CS5005
Parallel Programming
Assignment - 2

(Performance profile of synchronization constructs)

Himanshu Rai - 111601032
Date - 23/02/2019

Problem Statement

We discussed various synchronization primitives such as busy-waiting, mutexes, conditional variables, barriers, and read-write locks. You are required to design experiments (or test cases), execute them and measure the performance of each of these synchronization constructs.

Processor Specs

```
himanshu@Lenovo-ideapad-320-15IKB:~$ lscpu
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  4
On-line CPU(s) list:     0-3
Thread(s) per core:      2
Core(s) per socket:      2
Socket(s):               1
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   142
Model name:              Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
Stepping:                9
CPU MHz:                 700.201
CPU max MHz:             3100.0000
CPU min MHz:             400.0000
BogoMIPS:                5424.00
Virtualization:          VT-x
L1d cache:               32K
L1i cache:               32K
L2 cache:                256K
L3 cache:                3072K
NUMA node0 CPU(s):       0-3
```

Experiment - 1 (array_sum)

The first experiment is of calculating sum of numbers in an array. Each thread computes a partial sum of the array. These sums are combined into a global sum using critical section. The synchronisation constructs compared using this experiment are

- Busy Waiting
- Mutexes
- Semaphores
- Read Write Locks

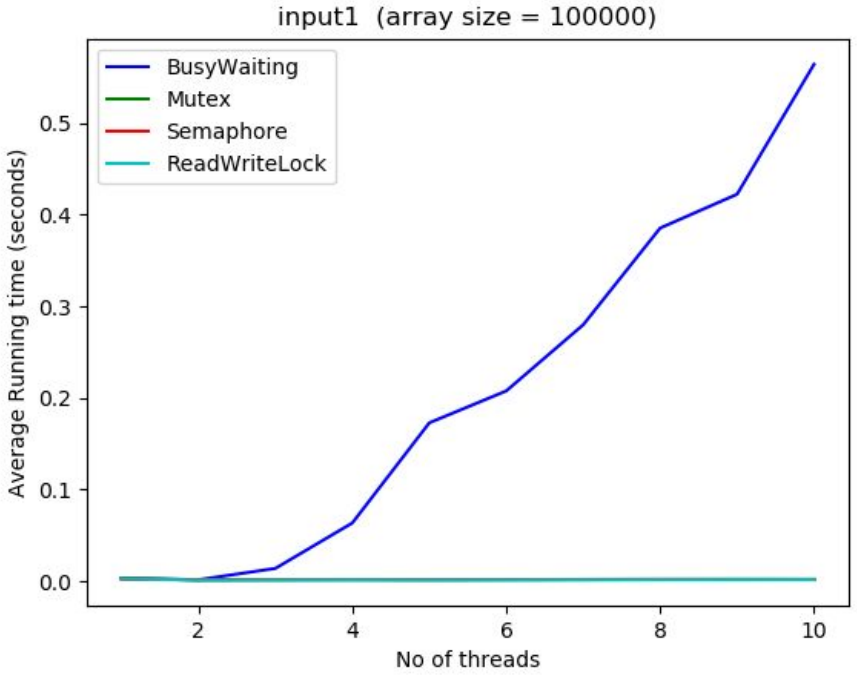
The synchronisation constructs are compared for different array sizes and also for different number of threads.

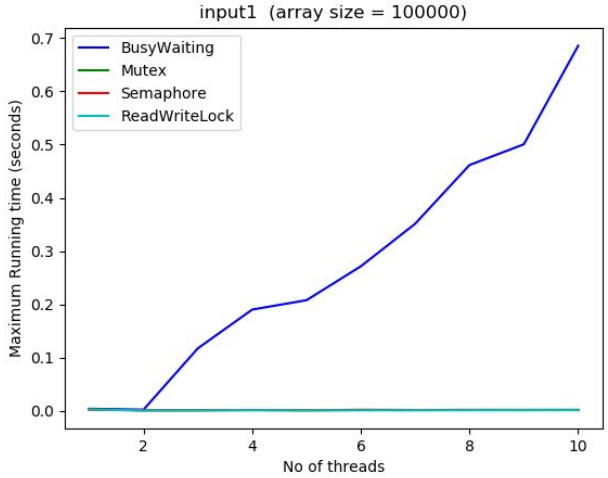
The functions running under the threads for the four constructs are kept similar apart except for the codes specific to them. This ensures uniformity and no extra overhead for any of the three constructs.

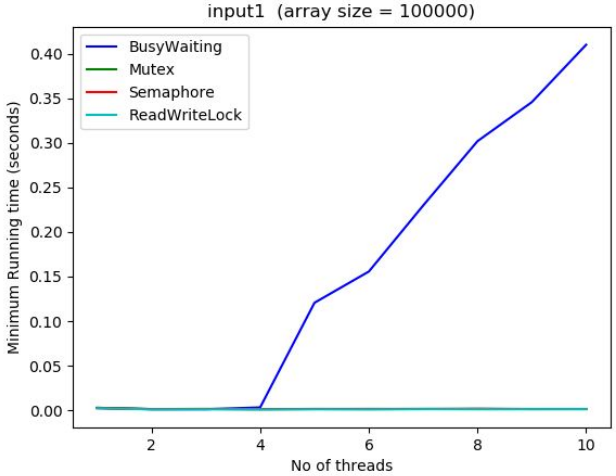
Also the partial sums are calculated five times to minimize the effect of other overheads in the relative performance when considering runtime of different number of threads.

Also since the running time for a program depends on multiple factors an average of running time over various runs is taken.

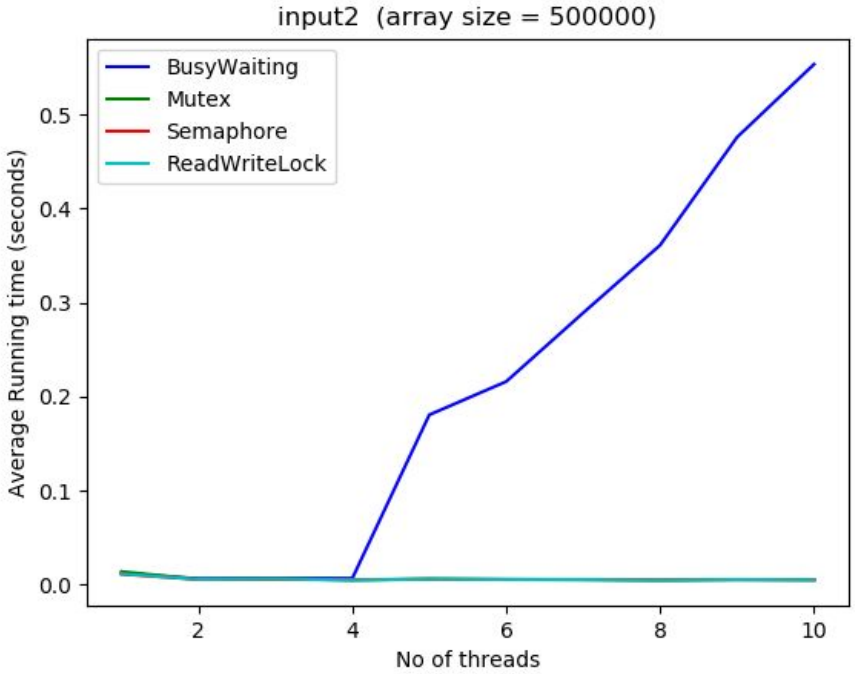
Observations (array_sum)

