Advanced Operating System

Unit – IV

Dr.M.Lalli

Dept of CS, Trichy

Unit - 4

Database operating systems:

- Requirements of Database OS
- Transaction Process model
- Synchronization primitives
- Concurrency control algorithms

Database Operating Systems

- Database system have been implemented as an application on top of general purpose OS
- Requrements of DBOS
  - Transaction Management
  - Support for complex, persistent data
  - Buffer Management

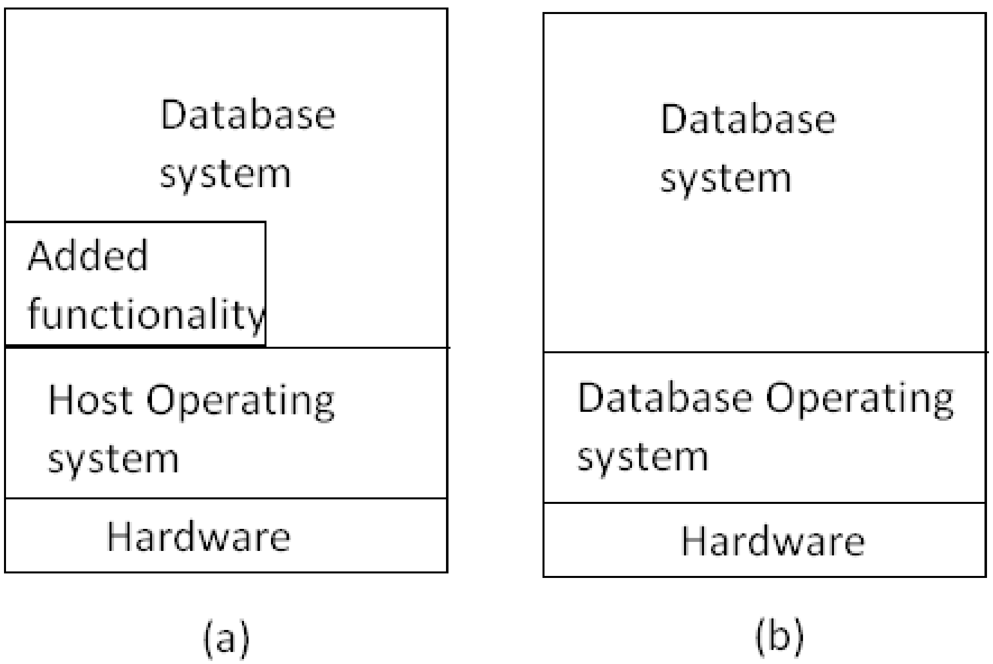


Fig: Two approaches to database systems design

Distributed Database System

- A distributed database is a database in which storage devices are not all attached to a common processing unit such as the CPU.

- It may be stored in multiple computers, located in the same physical location; or may be dispersed over a network of interconnected computers.

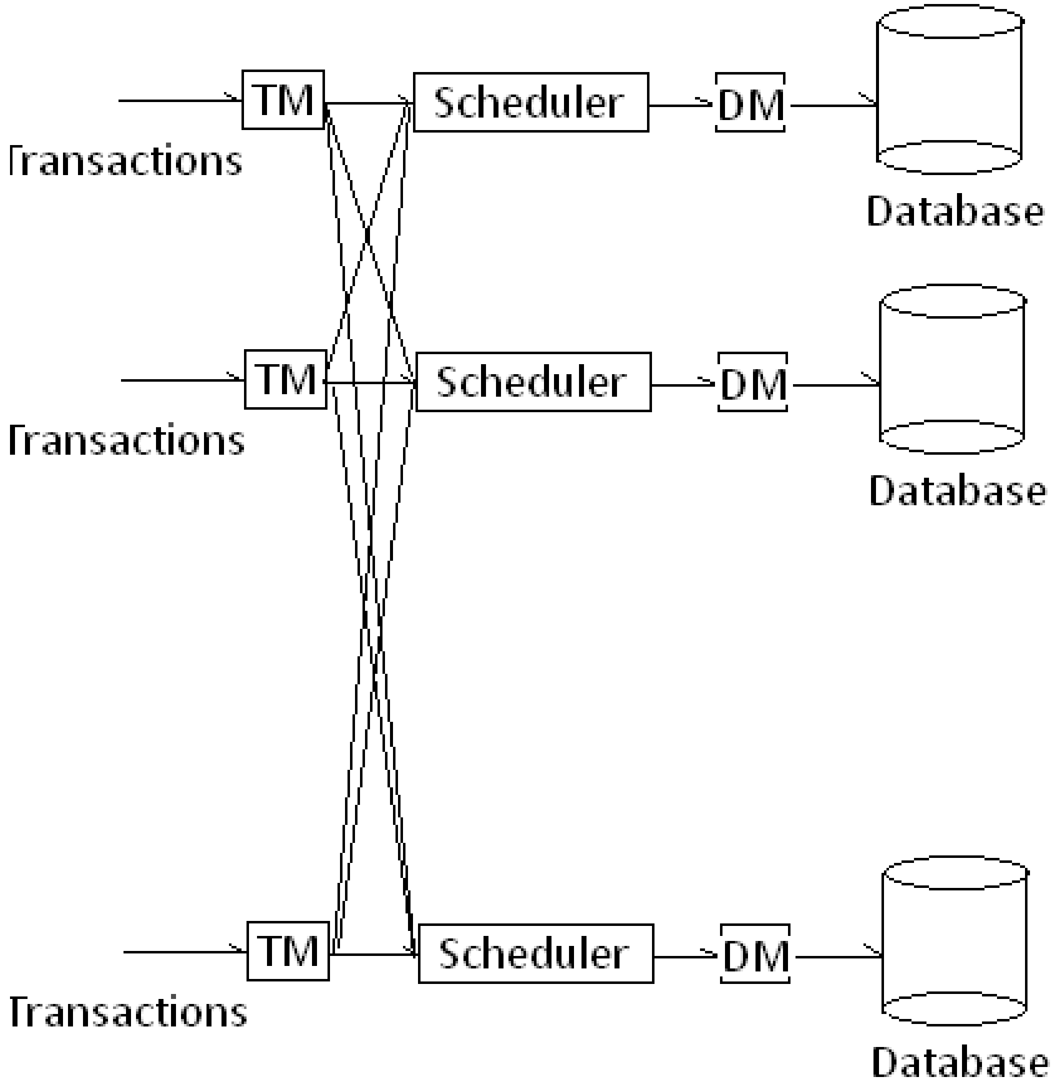
- Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely coupled sites that share no physical components.

Distributed Database System

- Motivations: DDBS offers several advantages over a centralized database system such as
  - Sharing
  - Higher system availability (reliability)
  - Improved performance
  - Easy expandability
  - Large databases
- Transaction Processing Model
- Serializability condition in DDBS
- Data replication
- Complications due to Data replication
- Fully Replicated Database Systems

1. Enhanced reliability 2. Improved responsiveness 3. No directory management 4. Easier load balancing

Model of Distributed Database System

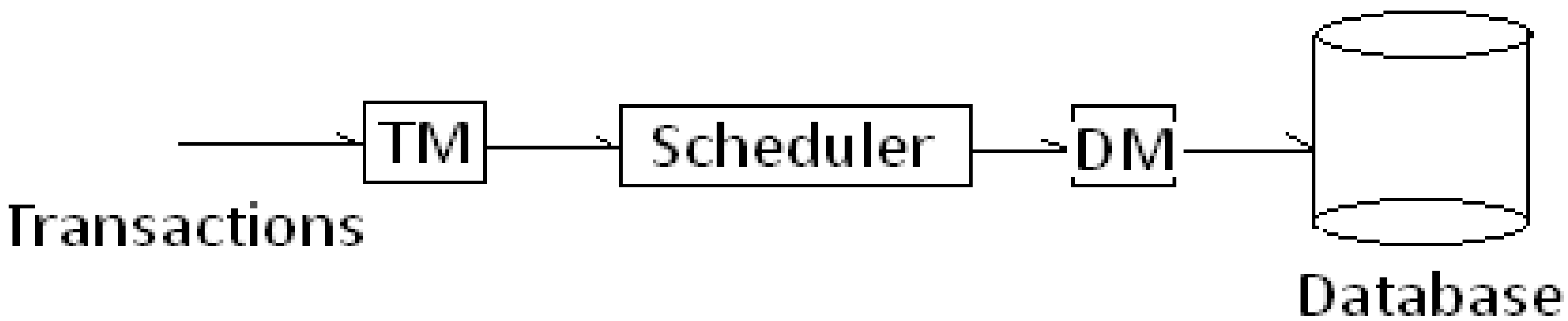


A concurrency control model of DBS

- 3 software modules
  - Transaction manager (TM)

    Supervises the execution of a transaction
  - Data manager (DM)

    Responsible for enforcing concurrency control
  - Scheduler



Concurrency Control

- CC is the process of controlling concurrent access to a database to ensure that the correctness of the database is maintained.
- Database systems

Set of shared data objects that can be accessed by users.

- Transactions

A transaction consists of a sequence of R, compute & W s/m that refer to the data objects of a database.

- Conflicts

Transactions conflicts if they access the same data objects.

- Transaction processing

A transaction is executed by executing its actions one by one from the beginning to the end.

Concurrency Control Algorithms

- It controls the interleaving of conflicting actions of transactions so that the integrity of a database is maintained, i.e., their net effect is a serial execution.
- Basic synchronization primitives
- Locks
  - A transaction can request, hold or release the lock on a data object.
  - lock a data object in 2 modes: exclusive and shared
- Timestamps
  - Unique number is assigned to a transaction or a data object and is chosen from a monotonically increasing sequence.
  - Commonly generated using Lamport's scheme

Lock based algorithms

- ❑ Static locking
- ❑ Two Phase Locking (2PL)
- ❑ Problems with 2PL: Price for Higher concurrency
- ❑ 2PL in DDBS
- ❑ Timestamp Based locking
- ❑ Conflict Resolution
- ✓ Wait Restart Die Wound
- ❑ Non-two-phase locking

Timestamp Based Algorithms

- Basic timestamp ordering algorithm
- Thomas Write Rule (TWR)
- Multiversion timestamp ordering algorithm
- Conservative timestamp ordering algorithm

Optimistic Algorithms & Concurrency Control Algorithms

- Optimistic Algorithms
  - Kung Robinson Algorithm
- Concurrency Control Algorithms
  - Completely Centralized Algorithm
  - Centralized Locking Algorithm
  - INGRES' Primary-Site Locking Algorithm
  - Two-Phase Locking Algorithm

Optimistic Algorithms

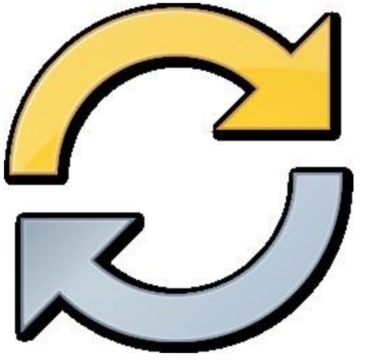
- Based on the assumption

“conflicts do not occur during executiontime”

Optimistic Algorithms

- Thus,

No synchronization is performed when a transaction is executed.



