HIGH PERFORMANCE COMPUTING PROFILING

Implementation of Durand Kerner Method to solve Polynomial Equations

Durand Kerner

$$r_{n+1} = r_n - \frac{p(r_n)}{(r_n - s_n)(r_n - t_n)}$$

$$s_{n+1} = s_n - \frac{p(s_n)}{(s_n - r_n)(s_n - t_n)}$$

$$t_{n+1} = t_n - \frac{p(t_n)}{(t_n - r_n)(t_n - s_n)}$$

-Report By:
Sharan SK
CED18I049

Serial Code:

```
#include <stdio.h>
#include <math.h>
#include <complex.h>
#include<omp.h>

#define M_PI 3.14159265358979323846
#define coff_size 500

double R=0;
double complex z[coff_size];
double complex deltaZ[coff_size];
double deltaZMax;
double epsilon = 1e-6;
double complex QsubJ,fz;
int max_iter = 1000;


//----------------------Function Prototypes-------------------------------
void durand_kerner(); //Prototypes
void calc_theta();
double max_cof();
void printz();
void update_z();
void update_fz();


int main() {

   double complex cList[coff_size];  //List of coefficients
   double complex z;
   double x,y; //x for real and y for imaginary parts of the coefficient
   int n=0; //n is number degree of polynomial


//------Read Coefficients-------------------------------------------------
```

```
   printf("Enter coefficients and enter any char other than number when
done:\n");
   while(scanf("%lf %lf",&x,&y) == 2)  { //Read coefficients from stdin
       cList[n] = (x + y*I);
       n++;
   }
   x = 1;  //Cn = 1, because the equation has to be normalized
   y = 0;
   z = (x + y*I);
   cList[n] = z; //Store in cList[]


   durand_kerner(cList,n);


}
//----------------------------------Function
Definition-------------------------------


void durand_kerner(double complex cList[],int n) {
   float st;
   st=omp_get_wtime();
   R = 1 + max_cof(cList,n);  //End Equation 5

   calc_theta(n);
   int k;
   for(k=1;k <= max_iter;k++) {

       //printz(cList,n,k);

       deltaZMax = 0;

       update_fz(cList,n);
       update_z(n);

       if(deltaZMax <= epsilon) {
```

```
            break;
        }

    }
    st=omp_get_wtime()-st;
    printf("%d\n",k);
    printz(cList,n);
    printf("Time Taken=%f\n",st);

}

void calc_theta(int n) {
    for(int j=0;j < n;j++) {
        z[j] = ( cos( j*((2*M_PI)/n) ) + (I*sin( j*((2*M_PI)/n) )) )*R;
    }

}

double max_cof(double complex cList[],int n)
{
    double r;
    for(int j=0;j < n;j++) {
        if(cabs(cList[j]) > R) {
            r = cabs(cList[j]);
        }
    }

    return r;
}

void printz(double complex cList[],int n)
{
        printf("Final Output:(Note: if the roots repeat then there exist
less than n-1 roots for the equation)\n");
        for(int i=0;i < n;i++) {
                    printf("z[%d] = %0.10f +
%0.10f*I\n",i,creal(z[i]),cimag(z[i]));
```

```
                fflush(stdout);
            }
}

void update_z(int n)
{
   for(int j=0;j < n;j++) {
            z[j] = z[j] + deltaZ[j];
       }
}

void update_fz(double complex cList[],int n)
{
   for(int j=0;j < n;j++) {

            QsubJ = 1;
            for(int i=0;i < n;i++) {
                if(i != j) {
                    QsubJ = (z[j]-z[i])*QsubJ;
                }
            }
            fz = 1;
            for(int k = n-1;k >= 0;k--) {
                fz = fz*z[j] + cList[k];

            }

            deltaZ[j] = (-fz/QsubJ);

            if(cabs(deltaZ[j]) > deltaZMax) {
                deltaZMax = cabs(deltaZ[j]);
            }
       }
}
```

Profiling:

Three types of profiling are followed in this report.Namely

- ➢ Function Profiling
- ➢ Line Profiling
- ➢ Hardware Profiling

Function Profiling:

For function profiling we will be using GPROF.GPROF helps to identify the number of times each function is called.Using the following commands,we can get a GPROF report.

