Distributed Systems
CS 15-440

Consistency and Replication – Part II

Lecture 11, Oct 10, 2011

Majd F. Sakr, Vinay Kolar, Mohammad Hammoud

Today…

- Last Session
  - Consistency and Replication
    - Introduction and Data-Centric Consistency Models
- Today's session
  - Consistency and Replication – Part II
    - Finish Data-centric Consistency Models
    - Client-Centric Consistency Models
    - Replica Management
- Announcement:
  - Interim design report for Project 2 due today

Recap: Trade-offs in Maintaining Consistency

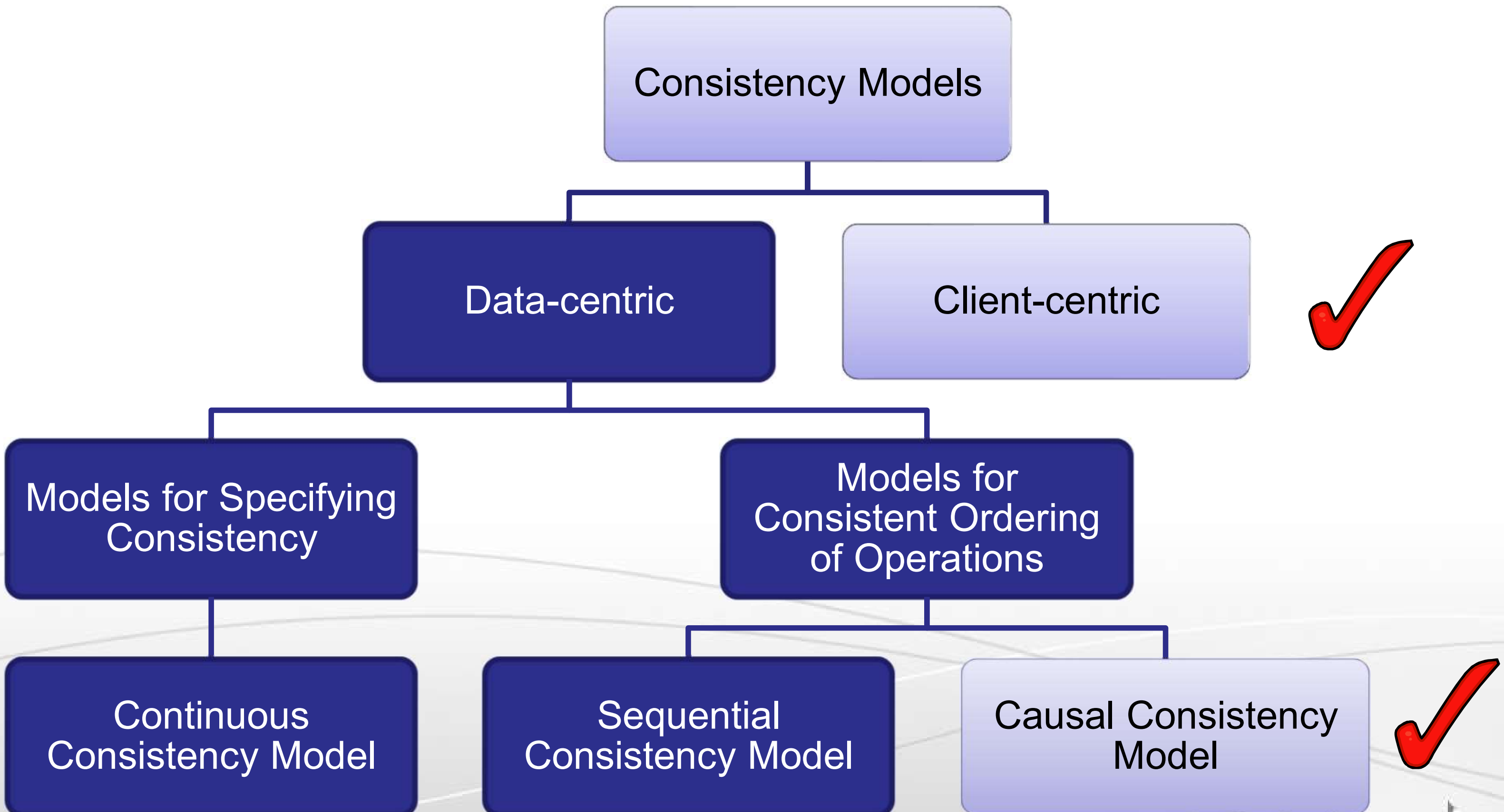
- Maintaining consistency should balance between the strictness of consistency versus efficiency
  - How much consistency is “good-enough” depends on the application

Easier to implement, and is efficient

Generally hard to implement, and is inefficient

Recap: Consistency Models

+ A consistency model states the level of consistency provided by the *data-store* to the processes while reading and writing the data



Types of Ordering

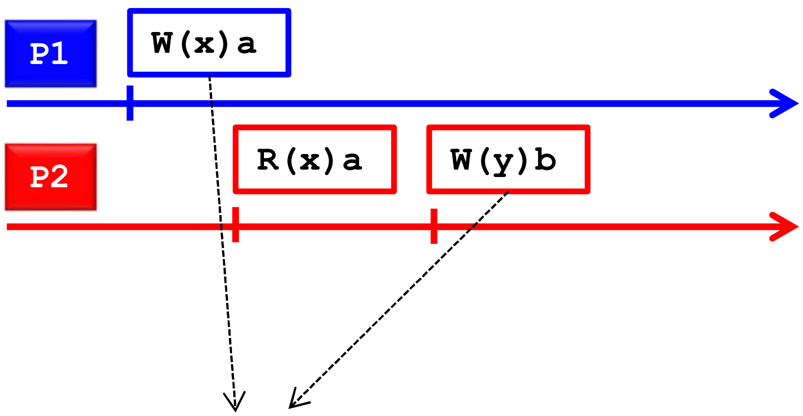
1. Total Ordering
2. Sequential Ordering
3. Causal Ordering

Causality (Recap)

- Causal relation between two events
  - If `a` and `b` are two events `a` and `b` such that `a` **`happened-before b`** or $a \rightarrow b$, and
  - If the (logical) time when event `a` and `b` is received at a process P_i is denoted by $C_i(a)$ and $C_i(b)$
  - Then, if we can infer that $a \rightarrow b$ by observing that $C_i(a) < C_i(b)$, then `a` and `b` are causally related
- Causality can be implemented using Vector Clocks

Causal vs. Concurrent events

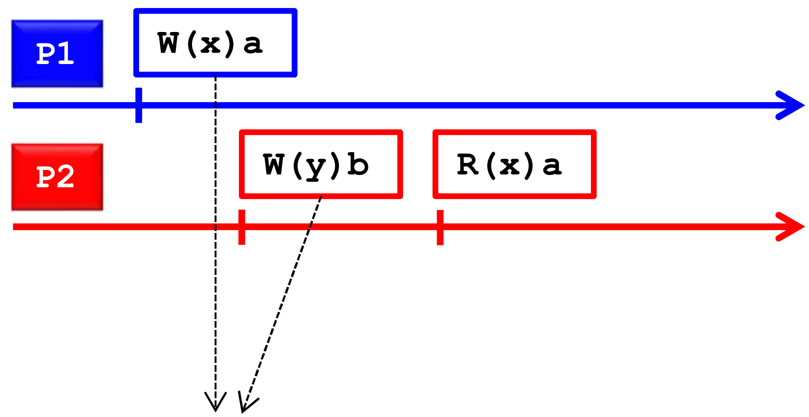
+ Consider an interaction between processes $\mathbf{P_1}$ and $\mathbf{P_2}$ operating on replicated data $\mathbf{x}$ and $\mathbf{y}$



Events are causally related

Events are not concurrent

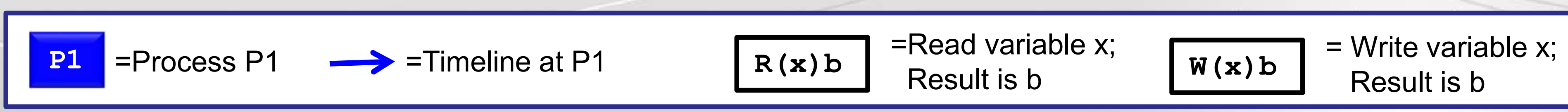
- Computation of $\mathbf{y}$ at $\mathbf{P_2}$ may have depended on value of $\mathbf{x}$ written by $\mathbf{P_1}$



