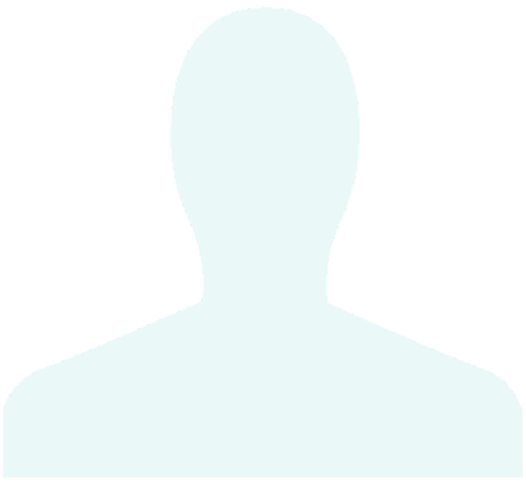
Grokking the System Design Interview

Search

- System Design Basics
  - Why System Design Interviews?
  - System Design Basics
  - Load Balancing
  - Caching
  - Sharding or Data Partitioning
  - Indexes
  - Proxies
  - Queues
  - Redundancy and Replication
  - SQL vs. NoSQL
  - CAP Theorem
  - Consistent Hashing
  - Long-Polling vs WebSockets vs Server-Sent Events (*New*)
- System Design Problems
  - System Design Interviews: A step by step guide
  - Designing a URL Shortening service like TinyURL
  - Designing Pastebin
  - Designing Instagram
  - Designing Dropbox
  - Designing Facebook Messenger
  - Designing Twitter
  - Designing Youtube or Netflix
  - Designing Typeahead Suggestion
  - Designing an API Rate Limiter (*New*)
  - Designing Twitter Search
  - Designing a Web Crawler
  - Designing Facebook's Newsfeed
  - Designing Yelp or Nearby Friends
  - Designing Uber backend
  - Design BookMyShow (*New*)
- Contact Us
  - Feedback

LearnTeach



- My Profile
  - View
  - Edit
- Logout

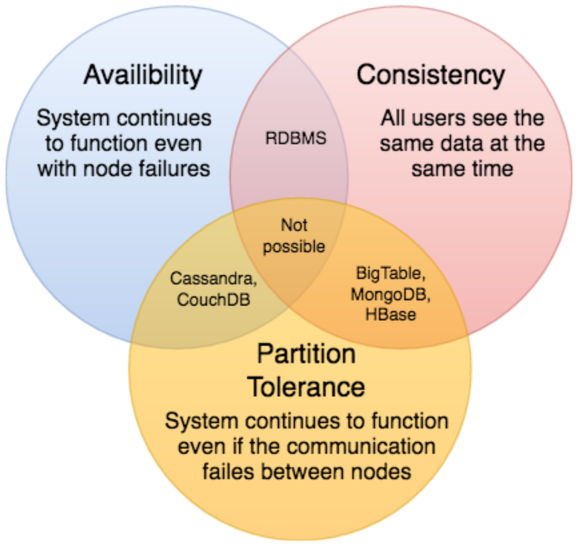
CAP Theorem

CAP theorem states that it is impossible for a distributed software system to simultaneously provide more than two out of three of the following guarantees (CAP): Consistency, Availability and Partition tolerance. When we design a distributed system, trading off among CAP is almost the first thing we want to consider. CAP theorem says while designing a distributed system we can pick only two of:

Consistency: All nodes see the same data at the same time. Consistency is achieved by updating several nodes before allowing further reads.

