Alternative Technologies

Ricardo Alfredo Calvo Pérez

14/5/2024

TC2037 - Computers methods implementation

Alternative Tools

During this project we used threading to optimize the program's performance. This technique divides the program into smaller tasks that run concurrently on different CPU cores. Each thread processes a specific file, distributing the workload efficiently and maximizing system resource usage.

Threading is particularly useful for intensive operations. By allowing multiple threads to work simultaneously, we can significantly reduce the total processing time. Instead of sequentially processing files, threading enables multiple files to be processed at once, speeding up the overall process.

Although we can find alternative tools to solve this problem like using the language's own libraries such as Elixir libraries and frameworks like ExDoc and Makeup also provide robust solutions. ExDoc is an official tool for generating documentation for Elixir projects, which includes syntax highlighting and can be customized to output HTML files with highlighted Elixir code. Makeup, an Elixir library specifically designed for syntax highlighting, offers seamless integration and customization within Elixir projects, ensuring compatibility and leveraging the strengths of the Elixir ecosystem.

But we can scale this alternative by changing the programming language to other that has more options like is Python.

Comparison

Using our function `Project.calc_time("function")`, we measured the time in seconds that our program takes to process files in a folder using the lexer. With a folder containing 8 files, the function using threads has an average completion time of 1.5 seconds. This is a noticeable improvement compared to the function that doesn't use threads, which takes an average of 5.5 seconds. The significant difference in performance highlights the efficiency of threading in speeding up the processing time.

On the other hand, using alternative libraries can be unpredictable in terms of specific results. However, from experience, we know that incorporating libraries into a codebase generally makes the development process easier and more efficient. These libraries often provide pre-built functionalities and abstractions that simplify complex tasks, reducing the amount of code developers need to write from scratch.

Nevertheless, there is a trade-off. While libraries can enhance productivity, they can also introduce overhead and dependencies that may slow down the overall performance of the program. Each library adds additional layers of abstraction and, depending on how they are implemented, can increase the memory footprint and processing time. This can be particularly noticeable in performance applications where every millisecond counts.

New technologies

The appearance of new technologies varies greatly across different areas, driven by the unique demands and innovations within each field.

New tools and frameworks appear frequently, often driven by the need for more efficient coding practices, better performance, and enhanced user experiences. For instance, utilizing multiple cores in a computer to make a program run faster showcases how technology evolves to meet performance demands. The tech industry is constantly evolving, with new front-end and back-end technologies, frameworks, and tools emerging regularly. This rapid pace is driven by the need for responsive, user-friendly web applications and the constant push for better performance and security.

Libraries, in particular, appear often and are mass-produced to address common programming needs. They provide pre-built functionalities that simplify complex tasks, reducing development time and effort. Since libraries make coding more accessible and efficient by offering reusable components and standard practices, most programmer upload their own libraries to their favorite programming language, or they upload their code to different public web pages.

Final reflexion

This new technology is helpful for everyone since, as mentioned before, it is more user-friendly and developer-friendly. It allows programs to run faster and makes development easier in many ways.

For users, new technologies often translate into more intuitive and responsive interfaces. For example, modern web development frameworks like React and Angular enable the creation of dynamic and interactive web applications that enhance user engagement and satisfaction.

For developers, these technologies provide powerful tools and libraries that simplify complex tasks. For instance, multithreading libraries and frameworks enable developers to write concurrent and parallel code more easily, improving the performance of applications and reducing development time.

In conclution, new technologies provide significant benefits by improving user experience, streamlining development processes, enhancing performance, and facilitating the maintenance and scalability of applications. These advancements contribute to creating more robust, efficient, and user-friendly software solutions.

References

elixir forum. (2023a, January 8). Any must have libraries lacking from elixir? Retrieved June 15, 2024, from Elixir Programming Language Forum website: https://elixirforum.com/t/any-must-have-libraries-lacking-from-elixir/53048/12

elixir forum. (2023b, November 13). Uniform Elixir Documentation Experience - ExDoc as a Server. Retrieved June 15, 2024, from Elixir Programming Language Forum website: https://elixirforum.com/t/uniform-elixir-documentation-experience-exdoc-as-a-server/59695

Manuel, J., & Carrasco, G. (n.d.). HERRAMIENTAS PARA PROGRAMACION PARALELA. Retrieved from https://webs.um.es/jmgarcia/miwiki/lib/exe/fetch.php?media=esc-ver.pdf

Python. (2021). ParallelProcessing - Python Wiki. Retrieved June 15, 2024, from wiki.python.org website: https://wiki.python.org/moin/ParallelProcessing

stackoverflow. (2010). Does using large libraries inherently make slower code? Retrieved June 15, 2024, from Stack Overflow website: https://stackoverflow.com/questions/2247465/does-using-large-libraries-inherently-make-slower-code