Events are not causally related

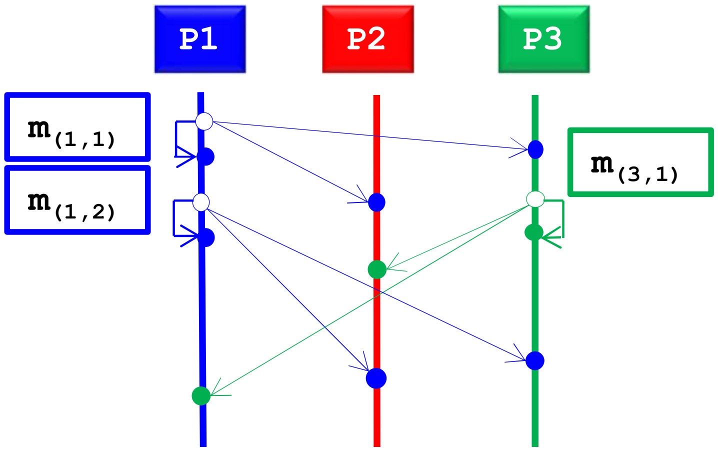
Events are concurrent

- Computation of $\mathbf{y}$ at $\mathbf{P_2}$ does not depend on value of $\mathbf{x}$ written by $\mathbf{P_1}$



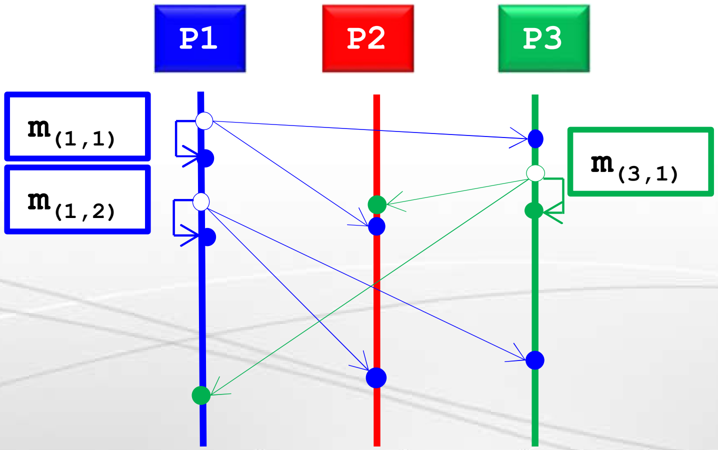
Causal Ordering

- Causal Order
  - If process P_i sends a message m_i and P_j sends m_j, and if $m_i \rightarrow m_j$ (operator '$\rightarrow$' is Lamport's **happened-before** relation) then any correct process that delivers m_j will deliver m_i before m_j



Valid Causal Orders

- In the example, $m_{(1,1)}$ and $m_{(3,1)}$ are in Causal Order



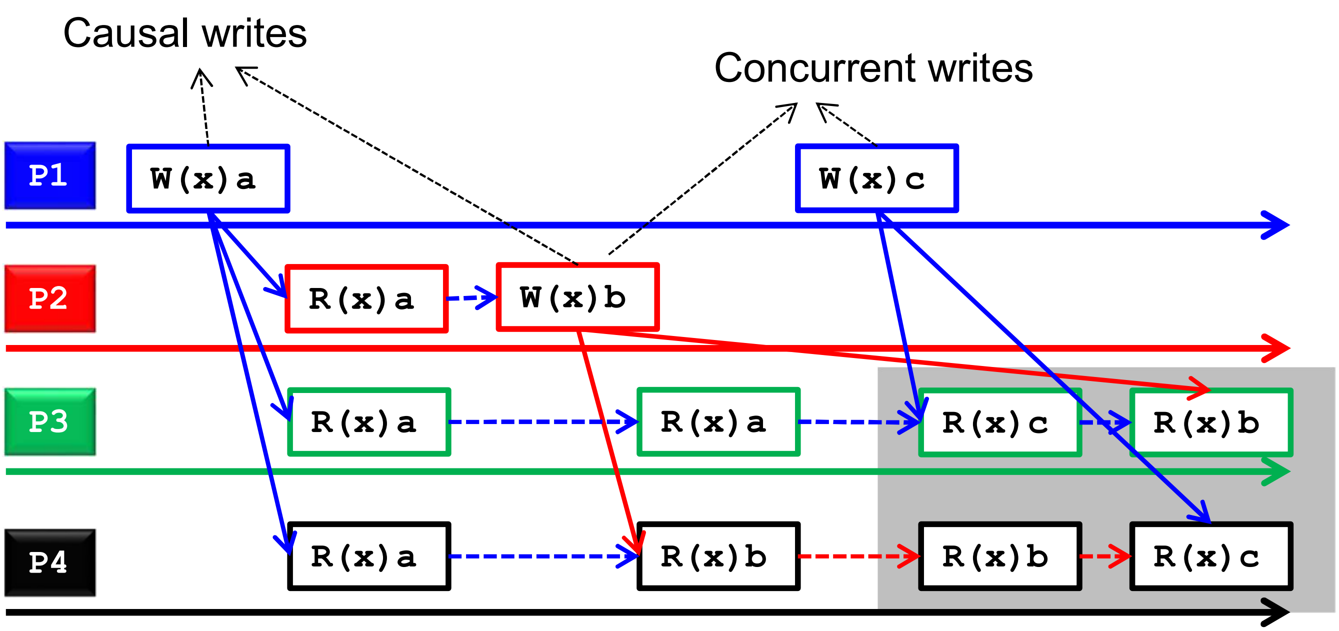
Invalid Causal Order

- Drawback:
  - The **happened-before** relation between m_i and m_j should be induced before communication

Causal Consistency Model

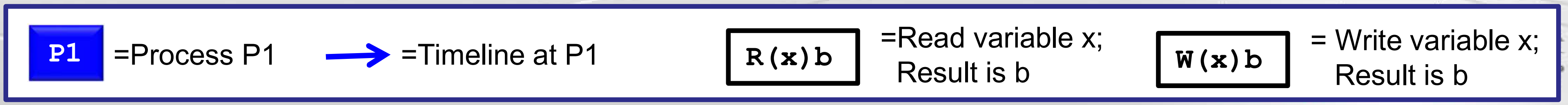
- A data-store is causally consistent if:
  - Writes that are potentially causally related must be seen by all the processes in the same order
  - Concurrent writes may be seen in a different order on different machines

Example of a Causally Consistent Data-store



A Causally Consistent Data-Store

But not a Sequentially Consistent Data-Store

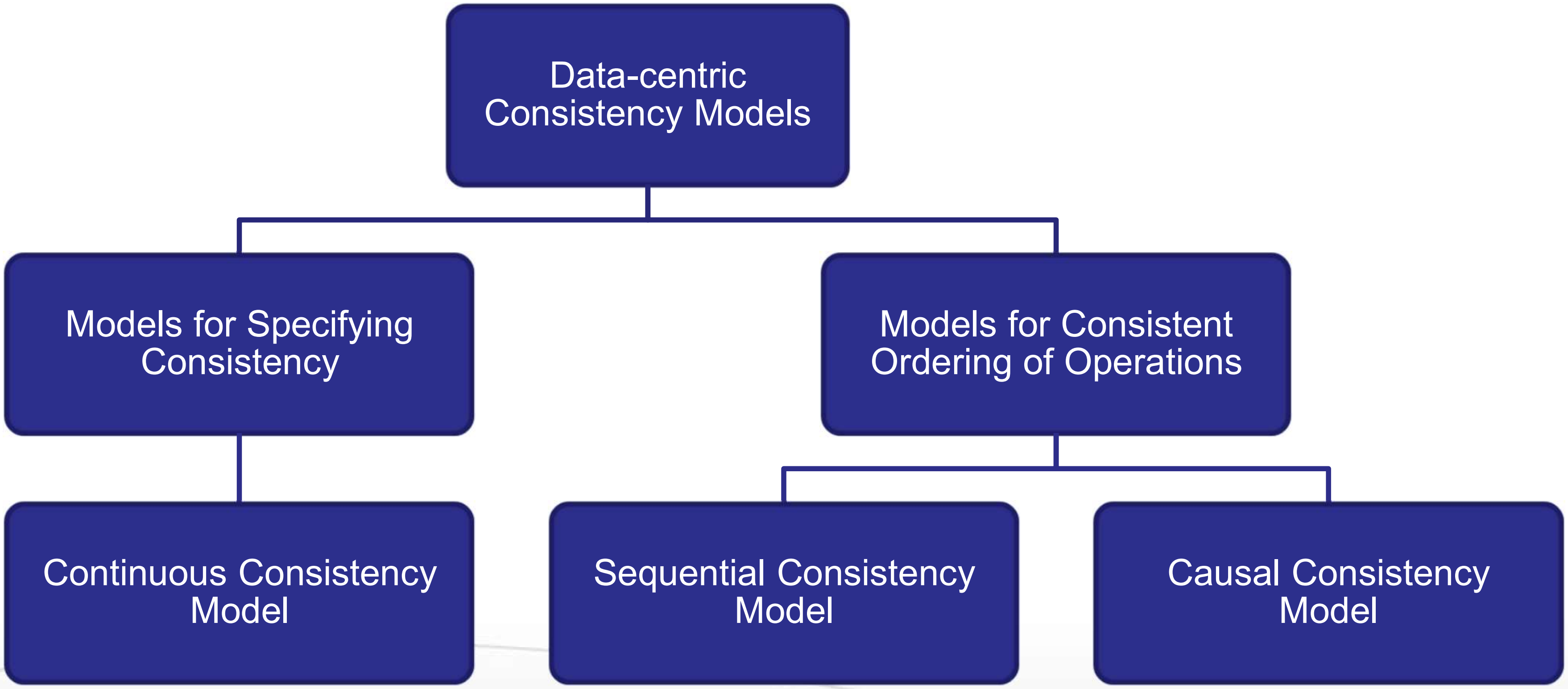


Implications of adopting a Causally Consistent Data-store for Applications

- Processes have to keep track of which processes have seen which writes

- This requires maintaining a dependency graph between write and read operations
  - Vector clocks provides a way to maintain causally consistent data-base

Topics Covered in Data-centric Consistency Models

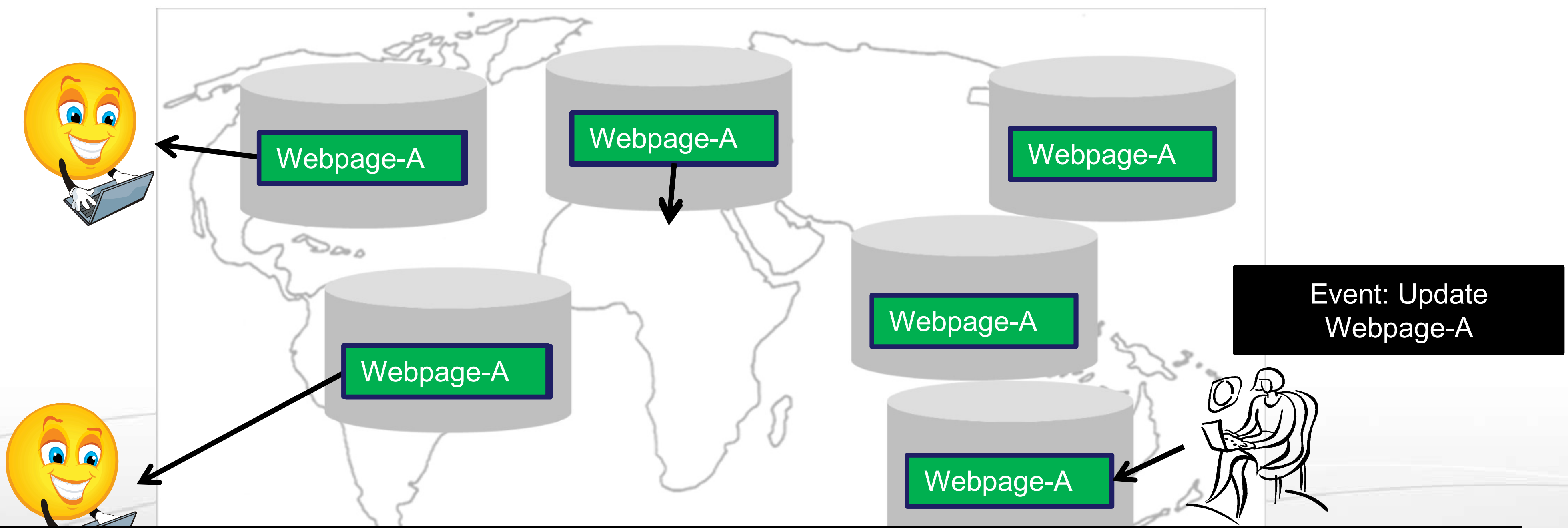


But, is Data-centric Consistency Model good for all applications?

Applications that can use Data-centric Models

+ Data-centric models are applicable when many processes are concurrently updating the data-store

+ But, do all applications need all replicas to be consistent?



Data-Centric Consistency Model is too strict when

- One client process updates the data
- Other processes read the data, and are OK with reasonably stale data

Summary of Data-Centric Consistency Models

Data-centric consistency models describe how the replicated data is kept consistent across different data-stores, and what the process can expect from the data-store

- Data-centric Consistency Models
  - Models for Specifying Consistency
    - Continuous Consistency Model
  - Models for Consistent Ordering of Operations
    - Sequential Consistency Model
    - Causal Consistency Model

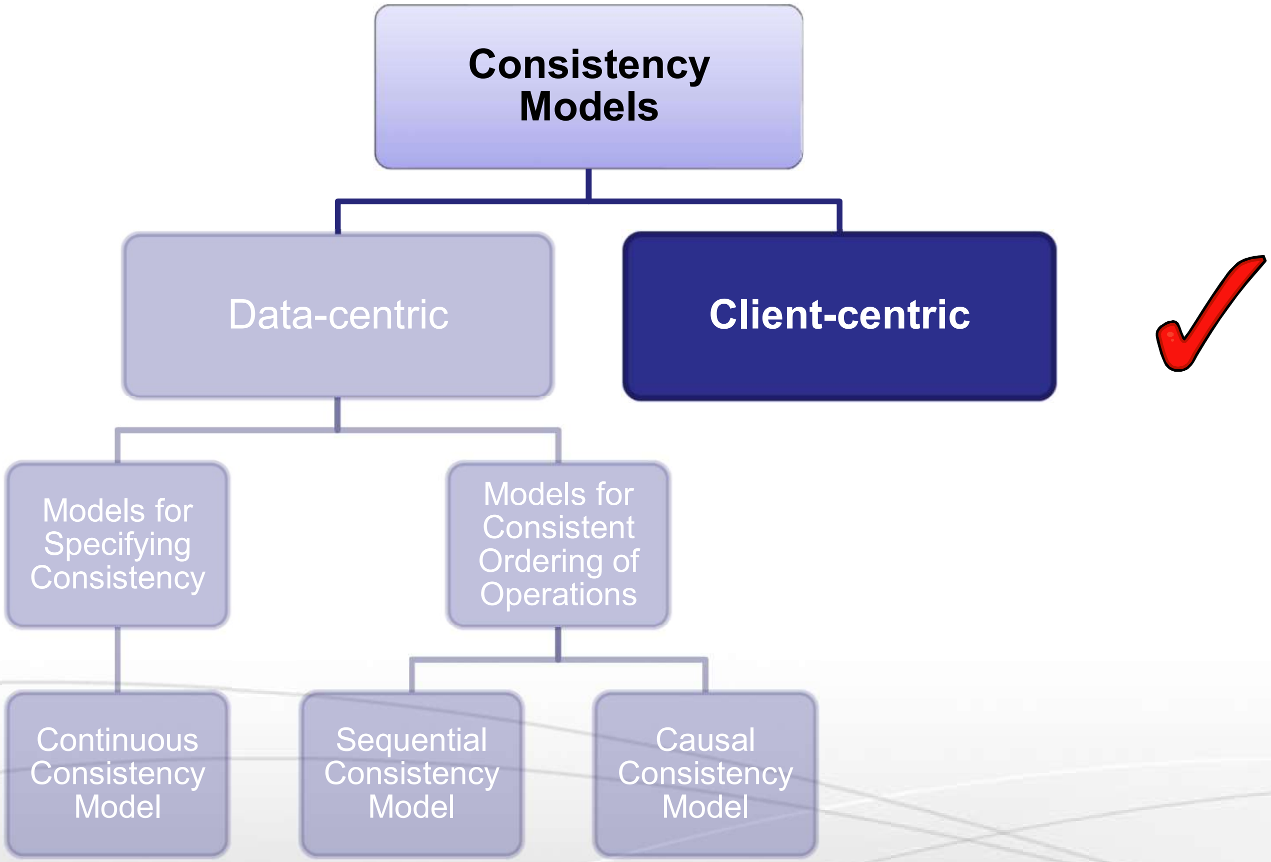
These models allow measuring and specifying the consistency levels that are tolerable to the application

These models specify what ordering of operations are ensured at the replicas

Data-centric models are too strict when:

- most operations are read operations
- updates are generally triggered from one client process

Overview



Consistency Models
Data-centric
Client-centric
Models for Specifying Consistency
Models for Consistent Ordering of Operations
Continuous Consistency Model
Sequential Consistency Model
Causal Consistency Model

