第2回　数値計算レポート

g99p1200
藤井　優尚

出題日：2001年5月18日
提出期限：2001年6月2日正午まで
提出日：2001年6月2日

1　問題

1000 × 1000double型行列A,Bの積を計算するCプログラムを作れ。そして、10種の行列の組み合わせで計算した平均計算時間をできるだけ短縮せよ。

2　解答

プログラムの説明
このプログラムはホームページのものを参考にした。
その中で使った技術は、ループ変換、ループのアンローリング、ブロック化法、内積化である。
ループ変換では、(j,i,k)が最速であったのでこれを用いた。
ループのアンローリングでは２重または３重ループで用いた。このとき、２重ループでは内のループで１０飛ばし、外では４飛ばした。また、３重ループでは一番外のループはそのまま、内は１０飛ばし、外は５飛ばしにした。
ブロック化法では、キャッシュの有効範囲を考え、１００ずつ処理をさせた。
コンパイル時には、最高速化するオプションとして、cl /O2 ～.cを用いた。

作成したプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define NMAX 1000 /* Maximum Matrix Size */
#define NBLC 100 /* Block Matrix Size */
#define Timer clock()
```

```
/* Prototype */
int mflops(double flops, double time);
void set_mat(double A[NMAX][NMAX], double B[NMAX][NMAX],double C[NMAX][NMAX]);
double check(double C[NMAX][NMAX], double D[NMAX][NMAX]);
/* Prototype */

double A[NMAX][NMAX], B[NMAX][NMAX], C[NMAX][NMAX],D[NMAX][NMAX];

void multmats(double A[NMAX][NMAX], double B[NMAX][NMAX]){
   int i, j, k, iloop, jloop, kloop, is, js, ks;
   double s, s2, s3, s4, s5;
   double WA[NBLC][NBLC], WB[NBLC][NBLC], WC[NBLC][NBLC];

   for(iloop=0; iloop<10; iloop++){
      is = NBLC*iloop;
      for(jloop=0; jloop<10; jloop++){
         js = NBLC*jloop;
         /* Copy A to WA */
         for(i=0; i<NBLC; i=i+4){
            for(j=0; j<NBLC; j=j+10){
               WA[i][j] = A[is+i][js+j];
               WA[i][j+1] = A[is+i][js+j+1];
               WA[i][j+2] = A[is+i][js+j+2];
               WA[i][j+3] = A[is+i][js+j+3];
               WA[i][j+4] = A[is+i][js+j+4];
               WA[i][j+5] = A[is+i][js+j+5];
               WA[i][j+6] = A[is+i][js+j+6];
               WA[i][j+7] = A[is+i][js+j+7];
               WA[i][j+8] = A[is+i][js+j+8];
               WA[i][j+9] = A[is+i][js+j+9];

               WA[i+1][j] = A[is+i+1][js+j];
               WA[i+1][j+1] = A[is+i+1][js+j+1];
               WA[i+1][j+2] = A[is+i+1][js+j+2];
               WA[i+1][j+3] = A[is+i+1][js+j+3];
               WA[i+1][j+4] = A[is+i+1][js+j+4];
               WA[i+1][j+5] = A[is+i+1][js+j+5];
               WA[i+1][j+6] = A[is+i+1][js+j+6];
               WA[i+1][j+7] = A[is+i+1][js+j+7];
               WA[i+1][j+8] = A[is+i+1][js+j+8];
               WA[i+1][j+9] = A[is+i+1][js+j+9];

               WA[i+2][j] = A[is+i+2][js+j];
               WA[i+2][j+1] = A[is+i+2][js+j+1];
               WA[i+2][j+2] = A[is+i+2][js+j+2];
               WA[i+2][j+3] = A[is+i+2][js+j+3];
               WA[i+2][j+4] = A[is+i+2][js+j+4];
               WA[i+2][j+5] = A[is+i+2][js+j+5];
               WA[i+2][j+6] = A[is+i+2][js+j+6];
               WA[i+2][j+7] = A[is+i+2][js+j+7];
               WA[i+2][j+8] = A[is+i+2][js+j+8];
               WA[i+2][j+9] = A[is+i+2][js+j+9];

