Name - PRABHAKAR KUMAR

Roll - IRM2017008.

Page No.:

Date:

MDM Assignment-3

(Q1.) An emergency patient dispatch query can be ... Moflex transaction structure.

The mobile transaction model called Moflex, has 7 components :-

Moflex Transaction $T = \{M, S, F, D, H, J, G\}$

Where

$M = \{t_1, t_2 \ldots t_n\}$ where t_i are compensatable and noncompensatable subtransaction

S = a set of success-dependencies b/w t_i and t_j.

F = a set a failure dependencies to indicated that t_i can execute only after failure of t_j.

D = a set of external dependencies to indicated that t_i can execute iff it satisfies predefined external predicates. These predicates are defined on time(P), Cost(C) & location(L).

H = Set of handoff (control) rules that manage the exec. of subtransactions in the presence of a handoff.

J = Set of acceptable join rules to determine correct execution

of a subtransaction.
G = Set of all acceptable states of T.

An emergency patient dispatch query can be stated as follows. The objective of this is to illustrate how it fits into Moflex transaction. Find the right hospital or take the patient to the default hospital then dispatch patient status to the emergency doctor for getting the correct treatment. Here

$$M = \{t_1(C), t_2(C), t_3(NC), t_4(C), t_5(C)\}$$
$$S = \{t_1 <_s t_3,\ t_2 <_s t_3, t_1 <_s t_4\}$$
$$D = \{t_1, t_4\}$$
$$H = \{\text{restart}(t_1), \text{continue}(t_2), \text{continue}(t_3), \text{split-resume}(t_4), \text{continue}(t_5)\}$$
$$J = \{\text{user}(t_4)\}$$
$$G = \{(S, -, S, S, S), (-, S, S, -, S)\}$$

Here G,S indicates successful exec. and '–' means that state of subtransaction can be any, though other than the defined ones.

Here in the above model we must notes that :-

t_1 ⇒ Find the right hospital
t_2 ⇒ Take patient to default hospital
t_3 ⇒ Give patient status to emergency doctor
t_4 ⇒ Receive treatment
t_5 ⇒ Dispatch the patient.

(Q2) What is CODA file system? How does comm. perform in CODA file system?

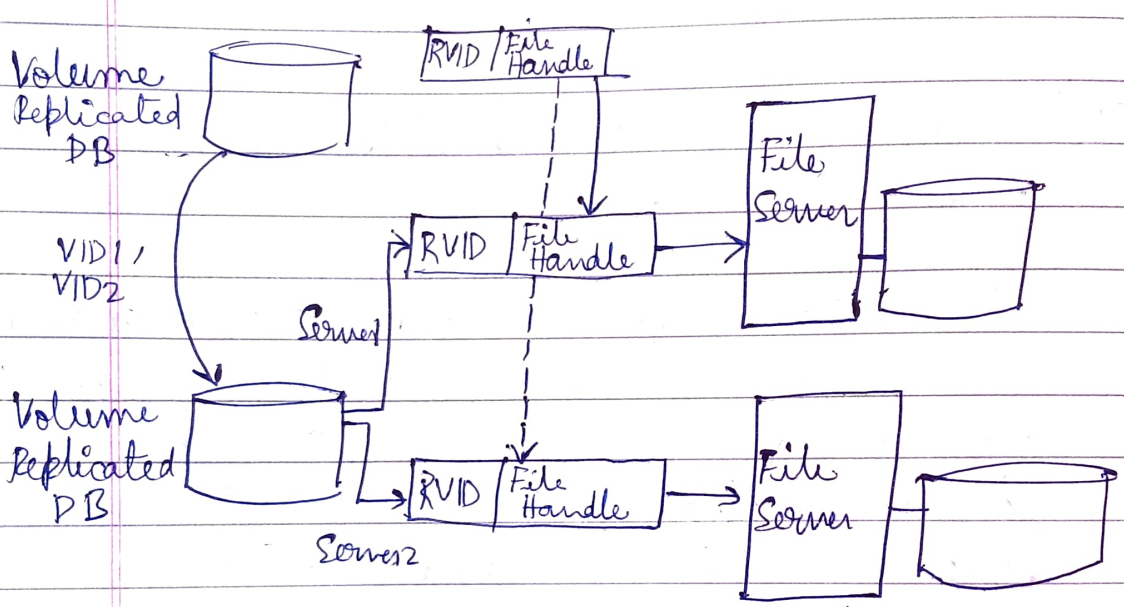
// Coda, stands for Constant Data Availability is a distributed file system that was developed as a research project at Carnegie Mellon University in 1987 under the direction of Mahadev Satyanarayan.

