PARALLEL AND DISTRIBUTED SYSTEM

Project Report

TITLE

“WORD SEARCH USING OPENMP”

Under the guidance of

Prof. BALAMURGAN R

Submitted by –

SAURABH AMBAR (17BCI0058)

KUMAR ABHISHEK (17BCI0145)

VISHAL SAINI (17BCI0188)

DECLARATION

I hereby declare that the report entitled “Word Search Using Open MP” submitted by me, for the CSE4001 Parallel and Distributed Computing (EPJ) to VIT is a record of Bonafede work carried out by me under the supervision of Prof. Balamurgan R.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

Place:

Date: Signature of Candidate

ACKNOWLEDGEMENT

I would like to extend my sincere and heartfelt gratitude to my faculty Prof. Balamurgan R who has helped me in this endeavour and has always been very cooperative and without his help, cooperation, guidance and encouragement, the project couldn’t have been what it evolved to be.

I would also like to thanks my teammates who worked together with me to made this project successful.

At last but not least, gratitude to all my friends who helped me (directly or indirectly) to complete this project within a limited time frame.

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

1.1 Objective

- In this project, we developed the parallel algorithm for fast word search to determine the number of words/ presence of word in the document.

- The algorithm is designed to scale well on state-of-the-art multiprocessor/multicore systems for large inputs and large maximum word sizes.
- As we show, this pattern does not lend itself to straightforward standard parallelization techniques.

- The proposed algorithm aims to achieve three major goals to overcome the drawbacks of embarrassingly parallel solution techniques:

  (i) to impose a high degree of cache locality on a problem that, by nature, tends to exhibit nonlocal access patterns,

  (ii) to be lock free or largely reduce the need for data access locking, and

  (iii) to enable an even distribution of the overall processing load among multiple threads.

  We compare the performance of the parallel word search implementation with a sequential implementation and with an "embarrassingly" parallel implementation based on OpenMP.

2. Literature Survey: -

1. Parallel Quick Search Algorithm for the Exact String-Matching Problem Using OpenMP

The key reason of examining the behaviour of the sequential Quick Search algorithm which involve the preprocessing phase as well as searching phase is to find out the compute-intensive portions of the code, that could be parallelize using OpenMP.

Quick Search algorithm is a simplified version of Boyer Moore algorithm solves the string-matching problem. In general, the Quick Search algorithm composes from two logical phases, pre-processing and searching phase

2. Parallel Computing using OpenMP

OpenMP is beneficial for multi core programming. We can improve system performance by lots of parallelization.it is very good parallel platform to achieve high performance

3. Scalable parallel word search in multicore/multiprocessor systems

This paper presents a parallel algorithm for fast word search to determine the set of biological words of an input DNA sequence. The pattern exhibited by many sequential solutions to this problem is a repetitive execution over a large input DNA sequence, and the generation of large amounts of output data to store and retrieve the words determined by the algorithm.

4. Parallel Quicksort Algorithm using OpenMP

This paper aim to parallelization the Quicksort algorithm using multithreading (OpenMP) platform. The proposed method examined on two standard dataset (File 1: Hamlet.txt 180 KB and File 2: Moby Dick.txt 1.18 MB) with different number of threads. The fundamental idea of the proposed algorithm is to creating many additional temporary sub-arrays according to a number of characters in each word, the sizes of each one of these sub-arrays are adopted based on a number of elements with the exact same number of characters in the input array.

3. TECHNICAL SPECIFICATION:

Programming Language and Technology Used:

- C++
- OpenMp
- Intel VTune Amplifier

Overview of proposed work: -

1. User will enter the word that he has to search in number of files.
2. Now we have three code to check the efficiency of the parallel searching: serial, simple parallel, optimised parallel.

    a. Serial Search

    It will search the word one by one in each file and count the word the we have to search.

    b. Simple Parallel Search

    In case of this, each thread is allotted to single file, and each thread count the word in their allotted file and sum up at the last.

    This will reduce the lots of time.

    c. Optimised Parallel Search

    In this case, we are removing the unwanted words like “is”, “the”,temporary file is created

4. IMPLEMENTATION:

Following files are the part of the project

1. PagerankSerial.cpp (Serial Code)
2. PagerankParallel.cpp (Parallel Code)
3. Optimised PagerankParallel.cpp (Parallel Code)
4. File1.txt (Input file)
5. File2.txt (Input file)
6. File3.txt (Input file)
7. File4.txt (Input file)

CODE:

1. PageRankSerial

```
#include<iostream>
#include<omp.h>
#include<fstream>
#include<chrono>
#include<ctime>
using namespace std;

int main()
{
    int sum = 0;
    fstream file;
    string s,word;
    double t1,t2;

    auto start = chrono::system_clock::now();
    for(int i=1;i<=4;i++)
    {
        s = "file"+to_string(i)+".txt";
        file.open(s.c_str());
        while(file >> word)
        {
            if(word == "hello")
```

```
                {
                    sum++;
                }
            }
            file.close();
        }
        auto end = chrono::system_clock::now();
        chrono::duration<double> elapsed_seconds = end-start;
        cout<<"elapsed time: " << elapsed_seconds.count()*1000<<endl;
        cout<<sum<<endl;
    }
```

2. PageRankParallel

```
//Simple Parallel Code
#include<iostream>
#include<omp.h>
#include<fstream>
#include<string>
#include<chrono>
#include<ctime>

using namespace std;


int main()
{
    double t1=0;
    int z=1;
    while(1)
    {
        string option;
        cout<<"Do u wnat to search (Y/N)";
        cin>>option;
        if(option == "N" )
        {
            cout<<"Bye-Bye"<<endl;
            cout<<"Project has been made by: "<<endl;
            cout<<"Saurabh Ambar 17BCI0058"<<endl;
            cout<<"Kumar Abhishek 17BCI0145"<<endl;
            cout<<"Vishal Saini 17BCI0188"<<endl;
            exit(0);
        }
        else if(option == "Y")
        {
            int totalsum=0;
            string searchword;
            cout<<"Enter the word u want to search: ";
            cin>>searchword;
    //
            long arr[]={0,0,0,0};
            auto start = std::chrono::system_clock::now();
            #pragma omp parallel
            {
                int tid;
                int numt;
                int sum = 0;
                #pragma omp single
                {
                    numt = omp_get_num_threads();
                }
                tid = omp_get_thread_num();
                fstream file;
                string s = "file"+to_string(tid+1)+".txt";
                string word;
                file.open(s.c_str());

                while(file >> word)
                {
                    if("hello" == word)
                    {
                        sum++;
                    }
                }
                #pragma omp critical
                {
                    arr[tid] = sum;
                    totalsum+=sum;
                }
            }
            //t2 = omp_get_wtime();
            auto end = std::chrono::system_clock::now();
            cout<<"Number of times word found: "<<totalsum<<endl;
            chrono::duration<double> elapsed_seconds = end-start;
            cout<<"elapsed time: " << elapsed_seconds.count()*1000<<endl;
            t1+=elapsed_seconds.count()*1000;
            //t1 = 0;
            cout<<"Words according to file number: "<<endl;
            for(int i=0;i<4;i++)
            {
                cout<<"file"<<(i+1)<<" : "<<arr[i]<<endl;
            }
            cout<<"Tota time taken after "<<z++<<" searches: "<<t1<<endl;
```

```
        }
        else
        {
            cout<<"Invalid Input. Try Again!!!"<<endl;
        }
    }
}
```

