System Design Interesting Reads

Basics:

[System Design Cheatsheet · GitHub](https://gist.github.com/vasanthk/485d1c25737e8e72759f)

[System Design Primer](https://github.com/donnemartin/system-design-primer)

[How to Succeed in a System Design Interview](https://blog.pramp.com/how-to-succeed-in-a-system-design-interview-27b35de0df26)

[Approach a System Design Interview](https://medium.com/system-designing-interviews/approach-a-system-design-interview-f3594e243730)

[System Design, Chapter 2: Sharding](https://medium.com/system-designing-interviews/system-design-chapter-2-sharding-484960c18f6)

[System Design, Chapter 3: Load Balancing](https://medium.com/system-designing-interviews/system-design-chapter-3-load-balancing-e1c89148e37)

[System Design, Chapter 4: Caching](https://medium.com/system-designing-interviews/system-design-chapter-4-caching-b59a4cf83f10)

[System Design, Chapter 5: Indexes in Databases](https://medium.com/@nishantnitb/system-design-chapter-5-indexes-in-databases-beb90295dbcf)

[System Design, Chapter 6: Proxies](https://medium.com/system-designing-interviews/system-design-chapter-6-proxies-f77be8858023)

[System Design, Chapter 7: Queues](https://medium.com/system-designing-interviews/system-design-chapter-7-queues-5f3f9bed81)

Milestone Research Papers:

[The Google File System](https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf)

[Dynamo: Amazon's Highly Available Key-value Store](https://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf)

[MapReduce: Simplified Data Processing on Large Clusters](https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf)

[TAO: Facebook's Distributed Data Store for the Social Graph](https://www.usenix.org/system/files/conference/atc13/atc13-bronson.pdf)

[Kafka: a Distributed Messaging System for Log Processing](http://notes.stephenholiday.com/Kafka.pdf)

[Bigtable: A Distributed Storage System for Structured Data](https://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf)

[Spark: Cluster Computing with Working Sets](https://www.usenix.org/legacy/event/hotcloud10/tech/full_papers/Zaharia.pdf)

Popular Datastores:

[Choose the right data store - Azure Application Architecture Guide](https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/data-store-overview)

[Redis vs Aerospike](https://lynnlangit.com/2015/01/28/lessons-learned-benchmarking-nosql-on-the-aws-cloud-aerospikedb-and-redis/)

[Elasticsearch vs MongoDB](https://medium.com/@ranjeetvimal/elasticsearch-vs-mongodb-631f410cd317#:~:text=MongoDB%20is%20a%20general%20purpose,search%20engine%20backed%20by%20Lucene.&text=In%20practice%2C%20ElasticSearch%20is%20often,queries%2C%20based%20on%20data%20content.)

[MongoDB vs CouchDB](https://blog.scottlogic.com/2014/08/04/mongodb-vs-couchdb.html#:~:text=Both%20scale%20across%20multiple%20nodes,defined%20as%20a%20replica%20set.&text=CouchDB%20uses%20a%20replication%20model%20called%20Eventual%20Consistency.)

*Cassandra*

[Cassandra writes in depth. Surprises you can expect from a… | by Andrzej Ludwikowski](https://blog.softwaremill.com/cassandra-writes-in-depth-6ea8d7581eb)

[Cassandra Data Modelling](https://cassandra.apache.org/doc/latest/data_modeling/data_modeling_logical.html)

[Why Cassandra writes faster than Traditional RDBMS?](https://www.linkedin.com/pulse/why-cassandra-writes-faster-than-traditional-rdbms-vishal-kharjul/)

[When to use Cassandra and when to steer clear](https://towardsdatascience.com/when-to-use-cassandra-and-when-to-steer-clear-72b7f2cede76)

[Log Structured Merge Trees](http://www.benstopford.com/2015/02/14/log-structured-merge-trees/)

[How is data read? | Apache Cassandra 3.0](https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlAboutReads.html)

*Redis*

[Redis Persistence – Redis](https://redis.io/topics/persistence)

[Goodbye Cache: Redis as a Primary Database](https://redislabs.com/blog/goodbye-cache-redis-as-a-primary-database/)

[Transactions in Redis](https://redis.io/topics/transactions)

[Partitioning: how to split data among multiple Redis instances. – Redis](https://redis.io/topics/partitioning)

[Horizontal scaling in/out techniques for redis cluster](https://medium.com/@iamvishalkhare/horizontal-scaling-in-out-techniques-for-redis-cluster-dcd75c696c86)

[Redis Cluster](https://redis.io/topics/cluster-tutorial)

[Key Problems with Redis Persistence](http://oldblog.antirez.com/post/a-few-key-problems-in-redis-persistence.html#:~:text=The%20problem%20is%20that%20we,memory%20at%20a%20given%20time.)

