software engg interviewprep . 10 Mar 2024

Best of the linkedin posts from Arpit Adlakha

Arpit's Linkedin Profile : https://www.linkedin.com/in/arpit-adlakha-30691a101/
Arpit's Bio : Engineering @Quizizz | Ex- AWS Vancouver | LinkedIn Top Voice 2024

Playlist Items

Getting a High Paying Software Engineer Job in 2024 🚀 !

DSA Problem Solving 👨‍🏭
1. Grind 75 (Customised plan for the time you have for preparation and hours in a day and week): https://lnkd.in/gKN2TZKY
3. Dynamic Programming Patterns: https://lnkd.in/gVNgiDWH
4. Tree Patterns: https://lnkd.in/gYB7zUX6
5. Graph Patterns: https://lnkd.in/geZGw4Vt
6. Substring Problem Patterns: https://lnkd.in/gt23kRen
7. Backtracking Problem Pattern: https://lnkd.in/gk6JqQD4
8. Two Pointers Patterns: https://lnkd.in/gfea3T9v
9. Binary Search Patterns: https://lnkd.in/gHhq5MrR
10. Cloning Problems Patterns: https://lnkd.in/gJWqTDV8
11. Bit Manipulation Pattern: https://lnkd.in/gE6cdc-g
12. Heap Patterns: https://lnkd.in/gugVTJsT
13. Sliding Window Patterns: https://lnkd.in/gb5NeskQ
14. Mastering Recursion: https://lnkd.in/gJ_5VxQZ

OOPS and Low Level Design 🏝️
1. Design Patterns : https://lnkd.in/g4QZ5s7n
2. Patterns Implementation in Java : https://lnkd.in/gvAQh5H3
3. Patterns Implementation in Javascript: https://www.patterns.dev/
4. Patterns implementation in Golang: https://lnkd.in/gsWkWy_4
5. Low Level Design Questions and Solutions: https://lnkd.in/g2DF22wP

High System Design ⛰️
1. System Design Primer: https://lnkd.in/g89Mtjea
2. System Design Repo by Alex Xu: https://lnkd.in/gPmCmvVg
3. Byte Byte Go Youtube: https://lnkd.in/gaDNg8xr
4. Computer Science Papers Biggest Relevant Collection: https://lnkd.in/gkxJ8QAM
5. System Design Repo With Concepts explained in the repo only: https://lnkd.in/gtYDU8UD
6. Systems that scale Newsletter: https://lnkd.in/gG7hSwf6
7. Gaurav Sen Youtube: https://lnkd.in/grjkY55Z

Blogs you need to read for real life case studies 📖 :

1. Building and Operating S3 the biggest storage system: https://lnkd.in/g65anHkf
2. Modern Load Balancing and Networking by the creator of Envoy: https://lnkd.in/gn4BA3sD
3. The Log: What every software engineer should know about real-time data's unifying abstraction: https://lnkd.in/gQ7WsE8k
4. Keeping Netflix reliable using Load Shedding: https://lnkd.in/gSx8p32U
5. How Discord stores billions of messages: https://lnkd.in/gduE8Kgx
6. How Discord stores trillions of messages: https://lnkd.in/gRNwqnuH
7. Netflix Fault Tolerance in a high volume distributed system: https://lnkd.in/gGFensR4
8. How Pinterest runs Kafka at scale: https://lnkd.in/gb5skEtU
9. How browsers work: https://lnkd.in/gumG6WQw
10. Practical Guide to Kafka Storage Internals: https://lnkd.in/gA9XJteP

Top system design articles you need to read this week !

1. Caching strategies at Twitter: https://lnkd.in/dUAque7i
2. Improving Multi-CDN Delivery on Netflix: https://lnkd.in/dSBVbBMz
3. Amazon Physalia - Millions of tiny databases: https://lnkd.in/dbaUXqcy
4. Airbnb Microservice Architecture - https://lnkd.in/duzNCmqw
5. Evolution of the Netflix API Architecture - https://lnkd.in/dKgZnWsD
6. How Zapier Automates Billions of Tasks - https://lnkd.in/d74kktbV
7. How LinkedIn Scaled to 930 Million Users - https://lnkd.in/dSCZG9bx
8. How Pinterest scaled to 11 million users with only 6 engineers - https://lnkd.in/drx93zhb
9. How Instagram scaled to 14 million users with only 3 engineers - https://lnkd.in/dJ54KJne
10. Intro to Change Data Capture: https://lnkd.in/dsk4gv5d

Follow Arpit Adlakha for more !