Optimistic Algorithms

- However,

A check is performed at the end of the transaction (to ensure no conflicts have occurred)

Optimistic Algorithms

- If conflict == true
  - Transaction aborted
- Else
  - Commit Transaction

Optimistic Algorithms

- Since conflicts do not occur very often, this algorithm is very efficient compared to other locking algorithms.

Kung-Robinson Algorithm

- H. T. *KUNG* and JOHN T. *ROBINSON* were the first to propose an optimistic method for concurrency control.
- The optimistic situation for this algorithm happens when conflicts are unlikely to happen; the system consists mainly read-only transactions (such as a query dominant (powerful) system)
- Basic Idea: No synchronization check is performed during transaction processing time, however, a validation is performed to make sure that no conflicts occurred. If a conflict is found, the tentative write is discarded and the transaction is restarted.

The Algorithm (Kung-Robinson)

- Divided into three phases :

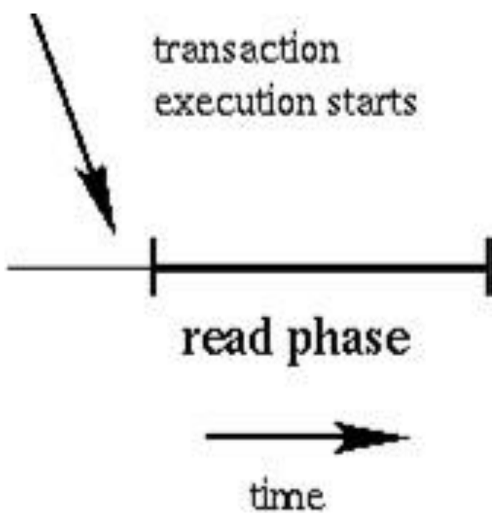
Read Phase Validation Phase Write Phase

The Algorithm (Kung-Robinson)

- Divided into three phases :

Read Phase Validation Phase Write Phase

data objects are read, the intended computation of the transaction is done, and writes are made on a **temporary** storage.

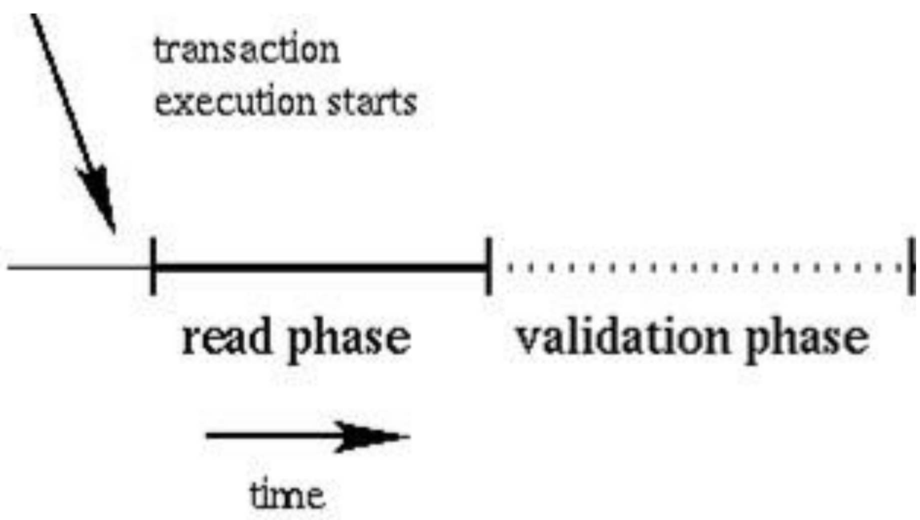


The Algorithm (Kung-Robinson)

- Divided into three phases :

Read Phase | Validation Phase | Write Phase

check to see if writes made by the transaction violate the consistency of the database.

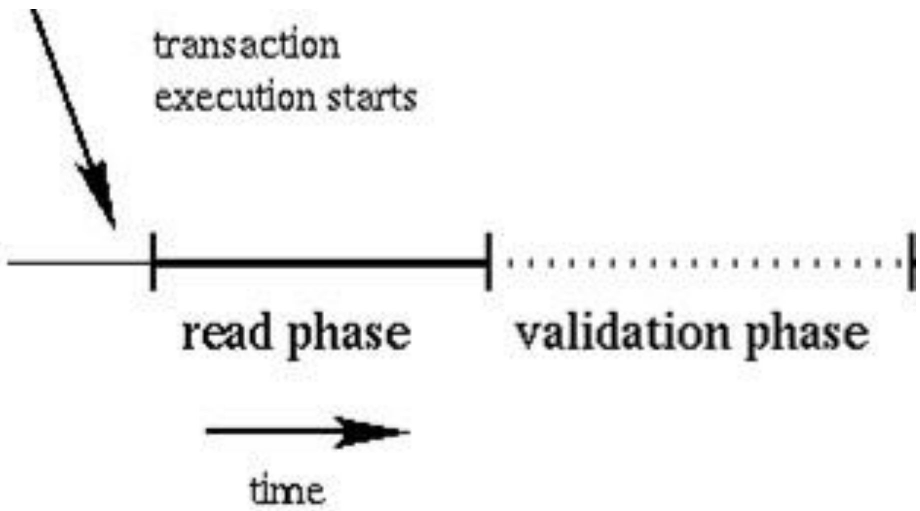


The Algorithm (Kung-Robinson)

- Divided into three phases :

Read Phase Validation Phase Write Phase

If the check finds out any conflicts, the data in the temporary storage will be discarded. Otherwise, the write phase will write the data into the **database**.

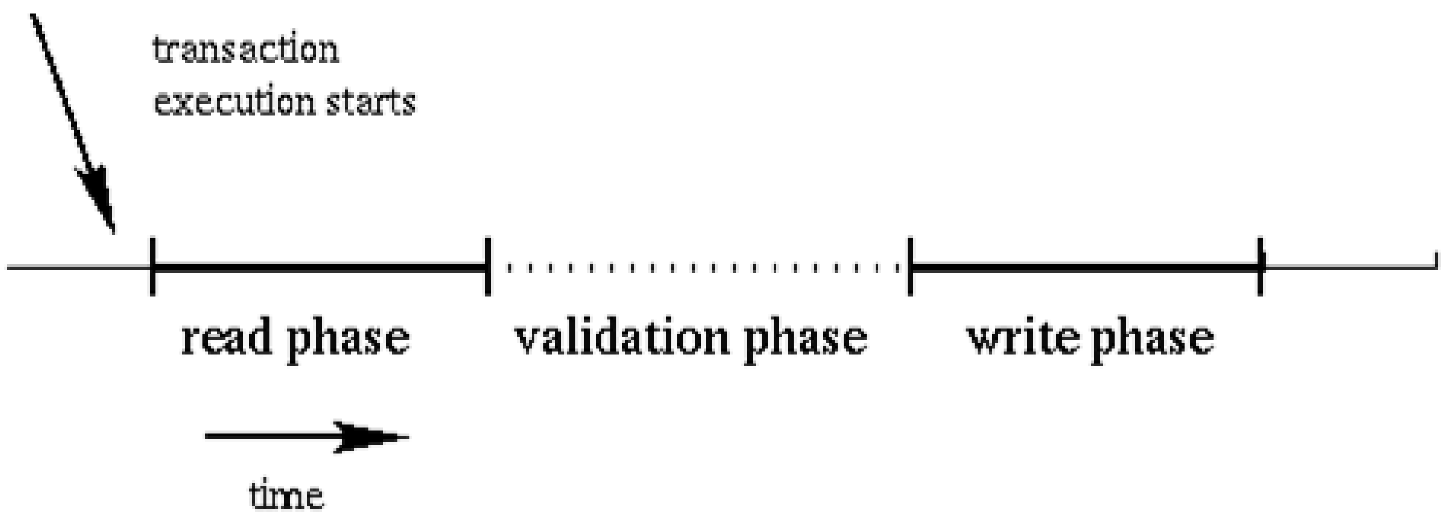


The Algorithm (Kung-Robinson)

- Divided into three phases :

Read Phase Validation Phase Write Phase

If the validation phase passes ok, write will be performed to the database. If the validation phase fails to pass, all temporary written data will be aborted.



The Algorithm (Kung-Robinson)

- Validation Phase (can be described as)
  - T: a transaction
  - ts: the highest sequence number at the start of T
  - tf: the highest sequence number at the beginning of its validation phase

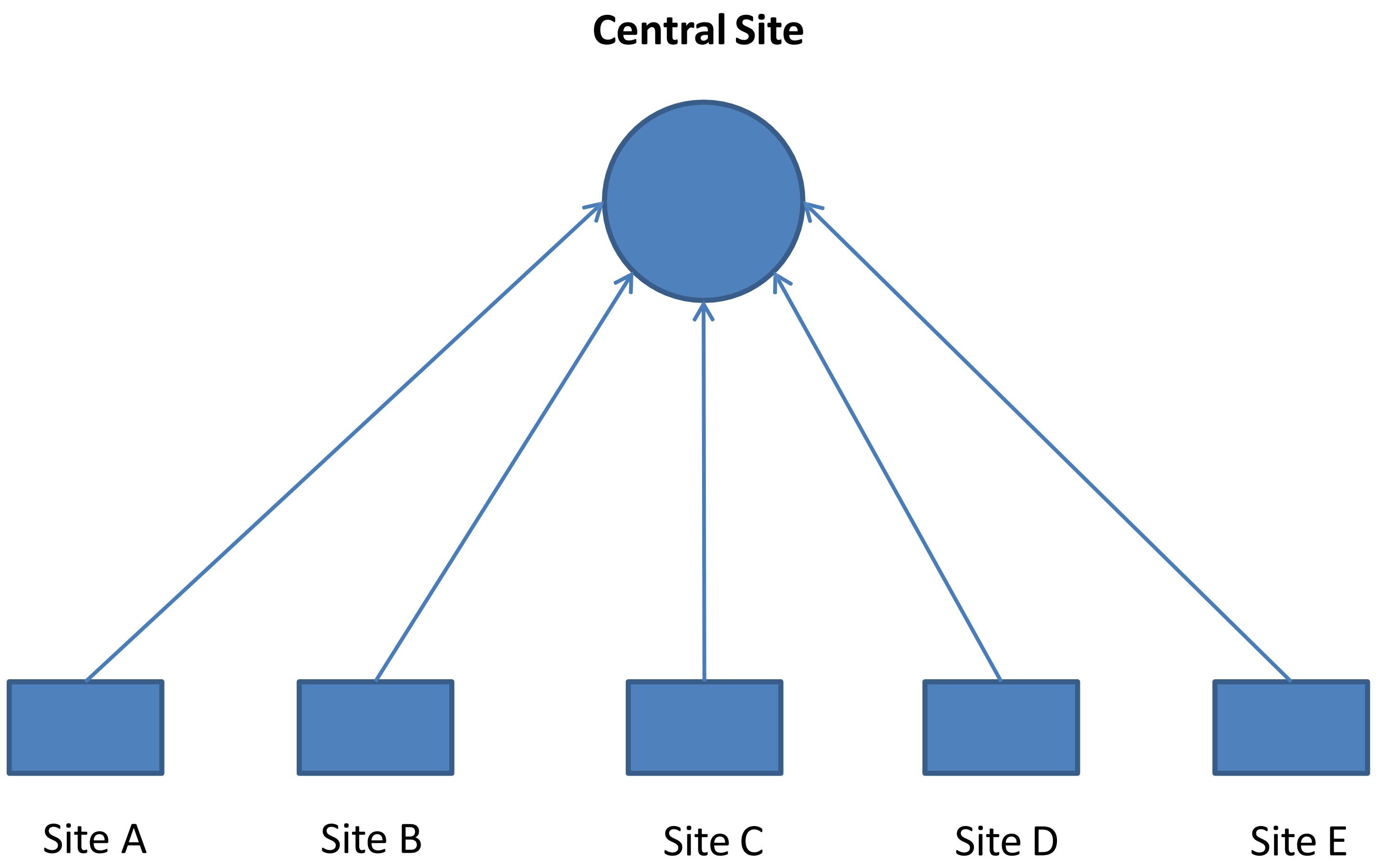
```
valid:=true;
for t:=ts+1 to tf do
        if (writeset(t) & readset[T] != {} ) then
                valid :=false;
        if valid then {write phase; increment counter, assign T a
        sequence number}
```

Concurrency Control Algorithms

- Fully replicated database systems i.e.