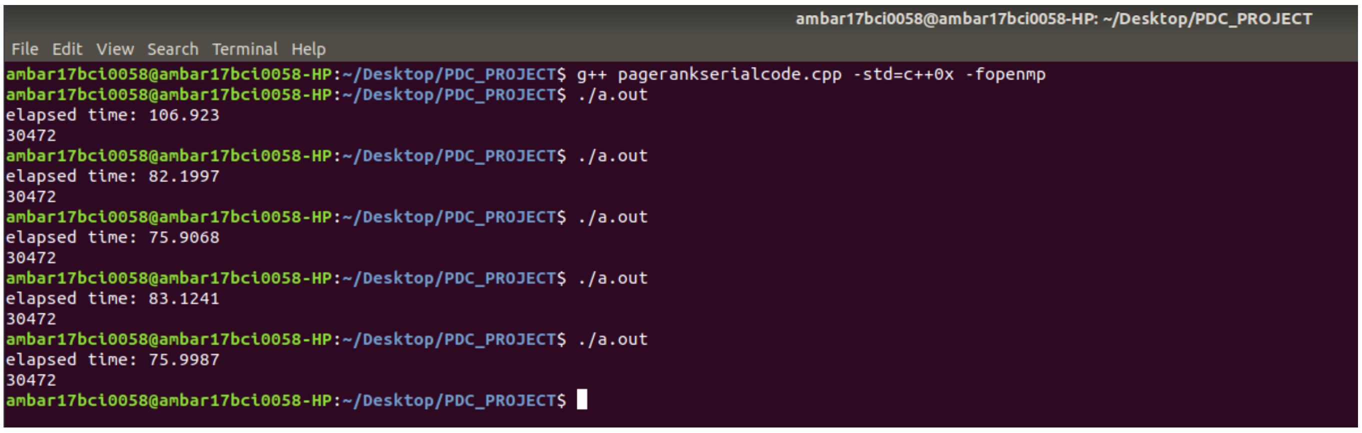
3. Optimised PageRankParallel

```
//parallel code for word search
#include<iostream>
#include<omp.h>
#include<vector>
#include<fstream>
#include<chrono>
#include<ctime>

using namespace std;
int linearSearch(vector<string> A,string tos);
int main()
{
    vector<string>  vect = {"The","knowns","known","also","ourselves", "hers",
"between", "yourself", "but", "again", "there", "about", "once", "during", "out",
"very", "having", "with", "they", "own", "an", "be", "some", "for", "do", "its", "yours",
"such", "into", "of", "most", "itself", "other", "off", "is", "s", "am", "or", "who", "as",
"from", "him", "each", "the", "themselves", "until", "below", "are", "we", "these",
"your", "his", "through", "don", "nor", "me", "were", "her", "more", "himself", "this",
"down", "should", "our", "their", "while", "above", "both", "up", "to", "ours", "had",
"she", "all", "no", "when", "at", "any", "before", "them", "same", "and", "been",
"have", "in", "will", "on", "does", "yourselves", "then", "that", "because", "what",
"over", "why", "so", "can", "did", "not", "now", "under", "he", "you", "herself", "has",
"just", "where", "too", "only", "myself", "which", "those", "i", "after", "few", "whom",
"t", "being", "if", "theirs", "my", "against", "a", "by", "doing", "it", "how", "further",
"was", "here", "than"};
    double t1,t2;
    int count=0;
    auto start = std::chrono::system_clock::now();
    #pragma omp parallel
    {
        int tid,numt;
        #pragma omp single
            numt = omp_get_num_threads();
            tid = omp_get_thread_num();
            fstream fileread,filewrite;
            string s,w;
            s = "file"+to_string(tid+1)+".txt";
            w = "tempfile"+to_string(tid+1)+".txt";
            fileread.open(s.c_str());
            filewrite.open(w,ios::out);
            string word;
           while(fileread >> word)
            {
                int count = 0;
                int index = linearSearch(vect,word);
                if(index == 0)
                {
                    filewrite << word+" ";
                    count++;
                }
            }
        }
        auto end = std::chrono::system_clock::now();
        chrono::duration<double> elapsed_seconds = end-start;
        t1 = 0;
        while(1)
        {
            int totalsum = 0;
            string choice;
            cout<<"Do u want to search word or not (Y/N)";
            cin>>choice;
            if(choice == "N")
            {
             system("cls");
                cout<<"Bye-Bye"<<endl;
                cout<<"Project has been made by: "<<endl;
                cout<<"Saurabh Ambar 17BCI0058"<<endl;
                cout<<"Kumar Abhishek 17BCI0145"<<endl;
                cout<<"Vishal Saini 17BCI0188"<<endl;
                cout<<"\nTotal time taken: "<<t1<<endl;
                exit(0);
            }
            else if(choice == "Y")
            {
                string searchword;
                cout<<"Enter the search word: ";
                cin>>searchword;
                double arr[] = {0,0,0,0};
                auto start1 = std::chrono::system_clock::now();
                #pragma omp parallel
                {
                    int tid;
                int numt;
                int sum = 0;
                #pragma omp single
                {
                    numt = omp_get_num_threads();
                }
                tid = omp_get_thread_num();
                fstream file;
                string s = "tempfile"+to_string(tid+1)+".txt";
                string word;
                file.open(s.c_str());

                while(file >> word)
                {
                    if(searchword == word)
                    {
                        sum++;
                    }
                }
                #pragma omp critical
                {
                    totalsum+=sum;
                    arr[tid] = sum;
                }
            }
            auto end1 = std::chrono::system_clock::now();
            //t2 = omp_get_wtime();
            for(int i=0;i<4;i++)
            {
                cout<<"file"<<i+1<<" : "<<arr[i]<<endl;
            }
            cout<<totalsum<<endl;
            elapsed_seconds = end1-start1;
            cout<<"elapsed time: " << elapsed_seconds.count()*1000<<endl;
            t1+=elapsed_seconds.count()*1000;
            cout<<"Total time after "<<count++<<" : "<<t1<<endl;

        }
        else
        {
            cout<<"Invalid Input. Try Again!!!"<<endl;
        }
    }
}

int linearSearch(vector<string> A,string tos){

     int foundat = -1;
```

```
         //Simple OpenMP for loop in parallel
    bool flag = false;
         #pragma omp parallel for
         for(int iter =0; iter< A.size(); iter++)
    {
        if(!flag)
        {
            if(A[iter] == tos)
            {
                foundat = iter+1;
                flag = true;
            }
        }


         }
         if(foundat == -1)
        return 0;
    return 1;
}
```

5. Result and Discussion

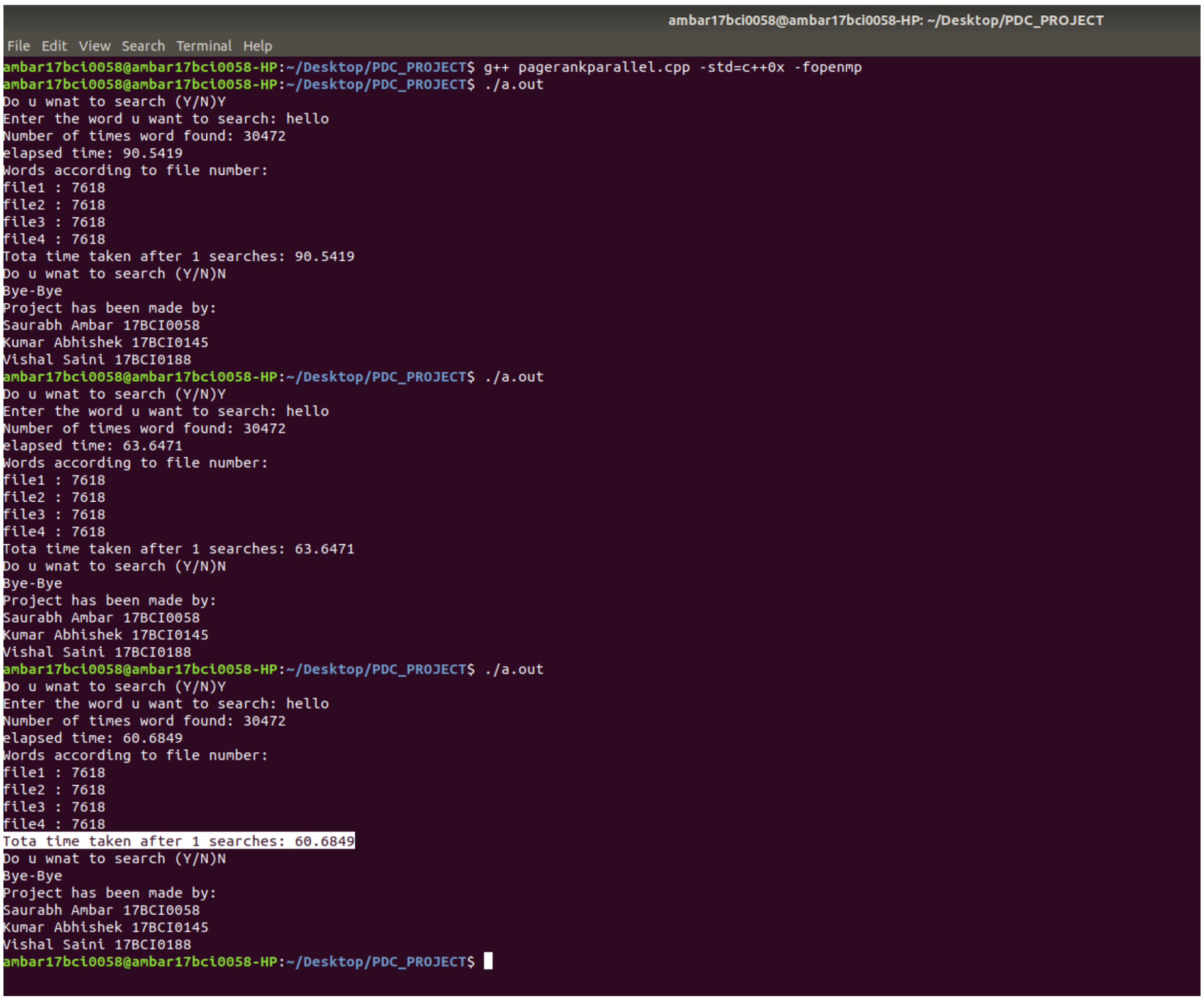
1. Serial Code



```
ambar17bci0058@ambar17bci0058-HP: ~/Desktop/PDC_PROJECT
File  Edit  View  Search  Terminal  Help
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ g++ pagerankserialcode.cpp -std=c++0x -fopenmp
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
elapsed time: 106.923
30472
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
elapsed time: 82.1997
30472
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
elapsed time: 75.9068
30472
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
elapsed time: 83.1241
30472
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
elapsed time: 75.9987
30472
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ 
```