Some of the best System Design Resources

Some of the best System Design Resources

1. Uber Time Series DB: https://lnkd.in/dzpbrmAw

2. Airbnb Idempotency: https://lnkd.in/dmnevUzk

3. Facebook Cluster Management: https://lnkd.in/dyB_rVRU

4. Google Autopilot - Autoscaling: https://lnkd.in/dsY3u5VM

5. Netflix Workflow Orchestration: https://lnkd.in/dZcfsTxZ

6. Opensource WorkflowManagement: https://lnkd.in/d3RqkjDq

7. Facebook Video Broadcasting:

https://lnkd.in/dvnXntNF

8. LinkedIn Brooklin- Real time data streaming:
https://lnkd.in/dfbFcB2r

9. Amazon S3 Performance hacks:
https://lnkd.in/dFgxQifh

10. Amazon S3 object expiration:
https://lnkd.in/ddb5RPgg

11. Circuit Breaker Algorithm:
https://lnkd.in/dtMAgFgN

I maintain a GitHub repo that points to a collection of Resources like these:
https://lnkd.in/g8W4NrP5

hashtag#design hashtag#data hashtag#github hashtag#google hashtag#facebook hashtag#amazon hashtag#netflix hashtag#streaming hashtag#video hashtag#linkedin

Top Books for System Design and Architecture 👨‍💻

Here are the top 11 free resources to clear any system design interview

There are tons of system design resources on the internet. Here are the top 11 free resources to clear any system design interview.

1. System Design Concepts : https://lnkd.in/gmhYfk3u

2. HighScalibility Website: https://lnkd.in/g-BGfKRu

3. Hussein Nasser Youtube: https://lnkd.in/gvJ3bjgQ

4. CodeKarle System Design Playlist: https://lnkd.in/g4yacWgy

5. Gaurav Sen's Youtube Channel : https://lnkd.in/gqnCX_bF

6. ByteByteGo Youtube Channel: https://lnkd.in/gtvAtCxR

7. Level Up System Design Survival Guide: https://lnkd.in/gDMiqHN3

8. Azure's Architecture Guide: https://lnkd.in/gyQvqHhc

9. Amazon CTO's website on all things distributed: https://lnkd.in/gz5h_ffY

10. Best Architecture Notes: https://lnkd.in/get4KpZ5

11. System Design Primer on Github: https://lnkd.in/g89Mtjea

Follow Arpit Adlakha for more.

The biggest System Design BluePrint 🖨️

The biggest System Design BluePrint 🖨️

🔍 Identify the Scope & Use Cases
1. Clearly define the scope of the system to avoid feature creep.
2. Create user stories that describe sequences of events leading to a useful outcome.
3. Identify your target users and understand their needs and how they'll interact with the system.

📊 Constraints & Scale
1. Identify constraints like network latency, data storage limits, and user load.
2. Measure the scale in terms of requests per second, types of requests, data read/written per second.
3. Consider special requirements like multi-threading, read or write orientation.

🏗️ High-Level Architecture
1. Sketch out the main components like frontend, backend, database, and their interactions.
2. The application service layer should be robust enough to handle various types of requests.
3. Data storage layer often includes a load balancer, service partition, and master/slave database clusters.

🛠️ Component Design
1. Define specific APIs for each component to interact with others.
2. Use object-oriented design principles for functionalities.
3. Map features to modules and consider relationships like composition and inheritance among them.

📈 Understanding Bottlenecks
1. Evaluate if a load balancer is needed to distribute user requests across multiple servers.
2. Consider data partitioning if the database becomes a bottleneck.
3. Evaluate the need for in-memory caching solutions like Memcached or Redis to speed up data retrieval.

🔪 Scaling Your Design
1. Vertical Scaling: Enhance the power of your existing machine by adding more CPU, RAM.
2. Horizontal Scaling: Add more machines to your existing pool of resources to distribute the load.

🔥 Caching
1. Application Caching: Integrate caching logic into the application code to check and retrieve values.
2. Database Caching: Utilize the database's built-in caching mechanisms, which can be optimized for your specific use case.

⚖️ Load Balancing
1. Types of Load Balancers:Smart Client: Difficult to perfect but highly customizable.

2. Hardware Load Balancers: Expensive but extremely reliable.
3. Software Load Balancers: A hybrid approach that works well for most systems.

🗄️ Database Replication & Partitioning
1. Replication: Copy data from one database to another for consistency.
2. Partitioning: Decompose tables either row-wise (horizontally) or column-wise (vertically) to improve performance.

🌐 Platform Layer
Separate the platform and application layers to scale them independently.This allows you to reuse the same infrastructure for different products or interfaces.

🔑 Key Topics for Robust System Design
1. Concurrency: Understand threads, deadlock, and starvation.
2. Networking: Know the basics of TCP/IP.
3. Abstraction: Understand OS, file system, and database.
4. Real-World Performance: Know the speed of RAM, disk, SSD, and network.
4. Estimation: Perform back-of-the-envelope calculations for feasibility checks.

Follow Arpit Adlakha for more !

These are the best engineering blogs for learning the real system design !

AWS Architecture: https://lnkd.in/eEchKJif
Microsoft Tech: https://lnkd.in/etw_7_bN
Engineering at Microsoft: https://lnkd.in/eEKz4ECi
Meta: https://lnkd.in/e8tiSkEv
All Things Distributed: https://lnkd.in/emXaQDaS
Cloudflare: https://lnkd.in/gCPjyTFk
Nextflix Tech: https://lnkd.in/efPuR39b
LinkedIn Engineering: https://lnkd.in/ehaePQth
Uber Engineering: https://eng.uber.com/
Engineering at Quora: https://lnkd.in/em-WkhJd
Pinterest Engineering: https://lnkd.in/esBTntjq
Lyft Engineering: https://eng.lyft.com/
Twitter Engineering: https://lnkd.in/evMFNhEs
Dropbox Engineering: https://dropbox.tech/
Spotify Engineering: https://lnkd.in/eJerVRQM
Github Engineering: https://lnkd.in/eCADWt8x
Instagram Engineering: https://lnkd.in/e7Gag8m5
Canva Engineering: https://canvatechblog.com/
Etsy Engineering: https://lnkd.in/eddzzKRt
Booking.com Tech: https://blog.booking.com/
Expedia Technology: https://lnkd.in/ehjuBE5J
The Airbnb Tech: https://lnkd.in/emGrJbGM
Stripe Engineering: https://lnkd.in/em6Svgyx
Ebay Tech: https://tech.ebayinc.com/

Flickr's Tech: https://code.flickr.net/
Hubspot Product and Engineering: https://lnkd.in/eRGZkBd4
Zynga Engineering: https://lnkd.in/eex5Ddry
Yelp Engineering: https://lnkd.in/epgBW_4J
Heroku Engineering: https://lnkd.in/evgctQjh
Discord Engineering: https://lnkd.in/evY4gpUA
Zomato: https://lnkd.in/e9gf3APD
Hotstar: https://blog.hotstar.com/
Swiggy: https://bytes.swiggy.com/
Shopify Engineering: https://lnkd.in/evvnqQTj
MongoDB Engineering: https://lnkd.in/e9iaqcmZ
Slack Engineering: https://slack.engineering/
DoorDash Engineering: https://lnkd.in/ep5raBZv
Reddit Engineering: https://lnkd.in/e4E_XzaX
Snap Engineering: https://eng.snap.com/blog
Indeed Engineering: https://lnkd.in/ecFS87Dt
Gusto Engineering: https://lnkd.in/e7yVxDKs
Engineering at Birdie: https://lnkd.in/eUqJTpje
Forethough Engineering: https://lnkd.in/esCKvedJ
Ramp: https://lnkd.in/eZHSnx-j
Capital One: https://lnkd.in/ezsKUf_H
Disney Streaming: https://lnkd.in/eWe6w6TT
Dunelm: https://lnkd.in/enJS53DS
The Guardian: https://lnkd.in/e73WNZNN
GiffGaff: https://lnkd.in/eBakqffd