The major features that CODA file system attains are as follows :-

- Scalability
- Constant Data Availability
- Transparency
- Security
- Consistency.

Coda is hierarchically structured as in Unix and is partitioned into disjoint volumes, which consists of files & directories, and is the unit of replication.

Each file is identified by a 96 bit FID. Each file in Coda belongs to one volume, while volume may be replicated across several servers. Multiple logical volumes map to the same physical volume.



Coda file system works by implementing the following functionalities :-

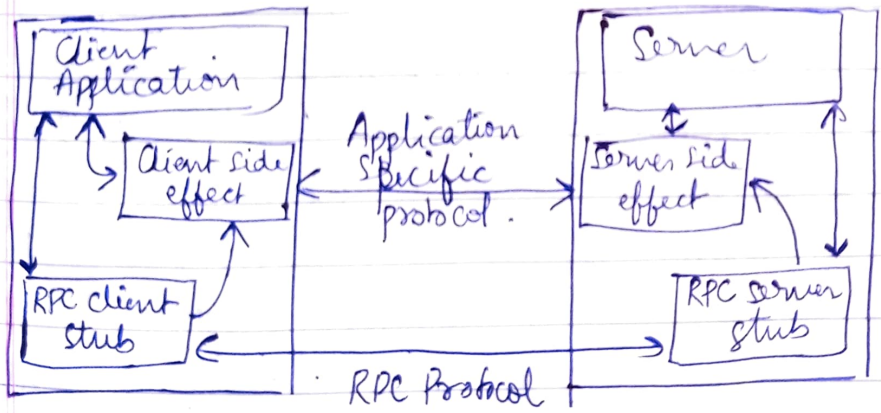
1) Availability of files by replicating a file volume across many servers

2) Disconnected mode of operation by caching files at the client machine

Page No :
Date :

Coda uses a local cache to provide access to server data when the network connection is lost. During normal operation, a user reads and writes to the file system normally, while the client fetches, or hoards all of the data the user has listed as important. If the network connection is lost, the Coda client's local cache servers data from this cache and logs all updates. Upon reconnection, it sends logged updates to the servers.