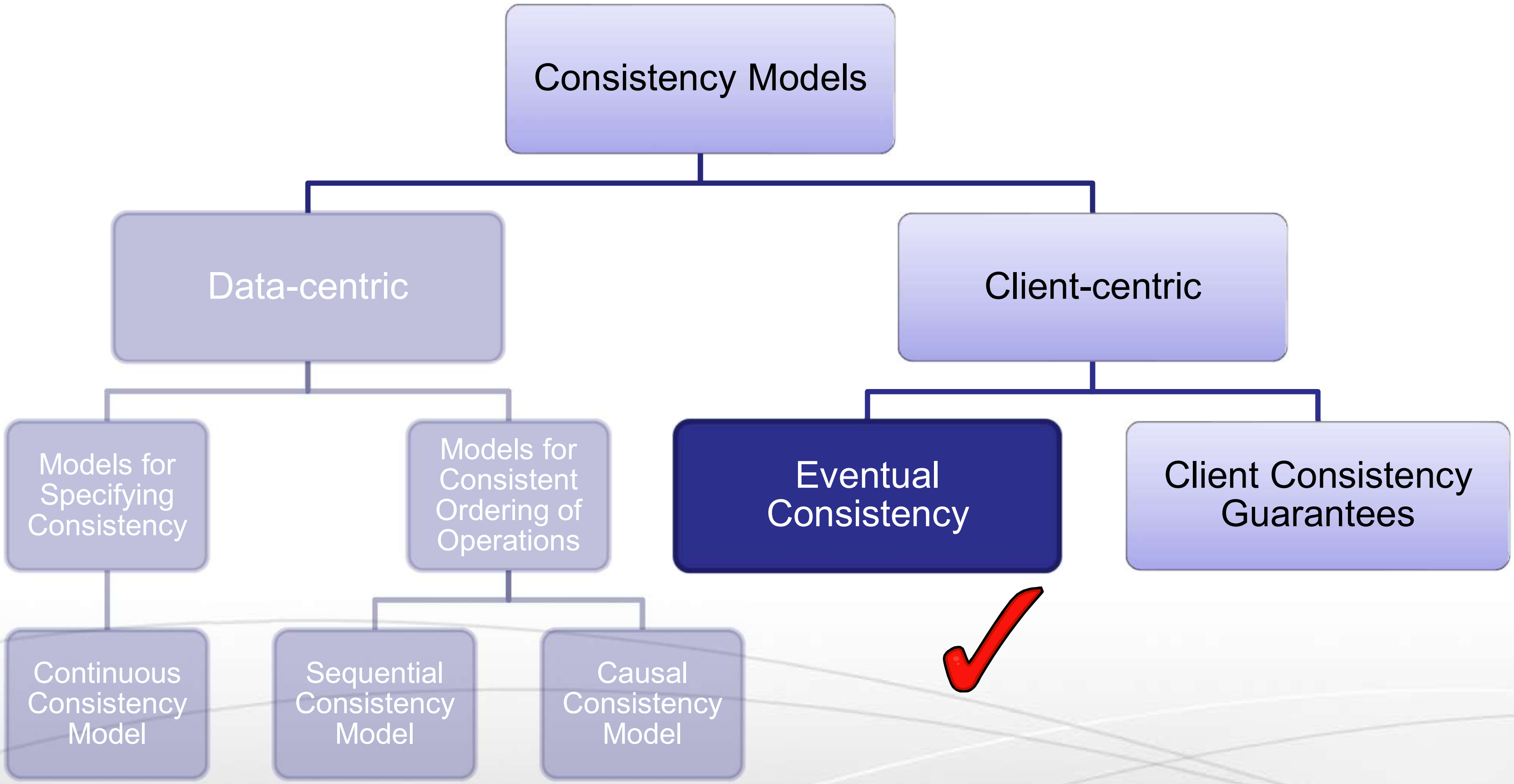
Client-Centric Consistency Models

- Data-centric models lead to excessive overheads in applications where:
  - a majority operations are reads, and
  - updates occur frequently, and are often from one client process

- For such applications, a weaker form of consistency called *Client-centric Consistency* is employed for improving efficiency

- Client-centric consistency models specify two requirements:
  1. Eventual Consistency
     - All the replicas should *eventually* converge on a final value
  2. Client Consistency Guarantees
     - Each client processes should be guaranteed some level of consistency while accessing the data value from different replicas

Overview



Consistency Models
Data-centric
Client-centric
Models for Specifying Consistency
Models for Consistent Ordering of Operations
Eventual Consistency
Client Consistency Guarantees
Continuous Consistency Model
Sequential Consistency Model
Causal Consistency Model

Eventual Consistency

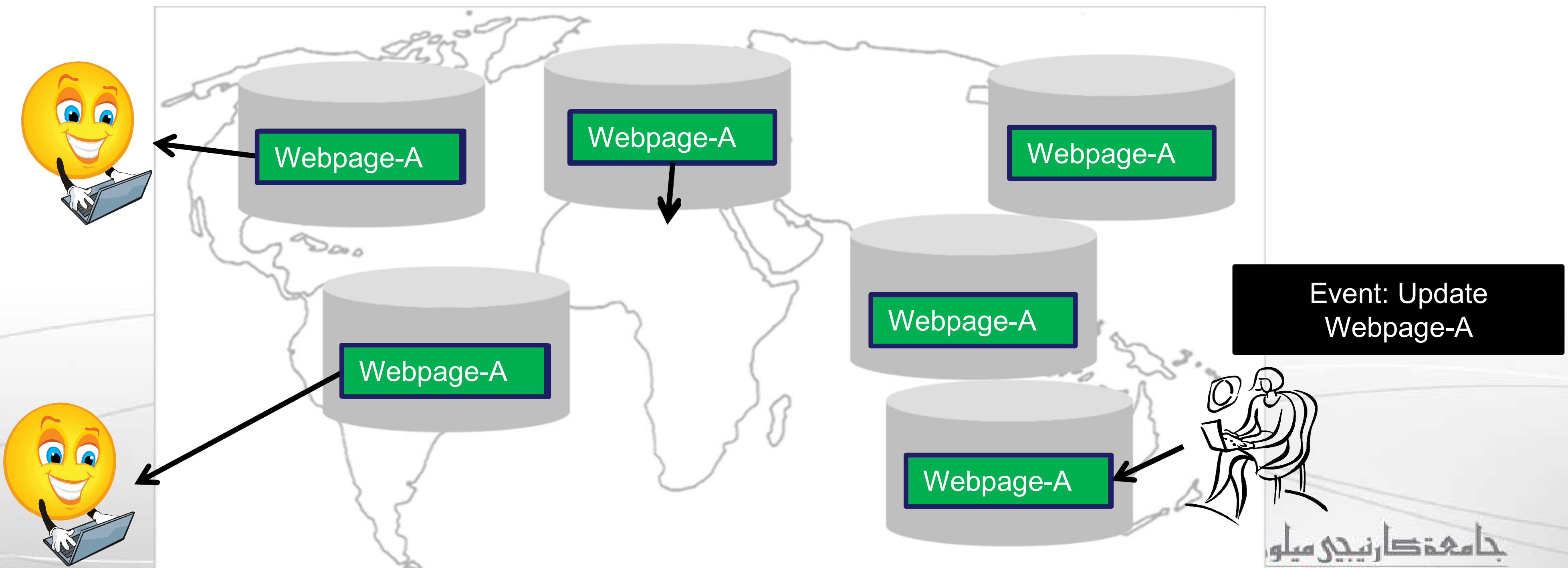
+ Many applications can tolerate a inconsistency for a long time
  + Webpage updates, Web Search – Crawling, indexing and ranking, Updates to DNS Server
+ In such applications, it is acceptable and efficient if replicas in the data-store rarely exchange updates
+ A data-store is termed as *Eventually Consistent* if:
  + All replicas will gradually become consistent in the absence of updates
+ Typically, updates are propagated infrequently in eventually consistent data-stores

Designing Eventual Consistency

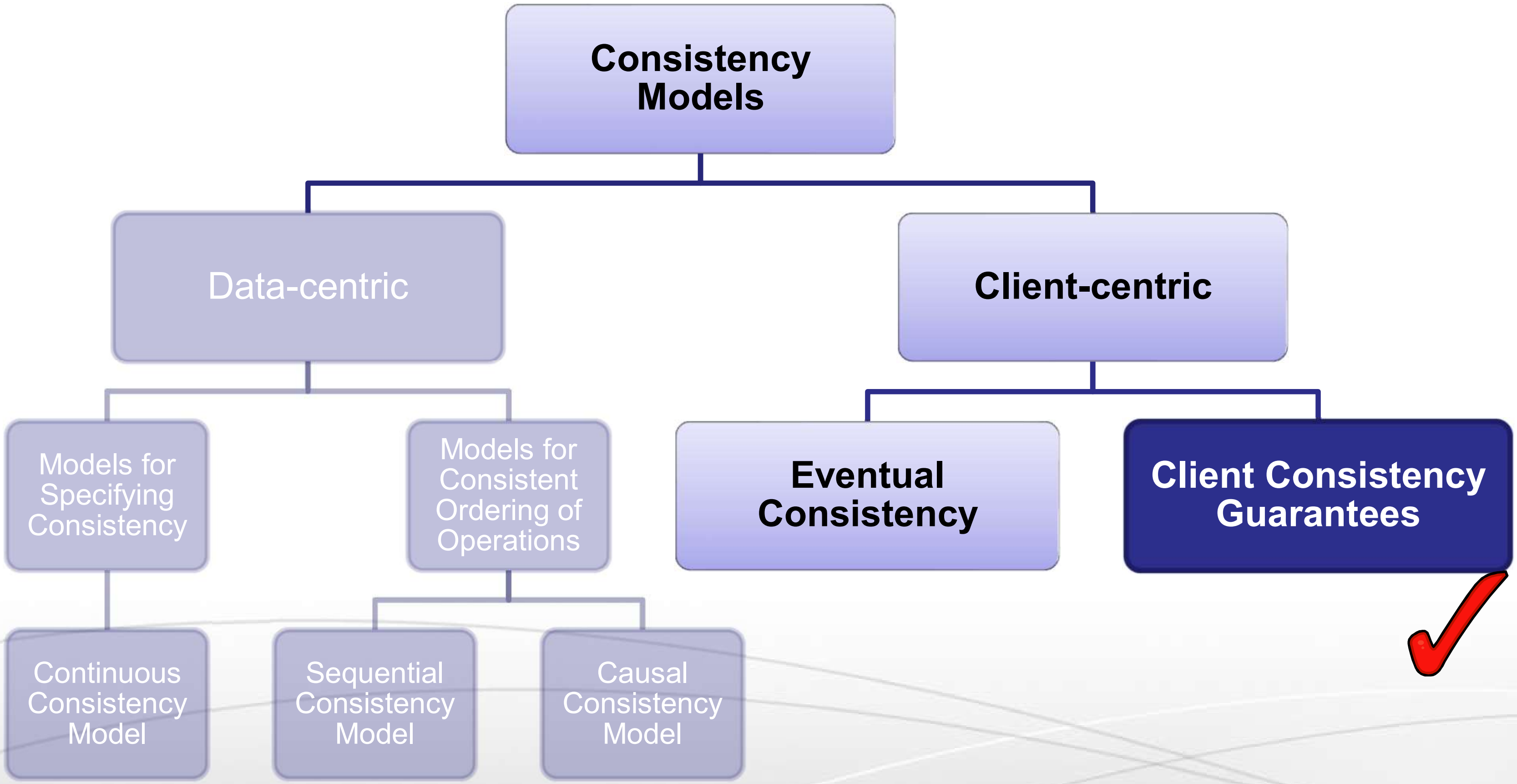
- In eventually consistent data-stores,
  - *Write-write conflicts* are rare
    - Two processes that write the same value are rare
    - Generally, one client updates the data value
      - e.g., One DNS server updates the name to IP mapping
    - Such rare conflicts can be handled through simple mechanisms, such as mutual exclusion
  - R*ead-write conflict* are more frequent
    - Conflicts where one process is reading a value, while another process is writing a value to the same variable
    - Eventual Consistency Design has to focus on efficiently resolving such conflicts

