Report:

Experimental setup

- Processor: Intel® Core™ i3-5005U × 4
- Memory: 8GB
- OS: Manjaro Linux
- OS Type: 64-bit
- Kernel: 5.15.78-1-MANJARO

Additional information

- The data present in the tables is a mean of 100 simlations realized under similar conditions.
- The accuracy is evaluated by the frequency of the final counter holding the right value.
- The time is in seconds.

Race condition thread

Accuracy

Size Threads	1	2	4	8	16	32	64
100	100%	100%	100%	100%	100%	100%	100%
1,000	100%	100%	98%	100%	100%	100%	100%
10,000	100%	6%	83%	86%	97%	97%	95%
100,000	100%	1%	0%	0%	0%	1%	48%
1,000,000	100%	0%	0%	0%	0%	0%	0%
10,000,000	100%	0%	0%	0%	0%	0%	0%
100,000,000	100%	0%	0%	0%	0%	0%	0%

Running Time

Size Threads	1	2	4	8	16	32	64
100	0.000036	0.000057	0.000104	0.000273	0.000602	0.001179	0.002406
1,000	0.000041	0.000059	0.000112	0.000275	0.000602	0.001172	0.002411
10,000	0.000123	0.000102	0.000139	0.000312	0.000634	0.001240	0.002499
100,000	0.000703	0.000795	0.000721	0.000842	0.000912	0.001688	0.003898
1,000,000	0.006326	0.007211	0.006577	0.006631	0.006672	0.006902	0.007537
10,000,000	0.062638	0.070826	0.064598	0.064805	0.064828	0.064957	0.065386
100,000,000	0.614216	0.708994	0.638655	0.643387	0.644816	0.643185	0.645343

Analysis

- The accuracy is 100% when a unique thread is created, because race conditions only occur with multiple threads.
- The accuracy is only acceptable when the number of elements is small (100 or 1000) since the less the global variable is modified, the less race conditions happen.
- Adding an escessive amount of threads for small arrays significantly increases the running time.

Mutex thread

Running Time

Size Threads	1	2	4	8	16	32	64
100	0.000035	0.000061	0.000111	0.000280	0.000588	0.001180	0.002672
1,000	0.000044	0.000062	0.000119	0.000285	0.000581	0.001192	0.002430
10,000	0.000144	0.000252	0.000305	0.000318	0.000627	0.001275	0.002532
100,000	0.001058	0.002747	0.003992	0.003507	0.003408	0.003177	0.002775
1,000,000	0.009667	0.026205	0.039076	0.038037	0.037548	0.037736	0.037014
10,000,000	0.095872	0.249729	0.387783	0.384026	0.387617	0.388363	0.389898
100,000,000	0.959138	2.525131	3.882493	3.943893	3.919308	3.931774	3.929282

Analysis

- The running time is significantly higher than for the race condition thread, especially when the array and number of thread increase, this is because threads have to wait for each other before accessing the global counter.
- The running time doesn't seem to increase with the number of threads, this is probably due to false sharing of the `count` variable in caches.

Private count thread

Running Time

Size Threads	1	2	4	8	16	32	64
100	0.000034	0.000057	0.000115	0.000281	0.000581	0.001172	0.002408
1,000	0.000039	0.000060	0.000123	0.000272	0.000599	0.001190	0.002408
10,000	0.000101	0.000121	0.000149	0.000303	0.000643	0.001226	0.002471
100,000	0.000731	0.000783	0.000730	0.000818	0.000847	0.001400	0.002627
1,000,000	0.006650	0.007440	0.006838	0.006673	0.004944	0.004606	0.005469
10,000,000	0.065404	0.072937	0.065336	0.066021	0.046670	0.037182	0.031560
100,000,000	0.653064	0.727045	0.646689	0.650325	0.466532	0.366785	0.301947

Analysis

- The running is better than for the race condition threads and the mutex threads, especially for big numbers of threads.
- The running time could still be improved by reducing the false sharing of the private counters. Since they are located in contiguous arrays, when their values are updated, the caches of the other cores containing the array are flushed.

Private count thread with cache padding

Running Time

Size Threads	1	2	4	8	16	32	64
100	0.000046	0.000080	0.000132	0.000339	0.000689	0.001428	0.002944
1,000	0.000049	0.000080	0.000132	0.000343	0.000689	0.001441	0.003150
10,000	0.000104	0.000115	0.000174	0.000354	0.000699	0.001469	0.002986
100,000	0.000747	0.000433	0.000389	0.000648	0.000865	0.001502	0.003394
1,000,000	0.006662	0.003379	0.002599	0.003204	0.003247	0.004408	0.005248
10,000,000	0.064872	0.033071	0.024230	0.025234	0.025228	0.025540	0.026997
100,000,000	0.647457	0.327024	0.239786	0.244141	0.241310	0.241042	0.244081

Analysis

- The running is better than for all of the other threads, this is because false sharing is avoided by filling the caches only with the needed counter, and with trash values instead of other significant values that would cause the cache to be flushed.
- The running time significantly increases when the number of threads increase. But this stops when the number of threads gets over 4, because my CPU has 4 cores, and the running time increases after that.
- The running time could probably still be improved by having a private counter instead of a public one, and incrementing the global counter by the value in the private counter when the thread is done, using a mutex.