1. How would you design this system? (Feel free to include a drawing or descriptions)

For designing this system, I consider the following important steps:

1. processing and servers
2. storage
3. caching
4. concurrency and communication
5. security
6. load balancing and proxy
7. CDN: Content Delivery Network
8. Monetization: if relevant, how will you monetize? e.g. What kind of DB (Is Postgres enough, if not why?), do you need caching and how much, is security a prime concern?

The below flow chart is very high-level design for my load balancer. The load balancer evenly distributes load (requests from your users) onto group/cluster of application servers.

**Load Balancer:**



Along with this, I would use Map Reduce parallel programming along with Hadoop for the distribution task to process the set of documents and return the word count results. Adding a map-reduce layer makes it possible to perform data and/or processing intensive operations in a reasonable amount of time. E.g. Hadoop.

2. How can you make your system scale?

To make the system scale, I follow the following steps:

1. Load Balancing: Scalability & Redundancy:

2. Caching: Make better use of resources you already have

a). Application Versus Database Caching

b). In Memory Caches

c). Content Distribution Networks

d). Cache Invalidation

3. Off-Line Processing

a). Message Queues

b). Scheduling Periodic Tasks

c). Map-Reduce: When your system becomes too large for ad hoc queries then move to using a specialized data processing infrastructure.

4. Platform Layer: Disconnect application code from web servers, load balancers, and databases using a service level API.

5. Avoid the single point of failure: "If you can’t split it, you can’t scale it", Hence, we can never just have one of anything, we should always assume

and design for having at least two of everything

6. Scale horizontally, not vertically and maintain Asynchronous rather than synchronous requests.

3. What type of database would you use? Why?

NoSQL database suited for this purpose like Redis, Memcached because it does not need data persistence. We can use in-memory data store. Redis supports more complex data structures which are very helpful in sorting the results based on the document creation date Use a key to sorted time mapping. This database is very high scalable.

4. What web framework would you use? Why?

For this design, I would use Angular Web Framework along with Node.js as the backend server. Angular helps in building a very large scale and high-performance web application while keeping the as easy-to-maintain for the users. Since, we have a very huge data (1000000 \* 1 TB), angular will help in fetching the data for users very fast with proper accurate data and out web application will be very light weight which will help the users to use it without any delay or crash. Node.js is lightweight and efficient platform for app backend for querying the data from the database. The user can use this angular built web portal for the querying the data from Redis database.