Average Time: 84.2

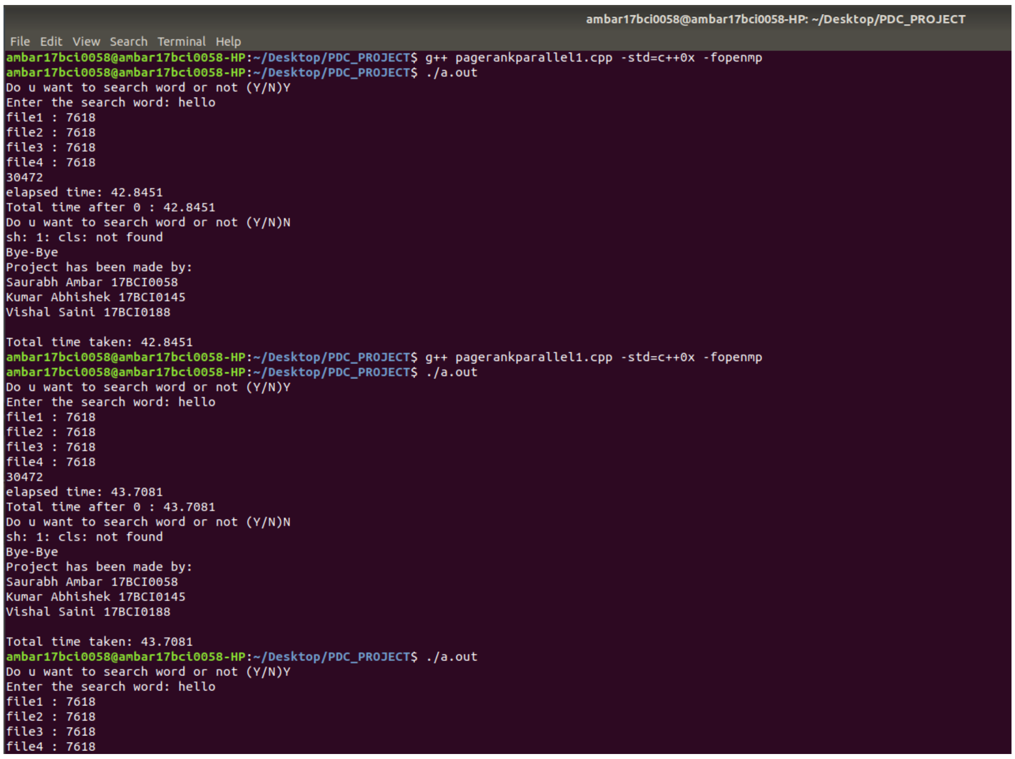
2. Parallel Code:



```
ambar17bci0058@ambar17bci0058-HP: ~/Desktop/PDC_PROJECT
File  Edit  View  Search  Terminal  Help
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ g++ pagerankparallel.cpp -std=c++0x -fopenmp
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
Do u wnat to search (Y/N)Y
Enter the word u want to search: hello
Number of times word found: 30472
elapsed time: 90.5419
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Tota time taken after 1 searches: 90.5419
Do u wnat to search (Y/N)N
Bye-Bye
Project has been made by:
Saurabh Ambar 17BCI0058
Kumar Abhishek 17BCI0145
Vishal Saini 17BCI0188
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
Do u wnat to search (Y/N)Y
Enter the word u want to search: hello
Number of times word found: 30472
elapsed time: 63.6471
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Tota time taken after 1 searches: 63.6471
Do u wnat to search (Y/N)N
Bye-Bye
Project has been made by:
Saurabh Ambar 17BCI0058
Kumar Abhishek 17BCI0145
Vishal Saini 17BCI0188
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
Do u wnat to search (Y/N)Y
Enter the word u want to search: hello
Number of times word found: 30472
elapsed time: 60.6849
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Tota time taken after 1 searches: 60.6849
Do u wnat to search (Y/N)N
Bye-Bye
Project has been made by:
Saurabh Ambar 17BCI0058
Kumar Abhishek 17BCI0145
Vishal Saini 17BCI0188
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ 
```

Average Time: 60

3. Optimised Parallel Algorithm:

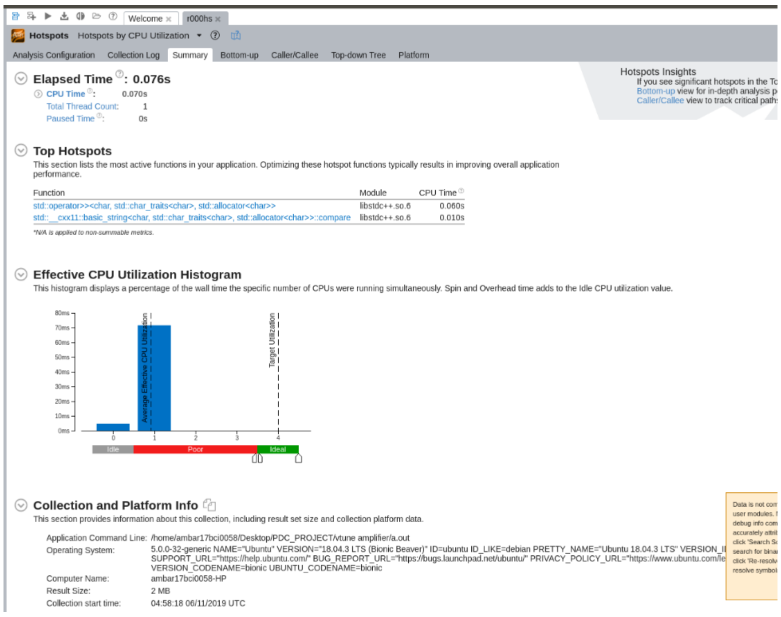


```
ambar17bci0058@ambar17bci0058-HP: ~/Desktop/PDC_PROJECT
File  Edit  View  Search  Terminal  Help
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ g++ pagerankparallel1.cpp -std=c++0x -fopenmp
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
Do u want to search word or not (Y/N)Y
Enter the search word: hello
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 42.8451
Total time after 0 : 42.8451
Do u want to search word or not (Y/N)N
sh: 1: cls: not found
Bye-Bye
Project has been made by:
Saurabh Ambar 17BCI0058
Kumar Abhishek 17BCI0145
Vishal Saini 17BCI0188

Total time taken: 42.8451
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ g++ pagerankparallel1.cpp -std=c++0x -fopenmp
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
Do u want to search word or not (Y/N)Y
Enter the search word: hello
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 43.7081
Total time after 0 : 43.7081
Do u want to search word or not (Y/N)N
sh: 1: cls: not found
Bye-Bye
Project has been made by:
Saurabh Ambar 17BCI0058
Kumar Abhishek 17BCI0145
Vishal Saini 17BCI0188

Total time taken: 43.7081
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT$ ./a.out
Do u want to search word or not (Y/N)Y
Enter the search word: hello
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
```

Average Time: 45

6. OBSERVATION

1. Serial Search



Elapsed Time: 0.076s

CPU Time: 0.070s
Total Thread Count: 1
Paused Time: 0s

Hotspots Insights
If you see significant hotspots in the To
Bottom-up view for in-depth analysis p
Caller/Callee view to track critical path

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
std::operator>><char, std::char_traits<char>, std::allocator<char>>	libstdc++.so.6	0.060s
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::compare	libstdc++.so.6	0.010s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

Collection and Platform Info

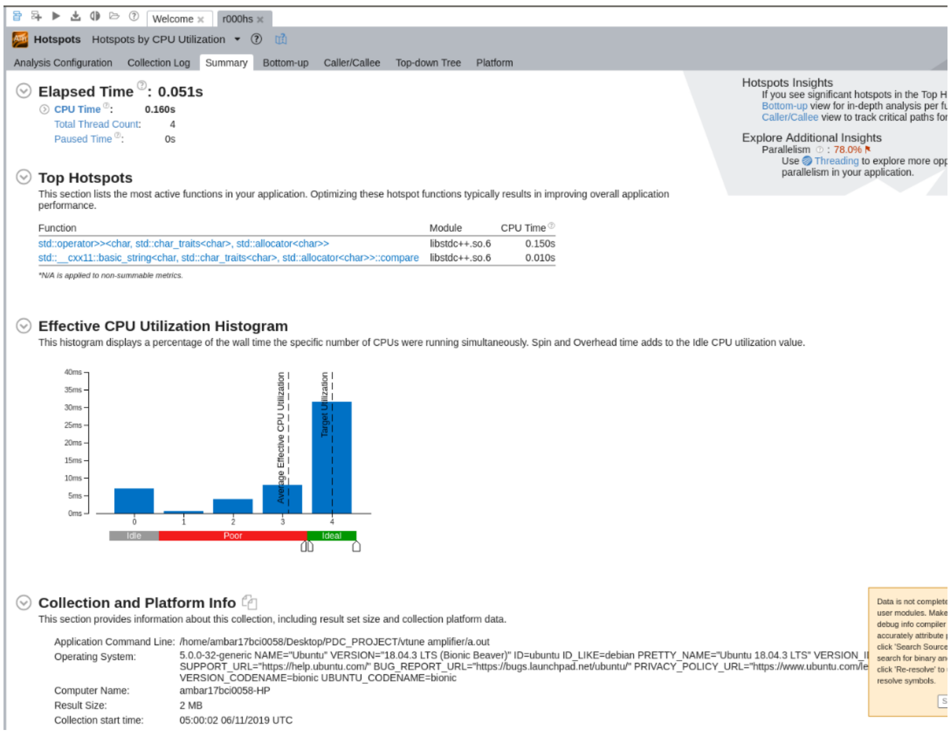
This section provides information about this collection, including result set size and collection platform data.

