OBJECT REPLICATION

Problem:-

Can be described from a real-life graph problem based on web/data replication.

Consider a graph network; K users,
N nodes, &
r replicas of data item.

What if optimal placement of replicas if K > r & users access data item in read mode only?

Solution:-

A sol. requires min distance to be found i.e; cost from node to replica, a nearest one should be found.

Many such graph problems do not always have polynomial solutions

Use: -

In large distributed system; data replication is useful for
* rapid access of data
* fault tolerance.

Wolfson's ADR

Adaptive Data Replication Algorithm.

This ADR; algo adapts to read & write from diff nodes.

Defining Replication Scheme: as ~

Subset R of V & each node in R has a replica of object.
r_i & w_i denotes rates of read & write issued by node i.
$C_r(i)$ & $C_w(i)$ denotes cost of read & write.
R denote set of all possible replication schemes.

Goal is to minimize the cost of replication scheme

$$\min \left[\sum_{i \in V} r_i \cdot C_r(i) + \sum_{i \in V} w_i \cdot C_w(i) \right]$$

Algo assumes one copy serializability, which can be implemented by read-one-write-all (ROWA) policy.

ALGORITHM:-

Consider the n/w.

We can define as; set of nodes R containing the replicas around tree T having read & write activity.

So graph expands when the relative cost of read is more than that of write &
graph shrinks if $c_w > c_r$.

The equilibrium state subgraph for replication scheme is optimal.
This algorithm executes in steps that are separated by predetermined time periods / EPOCHS.
Irrespective of initial replication scheme; algo converges to optimal replication scheme in (diameter +1) no: of steps once read & write pattern stabilizes.

READ:-

A read operation is done from closest replica on tree/n/w.
If node issuing read query / receiving a forwarded read query is not in R; it forwards the query towards nodes in R along edges.
Once query reaches node in R; the value read is returned along same path.

WRITE:-

A write is performed to every replica in the current replication scheme R.
If write operation is issued by a node not in R, the operation request is propogated to closest node in R.
Once write operation reaches a node i in R; the local replica is updated & operation is propogated to

all neighbors of i that belong to R.
To implement this, a node need to track the set of its neighbors that belong to R. This is done using a variable, R-neighbor.

IMPLEMENTATION:-

To execute a read/write operation, a node needs to know;

* whether it is in R (so that it can read/write from local replica)
* which of its neighbors are in R. (to propogate write request)
* if node is not in R, then which of its neighbors is the unique node leads to R in n/w (so it can propogate read & write requests).

CONVERGING TO REPLICATION SCHEME:-

Types of Nodes in RS:-

R-Neighbor:-
Such a node i belongs to R but has atleast one neighbor j that doesnot belong to R.

R-Fringe:-
Such a node i belongs to R & has only one neighbor j that belongs to R. [Leaf node]

Singleton:-
$|R| = 1$.

Types of Tests in RS:-

EXPANSION TEST:- (A, E, D)

For each neighbor "j" not in R, compare.
R-Neighbor examines j to determine whether j can be included in replication scheme using ET.

$x \to$ no. of read that i received from j during last timeperi

$y \to$ no. of write

$\boxed{\text{read}(r) > \text{write}(w)}$

If $x > y$, i send to j a copy of object with an indication to save the copy in local database.
Thus j joins RS.

CONTRACTION TEST:- (A, E, C)

R-fringe node i examines whether it can exclude itself from RS; using CT.

$w > R$; node i excludes itself from RS.

Before leaving; node i must seek permission from j to prevent a situation where $R = \{i, j\}$; leaving no copies of the object.

SWITCH TEST:-

A singleton node i executes the ST to determine if it can transfer its replica to some neighbor to optimize the objective function.

Node i transfers replica to j if

r+w being forwarded by j > r+w that node i received.

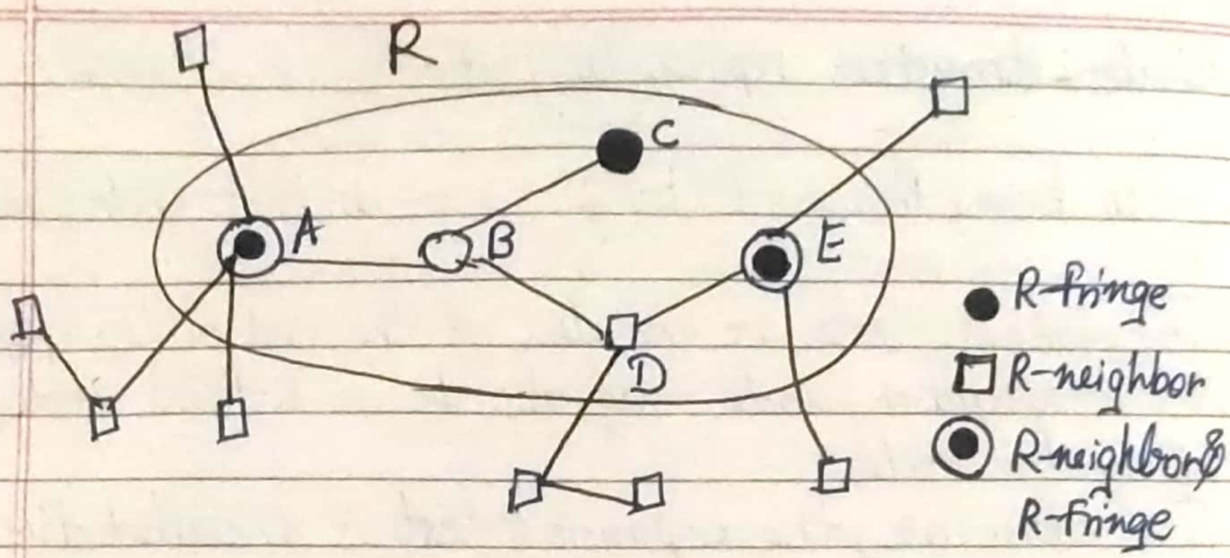
The various tests are executed at the end of the epoch.
An R-neighbor node may also be an R-fringe node/ singleton node.
In either case; the expansion test is executed first & if it fails the contraction test / switch test is executed.

