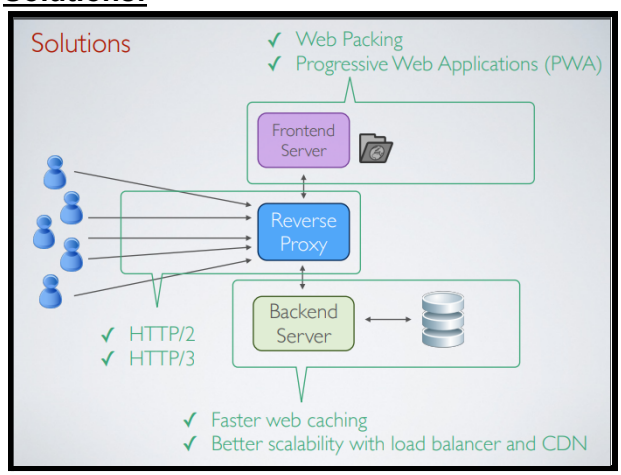
Users respond to speed:

- Amazon found every 100ms of latency cost them 1% in sales.
- Google found an extra .5 seconds in search page generation time dropped traffic by 20%.

Problems to consider:

- How to increase the throughput?
  Throughput refers to how much data can be transferred from one location to another in a given amount of time. It is used to measure the performance of hard drives and RAM, as well as Internet and network connections.
- How to scale to serve millions of users?

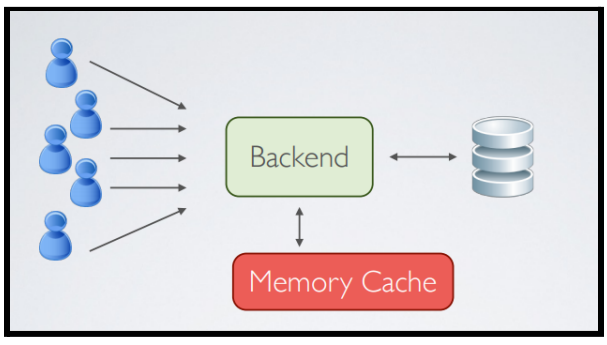
Solutions:



Backend Web Caching:

- Processing the request means:
  1. Parsing the HTTP request.
  2. Mapping the URL to the handler.
  3. Querying the database or third-party API.
  4. Computing the HTTP response.

  Of all these tasks, step 3 is the most expensive. DB and API accesses are expensive both in terms of time and money.
- **Caching:**
- In computing, a **cache** is a high-speed data storage layer which stores a subset of data, typically transient in nature, so that future requests for that data are served up faster than is possible by accessing the data's primary storage location. Caching allows you to efficiently reuse previously retrieved or computed data.
- The data in a cache is generally stored in fast access hardware such as RAM and may also be used in correlation with a software component. A cache's primary purpose is to increase data retrieval performance by reducing the need to access the underlying slower storage layer.
- Instead of making the same queries to the database multiple times, we can store the information in cache to make it faster to retrieve in the future.
- I.e.



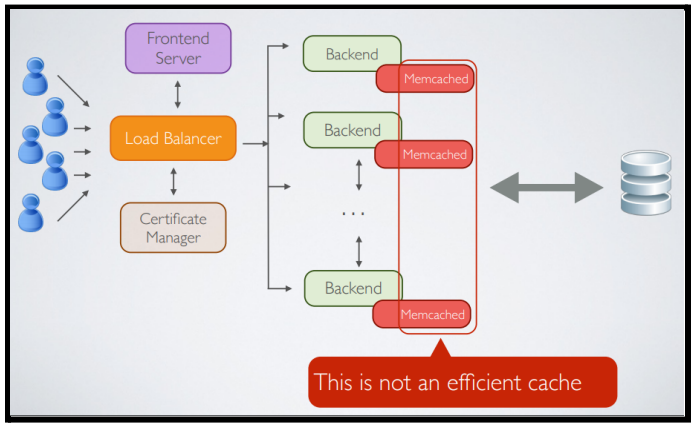
- The cache is controlled by the program and is specific for each app.
- We can cache database requests and session information.
- A popular memory cache is memcached.
- Memcached is a distributed shared cache.
- Memcached stores key/value pairs in memory and throws away data that is the least recently used.
- A typical cache algorithm:

```
retrieve from cache
if data not in cache:
        # cache miss
        query the database or API
        update the cache
return result
```

