HPC Assignment 2
Profiling Report

Recommendation System

(Collaborative Filtering)

Source code and output files can be found here:
[ced18i042-recommendation-system-1]

Roll No: CED18I042
Name: Reuben Skariah Mathew

Date: 20th September, 2021

Hardware Configuration:

Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Sockets: 1
Cores per Socket: 4
Threads per Core: 2
L1 Cache: 32 kB
L2 Cache: 256 kB
L3 Cache: 6 MB
RAM: 8 GB

Serial Code:

```
#include <vector>
#include <queue>
#include <string>
#include <cmath>
#include <vector>
#include <iostream>
#include <fstream>
#include <assert.h>
#include <functional>

using namespace std;

vector<string> moviesList;

void topRatings(vector<vector<double>> ratingsMat, int user)
{
    std::priority_queue<pair<double, int>> q;
    for (int i = 0; i < ratingsMat[user].size(); ++i)
    {
        q.push(pair<double, int>(ratingsMat[user][i], i));
    }
    int k = 5; // number of movies to be shown
    cout << "\nTop rated movies by User " << user << endl;
    for (int i = 0; i < k; ++i)
    {
        int ki = q.top().second;
        printf("%s\n", moviesList[ki].c_str());
        q.pop();
    }
}

void makeRec(vector<vector<double>> predict, int user)
{
```

```
    std::priority_queue<pair<double, int>> q;
    for (int i = 0; i < predict[user].size(); ++i)
    {
        q.push(pair<double, int>(predict[user][i], i));
    }
    int k = 5; // number of recommendations to be shown
    cout << "\nRecomendations for User " << user << endl;
    for (int i = 0; i < k; ++i)
    {
        int ki = q.top().second;
        printf("%s\n", moviesList[ki].c_str());

        q.pop();
    }
}

vector<vector<double>> matRead(string file, int row, int col)
{
    ifstream input(file);
    if (!input.is_open())
    {
        cerr << "File is not existing, check the path: \n"
             << file << endl;
        exit(1);
    }
    vector<vector<double>> data(row, vector<double>(col, 0));
    for (int i = 0; i < row; ++i)
    {
        for (int j = 0; j < col; ++j)
        {
            input >> data[i][j];
        }
    }
    return data;
}

vector<string> movieRead(string file)
{
    vector<string> movies;
    ifstream input(file);
    if (!input.is_open())
    {
        cerr << "File is not existing, check the path: \n"
             << file << endl;
        exit(1);
    }
    string str;
    while (getline(input, str))
    {
        if (str.size() > 0)
```

```
            movies.push_back(str);
    }
    return movies;
}

void matWrite(vector<vector<double>> mat, string file)
{
    ofstream output(file);
    int row = mat.size();
    int col = mat[0].size();

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
            output << mat[i][j] << " ";
        output << endl;
    }
}

double norm(vector<double> A)
{
    double res = 0;
    for (int i = 0; i < A.size(); ++i)
    {
        res += pow(A[i], 2);
    }
    return sqrt(res);
}

double dotProduct(vector<double> A, vector<double> B)
{
    double res = 0;
    for (int i = 0; i < A.size(); ++i)
    {
        res += A[i] * B[i];
    }
    return res;
}

double adjCosineSimilarity(vector<double> A, vector<double> B) //adjusted cosine
similarity (cosine similarity - mean)
{
    double A_mean = 0;
    double B_mean = 0;
    for (int i = 0; i < A.size(); ++i)
    {
        A_mean += A[i];
        B_mean += B[i];
    }
    A_mean /= A.size();
```

```
