AHMAD OBEID

[System programming]

[Dr. Samer Sulaiman]

**Discuss the results of the thread management component and report other possible parallelization techniques**

When two threads attempt to access a shared variable simultaneously, a race condition happens. The variable is read by the first thread, and the same value is read by the second thread. The value is then processed by the first and second threads, who compete to be the first to write the value to the shared variable the fastest. Because the thread is writing over the value that the preceding thread wrote, the value of the thread that publishes its value last is retained. The multi threads program was created to handle multiple tasks and distribute them among the threads. The tasks are file generation, functions for counting letters, numbers, and lines, encryption service task, and compressing service tasks. File generation was handled by one thread, the functions for file text analysis were handled by 3 different threads and they worked concurrently. Moreover, additional thread was used for the compressing function or encryption function based on user choice. Race problems were avoided by using mutex to allow ICP to lock before writing and unlock after it, as well as by having threads access the file in read-only mode for task counting and by not sharing writable resources between threads. Fine-Grained jobs: Dividing a large file into smaller, more manageable jobs can improve workload distribution among several threads. Thread pooling allows a fixed number of threads to be managed rather than having to create new threads for each task. By doing this, the overhead of starting and stopping threads is avoided, and system resources can be managed more effectively.   
I could assign jobs to threads from the pool dynamically using a queue of tasks (count letters, numerals, and lines). The thread pool manager is in charge of assigning and finishing tasks. Other potential parallelization strategies are thread pooling and fine-grained tasks.

**Evaluate the effectiveness of the system's design and the developed components in terms of efficiency and execution time?**

First, I wrote to code file for the system, one is multi-process, and one was multi-threads. The first code file I uses the fork to create multi processes for the functions that I have in my program like counting letters, numbers, lines, encryption service, and compressing service. I distributed these functions in the process that I created using fork, and then implemented IPC for the compressing and encryption services. In the second file I did the same, but I used threads instead of fork to make it multi threads. Threads were efficient on some sides and other sides were not. threads share the same memory space and less resource-intensive, which lead to an efficient data sharing and communication. Also, threads require proper synchronization for better results and less execution time, but it can be complex to implement and might lead to complex errors which will be difficult to handle. the execution time in multi threads program was divided on the tasks which are file generation, counting letters, numbers, and lines are handled concurrently by different threads, which should improve execution time, and finally encryption/compression services which are based on user choice. In multi processes, since each process operates in its own memory area, there is less chance that it would crash or cause concurrency problems that could affect other processes. Additionally, controlling processes may be easier to design when shared state is not required. However, compared to threads, processes have higher overhead because of their own memory regions and more costly context shifts. the execution time in multi processes program was divided on the tasks and the same way as multi threads, which are file generation, counting letters, numbers, and lines are handled concurrently by different threads, which should improve execution time, and finally encryption/compression services which are based on user choice. Because they require mutexes for synchronization to handle concurrent access, threads are often more efficient due to shared memory and lower overhead in maintaining threads. They are also more scalable inside a single process. However, because they use more resources and don't have concurrency problems, multiple processes are less scalable and may have problems with IPC and process synchronization.

Execution time for multi threads program



Execution time for multi processes program



Because threads don't have to use slower IPC protocols, multi-threading is typically faster when activities require a lot of data to be shared or when memory access is required frequently. It also performs well in contexts where memory use is a problem because threads use less memory than processes. However, because there is little data exchange, most jobs are autonomous, and there is process creation overhead, multi-processing may perform better than multi-threading.

**Discuss the usage of shell scripting compared to C programming for system services.**

Regarding system services, shell scripting and C programming have various uses. Shell scripts are simpler to write and edit, making them perfect for short automation jobs. They are adept at manipulating files, executing system instructions, and processing text using basic commands such as grep, wc, or awk. Because shell scripts don't need to be compiled, they can be changed and used quickly. But they don't have the same control and performance as C.

However, sophisticated and high-performance applications are better suited for C programming. It offers fine control over memory management, process handling, and system calls by granting low-level access to system resources. Because of this, C is a superior option for developing reliable, effective server-side services like managing multiple clients or encrypting and compressing files. However, compared to shell scripting, C needs more development work, including compilation and debugging, and is less suited for rapid updates.

**References**

1. GeeksforGeeks (2024) Thread pool in C++, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/thread-pool-in-cpp/ (Accessed: 06 September 2024).
2. HaiyingYu (no date) Race conditions and deadlocks - visual basic, Visual Basic | Microsoft Learn. Available at: https://learn.microsoft.com/en-us/troubleshoot/developer/visualstudio/visual-basic/language-compilers/race-conditions-deadlocks (Accessed: 06 September 2024).
3. Ippolito, G. (no date) POSIX thread (pthread) libraries, Linux Tutorial: POSIX Threads. Available at: https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html (Accessed: 06 September 2024).
4. (No date) Intel Guide for Developing Multithreaded Application. Available at: https://www.intel.com/content/dam/develop/external/us/en/documents/gdma-2-165938.pdf (Accessed: 06 September 2024).
5. Edited byMarek SuchánekRed Hat Customer Content Servicesmsuchane@redhat.comMilan NavrátilRed Hat Customer Content ServicesLaura BaileyRed Hat Customer Content ServicesCharlie BoyleRed Hat Customer Content Services et al. (no date) Performance tuning guide, Performance Tuning Guide | Red Hat Product Documentation. Available at: https://docs.redhat.com/en/documentation/red\_hat\_enterprise\_linux/7/html-single/performance\_tuning\_guide/index (Accessed: 06 September 2024).
6. Matb et al. (1960) Measuring time in a multithreaded C++ Application, Stack Overflow. Available at: https://stackoverflow.com/questions/26141484/measuring-time-in-a-multithreaded-c-application (Accessed: 06 September 2024).
7. Gillis, A.S. (2023) What is performance testing and how does it work? – TechTarget definition, Software Quality. Available at: https://www.techtarget.com/searchsoftwarequality/definition/performance-testing#:~:text=Performance%20testing%20helps%20determine%20if,once%20functional%20testing%20is%20completed. (Accessed: 06 September 2024).
8. What is shell scripting? (no date) Coursera. Available at: https://www.coursera.org/articles/what-is-shell-scripting (Accessed: 06 September 2024).
9. GeeksforGeeks (2022) Isalpha() and ISDIGIT() functions in C with cstring examples., GeeksforGeeks. Available at: https://www.geeksforgeeks.org/isalpha-isdigit-functions-c-example/ (Accessed: 06 September 2024).