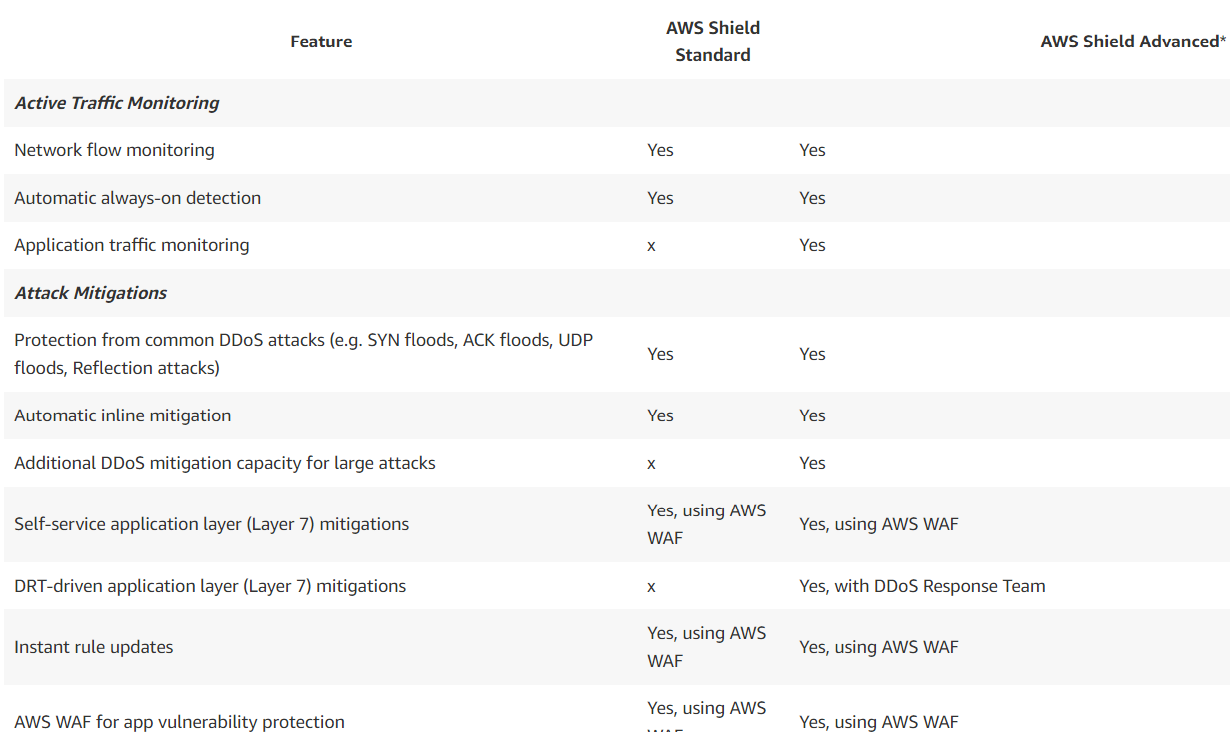
<https://www.geeksforgeeks.org/top-10-system-design-interview-questions-and-answers/>

<https://www.freecodecamp.org/news/systems-design-for-interviews/>

Requirement Clarification.

1. This feature will support only application traffic monitoring?
2. What would be the criteria for allowed request, request drop or request rejection policies?
3. Scope of rate limiting. Meaning only for data access api or any future service will leverage this rate limiting features?
4. Will rate limit features support distributed request?
5. Client identifications based on what? Meaning ip, location, tenant id, token etc.
6. To resolving rate limiting features we will go ahead use OOTB tools(Zull, NGINX etc.) or we will write it our own based on requirement?
7. Any idea on rate limiting parameters for allowing request, per minutes, second, hour etc.
8. 1st Rate limit will intercept request or auth api? Meaning Public URL🡪Rate Limit🡪Auth API🡪Resouce URL.
9. Will this rate limiting feature will support for distributed environment?
10. Any specific algorithm we are looking for to resolve this requirment. (Optional).
11. Can we know how many request PROD reporting module is getting daily, hourly, minutes & seconds.
12. Incase rate limiting feature support extended services then what would be the request drop or rejection policies?





<https://aws.amazon.com/shield/getting-started/>

1. Drop or Rejection policy?

Kind of Rate Limiting?

1. Tenant/User based rate limiting?
2. Concurrent Rate limiting (For a given user how many parallel connection and parallel session can be allowed to mitigate the DDOS attack?
3. Location and IP based rate limiting?
4. Token based rate limiting.

**Resolving Rate Limiting Algorithms.**

Approach 1.

1. Fetch Token.
2. Update Token.

Approach 2.

Leaky Bucket (FIFO).