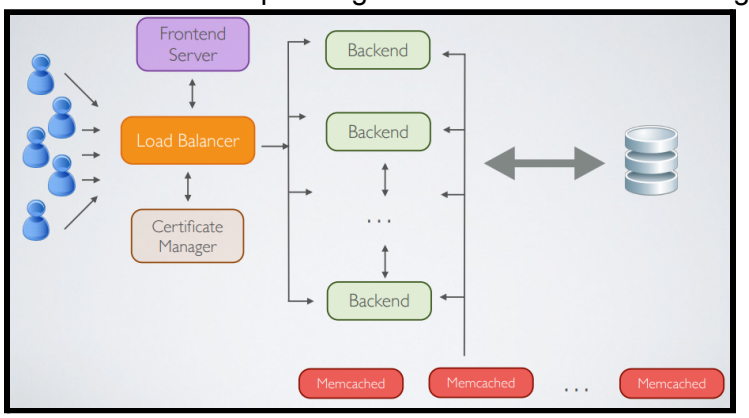
- **Cache stampede/dog piling** is a problem that occurs when multiple concurrent requests are doing the same request because cache was cleared. A cache stampede occurs when several threads attempt to access a cache in parallel. If the cached value doesn't exist, the threads will then attempt to fetch the data from the database at the same time. Due to the sudden spike in CPU usage, the database will crash.
- E.g.
  Imagine you are doing an expensive SQL query that takes 3 seconds to complete and spikes CPU usage of your database server. You want to cache that query as running multiple of those in parallel can cause database performance problems and could bring your entire app down. Let's say you have a traffic of 100 requests a second. With our 3-second query example, if you get 100 requests in one second and your cache is cold, you'll end up with 300 processes all running the same uncached query.
- A solution is **cache warming**. Cache warming is when websites artificially fill the cache so that real visitors will always get a cache hit. Essentially, sites that engage in cache warming are preparing the cache for visitors, rather than allowing the first visitor to get a cache miss. This ensures every visitor has the same experience.
- Warm cache maintains useful data that your system requires. It helps you achieve faster processing. It will be time-consuming if for each request the process has to query a DB to get this information. so it would be a good idea to cache it; and that would be feasible through warm cache. This cache should be maintained regularly otherwise your cache may grow in size with unnecessary data and you might notice performance degradation.
- I.e.
  Update the cache instead of clearing it after an insert.
  Furthermore, you warm the cache when you start the server.

Scaling The Backend:

- **Load Balancing:**
- We can use a load balancer to distribute the weight. A **load balancer** acts as the "traffic cop" sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no one server is overworked, which could degrade performance. If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

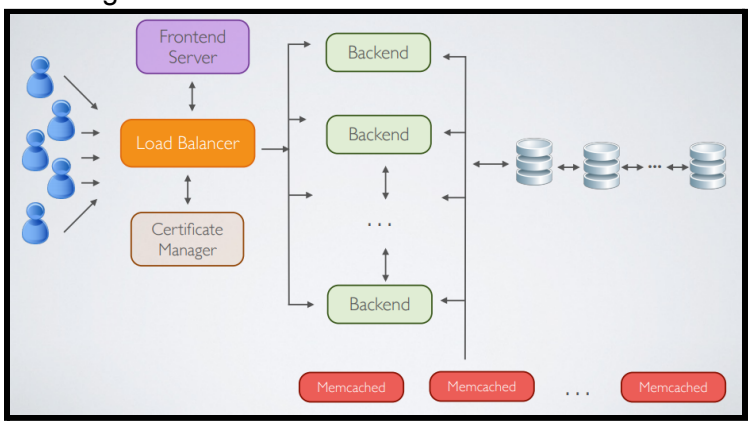


This is a bad design because each memcached in each backend will contain the same information. Your duplicating information over and over again.

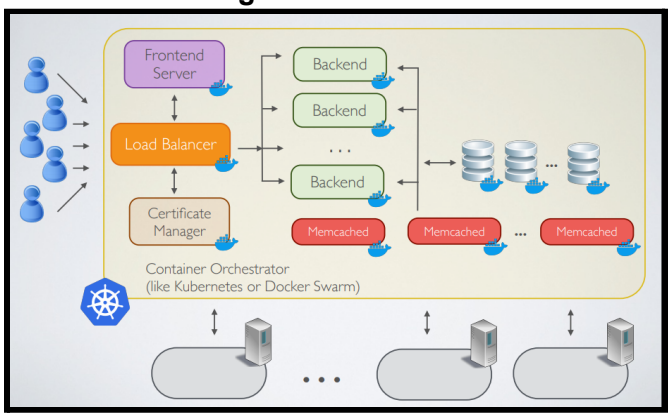


Memcached is a distributed shared cache. This means that while there are many memcached servers, in reality, we're dealing with a unified memcached.