"data objects are replicated at all sites"

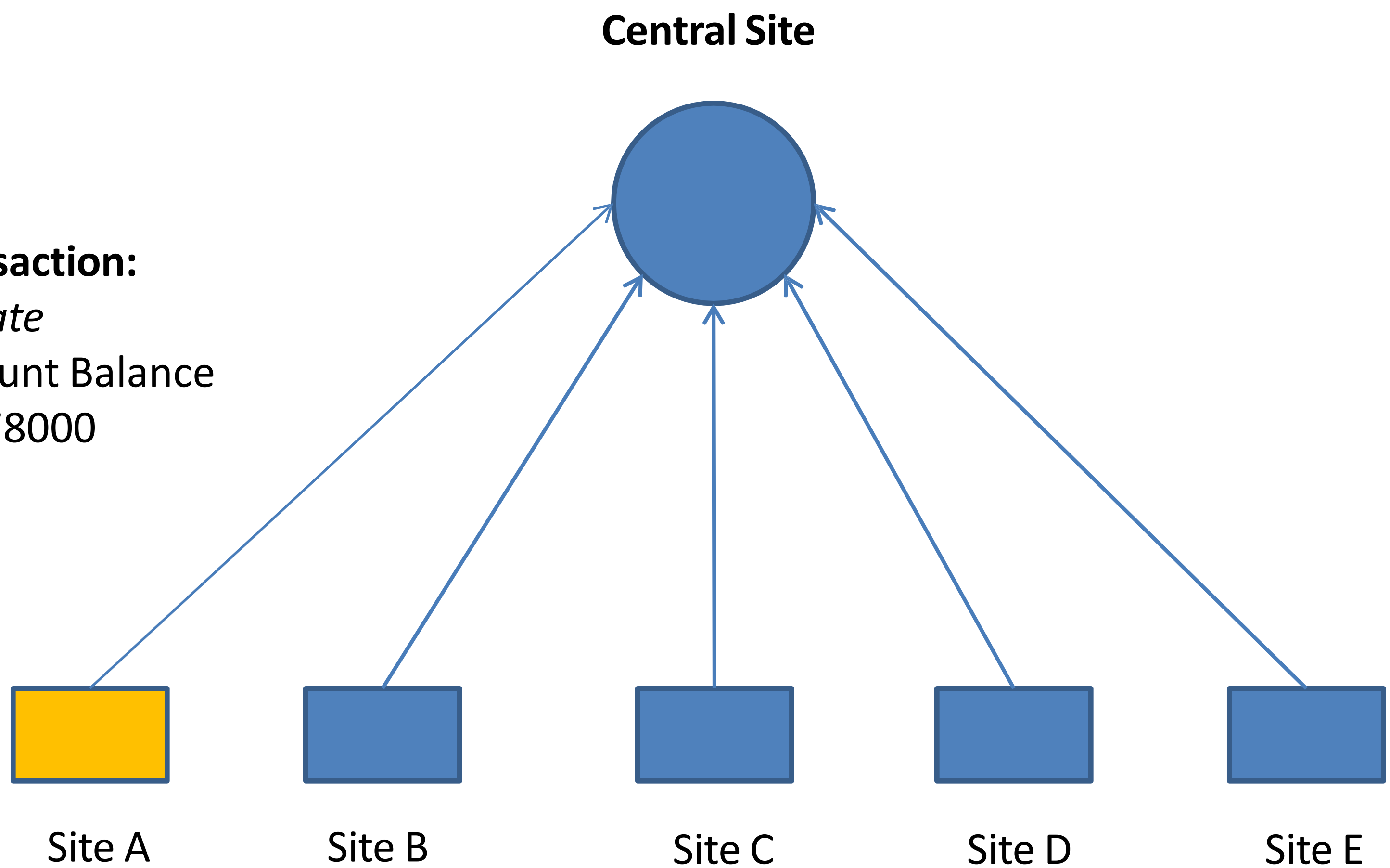
Completely Centralized Algorithm



A site is designated as the central site.

Completely Centralized Algorithm

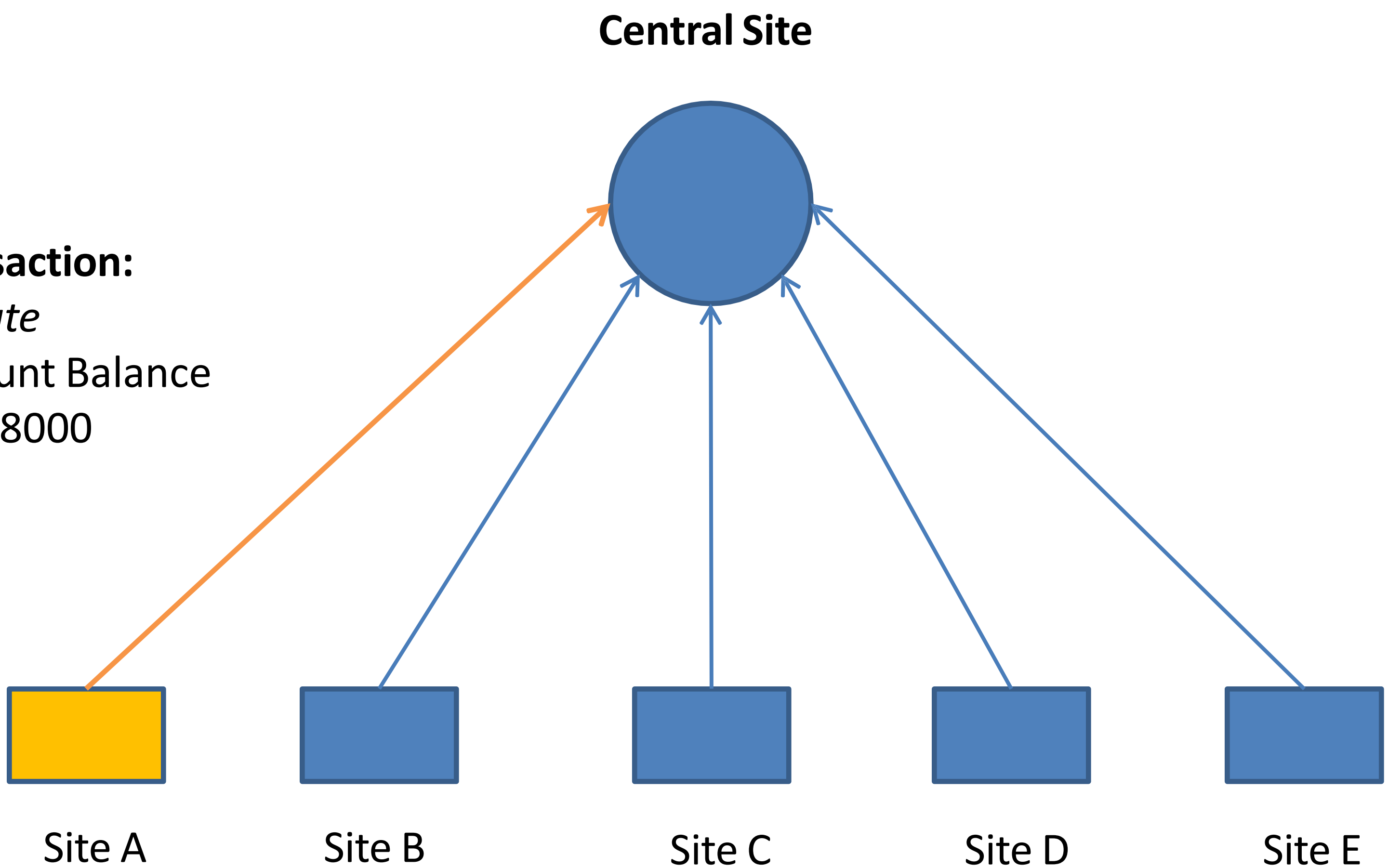
Transaction:
Update
Account Balance
#B=78000



Transaction at site A occurs.

