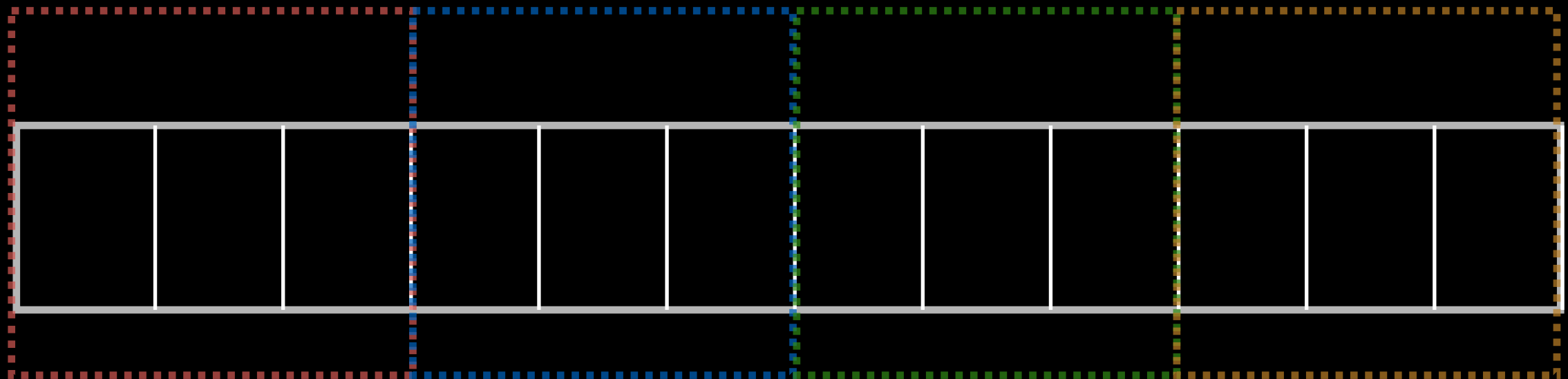


thread1

thread2

thread3

thread4



```
template<typename I>
long async_accumulate(I begin, I end)
{
    auto len = std::distance(begin, end);

    if (len == 0)
        return 0;

    const unsigned num_threads = compute_number_of_threads(len);
    std::vector<std::future<long>> res(num_threads-1);
    const unsigned block_size = len / num_threads;

    auto f = [](I b, I e, long init) { return std::accumulate(b,e,init); };

    auto start_block = begin;
    for(unsigned i =0; i < num_threads-1; ++i)
    {
        auto end_block = start_block;
        std::advance(end_block, block_size);
        res[i] = std::async(f, start_block, end_block, 0);
        start_block = end_block;
    }

    long sum = f(start_block, end, 0);
    for(auto& f : res )
        sum += f.get();

    return sum;
}
```