Scaling APIs to Billions: Architectures and Algorithms for Extreme-Scale Reliability

Nikhil Kokal · The Walt Disney Company · Conf42 DevOps 2026

The Modern API Challenge

The Scale We're Facing

APIs now power the world's largest digital ecosystems: e-commerce platforms processing millions of transactions, social networks serving billions of users, IoT devices generating endless streams of data, and scientific computing infrastructure handling petabyte-scale workloads.

Modern applications demand APIs that can scale from millions to billions of daily requests while maintaining sub-100ms latency and five-nines reliability.

The Engineering Reality

Traditional scaling approaches fail at extreme scale. Simple vertical scaling hits physical limits. Naive horizontal scaling introduces coordination overhead that destroys performance.

Today's engineers need a systematic framework combining architectural patterns, operational strategies, and algorithmic approaches to build APIs that remain reliable, fast, and cost-efficient at unprecedented scale.

Proven Success Stories

Netflix

Serves 250M+ subscribers globally with edge caching and adaptive streaming algorithms that dynamically adjust to network conditions and regional demand patterns.

Uber

Processes billions of location updates daily using geo-sharded architectures and distributed throttling to handle massive surge events during peak hours.

RevenueCat

Built subscription infrastructure handling millions of purchases with sophisticated request coalescing and cache key design patterns that reduce backend load by 90%.

Four-Layer Scaling Framework

Edge Layer

CDN caching, geographic routing, and TLS termination reduce origin load while improving global latency through strategic PoP placement.

Gateway Layer

Rate limiting, authentication, request routing, and protocol translation provide unified control plane for all API traffic.

Microservice Layer

Service mesh orchestration, circuit breaking, and adaptive load balancing enable resilient communication between hundreds of services.

Data Layer

Sharding strategies, read replicas, and distributed caching patterns ensure databases scale horizontally without sacrificing consistency.

Edge Caching Architecture

Cache Key Design

Strategic cache key composition determines hit rates and backend offload effectiveness.

- User segment-based keys
- Device type awareness
- Geographic bucketing
- Version-aware invalidation

Adaptive TTL Strategies

Dynamic time-to-live adjustments based on content freshness requirements, traffic patterns, and backend health signals dramatically improve cache efficiency.

Popular content gets extended TTLs during high traffic, while stale-while-revalidate patterns ensure zero-downtime cache refreshes. This approach reduced origin requests by 85% at scale.

Request Coalescing Patterns

1

Request Deduplication

Identical concurrent requests are collapsed into a single backend call, with results broadcast to all waiting clients.

2

Batch Aggregation

Multiple similar requests are intelligently grouped and processed together, reducing network overhead and database query load.

3

Result Sharing

Fresh responses are temporarily cached in-memory and served to subsequent identical requests within microseconds.

Real-World Traffic Patterns

Diurnal Surges

Predictable daily peaks require proactive autoscaling with 15-minute lead time. Traffic can spike 10x between 8 AM and noon in key regions.

Burst Traffic

Unpredictable events (viral content, breaking news, product launches) create 50-100x load spikes within seconds. Systems must absorb without cascading failures.

Adversarial Loads

Bot traffic, DDoS attacks, and scraping operations create abnormal patterns. Distributed rate limiting and anomaly detection provide defense layers.