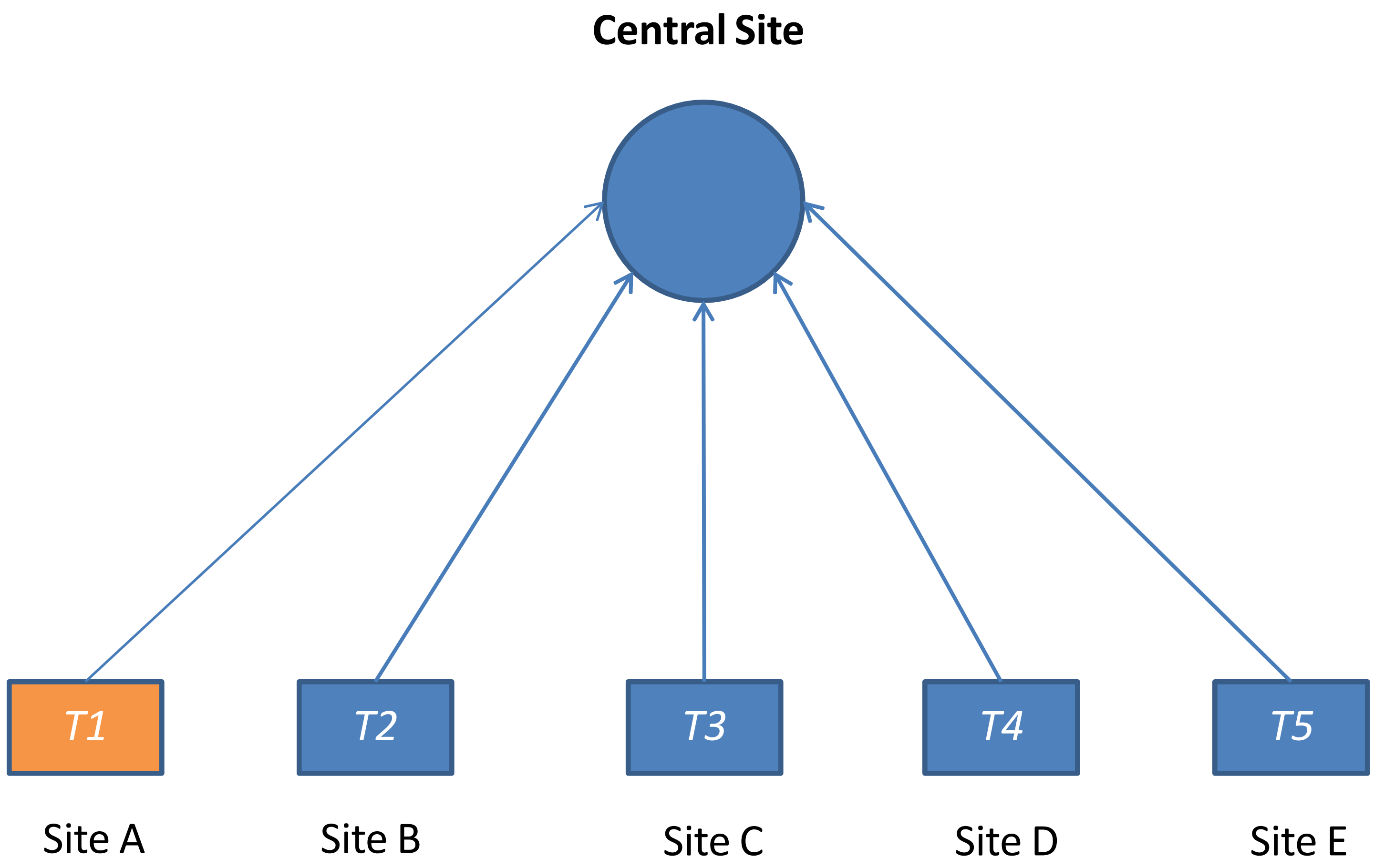
Completely Centralized Algorithm

Transaction:
Update
Account Balance
#B=78000



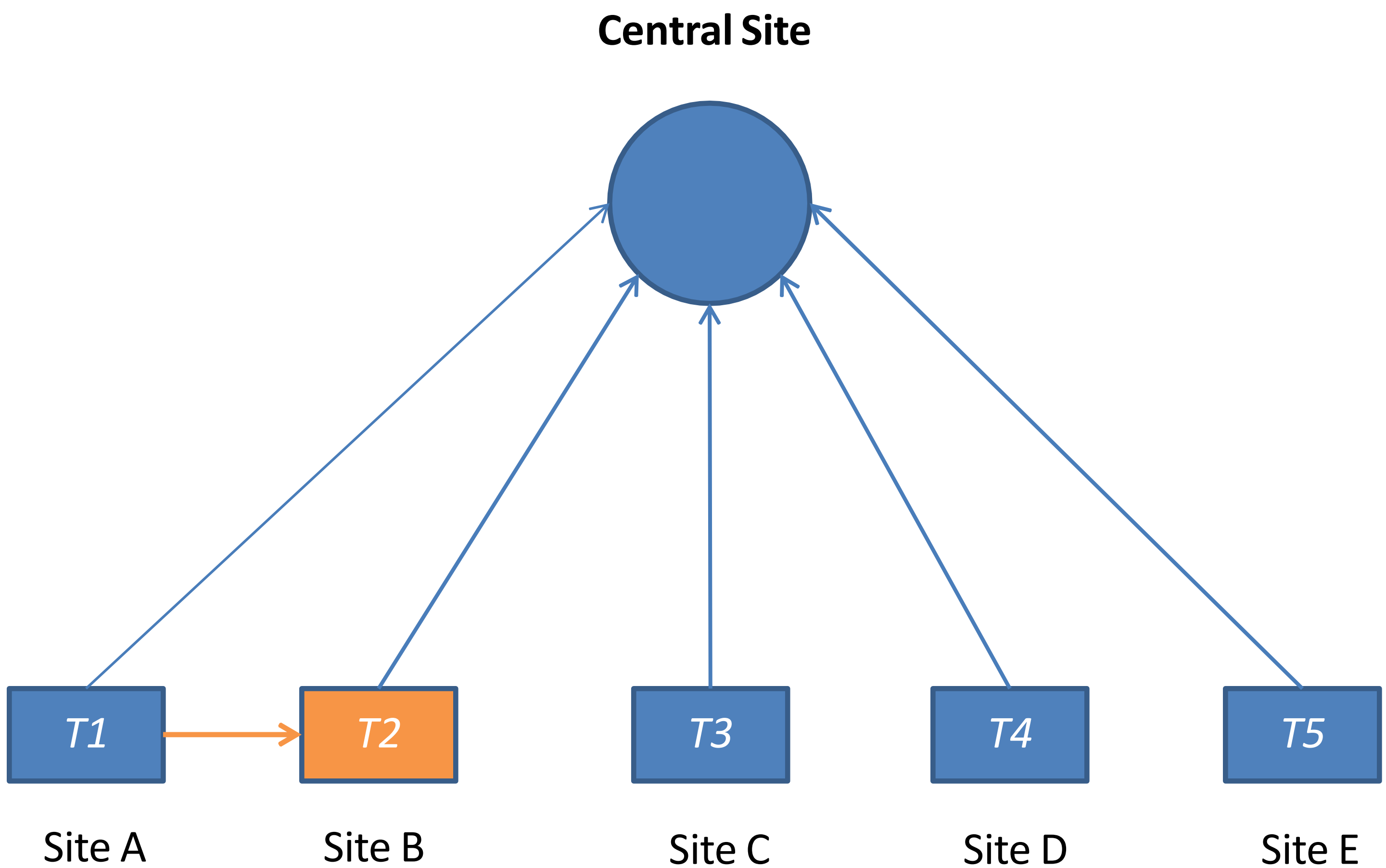
Transaction is forward to the central site for execution.

Completely Centralized Algorithm



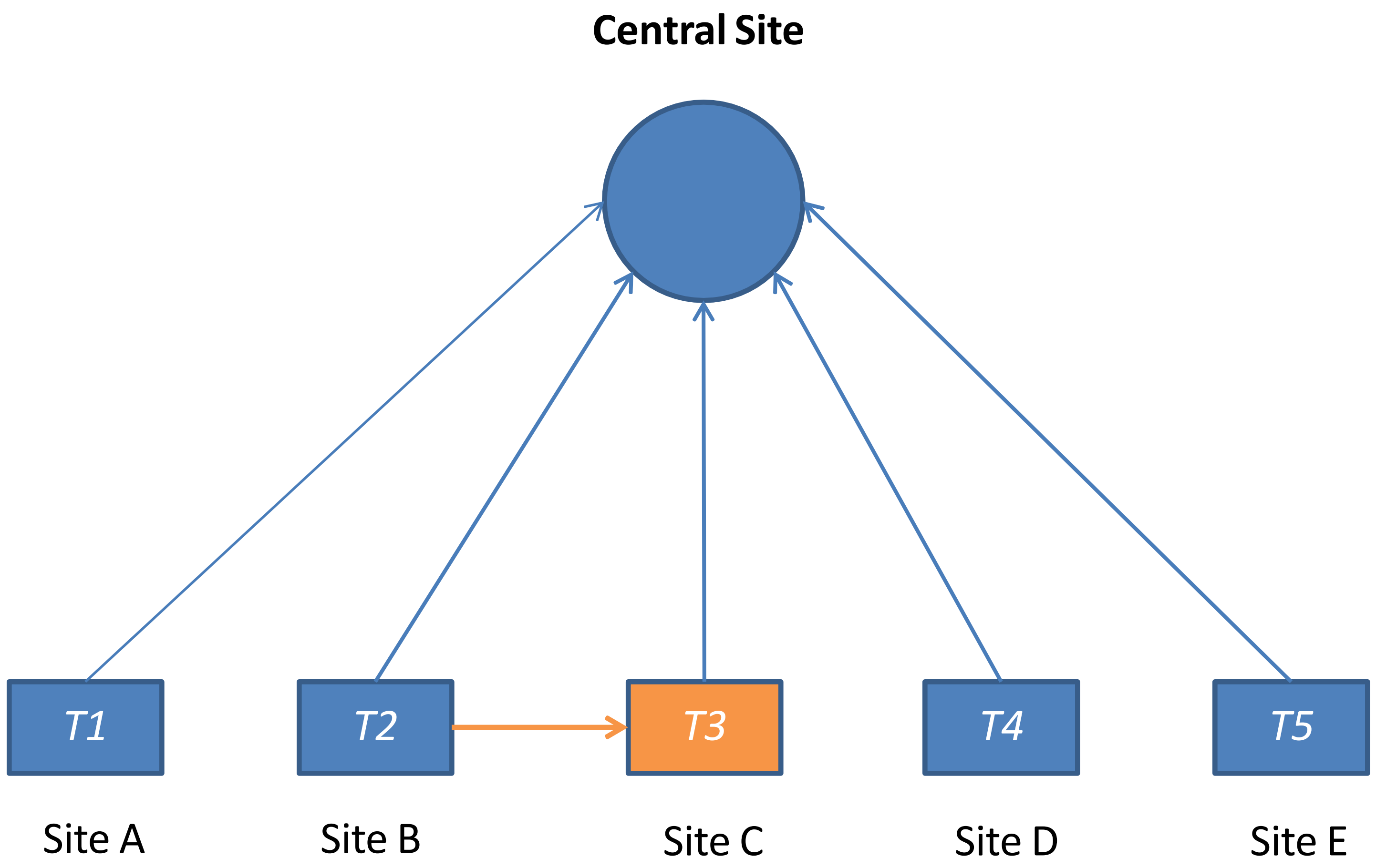
Serial execution effect.

