; i < N; i = i+1)

{

    for (j = 0; j < N; j = j+1)

    {

         d[i][j] = a[i][j] + c[i][j];

    }

}

/\* After \*/

for (i = 0; i < N; i = i+1)

{

    for (j = 0; j < N; j = j+1)

    {

a[i][j] = 1/b[i][j]\_ \* c[i][j];

         d[i][j] = a[i][j] + c[i][j];

    }

}  
通过上面的循环合并，访问a和c的2次不命中降为1次，但是这里可能有另外一个问题，就是a[i][j]因为流水线问题可能导致阻塞，不过编译器一般能够解决这样的问题，所以程序员可以不考虑这个问题。

·分块

集中访问可取入cache的块状矩阵，避免全行或全列的读写，以增强时间局部性。

/\* Before \*/

for (i = 0; i < N; i = i+1)

{

  for (j = 0; j < N; j = j+1)

  {

      r = 0;

      for (k = 0; k < N; k = k+1){

            r = r + y[i][k]\*z[k][j];

      }

      x[i][j] = r;

  }

}

上面程序中两个内层循环中:读了z[ ]的所有NxN个元素，重复读y[ ]的某一行N次，写x[ ]的某一行1次，Cache不命中次数是N及cache大小的函数。

当3NxNx4 小于cache容量时，没有不命中。

分块的思想：计算cache中放得下的 BxB子矩阵。

/\* After \*/

for (jj = 0; jj < N; jj = jj+B)

{

for (kk = 0; kk < N; kk = kk+B){

     for (i = 0; i < N; i = i+1){

        for (j = jj; j < min(jj+B-1,N); j = j+1)

         {

             r = 0;

             for (k = kk; k < min(kk+B-1,N); k = k+1) {

                 r = r + y[i][k]\*z[k][j];

             }

             x[i][j] = x[i][j] + r;

         }

     }

}

}

B称为分块因子Blocking Fa[cto](http://blog.vsharing.com/Tag/cto" \t "http://bbs.vsharing.com/Information/Soft/_blank)r，不命中数从 2N3 + N2 降到 2N3/B +N2，但还存在因冲突导致的不命中，如下图：

[分享]提升系统性能方法初步探讨 - 软件行业 - 畅享论坛.html