Navneet Nandan
ID: 201451076
4th year
B. Tech CSE

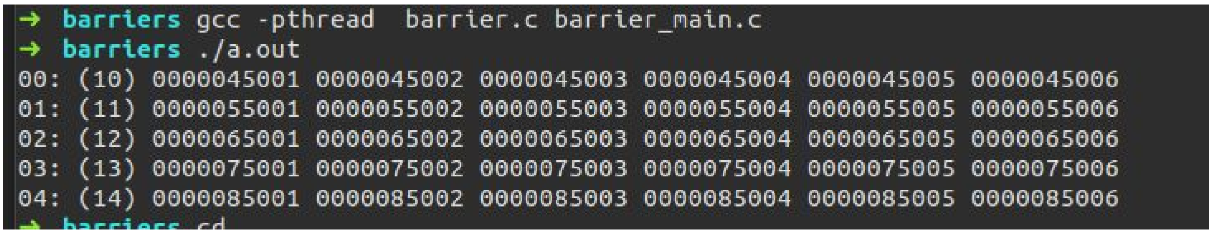
Lab 2 Report

Question 1

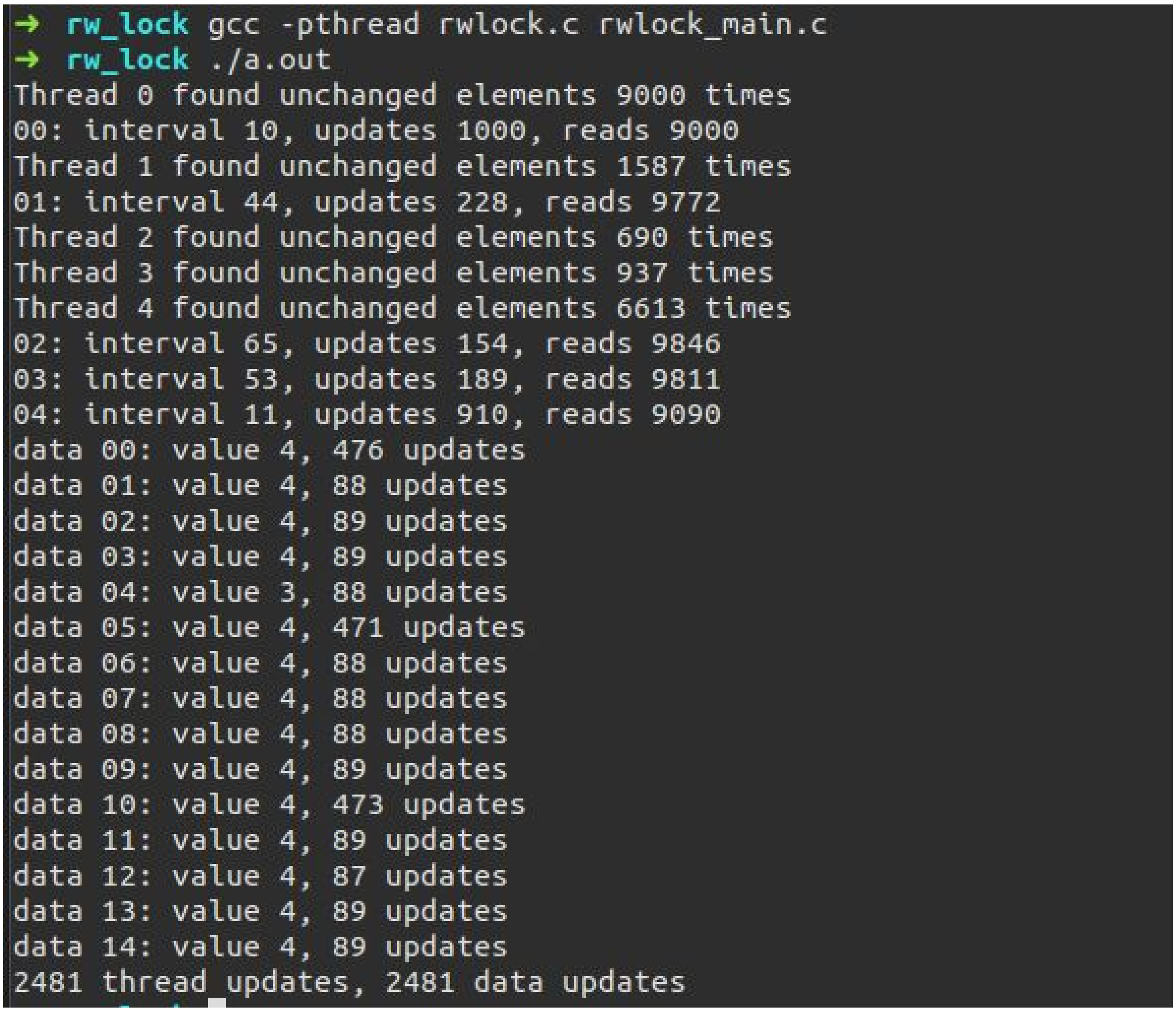
Objective:

To understand given read write lock and barrier program.

