15-440/15-640: Homework 2

Due: October 13, 2017 10:30am (**NO LATE DAY**)

Name:
Andrew ID:

1 Concurrency Control

1. Is 2-phase commit blocking or non-blocking? What about 3-phase commit? Elaborate on your answers by explaining the situations where a transaction blocks or why it does not block.

   2-phase commit is blocking, 3-phase commit is non-blocking. In 2PC, when the coordinator and a member crashes, the new coordinator cannot decide which state the crashed member was in and thus cannot proceed until the crashed member recovers. Therefore 2PC is blocking. In 3PC, the coordinator can only COMMIT after collecting all PRECOMMIT ACKs from members, such that all members are awared of the decisions of other members in the COMMIT stage. Hence 3PC is non-blocking.

2. Consider the following program:

```
// DB is a key/value database
acquireWriteLock(key1)
acquireWriteLock(key2)
x = DB[key1]
DB[key2] = x
releaseLock(key2)
acquireReadLock(key3)
y = DB[key3]
DB[key1] = y
releaseLock(key3)
releaseLock(key1)
```

   (a) Which ACID property(s) does this program violate?

   Atomicity: The operation could crash halfway through and commit some of its changes but not others. Isolation: Another action could depend on the value at `key2` and affect the value at `key3` in the middle of this operation.

   (b) How would strong strict 2-phase locking fix it?

   First, the program would not be able to release a lock and acquire another halfway through. Second, the program would have to commit all of its changes at once, instead of piecemeal.

2 Distributed Mutual Exclusion

1. In class we discussed enforcing mutual exclusion, among other ways, via a central server, majority voting, and token ring.

   (a) How many messages are required per request under heavy contention? What are the messages used for? Write your answer for each algorithm.

Under heavy contention, Token Ring: One message, assuming every node has requests for executing critical section. The message is used to pass the token (from previous node who just exited critical section to next node). Central Server: 3 messages: one request message for entering critical section, one grant message from coordinator to give permission, and one release message to notify the coordinator after exiting critical section. Majority Voting: 3mk messages. m is the number of nodes needed for a majority, k is the number of retries.

(b) List at least one disadvantage of each algorithm.

Token Ring: Token loss causes the system to shut down and it is costly to regenerate token. Central Server: Coordinator is performance bottleneck in large systems. Centralized algorithm also suffers single point of failure. Majority Voting: Majority voting may suffer from starvation during concurrent voting.

(c) Which of these systems is most robust to failure? Why?

A majority voting system is most robust to failure. Token ring suffers from failure when the token is lost, and the prevention measure is costly to implement. The coordinator in central server system is a single point of failure; the system will only recover after a new coordinator is created. In majority voting system, since there are multiple coordinators, crashing of one coordinator affects little for the node to gain majority votes from voters.

2. Consider three processes. The system has totally ordered clocks by breaking ties by process ID. It uses the Ricard & Agrawala algorithm. The timestamp for each process of id i is $T(p) = 10 * L(p) + i$, where $L(p)$ is a regular Lamport clock.

Each message takes 2 "real-time" steps to get delivered. Critical section takes 2 real-time steps. Fill in the table with the messages that are being broadcast, sent, or received between the processes until all nodes have executed their critical sections. Write "execute critical section" as the action for a node when it enters its critical section. The first three rows have been filled in for you, and the fourth row has been started. Assume that if a process receives messages from the other two processes at the same time, the message that comes from the lower process ID will be received first.

Action Types: Broadcast (B), Receive (R), Send (S), Execute Critical Section (ExCS) Initial timestamps: P1 → 111, P2 → 212 and P3 → 103

Real Time	Process	Lamport Time	Action(to/from)	Contents	Q at P1	Q at P2	Q at P3
1	1	111	B	(request 111)	111		123
	3	123	B	(request 123)			
2	2	222	B	(request 222)	121	222	123
3	1	131	R from 3	(request 123)	111	111	111
	2	232	R from 1	(request 111)	123	123	123
	2	242	R from 3	(request 123)		222	
	3	133	R from 1	(request 111)			
4	1	231	R from 2	(request 222)	111	222	123
	2	252	S to 1	(reply 111)	123		222
	2	262	S to 3	(reply 123)	222		
	3	233	R from 2	(request 222)			
	3	243	S to 1	(reply 111)			
5					111	222	123
					123		222
					222		
6	1	R from 2	261	(reply 111)	111	222	123
	1	R from 3	271	(reply 111)	123		222
	3	R from 2	273	(reply 123)	222		
7	1	ExCs	281		123	222	123
					222		222
8							
9	1	291	S to 3	(reply 123)		222	123
	1	301	S to 2	(reply 222)			222
10							
11	2	312	R from 1	(reply 222)		222	123
	3	303	R from 1	(reply 123)			222
12	3	313	ExCs			222	222
13						222	222
14	3	323	S to 2	(reply 222)		222	
15							
16	2	332	R from 3	(reply 222)		222	
17	2	342	ExCs				

3 Logging and Crash Recovery

1. The ARIES logging and crash recovery design we talked about can also recover from a crash during the recovery phase for a previous crash. How is this achieved?