Commands Used:

- ➢ Use -pg command during gcc compilation to enable profiling
- ➢ Execute the binary file to generate the profiling data in the form of gmon.out file

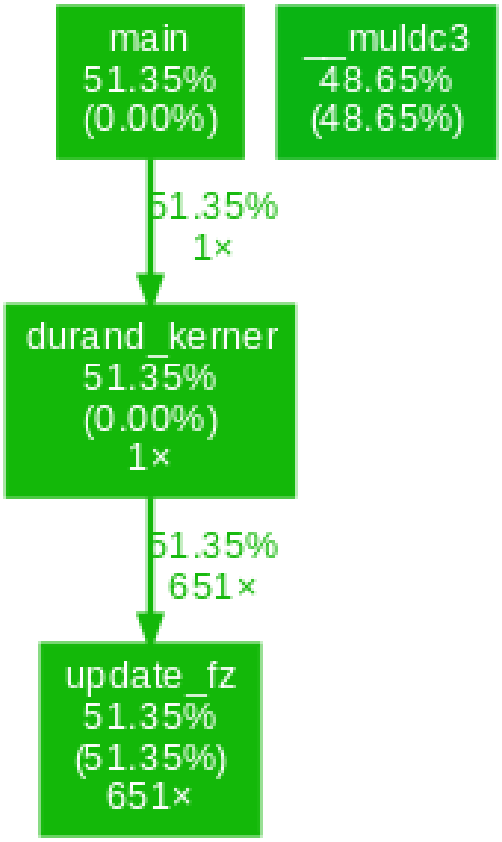
Following is the gprof report:

Flat profile:

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 51.49      0.19     0.19      651     0.29     0.29  update_fz
 48.78      0.37     0.18                             __muldc3
  0.00      0.37     0.00      651     0.00     0.00  update_z
  0.00      0.37     0.00        1     0.00     0.00  calc_theta
  0.00      0.37     0.00        1     0.00   190.53  durand_kerner
  0.00      0.37     0.00        1     0.00     0.00  max_cof
  0.00      0.37     0.00        1     0.00     0.00  printz
```

Call Graph :

```
Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 2.70% of 0.37 seconds

index % time    self  children    called     name
                0.19    0.00     651/651         durand_kerner [2]
[1]     51.4    0.19    0.00     651         update_fz [1]
-----------------------------------------------
                0.00    0.19       1/1           main [3]
[2]     51.4    0.00    0.19       1         durand_kerner [2]
                0.19    0.00     651/651         update_fz [1]
                0.00    0.00     651/651         update_z [5]
                0.00    0.00       1/1           max_cof [7]
                0.00    0.00       1/1           calc_theta [6]
                0.00    0.00       1/1           printz [8]
-----------------------------------------------
                                                <spontaneous>
[3]     51.4    0.00    0.19                 main [3]
                0.00    0.19       1/1           durand_kerner [2]
-----------------------------------------------
                                                <spontaneous>
[4]     48.6    0.18    0.00                 __muldc3 [4]
-----------------------------------------------
                0.00    0.00     651/651         durand_kerner [2]
[5]      0.0    0.00    0.00     651         update_z [5]
-----------------------------------------------
                0.00    0.00       1/1           durand_kerner [2]
[6]      0.0    0.00    0.00       1         calc_theta [6]
-----------------------------------------------
                0.00    0.00       1/1           durand_kerner [2]
[7]      0.0    0.00    0.00       1         max_cof [7]
-----------------------------------------------
                0.00    0.00       1/1           durand_kerner [2]
[8]      0.0    0.00    0.00       1         printz [8]
-----------------------------------------------
```



Observation:

- ➢ It is to be noted that the predominant operations are update_fz and __muldc3 which is called 51% and 48% of the time respectively .
- ➢ __muldc3 is not a user-built function but rather of the c programming inbuilt function which is invoked when two complex numbers are being multiplied together.The Runtime Function is of the form : complex double __muldc3 (double a, double b, double c, double d) which returns the product of a + ib and c + id, following the rules of C99 Annex G.

Line Based Profiling:

GCOV helps to find the no of times each line is executed ,number of times a branch has been taken and other informations related to each line of the code. Following commands can be used to generate GCOV report:

- ➢ Enable profiling using -fprofile-arcs -ftest-coverage during compilation
- ➢ Execute binary file to generate the data
- ➢ Execute gcov <filename> with -b -c parameters to generate coverage data
- ➢ Open <filename>.gcov to see the report