Challenges in Eventual Consistency

+ Eventual Consistency is not good-enough when the client process accesses data from different replicas
  + We need consistency guarantees for a single client while accessing the data-store

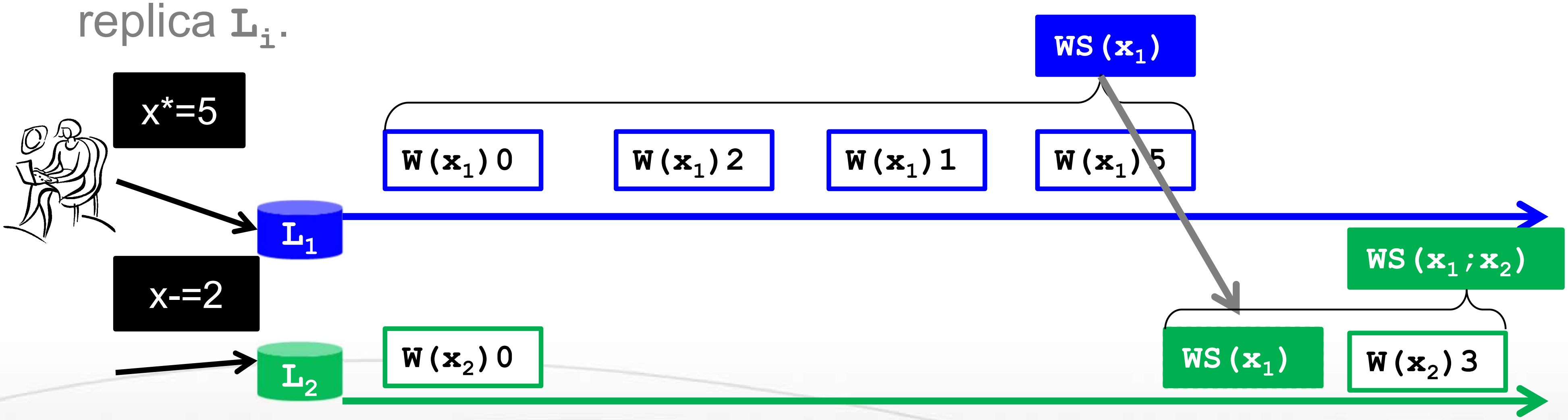


Overview


Consistency Models
Data-centric
Client-centric
Models for Specifying Consistency
Models for Consistent Ordering of Operations
Eventual Consistency
Client Consistency Guarantees
Continuous Consistency Model
Sequential Consistency Model
Causal Consistency Model

Client Consistency Guarantees

- Client-centric consistency provides guarantees for a single client for its accesses to a data-store
- Example: Providing consistency guarantee to a client process for data $\mathbf{x}$ replicated on two replicas. Let $\mathbf{x_i}$ be the local copy of a data $\mathbf{x}$ at replica $\mathtt{L_i}$.



WS(x_1) = **Write Set for $\mathbf{x_1}$** = Series of ops being done at some replica that reflects how $\mathtt{L_1}$ updated $\mathbf{x_1}$ till this time

WS(x_1;x_2) = **Write Set for $\mathbf{x_1}$ and $\mathbf{x_2}$** = Series of ops being done at some replica that reflects how $\mathtt{L_1}$ updated $\mathbf{x_1}$ and, later on, how $\mathbf{x_2}$ is updated on $\mathtt{L_2}$

$\mathtt{L_i}$ = Replica i	R(x_i)b = Read variable x at replica i; Result is b	W(x)b = Write variable x at replica i; Result is b	WS(x_i) = Write Set

Client Consistency Guarantees

+ We will study four types of client-centric consistency models[1]
  1. Monotonic Reads
  2. Monotonic Writes
  3. Read Your Writes
  4. Write Follow Reads

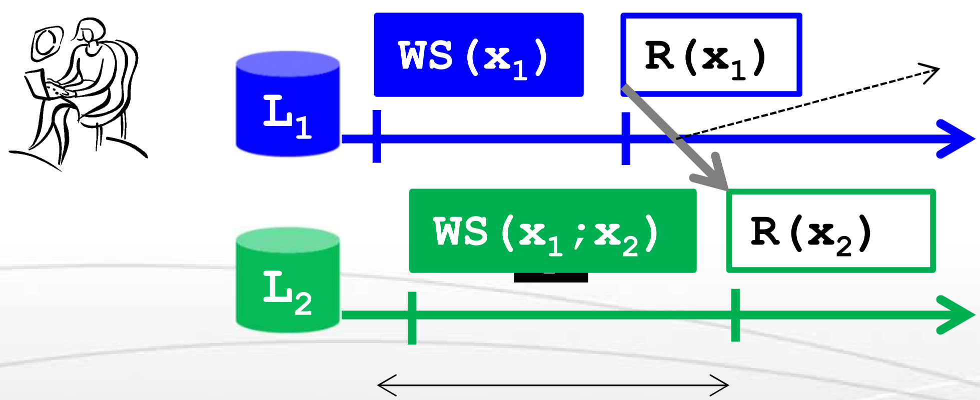
1. The work is based on the distributed database system built by Terry et al. [1]

Overview


Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees
Monotonic Reads
Monotonic Writes
Read Your Writes
Write Follow Reads

Monotonic Reads

- The model provides guarantees on successive reads
- If a client process reads the value of data item **x**, then any successive read operation by that process should return the <u>same</u> or a <u>more recent value</u> for **x**



Return of R(x_2) should at least as recent as R(x_1)