Distributed Rate Limiting

Algorithm Selection

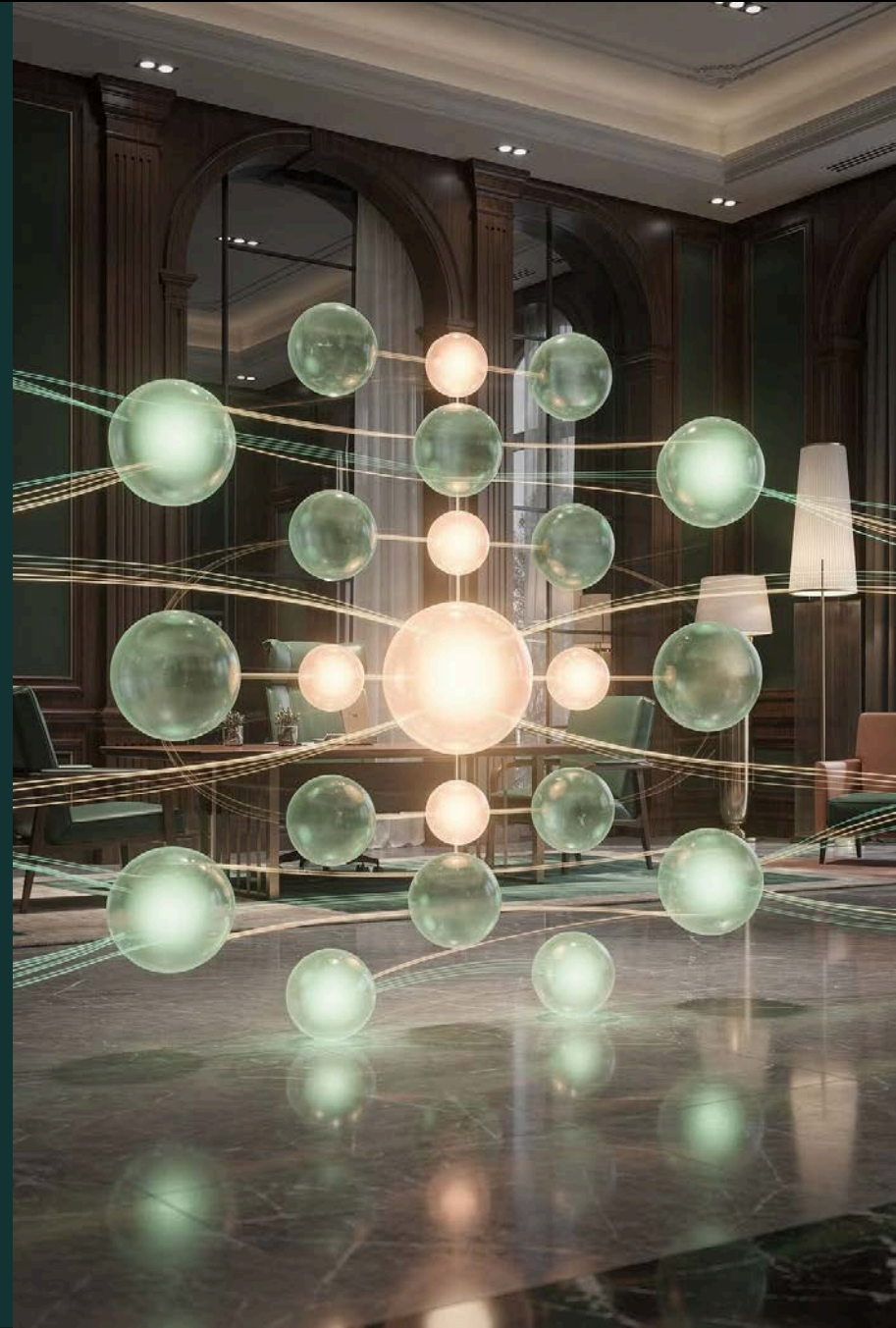
Token Bucket: Allows burst capacity while enforcing average rate limits. Ideal for API endpoints serving bursty client behavior.

Leaky Bucket: Smooths traffic into consistent rates. Perfect for protecting downstream services with strict capacity limits.

Cross-Region Coordination

Global rate limits require distributed coordination without introducing unacceptable latency. Hybrid approaches combine local counters with eventual consistency synchronization.

Regional quotas with periodic reconciliation balance accuracy with performance, allowing 99.9% enforcement with sub-millisecond overhead.



Multi-Region Active-Active Architecture

1. **Request Routing**

   GeoDNS routes users to nearest region, with automatic failover to healthy regions during outages.

2. **Data Replication**

   Bidirectional replication with conflict resolution ensures eventual consistency across regions.

3. **Cross-Region Writes**

   Distributed transactions with two-phase commit handle writes affecting multiple regions.

4. **Regional Autonomy**

   Each region operates independently, surviving complete loss of inter-region connectivity.

Sharding Strategies at Scale

Consistent Hashing

Distributes data across nodes with minimal reshuffling during cluster changes. Virtual nodes ensure even distribution and smooth rebalancing.