```
(base) sharan@Omen:~/Desktop/Sem7/HPC/Project$ cat durand-kerner.c.gcov
        -:    0:Source:durand-kerner.c
        -:    0:Graph:durand-kerner.gcno
        -:    0:Data:durand-kerner.gcda
        -:    0:Runs:1
        -:    1:#include <stdio.h>
        -:    2:#include <math.h>
        -:    3:#include <complex.h>
        -:    4:#include<omp.h>
        -:    5:
        -:    6:#define M PI 3.14159265358979323846
        -:    7:#define coff size 500
        -:    8:
        -:    9:double R=0;
        -:   10:double complex z[coff size];
        -:   11:double complex deltaZ[coff size];
        -:   12:double deltaZMax;
        -:   13:double epsilon = 1e-6;
        -:   14:double complex QsubJ,fz;
        -:   15:int max iter = 1000;
        -:   16:
        -:   17:
        -:   18://---------------------Function Prototypes-----------------------------
        -:   19:void durand kerner(); //Prototypes
        -:   20:void calc theta();
        -:   21:double max cof();
        -:   22:void printz();
        -:   23:void update z();
        -:   24:void update fz();
        -:   25:
        -:   26:
        1:   27:int main() {
        -:   28:
        -:   29:        double complex cList[coff size];  //List of coefficients
        -:   30:        double complex z;
        -:   31:        double x,y; //x for real and y for imaginary parts of the coefficient
        1:   32:        int n=0; //n is number degree of polynomial
        -:   33:
        -:   34:
        -:   35://------Read Coefficients-------------------------------------------------
        1:   36:        printf("Enter coefficients and enter any char other than number when done:\n");
      201:   37:        while(scanf("%lf %lf",&x,&y) == 2)  { //Read coefficients from stdin
      200:   38:                cList[n] = (x + y*I);
      200:   39:                n++;
        -:   40:        }
        1:   41:        x = 1;  //Cn = 1, because the equation has to be normalized
        1:   42:        y = 0;
        1:   43:        z = (x + y*I);
        1:   44:        cList[n] = z; //Store in cList[]
        -:   45:
        -:   46:
        1:   47:        durand kerner(cList,n);
```

```
    -:   46:
    1:   47:        durand kerner(cList,n);
    -:   48:
    -:   49:
    -:   50:}
    -:   51://---------------------------------Function Definition-----------------------------
    -:   52:
    -:   53:
    1:   54:void durand kerner(double complex cList[],int n) {
    -:   55:        float st;
    1:   56:        st=omp get wtime();
    1:   57:        R = 1 + max cof(cList,n);  //End Equation 5
    -:   58:
    1:   59:        calc theta(n);
    -:   60:        int k;
  651:   61:        for(k=1;k <= max iter;k++) {
    -:   62:
    -:   63:                //printz(cList,n,k);
    -:   64:
  651:   65:                deltaZMax = 0;
    -:   66:
  651:   67:                update fz(cList,n);
  651:   68:                update z(n);
    -:   69:
  651:   70:                if(deltaZMax <= epsilon) {
    1:   71:                        break;
    -:   72:                }
    -:   73:
    -:   74:        }
    1:   75:        st=omp get wtime()-st;
    1:   76:        printf("%d\n",k);
    1:   77:        printz(cList,n);
    1:   78:        printf("Time Taken=%f\n",st);
    -:   79:
    1:   80:}
    -:   81:
    1:   82:void calc theta(int n) {
  201:   83:        for(int j=0;j < n;j++) {
  200:   84:                z[j] = ( cos( j*((2*M PI)/n) ) + (I*sin( j*((2*M PI)/n) )) )*R;
    -:   85:        }
    -:   86:
    1:   87:}
    -:   88:
    1:   89:double max cof(double complex cList[],int n)
    -:   90:{
    -:   91:        double r;
  201:   92:        for(int j=0;j < n;j++) {
  200:   93:                if(cabs(cList[j]) > R) {
  200:   94:                        r = cabs(cList[j]);
    -:   95:                }
    -:   96:        }
    -:   97:
```

