Today

- Data Replication
  - Examples and Issues
- Data Consistency
  - Consistency Models

Data Replication

- Using multiple copies of same data
- Why do we need data replication?

Example: Distributed Shared Memory

- Multiprocessor system with per-CPU cache
  - Different caches may hold same cache line
- Why do we need replication?
- What happens when a CPU writes a data item?
Example: DNS
- Each zone has multiple replicas
  - One primary and other secondary
  - Other servers/clients may cache data
- Why do we need replication?
- What happens if name resolution changes?

Example: Web Caching
- Server is primary replica
- Web pages are cached at:
  - Server replicas
  - Client browsers
  - Proxy caches
  - Content-distributions networks (CDNs)
- Why do we need replication?
- What happens when a Web page changes?
- What about stock tickers, live sports scores, weather reports?

Data Replication: Issues
- What happens if multiple processes write concurrently?
- How do we propagate updates to all replicas?
- What is the cost of updation/consistency?

Data Consistency
- How do we define “consistency”?
- What level of consistency is required in case of:
  - DSM?
  - DNS?
  - Web caching?
Consistency Models

- “Contract” between processes and data store
  - "Guarantees" on the view of data store visible to each process
  - What writes will be visible to whom and when?

Data-Centric Consistency Models

- Defined in terms of the values stored in the replicas
  - How much can the values differ from each other?

- Consistency can be defined in terms of:
  - Ordering of reads/writes
  - Deviation in numerical values or staleness of replicas

Consistency Models

- Data-centric consistency models
  - How to provide consistent views of the data store to all replicas?
  - Typically assumes multiple concurrent writes/reads

- Client-centric consistency models
  - How to provide consistent views of the data store to a client?
  - Typically assumes limited concurrent writes, but client can move

Ordering-based Consistency

- Different processes read and write to replicas of shared data concurrently
- What ordering will these reads and writes appear to different processes?
**Strict Consistency**

- Any read to x returns the most recent write to x
  - Assumes notion of absolute global time

  
<table>
<thead>
<tr>
<th>Process</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>W(x)a</td>
</tr>
<tr>
<td>P2</td>
<td>R(x)a</td>
</tr>
<tr>
<td>P1</td>
<td>W(x)a</td>
</tr>
<tr>
<td>P2</td>
<td>R(x)b</td>
</tr>
</tbody>
</table>

- Is it feasible in a distributed system?

**Sequential Consistency**

- All processes see the same sequence of operations
  - Each process’s operations appear in program order
  - Any valid interleaving of multiple process operations
  - No notion of absolute time

  
<table>
<thead>
<tr>
<th>Process</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>W(x)a</td>
</tr>
<tr>
<td>P2</td>
<td>W(x)b</td>
</tr>
<tr>
<td>P3</td>
<td>R(x)b</td>
</tr>
<tr>
<td>P4</td>
<td>R(x)c</td>
</tr>
</tbody>
</table>

**Causal Consistency**

- Causally related writes must be seen in the same order by all processes
  - Concurrent writes can be seen in any order

  
<table>
<thead>
<tr>
<th>Process</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>W(x)a</td>
</tr>
<tr>
<td>P2</td>
<td>R(x)a</td>
</tr>
<tr>
<td>P3</td>
<td>R(x)b</td>
</tr>
<tr>
<td>P4</td>
<td>R(x)c</td>
</tr>
</tbody>
</table>

  
<table>
<thead>
<tr>
<th>Process</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>W(x)a</td>
</tr>
<tr>
<td>P2</td>
<td>R(x)b</td>
</tr>
<tr>
<td>P3</td>
<td>R(x)c</td>
</tr>
<tr>
<td>P4</td>
<td>R(x)a</td>
</tr>
</tbody>
</table>

**FIFO Consistency**

- All writes from a process are seen in the same order by all processes
  - Order is the order of issue
  - Writes by different processes may be seen in different order

  
<table>
<thead>
<tr>
<th>Process</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>W(x)a</td>
</tr>
<tr>
<td>P2</td>
<td>W(x)b</td>
</tr>
<tr>
<td>P3</td>
<td>R(x)b</td>
</tr>
<tr>
<td>P4</td>
<td>R(x)c</td>
</tr>
</tbody>
</table>