Once read & write pattern stabilizes; resulting replication scheme is optimal.

TERMINATION:-

* After switch test succeeds, no other expansion test can succeed.
* If a node exits the replication scheme in a contraction test, it cannot re-enter the replication scheme via an ET.
* If a node exits the replication scheme in a ST, it can't re-enter the replication scheme again.

It can be seen that RS first expands wherever possible & then contracts.
Thus if node exits the RS, it can re-enter only by ST & that too if the exit was via CT. But then no ET can succeed. Hence; a node can exit the RS atmost twice, & after the first switch test, no expansion can occur. Hence the RS stabilizes.



R-fringe :- C, A, E ET- A,E,D

R-Neighbors :- A, E, D CFA, E, C

Both :- A, E

~~Suppose as no: of (r) by P2 is 4 & no: of (w) by P1 & P3 is 20; P2 doesn't enter RS.~~

~~No: of read by P3 is 32 & no: of write by~~

CONSISTENCY MODELS:-

Sequential Consistency → Casual Consistency.

In seq con, all writes must be seen in same order by all processes.
In casual con, causually related writes must be seen in the same order.
As the causally related writes form a subset of all writes, the requirement for causal consistency is certainly fullfilled ~~by~~ if requirement for seq con is fulfilled.

SEQUENTIAL CONSISTENCY:-

The result of any execution is same as if :-

* the memory operations (R/W) of all processes were executed in some sequential order &
* the operations of each individual processor appear in this sequence in the order specified by its pgm.

However; in seq con ~ writes are not instantly visible throughout the system. (impossible to implement).

P_1 : $W(x)1$

P_2 : $R(x)0$ $R(x)1$

(a)

P_1 : $W(x)1$

P_2 : $R(x)1$ $R(x)1$

(b)

T-Diagram·

a) Seq cons because only 1 write so globally consistent even if $R(x)1$ is not immediate to $W(x)1$. i.e; inspite ~~that~~ $W(x)1$ in P_1 is not instantly seen in P2.

b) Maintain order thus seq con.

CAUSAL CONSISTENCY:-

2 events are causally connected/related if one can influence the other.

P_1 : $W(x)1$

P_2 : $R(x)1$ $W(y)2$

Here, the write to x could influence write to y because, P2 might have read x & used its value to calculate y.

On the other hand, without the intervening read, 2 writes would not have been causally connected.

P_1: $W(x)_1$

P_2: $W(y)_2$

The following pairs of operations are potentially causally related.

* A read followed by a later write by same process.
* A write followed by a later read to same process.
* The transitive closure of the above 2 types of pairs of operations.

Operations that are not causally related are said to be concurrent.

CAUSAL CONSISTENCY:- Writes that are potentially causally related must be seen in same order by all processes.

Concurrent writes may be seen in diff. order in diff. process

Here is a sequence of events that is allowed with a causal consistent memory, but disallowed by a sequentially consistent memory.

P_1:	$W(x)1$	$W(x)3$		
P_2:	$R(x)1$	$W(x)2$		
P_3:	$R(x)1$		$R(x)3$	$R(x)2$
P_4:	$R(x)1$		$R(x)2$	$R(x)3$

Why is this not allowed by sequential consistency?

$W(x)2$ & $W(x)3$ are seen in a diff. order by diff. process i.e; it is $W(x)3$ happened 1^{st} & then $W(x)2$. But in P3 $R(x)3$ happened 1^{st} & then $R(x)2$ but vice-versa in P4.

Why is this allowed by causal consistency?

$W(x)2$ & $W(x)3$ aren't causally related, so may be seen in either order. $W(x)2$ & $W(x)3$ are concurrent in T-diagram.

What is the violation of causal consistency in sequence below?

P1: W(x)1

P2: R(x)1 W(x)2

P3: R(x)2 R(x)1

P4: R(x)1 R(x)2

$W(x)1$ & $W(x)2$ are potentially causally related, & so must be seen in same order by all processes.

Without $R(x)1$ by P2, this sequence would have been legal.

Some more eg^s:-

P1: W(x,2) W(x,4)

P2: R(x,4) W(x,7)

P3: R(x,2) R(x,7)

P4: R(x,4) R(x,7)

Sequentially consistent & causally consistent

P1: W(x,2) W(x,4)

P2: W(x,7)

P3: R(x,7) R(x,2)

P4: R(x,4) R(x,7)

Causally consistent but not sequentially consistent.