1 CheatSheet: System Design For Job Interview INTERVIEW

- PDF Link: cheatsheet-systemdesign-A4.pdf, Category: interview
- Blog URL: `https://cheatsheet.dennyzhang.com/cheatsheet-systemdesign-A4`
- Related posts: CheatSheet: Leetcode For Code Interview, CheatSheet: Well-Known Papers For IT Industry, #denny-cheatsheets

File me Issues or star this repo.

1.1 Reference

Name	Summary
YouTube	YouTube: Intro to Architecture and Systems Design Interviews, YouTube: System Design Interview
YouTube	YouTube Channel: Success in Tech, YouTube: Scalability Harvard Web Development
YouTube	YouTube: Prepare for Your Google Interview: Systems Design
Papers	CheatSheet: Well-Known Papers For IT Industry, Github: papers-we-love
Github	Github: donnemartin/system-design-primer, Github: checkcheckzz/system-design-interview
Github	Github: puncsky/system-design-and-architecture, Github: yangshun/tech-interview-handbook
Website	Website: hiredintech - System Design, Website: blog.gainlo.co
Website	Website: interviewing.io, Website: interviewbit.com
Cheatsheet	CheatSheet: Leetcode For Code Interview, CheatSheet: Common Code Problems & Follow-ups
Cheatsheet	CheatSheet: System Design For Job Interview, CheatSheet: SRE/DevOps/Sysadmin
Cheatsheet	CheatSheet: Behavior Questions For Coder Interview, Programming Language Implemenetations CheatS
Cheatsheet	CheatSheet: Concurrency & Parallel Programming
Coding	Code problems for #oodesign
Individual Tech Blog	Blog: highscalability, Blog: All Things Distributed - Amazon CTO
Company Tech Blog	Website: Facebook Engineering, Website: Google Developers
Company Tech Blog	Medium: Netflix Blog, Medium: Airbnb Engineering & Data Science
Company Tech Blog	Medium: Instagram Engineering, Medium: Mixpanel
Company Tech Blog	Shopify Engineering, Github Engineering
Company Tech Blog	Doordash Engineering, Uber Engineering
Reference	Link: Facebook Engineering Interview, Link: The System Design Process

1.2 Design Problems Per Category

Num	Name	Summary
1	K/V DB store	Design K/V DB; Design memcache/redis
2	Data synchronization	Design dropbox client sync
3	Resource/Task scheduling	Design web crawler; Delayed task queue; Design a distributed message queue
4	Design a distributed component	Design a distributed hit counter, Design a distributed UUID generator
5	Design a SNS system	Design Twitter News Feed
6	Design API Gateway	Design An API Rate Limiter
7	Design a logging & metrics system	Pull vs Push model
8	Design a gaming system	Design: Leaderboard Ranking
10	Design a small scale MIS system	Design: Flight booking service, Design a payment processor
11	Recommendation system	Design amazon book recommendation system
12	Design a communication system	Design a message chat room
13	Design an ads system	

1.3 Top 50 Component Design

Num	Name	Summary
1	Top K Frequent Elements in Recent X mins	Github: link
2	Design An API Rate Limiter	
3	Design: Leaderboard Ranking	
4	Delayed task queue	
5	Spam Filter: design a system to block malicious IPs	Github: link
6	Find duplicates files across 1000 servers with 10 million files	Github: link
7	Design a monitoring system to check 10,000 nodes	Github: link
8	Design a scalable and reliable notification service	Github: link
9	Web crawler for 1 billion URL from 1 seed URL	Github: link
10	Design twitter timeline feature	Github: link
11	How to upload large videos at scale	Github: link
12	Real-time Deduping At Scale	Github: link
13	How quoram based DB works when nodes join or leave	
14	How to implement redis clustering	Github: link
15	How to deployment 1GB binary to 10,000 servers	Github: link
16	How to distribute TB data from a server to 10,000 nodes	
17	Merge big datasets across different servers	Github: link
18	Unique url hits	
19	Design a distributed counter	
20	Design a distributed message queue	
21	Design a distributed cache service	
22	Design a distributed Hashmap	
23	Design a distributed UUID generator	
24	Design a git service	
25	Design: A Parking Lot Service	
26	Design a distributed transaction	
27	Design: A URL Redirecting Feature	
28	Give three 1TB disks, how to store 2TB data with redundancy	Github: link. XOR bit manipulation
29	How to support feature of "diff big1.bin big2.bin"	#lcs - Longest Common Subsequence
30	How to support "rsync big1.bin ssh:/big2.bin" in a doggy network	delta-transfer algorithm. Weak Hashing + Strong Ha
31	Avoid double payment in a distributed payment system	Link: Avoiding Double Payments in a Distributed Pa