  
<table>
<thead>
<tr>
<th>Process</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>W(x)a</td>
</tr>
<tr>
<td>P2</td>
<td>W(x)b</td>
</tr>
<tr>
<td>P3</td>
<td>R(x)a</td>
</tr>
<tr>
<td>P4</td>
<td>R(x)b</td>
</tr>
</tbody>
</table>

  (a)
**Synchronization-based Consistency**

- Many processes access shared data inside critical sections
  - Do not care about all reads/writes to be consistent
  - Only require values to be consistent at beginning and end of critical sections
  - Do not need to pass intermediate updates
  - Synchronization variables used to trigger data synchronizations
  - Makes all copies consistent

**Weak Consistency**

- At synchronization:
  - All local writes are flushed out everywhere
  - All remote writes are gathered in
  - All accesses to synchronization variables are sequentially consistent
  - Ensures sequential consistency on groups of operations

\[
\begin{array}{ccc}
P_1 & W(x)a & W(x)b & S \\
P_2 & & R(x)b & R(x)a & S & R(x)b \\
P_3 & R(x)a & R(x)b & S & R(x)b \\
\end{array}
\]

**Variations of Weak Consistency**

- Release Consistency:
  - Separate synchronization operations for entry and exit from critical sections
  - Gather remote writes on entry (acquire), flush out local writes on exit (release)

- Entry Consistency:
  - Separate synchronization variable for each data item
  - Avoids false sharing, multiple non-overlapping critical sections

**Summary of Ordering-based Data-Centric Consistency Models**

- Strict: Absolute time-based
- Sequential: All processes see same order of operations
- Causal: Causally-related operations in same order
- FIFO: Ordered per-process basis
- Weak: Flush/gather at each synchronization
Continuous Consistency

- Consistency defined as a bound on deviations between replicas
  - Bound on a continuous scale
  - Could be numerical, time-based
- Conit: Consistency unit
  - Data unit over which consistency is defined
  - E.g.: Individual stocks in a stockticker, OR, whole set of stocks in an index
- What is the tradeoff between having a fine-grained vs. coarse-grained conit?

Continuous Consistency - Deviations

- Inconsistencies between replicas are measured in terms of deviations
- Numerical deviation: If data is numerical
  - Absolute or relative
  - Number of updates: Referred to as "weight"
- Staleness: How fresh is a replica?
  - Must be updated with certain frequency

Eventual Consistency

- In absence of updates, all replicas converge towards identical copies
- Applied to a replicated data store with few updaters and many readers
  - Only requirement: an update should eventually propagate to all replicas
  - Nothing assumed about the timeliness of update propagation
  - Cheap to implement
- E.g.: Web, DNS

Client-Centric View of Data

- A client may only care about the data it is reading and writing
  - E.g.: a user may only care about the posts on their Facebook wall
- These should be in consistent order. Can use:
  - Data-centric consistency models. Problem?
  - Eventual consistency model. Problem?
Client-Centric Consistency Models
- Defined in terms of the values seen by a single client at different replicas
  - Assume: a client can move between replicas
- Useful for:
  - Mobile applications
  - Applications with multiple access points, e.g.: email
- Ordering-based consistency:
  - In what order will a single client see its reads and writes on different replicas?
  - Different combinations based on read vs. write ordering

Monotonic Reads
- If a process reads a value of x, any successive read of x by it will return the same or a more recent value
  - E.g.: Reading the posts from different locations
    - L1: WS(x1) R(x1)
    - L2: WS(x1; x2) R(x2)

Monotonic Writes
- If a process writes to x, this write will be completed before any successive write to x by it
  - E.g.: All outgoing posts from different locations
    - L1: W(x1)  L1: W(x1)
    - L2: WS(x1) W(x2)  L2: W(x2)

Read Your Writes
- A write to x by a process will always be seen by a successive read of x by it
  - E.g.: You can see your earlier posts
    - L1: W(x1)  L1: W(x1)
    - L2: WS(x1; x2) R(x2)  L2: WS(x2) R(x2)
**Writes Follow Reads**

- If a process reads a value of x, any successive write to x by it will take place on the same or a more recent value.
- Your post will reflect any postings you’ve read earlier.

<table>
<thead>
<tr>
<th></th>
<th>Read first</th>
<th>Write first</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read next</strong></td>
<td>Monotonic Reads</td>
<td>Read Your Writes</td>
</tr>
<tr>
<td><strong>Write next</strong></td>
<td>Writes Follow Reads</td>
<td>Monotonic Writes</td>
</tr>
</tbody>
</table>