               WA[i+3][j] = A[is+i+3][js+j];
```

```
      WA[i+3][j+1] = A[is+i+3][js+j+1];
      WA[i+3][j+2] = A[is+i+3][js+j+2];
      WA[i+3][j+3] = A[is+i+3][js+j+3];
      WA[i+3][j+4] = A[is+i+3][js+j+4];
      WA[i+3][j+5] = A[is+i+3][js+j+5];
      WA[i+3][j+6] = A[is+i+3][js+j+6];
      WA[i+3][j+7] = A[is+i+3][js+j+7];
      WA[i+3][j+8] = A[is+i+3][js+j+8];
      WA[i+3][j+9] = A[is+i+3][js+j+9];
   }
}
for(kloop=0; kloop<10; kloop++){
   ks = NBLC*kloop;
   /* Copy B to WB */
   for(i=0; i<NBLC; i=i+4){
      for(j=0; j<NBLC; j=j+10){
         WB[i][j] = B[js+i][ks+j];
         WB[i][j+1] = B[js+i][ks+j+1];
         WB[i][j+2] = B[js+i][ks+j+2];
         WB[i][j+3] = B[js+i][ks+j+3];
         WB[i][j+4] = B[js+i][ks+j+4];
         WB[i][j+5] = B[js+i][ks+j+5];
         WB[i][j+6] = B[js+i][ks+j+6];
         WB[i][j+7] = B[js+i][ks+j+7];
         WB[i][j+8] = B[js+i][ks+j+8];
         WB[i][j+9] = B[js+i][ks+j+9];

         WB[i+1][j] = B[js+i+1][ks+j];
         WB[i+1][j+1] = B[js+i+1][ks+j+1];
         WB[i+1][j+2] = B[js+i+1][ks+j+2];
         WB[i+1][j+3] = B[js+i+1][ks+j+3];
         WB[i+1][j+4] = B[js+i+1][ks+j+4];
         WB[i+1][j+5] = B[js+i+1][ks+j+5];
         WB[i+1][j+6] = B[js+i+1][ks+j+6];
         WB[i+1][j+7] = B[js+i+1][ks+j+7];
         WB[i+1][j+8] = B[js+i+1][ks+j+8];
         WB[i+1][j+9] = B[js+i+1][ks+j+9];

         WB[i+2][j] = B[js+i+2][ks+j];
         WB[i+2][j+1] = B[js+i+2][ks+j+1];
         WB[i+2][j+2] = B[js+i+2][ks+j+2];
         WB[i+2][j+3] = B[js+i+2][ks+j+3];
         WB[i+2][j+4] = B[js+i+2][ks+j+4];
         WB[i+2][j+5] = B[js+i+2][ks+j+5];
         WB[i+2][j+6] = B[js+i+2][ks+j+6];
         WB[i+2][j+7] = B[js+i+2][ks+j+7];
         WB[i+2][j+8] = B[js+i+2][ks+j+8];
         WB[i+2][j+9] = B[js+i+2][ks+j+9];

         WB[i+3][j] = B[js+i+3][ks+j];
         WB[i+3][j+1] = B[js+i+3][ks+j+1];
         WB[i+3][j+2] = B[js+i+3][ks+j+2];
         WB[i+3][j+3] = B[js+i+3][ks+j+3];
         WB[i+3][j+4] = B[js+i+3][ks+j+4];
```

```
      WB[i+3][j+5] = B[js+i+3][ks+j+5];
      WB[i+3][j+6] = B[js+i+3][ks+j+6];
      WB[i+3][j+7] = B[js+i+3][ks+j+7];
      WB[i+3][j+8] = B[js+i+3][ks+j+8];
      WB[i+3][j+9] = B[js+i+3][ks+j+9];
   }
}
for(j=0; j<NBLC; j++){
   for(i=0; i<NBLC; i=i+5){
      s = 0.0; s2 = 0.0; s3 = 0.0; s4 = 0.0; s5 = 0.0;
      for(k=0; k<NBLC; k=k+10){
         s = s+WA[i][k]*WB[k][j]
              +WA[i][k+1]*WB[k+1][j]
              +WA[i][k+2]*WB[k+2][j]
              +WA[i][k+3]*WB[k+3][j]
              +WA[i][k+4]*WB[k+4][j]
              +WA[i][k+5]*WB[k+5][j]
              +WA[i][k+6]*WB[k+6][j]
              +WA[i][k+7]*WB[k+7][j]
              +WA[i][k+8]*WB[k+8][j]
              +WA[i][k+9]*WB[k+9][j];

         s2 = s2+WA[i+1][k]*WB[k][j]
                +WA[i+1][k+1]*WB[k+1][j]
                +WA[i+1][k+2]*WB[k+2][j]
                +WA[i+1][k+3]*WB[k+3][j]
                +WA[i+1][k+4]*WB[k+4][j]
                +WA[i+1][k+5]*WB[k+5][j]
                +WA[i+1][k+6]*WB[k+6][j]
                +WA[i+1][k+7]*WB[k+7][j]
                +WA[i+1][k+8]*WB[k+8][j]
                +WA[i+1][k+9]*WB[k+9][j];

