MCAP Project

Title: **Implementing and Analyzing the PageRank Algorithm Using Parallel Processing**

Guide : Dr . Prabha C N

Team Details				
Sl.No	**Roll No.**	**Div**	**SRN**	**Name**
1	560	E	01FE21BEC310	Rakshith S
2	150	A	01FE22BEC410	Samit P
3	152	A	01FE22BEC412	Sarvesh K
4	153	A	01FE22BEC413	Nitin S

Contents:

Problem Statement :

The goal of this project is to implement and optimize the PageRank algorithm using parallel processing techniques. The OpenMP for parallel execution in the program aims to significantly reduce computation time.

Objectives: For PageRank Algorithm Program

1. Implement PageRank:
•Develop a program to compute the PageRank of websites based on their link structure using the damping factor.
2. Data Handling:
•Read and parse website and link data from a structured input file.
3. Matrix Construction:
•Construct and transpose the stochastic matrix to represent link probabilities.
4. Parallel Computation:
•Utilize OpenMP for parallel processing to speed up PageRank calculations.
5. Convergence Criteria:
•Iterate until the PageRank values converge to a specified precision.
6.Performance Analysis:
•Measure and analyze the performance and speedup achieved through parallel computation.
7. Ranking and Sorting:
•Sort and display the top 10 websites based on their PageRank scores.
8. Memory Management:
•Ensure efficient memory usage and proper cleanup in the program.

Literature survey

Sl.No	Title	Methodology/ Algorithm	Description
1	An Improved PageRank Algorithm Based on Time Feedback and Topic Similarity	improved PageRank algorithm.	The paper proposes an improved PageRank algorithm that addresses limitations of the traditional algorithm by incorporating topic similarity and time feedback between web pages. Experimental results show that the enhanced algorithm improves search result precision and reduces theme drift, providing more relevant and up-to-date information for users.
2	An improved PageRank algorithm based on web content.	Valid links are identified by matching keywords in <anchor> and <title> tags, contributing to PageRank value.	The PDF explores challenges in delivering relevant search results on the web and proposes enhancements to the PageRank algorithm through web content analysis. By assessing the relationship between webpages based on content, the improved algorithm seeks to boost user search precision. It addresses shortcomings in traditional PageRank and Weighted PageRank algorithms, aiming to provide more accurate and targeted search results. The algorithm's focus on web content relevance aims to overcome issues like topic drift and user interest neglect. Through these modifications, the algorithm aims to offer users more precise and relevant search results in specific fields.

Sl.No	Title	Methodology/ Algorithm	Description
3	Research and Improvement of PageRank Sort Algorithm based on Retrieval Results	Different methods PageRank techniques	The improved algorithm combines the idea of HITS and assigns PageRank values based on the relevance of page concepts and user retrieving concepts. The document also explores the selection of concepts, design of the improved algorithm, and possible applications. Overall, it provides insights into the evolution and enhancement of retrieval result sorting algorithms for improved user satisfaction.
4	The PageRank Citation Ranking: Bringing Order to the Web	Enhanced version PageRank algorithm using OpenMP	A method for objectively ranking web pages based on importance. It addresses challenges faced by search engines on the diverse and vast World Wide Web, utilizing link structure to provide a global ranking of web pages. PageRank considers not just the content but also the connections between pages, offering a unique approach to information retrieval. By comparing PageRank to a random web surfer model, the document highlights the efficiency of computing PageRank for numerous pages and its application in search and user navigation. Overall, PageRank offers a valuable tool for understanding and navigating the complex web landscape.

Execution Plan:

1) Project Setup :

Understand and Collect necessary data (e.g., hollins.dat) and identify tools (e.g., GCC compiler, OpenMP library).

2) Data Preparation and Matrix Construction :

Develop functions to read the number of websites and their interconnections from the input file. Parse and store website names and their respective link data.

Initialize Adjacency Matrix (S) :- Create and populate the adjacency matrix representing the link structure.

Stochastic Matrix Calculation :- Convert the adjacency matrix into a stochastic matrix (S) by normalizing the link probabilities.

Transpose Matrix (St) :- Compute the transpose of the stochastic matrix to facilitate PageRank calculations.

3) Optimization and Parallelization :

Optimize Matrix Operations :- Ensure efficient computation of matrix-vector multiplications and adjustments.

OpenMP Integration :- Apply OpenMP pragmas to parallelize the iterative PageRank calculation.

Performance Testing :- Measure execution time and speedup achieved through parallelization. Adjust the number of threads and test for optimal performance.

4) Result Processing and Output:

Sort PageRank Scores :- Implement sorting (e.g., Bubble Sort) to rank websites based on their PageRank scores.

Display Top Results :- Print the top 10 websites with the highest PageRank values.

Parallelization Strategies for the PageRank Algorithm

	Name	Strategy	Benefit	Implementation
1	Partitioning the Graph	Divide the web graph into smaller subgraphs, distributing the nodes and their connections across different processing units.	Reduces the workload per unit and allows simultaneous computation of PageRank values for different sections of the graph.	This can be done using adjacency lists or matrices, distributing the columns or rows of the matrix across processors.
2	Parallel Matrix-Vector Multiplication	PageRank computation involves iterative multiplication of the rank vector with the transition matrix.	Speeds up the convergence process by concurrently calculating multiple components of the rank vector.	Use libraries like OpenMP or MPI to distribute matrix-vector operations across threads or processors.
3	Asynchronous Updates	Instead of synchronizing updates after each iteration, allow processors to update the PageRank values of nodes asynchronously.	Can reduce synchronization overhead and improve convergence speed as processors do not have to wait for each other.	Implement non-blocking communication &independent updates, ensuring convergence by carefully managing dependencies.
4	Dynamic Load Balancing	Distribute the computational load dynamically based on the processing capabilities of each unit and the complexity of the graph sections they are handling.	Maximizes resource utilization and prevents bottlenecks due to uneven workload distribution.	Use dynamic scheduling algorithms and tools like OpenMP's dynamic scheduling features to assign tasks.
5	Using GPU Acceleration	Leverage the parallel processing power of GPUs for matrix-vector operations and other computations involved in the PageRank algorithm.	GPUs can handle large-scale parallel tasks more efficiently than CPUs, greatly speeding up the PageRank computation.	Utilize GPU programming frameworks like CUDA or OpenCL to offload intensive computations to the GPU.

Results : For Serial

serial.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

parallel.c | serial.c | hollins.dat

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <sys/time.h>

int n;
double d = 0.75; // Damping factor

void swap(double *xp, double *yp, char *str1, char *str2) {
    double temp = *xp;
    *xp = *yp;
    *yp = temp;

    char *tempStr = malloc((strlen(str1) + 1) * sizeof(char));
    strcpy(tempStr, str1);
    strcpy(str1, str2);
    strcpy(str2, tempStr);
    free(tempStr);
}

// BubbleSort algorithm that sorts PageRank scores and corresponding websites
void bubbleSort(double* arr, char** web, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] < arr[j + 1]) {
                swap(&arr[j], &arr[j + 1], web[j], web[j + 1]);
            }
        }
    }
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | Build messages | CppCheck/Vera++ | CppCheck/V

File	Line	Message
		=== Build file: "no target" in "no project" (compiler: unknown) ===
		=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===

C:\Users\HP\Desktop\6th Sem\MCAP\Parallel-GaussSeidel-PageRank-main\serial.c   C/C++

"C:\Users\HP\Desktop\6th Se

```
Using damping factor d = 0.750000
Non-parallel time = 2.700139 seconds
The 10 biggest sites are:
1(0.018317): http://www.hollins.edu/
2(0.007229): http://www.hollins.edu/admissions/visit/visit.htm
3(0.006732): http://www.hollins.edu/about/about_tour.htm
4(0.006245): http://www.hollins.edu/admissions/info-request/info-request.cfm
5(0.006210): http://www.hollins.edu/academics/library/resources/web_linx.htm
6(0.006176): http://www.hollins.edu/htdig/index.html
7(0.005657): http://www.hollins.edu/admissions/apply/apply.htm
8(0.004447): http://www.hollins.edu/academics/academics.htm
9(0.004437): http://www.hollins.edu/admissions/admissions.htm
10(0.003219): http://www.hollins.edu/grad/coedgrad.htm
The number of iterations is: 21

Process returned 0 (0x0)   execution time : 5.099 s
Press any key to continue.
```

Results : For Prallel

parallel.c - Code::Blocks 20.03

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <sys/time.h>
#include <omp.h>

int n;
double d = 0.75; // Damping factor

void swap(double *xp, double *yp, char *str1, char *str2) {
    double temp = *xp;
    *xp = *yp;
    *yp = temp;

    char *tempStr = malloc((strlen(str1) + 1) * sizeof(char));
    strcpy(tempStr, str1);
    strcpy(str1, str2);
    strcpy(str2, tempStr);
    free(tempStr);
}

// BubbleSort algorithm that sorts PageRank scores and corresponding websites
void bubbleSort(double* arr, char** web, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] < arr[j + 1]) {
                swap(&arr[j], &arr[j + 1], web[j], web[j + 1]);
            }
        }
```

```
=== Build file: "no target" in "no project" (compiler: unknown) ===
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

C:\Users\HP\Desktop\6th Sem\MCAP\Parallel-GaussSeidel-PageRank-main\parallel.c

```
Using damping factor d = 0.750000
Using 8 threads
Parallel time with 8 threads is = 0.585840 seconds
The 10 biggest sites are:
1(0.018317): http://www.hollins.edu/
2(0.007229): http://www.hollins.edu/admissions/visit/visit.htm
3(0.006732): http://www.hollins.edu/about/about_tour.htm
4(0.006245): http://www.hollins.edu/admissions/info-request/info-request.cfm
5(0.006210): http://www.hollins.edu/academics/library/resources/web_linx.htm
6(0.006176): http://www.hollins.edu/htdig/index.html
7(0.005657): http://www.hollins.edu/admissions/apply/apply.htm
8(0.004447): http://www.hollins.edu/academics/academics.htm
9(0.004437): http://www.hollins.edu/admissions/admissions.htm
10(0.003219): http://www.hollins.edu/grad/coedgrad.htm
The number of iterations is: 21

Process returned 0 (0x0)   execution time : 3.185 s
Press any key to continue.
```

Conclusion

Parallel PageRank code can significantly outperform serial code in terms of execution time, especially for large datasets. This is due to concurrent processing on multiple cores. However, achieving optimal efficiency requires careful design to minimize overhead and ensure coarse-grained tasks. Additionally, Amdahl's Law limits the achievable speedup due to inherent sequential portions of the algorithm. Parallel code offers better scalability as the number of websites increases by distributing the workload among more processing cores.

References

1) Bin Y, Muning K. Concept-based Weighted PageRank Algorithm[J]. Journal of Information, 2006 .

2) Page L, Brin S, MotW.llli R, eta!. The PageRank citation ranking: Bringing order to the Web . Technical report , Stanford University Database Group, 1998

3) Brin S, Page L. The Anatomy of a Large-Scale Hypertextual Web Search Engine[J]. Computer Networks and ISDN Systems, 1998.

4) Xie Chengshun, Zhou Lianying. Research on Evaluation Model of Influence of Microblog Users Based on Page Rank [J]. Information Technology. 2018(05).75-78.

5) Sun Hong, Zuo Teng. Research on Influence Algorithm of Microblog Users Based on Page Rank. [J]. Computer Application Research. 2018(04):1028-1032.

6) Zhang Xin, Ma Ruimin. Research on Core Patent Discovery Based on Improved Page Rank Algorithm [J]. Library and Information Service. 2018(10).106-115.