Beyond Short String Optimisation: The Merits of Fixed-Length Strings

Jamie Taylor
jtaylor91@protonmail.com

So… std::string… It’s pretty nice!

- Class-based representation of a string: `std::string mystring( “Hello C++ London!” );`
  - Easier and safer to work with than a C string
- This is much better than a C string! :D
  - `void foo(const char*, size_t)` becomes `void foo(const std::string&)` etc.
  - Memory allocation is handled for you
  - Short String Optimisation (SSO) buffer: short strings go on the stack

But it’s not perfect...

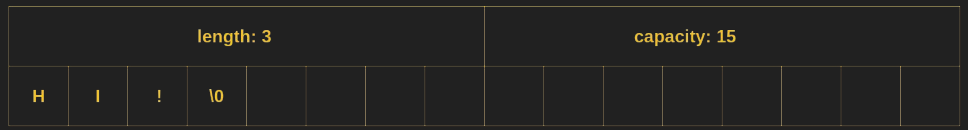
- SSO is implementation-defined
- Absence of an SSO buffer will result in dynamic allocation
- Strings too long for the SSO buffer will result in dynamic allocation
- std::string instance size is unavoidable, regardless of actual string length
- Non-optimal cache utilisation
- Some use-cases could benefit from an alternative...

Introducing: fl::string

- A fixed-length string implementation

```
class std::string {
        // ...
        size_t length, capacity;
        char sso[16]; // union with char*
};
```

```
template<size_t length>
class fl::string {
        // ...
        char m_data[length];
};
```



- Interoperability with std::string and C strings using std::string_view
    - inline operator string_view() const noexcept { return string_view{m_data, length()}; }

