Exploring Compiler Explorer

Marc Paterno

14 October 2020

Section 1

Demo

Is `int` faster as a loop index than `size_type`?

- A C-style array is accessed by an index of type `int`.
- A C++ `std::vector` or `std::array` is accessed by an index of type `size_type` (often `unsigned long`).
- Some people prefer to use `int` to access `std::vector` values, because “the smaller type is faster”.
- Is this true?
  - g++ 10.2: https://godbolt.org/z/dbEcWj.
  - clang++ 10.0.1: https://godbolt.org/z/P9r93b.

How does g++ compare to clang for this code?

- We can compare the output of two (or more) compilers on the same code.
- For this, we'll see the graph output can be useful.
  - clang++ and g++ comparison: https://godbolt.org/z/bnz7vo.

How do the range algorithms perform?

- We can use a selection of 3rd party libraries
  - https://godbolt.org/z/98e7sG.
- To see the difference betwen the generated code, use the top-level “Add” button and add a “diff view”.
- Compiler Explorer has Intel C++ also. On code like this, it can vectorize the sum. Let’s compare g++, clang++, and icc:
  - https://godbolt.org/z/4oKbfj.

Use llvm-mca to understand a bit more

- We can select the loop from the assembly code, and paste the results into the llvm-mca tool for *machine code analysis*.
  - Start from https://godbolt.org/z/ohKbb9.
- Copy code; switch editor to new “language” *analysis*, and paste in code
- Set the “compiler flags” `-timeline -mcpu=skylake` (or whatever hardware you like)
  - Result is https://godbolt.org/z/sf44vK.

Compiler conformance view

- One of my favorite features of C++17 is *structured bindings*:

```
#include <map>
#include <string>
int foo(std::map<std::string, int> const& m) {
   int result = 0;
   for (auto const& [k, v] : m) result += v;
   return result;
```

- How new a compiler does one need to use this?
  - start from https://godbolt.org/z/Yn6aaW.
  - Use “add new. . . ” to add a new “conformance view”; add compilers.
  - result: https://godbolt.org/z/vo8frr.

Online microbenchmarks

- Compiler Explorer has a link to https://quick-bench.com, but I have not got it to work.
- I just paste code directly in to QuickBench.
- You can try a few difference compilers.
- You can try different libraries for Clang.