CSCI 5105

Instructor: Abhishek Chandra

Today

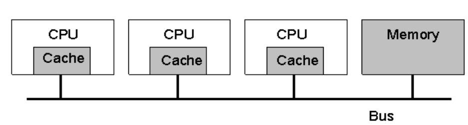
- Data Replication
  - Examples and Issues
- Data Consistency
  - Consistency Models

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?

5

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?

6

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?

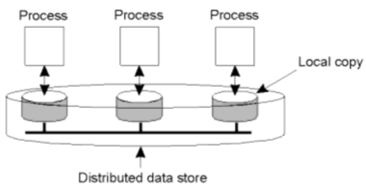
7

Data Consistency

- How do we define "consistency"?
- What level of consistency is required in case of:
  - DSM?
  - DNS?
  - Web caching?

8

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?

9

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

10

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

11

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?

12

Strict Consistency

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

P1	W(x)a		
P2		R(x)a	R(x)a

P1	W(x)a		
P2		R(x)NIL	R(x)a

P1	W(x)a	W(x)b	
P2			R(x)b

P1	W(x)a	W(x)b	
P2			R(x)a

- Is it feasible in a distributed system?

13

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

P1:	W(x)a				
P2:		W(x)b			
P3:			R(x)b		R(x)a
P4:				R(x)b	R(x)a

(a)

14

Causal Consistency

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

P1:	W(x)a			
P2:	R(x)a	W(x)b		
P3:			R(x)a	R(x)b
P4:			R(x)a	R(x)b

P1:	W(x)a			
P2:	R(x)a	W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

P1:	W(x)a		
P2:	W(x)b		
P3:		R(x)b	R(x)a
P4:		R(x)a	R(x)b

P1:	W(x)a			
P2:	R(x)a	W(y)b		
P3:			R(y)b	R(x) ?
P4:			R(x)a	R(y) ?

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

P1	W(x)a					
P2	R(x)a	W(x)b	W(x)c			
P3				R(x)a	R(x)b	R(x)c
P4				R(x)b	R(x)a	R(x)c

16

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 (or locks) used to trigger data synchronizations
  - Makes all copies consistent

17

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

P1	W(x)a	W(x)b	S					
P2				R(x)b	R(x)a	S	R(x)b	
P3				R(x)a	R(x)b	S	R(x)b	

18

Other Synchronization-based Consistency Models

- 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

19

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
- Synchronization-based: Flush/gather at each synchronization

20

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?

21

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

22

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

23

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?

24

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

25

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 W1(x1)**R1(x1)**
L2 W2(x1;x2) **R1(x2)**

L1 W1(x1)**R1(x1)**
L2 W2(x1|x2) **R1(x2)**

26

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 **W1(x1)**
L2 W2(x1;x2) **W1(x2;x3)**

L1 **W1(x1)**
L2 W2(x1|x2) **W1(x1|x3)**

L1 **W1(x1)**
L2 W2(x1|x2) **W1(x2;x3)**

L1 **W1(x1)**
L2 W2(x1|x2) **W1(x1;x3)**

27

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 **W1(x1)**
L2 W2(x1; x2) **R1(x2)**

L1 **W1(x1)**
L2 W2(x1|x2) **R1(x2)**

28

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

L1 W1(x1) **R2(x1)**
L2 W3(x1;x2) **W2(x2;x3)**

L1 W1(x1) **R2(x1)**
L2 W3(x1|x2) **W2(x1|x2)**

29

Summary of Client-Centric Consistency Models

	Read first	**Write first**
Read next	Monotonic Reads	Read Your Writes
Write next	Writes Follow Reads	Monotonic Writes

30