Monotonic Reads – Puzzle

Recognize data-stores that provide monotonic read guarantees

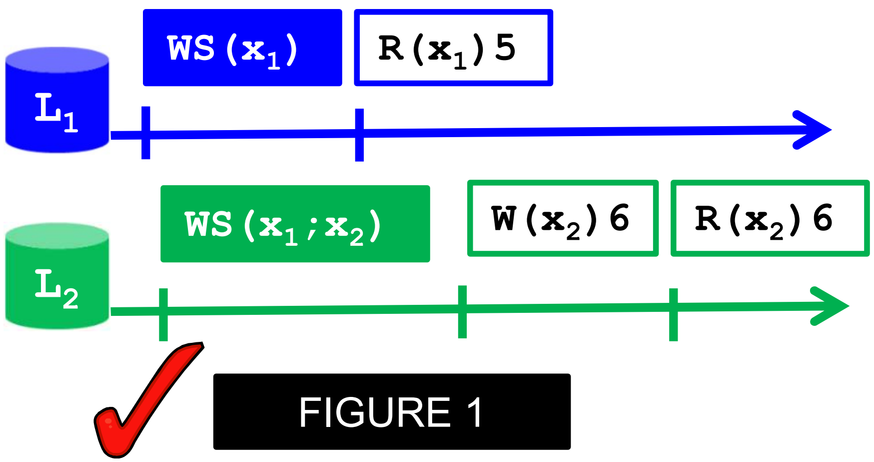


FIGURE 1

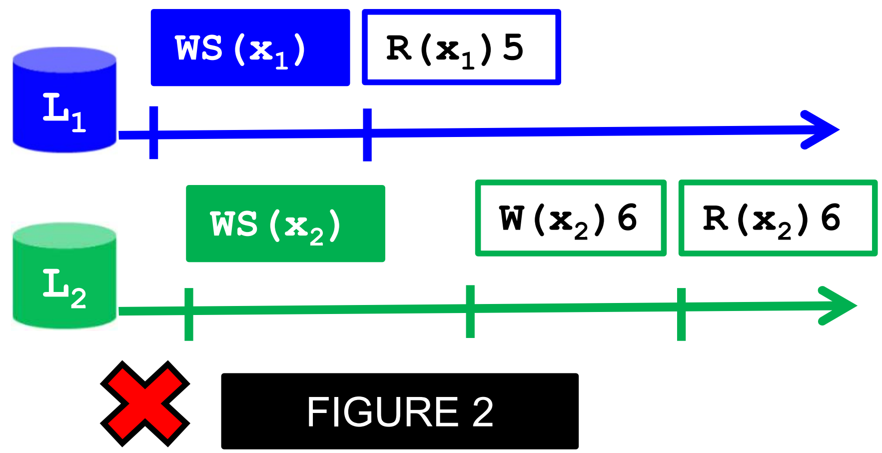


FIGURE 2

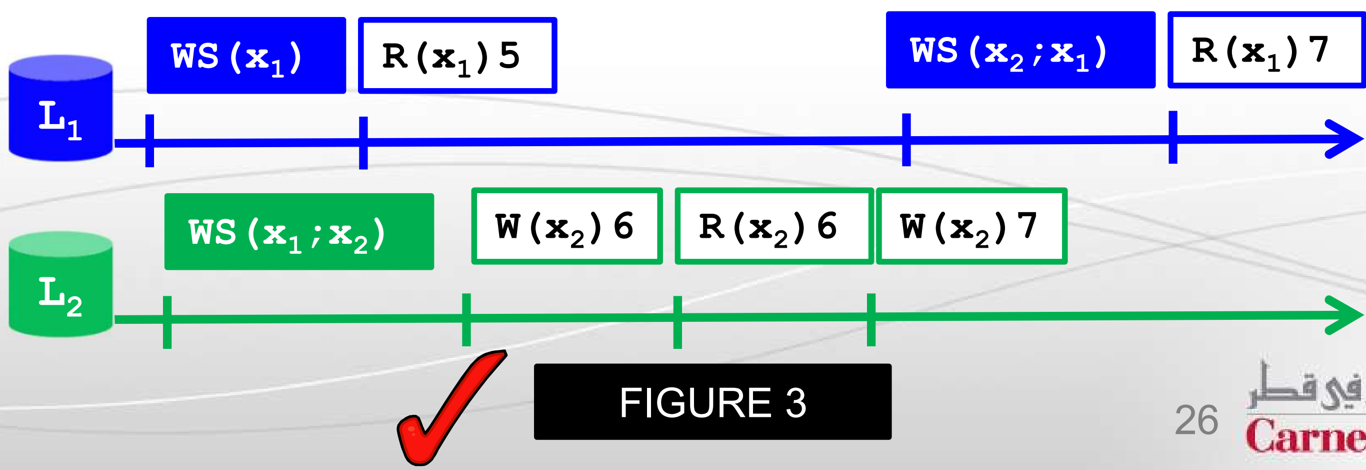


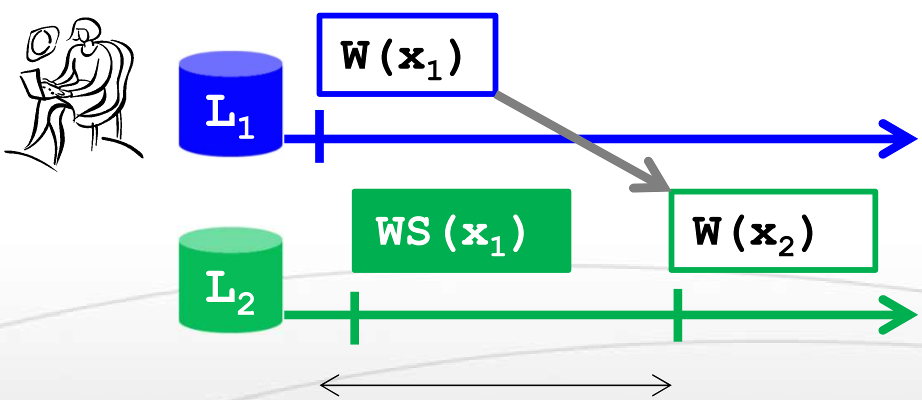
FIGURE 3

Overview

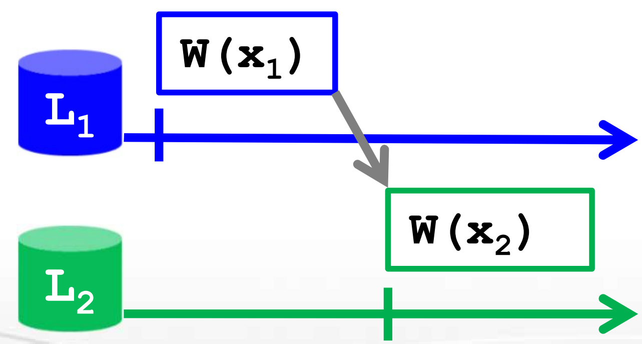

Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees
Monotonic Reads
Monotonic Writes
Read Your Writes
Write Follow Reads

Monotonic Writes

- This consistency model assures that writes are monotonic
- A write operation by a client process on a data item **x** is completed <u>before any successive write</u> operation on **x** by the <u>same process</u>
  - A new write on a replica should wait for all old writes on any replica



$W(x_2)$ operation should be performed only after the result of $W(x_1)$ has been updated at L_2



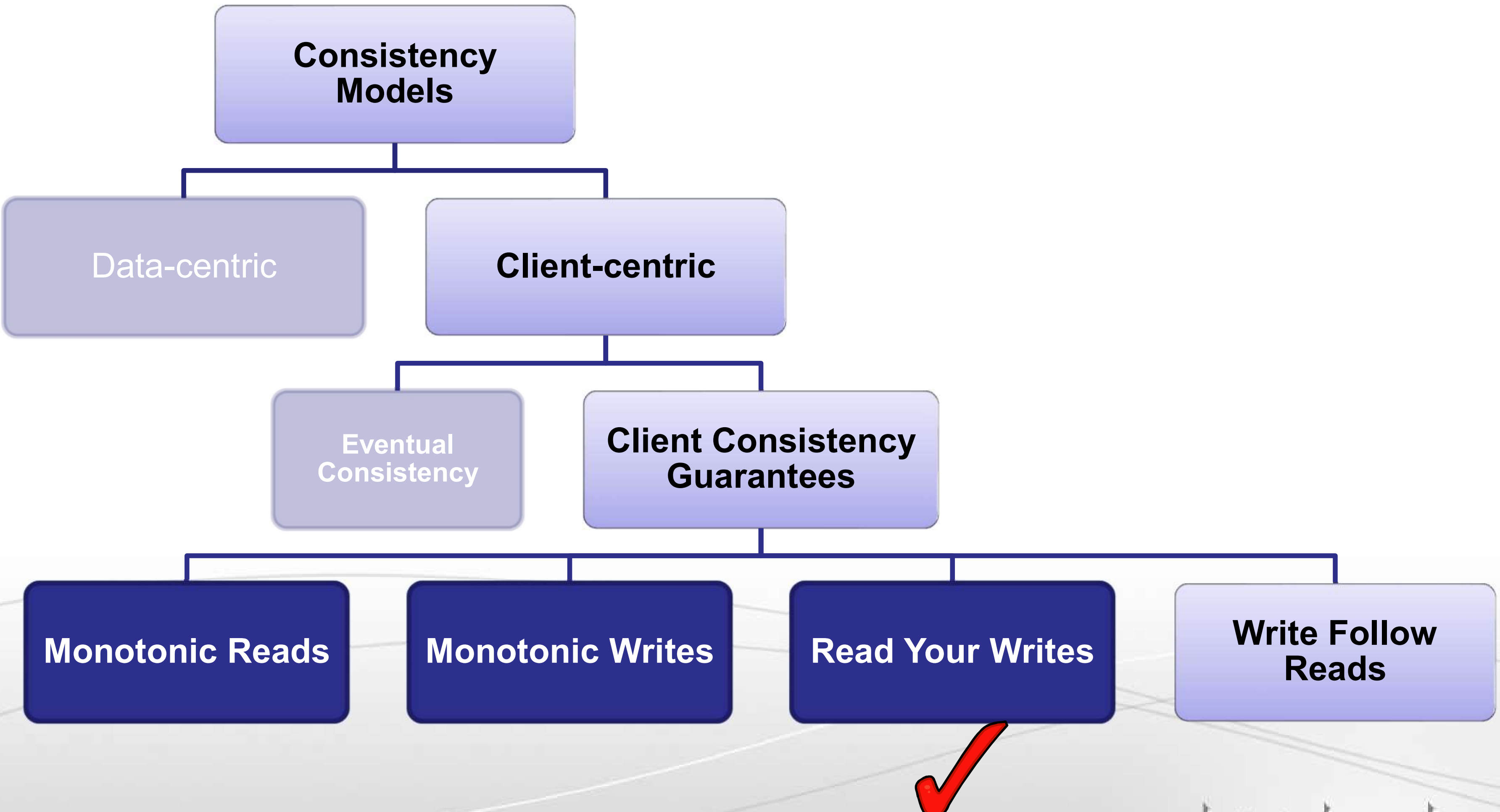
The data-store does not provide monotonic write consistency

Monotonic Writes – An Example

- Example: Updating individual libraries in a large software source code which is replicated
  - Updates can be propagated in a lazy fashion
  - Updates are performed on a part of the data item
    - Some functions in an individual library is often modified and updated
  - Monotonic writes: If an update is performed on a library, then all preceding updates on the same library are first updated

- Question: If the update overwrites the complete software source code, is it necessary to update all the previous updates?

