Graphic Era deemed to be University DEHRADUN

PROJECT AND TEAM INFORMATION

Project Title

CryptoCore: Leveraging Parallelism for Rapid File Encryption

Student / Team Information

Team #OS-V-T002	Team Name: Mighty Bytey
Team member 1 (Team Lead) (Last Name, name: student ID: email, picture):	Maini, Gunottam – 230211521 gunottammaini@gmail.com
Team member 2 (Last Name, name: student ID: email, picture):	Dhyani, Dishant – 23021797 dihantdhyani01@gmail.com

Team member 3 (Last Name, name: student ID: email, picture):	Rawat, Satyam Singh – 230211536 rawatsatyam058@gmail.com
Team member 4 (Last Name, name: student ID: email, picture):	Gusain, Abhinav Singh – 23021580 abhinavgusain5108@gmail.com

PROPOSAL DESCRIPTION (15 pts)

Motivation

In an era of ever-increasing data generation, the need for efficient and secure data handling is paramount. Cryptographic operations, such as file encryption and decryption, are computationally intensive and can become a significant bottleneck, especially when dealing with large files. Traditional, single-threaded encryption tools often fail to fully utilize the power of modern multi-core processors, leading to suboptimal performance. This project is motivated by the need to address this performance gap by exploring and implementing parallel processing techniques in the context of file encryption and decryption. By leveraging multiprocessing and multithreading, we aim to significantly reduce the time required for these critical operations, thereby enhancing both efficiency and user experience. The importance of this project lies in its practical application in various domains, including data security, cloud storage, and secure data transfer, where both speed and security are of utmost importance. This project will also provide a deep, hands-on understanding of fundamental operating system concepts like process and thread management, inter-process communication, and synchronization.

State of the Art / Current solution

Currently, many file encryption solutions rely on a sequential, single-threaded approach. While effective, this method can be time-consuming for large datasets, as it processes data in a linear fashion, one block at a time. More advanced solutions and libraries (like OpenSSL) do incorporate parallelism, but often as a black box, hiding the underlying implementation details from the user. This project distinguishes itself by not only implementing a parallel solution but by also providing a comparative study of two distinct parallel processing paradigms: **multiprocessing and multithreading**. This hands-on approach allows for a deeper understanding of the trade-offs between these two models in the context of a real-world application.

Project Goals and Milestones

The primary goal of this project is to design, implement, and evaluate a high-performance file encryption and decryption tool that utilizes parallel processing. We aim to compare the performance of a multiprocessing-based approach with a multithreading-based approach to determine the most effective strategy for this particular problem.

Milestones:

- **Weeks 1-2: Research & Core Logic**
  - **Week 1:** Conduct in-depth research on parallel processing concepts. Finalize the choice of a symmetric encryption algorithm (e.g., AES) for the implementation.
  - **Week 2:** Design and implement the core single-threaded encryption/decryption logic. This will serve as the baseline for performance comparison.
- **Weeks 3-5: Parallel Implementation**
  - **Weeks 3-4**: Implement the multiprocessing version of the tool using the fork() system call. Focus on inter-process communication and task distribution.
  - **Week 5:** Implement the multithreading version using POSIX threads (pthread), including necessary synchronization mechanisms like mutexes or semaphores.
- **Weeks 6-8: Analysis & Final Deliverables**
  - **Week 6:** Perform rigorous testing of all three versions (single-threaded, multiprocessing, and multithreading) with files of varying sizes to ensure correctness and stability.
  - **Week 7:** Conduct performance analysis. Benchmark the execution times, generate graphs and charts to visually compare the results, and draw conclusions.

- **Week 8:** Finalize the project report, complete the source code documentation, and prepare the final presentation.

