


```
#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);
```

```
    }
```

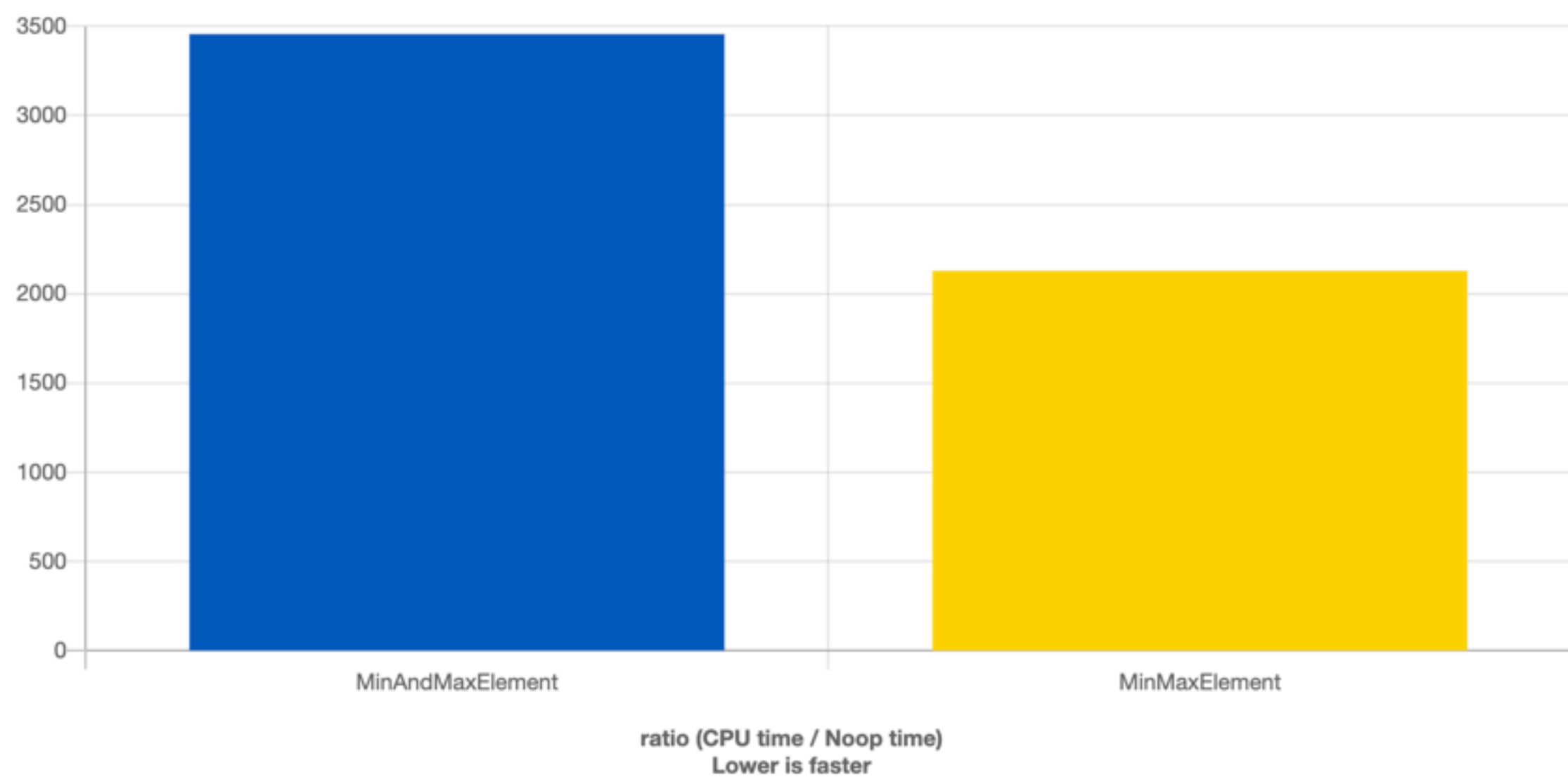
```
}
```

```
// Register the function as a benchmark
```

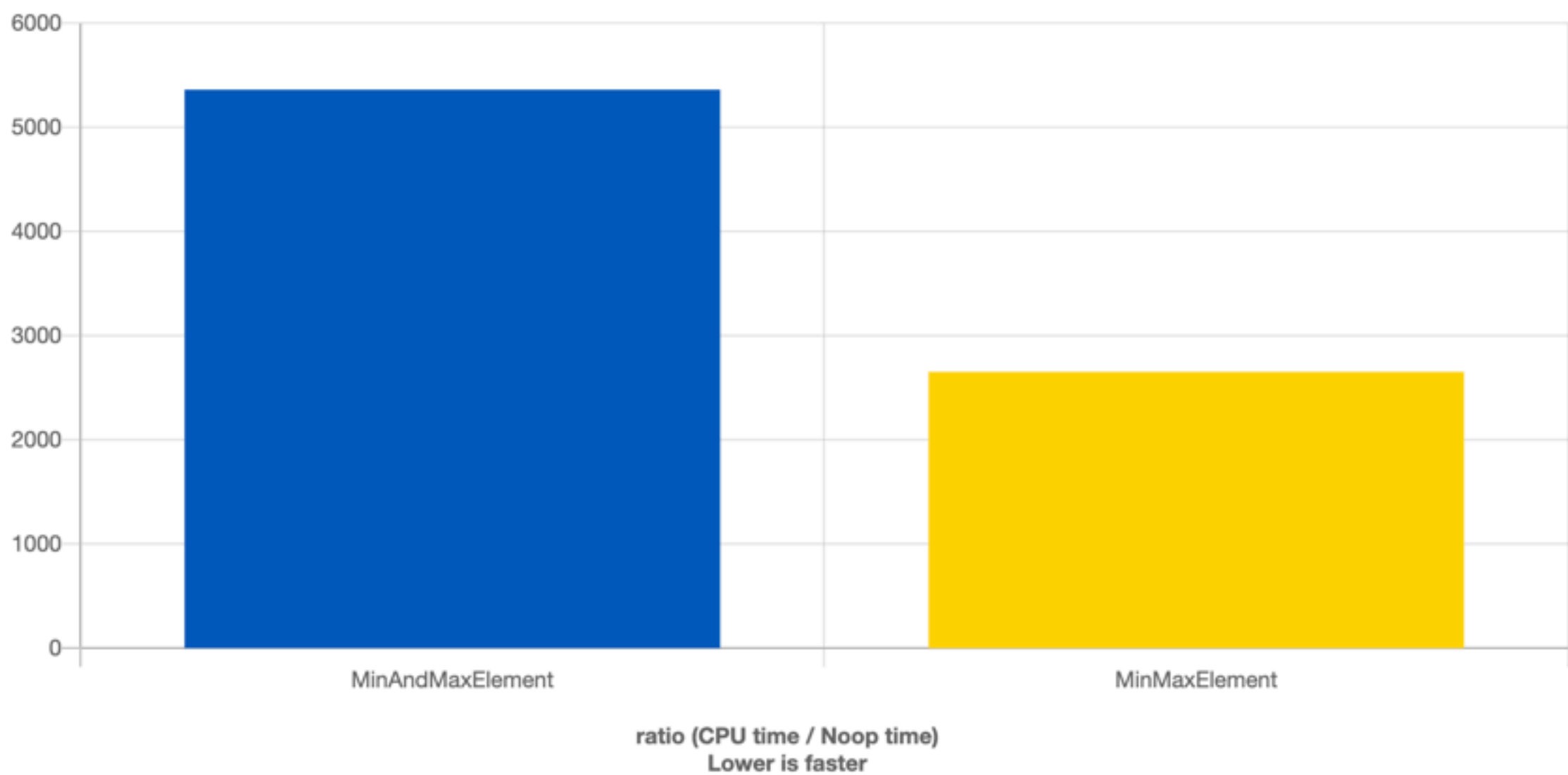
```
BENCHMARK(MinMaxElement);
```


4

9

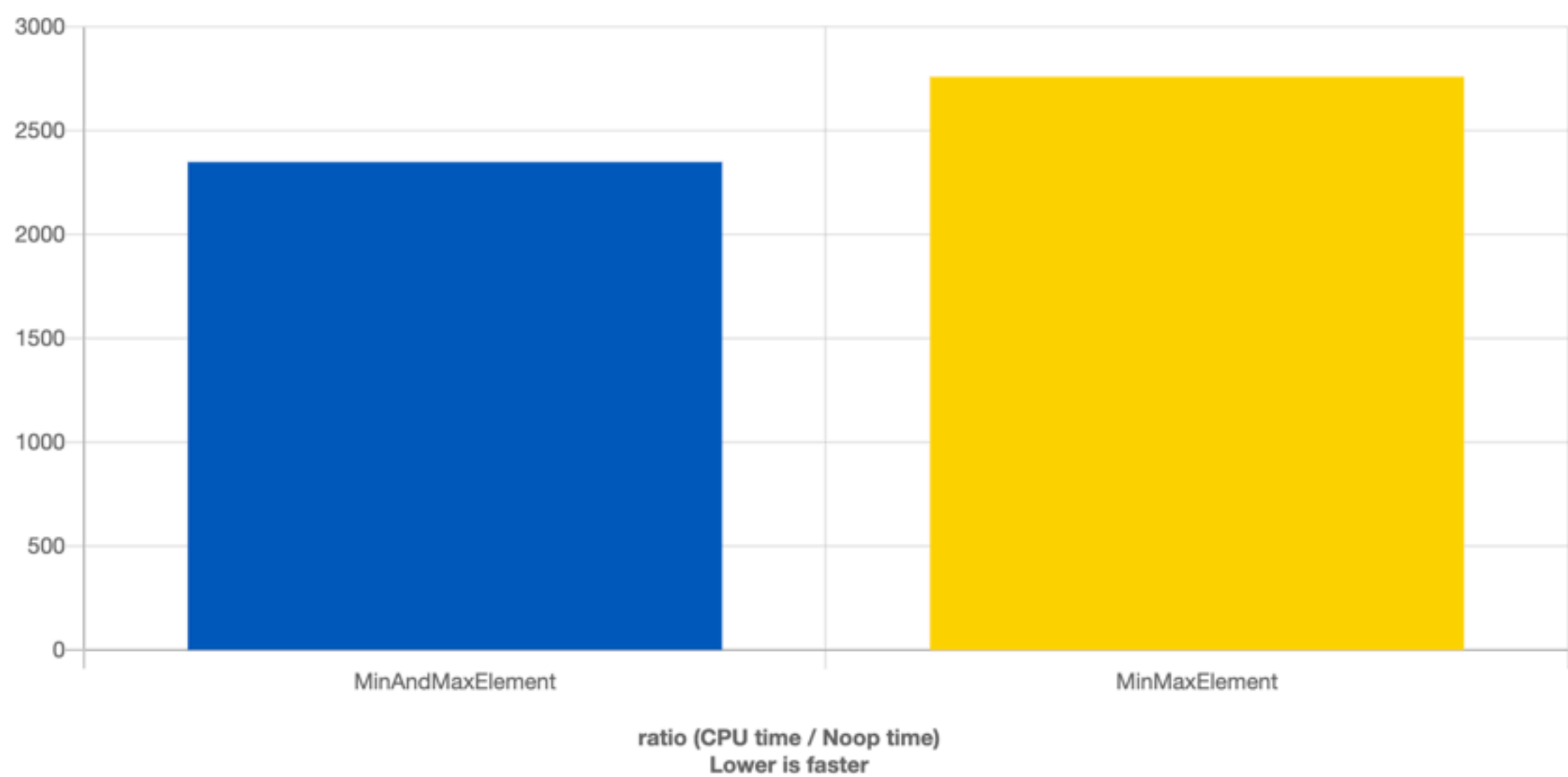


Google12.22/libstdc++



clang 15/libstdc++

<https://quick-bench.com/q/QnAsARJ8vPBHjkdkCUpxW-4m5f0>



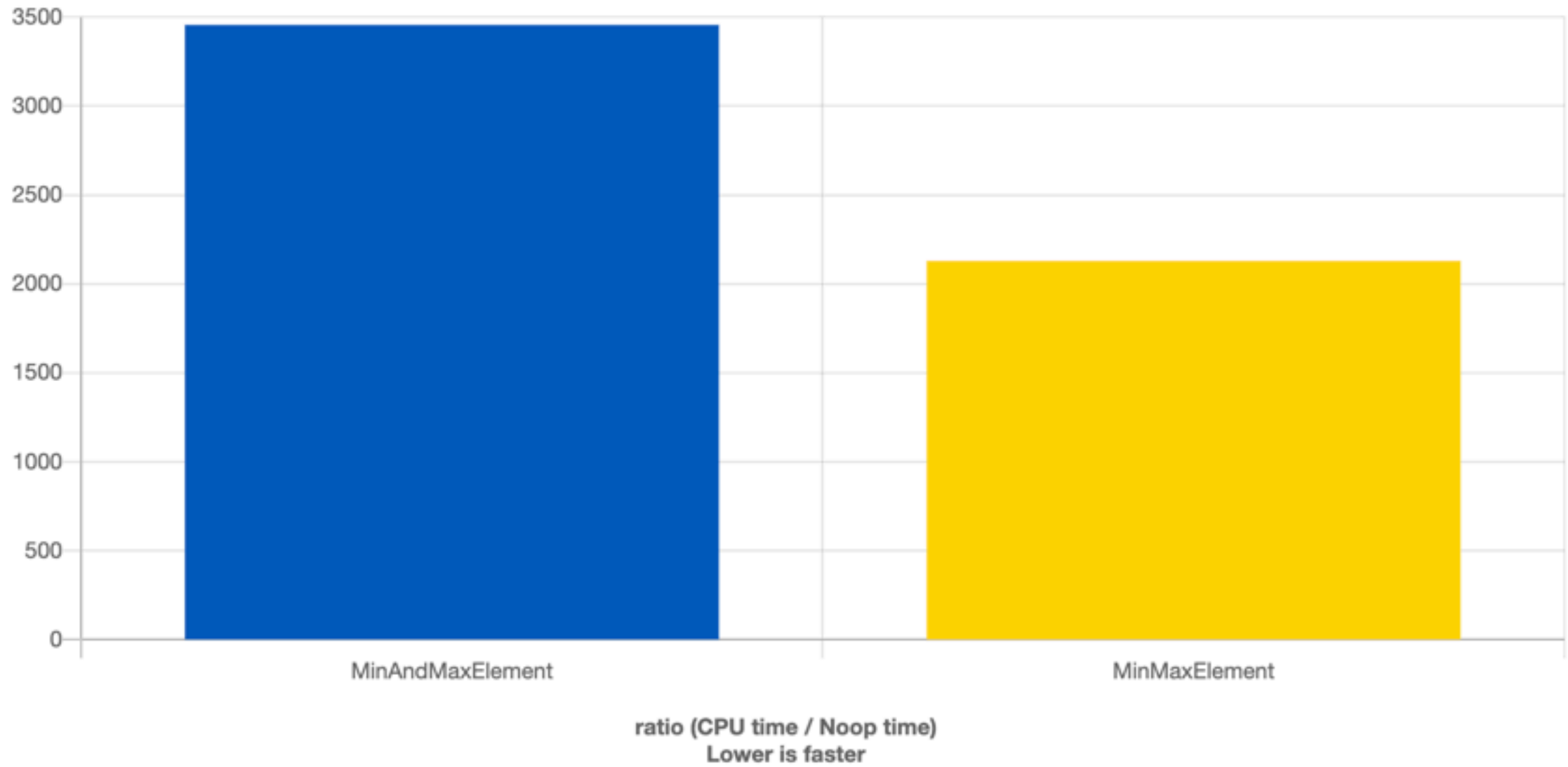
clang 14/libc++

```
#include <algorithm>

const auto v = { 566, ... };
```

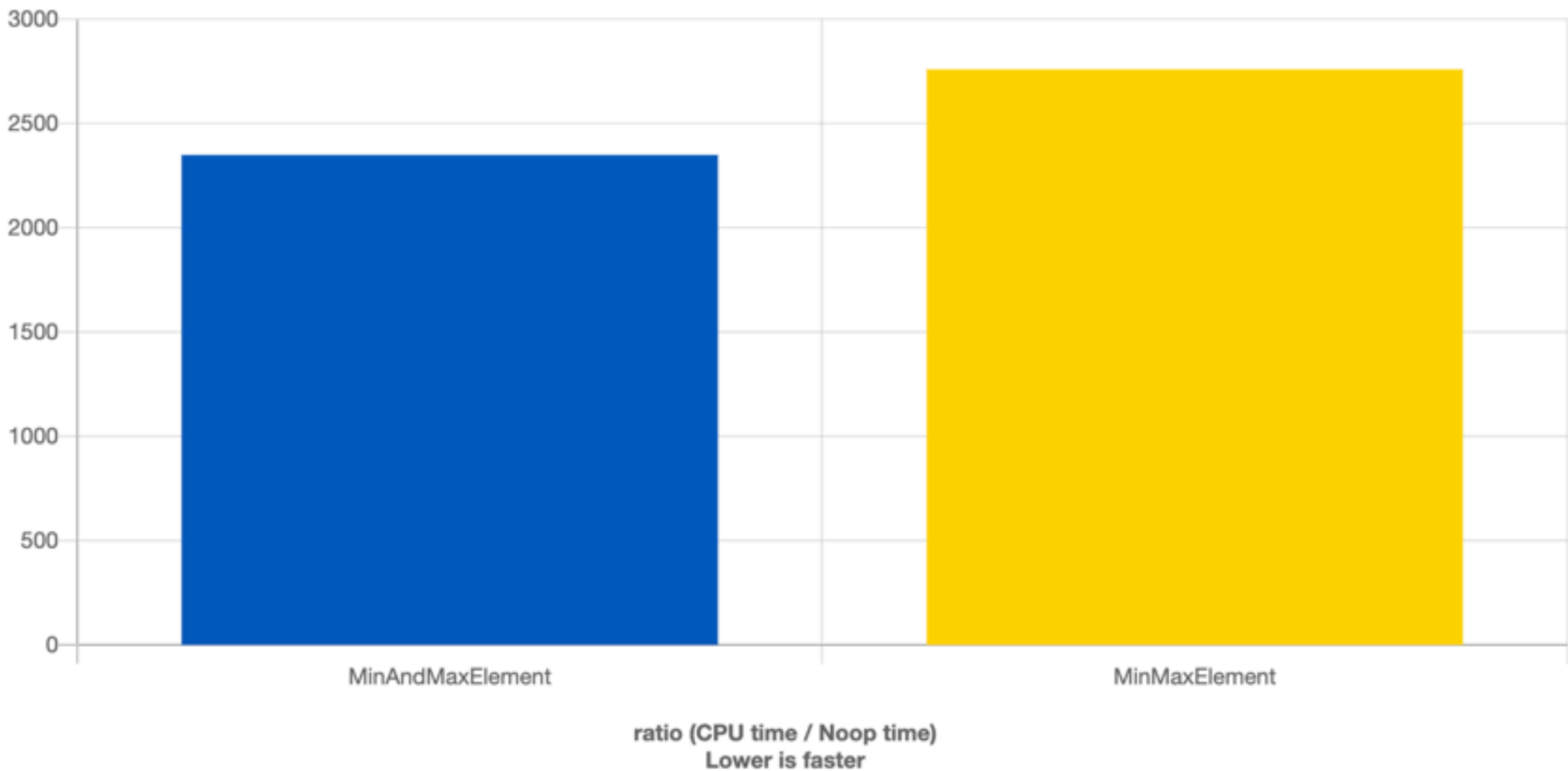
```
const auto min = std::min_element (begin(v), end(v));
const auto max = std::max_element (begin(v), end(v));
```

GCC 12.2/libstdc++



Clang 14/libc++

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



Techniques for Optimisation

3. Reducing algorithmic complexity