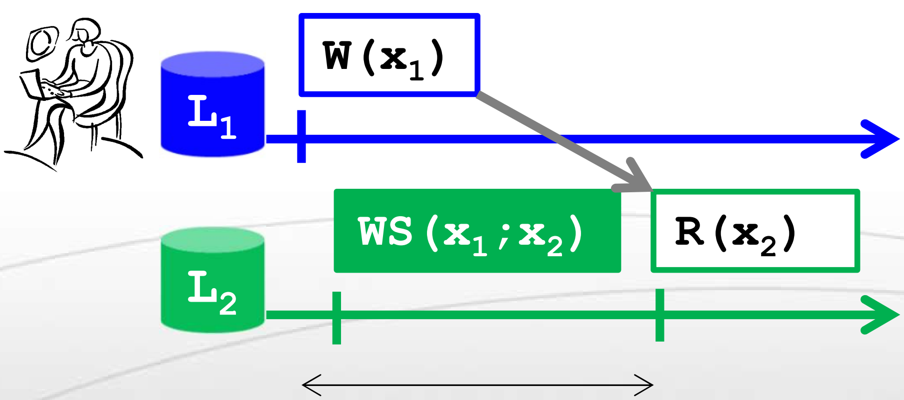
Overview



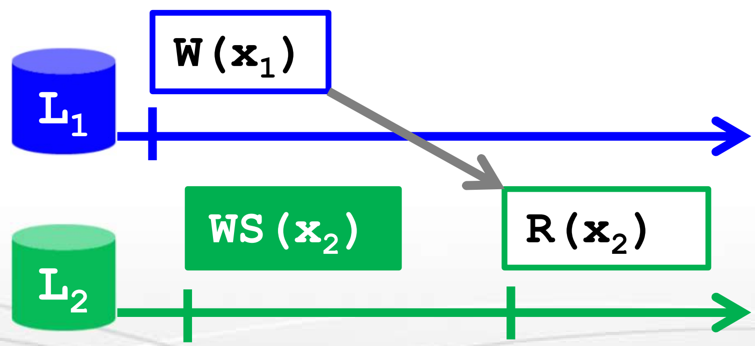
Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees
Monotonic Reads
Monotonic Writes
Read Your Writes
Write Follow Reads

Read Your Writes

+ The <u>effect of a write</u> operation on a data item **x** by a process will <u>always be seen by a successive read</u> operation on **x** by the same process
+ Example scenario:
  + In systems where password is stored in a replicated data-base, the password change should be seen immediately

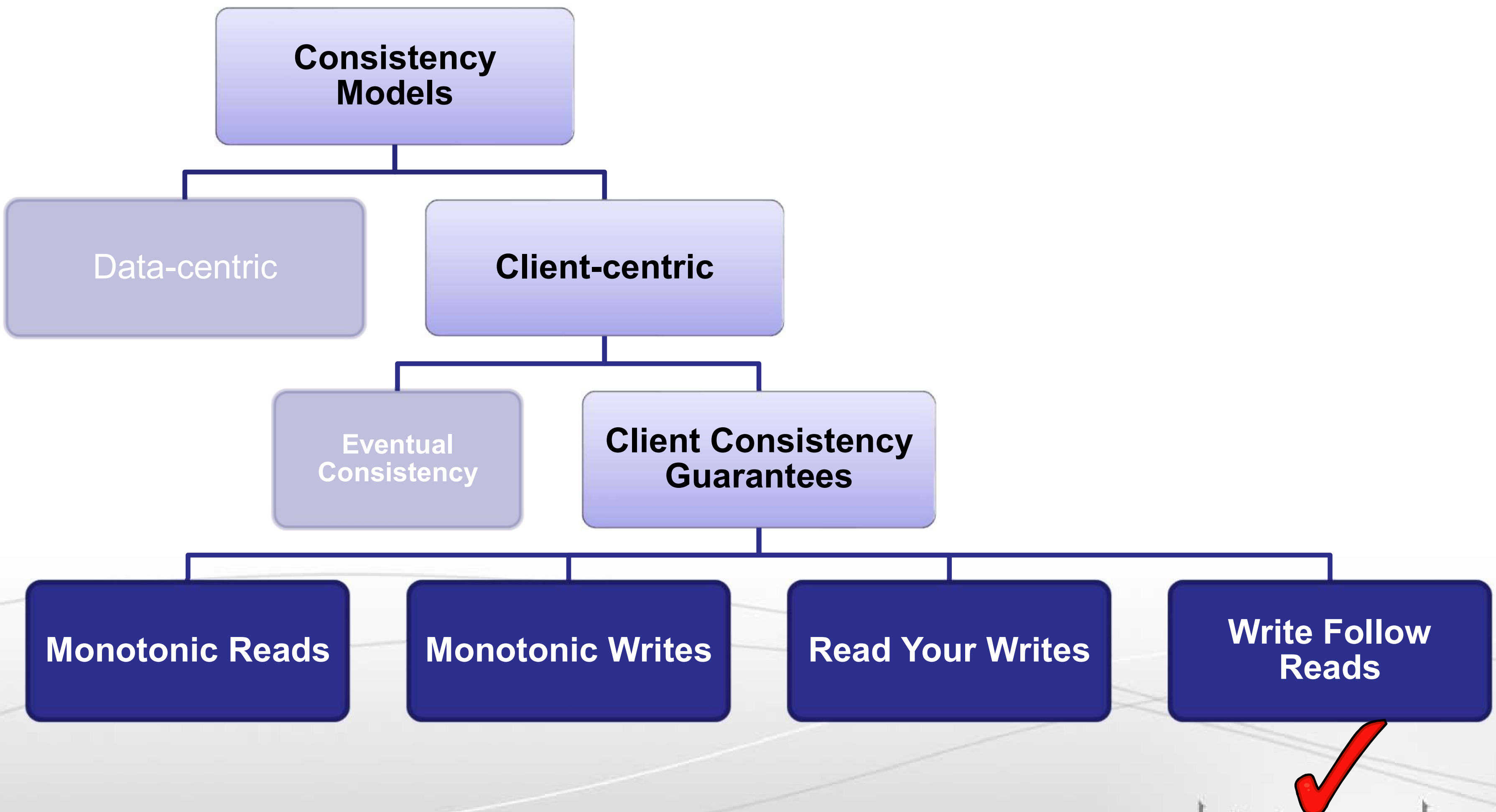


$R(x_2)$ operation should be performed only after the updating the Write Set $WS(x_1)$ at L_2



A data-store that does not provide *Read Your Write* consistency

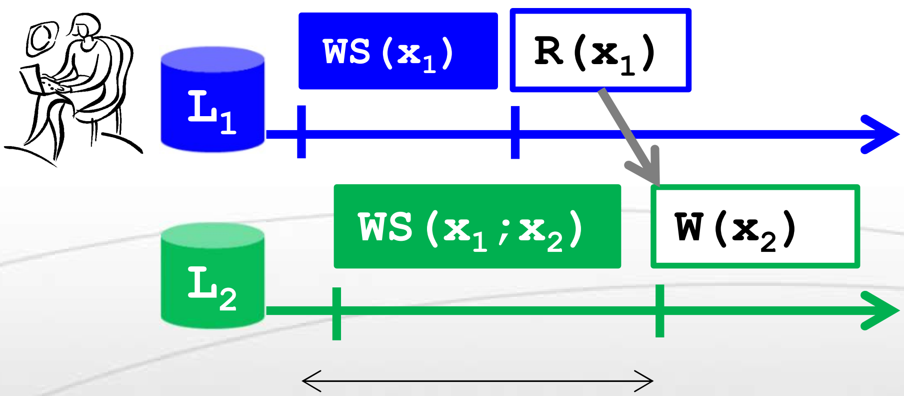
Overview



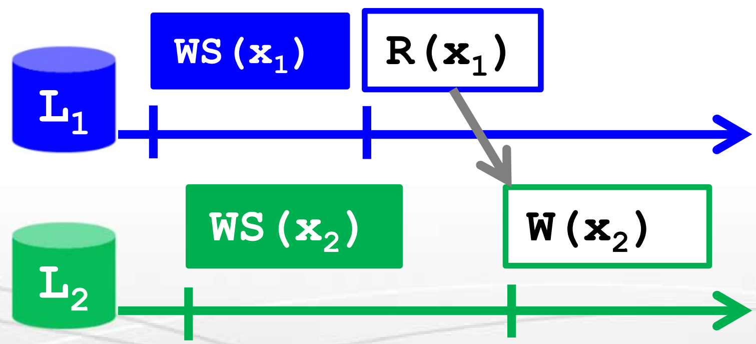
Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees
Monotonic Reads
Monotonic Writes
Read Your Writes
Write Follow Reads

