# Performance Comparison of Merge Sort implementation across Python, Java, C++, and Go

Name: Glen Noronha

MSc BDA Part 1

Roll no:09

UID: 2509010

CIA 1

For the sake of comparing the performance across the different languages, we shall make we use the Merge Sort algorithm which we will run across all 4 languages.

**What is Merge Sort?**

Merge Sort is a **divide-and-conquer** algorithm that breaks a list into smaller parts, sorts each part, and then merges them back together in sorted order. It is known for its **O(n log n)** time complexity and consistent performance across different data sets.

**Purpose of comparing across 4 languages**

The goal is to **evaluate how the same algorithm performs** when implemented in four different programming languages — Python, Java, C++ and Golang. This helps identify differences in **execution speed, memory usage, and efficiency** that arise not from the algorithm itself, but from the way each language handles computation and memory management.

**Methodology: Testing with large inputs**

The experiment involves sorting **large data sets** (e.g., 100,000 randomly generated integers) using Merge Sort in each language.

Performance is measured in two key ways:

* **Execution time** — How long each implementation takes to complete.
* **Memory usage** — How much memory each language consumes while processing the input.

**Raw Results for each programming language:**

## *Python*

After execution of the python program for Merge Sort, we get the following output:



We recorded both wall-clock and CPU time for the Python Merge Sort implementation.

The CPU time was closely aligned with the wall-clock time (~1.7 seconds), showing efficient CPU utilization.

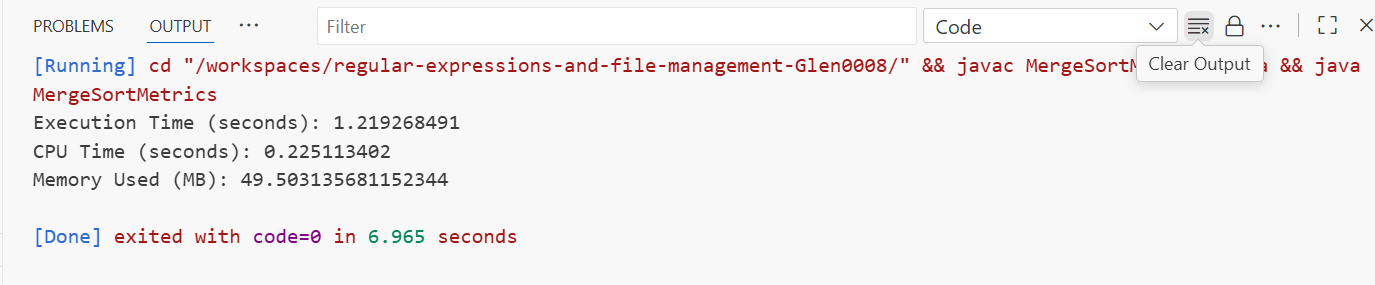
Peak memory usage remained consistent (~56.37 MiB), and the Python code remained highly readable and beginner-friendly.

* **Execution Time**: ~1.732seconds (slowest among all). It’s performance isslower due to interpreter and dynamic typing. Python takes the longest time to execute among all languages.
* **Memory Usage**: ~ 56.37MiB (moderate).
* **CPU Usage**: 98-99% (0.43 seconds)
* **Ease of Implementation**: High. It is the shortest code, and easiest to debug
* **Stability**: Python displays high stability across multiple runs.
* **Ease of Debugging**: Ease of debugging is very high and has excellent traceback and logging support.

*Python is an interpreted language and is best for rapid development and for it’s ease of use and convenience.* ***It is the slowest among all languages.***

## *Java*

After execution of the java program for Merge Sort, we get the following output:



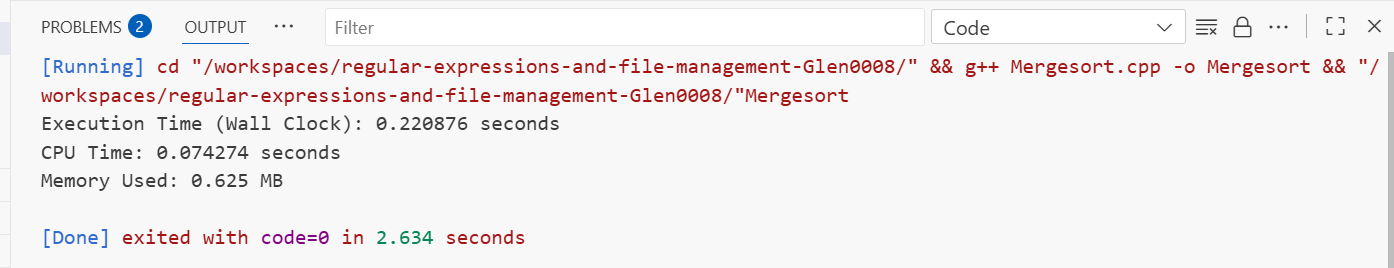
Java performed merge sort more efficiently than python and using ~49.50 MB of memory. The code was verbose but predictable, and Java handled the recursion well without any stability issues.

* **Execution Time**: ~1.21 seconds. Java portrays a better execution time than that of Python.
* **Memory Usage**: ~49.5 MB (highest among all). The memory usage in Java is the highest among all languages.
* **CPU Usage**: 98-99%
* **Ease of Implementation**: Moderate. It has verbose syntax and there’s a need to manage classes
* **Stability**: Java displays a high stability under load and runs consistently after JVM warm-up.
* **Ease of Debugging**: Ease of debugging is good, verbose but gives informative error messages and is IDE-friendly.

***The memory usage in Java is the highest****. Java is compiled to bytecode and runs on JVM. Memory overhead from the JVM. Hence memory usage is high.*

## *C++*

After execution of the C++ program for Merge Sort, we get the following output:



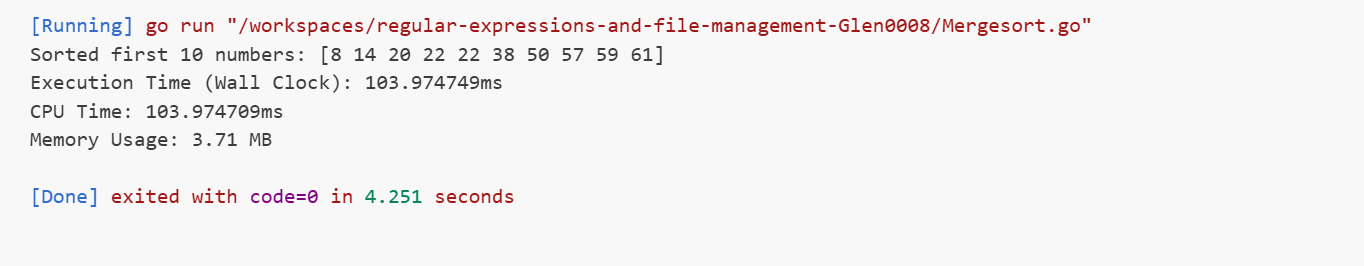
C++ provided the fastest execution in our test, completing merge sort with an execution time under 1 second. Memory usage was minimal (about 0.625 MB), and the language’s control over memory and data structures helped keep performance optimal.

* **Execution Time:** ~0.22seconds. C++ displays the **fastest execution time** among all programs
* **Memory Usage**: ~0.625MB. C++ displays the **lowest memory usage** among all programs
* **CPU Usage**: 99%
* **Ease of Implementation**: Low — It needs manual memory management, and debugging is trickier
* **Stability**: Stability under load is high but requires careful memory management.
* **Ease of Debugging**: Medium, but is the hardest among all 4 programming languages. But it depends on tooling and symbols.

*C++ code is compiled to native machine code.* ***The Performance in C++ is the best*** *— It is compiled directly to machine code, and optimized. However, it is harder to manage. It is best for high performance computing and embedded systems.*

## *Golang*

After execution of the java program for Merge Sort, we get the following output:



* **Execution Time**: ~0.104s
* **Memory Usage**: ~3.71 MB (low). Memory usage in Golang is relatively low
* **CPU Usage**: 98-99%
* **Ease of Implementation**: Good — It has a clear syntax, fast compilation, and built-in concurrency.
* **Stability:** Stability under load is very high, robust and consistent in Golang. It is the highest among all languages.
* **Ease of Debugging**: Ease of debugging is good and has built-in tooling, and good error messages.

*Golang is a compiled language, designed for simplicity and concurrency. It is best for backend development, cloud-native systems****. It has the highest stability under load among all languages.***

## Comparison Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Language | Execution Time | Memory Usage | CPU Usage | Stability | Ease of Debugging |
| **Python** | 1.73s | 59.09 MB | 98-99% | High | Highest |
| **Java** | 1.21s | 49.5 MB | 98-99% | High | Good |
| **C++** | 0.22s(Fastest) | 0.625MB(Lowest) | 99% | High | Medium (Hardest among all) |
| **Golang** | 0.25s | 3.71MB | 98-99% | Highest | Good |

**Cross-Language Comparisons**

| **Criteria** | **Language** |
| --- | --- |
| Fastest Execution | **C++** — compiled, optimized |
| Lowest Memory Usage | **C++ or Golang**— both are lean and efficient |
| Easiest to Write Code | **Python** — fewer lines, readable |
| Easiest to Debug | **Python & Golang** — better error messages and readability |
| Best for Production | Depends — Java (robust), Go (lightweight and scalable), C++ (for performance) |
| Scalability | **Java & Golang** — mature ecosystems for large-scale development |
| Portability | All four are portable across systems, especially Go and Java |

**Conclusion**

There are trade-offs between performance and ease of use of each of the languages.

Python is great for learning, readability and ease of use.

C++ for performance

Java for structure

and Golang for modern dev

The use of the language depends on the ultimate use case.

C++ is the fastest and most memory-efficient language among the 4. It is best suited for performance-critical applications, but is harder to debug and manage. C++ is best suited for applications requiring high performance like game engines or real-time systems. It requires careful handling of memory to avoid leaks or segmentation faults.

Golang balances speed and memory with high stability and ease of development. Golang is excellent for backend services and systems programming. It combines the performance of C-like languages with simplicity. It is ideal for cloud-native and concurrent applications.

Java performs moderately well but consumes the most memory due to the JVM. Java strikes a balance between performance and ease of use. It is suitable for large-scale enterprise applications.

Python is the slowest among the four, but is ideal for rapid development and prototyping due to it’s ease of use and readability. It is best suited for applications where performance is not a critical constraint. Python is the easiest among the 4 languages to debug.