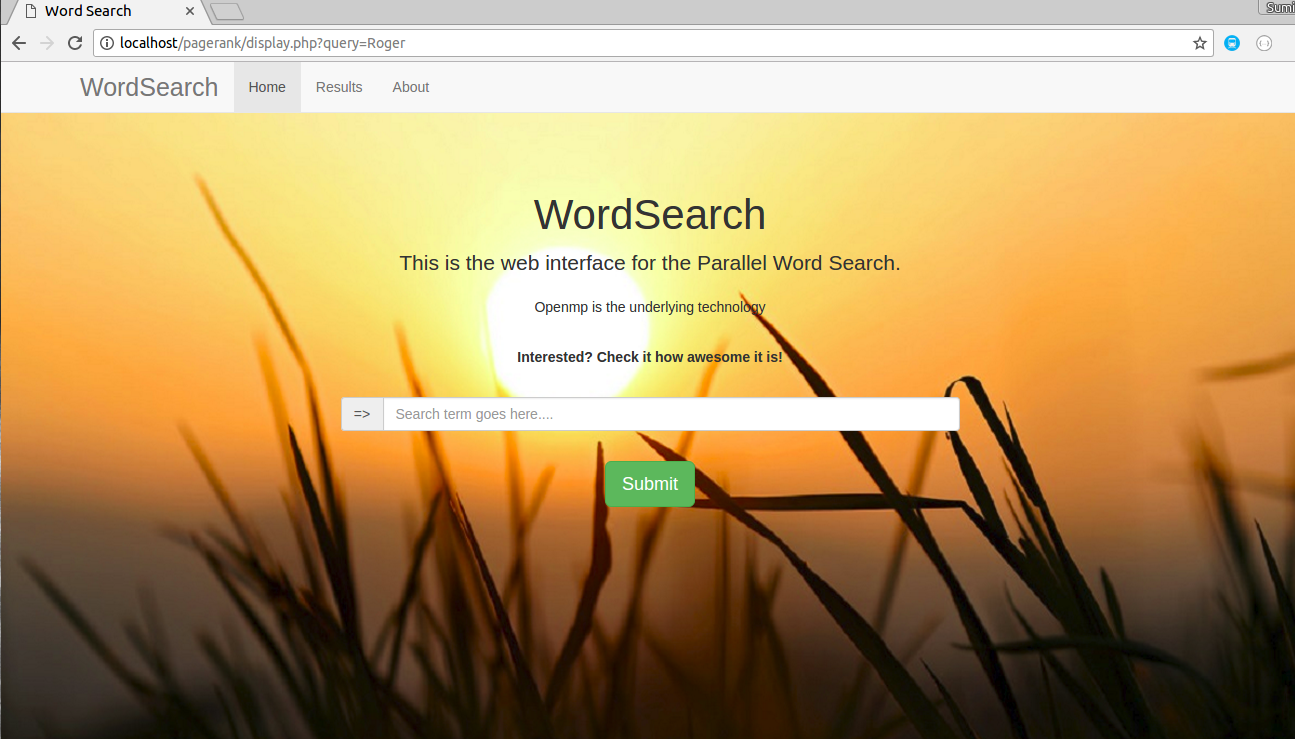
Mini Project (Parallel Processing)

Word Search

(USing openmp and PHP)

Avesh Shaikh (B12053) | Puja Tapadiya (B12060) | Kirtikumar Waykos (B12066)

BE –C (VII Sem.)



# **Abstract** :

As the name suggests “word search” is about searching words in documents parallely using **openmp**. This is not just a simple word search but it also ranks the documents based on the relevance of the documents with respect to the searched word.The Results are the displayed on a web page which uses **PHP**(Server side Scripting language) to render the result.

