Assignment 8 - Solution

Problem 1 - We replicate database DB1.
1.A. Assume DB1 is a read-only database.

- **Could availability decrease due to replication? Why?**
  - Every transaction accesses one copy only.
  - Therefore, availability improves with more replicas.

- **Is there a bound on system throughput (read-only tx/sec) as we increase the number of replicas?**
  - There is no bound on system throughput insofar as the database system is concerned.
  - More replicas give more throughput.
  - Network may impose a bound, depending on the capacity of connections between clients and the database.

Problem 1.B.1

Assume DB1 processes update-only transactions.

- **Describe a situation where availability is decreased.**
  - There are several ways availability can be reduced.
  - For example, use an algorithm that requires a transaction to update more than one replica to commit:
    - Replication protocol specifies that *all replicas* must be updated within the transaction.
    - Failure of one replica makes system unavailable.
    - The more replicas, the higher the probability that one of them will fail.

Problem 1.B.2

- **Assuming infinite network bandwidth, what factor(s) bound the number of update-transactions per second?**
- **How is each factor affected as the number of replicas increases?**
  - System throughput is capped by the number of updates a single replica can perform because updates are performed on all replicas.
    - This is an upper bound on system throughput.
  - In a primary-copy system with synchronous updates of replicas, network latency bounds the update rate, since each transaction must be acknowledged by (a quorum of) the replicas before committing at the primary.
  - Two-phase commit imposes another limit.
  - The latter two factors reduce the maximum throughput as the number of replicas increase.

Problem 2

A database is replicated on two servers, server A and server B. Each transaction executes on one server using two phase locking and propagates its updates within the transaction boundary to the other server. Transactions commit using two phase commit.

Does this replication protocol provide one-copy-serializability? If "yes", provide an argument, and if "no", provide a history.

No. 1SR is not guaranteed even if each site uses 2PL (providing local serializability) and 2PC (providing atomic commitment).

To show this, suppose the database has two elements x and y and is replicated on two servers A and B.

$T_1 = r_1(x), w_1(y)$

$T_2 = r_2(y), w_2(x)$

$T_1 = r_1(x), w_1(y)$
$T_2 = r_2(y), w_2(x)$

Site A:	$rl_1(x_a)$	$r_1(x_a)$	$wl_1(y_a)$	$w_1(y_a)$	$ru_1(x_a)$	$wl_2(x_a)$	$w_2(x_a)$	C_1	C_2
Site B:	$rl_2(y_b)$	$r_2(y_b)$	$wl_2(x_b)$	$w_2(x_b)$	$ru_2(y_b)$	$wl_1(y_b)$	$w_1(y_b)$	C_1	C_2

The execution is not 1SR since in a serial execution on a one-copy database, one of the transactions would have read the other one's output, which did not occur in this execution.

The problem here is the early release of read locks. If read locks were held until after commit, then the execution couldn't arise.

Problem 3.
A database is replicated on two servers. Transactions run serially at both servers. Updates are propagated by executing each transaction on both servers.

Suppose the system executes the transaction program:
T = { read x; read y; write z = x + y; write w = time.now()}
Where time.now() is a database function that returns the current time.

What could go wrong using this approach?
T is non-deterministic, writing different values with each execution.
If T is used for update propagation, replicas will contain different values for data item w.