Back-of-the-Envelope Calculations Guide for System Design Interviews

Table of Contents

Introduction

> **"In system design interviews, being approximately right is better than being precisely wrong."**

Back-of-the-envelope calculations are rough estimates used to evaluate the feasibility of a system design. Interviewers want to see:

- Can you estimate scale and resource requirements?
- Do you understand trade-offs between different approaches?
- Can you identify potential bottlenecks?

You don't need a calculator! Simple multiplication and division with rough numbers is enough.

Core Principles

1. Round Numbers Liberally

Bad: "We have 387,654,321 daily active users..."
Good: "We have ~400 million daily active users..."

Why: Easier to calculate, equally accurate for capacity planning

2. Use Powers of 10/2

Instead of: 1,234,567 bytes
Use: ~1 MB (1 million bytes)

Instead of: 987 requests
Use: ~1K requests (1,000)

3. State Your Assumptions Clearly

Always say:

- "Assuming 100 million daily active users..."
- "Let's assume average photo size is 200 KB..."
- "I'll estimate read-to-write ratio as 100:1..."

4. Work in Standard Units

Storage: Bytes → KB → MB → GB → TB → PB
Time: Seconds → Minutes → Hours → Days → Years
Traffic: QPS (Queries Per Second)

5. Show Your Work

Don't say: "We need 10 TB of storage"
Do say: "100M photos × 200 KB = 20 TB, minus 50% compression = 10 TB"

Essential Numbers to Memorize

Powers of Two (Storage)

```
2^10 = 1,024      ≈ 1 Thousand  = 1 KB
2^20 = 1,048,576  ≈ 1 Million   = 1 MB
2^30 = 1 Billion  ≈ 1 GB
2^40 = 1 Trillion ≈ 1 TB
2^50 = 1 Quadrillion ≈ 1 PB

Pro Tip: Just remember "1024" and multiply!
- 1 KB = ~1,000 bytes
- 1 MB = ~1,000 KB = 1 million bytes
- 1 GB = ~1,000 MB = 1 billion bytes
- 1 TB = ~1,000 GB = 1 trillion bytes
```

Time Conversions

```
1 minute = 60 seconds
1 hour = 3,600 seconds (60 × 60)
1 day = 86,400 seconds (24 × 3,600)
1 month = ~2.5 million seconds (30 × 86,400)
1 year = ~31.5 million seconds (365 × 86,400)

Pro Tip: Remember "86,400 seconds per day"
```

```
- Daily to QPS: Divide by 86,400 (≈ 100K)
- Monthly to QPS: Divide by 2.5M
```

Latency Numbers (Google's Famous List)

```
L1 cache reference:              0.5 ns
Branch mispredict:               5 ns
L2 cache reference:              7 ns
Main memory reference:           100 ns
Compress 1KB with Snappy:        10 µs
Send 1 KB over 1 Gbps network:   10 µs
Read 1 MB from memory:           250 µs
Round trip within datacenter:    500 µs
Read 1 MB from SSD:              1 ms
Disk seek:                       10 ms
Read 1 MB from network:          10 ms
Read 1 MB from disk:             30 ms
Send packet CA→Netherlands→CA:   150 ms

Takeaways:
- Memory is ~100x faster than SSD
- SSD is ~10x faster than HDD
- Network within datacenter is fast (0.5ms)
- Cross-continent is slow (150ms)
```

Common Data Sizes

```
1 character = 1 byte (ASCII)
1 character = 2-4 bytes (Unicode/UTF-8)

User ID (integer): 8 bytes
Timestamp: 8 bytes
IP address (IPv4): 4 bytes
IP address (IPv6): 16 bytes

Tweet text (280 chars): ~280 bytes
Small image (thumbnail): ~50 KB
Regular image (compressed): ~200 KB
High-res image: ~2 MB
Short video (1 min): ~10 MB
HD video (1 min): ~50 MB

User profile: ~1-2 KB
Email: ~75 KB (average)
Web page: ~2 MB (with assets)
```

Step-by-Step Framework

Step 1: Clarify Scale Requirements

Questions to Ask:

```
Users:
- How many daily active users (DAU)?
- How many monthly active users (MAU)?
- User growth rate?

Usage:
- How many posts/uploads per user per day?
- How many reads per user per day?
- Read-to-write ratio?

Data:
- Average size of user data?
- Average size of content (posts, photos, videos)?
- Data retention period?
```

Step 2: Calculate Traffic

Formula:

```
QPS = Total Operations per Day / 86,400 seconds

Example:
100M tweets per day
= 100,000,000 / 86,400
= 1,157 QPS (round to ~1.2K QPS)

Peak QPS = Average QPS × Peak Factor
Peak Factor typically: 2-5x
= 1.2K × 3 = 3.6K QPS (round to ~4K QPS)
```

Step 3: Calculate Storage

Formula:

```
Total Storage = Number of Items × Size per Item × Time Period

Example:
100M photos per month × 200 KB × 12 months × 10 years
= 100M × 200 KB × 120
= 2.4 PB
```

Step 4: Calculate Bandwidth

Formula:

```
Bandwidth = QPS × Average Response Size

Example:
Read QPS: 10K
Average response: 100 KB
Bandwidth = 10,000 × 100 KB = 1 GB/s
```

Step 5: Calculate Memory (Cache)

Formula:

```
Cache Size = % of Data to Cache × Data Size

Example (80-20 rule):
Total data: 10 TB
Cache 20% hot data: 10 TB × 0.2 = 2 TB
Distributed across 20 servers: 2 TB / 20 = 100 GB per server
```

Traffic Estimation

Calculation Method

Given: Daily Active Users (DAU) and actions per user

Formula:

```
Total Daily Operations = DAU × Operations per User
QPS = Total Daily Operations / 86,400
Peak QPS = QPS × Peak Factor (2-5x)
```

Example 1: Twitter

Given:

- 400M DAU
- Each user tweets 0.5 times per day (average)
- Each user reads 100 tweets per day (scrolling feed)

Calculate:

```
Write Traffic (Tweets):
- Tweets per day: 400M × 0.5 = 200M
- Write QPS: 200M / 86,400 ≈ 2,315 QPS
- Peak write QPS: 2,315 × 3 ≈ 7K QPS

Read Traffic (Timeline):
- Reads per day: 400M × 100 = 40 billion
- Read QPS: 40B / 86,400 ≈ 462K QPS
- Peak read QPS: 462K × 3 ≈ 1.4M QPS

Read-to-Write Ratio: 40B / 200M = 200:1
```

Interview Insight: "This is read-heavy, so we need caching and read replicas"

Example 2: Instagram

Given:

- 500M DAU
- Each user uploads 0.1 photos per day (1 photo per 10 days)
- Each user views 50 photos per day

Calculate:

```
Write Traffic (Photos):
- Photos per day: 500M × 0.1 = 50M
- Write QPS: 50M / 86,400 ≈ 578 QPS
- Peak: ~1.7K QPS

Read Traffic (Feed):
- Views per day: 500M × 50 = 25 billion
- Read QPS: 25B / 86,400 ≈ 289K QPS
- Peak: ~867K QPS

Read-to-Write Ratio: 25B / 50M = 500:1
```

Interview Insight: "Extremely read-heavy, CDN caching is critical"

Example 3: WhatsApp

Given:

- 2 billion MAU
- 50% daily active = 1B DAU
- Each user sends 40 messages per day
- Each user receives 40 messages per day

Calculate:

```
Messages per day: 1B × 40 = 40 billion

Message writes: 40B / 86,400 ≈ 463K QPS
Peak: ~1.4M QPS

With reads (checking for new messages):
- Check every 10 seconds: 1B users × 8,640 checks = 8.64 trillion checks/day
- Check QPS: 8.64T / 86,400 = 100M QPS (too high!)
- Solution: WebSocket connections, push notifications
```

Interview Insight: "Need WebSocket for efficiency, not polling"

Storage Estimation

Calculation Method

Formula:

```
Storage = Number of Objects × Size per Object × Time Period
```

Example 1: YouTube

Given:

- 500 hours of video uploaded per minute
- Average video size: 50 MB per minute
- Store for 10 years

Calculate:

```
Daily uploads:
- 500 hours/min × 60 min/hour × 24 hours = 720,000 hours per day
- 720K hours × 60 min × 50 MB = 2,160,000,000 MB per day
- = 2,160 TB per day = ~2 PB per day

Yearly: 2 PB × 365 = 730 PB per year
10 years: 730 PB × 10 = 7.3 exabytes (7,300 PB)

With multiple resolutions (360p, 720p, 1080p, 4K):
- 4 versions per video
- Total: 7,300 PB × 4 = 29 exabytes

With compression and deduplication:
- Assume 30% savings
- Effective: 29 EB × 0.7 = ~20 exabytes
```

```
```

Interview Insight: "This is why YouTube needs Google's infrastructure!"

Example 2: Instagram

Given:

- 500M DAU
- 50M photos uploaded per day
- Average photo: 200 KB
- Store for 10 years

Calculate:

```
Daily storage:
- 50M photos × 200 KB = 10,000,000,000 KB
- = 10 TB per day

Yearly: 10 TB × 365 = 3.65 PB
10 years: 3.65 PB × 10 = 36.5 PB

With thumbnails (3 sizes per photo):
- Original: 200 KB
- Large: 100 KB
- Thumbnail: 20 KB
- Total per photo: 320 KB

Adjusted: 50M × 320 KB = 16 TB per day
10 years: 16 TB × 365 × 10 = 58 PB
```

Interview Insight: "Need S3 or similar object storage, with lifecycle policies"

Example 3: Twitter

Given:

- 400M DAU
- 200M tweets per day
- Average tweet: 280 characters = ~280 bytes
- Store for 5 years
- 20% of tweets have images (~500 KB each)

Calculate:

```
Text storage:
- 200M tweets × 280 bytes = 56 GB per day

Image storage:
```

```
- 200M × 0.2 = 40M images per day
- 40M × 500 KB = 20 TB per day

Total per day: 56 GB + 20 TB ≈ 20 TB
Yearly: 20 TB × 365 = 7.3 PB
5 years: 7.3 PB × 5 = 36.5 PB

With metadata (user_id, timestamp, likes, etc.):
- Additional 100 bytes per tweet
- 200M × 100 bytes = 20 GB per day
- Negligible compared to images

Total: ~37 PB for 5 years
```

Interview Insight: "Text is cheap, media is expensive"

Bandwidth Estimation

Calculation Method

Formula:

```
Bandwidth = QPS × Average Object Size
```

Example 1: Instagram Feed

Given:

- 289K read QPS (from earlier calculation)
- Average feed response: 50 photos × 200 KB thumbnails = 10 MB