1.4 Concurrency Problems

- CheatSheet: Concurrency & Parallel Programming

1.5 Top 30 Product Design

Num	Name
1	Design: TinyURL - A URL Shorterner Service
2	Design Twitter News Feed
3	Design K/V DB
4	Design autocomplete/typeahead
5	Design a online contest system like leetcode.com
6	Design Google Calendar
7	Design a load balancer
8	Design: Flight booking service
9	Design: Uber Backend
10	Design: An Elevator Service
11	Design amazon shopping cart
12	Design: Google Suggestion Service
13	Design a payment processor
14	Design google doc
15	Design gmail
16	Design RSS news reader
17	Design a client-server API to build a rich document editor
18	Design instagram, a photo sharing app
19	Design Yelp, a location-based system
20	Design Pastebin.com
21	Design amazon book recommendation system
22	Design Google PageRank
23	Design messaging/notification system
24	Design memcache/redis
25	Design a voice conference system
26	Design an API gateway
27	Design slack
28	Design a service auto-discovery feature
29	Design a secrets management system
30	Design Google Adsense fraud detection
31	Design The Great Firewall

1.6 Process Of System Design

Num	Name	Summary
1	Outline use cases: List major and focus on some	Show good sense. The questions you asked define your level
2	Estimate scale: **Data** + **Traffic**	Back-of-the-envelope estimation
3	Defining data model	It helps to clarify how data will flow among different components
4	Abstract design	Sketch main components, explain workflow, avoid too deep for details
5	Detailed design + discussion with interviewers	Explain trade-off of your proposal + on-demand deep dive
6	Identify and resolve Bottlenecks	**Key challenges** + **Trade-Offs**. Usuaully no optimal solution(s)
7	Scale your design	Availability, Resiliency, Scalability, Security, Serviceability, etc
8	Show your relevant experience and learning	Industry best practice; You experience of scaling/trade-off/resiliency

1.7 Common Mistakes Of System Design

Num	Name	Summary
1	Run into an opinioned solutions before clarification	Inexperienced; Hard to communicate
2	Not driving the conversation	Inexperienced
3	General answers without your personal experience/thinking	
4	Makes interviewers feeling you're stubborn	

1.8 Top 30 Concepts For Feature/System Design

Num	Name	Summary
1	Caching	Stores data so that future requests of data retrieval can be faster
2	Message Queue	Provides an asynchronous communications protocol,
3	Data Partition & Sharding	Break up a big data volume into many smaller parts
4	DB Indexing	Create indexes on multiple columns to speed up table look up
5	DB replication	Duplicate data to increase service availability
6	CAP: Consistency/Availability/Partition	A distributed database system can only have 2 of the 3
7	DB: SQL & NoSQL	Relational databases and non-relational databases
8	Concurrency & Communication	
9	Pessimistic And Optimistic Locking	
10	Consistency Module	weak consistency, eventual consistency, strong consistency
11	Conflict resolution	Quorum, vector lock, reconcile on read/write, CRDTs
12	B+ Tree	
13	Networking: HTTP	
14	Networking: TCP/UDP	
15	Pull vs Push model	
16	Garbage Collection	
17	Memory Management	
18	Heartbeats	
19	Self Protection	API Rate limit, Circuit breaker, bulkhead, throttling
20	Filesystem	
21	API: RPC vs gRPC vs REST	
22	Load balancer	
23	Scale up vs Scale out	Vertical scaling and Horizontal scaling
24	API Design	
25	Session management	
26	Networking: TCP vs UDP	
27	Consistency patterns	Weak consistency, Eventual consistency, Strong consistency
28	Availability patterns	Fail-over vs Replication
29	CDN - Content Delivery Network	Edge caching
30	Monitoring	
31	Security	
32	Networking: DNS	
33	Linux signals	

