HPC Week 3 - Project Profling

Week-3

Profiling

Roll no. - CED19I011
Name - Sandeep Ahirwar

Project : Parallelize Fitness calculation in Genetic algorithm

Week_3-Serial Code Profiling

- Submit a report on following profiling techniques for your chosen project:

1. Functional profiling - gprof
2. Line Profiling - gconv
3. Hardware resource profiling- likwid

Serial Code

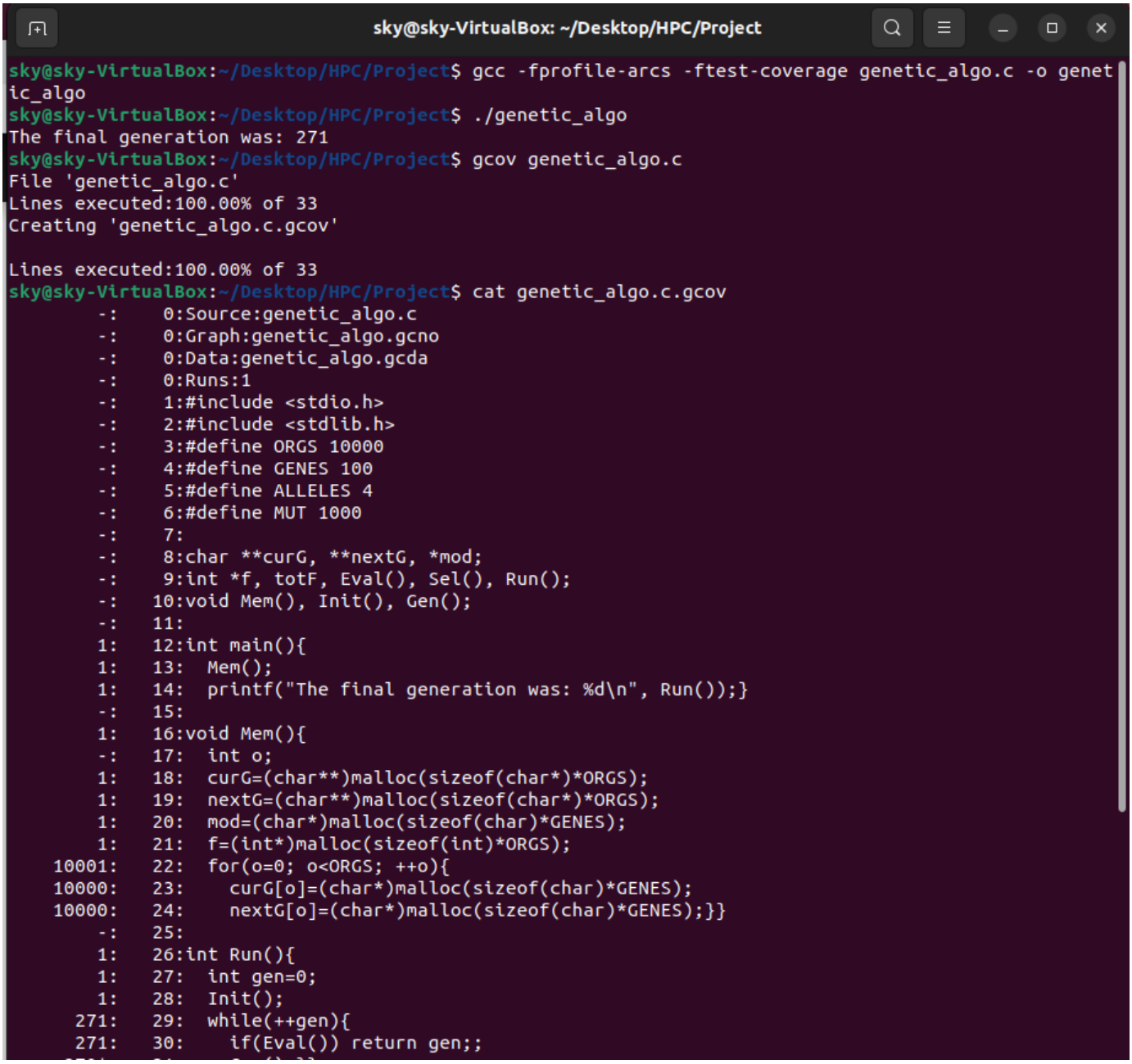
```
#include <stdio.h>
#include <stdlib.h>
#define ORGS 10000
#define GENES 100
#define ALLELES 4
#define MUT 1000

char **curG, **nextG, *mod;
int *f, totF, Eval(), Sel(), Run();
void Mem(), Init(), Gen();

int main(){
  Mem();
  printf("The final generation was: %d\n", Run());}

void Mem(){
  int o;
  curG=(char**)malloc(sizeof(char*)*ORGS);
  nextG=(char**)malloc(sizeof(char*)*ORGS);
  mod=(char*)malloc(sizeof(char)*GENES);
```

```
  f=(int*)malloc(sizeof(int)*ORGS);
  for(o=0; o<ORGS; ++o){
    curG[o]=(char*)malloc(sizeof(char)*GENES);
    nextG[o]=(char*)malloc(sizeof(char)*GENES);}}

int Run(){
  int gen=0;
  Init();
  while(++gen){
    if(Eval()) return gen;;
    Gen();}}

void Init(){
  int o, g;
  for(o=0; o<ORGS; ++o) for(g=0; g<GENES; ++g) curG[o][g]=rand()%ALLELES;
  for(g=0; g<GENES; ++g) mod[g]=rand()%ALLELES;}

int Eval(){
  int o, g, curF;
  for(totF=0, o=0; o<ORGS; ++o){
    for(curF=0, g=0; g<GENES; ++g) if(curG[o][g]==mod[g]) ++curF;
    if(curF==GENES) return 1;
    totF += f[o]=curF;}
  return 0;}

void Gen(){
  int o, g, p1, p2, cp;
  for(o=0;o<ORGS;++o) for(p1=Sel(), p2=Sel(),
cp=rand()%GENES,g=0;g<GENES;++g)
      nextG[o][g]=(rand()%MUT)? ((g<cp)? curG[p1][g]: curG[p2][g]):
rand()%ALLELES;
  for(o=0; o<ORGS; ++o) for(g=0; g<GENES; ++g) curG[o][g]=nextG[o][g];}

int Sel(){
  int o, tot=0, pt=rand()%(totF+1);
  for(o=0; o<ORGS; ++o) if((tot+=f[o])>=pt) return o;}
```