Calculate:

```
Incoming bandwidth (uploads):
- 578 QPS (writes) × 200 KB = 115 MB/s

Outgoing bandwidth (feed loads):
- 289K QPS × 10 MB = 2,890 GB/s = 2.8 TB/s

Wait, that's too high! Let's optimize:

With CDN caching (95% cache hit ratio):
- Only 5% hits origin: 2.8 TB/s × 0.05 = 140 GB/s
- CDN bandwidth: 2.8 TB/s (distributed globally)

Actual Instagram approach:
- Serve thumbnails first (150 KB instead of 200 KB)
```

```
- Lazy load full images
- Reduced to ~1 TB/s from CDN
```

Interview Insight: "CDN is essential, not optional"

Example 2: YouTube

Given:

- 1 billion video views per day
- Average video: 10 minutes at 720p = 100 MB

Calculate:

```
Views per second: 1B / 86,400 ≈ 11,574 QPS

Bandwidth: 11,574 × 100 MB = 1,157 GB/s = 1.1 TB/s

Peak (3x): 3.3 TB/s

With CDN (80% cache hit):
- Origin: 3.3 TB/s × 0.2 = 660 GB/s
- CDN: 3.3 TB/s × 0.8 = 2.6 TB/s

Optimization with adaptive bitrate:
- Serve lower quality for slow connections
- Average drops to 50 MB per view
- New bandwidth: 578 GB/s origin
```

Memory/Cache Estimation

The 80-20 Rule (Pareto Principle)

Key Insight: 20% of data generates 80% of traffic

Cache Strategy: Cache the hot 20%, serve 80% of requests

Example 1: Twitter Timeline Cache

Given:

- 400M DAU
- Each timeline: 50 tweets × 500 bytes = 25 KB
- Cache 20% of active users

Calculate:

```
Users to cache: 400M × 0.2 = 80M users
Cache size: 80M × 25 KB = 2,000 GB = 2 TB

Distributed across servers:
- 20 cache servers × 100 GB RAM each = 2 TB total
- Cost: ~$2,000/month (AWS ElastiCache)

Cache hit ratio achieved: ~80%
Database queries saved: 80% of 462K QPS = 370K QPS
```

Interview Insight: "Caching 20% of users saves 80% of database queries"

Example 2: Netflix Thumbnails

Given:

- 10,000 videos in catalog
- Each thumbnail: 50 KB
- Cache most popular 1,000 videos (10%)

Calculate:

```
Cache size: 1,000 videos × 50 KB = 50 MB

With 50% cache hit ratio:
- 11,574 QPS × 0.5 = 5,787 QPS to origin
- Saves significant bandwidth and latency

Even full catalog cache:
- 10,000 × 50 KB = 500 MB
- Tiny! Cache everything!
```

Interview Insight: "Sometimes you can cache the entire dataset"

Real-World Examples

Example 1: Design Twitter

Step 1: Clarify Requirements

```
Assumptions:
- 400M DAU
- 200M tweets per day (0.5 tweets per user)
- Each user views 100 tweets per day
- Average tweet: 280 characters = 280 bytes
```

```
- 20% tweets have images (500 KB)
- Store for 5 years
```

Step 2: Traffic Estimation

```
Writes (Tweets):
200M tweets/day ÷ 86,400 = 2,315 QPS
Peak: 2,315 × 3 = ~7K QPS

Reads (Timeline):
400M users × 100 tweets = 40B reads/day
40B ÷ 86,400 = 462,963 QPS ≈ 463K QPS
Peak: 463K × 3 = ~1.4M QPS

Read-to-Write Ratio: 463K / 2.3K = 200:1
```

Step 3: Storage Estimation

```
Daily storage:
- Text: 200M × 280 bytes = 56 GB
- Images: 200M × 0.2 × 500 KB = 20 TB
- Total: ~20 TB/day

5 years: 20 TB × 365 × 5 = 36.5 PB

With metadata and indexes (+20%):
Total: 36.5 PB × 1.2 = ~44 PB
```

Step 4: Bandwidth Estimation

```
Incoming:
- 2.3K QPS × 280 bytes = 644 KB/s (text)
- Images: 40M/day ÷ 86,400 = 463 QPS
- 463 QPS × 500 KB = 231 MB/s
- Total incoming: ~232 MB/s

Outgoing:
- 463K QPS × 280 bytes = 130 MB/s (text)
- Images: 463K × 0.2 × 200 KB (thumbnails) = 18.5 GB/s
- Total: ~19 GB/s

With CDN (95% cache hit):
- Origin: 19 GB/s × 0.05 = 950 MB/s
- CDN: 19 GB/s × 0.95 = 18 GB/s
```

Step 5: Memory Estimation

```
Cache 20% of active user timelines:
- 400M × 0.2 = 80M users
- Each timeline: 50 tweets × 280 bytes = 14 KB
- Total: 80M × 14 KB = 1.1 TB

Cache servers needed:
- 1.1 TB ÷ 100 GB per server = 11 servers
- Round up to 20 servers for redundancy
```

Summary for Interview:

```
Traffic: ~7K write, ~1.4M read (peak)
Storage: ~44 PB for 5 years
Bandwidth: ~19 GB/s (950 MB/s with CDN)
Cache: ~1.1 TB across 20 servers
Database: Cassandra (write-heavy), PostgreSQL (users), Redis (cache)
```

Example 2: Design Instagram

Step 1: Requirements

```
- 500M DAU
- 50M photos uploaded per day (0.1 per user)
- Each user views 50 photos per day
- Photo: 2 MB original, 200 KB feed, 20 KB thumbnail
- Store for 10 years
```

Step 2: Traffic

```
Uploads: 50M ÷ 86,400 = 578 QPS (peak: 1.7K QPS)
Views: 500M × 50 = 25B/day
Views QPS: 25B ÷ 86,400 = 289K QPS (peak: 867K QPS)
Ratio: 500:1 (extremely read-heavy)
```

Step 3: Storage

```
Per photo:
- Original: 2 MB
- Feed size: 200 KB
- Thumbnail: 20 KB
```

```
- Total: 2.22 MB per photo

Daily: 50M × 2.22 MB = 111 TB
Yearly: 111 TB × 365 = 40.5 PB
10 years: 405 PB

With compression (save 30%):
Actual: 405 PB × 0.7 = ~283 PB
```

Step 4: Bandwidth

```
Upload: 578 QPS × 2 MB = 1.2 GB/s
Download: 289K QPS × 200 KB = 57.8 GB/s

With CDN (90% cache hit):
- Origin: 57.8 GB/s × 0.1 = 5.8 GB/s
- CDN: 57.8 GB/s × 0.9 = 52 GB/s
```

Step 5: Cache

```
Cache hot photos (20% of last 30 days):
- 50M/day × 30 days = 1.5B photos
- 1.5B × 0.2 = 300M hot photos
- 300M × 200 KB = 60 TB

Distributed: 60 TB ÷ 100 servers = 600 GB each
```

Example 3: Design WhatsApp

Step 1: Requirements

```
- 2B MAU, 1B DAU (50% active daily)
- Each user sends 40 messages/day
- Each user receives 40 messages/day
- Average message: 100 bytes
- Store for 1 year
```

Step 2: Traffic

```
Messages per day: 1B × 40 = 40B
Message QPS: 40B ÷ 86,400 = 462K QPS
Peak: 462K × 3 = 1.4M QPS
```

```
This is WRITE-heavy (send/receive are both writes)
```

Step 3: Storage

```
Daily: 40B × 100 bytes = 4 TB
Yearly: 4 TB × 365 = 1.46 PB

With metadata (timestamp, read status, etc.):
- +50 bytes per message
- 40B × 150 bytes = 6 TB/day
- Yearly: 2.19 PB

With images/videos (10% of messages):
- 40B × 0.1 = 4B media messages
- Average: 1 MB per media
- 4B × 1 MB = 4 PB/day
- Yearly: 4 PB × 365 = 1,460 PB = 1.46 exabytes
```

Step 4: Bandwidth

```
Text: 462K QPS × 100 bytes = 46 MB/s
Media: 462K × 0.1 × 1 MB = 46 GB/s
Total: ~46 GB/s

Note: This is why WhatsApp compresses images aggressively!
With compression (80% reduction):
Actual: 46 GB/s × 0.2 = ~9 GB/s
```

Example 4: Design Uber

Step 1: Requirements

```
- 50M daily rides
- Each ride: 1 driver, 1 rider
- Location updates every 3 seconds during ride
- Average ride: 15 minutes
- Store location history for 1 year
```

Step 2: Traffic

```
Location Updates:
- Updates per ride: (15 min × 60 sec) / 3 sec = 300 updates
```

```
- Updates per day: 50M rides × 300 × 2 (driver + rider) = 30B updates
- QPS: 30B ÷ 86,400 = 347K QPS
- Peak (during rush hour, 3x): ~1M QPS

This is WRITE-HEAVY (real-time location tracking)
```

Step 3: Storage

```
Per location update:
- User ID: 8 bytes
- Latitude: 8 bytes
- Longitude: 8 bytes
- Timestamp: 8 bytes
- Total: 32 bytes

Daily: 30B × 32 bytes = 960 GB per day
Yearly: 960 GB × 365 = 350 TB

With trip metadata (pickup, dropoff, fare, etc.):
- Additional 1 KB per trip
- 50M × 1 KB = 50 GB per day
- Yearly: 18 TB

Total: ~368 TB per year
```

Step 4: Memory (Real-time Matching)

```
Active drivers at any time:
- Assume 5M drivers online (10% of daily drivers)
- Each driver location: 32 bytes
- Total: 5M × 32 bytes = 160 MB

Cache all active drivers in memory:
- Easily fits in single Redis instance!
- Use Redis Geo data structure (GEOADD, GEORADIUS)

Active riders:
- 5M riders waiting (matching period)
- Total: 160 MB

Combined: 320 MB (tiny!)
```

Quick Calculation Shortcuts

Shortcut 1: Daily to QPS

Formula: `Daily Requests ÷ 100K ≈ QPS`

Why: 86,400 ≈ 100,000 (close enough for estimates)

Examples:

```
10M requests/day ÷ 100K = 100 QPS
1B requests/day ÷ 100K = 10K QPS
100B requests/day ÷ 100K = 1M QPS
```

Shortcut 2: Monthly to Daily

Formula: `Monthly ÷ 30 = Daily`

Example:

```
3B requests/month ÷ 30 = 100M requests/day
100M ÷ 100K = 1K QPS
```

Shortcut 3: Storage Growth

Formula: `Daily × 365 × Years = Total`

Example:

```
10 TB/day × 365 × 5 = 18,250 TB = ~18 PB
Round to 20 PB for safety margin
```

Shortcut 4: Bandwidth from QPS

Formula: `QPS × Object Size`

Example:

```
10K QPS × 100 KB = 1,000,000 KB/s = 1 GB/s
Round to 1 GB/s for simplicity
```

