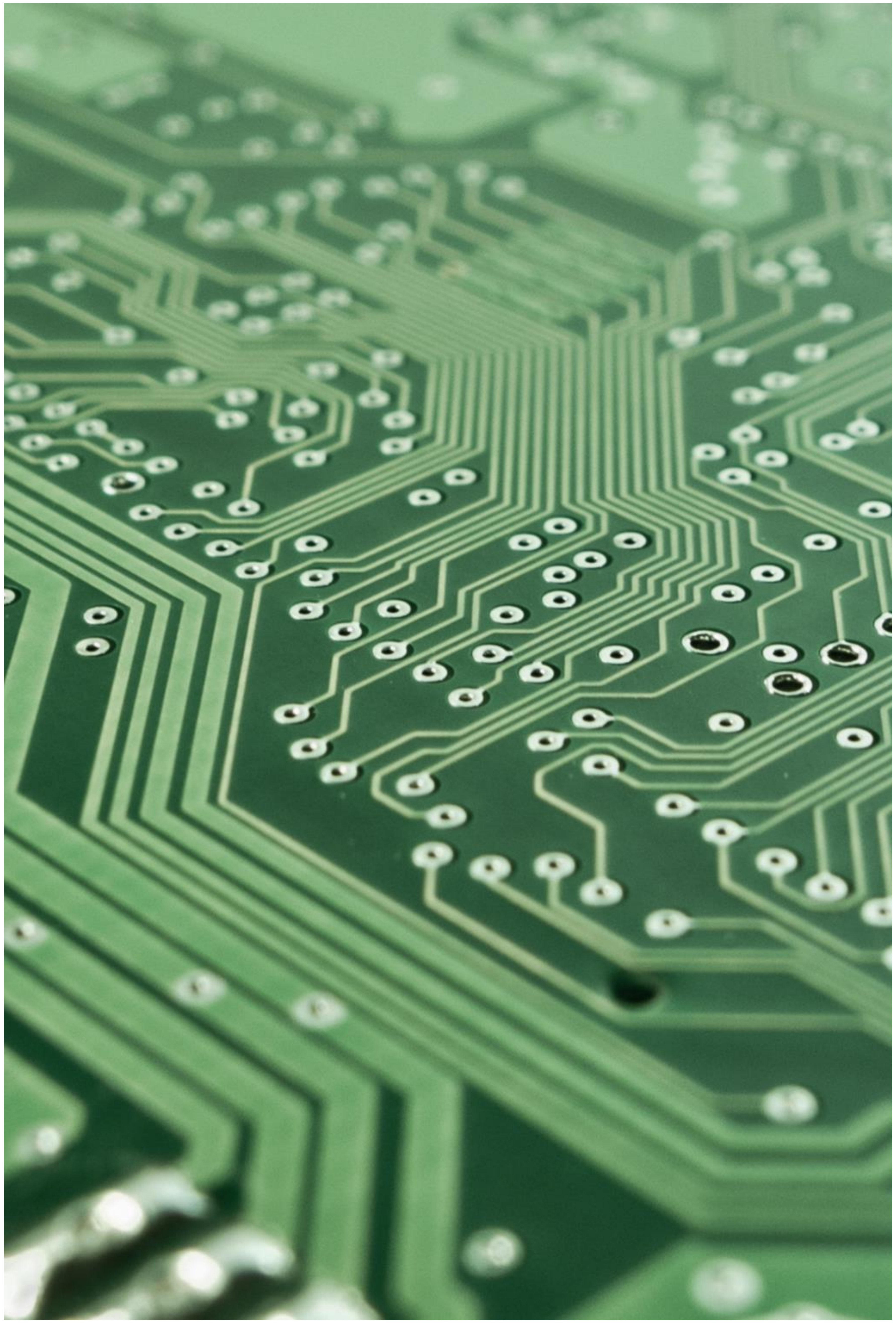
Challenges in Modern Embedded Development Using C++

Glyn Matthews, SoftKinetic

Belgian C++ User Group – 11/04/2017

Solving hard problems is why we do what we do: we must remember that we should be problem solvers first, programmers second.

The technology of tomorrow… today!