Github Repository for categorised engineering blogs : https://lnkd.in/gxDzj57b

Compensation Details for Top 300 Software Companies in India 🔥

Compensation Details for Top 300 Software Companies in India 🔥

✅ The data is for freshers and for the years 2022-2023.
✅ The data has been sorted in descending order.
✅ The CTC is the initial package that is usually being offered for freshers. It may include the 4 Years Vested Tax - Check in Details Column.
✅ There is data for the internship stipend offered by these companies as well.
✅ Please check the details for the complete breakdown as CTC might not give you a clear picture.

🚩 Also please remember compensation should not be the only criteria while deciding the company you want to join. There are many other factors like work culture, team, learning curve etc which you should consider. But still this gives you a good sense on the top companies to apply for in terms of pay as a fresher.

I used the data from this amazing Leetcode Post https://lnkd.in/gMzj-qK4 and made a pdf out of it.

Share with friends not working in tech 🙈 !

From Novice to Pro in DSA 🚀 . This will change how you do DSA !

From Novice to Pro in DSA 🚀 . This will change how you do DSA !

Save tons of your time, you really don't have to do 1000 Leetcode problems to figure out these patterns and instead you can learn from these Leetcode articles directly.

Dynamic Programming Patterns: https://lnkd.in/gVNgiDWH

Tree Patterns: https://lnkd.in/gYB7zUX6

Graph Patterns: https://lnkd.in/geZGw4Vt

Substring Problem Patterns: https://lnkd.in/gt23kRen

Backtracking Problem Pattern: https://lnkd.in/gk6JqQD4

Two Pointers Patterns: https://lnkd.in/gfea3T9v

Binary Search Patterns: https://lnkd.in/gHhq5MrR

Cloning Problems Patterns: https://lnkd.in/gJWqTDV8

Bit Manipulation Pattern: https://lnkd.in/gE6cdc-g

Heap Patterns: https://lnkd.in/gugVTJsT

Sliding Window Patterns: https://lnkd.in/gb5NeskQ

You will honestly save tons of time, I figured out the patterns probably after doing some 500 questions.

How I Cracked the Google India and Amazon Web Services (AWS) Vancouver, Canada Interview.

How I Cracked the Google India and Amazon Web Services (AWS) Vancouver, Canada Interview.

1. Setting a target for 2-3 months of Preparation. It includes refreshing on dsa and system design and preparing a bit on the behavioural aspects. Allocating time in the evenings and weekends for preparation and spare time to refresh to not get burnt out.

2. Coding and Problem Solving: Studying DSA strictly following leetcode and doing only medium questions. I solved around 100 questions in this round of preparation and leetcode premium is must for

quick detailed solutions and alternatives to what you came up with.
Practice all patterns and types of questions
* Greedy problems
* Sliding Window
* Dynamic Programing
* DFS And BFS in Graphs (Very imp for google)
* Backtracking
* Trees
* Sorting
* Stack, Queue, Linked List
* Bit Manipulation

3. System Design: As we all know there is no one correct answer and after taking almost 100 system design interviews I figured you need to go principles first in these interviews and never try to cram answers from the resources out there. Also all questions asked were new and were not there on the internet.
* Funtional Requirements
* Non Funtional Requirements
* Back of the envelope calculations
* Sql and NoSql databases and understanding popular databases like Mysql, Mongodb and Cassandra.
* Kafka, RabbitMq type systems their differences and use cases.
* Load Balancing, Api Gateway, Service Discovery,CI/CD, Kubernetes etc
* Strong skills for Database Schema both for sql and Nosql
* Deploying and Scaling Web Services
* Writing APIs and security of APIs
* Understand sharding, gossip, consensus algorithms.

4. Behavioural Aspect: Study for the Googlyness and Amazon Leadership principles. You need to have solid examples to back up your answers. Questions will always be like “Tell me a time you did this”.

Some surprises which I did not prepare for :

1. Strong understanding of Concurrency and problems associated with it.
2. Strong understanding of the language you are coding in, for me it was Java.
3. Great testing and code review skills, was tested in google.

Link to all the resources in first comment.

Complete this list and you will never have any problem doing DP problems 👇🔥

Dynamic Programming is tricky but is still asked a lot in programming interviews. The last interviews I had for google and amazon, I got one DP problem in both.

Complete this list and you will never have any problem doing DP problems 👇🔥

Dynamic Programming :

1. DP for Beginners(https://lnkd.in/gHqQqbHM)

2. DP Patterns (https://lnkd.in/g-KQMC8j)

3. How to solve DP - String? Template and 4 Steps to be followed (https://lnkd.in/gaZNqfKM)

4. Dynamic Programming Questions thread (https://lnkd.in/gPCCfG8W)

5. DP Classification (https://lnkd.in/gGF4uNmw)

6. How to approach DP problems(https://lnkd.in/gkAnsSsM)

7. Iterative DP for subset sum problems(https://lnkd.in/gkAnsSsM)

8. DP problems summary (problem categorization) (https://lnkd.in/g5VmKrVj)

9. Categorization of Leetcode DP problems(https://lnkd.in/gHrUAwnV)

10. Must do Dynamic Programming Category wise(https://lnkd.in/gqekyp_V)

11. Dynamic programming is simple(https://lnkd.in/gyrFWrXF)

12. Dynamic Programming on subsets(https://lnkd.in/gNEjM_z4)

13. DP is easy (Thinking process)(https://lnkd.in/gKt85wRz)

Okay you still might not be able to solve a problem its dp 😅

best way to learn real and deep system design concepts

What is the best way to learn real and deep system design concepts ?

Engineering blogs and Computer Science Papers, where people share their experiences from building real systems. If you really want to stand out, you have to give real examples from systems that actually scaled and not from some dummy system design interview book. So here are the two repos you should check out and follow regularly.

Engineering blogs categorised By Companies and popular individuals : https://lnkd.in/gxDzj57b

Computer Science Papers Collection: https://lnkd.in/gkxJ8QAM

11 Things You Should Know About Databases 📚 🚀

11 Things You Should Know About Databases 📚 🚀

1. Database Paradigms: https://lnkd.in/gtmnqRxg
2. CAP theorem: https://lnkd.in/g_gZWd5w
3. Evolutionary Database Design: https://lnkd.in/gABKkirQ)
4. ACID vs BASE in Databases: https://lnkd.in/gU65xBp6
5. Understanding Database Sharding: https://lnkd.in/g57A8ytQ
6. Database Replication: https://lnkd.in/gHRAbJD3
7. SQL vs. NoSQL Database: When to Use, How to Choose https://lnkd.in/gvT5xx-W
8. How do database indexes work?: https://lnkd.in/gj8CJYnW
9. Redis Explained: https://lnkd.in/gfRM-9Us
10. Database Sharding Explained: https://lnkd.in/gZGHZvQR
11. Things you should know about databases: https://lnkd.in/gPGhe9iM

Great System Design Articles from the learnings at AWS to build really high scale, resilient and available systems.

Great System Design Articles from the learnings at AWS to build really high scale, resilient and available systems.

1. Dependency Isolation : https://lnkd.in/gFFPF9xZ
2. Using CI/CD : https://lnkd.in/g2rzWgSZ
3. Correlated Failures: https://lnkd.in/gh-avD-V
4. Reliability and Constant Work: https://lnkd.in/gTXbDHhM
5. Making retries safe with Idempotent APIs : https://lnkd.in/gKQAFMWR
6. Fairness In Multitenant Systems: https://lnkd.in/gJVh5P2R
7. Shuffle Sharding and Magical Fault Isolation: https://lnkd.in/gjzyc58c
8. Avoiding Overload in Distributed Systems: https://lnkd.in/gV7pik_d
9. Dashboards For Visibility: https://lnkd.in/gngJZW5e
10. Caching Challenges And Strategies: https://lnkd.in/gNPua7tM
11. Timeouts, Retries, Back Off with Jitter: https://lnkd.in/gqtpfJVU
12. Load Shedding: https://lnkd.in/gna9EFZv

DSA and System Design Guide : https://lnkd.in/g8W4NrP5
SRE Interview Prep Guide: https://lnkd.in/geVJgcjx
Newsletter- AI Toast: https://lnkd.in/gH2FQ2rK
Follow me on twitter for Byte Sized System Design Content: https://lnkd.in/gbuyTGYw