```
      1:   89:double max cof(double complex cList[],int n)
      -:   90:{
      -:   91:        double r;
    201:   92:        for(int j=0;j < n;j++) {
    200:   93:                if(cabs(cList[j]) > R) {
    200:   94:                        r = cabs(cList[j]);
      -:   95:                }
      -:   96:        }
      -:   97:
      1:   98:        return r;
      -:   99:}
      -:  100:
      1:  101:void printz(double complex cList[],int n)
      -:  102:{
      1:  103:                printf("Final Output:(Note: if the roots repeat then there exist less than n-1 roots for the equation)\n");
    201:  104:                for(int i=0;i < n;i++) {
    200:  105:                        printf("z[%d] = %0.10f + %0.10f*I\n",i,creal(z[i]),cimag(z[i]));
    200:  106:                fflush(stdout);
      -:  107:                }
      1:  108:}
      -:  109:
    651:  110:void update z(int n)
      -:  111:{
 130851:  112:        for(int j=0;j < n;j++) {
 130200:  113:                        z[j] = z[j] + deltaZ[j];
      -:  114:                }
    651:  115:}
      -:  116:
    651:  117:void update fz(double complex cList[],int n)
      -:  118:{
 130851:  119:        for(int j=0;j < n;j++) {
      -:  120:
 130200:  121:                        QsubJ = 1;
26170200:  122:                        for(int i=0;i < n;i++) {
26040000:  123:                                if(i != j) {
25909800:  124:                                        QsubJ = (z[j]-z[i])*QsubJ;
      -:  125:                                }
      -:  126:                        }
 130200:  127:                        fz = 1;
26170200:  128:                        for(int k = n-1;k >= 0;k--) {
26040000:  129:                                fz = fz*z[j] + cList[k];
      -:  130:
      -:  131:                        }
      -:  132:
 130200:  133:                        deltaZ[j] = (-fz/QsubJ);
      -:  134:
 130200:  135:                        if(cabs(deltaZ[j]) > deltaZMax) {
  44584:  136:                                deltaZMax = cabs(deltaZ[j]);
      -:  137:                        }
      -:  138:                }
    651:  139:}
```

Observation:

- From the above gcov report,we can see that update_fz has lines of code that are being run for the highest no of time.For example:updating fz value executed more no of times.
- Second most executed lines are updating z value and QsubJ value which belongs to update_fz and update_z function.

Hardware Profiling :

LIKWID tool is used for hardware profiling.Following Commands are used to generate hardware related data:

- ➢ likwid-topology: to generate hardware data
- ➢ likwid-perfctr -c 0-3 -g <Hardware/Option> <binary file>: To generate execution of the program related to Hardware.Example Giving option as L2 gives data related to execution of program and how L2 cache is affected during execution.

Hardware Specification:

```
(base) sharan@Omen:~$ likwid-topology
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz
CPU type:       Intel Coffeelake processor
CPU stepping:   9
********************************************************************************
Hardware Thread Topology
********************************************************************************
Sockets:                1
Cores per socket:       4
Threads per core:       1
--------------------------------------------------------------------------------
HWThread        Thread          Core            Socket          Available
0               0               0               0               *
1               0               1               0               *
2               0               2               0               *
3               0               3               0               *
--------------------------------------------------------------------------------
Socket 0:               ( 0 1 2 3 )
--------------------------------------------------------------------------------
********************************************************************************
Cache Topology
********************************************************************************
Level:                  1
Size:                   32 kB
Cache groups:           ( 0 ) ( 1 ) ( 2 ) ( 3 )
--------------------------------------------------------------------------------
Level:                  2
Size:                   256 kB
Cache groups:           ( 0 ) ( 1 ) ( 2 ) ( 3 )
--------------------------------------------------------------------------------
Level:                  3
Size:                   6 MB
Cache groups:           ( 0 1 2 3 )
--------------------------------------------------------------------------------
```