## **Programming Languages and Web Technologies Used**:

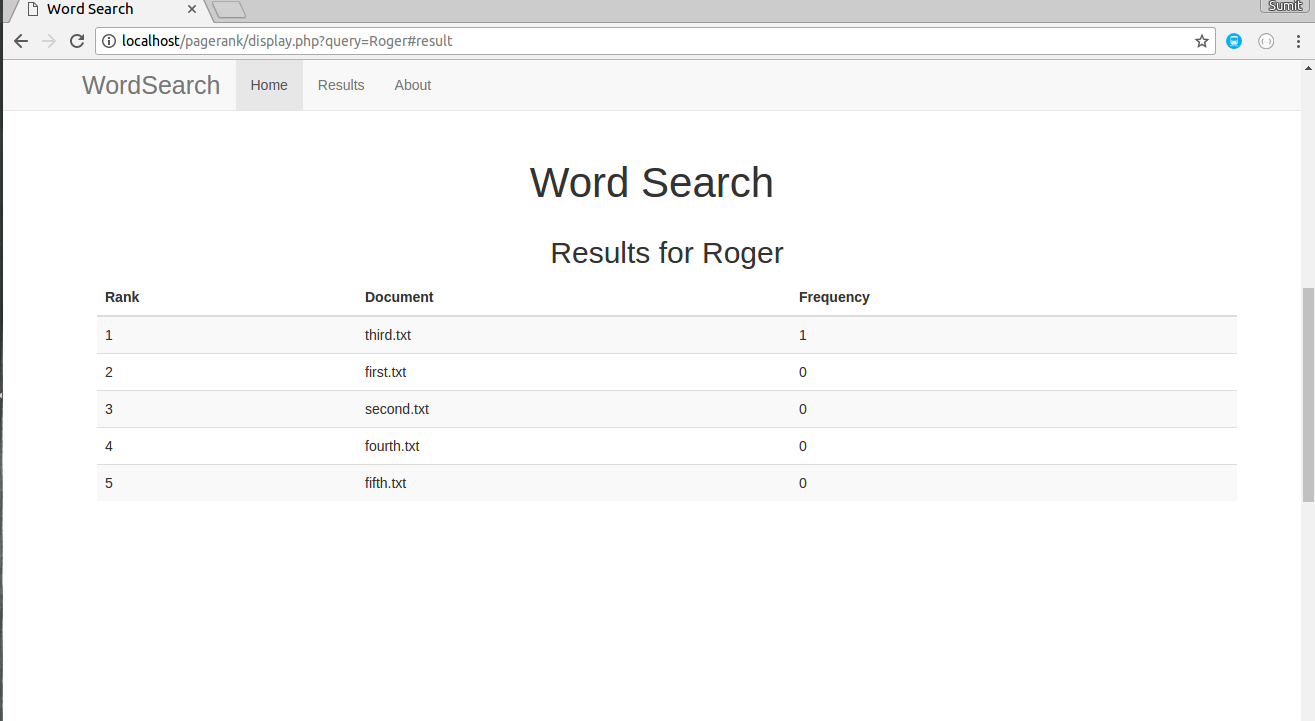
C++ (using Standard Template Library) , Openmp (for parallelization of the code), PHP , HTML ,CSS , Javascript ( Jquery) , Bootstrap , Mysql(Database).

## **Implementation**:

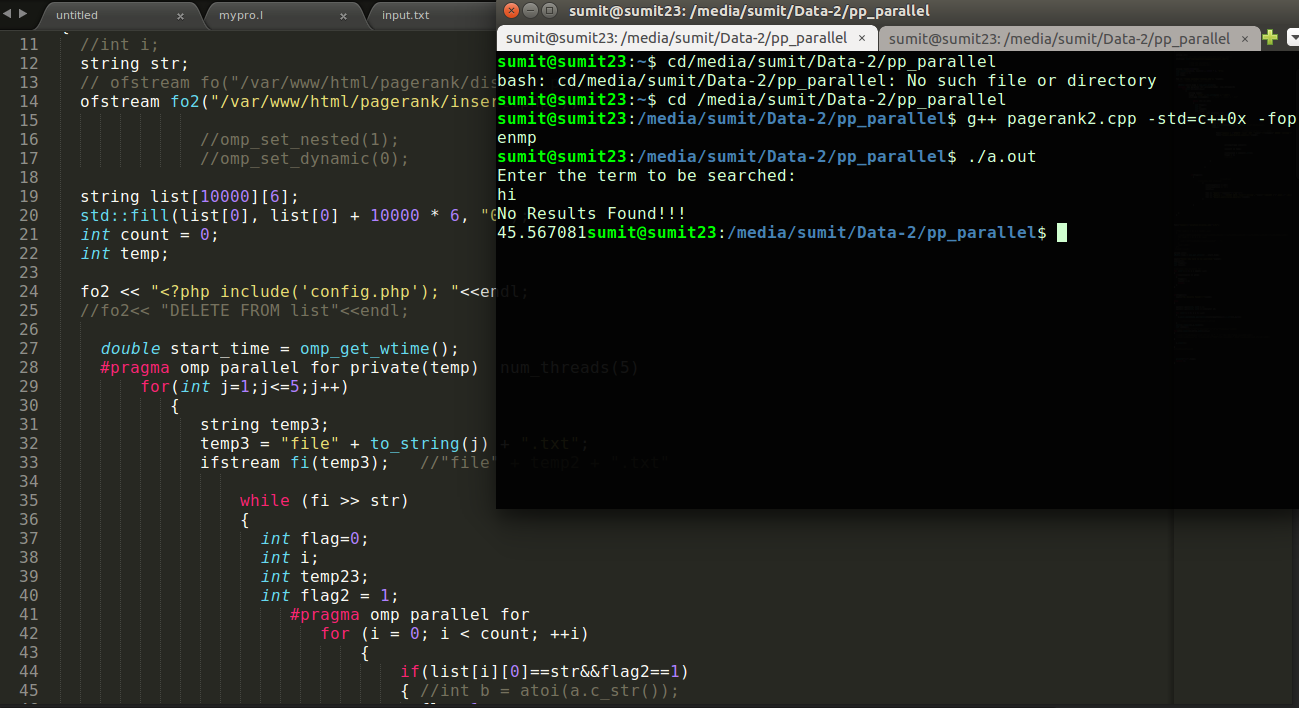
The basic code is written in C++ which uses openmp to parallelize the code.