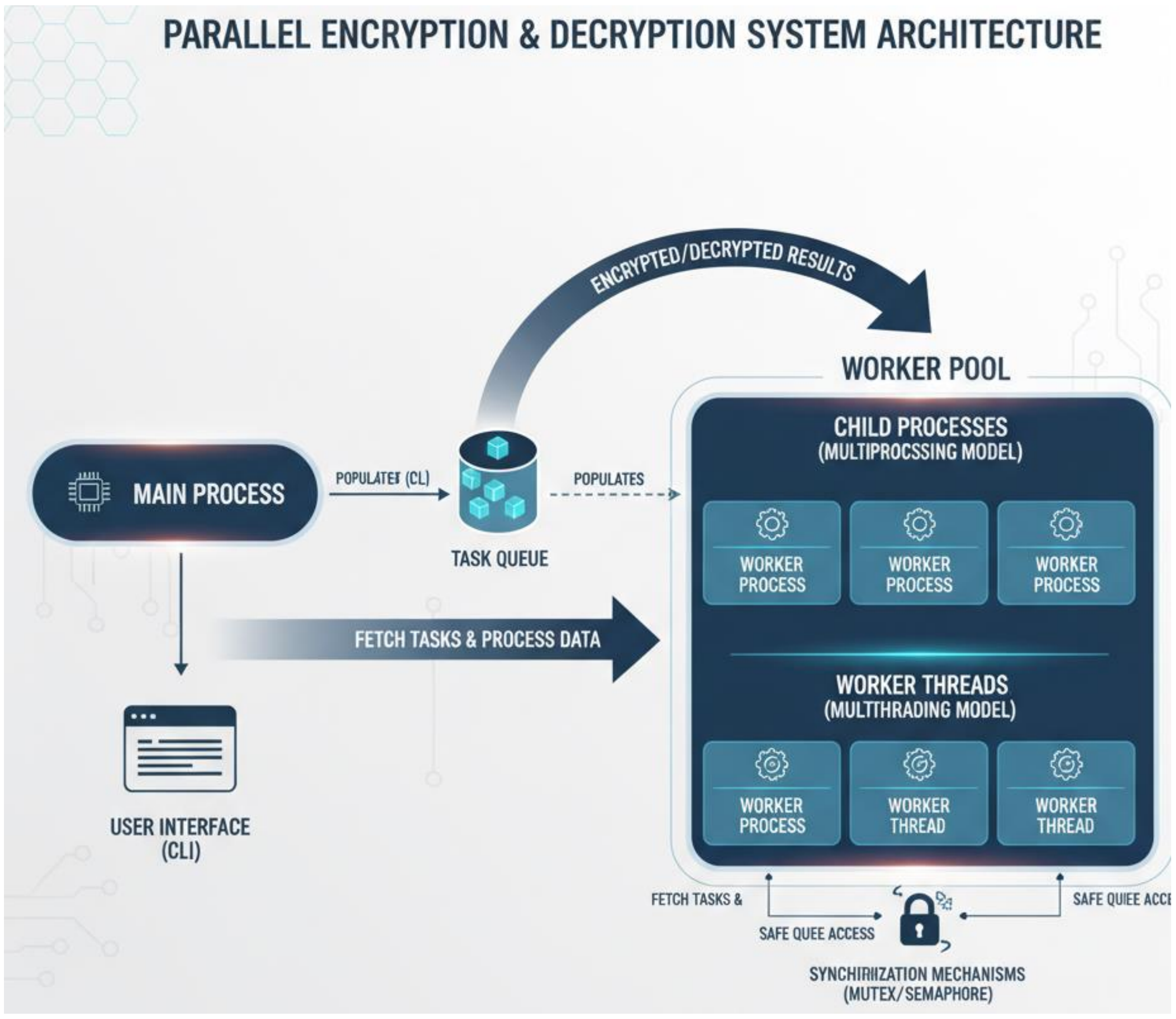
Project Approach

The project will be developed in **C++** on a **Linux/UNIX-based operating system** to take advantage of its rich support for system calls and parallel programming libraries.

- **Platform:** A Linux distribution (like Ubuntu) will be the primary development and testing environment.
- **Technologies:**
  - **C++:** For its performance and low-level control over system resources.
  - **Multiprocessing:** Using the fork() system call to create child processes that will handle encryption/decryption tasks in parallel.
  - **Multithreading:** Using the **POSIX threads (pthread) library** to create and manage threads for concurrent execution.
  - **Synchronization: Semaphores and mutexes** will be used to manage access to shared data structures (like a task queue) in the multithreaded version, preventing race conditions.
  - **Makefile:** To automate the compilation and build process.

The approach will involve creating a central task queue that holds the files (or chunks of files) to be encrypted or decrypted. A main process/thread will populate this queue, and a pool of worker processes/threads will concurrently dequeue tasks and execute them.

System Architecture (High Level Diagram)



Project Outcome / Deliverables

The project will result in the following deliverables:

1. **A fully functional command-line tool, "**CryptoCore,**"** capable of encrypting and decrypting files using single-threaded, multiprocessing, and multithreading modes.
2. **The complete source code** of the project, well-documented and hosted on a Git repository.
3. **A comprehensive project report** detailing the project's design, implementation, and a thorough performance analysis comparing the three different execution models.
4. **A presentation** summarizing the project's objectives, methodology, results, and conclusions.

Assumptions

- The project will be developed and tested on a system with a **multi-core processor** to effectively demonstrate the benefits of parallel processing.
- The primary focus of this project is on the **implementation and comparison of parallel processing techniques**, not on the development of a new, cryptographically secure encryption algorithm. A standard and well-understood algorithm will be used for demonstration purposes.
- The files to be encrypted are accessible to the program and the necessary read/write permissions are available.

References

Books

- Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley.
- Stevens, W. R., & Rago, S. A. (2013). *Advanced Programming in the UNIX Environment*. Addison-Wesley.
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts*. Wiley.
- Pacheco, P. S. (2011). *An Introduction to Parallel Programming*. Morgan Kaufmann.
- Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice*. Pearson.

Technical Documentation & Online Resources

Core Language & Compiler

- C++ Standard Library Reference: https://en.cppreference.com/
- Official C++ Standards Body: https://isocpp.org/
- GCC (GNU Compiler Collection) Docs: https://gcc.gnu.org/onlinedocs/

Parallelism & Synchronization

- POSIX Threads (pthreads) Manual: man pthreads
- The fork() System Call Manual: man 2 fork
- System V Inter-Process Communication (IPC) Overview: man svipc
- POSIX Semaphores Overview: man sem_overview

Build Systems

- GNU Make Manual: https://www.gnu.org/software/make/manual/
- CMake Documentation: https://cmake.org/documentation/

Cryptography & Performance

- NIST FIPS 197 (AES Standard): https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
- Google Benchmark Library: https://github.com/google/benchmark