Application Command Line:	/home/ambar17bci0058/Desktop/PDC_PROJECT/vtune amplifier/a.out
Operating System:	5.0.0-32-generic NAME="Ubuntu" VERSION="18.04.3 LTS (Bionic Beaver)" ID=ubuntu ID_LIKE=debian PRETTY_NAME="Ubuntu 18.04.3 LTS" VERSION_I SUPPORT_URL="https://help.ubuntu.com/" BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/" PRIVACY_POLICY_URL="https://www.ubuntu.com/le VERSION_CODENAME=bionic UBUNTU_CODENAME=bionic
Computer Name:	ambar17bci0058-HP
Result Size:	2 MB
Collection start time:	04:58:18 06/11/2019 UTC

Data is not com
user modules.
debug info com
accurately attri
click 'Search S
search for bina
click 'Re-resolv
resolve symbol

➢ We can observe the CPU Utilization is poor in case of Serial Search.

2. Parallel Search



Elapsed Time: 0.051s

CPU Time:	0.160s
Total Thread Count:	4
Paused Time:	0s

Hotspots Insights
If you see significant hotspots in the Top H
Bottom-up view for in-depth analysis per fu
Caller/Callee view to track critical paths for

Explore Additional Insights
Parallelism : 78.0%
Use Threading to explore more opp
parallelism in your application.

Top Hotspots
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
std::operator>><char, std::char_traits<char>, std::allocator<char>>	libstdc++.so.6	0.150s
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::compare	libstdc++.so.6	0.010s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

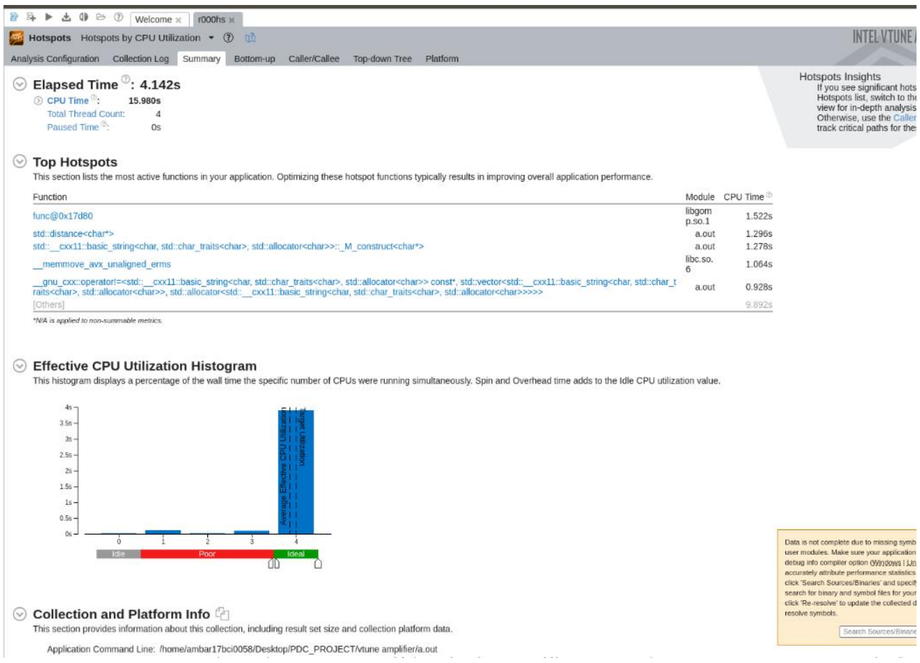
Collection and Platform Info
This section provides information about this collection, including result set size and collection platform data.

Application Command Line:	/home/ambar17bci0058/Desktop/PDC_PROJECT/vtune amplifier/a.out
Operating System:	5.0.0-32-generic NAME="Ubuntu" VERSION="18.04.3 LTS (Bionic Beaver)" ID=ubuntu ID_LIKE=debian PRETTY_NAME="Ubuntu 18.04.3 LTS" VERSION_II SUPPORT_URL="https://help.ubuntu.com/" BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/" PRIVACY_POLICY_URL="https://www.ubuntu.com/le VERSION_CODENAME=bionic UBUNTU_CODENAME=bionic
Computer Name:	ambar17bci0058-HP
Result Size:	2 MB
Collection start time:	05:00:02 06/11/2019 UTC

Data is not complete
user modules. Make
debug info compiler
accurately attribute p
click 'Search Source
search for binary an
click 'Re-resolve' to
resolve symbols.

- ➢ We can observe the CPU Utilization is better than Serial.

3. Optimised Parallel



Elapsed Time: 4.142s

CPU Time:	15.980s
Total Thread Count:	4
Paused Time:	0s

Hotspots Insights
If you see significant hots
Hotspots list, switch to the
view for in-depth analysis
Otherwise, use the Caller
track critical paths for the

Top Hotspots
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
func@0x17d80	libgomp.so.1	1.522s
std::distance<char*>	a.out	1.296s
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::_M_construct<char*>	a.out	1.278s
__memmove_avx_unaligned_erms	libc.so.6	1.064s
__gnu_cxx::operator!=<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>> const*, 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>>>>>	a.out	0.928s
[Others]		9.892s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

Data is not complete due to missing symb
user modules. Make sure your application
debug info compiler option (Windows | Lin
accurately attribute performance statistics
click 'Search Sources/Binaries' and specif
search for binary and symbol files for your
click 'Re-resolve' to update the collected d
resolve symbols.

