Chapter 20: Database System Architectures

Database System Concepts, 7th Ed.

Outline

- Centralized Database Systems
- Server System Architectures
- Parallel Systems
- Distributed Systems
- Network Types

Centralized Database Systems

- Run on a single computer system
- **Single-user system**
  - Embedded databases
- **Multi-user systems** also known as **server systems**.
  - Service requests received from client systems
  - Multi-core systems with **coarse-grained parallelism**
    - Typically a few to tens of processor cores
    - In contrast, fine-grained parallelism uses very large number of computers

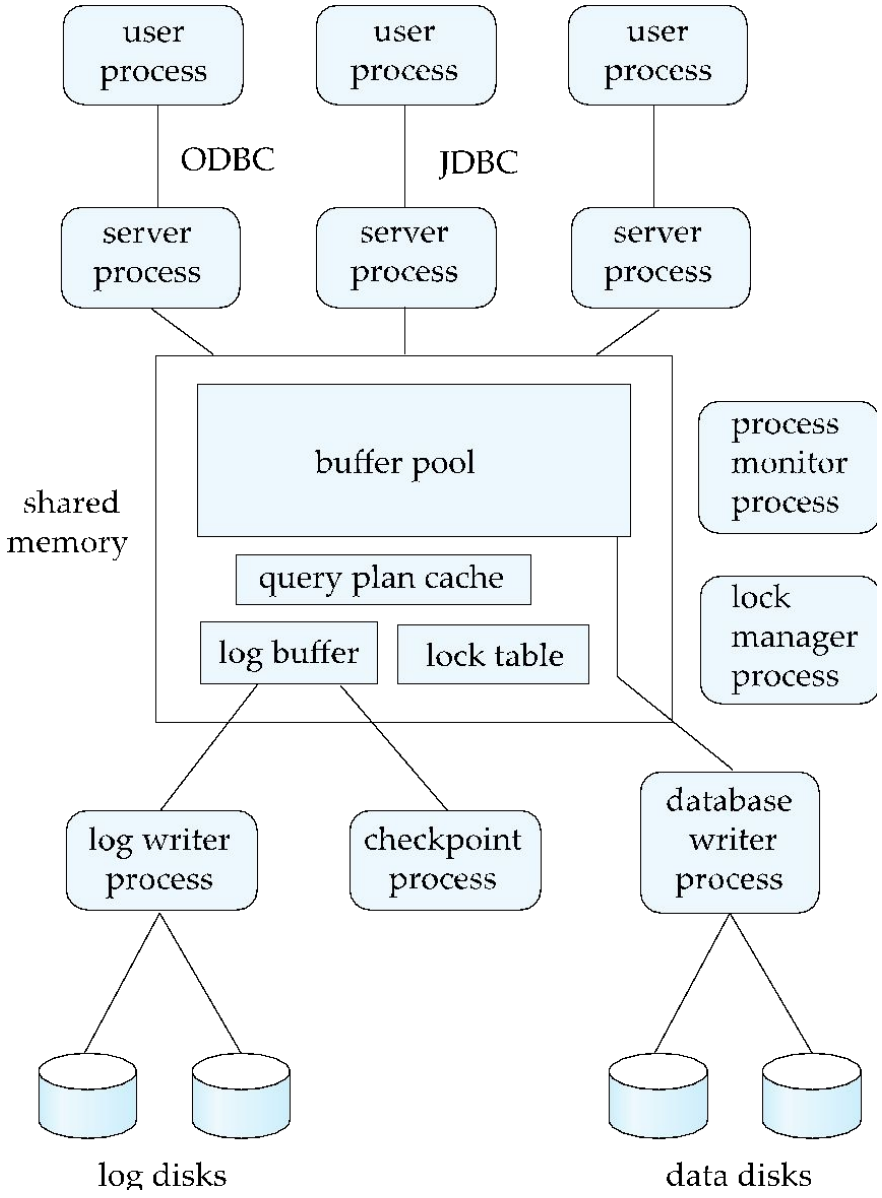
Server System Architecture

- Server systems can be broadly categorized into two kinds:
  - **transaction servers**
    - Widely used in relational database systems, and
  - **data servers**
    - Parallel data servers used to implement high-performance transaction processing systems

Transaction Servers

- Also called **query server** systems or SQL *server* systems
  - Clients send requests to the server
  - Transactions are executed at the server
  - Results are shipped back to the client.
- Requests are specified in SQL, and communicated to the server through a *remote procedure call* (RPC) mechanism.
- Transactional RPC allows many RPC calls to form a transaction.
- Applications typically use ODBC/JDBC APIs to communicate with transaction servers

Transaction System Processes (Cont.)



