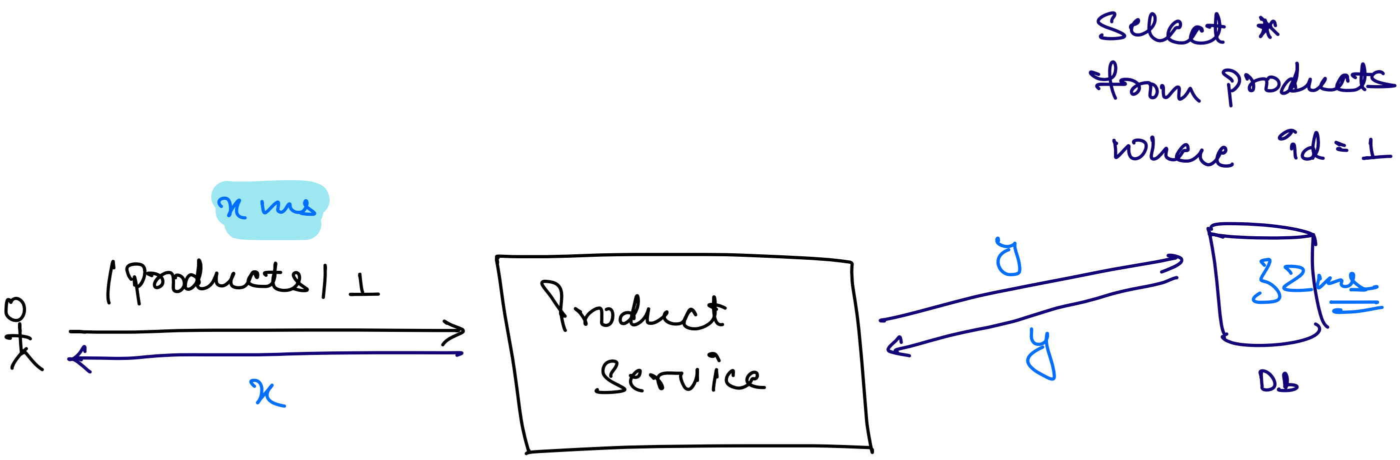
Caching : Implement Redis Cache in Our Product Microservice.

Outcome : latency improved from 500ms to 20ms.

90+%

* Latency Metric improvement, you can add in the Resume Project section.

