System Design Calculations Cheat Sheet

Formulas, typical values, and worked examples for fast interview math

Five Essential Calculations

1) REQUESTS PER SECOND (RPS)

Formula: `(DAU × actions_per_user × peak_ratio) ÷ 86,400`

- Typical **peak_ratio**: 2-3x average (can be 5-10x for events).
- Always sanity-check against product behavior (bursty vs steady).

Example: 100M DAU, 10 actions/day, 3x peak → (100,000,000 × 10 × 3) ÷ 86,400 ≈ 34,722 RPS (~35K).

2) STORAGE REQUIREMENTS

Formula: `entities × size_per_entity × growth_years × replication_factor`

- Typical **replication_factor**: 2-3x (storage systems often keep multiple copies).
- Separate **hot** vs **cold** data if possible (cost and access patterns differ).

Example: 1B photos, 2MB each, 3 years, 3x replication → 1,000,000,000 × 2MB × 3 × 3 ≈ 18,000,000,000 MB ≈ 18 PB.

3) BANDWIDTH

Formula: `RPS × avg_request_size (compute read and write separately)`

- Typical request size: 1KB-1MB depending on content type.
- Remember: 1 byte = 8 bits. Convert to Gbps when talking networks.

Example: 10K RPS × 100KB = 1,000,000 KB/s ≈ 1 GB/s ≈ 8 Gbps.

4) CACHE SIZING

Formula: `total_data × 0.20 (80/20 hot set) × 1.5 (overhead)`

- Typical cache hit rate target: 70-90% for well-designed systems.
- Overhead accounts for metadata, eviction policies, and fragmentation.

Example: 100GB total data → 100 × 0.20 × 1.5 = 30GB cache needed.

5) DATABASE CAPACITY

Rule of thumb: If calculated RPS exceeds a single instance capacity, plan for replication, caching, and eventually sharding.

Store Type	Typical Single Instance Throughput (very workload-dependent)
SQL (e.g., MySQL/PostgreSQL)	1K-10K QPS
NoSQL (e.g., Cassandra/Dynamo-style)	10K-100K+ QPS

Reality Check Benchmarks

Component	Benchmark (rough)	Why it matters
Single app server	1,000-5,000 RPS (simple ops)	If you need 100K RPS, you will need many app servers + load balar
Redis	100,000+ ops/sec	Great for hot data and rate-limiting; reduces DB load.
PostgreSQL	5,000-15,000 QPS (indexed)	Indexes and query shape dominate; treat as workload-dependent.
CDN	70-90% origin load reduction	For static/media-heavy apps, CDN can cut bandwidth and latency m
Most mobile apps	< 100 RPS	Many interview designs are over-engineered; scale should drive cor

Worked Example: Design Twitter Feed

Requirements: 300M DAU, each user loads feed 10 times/day, assume 3x peak traffic.

Step	Calculation	Result (approx)
1) RPS	300M × 10 × 3 ÷ 86,400	≈ 104,167 RPS (~104K)
2) Storage	300M users × 100 tweets avg × 280 chars × 3 years	≈ 25 TB (order of magnitude)
3) Bandwidth	104K RPS × 10KB avg response	≈ 1.0 GB/s (~8 Gbps)
4) Cache	25TB × 0.20 × 1.5	≈ 7.5 TB cache
5) DB capacity	104K RPS vs single-instance capacity	Exceeds → sharding + replication + caching

Interpretation: ~104K RPS is beyond a single database instance. You need to reduce read load (cache + precompute), replicate for reads, and shard when a single primary cannot keep up. Use calculations to justify architecture decisions.