    B_mean /= B.size();
    vector<double> C(A);
    vector<double> D(B);
    for (int i = 0; i < A.size(); ++i)
    {
        C[i] = A[i] - A_mean;
        D[i] = B[i] - B_mean;
    }
    return dotProduct(C, D) / (norm(C) * norm(D)); //if output is nan then there
is no correlation
}

void checkCommon(vector<double> A, vector<double> B, vector<double> &C,
vector<double> &D) //to check if both A and B have rated
{
    for (int i = 0; i < A.size(); ++i)
    {
        if (A[i] && B[i])
        {
            C.push_back(A[i]);
            D.push_back(B[i]);
        }
    }
}

vector<vector<double>> colabFilter(vector<vector<double>> ratingsMat, int
usersNum, int itemsNum)
{
    vector<vector<double>> predict(usersNum, vector<double>(itemsNum, 0));
    for (int i = 0; i < usersNum; i++) //Make predictions for each user
    {
        for (int j = 0; j < itemsNum; j++) //Find item j that user i has not
scored, and predict user i's score for item j
        {
            if (ratingsMat[i][j]) //if movie has already been rated by the user
                continue;
            else //If item j has not been rated by user i, find out users who
have rated item j
            {
                vector<double> cosSim;
                vector<double> ratingsOld;
                for (int k = 0; k < usersNum; k++) //If user k has rated item j,
calculate the cosSimilarity between user k and user i
                {
                    if (ratingsMat[k][j]) //Find user k who has rated item j
                    {
                        vector<double> commonA, commonB; //  Store the scores of
the two items that have been jointly rated in two vectors respectively
                        checkCommon(ratingsMat[i], ratingsMat[k], commonA,
commonB); //  check if item has been rated by both users
```

```
                        if (!commonA.empty()) //If the two have jointly rated
items, calculate the cosine similarity
                        {
                            cosSim.push_back(adjCosineSimilarity(commonA,
commonB)); //cosine similarity
                            ratingsOld.push_back(ratingsMat[k][j]); //old ratings
                        }
                    }
                }
                double cosSimSum = 0; //dot product of ratingsOld and cosSim
                if (!cosSim.empty())
                {
                    for (int m = 0; m < cosSim.size(); m++)
                    {
                        cosSimSum += cosSim[m];
                    }
                    predict[i][j] = dotProduct(cosSim, ratingsOld) / (cosSimSum);
                    cout << "user " << i << " item " << j << " with predicted
rating " << predict[i][j] << endl;
                }
            }
        }
    }
    return predict;
}

int main()
{
    string file1("ratings.txt");
    string file2("movies.txt");

    int row = 268;
    int col = 450;
    vector<vector<double>> ratingsMat = matRead(file1, row, col);
    moviesList = movieRead(file2);
    vector<vector<double>> predict = colabFilter(ratingsMat, row, col);
    matWrite(predict, "predict.txt");

    int uid, check;
    do
    {
        cout << "\nEnter User ID:" << endl;
        cin >> uid;
        topRatings(ratingsMat, uid);
        makeRec(predict, uid);
        cout << "\nRecommend for another user? (1 = Yes, 0 = No)" << endl;
        cin >> check;
    } while (check == 1);

    return 0; }
```

