Without a way to properly express lifetimes (in terms of borrows/relocations/drops) we don't get the same level of safety and performance

Back to C++

```
void entry_point (std::shared_ptr<synchronized_value<std::string>> sync_s, int tid)
{
    apply ([tid] (auto& s) {
        s.append ("🔥");
        std::println ("{} {}", s, tid);
        return s;
    },
    *sync_s);
}

int main()
{
    auto s = std::make_shared<synchronized_value<std::string>> ("Hello threads");

    std::vector<safe_thread> threads { };

    const int num_threads = 15;

    for (int i : std::views::iota (0, num_threads))
        threads.push_back (safe_thread (entry_point, auto (s), auto (i)));
}
```