```cpp
template<queue_end queue_end, typename queue_type>
struct queue_thread
{
    queue_thread(queue_type &queue_, std::latch &start_latch_, size_t num_iterations)
        : queue(queue_), start_latch(start_latch_), iters(num_iterations)
    {}

    void run_async()
    {
        thread = std::thread([this]
        {
            start_latch.arrive_and_wait();
            queue_result res;

            stopwatch sw;
            time_statistics stats;
            int sum = 0;

            if constexpr (queue_end == queue_end::producer)
            {
                //...
            }
            else
            {
                //...
            }

            res.duration = sw.get();
            res.stats = stats;

            run_result = res;
        });
    }

    queue_result join()
    {
        assert (thread.joinable());
        thread.join();
        return run_result;
    }

private:
    queue_type& queue;
    std::latch& start_latch;
    const size_t iters;
    std::thread thread;
    queue_result run_result;
};
```

```cpp
if constexpr (queue_end == queue_end::producer)
{
    for (size_t i = 0; i < iters; ++i)
    {












        if (queue.try_push(i))
        {



        }




    }
}
```

```cpp
else
{
    for (size_t i = 0; i < iters; ++i)
    {



        int v;
        if (queue.try_pop (v))
        {








        }









    }
}
```

```cpp
template<queue_end queue_end, typename queue_type>
struct queue_thread
{
    queue_thread(queue_type &queue_, std::latch &start_latch_, size_t num_iterations)
        : queue(queue_), start_latch(start_latch_), iters(num_iterations)
    {}

    void run_async()
    {
        thread = std::thread([this]
        {
            start_latch.arrive_and_wait();


            stopwatch sw;



            if constexpr (queue_end == queue_end::producer)
            {
                //...
            }
            else
            {
                //...
            }

            res.duration = sw.get();


            run_result = res;
        });
    }

    queue_result join()
    {
        assert (thread.joinable());
        thread.join();
        return run_result;
    }

private:
    queue_type& queue;
    std::latch& start_latch;
    const size_t iters;
    std::thread thread;
    queue_result run_result;
};
```

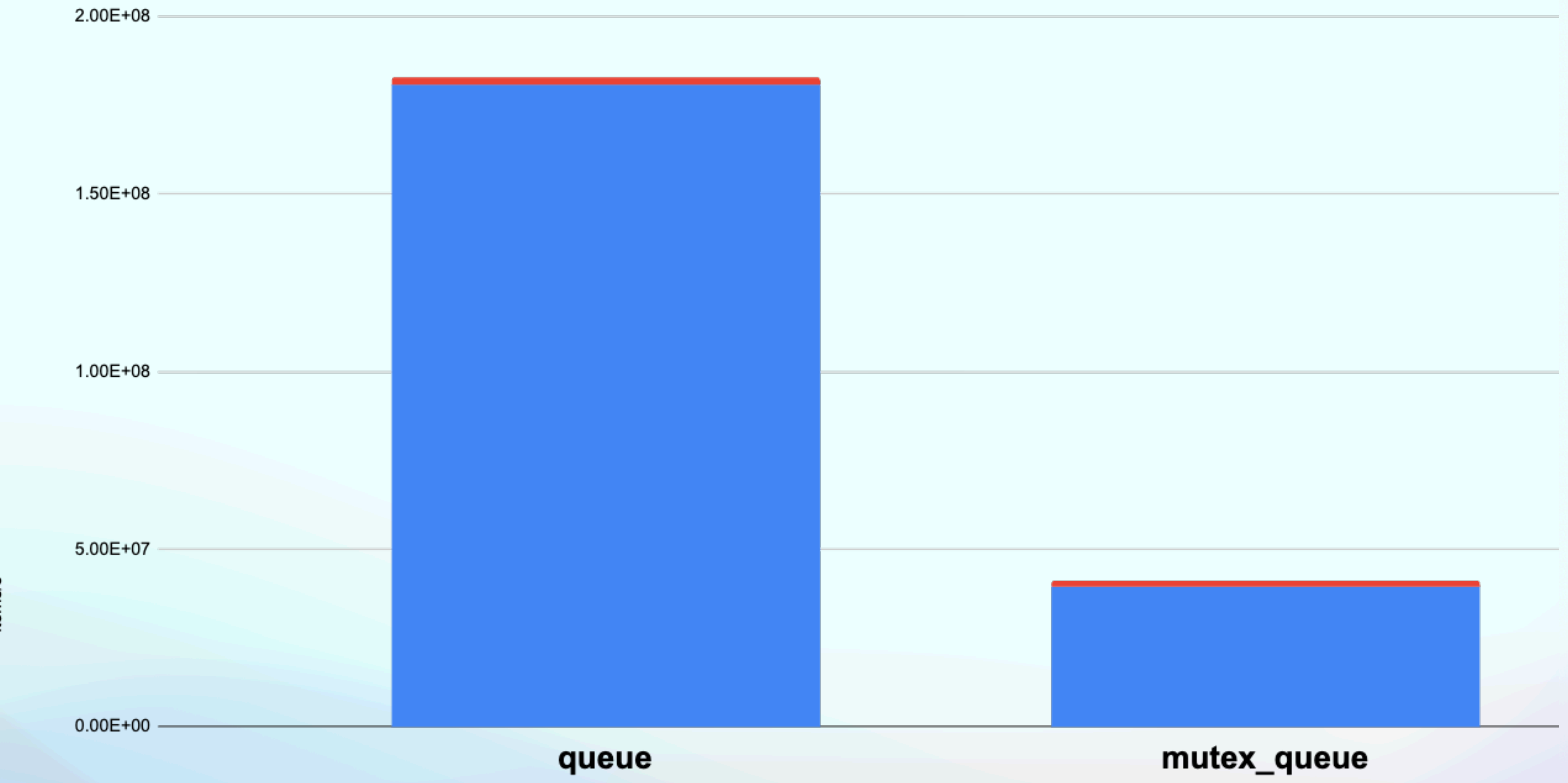
```cpp
if constexpr (queue_end == queue_end::producer)
{
    for (size_t i = 0; i < iters; ++i)
    {









        if (queue.try_push(i))
        {


        }




    }
}
```

```cpp
else
{
    for (size_t i = 0; i < iters; ++i)
    {




        int v;
        if (queue.try_pop (v))
        {




        }









    }
}
```

182,987,000

x4.5

41,400,000


2.00E+08
1.50E+08
1.00E+08
5.00E+07
0.00E+00
queue
mutex_queue