# ACM40640 High Performance Comp Assignment 1

Ian Towey

04128591

June 24, 2018

#### Abstract

OpenMP code analysis

## Contents

0	Mat	trix-Matrix Multiplication	2
1	$\mathbf{Q2}$		4
2	2.1	Parallelize code using synchronous directives	<b>5</b> 5
$\mathbf{A}_{]}$	ppen	dices	7
$\mathbf{A}_{\mathbf{j}}$	ppen	dices	7

# 0 Matrix-Matrix Multiplication

	Matrix Dim: 5						
No. Threads	Time Taken (s)	Relative speedup					
1	0.136818	1					
5	0.032769	4.17522658610272					
10	0.03897	3.5108545034642					
15	0.061737	2.21614266971184					
20	0.109617	1.24814581679849					
25	0.094764	1.44377611751298					
30	0.085122	1.60731655741172					
35	0.074294	1.841575362748					
40	0.061368	2.22946812671099					
	Matrix Dim: 10	000					
No. Threads	Time Taken (s)	Relative speedup					
1	8.32384	1					
5	1.789278	4.6520663641983					
10	0.98218	8.47486204158097					
15	0.670694	12.4107864391213					
20	0.627197	13.2714920511418					
25	0.687821	12.1017532177703					
30	0.638659	13.0333088549602					
35	0.546503	15.2310966270999					
40	0.499063	16.6789363266762					
Matrix Dim : 2000							
No. Threads	Time Taken (s)	000 Relative speedup					
1	Time Taken (s) 81.919144	Relative speedup					
1 5	Time Taken (s) 81.919144 17.612401	Relative speedup  1 4.65121955831008					
1 5 10	Time Taken (s) 81.919144 17.612401 9.42893	Relative speedup  1 4.65121955831008 8.68806365091267					
1 5 10 15	Time Taken (s) 81.919144 17.612401 9.42893 6.309168	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985					
1 5 10 15 20	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987					
1 5 10 15 20 25	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423					
1 5 10 15 20 25 30	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403					
1 5 10 15 20 25 30 35	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797					
1 5 10 15 20 25 30	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067					
1 5 10 15 20 25 30 35 40	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649 Matrix Dim: 30	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067					
1 5 10 15 20 25 30 35 40	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649 Matrix Dim: 36 Time Taken (s)	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067					
1 5 10 15 20 25 30 35 40 <b>No. Threads</b>	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649 Matrix Dim: 30 Time Taken (s) 379.905421	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067  000  Relative speedup  1					
1 5 10 15 20 25 30 35 40  No. Threads 1 5	Time Taken (s)  81.919144  17.612401  9.42893  6.309168  4.834022  5.673005  4.915481  4.304687  4.444649  Matrix Dim: 30  Time Taken (s)  379.905421  72.595633	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067  000  Relative speedup  1 5.23317182178162					
1 5 10 15 20 25 30 35 40  No. Threads 5 10	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649 Matrix Dim: 30 Time Taken (s) 379.905421 72.595633 38.912887	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067  000  Relative speedup  1 5.23317182178162 9.76297186584999					
1 5 10 15 20 25 30 35 40  No. Threads 1 5 10 15	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649 Matrix Dim: 30 Time Taken (s) 379.905421 72.595633 38.912887 26.133908	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067  000 Relative speedup  1 5.23317182178162 9.76297186584999 14.5368775691718					
1 5 10 15 20 25 30 35 40  No. Threads 1 5 10 15 20	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649  Matrix Dim: 36 Time Taken (s) 379.905421 72.595633 38.912887 26.133908 19.820675	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067  000  Relative speedup  1 5.23317182178162 9.76297186584999 14.5368775691718 19.1671283142476					
1 5 10 15 20 25 30 35 40  No. Threads 1 5 10 15 20 25	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649 Matrix Dim: 30 Time Taken (s) 379.905421 72.595633 38.912887 26.133908 19.820675 21.94133	1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067  000 Relative speedup  1 5.23317182178162 9.76297186584999 14.5368775691718 19.1671283142476 17.3146031256993					
1 5 10 15 20 25 30 35 40  No. Threads 1 5 10 15 20 25 30	Time Taken (s)  81.919144  17.612401  9.42893  6.309168  4.834022  5.673005  4.915481  4.304687  4.444649  Matrix Dim: 30  Time Taken (s)  379.905421  72.595633  38.912887  26.133908  19.820675  21.94133  19.670081	Relative speedup  1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067  000 Relative speedup  1 5.23317182178162 9.76297186584999 14.5368775691718 19.1671283142476 17.3146031256993 19.3138717120687					
1 5 10 15 20 25 30 35 40  No. Threads 1 5 10 15 20 25	Time Taken (s) 81.919144 17.612401 9.42893 6.309168 4.834022 5.673005 4.915481 4.304687 4.444649 Matrix Dim: 30 Time Taken (s) 379.905421 72.595633 38.912887 26.133908 19.820675 21.94133	1 4.65121955831008 8.68806365091267 12.9841437095985 16.9463738476987 14.4401677770423 16.6655397508403 19.0302207802797 18.4309591151067  000 Relative speedup  1 5.23317182178162 9.76297186584999 14.5368775691718 19.1671283142476 17.3146031256993					