Write Follow Reads

- A write operation by a process on a data item **x** following a previous read operation on **x** by the same process is guaranteed to take place on the same or a more recent value of **x** that was read
- Example scenario:
  - Users of a newsgroup should post their comments only after they have read all previous comments



$\mathtt{W(x_2)}$ operation should be performed only after the all previous writes have been seen

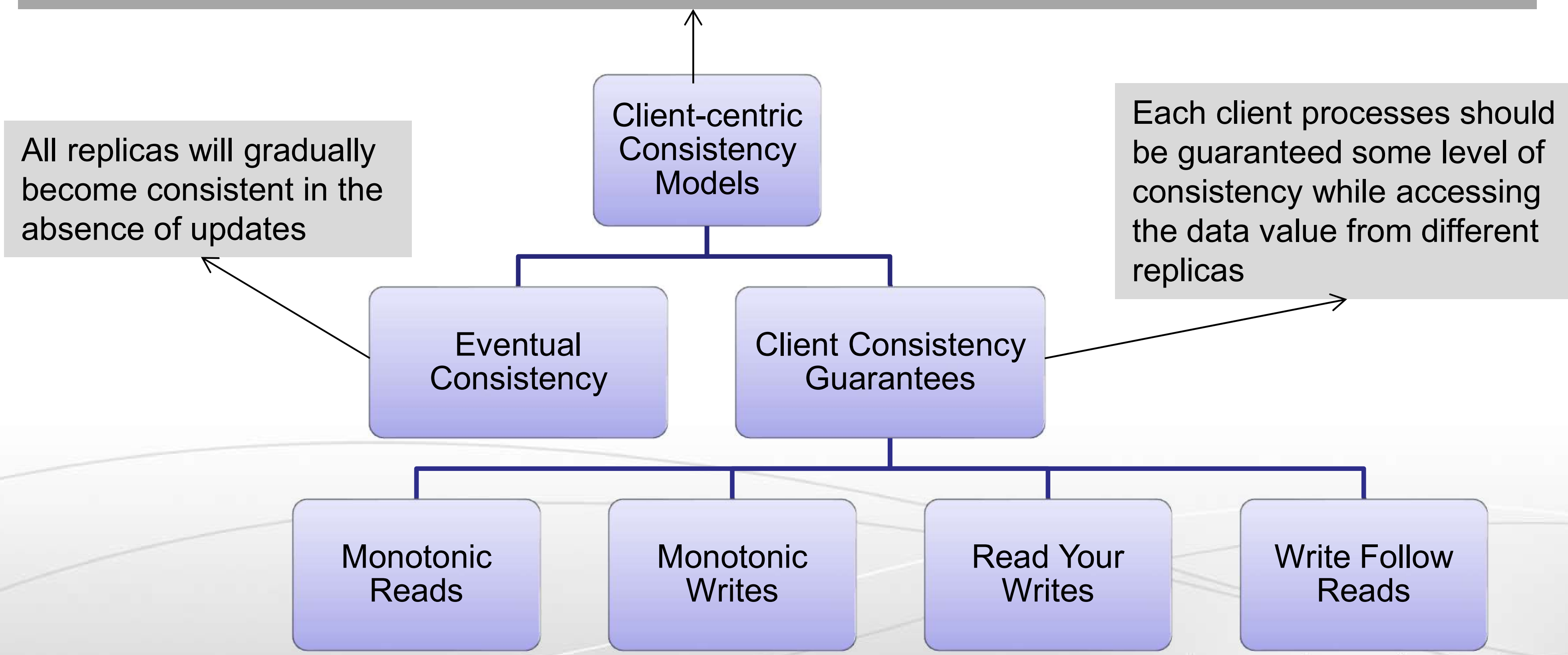


A data-store that does not guarantee Write Follow Read Consistency Model

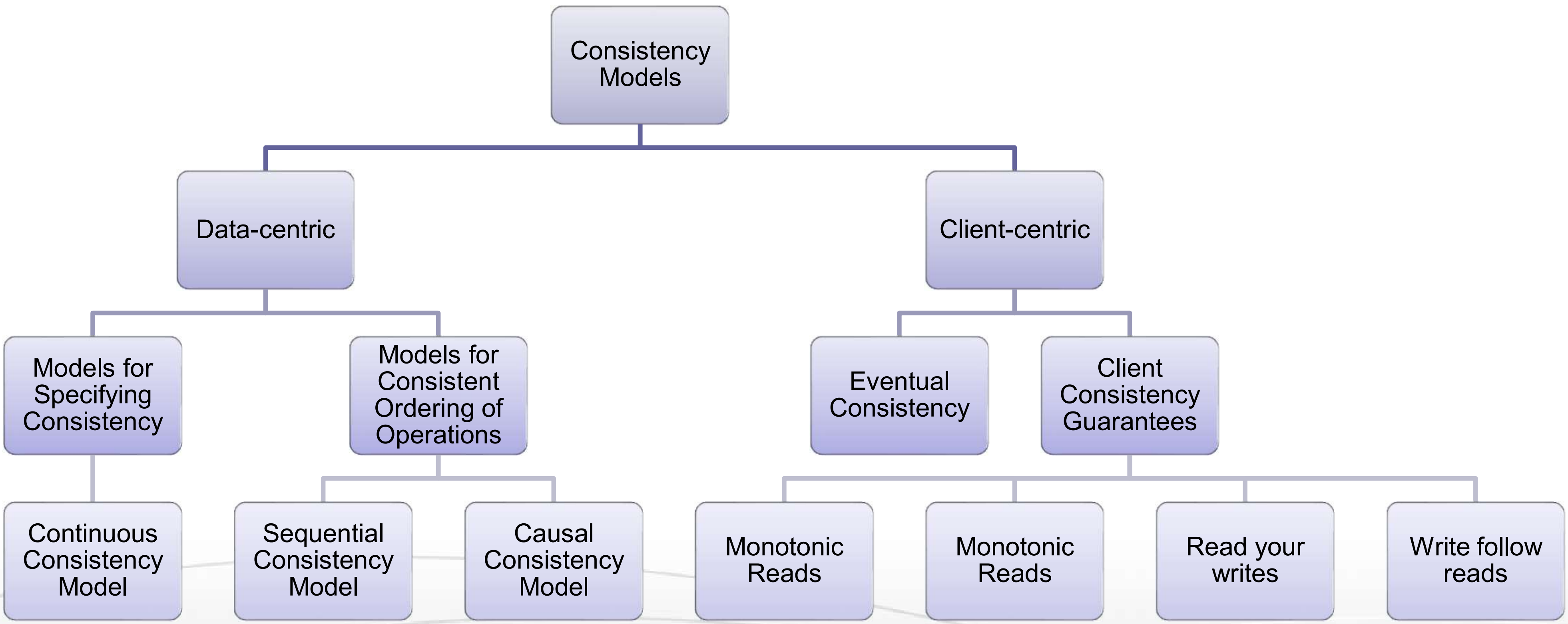
Summary of Client-centric Consistency Models

Client-centric Consistency Model defines how a data-store presents the data value to an individual client when the client process accesses the data value across different replicas.
It is generally useful in applications where:

- one client always updates the data-store.
- read-to-write ratio is high



Topics covered in Consistency Models



Summary of Consistency Models

- Different applications require different levels of consistency
  - Data-centric consistency models
    - Define how replicas in a data-store maintain consistency
  - Client-centric consistency models
    - Provide an efficient, but weaker form of consistency when
    - Here, one client process updates the data item, and many processes read the replica

Next Class

+ Replica Management
  + Describes where, when and by whom replicas should be placed

+ Consistency Protocols
  + We study “how” consistency is ensured in distributed systems

References

+ [1] Terry, D.B., Demers, A.J., Petersen, K., Spreitzer, M.J., Theimer, M.M., Welch, B.B., "Session guarantees for weakly consistent replicated data", Proceedings of the Third International Conference on Parallel and Distributed Information Systems, 1994
+ [2] Lili Qiu, Padmanabhan, V.N., Voelker, G.M., “On the placement of Web server replicas”, Proceedings of IEEE INFOCOM 2001.
+ [3] Rabinovich, M., Rabinovich, I., Rajaraman, R., Aggarwal, A., “A dynamic object replication and migration protocol for an Internet hosting service”, Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS), 1999
+ [4] http://www.cdk5.net