Output:

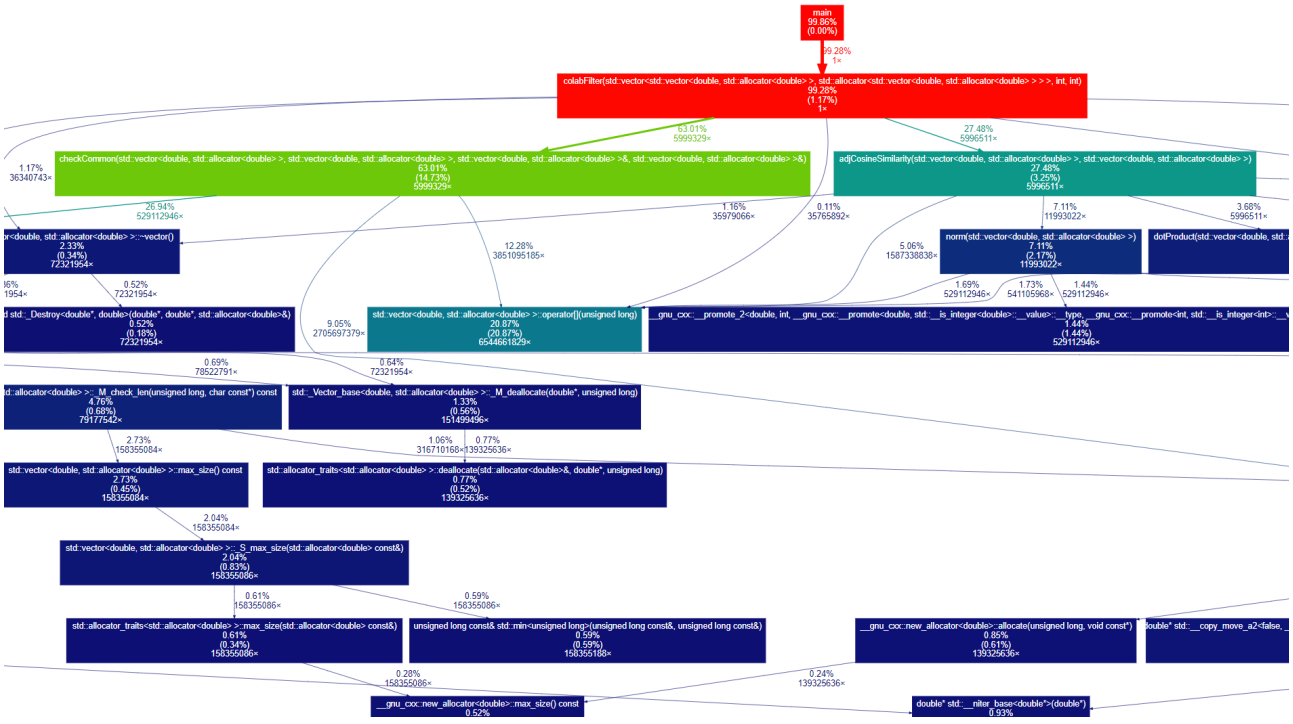
```
Enter User ID:
123

Top rated movies by User 123
Inception (2010)
Inglourious Basterds (2009)
Hot Fuzz (2007)
Kill Bill: Vol. 2 (2004)
Eternal Sunshine of the Spotless Mind (2004)
Kill Bill: Vol. 1 (2003)
Scarface (1983)

Recomendations for User 123
Amelie (Fabuleux destin d'Am◆lie Poulain, Le) (2001)
Chinatown (1974)
Graduate, The (1967)
L.A. Confidential (1997)
Wallace & Gromit: The Wrong Trousers (1993)
Sting, The (1973)
Annie Hall (1977)

Recommend for another user? (1 = Yes, 0 = No)
0
→  recommeder-system
```

Profiling

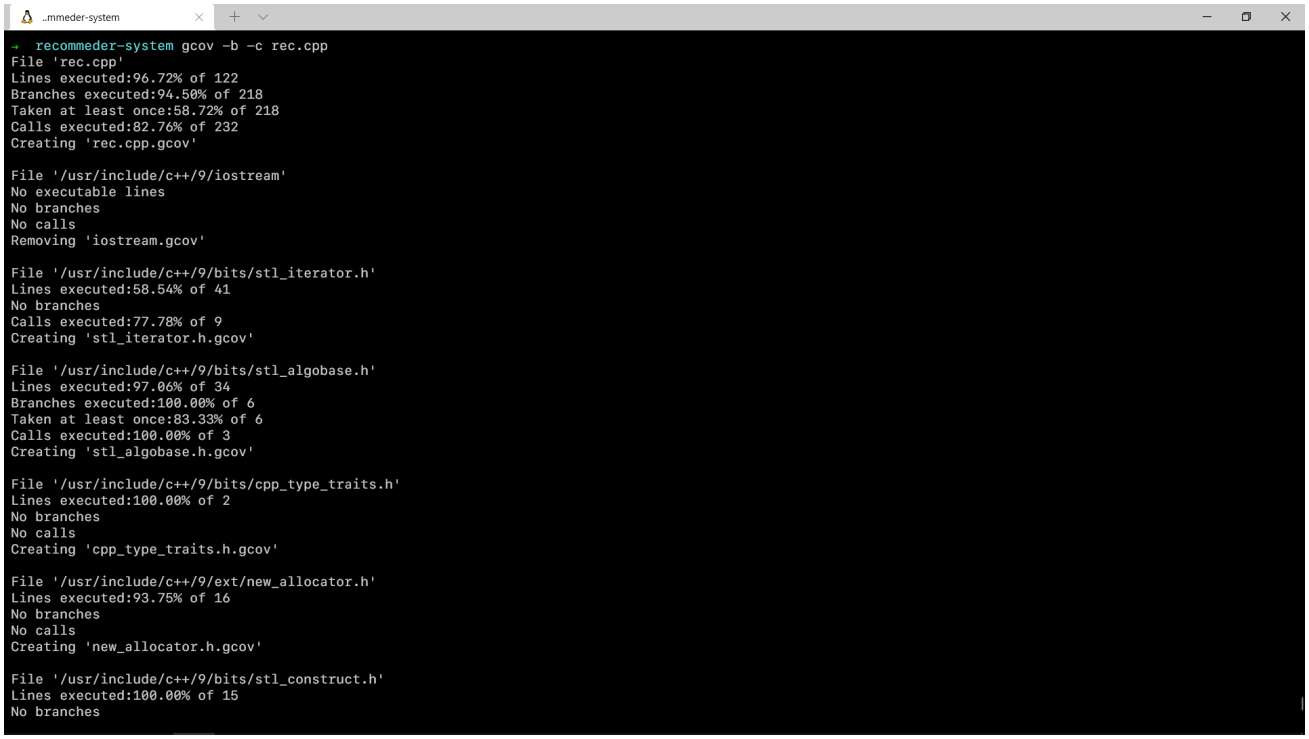
- ## Functional Profiling

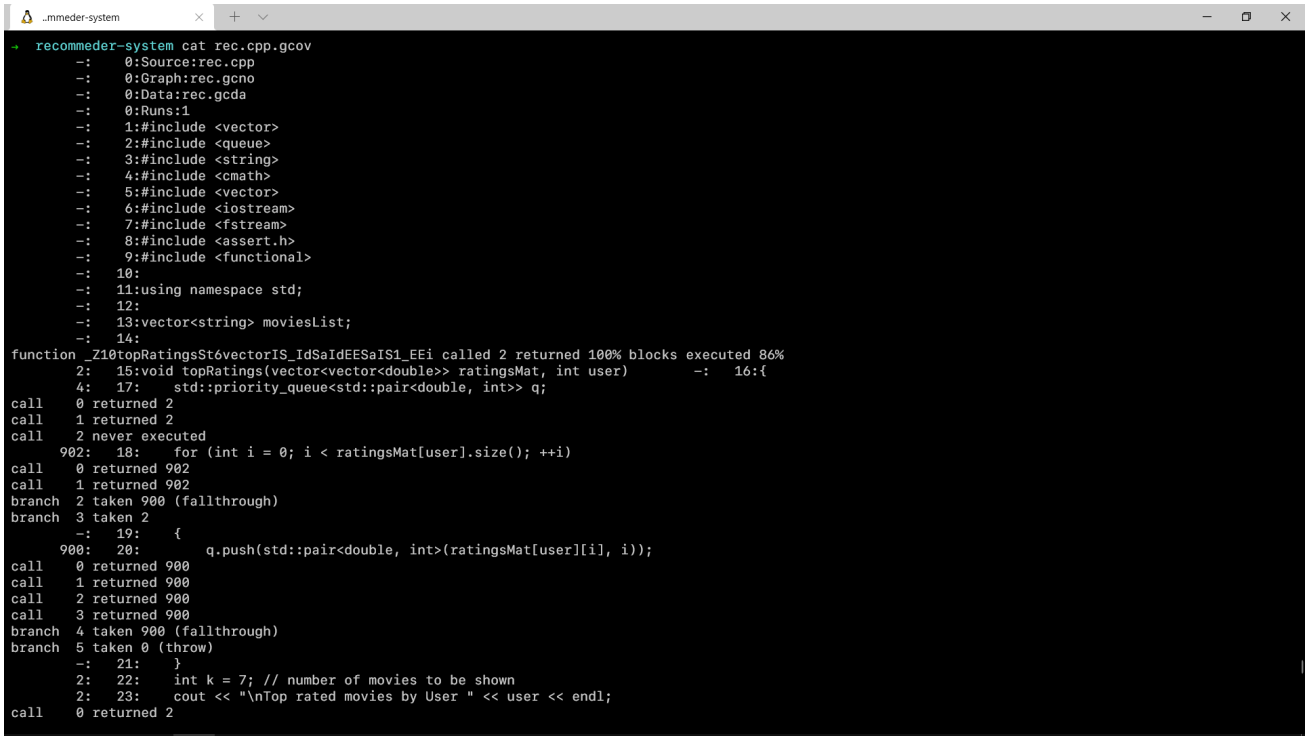
```
..mmeder-system
→  recommeder-system gprof rec > rec.gprof
→  recommeder-system python3 gprof2dot.py < rec.gprof | dot -Tsvg -o gprof-output.svg
→  recommeder-system gprof -b rec
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 20.94      9.25     9.25 6544661829     0.00     0.00  std::vector<double, std::allocator<double> >::operator[](unsigned long)
 14.95     15.86     6.61 4459483610     0.00     0.00  std::vector<double, std::allocator<double> >::size() const
 14.78     22.39     6.53  5999329     0.00     0.00  checkCommon(std::vector<double, std::allocator<double> >, std::vector<double, std::allocator<double> >
, std::vector<double, std::allocator<double> >&, std::vector<double, std::allocator<double> >&)
  3.26     23.83     1.44  5996511     0.00     0.00  adjCosineSimilarity(std::vector<double, std::allocator<double> >, std::vector<double, std::allocator<d
ouble> >)
  2.69     25.02     1.19 535109457     0.00     0.00  std::vector<double, std::allocator<double> >::push_back(double const&)
  2.57     26.16     1.14 535109457     0.00     0.00  void __gnu_cxx::new_allocator<double>::construct<double, double const&>(double*, double const&)
  2.35     27.20     1.04 535109457     0.00     0.00  void std::allocator_traits<std::allocator<double> >::construct<double, double const&>(std::allocator<
double>&, double*, double const&)
  2.23     28.19     0.99 78522791     0.00     0.00  void std::vector<double, std::allocator<double> >::_M_realloc_insert<double const&>(__gnu_cxx::__norma
l_iterator<double*, std::vector<double, std::allocator<double> > >, double const&)
  2.17     29.15     0.96 11993022     0.00     0.00  norm(std::vector<double, std::allocator<double> >)
  2.06     30.06     0.91 1148741705     0.00     0.00  double const& std::forward<double const&>(std::remove_reference<double const&>::type&)
  1.45     30.70     0.64 529112946     0.00     0.00  __gnu_cxx::__promote_2<double, int, __gnu_cxx::__promote<double, std::__is_integer<double>::__value>:
:__type, __gnu_cxx::__promote<int, std::__is_integer<int>::__value>::__type>::__type std::pow<double, int>(double, int)
  1.44     31.33     0.64 42012844     0.00     0.00  std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std::allo
