CS744 (Autumn 2017)

Midsem Solutions

1.
```
Variables:
  lock mutex
  condvar reader_can_enter, writer_can_inter
  bool writer_present = false
  int readcount = 0, writer_waiting = 0

Read lock:
lock(mutex)
while(writer_present || writer_waiting > 0)
    wait(reader_can_enter,mutex)
readcount++
unlock(mutex)

Read unlock:
lock(mutex)
readcount--
if(readcount==0)
    signal(writer_can_enter)
unlock(mutex)

Write lock:
lock(mutex)
writer_waiting++
while(readcount > 0 || writer_present)
    wait(writer_can_enter, mutex)
writer_waiting--
writer_present = true
unlock(mutex)

Write unlock:
lock(mutex)
writer_present = false
if(writer_waiting==0)
    signal(reader_can_enter)
else
    signal(writer_can_enter)
```

2. Several possible solutions exist. The main thing to keep in mind is that the server should be able to assign a certain request in the buffer to a worker, and the worker must be able to notify completion. For example, the master can use pipes or sockets or message queues with each worker. When it places a request in the shared memory, it can send the position of the request to one of the workers. Workers listen for this signal from the master, process the request, write the response, and send a message back to the master that it is done. The master monitors the pipes/sockets of all workers, and assigns the next request once the previous one is done.

3. (a) Number of threads = 5.1/0.1 = 51. throughput at saturation is 1/0.1ms = 10,000 requests/sec.
   (b) 10/5.1 * 1000 req/s
   (c) By Little's law, avergae number of threads = 1000 * 10 ms = 10
   (d) Throughput = 1000 req/s, utilization = 1000 / 10000 = 0.1
   (e) Throughput and utilization do not change, but response time will reduce.

4. (a) Array A fits in cache, but every 64 byte line will lead to a miss. For every access of A, 1 in 16 will be cache miss, and next 15 will be hits. So average access time = (1/16) * 145 + (15/16) * 1 = 10 cycles.
   (b) A[i] and B[i] will always map to the same cacheline, because they are 4MB apart. So every access of A will be a miss because accessing B will flush out previous fetched line of A. So average access time = 145 cycles.
   (c) There will be one miss per line as A, and the lines of A and B will no longer conflict. Answer is the same as part (a).
   (d) Again, there will be one miss per line of A, so same as part (a).
   (e) The two halfs of A will occupy both slots in a set. So all of A will fit into cache. The first N accesses will have average time of part (a), and next N accesses will have average time of 1. So average = 5.5 cycles.

5. (a) ceil $\frac{v-k}{k-e}$
   (b) $v - k - (l - 1) * (k - e)$

6.
```
sem buyer = 0; sem seller = 0;

Buyer thread:

up(buyer)
down(seller)
completeBuy()
```

7. (a) Size of each cache set = 64 * 8 = 2^9. Number of cache sets = $8 * 2^{20}/2^9 = 2^{14}$.
   Tag = 32 - 6 (offset) - 14(index) = 12.
   (b) Size of each cache set = 64 = 2^6. Number of cache sets = $2^{23}/2^6 = 2^{17}$.
   Tag = 32 - 6 (offset) - 17 (index) = 9.

8. The master process can open a socket on port 80, fork multiple processes, and all child processes can accept connections from the same socket on port 80 using locks for mutual exclusion. Or, the master process can alone listen to requests on port 80 and assigns all blocking disk I/O to worker processes.

9. Apply little's law to easy and hard requests separately.

```
mu (hard) = 100 req/s
mu (easy) = 500 req/s

lambda (hard) = 10 req/s
lambda (easy) = 10 req/s

rho (hard) = 10 / 100 = 10%
rho (easy) = 10/500 = 2%

Total rho = 12%
```

10. 0 (TLB hit implies a physical page has been mapped, so a page fault cannot occur)

11. 2

12. Shared

13. Invalid

14. No

15. Yes

16. False (C is adopted by init and continues to run)

17. True

18. False (physical frame is allocated when accessed, but PTE can be created earlier).

19. False (if integers are on same cache line, then it is shared between cores, false sharing)