Transaction Server Process Structure

- A typical transaction server consists of multiple processes accessing data in shared memory
- Shared memory contains shared data
  - Buffer pool
  - Lock table
  - Log buffer
  - Cached query plans (reused if same query submitted again)
- All database processes can access shared memory
- Server processes
  - These receive user queries (transactions), execute them and send results back
  - Processes may be **multithreaded**, allowing a single process to execute several user queries concurrently
  - Typically multiple multithreaded server processes

Transaction Server Processes (Cont.)

- Database writer process
  - Output modified buffer blocks to disks continually
- Log writer process
  - Server processes simply add log records to log record buffer
  - Log writer process outputs log records to stable storage.
- Checkpoint process
  - Performs periodic checkpoints
- Process monitor process
  - Monitors other processes, and takes recovery actions if any of the other processes fail
    - E.g. aborting any transactions being executed by a server process and restarting it

Transaction System Processes (Cont.)

- Lock manager process
  - To avoid overhead of interprocess communication for lock request/grant, each database process operates directly on the lock table
    - instead of sending requests to lock manager process
  - Lock manager process still used for deadlock detection
- To ensure that no two processes are accessing the same data structure at the same time, databases systems implement **mutual exclusion** using either
  - Atomic instructions
    - Test-And-Set
    - Compare-And-Swap (CAS)
  - Operating system semaphores
    - Higher overhead than atomic instructions

Data Servers/Data Storage Systems

- Data items are shipped to clients where processing is performed
- Updated data items written back to server
- Earlier generation of data servers would operated in units of data items, or pages containing multiple data items
- Current generation data servers (also called data storage systems) only work in units of data items
  - Commonly used data item formats include JSON, XML, or just uninterpreted binary strings

Data Servers/Storage Systems (Cont.)

- **Prefetching**
  - Prefetch items that may be used soon
- **Data caching**
  - Cache coherence
- **Lock caching**
  - Locks can be cached by client across transactions
  - Locks can be **called back** by the server
- **Adaptive lock granularity**
  - **Lock granularity escalation**
    - switch from finer granularity (e.g. tuple) lock to coarser
  - **Lock granularity de-escalation**
    - Start with coarse granularity to reduve overheads, switch to finer granularity in case of more concurrency conflict at server
    - Details in book

Data Servers (Cont.)

- **Data Caching**
  - Data can be cached at client even in between transactions
  - But check that data is up-to-date before it is used (**cache coherency**)
  - Check can be done when requesting lock on data item
- **Lock Caching**
  - Locks can be retained by client system even in between transactions
  - Transactions can acquire cached locks locally, without contacting server
  - Server **calls back** locks from clients when it receives conflicting lock request. Client returns lock once no local transaction is using it.
    - Similar to lock callback on prefetch, but across transactions.

Parallel Systems

- Parallel database systems consist of multiple processors and multiple disks connected by a fast interconnection network.
- Motivation: handle workloads beyond what a single computer system can handle
- High performance **transaction processing**
  - E.g. handling user requests at web-scale
- **Decision support** on very large amounts of data
  - E.g. data gathered by large web sites/apps

Parallel Systems (Cont.)

- A **coarse-grain parallel** machine consists of a small number of powerful processors
- A **massively parallel** or **fine grain parallel** machine utilizes thousands of smaller processors.
  - Typically hosted in a **data center**
- Two main performance measures:
  - **throughput** --- the number of tasks that can be completed in a given time interval
  - **response time** --- the amount of time it takes to complete a single task from the time it is submitted

Speed-Up and Scale-Up

- **Speedup**: a fixed-sized problem executing on a small system is given to a system which is *N*-times larger.
  - Measured by:

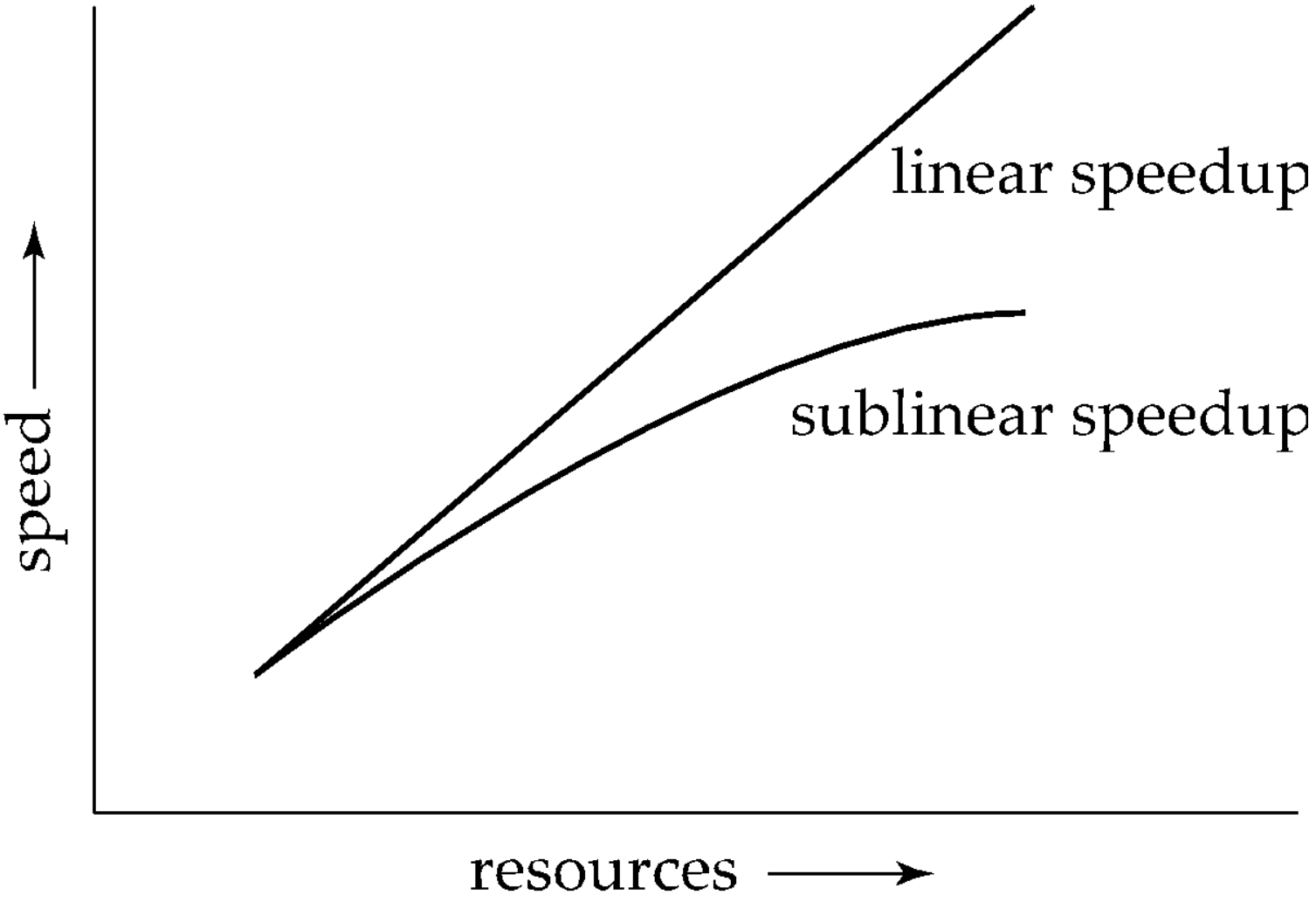
    $$speedup = \frac{small\ system\ elapsed\ time}{large\ system\ elapsed\ time}$$

  - Speedup is **linear** if equation equals N.
- **Scaleup**: increase the size of both the problem and the system
  - *N*-times larger system used to perform *N*-times larger job
  - Measured by:

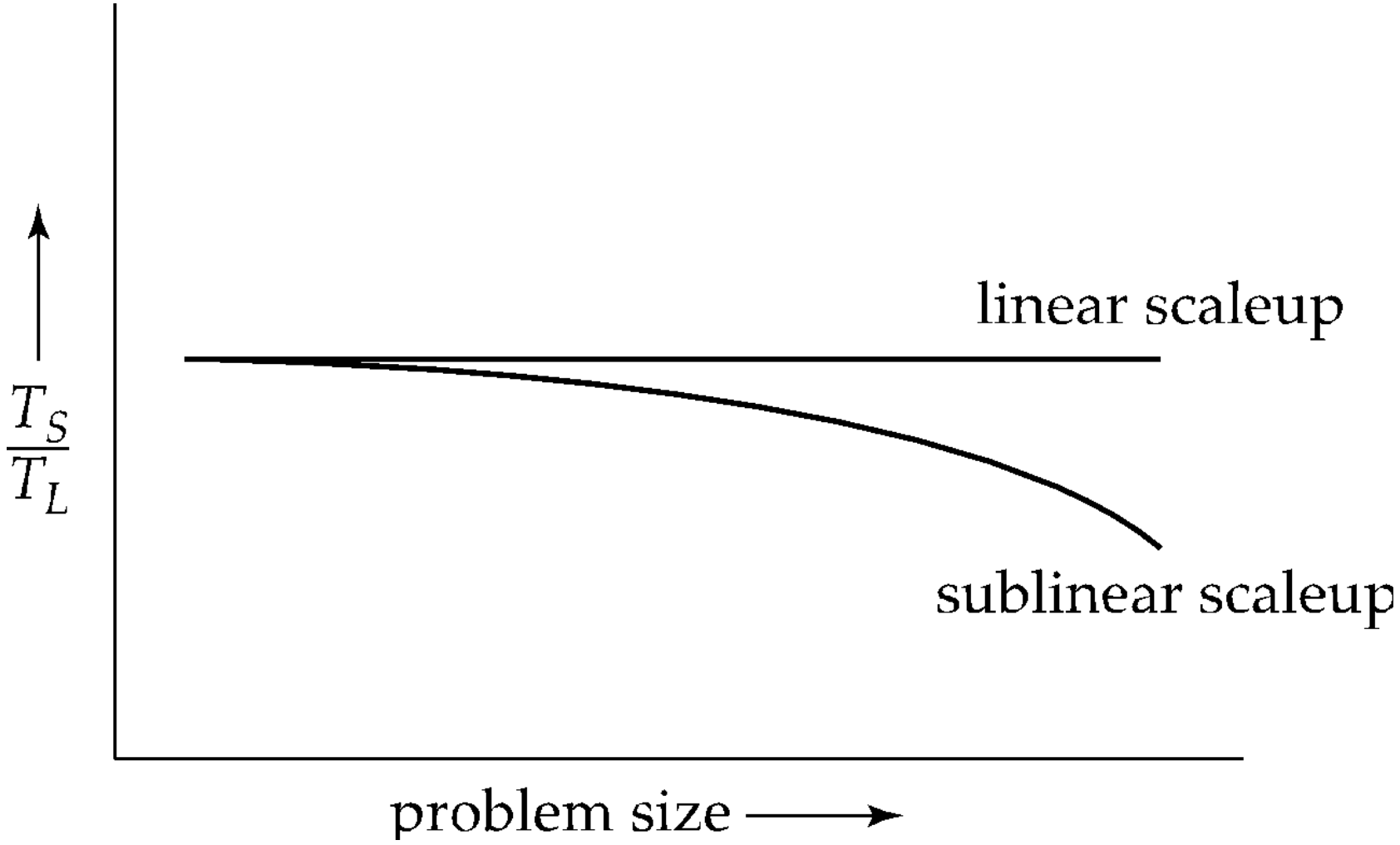
    $$scaleup = \frac{small\ system\ small\ problem\ elapsed\ time}{big\ system\ big\ problem\ elapsed\ time}$$

  - Scale up is **linear** if equation equals 1.

