Maintainability, Readability, Performance

What are we optimizing for?

Dr. Matthias Kretz

GSI Helmholtz Centre for Heavy Ion Research

2025-01-22

@mkretz@floss.social

github.com/mattkretz

Why we use C++

We use C++ because ...

1. we maintain and extend an existing C++ codebase?
2. important dependencies we'd like to use are C++ libraries?
3. because the higher-ups say so?
4. because our last project(s) was/were in C++ (familiarity)?
5. because we need the performance?
6. because it's a dependable, stable, and standardized language?
7. ...

Using C++ comes at a cost

The cost depends on available alternatives and the task at hand.

- Every language has to make trade-offs.
- Evolution of a language at some point either breaks compatibility or overall consistency.

C++ is old and inherts C which is even older …

… how does this affect performance and software quality?

Using C++ comes at a cost

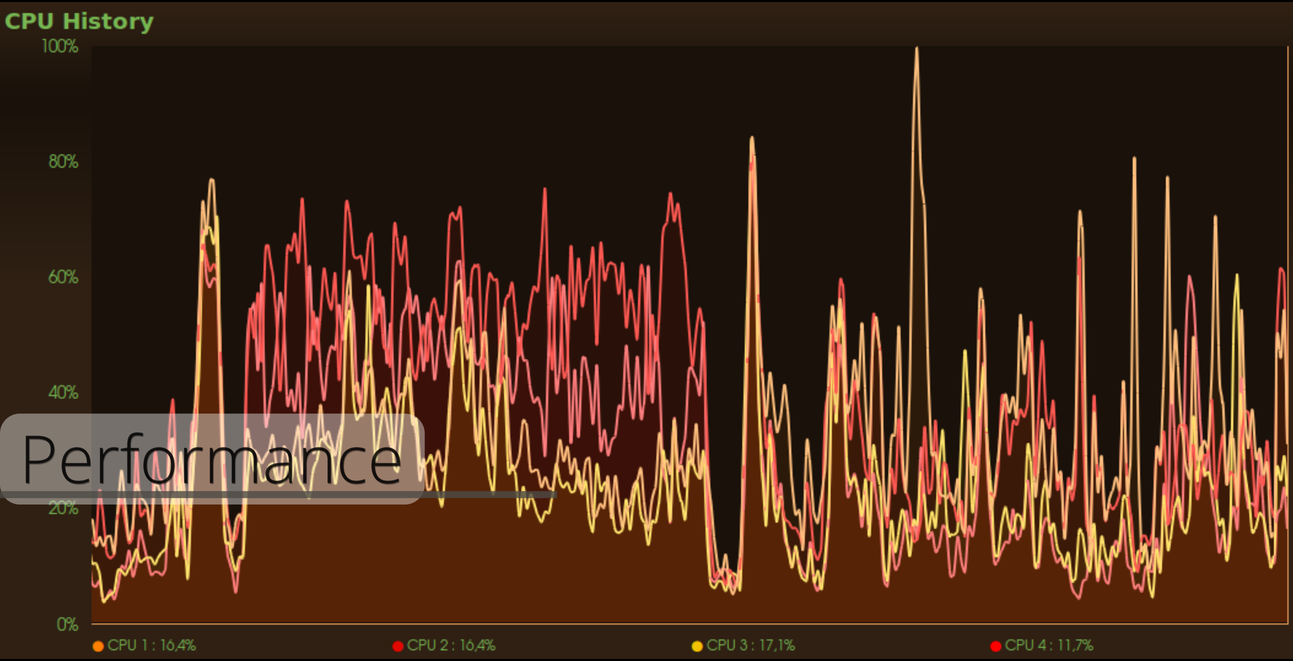
The cost depends on available alternatives and the task at hand.

- Every language has to make trade-offs.
- Evolution of a language at some point either breaks compatibility or overall consistency.

C++ is old and inherts C which is even older ...

... how does this affect performance and software quality?

Performance



Performance

Performance

The efficiency of executing a program with regard to time and computing resources.

Wikipedia: Computer performance

the amount of useful work accomplished by a computer system

Which of the following do you call “performance increase”?

	shorter response time	lower resource utilization
higher CPU core count	(1)	(2)
different/newer CPU architecture	(3)	(4)
newer/different compiler	(5)	(6)
the source code was changed	(7)	(8)

Performance

Performance

The efficiency of executing a program with regard to time and computing resources.