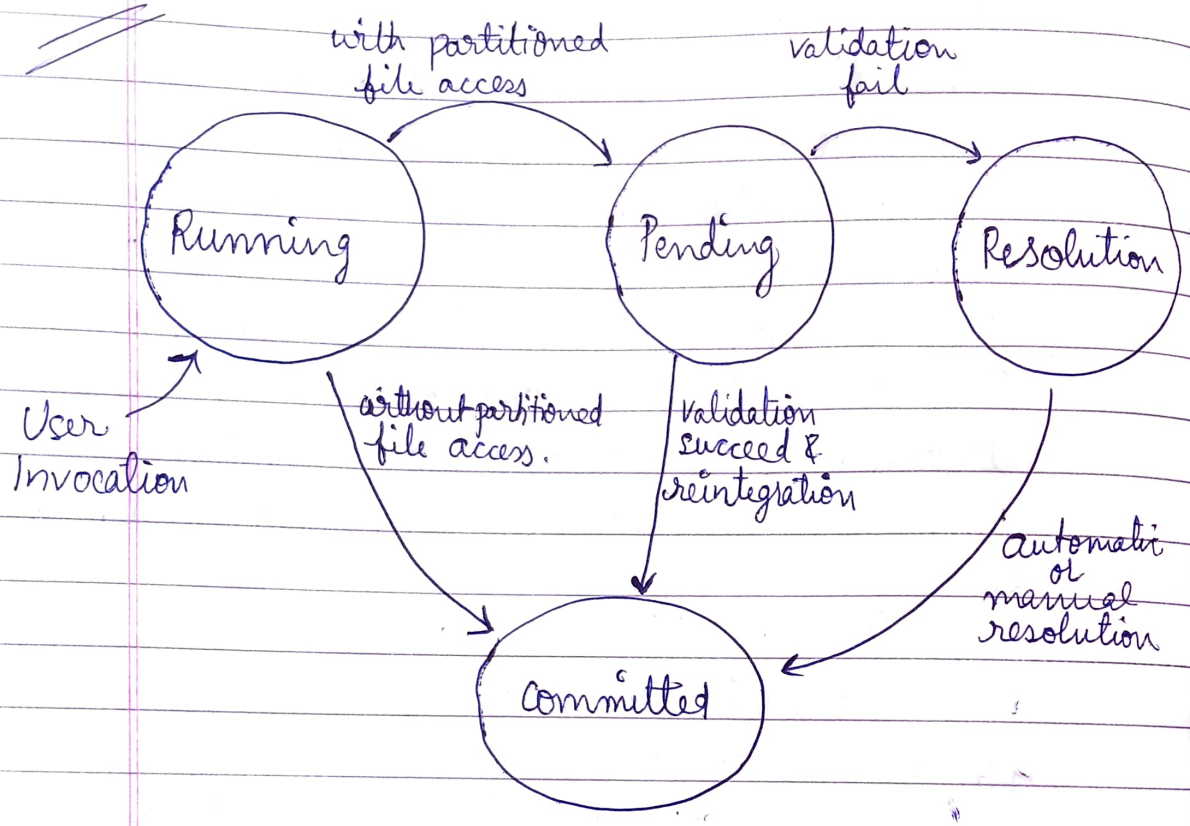
It uses RPC2, a sophisticated reliable RPC system, starts a new thread for each request, server periodically informs client it is still working on the request.



Modification by other clients are notified through invalidation messages which require multicast RPC.

Page No.
Date :

Q3 Draw transition diagram for Isolation only transaction in Code. Discuss ... transaction



An Isolation of Transaction is a flat sequence of file access operation bracketed by begin-iot and end-iot. An IOT provides strong consistency gurantees depending on the system connectivity conditions.

Transactions are classified into two categories : a first class transaction is one whose execution does not contain any partitioned file accesses.

Page No.
Date :

Otherwise, it is a second class transaction.

The execution of any first class transaction is guaranteed to be serializable with all committed transactions. The execution of any second class transaction is guaranteed to be serializable with other second class transactions executed on the same client.

Further we need to follow Global Serializability procedures and Global Certification Ordering for Second Class Transaction to ensure consistency of the system.

(Q4) Consider the architecture of a given mobile ... successfully.

The scenarios a transaction may encounter during execution, based upon the architecture of a mobile database systems can be read as follows :—

1) Mobile unit does not move, hence a transaction arrives and completes

its execution entirely on the same mobile unit. This is similar to conventional centralized data process.

(2) Mobile unit moves, hence a transaction arrives and completes its execution at different mobile unit. Required data items are moved from one node to the other, called 'Local Atomic' execution.

(3) Mobile unit moves and the transaction is distributed fragmented into subtransactions and these are distributed over the MUs and a set of databases.

To manage these situations, we may have two different approaches :- A static approach and a dynamic approach.

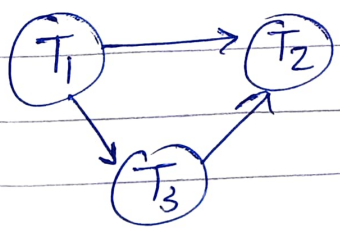
In static approach, when a transaction originates at a mobile unit, then the Base station of this MU becomes the coordinator of the transaction and remains the coordinator. Until the transaction commits

the MU may continue to migrate from one cell to another while processing its sub-transaction but the coordinator does not change. The MU moves from cell C1 to C2 with subtransaction C1 leaving behind the coordinator at BS1, which continue to manage the execution with the help of BS2.

In case of Dynamic Method, the role of coordinator moves with the Mobile unit. When mobile unit moves to cell C2, its base station BS2 becomes the coordinator of the transaction being executed by the MU. Since a transaction is being processed by multiple databases, they must know when a new coordinator is assigned to an existing transaction.

