

## Homework 4 (Distributed File Systems, Self-Stabilization, Transactions and Concurrency Control, Replication) - 100 Points - Solutions

CS425/ECE 428 Distributed Systems, Fall 2009, Instructor: Klara Nahrstedt

**Out:** Tuesday, November 10, **Due Date:** Tuesday, December 1

**Instructions:** (1) Please, hand in hardcopy solutions that are typed (you may use your favorite word processor). We will not accept handwritten solutions. Figures and equations (if any) may be drawn by hand. (2) Please, start each problem on a fresh sheet and type your name at the top of each sheet. (3) Homework will be due at the **beginning of class** on the day of the deadline.

### Problem 1 (15 Points) :

Consider a Calendar composed of “week”, “days” and “hourly time slots”. Indicate what actions (block , grant lock, release lock) is taken and what type of locks are set for each of the following actions in the following sequence:

- (i) – Process **A** wants to insert an appointment for slot 11:00 am of Monday of week-10.
- (ii) – Before Process **A** completes, process **B** requires to read entries in week-10;
- (iii) – Process **A** finishes and commits.
- (iv) -- Process **C** wants to block off Monday of week-10 for a conference.

### Solutions:

- (i) – Process A wants to insert an appointment for slot 11:00 am of Monday of week-10.

A acquires (action: grant Write Lock)

- Write Lock on 11:00 AM time slot of Monday of Week-10; and
- Write on Monday of week 10; and
- Write on Week 10

- (ii) – Before Process A completes, process B requires to read entries in week-10;

B is blocked (waits) because the Read-Lock it needs is in conflict with the Write on Week 10; (i.e., B blocks until A releases the Write Lock)

(iii) – Process A finishes and commits.

Process A releases its Write-Lock on 11:00 am slot of Monday of Week-10; write locks on Monday of Week-10 and Week-10 are also released as there are no other locks on components of Monday of week 10 and week-10 itself.

Process B acquires

- Read lock on Week 10
- Read lock for every day and hour slot of every day of week 10

(iv) Process C wants to block off Monday of week-10 for a conference.

Process C waits since the Write Lock it needs for Monday of Week-10 is blocked by the Read Lock on Week 10.

**Problem 2 (25 Points):**

A server manages the objects  $a_1; a_2; \dots; a_n$ . The server provides two operations for its clients: (1) **read(i)** returns the value of  $a_i$ ; and (2) **write(i; value)** assigns “value” to  $a_i$ . Consider the following interleaving of transactions **T**, **U**, and **V**:

Time	T	U	V
1	openTransaction	openTransaction	openTransaction
2	y=read(k);		
3		x=read(k);	
4		write(i,55);	
5		write(j,66);	write(i,77);
6		commit;	
7			
8	x=read(i);		
9	write(k,44);		
10			write(k,88);

(a) (strict 2-phase locking) Suppose the strict 2-phase locking mechanism is used for concurrency control. Answer the following questions:

- (i) Does U have to wait to acquire the lock for  $x = \text{read}(k)$ ?
- (ii) Does V have to wait to acquire the lock for  $\text{write}(k; 88)$ ?
- (iii) Does T have to wait for acquire the lock for  $x = \text{read}(i)$ ? If the answer to any of the questions is yes, specify at what time the transaction can acquire the lock.
- (iv) Do transactions T and V eventually commit? If the answer is yes, in what order?

**Solutions:**

- (i) Does U have to wait to acquire the lock for  $x = \text{read}(k)$ ?  
No, T and U can share the read lock on 'k' (read-read are not in conflict).
  
- (ii) Does V have to wait to acquire the lock for  $\text{write}(k; 88)$ ?  
Yes, V must wait for the Write lock on 'k' until after T commits.
  
- (iii) Does T have to wait to acquire the lock for  $x = \text{read}(i)$ ? If the answer to any of the questions is yes, specify at what time the transaction can acquire the lock.  
  