Post 3 on Papers to read to learn System Design 🎯

Post 3 on Papers to read to learn System Design 🎯

Consensus and replicated state machines

1. Paxos Made Simple (https://lnkd.in/gk6nxyVj)
2. Implementing Fault-Tolerant Services Using the State Machine (https://lnkd.in/gPwNde-i)
3. The Chubby lock service for loosely-coupled distributed systems (https://lnkd.in/gFXKTrXR)
4. ZooKeeper: Wait-free coordination for Internet-scale systems (https://lnkd.in/gWTYBxQN)
5. In Search of an Understandable Consensus Algorithm (https://lnkd.in/gqrKhvsK)
6. Virtual Consensus in Delos (https://lnkd.in/g5bitkdM)

Peertopeer systems and information dessimination

1. Gossip-Based Broadcast (https://lnkd.in/gT74Zb8Z)
Gossiping in Distributed Systems (https://lnkd.in/g55DFbuP)
2. Peer-to-peer membership management for gossip-based protocols (https://lnkd.in/g_XE4TiE)
3. Gossip-based Peer Sampling (https://lnkd.in/gSPwEkaW)
4. SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol (https://lnkd.in/gxZtR3Nh)
5. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems (https://lnkd.in/gyURBizm)
6. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications (https://lnkd.in/grVF9crk)

Post 2 of Papers for System Design on

Post 2 of Papers for System Design on hashtag#Streaming

🔖 MillWheel: Fault-Tolerant Stream Processing at Internet Scale (https://lnkd.in/gC7VjCfG)

🔖 The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing (https://lnkd.in/g-PyJUPa)

🔖 Apache Flink™: Stream and Batch Processing in a Single Engine (https://lnkd.in/gpzRA6v3)

🔖 Drizzle: Fast and Adaptable Stream Processing at Scale (https://lnkd.in/g9Hbnvp7)

🔖 Kafka, Samza and the Unix Philosophy of Distributed Data (https://lnkd.in/grtHkFWN)

🔖 Discretized Streams: Fault-Tolerant Streaming Computation at Scale (https://lnkd.in/gbzc3_Ke)

🔖 Structured Streaming: A Declarative API for Real-Time Applications in Apache Spark (https://lnkd.in/gnQQP2UY)

🔖 Noria: dynamic, partially-stateful data-flow for high-performance web applications (https://lnkd.in/gYtpef34)

Previous Post on hashtag#StorageSytems : https://lnkd.in/gDMRmkdw

First post today on #Storagesystems

Following are some amazing papers to learn about System Design in great depth.

I will be posting a series of categories and the papers associated with each.
First post today on #Storagesystems

1. Haystack (https://lnkd.in/gSZYcmmB)

2. f4: Facebook's Warm BLOB Storage System (https://lnkd.in/gMEfTpAh)

3. The Hadoop Distributed File System (https://lnkd.in/gSUqafDg)

4. The Google File System (https://lnkd.in/giUResea)

5. Facebook's Tectonic Filesystem: Efficiency from Exascale (https://lnkd.in/geg7-ub9)

6. Pelican: A Building Block for Exascale Cold Data Storage (https://lnkd.in/gSse26YK)

7. CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data (https://lnkd.in/gUbnK4rH)

8.RADOS: a scalable, reliable storage service for petabyte-scale storage (https://lnkd.in/gKwbmzTx)

9. Megastore: Providing Scalable, Highly Available Storage for Interactive Services (https://lnkd.in/gT7mSDQN)

10. The Design and Implementation of a Log-Structured File System (https://lnkd.in/gVuka_Ym)

11. The RAMCloud Storage System (https://lnkd.in/gC3SQccF)

12. Tao-Facebook-Distributed-Datastore (https://lnkd.in/g5BahTyz)

13. Spanner Google's-globally-distributed-database (https://lnkd.in/gGxdy_nw)

14. Bigtable A-Distributed-Storage-System-For-Structured-Data (https://lnkd.in/gyDtV96u)

15. RocksDb An in memory key value store (https://lnkd.in/gUhPQ-2d)

Follow Arpit Adlakha for more of these.