```
********************************************************************************
NUMA Topology
********************************************************************************
NUMA domains:           1
--------------------------------------------------------------------------------
Domain:                 0
Processors:             ( 0 1 2 3 )
Distances:              10
Free memory:            4769.25 MB
Total memory:           15965.7 MB
--------------------------------------------------------------------------------
```

Likwid FLOPS_DP Report:

```
(base) sharan@Omen:~/Desktop/Sem7/HPC/Project$ sudo likwid-perfctr -c 0-7 -g FLOPS_DP ./dk < input.in
[sudo] password for sharan:
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz
CPU type:       Intel Coffeelake processor
CPU clock:      2.50 GHz
--------------------------------------------------------------------------------
Sleeping longer as likwid sleep() called without prior initialization
Sleeping longer as likwid sleep() called without prior initialization
Enter coefficients and enter any char other than number when done:
495
Time Taken=0.295464
--------------------------------------------------------------------------------
Group 1: FLOPS_DP
+----------------------------------------+---------+-----------+-----------+-----------+-----------+
|                 Event                  | Counter |   Core 0  |   Core 1  |   Core 2  |   Core 3  |
+----------------------------------------+---------+-----------+-----------+-----------+-----------+
|           INSTR_RETIRED_ANY            |  FIXC0  | 135775957 | 119238819 | 2468335368 | 204992325 |
|         CPU_CLK_UNHALTED_CORE          |  FIXC1  | 163749715 | 133500282 |  909249738 | 187059720 |
|         CPU_CLK_UNHALTED_REF           |  FIXC2  | 132371720 | 108792320 |  706702360 | 150110792 |
| FP_ARITH_INST_RETIRED_128B_PACKED_DOUBLE |  PMC0  |      1060 |      8343 |       2112 |      4820 |
|   FP_ARITH_INST_RETIRED_SCALAR_DOUBLE  |  PMC1  |    198342 |     20038 |  315201691 |     39002 |
| FP_ARITH_INST_RETIRED_256B_PACKED_DOUBLE |  PMC2  |         2 |         0 |          0 |         0 |
+----------------------------------------+---------+-----------+-----------+-----------+-----------+

+----------------------------------------+---------+-----------+-----------+-----------+--------------+
|                 Event                  | Counter |    Sum    |    Min    |    Max    |      Avg     |
+----------------------------------------+---------+-----------+-----------+-----------+--------------+
|           INSTR_RETIRED_ANY_STAT       |  FIXC0  | 2928342469 | 119238819 | 2468335368 | 7.320856e+08 |
|         CPU_CLK_UNHALTED_CORE_STAT     |  FIXC1  | 1393559455 | 133500282 |  909249738 | 3.483899e+08 |
|         CPU_CLK_UNHALTED_REF_STAT      |  FIXC2  | 1097977192 | 108792320 |  706702360 |    274494298 |
| FP_ARITH_INST_RETIRED_128B_PACKED_DOUBLE_STAT |  PMC0  |      16335 |      1060 |       8343 |    4083.7500 |
|   FP_ARITH_INST_RETIRED_SCALAR_DOUBLE_STAT  |  PMC1  |  315459073 |     20038 |  315201691 | 7.886477e+07 |
| FP_ARITH_INST_RETIRED_256B_PACKED_DOUBLE_STAT |  PMC2  |          2 |         0 |          2 |       0.5000 |
+----------------------------------------+---------+-----------+-----------+-----------+--------------+
```