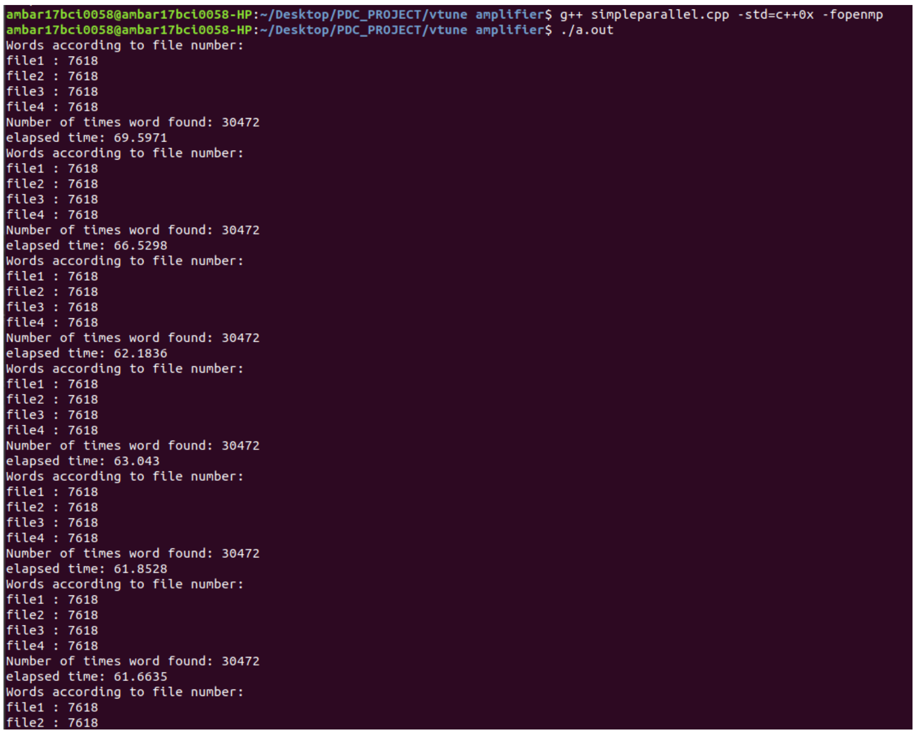
Collection and Platform Info
This section provides information about this collection, including result set size and collection platform data.

Application Command Line: /home/ambar17bci0058/Desktop/PDC_PROJECT/vtune amplifier/a.out

- ➢ We can observe the maximum CPU Utilization in this case compare to other cases.

After that we increase the file size 100 times the actual text to observe the change in efficiency:

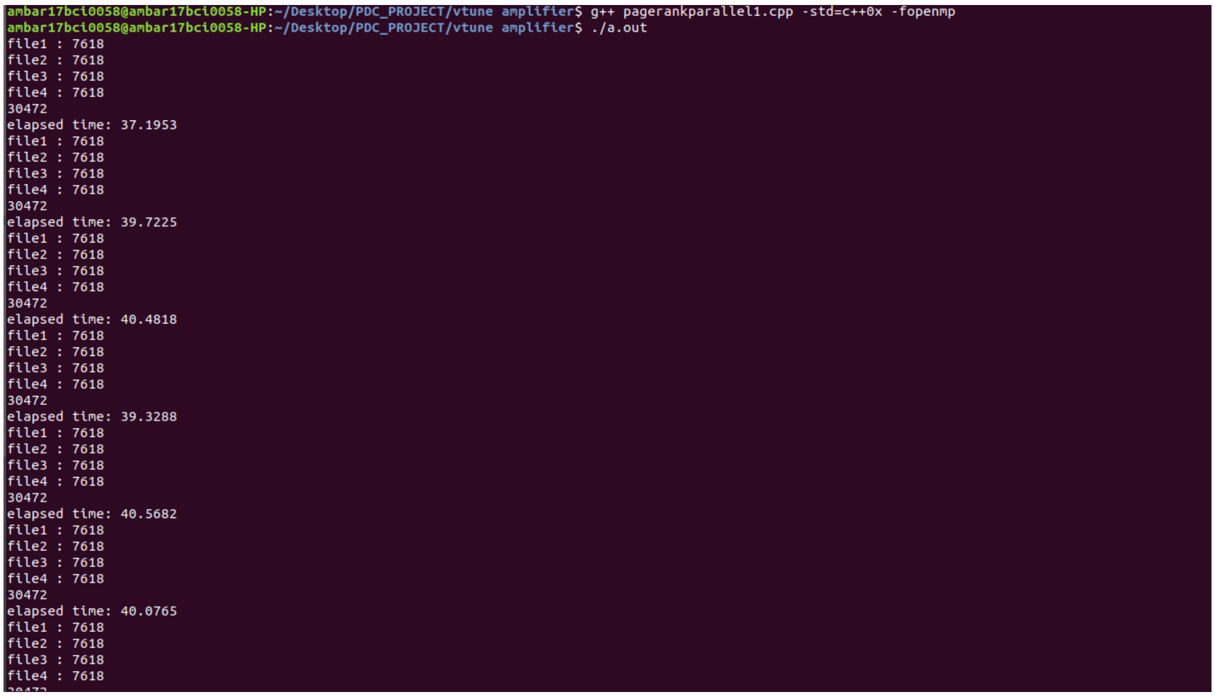
1. Parallel Search



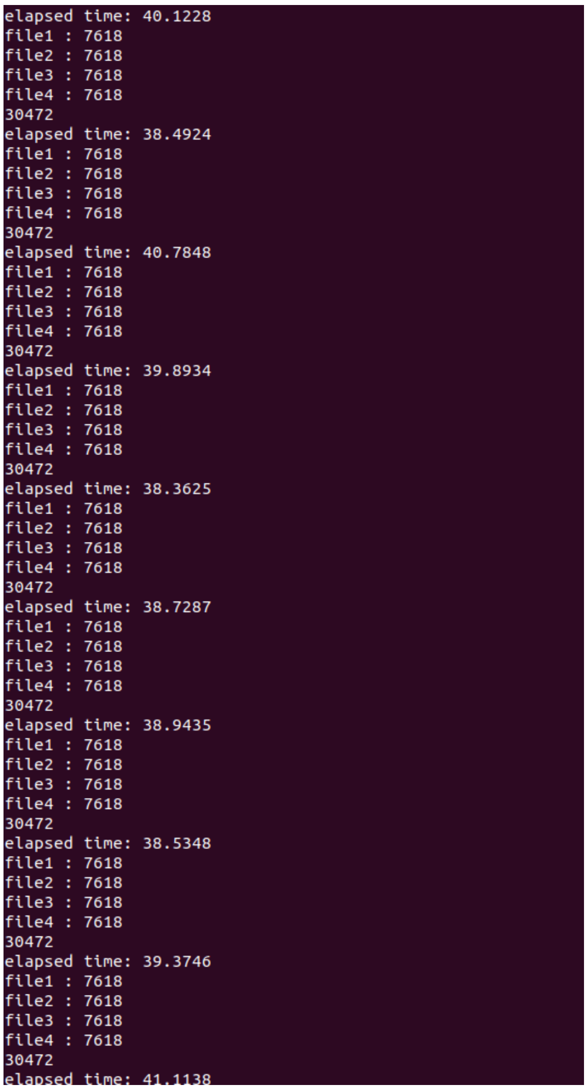
```
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT/vtune amplifier$ g++ simpleparallel.cpp -std=c++0x -fopenmp
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT/vtune amplifier$ ./a.out
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Number of times word found: 30472
elapsed time: 69.5971
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Number of times word found: 30472
elapsed time: 66.5298
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Number of times word found: 30472
elapsed time: 62.1836
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Number of times word found: 30472
elapsed time: 63.043
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Number of times word found: 30472
elapsed time: 61.8528
Words according to file number:
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
Number of times word found: 30472
elapsed time: 61.6635
Words according to file number:
file1 : 7618
file2 : 7618
```

