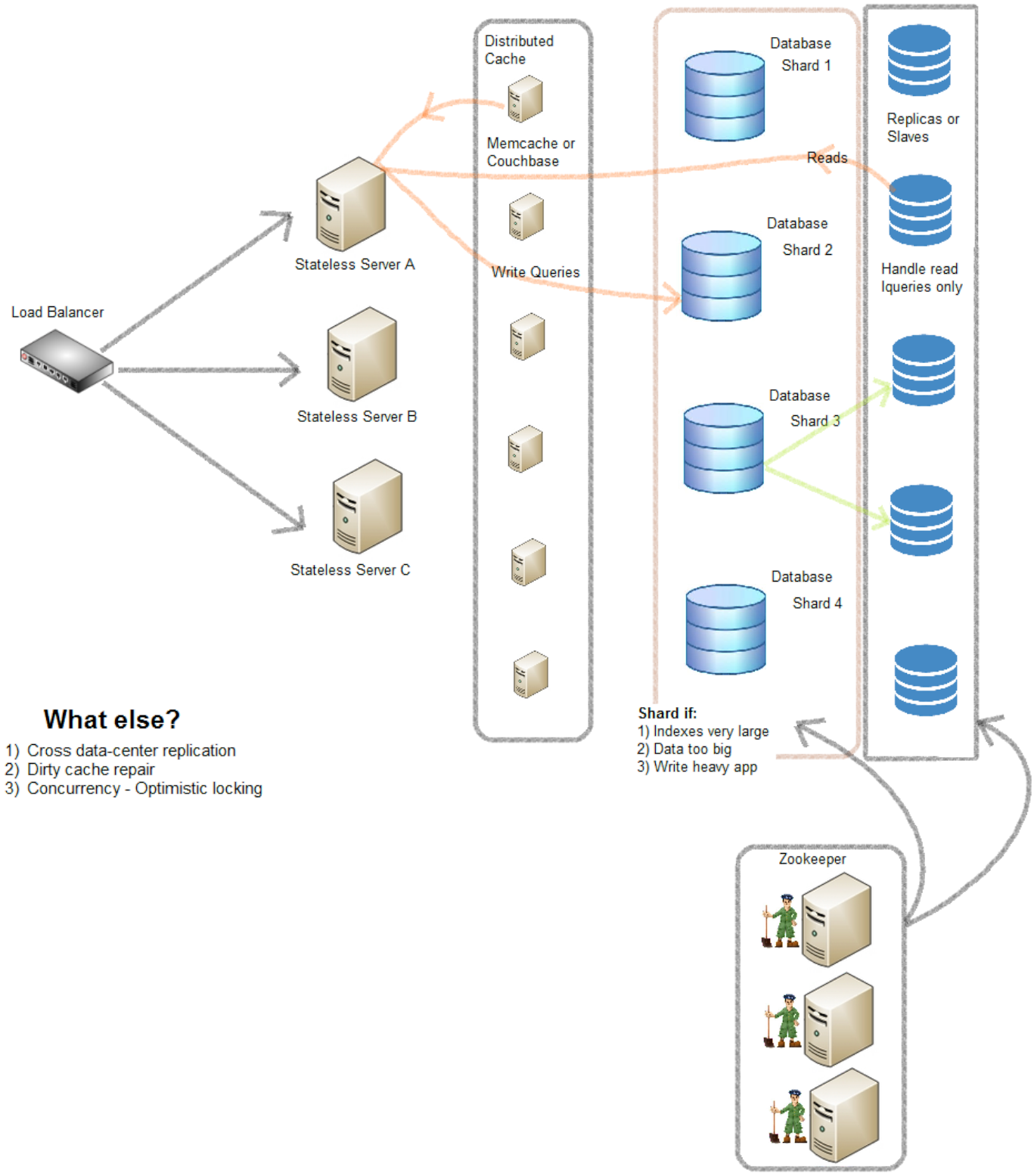
System Design

When designing a complete system, we need to consider the following components' detail:

1. Storage (Database)
2. Cache
3. High Availability - Have some redundant nodes running in active/active mode.
4. Cross country replication - To serve data closer to the user-machine like Akamai
5. Service Level Agreement (SLA): Time required to reflect the changes made in the system
6. Use a load balancer if one server is not sufficient
7. Try to put queuing systems for asynchronous consumption of offline loads.

8. Find out the columns you want to index if using an RDBMS. Find out the memory required by those indexes.
   If indexes per machine are taking more than 10-100 GB per machine, then you have to shard the DB.

9. If read traffic is very high than write traffic, it may be a good idea to increase some slaves replicas.
   Slave replicas will only respond to read traffic but will keep themselves updated by getting data from the server.



Approach to system design

1. Find requests per day.
2. Break it into read and write traffic.
3. Then get requests per second to see if a single machine can support it or not.
4. Estimate the amount of data in a request and response to see the network traffic requirements.
5. Find amount of storage required

6. Find/estimate the expiration policy of data
7. Some numbers from big companies might come in handy. As of 2014-2015:
    a. 12 billion google searches per month
    b. 1.5 billion tweets per month
    c. 54 million total facebook pages
    d. 1.3 billion monthly active users on facebook
    e. 2 billion links shared per month on facebook
    f. 60 billion photos shared per month across all facebook, whatsapp, snapchat etc.
    g. Apple downloads 2 billion apps per month
    h. Youtube uploads 13 million hours of video per month
    i. 500 million google maps users

Statelessness for scalability

1. Make sure your servers are stateless.
   That ways a load balancer can send requests to any server (maybe in a round robin fashion)
   To do this, all the state information must be kept in the database.
   This includes session information as well.
   However, since session information must be available extremely fast, it is best to keep this information in a distributed cache.

2. If servers cannot be made stateless, then the load-balancer must be made smarter.
   It should route stateful requests to the proper server.
   This is done by making the load balancer inspect the session-ID of each request and matching that with the appropriate server.

Load balancer (Also called "Reverse Proxy")

RP systems hide the internal topology from outside clients and in doing so they can provide several features:

1. Caching static content
2. Load balancing
3. Firewalling
4. Compression
5. Logging

A load balancer is useful not only for load balancing, but it also brings:

1. Redundancy and tolerance to machine failures
2. Elastic load-balancers can shut-down some of the servers during non-peak hours.

vmtouch

vmtouch is an application to see what is currently cached by the system.
Its a very simple c program in one file only and can help to see, remove or put things into cache.

Practice

Try system design and estimation on some big websites
1) Google
2) Email services like gmail, yahoo etc.
3) Maps
4) Go Daddy
5) Amazon Web Services
6) Youtube
7) E-commerce
8) Facebook
9) App stores
10) Box