E can server N request at Max where N is the Queue size.

Approach 3.

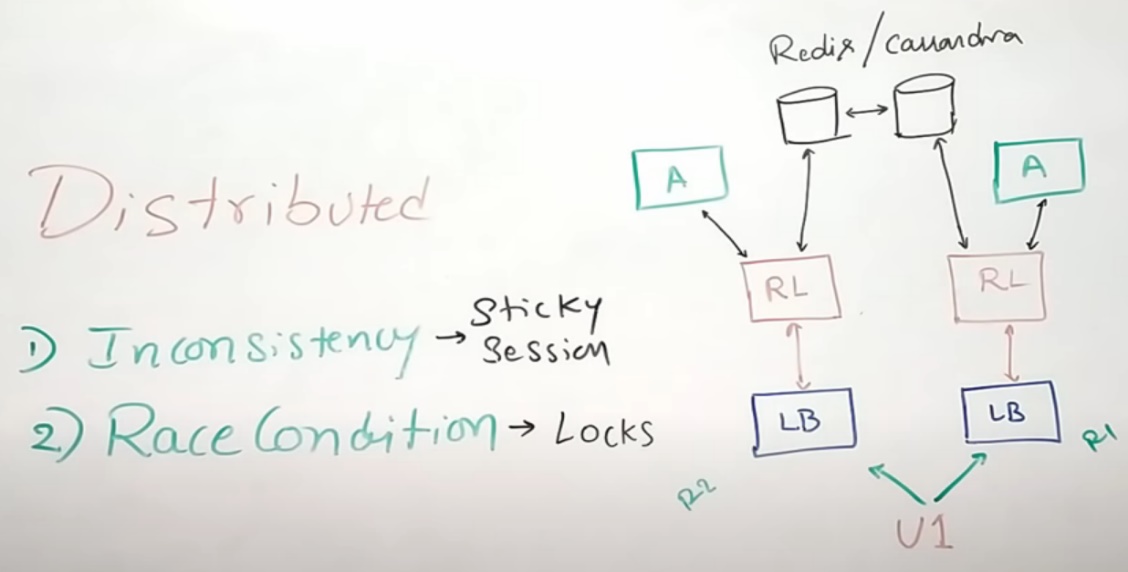
Fixed window counter (10(window size) request per minutes). In window size exceeds then drop the request.

Approach 4:

Sliding window log. (consumes more memory)

Approach 5.

Sliding window Counter algo.

Design for Distributed application:

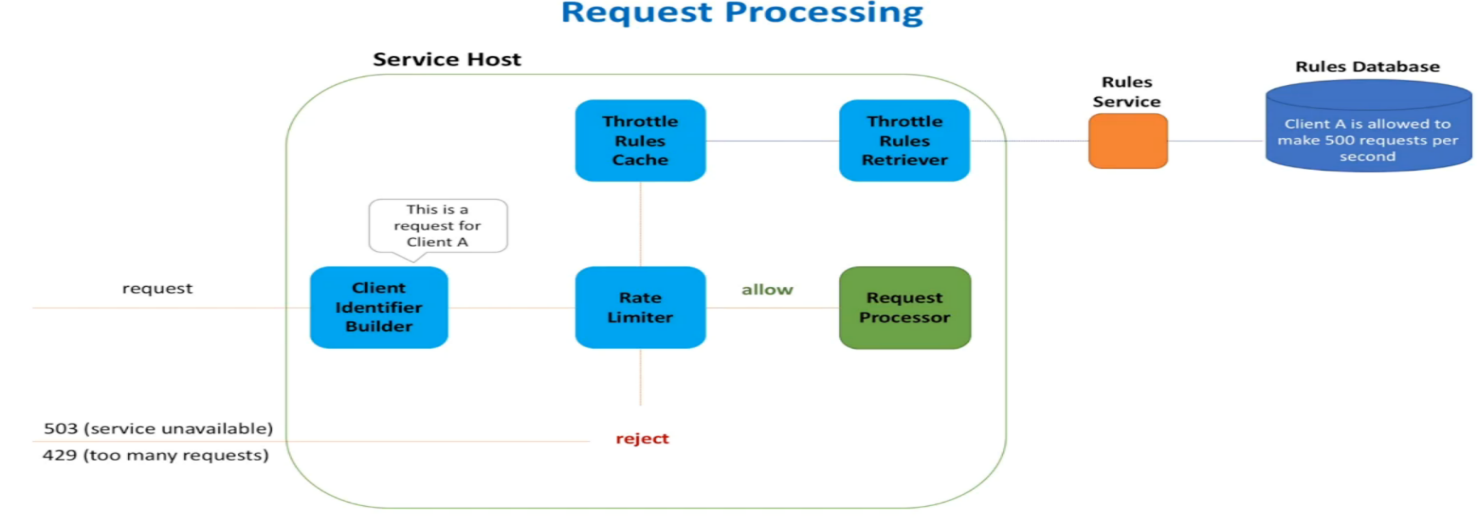
Consistency Rate Limit:

1. Sticky Session
2. Locks – ex. Optimistic or pessimistic

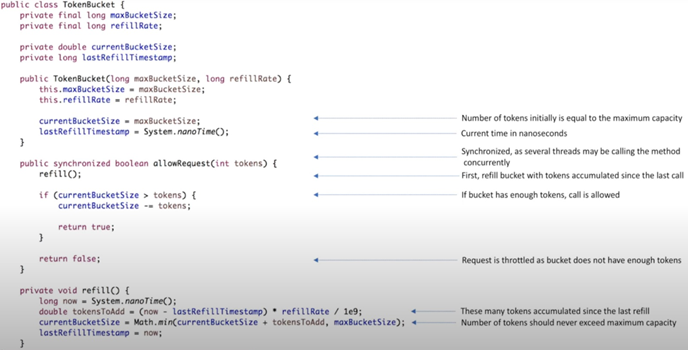
Relaxing Rate Limiting.

Local Memory + sync service

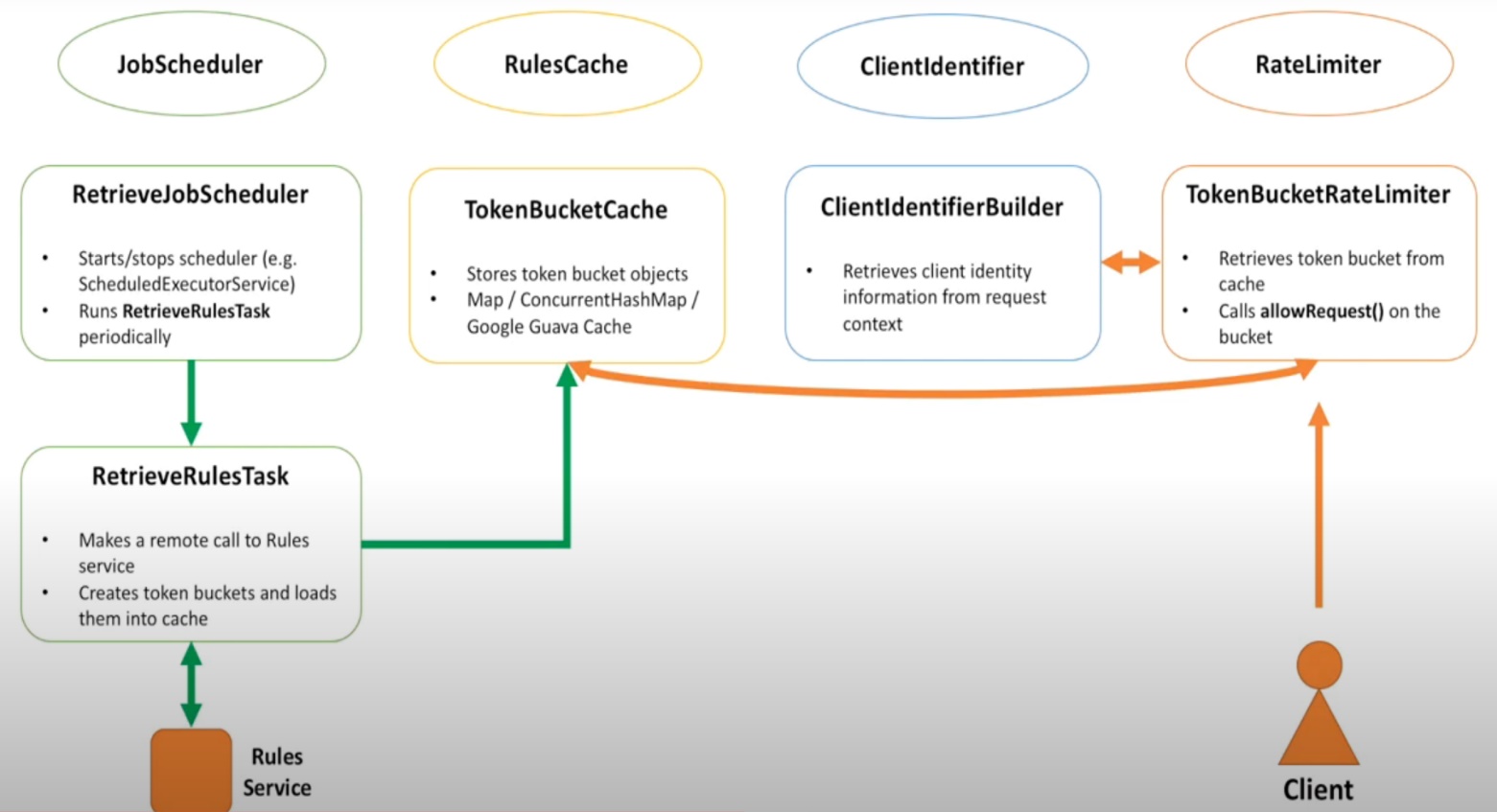
**High Level Request Flow**



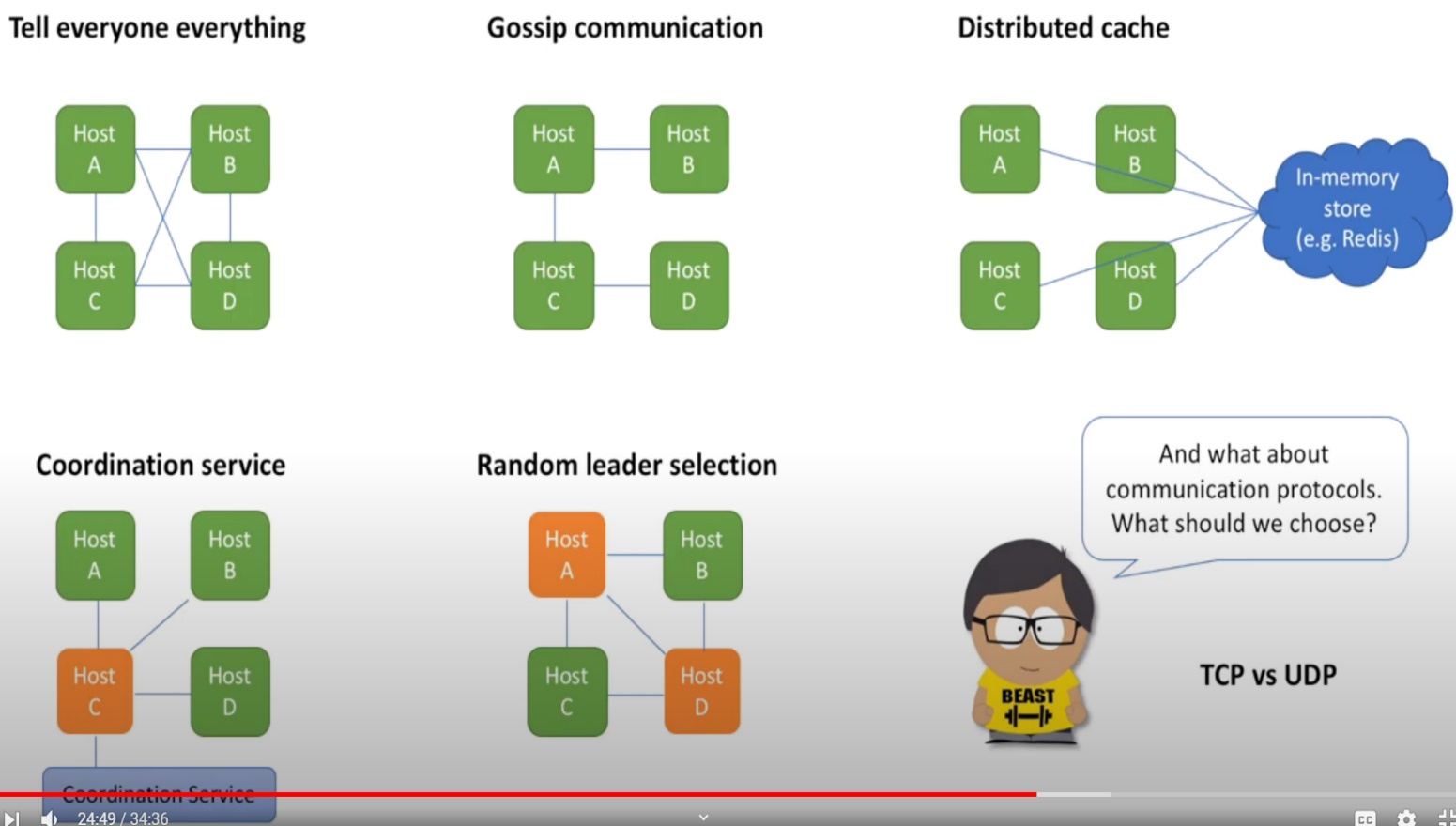
**Token Bucket Algorithm**

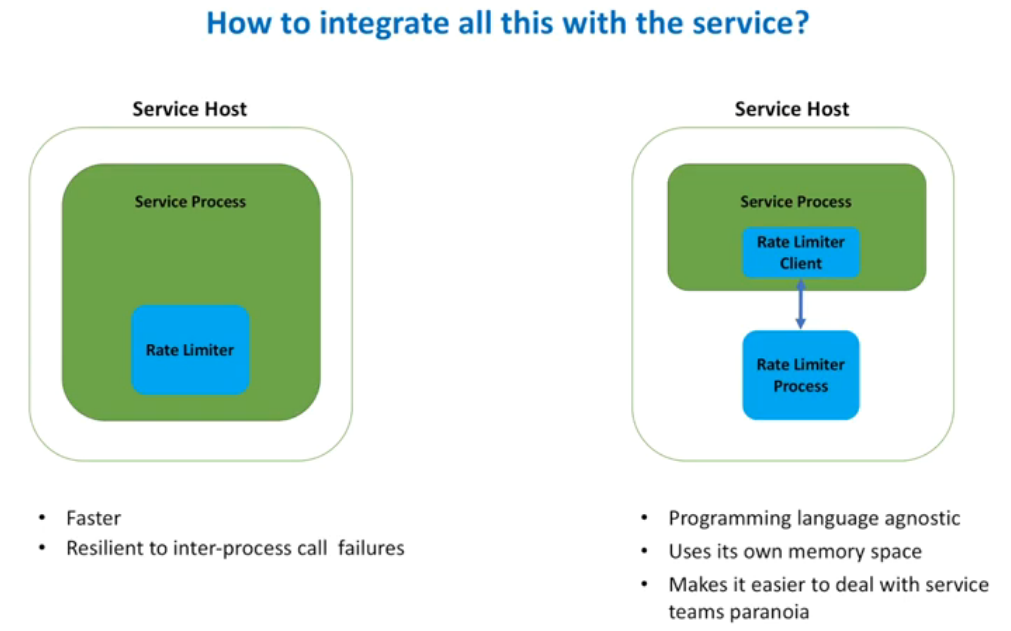