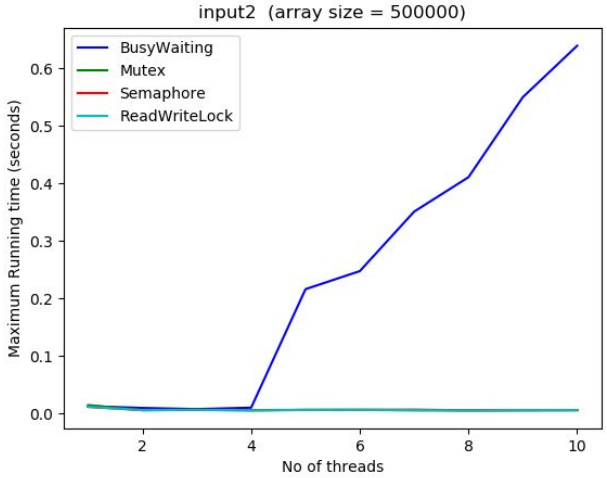


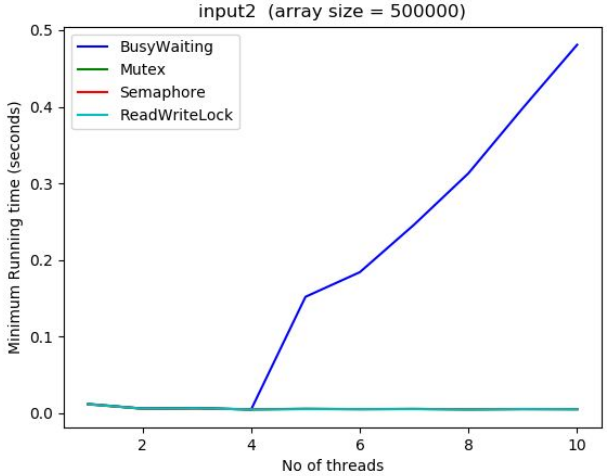




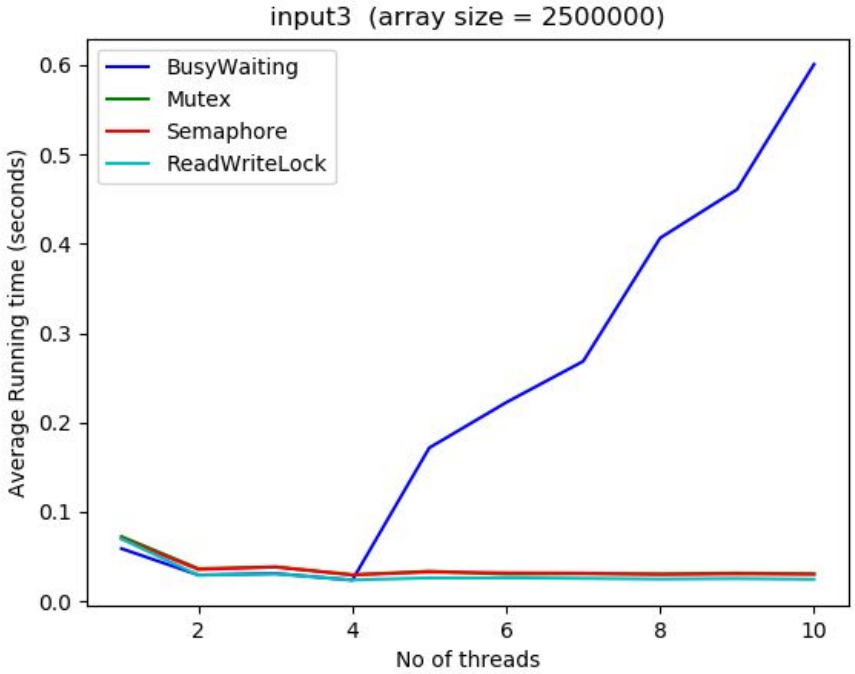
Observations (array_sum)

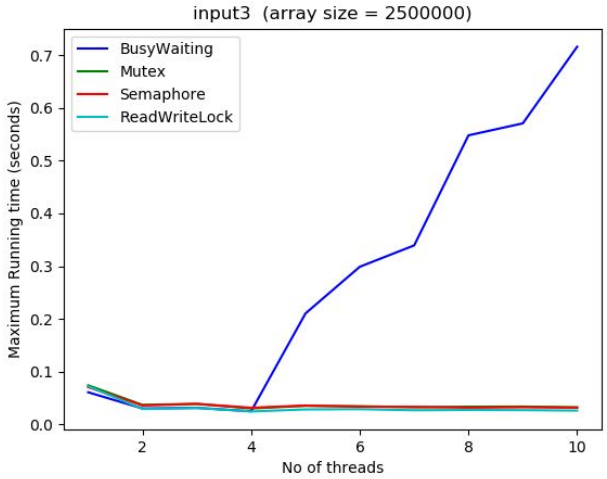


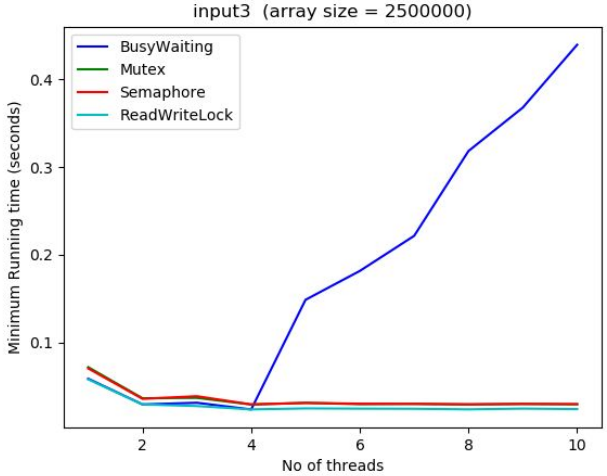




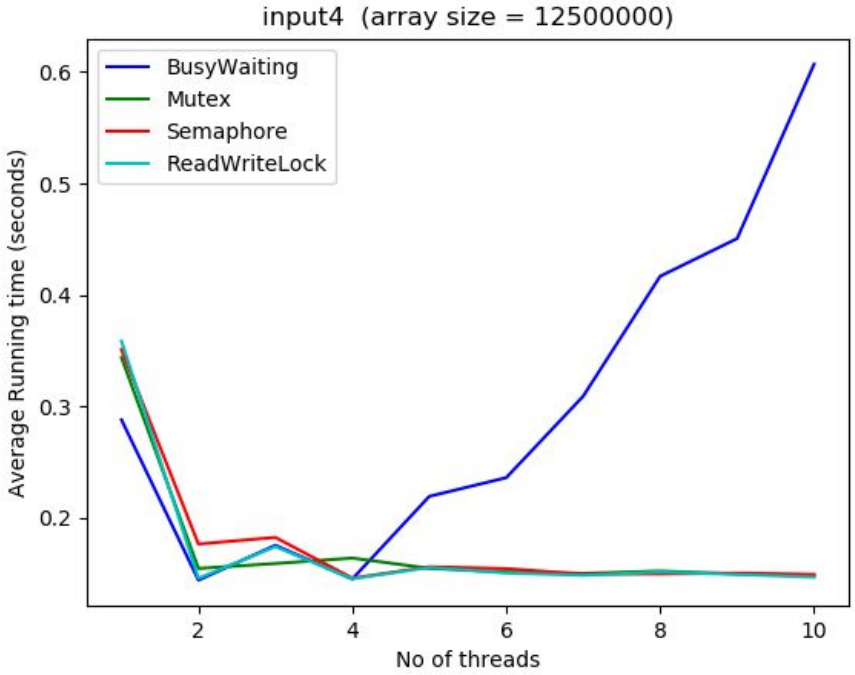
Observations (array_sum)

