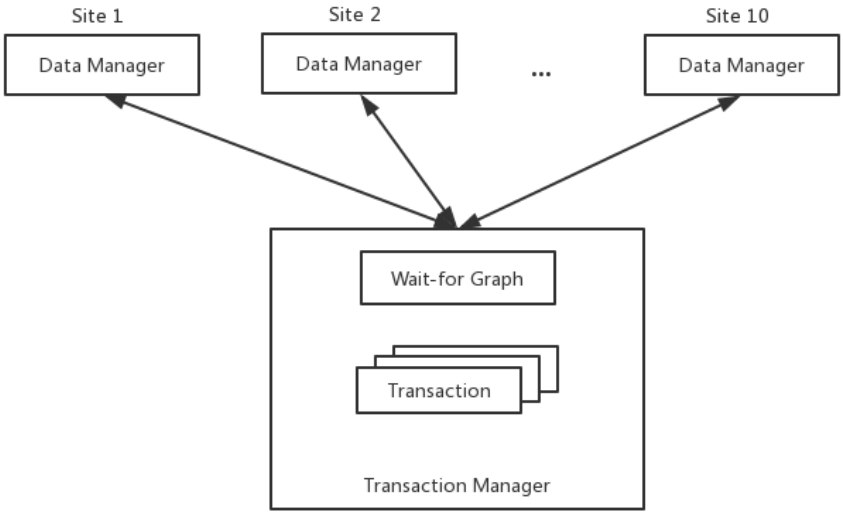
RepCRec Design Doc

Fall 2019 Advanced Database Course Project
Github repo(private): https://github.com/iamnwi/adv-db-project

Author

Jialiang Cao (jc8343)
Wai I Ng (win205)

System Architecture



Programming Language

Java

Classes

Class Main

- Functions:
    - init
        - Description: initialize sites and a TM

        - Input: void
        - Output: void
    - run
        - Description: run this distributed database system
        - Input: transaction instructions
        - Output: void

Class TransactionManager

- Fields:
    - Command Buffer: A queue for accumulated command in orders
        - FIFO queue
        - At each tick, add new instruction append to the end
        - Execute commands in the buffer in orders
    - DM list: Array<DM>
    - Site status table:
        - Format: Map<site id, Tuple({down/up}, last down-time(used for commit validation))>
    - Pointer to the next site for access
        - Purpose: balance the load for even indexed variable
        - Format: An integer, the next site number
        - How: This pointer points to the site that should be accessed in the current tick, if this pointed site is down, move the pointer to the next site until the pointer site is up. After this tick, move the pointer to the next site then go to the next tick
    - Check deadlock(bool): A flag to tell if a dead-lock detection is needed at the beginning of next tick
        - Set to true if the instruction at the current tick is blocked. Otherwise set to false. Set it at every tick, or set true when blocked and set false when detection run.
    - Wait-for graph(for deck-lock detection)
    - Time(integer)
        - Init to 0, add one when a newline is encountered
    - Transaction list
        - Format: array<Transaction>
- Functions:
    - init
        - Description: initialize all fields
        - Input: sites' DM objects
        - Output: void
    - run
        - Description: run all the instructions in a given text
        - Input: instructions text

    - Output: void
    - Side effect:
        - Add one tick to time when meeting a newline
        - Parse the instructions text line by line
        - Append parsed instructions into command buffer one by one
        - Perform the instructions in the command buffer in orders at each tick
- parse
    - Description: Parse a given instruction
    - Input: one instruction in string
    - Output: name of instruction and arguments
- begin
    - Description: handle transaction begin instruction
    - Input: transaction name
    - Output: void
    - Side effect:
        - Create Transaction with begin time and type, append to TM transaction list
- beginRO
    - Description: handle read-only transaction begin instruction
    - Input: transaction name
    - Output: void
    - Side effect:
        - Create Transaction with begin time and type(RO), append to TM transaction list
        - Notify all up sites to create a snapshot with a given timestamp(the begin time of this Transaction)
        - Append to the command buffer if blocked(all sites are down, no one can create a snapshot)
- read
    - Description: handle transaction read data instruction
    - Input: transaction name, variable name
    - Output: void
    - Side effect:
        - If acquire read lock success:
            - Notify the accessed site to update its locks table
            - Print the variable value in the format “x{number}: {val}”
            - Add accessed site to Transaction’s access sites set
        - If fail(no sites can provide a read lock for the target variable):
            - Add one edge to the wait-for graph if it is a read-write Transaction
            - Append to the command buffer if blocked