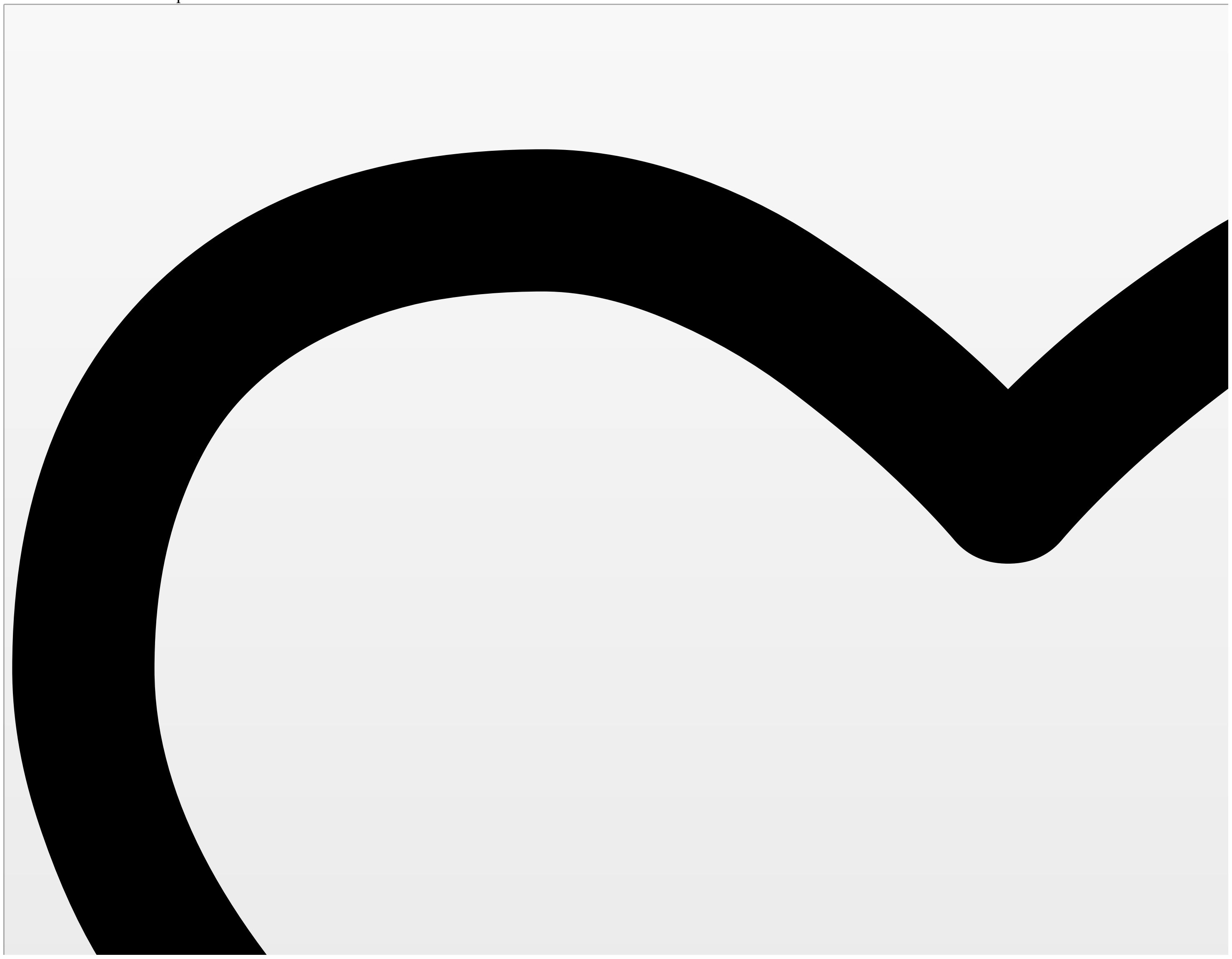
Availability: Every request gets a response on success/failure. Availability is achieved by replicating the data across different servers.