*Elasticsearch*

[Elasticsearch from the Top Down](https://www.elastic.co/blog/found-elasticsearch-top-down)

[Uses of Elasticsearch, and Things to Learn](https://www.elastic.co/blog/found-uses-of-elasticsearch)

[Elasticsearch from the Bottom Up, Part 1](https://www.elastic.co/blog/found-elasticsearch-from-the-bottom-up)

*PostgreSql*

[Comparing Data Stores for PostgreSQL - MVCC vs InnoDB](https://severalnines.com/database-blog/comparing-data-stores-postgresql-mvcc-vs-innodb)

[Postgres MVCC](https://malisper.me/postgres-mvcc/)

[PostgreSQL 10: Partitions of partitions!](https://joaodlf.com/postgresql-10-partitions-of-partitions.html)

*Mongo*

[How To Decide If MongoDB Is Right For You](https://medium.com/better-programming/mongodb-insights-20e36c8f2fcd)

Interesting Blogs:

[All Company Blogs !!](https://github.com/kilimchoi/engineering-blogs)

[Building Reliable Reprocessing and Dead Letter Queues with Kafka](https://eng.uber.com/reliable-reprocessing/)

[A Practical Introduction to the Internals of Kafka Storage](https://medium.com/@durgaswaroop/a-practical-introduction-to-kafka-storage-internals-d5b544f6925f)

[How To Design A Scalable Rate Limiting Algorithm](https://konghq.com/blog/how-to-design-a-scalable-rate-limiting-algorithm/)

[Data Compression for Large-Scale Streaming Experimentation | by Netflix Technology Blog](https://netflixtechblog.com/data-compression-for-large-scale-streaming-experimentation-c20bfab8b9ce)

[WhatsApp-Engineering Inside-1. Real Time messaging are now an…](https://medium.com/codingurukul/whatsapp-engineering-inside-1-1ef4845ff784)

[WhatsApp-Engineering Inside-2. In “WhatsApp-Engineering Inside-1” we…](https://medium.com/codingurukul/whatsapp-engineering-inside-2-bdd1ec354748)

[NETFLIX system design. System Design](https://medium.com/@narengowda/netflix-system-design-dbec30fede8d)

[A Design Analysis of Cloud-based Microservices Architecture at Netflix](https://medium.com/swlh/a-design-analysis-of-cloud-based-microservices-architecture-at-netflix-98836b2da45f)  
[Microservice Architecture pattern](https://microservices.io/patterns/microservices.html)

[Resiliency: Cache Me If You Can](https://medium.com/@adhorn/patterns-for-resilient-architecture-part-4-85afa66d6341#:~:text=Request%20coalescing&text=These%20situations%20often%20occur%20during,request%20to%20the%20downstream%20storage.)

[Microservices Design Patterns](https://vslive.com/Blogs/News-and-Tips/2018/02/Go-Fast-by-Going-Micro-Microservices-Design-Patterns-You-Should-Know.aspx#:~:text=Microservices%20design%20patterns%20are%20software,individual%20microservice%20independently%2C%20if%20desired.)