Functional Profiling

```
sky@sky-VirtualBox:~/Desktop/HPC/Project$ gcc -pg -o TestGprof genetic_algo.c
sky@sky-VirtualBox:~/Desktop/HPC/Project$ ./TestGprof
The final generation was: 271
sky@sky-VirtualBox:~/Desktop/HPC/Project$ gprof -b TestGprof gmon.out > analysis.out
sky@sky-VirtualBox:~/Desktop/HPC/Project$ gprof -b -q TestGprof
                     Call graph


granularity: each sample hit covers 4 byte(s) for 0.02% of 54.18 seconds

index % time    self  children    called     name
                0.00   54.12       1/1           main [2]
[1]     99.9    0.00   54.12       1         Run [1]
                1.89   50.95     270/270         Gen [3]
                1.28    0.00     271/271         Eval [5]
                0.00    0.00       1/1           Init [7]
-----------------------------------------------
                                                <spontaneous>
[2]     99.9    0.00   54.12                 main [2]
                0.00   54.12       1/1           Run [1]
                0.00    0.00       1/1           Mem [8]
-----------------------------------------------
                1.89   50.95     270/270         Run [1]
[3]     97.5    1.89   50.95     270         Gen [3]
               50.95    0.00 5400000/5400000     Sel [4]
-----------------------------------------------
               50.95    0.00 5400000/5400000     Gen [3]
[4]     94.0   50.95    0.00 5400000         Sel [4]
-----------------------------------------------
                1.28    0.00     271/271         Run [1]
[5]      2.4    1.28    0.00     271         Eval [5]
-----------------------------------------------
                                                <spontaneous>
[6]      0.1    0.06    0.00                 _init [6]
-----------------------------------------------
                0.00    0.00       1/1           Run [1]
[7]      0.0    0.00    0.00       1         Init [7]
-----------------------------------------------
                0.00    0.00       1/1           main [2]
[8]      0.0    0.00    0.00       1         Mem [8]
-----------------------------------------------


Index by function name

   [5] Eval                    [8] Mem                     [6] _init
   [3] Gen                     [1] Run
   [7] Init                    [4] Sel
sky@sky-VirtualBox:~/Desktop/HPC/Project$ gprof -b -p TestGprof
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 94.04     50.95    50.95  5400000     0.00     0.00  Sel
  3.49     52.84     1.89      270     0.01     0.20  Gen
  2.36     54.12     1.28      271     0.00     0.00  Eval
  0.11     54.18     0.06                             _init
  0.00     54.18     0.00        1     0.00     0.00  Init
  0.00     54.18     0.00        1     0.00     0.00  Mem
  0.00     54.18     0.00        1     0.00    54.12  Run
sky@sky-VirtualBox:~/Desktop/HPC/Project$
```