Shortcut 5: 80-20 Rule for Caching

Formula: `Total Data × 0.2 = Cache Size`

Example:

```
100 TB total data
Cache: 100 TB × 0.2 = 20 TB
Achieves ~80% cache hit ratio
```

Common Estimation Patterns

Pattern 1: Social Media (Twitter, Instagram)

Characteristics:

- Read-heavy (100:1 to 1000:1 ratio)
- Small writes (text, metadata)
- Large reads (media files)

Formula:

```
Write QPS = DAU × Posts per User / 86,400
Read QPS = DAU × Feed Loads × Items per Load / 86,400
Storage = Posts per Day × Size × Retention Period
```

Pattern 2: Messaging (WhatsApp, Telegram)

Characteristics:

- Balanced read/write (1:1 ratio)
- Small messages (bytes to KB)
- High frequency (billions per day)

Formula:

```
Message QPS = DAU × Messages per User / 86,400
Storage = Messages per Day × Size × Retention
Real-time Connections = DAU (for WebSocket)
```

Pattern 3: Video Streaming (YouTube, Netflix)

Characteristics:

- Extremely read-heavy (1000:1)
- Large files (GB per video)
- CDN is mandatory

Formula:

```
Upload QPS = Uploads per Day / 86,400
View QPS = Views per Day / 86,400
Storage = Videos per Day × Size × Retention
Bandwidth = View QPS × Bitrate / 8 (bits to bytes)
```

Pattern 4: E-Commerce (Amazon, eBay)

Characteristics:

- Read-heavy (browse > buy)
- Need ACID (transactions)
- Spike during sales events

Formula:

```
Browse QPS = DAU × Page Views / 86,400
Purchase QPS = Orders per Day / 86,400
Storage = Products × Size + Orders × Size
Peak Factor = 10x (during Black Friday)
```

Interview Example Walkthroughs

Walkthrough 1: "Design a News Feed"

Interviewer: "Design a news feed like Facebook. 1 billion users."

You:

```
"Let me estimate the scale:

Assumptions:
- 1B total users, 500M DAU (50% active)
- Each user posts 0.5 times per day
- Each user refreshes feed 10 times per day
- Each feed load shows 20 posts
- Average post: 1 KB (text + metadata)
- Store for 2 years

Traffic:
Posts: 500M × 0.5 = 250M/day ÷ 100K = 2.5K write QPS
Reads: 500M × 10 × 20 = 100B reads/day ÷ 100K = 1M read QPS
Ratio: 400:1 (very read-heavy)

Storage:
Posts: 250M/day × 1 KB = 250 GB/day
2 years: 250 GB × 365 × 2 = 182 TB
```

```
Round to 200 TB with indexes

Bandwidth:
Outgoing: 1M QPS × 1 KB = 1 GB/s

Cache:
Cache 20% of 500M users' feeds: 100M users
Each feed: 20 posts × 1 KB = 20 KB
Cache size: 100M × 20 KB = 2 TB across 20 servers

Database:
- PostgreSQL/MySQL for users (ACID)
- Cassandra for posts (write-heavy, 2.5K QPS)
- Redis for feed cache (1M read QPS × 80% = 800K QPS served from cache)
"
```

Walkthrough 2: "Design a URL Shortener"

Interviewer: "Design bit.ly. 100M URLs created per month."

You:

```
"Let me calculate the scale:

Assumptions:
- 100M new URLs per month
- 10B redirects per month (100:1 read-to-write)
- Average URL: 200 bytes
- Store for 10 years

Traffic:
Writes: 100M/month ÷ 30 ÷ 100K = 33 QPS (round to 40 QPS)
Reads: 10B/month ÷ 30 ÷ 100K = 3,333 QPS (round to 4K QPS)
Peak: 40 write QPS, 12K read QPS (3x)

Storage:
URLs: 100M × 12 × 10 = 12B URLs
Size: 12B × 200 bytes = 2.4 TB
With analytics: +6 TB (assume 10 clicks per URL)
Total: ~9 TB over 10 years

Cache:
Cache hot URLs (20% of last 30 days):
- 100M/month = 3.3M/day
- 3.3M × 30 days = 100M URLs
- 100M × 0.2 = 20M hot URLs
- 20M × 200 bytes = 4 GB
This is tiny! Cache everything recent!

Bandwidth:
```

```
Incoming: 40 QPS × 200 bytes = 8 KB/s
Outgoing: 4K QPS × 200 bytes = 800 KB/s

Database:
- PostgreSQL (simple, ACID, 4K QPS easily handled)
- Redis cache (80% hit ratio, < 1ms latency)
- Cassandra for analytics (time-series clicks)
"
```

Walkthrough 3: "Design Dropbox"

Interviewer: "Design file sync service. 50M users."

You:

```
"Let me estimate:

Assumptions:
- 50M registered users, 10M DAU (20% active)
- Each user has 200 files (average)
- Average file: 1 MB
- Each user syncs 5 files per day

Traffic:
Syncs: 10M × 5 = 50M syncs/day ÷ 100K = 500 QPS
Peak: 500 × 3 = 1.5K QPS

Storage:
Total files: 50M users × 200 files = 10 billion files
Total storage: 10B × 1 MB = 10 PB

With 3 replicas (redundancy):
Total: 10 PB × 3 = 30 PB

Bandwidth:
Upload/Download: 500 QPS × 1 MB = 500 MB/s
Peak: 1.5 GB/s

Metadata cache:
- File paths, versions, permissions
- 10B files × 1 KB metadata = 10 TB
- Cache 20%: 2 TB across servers
"
```

Quick Reference Cheat Sheet

Time Conversions

```
1 second   = 1,000 ms
1 minute   = 60 seconds
1 hour     = 3,600 seconds
1 day      = 86,400 seconds (~100K for rough estimates)
1 month    = 2,592,000 seconds (~2.5M)
1 year     = 31,536,000 seconds (~30M)
```

Storage Conversions

```
1 KB  = 1,024 bytes        (~1 thousand)
1 MB  = 1,024 KB           (~1 million bytes)
1 GB  = 1,024 MB           (~1 billion bytes)
1 TB  = 1,024 GB           (~1 trillion bytes)
1 PB  = 1,024 TB           (~1 quadrillion bytes)
1 EB  = 1,024 PB           (~1 quintillion bytes)
```

QPS Estimation Shortcuts

```
Daily Operations → QPS:
1M/day     = 10 QPS
10M/day    = 100 QPS
100M/day   = 1K QPS
1B/day     = 10K QPS
10B/day    = 100K QPS
100B/day   = 1M QPS
```

Typical Ratios

```
Read-to-Write Ratios:
- Social media: 100:1 to 1000:1
- Messaging: 1:1
- E-commerce: 10:1 to 100:1
- Analytics: Write-heavy (opposite)

Cache Hit Ratios:
- Well-designed: 80-90%
- Average: 70-80%
- Poor: < 70%

Peak to Average:
- Normal: 2-3x
- Social (viral): 5-10x
- E-commerce (Black Friday): 10-20x
```

Storage Per Object

```
User profile: 1-2 KB
Tweet/Post: 280 bytes - 1 KB
Email: 50-100 KB
Small image: 50-100 KB
Medium image: 200-500 KB
Large image: 1-3 MB
Short video (1 min): 10-50 MB
HD video (1 min): 50-100 MB
```

Interview Tips & Tricks

Tip 1: Start with Round Numbers

Example:

```
Interviewer: "500 million daily active users"
You: "So roughly 500M DAU, I'll round to 500M for calculations"

Interviewer: "Each user posts 2.3 times per day"
You: "I'll approximate that as 2 posts per day"
```

Why: Makes math easier, shows you understand precision isn't critical

Tip 2: Show Units in Every Step

Bad:

```
100 × 200 = 20,000
20,000 × 365 = 7,300,000
```

Good:

```
100M photos × 200 KB = 20,000,000,000 KB = 20 TB per day
20 TB per day × 365 days = 7,300 TB = 7.3 PB per year
```

Tip 3: Sanity Check Your Numbers

After calculating, ask yourself:

- "Does this make sense?"

- "Is 1 PB reasonable for Instagram photos? Yes!"
- "Is 10 EB reasonable for Twitter text? No, that's too high!"

Common Sanity Checks:

```
Twitter text storage should be < 100 TB (tiny)
Instagram photo storage should be > 10 PB (huge)
WhatsApp messages should be > 1 PB but < 10 PB
YouTube videos should be > 1 EB (massive)
```

Tip 4: Mention Optimizations

After calculations, always mention:

```
Storage optimization:
"With compression, we can reduce this by 30-50%"

Bandwidth optimization:
"With CDN caching at 90% hit ratio, origin only serves 10%"

Memory optimization:
"Using the 80-20 rule, we only need to cache 20% of data"
```

Tip 5: Compare to Real Systems

Say things like:

```
"This is similar to Twitter's scale"
"Instagram actually uses ~100 PB of storage"
"Netflix serves petabytes per day from CDN"
"WhatsApp handles billions of messages"
```

Common Mistakes to Avoid

❌ Mistake 1: Being Too Precise

Bad: "387,654,321 requests per day"
Good: "~400 million requests per day"

Why: Precision gives false confidence, rough numbers are sufficient

❌ Mistake 2: Forgetting Peak Traffic

Bad: "10K QPS average"
Good: "10K QPS average, 30K QPS peak (3x factor)"

Why: Systems must handle peak load, not average

❌ Mistake 3: Not Showing Units

Bad: "Storage is 100"
Good: "Storage is 100 TB"

Why: Interviewer doesn't know if you mean MB, GB, or TB

❌ Mistake 4: Ignoring Media

Bad: "Twitter stores 1 TB of tweets"
Good: "Twitter stores 50 GB of text + 10 PB of images"

Why: Media dominates storage, don't ignore it

❌ Mistake 5: Forgetting Replication

Bad: "Need 10 TB storage"
Good: "Need 10 TB × 3 replicas = 30 TB storage"

Why: Replication is standard for reliability

Practice Problems

Problem 1: Design Google Photos

Given:

- 1B users, 500M DAU
- Each user uploads 5 photos per day
- Each photo: 3 MB
- Each user views 50 photos per day
- Store for lifetime (20 years)

Your turn! Calculate:

1. Write QPS?
2. Read QPS?
3. Total storage for 20 years?
4. Bandwidth required?
5. Cache size?

▶ Click for Answer

```
Traffic:
- Uploads: 500M × 5 = 2.5B/day ÷ 100K = 25K write QPS
- Views: 500M × 50 = 25B/day ÷ 100K = 250K read QPS
- Peak: 75K write, 750K read QPS

Storage:
- Daily: 2.5B × 3 MB = 7.5 PB/day
- Yearly: 7.5 PB × 365 = 2,737 PB = 2.7 EB
- 20 years: 2.7 EB × 20 = 54 EB
- With compression (50%): 27 EB

Bandwidth:
- Upload: 25K QPS × 3 MB = 75 GB/s
- Download: 250K QPS × 3 MB = 750 GB/s
- With CDN (95% cache): 37.5 GB/s origin

Cache:
- Hot photos (20% of last 90 days):
- 2.5B × 90 × 0.2 = 45B photos
- 45B × 3 MB = 135 PB (too large to cache originals!)
- Cache thumbnails instead: 45B × 200 KB = 9 PB
- Still large! Cache metadata only: 45B × 1 KB = 45 TB
```

Problem 2: Design Spotify

Given:

- 400M users, 100M DAU
- Each user streams 10 songs per day
- Each song: 5 MB (compressed)
- Catalog: 100M songs
- Store catalog forever

Calculate: Traffic, Storage, Bandwidth, Cache

▶ Click for Answer

```
Traffic:
- Streams: 100M × 10 = 1B/day ÷ 100K = 10K QPS
- Peak: 30K QPS

Storage:
- Catalog: 100M songs × 5 MB = 500 TB
- With multiple qualities: 500 TB × 3 = 1.5 PB

Bandwidth:
- Download: 10K QPS × 5 MB = 50 GB/s
- With CDN (90% cache): 5 GB/s origin
```

```
Cache:
- Hot songs (20% of catalog): 100M × 0.2 = 20M songs
- 20M × 5 MB = 100 TB
- Distributed: 100 TB ÷ 100 servers = 1 TB each
```

The Universal Template

Use this template for ANY system design question:

```
1. TRAFFIC (QPS):
   Write QPS = Writes per Day ÷ 100K
   Read QPS = Reads per Day ÷ 100K
   Peak QPS = Average × 3
   Ratio = Reads / Writes

2. STORAGE:
   Daily = Items per Day × Size per Item
   Total = Daily × 365 × Years
   With Replication = Total × 3

3. BANDWIDTH:
   Incoming = Write QPS × Object Size
   Outgoing = Read QPS × Object Size
   With CDN = Outgoing × (1 - Cache Hit Ratio)

4. MEMORY (Cache):
   Cache Size = Total Data × 0.2 (80-20 rule)
   Servers = Cache Size ÷ 100 GB
   Cache Hit Ratio = ~80%

5. DATABASE:
   If Read-Heavy → SQL + Read Replicas + Cache
   If Write-Heavy → Cassandra/NoSQL
   If < 1 TB → Single SQL instance
   If > 10 TB → Sharding or NoSQL
```

Real Numbers from Production Systems

Twitter (Actual)

```
DAU: 400M
Tweets/day: 500M
QPS: ~6K write, ~300K read
Storage: ~100 PB
Bandwidth: ~1 TB/s (with CDN)
```

Instagram (Actual)

```
DAU: 500M
Photos/day: 95M
QPS: ~1K write, ~500K read
Storage: ~100 PB
Cache: Redis Cluster (50+ nodes)
```

WhatsApp (Actual)

```
DAU: 2B
Messages/day: 100B
QPS: ~1M write, ~1M read
Storage: ~10 PB
Erlang servers: 1000+
```

YouTube (Actual)

```
Videos watched/day: 5B
Upload: 500 hours/minute
Storage: ~1 EB (yes, exabyte!)
Bandwidth: ~1 PB/s globally
CDN: Critical infrastructure
```

Interview Scoring Rubric

What Interviewers Look For:

✅ **Structured Approach** (30%)

- Clear methodology
- Step-by-step calculations
- Organized presentation

✅ **Reasonable Assumptions** (20%)

- States assumptions clearly
- Numbers are realistic
- Justifies choices

✅ **Correct Math** (20%)

- Gets order of magnitude right
- Shows work

- Unit conversions correct

✅ **Identifies Bottlenecks** (15%)

- "At 1M QPS, we need caching"
- "44 PB requires distributed storage"
- "This is write-heavy, need Cassandra"

✅ **Mentions Optimizations** (15%)

- CDN for bandwidth
- Caching for reads
- Compression for storage

The Mental Math Trick

Multiplication by Powers of 10

```
× 10:    Add one zero
× 100:   Add two zeros
× 1,000: Add three zeros (1K)
× 1,000,000: Add six zeros (1M)
× 1,000,000,000: Add nine zeros (1B)
```

Example:

```
5 MB × 1,000 = 5,000 MB = 5 GB
100 KB × 1,000,000 = 100,000,000 KB = 100 GB
```

Division by Large Numbers

```
÷ 100:     Remove two zeros
÷ 1,000:   Remove three zeros
÷ 86,400:  Divide by 100,000 (close enough)
```

Example:

```
10,000,000 ÷ 100,000 = 100
1,000,000,000 ÷ 100,000 = 10,000
```

Advanced: Server Capacity Estimation

Typical Server Specs

Small Instance (t3.medium):

- 2 vCPU, 4 GB RAM
- Handles: ~500 QPS
- Cost: ~$50/month

Medium Instance (m5.xlarge):

- 4 vCPU, 16 GB RAM
- Handles: ~2,000 QPS
- Cost: ~$150/month

Large Instance (m5.4xlarge):

- 16 vCPU, 64 GB RAM
- Handles: ~10,000 QPS
- Cost: ~$600/month

Calculating Server Count

Formula:

```
Servers = (Peak QPS / QPS per Server) × Safety Factor

Example:
Peak: 30K QPS
Server capacity: 2K QPS
Safety factor: 1.5 (50% headroom)

Servers = (30,000 / 2,000) × 1.5 = 22.5
Round up: 25 servers
```