Tech/Papers you should read and know for Senior and Staff Engineering Roles !

🔷 Distributed Databases & Storage Systems
1. C-store: A column-oriented database
2. The Design and Implementation of Modern Column-Oriented Database Systems

3. Dynamo, Cassandra, GFS, HDFS
4. Vector Database: Challenges in Storage and Retrieval
5. MyRocks: Serving Facebook's Social Graph
6. Amazon DynamoDB: Fully Managed NoSQL Service
7. Windows Azure Storage, Building a Database on S3
8. Google Megastore & Spanner, SILT: High-performance key-value store

🔷 Consensus and Coordination
1. Chubby: Distributed Locks, Zookeeper
2. Paxos Made Simple, Raft Algorithm
3. State Machine Replication

🔷 Caching & In-Memory Databases
1. Memcached
2. Redis

🔷 Message Queues and Streaming
1. Kafka
2. RabbitMQ
3. Apache Spark
4. Apache Flink

🔷 Job Scheduling
1. Quartz: A Job Scheduling Library

🔷 Authentication & Authorization
1. Zanzibar: Google's Auth System

🔷 Cryptography & Blockchain
1.Bitcoin, SSL/TLS

🔷 Network & Protocols
1. Maglev, TCP, QUIC

🔷 Data Processing
1. MapReduce
2. Relational Database Model
3. SQLite

🔷 Ad-hoc
1. CAP Theorem
2. Scuba by Facebook
3. Google Colossus
4. Facebook Tectonic
5. Edge Networks by Cloudflare/Akamai

You will most of the papers https://lnkd.in/gkxJ8QAM here !

Follow Arpit Adlakha for more !

🚀 Understanding Index Storage in Databases 🚀

🚀 Understanding Index Storage in Databases 🚀
When it comes to database performance, index storage plays a crucial role in optimizing query execution and data retrieval. Let's take a closer look at how three popular databases handle index storage:

🔵 MongoDB:
MongoDB prioritizes index storage in memory, leveraging the available RAM to efficiently hold index data. This approach is pivotal for MongoDB's emphasis on fast query execution and data access. However, MongoDB also utilizes disk-based storage for indexes, especially when memory resources are constrained or needed for other operations. With the WiredTiger storage engine, MongoDB strikes a balance by utilizing a combination of in-memory and on-disk storage for indexes.

🔶 PostgreSQL:
Similarly, PostgreSQL primarily maintains indexes in memory to ensure optimal performance. By employing a shared buffer pool, PostgreSQL caches frequently accessed data pages, including index pages, enhancing query execution efficiency. PostgreSQL's versatile query planner further enhances performance by employing various index access methods, such as bitmap index scans and index-only scans. Despite its focus on in-memory storage, PostgreSQL also writes index updates to disk, ensuring durability and consistency, particularly in the face of system crashes.

🔷 MySQL:
MySQL adopts a hybrid approach to index storage, utilizing both memory and disk resources. With an in-memory key buffer, MySQL caches index blocks for frequently accessed tables, bolstering query performance. However, MySQL also persists index changes to disk through transaction logging and storage engine mechanisms. InnoDB, MySQL's default storage engine, manages indexes in a shared buffer pool in memory while simultaneously writing index changes to InnoDB data files on disk.

Understanding how MongoDB, PostgreSQL, and MySQL handle index storage is important for optimizing database performance and ensuring efficient query execution.
hashtag#DatabasePerformance hashtag#IndexStorage hashtag#MongoDB hashtag#PostgreSQL hashtag#MySQL

creating disaster recovery strategies for our systems and their implementation

For the last few months I have been involved in creating disaster recovery strategies for our systems and their implementation.

Here are some learnings from designing real systems !

1. Cost was always one of the most important discussion so you can't skip it in your system design interview.
2. The recovery strategy should always work when we need it, so it has to be reliable.

3. The solution should be ready within a week or two for one service, like MongoDb, Cassandra or Redis. It cannot take a long time because this is critical for us.
4. The technologies we used were terraform to create the infra using the backup EBS snapshot, Ansible for automation of installation/running processes and Jenkins to write jobs to run both terraform and ansible to recover in a single click.
5. Final outcome was jobs that can recover a completely deleted database in less than 10 minutes.