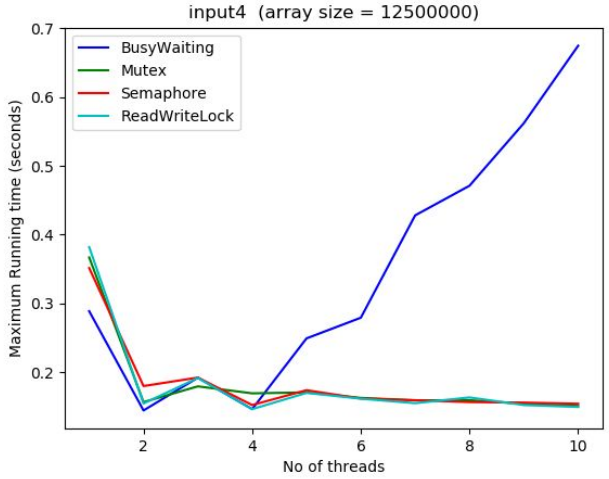


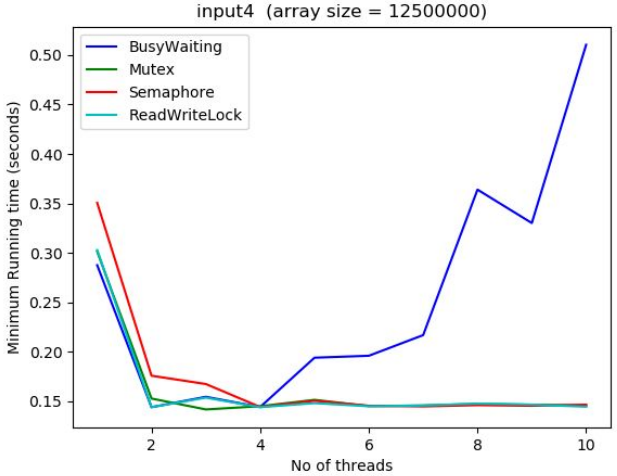




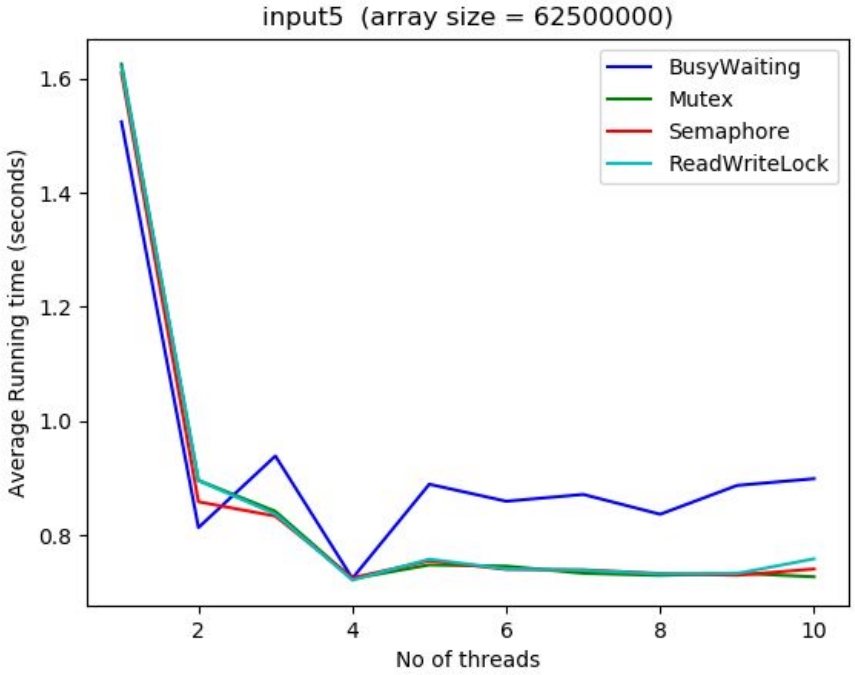
Observations (array_sum)

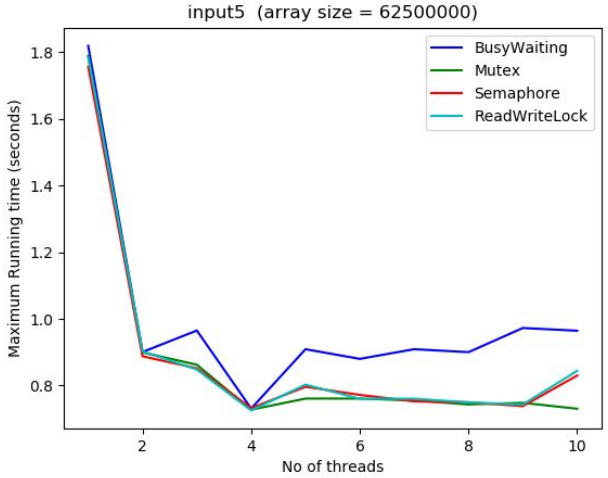


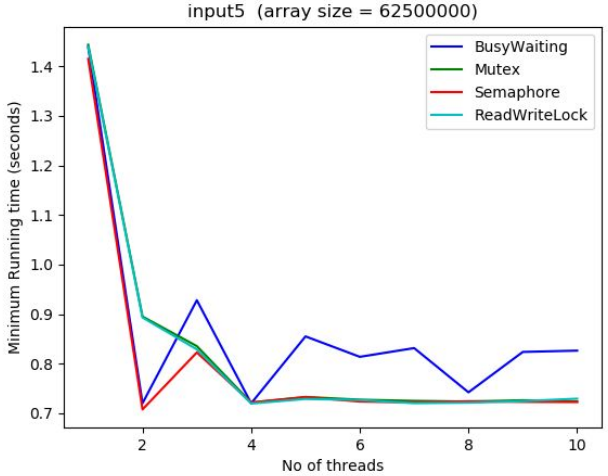




Observations (array_sum)







Analysis (array_sum)

- For all the constructs the time to sum decreases initially as the number of threads increases and eventually increases. The number of threads at the transition point is dependent on the communication overhead and the amount of work being parallelized.

- When the array size is small, the initial decrease is small. However as the size of the array increases, the decrease becomes more and more sharp - because of the increased amount of parallel work being done compared to same communication overhead.

- The transition point is mostly centered at 4 threads which is expected since at most four threads can be run in parallel. However this may occur early when the size of array is small.

- The overhead of busy waiting is much more than the amount of work being parallelized, with increasing number of threads (after the transition point mentioned above). This is visible as the constant increase in the busy-wait runtimes with increasing number of threads for a particular graph.

- However as mentioned in the second point as the array size increases the amount of parallel work being done with increase of threads is more and it appears that at somewhere around when the array size is 62500000 it increases to counteract the increasing communication and the graph shape is somewhat flat.

Analysis (array_sum)

- Performance wise mutex, semaphores and read-write locks are the same.
- As observed with busy-wait initially performance improves but after a certain point it becomes almost constant, as opposed to busy-wait for which the runtime increases because the overheads in implementation of mutex, semaphores and read-write locks are small.
- When the number of threads is small, busy-wait slightly outperforms the other three. However when the number of threads increases the busy wait performance degrades too much whereas mutex, semaphores and read-write locks maintains there performance. This difference is due to their implementation overheads involved.

Experiment - 2 (heat_eqlb)

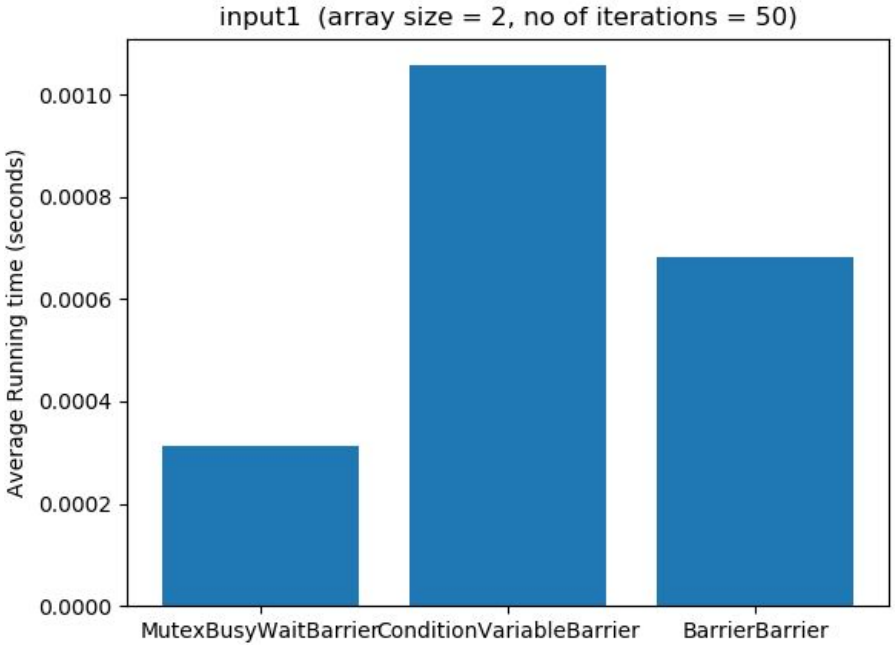
The second experiment is an algorithmic analogy of heat flow through a rod with varying temperature at various points. Ultimately, the rod reaches thermal equilibrium by heat flow from one part of the rod to the other. This experiment simulates the same process using threads. The synchronization constructs compared are

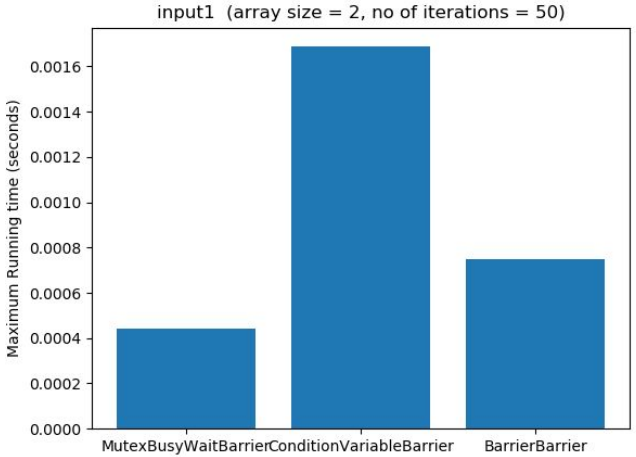
- Synchronisation using mutex and busy waiting
- Condition Variables
- Barriers

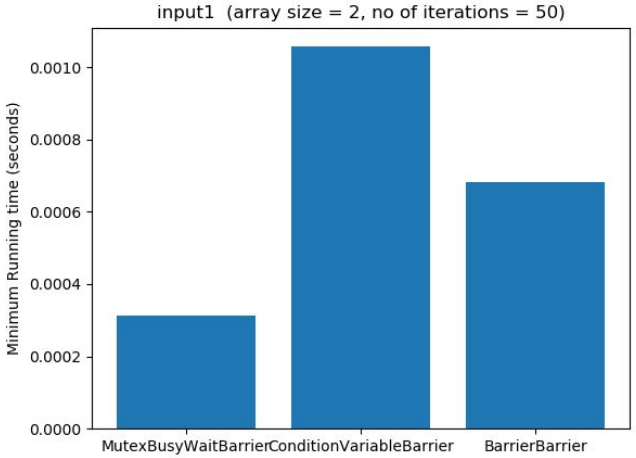
To maintain uniformity each of these is repeated a fixed prespecified number of iterations for an easy comparison.

