Advanced Database Systems

Final Project Design Document

Shubham Srivastava
NetID: ss14687

Adit Kotwal
NetID: ask9100

Goals:

Track: Distributed Replicated Concurrency Control and Recovery

Key Highlights:

We will implement a distributed database with:

1. Concurrency Control
2. Deadlock Detection
3. Replication
4. Failure Recovery

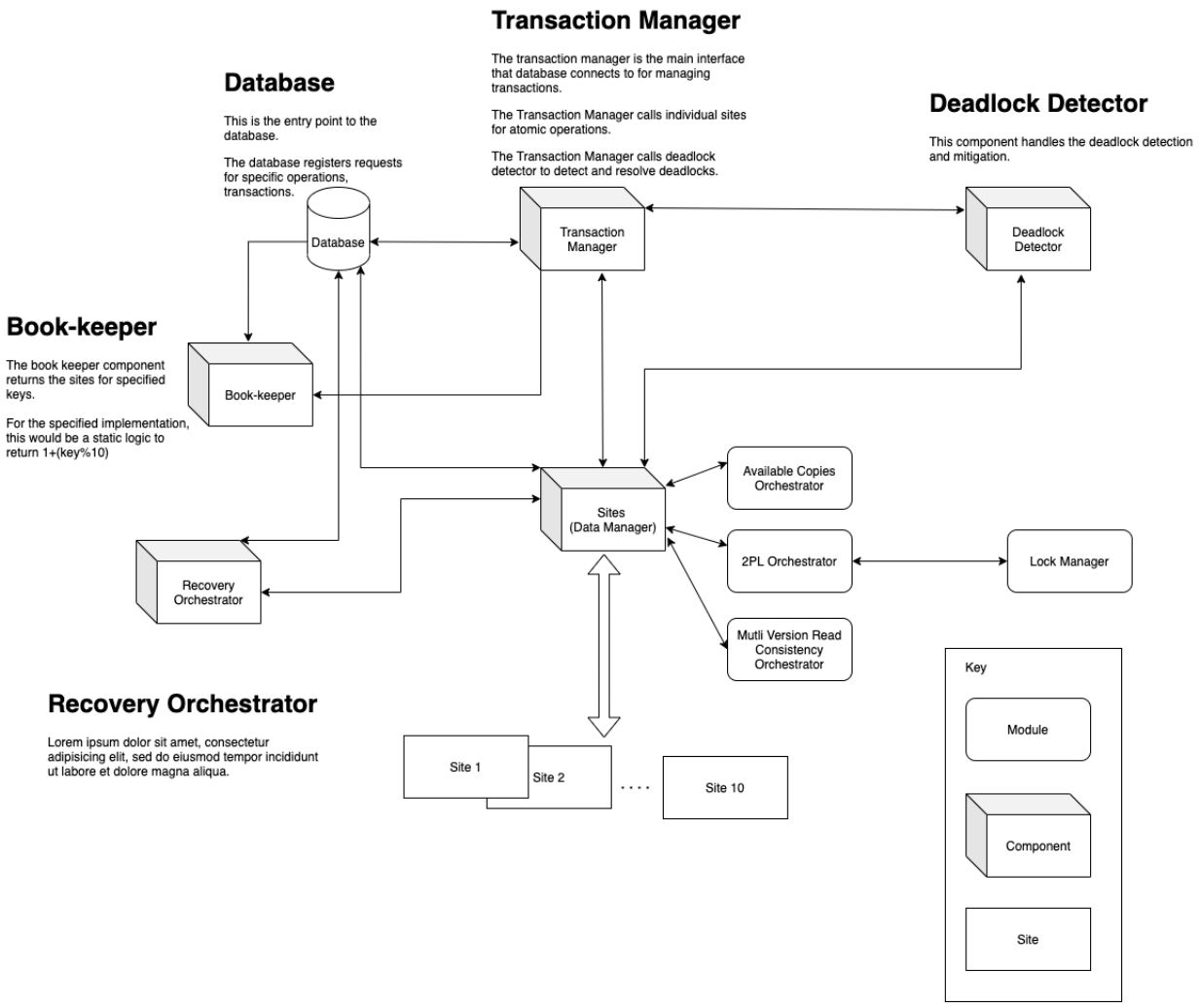
Algorithms to be used:

1. We will implement the available copies approach to replication using strict two-phase locking (using read and write locks) at each site and validation at commit time.
2. Deadlock detection using cycle detection. Deadlock mitigation by aborting the youngest transaction.
3. Read-only transactions should use multi-version read consistency.

Components in the database:

1. Database:
   - Entry point to the database
   - Registers all the requests to the database
   - Contacts transaction manager for handling transactions, other operations, and deadlock detection

2. Data Manager or Site:
   - Each site holds a set of records (versioned)
   - Each site takes care of its own 2PL
   - Each site takes care of its own Available copies algorithm
   - Each site takes care of read only operations through multi-version read consistency
   - Each site has its own lockmanager that facilitates locking
3. Deadlock Detector:
   - Handles deadlock detection through cycle detection in waits-for graph
   - Deadlock mitigation by removal of youngest transaction in the cycle
4. Bookkeeper:
   - Handles all the book keeping (key to sites and replicas)
5. Recovery Orchestrator:
   - Handles recovery of a site in case a site fails
   - **UPDATE: for simplicity of the implementation, this has been implemented along with the transaction manager itself**



The above document shows the high-level interaction between components.

In the page below, you would have the details for classes and methods. I have avoided mentioning additional details (around return types / comments) to remain within the size limit of the doc by naming methods accordingly.

Main Classes and Functions:

Operation:

- type (READ | WRITE) -> this now has other types BEGIN, BEGINRO, …
- key
- value

Transaction:

- List<Operation> operations
- status (PENDING | ACTIVE | COMMITTED) -> ABORT | ABORTED
- timestamps

Site:

- executeOperation(Operation x) (read, write)
- executeTransaction(Transaction t)
- failSite()
- recoverSite()
- getSiteState() # for debugging and demo
- registerLockManager()
- getActiveTransactions() #returns list of transactions
- getWaitsForGraph() # pass through for the method in lock manager
- commitValues()
- rollback()
- readAllowed() | writeAllowed()

LockManager:

- public boolean isReadAllowedForVariable(Transaction transaction, Integer variable);
- public boolean isWriteAllowedForVariable(Transaction transaction, Integer variable);
- public void acquireReadLock(Transaction transaction, Integer variable);
- public void acquireWriteLock(Transaction transaction, Integer variable);
- public Optional<Transaction> waitsForWriteTransaction(Integer variable);
- public List<Transaction> waitsForReadTransaction(Integer variable);
- public void clearAllLocks();
- public void releaseLocksForTransaction(Transaction transaction);
- public Set<Integer> getAllVariablesHeldByTransaction(Transaction transaction);
- public Set<Integer> getWriteVariablesHeldByTransaction(Transaction transaction);
- public Map<Integer, List<Transaction>> getReadLocks();
- public Map<Integer, Transaction> getWriteLocks();

DeadlockDetector:

- detectDeadlocks()
- findYoungestTransaction()
- abortYoungestTransaction()

Bookkeeper:

- getSiteForRecord(key) #returns Site(s) -> implementation using odd variable, even variable.