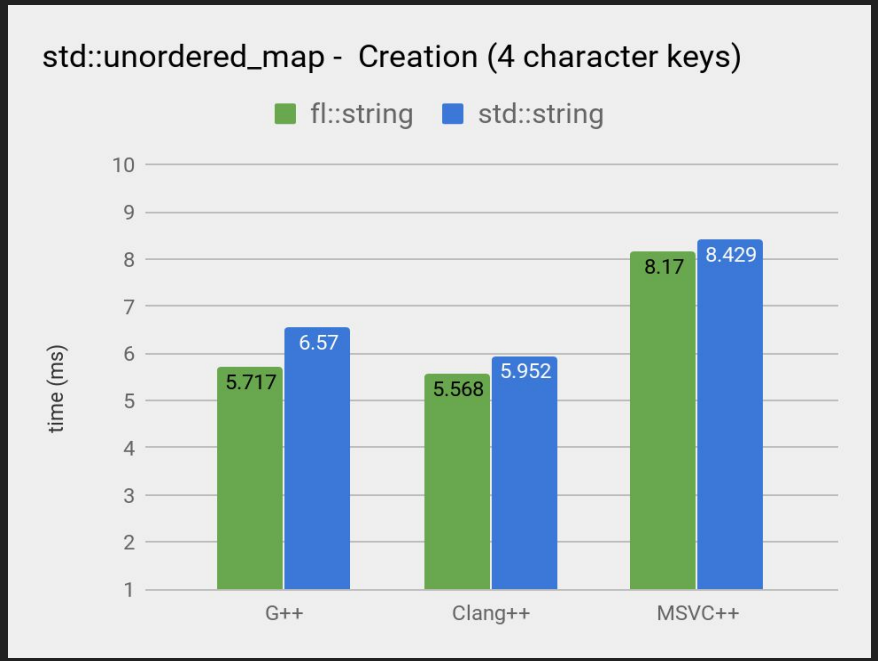
Test: Common string operations

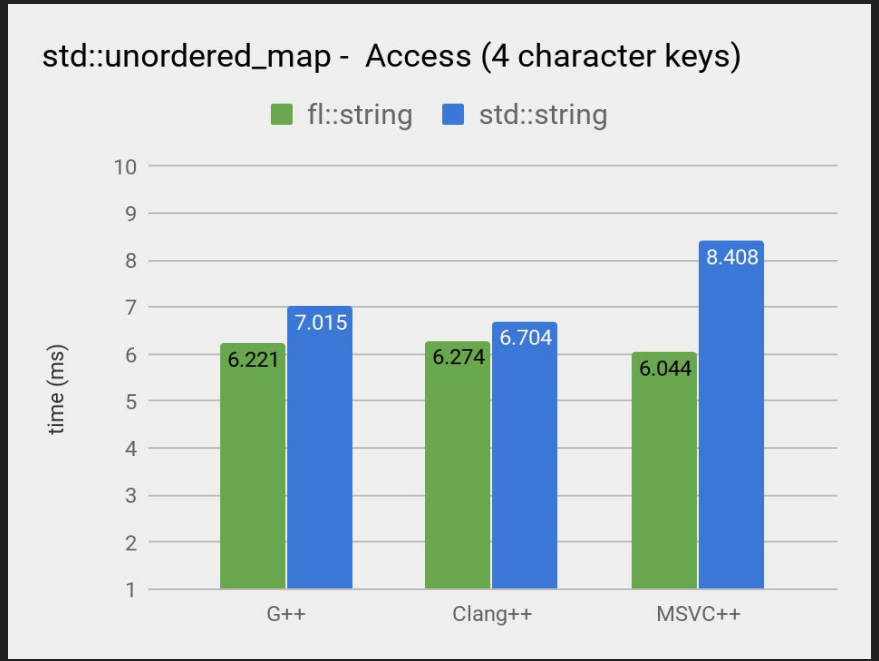
- % improvement in string operations (8 and 32 character):
  - Construction: string::string(const char *c) - 259% and 1049%
  - Assignment: string::operator=(const char *c) - 939% and 1750%
  - Concatenate: string::operator+=(const char *c) - 668% and 1360%
  - Avg. 6x and 13x speedup respectively

Test: std::unordered_map

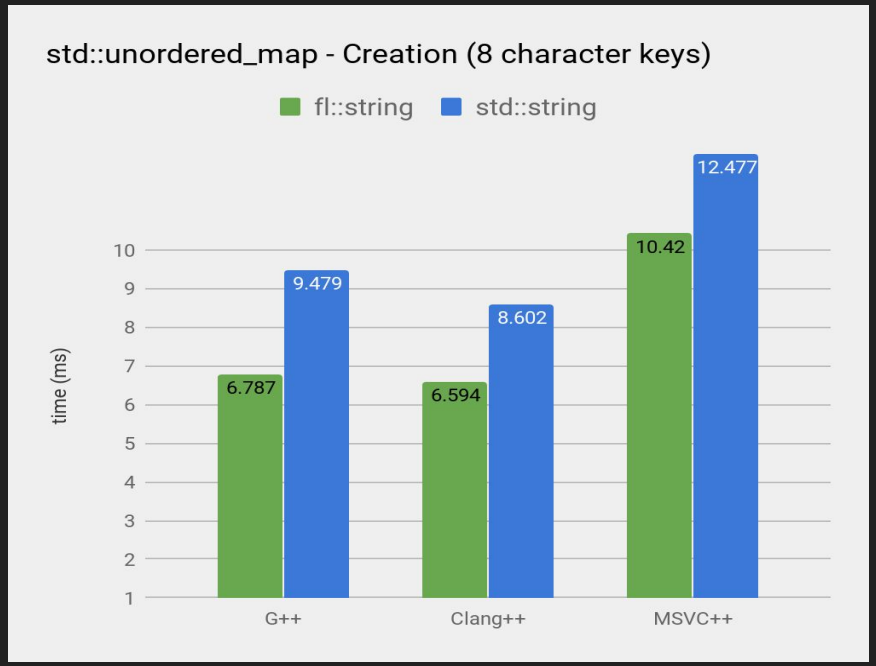
- Take a std::unordered_map<> of 128 <string, unsigned int> pairs
  - Key is a 4 or 8 character std::string or fl::string
  - Value represents the number of times the key has been searched for
- Record the time taken to:
  - 0) Create the map
  - 1) Perform 256 randomly-generated look-ups

