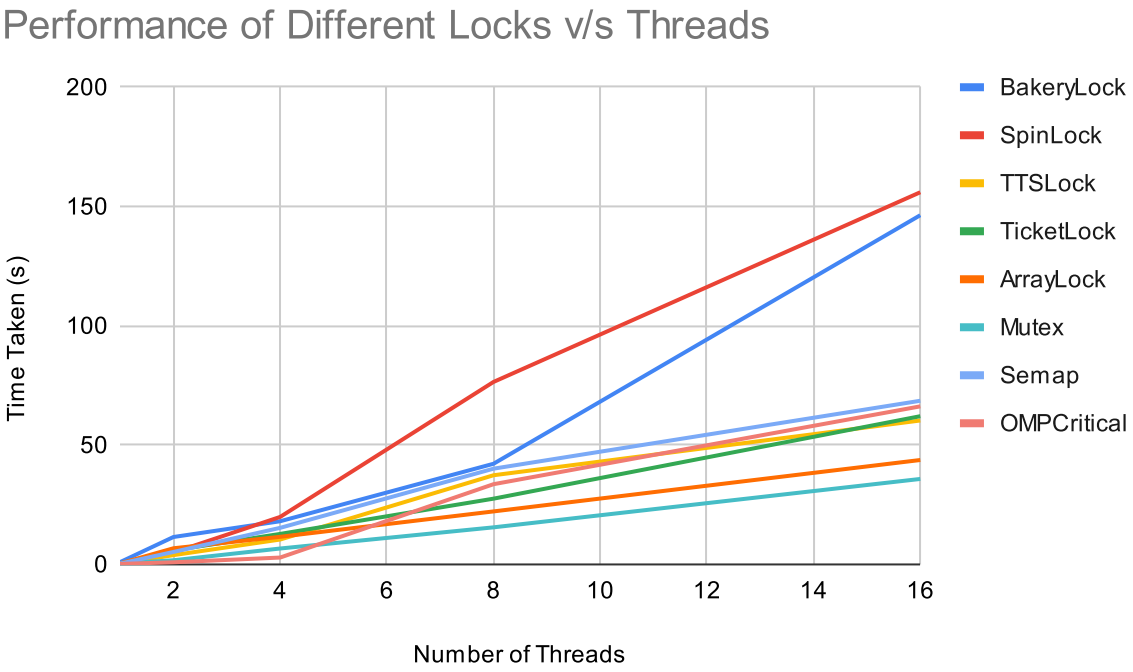
CS433
Design Assignment - 2

190772
Sarthak Rout
sarthakr@iitk.ac.in
Group - 21

April 2, 2022

Machine Specification: Intel x86 CPU gpu01.cc.iitk.ac.in with 16 threads and 64K L1 cache size.
The full machine specification is available at the end of the report.