Q5

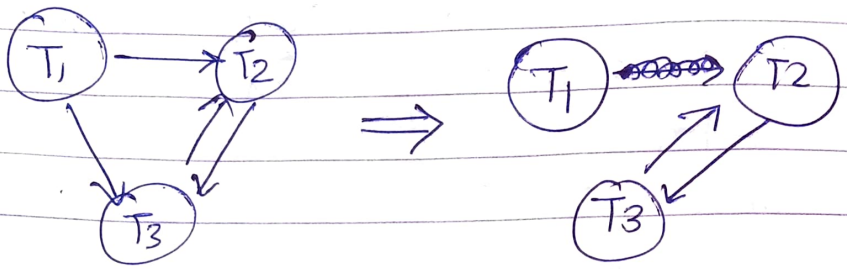
(i)



* No cyclic dependencies

⇒ No deadlock.

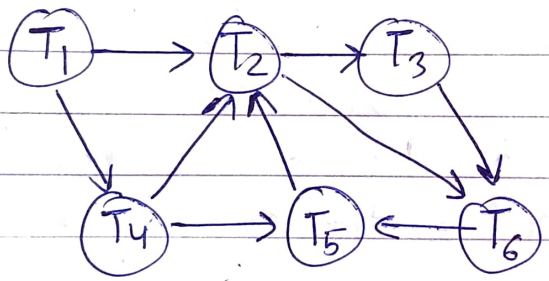
(ii)



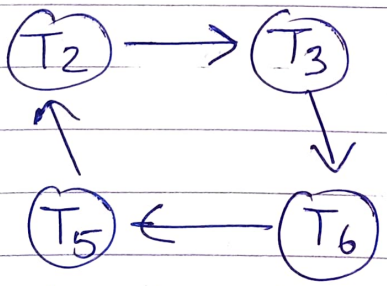
⇒ Cyclic dependency b/w T2 and T3

⇒ Deadlock.

(iii)



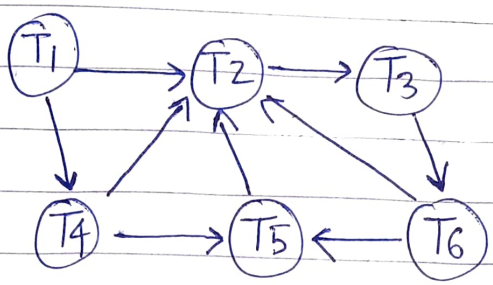
⇒



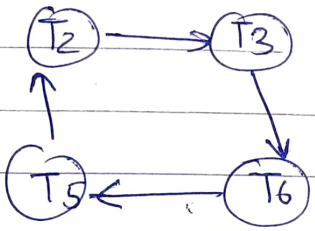
⇒ Cyclic dependency

⇒ Deadlock.

(iv)



⇒



⇒ Cyclic dependency

⇒ Deadlock.

(Q6) Develop your own mobile transaction model and a way of executing them on a mobile database system.

We propose a transaction model that would use some part of Time Based Consistency Model along with the Clustering Based Model. Here we assume a fully distributed system and the database to be divided into clusters, having mutually consistent data. Bounded inconsistencies are allowed to exist between clusters. These inconsistencies

may lead to a global conflict and hence we seek a Timestamp after which the clusters must be joined or merged for global consistency.

For any transaction, lets say T_i generated at Mobile unit MU_i, we divide the transaction into two parts:- Weak and Strict subtransactions. Weak Subtransaction are those which require data from the same cluster of which MU_i is a part of, and Strict Subtransaction are those that require data outside the cluster. We send the requests of strict transactions to the respective clusters, where they are executed locally and upon completion send a response of success. Upon completion of all weak transaction and upon receiving success from all strong subtransactions the cluster marks itself and all those clusters on which the strong sub-transactions were executed to be a part of the next timestamped global consistency based merging process.

Page No.:
Date:

Further as in the Time Based Model, we must have a Modification Time Bound upon each cluster that has performed any modification in data. This will be helpful in case of network partitioning and communication delay, that when global merge protocol is not executed at a cluster under its Modification Time Bound, it may recognise either network partitioning in which case it may rollback and send error message for all recently executed transactions until the rollback, or in case of network traffic, it may forcefully call for a merging of the clusters, so that the change log does not surpass a limit.