Database Server Capacity

PostgreSQL (single instance):

- Reads: 10K-50K QPS
- Writes: 1K-10K QPS
- Storage: Up to 16 TB

Cassandra (per node):

- Reads: 10K-50K QPS
- Writes: 10K-100K QPS
- Storage: Up to 8 TB per node

Redis (single instance):

- Reads: 100K-1M QPS
- Writes: 100K-1M QPS
- Storage: Up to 512 GB RAM

Estimation Worksheet Template

Copy this for every interview:

```
SYSTEM: ________________

ASSUMPTIONS:
- DAU: ________________
- Operations per user: ________________
- Data size: ________________
- Retention: ________________

TRAFFIC CALCULATIONS:
- Daily operations: DAU × ops = ________________
- Average QPS: daily ÷ 100K = ________________
- Peak QPS: avg × 3 = ________________
- Read/Write ratio: ________________

STORAGE CALCULATIONS:
- Per item size: ________________
- Daily: items/day × size = ________________
- Yearly: daily × 365 = ________________
- Total: yearly × years = ________________
- With replication: total × 3 = ________________

BANDWIDTH CALCULATIONS:
- Incoming: write QPS × size = ________________
- Outgoing: read QPS × size = ________________
- With CDN: outgoing × (1 - hit ratio) = ________________

MEMORY/CACHE CALCULATIONS:
- Hot data (20%): total × 0.2 = ________________
- Cache servers: cache ÷ 100 GB = ________________
- Cache hit ratio: ________________

DATABASE CHOICE:
- Primary: ________________ (because: ________________)
- Cache: ________________
- Analytics: ________________
```

Final Checklist

Before saying "I'm done with calculations", verify:

✅ **Traffic Estimation**

- Calculated write QPS
- Calculated read QPS
- Mentioned peak factor (2-3x)
- Identified if read-heavy or write-heavy

✅ **Storage Estimation**

- Calculated total storage
- Considered retention period
- Mentioned replication (3x)
- Separated data types (text vs media)

✅ **Bandwidth Estimation**

- Calculated incoming bandwidth
- Calculated outgoing bandwidth
- Mentioned CDN impact

✅ **Memory Estimation**

- Applied 80-20 rule
- Calculated cache size
- Determined number of servers

✅ **Identified Bottlenecks**

- Database: "At 10K write QPS, need sharding"
- Network: "At 1 TB/s, need CDN"
- Memory: "At 10 TB cache, need 100 servers"

Summary: The Golden Rules

1. **Round liberally** - 387M → 400M
2. **State assumptions** - "Assuming X..."
3. **Show your work** - Write calculations down
4. **Use shortcuts** - Daily ÷ 100K = QPS
5. **Apply 80-20 rule** - Cache 20%, serve 80%
6. **Mention peak traffic** - Average × 3
7. **Don't forget media** - Images dominate storage
8. **Include replication** - × 3 for redundancy
9. **Sanity check** - Does 1 EB for Twitter make sense? No!
10. **Mention optimizations** - CDN, compression, caching

References

Books

1. **"Designing Data-Intensive Applications"** - Martin Kleppmann
2. **"System Design Interview"** - Alex Xu

Online Resources

1. **System Design Primer** - GitHub (donnemartin)
2. **Google's Latency Numbers** - Jeff Dean
3. **AWS Calculator** - For cost estimates
4. **High Scalability Blog** - Real-world numbers

Videos

1. **"System Design Course"** - YouTube (Gaurav Sen)
2. **"Back of Envelope Calculations"** - YouTube (SystemDesignInterview)

Appendix: Pre-Calculated Common Scenarios

Scenario: 1 Million Users

```
Assumptions: 1M DAU, 10 operations/user/day
Traffic: 10M/day ÷ 100K = 100 QPS
Storage (1 KB per op): 10M × 1 KB = 10 GB/day = 3.6 TB/year
Cache: 1M users × 10 KB = 10 GB
Servers: 5–10 app servers
```

Scenario: 10 Million Users

```
Assumptions: 10M DAU, 10 operations/user/day
Traffic: 100M/day ÷ 100K = 1K QPS
Storage: 100M × 1 KB = 100 GB/day = 36 TB/year
Cache: 10M × 10 KB = 100 GB
Servers: 10–20 app servers, single DB with replicas
```

Scenario: 100 Million Users

```
Assumptions: 100M DAU, 10 operations/user/day
Traffic: 1B/day ÷ 100K = 10K QPS
Storage: 1B × 1 KB = 1 TB/day = 365 TB/year
Cache: 100M × 10 KB = 1 TB across 10 servers
Servers: 50–100 app servers, sharded DB or NoSQL
```

Scenario: 1 Billion Users (Facebook Scale)

```
Assumptions: 1B DAU, 10 operations/user/day
Traffic: 10B/day ÷ 100K = 100K QPS
Storage: 10B × 1 KB = 10 TB/day = 3.6 PB/year
Cache: 200M hot users × 10 KB = 2 TB across 20 servers
Servers: 500–1000 app servers, distributed NoSQL
```

Document Version: 1.0
Last Updated: January 8, 2025
For: System Design Interview Preparation
Status: Complete & Interview-Ready ✅

Remember: In interviews, being roughly right with clear reasoning beats being precisely wrong!