Yes, T can't read 'i' until V commits and releases the write lock on 'i'.
  
- (iv) Do transactions T and V eventually commit? If the answer is yes, in what order does it happen?

No, T and V are deadlocked. T is waiting on 'i' while holding 'k' and V is holding 'i' while waiting on k.

(b) - Describe the information written to the recovery file on behalf of these transactions if strict 2-phase locking is used.

**Solutions:**

p0			p1	p2	p3 U prepped	P4 U commit
i	j	k	i	j	<i, p1 >	p3
initial	initial	initial	55	66	<j, p2>	
					p0	

**Problem 3 (20 Points):**

Replication

- (a) A number of processes are collaborating in a group communication. Initially, P joins the group and knows nothing about other members in the group. Then, it learns that Q and R and W have joined the group. Then it learns that X joins the group. A while later it learns that R leaves and then learns that X crashes. Show different views of the process P from the initial to the final stages of the above scenario.

**Solutions:**

$P_0 = \{P\}$

$P_1 = \{P, Q, R, W\}$

$P_2 = \{P, Q, R, W, X\}$

$P_3 = \{P, Q, W, X\}$

$P_4 = \{P, Q, W\}$

- (b) Given 9 replicas, if updates are reflected on 5 replicas and include timestamps, what is the minimum number of replicas we must read to ensure that we have at least one up-to-date data.

**Solutions:**

To ensure  $W + R > N$ ,  $R > 9 - 5$ ,  $R \geq 5$

- (c) Suppose we have 6 replicas with write-all, read-all policy in which view-based quorum is used with a write threshold of 4, in case of partition. If the system is partitioned into two parts of 3 nodes each, can we read and write in any of these partitions? Explain your answer.

**Solutions:**

The read threshold must be  $\geq 6 - \text{write-threshold} + 1$ ; read-threshold  $\geq 3$ .

With a read threshold of 3, reads can be done in either of the partitions.

Write threshold of 4 cannot be met in any partition, so no writes are performed.

**Problem 4 (15 Points):**

Consider the following three transactions:

<u>T:</u>	<u>U:</u>	<u>W:</u>
getbalance (A)	getbalance(B)	getbalance(B)
Getbalance (C)	deposit(A, v1)	getbalance (A)
Deposit(B, v2)	deposit(C, v3)	

Object A is replicated at servers X, Y and Z. Object B is available at M, N and P. Object C is replicated at X, M and Z.

T reads from X for A and C. Later, X fails. U reads from N for B. Later, N fails. W reads from M for B and from Z for A.

If “Available Copies” policy is used for replication control, show what failure timelines must be met for each transaction to be able to commit.

**Solutions:**

All transactions must commit in the view they start. T must commit in a view which includes X (it reads A and C from X) and any available replica of B when it started.

So X must have failed after T commits. N must have failed either before T starts or after it commits.

N is in the view of U when it starts (it reads B from N), so N must be in the U’s view when it commits. N must have failed after U commits. Available copies of A and C at the start of U must remain the same at U’s commit. So, X must have failed either before U starts or after U commits.

W’s view (M and Z) does not change.

**Problem 5 (10 Points) :**

Give three reasons why fetching entire files at clients is more preferable to fetching file blocks as they are accessed (hint: AFS)?

**Solutions:**

(i) Most files are small in UNIX file systems. (ii) Locality of reference. (iii) Fetching the whole under these assumptions has lower overhead compared to a block by block approach.

(Any reasonable answer will be accepted)

**Problem 6 (15 Points):****Consider Problem 14.1 in Textbook****Solutions:**

This modified protocol takes 2 rounds, which is faster than usual 2PC, but sends more messages ( $n^2 - n$ ). Because it relies on the receipt of more messages, the probability of a consistency problem increases. This arises from the fact that the protocol requires all  $n^2 - n$  "YES" messages to be successfully received for every participant to commit. This can be mitigated with resends, as in standard 2PC.