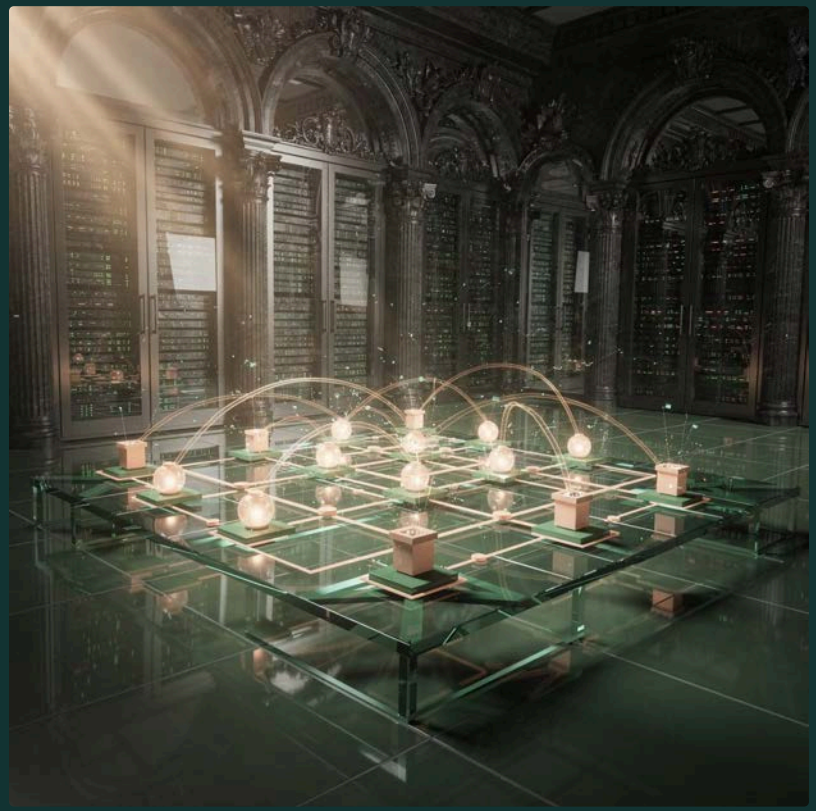
Geographic Sharding

Partitions data by region to optimize latency and comply with data residency regulations. Critical for global applications.

Hybrid Approaches

Combines multiple sharding dimensions (user ID, geography, tenant) for optimal data distribution and query performance.

Service Mesh Orchestration



Traffic Management

- Intelligent load balancing with health-aware routing
- Circuit breakers preventing cascading failures
- Automatic retry with exponential backoff
- Traffic splitting for canary deployments

Observability Integration

Service mesh provides unified telemetry collection point, capturing distributed traces, metrics, and logs without application instrumentation. This visibility is essential for debugging issues across hundreds of microservices.

Observability at Billion-Request Scale

01

High-Cardinality Metrics

Track latency, error rates, and throughput across millions of dimensions without overwhelming time-series databases.

02

Tail-Based Tracing

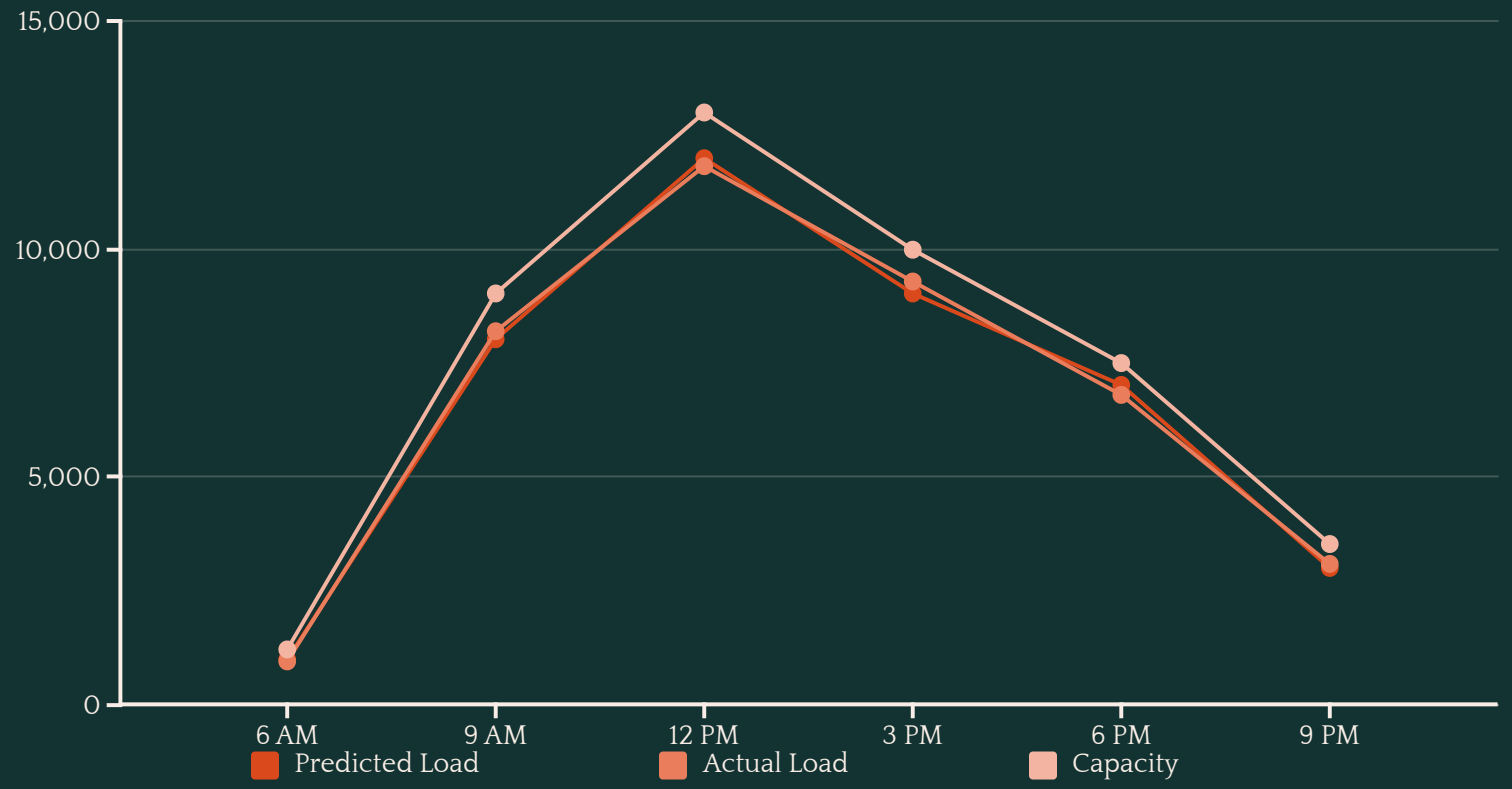
Intelligently sample slow requests and errors rather than random sampling, capturing anomalies while controlling costs.