Line Profiling



```
      271:   30:    if(Eval()) return gen;;
     270*:   31:    Gen();}}
        -:   32:
        1:   33:void Init(){
        -:   34:  int o, g;
  1010001:   35:  for(o=0; o<ORGS; ++o) for(g=0; g<GENES; ++g) curG[o][g]=rand()%ALLELES;
      101:   36:  for(g=0; g<GENES; ++g) mod[g]=rand()%ALLELES;}
        -:   37:
      271:   38:int Eval(){
        -:   39:  int o, g, curF;
  2702172:   40:  for(totF=0, o=0; o<ORGS; ++o){
272892102:   41:    for(curF=0, g=0; g<GENES; ++g) if(curG[o][g]==mod[g]) ++curF;
  2701902:   42:    if(curF==GENES) return 1;
  2701901:   43:    totF += f[o]=curF;}
      270:   44:  return 0;}
        -:   45:
      270:   46:void Gen(){
        -:   47:  int o, g, p1, p2, cp;
272700270:   48:  for(o=0;o<ORGS;++o) for(p1=Sel(), p2=Sel(), cp=rand()%GENES,g=0;g<GENES;++g)
270000000:   49:      nextG[o][g]=(rand()%MUT)? ((g<cp)? curG[p1][g]: curG[p2][g]): rand()%ALLELES;
272700270:   50:  for(o=0; o<ORGS; ++o) for(g=0; g<GENES; ++g) curG[o][g]=nextG[o][g];}
        -:   51:
  5400000:   52:int Sel(){
  5400000:   53:  int o, tot=0, pt=rand()%(totF+1);
26986862383*:   54:  for(o=0; o<ORGS; ++o) if((tot+=f[o])>=pt) return o;}
sky@sky-VirtualBox:~/Desktop/HPC/Project$
```

Hardware Resource Profiling

```
--------------------------------------------------------------------------------
CPU name:        Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz
CPU type:        Intel Kabylake processor
CPU stepping:    12
********************************************************************************
Hardware Thread Topology
********************************************************************************
Sockets:                1
Cores per socket:       1
Threads per core:       1
--------------------------------------------------------------------------------
HWThread        Thread          Core            Socket          Available
0               0               0               0               *
--------------------------------------------------------------------------------
Socket 0:               ( 0 )
--------------------------------------------------------------------------------
********************************************************************************
Cache Topology
```

```
********************************************************************************
****
Level:                  1
Size:                   32 kB
Cache groups:           ( 0 )
--------------------------------------------------------------------------------
----
Level:                  2
Size:                   256 kB
Cache groups:           ( 0 )
--------------------------------------------------------------------------------
----
Level:                  3
Size:                   6 MB
Cache groups:           ( 0 )
--------------------------------------------------------------------------------
----
********************************************************************************
****
NUMA Topology
********************************************************************************
****
NUMA domains:           1
--------------------------------------------------------------------------------
----
Domain:                 0
Processors:             ( 0 )
Distances:              10
Free memory:            777.762 MB
Total memory:           3924.27 MB
--------------------------------------------------------------------------------
----


********************************************************************************
****
Graphical Topology
********************************************************************************
****
Socket 0:
+------------+
| +--------+ |
| |    0   | |
```

```
| +--------+ |
| +--------+ |
| |  32 kB | |
| +--------+ |
| +--------+ |
| | 256 kB | |
| +--------+ |
| +--------+ |
| |  6 MB  | |
| +--------+ |
+------------+
```

Observation

Here we did profiling of our project serial code. We performed line profiling as well as function profiling.
in line profiling we get the result for every line of our code.
From line profiling we can observe that 'for loops' and 'sel' are executed more than a million times.
In function profiling we get the result for every function present in our code.
From function profiling we can see that 'sel' function is taking most of the time.

Thank You