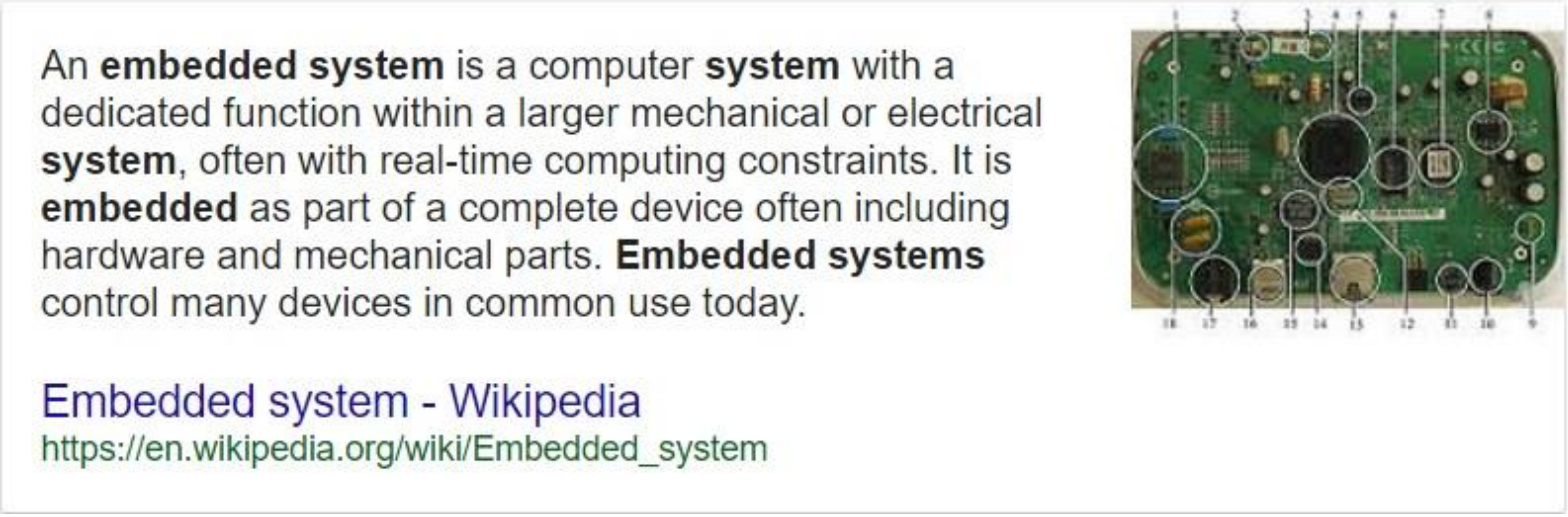


SONY

WALL·E

CHROMA

Systems



An **embedded system** is a computer **system** with a dedicated function within a larger mechanical or electrical **system**, often with real-time computing constraints. It is **embedded** as part of a complete device often including hardware and mechanical parts. **Embedded systems** control many devices in common use today.

Embedded system - Wikipedia
https://en.wikipedia.org/wiki/Embedded_system

Not about microcontrollers (deep embedded)
https://www.youtube.com/watch?v=TYqbgvHfxjM

?

Constrained systems for:

- high performance
- real-time
- low latency
- high availability
- mission/safety critical

Using:

- computer vision
- machine learning
- other cool stuff

Why do people use C++ to solve these problems?

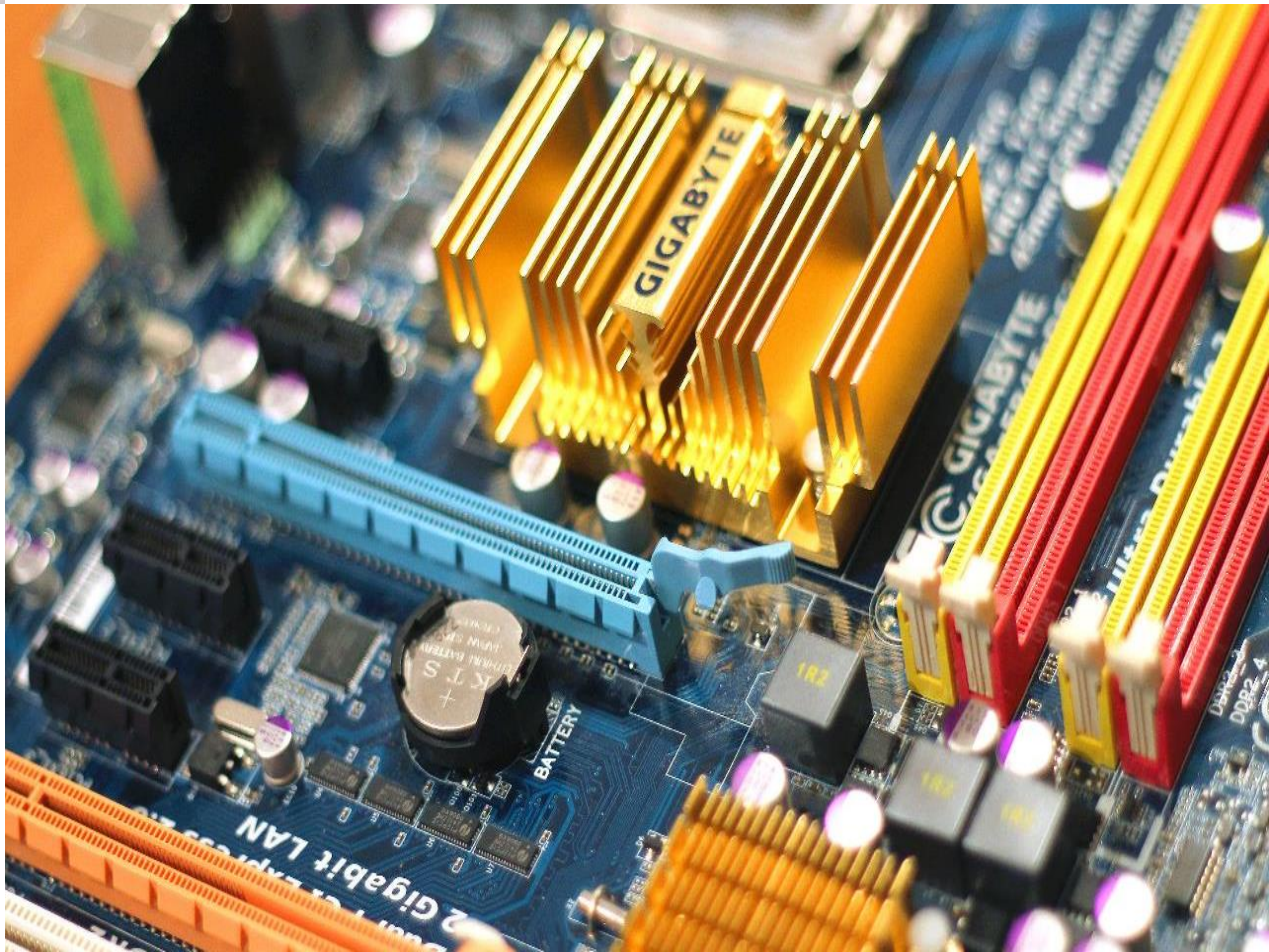
C++ doesn’t give you performance, it gives you control over performance

