Report

Thread safe:

Relation between number of threads and performance:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | No of Threads | Time(in ms) | | 1 | 216568 | | 2 | 217413 | | 3 | 382767 | | 4 | 515249 | | 5 | 516626 | | 6 | 501114 | | 7 | 499215 | | 8 | 475956 | | 9 | 477129 | | 10 | 508732 | |  |

From the figure, one can see that the performance of the synchronized counter scales poorly. Whereas a single thread can complete the million counter updates in a tiny amount of time, having two threads each update the counter one million times concurrently leads to a massive slowdown. It only gets worse with more threads.

Ideally, one would like to see the threads complete just as quickly on multiple processors as the single thread does on one. Achieving this end is called perfect scaling; even though more work is done, it is done in parallel, and hence the time taken to complete the task is not increased. Due to locks, the time increases.

Relation between number of threads and correctness:

|  |  |
| --- | --- |
| No of threads | Value |
| 1 | 1000000000 |
| 2 | 1000000000 |
| 3 | 1000000000 |
| 4 | 1000000000 |
| 5 | 1000000000 |
| 6 | 1000000000 |
| 7 | 1000000000 |
| 8 | 1000000000 |
| 9 | 1000000000 |
| 10 | 1000000000 |

From the figure, one can see that the counter is perfectly correct. It always reaches 1 billion if we use 1, 2 or 10 threads. Only one thread enters the critical region.

Non-thread safe:

Relation between number of threads and performance:

|  |  |
| --- | --- |
| No of Threads | Time(in ms) |
| 1 | 4251 |
| 2 | 12474 |
| 3 | 15904 |
| 4 | 42371 |
| 5 | 46290 |
| 6 | 51278 |
| 7 | 57849 |
| 8 | 61906 |
| 9 | 68993 |
| 10 | 69821 |

From the figure, one can see that the performance of the counter. Whereas a single thread can complete the million counter updates in a tiny amount of time, having two threads each update the counter one million times concurrently leads to a massive slowdown. It only gets worse with more threads.

Relation between number of threads and correctness:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| |  |  | | --- | --- | | No of Threads | Value | | 1 | 1000000000 | | 2 | 1000000000 | | 3 | 1000000001 | | 4 | 1000000001 | | 5 | 1000000001 | | 6 | 1000000002 | | 7 | 1000000000 | | 8 | 1000000002 | | 9 | 1000000000 | | 10 | 1000000002 | |
|  |

From the figure, one can see that the counter is not perfectly correct. It has varying values as multiple threads can enter critical region causing multiple updates. That why value even reaches 1billion + 1 too.