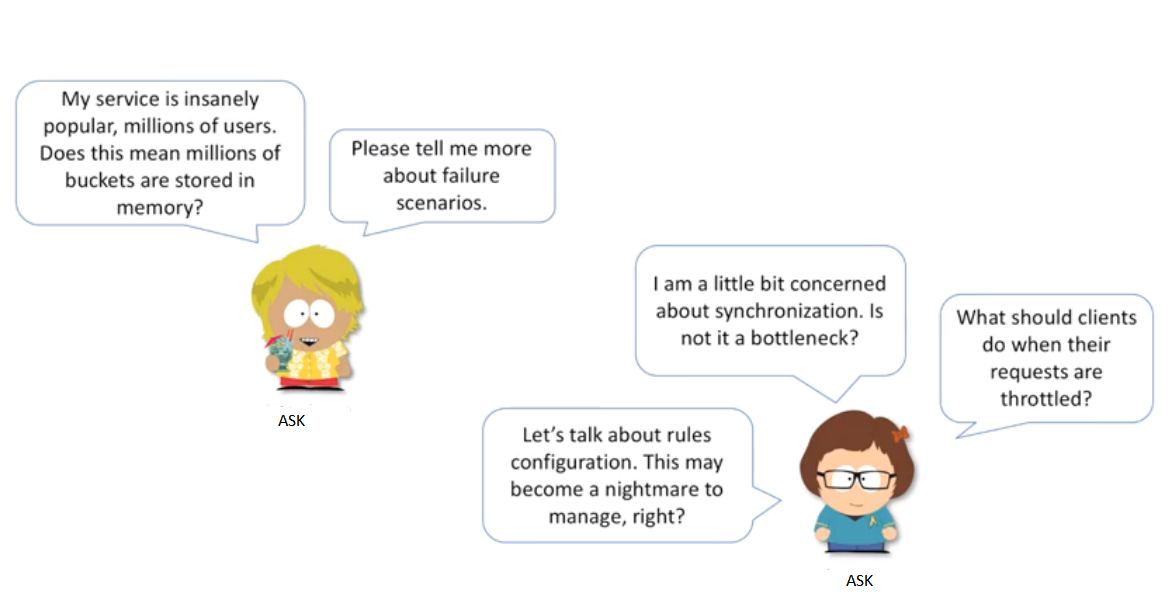
**Interfaces & Classes**



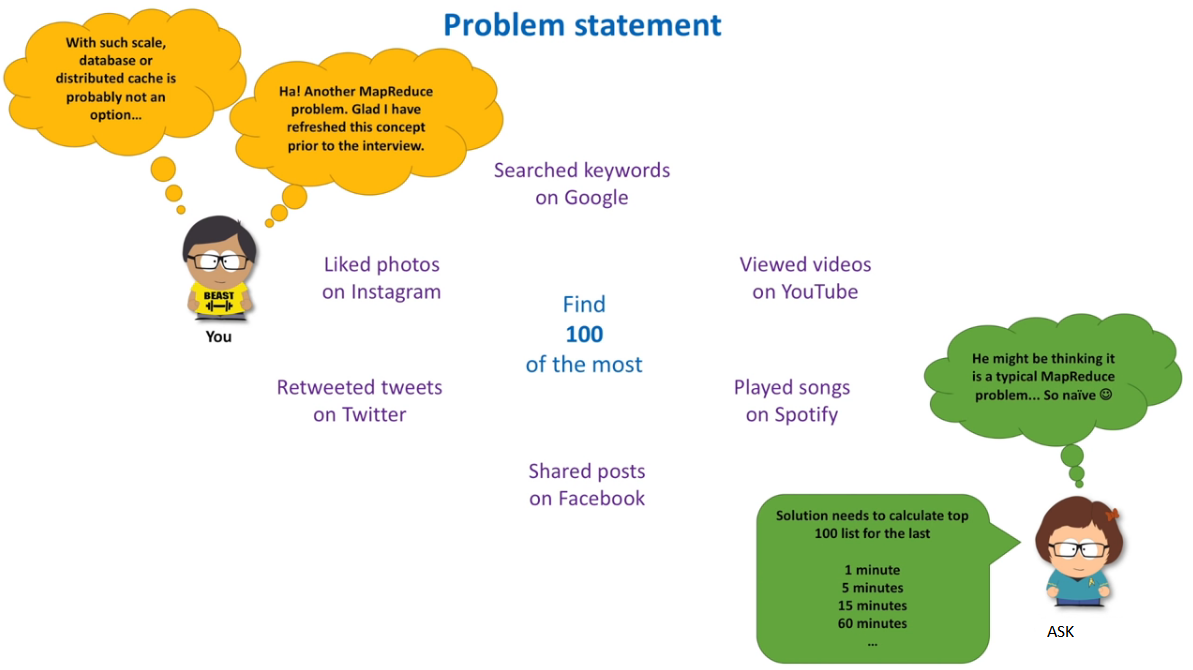
**Message Broad Casting – Distributed Communication**

