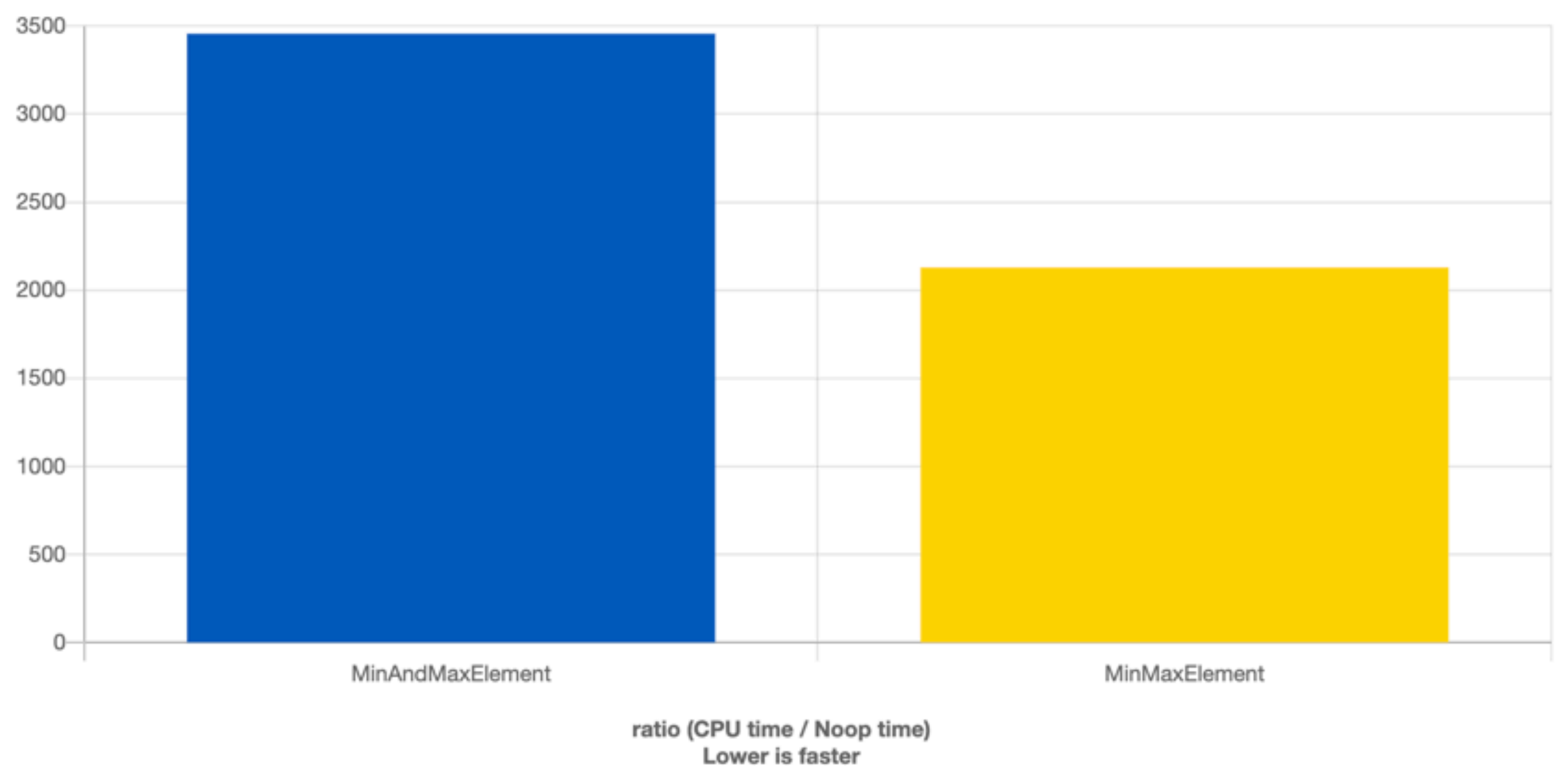
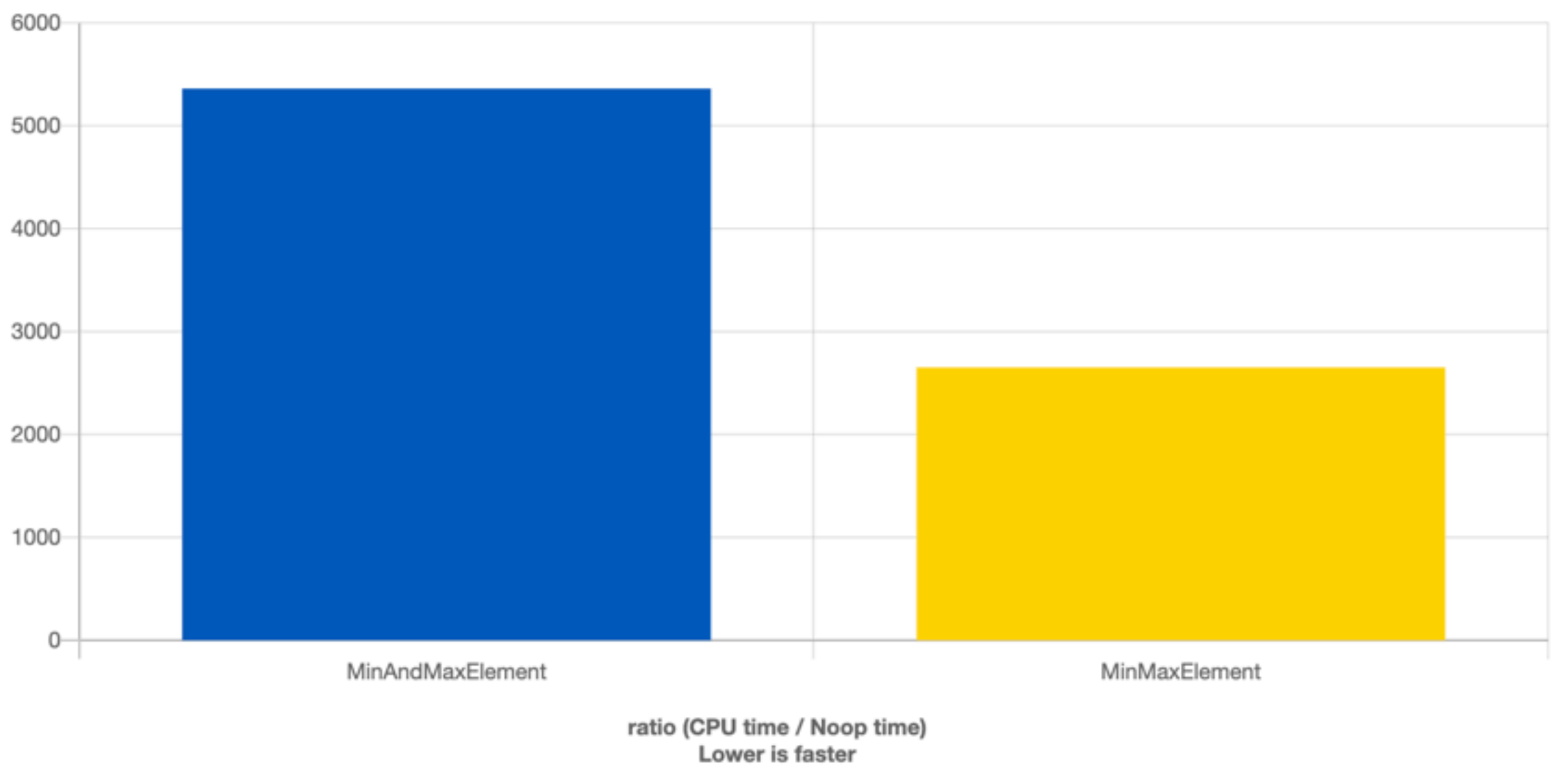
```cpp
#include <algorithm>

const auto v = { 566, … };

static void MinAndMaxElement(benchmark::State& state)
{
  for (auto _ : state)
  {
    const auto min = std::min_element (begin(v), end(v));
    const auto max = std::max_element (begin(v), end(v));

    // Make sure the variable is not optimized away by compiler
    benchmark::DoNotOptimize(min);
    benchmark::DoNotOptimize(max);
  }
}
// Register the function as a benchmark
BENCHMARK(MinAndMaxElement);

static void MinMaxElement(benchmark::State& state)
{
  for (auto _ : state)
  {
    const auto [min, max] = std::minmax_element (begin(v), end(v));

    // Make sure the variable is not optimized away by compiler
    benchmark::DoNotOptimize(min);
    benchmark::DoNotOptimize(max);
  }
}
// Register the function as a benchmark
BENCHMARK(MinMaxElement);
```