Average Time for Parallel: 62

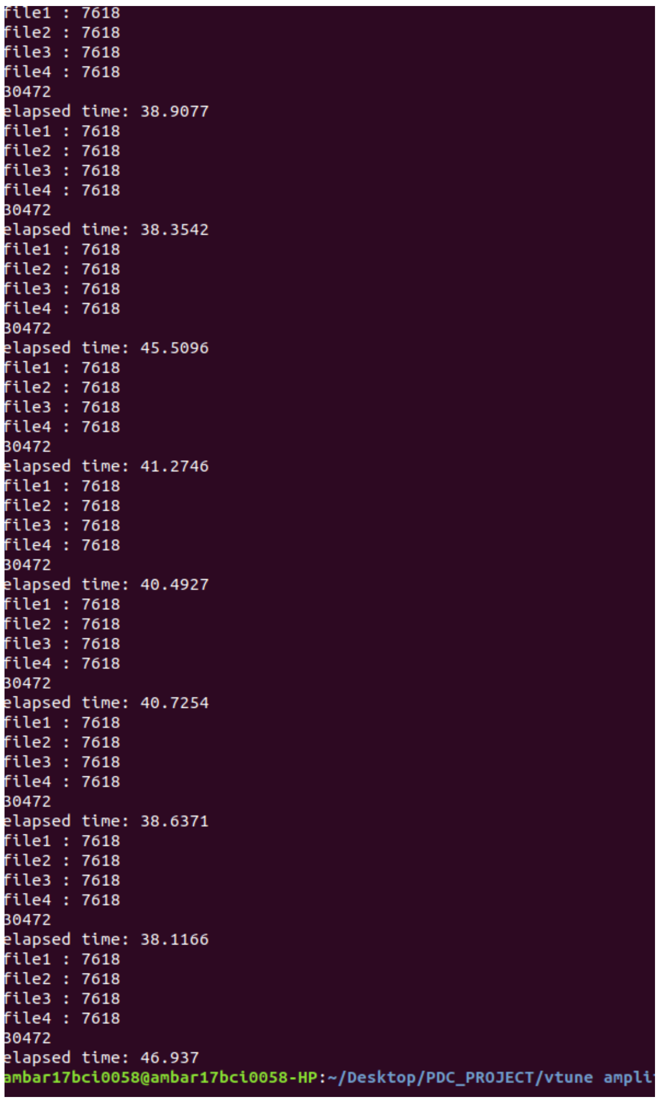
2. Optimised Parallel:



```
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT/vtune amplifier$ g++ pagerankparallel1.cpp -std=c++0x -fopenmp
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT/vtune amplifier$ ./a.out
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 37.1953
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 39.7225
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 40.4818
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 39.3288
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 40.5682
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 40.0765
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
```



```
elapsed time: 40.1228
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.4924
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 40.7848
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 39.8934
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.3625
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.7287
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.9435
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.5348
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 39.3746
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 41.1138
```

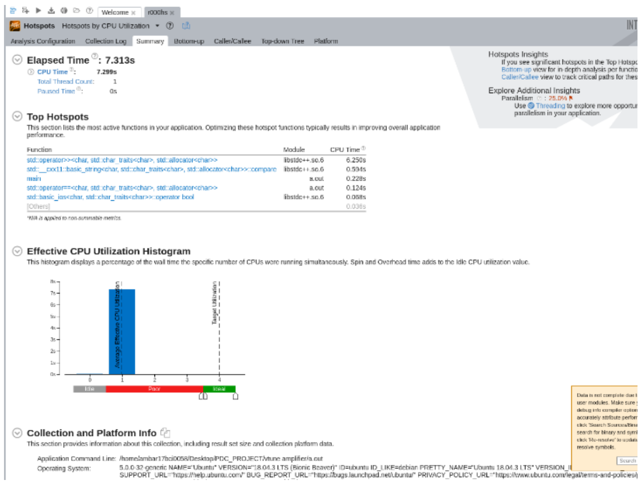


```
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.9077
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.3542
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 45.5096
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 41.2746
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 40.4927
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 40.7254
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.6371
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 38.1166
file1 : 7618
file2 : 7618
file3 : 7618
file4 : 7618
30472
elapsed time: 46.937
ambar17bci0058@ambar17bci0058-HP:~/Desktop/PDC_PROJECT/vtune ampli
```

Average Time for Serial: 40

OBSERVATION: After the increase of text file.

1. Serial Search



Hotspots Hotspots by CPU Utilization

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Elapsed Time: 7.313s

CPU Time:	7.299s
Total Thread Count:	1
Paused Time:	0s

Hotspots Insights
If you see significant hotspots in the Top Hotspo
Bottom-up view for in-depth analysis per functio
Caller/Callee view to track critical paths for thes

Explore Additional Insights
Parallelism: 25.0%
Use Threading to explore more opportu
parallelism in your application.

Top Hotspots
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
std::operator>><char, std::char_traits<char>, std::allocator<char>>	libstdc++.so.6	6.250s
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::compare	libstdc++.so.6	0.594s
main	a.out	0.228s
std::operator==<char, std::char_traits<char>, std::allocator<char>>	a.out	0.124s
std::basic_ios<char, std::char_traits<char>>::operator bool	libstdc++.so.6	0.068s
[Others]		0.036s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

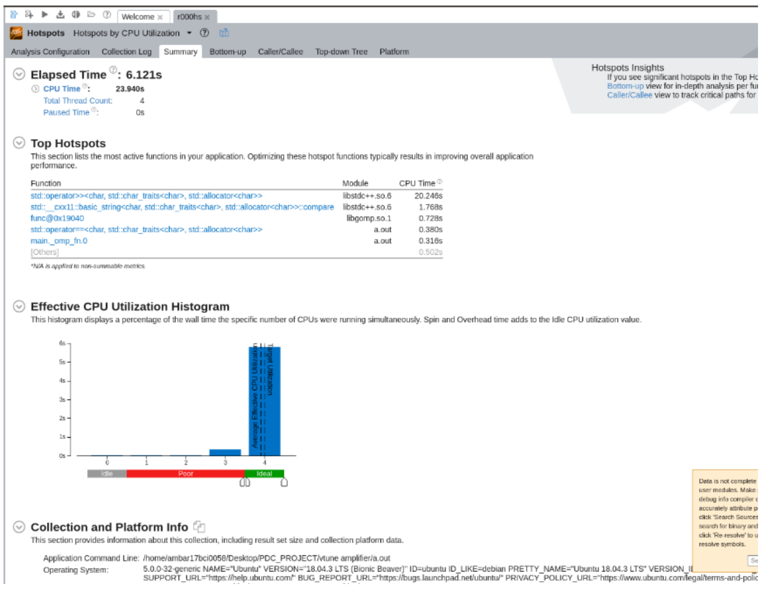
Data is not complete due t
user modules. Make sure
debug info compiler option
accurately attribute perfor
click 'Search Sources/Bina
search for binary and sym
click 'Re-resolve' to updat
resolve symbols.

