Performance evaluation of concurrent languages: C++, Go & Rust

CS550 Advanced Operating Systems

Florentin Bekier & Rémi Blaise
Supervised by Neeraj Rajesh

Introduction & motivations

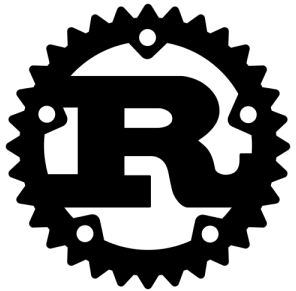
- C++:
  - Extension of C (1972)
  - First stable release: 1985
  - Latest stable release: C++17 (December 2017)
  - Object-oriented and many more features

Introduction & motivations

- Go:
  - Developed by Google
  - First stable release: 2012
  - Motivated by a dislike of C++
  - Also similar to C but with more safety, readability and usability

Introduction & motivations

- Rust:
  - Developed by Mozilla
  - First stable release: 2015
  - Similar to C and C++
  - Multi-paradigm language focused on performance, safety and ease of development



Introduction & motivations

- Goal: perform benchmark on C++, Go and Rust to compare their performance and features
- Why?
  - Go and Rust are concurrent of C++
  - They were developed to be more efficient
  - C and C++ are still among the most used languages almost 50 years after their creation

Methodologies

1. Establishment of a set of benchmarks based on the common features of the languages, their specificities and the most critical features for programming
2. Selection of the metrics that will be measured and used to compare the languages
3. Implementation of the benchmarks for each language, paying attention to evaluate elements comparable
4. Compilation and execution of the benchmarks in the same environment and with an equivalent level of optimization for each language
5. Evaluation based on the metrics

Methodologies

Benchmark list:

- Hello World (basic program)
- Memory management (allocation, …)
- Iteration & recursion
- Data structures (arrays, vectors, trees)
- Matrix operations (multiplication)
- Concurrency mechanisms
- Strings
- Data serialization (binary, XML, JSON)
- Hashing (hash maps, cryptographic hash)
- Sockets (TCP)
- Timers precision and time read
- Computation: complex numbers

Metrics:

- Binary size
- Execution time
- Memory usage
- CPU usage

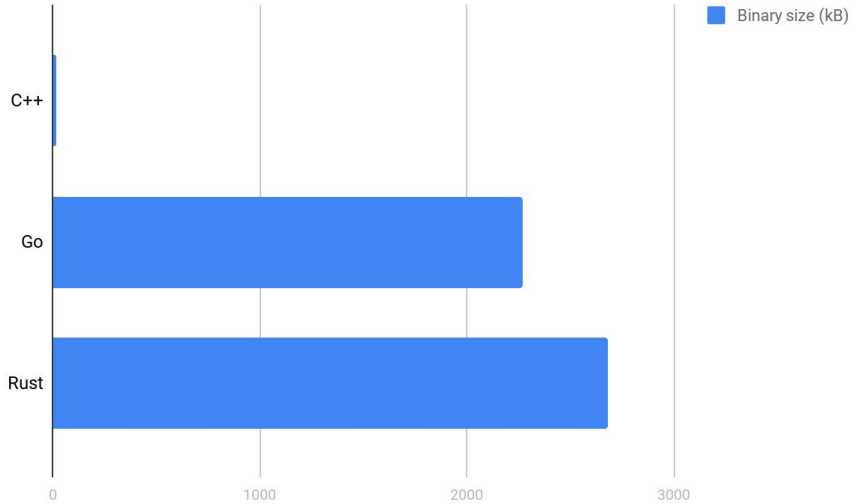
Results: compilation

Language	Total compilation time (s)	Number of scripts	Avg compilation time (s)
C++	13.75	40	0.34
Go	7.43	38	0.19
Rust (first time)	53.12	39	1.36
Rust (second time)	1.11		0.03