```
+----------------------+-------------+----------+----------+----------+
|        Metric        |    Core 0   |  Core 1  |  Core 2  |  Core 3  |
+----------------------+-------------+----------+----------+----------+
|  Runtime (RDTSC) [s] |      0.3056 |   0.3056 |   0.3056 |   0.3056 |
| Runtime unhalted [s] |      0.0656 |   0.0535 |   0.3643 |   0.0749 |
|      Clock [MHz]     |   3087.6345 | 3062.8413 | 3211.3467 | 3110.3475 |
|          CPI         |      1.2060 |   1.1196 |   0.3684 |   0.9125 |
|     DP [MFLOP/s]     |      0.6560 |   0.1202 | 1031.5049 |   0.1592 |
|   AVX DP [MFLOP/s]   | 2.617983e-05 |        0 |        0 |        0 |
|   Packed [MUOPS/s]   |      0.0035 |   0.0273 |   0.0069 |   0.0158 |
|   Scalar [MUOPS/s]   |      0.6491 |   0.0656 | 1031.4910 |   0.1276 |
|  Vectorization ratio |      0.5326 |  29.3964 |   0.0007 |  10.9990 |
+----------------------+-------------+----------+----------+----------+

+--------------------------+-------------+----------+-------------+-------------+
|          Metric          |     Sum     |    Min   |     Max     |     Avg     |
+--------------------------+-------------+----------+-------------+-------------+
|  Runtime (RDTSC) [s] STAT |      1.2224 |   0.3056 |      0.3056 |      0.3056 |
| Runtime unhalted [s] STAT |      0.5583 |   0.0535 |      0.3643 |      0.1396 |
|      Clock [MHz] STAT     |  12472.1700 | 3062.8413 |   3211.3467 |   3118.0425 |
|          CPI STAT         |      3.6065 |   0.3684 |      1.2060 |      0.9016 |
|     DP [MFLOP/s] STAT     |   1032.4403 |   0.1202 |   1031.5049 |    258.1101 |
|   AVX DP [MFLOP/s] STAT   | 2.617983e-05 |        0 | 2.617983e-05 | 6.544957e-06 |
|   Packed [MUOPS/s] STAT   |      0.0535 |   0.0035 |      0.0273 |      0.0134 |
|   Scalar [MUOPS/s] STAT   |   1032.3333 |   0.0656 |   1031.4910 |    258.0833 |
|  Vectorization ratio STAT |     40.9287 |   0.0007 |     29.3964 |     10.2322 |
+--------------------------+-------------+----------+-------------+-------------+
```

LIkwid L3 Report:

```
(base) sharan@Omen:~/Desktop/Sem7/HPC/Project$ sudo likwid-perfctr -c 0-7 -g L3 ./dk < input.in
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz
CPU type:       Intel Coffeelake processor
CPU clock:      2.50 GHz
--------------------------------------------------------------------------------
Sleeping longer as likwid sleep() called without prior initialization
Sleeping longer as likwid sleep() called without prior initialization
Enter coefficients and enter any char other than number when done:
495
Time Taken=0.292505
--------------------------------------------------------------------------------
Group 1: L3
+----------------------+---------+----------+----------+------------+----------+
|        Event         | Counter |   Core 0 |  Core 1  |   Core 2   |  Core 3  |
+----------------------+---------+----------+----------+------------+----------+
|   INSTR RETIRED ANY  |  FIXC0  | 151683170 | 80226953 | 2454470654 | 131935662 |
| CPU CLK UNHALTED CORE |  FIXC1  | 139069053 | 91748366 |  893507653 | 103578560 |
|  CPU CLK UNHALTED REF |  FIXC2  | 109833568 | 72960888 |  685560512 |  81989856 |
|    L2 LINES IN ALL   |   PMC0  |   5043529 |  4731644 |     333723 |   4019020 |
|     L2 TRANS L2 WB   |   PMC1  |    858862 |   867494 |      68571 |    789864 |
+----------------------+---------+----------+----------+------------+----------+
```