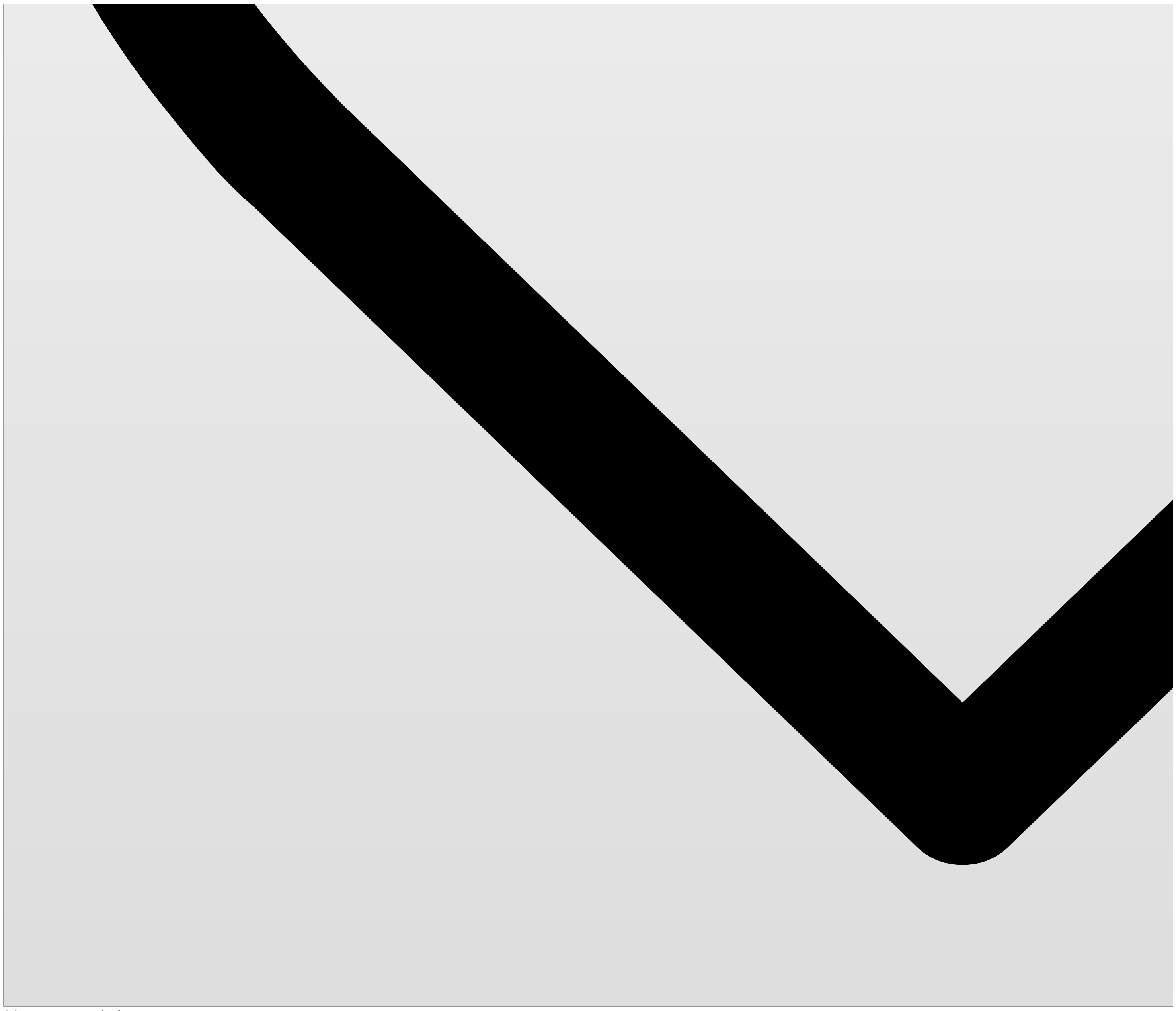
Partition tolerance: System continues to work despite message loss or partial failure. A system that is partition-tolerant can sustain any amount of network failure that doesn't result in a failure of the entire network. Data is sufficiently replicated across combinations of nodes and networks to keep the system up through intermittent outages.


Availibility
System continues to function even with node failures
Consistency
All users see the same data at the same time
RDBMS
Not possible
Cassandra, CouchDB
BigTable, MongoDB, HBase
Partition Tolerance
System continues to function even if the communication failes between nodes

We cannot build a general data store that is continually available, sequentially consistent and tolerant to any partition failures. We can only build a system that has any two of these three properties. Because, to be consistent, all nodes should see the same set of updates in the same order. But if the network suffers a partition, updates in one partition might not make it to the other partitions before a client reads from the out-of-date partition after having read from the up-to-date one. The only thing that can be done to cope with this possibility is to stop serving requests from the out-of-date partition, but then the service is no longer 100% available.

Mark as completed
← *Previous*SQL vs. NoSQL*Next* →Consistent Hashing
Send feedback or ask a question





23 recommendations

- Home
- Featured
- Team
- Blog
- FAQ
- Terms of Service
- Contact Us