Matrix Dim: 4000					
No. Threads	Time Taken (s)	Relative speedup			
1	732.608332	1			
5	183.152083	4			
10	102.377162	7.15597422010976			
15	67.713204	10.8192832228113			
20	51.827723	14.1354527961801			
25	55.588827	13.1790572231359			
30	50.389406	14.5389356643736			
35	43.283283	16.9258956627666			
40	42.753439	17.1356585373167			
Matrix Dim: 5000					
	Matrix Dim: 50	000			
No. Threads	Matrix Dim: 50 Time Taken (s)	000 Relative speedup			
No. Threads					
	Time Taken (s)				
1	Time Taken (s) 1548.988168	Relative speedup			
1 5	Time Taken (s) 1548.988168 387.247042	Relative speedup  1 4			
1 5 10	Time Taken (s) 1548.988168 387.247042 209.308222	Relative speedup  1  4  7.40051276151015			
1 5 10 15	Time Taken (s) 1548.988168 387.247042 209.308222 141.228929	Relative speedup  1  4  7.40051276151015  10.9679240575421			
1 5 10 15 20	Time Taken (s) 1548.988168 387.247042 209.308222 141.228929 107.211277	Relative speedup  1  4  7.40051276151015  10.9679240575421  14.4479966225941			
1 5 10 15 20 25	Time Taken (s) 1548.988168 387.247042 209.308222 141.228929 107.211277 111.409326	Relative speedup  1 4 7.40051276151015 10.9679240575421 14.4479966225941 13.9035772283552			

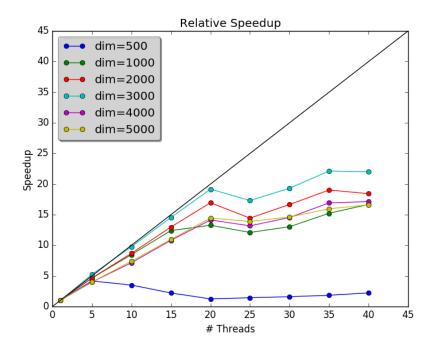


Figure 1: Relative Speed

1 Q2

## 2 Q3

### 2.1 Parallelize code using synchronous directives

```
void main_q1(){
     int niter, i, j;
     long seed;
     double count;
     double x,y,z,pi;
     extern float ran2();
     niter = 10000;
    count = 0;
10
    \#pragma\ omp\ parallel\ \textbf{for}\ ordered\ private(i)\ shared(count,x,y,z,
11
     for (i=1;i<=niter;i++){
      #pragma omp critical
13
14
15
            seed{=}i\;;
            x=ran2(\&seed);
16
           y=ran2(&seed);
17
18
            z \!\!=\!\! x \! *\! x \!\!+\!\! y \! *\! y \; ;
19
            if(z<1){
              count+=1;
20
21
22
23
     pi=count *4.0 / niter;
     printf("The value of pi is %8.14f\n",pi);
25
26 }
```

#### 2.2 Ran2 thread safe

7 #define IA1 40014 8 #define IA2 40692 9 #define IQ1 53668 10 #define IQ2 52774 11 #define IR1 12211

```
1 #ifndef MAIN_H_INCLUDED
2 #define MAIN_H_INCLUDED
4 #include <stdlib.h>
5 #include <stdlib.h>
6 #include <stdio.h>
7 #include <omp.h>
9 typedef struct {
      long seed;
      float val;
11
12 } rand_tuple;
13
14
#endif // MAIN_H_INCLUDED
1 #include <main.h>
3 #define IM1 2147483563
4 #define IM2 2147483399
5 #define AM (1.0/IM1)
6 #define IMM1 (IM1-1)
```

```
12 #define IR2 3791
13 #define NTAB 32
<sup>14</sup> #define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
<sup>16</sup> #define RNMX (1.0-EPS)
17
18 rand_tuple *ran3(long seed) {
19
     int j;
20
      long k;
21
      static long idum2=123456789;
      static long iy=0;
22
      static long iv [NTAB];
23
24
      float temp;
      rand_tuple *resp = (rand_tuple*) malloc(sizeof(rand_tuple));
25
      resp \rightarrow seed = seed;
      resp \rightarrow val = 0;
27
      if (resp->seed <= 0) {
28
         if (-(resp \rightarrow seed) < 1) resp\rightarrow seed = 1;
         else resp->seed = -(resp->seed); idum2=(resp->seed);
30
         for (j=NTAB+7; j>=0; j--) {
31
            k=(resp->seed)/IQ1;
32
          \begin{array}{l} resp \mathop{\rightarrow}\! seed = IA1*(resp \mathop{\rightarrow}\! seed - k*IQ1) - k*IR1\,;\\ if\ (resp \mathop{\rightarrow}\! seed\ <\ 0)\ resp \mathop{\rightarrow}\! seed\ +=\ IM1\,; \end{array}
33
34
           if (j < NTAB) iv [j] = resp \rightarrow seed;
35
36
         iy=iv [0];
37
38
39
      k = (resp - seed)/IQ1;
40
      resp\mathop{->}seed\mathop{=}IA1*(resp\mathop{->}seed\mathop{-}k*IQ1)\mathop{-}k*IR1;
41
      if (resp->seed < 0) resp->seed += IM1;
      k=idum2/IQ2;
43
      idum2=IA2*(idum2-k*IQ2)-k*IR2;
44
45
      if (idum2 < 0) idum2 += IM2;
      j=iy/NDIV;
46
      iy=iv[j]-idum2;
47
48
      iv[j] = resp \rightarrow seed;
      if (iy < 1) iy += IMM1;
49
50
      if ((temp=AM*iy) > RNMX) {
51
        resp \rightarrow val = RNMX;
52
53
54
      else {
        resp \rightarrow val = temp;
55
56
57
58
      return resp ;
59
60 }
   void main_q2(){
 2
      int niter, i, j;
      long seed;
      double count;
      double x,y,z,pi;
      extern rand_tuple *ran3();
      niter = 10000;
     count=0;
10
     \#pragma \ omp \ parallel \ for \ ordered \ private(i) \ shared(count,x,y,z,
11
   for (i=1; i \le niter; i++){
```

```
#pragma omp critical
13
14
            seed=i;
15
            rand_tuple *ran_val1 = ran3(seed);
16
           x = ran_val1 \rightarrow val;
17
           rand_tuple *ran_val2=ran3(ran_val1->seed);
y = ran_val2->val;
18
19
            free(ran_val1); free(ran_val2);
20
            z=x*x+y*y;
21
            if (z<1){
22
23
              count+=1;
24
       }
25
     }
26
     pi=count *4.0 / niter;
27
     printf("The value of pi is %8.14f\n",pi);
28
29 }
```

# Appendices