1.9 Top 20 Advanced Data Structure & Algorithms

Num	Name	Summary
1	Consistent Hash	
2	Bloom filter	A space-effcient query returns either "possibly in set" or "definitely not"
3	hyerloglog for count-distinct problem	Estimation: the count of unique values with relatively high accuracy(98
4	Reservoir Sampling	
5	Merkle Tree	
6	LPM(Longest Prefix Match)	
7	Frugal Streaming	
8	Gossip	Propagate cluster status
9	Vector Clocks/Version Vectors	
10	Lossy Counting	
11	Skip list	
12	CRDTs (Conflict-Free Replicated Data Types)	
13	choice-of-2 in load balancer	
14	Range-based query	
15	SSTable (Sorted Strings Table)	
16	MemTable	
17	LSM (Log Structured Merge Trees)	
18	Two-phase commit/Three-phase commit	Github: link
19	Paxos and raft protocol	
20	Ring buffer	
21	cuckoo hashing	Resolve hash collisions with worst-case constant lookup time
22	snappy/lzss	Fast data compression and decompression
23	S2 Geometry	Build geographic database in a better way
24	geohash	
25	Quadtree	
26	DHT - distributed hash table	

https://raw.githubusercontent.com/dennyzhang/cheatsheet.dennyzhang.com/master/cheatsheet-systemdesign-A4/dynamo-summary.png

1.10 Explain tools: how XXX supports XXX?

Num	Name	Summary
1	How JDK implement hashmap?	
2	Explain java garbage collection model	
3	Explain raft/etcd	
4	How OS supports XXX?	

1.11 Cloud Design Principles

Num	Name	Summary
1	Fail fast	
2	Design for failure	
3	Immutable infrastructure	
4	Cats vs Cattle	Avoid snowflake servers
5	Auto healing	
6	Async programming	
7	GitOps operational model	
8	Event-Driven Architectures	

1.12 Cloud Design Patterns

Num	Name	Summary
1	Ambassador pattern	Create helper service to send network requests, besides the main sevice
2	Cache-Aside pattern	Load data on demand into a cache from a data store
3	Circuit Breaker pattern	If a request takes too many reousrce, abort it
4	Bulkhead pattern	Isolate elements into pools, so that one fire won't burn all
5	Gateway Aggregation pattern	Aggregate multiple individual requests into a single request
6	Priority Queue pattern	Support different SLAs for different individual clients
7	Strangler pattern	Incrementally migrate a legacy system piece by piece

1.13 Engineering Of Well-Known Products

Name	Summary
Google	Link: Google Architecture
Facebook	Link: Facebook Live Streams
Twitter	Link: Twitter Image Service, YouTube: Timelines at Scale
Uber	Link: Lessons Learned From Scaling Uber
Tumblr	Link: Tumblr Architecture
StackOverflow	Link: Stack Overflow Architecture

1.14 Grow Design Expertise In Daily Work

Num	Name	Summary
1	Keep the curiosity	Thinking about interesting/weird questions helps
2	Deep dive into your daily work	Unify and normalize problems from daily work
3	Learn the work of your coleagues	Indirect working experience also help
4	Popular products under the hood	Once you notice an interesting feature, think about how it's supported?
5	Read engineering blogs	Especially for big companies
6	Tools under the hood	Common tools/frameworks
7	Try tools	Use cases; Alternatives; Pros and Cons
8	Read papers	Best practices in papers
9	Try new things	Gain hands-on experience; evaluate alternatives
10	Datastore & OS	Learn how databases and operating systems work
11	Language implementation	Deep dive into one programming language. Java, Python, Golang, etc

1.15 More Resources

https://github.com/binhnguyennus/awesome-scalability
https://highscalability.com/blog/2013/4/15/scaling-pinterest-from-0-to-10s-of-billions-of-page-views-a.html
https://medium.com/hackernoon/top-10-system-design-interview-questions-for-software-engineers-8561290f044
https://draveness.me/
https://docs.microsoft.com/en-us/azure/architecture/patterns/
https://github.com/sdmg15/Best-websites-a-programmer-should-visit
https://www.infoq.com/presentations/Pinterest/