Gordon Ng U82744816

1a) Yes when you have 3 machines and machine 1 sets x=7, applies a lock and sets y = next address x and unlocks it. M1 comes along and uses a local lock to set z=y, M2 comes along and gets z=y, but when asking to print the pointer to z, the next address x value is garbage, M2 doesn't know this and reads stale data.

1b) Release consistency waits for all operations are ready and then runs the next operation on thread M1. LRC on the otherhand does not need to wait until acquire access has completed. So LRC outpreforms in a DSM structure when there's many small packets and little big packets.

1c) No in a casual consistency model, contents could be left behind. In question 1a, we can counter the false stale read by leaving contents behind.

Example:

M0: x=7
a.lock
y=&x
a.unlock()

M1:
a.Lock()
b.lock()
z=y
b.unlock()
a.unlock()

M2:
b.Lock(); print*z; b.unlock()

3a) Actions Variables:

U_1: Remove Bob from Permissions

U_2: Add P_{10} to album A from Albums

In a eventual consistency model, U_1 will go to $Replica_1$, and U_2 would go to $Replica_2$. However updates are applied oppositely and R_2 updates U_2 with stale version. Bob reads a state of the record that should have never existed.

3b) Use a pub/sub mechanism that replays updates that are lost. Read_any also allows us to read to not most up-to-date value. You could also shuffle the people around and turn off notifications, instead of shuffling the photos around.

3c) As listings Management data, PNUTS ordered table can be used to store ratings such as likes. Storage units store tablets use get() and scan() to retrieve and set() to update to message broker.

4a) True

b) True

c) False

d) False

e) False

f) Yes, because in section 3, it describes a linearizability checker, and since sequentially is weaker than per obj linearizability, we can check the graph using a subfunction/parts of the linearizability checker to check the graph.

g) Yes, all values remember what was before/ has a causality relationship from before. Here, only x0 and y0 have to remember what was before but they don't get called again in the future.

5a) We can handle duplicates so easily because of duplicate filtering and it is no problem for cache, An update can be truncated (data in log for 1-2 days) & Applications A and B are not affected by this failure.

b) wormhole stores it's data markers everytime it acknowledges the translog. It is expensive to acknowledge each update, but it also uses TCP for delivery resulting in no package loss. Subscriber does not keep tract of the data markers.

6) a) Borg automates deployment of apps and resource management on large clusters. Hides details of resource allocation and failure handling, must have available resources

b) Borg Master
- distribute load over BorgMaster replicas
- threading
- co design of master and scheduler to use read only data

Scheduler
- 20k Borgcell trace: 150+ tasks
- cant brute force it - 20k cycles per task
- Decomposition/partition tasks into classes
- Score caching
- relaxed randomization

c) No all data goes through the scheduler which checks for required resources.

d) No Borg should pack its resources according to the scheduler's scorer and assign priority that way due to the graphic nature of Cluster compaction matrix.

e)

8a) a flatmap has 3 partitions, Map has 4 partitions and reduceByKey has 4 partitions

b)

c) The RDDs that are recomputed is the entire map.

d) Spark will materialize on the last line because they create a saveAsTextFile in execution.

9) Yes, bounded staleness does ~~effect~~ affect accuracy by decreasing accuracy in the model. Adding more outlier data/overwhelming the model with skewed/biased data will certainly decrease accuracy, for the original model already modelled against its own test set. Bounded staleness does not provide benefit of being more flexible than eventual consistency or sequential consistency.

b) Vector clocks coordinate task-workers with ~~a~~ clock and show task dependencies They provide a means to track progress.

c) M_1 causally precedes M_2 because ~~M vector clock updates was at step 2~~ ~~M's last update of its vector clock~~

M1's last update shown by it's vector clock is at index 2, (time)

M2's last update shown by it's vector clock is at index 4. (time)

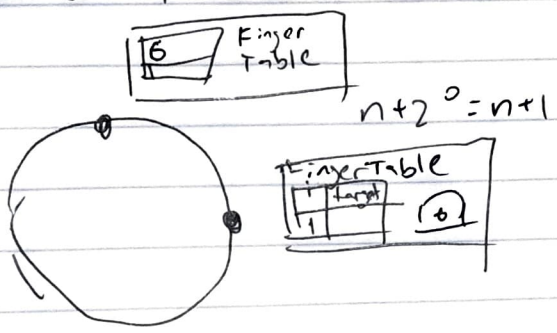
d) Parameter server recovers faster, because during failure, ~~training~~ data is sent to the next server and restart. During a crash, data can be replicated to different servers but in spark ~~it's~~ ~~[illegible]~~ data is completely lost during crash.

11a) Malicious users can succeed because even in the original bitcoin ~~code~~ crypto there was already a person that could be malicious, The fact that 40% of people are benevolent just increses that change. You could have the fastest blockchain.

a) No the time or computational power might not go through within the time frame. Bitcoin transaction can take ~~many~~ up to many days.

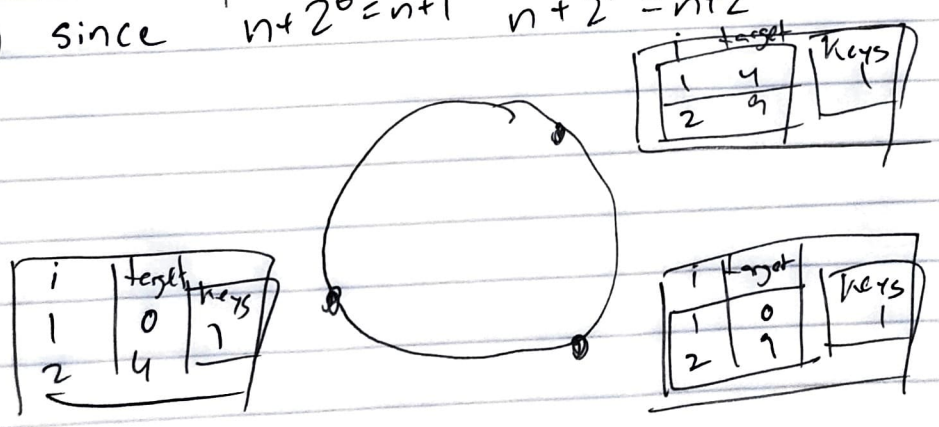
b) No, you increase P by adding time, but if you give them more time it will never hit 1 but they will be at .99999... or close to 1, never 1.

10)



b) Successor ~~node~~ node 4 for k_1, node 0 for k_2 and 0 for k_3
The amount of RPC calls are 1 for k_1, k_2 and 0 for k_3.
Predecessor node keeps tract of it's successor node, RPC until you find the predecessor to the keys in the successor.

c) since $n+2^6 = n+1$ $n+2^1 = n+2$



d) You need 4 RPCs, node 4 needs to tell its neighbors it is available it will update its own ~~successor~~ successor and predecessor pointers and each of its neighbors $(1+2+1)$ and get its finger table

12) a) PNUTS Paper #1 for sure #1
Dynamo Paper #2
Chord Paper #3
Primary Backup Replication #4
Trackerless Bittorrent #5
Parameter Server #runner-up

b) yes. Existential Consistency. Very boring.

c) PNUTS ~~was intro.~~/Treadmarks/ really structure my foundations
As for new ideas, Bittorrent, dynamo,
Learning from FB (over) google

d) Borg, Thor, (not including Raft & Zookeeper because they were assignments)
↑ Bitcoin