Chandler Carruth

https://www.youtube.com/watch?v=fHNmRkzxHWs

Large Scale

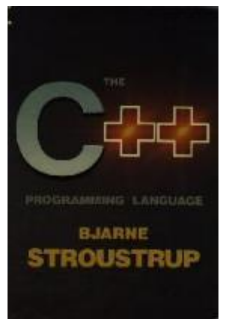
Small Scale



C++ History

C++ History

1979:
C with classes



1985: The C++ Programming Language: 1st Edition

1987: C++ support in GCC

1990: ANSI C++ Committee founded

C++ History

1991: ISO Committee founded

1995: MSVC Initial release

1998: C++98 Standard published

2003: C++03 Standard published

C++ History

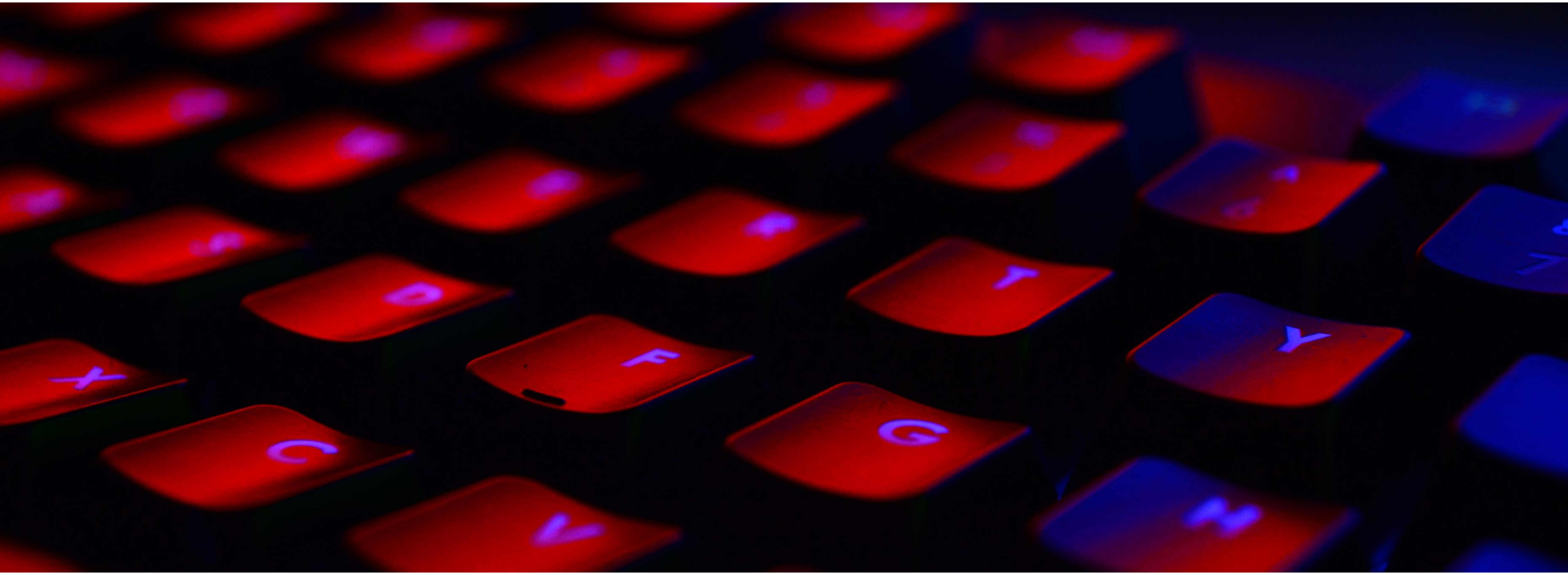
2007: Clang
Initial release

2012: C++11
Standard published

2015: C++14
Standard published

**2018: C++17
Standard will
be published**

Modern C++



Computers

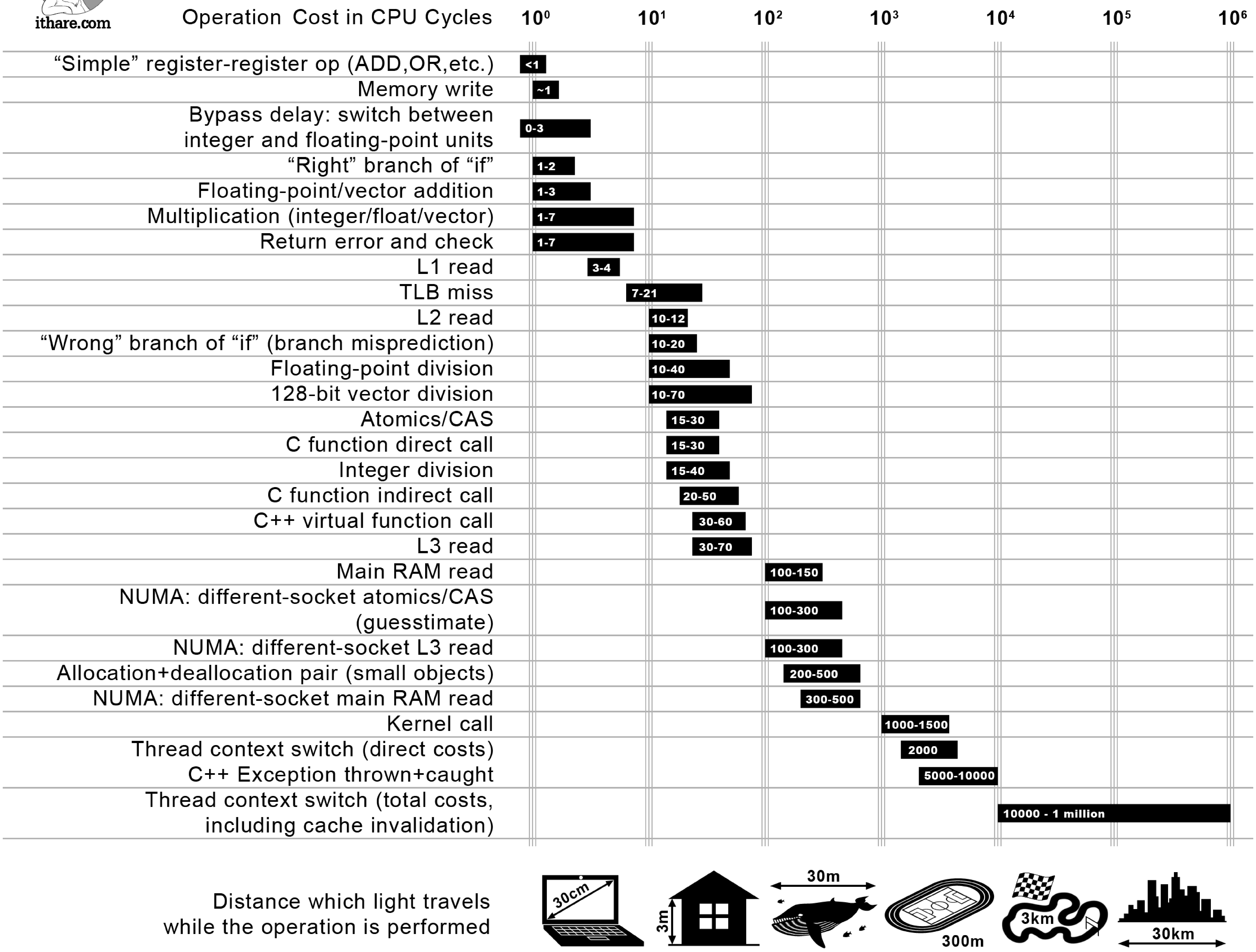
Computer Resources

We care about:

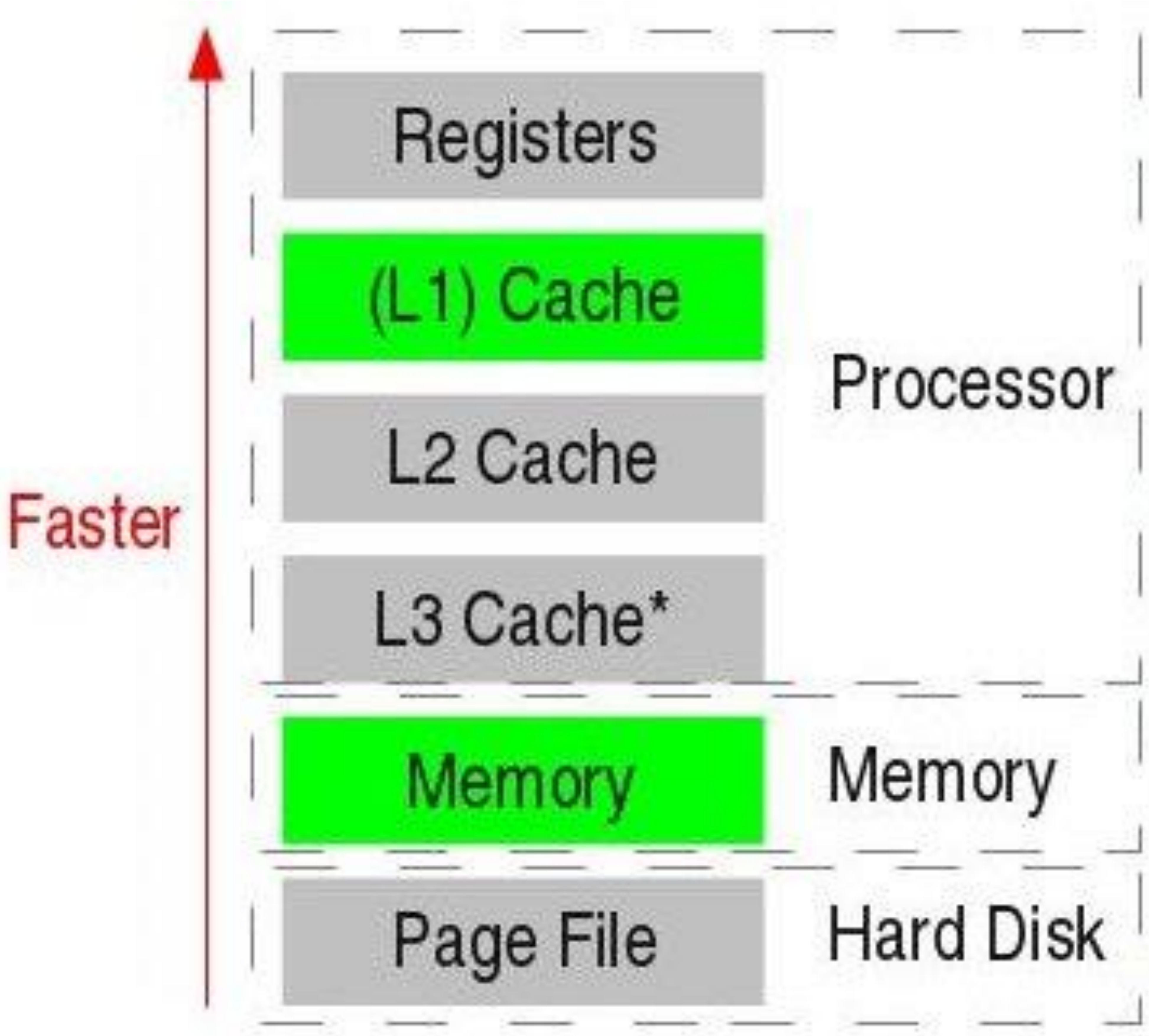
- The processor(s) and it’s core(s)
- Memory
- Filesystem
- Network
- Etc.

ithare.com

Not all CPU operations are created equal



Memory Hierarchy



C++ Features

A FILM BY ANTON CORBIJN
CONTROL
SAMANTHA MORTON SAM RILEY ALEXANDRA MARIA LARA
WINNER BEST ACTOR-SAM RILEY EDINBURGH FILM FESTIVAL
WINNER BEST NEW BRITISH FILM EDINBURGH FILM FESTIVAL
WWW.CONTROL-MOVIE.COM

How do you take control using modern C++?

}

Some Opinions About Smart Pointers

- Use unique_ptr to indicate ownership
- Take advantage of the deleter for resources such as files
- reference_wrapper is your friend when using containers

Use unique_ptr to indicate ownership

```cpp
struct Vector3 {
  int x, y, z;
  Vector3() : x(0), y(0), z(0) {}
};

void compute_point_cloud(Vector3 *point_cloud, std::size_t length);

{
  auto point_cloud = std::make_unique<Vector3[]>(100);
  auto t1 = std::thread(compute_point_cloud, point_cloud.get(), 50);
  compute_point_cloud(point_cloud.get() + 50, 50);
  t1.join();
}
```

Take advantage of the deleter for resources such as files

```
using file_ptr = std::unique_ptr<FILE, decltype(&std::fclose)>;

{
  auto file = file_ptr(std::fopen(“cpp.txt”, “r”), &std::fclose);
  if (file) {
    // process file
  }
}
```

reference_wrapper is your friend when using containers

```cpp
struct BigData { bool is_interesting() const; };

auto data = std::vector<std::unique_ptr<BigData>>{ 100 };

{
  auto filtered_data = std::vector<std::reference_wrapper<BigData &>>{};

  std::for_each(
    std::begin(data), std::end(data),
    [&filtered_data](auto &data) {
    if (data->is_interesting()) {
      filtered_data.emplace_back(std::ref(*data.get()));
    }});
}
```

