Performance improvement opportunities in the open source C++ standard libraries

Aditya Kumar
Software Engineer, Facebook

linkedin.com/in/hiraditya
twitter.com/_hiraditya_

Why bother?

Back-end (Server-side) table in most popular websites

Websites	C#	C	C++	D	Erlang	Go	Hack	Java	JavaScript	Perl
Google	No	Yes	Yes	No	No	Yes	No	Yes	Yes	No
YouTube	No	Yes	Yes	No	No	Yes				No
Facebook	No	No	Yes	Yes	Yes	Yes				No
Yahoo	No	Yes	Yes	No	No	Yes				Yes
Amazon	No	No	Yes	No	No	No	No	Yes	No	Yes
Wikipedia	No	No	No	No	No	No	No	No	No	No
Twitter	No	No	Yes	No						
Bing	Yes	No	Yes	No	No	No	No	No	No	No

Also:

- **All smartphones**
- **Airplanes**
- **Browsers**

- ***Where there is C++, there is a C++ standard library***

Source: https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites

Opportunities in

- Standard library Containers
- Standard library algorithms
- Source code annotations
- Compiler optimizations

Standard Library Containers (string::rfind)

```
template<class _CharT, class _SizeT, class _Traits, _SizeT __npos>
inline _SizeT
__str_rfind(const _CharT *__p, _SizeT __sz,
            _CharT __c, _SizeT __pos) _NOEXCEPT
{
    if (__sz < 1)
        return __npos;
    if (__pos < __sz)
        ++__pos;
    else
        __pos = __sz;
    for (const _CharT* __ps = __p + __pos; __ps != __p;)
    {
        if (_Traits::eq(*--__ps, __c))
            return static_cast<_SizeT>(__ps - __p);
    }
    return __npos;
}

// Similarly string::find first of, 81 string::find first not of
```

Standard Library Containers (std::iostream, std::locale)

- Parsing of characters
  - libcxx uses std::find to search for a character while parsing [O(n) slow!]
- The atom string have different layout for libstdc++ vs libcxx

libstdc++

```
constexpr char hex_digits[] = "abcdefABCDEF0123456789";
constexpr auto dec_digits = hex_digits + 12; // alignment issue?
const char* __num_base::_S_atoms_in = "-+xX0123456789abcdefABCDEF"; // +-xX occurs strictly less frequently than 0..9
```

libcxx

```
const char __num_get_base::__src[33] = "0123456789abcdefABCDEFxX+-pPiInN"; // parsing negative numbers slower than positive numbers?
```

Standard Library Algorithms(std::sort)

- libc++’s sort does not have worst case O(n^2) guarantee

Standard Library Algorithms(std::find)

```cpp
#include <algorithm>
#include <cstdlib>

int arr[] = {1,2,3,4,5,6,7,8,9,10};
int BenchmarkLinearSearch(int n){
    int r = std::rand() % n;
    auto result = std::find(begin(arr), end(arr), arr[r]);
    return *result;
}
```

clang unrolls by different factors with libc++ (5) and libstdc++ (8), gcc unrolls by a factor of 4 consistently.

https://godbolt.org/z/WerYE1

Source code annotations

- Annotating non-returning functions
- Annotating branches with builtin-expect
- Annotating pointers with restrict
- Annotating functions with noexcept
- Visibility attributes

Source code annotations (Annotating non-returning functions)

```
template <bool>
class __split_buffer_common
{
protected:
    void __throw_length_error() const;
    void __throw_out_of_range() const;
};
```

Source code annotations (Annotating branches with builtin-expect)

```
template <class _CharT, class _Traits>
streamsize
basic_streambuf<_CharT, _Traits>::xsgetn(char_type* __s, streamsize __n)
{
    const int_type __eof = traits_type::eof();
    int_type __c;
    streamsize __i = 0;
    while(__i < __n)
    {
        if (__ninp_ < __einp_) // [[likely]]
        {
            const streamsize __len = _VSTD::min(static_cast<streamsize>(INT_MAX),
                                     _VSTD::min(__einp_ - __ninp_, __n - __i));
            traits_type::copy(__s, __ninp_, __len);
            __s +=  __len;
            __i +=  __len;
            this->gbump(__len);
        }
        else if ((__c = uflow()) != __eof)
        {
            *__s = traits_type::to_char_type(__c);
            ++__s;
            ++__i;
        }
        else
            break;
    }
    return __i;
}
```

Source code annotations (Annotating pointers with restrict)

- string::copy(value_type* __s, size_type __n, size_type __pos) calls
  - char_traits::copy(char_type* __s1, const char_type* __s2, size_t __n)
  - Could use restrict as overlap is not allowed

Annotating functions with noexcept

- Noexcept is already annotated consistently
- Should we annotate throwing functions with noexcept as well?
  - What if the application just aborts for all the exceptions. No need to have exception handling code in the first place

```
#ifdef ADD_UNSAFE_NOEXCEPT
#define MAY_NOEXCEPT noexcept
#else
#define MAY_NOEXCEPT
#endif
template <class _Tp, class _Allocator>
void vector<_Tp, _Allocator>::__vallocate(size_type __n) MAY_NOEXCEPT
{
    if (__n > max_size())
        this->__throw_length_error();
    this->__begin_ = this->__end_ = __alloc_traits::allocate(this->__alloc(), __n);
    this->__end_cap() = this->__begin_ + __n;
    __annotate_new(0);
}
```

Visibility attributes

- Make all the function definitions internal to a shared library
  - __attribute__((internal))

Question: What is the default visibility of functions in C/C++ programs?

Question: What does -fvisibility=hidden do?

Reference: https://gcc.gnu.org/wiki/Visibility

Compiler optimizations

- Discrepancies in compiler optimization flags of the runtime vs application
- Different OS distributions may build libraries with different compilation flags
  - -O2/-O3/-Os ?
- Which compiler flags are best for C++ standard libraries?
- Inconsistent exception flags in libc++. Issue with -fno-exceptions
  - The .cpp files in standard library may have been compiled with exceptions enabled

Compiler optimizations...

- Whole program devirtualization
- Inlining important function
- Auto vectorization
- Loop idiom recognition
  - assign to memset, copy to memcpy etc.
- Loop unrolling
  - Different compilers may unroll by different amount
- Jump threading
  - grep -r ‘ switch ’ gcc/libstdc++-v3/* | wc # > 30
  - grep -r ‘ switch ’ llvm-project/libcxx/* | wc # > 60

Additional Reading

- Performance analysis and optimization of C++ standard libraries [Kumar and Pop] https://cppnow2017.sched.com/event/A8J7

References

- https://gcc.gnu.org/
- https://github.com/llvm/llvm-project
- https://hiraditya.github.io/blog/2021/01/11/presentations
- https://cppnow2017.sched.com/event/A8J7

Performance improvement opportunities in the open source C++ standard libraries

Aditya Kumar
Software Engineer, Facebook

linkedin.com/in/hiraditya
twitter.com/_hiraditya_