[Kafka Message Delivery Semantics](https://kafka.apache.org/documentation/#semantics)

[Replication In Depth - Kafka](https://aphyr.com/posts/293-jepsen-kafka)

[Design Decisions for Scaling Your High Traffic Feeds](http://highscalability.com/blog/2013/10/28/design-decisions-for-scaling-your-high-traffic-feeds.html)

[The Architecture Twitter Uses to Deal with 150M Active Users, 300K QPS, a 22 MB/S Firehose, and Send Tweets in Under 5 Seconds](http://highscalability.com/blog/2013/7/8/the-architecture-twitter-uses-to-deal-with-150m-active-users.html)

[Instagram Saves Switching to Cassandra from Redis](https://www.datastax.com/blog/2014/10/facebooks-instagram-making-switch-cassandra-redis-75-insta-savings)

[Jepsen Blog](https://aphyr.com/posts)

[Aerospike Developer Blog – Medium](https://medium.com/aerospike-developer-blog)

[Patterns for distributed transactions within a microservices architecture](https://developers.redhat.com/blog/2018/10/01/patterns-for-distributed-transactions-within-a-microservices-architecture/)

[A Guide to Atomikos](https://www.baeldung.com/java-atomikos)

[How to do distributed locking](https://martin.kleppmann.com/2016/02/08/how-to-do-distributed-locking.html)

[Taming Garbage Collection](https://product.hubspot.com/blog/g1gc-fundamentals-lessons-from-taming-garbage-collection#HumongousObjects)

Real World Videos:

[Jeff Dean's talk at Stanford](https://www.youtube.com/watch?v=modXC5IWTJI)

[Building Billion user Load Balancer at Facebook](https://www.youtube.com/watch?v=bxhYNfFeVF4)

[Netflix Guide to Microservices](https://www.youtube.com/watch?v=CZ3wIuvmHeM)

[Amazon DynamoDB deep dive](https://www.youtube.com/watch?v=HaEPXoXVf2k)

[Twitter: Timelines at Scale](https://www.infoq.com/presentations/Twitter-Timeline-Scalability/)

Advanced Techniques:

[Bloom Filter](https://medium.com/system-design-blog/bloom-filter-a-probabilistic-data-structure-12e4e5cf0638#:~:text=Bloom%20filter%20is%20a%20probabilistic,of%20applications%20in%20software%20engineering.&text=It%20allows%20for%20membership%20lookups%20in%20constant%20space%20%26%20time.)

[Cache Stampede](https://medium.com/@vaibhav0109/cache-stampede-problem-5eba782a1a4f)

[Request Coalescing](https://wikitech.wikimedia.org/wiki/Varnish#:~:text=In%20case%20a%20request%20from,object%20from%20an%20origin%20server.&text=Once%20the%20response%20is%20fetched,feature%20is%20called%20request%20coalescing.)

[Three Tier Caching](https://blogs.oracle.com/oswald/cache,-cache,-cache-part-2:-architectures)

[Consistent Hashing](https://medium.com/system-design-blog/consistent-hashing-b9134c8a9062)

Actual Questions:

*Easy*

[Design Chess](https://medium.com/system-designing-interviews/design-a-chess-game-dddd7ba11bc0)

[Design Tic-Tac-Toe](https://medium.com/system-designing-interviews/design-tic-tac-toe-game-1b912bba64cf)

[Design Elevator System](https://medium.com/system-designing-interviews/design-a-elevator-system-fc5832ca0b8b)

*Medium*

[Design Uber](https://medium.com/@narengowda/uber-system-design-8b2bc95e2cfe)

[Design Instagram](https://medium.com/@dingdingsherrywang/system-design-instagram-4658eeb0423a)

[Design Facebook Messenger](https://medium.com/@eileen.code4fun/design-facebook-messenger-438d76639985)

[Design URL Shortener](https://medium.com/@narengowda/url-shortener-system-design-3db520939a1c)

[Design Dropbox / Google Drive](https://medium.com/@narengowda/system-design-dropbox-or-google-drive-8fd5da0ce55b)

[Design News Feed System / Quora](https://medium.com/@bansal_ankur/design-a-news-feed-system-6bf42e9f03fb)

[Design Price Surging Service](https://medium.com/software-system-design/price-surge-during-online-hotel-booking-ff060c33b2b6)

[Design AirBnb](https://medium.com/@nrkapri/system-design-interview-question-design-airbnb-11ac00a1d9b0)

*Hard*

[Design YouTube / Netflix](https://medium.com/@eileen.code4fun/system-design-interview-mini-youtube-5cae5eedceae)

[Design Web Crawler](https://medium.com/@morefree7/design-a-distributed-web-crawler-f67a8ebb8336)