⇒



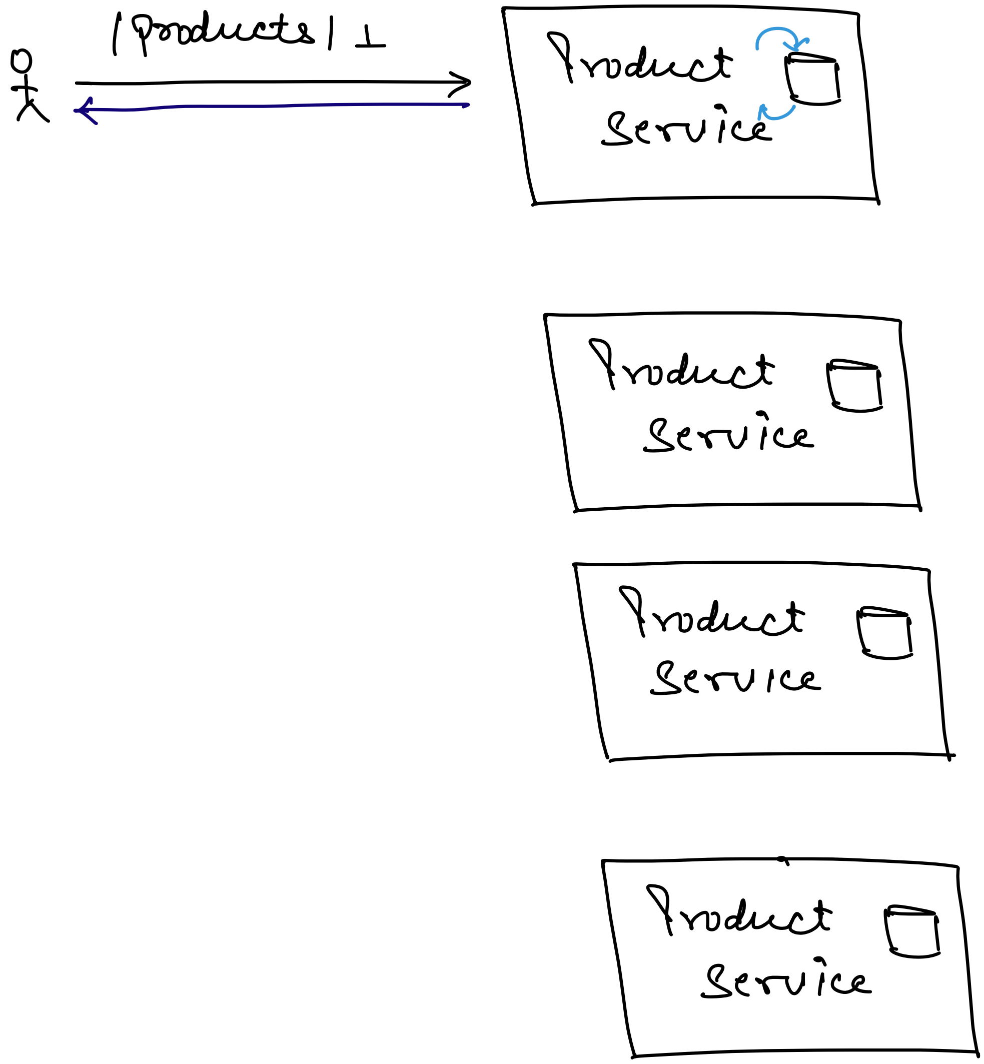
Total latency of the request =

$x + y + 2 + y + x$

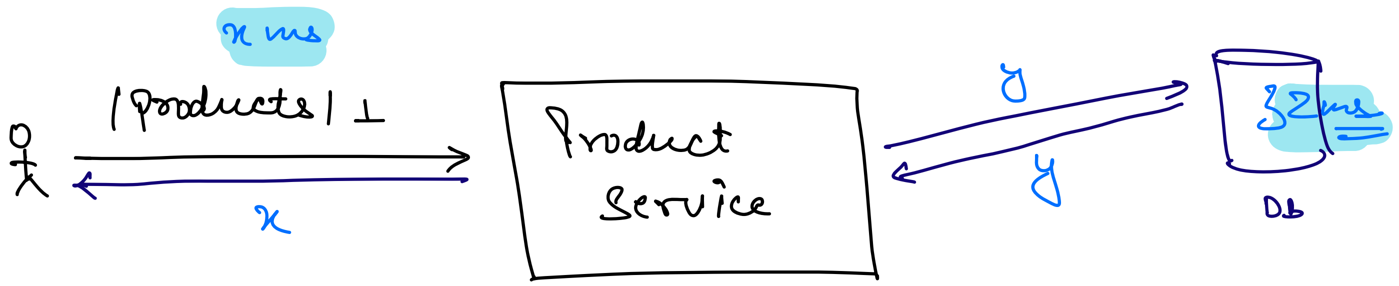
⇒ If we place the Database within the server, then latency will get improved but

① Cost will get High.

② How will we keep our DB's in sync.



Cost of RAM >>>> Cost of HDD.



⇒ Database : Hard Disk.