These are compared for different array sizes and also for different number of iterations. Again the functions running under the threads for the three constructs are kept similar apart except for the codes specific to them. This ensures uniformity and no extra overhead for any of the three constructs. Also an average of running time over various runs is taken.

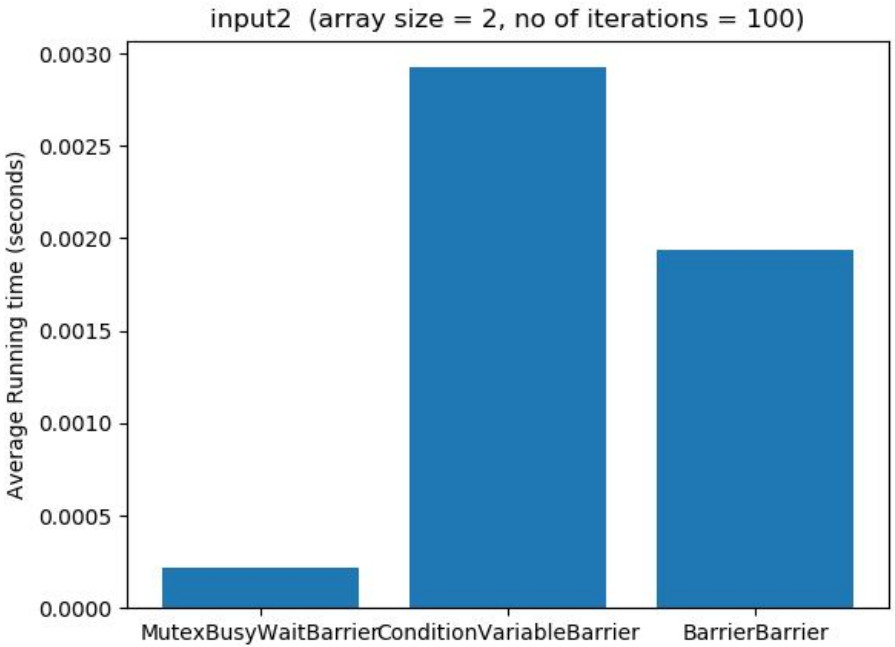
Observations (heat_eqlb)

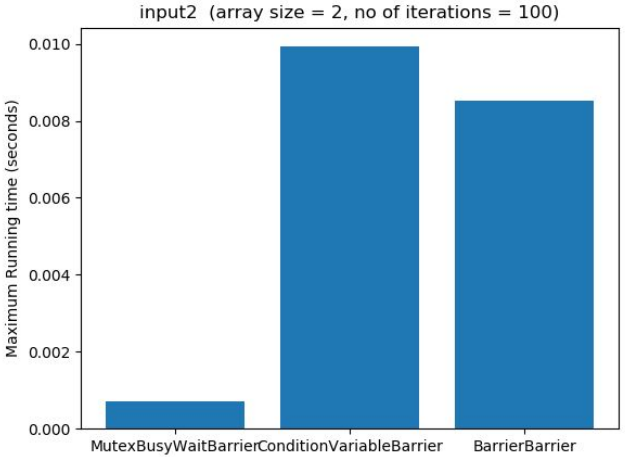


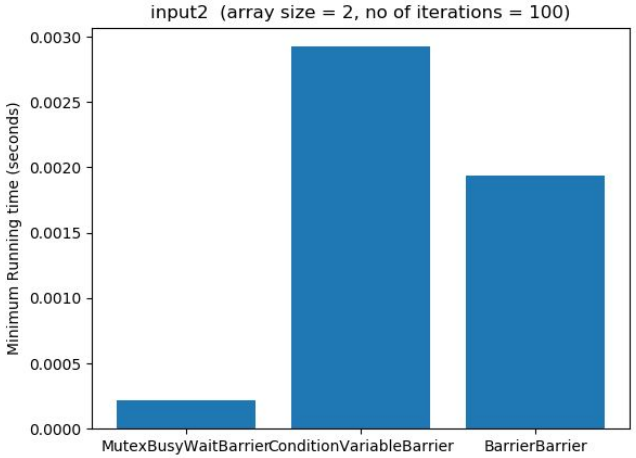




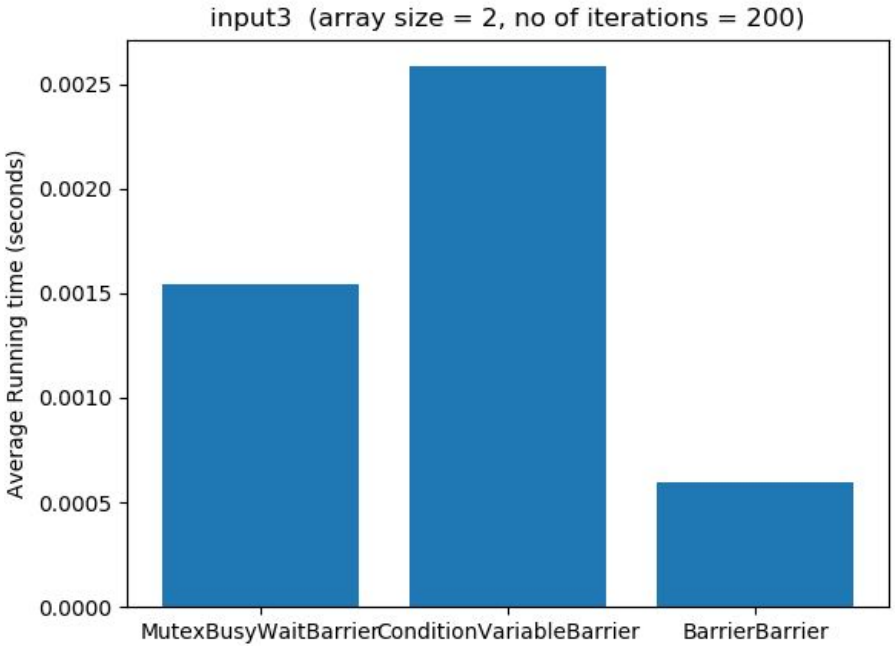
Observations (heat_eqlb)

