System Architect Interview Conversation

Google

Interviewer:
Give me a brief overview of your background and the kinds of systems you've architected.

Candidate:
I have **9 years of experience** building distributed backend systems. For the last **5 years**, I worked as a **Principal Systems Engineer** at a global SaaS company with **approximately 120 million monthly active users**. I led the architecture of a multi-region platform processing **2.8 million requests per second at peak**, deployed across **North America, Europe, and Asia-Pacific**, with a sustained **99.99% availability** target.

Interviewer:
Let's move into system design. Design a global URL shortening service at Google scale.

Candidate:
The core requirements are low-latency redirects, global availability, correctness of redirects, and operational simplicity. Quantitatively, I would assume **1 billion active short URLs**, **5 million new URLs per day**, and approximately **300 million redirects per day**, with a **p99 latency under 50 milliseconds**.

Interviewer:
What does the high-level architecture look like?

Candidate:
Traffic enters through Anycast DNS and global load balancers. Requests are routed to the nearest healthy region, primarily **us-central1**, **europe-west1**, and **asia-southeast1**. There are separate services for URL creation and redirects. Redirects are optimized for read-heavy traffic and sit behind an aggressive CDN layer. Writes go through a regional API service backed by a distributed ID generator and a globally replicated key-value store.

Interviewer:
How do you generate short URLs without collisions or coordination issues?

Candidate:
I use a **64-bit Snowflake-style ID**. The layout consists of 41 bits for timestamp, 10 bits for region ID, and 12 bits for a per-node sequence counter. This allows **4,096 IDs per millisecond per node** without coordination. The ID is Base62-encoded, producing URLs between **7 and 9 characters**.

Interviewer:
Why not random hashes?

Candidate:
Random hashes introduce collision handling, retries, and coordination overhead at scale. Sequential, time-ordered IDs are simpler to operate and easier to shard. Predictability improves operational

reliability.

Interviewer:
Talk about data storage.

Candidate:
The primary store is a **sharded, globally replicated key-value datastore**. Each short URL maps to the long URL, creation timestamp, optional expiration timestamp, and owner account ID. The replication factor is **three per region**, with quorum reads for correctness. Analytics and logging data are written asynchronously to append-only event streams and processed separately.

Interviewer:
How do you shard the data?

Candidate:
Data is sharded by hashing the short URL ID. This ensures uniform distribution and allows shard count changes without reshaping the entire dataset. The system starts with **2,048 shards globally** and scales horizontally as write throughput increases.

Interviewer:
Describe the redirect path in detail.

Candidate:
Most redirects are served directly from the CDN edge with a **24-hour TTL**. On a cache miss, the request is routed to the nearest regional redirect service, which performs a single key-value lookup and returns a 301 or 302 HTTP response. The redirect service is stateless and horizontally scalable.

Interviewer:
How high do you expect the cache hit ratio to be?

Candidate:
Above **95 percent**. Redirect traffic is highly skewed, with a small subset of links receiving the majority of traffic. The architecture is designed to exploit this access pattern.

Interviewer:
What about cache invalidation?

Candidate:
There is no synchronous cache invalidation. Administrative changes emit asynchronous invalidation events. Short-lived staleness is acceptable, but incorrect redirects are not. The write path enforces correctness before propagation.

Interviewer:
Let's talk about failures. A region goes completely offline.

Candidate:
DNS health checks remove the region within **30 to 60 seconds**. Reads continue from remaining

regions using replicated data. Writes are automatically routed to the nearest healthy region. The recovery point objective is **under one second**, and full traffic stabilization occurs in **under two minutes**.

Interviewer:
How do you ensure consistency for redirects?

Candidate:
Once a short URL is created, it must always resolve to the same long URL. Writes use quorum replication. Reads use quorum when accessing storage, though most reads are served from cache and never reach the datastore.

Interviewer:
What happens if two users try to create the same short URL?

Candidate:
They cannot. Short URLs are system-generated, not user-provided. This eliminates conflicts and simplifies abuse prevention.

Interviewer:
How do you monitor this system?

Candidate:
We track the golden signals: latency, traffic, error rate, and saturation. Key metrics include p50, p95, and p99 redirect latency, cache hit ratio, storage read latency, and write error rates. All requests carry trace IDs for distributed tracing. Alerts are driven by SLO violations rather than raw resource thresholds.

Interviewer:
What's the availability target?

Candidate:
99.99 percent monthly availability for redirects. The write path can tolerate brief degradation without immediate user impact.

Interviewer:
Traffic increases tenfold in a year. What breaks first?

Candidate:
The metadata write path. Mitigation involves increasing shard count, adding write batching, and extending cache TTLs for inactive URLs. The read path scales linearly due to CDN absorption.

Interviewer:
How do you handle abuse and malicious links?

Candidate:
URL creation events are asynchronously scanned using reputation services. Known malicious URLs

are disabled by updating metadata and pushing cache invalidations. The redirect service enforces these checks before responding.

Interviewer:
How do you roll out schema changes?

Candidate:
Only backward-compatible changes are allowed. New fields are added with default values. Readers are updated before writers. There are no in-place destructive migrations on hot paths.

Interviewer:
How do you document this architecture for large teams?

Candidate:
I use high-level diagrams for cross-team alignment, detailed design documents for implementers, and explicit ownership boundaries. Documentation is versioned and reviewed like code.

Interviewer:
What trade-offs did you intentionally make?

Candidate:
Eventual consistency for analytics, no synchronous cache invalidation, and system-generated IDs instead of user customization. These decisions reduce complexity and increase reliability.

Interviewer:
What makes a system well-architected?

Candidate:
A well-architected system scales without redesign, fails predictably, and can be operated by engineers who did not originally build it. If incidents can be diagnosed and mitigated quickly, the architecture is effective.

Interviewer:
That's all. Thank you.

Candidate:
Thank you.