- **Database Sharding:**
- **Database sharding** is the process of breaking up large tables into smaller chunks called shards that are spread across multiple servers. The idea is to distribute data that can't fit on a single node onto a cluster of database nodes.

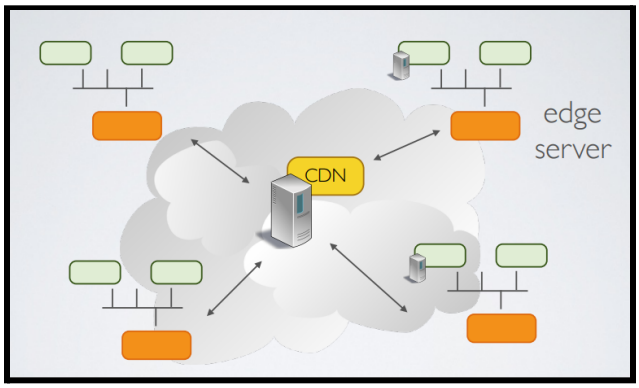


- **Automatic Scaling with container Orchestration:**



At certain times, our web application will have more users using it and at other times, our web application will have few or no users using it. For example, for Amazon, its 2 busiest days are Black Friday and Christmas. Furthermore, consider an Amazon-like website but only for Canada. At night, it will have few people using it. We can use Kubernetes to automatically scale our website based on the stress/load.

- **CDN (Content Distribution Network):**



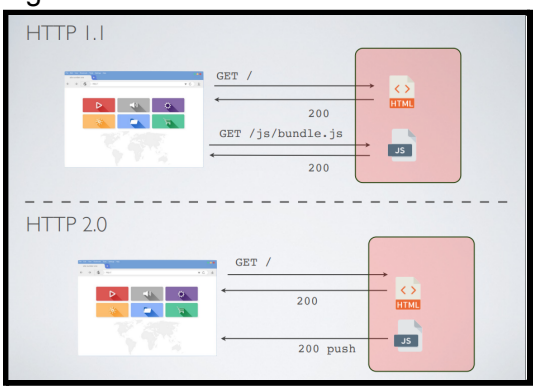
- A **content delivery network (CDN)** refers to a geographically distributed group of servers which work together to provide fast delivery of Internet content.
- A CDN allows for the quick transfer of assets needed for loading Internet content including HTML pages, javascript files, stylesheets, images, and videos. The popularity of CDN services continues to grow, and today the majority of web traffic is served through CDNs, including traffic from major sites like Facebook, Netflix, and Amazon.
- A CDN is a highly-distributed platform of servers that helps to minimize delays in loading web page content by reducing the physical distance between the server and the user. This helps users around the world view the same high-quality content without slow loading times.
- Without a CDN, content origin servers must respond to every single end user request. This results in significant traffic to the origin and subsequent load, thereby increasing the chances for origin failure if the traffic spikes are exceedingly high or if the load is persistent.
- By responding to end user requests in place of the origin and in closer physical and network proximity to the end user, a CDN offloads traffic from content servers and improves the web experience, thus benefiting both the content provider and its end users.

Frontend Packing:

- If your frontend has a lot of HTML/JS/CSS files, loading the files will take a long time.
- A solution is to use webpack.
- Webpack's mission is to take many different modules, files and assets and bundle them together. It transfers their content into a single file or a smaller group of files. It also manages all complexities regarding your dependencies, making sure that code is loaded in the correct order.

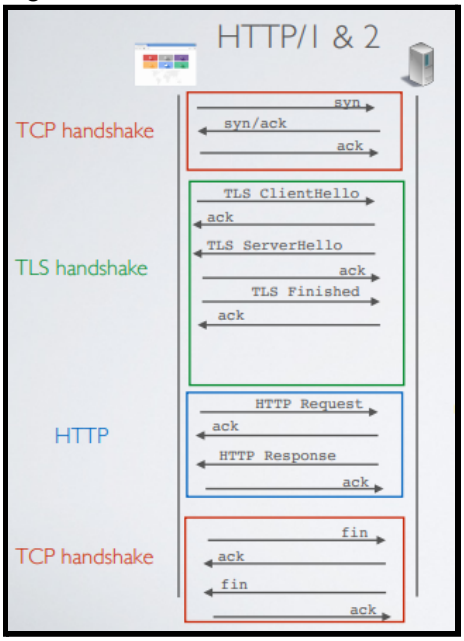
HTTP Versions 2.0 and 3.0:

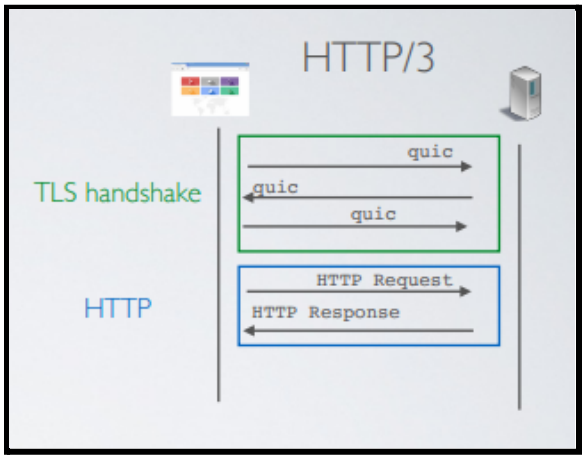
- **HTTP 2:**
- HTTP/2 enables multiplexing which means sending multiple HTTP responses for a given request (aka push).
- E.g.



With HTTP 1.1, you had to make GET requests separately for the HTML, JS, and CSS files. With HTTP 2.0, if a user makes a GET request for the HTML page, the JS and CSS files are also sent back with the HTML file.

- It was proposed by Google and was originally called SPDY.
- It was adopted as a standard in 2015 (RFC 7540).
- HTTP/2 is compatible with HTTP/1 (same protocols).
- **HTTP/3:**
- Is still a work in progress. However, Chrome is compatible with HTTP 3.
- HTTP/3 is compatible with HTTP/2 and HTTP/1.
- The main difference between HTTP/3 and HTTP/2 is that HTTP/3 uses the UDP protocol instead of the TCP protocol. UDP is a lot faster than TCP.
- E.g.





Short Polling vs Long Polling:

- **Short Polling:**
- **Short polling** is a technique where the frontend requests an update from the backend every few seconds and the backend replies right away regardless if there is an update or not. In this case, many requests/responses are wasted.
  I.e. Send a request to the server and get an instant answer. Do this every few seconds, minutes, etc to keep your application up-to-date. But, this costs a lot of requests.
- Twitter uses short polling.
- **Long Polling:**
- **Long polling** is a technique where the frontend requests an update from the backend and waits for the response and the backend replies to the update request only when there is an update. In this case, no requests/responses are wasted and updates are processed as soon as they arrive.
  I.e. Long polling is technique where the server elects to hold a client connection open for as long as possible, delivering a response only after data becomes available or timeout threshold has been reached. After receiving a response, the client immediately sends the next request.
- Facebook uses long polling.

WebSockets:

- A **WebSocket** is a persistent connection between a client and server. WebSockets provide a bidirectional, full-duplex communications channel that operates over HTTP through a single TCP/IP socket connection. **Note:** WebSockets do not rely on HTTP at all except for initialization.
- A full-duplex system allows communication in both directions to happen simultaneously. E.g. Land-line telephone networks are full-duplex since they allow both callers to speak and be heard at the same time.
- WebSockets are similar to low-level POSIX sockets.

WebRTC (Web Real-Time Communications):

- It is a full-duplex communication between clients (browsers).
- It provides peer-to-peer (P2P) communications, which is perfect for sending text, video, audio without going through the server except for initialization and signalling that goes through the server usually using WebSockets.
- WebRTC has no signaling of its own and this is necessary in order to open a WebRTC peer connection. WebRTC can achieve this by using other transport protocols such as HTTPS or secure WebSockets.
  I.e. To connect a WebRTC data channel you first need to signal the connection between the two browsers. To do that, you need them to communicate through a web server in some way. This is achieved by using a secure WebSocket or HTTPS.
- The main difference between WebSockets and WebRTC is that WebSockets are meant to enable bidirectional communication between a browser and a web server while WebRTC is meant to offer real time communication between browsers.

Progressive Web Applications (PWA):

- PWAs are web applications that can be installed on your system.
- It works offline when you don't have an internet connection, leveraging data cached during your last interactions with the app.
- It relies on the browser's local storage to store the frontend and checks for updates with the server.
- It relies on web-workers for caching and communication.