49

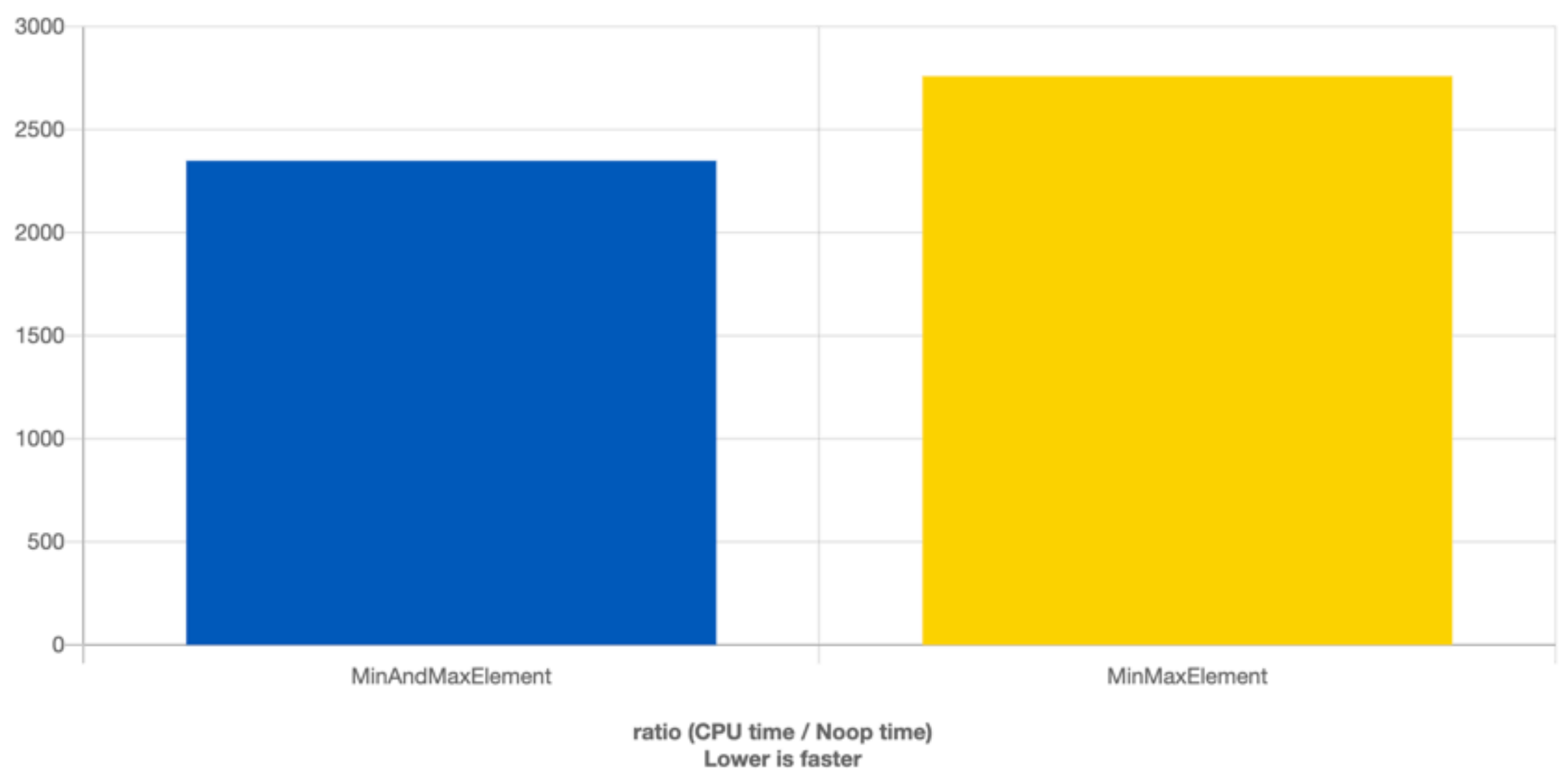

3500
3000
2500
2000
1500
1000
500
0
MinAndMaxElement
MinMaxElement
ratio (CPU time / Noop time)
Lower is faster