cator<double> > >::operator[](unsigned long)
  1.29     31.90     0.57 541112917     0.00     0.00  operator new(unsigned long, void*)
  1.22     32.44     0.54 158355084     0.00     0.00  double* std::__relocate_a<double*, double*, std::allocator<double> >(double*, double*, double*, std::
allocator<double>&)
  1.20     32.97     0.53 158355084     0.00     0.00  std::vector<double, std::allocator<double> >::_S_relocate(double*, double*, double*, std::allocator<d
ouble>&)
  1.18     33.49     0.52        1     0.52    43.92  colabFilter(std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<doubl
e, std::allocator<double> > >, int, int)
  1.09     33.97     0.48  6084112     0.00     0.00  dotProduct(std::vector<double, std::allocator<double> >, std::vector<double, std::allocator<double> >)

  1.00     34.41     0.44 158355084     0.00     0.00  std::enable_if<std::__is_bitwise_relocatable<double, void>::value, double*>::type std::__relocate_a_1
<double, double>(double*, double*, double*, std::allocator<double>&)
  0.93     34.82     0.41 535213346     0.00     0.00  double* std::__niter_base<double*>(double*)
  0.84     35.19     0.37 158355086     0.00     0.00  std::vector<double, std::allocator<double> >::_S_max_size(std::allocator<double> const&)
  0.68     35.49     0.30 79177542     0.00     0.00  std::vector<double, std::allocator<double> >::_M_check_len(unsigned long, char const*) const
  0.66     35.78     0.29 60148092     0.00     0.00  std::vector<double, std::allocator<double> >::vector(std::vector<double, std::allocator<double> > cons
t&)
```

```
                     Call graph


granularity: each sample hit covers 2 byte(s) for 0.02% of 44.22 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]     99.9    0.00   44.17                 main [1]
                0.52   43.40       1/1           colabFilter(std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, st
d::allocator<double> > > >, int, int) [2]
                0.00    0.08       1/1           matRead(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, int, int) [76]
           0.00    0.06       1/1           movieRead(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >) [84]
                0.00    0.05       2/2           topRatings(std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std
::allocator<double> > > >, int) [98]