            - Set check deck-lock to true to perform a deadlock at next tick
        - Update the pointer of next available site
- write
    - Description: handle transaction write data instruction
    - Input: transaction name, variable name, the new variable value
    - Output: void
    - Side effect:
        - Append this command to Transaction's write command
        - If acquire write locks success:
            - Notify all up sites to update its locks table
            - Add accessed site to Transaction's access sites set
        - If fail:
            - Add edge(s) to the wait-for graph
            - Append to the command buffer
            - Set check deck-lock to true to perform a deadlock at next tick
        - Update the pointer of next available site
- dump
    - Description: handle transaction write data instruction
    - Input: N/A
    - Output: void
    - Side effect:
        - Print data tables of all sites(no matter down or up)
- end
    - Description: Perform a commit validation to see if the given transaction can be committed. If it is a read-write transaction, release all the locks it holds.
    - Input: transaction id
    - Output: void
    - Side effect:
        - Print whether the given transaction is committed or aborted.
        - Update lock tables of sites that the given transaction accessed.
        - Update wait-for graph(remove edges related to the given transaction)
        - If can commit, update data tables of sites that the given transaction accessed.
- fail
    - Description: Mimic the situation that the given site is down
    - Input: site id
    - Output: void
    - Side effect:

            - Set the status of the given site as down in the site status table
            - Notify the given site to down
    - recover
        - Description: Mimic the situation that the given site is recovered from a failure
        - Input: site index
        - Output: void
        - Side effect:
            - Set the status of the given site as up
            - Notify the given site to recover
    - querystate
        - Description: Print out all status of TM and each DM
        - Input: void
        - Output: void
        - Side effect:
            - Print all status in TM
            - Print all status and data in each DM
    - detectDeadLock
        - Description: Perform deadlock detection and abort transactions if needed
        - Input: void
        - Output: void
        - Side effect:
            - If a deadlock is detected
                - abort the younger transaction
                - update the wait-for graph

Class Transaction

- Fields
    - Begin time(Integer)
    - Type: {Read-Only | Read-Write}
    - Transaction access list(what sites a T accessed)
        - Format: Transaction name | List of sites the transaction accessed
    - Write Commands
- Functions
    - init
        - Description: initialize all fields
        - Input: void
        - Output: void

Class WaitForGraph

- Fields:
    - Edges: Array<(T1, T2, Lock Type, Next Edge)> // T1 waits for a lock of T2

        - Vertices: Array<T, Start Edge>
- Functions:
    - init
        - Description: initialize all fields
        - Input: void
        - Output: void
    - addEdge
        - Description: Add one new wait-for edge into graph
        - Input: (S, E) edge from S to E
    - detectCycle
        - Description: Detect if cycle exists in current graph. If so, return which transaction to abort. Otherwise, return -1.
        - Input: empty
        - Output: null if no cycle detected, or some transaction ID T if cycle does exist.
    - removeEdges
        - Description: Abort all edges related to some transaction T.
        - Input: Transaction ID T.

Class DataManager

- Fields:
    - Data table
    - Snapshots for multi-version read
        - A queue for versions of each variable (all variables on that site, even indexed and some odd indexed)
        - Format: Map<time, Map<var id, var val>>
    - Lock table
        - Format: Array<Tuple(Lock type, transaction id)>
    - Replication variable readable status table
        - Format: Array<bool>
    -
- Functions:
    - init
        - Description: initialize site's data
        - Input: void
        - Output: void
    - checkLock
        - Description: check if the required lock can be acquired
        - Input: transaction id, variable id, lock type
        - Output: true if the transaction already had that lock or it had a more powerful lock. Otherwise false
    - acquireLock

        - Description: acquired the required lock if possible
        - Input: variable id, lock type, transaction id
        - Output: bool
        - Side effect:
            - Update lock table if the required lock can be acquired
- fail
    - Description: Mimic the situation that the given site is down
    - Input: void
    - Output: void
    - Side effect:
        - Erase the lock table
- recover
    - Description: Mimic the situation that the given site is recover from down
    - Input: void
    - Output: void
    - Side effect:
        - Set replication variables(even indexed variables) to be non-readable
- read
    - Description: Handle transaction's read instruction
    - Input:
        - variable ID
        - transaction type
        - transaction begin time
    - Output:
        - null if
            - site is down
            - variable is not available to read for read-write transaction
            - no corresponding snapshot for read-only transaction
        - an integer as variable value
- write
    - Description: Handle transaction's write instruction
    - Input:
        - variable ID
        - variable new value
    - Output:
        - true if write completes successfully
        - false if site is down
    - Side effect:
        - Change the value of current variable to new value
        - If current variable is non-readable, set it to readable
- queryState
    - Description: Print out current state of current DM (site)

- Input: empty
- Output: void
- Side effect:
    - Print out variable data table
    - Print out multiversion-read data table
    - Print out lock table
    - Print out replicated variable read status table