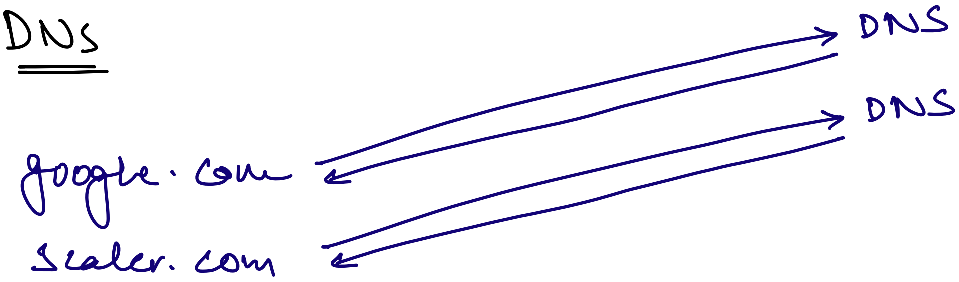
Stores the data permanently

Hard disk access >>>>> RAM access.

Cache.

⇒ Storing some piece of information at some other hardware to optimize the latency.

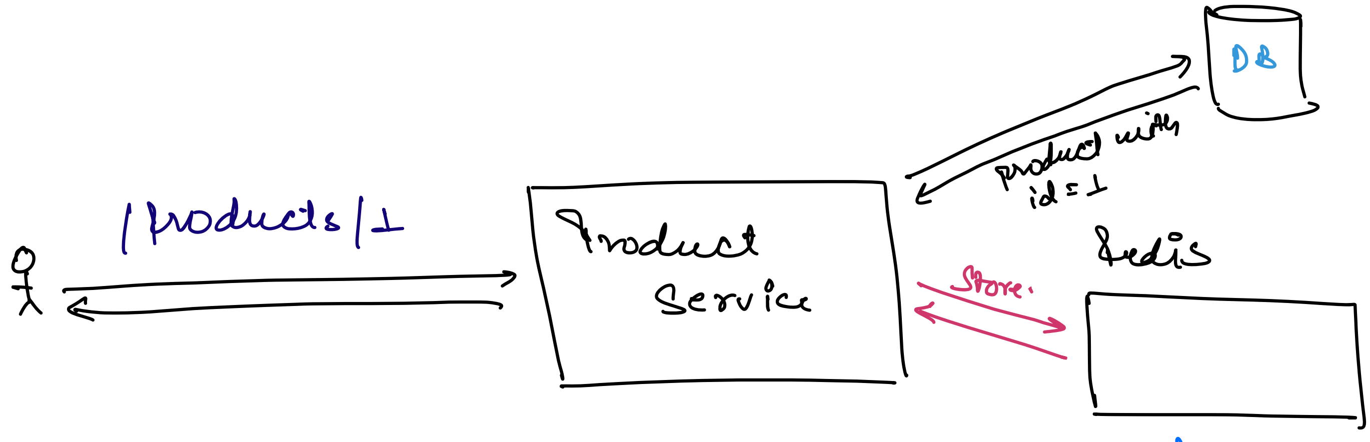
⇒ DNS



⇒ For the very first time, Browser sends a request to the DNS server to get the IP address of the website.

⇒ later it caches the IP address inside the Cache.

Caching



RAM

Volatile Memory.

→ Cache Miss.

→ Cache Hit.

→ Cache Eviction

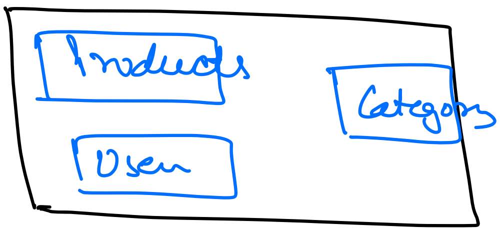
↳ Multiple policies are there to evict the data from Cache.

→ LRU : Most Commonly used.

DLL + Map.

⇒ Redis is like a Map.

<K,V>



⇒ FakeStore.

```
getProductById(Long id){
    Product p = Cache.get(id);
    if(p != null) {
        return p;
    }
    Product p = fakestore.get(id)
    Cache.put(id, p);
    return p;
}
```

⇒ Map. <$\underset{\uparrow}{K}$, V> : Hashing based DS.