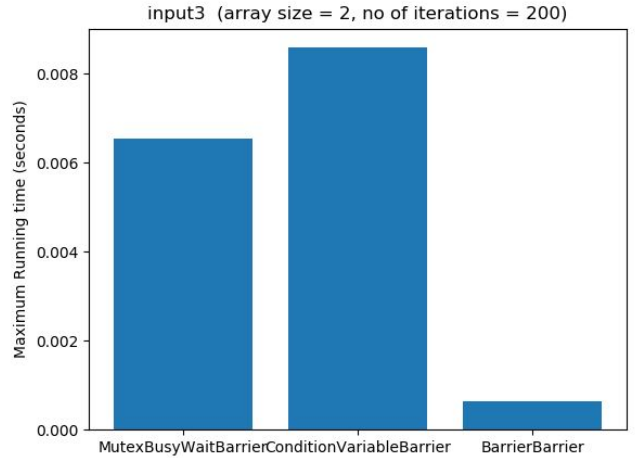


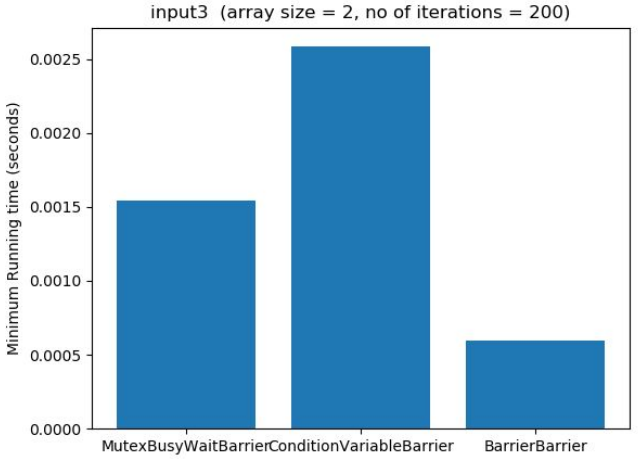




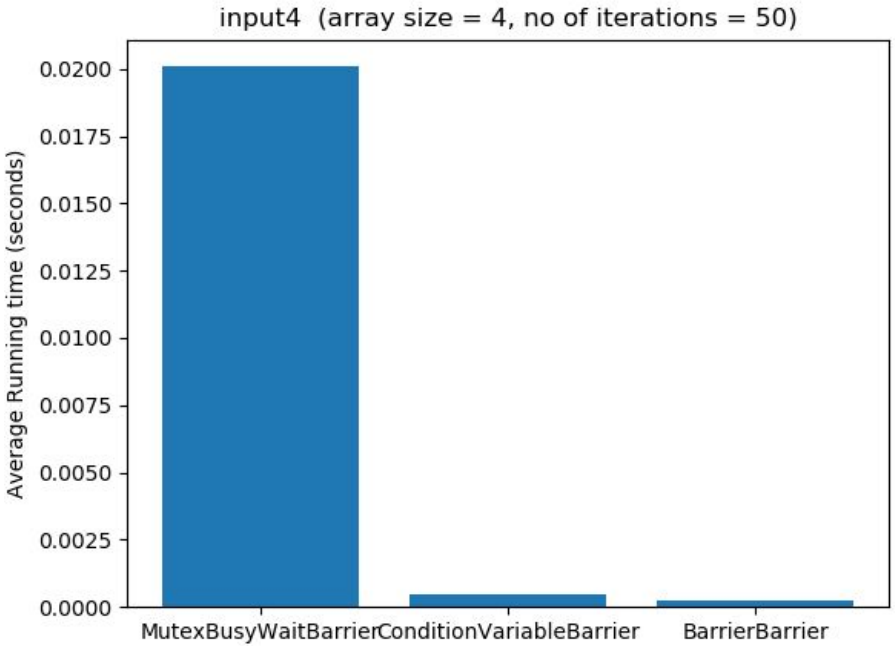
Observations (heat_eqlb)

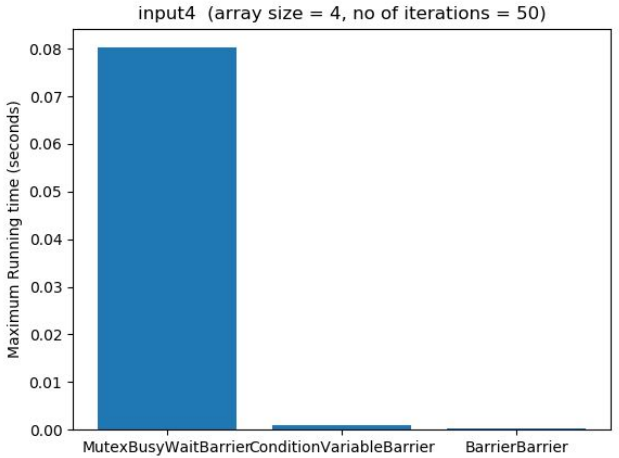


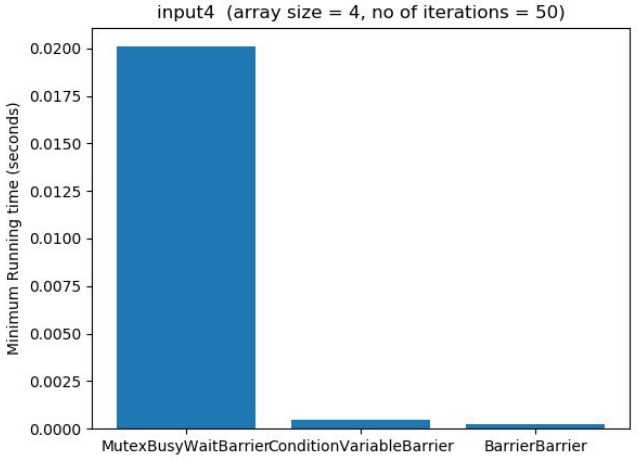




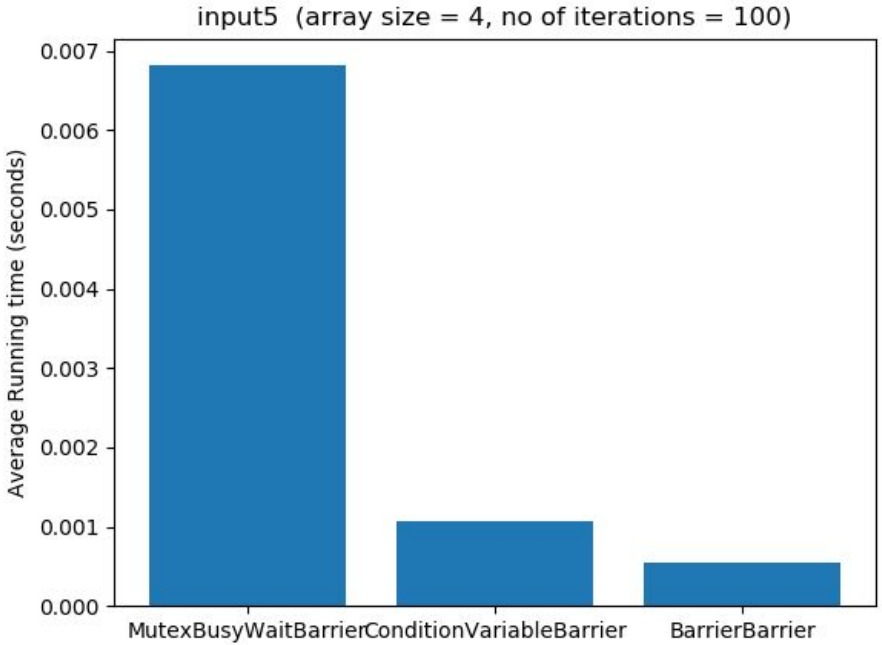
Observations (heat_eqlb)

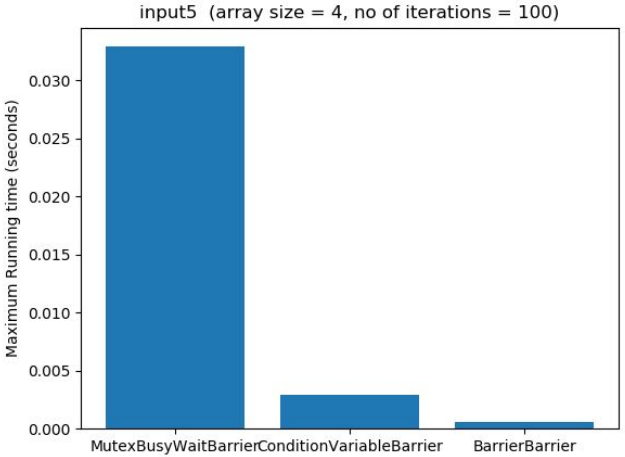


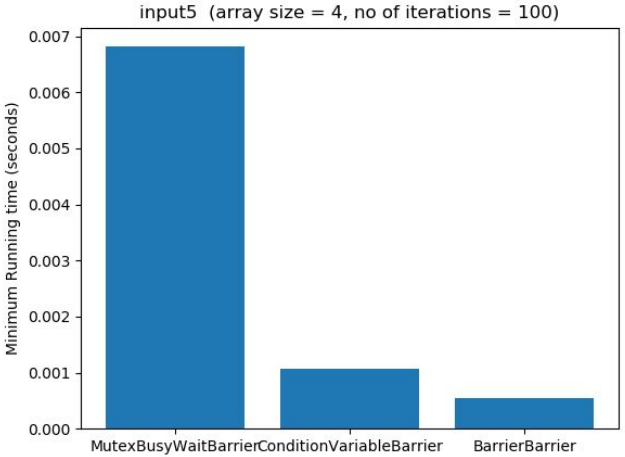




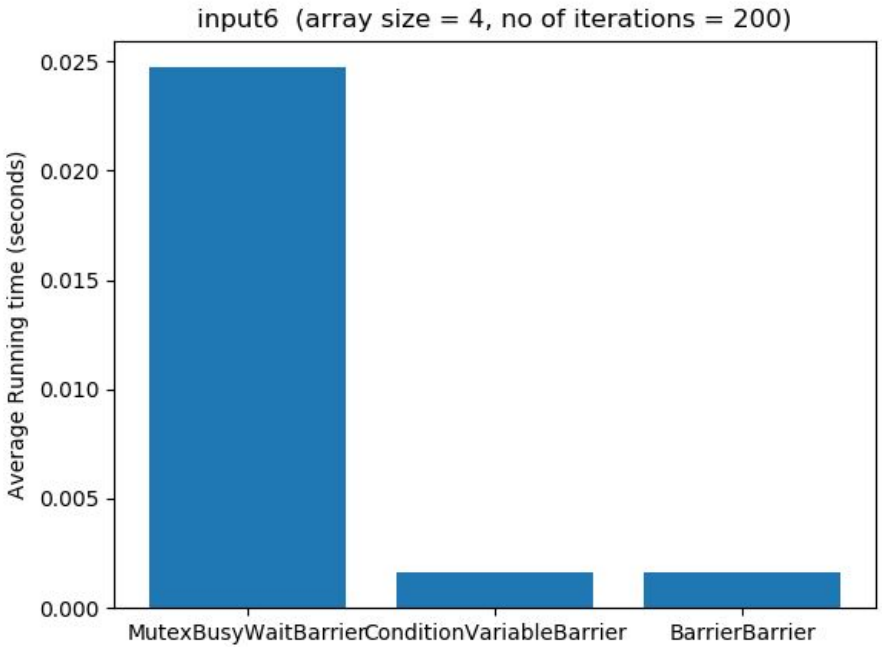
Observations (heat_eqlb)

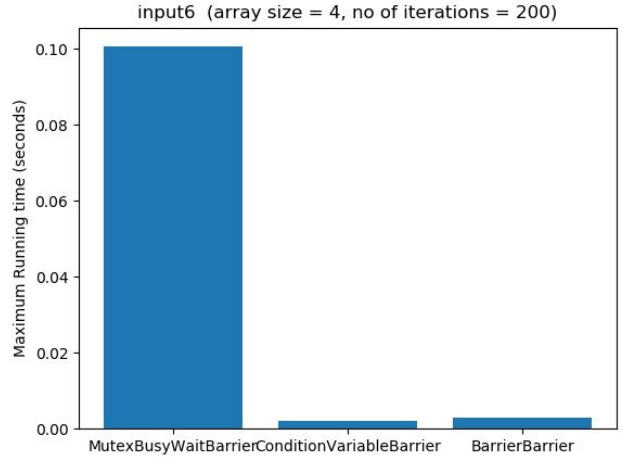


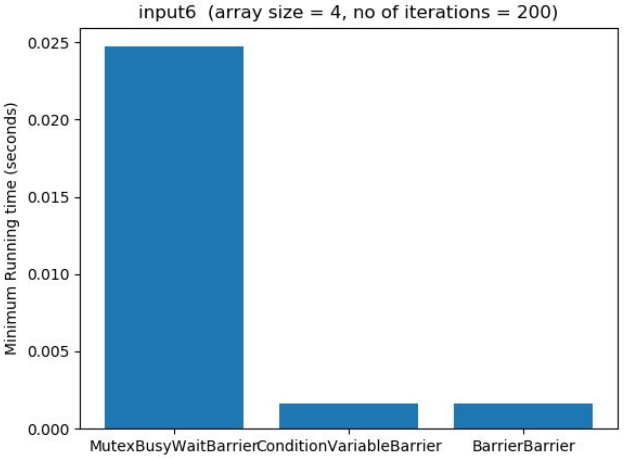




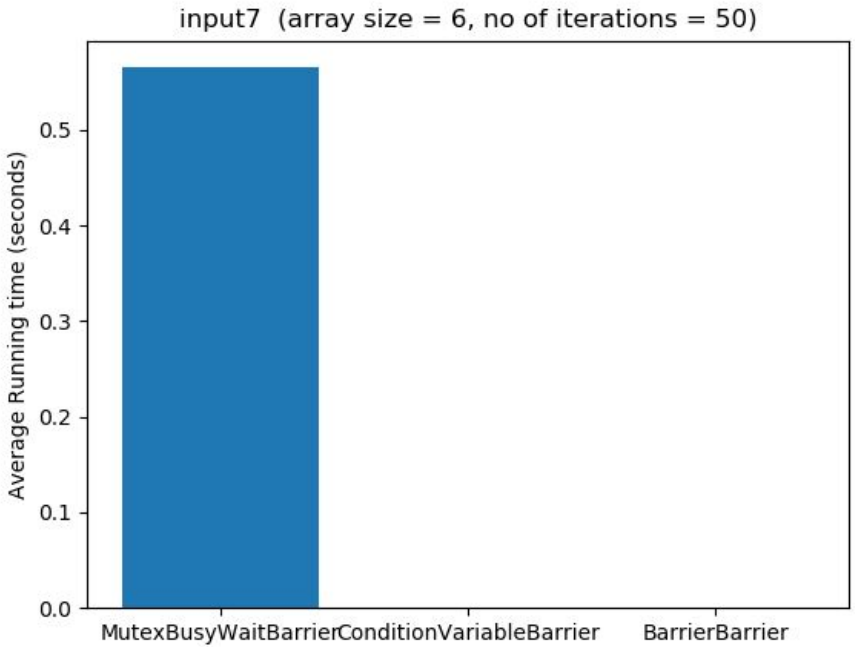
Observations (heat_eqlb)

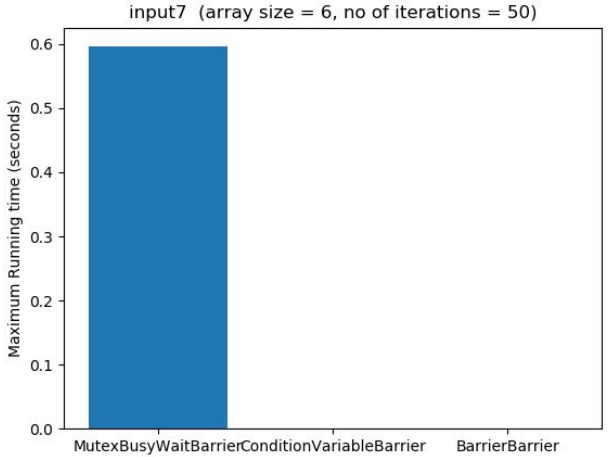


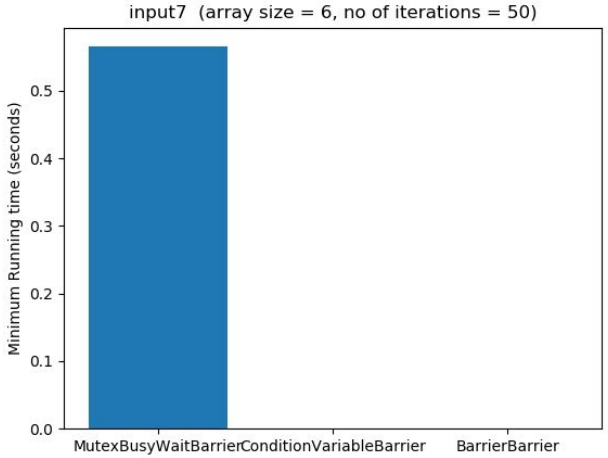




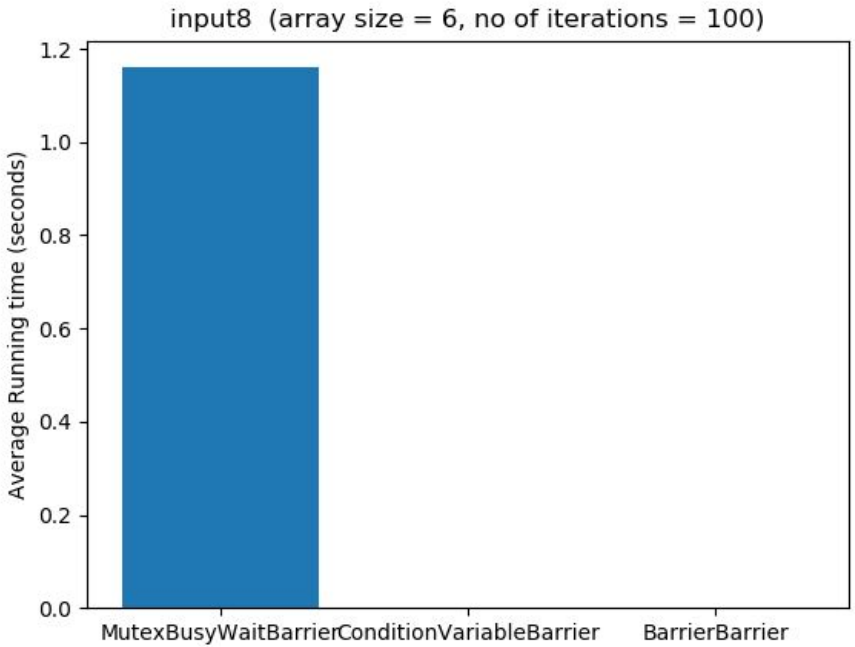
Observations (heat_eqlb)