Wikipedia: Computer performance

the amount of useful work accomplished by a computer system

Which of the following do you call “performance increase”?

	shorter response time	lower resource utilization
higher CPU core count	(1)	(2)
different/newer CPU architecture	(3)	(4)
newer/different compiler	(5)	(6)
the source code was changed	(7)	(8)

Performance of a Programming Language

- C++: leave no room for a lower-level language except Assembly
- C++: zero-overhead abstractions
- C++: provide primitive operations with higher-level abstractions on top
- …?

And what about compilation time?

Performance of a Programming Language

- C++: leave no room for a lower-level language except Assembly
- C++: zero-overhead abstractions
- C++: provide primitive operations with higher-level abstractions on top
- ...?

And what about compilation time?

Software Quality

Maintainability

Wikipedia: Maintainability

Maintainability is the ease of maintaining or providing maintenance for a functioning product or service. Depending on the field, it can have slightly different meanings.

[…] Closely related concepts in the software engineering domain are evolvability, modifiability, technical debt, and code smells.

The maintainability index is calculated with certain formulae from lines-of-code measures, McCabe measures and Halstead complexity measures.

The measurement and tracking of maintainability are intended to help reduce or reverse a system's tendency toward "code entropy" or degraded integrity, and to indicate when it becomes cheaper and/or less risky to rewrite the code than it is to change it.

Maintainability (2)

clean and easy-to-read code vs. unorganized and difficult-to-read code

- modularity
- understandability
- changeability
- testability
- reusability
- transferability between teams

These do not take the form of critical issues at the code level.

poor maintainability typically the result of thousands of minor violations with best practices in:

- documentation
- complexity avoidance strategy
- basic programming practices
- programming language idioms
- component reuse

`https://en.wikipedia.org/wiki/Software_quality#Maintainability`

Readability

Wikipedia: Readability of source code

[…] readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. It affects the aspects of quality above, including portability, usability and most importantly maintainability.

Readability is important because programmers spend the majority of their time reading, trying to understand, reusing, and modifying existing source code, rather than writing new source code. Unreadable code often leads to bugs, inefficiencies, and duplicated code.

Software Quality

Software Quality aspects:

- Reliability (coding errors)
- Robustness (handle invalid inputs)
- Usability
- Portability
- Maintainability
- Efficiency / Performance

https://en.wikipedia.org/wiki/Computer_programming#Quality_requirements

Software Quality

Software Quality aspects:

- Reliability (coding errors)
- Robustness (handle invalid inputs)
- Usability
- Portability
- Maintainability
- Efficiency / Performance

Performance is an aspect of software quality!

https://en.wikipedia.org/wiki/Computer_programming#Quality_requirements

Reliability

how often the results of a program are correct

- conceptual correctness of algorithms
- minimization of programming mistakes (resource management, logic errors)

Metric: FLOC

number of software faults per line of code (FLOC), usually expressed as faults per thousand lines of code

Theory

software reliability increases as the number of faults (or fault density) decreases

Practice

tests at several levels, starting with individual units, through integration and full-up system testing

https://en.wikipedia.org/wiki/Reliability_engineering#Software_reliability

Robustness

ability of a computer system to cope with

- errors during execution
- cope with erroneous input

Testing Practice

- fuzz testing
- fault injection

Coding Practice: Paranoia

- assume users are out to break their code
- assume code may fail or work incorrectly

digression: C++26 Contracts

```cpp
int
operator[](size_t i)
pre(i < size())
{
  return data_[i];
}
```

```cpp
class PositiveInt
{
  unsigned value_;
  // ...
  friend PositiveInt
  operator+(PositiveInt a, PositiveInt b)
  post(r: r >= a and r >= b)
  {
    return PositiveInt(a.value_ + b.value_);
  }
}
```

Contracts help with *Reliability* and *Robustness*

Portability

- CPU architecture
- also CPU vs GPU?
- different compiler
- performance portability

Strategies:

- avoid or abstract hardware features?
- avoid or abstract system libraries?
- use (and contribute to) existing abstractions

Usability

the ease with which a person can use the program for its intended purpose
or in some cases even unanticipated purposes

Often ignored when developers and users are the same group.
And that's often the case for Research Software.

Conjecture

bad usability hinders reuse / collaboration

FLOC

ease of use

compatibility

collaboration

reliability

trade-off

time

dependencies

Discuss!

portability

performance

testing

maintainability

robustness

reuse

money

usability

readability

familiarity

paranoia