**Parallelizing Summarizer and Plagiarism Checker**

A PROJECT REPORT

***Submitted by***

BL.EN.U4CSE20165 Sri Naga Jathin P BL.EN.U4CSE20169 Suraj G

BL.EN.U4CSE20186 Udith S



AMRITA SCHOOL OF COMPUTING, BANGALORE AMRITA VISHWA VIDYAPEETHAM BANGALORE 560 035

June – 2023

**Introduction**

The rapid growth of textual data in various domains has led to an increased demand for efficient text processing techniques. Summarization and plagiarism detection are two essential tasks in natural language processing. Summarization aims to condense the key information from a text, while plagiarism detection detects similarities between texts to identify instances of copied content. In this report, we present a comparative study of the performance of parallel and serial implementations of both the summarizer and plagiarism detection code.

**Literature Survey**

The literature survey focuses on understanding the problem statements of text summarization and plagiarism detection. It encompasses an overview of existing algorithms, techniques, and tools employed for these tasks. The survey explores the challenges faced in developing efficient summarization and plagiarism detection algorithms and highlights the importance of parallelism in improving their performance.

**Design and Implementation**

**Parallel Summarizer:**

The parallel summarizer is designed to leverage the power of parallel processing to accelerate the text summarization task. The implementation involves dividing the input text into smaller chunks and assigning them to different processors or threads for parallel processing. Each processor performs the summarization task independently, and the results are combined to generate the final summary.

**Serial Summarizer:**

The serial summarizer serves as a baseline for performance comparison. It follows a sequential approach, where the entire text is processed in a single thread. The summarization algorithm employed is similar to that used in the parallel summarizer.

**Parallel Plagiarism Detection:**

The parallel plagiarism detection code is designed to efficiently compare multiple documents in parallel to identify similarities and detect instances of plagiarism. The implementation involves dividing the input documents among different processors or threads for parallel comparison. Various algorithms and techniques, such as fingerprinting and string matching, are employed to achieve accurate and fast plagiarism detection.

**Serial Plagiarism Detection:**

The serial plagiarism detection code serves as a reference for performance evaluation. It compares documents sequ­entially, utilizing algorithms similar to those used in the parallel plagiarism detection code.

**Results**

The performance of the parallel and serial summarizers and plagiarism detection codes is evaluated based on several metrics, including execution time, efficiency, and speedup. Experimental results demonstrate the impact of parallelism on the performance of these tasks. The comparative analysis provides insights into the advantages and limitations of parallel processing in the context of text summarization and plagiarism detection.

**Conclusion**

The comparative study of parallel and serial summarizers and plagiarism detection codes reveals the benefits of parallel processing in terms of execution time, efficiency, and speedup. The parallel implementations significantly reduce the processing time compared to their serial counterparts. However, the achieved speedup and efficiency may vary depending on the nature of the input data and the parallelization techniques employed. The findings of this study contribute to the understanding of the effectiveness of parallelism in text processing tasks and can guide the development of more efficient and scalable algorithms for text summarization and plagiarism detection.

In conclusion, the parallel implementation of summarization and plagiarism detection codes demonstrates notable improvements in terms of performance. The study highlights the importance of considering parallel processing techniques to enhance the efficiency of text processing tasks, ultimately benefiting various applications in natural language processing and related domains. Future research can explore advanced parallelization techniques and optimizations to further enhance the performance of these tasks.