Some More Opinions

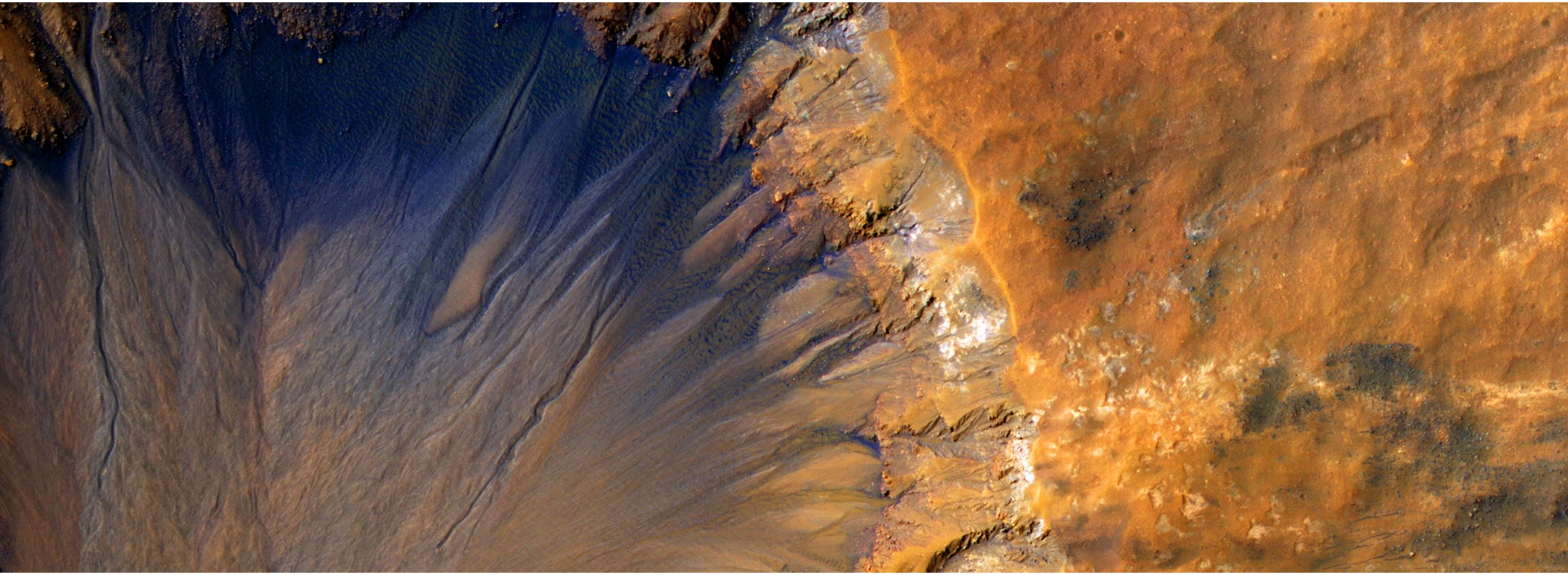
- Reference counting (using shared_ptr) is rarely needed in practice, especially in synchronous code
- Simply share data by passing by reference to functions (unless you’re transferring ownership)

C++ string_view (1)

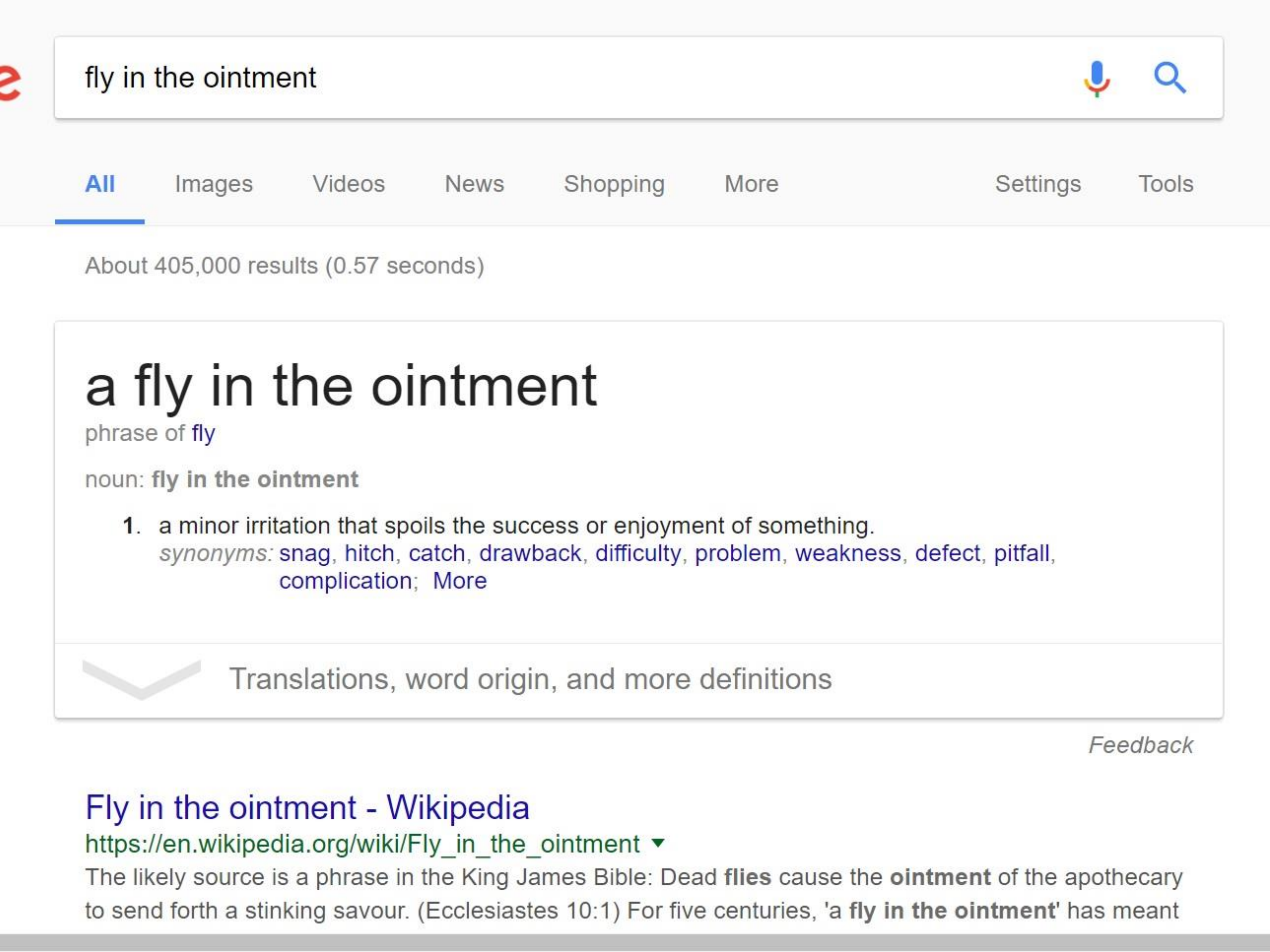
```
auto dna_sequence = std::string(“ACTGCGACGGTACGCTTCGACGTA”);

std::vector<std::size_t> find_occurrences(
  std::string_view dna_sequence,
  std::string sub_sequence);

auto sequence1 = std::string_view(dna_sequence.c_str(), 12);
auto sequence2 = std::string_view(dna_sequence.c_str() + 12, 12);

auto occurrences1 = std::async(find_occurrences, sequence1, “ACG”);
auto occurrences2 = find_occurrences(sequence2, “ACG”);
occurrences1.wait();
```