Screenshots



```
→ barriers gcc -pthread  barrier.c barrier_main.c
→ barriers ./a.out
00: (10) 0000045001 0000045002 0000045003 0000045004 0000045005 0000045006
01: (11) 0000055001 0000055002 0000055003 0000055004 0000055005 0000055006
02: (12) 0000065001 0000065002 0000065003 0000065004 0000065005 0000065006
03: (13) 0000075001 0000075002 0000075003 0000075004 0000075005 0000075006
04: (14) 0000085001 0000085002 0000085003 0000085004 0000085005 0000085006
→ barriers cd
```



```
→ rw_lock gcc -pthread rwlock.c rwlock_main.c
→ rw_lock ./a.out
Thread 0 found unchanged elements 9000 times
00: interval 10, updates 1000, reads 9000
Thread 1 found unchanged elements 1587 times
01: interval 44, updates 228, reads 9772
Thread 2 found unchanged elements 690 times
Thread 3 found unchanged elements 937 times
Thread 4 found unchanged elements 6613 times
02: interval 65, updates 154, reads 9846
03: interval 53, updates 189, reads 9811
04: interval 11, updates 910, reads 9090
data 00: value 4, 476 updates
data 01: value 4, 88 updates
data 02: value 4, 89 updates
data 03: value 4, 89 updates
data 04: value 3, 88 updates
data 05: value 4, 471 updates
data 06: value 4, 88 updates
data 07: value 4, 88 updates
data 08: value 4, 88 updates
data 09: value 4, 89 updates
data 10: value 4, 473 updates
data 11: value 4, 89 updates
data 12: value 4, 87 updates
data 13: value 4, 89 updates
data 14: value 4, 89 updates
2481 thread updates, 2481 data updates
```

Navneet Nandan
ID: 201451076
4th year
B. Tech CSE

Conclusion

Read write program can be used in the programmes where we need to handle the synchronisation of parallel thread as multiple threads are trying to read and write at same memory simultaneously.

Whereas barrier program can be used where we need to wait at a point until all threads working reach at that particular point.

Question 2

Objective:

To parallelize the code of dot product calculation of vector of given size and observe the speedup in comparison to serial program. To handle synchronisation use mutex lock.

Results:

Execution Time

Size of Vector	p=1	p=2	p=4	p=8
Vector Length = 100,000	0.03	0.03	0.04	0.04
Vector Length = 200,000	0.06	0.06	0.05	0.06
Vector Length = 100,000,000	1.310	1.225	1.051	1.05

Speedup

	p=2	p=4	p=8
Vector Length = 100,000	1	1.33	1.33
Vector Length = 200,000	1	0.833	1
Vector Length = 100,000,000	0.935	0.8023	0.8015

Navneet Nandan
ID: 201451076
4th year
B. Tech CSE

Conclusion:

When we parallelize the code of dot product, it is necessary to handle synchronisation as multiple threads would like to update value at same time. Mutex lock would be sufficient to solve this problem and can be used to do read write lock.
For vectors of small length there is no considerable speedup as there is very less calculation to do. While for larger vector there is considerable improvement in speedup due to parallelization.

Question 3

Objective:

Create a multi-access threaded queue on the principle of producer consumer problem such that at a time only one can produce(enqueue) or consume(dequeue) on the queue to avoid race conditions. 4 threads would be simultaneously enqueuing in the queue and 4 threads would be simultaneously dequeuing from it.

a) Ensure this mutual exclusion using mutex locks only also handle the case where queue is full or empty.
b) Ensure this mutual exclusion using mutex locks and conditional variables, also handle the case where queue is full or empty.

Observation:

a) Time observed for 1000 insertion and 1000 extractions each with 4 insertion threads (producers) and 4 extraction threads (consumers) is 0.03 sec
b) Time observed for 1000 insertion and 1000 extractions each with 4 insertion threads (producers) and 4 extraction threads (consumers) is 0.02 sec

Conclusion

Difference is 0.01 sec. This is observed because in second case when conditional variable is used if queue is full or empty, producer and consumer respectively give up their mutex lock and allow execution of other threads. This way there is no spinlock as in first case resulting in faster execution.