Completely Centralized Algorithm



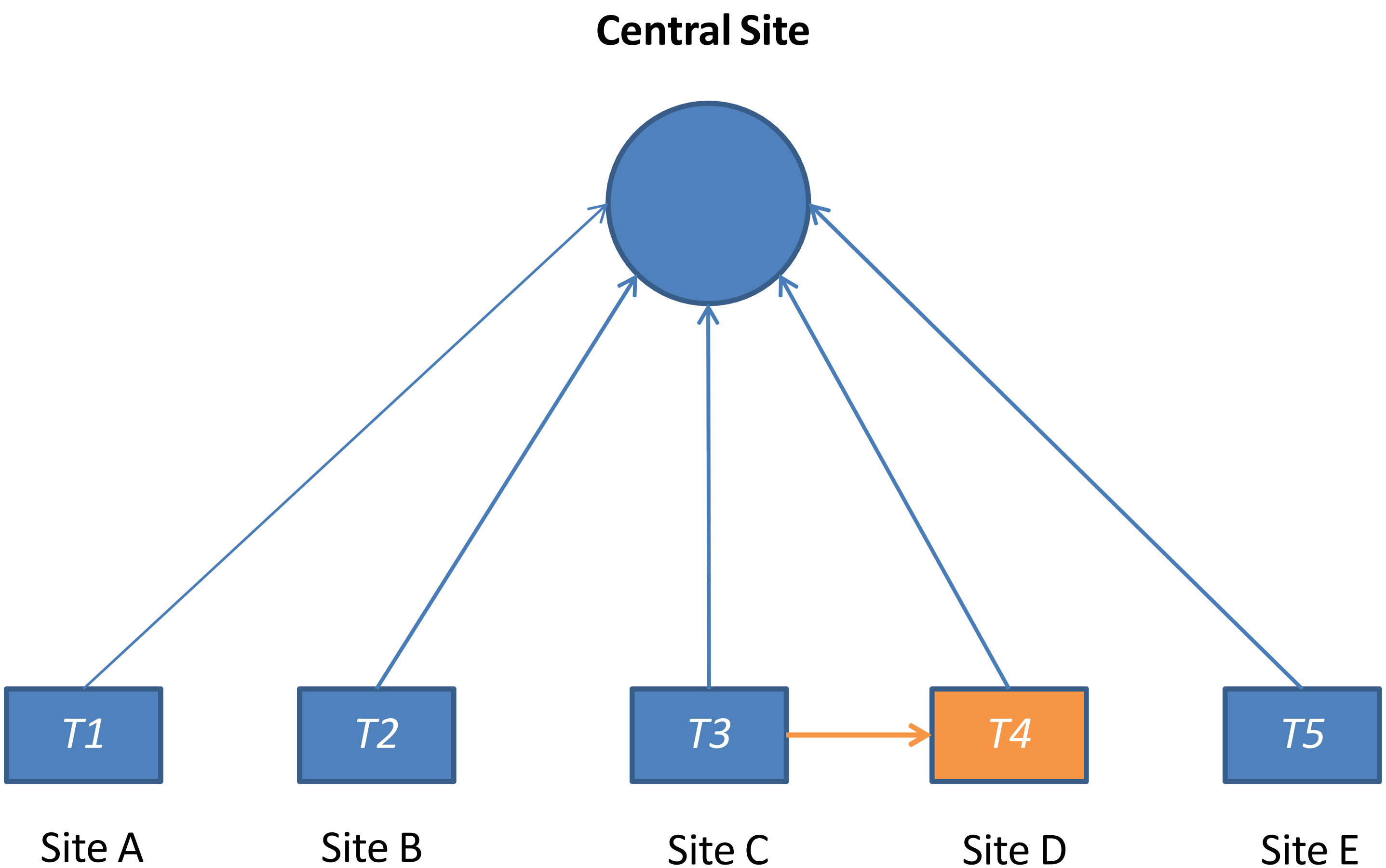
Serial execution effect.

Completely Centralized Algorithm



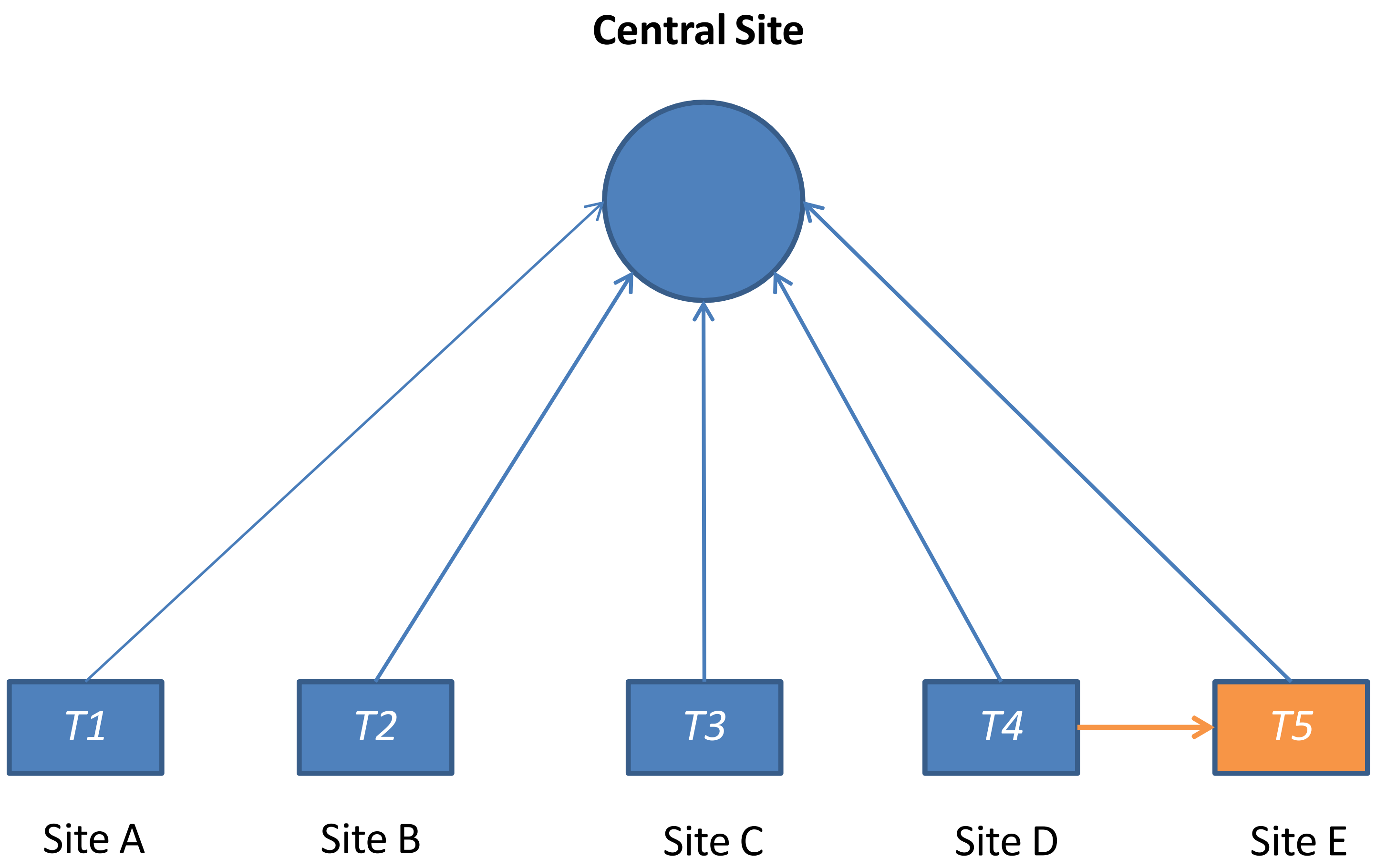
Serial execution effect.

Completely Centralized Algorithm



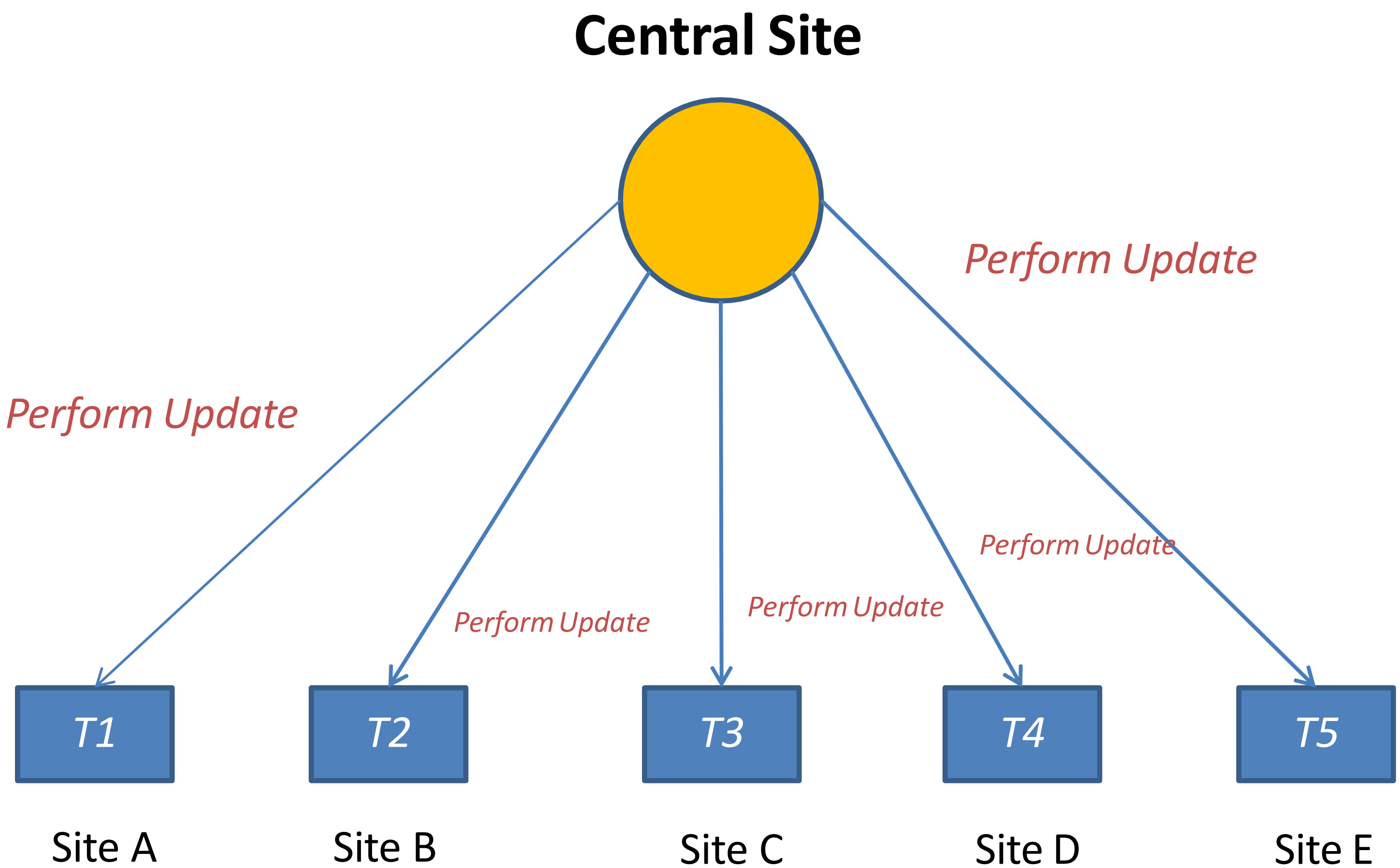
Serial execution effect.

