Unit 4 · Systems at Scale

ANSWER KEY · FOR TEACHERS

Full marks 32

Give the marks for the points in the marking scheme; an answer in other words that says the same is right.

1 Key words

1. stateless
2. load balancer
3. bottleneck
4. shard
5. transaction
6. network partition
7. hit
8. exponential backoff

Each carries 0.5 mark. **K · 4**

2 The parts of a big service

1. App servers
2. Load balancer
3. Cache in memory
4. Queue slow jobs

Each carries 0.5 mark. **U · 2**

3 Multiple choice

i. **b.** Every later request from that customer must reach the same server, and if it breaks or is removed, the cart is lost. **U · 1**

A server that keeps a user's data in its own memory isn't stateless: it can't be swapped for another. Keep the cart in a shared store every app server can reach, and any server can answer any request, so servers can be added, removed or lost freely.

ii. **b.** send each request to the server with the fewest requests in progress **A · 1**

Round robin and random choices give a server stuck on a long statement just as many new requests as an idle one, so queues build behind the slow ones. Counting the requests in progress sends new work to the servers that are actually free.

iii. **b.** The same seat can be sold on both sides, and the two sales only clash when the link is back. **U · 1**

Choosing availability means each side accepts changes without knowing what the other is doing. For seats or money, that allows two sales of one thing, so such systems choose consistency and refuse on the side without a majority.

iv. **d.** durability **K · 1**

Durability: a committed change survives crashes, because it was written to disk before the database said "done". Atomicity would be about the payment not being half done.

v. **b.** The refresh read from a follower copy that hadn't yet received your change. **U · 1**

That is replication lag: the change reached the leader at once and the followers a little later. Sending your own reads to the leader for a while after a change fixes it.

4 Think

ANSWER

No. When the network is split (a partition), the copy cut off can either refuse payments until it can reach the others, keeping every copy right, or accept them and tell the others later, staying available. If it accepts, two payments on the two sides can together spend money that isn't there (Rs. 700 twice from Rs. 1,000). This is the CAP theorem: during a

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

partition a system must give up consistency or availability. For balances, it should refuse on the side without a majority. HA · 2

MARKING SCHEME

- ✓ During a partition, a copy must refuse (stay consistent) or accept (stay available); it can't do both (CAP).
- ✓ If both sides accept payments, money can be spent twice, so a wallet must refuse on the side without a majority.

5 Waiting, and waiting for the slowest

ANSWER A · 4

i. 75% busy: 0.04 ÷ (1 − 0.75) = 0.04 ÷ 0.25 = **0.16 s**. 96% busy: 0.04 ÷ 0.04 = **1 s**, more than six times as long, for only about a third more work.

ii. All 20 are quick with a chance of $0.98^{20} \approx 0.67$, so at least one is slow on 1 − 0.67 ≈ **33%** of page loads, though each server is slow only 2% of the time.

MARKING SCHEME

- ✓ 0.16 s at 75% busy.
- ✓ 1 s at 96% busy.
- ✓ The chance all are quick: $0.98^{20} \approx 0.67$.
- ✓ About 33% of page loads are slow.

6 A lost update

MARKING SCHEME U · 3

- ✓ 800 left instead of 1,300.
- ✓ A lost update, breaking isolation.
- ✓ With a lock, the second program waits and reads the balance only after the first has committed.

7 A data center's electricity and water

ANSWER A · 3

A year is 365 × 24 = 8,760 hours. The computers use 8,000 kW × 8,760 h = 70,080,000 kWh.

i. The whole building: 1.4 × 70,080,000 ≈ 98,100,000 kWh ≈ **98 GWh** a year.

ii. Water = 2.0 L/kWh × 70,080,000 kWh (the computers' electricity) ≈ **140 million liters** a year.

iii. One person needs 100 × 365 = 36,500 L a year: 140,160,000 ÷ 36,500 ≈ **3,840 people.**

MARKING SCHEME

- ✓ About 98 GWh (PUE × the computers' 70,080,000 kWh).
- ✓ About 140 million liters (WUE × the computers' electricity, not the whole building's).
- ✓ About 3,840 people.

8 What does it print?

ANSWER

```
OUTPUT
1007 3 2
2026 2 1
3333 1 3
16
```

i. Changing mod 4 to mod 5 moves most accounts (16 of these 20, four in five) to a different shard, and moving that much data while it is in use takes days. With consistent hashing, a new shard takes over only the keys it should hold, about one in five, and every other key stays where it was.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

MARKING SCHEME A · 4

- ✓ `1007 3 2` and `2026 2 1`.
- ✓ `3333 1 3`.
- ✓ 16: only 1, 2, 3 and 20 stay in the same shard.
- ✓ Consistent hashing moves only the keys the new shard should hold, not most of them.

9 The four promises

K · 2

1. atomicity → **c.** a transaction happens completely or not at all
2. consistency → **a.** every rule of the database still holds after a transaction
3. isolation → **d.** transactions running at the same time don't see each other's half-done changes
4. durability → **b.** once committed, a change survives a crash or a power cut

Each carries 0.5 mark.

10 Find and fix the bug

ANSWER A · 3

i. The first UPDATE (line 13) took Rs. 500 from Sita; the second (line 14) was refused by the CHECK rule. The `except` prints "Refused" but leaves the transaction open with the half-done change still in it, so Sita shows Rs. 1,500 and the 500 is in nobody's wallet. A later COMMIT would keep it that way.

ii. In the `except` block, before line 18, add:

```
        db.execute("ROLLBACK")      # undo every change since BEGIN
```

Then it prints `Refused` and `[('Sita', 2000), ('Hari', 49800)]`.

MARKING SCHEME

- ✓ The first UPDATE happened and the second was refused, so the transfer is half done.
- ✓ Nothing undoes it: the transaction is left open with Sita's 500 taken.
- ✓ `db.execute("ROLLBACK")` in the `except` block.

11 Activity: Results day

(not marked)

WHAT STUDENTS SHOULD FIND

- Cache off: with 1 or 2 servers, more than a quarter of visitors give up; with 3 or more, 13% still give up however many are added, because the database is full: it is the bottleneck.
- Cache on: most requests never reach the database, so the web servers are the bottleneck again. 5 servers lose nobody, though some wait up to 10 seconds at 3:00; 6 or more answer everyone within about 2 seconds.

Marks by level	Knowledge	Understanding	Application	Higher ability	Total
Marks	7	8	15	2	**32**

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking