### Distributed Systems (COMP9243)

Lecture 5 (A): Replication & Consistency

---

**Slide 1**

- A Replication
- B Consistency
  - Models vs Protocols
- B Update propagation

---

**Slide 2**

**Replication**

Make copies of services on multiple machines.

Why?:

- Reliability
- Redundancy
- Performance
- Increase processing capacity
- Reduce communication
- Scalability (prevent centralisation)
  - Prevent overloading of single server (size scalability)
  - Avoid communication latencies (geographic scalability)

---

**Slide 3**

**Data vs Control Replication**

Data Replication (Server Replication/Mirroring):

- FTP Server
- FTP

---

**Slide 4**

Data Replication (Caching):

- Web Server
- HTTP

---

What’s the difference between mirroring and caching?
**Control Replication:**

Slide 5

What are the challenges of doing this?

**Data and Control Replication:**

Slide 6

We will be looking primarily at data replication (including combined data and control replication).

---

**Replication Issues**

Updates
- Consistency (how to deal with updated data)
- Update propagation

Replica placement
- How many replicas?
- Where to put them?

Redirect/Routeing
- Which replica should clients use?

**Distributed Data Store**

Data Store

Client’s Point of View:

Slide 8
Distributed Data-Store’s Point of View:

- Client A
- Client B
- Client C
- Client D
- Replica 1
- Replica 2
- Replica 3
- Replica 4

Data Model:
- data item: simple variable
- data item values: explicit (0, 1), abstract (a, b)
- data store: collection of data items

Operations on a Data Store:
- Read: R_i(x) \rightarrow b
  Client i performs a read for data item x and it returns b
- Write: W_i(x) \rightarrow a
  Client i performs write on data item x setting it to a
- Operations not instantaneous
  - Time of issue (when request is sent by client)
  - Time of execution (when request is executed at a replica)
  - Time of completion (when reply is received by client)
- Coordination among replicas

Replica Managers:

Timeline:
- Client A/Replica 1: WA(x)1, WA(x)0
- Client B/Replica 2: RB(x)1, RB(x)1, RB(x)0

Slide 10

Slide 11

Slide 12
CONSISTENCY

Conflicting Data:
✓ Do replicas have exactly the same data?
✓ What differences are permitted?

Consistency Dimensions:
✓ Time and Order

Time:
✓ How old is the data (staleness)?
✓ How old is the data allowed to be?
• Time, Versions

Operation order:
✓ Were operations performed in the right order?
✓ What orderings are allowed?

Real world examples of inconsistency?

ORDERING

Updates and concurrency result in conflicting operations

Conflicting Operations:
✓ Read-write conflict (only 1 write)
✓ Write-write conflict (multiple concurrent writes)
✓ The order in which conflicting operations are performed affects consistency

Partial vs Total Ordering:
✓ partial order: order of a single client’s operations
✓ total order: interleaving of all conflicting operations

Example:
Client A: x = 1; x = 0; Possible results:
Client B: print(x);
print(x);
- -, 11, 10, 00
How about 01?

What are the conflicting ops? What are the partial orders? What are the total orders?

Can you sanely use a system like this?

CONSISTENCY MODEL

Defines which interleavings of operations are valid (admissible)

Consistency Model:
✓ Concerned with consistency of a data store.
✓ Specifies characteristics of valid total orderings

A data store that implements a particular model of consistency will provide a total ordering of operations that is valid according to the model.
Data Coherence vs Data Consistency:

**Data Coherence** ordering of operations for single data item
- e.g. a read of x will return the most recently written value of x

**Data Consistency** ordering of operations for whole data store
- implies data coherence
- includes ordering of operations on other data items too

Non-distributed data store:
- Data coherence is respected
- Program order is maintained

---

**DATA-CENTRIC CONSISTENCY MODEL**

A contract, between a distributed data store and clients, in which the data store specifies precisely what the results of read and write operations are in the presence of concurrency.

- Multiple clients accessing the same data store
- Described consistency is experienced by all clients
  - Client A, Client B, Client C see same kinds of orderings
- Non-mobile clients (replica used doesn’t change)

---

**STRONG ORDERING VS WEAK ORDERING**

**Strong Ordering (tight):**
- All writes must be performed in the order that they are invoked
- Example: all replicas must see: \( W(z)_a \ W(z)_b \ W(z)_c \)
- Strict (Linearisable), Sequential, Causal, FIFO (PRAM)

**Weak Ordering (loose):**
- Ordering of groups of writes, rather than individual writes
- Series of writes are grouped on a single replica
- Only results of grouped writes propagated.
- Example: \( \{W(z)_a \ W(z)_b \ W(z)_c \} \Rightarrow \{W(z)_a \ W(z)_c \} \Rightarrow \{W(z)_c \} \)
- Weak, Release, Entry