C++ string_view (2)

```cpp
int __cdecl c_find_occurrences(
  const char *dna_sequence, size_t length, const char *sub_sequence,
  size_t **occurrences, size_t *occurrence_count) {
  auto occurrences = find_occurrences(
    std::string_view(dna_sequence, length),
    std::string(sub_sequence));

  // fill in return values

  return 0;
}
```



Flies in the Ointment



About 405,000 results (0.57 seconds)

a fly in the ointment

phrase of fly

noun: **fly in the ointment**

1. a minor irritation that spoils the success or enjoyment of something.
   synonyms: snag, hitch, catch, drawback, difficulty, problem, weakness, defect, pitfall, complication; More

Fly in the ointment - Wikipedia

https://en.wikipedia.org/wiki/Fly_in_the_ointment ▾

The likely source is a phrase in the King James Bible: Dead **flies** cause the **ointment** of the apothecary to send forth a stinking savour. (Ecclesiastes 10:1) For five centuries, 'a **fly in the ointment**' has meant

Inefficient Standard Data Structures

The standard specification for following data structures requires that they are sub-optimal:

- `list`
- `map`
- `set`
- `unordered_map`
- `unordered_set`

ordered_table (1)

```
template <class K, class V,
          class Allocator = std::allocator<std::pair<K, V>>>
class ordered_table {
public:
  ordered_table(std::initializer_list<std::pair<K, V>> values)
    : table_{values} {
    sort();
  }

private:
  void sort() {
    std::sort(begin(), end(), key_less_than{});
  }

  std::vector<std::pair<K, V>, Allocator> table_;
};
```

ordered_table (2)

```
bool try_insert(const K &key, const V &value) {
  auto found = std::binary_search(
    begin(), end(),
    std::make_pair(key, value), key_less_than{});

  if (!found) {
    table_.emplace_back(key, value);
    sort();
  }

  return found;
}
```

ordered_table (3)

```
const_iterator find(const K &key) const {
  auto value = std::make_pair(key, mapped_type{});
  auto it = std::lower_bound(
    begin(), end(), value, key_equals{});
  return (it != end()) && (!key_equals(*it, value)) ?
          it : end();
}
```

Cross-Platform Development

- When you don’t own the target platform
- When the target platform doesn’t support the language feature

C++ Trends

Low Latency – WG21/SG14

- Games
- Low Latency
- Real-time Applications
- Graphics
- Financial Trading

SG14 (the GameDev & low latency ISO C++ working group) - Guy Davidson - Meeting C++ 2016

https://www.youtube.com/watch?v=IuJ79Og-CfU

A library for Study Group 14 of Working Group 21 (C++)

https://github.com/WG21-SG14/SG14

C++ Core Guidelines

This document is a set of guidelines for using C++ well. The aim of this document is to help people to use modern C++ effectively. By “modern C++” we mean C++11 and C++14 (and soon C++17). In other words, what would you like your code to look like in 5 years’ time, given that you can start now? In 10 years’ time?