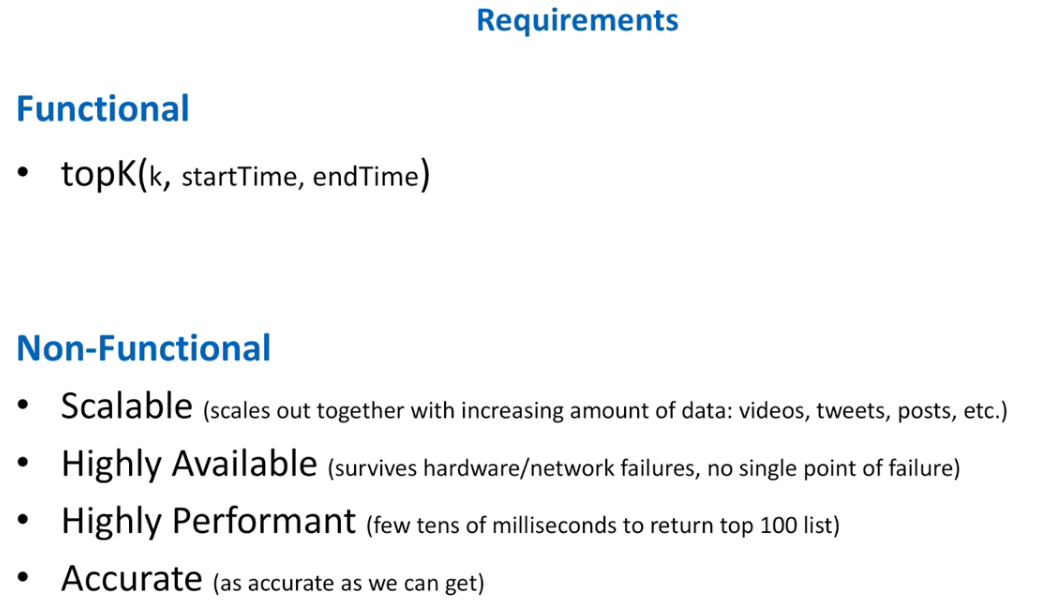
****

****

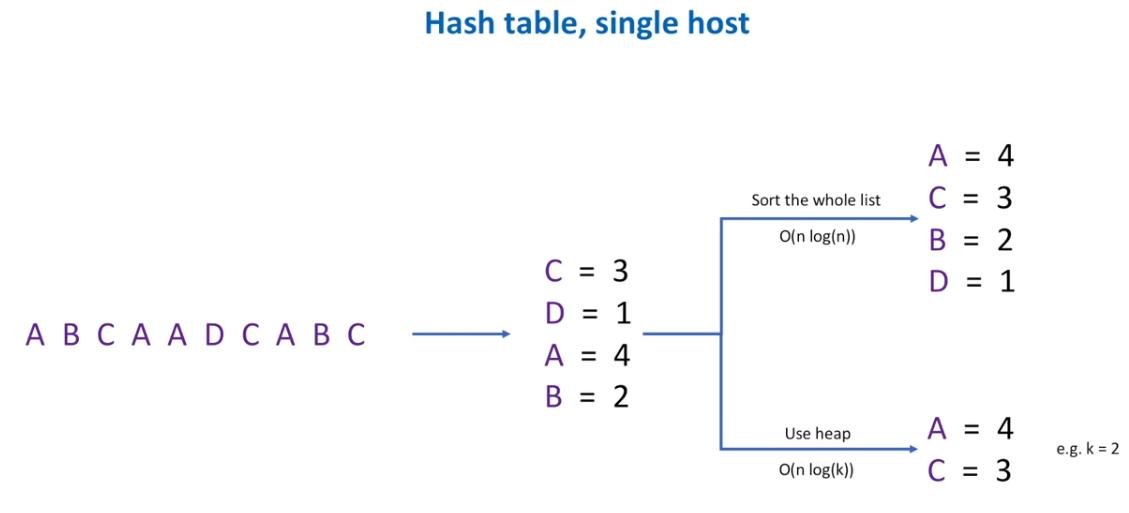
****

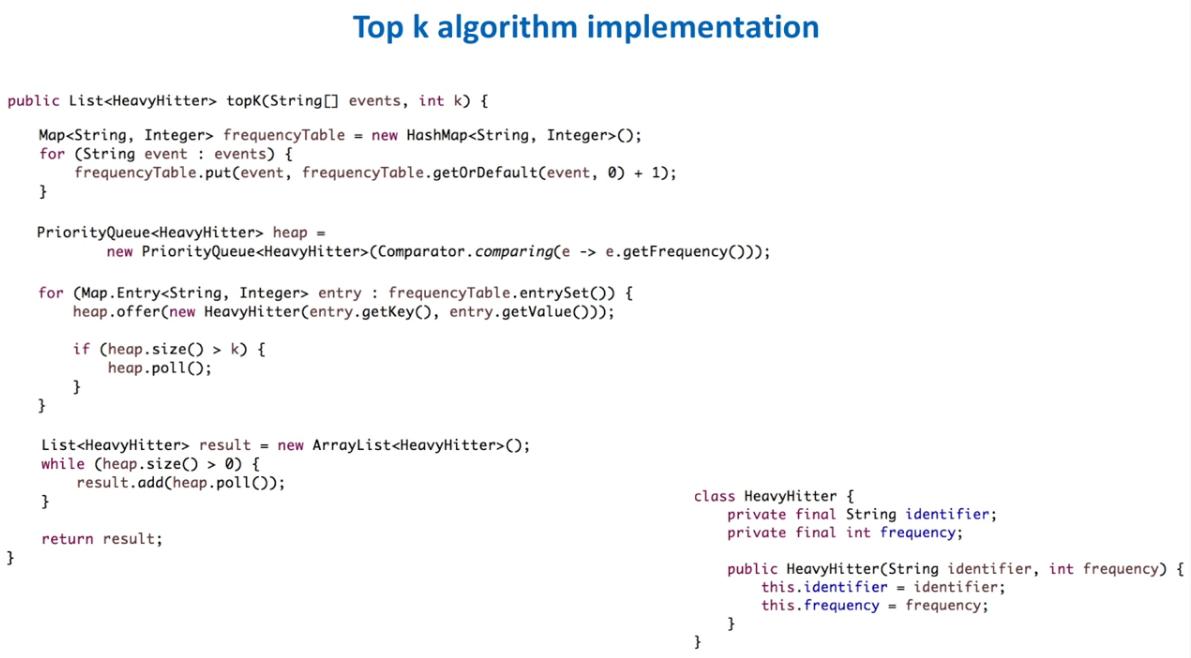
**Top K Problem (Heavy Hitters)**

****

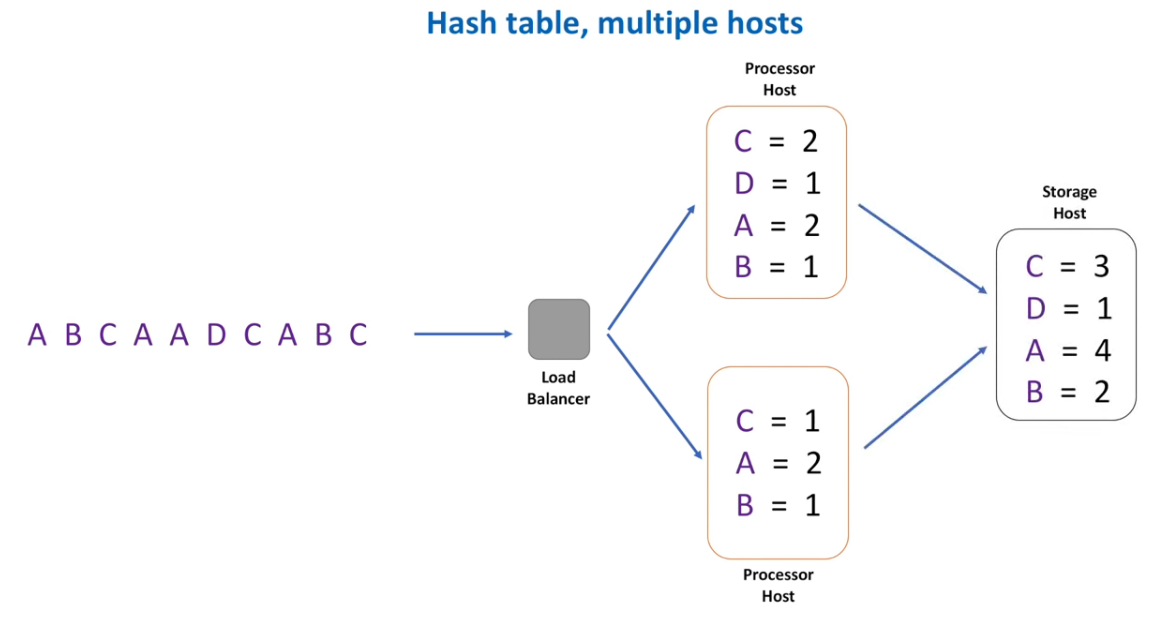
