Replicated State Machines:

**Goal:** - Reliable system from unreliable components
  - Consistent replicas of a server

**Single-copy consistency:** System behaves as if only one copy of server

**Eventual consistency:** - Replicas eventually agree on state
  - Perform conflict resolution
  - Might read stale data.

How to achieve single-copy consistency?

**Idea:**
  - Replicas start in same state
  - Apply same operations in deterministic order

**Key Issue:** How to agree on order
One method: Primary Backup

- Primary determines order, forwards operations to backup
- Backup ignores client requests
- When primary dies, backup is promoted to primary

Problem: How does backup know primary dead and not just partitioned? How do clients learn backup is new primary?

Solve: View server that monitors primary + backup and performs promotion on failure.

Problem: View server vulnerable

Want distributed consensus

Advanced topic, take 6.824 to learn more
**PNUTS:**

- Replicated storage system across Wide Area
  - Full replica in each region
    - Not single-copy consistent (Too slow)
    - Not Eventually consistent
      - Why not?

**Spring Break problem:**

1. update ACL
2. update photo list
3. read photo list

Eventually consistency causes this short-term problem
Solution: **Per-Record Timeline Consistency**

- updates to single record happen in same order on all replicas

**How?**

- writes go to **master Region** for record

- Yahoo Message Broker at master region chooses order and guarantees same order applied at other regions

- If master not local, writes are slow. Therefore master changes to region by majority of writes.

**API:**

- **read-any(key):** read local copy
  
  fast, but might be stale

- **read-critical(key, vern):** read local copy if local version # ≥ vern

  useful for read-your-writes, can be fast.
- `read-latest(key)`: reads from master region
- `test-and-set-write(key, value, vern)`:
  writes only if latest version $\geq$ vern
  can be used to increment a counter.

Practice problem on next page.
1 PNUTS Question

(From 6.824 Spring 2011 Quiz II)

You’re running a PNUTS system. Records X and Y both start with value zero. Here are two functions that use the API described in Section 2.2 of the PNUTS paper:

```
fn1:
  x1 = read-any(X)
  x1 = x1 + 1
  write(X, x1) // X = x1
  write(Y, x1) // Y = x1

fn2:
  x1 = read-any(X)
  x2 = read-latest(X)
  y1 = read-any(Y)
  print x1, x2, y1
```

You execute two calls to fn1, at different sites, at the same time. After both calls to fn1 have returned, you execute fn2 at a third site. There is no activity in the system other than described here, and no crashes or network failures.

What is it possible to see from fn2, given the design of PNUTS and the above scenario?

(A) 2, 2, 1
(B) 1, 2, 2
(C) 1, 1, 2
(D) 2, 1, 1
(E) 0, 0, 0
2 Answer

(A) Yes: fn1s happen in serial order. x1 and x2 represent latest X, y1 is slightly stale Y

(B) Yes: fn1s happen in serial order. x1 is slightly stale X, x2 is latest X, y1 is latest Y

(C) No: Y must be ≤ X, therefore y1 must be ≤ x2

(D) No: If latest X = 1, then stale X (x1) must be ≤ 1

(E) No: If both fn1s complete, latest X must be ≥ 1