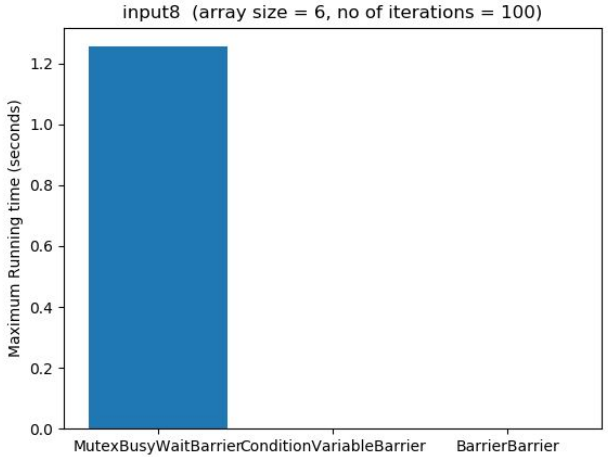


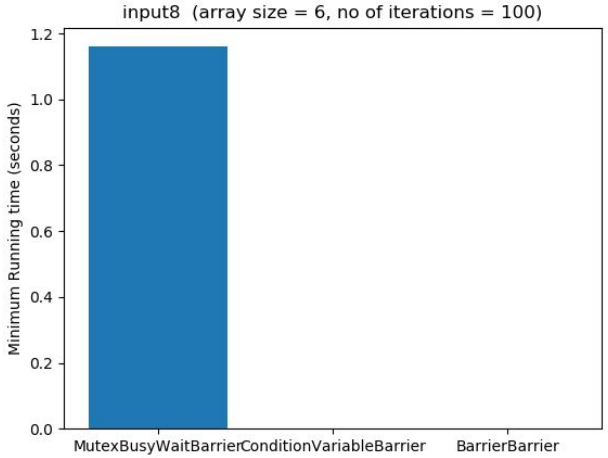




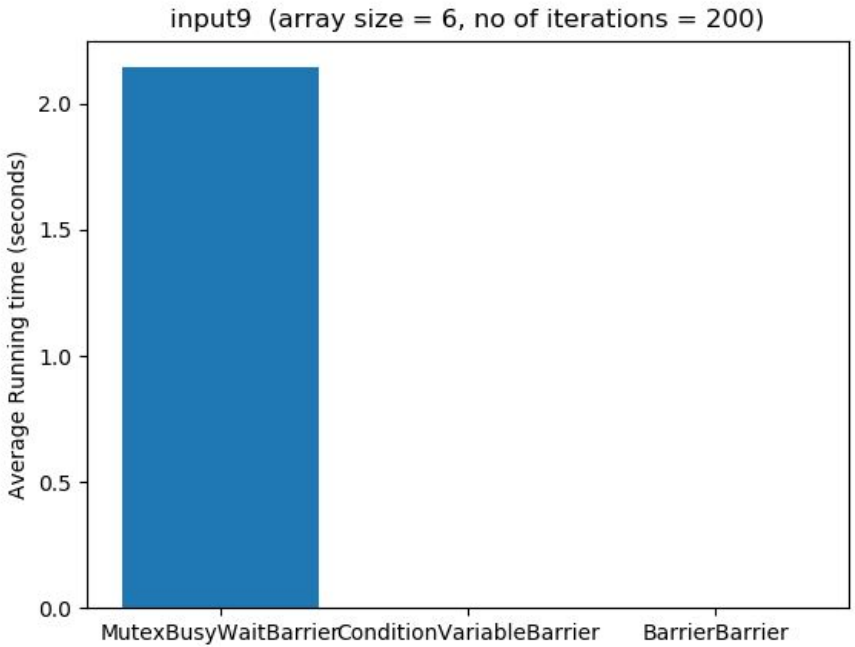
Observations (heat_eqlb)

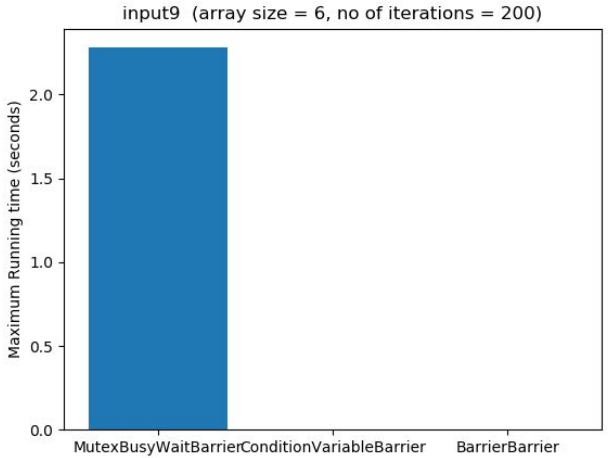


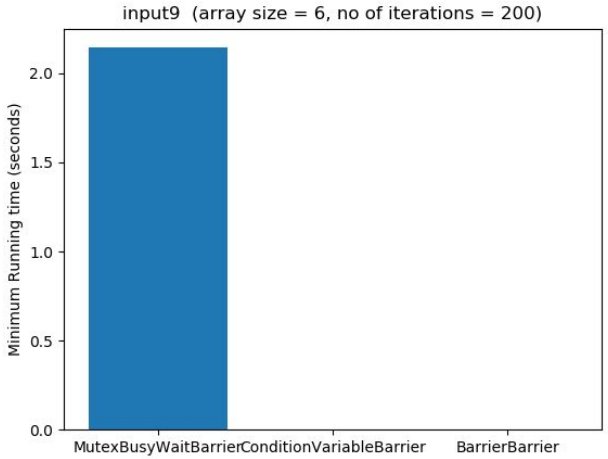




Observations (heat_eqlb)







Analysis (heat_eqlb)

- Initially when the number of threads is small busy waiting performs better than condition variables and barrier. However when the number of threads increases busy-wait is of no match to the performance of condition variables and barriers. This is because busy wait has much overhead which wastes CPU time.

- Also, when the number of iterations increases for small number of threads the performance of barrier outperforms busy wait. This shows that the increase in overhead with iterations is much more in busy wait than in barriers.

- Final thing to notice is that barriers are better than using condition variables. This may be because condition variables are more general constructs than barriers, which are specialised to handle only rendezvous.

Appendix - Raw Data (array_sum)

```
The sum of the array is: -338400
The size of the array is: 100000

The average time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.00288         0.00156         0.01398         0.06360         0.17289         0.20773         0.27994         0.38554         0.42247         0.56440

Mutex :
        0.00292         0.00151         0.00162         0.00175         0.00161         0.00169         0.00185         0.00185         0.00204         0.00185

Semaphore :
        0.00291         0.00145         0.00161         0.00162         0.00157         0.00175         0.00181         0.00202         0.00186         0.00199

ReadWriteLock :
        0.00277         0.00117         0.00130         0.00147         0.00127         0.00134         0.00151         0.00163         0.00161         0.00170

The maximum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.00431         0.00272         0.11773         0.19036         0.20800         0.27155         0.35141         0.46152         0.50036         0.68524

Mutex :
        0.00297         0.00155         0.00166         0.00202         0.00166         0.00219         0.00191         0.00217         0.00215         0.00219

Semaphore :
        0.00293         0.00146         0.00166         0.00199         0.00160         0.00217         0.00200         0.00207         0.00205         0.00221

ReadWriteLock :
        0.00296         0.00118         0.00134         0.00167         0.00135         0.00139         0.00155         0.00167         0.00172         0.00182

The minimum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.00253         0.00133         0.00129         0.00334         0.12067         0.15564         0.22965         0.30179         0.34557         0.40989

Mutex :
        0.00288         0.00146         0.00157         0.00149         0.00156         0.00135         0.00162         0.00139         0.00153         0.00145

Semaphore :
        0.00287         0.00143         0.00157         0.00123         0.00154         0.00166         0.00156         0.00199         0.00148         0.00158

ReadWriteLock :
        0.00242         0.00117         0.00126         0.00104         0.00125         0.00120         0.00139         0.00148         0.00137         0.00144
```

For input 1

Appendix - Raw Data (array_sum)

```
The sum of the array is: -2354630
The size of the array is: 500000

The average time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.01217         0.00653         0.00685         0.00690         0.18075         0.21615         0.28920         0.36121         0.47602         0.55358

Mutex :
        0.01406         0.00579         0.00626         0.00511         0.00607         0.00564         0.00542         0.00503         0.00548         0.00520

Semaphore :
        0.01138         0.00576         0.00624         0.00521         0.00599         0.00566         0.00558         0.00501         0.00548         0.00526

ReadWriteLock :
        0.01165         0.00581         0.00628         0.00506         0.00600         0.00570         0.00553         0.00494         0.00557         0.00517

The maximum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.01266         0.00965         0.00762         0.01014         0.21600         0.24737         0.35076         0.41056         0.54953         0.63891

Mutex :
        0.01452         0.00581         0.00634         0.00547         0.00621         0.00634         0.00549         0.00536         0.00553         0.00574

Semaphore :
        0.01150         0.00584         0.00627         0.00577         0.00625         0.00634         0.00675         0.00544         0.00562         0.00584

ReadWriteLock :
        0.01172         0.00586         0.00637         0.00541         0.00618         0.00630         0.00590         0.00528         0.00594         0.00585

The minimum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.01181         0.00587         0.00639         0.00470         0.15180         0.18395         0.24607         0.31321         0.39828         0.48096

Mutex :
        0.01176         0.00578         0.00621         0.00476         0.00565         0.00509         0.00538         0.00475         0.00529         0.00495

Semaphore :
        0.01132         0.00570         0.00622         0.00469         0.00539         0.00506         0.00539         0.00474         0.00505         0.00500

ReadWriteLock :
        0.01159         0.00576         0.00622         0.00475         0.00510         0.00506         0.00512         0.00473         0.00517         0.00494
```