03

Cost-Aware Log Sampling

Adaptive sampling rates based on error conditions and cost budgets prevent observability from consuming 20% of infrastructure spend.

Predictive Autoscaling



Machine learning models predict traffic patterns based on historical data, seasonality, and external signals, scaling infrastructure proactively. This approach reduces costs by 40% compared to reactive autoscaling while improving reliability during traffic spikes.

The Next Frontier

Cross-Region Rate Coordination

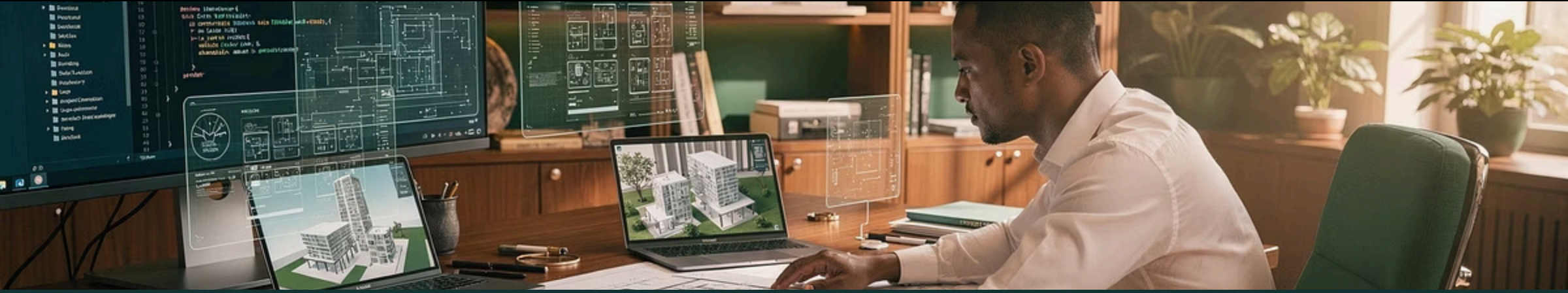
As applications become truly global, coordinating rate limits across regions without sacrificing latency remains unsolved. New protocols and eventual consistency models are emerging.

Extreme Tail Latency Diagnosis

Understanding and eliminating P99.9 latency spikes requires new profiling tools that can capture microsecond-level behavior across distributed systems without performance impact.

Energy-Aware Elasticity

The next generation of autoscaling will optimize for carbon footprint and energy costs, not just request latency—shifting workloads to regions with renewable energy availability.



Your Billion-Request Blueprint

Key Takeaways

- Apply the four-layer framework systematically
- Leverage edge caching with smart key design
- Implement distributed rate limiting early
- Build observability into architecture from day one
- Use predictive autoscaling to balance cost and performance

Start Your Journey

Scaling to billions requires disciplined application of proven patterns combined with continuous measurement and optimization. Start with your highest-traffic endpoints, instrument thoroughly, and evolve incrementally.

Build APIs that remain reliable, fast, and cost-efficient at any scale.

Thank You!
Questions?

Nikhil Kokal
The Walt Disney Company
Conf42 DevOps 2026.