Table 1: Compilation time

Machine: Intel Core i5 9300H (9th gen) 8-core, 2.4 GHz, 7.6GB RAM

Results: binary size

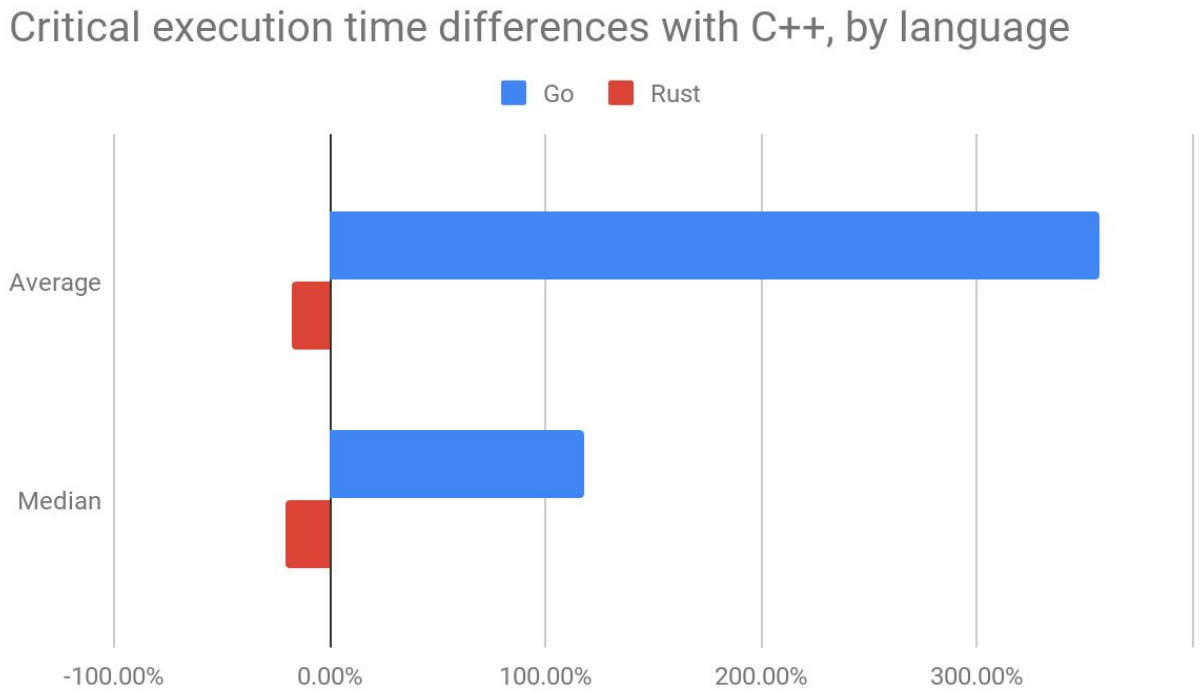


Language	Binary size (kB)
C++	17.95
Go	2269.22
Rust	2682.44

Table 2: Binary size

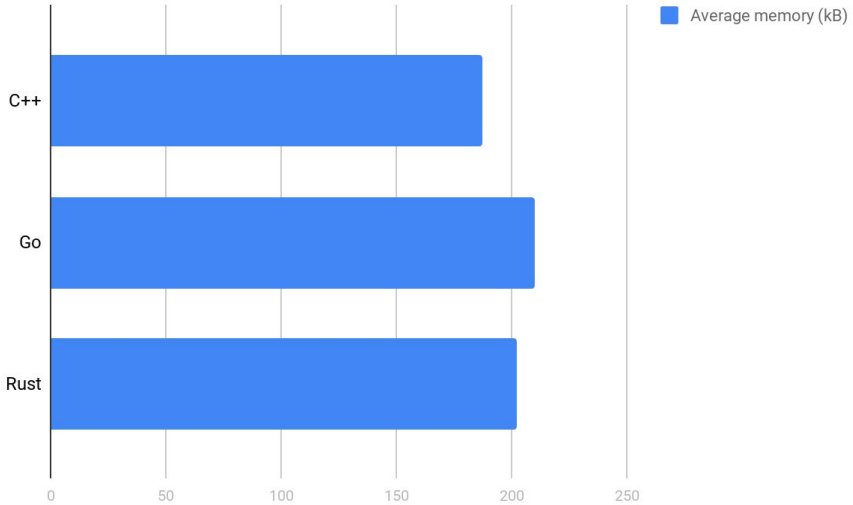
Machine: Intel Core i5 9300H (9th gen) 8-core, 2.4 GHz, 7.6GB RAM

Results: execution time



Machine: Intel Core i5 9300H (9th gen) 8-core, 2.4 GHz, 7.6GB RAM

Results: memory usage



Language	Average reserved memory (kB)
C++	187.38
Go	209.91
Rust	202.32

Table 3: Reserved memory

Machine: Intel Core i5 9300H (9th gen) 8-core, 2.4 GHz, 7.6GB RAM

Conclusion

- C++ is still the best language for programming on resource-limited devices (e.g. embedded chips) because of the memory usage and binary size
- Rust seems to be a good alternative to C++ in terms of performance
- Go is definitively slower than C++ and Rust
- Rust and Go introduce memory safety and programming efficiency
  - Better than C++ for developers
  - More frequent updates

References

[1] Bjarne Stroustrup. 1996. A history of C++: 1979--1991. History of programming languages---II. Association for Computing Machinery, New York, NY, USA, 699–769. DOI: https://doi.org/10.1145/234286.1057836

[2] Go language FAQ. https://golang.org/doc/faq

[3] Jason Kincaid. 2009. Google's Go: A New Programming Language That's Python Meets C++. Techcrunch. https://techcrunch.com/2009/11/10/google-go-language/

[4] Alexey Lozovsky. 2018. Rust vs C++ Comparison. Apriorit. https://www.apriorit.com/dev-blog/520-rust-vs-c-comparison

[5] Daniel Munoz. 2015. After all these years, the world is still powered by C programming. Toptal. https://www.toptal.com/c/after-all-these-years-the-world-is-still-powered-by-c-programming

Thank you!
Q&A