Collection and Platform Info
This section provides information about this collection, including result set size and collection platform data.

Application Command Line:	/home/ambar17bci0058/Desktop/PDC_PROJECT/vtune amplifier/a.out
Operating System:	5.0.0-32-generic NAME="Ubuntu" VERSION="18.04.3 LTS (Bionic Beaver)" ID=ubuntu ID_LIKE=debian PRETTY_NAME="Ubuntu 18.04.3 LTS" VERSION_I SUPPORT_URL="https://help.ubuntu.com/" BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/" PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/

- ➢ When we increase the file size by 100 times, the serial search becomes very poor

2. Parallel Search



Hotspots Hotspots by CPU Utilization

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Elapsed Time: 6.121s

CPU Time:	23.940s
Total Thread Count:	4
Paused Time:	0s

Hotspots Insights
If you see significant hotspots in the Top Ho
Bottom-up view for in-depth analysis per fu
Caller/Callee view to track critical paths for

Top Hotspots
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
std::operator>><char, std::char_traits<char>, std::allocator<char>>	libstdc++.so.6	20.246s
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::compare	libstdc++.so.6	1.768s
func@0x19040	libgomp.so.1	0.728s
std::operator==<char, std::char_traits<char>, std::allocator<char>>	a.out	0.380s
main._omp_fn.0	a.out	0.316s
[Others]		0.502s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

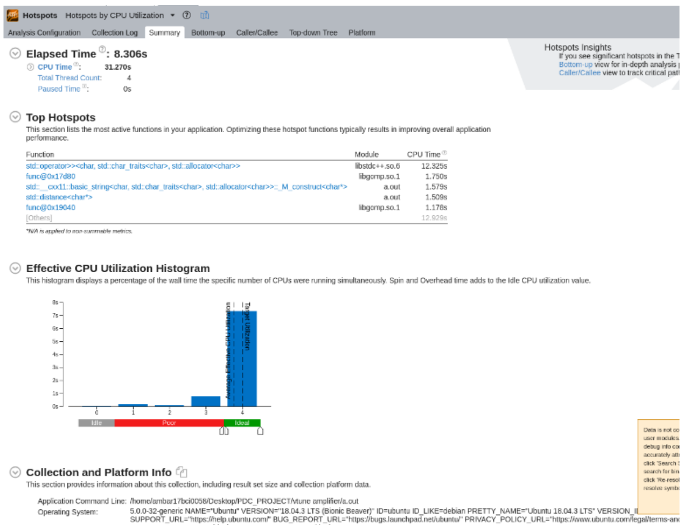
Data is not complete
user modules. Make
debug info compiler
accurately attribute p
click 'Search Sources
search for binary and
click 'Re-resolve' to u
resolve symbols.

Collection and Platform Info
This section provides information about this collection, including result set size and collection platform data.

Application Command Line:	/home/ambar17bci0058/Desktop/PDC_PROJECT/vtune amplifier/a.out
Operating System:	5.0.0-32-generic NAME="Ubuntu" VERSION="18.04.3 LTS (Bionic Beaver)" ID=ubuntu ID_LIKE=debian PRETTY_NAME="Ubuntu 18.04.3 LTS" VERSION_I SUPPORT_URL="https://help.ubuntu.com/" BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/" PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-polic

- ➢ When we increase the file size by 100 times, it show better CPU Utilization.

3. Optimised Parallel



Elapsed Time: 8.306s

CPU Time:	31.270s
Total Thread Count:	4
Paused Time:	0s

Hotspots Insights
If you see significant hotspots in the T
Bottom-up view for in-depth analysis
Caller/Callee view to track critical pat

Top Hotspots
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
std::operator>><char, std::char_traits<char>, std::allocator<char>>	libstdc++.so.6	12.325s
func@0x17d80	libgomp.so.1	1.750s
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::_M_construct<char*>	a.out	1.579s
std::distance<char*>	a.out	1.509s
func@0x19040	libgomp.so.1	1.178s
[Others]		12.929s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

Data is not co
user modules
debug info co
accurately att
click 'Search
search for bin
click 'Re-resol
resolve symb

Collection and Platform Info
This section provides information about this collection, including result set size and collection platform data.

Application Command Line:	/home/ambar17bci0058/Desktop/PDC_PROJECT/vtune amplifier/a.out
Operating System:	5.0.0-32-generic NAME="Ubuntu" VERSION="18.04.3 LTS (Bionic Beaver)" ID=ubuntu ID_LIKE=debian PRETTY_NAME="Ubuntu 18.04.3 LTS" VERSION_I SUPPORT_URL="https://help.ubuntu.com/" BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/" PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-an

- It has the better CPU Utilization than the others.

7. Conclusion: -

The Code was executed successfully and applying parallelism reduces running time significantly.

8. Application: -

Word search is used everywhere from local page search (Ctrl + F) to searching words on document viewer like "reader" in windows. In fact a whole branch called Information Retrieval was developed for this. This project was actually inspired by Information Retrieval. It has a lot of application in real word.

9. References:-

[1] Parallel Quick Search Algorithm for the Exact String Matching Problem Using OpenMP

https://pdfs.semanticscholar.org/cdb5/c9fb44e06e2dce2ec62c6b38a978e1f53dc4.pdf

[2] Parallel Computing using OpenMP

https://www.ijcsmc.com/docs/papers/February2017/V6I2201713.pdf

[3] Scalable parallel word search in multicore/multiprocessor systems

https://www.researchgate.net/publication/220358574_Scalable_parallel_word_search_in_multicorem ultiprocessor_systems

[4] Parallel Quicksort Algorithm using OpenMP

https://www.researchgate.net/publication/304676896_Parallel_Quicksort_Algorithm_using_OpenMP

[5] Punam V. Maitri ; Aruna Verma on "Secure file storage in cloud computing using hybrid cryptography algorithm"

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8101449