Completely Centralized Algorithm



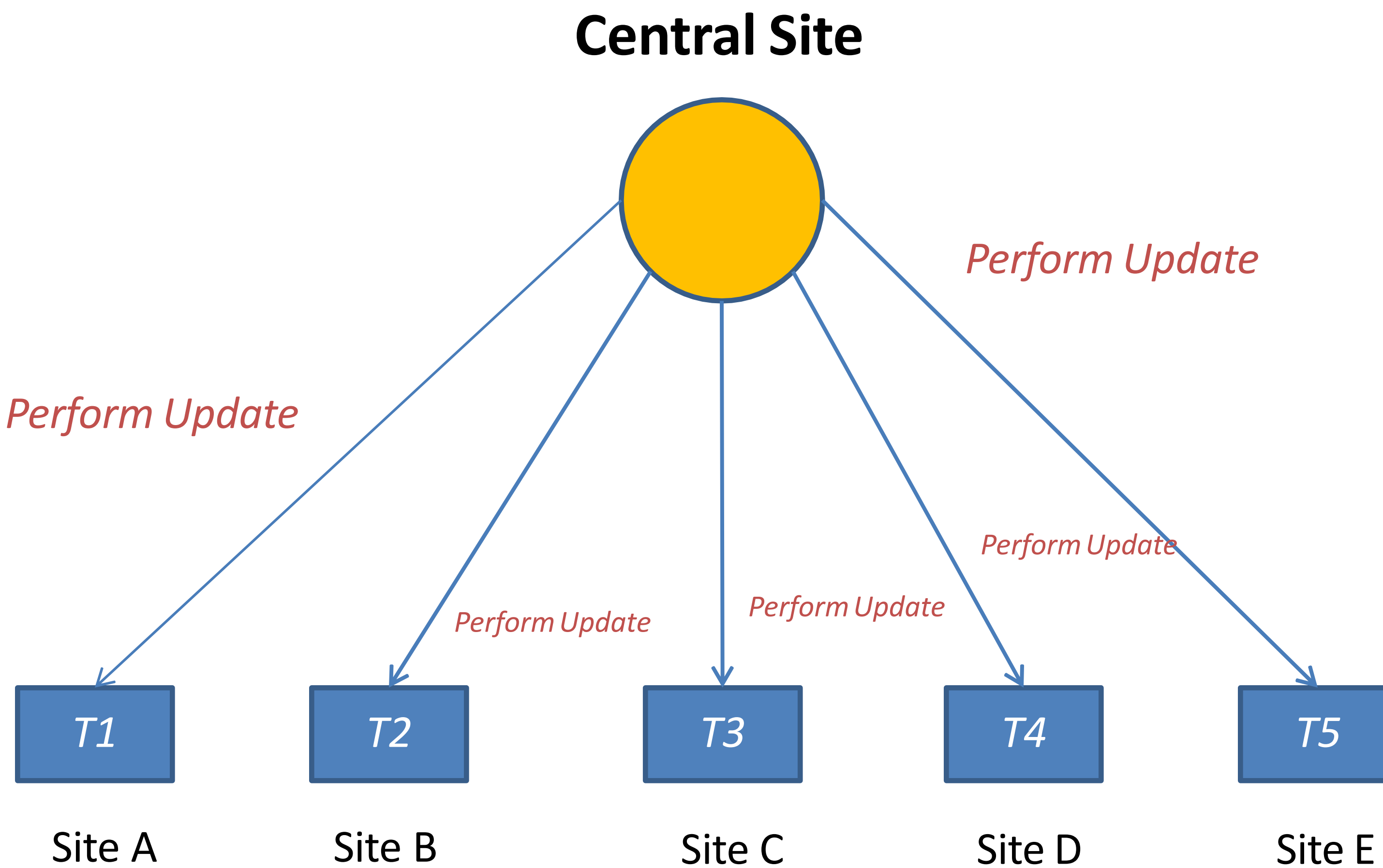
Serial execution effect.

Completely Centralized Algorithm



Perform update message broadcast to all other sites with sequence number.

Completely Centralized Algorithm

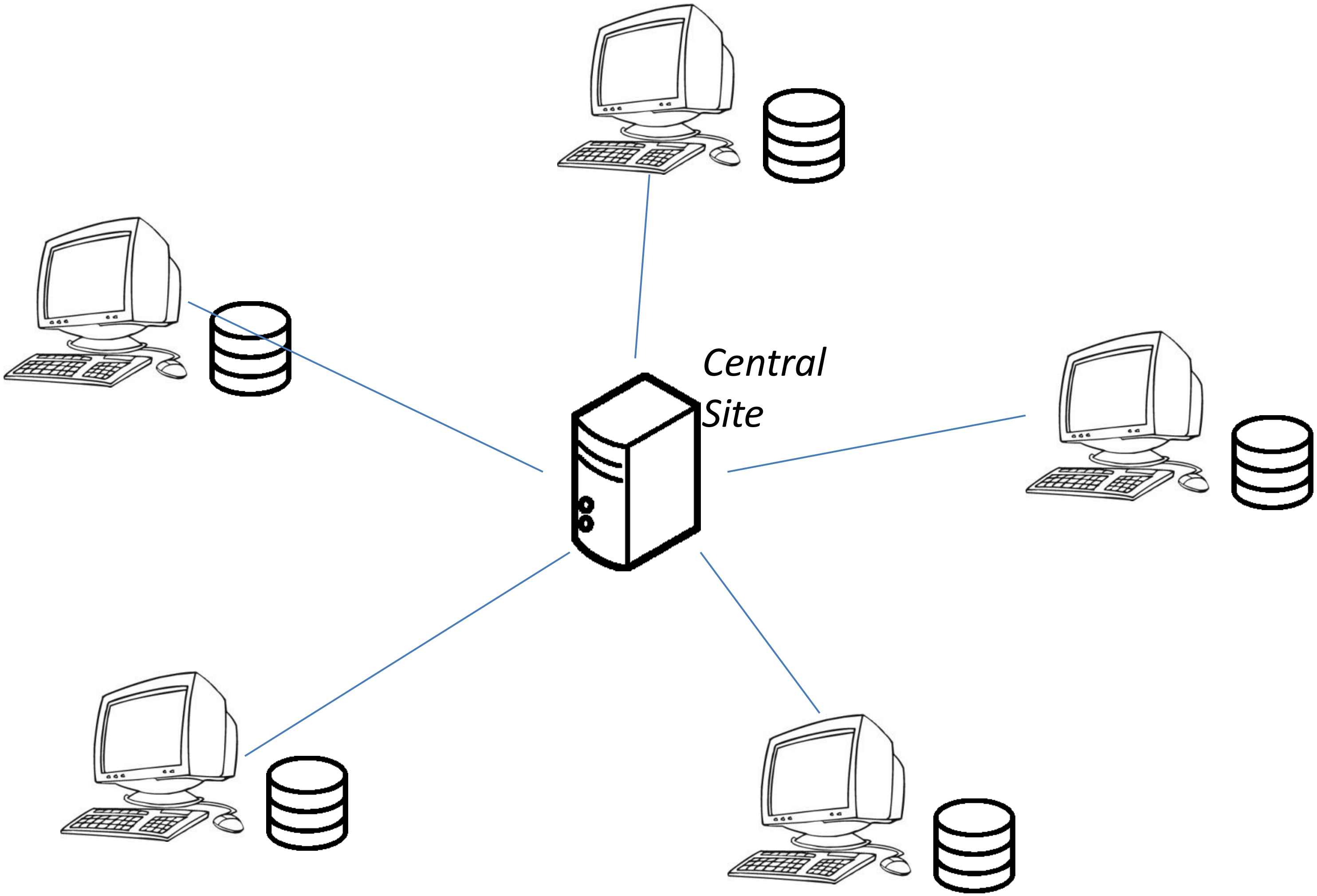


Messages processed then updates applied to local databases.

Centralized Locking Algorithm

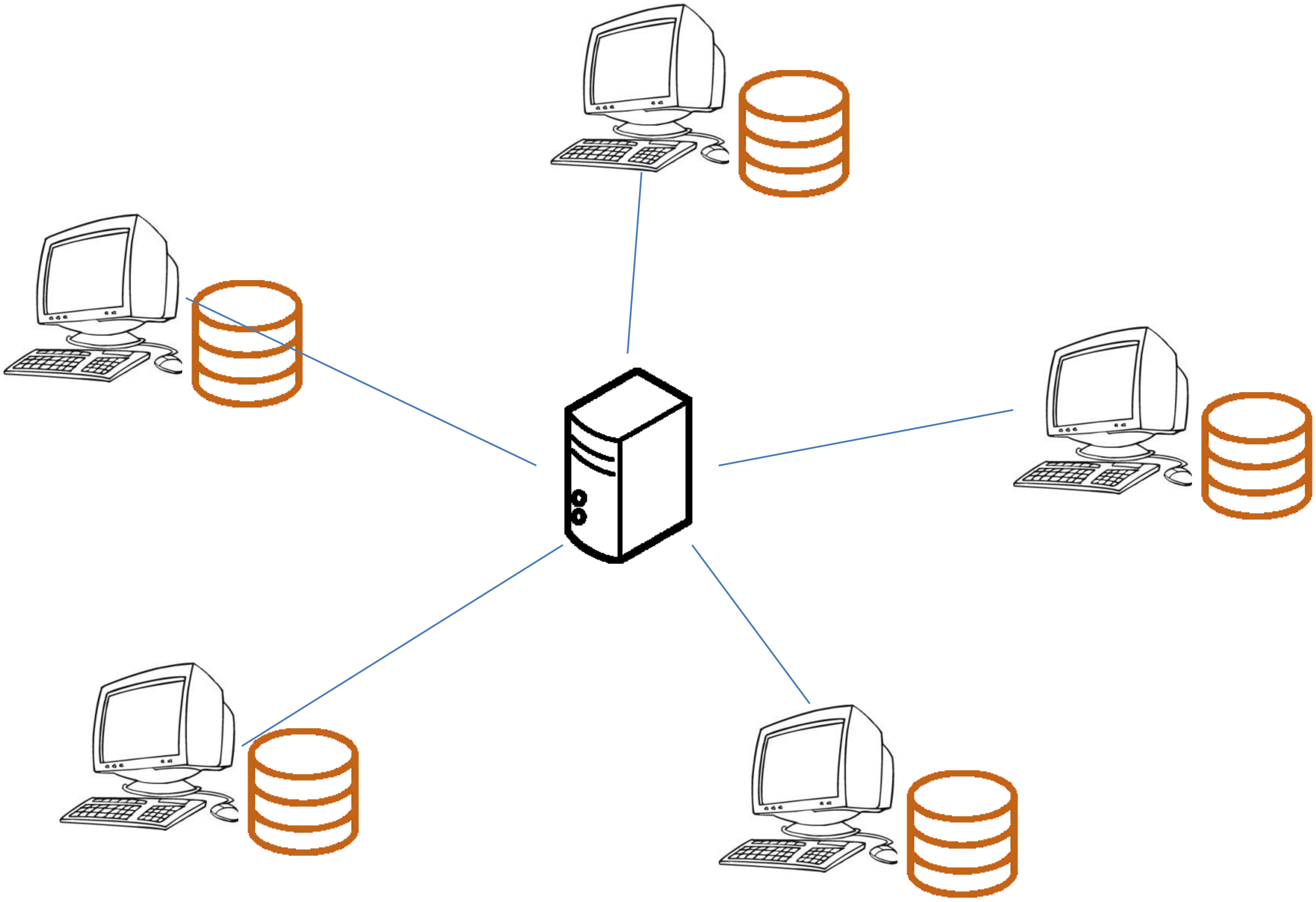
- Transactions processed in distributed manner
- Central site is requested with a *lock request message* for data objects to be accessed
- Site responds with a *lock grant* message with a sequence number
- Queue for each data object
- Site executes its transaction after receiving *lock grant* and broadcasts *perform update* after which the central site releases all locks

