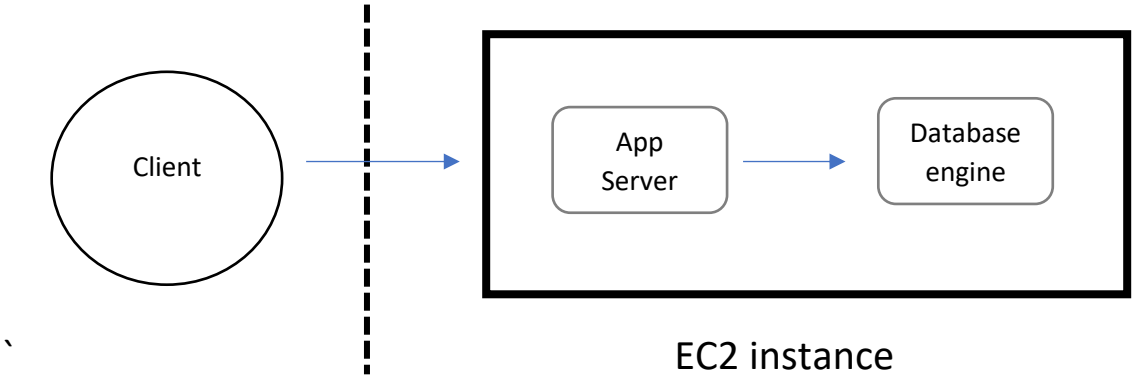
Question:

Once deployed, suppose this application became very famous and started to receive a ton of traffic. Your application now contains metadata about 5M movies and receives 15M API hits per day both from anonymous as well as authenticated users. Suggest an architecture to scale up this system to 5x of these specs. You can also think of potential bottlenecks at all layers of the stack and how you will solve for these.

Answer:

Initial state of the application:

Application is hosted on an AWS EC2 instance with a relational database server running on the same instance.



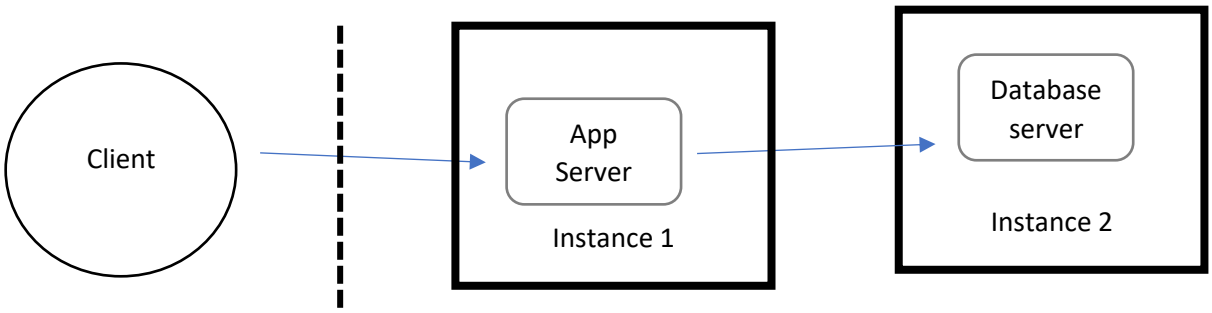
`

Now, if our app became very famous and starts having Millions of requests per day then this system design wants to be able to handle all increased traffic.

To overcome this issue, we can do either vertical scaling or horizontal scaling.

But, the problem with vertical scaling is that we won't be able to scale our application after one point or it'll become more expensive to scale it vertically than horizontally.

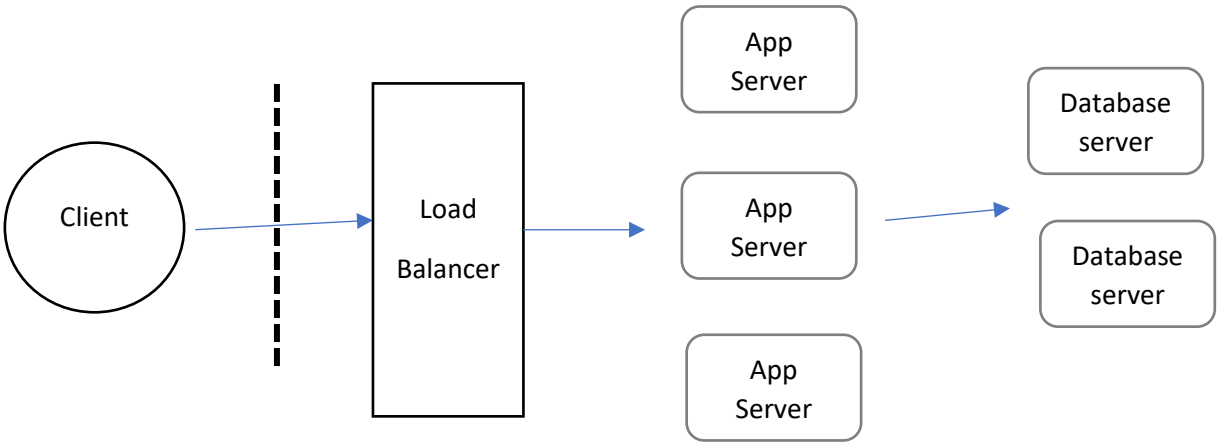
So, that's why we need to choose horizontal scaling. Now we will use different instances/nodes/machines for the app server and database server.



But this also won't solve our problem because if the load increases more than the app server can handle then our application will crash.

Also, traffic will come from different geographic locations so if we transfer all the traffic to one specific location then response time will decrease.

So, to overcome this problem we can use a load balancer that can act as a single point listener for clients and distributes incoming traffics across multiple instances.



Now, our backend will be able to handle the upcoming traffic, but as load on database will increase, we will be performing many similar operations on the database again and again which will impact the response time of the database server.

So, to overcome this issue we can use a **Cache Server** such as Redis to store the results of most frequent queries. So, now instead of directly making a query to the database, we will check cache for requested result and it's available we will simply return from cache memory.

This operation will be much faster than making query to the db.

Final System Design: