```cpp
inline void time_statistics::add_result
   (double secondsElapsed)
{
    if (numRuns == 0)
    {
        max = secondsElapsed;
        min = secondsElapsed;
    }
    else
    {
        max = std::max (max, secondsElapsed);
        min = std::min (min, secondsElapsed);
    }

    ++numRuns;
    total += secondsElapsed;

    // Seconds
    {
        const double delta = secondsElapsed - mean;
        mean += delta / (double) numRuns;
        const double delta2 = secondsElapsed - mean;
        m2 += delta * delta2;
    }
}
```

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

        int v;
        if (queue.try_pop (v))
        {
            sum += v;
            ++res.num_valid;
            stats.add_result (iter_sw.get());
        }
        else
        {
           ++res.num_failed;
        }
    }
}
```

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

        if (queue.try_push(i))
        {
            sum += i;
            stats.add_result (iter_sw.get());
            ++res.num_valid;
        }
        else
        {
            ++res.num_failed;
        }
    }
}
```

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

        if (queue.try_push(i))
        {
            sum += i;
            stats.add_result (iter_sw.get());
            ++res.num_valid;
        }
        else
        {
            ++res.num_failed;
        }
    }
}
```

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

        int v;
        if (queue.try_pop (v))
        {
            sum += v;
            ++res.num_valid;
            stats.add_result (iter_sw.get());
        }
        else
        {
           ++res.num_failed;
        }
    }
}
```

```cpp
inline void time_statistics::add_result
   (double secondsElapsed)
{
    if (numRuns == 0)
    {
        max = secondsElapsed;
        min = secondsElapsed;
    }
    else
    {
        max = std::max (max, secondsElapsed);
        min = std::min (min, secondsElapsed);
    }

    ++numRuns;
    total += secondsElapsed;

    // Seconds
    {
        const double delta = secondsElapsed - mean;
        mean += delta / (double) numRuns;
        const double delta2 = secondsElapsed - mean;
        m2 += delta * delta2;
    }
}
```

5.00E+08
4.00E+08
3.00E+08
2.00E+08
1.00E+08
0.00E+00
items/s
mutex_queue
realtime_mutex_queue
drow_spsc_v5
drow_spsc_v6
drow_spsc_v7
drow_spsc_v7_1
drow_spsc_v7_2
drow_spsc_v8
choc_spsc
rigtorp_spscqueue
farbot_spscqueue
mc_readerwriterqueue
boost_lockfree_spsc_queue