Centralized Locking Algorithm



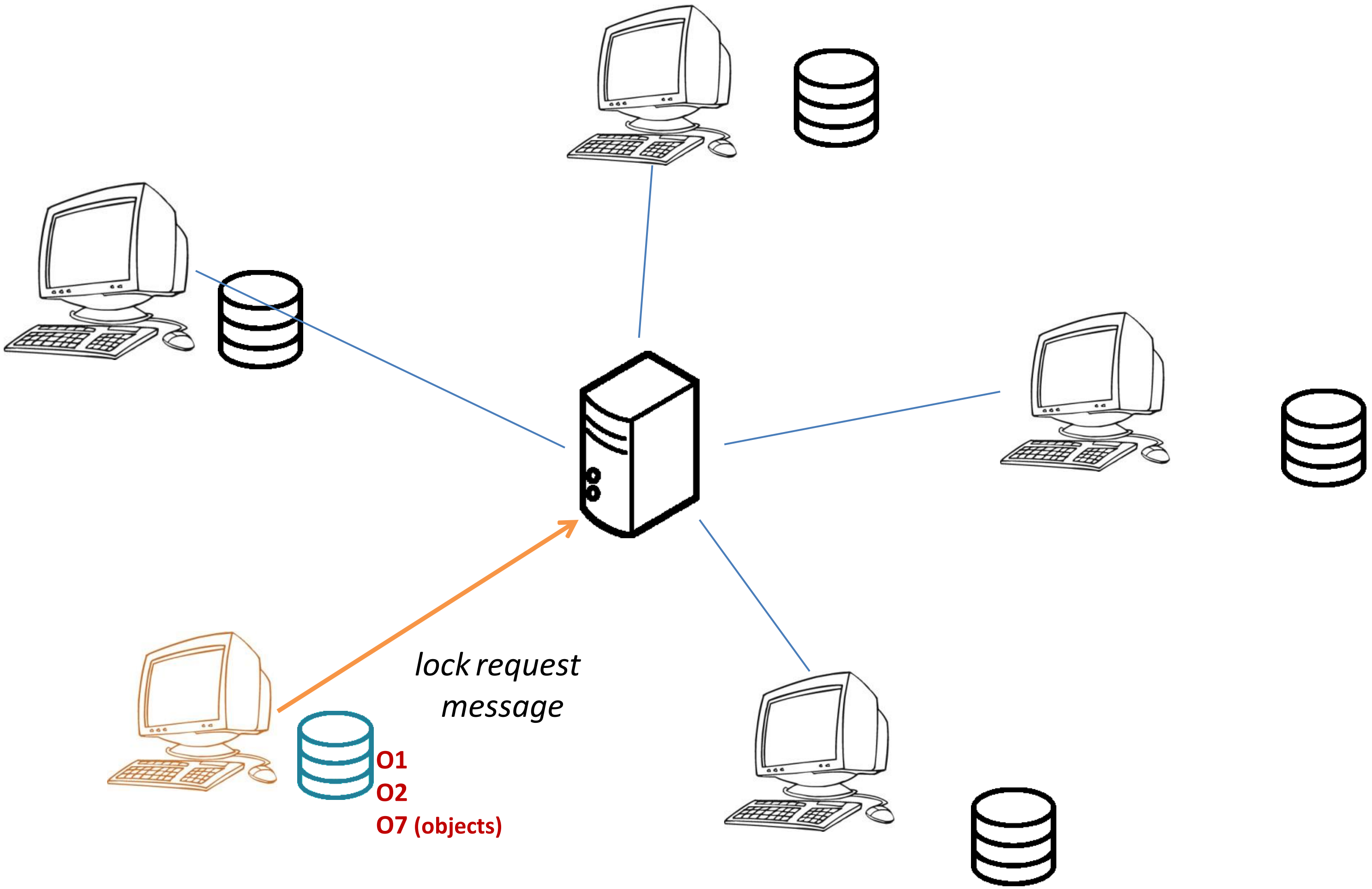
Many Sites with centralized lock management

Centralized Locking Algorithm



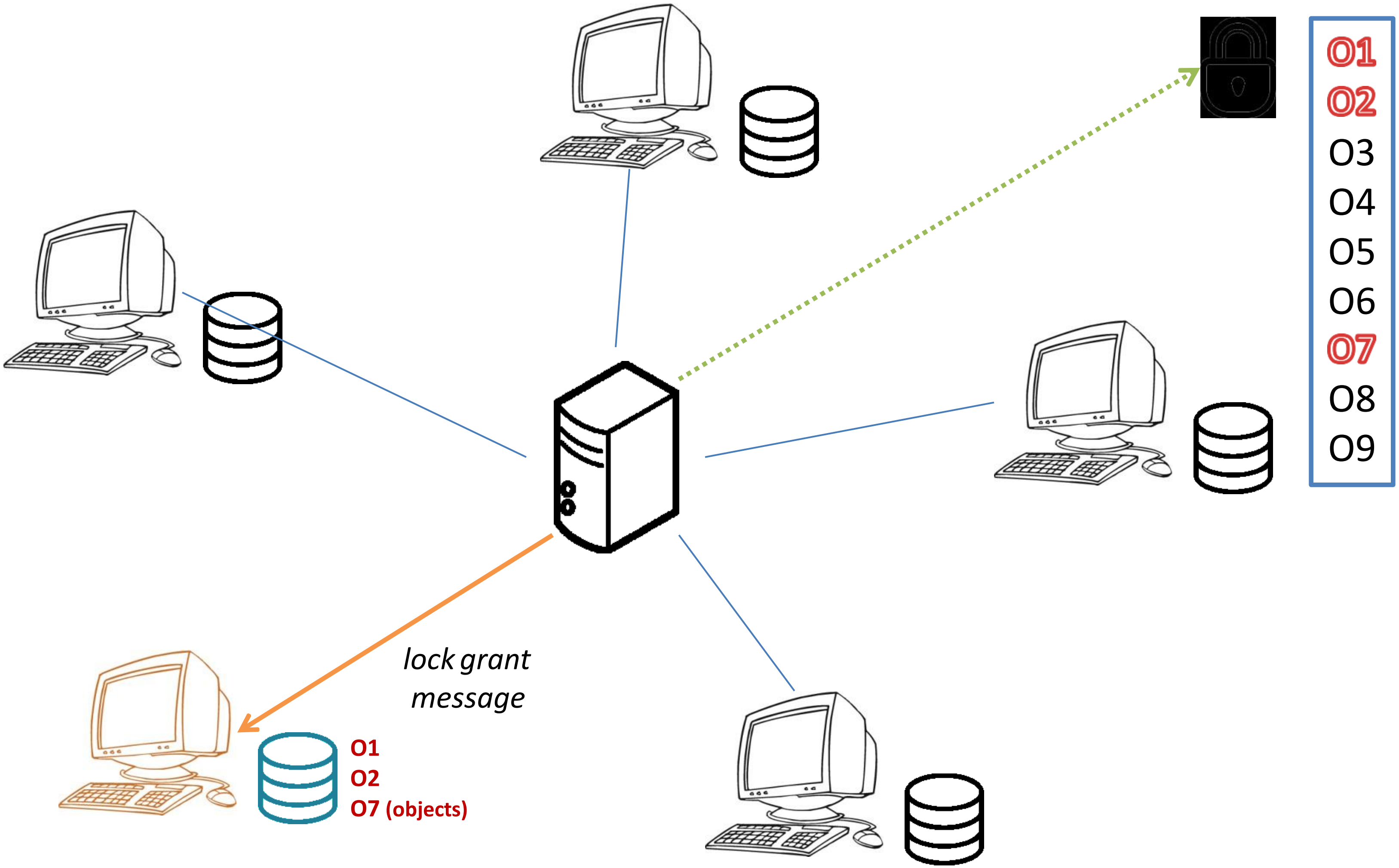
Transactions processed at their *home site* (distributed manner).

Centralized Locking Algorithm



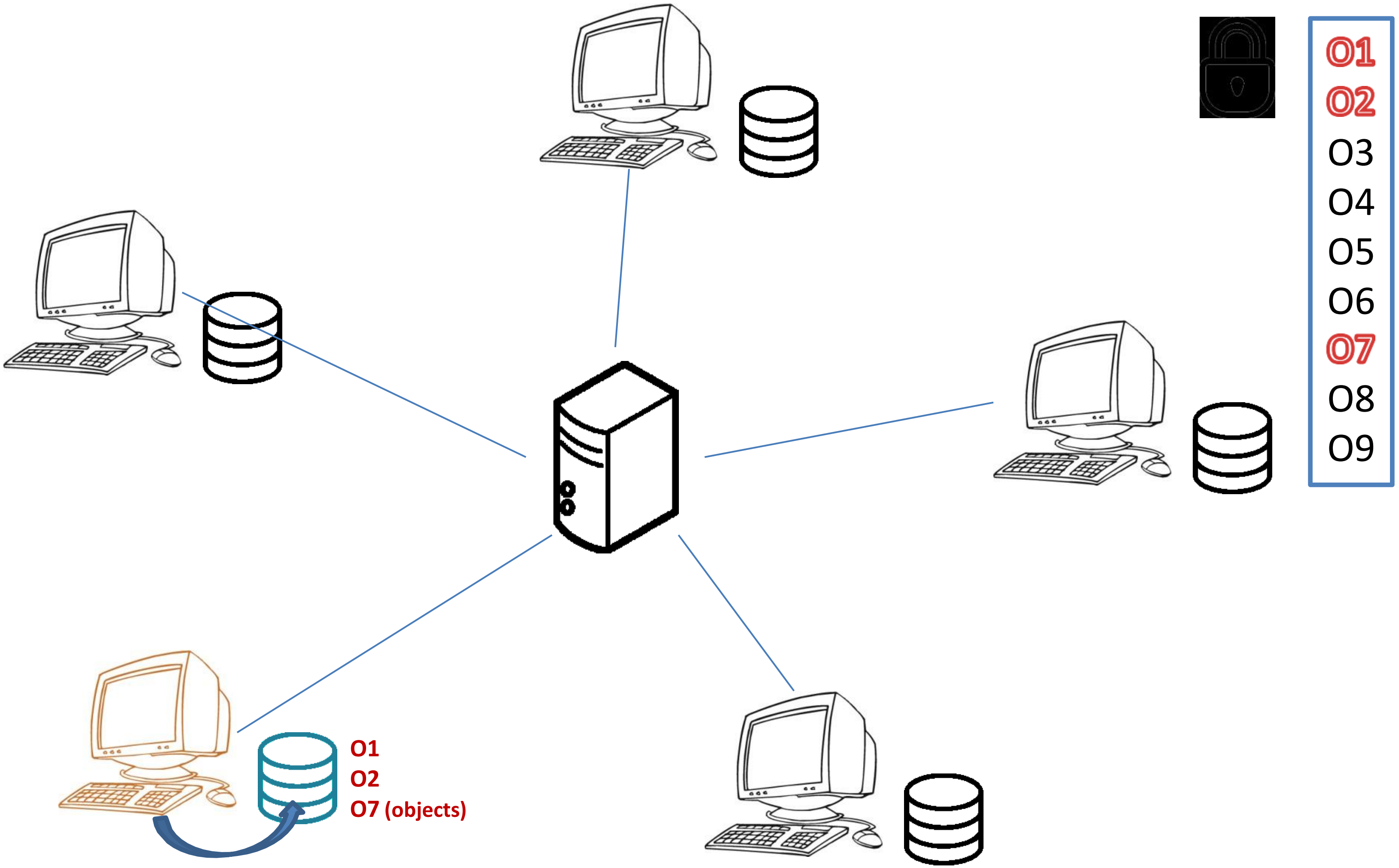
Site requests locks via lock request message for the data objects when it needs to update database