From the abstract of the C++ Core Guidelines

http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines

Write for portability and performance

- Be conservative in using language and library features
- Understand your target platform’s CPU, cache and memory
- Use cache-friendly data structures
- Consider memory locality

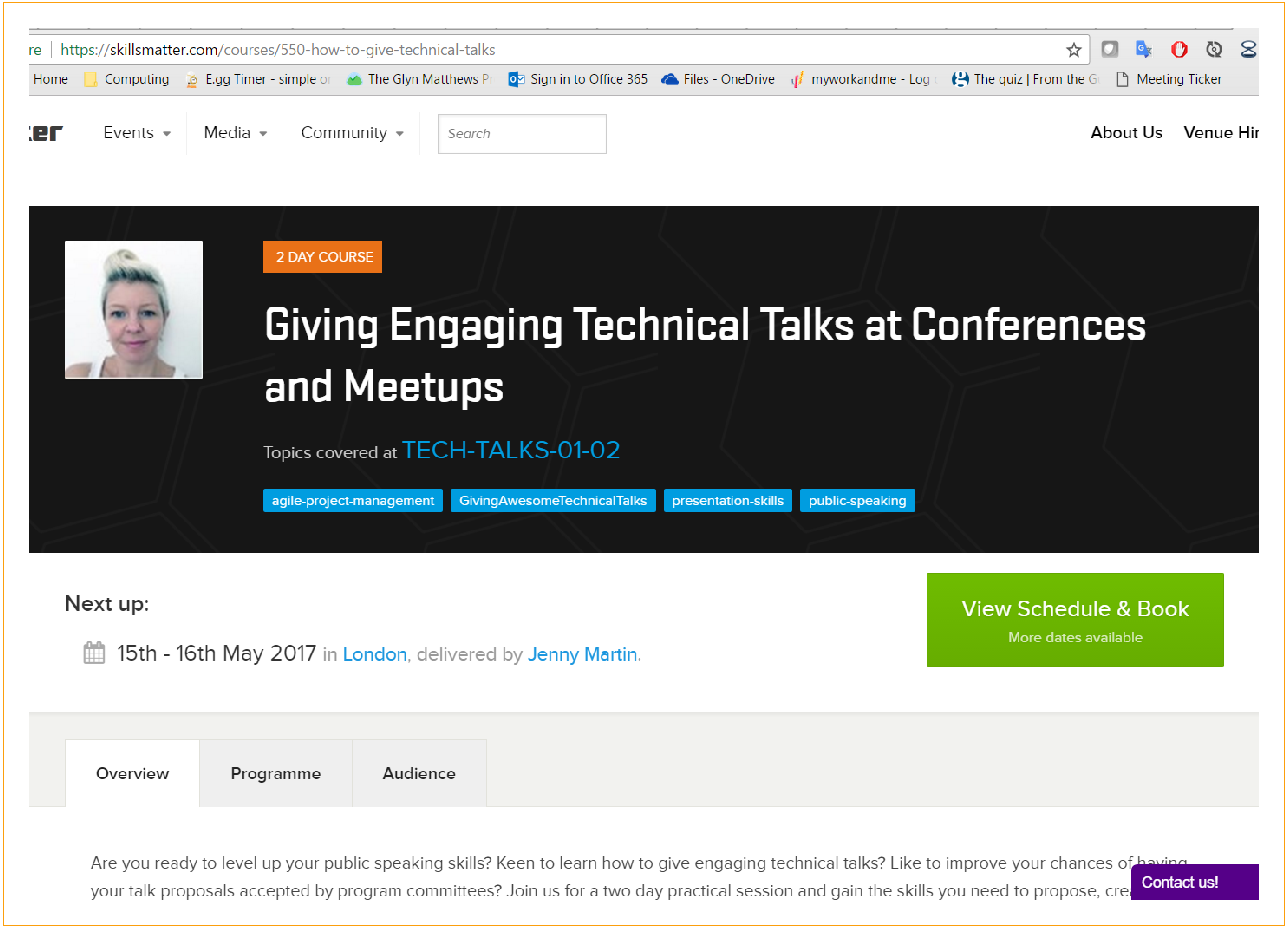
Measure and test

- Run benchmarks as part of acceptance testing

Follow core guidelines

- Helps understand trade-offs
- Still incomplete, but usable
- Tooling exists to support the core guidelines (e.g. c-tidy)

Solving hard problems is why we do what we do: we must remember that we should be problem solvers first, programmers second.



2 DAY COURSE

Giving Engaging Technical Talks at Conferences and Meetups

Topics covered at TECH-TALKS-01-02

agile-project-management GivingAwesomeTechnicalTalks presentation-skills public-speaking

Next up:

15th - 16th May 2017 in London, delivered by Jenny Martin.

View Schedule & Book
More dates available

Are you ready to level up your public speaking skills? Keen to learn how to give engaging technical talks? Like to improve your chances of having your talk proposals accepted by program committees? Join us for a two day practical session and gain the skills you need to propose, cre