         s3 = s3+WA[i+2][k]*WB[k][j]
                +WA[i+2][k+1]*WB[k+1][j]
                +WA[i+2][k+2]*WB[k+2][j]
                +WA[i+2][k+3]*WB[k+3][j]
                +WA[i+2][k+4]*WB[k+4][j]
                +WA[i+2][k+5]*WB[k+5][j]
                +WA[i+2][k+6]*WB[k+6][j]
                +WA[i+2][k+7]*WB[k+7][j]
                +WA[i+2][k+8]*WB[k+8][j]
                +WA[i+2][k+9]*WB[k+9][j];

         s4 = s4+WA[i+3][k]*WB[k][j]
                +WA[i+3][k+1]*WB[k+1][j]
                +WA[i+3][k+2]*WB[k+2][j]
                +WA[i+3][k+3]*WB[k+3][j]
                +WA[i+3][k+4]*WB[k+4][j]
                +WA[i+3][k+5]*WB[k+5][j]
                +WA[i+3][k+6]*WB[k+6][j]
                +WA[i+3][k+7]*WB[k+7][j]
                +WA[i+3][k+8]*WB[k+8][j]
                +WA[i+3][k+9]*WB[k+9][j];
```

```
                    s5 = s5+WA[i+4][k]*WB[k][j]
                           +WA[i+4][k+1]*WB[k+1][j]
                           +WA[i+4][k+2]*WB[k+2][j]
                           +WA[i+4][k+3]*WB[k+3][j]
                           +WA[i+4][k+4]*WB[k+4][j]
                           +WA[i+4][k+5]*WB[k+5][j]
                           +WA[i+4][k+6]*WB[k+6][j]
                           +WA[i+4][k+7]*WB[k+7][j]
                           +WA[i+4][k+8]*WB[k+8][j]
                           +WA[i+4][k+9]*WB[k+9][j];
                  }
                  WC[i][j] = s;
                  WC[i+1][j] = s2;
                  WC[i+2][j] = s3;
                  WC[i+3][j] = s4;
                  WC[i+4][j] = s5;
              }
            }
            /* Add WC to C */
            for(i=0; i<NBLC; i++){
               for(j=0; j<NBLC; j++)C[is+i][ks+j] += WC[i][j];
            }
         }
      }
   }
 }

int main(){
   int n;
   double sec, flops;
   clock_t tic, toc;

   set_mat(A, B, C); /* Matrices Setting */
   flops = 2.0*NMAX*NMAX*NMAX;

   printf("*** Type ***   Time[sec]    Speed[MFLOPS]    Check[Error]\n");

   /* Block Matrix Multiplication */
   tic = Timer;
   multmats(A, B);
   toc = Timer;
   sec = (double)(toc - tic)/CLOCKS_PER_SEC;
   printf(" multmats    %10.2f %13d %17.1e\n", sec, mflops(flops, sec), check(C, D));

   return 0;
}

int mflops(double flops, double sec){
   int mf;

   if (sec > 1.0e-14) mf = (int)(flops/sec/1000000);
   else mf = 0;
```

```
   return mf;
}

void set_mat(double A[NMAX][NMAX], double B[NMAX][NMAX], double C[NMAX][NMAX]){
   int i, j;

   srand( (unsigned)time( NULL ) );

   for(i=0; i<NMAX; i++){
      for(j=0; j<NMAX; j++){
         A[i][j] = (double)rand()/1000.0;
         B[i][j] = (double)rand()/1000.0;
         C[i][j] = 0.0;
      }
   }
}

double check(double C[NMAX][NMAX], double D[NMAX][NMAX]){
   int i, j;
   double temp, max_dif;

   max_dif = 0.0;
   for(i=0; i<NMAX; i++){
      for(j=0; j<NMAX; j++){
         temp = fabs(C[i][j] - D[i][j]);
         if(temp > max_dif) max_dif = temp;
      }
   }
   return max_dif;
}
```

実行結果は、１２回行ったものの中で最小値と最大値を除いた１０回分で平均を取る。
実行環境は
CPU：Pentium III 600Mhz
cash: 256kB
OS：Win2k
である。

	Time (sec)	*Speed* (MFLOPS)
	6.93	288
	6.92	289
	6.93	288
	6.94	288
	6.93	288
	6.93	288
	6.93	288
	6.94	288
	6.94	288
	6.93	288
average	6.93	288

3　考察

　ホームページのプログラムをそのままで実行（最高速化オプションでコンパイル）すると、9秒台だった。
これを速くするためにまず行ったのは、もとのプログラムではブロック化して計算するのに、関数を呼び出していたが、この処理を multmats 関数の中でやらせた。また、C[NMAX][NMAX] の初期化を違う関数内でやらせた。１秒ほど縮まった。
次に、アンローリングをほどこした。これでは２秒ほど縮まった。
ループの変換では (i,j,k) の順をいろいろと試し、上記のような順にしたが、それほど大きな差はなかった。大きく効いたのは、やはりコンパイルオプションであった。つぎに、ブロック化、アンローリングだったと思われる。