   For each record that is undone in the recovery phase, a compensation log record (CLR) is created. This ensures that changes that were alredy undone are not undone again in the case of a recovery phase crash or restart.

2. List one advantage and one disadvantage of checkpointing in a log based recovery system.

   Advantages: Need not scan the entire log to recover. Once a checkpoint is reached while traversing the log in reverse, the system can use that to construct the dirty page table and transaction table at that point for recovery. Disadvantages: May need to lock the whole disk, which will result in poor performance during checkpointing..

4 Distributed Replication/Paxos

1. Justify the need for the prepare phase in the Paxos algorithm.

   Consider a case where multiple servers send conflicting accepts. In that case, if a receiver chooses one of them and then crashes, there is no way to determine which one was accepted. This is because there isn't a single leader. The prepare phase determines the leader by getting a majority agreement. To read more: http://people.csail.mit.edu/alinush/6.824-spring-2015/stumbled/paxos-explained-from-scratch.pdf

2. You have set up a fault-tolerant banking service for the PNC bank (Paxos National Corporation bank). Based upon an examination of other systems, you've decided that the best way to do so is to use the Paxos algorithm to replicate log entries across three servers, and let one of your employees handle the issue of recovering from a failure using the log.

   The state on the replicas consists of a list of all bank account mutation operations that have been made, each with a unique request ID to prevent retransmitted requests, listed in the order they were committed.

   Assume that the replicas execute Paxos for every operation.[1] Each value that the servers agree on looks like "account action 555 transfers $1,000,000 from Yuvraj A. to Srini S.". When a server receives a request, it looks at its state to find the next unused action number, and uses Paxos to propose that value for the number to use.

   The three servers are S1, S2, and S3.

   At the same time:

   - S1 receives a request to withdraw $500 from Yuvraj.
     - S1 picks proposal number 501 (the n in Paxos)
   - S2 receives a request to transfer $500 from Yuvraj to Srini.
     - S2 picks proposal number 502

   Both servers look at their lists of applied account actions and decide that the next action number is 15. So both start executing Paxos as a leader for action 15.

   The first few messages are given below:

[1] In practice, most systems use Paxos to elect a primary and let it have a lease on the operations for a while, but that adds complexity to the homework problem.

```
S1 -> S1 PREPARE(501)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(501)
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(501)
S3 -> S1 RESPONSE(nil, nil)
```

For each of the following scenarios, give a sequence of events that could lead to it.

(a) The servers agree on the withdrawal as entry 15.

> If S2 does not send any proposal, and S1 sends the accept to the other three, the servers will agree upon the withdrawal as entry 15.

(b) The servers agree on the transfer as entry 15.

> If S2 sends a prepare with a higher number before S1 sends the accept, and then S2 proceeds to send accept, the servers will accept S2's proposal and not S1's. This will result in the transfer being agreed upon as entry 15.

5 Fault tolerance and RAID

A friend of yours bought 25 used hard drives from a sale. Each drive has the following characteristics:

- Capacity: 500 GB
- Sequential speed: 60 MB/s
- Random IOPS: 30 IOPS
- MTTF: 1 year

Your friend wants a lot of speed and a lot of storage, and decides to build a RAID-0 array with the disks.

1. Is this a good idea? Discuss this w.r.t. the effective capacity, sequential speed, random IOPS, the MTTF and data durability.

> This is not a very good idea because it does not provide durability. If the data is striped across the disks, the characteristics of the RAID array are as follows:
>
> - Random IOPS: 30 * 25 = 750 IOPS.
> - Capacity: 500 * 25 = 12.5 TB
> - Sequential speed: 60 * 25 = 1500 MB/s.
> - MTTF = 1 year / 25 = 15 days.
> - Failure of a single disk renders all data useless.

2. Another friend suggested that a RAID-5 array with 25 disks would be a better idea than RAID-0. Why? Compare all the characteristics of the RAID-5 arrangement with the RAID-0 arrange-

ment.

- Random IOPS(Read): 30 * 25 = 750 IOPS.
- Random IOPS(Write): 30 * 25 / 4 = 187.5 IOPS.
- Capacity: 500 * 24 = 12 TB
- Sequential speed (Read/Write): 60 * 24 = 1440 MB/s
- Failures: The mean time to a single disk failure is 1/25 year. The time to data loss because of failure of a second disk is 1/24 year after the failure of the first disk if the RAID array is not rebuilt.
- Failure of a single disk can be tolerated (data can be recovered), although the RAID array cannot be used after that.

3. Your friend was a little disappointed on hearing that the system would fail very soon and new disks would be needed to replace the failing ones. Your friend agrees to compromise a little on storage space and speed. What would be a good design so that no additional disks need to be purchased for about a year?

The idea is to use some of the disks as spare replacement disks. Use 13 disks in a RAID-5 array. This leaves 12 disks to use as spares. This is because on an average, around 12 disks will fail in the span of a year.