Cache:

- Load balancing help scale horizontally across a large number of servers, but caching will make it vastly better use of the resource we have.
- **Application caching:** requires explicit integration in the application code itself. Usually it will check if a value is in the cache; if not, retrieve the value from the database
- **Database caching:** (most database does that already); the initial settings will be optimized for a generic user case.
- **In-memory caches:** reddis (list set hash, can store values up to 512MB in size) and memcached (only supports key-value pairs and string).
    - **PRE CALCULATES RESULTS; PRE GENERATE EXPENSIVE INDECE**

Load Balancer

- Software hardware: Round Robin; Weighted Round Robbin, Least Connection, Least Time, IP Hash

Database partitioning

- Partitioning of relational data usually refers to decomposing your tables either low-wise (horizontally) or column wise.

Database replication

- Database replication is the frequent electronic copying data from a database in one computer or server to a database in another so that all users share the same level of information. The result is a distributed database in which users can access data relevant to their tasks without interferng with work of others.

Map Reduce:

- Map-reduce layer makes it possible to perform data and/or processing intensive operations in a reasonable amount of time. You might use it for calculating suggested users in a social graph, or for generating analytics reports. E.g. Hive or HBase.

Some Questions to think:

1. **Thread deadlock starvation questions? Parallelize algorithms? Consistency and Coherence?**
    A. **Async vs Parellel:** async is non-blocking, parellelism means dividing work up into chunks. Asynchronous basically means you do not wait results and do something else until the work comes back.
    B. **Mutex (mutual exclusion semaphore)** is a locking mechanism used to synchronize access to a resource. Only one task can acquire the mutex. **Semaphore** is a signaling mechanism. It can be acquired by multiple process threads to access the finite instance of the resource until available
    C. **Race condition:** A race condition is situation on concurrent programming where two concurrent threads or process compete for a resource and result final state depends on who gets the resource first. You don't know who will access the data first I.e. both are racing to access/change the data. You typically need a lock.
        a. If x==5: y *=2; you don't know the value of
    D. **Thread safe:** code will work even if many threads are executing it simultaneously within the same process. => internal data structures of operation that could run uninterrupted are protected against different modifications at the same time.
    E. **How to make sure it's only initiated once? Return once?** do a threadsafe counter behind the communication end point (use a lock to achieve it); if it needs to survive reboots, you can return a UUID (you cannot produce a duplicate one)
    F. **Two customer one cup:** 1). Not selling one low inventory 2. Leave in a basket for a period of time 3.
    G. **Starvation:** a thread is unable to gain regular access to shared resources and is unable to make progress. This happens when shared resources are made unavailable for long periods by

greedy threads. (One way to deal with is using a scheduling, like Queue) It's usually caused by poor context switch (poor scheduling algorithms). Aging Algorithm is something you can do.

H. **Process** can have multiple **threads**

1. **Networking? TCP/IP, IPC? Throughput and latency, when each is the relevant factor?**
    A. **Throughput vs Bandwidth:** throughput is a term used for how much data can be transferred from the source to its destination within a given time frame, while bandwidth is the term for MAXIMUM transfer capacity of a network.
    B. **Router**: it connects to modem and then to your device via Ethernet or wifi signal. It creates a local area network (LAN). It can understand different protocols. **Modem** connects your home to the internet, while a router creates the network inside your house. (A modem takes **Internet Service Provider ISP)** to your local device, this internet is WAN (Wide Area Network). Each Modem has a assigned public IP that is identified. **Router will then assign a local IP to each device on the network.**
    C. **Layers:**
        a. **1.** Physical Layer
        b. 2. Data Link Layer
        c. 3. Network Layer (Router)
        d. 4. Transport Layer
        e. 5. Session Layer: opening and closing communication between the two devices
        f. 6. Presentation Layer: encryption
        g. 7. Application Layer: HTML + SMTP(mail servers port 28 and 587) P2P
    D. **TCP (Transmission Control Protocol)** and the **UDP (User Datagram Protocol) (transport layer)**
        a. **TCP:** Guranteed order, actually receive; send long data (Transport Layer)OSI
        b. **UDP: faster,** maybe delivered twice and might not be delivered, and maybe delivered out of order. VOIP (voice over internet protocols) does not require a constant connection, low cost
    E. **On Application level, DNS, SNMP uses UDP.** DNS requests are very into so they have no problems fitting into UDP segments.

1. **How do you design a system to cope with network failures? Do you understand durability**
    A. **To increase throughput:** you can
        a. Reduce latency
        b. Use a different protocol (P2P. UDP, increase bandwidth)
    B. **To improve latency you can:**
        a. Better path (load balancer based on IP)
        b. Caching
        c. Protocol choice:
    C. **Failure Tolerant:**
    D. **Durability:** Something that is constantly involving.