1. The documents are scanned word by word and a dictionary is maintained so that words can be searched. Since documents can have several thousands of terms scanning each term can take a lot of time and hence there is a lot of scope for parallelization.
2. Along side scanning the terms using file pointer this dictionary is also sent to a php file **( Insert.php**) so that result can be displayed on the web page latter(**Display.php**).
3. This Page (Insert.php) is connected to mysql databse using LAMP (**Linux – Apache** – **Mysql-PHP**) server to insert the dictionary entries in to the tables.
4. Using the database entries whenever terms are searched it can be found on the web page and documents are ranked on the basis of relevence.
5. Maintaining the dictionary is the task that takes most of the time hence most of the parallelization as can be seen from the code is done here.

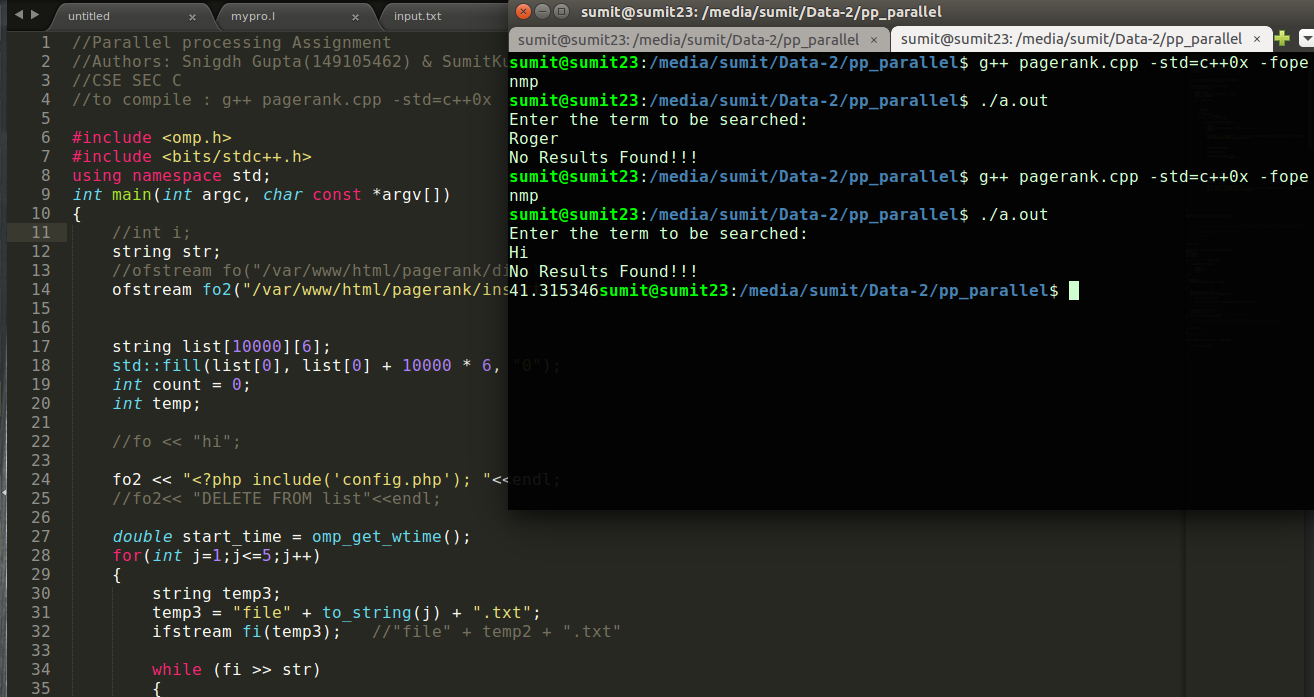
## **Observations**:



This is the web interface that displays the search result.It can be seen that it also ranks the documents based on the frequency of terms in the document.



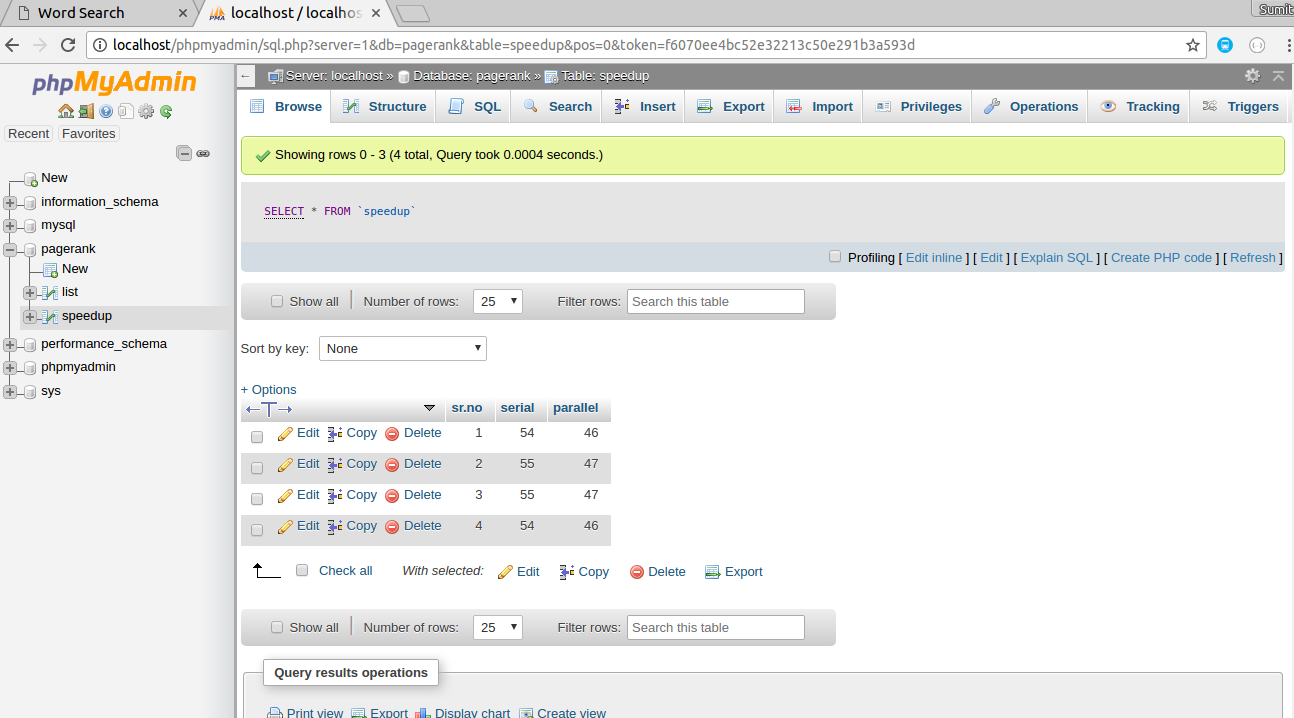
It can be seen when the serial code was executed it took 45.56 sec