LOCK DESIGN

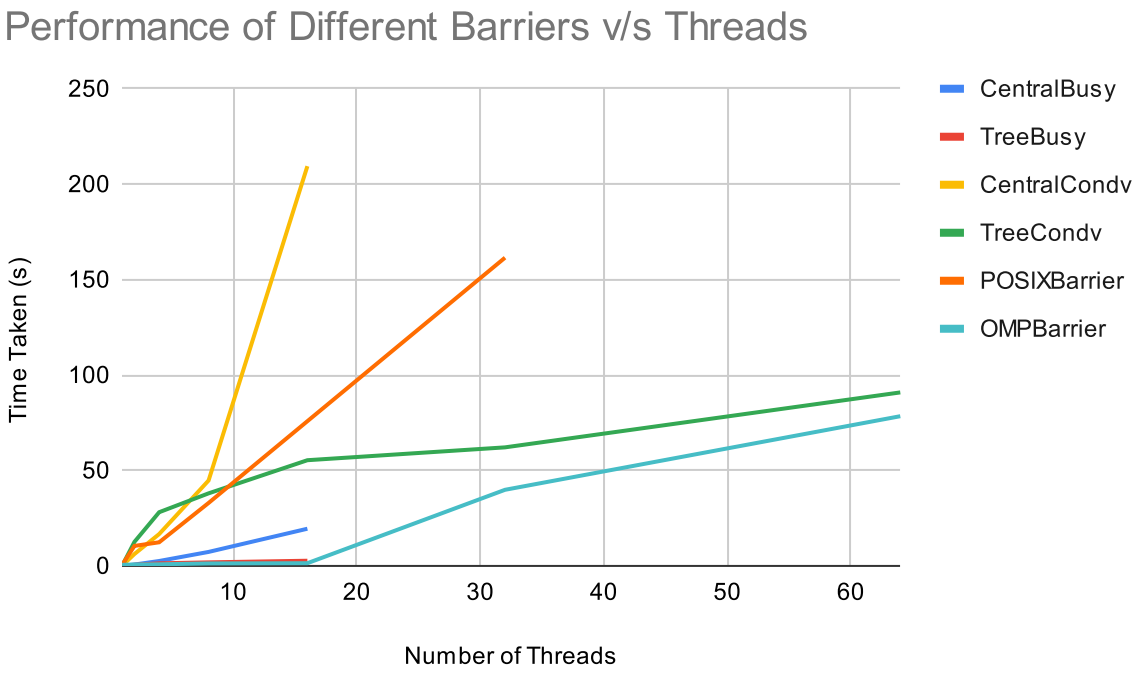


T	BakeryLock	SpinLock	TTSLock	TicketLock	ArrayLock	Mutex	Semap	OMPCritical	Best
1	0.93485	0.277	0.2634	0.275	0.394	0.303	0.307	0.182	OMPCritical
2	11.459	4.96	3.856	5.863	6.736	1.716	5.194	0.8607	OMPCritical
4	18.064	19.874	10.394	12.753	11.542	6.633	15.295	2.846	OMPCritical
8	42.1585	76.463	37.35	27.532	22.187	15.51	40.102	33.597	Mutex
16	146.361	155.979	60.302	62.103	43.735	35.794	68.573	66.2301	Mutex

Observed Trends and Explanations:

- We observe that implementations using OMP Critical directive and POSIX Mutex lock outperform every other kind of locks, as they are well optimized for production-grade environment.
  We note that POSIX Mutex performs better for larger number of threads.
- All locks show significant amount of jump with multiple number of threads when compared to that with single thread, due to lock contention (threads having to wait for acquiring the lock).
- Some locks such as SpinLock and TTSLock show superlinear increase in time taken with increase in number of threads initially, which shows that the algorithms are very sensitive to number of cores trying to acquire the lock at the same time (starvation)/ due to false sharing.
  To avoid false sharing, integer arrays were converted to higher dimensional arrays of each array element corresponding to 16 integers (64 byte) blocks.
- Test-and-test-and-set lock, Ticket Lock and Binary Semaphore have similar performance with that of ArrayLock marginally better with larger number of threads.

BARRIER DESIGN



T	CentralBusy	TreeBusy	CentralCondv	TreeCondv	POSIXBarrier	OMPBarrier	Best
1	0.0397	0.0177	0.0663	0.015	0.505	0.523	TreeCondv
2	0.674	0.5286	6.189	12.56	10.432	0.698	TreeBusy
4	2.53	1.3087	16.6526	28.09	12.34	0.807	OMPBarrier
8	7.282	1.8195	44.7079	37.98	32.946	1.2448	OMPBarrier
16	19.441	2.669	209.42	55.34	75.79	1.38187	OMPBarrier
32	-	-	-	62.08	161.469	39.8375	*OMPBarrier
64	-	-	-	90.867	-	78.408	*OMPBarrier

* $\equiv$ Some algorithms didn't terminate for 32/64 threads.

Observed Trends:

- Initially, Tree Barriers performed better than any other barrier algorithm, but later OMP Barrier fared better, due to better optimization/exploiting more features from the processor.
- Even with higher number of threads, Tree Barrier with condition variables was runner up to OMP Barrier in performance. We can clearly observe the logarithmic performance in the graph for it. For the one implemented with busy wait loops, it can be seen from the table that it also shows logarithmic increase with increase in number of threads.
- For Central sense reversal barrier with busy wait and condition variable and tree barrier with busy wait, it was not possible to get any sample with 32 threads as they took a large amount of time to complete. This was probably caused by a large number of context switches increasing the time taken significantly due to starvations. As the algorithms terminated for smaller value of N, the reason couldn't be a race condition or deadlock.

Machine Specification

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                16
On-line CPU(s) list:   0-15
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             2
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 26
Model name:            Intel(R) Xeon(R) CPU X5570  @ 2.93GHz
Stepping:              5
CPU MHz:               2927.000
CPU max MHz:           2927.0000
CPU min MHz:           1596.0000
BogoMIPS:              5866.89
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              8192K
NUMA node0 CPU(s):     0-15
```