(1) FEATURE EXPECTATIONS [5 min]

```
(1) Use cases
(2) Scenarios that will not be covered
(3) Who will use
(4) How many will use
(5) Usage patterns
```

(2) ESTIMATIONS [5 min]

```
(1) Throughput (QPS for read and write queries)
(2) Latency expected from the system (for read and write queries)
(3) Read/Write ratio
(4) Traffic estimates
        - Write (QPS, Volume of data)
        - Read  (QPS, Volume of data)
(5) Storage estimates
(6) Memory estimates
        - If we are using a cache, what is the kind of data we want to store in cache
        - How much RAM and how many machines do we need for us to achieve this ?
        - Amount of data you want to store in disk/ssd
```

(3) DESIGN GOALS [5 min]

```
(1) Latency and Throughput requirements
(2) Consistency vs Availability  [Weak/strong/eventual => consistency | Failover/replication => availabilit
```

(4) HIGH LEVEL DESIGN [5-10 min]

```
(1) APIs for Read/Write scenarios for crucial components
(2) Database schema
(3) Basic algorithm
(4) High level design for Read/Write scenario
```

(5) DEEP DIVE [15-20 min]

```
(1) Scaling the algorithm
(2) Scaling individual components:
        -> Availability, Consistency and Scale story for each component
        -> Consistency and availability patterns
(3) Think about the following components, how they would fit in and how it would help
        a) DNS
        b) CDN [Push vs Pull]
        c) Load Balancers [Active-Passive, Active-Active, Layer 4, Layer 7]
        d) Reverse Proxy
        e) Application layer scaling [Microservices, Service Discovery]
        f) DB [RDBMS, NoSQL]
                > RDBMS
                    >> Master-slave, Master-master, Federation, Sharding, Denormalization, SQL Tuning
                > NoSQL
                    >> Key-Value, Wide-Column, Graph, Document
                        Fast-lookups:
                        -------------
                            >>> RAM  [Bounded size] => Redis, Memcached
                            >>> AP [Unbounded size] => Cassandra, RIAK, Voldemort
                            >>> CP [Unbounded size] => HBase, MongoDB, Couchbase, DynamoDB
        g) Caches
                > Client caching, CDN caching, Webserver caching, Database caching, Application caching, Ca
                > Eviction policies:
                        >> Cache aside
                        >> Write through
                        >> Write behind
                        >> Refresh ahead
        h) Asynchronism
```

```
                        > Message queues
                        > Task queues
                        > Back pressure
                ii) Communication
                        > TCP
                        > UDP
                        > REST
                        > RPC
```

(6) JUSTIFY [5 min]

```
    (1) Throughput of each layer
    (2) Latency caused between each layer
    (3) Overall latency justification
```

Flash Cola

Sep 05, 2019

Great summary. Correction on caching part (3g). Cache aside, Write through, Write behind, Refresh ahead are not eviction techniques but caching patterns. Eviction techniques are LRU, LFU, FIFO etc.

208 Show 1 Replies Reply

vkalanji

Oct 14, 2019

@topcat Thank you very much for sharing this.
I would request that you take one or more sample use case/cases (Ex: Design a K-V Store) and explain how the template mentioned above applies to that use case/cases. That way we ll have both the template and an example.
Other leetcoders please upvote this if you think that this is a good idea.

127 Show 1 Replies Reply

ProblemSolver

Oct 16, 2022

@topcat You can add a few more points. These are to have a clear view of the system. Especially in a distributed system, these things are essential.

```
(7) Key metrics to measure
        For example, in the case of designing a search system, we will be interested to see how many keywords are
We can use tools like Graphana with Prometheus, AppDynamics, etc.
(8) System health monitoring
    Measuring app index, the latency of microservices
    We can use Newrelic, AppDynamics
(9) Log systems
    We can use ELK(Elastic, Logstash, Kibana) or Logtail, etc.
```

10 Reply

what's in a name

Aug 14, 2022

May I add 2 more sections in the deep dive:

1. Security aspect.
2. Telemetry & Logging aspect for quick mitigation of oncall issues.

9 Reply