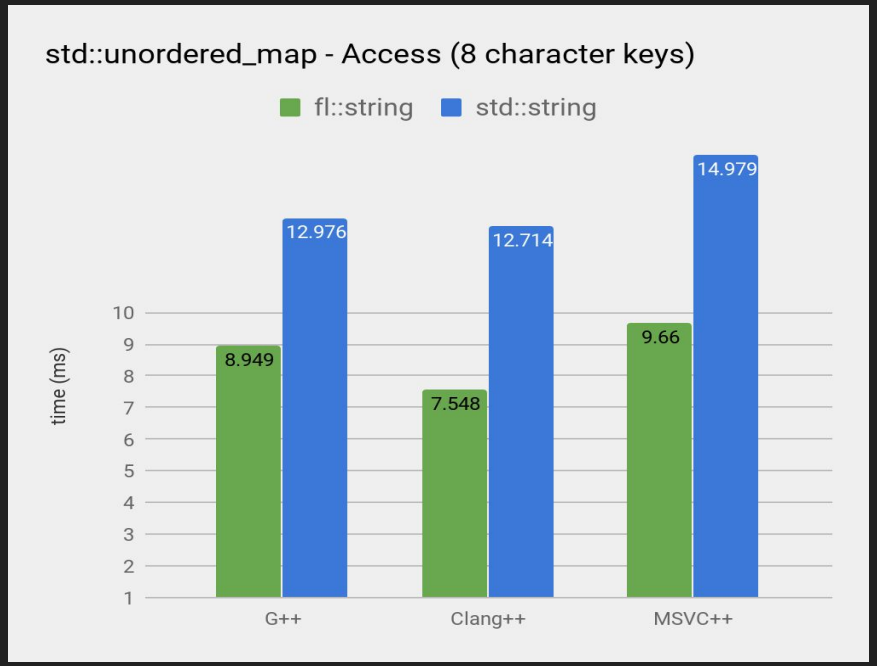
Results: std::unordered_map - 4 char keys





Results: std::unordered_map - 8 char keys





Conclusions

- std::string isn't the be-all-end-all
  - May dynamically allocate
  - Even a 1 char string will be sizeof(std::string) in size
- fl::string offers complete control over memory layout
  - No dynamic allocation
  - Improved cache-coherency
  - Last byte free-space/null-terminator double-up
- Demonstrable performance gains in real-world scenarios
  - String ops: construct, assign and concatenate
  - Containers: unordered maps
- Great fit for:
  - Embedded, Games, Trading - low-latency

Thank you!

jtaylor91@protonmail.com
https://gitlab.com/jt396/flstring

References

- https://david-grs.github.io/inplace_containers_for_fun_and_profit/
- https://shaharmike.com/cpp/std-string/
- https://www.facebook.com/Engineering/videos/10151029396848109

Appendix: Testing Methodology

- % improvement in string operations (8 and 32 character):
  - Construction: string::string(const char *c) - 259% and 1049%
  - Assignment: string::operator=(const char *c) - 939% and 1750%
  - Concatenate: string::operator+=(const char *c) - 668% and 1360%
- Avg. 6x and 13x speedup
- Performed with 8 and 32 character strings (7 and 31 + null-terminator)
- Compilers Used: Clang++5.0.0 G++7.0.0 (MSVC data incomplete? See git.)

Appendix: Testing Methodology cont.

- Take a std::unordered_map<> of 128 <string, unsigned int> pairs
  - Value is simply the number of times the key has been searched for - nothing special
- Perform N number of look-up operations on said maps
  - We're going to use 256
  - Look-ups are generated randomly and the same sequence is used for both the fl::string and std::string maps
- Record the time taken to:
  - 0) Create the map
  - 1) Perform the specified number of look-ups
- Repeat this M times
  - We're going to go for 1024 times
  - So: record the time taken to create the map and perform 256 look-ups 1024 times and average
  - Both 'fixed-length->std::string' and 'std::string->fixed-length' orderings used for fairness
- Performed with 4 and 8 character strings (3 and 7 + null-terminator)
- Compilers Used: Clang++5.0.0, G++7.0.0 and MSVC++2017 (19.10.25019)