```
+---------------------------+---------+------------+----------+------------+--------------+
|           Event           | Counter |     Sum    |    Min   |     Max    |      Avg     |
+---------------------------+---------+------------+----------+------------+--------------+
|   INSTR RETIRED ANY STAT  |  FIXC0  | 2818316439 | 80226953 | 2454470654 | 7.045791e+08 |
| CPU CLK UNHALTED CORE STAT |  FIXC1  | 1227903632 | 91748366 |  893507653 |    306975908 |
|  CPU CLK UNHALTED REF STAT |  FIXC2  |  950344824 | 72960888 |  685560512 |    237586206 |
|    L2 LINES IN ALL STAT    |   PMC0  |   14127916 |   333723 |    5043529 |      3531979 |
|     L2 TRANS L2 WB STAT    |   PMC1  |    2584791 |    68571 |     867494 |  646197.7500 |
+---------------------------+---------+------------+----------+------------+--------------+

+-------------------------------+-----------+-----------+-----------+-----------+
|             Metric            |   Core 0  |   Core 1  |   Core 2  |   Core 3  |
+-------------------------------+-----------+-----------+-----------+-----------+
|      Runtime (RDTSC) [s]      |    0.2947 |    0.2947 |    0.2947 |    0.2947 |
|      Runtime unhalted [s]     |    0.0557 |    0.0368 |    0.3580 |    0.0415 |
|          Clock [MHz]          | 3160.1109 | 3138.4495 | 3252.8153 | 3152.9470 |
|              CPI              |    0.9168 |    1.1436 |    0.3640 |    0.7851 |
|  L3 load bandwidth [MBytes/s] | 1095.1393 | 1027.4173 |   72.4638 |  872.6799 |
|  L3 load data volume [GBytes] |    0.3228 |    0.3028 |    0.0214 |    0.2572 |
| L3 evict bandwidth [MBytes/s] |  186.4911 |  188.3655 |   14.8893 |  171.5091 |
| L3 evict data volume [GBytes] |    0.0550 |    0.0555 |    0.0044 |    0.0506 |
|    L3 bandwidth [MBytes/s]    | 1281.6304 | 1215.7828 |   87.3531 | 1044.1890 |
|    L3 data volume [GBytes]    |    0.3778 |    0.3583 |    0.0257 |    0.3078 |
+-------------------------------+-----------+-----------+-----------+-----------+

+-----------------------------------+------------+----------+----------+----------+
|               Metric              |     Sum    |    Min   |    Max   |    Avg   |
+-----------------------------------+------------+----------+----------+----------+
|      Runtime (RDTSC) [s] STAT     |     1.1788 |   0.2947 |   0.2947 |   0.2947 |
|      Runtime unhalted [s] STAT    |     0.4920 |   0.0368 |   0.3580 |   0.1230 |
|          Clock [MHz] STAT         | 12704.3227 | 3138.4495 | 3252.8153 | 3176.0807 |
|              CPI STAT             |     3.2095 |   0.3640 |   1.1436 |   0.8024 |
|  L3 load bandwidth [MBytes/s] STAT |  3067.7003 |  72.4638 | 1095.1393 |  766.9251 |
|  L3 load data volume [GBytes] STAT |     0.9042 |   0.0214 |   0.3228 |   0.2260 |
| L3 evict bandwidth [MBytes/s] STAT |   561.2550 |  14.8893 |  188.3655 |  140.3137 |
| L3 evict data volume [GBytes] STAT |     0.1655 |   0.0044 |   0.0555 |   0.0414 |
|    L3 bandwidth [MBytes/s] STAT    |  3628.9553 |  87.3531 | 1281.6304 |  907.2388 |
|    L3 data volume [GBytes] STAT    |     1.0696 |   0.0257 |   0.3778 |   0.2674 |
+-----------------------------------+------------+----------+----------+----------+

(base) sharan@Omen:~/Desktop/Sem7/HPC/Project$
```

Observation:

- From the above report it can be seen that the CPI from both L3 and FLOPS_DP report that are not equal across each core ,which confirms that the program is yet to be optimizied which is expected from the serial code.
- The L3 data volume and bandwidth also indicates that the load is not balanced across each core
- The Clock across each core seems to be stable