****

**Approach 1**

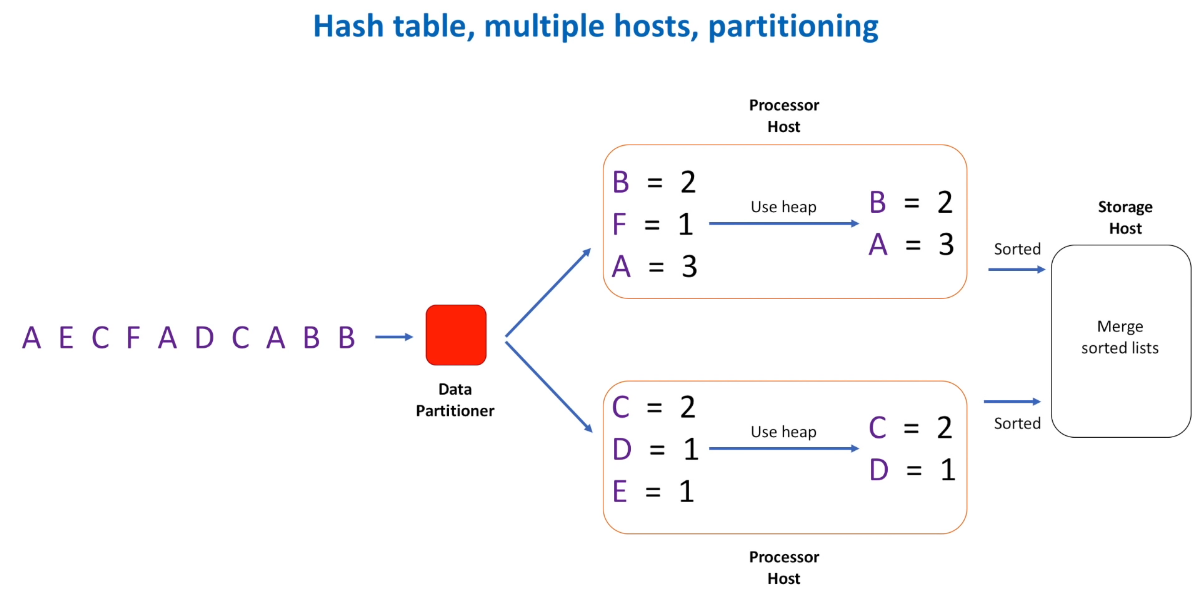
****

****

**Approach 2**

****

**Approach 3**

****

All above approaches are optimized and we have bounded data with limited data size. However, it splits into chunks with data partitioner. Then got the result merged again the result set of each processor host to storage host.

**Approach 4 (Most Optimized)**