                0.00    0.05       2/2           makeRec(std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std::a
llocator<double> > > >, int) [99]
                0.00    0.01       8/8           std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std::allocator
<double> > > >::~vector() [127]
                0.00    0.00       1/1           matWrite(std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std::
allocator<double> > > >, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >) [135]
                0.00    0.00       6/6           std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std::allocator
<double> > > >::vector(std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std::allocator<double> > > > const&) [13
9]
                0.00    0.00       1/1           std::vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator
<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > >::operator=(std::vector<std::__cxx11::basic_string<char, std::char_trait
s<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > >&&) [311]
                0.00    0.00       1/2           std::vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator
<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > >::~vector() [303]
-----------------------------------------------
                0.52   43.40       1/1           main [1]
[2]     99.3    0.52   43.40       1         colabFilter(std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std::a
llocator<double> > > >, int, int) [2]
                6.53   21.36 5999329/5999329     checkCommon(std::vector<double, std::allocator<double> >, std::vector<double, std::allocator<double> >, std
::vector<double, std::allocator<double> >&, std::vector<double, std::allocator<double> >&) [3]
                1.44   10.72 5996511/5996511     adjCosineSimilarity(std::vector<double, std::allocator<double> >, std::vector<double, std::allocator<double
> >) [4]
                0.12    1.47 24166882/60148092     std::vector<double, std::allocator<double> >::vector(std::vector<double, std::allocator<double> > const&)
 [9]
                0.63    0.00 41768039/42012844     std::vector<std::vector<double, std::allocator<double> >, std::allocator<std::vector<double, std::allocat
or<double> > > >::operator[](unsigned long) [30]
                0.08    0.44 36340743/72321954     std::vector<double, std::allocator<double> >::~vector() [22]
```