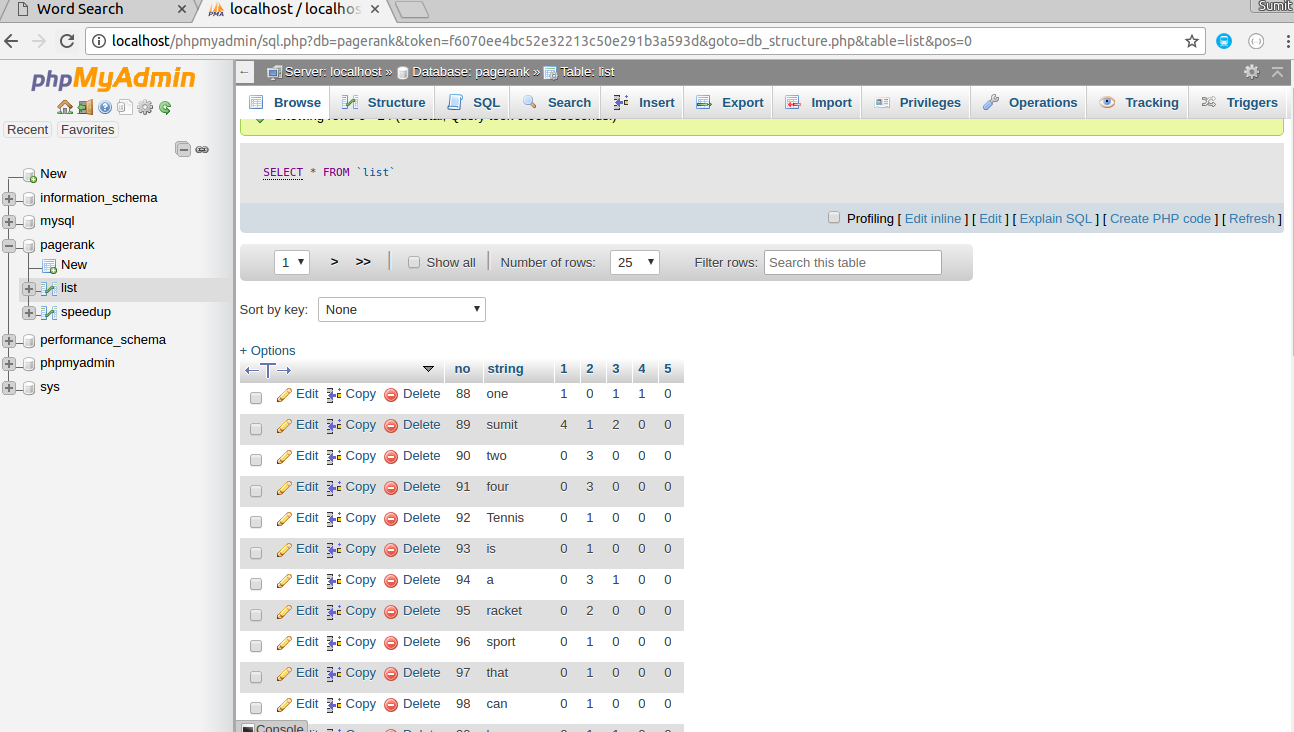
It took 41.31 secs when executed parallely.A significant improvement from serial code.

The above code was executed serially as well as parallely on multiple cores.

The result is shown in the graph in subsequent diagram .Following table shows the serial Vs Parallel time of execution.



Following diagram shows the dictionary which is made as result of running the code in mysql database.



## **Graph**:

The above Chart shows the speed up achieved on running the program serially and then parallel with variable number of threads .(**Note**:The above figures are the avg values calculated after running various times.)

## **Speed Up**:

As can be seen from the above diagram the speed up achieved was

(54.32-47.7) / 54.32 = 0.1218

**Speed up : 12.18%**

The Speed up achieved could have been more but there were several portions that had to be serialized to prevent race condition. Also since “while” loop cannot be serialized and while loop was used in the code.

## **Conclusion**:

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

## **Application**:

Word search is used everywhere from local page search ( Cntrl + F) to searching words on document viewer like “reader” in windows. Infact 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.

**Note**:

Following files are the part of the project

1. Pagerank.cpp (Parallel Code)
2. Pagerank2.cpp ( Serial Code)
3. Insert.php
4. Config.php
5. Display.php
6. File1.txt (Input file)
7. File2.txt (Input file)
8. File3.txt (Input file)
9. File4.txt (Input file)
10. File5.txt(Input file)