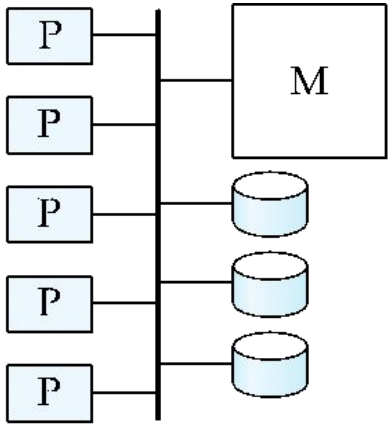
Speedup



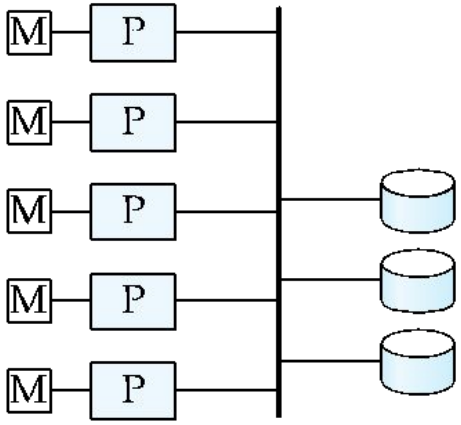
Scaleup



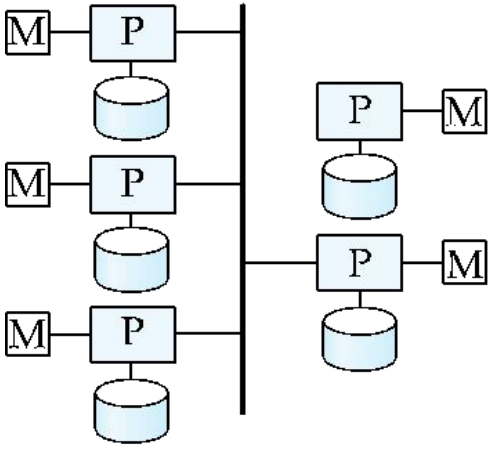
Parallel Database Architectures



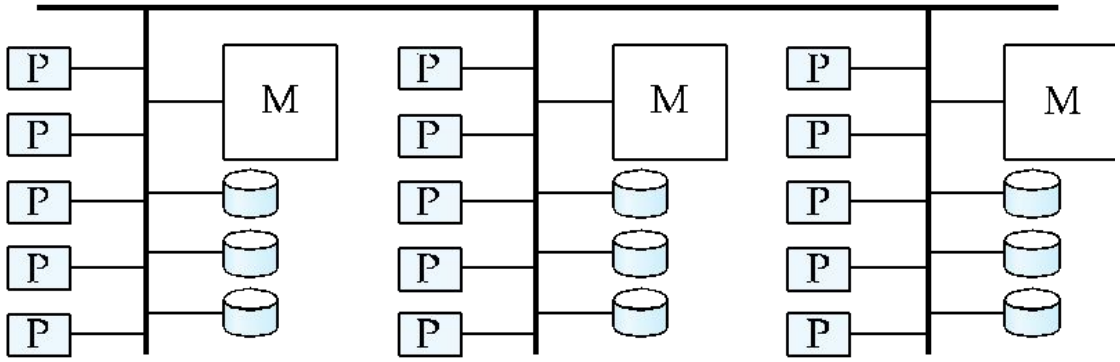
(a) shared memory



(b) shared disk



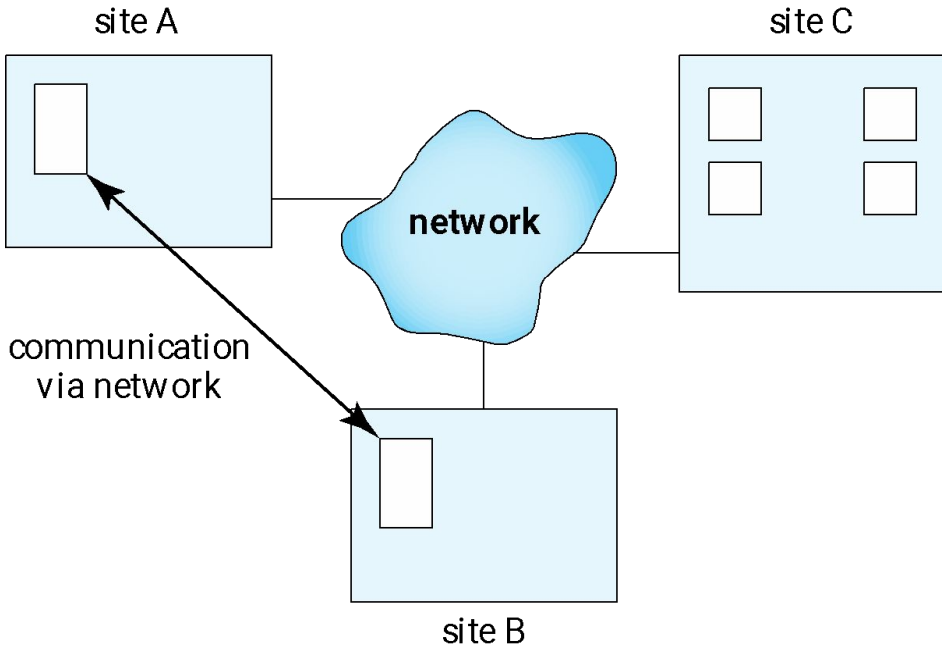
(c) shared nothing



(d) hierarchical

Distributed Systems

- Data spread over multiple machines (also referred to as **sites** or **nodes**).
- **Local-area networks** (**LANs**)
- **Wide-area networks** (**WAN**s)
  - Higher latency



Distributed Databases

- **Homogeneous distributed databases**
  - Same software/schema on all sites, data may be partitioned among sites
  - Goal: provide a view of a single database, hiding details of distribution
- **Heterogeneous distributed databases**
  - Different software/schema on different sites
  - Goal: integrate existing databases to provide useful functionality
- Differentiate between **local transactions** and **global transactions**
  - A local transaction accesses data in the *single* site at which the transaction was initiated.
  - A global transaction either accesses data in a site different from the one at which the transaction was initiated or accesses data in several different sites.