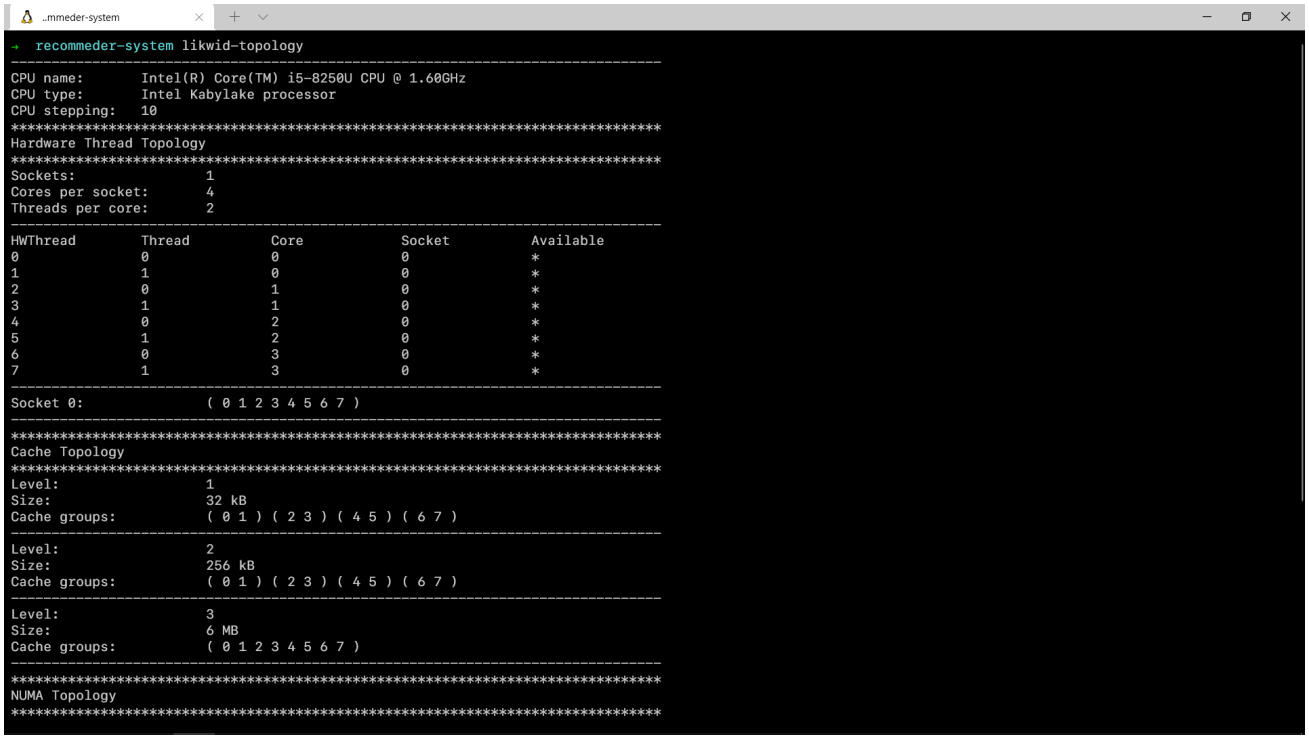


- # Line Based Profiling

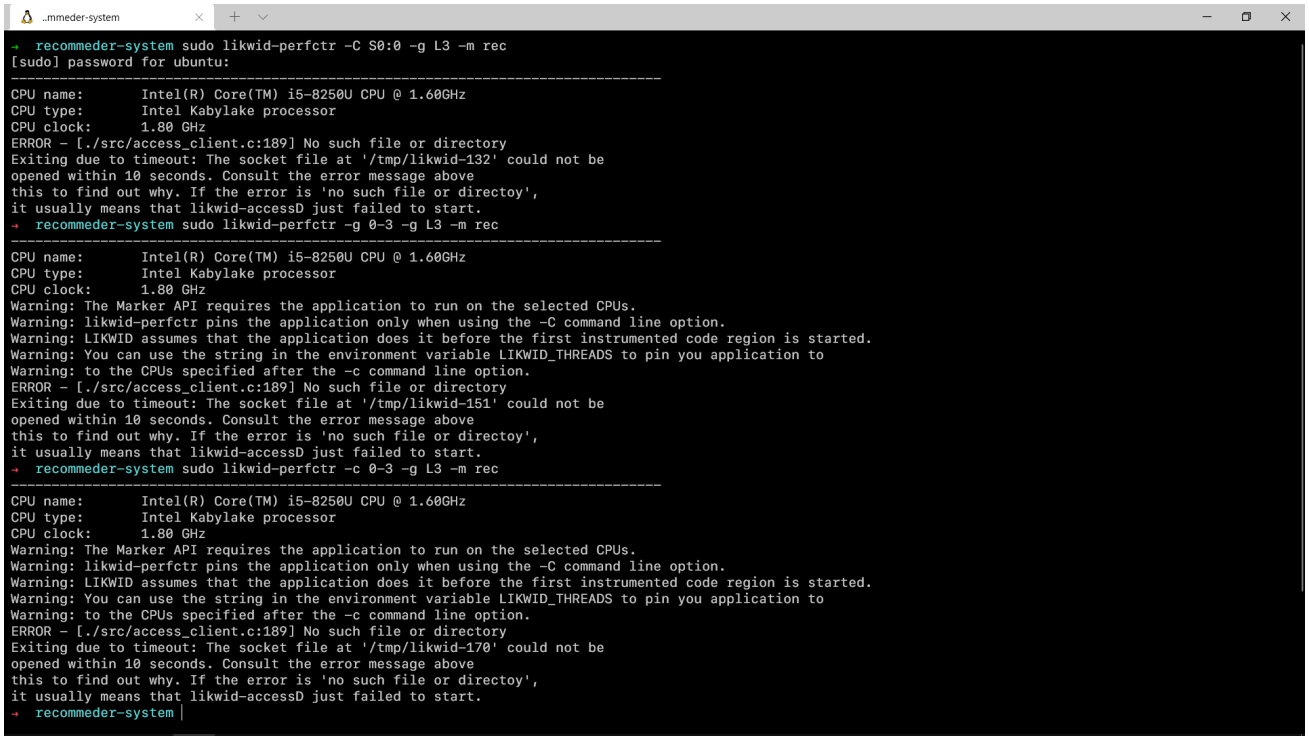




- # Hardware Profiling



```
..mmeder-system
→  recommeder-system likwid-topology
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU type:       Intel Kabylake processor
CPU stepping:   10
********************************************************************************
Hardware Thread Topology
********************************************************************************
Sockets:                1
Cores per socket:       4
Threads per core:       2
--------------------------------------------------------------------------------
HWThread        Thread          Core            Socket          Available
0               0               0               0               *
1               1               0               0               *
2               0               1               0               *
3               1               1               0               *
4               0               2               0               *
5               1               2               0               *
6               0               3               0               *
7               1               3               0               *
--------------------------------------------------------------------------------
Socket 0:               ( 0 1 2 3 4 5 6 7 )
--------------------------------------------------------------------------------
********************************************************************************
Cache Topology
********************************************************************************
Level:                  1
Size:                   32 kB
Cache groups:           ( 0 1 ) ( 2 3 ) ( 4 5 ) ( 6 7 )
--------------------------------------------------------------------------------
Level:                  2
Size:                   256 kB
Cache groups:           ( 0 1 ) ( 2 3 ) ( 4 5 ) ( 6 7 )
--------------------------------------------------------------------------------
Level:                  3
Size:                   6 MB
Cache groups:           ( 0 1 2 3 4 5 6 7 )
--------------------------------------------------------------------------------
********************************************************************************
NUMA Topology
********************************************************************************
```



```
..mmeder-system
→  recommeder-system sudo likwid-perfctr -C S0:0 -g L3 -m rec
[sudo] password for ubuntu:
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU type:       Intel Kabylake processor
CPU clock:      1.80 GHz
ERROR - [./src/access_client.c:189] No such file or directory
Exiting due to timeout: The socket file at '/tmp/likwid-132' could not be
opened within 10 seconds. Consult the error message above
this to find out why. If the error is 'no such file or directoy',
it usually means that likwid-accessD just failed to start.
→  recommeder-system sudo likwid-perfctr -g 0-3 -g L3 -m rec