Centralized Locking Algorithm



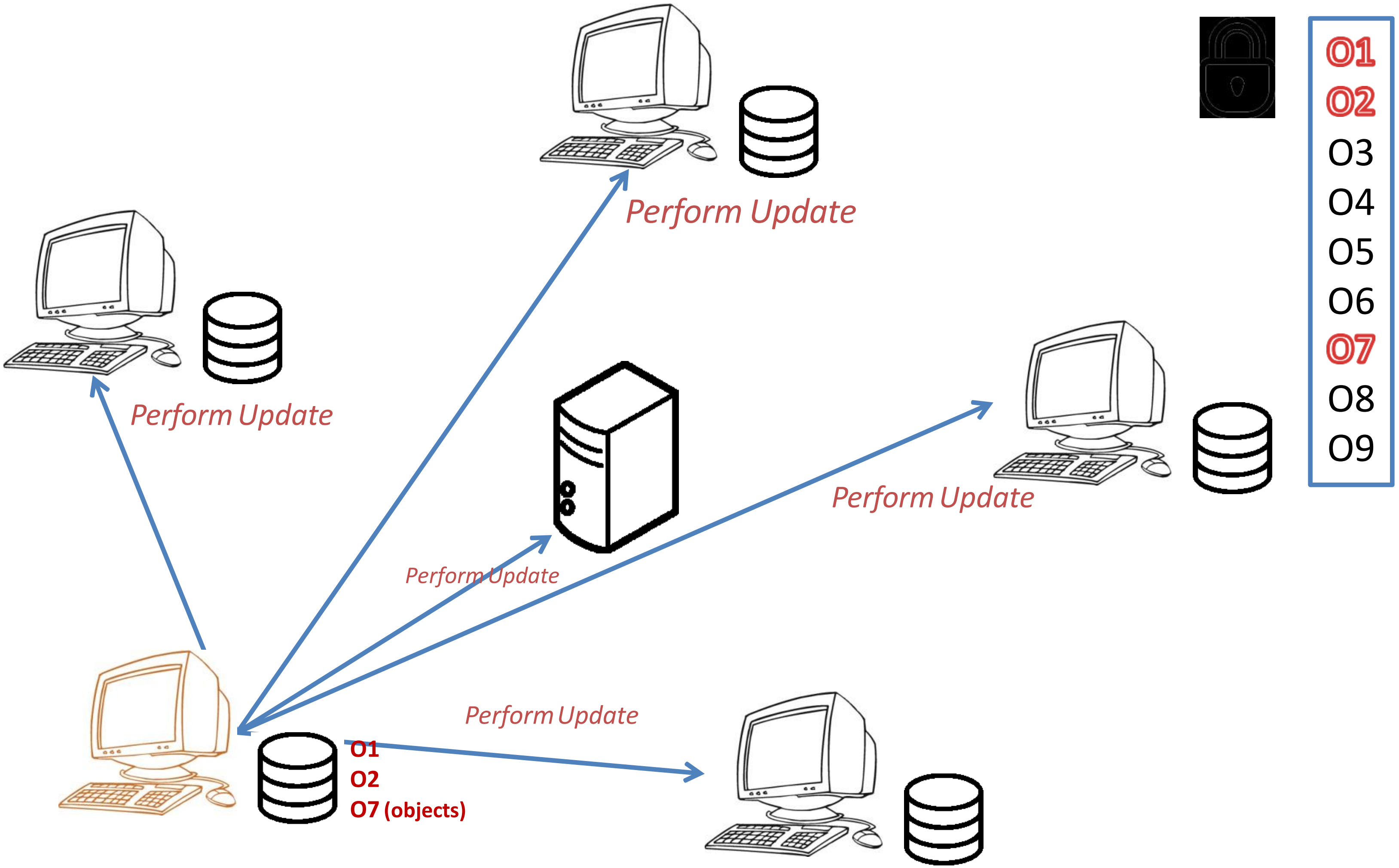
Central site responds with a *lock grant* message (if all locks can be granted)

Centralized Locking Algorithm



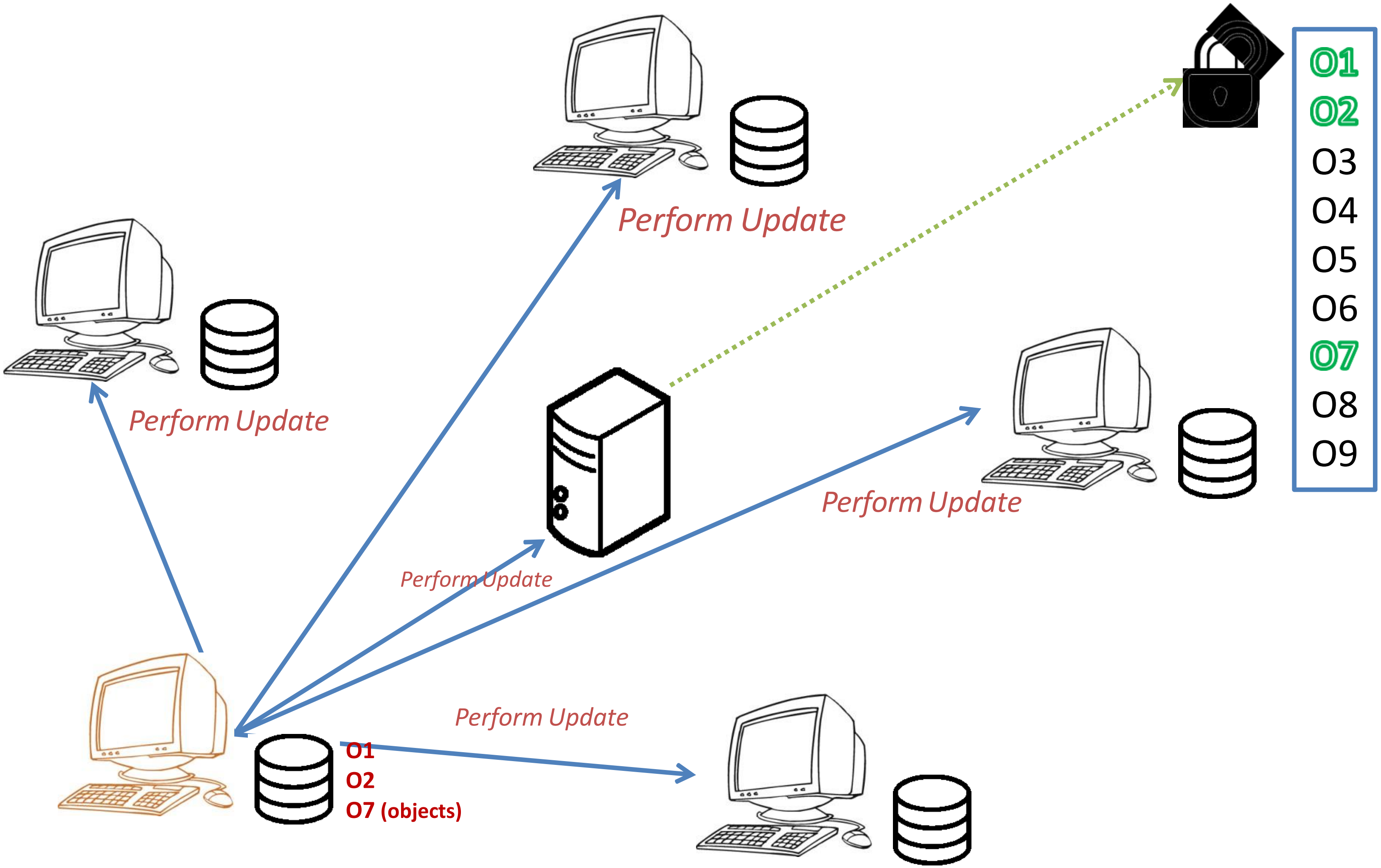
Site executes its transaction after receiving lock grant

Centralized Locking Algorithm



Site broadcasts *perform update* message

Centralized Locking Algorithm



Central site releases all locks when it receives perform update message

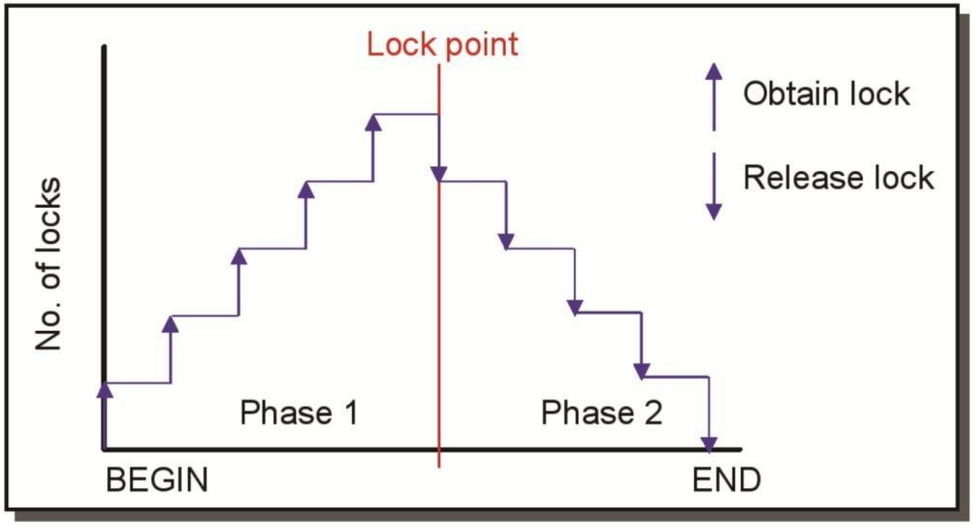
INGRES' Primary-Site Locking Algorithm

- Based on *primary site* method
- Lock management distributed among all the sites
- Each object of database designated with a single primary site

Two Phase Locking (2PL) Algorithm

- Two-phase locking protocol
  - Each transaction is executed in two phases
    - **Growing phase**: the transaction obtains locks
    - **Shrinking phase**: the transaction releases locks

The lock point is the moment when transitioning from the growing phase to the shrinking phase



Two Phase Locking (2PL) Algorithm

- Properties of the 2PL protocol
  - Generates conflict-serializable schedules
  - But schedules may cause cascading aborts
    - If a transaction aborts after it releases a lock, it may cause other transactions that have accessed the unlocked data item to abort as well
- Strict 2PL locking protocol

  Holds the locks till the end of the transaction

  Cascading aborts are avoided