GCC 12.2/libstdc++

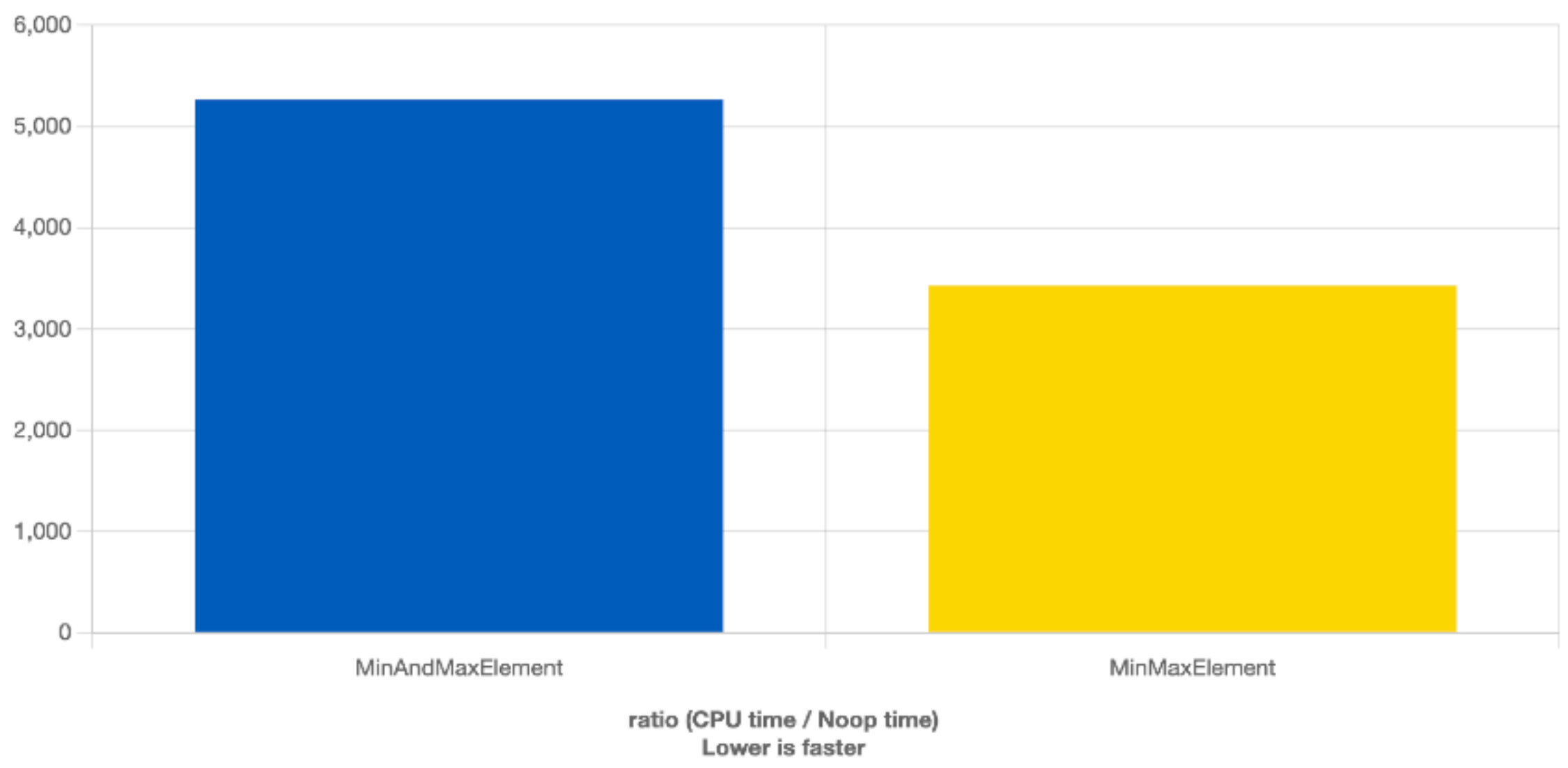

6000
5000
4000
3000
2000
1000
0
MinAndMaxElement
MinMaxElement
ratio (CPU time / Noop time)
Lower is faster

Clang 15/libstdc++

https://quick-bench.com/q/QnasARJ8vPBHjkdkCUpxW-4m5f0


3000
2500
2000
1500
1000
500
0
MinAndMaxElement
MinMaxElement
ratio (CPU time / Noop time)
Lower is faster

Clang 14/libc++


6,000
5,000
4,000
3,000
2,000
1,000
0
MinAndMaxElement
MinMaxElement
ratio (CPU time / Noop time)
Lower is faster

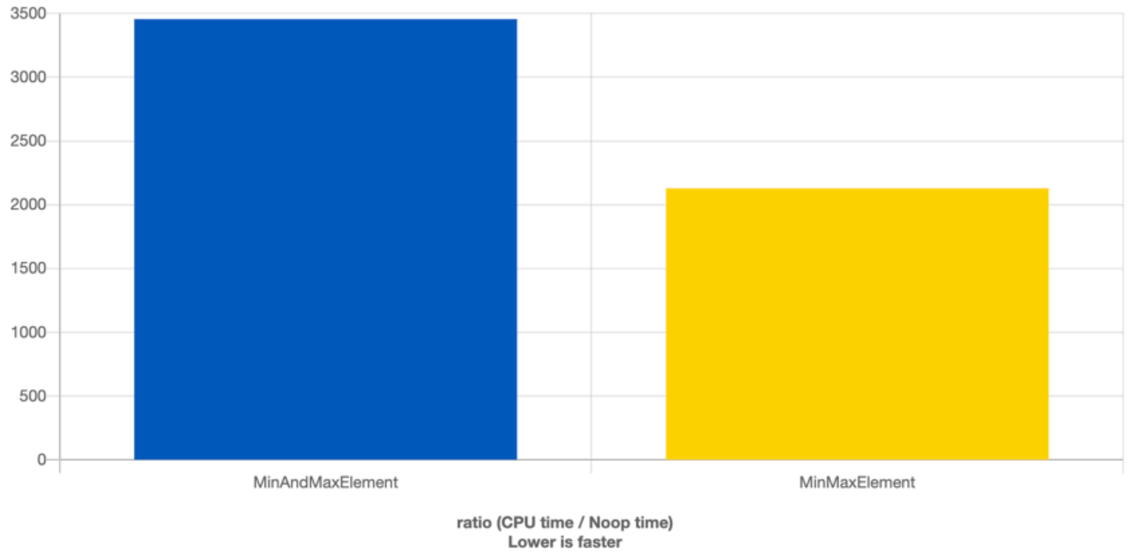
Clang 17/libc++

```cpp
#include <algorithm>

const auto v = { 566, … };
```

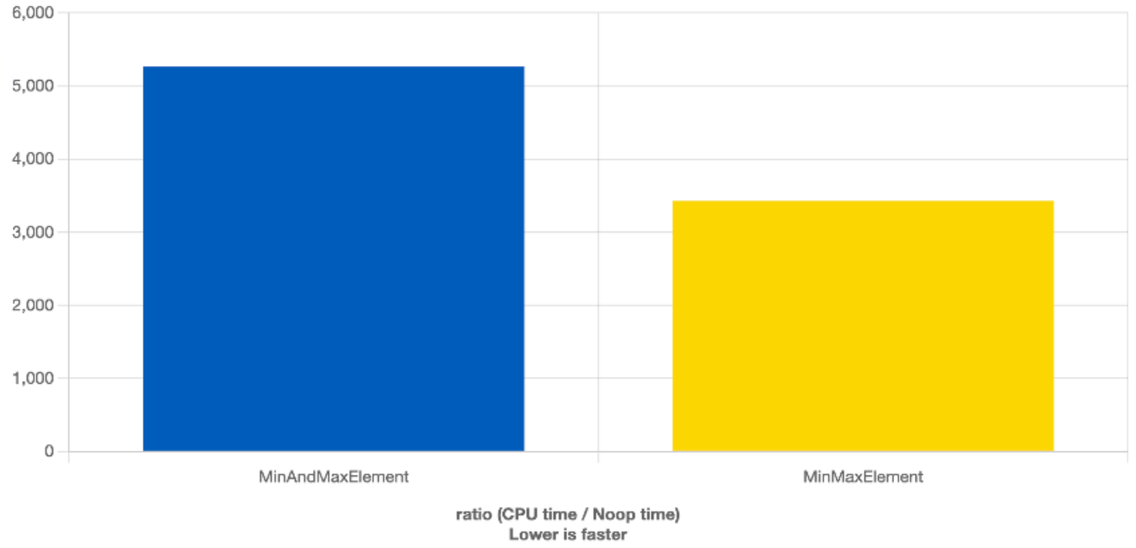
```cpp
const auto min = std::min_element (begin(v), end(v));
const auto max = std::max_element (begin(v), end(v));
```

```cpp
const auto [min, max] = std::minmax_element (begin(v), end(v));
```

GCC 12.2/libstdc++



Clang 17/libc++



https://quick-bench.com/q/QnasARJ8vPBHjkdkCUpxW-4m5f0

Techniques for Optimisation

3. Reducing algorithmic complexity