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU type:       Intel Kabylake processor
CPU clock:      1.80 GHz
Warning: The Marker API requires the application to run on the selected CPUs.
Warning: likwid-perfctr pins the application only when using the -C command line option.
Warning: LIKWID assumes that the application does it before the first instrumented code region is started.
Warning: You can use the string in the environment variable LIKWID_THREADS to pin you application to
Warning: to the CPUs specified after the -c command line option.
ERROR - [./src/access_client.c:189] No such file or directory
Exiting due to timeout: The socket file at '/tmp/likwid-151' could not be
opened within 10 seconds. Consult the error message above
this to find out why. If the error is 'no such file or directoy',
it usually means that likwid-accessD just failed to start.
→  recommeder-system sudo likwid-perfctr -c 0-3 -g L3 -m rec
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU type:       Intel Kabylake processor
CPU clock:      1.80 GHz
Warning: The Marker API requires the application to run on the selected CPUs.
Warning: likwid-perfctr pins the application only when using the -C command line option.
Warning: LIKWID assumes that the application does it before the first instrumented code region is started.
Warning: You can use the string in the environment variable LIKWID_THREADS to pin you application to
Warning: to the CPUs specified after the -c command line option.
ERROR - [./src/access_client.c:189] No such file or directory
Exiting due to timeout: The socket file at '/tmp/likwid-170' could not be
opened within 10 seconds. Consult the error message above
this to find out why. If the error is 'no such file or directoy',
it usually means that likwid-accessD just failed to start.
→  recommeder-system
```

Observations:

Functional Profiling:

- From functional profiling we see that colabFilter() is called once from main.
- This inturn leads to checkCommon being called 59,99,329 times, and adjCosineSimilarity being called 59,96,511 times. Other functions that are called a large number of times are norm and dotProduct with 1,19,93,022 and 60,84,112 calls respectively.
- The predefined function .push_back() to insert values into a vector is called 53,51,09,457 times.
- We also see that the checkCommon() function takes 14.73% of the total time, and that adjCosineSimilarity() takes 3.25% of the total time.
- Hence the functions: checkCommon and adjCosineSimilarity can been seen as hotspots.

Line Based Profiling:

- From line profiling we observe that 96.72 lines are executed, 94.50% branches are executed and 82.76% calls are executed.
- We see that lines 118, 123 - 140 have a large number of iterations.

Hardware Profiling:

- likwid-topology was used to view system information.
- But likwid-perfctr was not able to be used as it was not supported by my system (on Windows Subsystem for Linux)