For input 2

Appendix - Raw Data (array_sum)

```
The sum of the array is: -11039480
The size of the array is: 2500000

The average time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.05895         0.02942         0.03107         0.02366         0.17170         0.22244         0.26847         0.40640         0.46080         0.60080

Mutex :
        0.07245         0.03654         0.03821         0.02948         0.03364         0.03050         0.03082         0.03051         0.03152         0.03061

Semaphore :
        0.07032         0.03552         0.03851         0.02967         0.03271         0.03170         0.03132         0.03010         0.03058         0.03036

ReadWriteLock :
        0.06979         0.02911         0.03066         0.02387         0.02615         0.02620         0.02560         0.02488         0.02542         0.02457

The maximum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.06082         0.03012         0.03122         0.02490         0.21032         0.29885         0.33931         0.54802         0.57068         0.71596

Mutex :
        0.07397         0.03694         0.03891         0.03010         0.03515         0.03441         0.03239         0.03345         0.03369         0.03239

Semaphore :
        0.07063         0.03577         0.03866         0.03146         0.03565         0.03332         0.03318         0.03166         0.03241         0.03157

ReadWriteLock :
        0.07232         0.02916         0.03111         0.02445         0.02830         0.02878         0.02676         0.02733         0.02698         0.02611

The minimum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.05842         0.02904         0.03092         0.02328         0.14849         0.18148         0.22155         0.31842         0.36801         0.43977

Mutex :
        0.07158         0.03610         0.03651         0.02894         0.03079         0.02928         0.02968         0.02912         0.02967         0.02929

Semaphore :
        0.07010         0.03539         0.03843         0.02902         0.03073         0.02987         0.02960         0.02904         0.02949         0.02929

ReadWriteLock :
        0.05768         0.02904         0.02718         0.02334         0.02449         0.02422         0.02407         0.02338         0.02426         0.02370
```

For input 3

Appendix - Raw Data (array_sum)

```
The sum of the array is: -65517950
The size of the array is: 12500000

The average time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.28805         0.14420         0.17546         0.14524         0.21934         0.23610         0.30919         0.41680         0.45061         0.60700

Mutex :
        0.34387         0.15467         0.15910         0.16397         0.15477         0.15142         0.15016         0.15233         0.14980         0.14831

Semaphore :
        0.35101         0.17658         0.18251         0.14596         0.15613         0.15451         0.14956         0.14981         0.15048         0.14933

ReadWriteLock :
        0.35861         0.14587         0.17423         0.14535         0.15557         0.15060         0.14852         0.15150         0.14931         0.14687

The maximum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.28874         0.14431         0.19199         0.14624         0.24912         0.27921         0.42809         0.47119         0.56211         0.67480

Mutex :
        0.36690         0.15651         0.17937         0.16920         0.17057         0.16259         0.15903         0.15922         0.15401         0.15093

Semaphore :
        0.35150         0.17981         0.19212         0.15216         0.17378         0.16164         0.15917         0.15650         0.15573         0.15409

ReadWriteLock :
        0.38188         0.15443         0.19158         0.14625         0.16983         0.16143         0.15488         0.16327         0.15220         0.14939

The minimum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        0.28757         0.14395         0.15445         0.14414         0.19400         0.19596         0.21684         0.36397         0.33019         0.51046

Mutex :
        0.30139         0.15270         0.14167         0.14494         0.15138         0.14510         0.14565         0.14727         0.14646         0.14564

Semaphore :
        0.35061         0.17577         0.16738         0.14416         0.15039         0.14540         0.14470         0.14600         0.14536         0.14649
```

For input 4

Appendix - Raw Data (array_sum)

```
The sum of the array is: -317768020
The size of the array is: 62500000

The average time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        1.52415         0.81338         0.93883         0.72446         0.88934         0.85953         0.87148         0.83678         0.88724         0.89905

Mutex :
        1.62489         0.89611         0.84182         0.72373         0.74811         0.74586         0.73340         0.72984         0.73356         0.72720

Semaphore :
        1.61009         0.85862         0.83354         0.72533         0.75535         0.74027         0.73938         0.73295         0.72994         0.74088

ReadWriteLock :
        1.61957         0.89555         0.83713         0.72139         0.75813         0.74100         0.73878         0.73220         0.73303         0.75868

The maximum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        1.81866         0.90062         0.96497         0.73118         0.90898         0.88001         0.90899         0.90015         0.97254         0.96438

Mutex :
        1.78785         0.89933         0.86237         0.72762         0.76102         0.76098         0.75538         0.74348         0.74819         0.73051

Semaphore :
        1.75570         0.88785         0.85165         0.73234         0.79639         0.77193         0.75268         0.74987         0.73832         0.83005

ReadWriteLock :
        1.78387         0.90248         0.84812         0.72537         0.80283         0.76029         0.76066         0.74948         0.74306         0.84406

The minimum time spent for computing the array sum (in order of thread no from 1 to 10):
BusyWaiting :
        1.43874         0.72031         0.92806         0.72023         0.85518         0.81396         0.83158         0.74241         0.82385         0.82646

Mutex :
        1.44324         0.89520         0.83553         0.72158         0.73313         0.72752         0.72516         0.72385         0.72646         0.72409

Semaphore :
        1.41485         0.70786         0.82285         0.72195         0.73263         0.72374         0.72211         0.72448         0.72316         0.72276

ReadWriteLock :
        1.43945         0.89323         0.82892         0.71980         0.72906         0.72689         0.72054         0.72146         0.72469         0.72961
```

For input 5

Appendix - Raw Data (heat_eqlb)

```
For input1.txt
The size of the array is: 2
The number of iterations is: 50

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 0.00031 0.00044 0.00018
ConditionVariableBarrier : 0.00106 0.00169 0.00087
BarrierBarrier : 0.00068 0.00075 0.00063
python3 plot.py < data.txt
================================================================================
./array_sum input2.txt
For input2.txt
The size of the array is: 2
The number of iterations is: 100

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 0.00021 0.00072 0.00006
ConditionVariableBarrier : 0.00292 0.00993 0.00054
BarrierBarrier : 0.00194 0.00853 0.00022
python3 plot.py < data.txt
================================================================================
./array_sum input3.txt
For input3.txt
The size of the array is: 2
The number of iterations is: 200

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 0.00154 0.00655 0.00009
ConditionVariableBarrier : 0.00258 0.00859 0.00056
BarrierBarrier : 0.00060 0.00063 0.00055
python3 plot.py < data.txt
================================================================================
```

Appendix - Raw Data (heat_eqlb)

```
================================================================================
./array_sum input4.txt
For input4.txt
The size of the array is: 4
The number of iterations is: 50

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 0.02008 0.08023 0.00010
ConditionVariableBarrier : 0.00047 0.00086 0.00028
BarrierBarrier : 0.00023 0.00026 0.00021
python3 plot.py < data.txt
================================================================================
./array_sum input5.txt
For input5.txt
The size of the array is: 4
The number of iterations is: 100

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 0.00681 0.03290 0.00015
ConditionVariableBarrier : 0.00108 0.00295 0.00051
BarrierBarrier : 0.00054 0.00055 0.00053
python3 plot.py < data.txt
================================================================================
./array_sum input6.txt
For input6.txt
The size of the array is: 4
The number of iterations is: 200

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 0.02471 0.10052 0.00024
ConditionVariableBarrier : 0.00160 0.00201 0.00097
BarrierBarrier : 0.00160 0.00281 0.00100
python3 plot.py < data.txt
================================================================================
```

Appendix - Raw Data (heat_eqlb)

```
================================================================================
./array_sum input7.txt
For input7.txt
The size of the array is: 6
The number of iterations is: 50

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 0.56523 0.59601 0.49472
ConditionVariableBarrier : 0.00073 0.00077 0.00071
BarrierBarrier : 0.00039 0.00040 0.00038
python3 plot.py < data.txt
================================================================================
./array_sum input8.txt
For input8.txt
The size of the array is: 6
The number of iterations is: 100

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 1.15986 1.25531 1.08003
ConditionVariableBarrier : 0.00141 0.00154 0.00134
BarrierBarrier : 0.00068 0.00071 0.00064
python3 plot.py < data.txt
================================================================================
./array_sum input9.txt
For input9.txt
The size of the array is: 6
The number of iterations is: 200

The time spent for reaching equilibrium is (avg, max, min):
MutexBusyWaitBarrier : 2.14405 2.28026 2.01190
ConditionVariableBarrier : 0.00325 0.00341 0.00312
BarrierBarrier : 0.00158 0.00163 0.00152
python3 plot.py < data.txt
================================================================================
```

Sources

- Measurement of wall clock time - Geeksforgeeks
- Assignment source code - Bitbucket

Thank You