Real systems teach you tons of things. Execution speed with a balance of stability in the solution is really what we needed !

Follow Arpit Adlakha for more !

What is inside .git and how is a git repo really managed ?

What is inside .git and how is a git repo really managed ?

1. HEAD: .git/head
2. branch: .git/refs/heads/main
3. commit: .git/objects/10/93da429...
4. tree: .git/objects/9f/83ee7550...
5. blobs: .git/objects/5a/475762c...
6. reflog: .git/logs/refs/heads/main
7. remote-tracking branches: .git/refs/remotes/origin/main
8. tags: .git/refs/tags/v1.0
9. the stash: .git/refs/stash
10. .git/config
11. hooks: .git/hooks/pre-commit
12. the staging area: .git/index

You have to read this amazing blog by Julia Evans: https://lnkd.in/gHzCVfBq

The top 10 articles which will teach you tons about software engineering !

The top 10 articles which will teach you tons about software engineering !

1. Building and Operating S3 the biggest storage system: https://lnkd.in/g65anHkf

2. Modern Load Balancing and Networking by the creator of Envoy: https://lnkd.in/gn4BA3sD

3. The Log: What every software engineer should know about real-time data's unifying abstraction: https://lnkd.in/gQ7WsE8k

4. Keeping Netflix reliable using Load Shedding: https://lnkd.in/gSx8p32U

5. How Discord stores billions of messages: https://lnkd.in/gduE8Kgx

6. How Discord stores trillions of messages: https://lnkd.in/gRNwqnuH

7. Netflix Fault Tolerance in a high volume distributed system: https://lnkd.in/gGFensR4

8. How Pinterest runs Kafka at scale: https://lnkd.in/gb5skEtU

9. How browsers work: https://lnkd.in/gumG6WQw

10. Practical Guide to Kafka Storage Internals: https://lnkd.in/gA9XJteP

Follow Arpit Adlakha for more. Next up top 10 videos

Patterns of Distributed Systems for your System Design Interview 🖥️ 🚀

1. Clock-Bound Wait (https://lnkd.in/gHjr74cb)
2. Consistent Core (https://lnkd.in/g4XcrJUD)
3. Emergent Leader (https://lnkd.in/gZdf_AWx)
4. Fixed Partitions (https://lnkd.in/gSJuYfCc)
5. Follower Reads (https://lnkd.in/gmUw5b7a)
6. Generation Clock (https://lnkd.in/g92gJSHS)
7. Gossip Dissemination (https://lnkd.in/gBdD77Fc)
8. HeartBeat (https://lnkd.in/g6RR9swi)
9. High-Water Mark (https://lnkd.in/g3Kx3gux)
10. Hybrid Clock (https://lnkd.in/gjPiB_GM)
11. Idempotent Receiver (https://lnkd.in/gVnNEAS9)
12. Key-Range Partitions (https://lnkd.in/g9k_W2DN)
13. Lamport Clock (https://lnkd.in/gn_Mjn5h)
14. Leader and Followers (https://lnkd.in/guv739mY)
15. Lease (https://lnkd.in/gX9cpWYA)
16. Low-Water Mark (https://lnkd.in/ggEnd765)
17. Paxos (https://lnkd.in/gzXpcqzG)
18. Quorum (https://lnkd.in/g_zM69XD)
19. Replicated Log (https://lnkd.in/g8dh64wZ)
20. Request Batch (https://lnkd.in/gutqpHJB)
21. Request Pipeline (https://lnkd.in/gVXknRem)
22. Request Waiting List (https://lnkd.in/ghB6HfdH)
23. Segmented Log (https://lnkd.in/gzvS48Za)
24. Single Socket Channel (https://lnkd.in/gdJapnNc)
25. Singular Update Queue (https://lnkd.in/gAnmUBvp)
26. State Watch (https://lnkd.in/g8EuZTPG)
27. Two Phase Commit (https://lnkd.in/g3Btwu3Z)

learn DNS by doing it

If you ever wanted to learn DNS by doing it , this website https://messwithdns.net/ teaches you all about dns by giving practical exercises.

Some sample experiments there are in the image below.